Scaling out some of the Data Mining Algorithms for Big Data Analytics

A thesis submitted to University of Hyderabad in partial fulfillment

for the degree of

Doctor of Philosophy

by

Sk KAMARUDDIN

Reg.No. 14MCPC17

SCHOOL OF COMPUTER AND INFORMATION SCIENCES

UNIVERSITY OF HYDERABAD

HYDERABAD -500046

Telangana

India

July, 2020

بِسمِ اللَّهِ الرَّحمٰنِ الرَّحيمِ

In the name of Allah, Most Gracious, Most Merciful.

उस अल्लाह के नाम पर (शुरू करता हूँ) जो दयालु और करुणामय है।

IDRBT
Explore, Enable, Excel

CERTIFICATE

This is to certify that the thesis entitled "**Scaling out some of the Data Mining Algorithms for Big Data Analytics**" submitted by **Sk KAMARUDDIN** bearing registration number **14MCPC17** in partial fulfillment of the requirements for award of Doctor of Philosophy in the School of Computer & Information Sciences is a bonafide work carried out by him under my supervision and guidance at IDRBT, Hyderabad.

This thesis is free from plagiarism and has not been submitted previously in part or in full to this or any other University or Institution for award of any degree or diploma.

Parts of this thesis have been published online in following publications:

1. Book chapter in Handbook on Big Data Analytics, IET, UK & Proceedings of BDA 2017, vol. 10721, Springer, 2017, pp. 15–39. (Chapter 2)
2. Proceedings of ICDIM, 2015, pp. 223–228. (Chapter 3)
3. Proceedings of ICIA-16, 2016, pp. 33:1-33:8. (Chapter 4)
4. Proceedings of BDA 2017, vol. 10721, Springer, pp. 278–292. (Chapter 5)
5. Proceedings of ICDMAI 2019, vol. 1042, Springer, pp. 215–227. (Chapter 6)
6. Book chapter in Handbook on Big Data Analytics, IET, UK & Proceedings of TENCON 2019, IEEE, pp. 393-397. (Chapter 8)

Further, the student has passed the following courses towards fulfillment of coursework requirement for Ph.D:

	Course Code	Name	Credits	Pass/Fail
1	BT701	Data Structures and Algorithms	4	Pass
2	BT702	Operating System and Programming	4	Pass
3	BT712	Trends in Softcomputing	4	Pass
4	BT717	Datamining	4	Pass

01/7/2020

Supervisor	Director	Dean
Prof. Vadlamani Ravi	Dr. A.S. Ramasastri	Prof. Chakravarthy Bhagvati
Center of Excellence in Analytics, IDRBT	IDRBT	School of Computer and Information Sciences, UOH
Hyderabad-500 057, India	Hyderabad-500 057, India	Hyderabad-500 046, India

Dr. V. Ravi, M.Sc., M.S., Ph.D,
Professor
Head, Center of Excellence on CRM & Analytics
INSTITUTE FOR DEVELOPMENT AND RESEARCH IN BANKING TECHNOLOGY
(Established by Reserve Bank of India)
Castle Hills Road No. 1, Masab Tank,
Hyderabad - 500 057, (A.P.), India

Dr. A.S. RAMASASTRI
DIRECTOR
INSTITUTE FOR DEVELOPMENT AND RESEARCH IN BANKING TECHNOLOGY
(Established by Reserve Bank of India)
Castle Hills, Road No.1, Masab Tank,
Hyderabad - 500 057

DECLARATION

I, **Sk KAMARUDDIN**, hereby declare that this thesis entitled "**Scaling out some of the Data Mining Algorithms for Big Data Analytics**" submitted by me under the guidance and supervision of **Prof. Vadlamani Ravi**, is a bonafide research work and is free from plagiarism. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma. I hereby agree that my thesis can be deposited in Shodganga/INFLIBNET.

A report on plagiarism statistics from the University Librarian is enclosed.

Date: 16/07/2020

Sk Kamaruddin

Signature of the Student
(Sk Kamaruddin)
Reg. No.: 14MCPC17

//Countersigned//

Signature of the Supervisor:

Dr. V. Ravi, M.Sc., M.S., Ph.D.
Professor
Head, Center of Excellence on CRM & Analytics
INSTITUTE FOR DEVELOPMENT AND RESEARCH IN BANKING TECHNOLOGY
(Established by Reserve Bank of India)
Castle Hills Road No. 1, Masab Tank,
Hyderabad - 500 057, (A.P.), India

Dedicated

to

My Family & Teachers

ACKNOWLEDGEMENT

First and above of all, I praise **Allah, the Almighty**. I am very much grateful to Him for His immense blessing and mercy on me to withstand all the difficulties during the journey of Ph.D.

Next, I would like to seek blessings of my late parents **Sk Imamuddin** and **Maiman Nishan**, late father-in-law, Haji Azmat Ali Khan and mother-in-law, Khurshid Ara Begum. I would like to acknowledge the unfathomable love, support, and motivation from my wife **Sahista Parveen**, son **Sk Shoaib Akhtar**, and daughter **Aaliyah Parveen** for pursuing my dreams. I would also like to thank my younger brothers (Zafaruddin and Safaruddin), younger sisters (Rukhsana and Sohana), my nephews, and nieces for their immense love. I would also thank my brother and sister-in-laws for their support.

I especially acknowledge **University of Hyderabad** (UoH) for its academic support. I also extend my acknowledgments to **Institute for Development and Research in Banking Technology** (IDRBT), an R&D center established by RBI, for its kind financial support and conducible environment for pursuing the research. I express my sincere thanks to **Center of Excellence in Analytics** (CoEA), IDRBT, for making me part of experimenting with datasets of various banks and presenting the results to bankers throughout Ph.D.

This thesis would not have been possible without the help and support of the kind people around me. Among them, first of all, I owe my greatest gratitude to my supervisor **Prof. Vadlamani Ravi**, Professor, IDRBT, Hyderabad for his precious foundational & advanced lectures of Data Mining and related topics, strict supervision, regular encouragement and timely support from the introductory level to the concluding level that enabled me to understand and implement the concepts learned. He guided me in each stage of my research, including problem definition, idea creation, problem solution, and paper writing.

It is my privilege to thank **Prof. Chakravarthy Bhagvati**, Dean, School of Computer and Information Sciences (SCIS), UoH, Hyderabad for his academic support. It is an honor for me to thank **Dr. A.S. Ramasastri**, Director, IDRBT for reviewing the work and timely suggestions throughout my research. I would like to extend my sincere thanks to **Prof. B.M. Mehtre**, Professor, IDRBT, **Dr. G. R.**

Gangadharan, Ex-Associate Professor, IDRBT, and **Dr. M.V.N.K. Prasad**, Associate Professor, IDRBT for their regular reviews of the progress of research and kind inputs as the members of my Doctoral Research Committee. I also take a minute to thank **Prof. D.K.Subramanyam**, **Prof. B.L. Deekshatulu** and **Prof.Venu Govindaraju** for their periodical reviews of my work and their kind inputs. I also express my heartfelt thanks to **Dr. Rajarshi Pal, Dr. Nagesh B. Sristy**, for their valuable suggestions and support during my research period.

I would also like to express my sincere thanks and gratitude to the teaching staffs, including Prof. V.N. Sastry, Dr. V. Radha, Dr. Rajarshi Pal, Dr. P. Syam Kumar and Mr. Sandilya for their lectures as part of my coursework and timely suggestions.

I am always thankful to Mr. G. Raghuraj for his support in attending a conference in Malaysia. I am always grateful to the admin staff of IDRBT including Dr. Rashmi Dev, Mrs.Varsha Srivatsava, Mr. Vijay Belgurikar, Mr. Subramanyam, Smt. R. S. Sirisha, and Miss.T. Preethi for their support in attending conferences; Mr.Ratna Kumar, Mr. Srinivas, and Mr.Prakash Dhavale for their support in utilizing library resources to conduct research; Mr.K.V.R.Murthy for his technical support; Mr. K. Srinivas, Mrs. Pavani, Mr. Namdev, Mrs. Sony and Ms. Madhuri for administrative support; Mr. K. Dharmender and Mr. Anurag Nanda for the support during the stay in the quarters and finally, the staff of Prism Catering Services for their continuous support for serving lunch and refreshments throughout my doctoral study.

I would like to express my thanks to my seniors including Dr. R. K. Mohanty, Dr. D. Pradeep Kumar, Dr. B. Shravan Kumar, Dr. Hiran V. Nath, Dr. Ghanshyam Bopche, Dr. Gopal Narayan Rai, Dr. Deepnarayan Tiwari, Dr. Jatoth Chandrashekar, Dr. Kumar Ravi, Dr. M. Sandhya, Dr. Manu V.T., Dr. Anup Kumar Maurya, Dr. B. Sriramulu for their invaluable support. It is always a pleasure to express my sincere thanks to my research colleagues including G. Jayakrishna, U. Ravi, Dr. V. Dinesh, Dr. K. Sitara, Satya Krishna, Avinash, Shadab, Narottam, Malvika, Mallikarjun, Mahesh, Manoj, John, Nisanth, Praveen, Chandrashekar, JayaRao, Shalini, Amarajyothi, Asha Kiran, Ravi for their interactions with me and kind support.

Throughout my doctoral study, many research associates of IDRBT accompanied me. I would like to express my thanks to Siddeshwar, Jameel, Arvind, Tejasviram, Shiva Krishna, Sai Kiran, Prudhvikrishna, Mayank, Gautam, Raviteja, Harshal, Zaheer, Vikram, Arafat, Chandrabhusan. I also take time to thank UoH M.Tech students who worked in CoEA, including Raj Shukla, Maneesh Ojha, Amit Soni, Rohit, Bhaskar, Akhilesh, Tanveer, Vishnu, Sarveswar, and Basit.

I also want to extend my sincere thanks to the principal of Dawn Buds Model School, Begumpet, Mr. E. D. Nagaraju, where my kids are studying; for his continuous support. Finally, I would like to thank all the near and dear ones who have supported and prayed to The Almighty for me.

Sk Kamaruddin

ABSTRACT

Big Data Analytics is the analysis performed on Big Data, i.e., the data characterized by 5Vs viz. Volume, Velocity, Variety, Veracity, and Value. The analysis results in the extraction of knowledge from structured, unstructured, and semi-structured data. This knowledge helps us to make business decisions and necessary actions. Analyzing Big Data with traditional systems may not be possible; hence it is preferred to go for a cluster of traditional systems where the resources can be shared, and fault tolerance can be maintained. Such a computational framework is provided by the Apache Hadoop or Apache Spark. Hadoop or Spark computational framework has been employed for the research works presented in this thesis. The thesis starts with a comprehensive review presenting a critical analysis of two hundred and sixty-one (261) research articles that appeared between 2009 and 2020 dealing with parallel and distributed versions of regression, classification, clustering, association rule mining, recommendation systems, outlier detection problems in data mining.

Regression is one of the critical tasks in data mining. We proposed a hybrid model combining Auto-Associative Extreme Learning Machine and Multiple Linear Regression for performing Big Data regression. It is implemented with Hadoop MapReduce computational framework.

We proposed a one-class classification suitable for Big Data by employing Auto-Associative Neural Network optimized with Particle Swarm Optimization. The proposed method is implemented in a parallel, distributed manner. Credit card fraud is one of the severe challenges faced by the banking and financial industries. The detection of the scarce fraudulent transaction can be performed by one-class classification.

Clustering is the most often used unsupervised learning method. Implementation of clustering involving volume and velocity is a challenging task. We have presented a parallel distributed incrementally evolving clustering method for the Big Data paradigm employing a parallel distributed version of the Evolving Clustering Method.

Iterative training has its drawbacks while analyzing Big Data. So, we need training models where training can happen in one-pass. We have proposed a parallel distributed one-pass regression model for performing regression in the Big Data

paradigm. The work has implemented a parallel distributed version of the General Regression Neural Network.

Classification is the most popularly used data mining task across industries. Considering Big Data classification, we have proposed a parallel distributed one-pass classifier employing a parallel distributed version of Probabilistic Neural Network.

We have implemented a parallel distributed version of the Radial Basis Function Network, which is amenable for Big Data. The versatility of the architecture allows performing regression and classification for Big Data.

The last chapter of the thesis summarizes the contributions and suggests the research gaps as the future directions for the upcoming researchers. We observe that the Big Data analytical algorithms proposed in the thesis are eminently suitable for solving regression, one-class classification, binary classification, and clustering problems in various domains.

Contents

List of Figures

List of Tables

List of Acronyms

- **10-FCV** 10-Fold Cross-Validation
- **AAELM** Auto-Associative Extreme Learning Machine
- **AANN** Auto-Associative Neural Network
- **ABC** Artificial Bee Colony
- **ACO** Ant Colony Optimization
- **AF** Activation Function
- **AMR** Amazon Movie Review
- **ANN** Artificial Neural Networks
- **ARM** Association Rule Mining
- **AUC** Area Under Curve
- **BDA** Big Data Analytics
- **BPNN** Back Propagation Neural Network
- **CART** Classification and Regression Trees
- **CBR** Content-Based Recommendation
- **CF** Collaborative Filtering
- **CHAID** Chi-square Automatic Interaction Detection
- **CNN** Convolutional Neural Networks
- **CPNN** Counter Propagation Neural Network
- **CPU** Central Processing Unit
- **CR** Classification Rate
- **CRS** Computerized Reservations Systems
- **DE** Differential Evolution
- **DM** Data Mining
- **DT** Decision Tree
- **DTM** Document Term Matrix

- **ECM** Evolving Clustering Method
- **ELM** Extreme Learning Machine
- **ELM** Extreme Learning Machine
- **ENN** Ensemble Neural Networks
- **FCM** Fuzzy c-Means
- **FET** Field effect transistor
- **FN** False Negative
- **FP** Frequent Pattern
- **FP** False Positive
- **FPGA** Field Programmable Gate-Array
- **FPM** Frequent Pattern Mining
- **FPR** False Positive Rate
- **GA** Genetic Algorithm
- **GA** Genetic Algorithm
- **GB** Gigabytes
- **GBT** Gradient-Boosted Tree
- **GMDH** Group Method of Data Handling
- **GPGPU** General-Purpose Graphics Processing Unit
- **GRNN** General Regression Neural Network
- **HDFS** Hadoop Distributed File System
- **ID** Intrusion Detection
- **IoT** Internet of Things
- **K-NN** K-Nearest Neighborhood
- **LDA** Latent Dirichlet Allocation
- **LED** Light emitting diode
- **LSE** Least Square Estimation
- **LogR** Logistic Regression
- **MAE** Mean Absolute Error
- **MAPE** Mean Absolute Percentage Error
- **ML** Machine Learning
- **MLR** Multiple Linear Regression
- **MPI** Message Passing Interface
- **MRE** Mean Relative Error
- **MSE** Mean Squared Error
- **NB** Naive Bayes
- **NLPC** Non-Linear Principal Component
- **NLPCA** Non-Linear Principal Component Analysis

- **NN** Neural Networks
- **OCC** One-Class Classification
- **OCSVM** One-Class SVM
- **OD** Outlier Detection
- **PCA** Principal Component Analysis
- **PDF** Probability Distribution Function
- **PECM** Parallel Evolving Clustering Method
- **PNN** Probabilistic Neural Network
- **PRBFN** Parallel Radial Basis Function Network
- **PSO** Particle Swarm Optimization
- **PSOAANN** Auto-Associative Neural Network optimized with Particle Swarm Optimization
- **QRNN** Quantile Regression Neural Network
- **RBFN** Radial Basis Function Network
- **RF** Random Forest
- **RMSE** Root Mean Squared Error
- **RNN** Recurrent Neural Networks
- **SGD** Stochastic Gradient Descent
- **SMA** Social Media Analysis
- **SMOTE** Synthetic Minority Oversampling Technique
- **SNA** Social Network Analysis
- **SNAP** Stanford Network Analysis Project
- **SOM** Self Organizing Map
- **SVD** Singular Value Decomposition
- **SVM** Support Vector Machine
- **TF-IDF** Term Frequency–Inverse Document Frequency
- **TN** True Negative
- **TP** True Positive
- **TPR** True Positive Rate
- **UCI** University of California, Irvine
- **WNN** Wavelet Neural Network

Chapter 1

Introduction

This chapter presents an introduction to the thesis. This chapter provides the motivation and necessary context for the work presented in this thesis. The chapter begins with the introduction to Big Data and Big Data Analytics. It presents different application domains where big data analytics is performed. Later, it presents the motivation of work, defines the problem statement, describes the problems found in big data analytics, and lists the contributions of the thesis related to the problem statement.

The organization of the chapter is as follows: Section 1.1 presents an introduction to big data analytics. Section 1.3 presents the motivation that has driven the research in the area of big data analytics. The problem statement is outlined in Section 1.2. The organization of the whole thesis is presented in Section 1.4.

1.1 Introduction to Big Data and Big Data Analytics

1.1.1 Introduction to Big Data

Before moving to the introduction of Big Data, let the data be defined. Informally the data can be defined as, *the numerics, characters, or symbols on which several operations can be performed by a system, which can be stored and communicated in the form of electrical or optical signals and stored on magnetic, optical, or mechanical storage media.*

Big data is also about the data, which is of enormous volume. Big data is the term used for data, which is so huge and complex that none of the traditional data management tools can store or process the data efficiently. The multi-faceted challenge includes capturing or collection of data, curating the missing values or error present in data, storing the data, searching the data, sharing the data over memory, processes and network, transferring the data over the network, analyzing the data, and visualization of the data. Big data results from IoT sensor data, social media, e-commerce transactions, enterprise data, and public data.

Some examples of Big Data can be viz.

- About 1 TB/day new trade data is generated by the New York Stock Exchange.
- Statistics present that everyday 500+ TB of new data in the form of audio, video, image, and text are uploaded into the databases of Facebook.
- A single Jet engine can synthesize 20+ TB of data in 1 hour of flight time.
- The Walmart processes customer transactions that surpass 1 million every hour.
- The Tweeter receives 230+ millions of tweets every day.
- 48 hours of new video is uploaded to YouTube by its users every minute.
- The recommendation system of Amazon processes 15 million customer-click-stream data per day.
- The mailing systems analyze 294 billion emails that are sent every day to find the spams.
- Modern automobiles have numerous sensors for monitoring fuel level, tyre pressure, etc. Each vehicle generates a huge volume of sensor data.
- Weather monitoring system, defense surveillance, satellite imagery, and several other applications.

The above are some of the examples of Big data. The size of data generated by human-to-human interaction, machine-to-machine interaction, and human-to-machine interaction on social media itself is of huge volume.

The Big data may be of three different forms, i.e., (i) Structured, (ii) Unstructured, and (iii) Semi-structured.

The storage, access, and processing of any data if happen to be in the form of fixed-format is termed as a 'structured' data. The storage of data in a relational database management system (RDBMS) is an example of 'structured' data. Here, the data is stored with a fixed schema and which makes it easy to process. The data with unknown form, that can not be stored with RDBMS and can not be processed without converting into a structured format; is known as unstructured data. Apart from size being voluminous, unstructured data presents multiple challenges in terms of its processing for getting insights out of it. A typical example of unstructured data is a heterogeneous data source such as data uploaded by Facebook users containing a combination of simple text, images, videos, etc. Semi-Structured Data does not have a formal structure, i.e., like a given schema in a relational DBMS. Still, it has some structural properties like tags and other markers to group the data having similar semantics that makes it convenient to process. The examples of semi-structured data are XML files or JSON documents.

Big data has the following characteristics:

1. **Volume -** The name Big Data itself represents to an enormous size. Hence any data which is having a large size is considered to be big data, and 'Volume' is one of the significant features for Big Data. International Data Corporation (IDC) estimates 175 Zettabytes of data will be generated annually by 2025. It has been projected by IDC that the amount of generated digital data (also called Datasphere) will grow from 33 ZB in 2018 to 175 ZB by 2025, as shown in the Figure 1.1 [1].

2. **Velocity -** The term 'Velocity' indicates the speed of generation of data. Big data velocity refers to the speed of data generation and the flow of data from the source of generation. The flow of data is massive and continuous. If such velocity can be handled, then insights can be generated out of it, and decisions can be taken based on real-time data.

3. **Variety -** The term 'Variety' refers to heterogeneous sources of data generation and the nature of data, i.e., structured, unstructured, and semi-structured. Earlier, spreadsheets and databases were the only sources of data. Nowadays, data is getting generated in the form of emails, photos, videos, audios, PDFs, and many other file formats. This variety of unstructured data poses specific issues for storage, mining, and data analysis.

4. **Veracity –** The term 'Veracity' indicates the uncertainty present in the availability of the data due to data inconsistency and incompleteness. In the dataset, there are many missing values and many data which are inconsistent according to the business rule. This inconsistency and incompleteness is Veracity. The Big Data from different sources poses a challenge in maintaining accuracy and quality of data where the volume of Big Data plays a prime role in the veracity of data. For example, Twitter posts with hashtags, abbreviations, typos, and colloquial speech.

5. **Value -** The term 'Value' indicates the worth associated with the data. Unless the big data collected results into monetary worth; it is useless. Turning the big data into value means adding to the benefits of the organizations by adding to their profits.

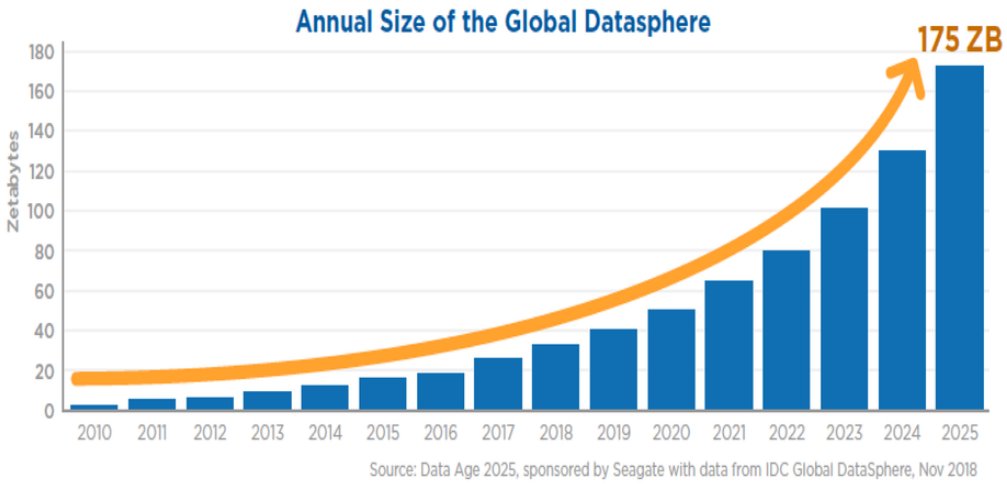

Figure 1.1: *Year-wise growth of Global Datasphere*

1.1.2 Application of Big Data

Now, Big Data has touched our lives in every aspect. The industries in almost all the domains today are reaping the benefits of Big Data applications in multiple ways. A few of them are listed here.

- **Smarter Healthcare:** The healthcare units analyze petabytes of patient's health data to extract insights out of it and then build applications for predicting the patient's health condition.
- **Telecom:** Telecom industries collect data, analyze it, and present solutions to different problems. With the help of Big Data applications, telecom companies are able to predict network failure, network load balance and thus have been able to significantly reduce data packet loss, and thus, providing a seamless connection to their customers.
- **Retail:** The retail industry has received the maximum benefits from Big Data. The retail industry has tried to understand its customer by analyzing the consumer behavioral data. Amazon's recommendation engine provides suggestion of a product based on past browsing history data of the consumer. Similarly, Netflix suggests a movie to a customer based on his past viewing preferences.
- **Traffic control:** Worldwide, many cities face the problem of traffic congestion. The effective analysis of sensor data with meaningful insights will be vital in managing traffic.
- **Manufacturing:** Analyzing data collected from the sensors in the production chain in the manufacturing industry can reduce component defects, improve product quality, increase efficiency, and save time and money. Apart from that inventory data, supply chain data all need big data analytics for the optimal demand-supply management.
- **Search Quality:** Whenever we are searching for some information on Google search engine, we are also generating data for Google in parallel.

Google saves this data and analyzes it to improve the search quality and consumer experience.

Some challenges of Big Data can be listed as follows:

(i) **Data Quality -** The major challenge is posed by Veracity. The ingested data from the source is very messy, inconsistent, and incomplete.

(ii) **Discovery –** Extracting insights from the high volume of data with multiple variety to receive value from it ultimately is a challenging task. Analyzing the petabytes of data with machine learning algorithms to find patterns and insights are very difficult.

(iii) **Storage –** When the collected data becomes enormous, it leads to more complex challenges of its management and maintenance. It presents the requirement of a storage system that can easily scale up or down on-demand.

(iv) **Analytics –** In Big Data, we may not be sure of the type and format of the data we are dealing with, so processing data with such variety is a great challenge.

(v) **Security –** As the data is voluminous, maintaining its security is also a challenge. It includes user authentication, restricted access, maintaining data access history, proper data encryption, etc.

(vi) **Lack of Talent –** Multiple application domains present their own Big Data projects. That requires a skillful team of developers, data scientists, and analysts with domain knowledge, which is the challenge still faced by the industry.

1.1.3 Big data analytics

Big Data Analytics (BDA) is the process of analyzing big data and derive insights out of it. BDA involves exploratory analytics, predictive analytics, and prescriptive analytics. For exploratory analytics, specific tools are used for Big Data visualization with their challenges and limitation. Supervised and unsupervised machine learning algorithms amenable for BDA is employed for prescriptive analytics. The

ultimate aim of BDA is to provide Business insights from the analyzed data to move forward in the business.

In order to perform BDA, we have to consider the task at hand and depending on that, we can utilize horizontal scaling or vertical scaling. We have to consider the cluster structure, the computational framework, the application tool to implement the task. The cluster should be fault-tolerant, robust, scalable, and cost-effective.

Some of the prominent challenges involved in BDA are listed below:

- Right degree of horizontal and vertical scaling for the hardware requirement for the BDA.
- Selection of a computational framework and the right tool for data mining or machine learning task.
- Tweaking the resources in the cluster and parameters in the application for optimal performance.
- Voluminous data and the velocity of its generation poses a challenge to its analysis.
- Selection of proper machine learning algorithms that can be parallelized for optimal performance.
- Preprocessing of a huge volume of data with high dimensionality poses its challenge.

1.2 Problem statement: Problems found in the area of BDA

Based on our literature survey, we defined problem statement according to the gaps identified in the area of big data analytics.

In view of the foregoing and based on our literature survey, we defined problem statement according to the gaps identified in the area of big data analytics.

1. To develop a parallel, distributed architecture to perform a regression task in a Big Data paradigm.

2. To implement a one-class classification algorithm for an imbalanced dataset in a parallel, distributed environment.

3. To develop an incremental clustering method that is amenable for a large volume of data in a parallel, distributed architecture.

4. To develop a novel approach for regression where the task can be achieved through one pass and suitable for voluminous data.

5. To implement an architecture in the parallel, distributed environment for the classification of large volume data with one pass.

6. To implement an architecture that is amenable for big data and able to perform regression as well as classification.

1.3 Motivation

The big data analytics requires special hardware architecture and software frameworks to process the huge volume of data. This constraint requires a parallel distributed computational framework to implement a parallel version of a machine learning algorithm to perform the analytics to extract the knowledge out of it. Because of the nature of the Big Data, a single computer is inadequate and inefficient for storage and processing the data in most cases. To provide the requirement of high processing power and storage space to analyze Big Data, computer clusters is a suitable choice. A cluster managing software aggregates the resources of many smaller individual machines to provide Big Data processing requirements. The following key factors should be present in a cluster.

- **Resource Pooling:** The resource pooling indicates aggregating available computing resources, e.g., CPU, memory, and storage space. Processing Big Data requires all three of the above resources in combination.

- **High Availability:** Clusters should provide high fault tolerance and the high availability. The cluster availability assures successful access of data and its processing instead of hardware or software failures. This fault tolerance

and availability becomes highly essential for real-time and quasi- real-time analytics.

- **Easy Scalability:** Clusters should be made easily horizontally scalable by adding machines to the group. This scalability indicates the system is adaptable to the changes in resource requirements.

Using clusters requires a solution for managing cluster membership, coordinating resource sharing, and scheduling actual work on individual nodes. Cluster membership and resource allocation can be handled by cluster manager software like Spark's standalone cluster manager, Hadoop's YARN, or Apache Mesos.

Looking into the requirements, the Spark standalone cluster has been selected to carry out all the research experiments, but one where the Hadoop MapReduce environment was chosen to perform the experiment. The different work-based research works are presented here as separate chapters, which is depicted by the overarching diagram (ref. Figure 1.2) of the thesis.

1.4 Organization of the thesis

The rest of the thesis is organized in 8 chapters, including a concluding chapter. The content of each of these chapters is summarized below:

1.4.1 Chapter 2: Literature Review

This chapter presents a comprehensive review of the articles related to data mining tasks viz., association rule mining, regression, classification, clustering, outlier detection, and recommendation, which are implemented with Hadoop MapReduce and Spark. This chapter discusses the articles related to big data analytics in horizontally scaled out environments on different data mining tasks and also some studies related to them. We summarized each of the surveyed articles in four aspects viz. the computational framework employed, data mining task addressed, machine learning technique implemented, analyzed dataset details. We included the obtained results, metrics for the evaluation of the model, and indicated future directions along with our critical reviews for each data mining task. We provided a detailed list of

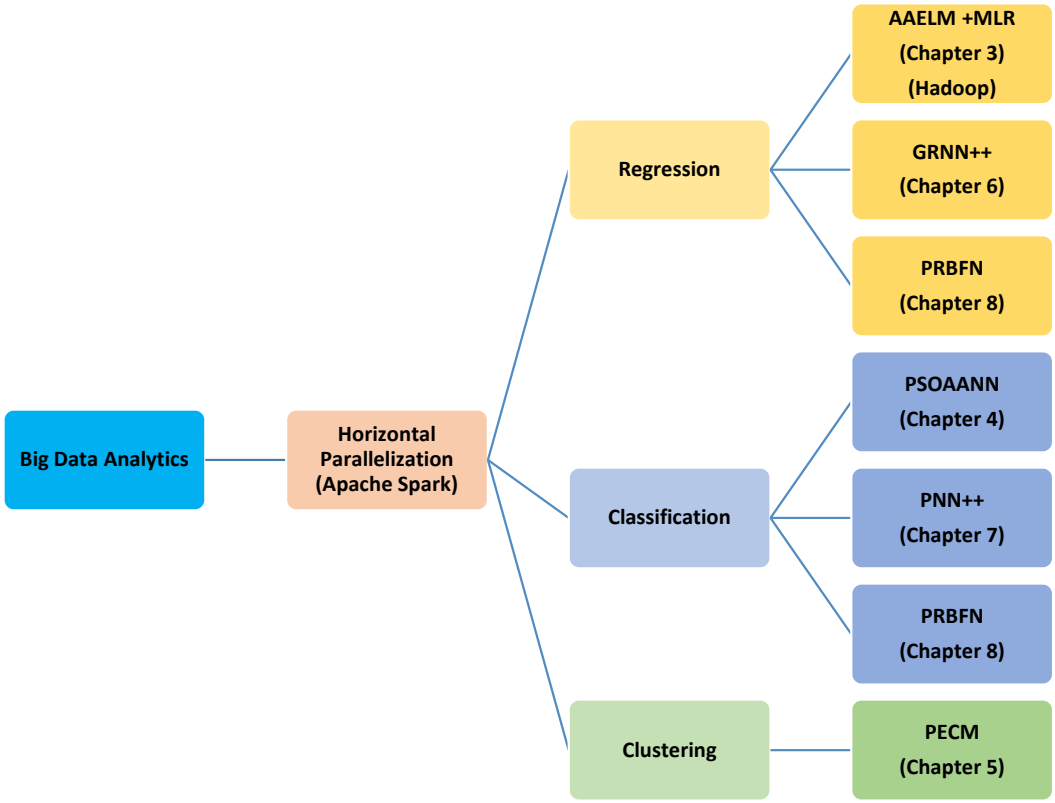

Figure 1.2: *Overarching Diagram for the Thesis*[†]

[†]AAELM: Auto-associative Extreme Learning Machine, MLR: Multiple Linear Regression, GRNN: General Regression Neural Network, PRBFN: Parallel Radial Basis Function Network, PSOAANN: Auto-associative Neural Network optimized with Particle Swarm Optimization, PNN: Probabilistic Neural Network, PECM: Parallel Evolving Clustering Method

analyzed real and synthetic datasets with references to the articles. We provided a list of employed machine learning techniques and evaluation metrics with the references to the articles. We also presented detailed exploratory analytics of the reviewed articles.

1.4.2 Chapter 3: Parallel Distributed Hybrid Regression Model

We propose a hybrid model that combines the AAELM with MLR (AAELM+MLR) for performing big data regression. It works using Hadoop MapReduce parallel computing model, which is implemented in Python using Dumbo API. It works in two phases. In the first phase, three-layered AAELM is trained. The output of the hidden nodes of AAELM is treated as NLPCs. In the second phase, the MLR model

is fitted using these NLPCs as input variables. Effectiveness of the AAELM+MLR model is demonstrated on two large datasets viz., airline flight delay dataset and gas sensor array dataset, taken from the web. It is observed that AAELM+MLR outperformed the MLR model by yielding less average Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE) values under the 10-fold cross-validation framework. A statistical test confirms its superiority at a 1% level of significance.

1.4.3 Chapter 4: Parallel Distributed One-class Classifier

Banking and financial industries are facing severe challenges in the form of fraudulent transactions. Credit card fraud is one example of them. To detect credit card fraud, we employed the one-class classification approach in the Big Data paradigm. We implemented a hybrid architecture of Particle Swarm Optimization and AutoAssociative Neural Network for one-class classification in the Spark computational framework. In this chapter, we implemented the parallelization of the autoassociative neural network in the hybrid architecture.

1.4.4 Chapter 5: Parallel Distributed Incremental One-pass Clustering Method

A novel parallel implementation of the ECM is proposed in this chapter. The original serial version of the ECM is the clustering method, which computes online and with a single-pass. The parallel version of ECM (Parallel ECM or PECM) is implemented in the Apache Spark framework, which makes it work in real-time. The parallelization of the algorithm aims to handle a dataset with a large volume. Many of the extant clustering algorithms do not involve a parallel one-pass method. The proposed method addresses this shortcoming. Its effectiveness is demonstrated on a credit card fraud dataset (with size 297MB), and a Higgs dataset was taken from Physics pertaining to particle detectors in the accelerator (with size 1.4GB). The experimental setup included a cluster of 10 machines having 32 GB RAM each with Hadoop Distributed File System (HDFS) and Spark computational environment. A remarkable achievement of this research is a dramatic reduction in computational time compared to the serial version of the ECM.

1.4.5 Chapter 6: Parallel Distributed One-pass Regression Model

Among the neural network architectures for prediction, Multi-layer Perceptron (MLP), RBFN, wavelet neural network (WNN), GRNN, and group method of data handling (GMDH) are popular. Out of these architectures, GRNN is preferable because it involves single-pass learning and produces reasonably good results. Although GRNN involves single-pass learning, it cannot handle big datasets because the pattern layer is required to store all the cluster centers after clustering all the samples. Therefore, this chapter proposes a hybrid architecture, GRNN++, which makes GRNN scalable for big data by invoking parallel, distributed clustering methods viz., K-Means|| in the pattern layer of GRNN. The whole architecture is implemented in the parallel distributed computational architecture of Apache Spark with HDFS. Further we proposed the extension of GRNN++ (EGRNN++) employing another clustering approach, parallel Bisecting K-Means, in place of K-Means||. Also, we proposed to employ Logistic, Cauchy activation functions in the pattern layer of EGRNN++ in place of the Gaussian activation function. The effectiveness of the variants of EGRNN++ was tested on two datasets taken from Chemistry and Amazon Movie Review dataset for customer review rating prediction under ten-fold cross-validation (10-FCV) setup. The proposed variants of EGRNN++ produced very low Mean Squared Error (MSE). It is worthwhile to mention that the primary objective of this article is to present a distributed and parallel version of the traditional GRNN to handle big datasets.

1.4.6 Chapter 7: Parallel Distributed One-pass Classifier

The popular neural network architectures for classification are MLP, RBFN, WNN, PNN, and GMDH. Out of these, PNN is preferable because it involves single-pass learning and produces reasonably good results. Although PNN involves single-pass learning, it cannot handle big data sets because the pattern layer is required to store all the cluster centers after clustering all the samples. Therefore, this chapter proposes two hybrid architectures, PNN_1++ and PNN_2++, which make PNN scalable for big data by invoking parallel, distributed clustering methods viz., K-Means||

and parallel Bisecting K-Means respectively, before the pattern layer of PNN. The whole architecture is implemented in the distributed parallel framework of Apache Spark with HDFS. The performance of the variants of PNN++ was demonstrated on HEPMASS, HIGGS, ccFraud, and Amazon Movie Review dataset under a ten-fold cross-validation setup. The proposed variants of PNN++ produced high AUC. They also turned out to be statistically the same with the help of a t-test conducted at a 1% level of significance and 18 degrees of freedom. It is worthwhile to mention that the primary objective of this article is to present a distributed and parallel version of the traditional PNN to handle big datasets.

1.4.7 Chapter 8: Parallel Distributed Versatile Architecture for Regression and Classification

RBFN is one of the versatile neural network architectures, in that it can be used to solve regression as well as classification tasks. The proposed work implements RBFN in a parallel and distributed environment of Apache Spark. The new version of RBFN is referred to as PRBFN. The main contribution in this work is embedding of K-Means|| or Bisecting K-Means (parallel, distributed clustering methods) in between input and hidden layer and implementation of parallel least square estimation employing outer product of matrices. The PRBFN employed Gaussian as well as Logistic activation functions in the hidden layer for non-linear transformation. The effectiveness of the PRBFN was tested on HEPMASS, HIGGS, ccFraud, and Amazon Movie Review (for sentiment classification) datasets under 10 Fold Cross-Validation (10-FCV) setup for classification problem. Again, the PRBFN was tested with gas sensor array dataset and Amazon Movie Review (for movie review prediction) datasets under 10-FCV setup for a regression problem. It is emphasized that the chief objective is to present a distributed and parallel version of the RBFN to handle regression and classification in the Big Data paradigm.

1.4.8 Chapter 9: Conclusions and future directions

Chapter 9 summarizes the contributions of the thesis to the research community in the parallel distributed computational framework for big data analytics in multiple

data mining tasks viz., regression, binary classification, one-class classification, and clustering. The chapter also presents a list of gaps in the current literature as the future directions, which can be addressed by the budding researchers.

Chapter 2

Literature Review

In this chapter, articles involving parallel distributed computation for different data mining tasks employing machine learning techniques are reviewed. The critical reviews are presented, and research gaps are suggested. At the end of the review, in the open problems section, the research opportunities for the new research work are identified. The chapter discusses the opportunities and challenges present in data analytics in the BFSI sector. The chapter also presents the literature study, for the different research work carried out in the further chapters. The next section of the chapter presents an introduction to the current review work. Section 2.2 lists out the previous review works carried out by the community in the big data area. The review methodology that has been conducted for the current review work has been presented in Section 2.3. Section 2.4 presents a review of the articles that were included in the review process. Section 2.5 discusses the insights gained out of the review work. Section 2.6 presents the conclusions and open problems.

2.1 Introduction

In the current digital era, there is enormous progress in the computational world; in other words, there is a spectacular growth of the internet. These developments, in return, account for a huge volume of data in a structured, semi-structured, and unstructured manner. These huge volume of data are generated by people, machines, and their interactions. So today, we have a gold mine of data from which

hidden insights can be drawn. These large voluminous data are generated in various forms like structured, semi-structured, and unstructured data. These data are also produced in a very fast manner from different types of the internet of things (IoT) devices and other digitization sources. This kind of data generation has led to the definition of Big Data defined by Gartner which states,

> "Big data is high-volume, high-velocity and/or high variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation [2]."

The voluminous data can be attributed to business sales records, the results gathered from scientific experiments, or real-time sensors used in the IoT. Data may also exist in a wide variety of file formats, including structured data, unstructured data, or streaming data. Further, Big Data may involve multiple, simultaneous data sources, which may not otherwise be integrated, resulting in a source of a variety of data formats. Finally, velocity refers to the speed at which big data is generated, processed and analyzed. Every Big Data Analytics (BDA) project will ingest, process, and analyze the data, and then provide an insight based on an existing business problem.

McKinsey Global Institute specified the potential of Big Data in five main topics: Healthcare, Public Sector, Retail, Manufacturing, and Personal location data [3].

The Big Data computation requires a scalable solution with distributed parallel computation. The scaling is defined as the ability of the system to adapt to increased demands regarding data processing. The BDA architectures can be classified into two types of scaling presented below:

- **Horizontal Scaling**: It involves distribution of the workload across multiple machines, which may be even commodity hardware. It is also known as "scale-out", where multiple independent computing devices are added together to increase the processing capability [4].
- **Vertical Scaling**: It involves increasing the number of processors, amount of memory, and installing other faster hardware, typically, within a single

machine. It is also known as "scale up", and it usually involves a single machine [2].

As part of this review chapter, we would like to address the following research questions:

Q1: Can statistical and machine learning algorithms meant for data mining tasks be made scalable for big datasets in Hadoop and Spark frameworks?

Q2: If 'yes' what are they and why they are scalable?

Q3: Can some pointers be suggested using which any researcher can attempt parallelizing hitherto unparallelized ML techniques?

These research questions shall be answered in Section 2.5 after the review of the papers.

In this chapter, we reviewed the papers concerned with a distributed parallel computing environment implemented with horizontal scaling. The review is performed, keeping in mind the different data mining tasks with the concerned machine learning techniques. The critical analysis of the papers is furnished, pointing the areas for future research work. There are some dedicated Big Data journals that came into existence in the recent past, viz. **Big Data Research**, **Journal of Big Data**, **IEEE Transactions on Big Data**, **International Journal of Big Data Intelligence**, **Big Data**. Articles from these journals were collected apart from the journal and proceeding articles from well-known publishers.

2.2 Existing Related Review Works

The current literature presents many review articles dealing with various aspects of big data processing. Some papers discussed hardware platform issues. The scope and challenges in different hardware platforms with different configuration and their performance was analyzed. Singh and Reddy [4] examined different platforms available for performing BDA. They reviewed different hardware platforms available for BDA and estimated the advantages and drawbacks of various platforms based on different metrics such as scalability, data I/O rate, fault tolerance, real-time processing, data size supported, and iterative task support. Saecker and Markl [5] presented a review of hardware architectures used for BDA. The authors

considered the horizontal and vertical scaling with the technical aspect for the BDA. Chen et al. [6] in the survey work reviewed the related technologies of Big Data such as cloud computing, Internet of Things, data centers, and Hadoop. The survey work highlighted the four phases of the value chain of Big Data, i.e., data generation, data acquisition, data storage, and data analysis. The review included the technical challenges and the latest advances for each phase. Liu et al. [7] presented an analysis of the opensource technologies that support big data processing in a real-time or near real-time fashion. They analyzed their system architectures and platforms.

There are some review works concerning the software aspects as data models supporting Big Data, discussing risks and challenges in Big Data computational environments. In addition, there are studies concerning the analysis of MapReduce computational environment, with its challenges. Sharma et al. [8] presented a review paper on leading Big Data models that are leading contributors to the NoSQL era and claim to address Big Data challenges in reliable and efficient ways. The survey paper compares the features of different data models serving for big data analysis. Al-Jarrah [9] presented a review of the theoretical and experimental data-modeling literature in large-scale data-intensive fields. The study is related to model efficiency while learning, the new algorithmic approaches with the least memory requirements, and the processing power to minimize computational cost while maintaining/improving its predictive/classification accuracy and stability. Sagiroglu and Sinanc [10] presented an overview of Big Data's content, scope, samples, methods, advantages, and challenges and discussed privacy concerns on it. Sri and Anusha [11] presented a survey of Big Data where they analyzed the Hadoop architecture, including HDFS and MapReduce environment and its security issues. Khan et al. [12], Ekbia et al. [13], Chen and Zhang [14] had reviewed the challenges, techniques, and technologies involved in the Big Data computational environment. The survey work also included the risks and security issues involved with the Big Data environment. Lee et al. [15] presented a review paper on the parallel data processing with MapReduce. They have analyzed various technical aspects of the MapReduce framework. The inherent pros and cons were discussed, along with the introduction to the optimization strategies for MapReduce programming. Ward and Barker [16] presented a survey on Big Data definitions. The review paper has collated various

definitions that have gained some degree of attraction and had furnished a clear and concise definition of the term '*Big Data*.'

The research community has also presented some review work considering individual data mining (DM) tasks like clustering, data analysis aids like visualization methods, and studied the in-memory data management system. Shirkhorshidi et al. [17], Fahad et al. [18] presented a survey of clustering algorithms for Big Data. The survey work included analysis of the improved clustering algorithms for the Big Data paradigm. Gorodov and Gubarev [19] presented a review of data visualization methods in application to Big Data. They reviewed the existing methods for data visualization in application to Big Data. The survey work included the classification of visualization methods in application to Big Data. Zhang et al. [20], presented a review of in-memory data management and processing on Big Data paradigm. The survey involved a broad range of in-memory data management and processing systems, including both data storage systems and data processing frameworks.

Table 2.1 presents the details of all the previous review papers. The following are the key reasons which necessitated writing the current, unique review work.

(i) The existing literature on the review of Big Data covered the hardware platforms, data models, challenges, risks, and security issues in BDA, data visualization, and a DM task like clustering.

(ii) It is evident that there is no survey paper existing, which exclusively deals with data mining tasks with different machine learning techniques applicable to parallel distributed programming considering horizontal scaling platform, viz., Apache Hadoop MapReduce and Apache Spark framework.

(iii) This review, by virtue of its theme, provides a plethora of opportunities to budding researchers in parallelizing various DM tasks applied to any domain. This will allow researchers and practitioners to not only employ the extant scaled versions of the popular ML techniques for data mining tasks in diverse domains but also suggest pointers to develop scaled versions of hitherto unparallelized algorithms.

Therefore, the present review will be a significant contribution to the BDA research from the viewpoint of parallelizing analytical techniques.

Table 2.1: Existing Review Articles

Ref.	Journal / Conference	Journal / Proceeding Name	Publisher	Scope of Review
[4]	Journal	*Journal of Big data*	Springer	Hardware Platforms
[5]	Conference	European Business Intelligence Summer School (eBISS), 2012	Springer	Hardware architectures
[6]	Journal	*Mobile Networks and Applications*	Springer	Challenges, and the latest advances in data generation, data acquisition, data storage, and data analysis
[7]	Conference	International Database Engineering & Applications Symposium, 2014	ACM	Technologies that support big data processing in a real-time or near real-time fashion
[8]	Journal	*Data Science Journal*	Ubiquity Press	Data Models
[9]	Journal	*Big Data Research*	Elsevier	Theoretical and experimental data-modeling literature for model efficiency while learning with least memory requirements
[10]	Conference	Int. Conf. on Collaboration Technologies and Systems	IEEE	Scope, methods, advantages, challenges and privacy concern of big data

Continued on next page…

Ref.	Journal / Conference	Journal / Proceeding Name	Publisher	Scope of Review
[11]	Journal	*Indonesian Journal of Electrical Engineering and Informatics*	Institute of Advanced Engineering and Science (IAES)	HDFS and MapReduce environment and its security issues
[12]	Journal	*The Scientific World Journal*	Hindawi	Challenges, techniques, and technologies involved in the big data computational environment. Risks and security issues.
[13]	Journal	*Journal of the Association for Information Science and Technology*	Wiley	
[14]	Journal	*Information Sciences*	Elsevier	
[15]	Journal	*ACM SIGMOD Record*	ACM	Various technical aspects of the MapReduce framework
[16]	Journal	*CoRR*	-	Definition of big data
[17]	Conference	-	Springer	Clustering algorithms for big data
[18]	Journal	*IEEE Transactions on emerging topics in computing*	IEEE	
[19]	Journal	*Journal of Electrical and Computer Engineering*	Hindawi	Data visualization methods in application to big data

Continued on next page…

Ref.	Journal / Conference	Journal / Proceeding Name	Publisher	Scope of Review
[20]	Journal	*IEEE Transactions on Knowledge and Data Engineering*	IEEE	In-memory data management and processing on big data paradigm
End of Table				

2.3 Review Methodology

The review covers all the articles in the different Data Mining (DM) tasks that discuss different Machine Learning (ML) techniques which are carried out in the parallel distributed computational environment of Apache Hadoop/Spark. Thus, the scope of the review can be considered as the intersection of these three (Figure 2.1). The current review work has followed four main steps (Figure 2.2) to bring the review work to its final form. The steps are Review definition, Search of Articles, Filtration of articles for review, and Analysis of the review process.

Step 1: *Review Definition*

Step 1.1: *Scope of review*. The present literature survey covers the research articles related to the data mining tasks employing different machine learning techniques in the Big Data paradigm. The review paper has focused only on the research articles related to parallel distributed processing with horizontal scaling. Thus, research articles related to Hadoop MapReduce and Spark are only considered for the review process.

Step 1.2: *Goal of the review process*. The review chapter aims to provide analytical insights from past papers with indications to the research gaps and suggesting directions for future work.

Step 2: *Search for Articles*

Step 2.1: *Scope of the articles used for the review process*. The articles are about Data Mining (DM) tasks, viz. Association Rule Mining (ARM), Regression, Classification, Clustering, Outlier detection, Recommendation along with the associated machine learning techniques form the focus of the present review. Further,

we restrict our attention to only horizontal scaling, i.e., parallel distributed processing with a cluster of nodes. Hence, the review process considered only Hadoop and Spark-based papers (Figure 2.1).

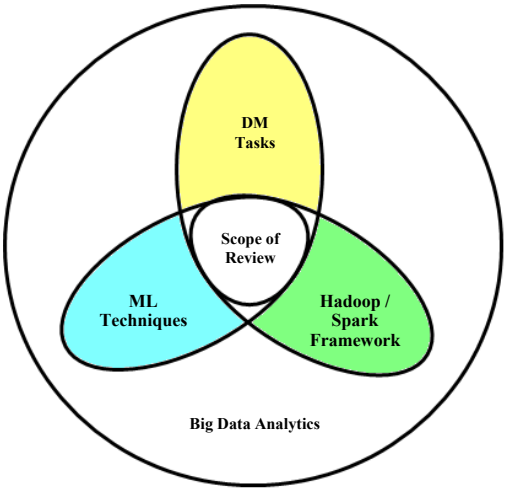

Figure 2.1: *Scope of Review*

It did not include parallelization using Field Programmable Gate-Array (FPGA), Message Passing Interface (MPI) or General-purpose graphics processing unit (GPGPU). The research works related to social media analytics; visualization was not included in the purview of the current review. This review did not focus on application domains such as healthcare, finance, bioinformatics, and telecommunication sectors. We have excluded the articles from paid journals. The edited volumes were excluded from the scope of the review.

Step 2.2: Defining *the search terms*. We refined our search using various words like "MapReduce", "Apache Spark", along with words representing data mining tasks such as, "Clustering", "Classification", "Association Rule Mining", "K-Means", "SVM (Support Vector Machine)", etc.

Step 2.3: *Scope of the online databases in the review process*. Primarily, we gathered research papers from the scientific articles databases such as ACM digital library, Taylor, and Francis, Science Direct, Wiley, Google Scholar, Springer, and IEEE-Xplore. We collected around 700 articles that were published during the 2009-2019.

Step 3: *Filtration of articles for review*

Step 3.1: *Elimination of unrelated articles*. The scope of the articles was defined, and the articles were filtered with the specified criteria.

The collection and filtration process resulted in 261 papers to review for this survey that was published during 2009–2019 in various conferences and journals.

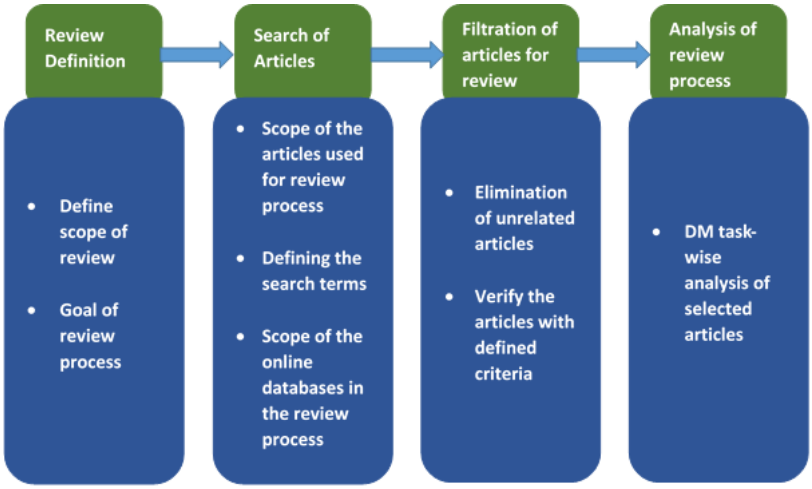

Figure 2.2: *Review Process Methodology*

Step 3.2: Verification of the articles with defined criteria. The articles were validated to have used real-world or synthetic datasets for the experiment with some results, compared with some ML techniques (Figure 2.3).

Step 4: *Analysis of review process*

Step 4.1: *DM task-wise analysis of selected papers with future research direction*. DM task-wise, the articles were reviewed and categorized with Hadoop and Spark-based articles, in chronological order. The critical analysis of the articles in each section of the DM task led to identifying research gaps and addressed for future research work. The bird's-eye view of the analysis of the reviewed articles with some suggestions for future research work is presented after the discussion of the reviewed articles.

2.4 Review of the articles

The review process involves articles related to distributed parallel computing environment of Hadoop and Spark with different DM tasks with ML techniques (Fig-

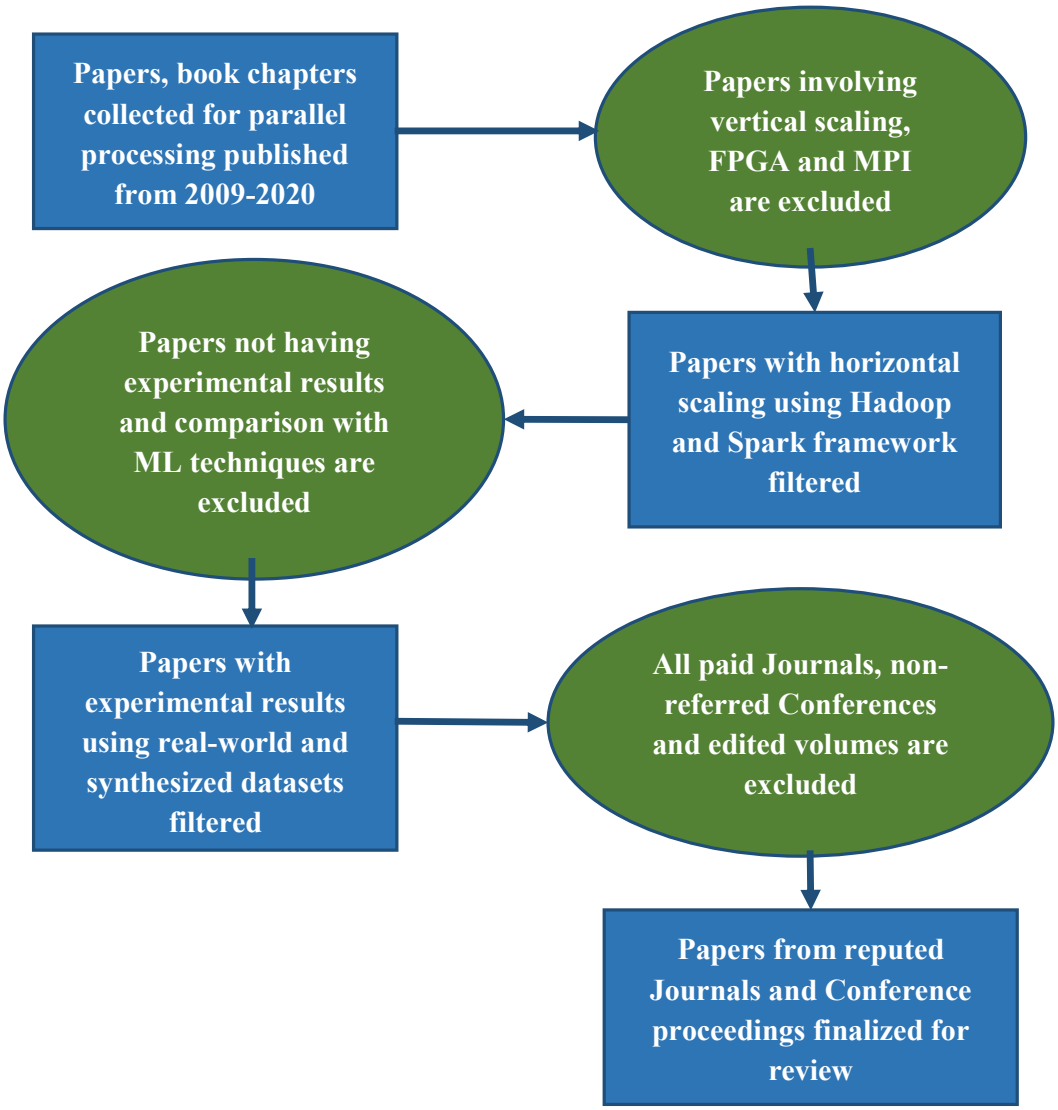

Figure 2.3: *Filtering Procedure for Selection of Articles to Review*

ure 2.1). Various data mining tasks with the machine learning techniques used for distributed parallel processing covered in the review process are depicted in Figure 2.4.

2.4.1 Association Rule Mining / Pattern Mining

2.4.1.1 Hadoop MapReduce-based Journal papers

Karim et al. [21] proposed an approach for extraction of frequent patterns (FPs) which is of maximum length and present contiguously. The concerned works included *Human Genome* (HomoSapiens GRCh37.64 DNA Chromosome Part 1), and *Bacteria DNA sequence* datasets downloaded from the National Center of Biotechnology Information Gene Expression Omnibus (NCBI GEO) repository. The results proved the approach to be efficient.

Karim et al. [22] presented a distributed model for extraction of useful patterns in large datasets. The proposed method was applied to the real world datasets, viz. *Connect-4*, *Mushroom*, and *Agaricas* were downloaded from UCI repository [23]. Experimental results exhibited that the proposed technique was efficient and scalable.

Bhuiyan and Al Hasan [24] advocated a frequent subgraph extraction method (FSM-H) implemented on MapReduce framework. The experimental work was carried out with the following datasets: (i) real-world graph datasets collected from a website [25] containing graphs from the PubChem [26], (ii) graph dataset from DBLP [27], (iii) synthetic datasets from Graphgen [28]. The results presented significantly better performance of the proposed FSM-H over that proposed by Hill et al. [29].

Xun et al. [30] proposed an approach FiDoop-HD, an extension of FiDoop, for parallel mining of frequent itemsets. The FiDoop-HD was evaluated with a synthetic dataset *D1000W* was generated using the IBM quest market-basket synthetic data generator, and a real dataset *Celestial Spectral Dataset* was used. The results presented that the suggested approach was sensitive to both data distribution and dimensions.

Salah et al. [31] suggested a method called Parallel Highly Informative K-ItemSet (PHIKS). The authors used three real-world data sets: the *complete 2013*

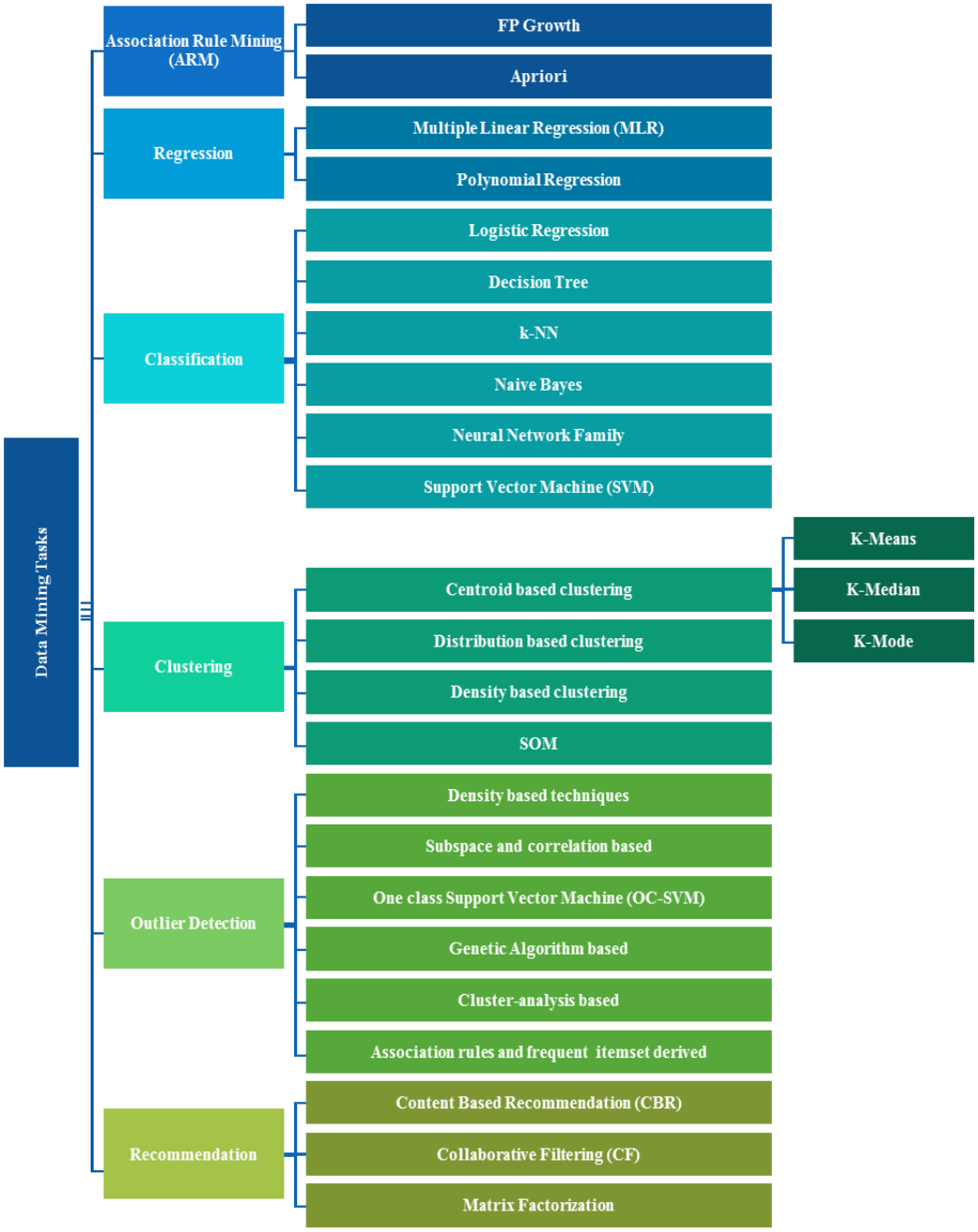

Figure 2.4: *Data Mining Tasks and Machine Learning Techniques parallelized for BDA*

Amazon Reviews dataset [32], the *2014 English Wikipedia* dataset [33], *a sample of ClueWeb English* dataset [34]. The proposed approach, PHIKS outperformed an alternative parallel approach.

Breier and Branišová [35] have proposed a dynamic rule creation based on an anomaly detection method for recognizing security breaches in log files. The datasets used were *1998 DARPA Intrusion Detection Evaluation Set* [36] and *Snort logs* [37] created by analyzing the DARPA dataset. The proposed approach found to perform better than FP-growth and Apriori methods.

Yan et al. [38] employed a parallel method for the constrained frequent pattern. The experimental work involved the *SDSS star spectrum* datasets. Experimental results claimed the proposed method to be scalable, expandable and with availability.

Leroy et al. [39] introduced item-centric mining, a new semantics for frequent itemset mining over long-tailed datasets. The proposed algorithm, TopPI, was compared with parallel FP-Growth. The research study involved the analysis of two real-world datasets, viz., *Supermarket*, and *LastFM*. The experimental outcomes presented the proposed method to be effective over standard itemset mining and global top-k algorithms.

Sohrabi and Taheri [40] proposed a parallel version of PrePost+ algorithm (HBPrePost+) based on Hadoop for frequent itemset mining. The experimental work was carried out with four datasets, viz., *pumsb*, *connect*, *chess*, and *mushroom* from Frequent Itemset Mining Implementations (FIMI) repository [41]. The experimental results showed that HBPrePost+ algorithm has a superior execution time.

Liu et al. [42] proposed a Hadoop-based Association rule mining method employing Maximal frequent itemsets mining. The experiments were conducted with synthetic datasets and real datasets collected from FIMI repository [41]. The proposed method was compared with state-of-the-art maximal frequent itemsets mining algorithms. The experimental results proved the proposed approach had less execution time with efficient memory utilization and scalability.

Singh et al. [43] proposed MapReduce based Apriori algorithms along with its optimized version. The experimental results showed that the optimized version is more efficient in terms of execution time. The above experimental work was

carried out with the synthetic dataset *c20d10k* generated by IBM Generator and real datasets, viz., *chess* and *mushroom* from FIMI repository [41].

2.4.1.2 Spark-based Journal papers

Zhang et al. [44] proposed a distributed frequent itemset mining algorithm. The proposed algorithm presented a reduction in the number of candidate itemsets. The datasets used in the research work are *T10I4D100K* and *T40I10D100K* [41]. The experimental result was compared with parallel FP growth (PFP) and found to outperform PFP in execution time for both the datasets.

Chen and An [45] proposed a Parallel High Utility Itemset (PHUI)-Miner algorithm, which is the parallel version of High Utility Itemset (HUI)-Miner algorithm. The experimental study was conducted on different datasets, viz. *kosarak*, *accidents*, *chess* [46], *twitter* [47], *T5000L10I1P10PL6*, *tafeng* [48] and *globe* where only *T5000L10I1P10PL6* was a synthetic dataset. The experimental results showed that the PHUI-Miner had superior performance to its serial counterpart HUI-Miner.

Karim et al. [49] proposed a method called Maximal Frequent Pattern for finding frequent patterns implemented with Apache Spark. The experiments were carried out with synthetic datasets generated by the IBM Quest [50]; a real dataset *Mushroom* from UCI ML data repository [23]; a real *retail dataset* [51]; and a *retail market basket data*. The experimental results indicated the superior performance of the proposed method over other state-of-the-art maximal frequent pattern algorithms.

Martín et al. [52] have proposed a generic parallel framework MRQAR for quantitative association rule mining in the Spark framework. The experimental work was carried out with *Susy*, *Higgs* dataset from UCI repository [23]; *epsilon* from LIBSVM [53] and ECBDL'14 competition. The proposed framework is to be implemented when any standard algorithm to find a quantitative association rule is not able to execute in the Big Data paradigm.

2.4.1.3 Hadoop MapReduce-based Conference papers

Hill et al. [29] proposed a method for frequent subgraph mining in biological datasets utilizing iterative MapReduce framework. The experimental study was

carried out with (i) synthetic dataset generated using a graph generator [54] and (ii) the real datasets extracted from the PubChem website [26]. The experimental results showed the proposed algorithm outperformed other methods.

Farzanyar and Cercone [55] proposed an approach for an efficient frequent itemset mining algorithm, called Improved MapReduce Apriori (IMRApriori) algorithm. The experimental study utilized IBM synthetic datasets *T10.I5.D1000K* [56] and real dataset *BMS-POS* [57]. The proposed approach had superior execution time in comparison with the MRApriori algorithm.

Farzanyar and Cercone [58] proposed a method of mining of frequent itemsets. The authors have demonstrated a comparative analysis of their proposed algorithm with the IMRApriori algorithm [55]. The authors have used synthetic *IBM dataset* for their experiments. The results demonstrated a reduction in communication overhead and execution time.

Mao and Guo [59] proposed an improved association rules mining algorithm, AprioriPMR, which involved the power set. The experimental study utilized *breast cancer* datasets from Weka. The results presented the superior performance of the proposed algorithm to the parallel Apriori algorithm regarding processing time and scalability.

Natarajan and Sehar [60] proposed an algorithm, named Association rule mining (ARM) based on Hadoop (ARMH), to extract FPs from large databases. The experimental work carried out with a dataset generated by IBM's Quest data generator [56]. The proposed algorithm had improved scalability and efficiency compared to BTP-tree.

Rong et al. [61] implemented a parallel version of the Apriori and FPGrowth algorithm for voluminous data. The study involved real datasets from FIMI repository [41]. The results showed the proposed method is efficient, scalable, and with reliability when compared to its serial counterpart.

Moens et al. [62] analyzed Frequent Itemset Mining (FIM) for significant-sized data. The experimental study involved datasets from FIMI repository [41]. The results indicated the scalability of the proposed method.

Chen et al. [63] presented a method for parallel mining FPs for large-scale data. The experimental study involved synthetic datasets generated by IBM synthetic data generator [56]. The results indicated the efficiency and scalability of the method.

Leung and Jiang [64] proposed a framework called BigAnt to extract FPs from large-scale data. The study utilized various datasets from the UCI Machine Learning Repository [23] and the FIMI Repository [41] and also the *IBM synthetic datasets*, which were generated using the IBM Quest Dataset Generator [65]. The experimental result was compared with the UF-growth algorithm [66] and found that the proposed method had superior execution time with scalability and speedup.

Yu et al. [67] designed an efficient algorithm Frequent Patterns Mining Algorithm based on MapReduce Framework (FAMR) by modifying the traditional Apriori algorithm. The experiments were carried out with data generated from the IBM data generator [56]. The experimental results showed FAMR could reduce the number of candidate itemsets and achieved an excellent reduction in execution time.

Yu et al. [68] proposed a new planted (l, d) motif discovery algorithm named MCES, which identified motifs by mining and combining emerging substrings. The experimental work was carried out with simulated data as well as real data from *mESC* (mouse embryonic stem cell) data. Experimental results showed that MCES could identify (l, d) motifs efficiently and effectively.

Zhou and Huang [69] implemented an improved parallel Apriori algorithm in which both count and candidate generation steps were employed. The experimental study utilized the synthetic data generated by an opensource tool package, Artool [70]. The results showed the proposed method had a superior performance regarding execution time.

Fumarola and Malerba [71] proposed a parallel algorithm for approximate frequent itemset mining using MapReduce called MrAdam. The experimental work was carried out with the datasets *mushroom*, *pumsb*, and *accidents*. The experimental results presented the superior performance and scalability of the proposed approach over other FP mining algorithms.

Bhuiyan and Al Hasan [24] proposed a frequent subgraph mining algorithm called FSM-H. The study involved four real-world graph datasets that were collected from the PubChem website [26]. Also, four synthetic datasets were created from a tool called Graphgen [54]. The experimental results showed that the proposed approach was efficient regarding execution time in comparison with existing methods.

Liu et al. [72] proposed a Frequent Itemset Mining Algorithm. The proposed approach used an improved Apriori algorithm that used the length of each transaction to determine the size of the maximum merge candidate itemsets. The experimental study utilized a synthetic dataset. The results indicated the proposed approach to be efficient.

Cao et al. [73] proposed an approach for mining of repetitive sequences across a collection of genomes implemented with the MapReduce framework called MRSMRS. The dataset used in the experiment contained the base pairs from the genome of six species, viz. human, mouse, rat, dog, chicken, and macaque. The results indicated that the proposed method had superior speedup, linear scalability, and less execution time.

Liu and Li [74] proposed an approach for parallel FP discovery in a noniterative manner. The experimental work utilized the file *SogouQ* provided by the Sogou lab. The proposed approach had a better execution time with comparison to algorithms bagging the idea of traditional Apriori.

Chang et al. [75] proposed the Parallel Block FP-Growth algorithm (PBFP-Growth) by combining the Apriori and FP-Growth algorithms. The experimental results proved that the performance of the PBFPGrowth algorithm was better than that of both the Apriori algorithm and the FPGrowth algorithm.

Sun et al. [76] proposed a method Vertical Apriori Map-reduce (VAMR) by aggregating MapReduce mechanism with Apriori and vertical frequent mining for extraction of frequent patterns from a large dataset. The experimental work involved datasets from FIMI repository [41] and synthesized datasets. The performance of the proposed algorithm was measured against the Apriori MapReduce and OPUS miner algorithm and found to be better than both.

Ferrucci et al. [77] proposed an architecture for developing parallel GAs. The approach was employed for feature subset selection problem. The *Chicago Crime* dataset was analyzed for experimental work. The results showed the proposed framework was superior to its serial counterpart with respect to the execution time and had comparable accuracy.

Jiang et al. [78] proposed a method for mining "Following" patterns in big social network. It was implemented with MapReduce framework and called MR-PFP. The experimental work was carried out with social network datasets from Stanford

Network Analysis Project (SNAP) [79]. The experiment compared the proposed algorithm MRPFP with its sequential counterpart, and the result showed a good speedup of the proposed algorithm.

Salah et al. [80] proposed an algorithm called Parallel Highly Informative *K*-ItemSet (PHIKS) algorithm. The experimental study utilized two real-world datasets, viz., *2014 English Wikipedia* dataset articles, *sample of ClueWeb English* dataset. The results indicated that PHIKS demonstrated superior execution time and scalability.

Liu et al. [81] proposed an algorithm for the detection of significant patterns called Pampas. The experimental study was carried out with three datasets *EMS-POS*, *connect4*, and *mushroom* collected from FIMI repository [41]. The results indicated that the Pampas was efficient, scalable on the dataset-size, and cluster-size.

Apiletti et al. [82] proposed Parallel Frequent Pattern Mining for High-Dimensional data, called PaMPa-HD. The experimental study involved the real *Kent Ridge Breast Cancer* dataset [83] and two synthetic datasets [84]. The study results showed the efficiency and scalability of PaMPa-HD, which outperformed avant-garde algorithms.

Baralis et al. [85] proposed a Parallel Weighted Itemset Mining (PaWI) algorithm. The experimental study involved a real-world dataset having a collection of 34 million of Amazon reviews. The experimental results showed that PaWI had less execution time with scalability.

Leung et al. [86] presented a model for FP analysis. The development of the model utilizes (i) IBM synthetic datasets [56], (ii) the online *retail* dataset from UCI Machine Learning Repository [23], and (iii) *ego-Facebook* dataset and *ego-Twitter* dataset from Stanford Network Analysis Project (SNAP) [79] for its evaluation. The proposed algorithm has outperformed several avant-garde algorithms.

Gonen and Gudes [87] proposed an algorithm for mining closed frequent itemsets. The study involved a real dataset *webdocs* from the FIMI repository [41] and synthetic dataset from the IBM data generator [56]. The experimental results showed the proposed algorithm had superior execution time and communication cost.

Thakare et al. [88] proposed an improved PrePost algorithm. The experimental work utilized the synthetic dataset constructed with the spawner data generator tool. *T15l106D100K*, *T15I106D1000K*, and *T15I106D2000K* were used as experimental data. The results showed that with an increase in dataset size, the proposed algorithm performed better.

Chang et al. [89] proposed an approach for parallel association rule mining, called Improved Parallel Association Rule Based on Combination (IPARBC). The experimental study involved a synthetic dataset. The experimental results showed the superior execution time of the proposed method.

Sheshikala et al. [90] proposed a parallel approach for finding a colocation pattern. The proposed method determines the spatial neighbor relationship to identify co-location instances and co-location rules. The experiment involved synthetic datasets. The experimental results showed the proposed approach was computationally more efficient.

2.4.1.4 Spark-based Conference papers

Gui et al. [91] proposed a distributed approach for frequent itemset mining employing matrix-based pruning algorithm. The research work involved synthetic dataset *T10I4D100K*. The proposed method was compared with the MapReduce-based Apriori algorithm and found to have less execution time with scalability.

Deng and Lou [92] proposed an FP-Growth algorithm. The experimental study involved datasets from the UCI ML and FIMI repository, viz. *Mushroom* [23] and *Accidents* [41], respectively. The results indicated superior efficiency of the proposed algorithm.

Rathee et al. [93] proposed an algorithm Reduced-Apriori (R-Apriori), which is an efficient apriori-based algorithm. The experimental study utilized five datasets, viz. (i) *T10I4D100K*, a synthetic dataset (ii) *Retail* dataset (iii) *Kosarak* dataset, contains the click-stream data of a Hungarian on-line news portal (iv) *BMSWeb-View2*, from KDD 2000 (v) *T25I10D10K*, a synthetic dataset. The results indicated that the proposed R-Apriori outperformed other avant-garde algorithms.

Utama and Distiawan [94] proposed a method for mining of frequent N-grams, called Spark-gram. The study included *Wikipedia articles collection* dataset col-

lected from Wikipedia dump repository. The experimental results showed that Spark-gram outperformed its Hadoop MapReduce equivalent when more frequent n-grams were considered.

Joy and Sherly [95] proposed a method for prediction of chances of heart attack using parallel frequent itemset mining. The experimental study involved dataset containing information about heart disease collected from UCI repository [23]. The results showed the proposed approach to be efficient.

The review of articles related to ARM in Hadoop and Spark framework revealed the following research insights.

- *In all the reviewed articles, we can observe that frequent itemset mining had been more than any other ARM method. The frequent itemset mining is inherently data-independent, i.e., the algorithm can be implemented in different chunks of data in a parallel manner.*
- *Frequent subgraph mining has been less explored in distributed parallel computing environments leading to a fertile ground for research work.*
- *The evolutionary computation has not been used to its full potential in BDA using a distributed framework for frequent pattern mining, which the researchers can explore.*
- *FP-growth and FP-tree are less explored areas in comparison to the Apriori method in Hadoop or Spark environment.*
- *High Utility Itemset miner presents an open space for research in the distributed parallel framework.*

2.4.2 Regression / Prediction / Forecasting

2.4.2.1 Hadoop MapReduce-based Journal papers

He et al. [96] proposed an efficient parallel ELM (PELM) for regression. The experimental work involved the *stock* dataset from UCI [23]. The results showed that the proposed algorithm, PELM, had a good speedup, scaleup, and sizeup performance.

Luts [97] employed semi-parametric regression analysis of large-scale data. The research work analyzed *domestic flight data* from United States. The proposed method analyzed air traffic delays in real-time, which was published on a website [98].

Naimur Rahman et al. [99] utilized a back-propagation neural network (BPNN) for the electricity generation forecasting system. The experimental study involved US power consumption data, which is 20 years of historical data. The experimental results showed the proposed forecasting system could predict the power generation which is required for consumption accurately near to 99 % of the real utilization.

Saranya and Nagarajan [100] implemented a metaheuristic optimized artificial neural network for agricultural yield prediction using satellite image employing Hadoop computational framework. The work has analyzed remote sensing data. The experimental results indicated the Artificial Neural Network models to be efficient in finding complex correlations among the dependent variable, crop yield and the independent variables, spectral reflectance values.

2.4.2.2 Spark-based Journal papers

Chen et al. [101] proposed a patient treatment time prediction algorithm and a Hospital Queuing-Recommendation (HQR) system. The experimental work carried out with datasets covering three years of data gathered from an actual hospital. The results indicated that the proposed algorithm achieved high accuracy and performance.

Rodríguez-Fdez et al. [102] proposed a Scalable Fuzzy Rule Learning through Evolution for Regression (S-FRULER) which is a distributed version of FRULER (a genetic fuzzy system (GFS)) that learns for regression problems. The study included ten regression datasets from the KEEL project repository [103], viz. *Delta Ailerons*, *Delta Elevators*, *California Housing*, *MV Artificial Domain*, *House-16H*, *Elevators*, *Computer Activity*, *Pole Telecommunications*, *Pumadyn*, and *Ailerons* and a dataset from bioinformatics problem also used. Experimental results showed that S-FRULER scaled well with comparable precision.

Galicia et al. [104] proposed a method for predicting time series data in a Big Data paradigm, i.e., time series with high-frequency measurement. It is used to

predict a time horizon whereby the generation of prediction models was carried out with linear models like regression and with non-linear methods based on decision trees (DT). The experiments were carried out with real-time series data related to the *consumption of electric energy* in Spain. The proposed method presented a reasonable high accuracy.

Talavera-Llames et al. [105] proposed an approach for forecasting with huge volume of data employing the k-weighted nearest neighbor algorithm. The experiment was conducted with *energy demand time series data*. The proposed algorithm was compared with deep learning, and other machine learning techniques, viz., decision tree (DT), gradient-boosted tree (GBT), random forest (RF), and linear regression and found to be outperforming them.

Xu et al. [106] proposed a distributed computational framework for wind speed prediction employing extreme learning machines in the distributed computational framework of Apache Spark. Three datasets of real wind speed data and a dataset of analog wind speed big data are used to analyze the performance of the proposed method. The performance was evaluated by Mean Absolute Percentage Error (MAPE), the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE).

Ding et al. [107] proposed a system that will issue early warning for an electric vehicle charging and also planning an optimal path based on Spark. The Spark computing framework was utilized to parallelize the optimized Dijkstra algorithm. The parallel algorithm is applied to the path planning problem.

2.4.2.3 Hadoop MapReduce-based Conference papers

Yin et al. [108] proposed a scalable regression tree learning on the Hadoop MapReduce framework using OpenPlanet, an opensource counterpart of a proprietary regression tree algorithm PLANET. The experimental study involved two years of power consumption data from 24 buildings at the University of Southern California campus. The above data was extrapolated to larger synthetic datasets utilized in the study. The experimental outcomes indicated that the proposed method was efficient and scalable.

Tejasviram et al. [109] proposed a hybrid model combining the Auto-Associative Extreme Learning Machine (AAELM) with Multiple Linear Regression (MLR) for performing regression on scalable data. The experimental work was carried out with (i) *Gas sensor array under dynamic gas mixtures* and (ii) *Airlineflight* datasets collected from UCI repository [23]. The hybrid model was compared with MLR, and the results proved the proposed model outperformed the MLR in the context of MSE and MAPE.

Rehab and Boufares [110] proposed a parallel MLR. The experimental study analyzed a real-world dataset, i.e., *Airline on-time* dataset and a synthetic dataset. The outcomes indicated that the proposed approach has superior efficiency and scalability with respect to the dataset size and the count of nodes present in the cluster.

Chavda and Dhobi [111] proposed a method for the prediction of web users browsing behavior using SVM. The experimental study involved *http log* from the NASA server. The results showed the proposed approach to be efficient and scalable.

Xu et al. [112] proposed a prediction model for forecasting user behavior for smart home employing BPNN. The experimental study used the smart home dataset generated from intelligent residential districts. The proposed algorithm had an outstanding superiority in accuracy, efficiency, and scalability.

2.4.2.4 Spark-based Conference papers

Vluymans et al. [113] proposed a method for weighted k-NN regression along with fuzzy rough set theory. The experimental work carried out with two healthcare datasets [114], [115]. The experimental outcomes demonstrated that the proposed approach had improved performance of k-NN.

Oneto et al. [116] proposed a method for predicting train delays based on the ELM. The study was carried out with the real-world data collected from Rete Ferroviaria Italiana (RFI). The results showed that the proposed approach had improved delay prediction.

Kamaruddin and Ravi [117] have implemented a parallel version of GRNN for regression in the Big Data paradigm. They have implemented a parallel distributed

version of GRNN in Apache Spark, where the unsupervised learning part is implemented through K-Means|| and parallel Bisecting K-Means. They have analyzed the gas sensor dataset. The proposed method produces a very low mean squared error.

After reviewing the articles related to the regression in a distributed parallel computational environment, the following research gaps are discovered.

- *It is not explored to its full extent.*
- *There has been very less research work carried out in the field of regression task using the distributed parallel framework, which is the goldmine for budding researchers.*
- *During the review process, we found few articles had utilized Artificial Neural Network (ANN) in the form of BPNN. The other types of Neural Network (NN) like General Regression Neural Network (GRNN), Radial Basis Function Network (RBFN), Wavelet Neural Network (WNN), Quantile Regression Neural Network (QRNN) are not considered at all.*
- *Evolutionary computation and Fuzzy methods for regression present an excellent opportunity for researchers to explore it in the distributed parallel framework.*
- *Nonlinear regression has not been explored in the distributed parallel environment leading it to be a fertile area for research.*
- *Support vector regression has not been explored to its extent. It has not been explored in the Spark environment at all.*
- *Quantile regression, Ridge regression, Lasso regression, Cox regression, and Poisson regression are not explored at all.*
- *Kernel methods are not explored at all.*
- *Multivariate Adaptive Regression Splines (MARS), Treenet, extreme gradient boosting are not explored at all.*

2.4.3 Classification

2.4.3.1 Hadoop MapReduce-based Journal papers

He et al. [118] implemented an approach comprising a parallelized version of extreme support vector machine (ESVM), (PESVM) and an incremental learning algorithm for ESVM (IESVM), which can incorporate online learning to update the existing model. The proposed approach implemented the parallelization of IESVM, called PIESVM. The experimental works utilized the *Australian* dataset from the public UCI repository [23]. The large-scale datasets were synthesized by duplicating the data. The experimental results showed that the proposed algorithms PESVM and PIESVM were efficient and scalable.

Janaki Meena et al. [119] proposed a parallel version of the Ant Colony Optimization (ACO) algorithm for feature selection which is to be implemented for text categorization. The experimental study involved two datasets formed with documents from the *20Newsgroup* benchmark. The experimental results exhibited that the features selected by the proposed method has improved performance.

Caruana et al. [120] presented a parallel SVM algorithm for scalable spam filter training. The experimental work carried out with the *SpamBase* dataset from UCI repository [23]. The experimental results exhibited an improved accuracy of the proposed approach.

You et al. [121] proposed a parallel SVM model for predicting protein-protein interactions (PPI). The experimental work utilized the PPI dataset downloaded from the *human protein references database* (HPRD) [122]. Experimental results exhibited the proposed method had an excellent speedup performance, scalability with dataset size with comparable performance to its serial counterpart.

Singh et al. [123] proposed a framework for peer-to-peer botnet detection using RF-based DT-model. The experiment was carried out with the CAIDA UCSD dataset [124]. The experimental results showed the proposed method produced better accuracy than many other classifiers.

Chen et al. [125] proposed a classifier employing a parallel and scalable approach for network intrusion detection. The experimental study involved two intrusion detection datasets, viz., *KDD99* and *CMDC2012*. The experimental results demonstrated the proposed approach had a faster execution speed.

Kumar and Rath [126] implemented proximal support vector machine (mrPSVM) classifier to classify the microarray data with selected relevant features. The dataset for classification was collected from Kent Ridge Bio-medical Data Set Repository [127] and NCBI GEO [128]. The datasets were *LEukemia*, *MULTMYEL*, *ovarian cancer*, and *breast cancer*. The results showed the mrPSVM was efficient with less execution time in comparison with its sequential counterpart.

Han et al. [129] proposed a Distributed Extreme Learning Machine (DELM) for matrix operations over voluminous data along with Weighted Ensemble classifier based on DELM (WE-DELM), for efficiently classifying uncertain streaming data. The experimental study involved real-world dataset *KDDcup99* [130], and the popular static datasets *Iris* and *Spambase* [23]. The experimental results showed that the proposed algorithms had better efficiency, accuracy, and speedup.

Barkhordari and Niamanesh [131] proposed a scalable and distributable method to solve Patient Similarity (ScaDiPaSi) problems. The study utilized real *patient Electronic Health Records* (EHRs) from laboratories and hospitals. The experimental results showed better execution time and accuracy.

López et al. [132] proposed a cost-sensitive linguistic fuzzy rule-based classification systems for imbalanced large-scale data. The proposed method was analyzed with datasets from the UCI ML repository [23]. The outcomes indicated that the proposed approach had a competitive accuracy and execution time.

Xin et al. [133] proposed an Adaptive Distributed Extreme Learning Machine (A-ELM*). It overcomes the weakness of ELM* in learning large datasets. The experiment was carried out with the synthetic dataset. A-ELM* was compared with ELM* while increasing dimensionality, the number of records in the dataset, and found to outperform the latter.

Xia et al. [134] presented a Nearest Neighborhood approach with MapReduce framework for predicting the traffic flow employing correlation analysis. The authors have utilized real-world *trajectory* dataset [135], which comprises large-scale GPS trajectories generated by 12,000 taxis and developed a parallel k-nearest neighbor optimized classifier to predict the traffic flow in a real-time basis.

Bechini et al. [136] proposed a distributed association rule-based classification. The experimental study employed seven big datasets extracted from the UCI repos-

itory [23] and LIBSVM repository [53]. The results presented the superiority of the proposed method over DT and were comparable with RF in terms of accuracy.

Kumar et al. [137] proposed various statistical methods for feature selection and then implemented a MapReduce-based K-nearest neighbor classifier. The proposed approach classified microarray leukemia data obtained from the NCBI GEO repository [128], viz. dataset with accession number *GSE13159*, *GSE13204*, and *GSE15061*. The experimental results exhibited superiority in performance.

Chen et al. [138] proposed a MapReduce-based extreme learning machine (MR-ELM). The experimental study involved classification and regression datasets collected from UCI ML repositories [23] and FCUP [139]. The experimental results showed the scalability of the proposed algorithm when compared with original ELM and with other parallel versions of ELM.

Huang et al. [140] suggested a MapReduce-based parallel ensemble of an online sequential extreme learning machine (PEOS-ELM) algorithm. The experimental work involved the evaluation of the PEOS-ELM algorithm with the real and synthetic dataset. The real-world datasets used were MNIST [53], DNA [53] and KDDCup99 [130]. The result demonstrated that PEOS-ELM could learn large-scale data accurately and efficiently in comparison with EOS-ELM and POS-ELM algorithms. The proposed algorithm had excellent scalability and speed-up ratio.

Zhai et al. [141] employed a voting-based instance selection with random weight networks and MapReduce framework. The proposed algorithm is called MapReduce, and Voting-based Instance Selection (MRVIS). The experimental study was conducted with 8 data sets, including two synthetic datasets and 6 UCI datasets, viz. *banana*, *cloud*, *Gaussian*, *shuttle*, *artificial*, *cod_rn*, *poker* and *susy*. The proposed MRVIS was compared with Convolutional Neural Networks (CNN), Ensemble Neural Networks (ENN), and Recurrent Neural Networks (RNN). The outcomes indicated MRVIS to be effective and efficient.

Huang et al. [142] presented a MapReduce-based parallel method for batched online sequential extreme learning machine (BPOS-ELM). The proposed method of BPOS-ELM has analyzed real and synthetic data. There were three real-world datasets, viz. *MNIST*, *DNA* [53], and *KDDCup99* [130]. Two synthetic datasets were generated based on *Flower* [143] and *CIFAR-10* respectively. The experimental results exhibited a comparable accuracy with higher execution speed.

Li et al. [144] proposed a parallel feature selection method for text classification implemented on iterative MapReduce with Twister. The experimental work involved web page documents in Chinese from the Internet and abstracts of papers from CNKI, an electronic journal database of China. The experimental results presented the proposed approach to be efficient and scalable.

Zhai et al. [145] implemented the classification of large imbalanced datasets based on MapReduce and the ensemble of ELM classifiers. The experimental work involved one synthetic dataset and six datasets collected from UCI machine learning repository, viz. *Yeast*, *Abalone*, *Shuttle*, *Skin_segment*, *MiniBooNE*, and *Cod_rna*. The experimental results showed that the performance of the proposed algorithm statistically outperformed the three state-of-the-art approaches, viz. SMOTE-Vote, SMOTEBoost, and SMOTE-Bagging. The proposed algorithm also had a good speed-up and scale-up performance.

Jedrzejowicz et al. [146] proposed a parallel distributed framework for imbalanced data classification employing a MapReduce computation environment. The proposed framework involves the implementation of the splitting-based data balancing method (SplitBal) [147] and the dissimilarity-based imbalanced data classification (EDBC) [148]. The proposed framework was tested over the benchmark datasets obtained from KEEL [103] and UCI machine learning repository [23]. The experimental outcomes indicated that the proposed framework is scalable.

Elkano et al. [149] proposed a Fuzzy Rule-Based Classification Systems (FRBCS) for Big Data classification system with the MapReduce framework. The experiment was conducted with 20 binary datasets obtained from 8 different multiclass datasets available at the UCI repository [23]. The experimental outcomes indicated that the proposed distributed algorithm outperformed its local counterpart in accuracy and execution time.

Thakur and Deshpande [150] proposed a parallel approach for sentiment classification using the MapReduce framework. They implemented Kernel Optimized SVM classifier to analyze *train review* [151] and *movie review* [152] dataset. The performance of the classifier was compared with SentiWordNet, Naïve Bayes, Linear SVM, and Neural Network. The proposed approach had superior accuracy, sensitivity, and specificity.

2.4.3.2 Spark-based Journal papers

Maillo et al. [153] implemented k-NN Classification based on Apache Spark. The study included three datasets, viz. *PokerHand*, *Susy*, and *Higgs* collected from the UCI ML repository [23] and one dataset from the *ECBDL'14* competition. The proposed approach kNN-IS achieved the same accuracy as k-NN, and the execution time was reduced by almost ten times with respect to MR-kNN based on Hadoop.

Arias et al. [154] implemented Bayesian Network Classifiers for studying their adaptability to MapReduce and Apache Spark frameworks. The experimental work was carried out with *splice* [23], *epsilon* [53], and *ECBDL'14* dataset along with some synthetic dataset for the test of scalability of the proposed approach. The outcomes proved that the proposed approach was scalable and efficient.

Liu et al. [155] proposed a Parallel Back Propagation Neural Network (PBPNN) implemented in three distributed computing environments, i.e., Hadoop, HaLoop, and Spark. The experimental work involved *Iris* dataset. The outcomes demonstrated, the PBPNN algorithm outperformed standalone BPNN in terms of accuracy and stability.

Chen et al. [156] proposed a Parallel Random Forest (PRF) algorithm. The experimental study carried out by two groups of datasets with large scale and high dimensionality, viz. (i) URL Reputation, YouTube Video Games, Bag of Words and Gas sensor arrays datasets from the UCI ML repository and (ii) Patient, Outpatient, Medicine, and Cancer datasets from an actual medical project. The classification accuracy of PRF was evaluated by comparison with RF, dynamic random forest (DRF), and Apache Spark Mllib parallelized RF. Experimental results indicated the superiority of PRF in terms of accuracy, performance, and scalability.

Shi et al. [157] presented an integrated data pre-processing framework implemented with the Spark computational environment for fault diagnosis of power grid equipment. The classification was achieved with logistic regression and SVM. The experiment was carried out with data collected from the State Grid of China. The proposed method yielded higher classification accuracy than the traditional ones with scalability.

Lin et al. [158] proposed an ensemble random forest algorithm implemented with Spark. The experiments were conducted with the data from China Life Insur-

ance Company. The experimental results proved that the proposed methodology outperformed SVM and logistic regression in both performance and accuracy.

Nair et al. [159] presented a real-time remote health status prediction system. The system was deployed on streaming Big Data through user tweets with the Apache Spark environment. The experiments were carried out with a Heart disease dataset [23]. The health status was predicted as whether heart disease is present or absent by employing the decision tree model.

Gonzalez-Lopez et al. [160] proposed a distributed multi-label k-nearest neighbor (ML-KNN) implemented with Apache Spark. The proposed work involved a comparison of three strategies while analyzing 22 benchmark datasets. The outcomes prove that the tree-based index strategy outperforms the other approaches.

Venkataramana et al. [161] proposed a parallel multilevel feature selection procedure for cancer classification. The proposed method selects optimal and important features and evaluate them employing parallel Random Forest on Spark. The work analyzed three biological datasets. The proposed method was compared with extant parallel feature selection methods and sequential methods. The proposed approach is found to be better in performance in terms of accuracy and execution time.

2.4.3.3 Hadoop MapReduce-based Conference papers

Magnusson and Kvernvik [162] proposed a subscriber classification within telecom networks. The experimental study involved real traffic data from a telecom operator and synthetic dataset from a randomly generated graph, and another one generated based on the ErdõsRényi algorithm [163]. The results showed the performance of the proposed method.

Khuc et al. [164] proposed large-scale distributed systems for real-time Twitter sentiment analysis. The experimental study involved *Twitter* dataset. The experimental result showed that the proposed classifier obtained superior accuracy. The proposed approach was scalable.

Wang et al. [165] proposed a hybrid method combining DT and SVM- for stock futures prediction. The work carried out with real-world data from stock exchange.

The research outcomes exhibited the superior performance of the hybrid architecture in comparison with BootstrapSVM, Bootstrap-DT, and BPNN.

Han et al. [166] proposed SMRF algorithm, an improved scalable RF algorithm. The study analyzed ten UCI publicly available datasets. The experimental results demonstrated that the proposed algorithm had excellent scalability and comparable accuracy with traditional RF algorithm.

Chen et al. [167] proposed an extreme learning machine (ELM) ensemble classifier based on the MapReduce framework (ELM-MapReduce). The experimental study amalyzed the Remote Sensing (RS) images captured by satellite. The experimental results showed that the ELM-MapReduce presented scalability, higher accuracy, and efficiency with large datasets.

Liu et al. [168] proposed an approach to machine learning for large-scale dataset with meta-learning. The experiments utilized eleven real-world datasets, viz., *yeast*, *wineRed*, *wineWhite*, *pendigits*, *spambase*, *musk*, *telescope*, *kdd*, *isolet*, *org*, and *census* [23]; and two synthetic datasets, viz., *S1* and *S2*, generated by applying the RDG1 data generator in WEKA data mining tool [169]. The outcomes indicated that the proposed approach could reduce computational complexity with less error rate.

Lubell-Doughtie and Sondag [170] proposed an approach for distributed classification using the Alternating Direction Method of Multipliers (ADMM) algorithm. The experimental work was carried out with historical data consisting of approximately 25 million records of website visit attributes with 440 features per record. The performance of the proposed approach was measured with a change in loss function per iteration. It was found that the loss decreased, or in other words, the accuracy consistently improved as more iterations passed.

Al-Madi and Ludwig [171] proposed scalable genetic programming for data classification called MRGP, implemented with MapReduce framework. The experimental study involved six datasets, viz. *Ionosphere*, *Vertebral Column* (Vertebral-2C and Vertebral-3C), *Blood Transfusion Service Center*, *Balance Scale*, *Cardiotocography*. The research results showed that the proposed approach had higher accuracy with speedup and scalability properties.

Kiran et al. [172] proposed an approach for Parallel SVM algorithm. The experimental work involved a comparison of sequential SVM and parallel SVM on the

MapReduce framework. The observed results showed that the proposed approach had superior efficiency in terms of execution time and with scalability.

Wang et al. [173] have presented an instance-weighted variant SVM with a parallel meta-learning algorithm analyzing eleven datasets from UCI ML repository [23] and a real-world dataset *Maritime*. The outcomes proved that the proposed method could improve the prediction performance of the classifier.

Park and Ha [174] proposed an approach for data classification over imbalanced data for traffic accident prediction. The experimental study utilized traffic data collected from the Korea Highway Corporation. The results showed that the proposed approach was efficient with reasonable accuracy.

López et al. [175] proposed a method for linguistic fuzzy rule-based classification systems for large-scale data achieved with the MapReduce framework, called Chi-FRBCS-BigData. The algorithm had been developed in two versions: Chi-FRBCSBigData-Max and Chi-FRBCS-BigDataAve. The study involved six datasets from the UCI machine learning repository. The two versions of the proposed algorithm produced different classification results. The experimental results presented that Chi-FRBCS-BigData-Ave generated more accuracy with higher execution time, whereas Chi-FRBCSBigDataMax produced a faster but less accurate result.

Kolias et al. [176] proposed a method RuleMR for generating classification rules out of large-scale data. The experimental work involved datasets from UCI repository [23] for measuring the accuracy, viz. *breast*, *car eval*, *connect-4*, *weather*, *mushroom*, *nursery*, *vote*, and *tic*. Four synthetic datasets were used for measuring execution time. The outcomes proved that the proposed method had scalability with respect to the size of the dataset and a comparable accuracy with state-of-the-art algorithms.

Kakade et al. [177] proposed a spam filtering technique implemented with SVM along with Sequential Minimal Optimization (SMO). The experimental study was carried out with *SpamBase* dataset available on UCI repository [23]. The research outcomes showed that the proposed approach with SMO was efficient and had a superior speed up to Linear as well as the Gaussian kernel.

Anchalia and Roy [178] proposed a MapReduce-based k-NN algorithm. The experimental work involved a comparison of MapReduce k-NN with sequential

k-NN. The work included a synthetic dataset for the study. The experimental outcomes proved that the MapReduce k-NN outperformed the sequential k-NN with voluminous data.

Ke et al. [179] proposed a method for distributed SVM for binary and multiclass classification. The proposed method involves parallel decomposition solver on two datasets, *MNIST*, and *Cover-Type*. The experimental results projected substantial growth in speed-up.

Kolias et al. [180] implemented a classification rule induction algorithm. The algorithm produces an ordered list of classification rules from massive categorical data. The datasets from UCI ML Repository [23] used for comparing the accuracy and quality of the proposed approach and four synthesized datasets used for verification of the algorithm for scaled-up datasets. The algorithm does not work on the numerical dataset.

Maillo et al. [181] employed k-NN classification. The experimental study was carried out over the *PokerHand* dataset, collected from the UCI repository [23]. The experimental results exhibited the reduction of computational time achieved by the proposed algorithm in comparison to its sequential version.

Chatzigeorgakidis et al. [182] proposed a new k-Nearest Neighbour (k-NN)-based algorithm. The proposed Flink zkNN (F-zkNN) algorithm was extended over a Hadoop-based implementation of the k-NN (H-zkNN). The experimental work involved water usage time-series data. The algorithm was evaluated with respect to forecasting and prediction precision.

Wang et al. [183] proposed an ordinal RF algorithm based on the variable consistency dominance-based rough set approach (VC_DRSA). The experimental work utilized two synthetic datasets. The experimental results confirmed that the proposed algorithm was effective and efficient.

Cui and Zhao [184] proposed a method for gender classification. Three classification algorithms were involved, viz. (i) SVM, (ii) k-NN, and (iii) Adaboost to implement gender parallelize machine learning (GPML). The experimental work was carried out with images from the *CAS-PEAL* dataset. The proposed algorithm was compared with parallelized Adaboost. The experimental results showed that GPML had higher recognition rates.

Yuan et al. [185] proposed a GA optimized DT algorithm (MRGAOT) implemented in MapReduce environment. The parallelized MR-GAOT was compared with the traditional decision tree algorithm, and the results showed higher classification accuracy and shorter execution time.

Wakayama et al. [186] proposed a distributed RF. The experiments were carried out with *Letter Recognition* dataset from UCI ML Repository. The results indicated that the proposed approach had an excellent classification performance along with lower computational costs compared to the naïve implementation of RFs.

Triguero et al. [187] proposed an evolutionary under-sampling method for imbalanced Big Data classification. The experimental study involved the *KDDCup99* dataset from the UCI ML repository. The research outcomes proved the scalability of the proposed method.

Zdravevski et al. [188] proposed an approach for feature ranking based on information gain for large data classification problems. The ranking of features led to feature selection for which the *FedCSIS AAIA'14 data mining competition* dataset [189] was used for the study. The proposed approach had excellent scalability.

Kumar et al. [190] proposed a method for feature selection and classification of microarray data. The experimental study utilized large datasets from NCBI GEO [128], viz. *Leukemia*, *Ovarian Cancer*, *Breast Cancer*, *MULTMYEL*, and *LEukemia*. The research outcomes proved that the proposed approach was efficient and scalable.

Bhukya and Gyani [191] proposed a fuzzy associative classification algorithm. The experimental study involved *Record linkage comparison pattern* dataset from UCI machine learning repository [23]. The study showed the proposed approach had superior accuracy and efficiency with a comparison to its sequential counterpart.

Arias et al. [192] proposed a k-dependence Bayesian Classifier (KDB). The proposed approach was evaluated with three datasets, viz. *Splice*, *ECBLD14*, and *Epsilon*. The experimental work showed the proposed method to be efficient.

Sukanya et al. [193] implemented SVMs in linear and parallel distributed frameworks. The experimental study utilized three datasets, namely *Spam*, *Wine*, and *Heart*. The experimental results showed MapReduce-based SVM performed superiorly regarding execution time and accuracy with scalability on dataset size.

Yang et al. [194] proposed a method for a parallelized Rocchio relevance feedback algorithm implemented with the MapReduce framework called, MR-Rocchio algorithm. The experimental study involved the earthquake information source text for Beijing. The experimental results showed that the performance of the MR-Rocchio algorithm was significantly improved in comparison with the traditional Rocchio algorithm, better than the KNN algorithm, and only slightly inferior to the SVM algorithm.

2.4.3.4 Spark-based Conference papers

Peng et al. [195] implemented and evaluated parallel LogR models in the Big Data paradigm. The experimental study involved a synthetic dataset, *2d*, and four real-world datasets, viz., *20NewsGroup* dataset, *Gisette* dataset, *ECUESpam* dataset, and *URL-Reputation* dataset. The experimental work involved an analysis of three optimization approaches implemented with two computing platforms, viz. Hadoop and Spark framework to train the LogR model on high-volume, high-dimensional datasets for classification. The results showed that Spark outperformed Hadoop for LogR model implementation.

Tao et al. [196] proposed a budgeted mini-batch parallel gradient descent algorithm (BMBPGD) for large-scale kernel SVM training. The experimental work involved three datasets, viz. *a9a*, *w8a*, *covtype* from UCI/Adult, JP98a, and UCI/Covertype respectively. The work included a comparative analysis of the proposed approach with SVMWithSGD and LibSVM. The experimental outcomes proved that the proposed approach had a superior accuracy than the SGD-based SVM.

Lin et al. [197] implemented LogR and linear SVMs in largescale data. The study involved datasets from the LIBSVM dataset page along with *Yahoo-Japan* and *Yahoo-Korea* dataset. The implemented method was efficient and scalable.

Wang et al. [198] proposed Weighted Label Propagation Algorithm with Probability Threshold (P-WLPA) algorithm. The experimental study utilized *Iris* dataset from UCI repository [23] and a synthetic dataset. The work involved a comparison of Serial and parallel P-WLPA. The results showed the feasibility and efficiency of the proposed parallel P-WLPA algorithm.

Roy [199] proposed a method for online feature selection by employing an ensemble of Kohonen neurons which is trained through high-dimensional streaming data. The experimental work was carried out with five datasets, viz. (i) *Leukemia*, (ii) *Central Nervous System (CNS)*, (iii) *Colon Tumor*, (iv) *Lymphoma* and (v) *small round blue-cell tumors (SRBCT)*. The work involved a comparison of the proposed algorithm with the state-of-the-art algorithm for average test error, standard deviation, and an average number of selected features. The Kohonen ensemble had superior performance in almost all cases.

Ramírez-Gallego et al. [200] proposed a distributed implementation of the entropy minimization discretizer. The datasets employed in the experiment are (i) *ECBDL14*, (ii) *epsilon*. The experimental results demonstrated the improvement in both classification accuracy and execution time for the proposed algorithm.

Bhagat and Patil [201] proposed an Enhanced SMOTE algorithm for the classification of imbalanced data using RF. The experimental study involved datasets, viz., *Landsat*, *Lymphography*, *Zoo*, *Segment*, *Iris*, *Car*, *Vehicle*, and *Waveform* [23]. The proposed approach was implemented on the Hadoop MapReduce framework and the Apache Spark platform. The results showed that the proposed method carried out with Spark outperformed other methods.

Chandorkar et al. [202] proposed an approach for SVM employing fixed-size least squares for high-volume data classification. The model was evaluated with datasets available in UCI repository [23]. The experimental outcomes indicated that the proposed approach had a substantial speed up over the existing implementations.

Venturini et al. [203] proposed a distributed Bagged Associative Classifier (BAC), employing an ensemble techniques with voting to provide a unique classification outcome. The experiment was carried out with three datasets, viz. *yeast*, *nursery*, and *census*, from the UCI repository [23] and a synthetic dataset, generated from IBM data generator [65]. The experimental results showed that bagging achieved an accuracy above or at par with the sampling-only approach.

Semberecki and Maciejewski [204] implemented classification of text documents. The classifier was verified with documents belonging to three categories, viz. Art, History, and Law. ML algorithms, viz., Naive Bayes Classifier, DT, and RFs were utilized for building the classification model. The outcomes indicated that the Naive Bayes algorithm had the best accuracy.

Nodarakis et al. [205] proposed a method for sentiment analysis on Twitter. The proposed method analyzed two Twitter datasets collected through the Twitter Search API. The research outcomes indicated that the proposed approach was efficient, robust, and scalable.

Ray et al. [206] proposed a method that employed a mutual information feature selection method based on spark framework (sfMIFS) to determine the relevant features. After the feature selection process, the machine learning techniques, viz., Logistic Regression (LogR) and Naive Bayes (NB) using Spark were implemented to classify the microarray datasets. The dataset utilized for the study was *LEukemia* dataset collected from NCBI GEO [128]. The experimental results showed that sf-MIFS provided superior accuracy with NB as compared to LogR.

Tayde and Patil [207] proposed an approach for genome data classification employing n-gram for genome sequence encoding. The features were gathered implementing n-gram. The SVM classifier was used to classify the genome data of the cat and rat species. The Spark-based approach was found to be efficient in terms of execution time.

Kamaruddin and Ravi [208] proposed a hybrid architecture involving AutoAssociative Neural Network (AANN) and PSO called PSOAANN for credit card fraud detection using one-class classification. The experimental study utilized *ccFraud* dataset [209]. The research outcomes proved that the proposed approach had superior accuracy.

Talavera-Llames et al. [210] proposed nearest neighbors based algorithm for long time series data forecasting. The experimental work involved datasets related to electrical energy consumption from a Spanish public university. The experimental work showed satisfactory results regarding both mean relative error (MRE) and execution time.

Lincy and Nagarajan [211] implemented a parallel distributed SVM along with data augmentation for Semi-supervised Classification. The experimental work analyzed real-world datasets. The proposed methodology was compared with Spark-based Logistic regression with Stochastic gradient descent, Random Forest, LinearSVM, and Decision Tree algorithms. The outcomes indicated that the proposed methodology has better performance with respect to execution time and accuracy.

We identified the following research insights during the review of articles related to classification.

- *It has been observed that SVM, ELM, K-NN, DT, and RF are the most implemented in a parallel and distributed manner. Maintaining the same distribution of the samples in all data partitions, we can implement SVM, K-NN, and similar algorithms to run in a parallel manner. The algorithms like DT, RF are inherently parallel as they partition the sample space independently and so can be implemented in a parallel manner.*
- *Logistic regression and Fuzzy rule-based classifiers are the least explored in distributed parallel architecture leading to an open research space for the researchers.*
- *Only a few papers were found using the Evolutionary computational method for classification in Hadoop and Spark environment leading to the fertile ground to explore.*
- *The most explored ML technique is SVM with its variants for classification in the distributed parallel environment.*
- *Any architecture of NN is conspicuous by its absence.*
- *The algorithms like Very Fast Decision Tree (VFDT), Classification and Regression Trees (CART), and Chi-square Automatic Interaction Detection (CHAID) are not explored at all.*
- *Extreme gradient boosting is not explored at all.*
- *Kernel methods and class association rule mining are not explored at all.*

2.4.4 Clustering

2.4.4.1 Hadoop MapReduce-based Journal papers

Sun et al. [212] proposed a clustering method using a parallel information bottleneck (IB) theory clustering method along with a centroid-based clustering method

to determine the clusters. The experimental study was carried out with *16S rRNA* dataset [213]. Interpolation Multi-dimensional scaling (MDS) was used for feature dimension reduction for visualization of clustering results in 2D and 3D. The outcomes indicated scalability of the proposed method.

Xu et al. [214] proposed a clustering approach K-means++. This proposed approach had a significant reduction in communication and I/O costs. The study involved one real dataset *Oxford Buildings* dataset and a synthetic dataset. The experimental results indicated that the proposed MapReduce K-means++ method was much more efficient and scalable.

Cui et al. [215] proposed the K-Means algorithm in MapReduce environment to eliminate the iteration dependence and obtain high performance. The experimental study involved a synthetic dataset, Gauss Distribution Set, and two real datasets, *Bag of Words* and *House* collected from UCI ML repository. The experimental results showed that the proposed algorithm was efficient and superior to parallel K-Means, K-Means||, and stand-alone K-Means++ algorithms.

Ludwig [216] proposed a parallel approach for the fuzzy c-means clustering algorithm implemented with the MapReduce framework (MR-FCM). The experimental study involved *Covertype* dataset [23]. The proposed algorithm was compared with other clustering approaches. The experimental evaluation indicated a comparable purity along with excellent scalability.

Yang et al. [217] proposed a semi-supervised multi-ant colonies consensus clustering algorithm. The experimental study involved datasets *Iris*, *Wine*, *Balance-scale*, *Sonar*, *Covertype*, *Shuttle*, *MinibooNE* from the UCI machine learning repository [23] along with *USCensus1990*, and the image database [218]. The research outcomes demonstrated the proposed method to be effective.

Bi et al. [219] proposed Semantic Driven Subtractive Clustering Method (SDSCM) based on the Subtractive Clustering Method (SCM) and fuzzy cmeans (FCM), to alleviate the risk of customer churn. Business Support System and Operations Support System data of *China Telecom* used for customer churn management.

Liu et al. [220] proposed a method for similarity join using Similarity Join Tree (SJT) and Extended Fiduccia–Mattheyses (EFM) algorithm. The work involved two real datasets, viz. dataset containing time-series recordings from UCI ML repository [23] and articles extracted from Wikipedia in the English domain

[221]. The research outcomes depicted the proposed method to be more efficient and scalable.

Zhang et al. [222] proposed a Distributed Density Peaks clustering algorithm with Locality Sensitive Hashing (LSH-DDP). The experimental study involved nine datasets of different dimensions [23], [223]. Experimental results on local cluster and cloud showed that LSH-DDP achieved a superior speedup over other distributed density peaks clustering algorithms.

Alewiwi et al. [224] proposed a document similarity approach with a filtering method of cosine similarity measure. The experimental study utilized the *Enron* dataset [223] and *Reuters* dataset [225]. The observed results demonstrated the proposed method had superior efficiency.

Fang et al. [226] proposed algorithms for nearest-neighbor joins amenable for high-volume trajectory data rendering scalability. The experimental study analyzed two synthetic datasets *DS1* and *DS2*, generated from GSTD [227]; and another synthetic dataset *DS3* was constructed from Brinkhoff [228]. One real dataset *Beijing taxi* dataset [229] was also analyzed in the study. The experimental outcomes proved that the proposed algorithm was efficient and scalable.

Wu et al. [230] proposed a SIMPLifying and Ensembling (SIMPLE) framework for parallel community detection. The work involved six network datasets, out of which five were of friendship networks derived from different social networking sites, viz. *Oklahoma*, *UNC*, *Twitter*, *Gowalla* and *LiveJournal*; and *Skitter* is an Internet topology graph. The experimental results showed that SIMPLE could identify high-quality community structures on various networks demonstrated by the Q-function.

Shahrivari and Jalili [231] proposed a single-pass and linear time solution for the K-Means algorithm called MRK-Means. The experimental study utilized a set of synthesized datasets and five real-world datasets, viz. *USCensus*, *KDD04*, *Poker*, *Skin*, and *Birch* [23]. The experimental results showed the proposed approach, MRK-Means, had faster execution times, and higher quality of clustering with linear scalability.

Banharnsakun [232] proposed a MapReduce-based artificial bee colony called MR-ABC for data clustering. The author has used four synthesized datasets generated from *Iris*, *CMC*, *Wine*, and *Vowel* [23] by duplicating the records. The ex-

perimental outcomes presented that the proposed algorithm had a superior value in comparison with PKMeans and K-PSO algorithms with respect to F-measure.

Capó et al. [233] proposed an efficient approximation to the K-Means algorithm for voluminous data. The experimental results exhibited that the proposed method outperformed other methods like the K-Means++ and the minibatch K-Means.

Tidke et al. [234] proposed topic sensitive user clustering based on sentiment score and similarity measures employing the MapReduce computational framework of Hadoop. They have collected tweets from Tweeter API to detect the cluster of users having similar sentiments. The experimental work compared the performance of the method in terms of execution time against the serial approach.

2.4.4.2 Spark-based Journal papers

Lu et al. [235] proposed an improved K-means clustering algorithm with a Tabu Search strategy to enable it to handle Big Data applications. The experimental work involved *Iris*, *Wine*, *Yeast*, and *Seeds* datasets [23]. The research outcomes disclose that the proposed method had a superior solution to the K-Means algorithm of Spark MLLib.

Xia et al. [236] proposed a parallel adaptive canopy-K-Means clustering algorithm for a large dataset employed in a parallel computational environment of Apache Spark. The performance of the proposed method is analyzed with the Stanford Network Analysis Project (SNAP) dataset and the author-built Dimension Networks dataset. The research outcomes indicated that the proposed method is effective.

Yuan [237] proposed a parallel K-Means clustering, which is optimized by employing particle swarm optimization with the Spark computational environment. The proposed approach is employed for mining the anomaly from sensor networks. The work included analysis of the dataset *KDDCUP99*. The results of analysis demonstrated that the proposed method is having high accuracy and better execution time when compared with other parallel clustering approaches.

Ianni et al. [238] proposed a complex hierarchical clustering algorithm CLUBS+ in a parallel computational environment of MapReduce and Spark. The

work has analyzed synthetic datasets from an open-source synthetic data generator[1]. The experimental results show that CLUBS+ with Spark generates high-quality clusters of data clustered around their centroids. The accuracy and scalability of the CLUBS+ with Spark is improved over other parallel clustering algorithms.

2.4.4.3 Hadoop MapReduce-based Conference papers

Zhao et al. [239] proposed a parallel K-Means clustering. The experimental study utilized a synthetic dataset. The experimental results demonstrated the scalability and efficiency of the proposed algorithm.

Ene et al. [240] proposed clustering algorithms, viz. k-center and K-Median implemented with the MapReduce framework. The study involved a random set of points in R^3 as the dataset. The research outcomes demonstrated that the proposed algorithms had comparable accuracy with better execution time in comparison with other tested algorithms.

Zongzhen et al. [241] presented a fuzzy clustering approach for document categorization. In the experiment, five different classes, viz. *Diabetes*, *Happiness*, *Yoga*, *Ebook*, and *Security* selected and one hundred articles of each class chosen for training and to test the model. The experimental results were containing the F1 measure that exhibited the performance of clustering.

Liao et al. [242] proposed an improved parallel K-Means clustering algorithm. The proposed algorithm presented a superior performance in both processing speed and accuracy than the traditional parallel K-Means algorithm.

Esteves et al. [243] proposed a Competitive K-Means algorithm. The experiments were carried out with four datasets, viz. (i) *Hypercube*, (ii) *Google*, (iii) *Electrical*, and (iv) *KDD99*. The work involved a comparative analysis of the proposed algorithm with serial K-Means++ and streaming K-Means. The results showed the proposed algorithm had increased accuracy and decreased variance with scalability related to the dimension of the dataset.

Kumar et al. [244] proposed an approach for the parallel K-Means algorithm. The experimental result showed that the proposed method had a superior efficiency with scalability.

[1] https://github.com/gmmazzeo/clugen

Lin et al. [245] proposed a K-Means clustering algorithm with optimized initial-centers based on data dimensional density. The experimental results demonstrated the stability of the algorithm with a cost of execution time.

Zhang and Wang [246] proposed an enhanced agglomerative fuzzy K-Means clustering algorithm. Experimental works were carried out on a synthetic data set, the *WINE* dataset. The observed outcomes presented that the proposed algorithm had superior accuracy and scalability.

Bousbaci and Kamel [247] proposed a hybrid architecture with K-Means and PSO. The experimental work was carried out with two synthetic numerical multi-dimensional datasets [248]. The experimental results presented that the proposed approach had improved execution time and cluster quality.

Garg and Trivedi [249] proposed fuzzy k-mean clustering. The study involved datasets viz. *Iris* dataset, *Synthetic control* dataset, and *KDDCUP99* dataset. The experimental results depicted the excellent execution time.

Anchalia [250] proposed an improved method to implement the K-Means Clustering Technique. The experiment involved generated synthetic data. The research outcomes showed the proposed method outperformed the regular implementation.

Zhu et al. [251] proposed an improved algorithm for the optimal search of medoids to cluster Big Data using K-Medoids clustering. The results showed that the proposed algorithm had high efficiency and effectiveness in comparison to its serial counterpart.

Choi and So [252] proposed a method for the K-Means algorithm with an FPGA-accelerated computer cluster. The experimental study used dataset collected from UCI machine learning repository [23]. The outcomes showed that the proposed FPGA K-Means implementation had superior performance compared to the baseline software implementation.

Garcia and Naldi [253] proposed a method for parallel K-Means clustering. The experimental work involved five synthetic datasets generated by the MixSim R Package [254]. The results presented that the proposed method outperformed other implementations of K-Means in terms of execution time.

Daoudi et al. [255] proposed a method for the parallel differential evolution clustering algorithm. The experimental study involved 18 publicly available gene

expression datasets. The experimental outcomes indicated that the proposed approach was efficient and produced comparable results with existing algorithms.

Al-Madi et al. [256] proposed an algorithm for clustering large-scale data using parallel glowworm swarm optimization implemented with MapReduce, called MRCGSO. The experimental study utilized four real-world datasets, viz. *magic*, *poker hand*, *cover type* collected from UCI repository [23] and *Electricity* collected from MOA [257]. Apart from real-world datasets, four synthetic datasets were generated using the data generator [258]. The experimental results demonstrated the proposed algorithm had good accuracy with scalability and with a linear speed up while maintaining cluster quality.

Jiang and Zhang [259] proposed a parallel K-Medoids clustering algorithm implemented with the Hadoop MR framework, called HK-Medoids. The experimental study involved *Iris* dataset from UCI repository [23]. The experimental results showed that the proposed HK-Medoids algorithm had scalability with cluster node count. The results showed the proposed algorithm had a linear speedup for large-scale data with good clustering results.

Yuan et al. [260] proposed a distributed link prediction algorithm based on clustering on social networks. The research work involved five classical datasets on the social network: *USAir*, *PB*, *Yeast*, *Power*, and *Router*. The experimental results proved that the proposed parallel algorithm had a superior performance in terms of execution time in comparison with its serial counterpart.

Shettar and Purohit [261] proposed an enhanced K-Means algorithm. The experimental work carried out with two types of datasets. One was randomly generated numbers, and another dataset was collected from the DEBS-2014 grand challenge [262]. The experimental outcomes presented that the proposed method had better accuracy than traditional counterparts.

Yu and Ding [263] proposed an improved Fuzzy C-Means (FCM) algorithm with the help of the canopy algorithm called canopy-FCM. The research work was carried out with *Church* dataset. The experimental results presented that the proposed canopy-FCM algorithm in MapReduce had lesser execution time in comparison with the FCM algorithm in MapReduce.

Boukhdhir et al. [264] proposed an advanced K-Means algorithm with the removal of outliers and selection of initial centroids, called IM-KMeans algorithm.

The experimental work involved a real-world stock exchange data. The experimental results presented the superior performance of IM-KMeans regarding execution time in comparison with traditional K-Means, Parallel K-Means (PK-Means), and Fast K-Means.

Lachiheb et al. [265] proposed an improved K-Means algorithm, called SMRK-Means. The algorithm was applied to identify stock investments with risks based on variation in stock data. The experimental work carried out with (i) a real-world stock exchange data and (ii) a synthetic dataset with random values. The research outcomes proved that the proposed method reduces the execution time while keeping 80% of the clustering accuracy. The proposed algorithm was compared with traditional K-Means, PK-Means, and FastK-Means.

Mao et al. [266] proposed an optimal distributed K-Means clustering algorithm. The proposed algorithm was improvised with, (i) partially random center selection algorithm (PRCSA), to select the initial points (ii) implementing Haloop, to support the iterative calculation model of machine learning while saving the intermediate results generated during each iteration to the cache. The experiments utilized data from the *household electricity consumption* dataset. The experimental results showed the proposed algorithm with haloop performed better in terms of execution time than its Hadoop counterpart.

Saranya and Sumalatha [267] proposed a dynamic neighborhood selection (DNS) clustering algorithm based on MapReduce framework (DNS-MR). The experimental work involved *person health care* dataset. The results demonstrated, the proposed DNS-MR algorithm was superior in efficiency and had less execution time in comparison to DBCURE-MR.

Ketu et al. [268] proposed an approach for large-scale text data clustering using a distributed K-Means algorithm combined with a corpus selection technique for a significant reduction of overall computational time. The experimental study utilized four large text datasets, viz. *Wikilanguage* [269], *Wikilinks* [269], *Enron* [223], and *Wikipedia* dataset [270]. The performance of the proposed algorithm was compared with traditional K-Means and parallel K-Means. The experimental results showed that the corpus selection technique was significantly effective in the reduction of overall processing time.

Karimov and Ozbayoglu [271] developed a hybrid model, H(EC)^{2}S, Hybrid Evolutionary Clustering with Empty Clustering Solution, for clustering of Big Data. The experimental study involved two datasets, i.e., *public* dataset and *ATM logs* dataset. The outcomes proved that the proposed method outperformed other models with a significant clustering quality gain.

Karimov et al. [272] proposed a method for the K-Means clustering algorithm with centroid calculation heuristics. The work involved a comparative study of serial and parallel implementations of the proposed algorithm and analyzed two real-world datasets. The research outcomes proved that the proposed method outperformed other compared methods.

Garg et al. [273] proposed a modified fuzzy K-Means clustering implemented with MapReduce. The experimental study involved datasets from UCI ML repository [23]. The experimental result showed that the proposed approach was efficient in terms of execution time as compared to its counterpart.

Ling and Yunfeng [274] proposed a distributed K-Means clustering algorithm based on set pair analysis. The experimental study used the extended *Iris* dataset and *Wine* dataset available on UCI machine learning repository [23]. The results showed that the proposed algorithm was more efficient than other compared algorithms.

Tao et al. [275] proposed the parallel K-Modes algorithm. K-Modes is a conventional categorical clustering algorithm. The experimental study used *US Census Data (1990)* dataset. The experimental results showed that parallel K-Modes achieved a good speedup ratio with large-scale categorical data.

Phan et al. [276] proposed an approach for range-based clustering supporting similarity searches in Big Data. The experimental study involved *Gutenberg* datasets. The research outcomes indicated that the proposed approach was superior to avant-garde algorithms.

Chunne et al. [277] proposed a method for real-time clustering of tweets using adaptive PSO technique. The parallel PSO was compared with the K-Means algorithm. The experimental results showed that the F-Measure was increasing with an increase in the number of particles.

Chen et al. [278] proposed two distributed clustering algorithms, i.e., Distributed Density-based Clustering (DDC) and Distributed Grid-based Clustering (DGC) algorithm with a reduction in communication and merging overheads. The

experimental study was carried out with a synthetic dataset generated from a data generator [279]. The experimental results showed that the proposed algorithms DDC and DGC were able to reduce the execution time and achieve scalability.

Wu et al. [280] proposed an approach for improved K-Means algorithm. The experimental study involved *Reuters news* set. The experimental outcomes proved that the proposed approach had superior efficiency in comparison with its serial counterpart and scalability.

Gao et al. [281] implemented K-Means clustering for fixed traffic bottleneck detection. The proposed method analyzed the data collected from the Jilin urban regional road net. The experimental outcomes indicated that the proposed method had scalability and was superior with respect to execution time.

de Oliveira and Nald [282] proposed a scalable evolutionary K-Means clustering method called, Scalable Fast Evolutionary Algorithm for Clustering (SF-EAC). The experimental work was carried out with three synthetic datasets generated by the MixSim R Package [254]. In addition to that, one real-world dataset from *Medline database* (PubMed) was also analyzed. The outcomes indicated that the proposed SF-EAC approach obtained comparable quality with MRMK-Means with lesser execution time.

Moertini and Venica [283] employed an enhanced parallel K-Means implemented with the MapReduce framework for clustering large datasets. The experimental study was carried out with *household energy consumption* dataset obtained from UCI ML repository [23]. The results showed the proposed method had linear scalability.

Akthar et al. [284] proposed a method for the K-Means clustering algorithm. The *20 Newsgroups* dataset [23] was considered for the evaluation of the proposed algorithm. The experimental results showed that the proposed approach had a superior performance in terms of precision, recall, F-measure with less execution time in comparison with the simple K-Means algorithm.

Zhong and Liu [285] suggested the application of the K-Means clustering algorithm for clustering spatial data. The user data of *Sina Weibo* was used for the study. The experimental result showed the efficiency of the proposed approach.

Budhraja et al. [286] proposed a fuzzy clustering-based classification for large-scale TCP dump dataset implemented with Hadoop Framework. The study included

KDD'99 dataset [130]. The outcomes indicated that the proposed fuzzy c-means algorithm had superior accuracy to the K-Means algorithm.

Alshammari et al. [287] proposed a genetic algorithm based parallel K-Means algorithm employing the MapReduce computational framework in the Hadoop environment. They have analyzed five synthetic datasets [288] (ADS1-ADS5) for the experiment. The outcomes indicated that the proposed method had superior execution time.

2.4.4.4 Spark-based Conference papers

Sarazin et al. [289] proposed Self Organizing Map (SOM) clustering. The experimental work involved datasets from UCI repository [23]. The experimental results showed the proposed approach had scalability and efficiency in terms of execution time in comparison with its serial counterpart.

Tsapanos et al. [290] proposed an approach for distributed implementation of the Nearest Neighbour and □-ball variations of Kernel K-Means, which was implemented with Apache Spark. The experimental work utilized the *MNIST handwritten digits* database and *BF0502* dataset, containing descriptors of the faces. The proposed approach provided improved results over baseline Kernel K-Means and approximated Kernel K-Means in a time-efficient manner.

Govindarajan et al. [291] proposed a method for Parallel Particle Swarm Optimization (PPSO) clustering. The experimental results showed the proposed algorithm outperformed sequential clustering algorithms and existing parallel algorithms regarding execution time, speedup, inter- and intra-cluster distance measures.

Ketu and Agarwal [292] proposed an approach for distributed K-Means clustering for large-scale data analytics. The study included four benchmark datasets, viz. *Wikilanguage* [269], *Wikilinks* [269], *Enron* [223], and *Wikipedia* dataset [270]. The experimental outcomes indicated that the proposed distributed K-Means is 10x faster than the Hadoop MapReduce implementation.

Zhu et al. [293] proposed an approach for distributed SAR image change detection. The proposed method employed kernel fuzzy c-means clustering algorithm with Spark called S-KFCM was used to group the changed area and unchanged area

in the difference map. The experimental work involved a comparison of Spark-based KFCM (S-KFCM) and Hadoop-based KFCM (H-KFCM). The SAR images were gathered as the data for the experimental study. The experimental results showed that the S-KFCM was efficient in terms of execution time on H-KFCM.

Peng et al. [294] designed a parallel nonlinear clustering algorithm DenPeak. The experimental task involved nine synthetic datasets that represent the typical nonlinearly separable datasets. The experimental results depicted the performance regarding the number of points, the dimension of data, and the number of nodes present in the Spark cluster.

Han et al. [295] proposed a parallel DBSCAN clustering algorithm. The experimental setup analyzed a synthetic dataset. The experimental results presented the scalability and efficiency of the proposed algorithm.

Jędrzejowicz et al. [296] proposed a classification algorithm based on Kernel-based fuzzy C-means clustering. The proposed algorithm was tested on several datasets from UCI Machine Learning Repository [23]. The experimental results showed the proposed approach had better performance in execution time with good accuracy.

Tsapanos et al. [297] proposed a Kernel K-Means clustering algorithm with a distributed implementation called Trimmed Kernel K-Means, which employed subsampling. The experimental study involved *Youtube Faces* dataset. The experimental results indicated that the proposed method runs much faster than the original Trimmed Kernel K-Means.

Bharill et al. [298] proposed a fuzzy-based clustering algorithm to handle voluminous data. The proposed algorithm was a partitional clustering algorithm called Scalable Random Sampling with Iterative Optimization Fuzzy c-Means algorithm (SRSIO-FCM). The experimental work involved analysis of four datasets collected from UCI repository [23], viz. *Minst8m*, *Replicated-USPS*, *Monarch-Skin*, and *SUSY* dataset. The experimental results showed that the proposed SRSIO-FCM had superior performance in terms of F-measure, Adjusted Rand index (ARI), an Objective function value, runtime, and scalability.

Gouineau et al. [299] implemented an algorithm called PatchWork, a scalable density-grid clustering algorithm. The proposed algorithm was evaluated with four synthetic datasets, viz. *Jain*, *Spiral*, *Aggregation* and *Compound* datasets. The

PatchWork was also tested with a real-world dataset, *SFPD (San Francisco Police Department) Incidents* dataset. The outcomes presented that the proposed algorithm was considerably faster in comparison with the K-Means implemented with Spark.

After reviewing articles in clustering with distributed parallel computing architecture, the following insights were discovered.

- The predominant algorithm that has been parallelized is the K-Means. It can be implemented on different chunks of data independently and finally merging the results, which will be the approximation of the optimal result but with lesser execution time.
- K-Means, Fuzzy c-means are highly explored in the distributed parallel environment.
- Density-based clustering algorithms are least explored in Hadoop and Spark framework leading to a fertile ground for the new researchers.
- Evolutionary methods for clustering are explored to a great extent with ACO, ABC, DE, Glowworm Swarm Optimization (GSO), PSO, Fireworks algorithm, and Cuckoo search. These algorithms with modifications and the other algorithms of evolutionary methods can be explored.
- SOM is the least exploited technique.

2.4.5 Outlier Detection / Intrusion Detection System

2.4.5.1 Hadoop MapReduce-based Journal papers

Zhu et al. [300] proposed an algorithm combining cell-based outlier detection and single-layer perceptron. The experimental data were two-dimensional datasets generated randomly by MATLAB 7.0. The experimental results exhibited, the parallelized cell-based outlier detection algorithm produced better accuracy than its serialized counterpart.

Soltani Halvaiee and Akbari [301] proposed a model for credit card fraud detection using the Artificial Immune System (AIS), known as AIS-based Fraud Detection Model (AFDM). The experiments worked on transactions collected from a

Brazilian bank. The improvisation made on the base algorithm AIS demonstrated an improvement of accuracy, reduction in cost, and reduction in system response time in the experimental results.

Natesan et al. [302] proposed a Parallel Binary Bat Algorithm for efficient feature selection and classification for network intrusion detection in the Hadoop platform (HPBBA). The proposed method analyzed the *KDDCup99* dataset [130]. The outcomes proved that the proposed method was superior to the sequential computing approach.

El-Alfy and Alshammari [303] proposed an approach based on rough sets for scalable attribute subset selection to detect intrusion using the parallel GA. The work utilized four cybersecurity datasets, viz. *Spambase*, *NSL-KDD*, *Kyoto*, and *CDMC2012*. The outcomes showed that the proposed approach reduced the execution time without degrading the solution quality regarding the reduct size.

Rathore et al. [304] propose a real-time intrusion detection system (IDS) for high-speed Big Data environments. The study used three publicly available datasets, viz. *DARPA* [36], *KDDCUP99* [130], and *NSL-KDD* dataset [305]. The proposed system employed five different ML classifiers, viz. J48, REPTree, RF tree, conjunctive rule, SVM, and Naïve Bayes classifiers. The experimental outcomes showed that REPTree, and J48 are the best in terms of accuracy and efficiency.

2.4.5.2 Spark-based Journal papers

Carcillo et al. [306] presented a SCAlable Real-time Fraud Finder (SCARFF), which is an amalgamation of Big Data tools, viz., Kafka, Spark and Cassandra with a machine learning approach for fraud detection in credit cards. The experimental work involved more than 8 million of *e-commerce transactions* from almost 2 million cardholders. The experimental outcomes proved that the proposed method is scalable.

2.4.5.3 Hadoop MapReduce-based Conference papers

Tanupabrungsun and Achalakul [307] introduced a feature reduction method with GA/kNN for anomaly detection in manufacturing. The experimental study involved

six standard datasets from UCI machine learning repository [23], viz. *Connectionist*, *WBDC*, *Ionosphere*, *Hill Valley*, *Musk*, and *Wine*. The results showed the proposed algorithm had a comparable accuracy with its sequential counterpart with excellent scalability.

Aljarah and Ludwig [308] proposed a method for IDS based on a parallel PSO clustering algorithm, called IDS-MRCPSO. The experimental study involved *KDD99* intrusion detection dataset for evaluation of the proposed approach. The results showed that the IDS-MRCPSO was efficient and scalable with dataset size. Also, it had close to linear speedup.

Xiang et al. [309] proposed an approach for intrusion detection employing extreme learning machine with the MapReduce framework, called MR_ELM. The experimental study involved *KDDcup99* dataset. The experimental result showed that the proposed MR_ELM had an excellent efficiency with respect to execution time, speedup, and sizeup in comparison with the local ELM.

Sharma et al. [310] proposed a classification approach for IDS. The experimental study used *NSL-KDD* dataset [305], which was derived from the *KDDcup99* dataset by refining it for missing and duplicate values. The experimental results showed the proposed classifiers in the MapReduce environment had more accuracy, specificity, precision, F1 scores than its corresponding WEKA implementations.

2.4.5.4 Spark-based Conference papers

Gupta and Kulariya [311] proposed a method whereby a fast and efficient intrusion detection in the massive network traffic is implemented. The experiment involved two real-time network traffic datasets: *DARPA KDD'99* dataset [130] and *NSL-KDD* dataset [305]. The implemented feature selection algorithms involved correlation-based feature selection and Chi-square feature selection, and classification algorithms included LogR, SVM, RF, Gradient Boosted Decision Trees, and Naive Bayes. The experimental results were compared and contrasted.

Kumari et al. [312] implemented K-Means clustering for anomaly detection in network traffic utilizing the MapReduce programming environment in the Apache Spark platform. The experimental work was carried out with the *KDDCup99*

dataset. The result showed the detection of anomalies in the dataset by implementing a threshold to the distance of a data point from the nearest centroid of a cluster.

The following research insights are obtained after reviewing the articles related to outlier detection in Hadoop and Spark frameworks.

- The outlier detection or intrusion detection has been implemented with classification and clustering approaches that are popular in parallel implementation. But, it has not been explored to its full scale.
- The DM task, outlier detection, is one of the less explored areas in the distributed parallel computational environment, leading it to be a fertile ground for the budding researcher to explore.
- ANN has not been utilized in outlier detection. Hence, the rich features of ANN can be exploited for outlier detection.
- Evolutionary methods have not been explored to its full extent for outlier detection. Hence, the researchers can exploit it to its full potential.

2.4.6 Recommendation

2.4.6.1 Hadoop MapReduce-based Journal papers

Veloso et al. [313] implemented a collaborative recommendation filter employing singular value decomposition with stochastic gradient descent (SVD-SGD). The work analyzed the *Yelp* dataset [314]. The experimental results showed that the proposed method was effective and scalable.

2.4.6.2 Hadoop MapReduce-based Conference papers

Schelter et al. [315] proposed a scalable similarity-based neighborhood methods. The proposed approach employed pairwise item comparison and top-N recommendation. The experimental study was carried out with *Movielens* dataset [316] and Flixster dataset [317]. The outcomes indicated that the proposed approach had linear scalability with the number of users and a linear speedup with the addition of new computational nodes.

Ghuli et al. [318] developed a collaborative filtering recommendation engine. They presented a comparison between item-based and user-based collaborative filtering (CF) with the help of *MovieLens* dataset [316], where the item-based CF showed better scalability than user-based CF algorithm.

Shang et al. [319] proposed a scalable collaborative filtering recommendation algorithm. The study involved a real-world dataset, *MovieLens* [316], and a synthetic dataset in analyzing the performance of the proposed algorithm. The experimental results demonstrate that the proposed implementation had scalability concerning numbers of users and items, ensuring recommendation accuracy.

Pozo and Chiky [320] proposed a method of a Distributed Stochastic Gradient Descent (DSGD) for Recommender Systems based on MapReduce computational environment. The performance of the proposed algorithm was evaluated with the implementation of *MovieLens* dataset [316]. The experimental outcomes presented a better performance of the proposed method in respect of accuracy and scalability.

Lu et al. [321] implemented a distributed item-based collaborative filtering algorithm on the Hadoop MapReduce framework. The experimental study utilized the *MovieLens* dataset [316]. The algorithm was evaluated with a root mean square error (RMSE) and mean absolute error (MAE). The proposed method improved the accuracy of the algorithm.

Subramaniyaswamy et al. [322] proposed an approach for unstructured data analysis on large-scale data. The study used the *Twitter* dataset. The unstructured data was converted to the structured format, and the proposed approach implemented collaborative filtering and sentiment analysis on the data. The proposed method showed scalability with respect to the size of the data.

Shen and Jiamthapthaksin [323] proposed an improved algorithm DIMSUM+ which is the improvised version of DIMSUM algorithm, an all-pair similarity algorithm The experimental work involved (i) the *MovieLens* 1M dataset provided by GroupLens project [316] and (ii) the *Yahoo! Music* dataset provided by Yahoo! Webscope Program [324]. The experimental results showed the proposed algorithm DIMSUM+ outperformed DIMSUM regarding accuracy and execution time.

2.4.6.3 Spark-based Conference papers

Panigrahi et al. [325] implemented a hybrid algorithm involving dimensionality reduction and Clustering techniques for user-oriented collaborative filtering method. The algorithm utilized benchmark dataset of *MovieLens* [316]. The experimental results proved the efficacy of the proposed algorithm.

Singh et al. [326] have proposed a distributed music recommendation engine using collaborative filtering. The collaborative filtering method is implemented on a distributed Apache Spark cluster. The experimental work has analyzed the ListenBrainz dataset. The performance of the proposed method is measured with root mean squared error.

Kumar et al. [327] have implemented music tagging and similarity analysis for recommendation systems employing the computational framework of Apache Spark. The proposed method is evaluated with precision, recall, F1 score, and accuracy. The proposed method gives high average true positive value and less average false positive value.

After reviewing the articles related to recommender system in the distributed parallel environment, following research insights are found.

- The recommendation system has been implemented with user-based and item-based recommendations. These approaches are susceptible to parallel implementation.
- There has been very little research work carried out by the research community in the recommendation system utilizing distributed parallel computation. A lot of exploration can be achieved in this rich area.
- A few research articles were published related to user-based and item-based collaborative filtering, thus, resulting in a most explorable area.
- Many similarity indices can be explored with different classification or clustering techniques to produce a better result.

2.4.7 Others

The articles reviewed under this category covers some articles utilizing ML techniques such as Bayesian Network, Rough set approximations, PSO, SGD, LDA, GA, ABC, etc.

2.4.7.1 Hadoop MapReduce-based Journal papers

Zhang et al. [328] suggested a parallel approach for computing rough set approximations. The research work implemented three parallel algorithms involving real dataset *KDDCup-99*, and three synthetic datasets generated with the help of the WEKA data generator [169]. The performance of the proposed parallel algorithms was evaluated with speedup, scaleup, and sizeup performance metrics and found to be superior to its serial counterpart.

Chen et al. [329] employed a feature selection method based on differential privacy and Gini index. The experimental study included validation of the proposed algorithm with the utilization of five benchmark datasets from UCI repository [23], viz. *Car Evaluation*, *Mushrooms*, *Connect-4*, *Covertype*, *Pokerhand*; and one synthetic dataset *S1*. The results indicated that the proposed algorithm was time efficient than its centralized counterpart.

Qian et al. [330] proposed a parallel feature reduction algorithm. The experimental study was carried out with two real datasets, i.e., *Mushroom* and *Gisette* [23] and four synthetic datasets (*DS3-6*). The research outcomes proved that the proposed parallel algorithm was efficient and scalable.

Yue et al. [331] proposed a parallel and incremental approach for learning of Bayesian Network (BN). The research work utilized (i) *Chest-clinic network* and (ii) *HepaerII* network. The experimental results depict the proposed methods to be scalable and efficient.

Zhang et al. [332] proposed i^{2}MapReduce, an incremental processing extension to MapReduce. Four iterative mining algorithms were implemented: (i) PageRank (one-to-one correlation) with *ClueWeb* dataset [34], (ii) Single Source Shortest Path (SSSP, one-to-one correlation) with *ClueWeb2* dataset [34], (iii) Kmeans (all-to-one correlation) with *BigCross* dataset [333], and (iv) GIM-V (many-to-one correlation) with *WikiTalk* dataset [334] and also a one-step mining algorithm, Apriori was also

implemented with Twitter dataset. Experimental results demonstrated significant performance enhancements of i^{2}MapReduce compared to both plain and iterative MapReduce.

Yue et al. [335] presented a method for measuring the similarity among the users utilizing Bayesian Network called User Bayesian Network (UBN). The authors analyzed *DBLP* and *Sina Weibo* dataset. The similarities depend on the social, behavioral interactions as well as on the contents involved in it.

Wang et al. [336] proposed a MapReduce-based cooperative particle swarm optimization (MRCPSO). The experimental work involved 11 scalable optimization problems. The experimental results showed that the proposed algorithm MRCPSO outperformed its sequential counterpart in terms of execution time and the quality of the solution.

2.4.7.2 Spark-based Journal papers

Qi et al. [337] proposed a parallel genetic algorithm based on Spark (PGAS). The experimental work carried out with synthetic and real-world benchmarks presented by Jia et al. [338] and Garvin et al. [339]. The results showed that the proposed approach outperformed its sequential counterpart in both size and execution time in almost all benchmarks.

Eiras-Franco et al. [340] implemented the parallel version of four traditional feature selection algorithms, i.e., InfoGain, RELIEF-F, CFS, and SVM-RFE. The study included seven high dimensional datasets, viz. *Higgs*, *Epsilon*, *KDD99*, *Isolet*, *USPS*, *Poker* and *KDDB*. The research outcomes demonstrated that the proposed approach had the speed-up and efficiency in comparison with sequential approach.

Ramírez□Gallego et al. [341] proposed a parallelized version of the minimum-redundancy-maximum-relevance method, called fast-mRMR for dimensionality reduction. The experimental work involved a comparison of CPU-based, GPU-based, and parallel distributed execution performance. The study included real datasets for a sequential version of fast-mRMR vs. mRMR, viz. *Lung*, *NCI*, *Colon*, *Leukemia*, and *Lymphoma*. Performance comparison of CPU- and GPU-based architecture was executed with the help of a synthetic dataset. The distributed parallel computational environment was studied with *ECBDL14*, *epsilon*, and *kddb* datasets. The

experimental results showed that fast-mRMR outperformed the original mRMR.

2.4.7.3 Hadoop MapReduce-based Conference papers

Gemulla et al. [342] proposed an approach for large-scale matrix factorization with distributed stochastic gradient descent (DSGD). The analyzed the *Netflix competition* dataset. They compared DSGD with its sequential counterpart. The outcomes proved that the proposed approach is efficient, scalable, and fast.

Zhang and Sun [343] proposed a method for microblog mining using distributed MicroBlog-Latent Dirichlet Allocation (MB-LDA). The experimental study involved a microblog dataset originally from the *Twitter* dataset for comparative analysis of MB-LDA and distributed MBLDA. The observed results presented that the proposed MB-LDA outperformed the baseline of LDA.

Thompson et al. [344] proposed a fast, scalable selection algorithm. The study involved a synthetic dataset generated using the *TeraGen* program included in the Hadoop distribution. The results showed the proposed method outperformed several other alternatives.

Hu et al. [345] proposed a MapReduce enabled simulated annealing genetic algorithm. The proposed approach was the amalgam of the conventional GA and the SA algorithm. The hybridization in the proposed algorithm led to maintain a higher probability of getting the optimal global solution than traditional GAs. The research work involved Traveling Salesman Problem (TSP) dataset [346]. The experimental results indicated that the convergence speed of the proposed algorithm significantly outperformed its traditional genetic rivals.

He et al. [347] employed a parallel feature selection using a positive approximation, called PSFPA. The experiment used *Shuttle* and *Handwritten* datasets [23]. The results demonstrated that the proposed algorithm was efficient to process large scale and high dimensional dataset.

Hilda and Rajalaxmi [348] proposed an approach for feature selection using GA for supervised learning through the K-Nearest Neighbor (k-NN) classifier. The experimental study used five real-world datasets. The experimental results indicated that the Parallel GA produced high accuracy than other methods.

Kourid and Batouche [349] proposed an approach for feature selection, using K-Means clustering combined with Binary Particle Swarm Optimization (BPSO). The experimental work was carried out with two datasets of *cancer RNA-seq gene expression* data (gastric cancer, ESCA (esophageal carcinoma)). The scalability of the proposed method was validated with a synthetic dataset by duplicating the genes of each dataset. The experimental results proved that the proposed method outperformed the sequential approaches in terms of accuracy.

Alshammari and El-Alfy [350] proposed an approach for minimum reduct using parallel GA. The study utilized four intrusion detection datasets that are publicly available, viz. *Spambase*, *NSL-KDD*, *Kyoto*, and *CDMC2012*. The outcomes showed that the proposed approach reduced the execution time with a comparable solution on reduct size.

Yu et al. et al. [351] proposed a MapReduce-based distributed keyword search (DKS). The proposed method modeled the keyword database as a data graph. The data graph was partitioned into multiple subgraphs from which the candidate Steiner tree was searched using map operation and later combined in reduce phase to generate the Steiner tree. The top-k keywords could be found my merging Steiner trees.

2.4.7.4 Spark-based Conference papers

Li et al. [352] proposed parallel multi-objective artificial bee colony (MOABC) algorithm. The experimental study analyzed real-world datasets and showed that MOABC performed efficiently for solving multi-objective optimization problems.

Hu and Tan [353] proposed an algorithm for feature selection to classify malware based on N-Gram partitioning. The experimental study involved malicious files from the *VX Heaven virus collection* [354]. The outcomes proved the proposed algorithm to be efficient and superior.

2.5 Discussion

The review work addressed the two research questions posed at the beginning of Section 2.1. Research question Q1 is answered affirmatively because we could find

261 studies where a majority of statistical and machine learning family of techniques that are relevant to data mining tasks could be scaled up in a distributed and parallel manner under the Hadoop and Spark frameworks. The reason could be that all of them involved several vector-matrix, matrix-matrix, and matrix-vector multiplications, which are easily amenable for parallelization and distributed processing.

Research question Q2 is answered as follows: The techniques left out are either not amenable to parallelization or not yet attempted because of lack of popularity and/or wide applicability. Examples could be LASSO [355], ridge regression [356], Linear discriminant analysis [357], Quadratic discriminant analysis [358], canonical correlation analysis [359], factor analysis [360], correspondence analysis [361], conjoint analysis [362], and hierarchical clustering [363].

Various publication resources of this review and distribution of these are presented in Table 2.2. The review noticed that the highest number of papers were from IEEE, followed by Elsevier and Springer. Figure 2.5 depicts an eagle-eye view of the distribution of articles publisher-wise that has been considered in the current review work. This illustrates that the IEEE has the maximum number of publications followed by Elsevier, Springer, and so on. Figure 2.6 depicts the count of articles from individual conference proceedings. It emphasizes that the highest number of articles is from IEEE, followed by ACM and Springer. Similarly, Figure 2.7 depicts the publisher-wise distribution of journal articles, which indicates Elsevier has produced the highest number of journal publications followed by Springer and IEEE. The Figure 2.9 depicts the count of articles from individual journals that have been surveyed. According to the count, the highest number of journals are published in Information Sciences, Knowledge-Based Systems, and Neurocomputing, followed by IEEE Transactions on Knowledge and Data Engineering, and so on.

The year-wise distribution of the reviewed papers with the categorization of Hadoop MapReduce-based and Apache Spark-based is depicted in Figure 2.8. The highest number of articles (i.e., 59) appeared in 2015, followed by 2014 and 2016 in the Hadoop MapReduce framework category. Similarly, in the case of Apache Spark-based papers, the highest number of articles (i.e., 24) appeared in 2016, followed by 2015.

Figure 2.10 depicts the distribution of the papers according to various data mining tasks. It reveals that there is an almost equal share of classification and cluster-

Table 2.2: Publication wise Reviewed Papers

Source	No. of Journal Papers	No. of Conference Papers	Total No. of Papers
ACM	-	17	17
Elsevier	46	7	53
IEEE	15	126	141
Springer	28	14	42
Taylor & Francis	4	-	4
Wiley	1	-	1
Others	1	2	3
Total	95	166	261

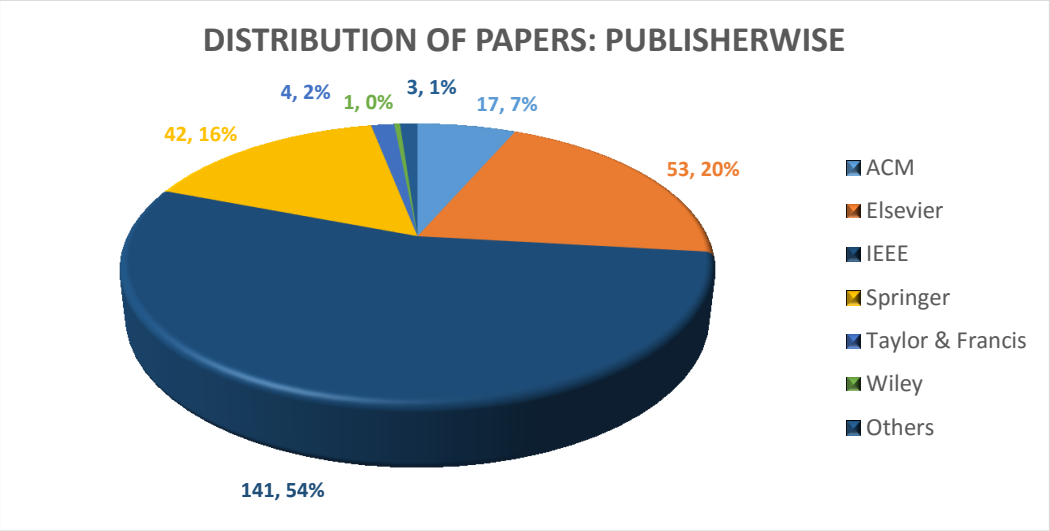

Figure 2.5: *Publisher-wise distribution of Papers*

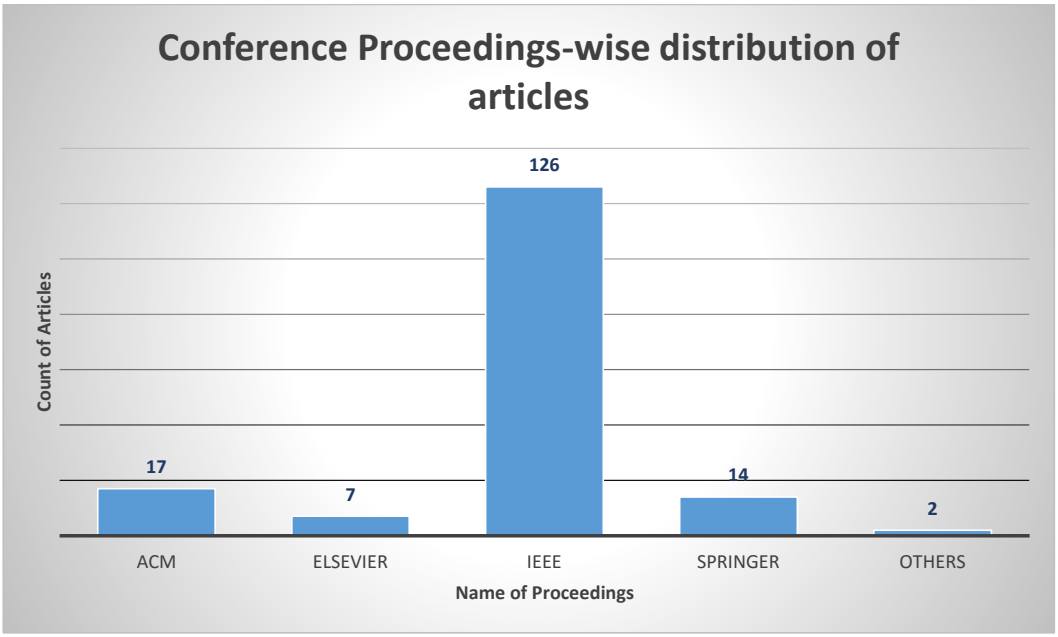

Figure 2.6: *Conference proceedings-wise distribution of articles*

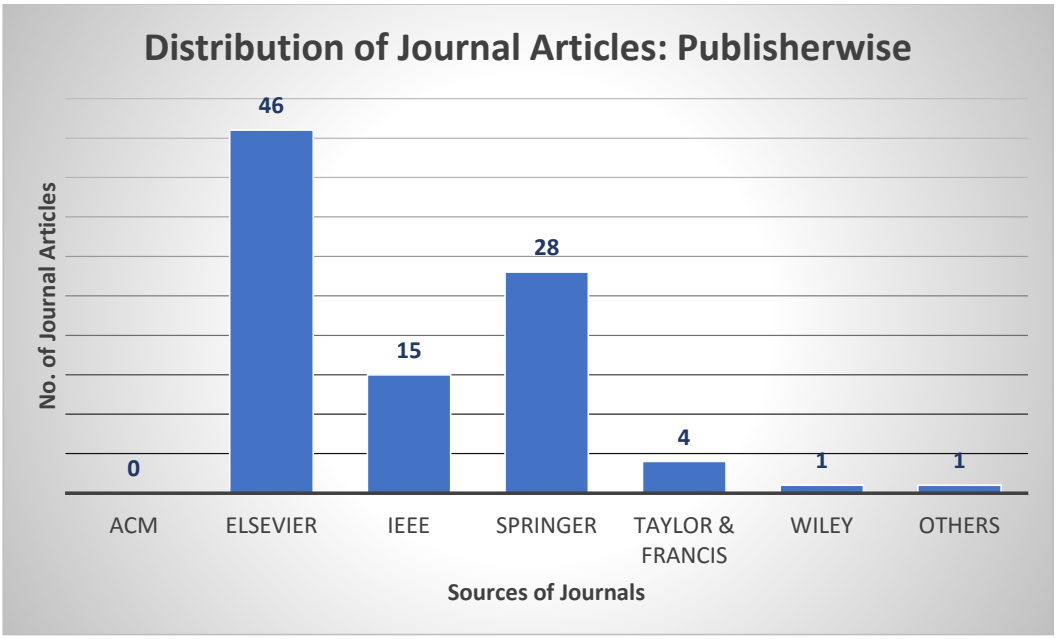

Figure 2.7: *Publisher-wise distribution of Journal articles*

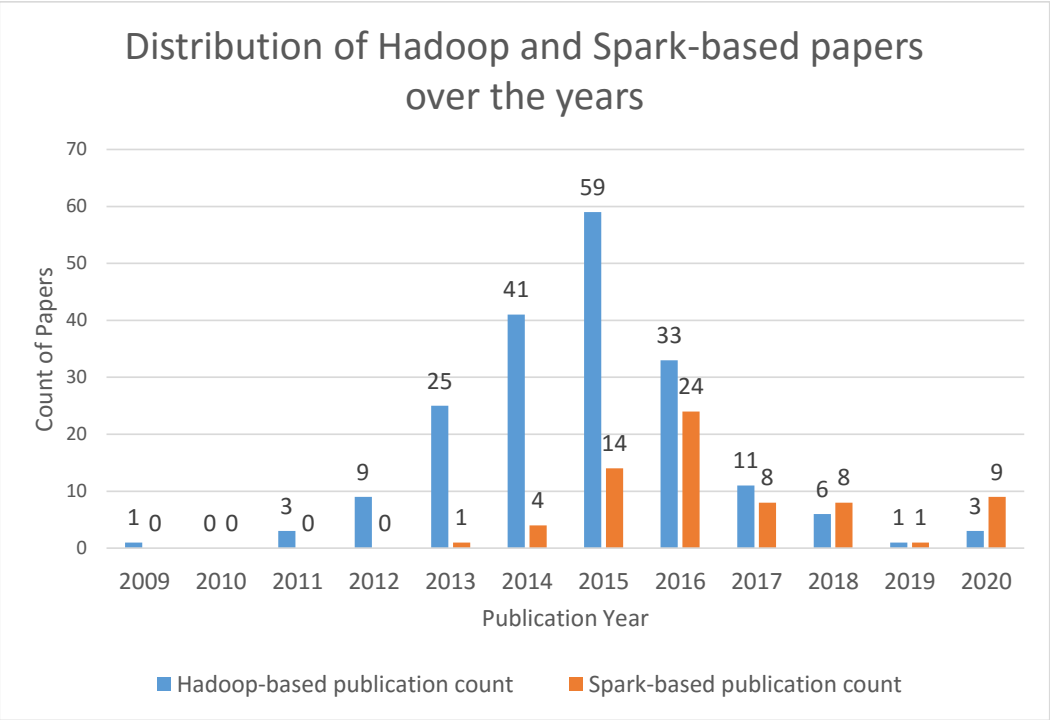

Figure 2.8: *Year-wise Hadoop and Spark-based publication count*

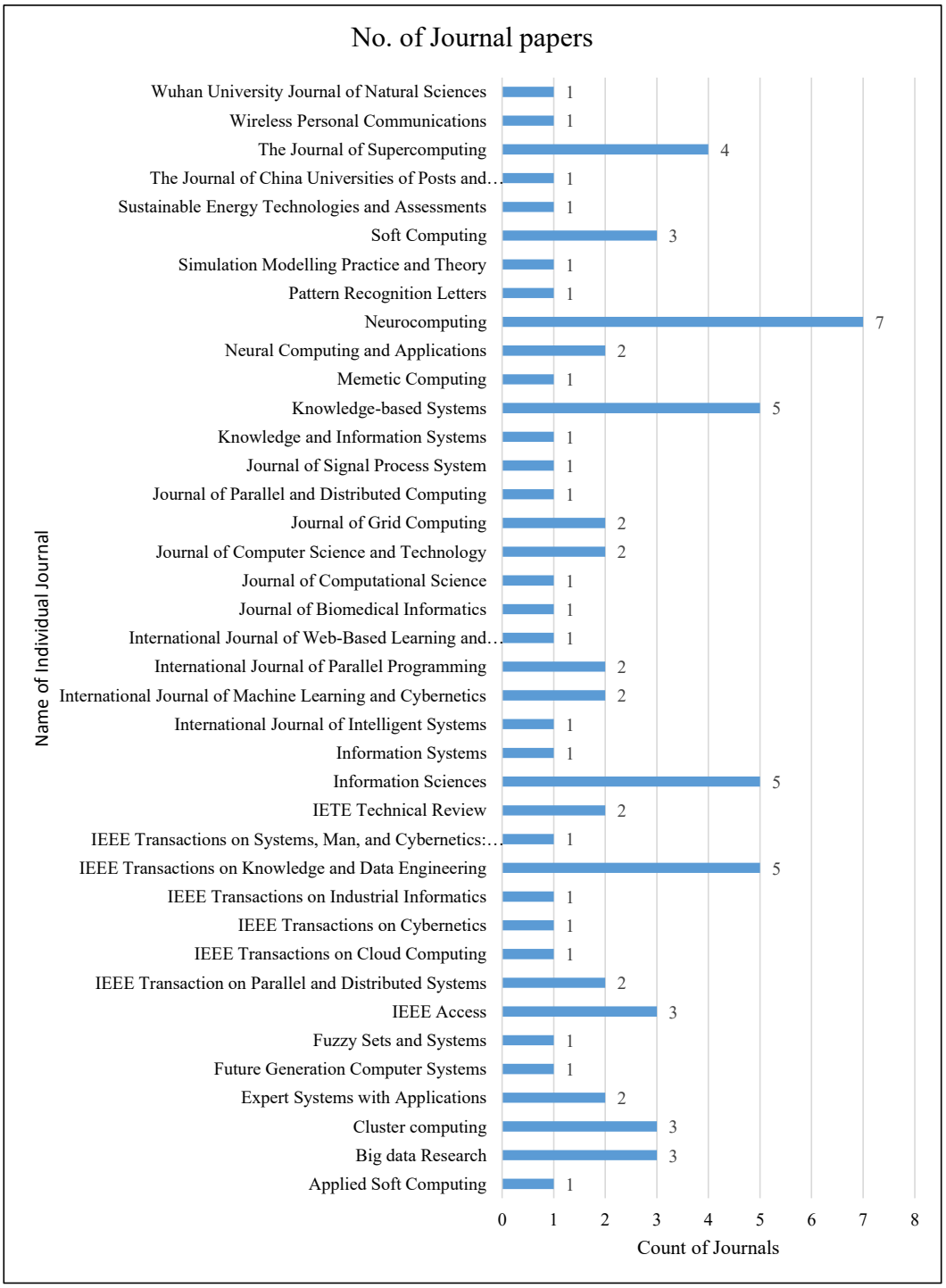

Figure 2.9: *Individual journal-wise distribution of articles*

ing articles in horizontal scaling platforms using Hadoop MapReduce and Apache Spark followed by ARM/pattern mining and so on.

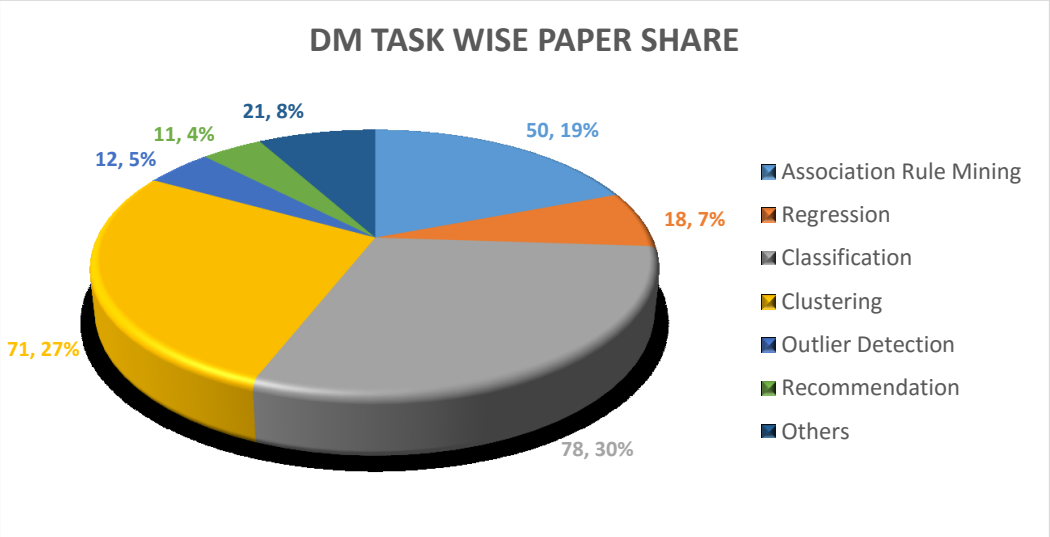

Figure 2.10: *Datamining task-wise paper distribution*

Figure 2.11 depicts one step further in detail, giving the paper distribution, again categorized on the computational framework, i.e., Hadoop MapReduce and Apache Spark with journal and conference distribution. It demonstrated that a maximum number of publications have occurred in clustering followed by classification and association rule mining task in Hadoop-based conference papers whereas, classification task is followed by clustering in the case of Hadoop-based journal articles. Similarly, the publications in the category of classification are followed by clustering and association rule mining in the case of Spark-based conference articles. Moreover, it depicts the total count of Hadoop-based articles in each data mining task category that presents clustering is the most explored task, followed by the classification, ARM, and so on. Similarly, the total count of Apache Spark-based articles present classification is followed by clustering, ARM, and so on.

Figure 2.12 depicts the distribution of the papers according to different machine learning techniques involved in various data mining tasks further categorized with Hadoop and Spark article. It presents the distribution of papers in different machine learning techniques used for Association Rule Mining (ARM) / Frequent Pattern Mining (FPM), Regression, Classification, and Clustering. It indicates some

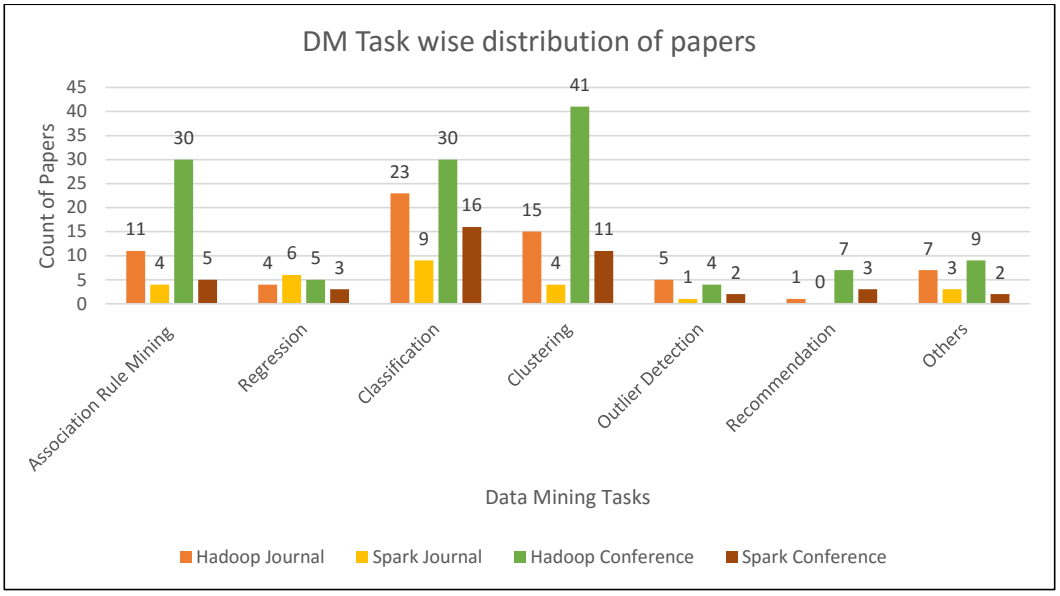

Figure 2.11: *Distribution of papers: data mining task and computational framework-wise*

research has been carried out in Apriori and FP-growth and FP-tree for ARM or FPM. The Apriori technique is the most widely used for ARM. BPNN and ELM have been explored for regression, but overall the regression task has been less explored in Hadoop and Spark platform. The SVM, k-NN, and RF are the most used techniques for classification, out of which the SVM is the most widely used technique for classification in both Hadoop and Spark platform. Similarly, the K-Means algorithm is rigorously studied by the research community, followed by fuzzy clustering. We can observe that the centroid-based clustering technique has fascinated many researchers for its study.

We can figure out that the Apriori for ARM; SVM, K-NN for classification; and K-Means for clustering are widely used in the MapReduce paradigm. It is due to the inherent nature of the algorithm and the fact that the model can be developed with partial data in a parallel and distributed manner.

The 'Others' mentioned in the ARM category represents all those articles not utilizing Apriori, FP growth, or FP tree approach. They have used other approaches, viz. parallel Genetic Algorithm (GA), GA with k-NN, High Utility Itemset (HUI) miner, etc. The 'Others' mentioned in the Regression category represents all ar-

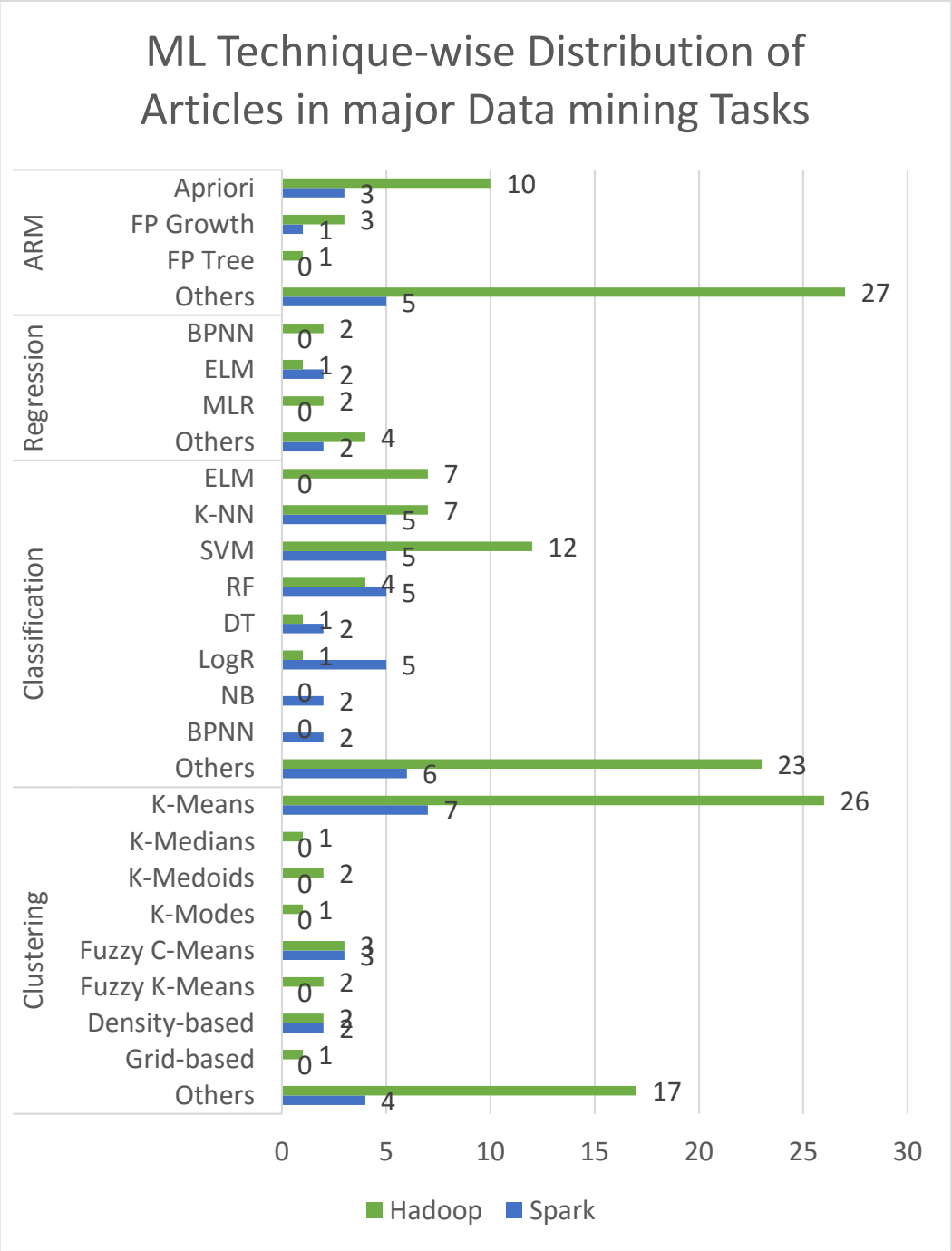

Figure 2.12: *ML Technique-wise Distribution of Articles for different DM tasks categorized with Hadoop and Spark*

ticles utilizing Support Vector Machine (SVM), Random Forest (RF), Stochastic Gradient Descent (SGD) for Extreme Learning Machine (ELM), etc. The 'Others' mentioned in the classification category represents all articles utilizing Ant Colony Optimization (ACO), Genetic programming, Auto-Associative Neural Network (AANN) with Particle Swarm Optimization (PSO), Social Network Analysis (SNA) algorithms, etc. The 'Others' in the clustering category represents all articles utilizing Latent Dirichlet Allocation (LDA), Differential Evolution (DE), Self Organizing Map (SOM), Artificial Bee Colony (ABC) algorithms, etc.

The Figure 2.13 depicts summarized information of the datasets analyzed in the papers reviewed. It demonstrates that 68% of the articles reviewed have used only real-world datasets, 11% of the articles have used only synthetic datasets, whereas 15% of the articles have used both real and synthetic datasets. This analysis indicates the trend of the researchers for having a preference for real-world datasets over a synthetic one. Figure 2.14 depicts the distribution of different types of datasets used over Hadoop MR and Spark articles.

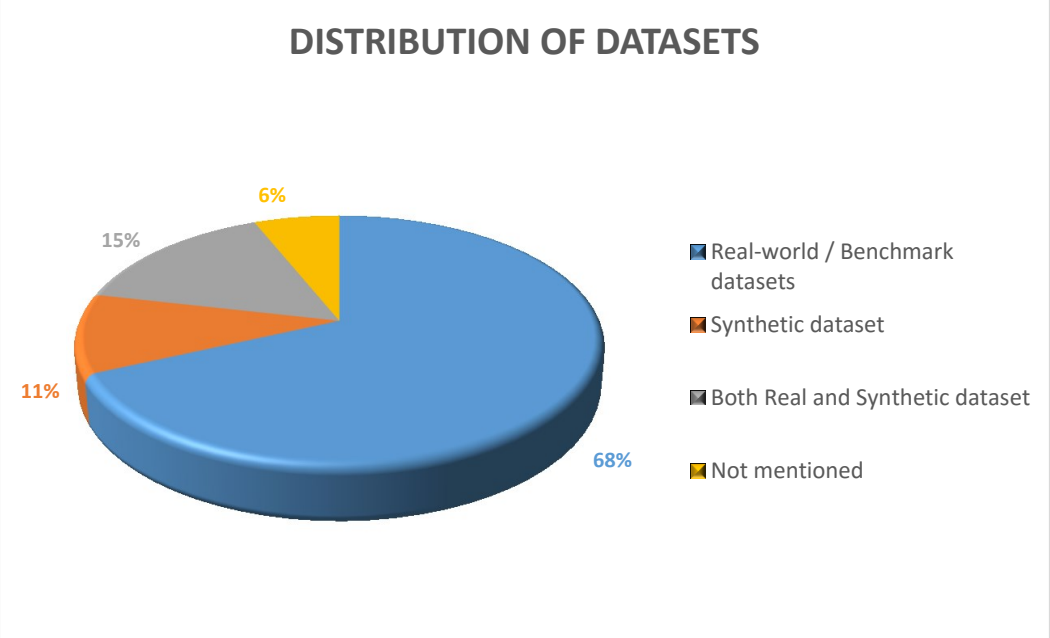

Figure 2.13: *Distribution of papers: Dataset wise*

These are the different descriptive insights we can draw from the review work. The survey work led us to some research gaps those are presented in the following

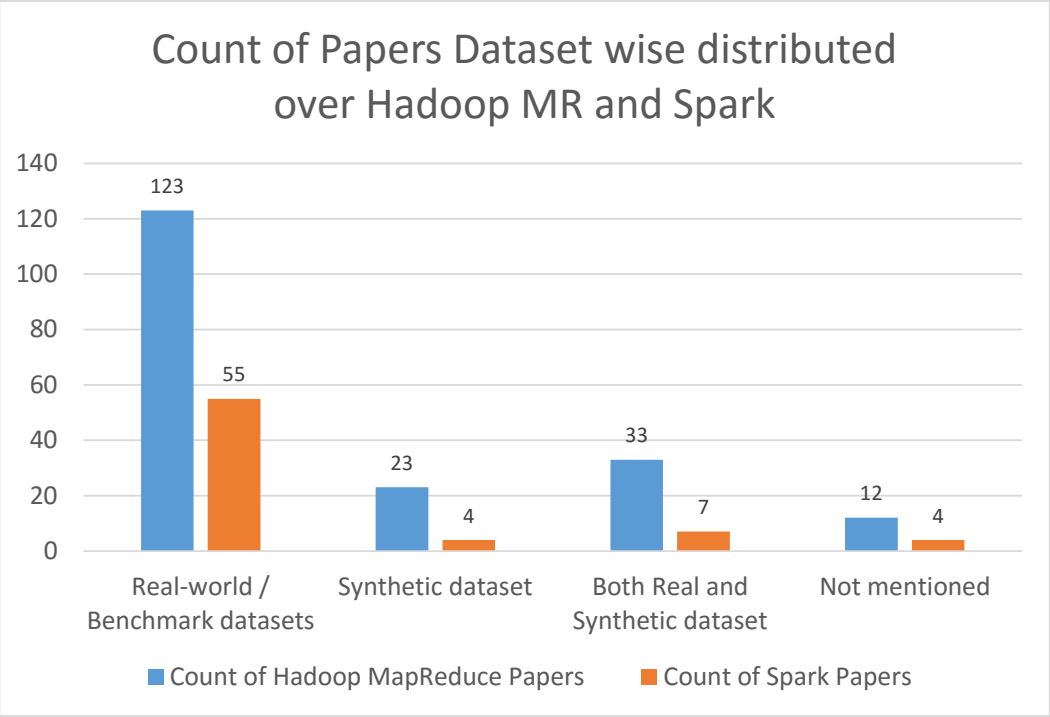

Figure 2.14: *Count of Papers distributed dataset-wise in Hadoop MR and Spark*

section in the open problems part.

Apart from the above, the review discovers the following insights:

- The advanced method of the ARM, such as Parallel High Utility Itemset Miner (PHUI-Miner), yielded better results compared to its sequential counterpart. Parallel Highly Informative K-ItemSet (PHIKS), and Complete Parallel Apriori (CPA) yielded better results compared to other parallel methods.
- The modified approaches such as parallel ELM (PELM), Parallel Auto-Associative ELM (AAELM) yielded better results for the regression task with respect to their sequential counterpart.
- The improved methods, viz. Parallel Incremental Extreme SVM (PIESVM), Adaptive Distributed Extreme Leaning Machine (A-ELM*), and Parallel Random Forest (PRF) produced better results while solving the classification task in comparison with its serial version. The improved method Weighted Ensemble classifier based on Distributed ELM (WE-DELM) yielded a better result in comparison with parallel ELM.
- The ANN, with its different form along with Evolutionary optimization techniques, can be employed for regression and classification due to the inherent nature of ANN to be parallelized. The parallelization of ANN is achievable as it involves vector-matrix, matrix-vector, or matrix-matrix multiplication.
- The modified method of FCM, called MRFCM, produced a better result than its sequential counterpart. The improved parallel K-Means and IM-K-Means approach yielded a better result in comparison to the parallel version of K-Means.
- Many papers are found to experiment with the real-world dataset(s). The synthetic dataset was used for the test of the scalability of the algorithms. The synthetic datasets were generated through some data generators or replicating the samples of small real-world datasets.

- ❖ Neither the large-scale real-world datasets nor the benchmark datasets for evaluating DM tasks in Big Data paradigm are available in the public domain. Therefore, many researchers replicated the existing benchmark dataset many times so as to get an artificial flavor of Big Data. This jugglery leads to the inaccurate modeling of a DM task and ultimately rendering the model not so useful.
- ❖ There are only a few works that have been carried out employing ML techniques with Evolutionary optimization techniques.

2.6 Conclusion and Open Problems

The review work included 261 papers from 2009 to 2019. This chapter presents a comprehensive review of data mining tasks with a distributed parallel computing environment with horizontal scaling. The different data mining tasks on which we categorize the review papers are Association Rule Mining / Pattern Mining, Regression / Prediction, Classification, Clustering, Outlier Detection (OD) / Intrusion Detection (ID), and Recommendation. We could see that Hadoop MapReduce was the preferred choice over Apache Spark, as seen by the number of publications reviewed. One reason for this is the latter is relatively new.

The study concludes with some of the key open problems:

- ❖ There remains a lot of work to be carried out in Regression / Prediction in both Hadoop MapReduce as well as Apache Spark environment.
- ❖ The Kernel density estimation has not been parallelized.
- ❖ The rough set approximation is scantily scaled out.
- ❖ A very few papers were found related to various NN architectures. The different NN architecture such as GRNN, QRNN, PNN, WNN, RBFN, Group Method of Data Handling (GMDH), Functional Link Neural Network (FLNN), Single Multiplicative Neural Network (SMNN), Sigma-Pi-Sigma Neural Network (SPSNN), Multi-Layer Morphological Neural Network (MLMNN) is not explored at all, thus, resulting in it into a fertile ground to explore.

- ❖ Though a lot of work is reported in clustering, classification, and ARM, they are carried out with the Hadoop MapReduce framework. Therefore, there is also a great scope for conducting the research in the Apache Spark environment, i.e., with the in-memory computational framework, in order to reap the benefits of the Spark environment.
- ❖ The least explored OD / ID, and Recommendation presents an open area to conduct more research work.
- ❖ The unbalanced datasets and high dimensional datasets pose significant challenges in Big Data paradigm too. These areas need full exploration.
- ❖ Streaming data analytics, Social Network Analysis (SNA), and Social Media Analysis (SMA) also pose challenges. There are very few papers on real-time or quasi-real-time streaming analysis. These are open areas for future work.
- ❖ Scalable Advanced Massive Online Analysis (SAMOA) has not been explored at all for streaming data analytics.
- ❖ Soft computing hybrid models and their applications were explored, but not as much as the field deserves.
- ❖ Evolutionary algorithms, in their stand-alone mode, were not explored for solving the data mining tasks. These are again some open areas for future research.
- ❖ A few papers were found on fuzzy logic-based techniques covering fuzzy classification and fuzzy clustering. Their combination with optimization techniques can be an area to explore further.
- ❖ SOM is scantily employed, thus rendering an open space for exploration.
- ❖ Deep learning is now a growing research area that can handle complex nonlinear data with high dimensionality. It can be applied to solve DM tasks. Deep learning architectures are computationally expensive due to the presence of multiple hidden layers, where there is a scope or parallelization. They can be explored with in-memory computing utilizing Apache Spark.

- ❖ The Relevance Vector Machine (RVM), Independent Component Analysis (ICA), Singular Value Decomposition (SVD), Latent Semantic Indexing (LSI), One-Class SVM (OCSVM), Class Association Rule Mining (CARM) techniques are not explored at all, thus, resulting into an open space for exploration.
- ❖ There are hardly any research papers on mining of spatial and temporal data, as they pose numerous challenges due to their high dimensionality, which has to be explored with respect to different DM tasks.
- ❖ It is also found that a few articles were published on GPGPU based vertical parallelization using MapReduce and Apache Spark. Therefore, this provides a fertile ground for budding and future researchers.
- ❖ To the best of our knowledge, we came across only a few research papers hybridizing horizontal parallelization with vertical parallelization, i.e., a cluster of GPGPU based machines. Therefore, the cluster of GPGPU is an area to explore.
- ❖ Surprisingly, Big Data visualization using MapReduce or Apache Spark is a conspicuously rarely researched topic.
- ❖ In our current review work, we could include only a few articles based on large-scale optimization algorithms implemented using Hadoop MapReduce or Apache Spark frameworks. Hence, there is a vast scope for the researchers to explore this exciting area.
- ❖ The absence of papers in banking, insurance, and finance sectors is conspicuous as far as the application domain of BDA is concerned.

The multiple research works that have been carried out after going through a thorough survey of literature and finding gaps in it also require a literature survey related to the proposed work. Those literature reviews are presented below.

For one-class classification, the following articles were studied: Ravi and Singh [364] performed OCC using Auto-Associative Extreme Learning Factory (AAELF) for bankruptcy prediction in banks, credit risk prediction, and phishing detection.

Tax and Duin [365] proposed Support Vector Data Description (SVDD) for the detection of outliers. Strackeljan et al. [366] demonstrated a fault detection system using SVDD and a feature selection method for OCC. Another approach used for OCC was the implementation of a One-class Support Vector Machine (OC-SVM) for patient classification by Mourão-Miranda et al. [367].

For one-class classification using auto-encoder or auto-associative neural network, the following articles were referred: Pandey and Ravi [368], where they have proposed a model using PSOAANN for phishing detection in emails. Ravi et al. [369] presented the classifying capability of PSOAANN in bankruptcy prediction in banks.

The experimental work involved the Apache Spark environment, and it led to the study of several research works related to the work, and they are as follows: the study of an open-source distributed machine learning library, MLLib, presented by Meng et al. [370]. MLI, an API for distributed machine learning proposed by Sparks et al. [371]. Also, some application-oriented articles implemented with Spark were studied, viz. Bharill et al. [298], where they have proposed a clustering algorithm called Scalable Random Sampling with Iterative Optimization Fuzzy c-Means algorithm(SRSIO-FCM) with Spark to handle the challenges associated with Big Data clustering. Panigrahi et al. [325] have proposed a Hybrid Distributed Collaborative Filtering Recommender Engine (HDCFRE) using Spark. McNeil et al. [372] implemented Scalable Real-time Anomalies Detection and Notification of Targeted Malware in Mobile Devices (SCREDENT) using Spark. Bello-Orgaz et al. [373] surveyed the different tools for machine learning for social media data on a Big Data paradigm using Spark for massive data processing.

The implementation of PECM led to the study of several articles on clustering. Foremost the K-Means algorithm proposed by Macqueen [374] was referred. Several articles implementing a parallel version of K-Means were studied, viz. [212], [233], [242], [243], [245]. other centroid-based clustering approaches were studied, such as the parallel k-median algorithm proposed by Ene et al. [240]. The parallel version of the k-medoid proposed by Zhu et al. [251], and Jiang and Zhang [259]. Also, fuzzy c-means proposed by [216], [263].

Some other clustering methods were also studied, viz. the parallel density-based clustering proposed by Han et al. [295], and Chen et al. [278]. Parallel grid-based

clustering proposed by Chen et al. [278], and Gouineau et al. [299]. Articles implementing clustering in parallel and distributed environment of Spark were referred, viz. [290], [292], [297]. And finally, the sequential version of ECM proposed by Song and Kasabov [375] was referred to conduct the work.

The work EGRNN++ was carried out after a strong literature review, which includes the following articles. For the implementation of EGRNN++, which is the parallel version of GRNN, proposed by Specht [376], was referred. The different articles involving applications of GRNN that were studied are as follows. Exchange rate forecasting employing GRNN by Leung et al. [377]. GRNN implemented by Kayaer and Yildirim [378] for medical diagnosis of diabetes, for image quality assessment by Li et al. [379]. Li et al. [380] implemented GRNN with a fruit fly optimization algorithm for power load distribution prediction. Software reliability prediction employed with GRNN by Mohanty et al. [381]. Pradeepkumar and Ravi [382] employed GRNN for forecasting financial time series volatility. Raj Kiran and Ravi [383] employed GRNN for software reliability prediction. Kamini et al. [384] implemented GRNN for the prediction of cash demands in ATMs. Ravi et al. [385] employed GRNN for financial time series prediction.

The work PNN++ was also carried with a sound literature study, which includes the article about PNN, which is proposed by Specht [386]. There are several articles spanning different application domains employing PNN that are referred to as follows. Othman and Basri [387][employed PNN for brain tumor classification; Virmani et al. [388] for Breast density classification which has a significant association with breast cancer; Sweeney et al. [389] for chromosome classification; Lozano et al. [390] for wine classification; Mo and Kinsner [391] for power line fault classification. In a novel application, Nishanth and Ravi [392] implemented PNN for categorical data imputation. Ravisankar et al. [393] employed PNN for financial statement fraud detection. Mohanty et al. [394] implemented PNN for web-services classification. Ravi et al. [395] implemented a hybrid version of PNN along with PCA, and the PNN was trained with a genetic algorithm for bank performance prediction. Ravisankar et al. [396] employed PNN for the failure prediction of dotcom companies. Nishanth et al. [397] implemented PNN for the prediction of the severity of the phishing attack. Sundarkumar and Ravi [398] proposed PNN for the classification of unbalanced datasets in the banking and insurance sector.

Ghosh et al. [399] hybridized PNN with Restricted Boltzmann Machine in a hybrid deep learning architecture for sentiment classification and reported the best results for this hybrid compared to several other hybrids.

The work PRBFN has led to the study of several articles related to regression and classification, which are mentioned in the Chapter 8. The parallel implementation of RBFN by De Souza and Fernandes [400] in a field programmable gate array (FPGA) was also referred. Among the other referred articles, there were articles related to parallel clustering, viz., parallel K-Means employing MapReduce framework proposed by Zhao et al. [239], an improved version of parallel K-means using MapReduce proposed by Liao et al. [242]. The K-Means++ [401] with optimized initial cluster centers selection. The parallel version of K-Means++ proposed by Bahmani et al. [402]. Bisecting K-Means [403], which is a combination of hierarchical clustering and K-Means clustering.

There are some related articles that are referred to for the completion of the research work are mentioned in the respective chapters and not discussed here to avoid repetition.

Chapter 3

Parallel Distributed Hybrid Regression Model

This chapter presents a proposed parallel distributed hybrid model in which the amalgamation of the Auto Associative Extreme Learning Machine (AAELM) and Multiple Linear Regression (MLR), performs Big Data regression. The next section of the chapter presents an introduction to the current work. Section 3.2 presents the proposed approach. The experimental setup is described in Section 3.3. Results and discussion are presented in Section 3.4. The chapter is concluded in Section 3.5.

3.1 Introduction

With the emergence of SMAC (Social, Mobile, Analytics & Cloud) stack in Information Technology, we are confronted with huge volumes of data coming at a high velocity and from a variety of sources in various forms in almost all branches of science, engineering, finance, and business. This has led to the crystallization of the definition of what is known as Big Data comprising three Vs [404]. The data is highly voluminous and exceeds the storage and computing capability of a single machine. Hence, it requires a parallel distributed architecture for storage and analytical purposes.

Voluminous datasets not only possess a massive number of records but also are plagued by the curse of dimensionality of the feature space. Therefore, dimension-

ality reduction of the feature space often becomes mandatory. Principal component analysis (PCA) [405], [406] is often used as a method of dimensionality reduction in many applications, such as image processing, pattern recognition, and signal processing. But, the fundamental drawback of the PCA is that it assumes there is a linear correlation among the features, also known as multi-collinearity. However, when multi-collinearity is absent, one could suspect non-linear correlations among the features. Therefore, researchers have shown keen interest in non-linear PCA (NLPCA), which is a non-linear generalization of PCA and overcomes the aforementioned drawback of PCA. Kramer [407] proposed an auto-associative neural network for NLPCA. Baldi and Hornik [408] proposed Back Propagation Neural Network (BPNN) for PCA.

In this chapter, we propose AAELM to perform NLPCA and extract the output of the hidden nodes of the AAELM and teat them as Non-Linear Principal Components (NLPCs), once the training was over. The literature review suggests that a hybrid architecture comprising AAELM and Multiple Linear Regression (MLR) in tandem for performing Big Data regression has not yet been reported. Therefore, we extracted the NLPCs from the trained AAELM and fed them to MLR for regression purposes. Since we intend to exploit this hybrid architecture in the Big Data paradigm, we analyzed two data sets, taken from the web, using the Hadoop MapReduce architecture. The advantage of AAELM is that its training algorithm, apart from being very fast and single-pass, is free from the drawbacks of the backpropagation algorithm used for training the traditional AANN. The points mentioned above motivated us to propose a hybrid architecture AAELM+MLR.

In this chapter, we propose and implement the AAELM+MLR for Big Data applications using MapReduce to improve the scalability and speed of the AAELM+MLR. Moreover, as a prominent parallel data processing technique, MapReduce [409], [410] can process the decomposable problems with huge amounts of data that can effectively run in a parallel manner on a distributed system and has been used in a variety of applications for its scalability, ease-of-use and fault tolerance [411], [412], [413], [414], [415]. In training the AAELM, one must remember that the most expensive computational part is a series of matrix multiplication operations used in the calculation of the Moore-Penrose generalized

inverse matrix. The matrix multiplication operation is decomposable, and the computational cost can be reduced drastically by parallelizing using Mapreduce [416].

3.2 Proposed Approach

In this chapter implementation of MLR hybridized with AAELM in the parallel distributed framework is presented for a regression problem. The next two sections present an overall introduction to ELM and NLPCA, followed by the proposed architecture.

3.2.1 ELM

Extreme learing machine (ELM) was proposed [417] as a faster alternative to the extant single layer feed forward neural networks. ELM is a 3 layered neural network, where the weights connecting the input and the hidden layers are assigned randomly, while that between the hidden and the output layers are estimated as a linear problem using Moore-Penrose inverse. Output produced by ELM is:

$$f_L(x) = \sum_{i=1}^{h} \beta_{iL} * g(w_i.x + b_i) \tag{3.1}$$

where, $\beta = [\beta_{1L}, \beta_{2L}, \ldots, \beta_{hL}]^T$ is the output weight vector connecting the hidden layer to the L^{th} output node; W_i is the weight vector connecting input nodes to the i^{th} hidden node; b_i is the bias corresponding to i^{th} hidden node. In the chapter, g is sigmoid function; x is the record vector.

After transforming the input records into a sigmoid hidden layer matrix using random weights and biases, the next step involves training, which involves estimating the weights between the hidden nodes and the output node. It is achieved by solving the minimum norm least square solution to the following problem.

$$\min_{\beta \in R^{L \times M}} \|H\beta - Y\|^2 \tag{3.2}$$

Optimal solution to above problem is given by $\beta = H^+Y$, H^+ is the Moore Penrose generalized inverse of matrix H, where β is minimum norm least square

solution to the problem $H\beta = Y$, where H is the hidden layer output matrix.

$$H = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_h \cdot x_1 + b_h) \\ \vdots & \ddots & \vdots \\ g(w_1 \cdot x_n + b_1) & \cdots & g(w_h \cdot x_n + b_h) \end{bmatrix}$$

and Y is the taining data target matrix

$$Y = \begin{bmatrix} y_1^T \\ \vdots \\ y_n^T \end{bmatrix} = \begin{bmatrix} y_{11} & \cdots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{n1} & \cdots & y_{nm} \end{bmatrix}$$

Whenever H is of full rank; $H^+ = \left(H^T H\right)^{-1} H^T$. In a rare situation, when H is rank deficient, H^+ can be calculated using Singular Value Decomposition [418].

ELM has universal approximation ability, and it maintains its ability with randomly generated hidden nodes, as shown in theoretical studies [417]. It has an excellent generalization performance as compared to a backpropagation based feed-forward neural network [417].

3.2.2 NLPCA

NLPCA is a non-linear generalization of linear PCA. The principal components are curves (non-linear), which are generalized forms of straight lines [419]. Non-linear PCA can be achieved by using an auto-associative neural network, also known as an autoencoder or replicator network. Initially, such an auto-associative neural network is a multi-layer perceptron that performs an identity mapping, meaning that the output of the network is considered to be identical to the input. However, in the middle layer of the network achieves dimension reduction non-linearly through the use of non-linear activation function. The nodes in this layer turn out to be the desired non-linear principal components [407]. If there exists a non-linear correlation between variables, then NLPCA can describe the data with greater accuracy than the traditional linear PCA.

3.2.3 Proposed AAELM driven NLPCA based regression

In this chapter, the proposed model is a hybrid model, wherein, the first phase involves AAELM, and the second phase comprises MLR (see Figure 3.1).

AAELM consists of three layers, namely the input layer, hidden layer, and the output layer, which performs identity mapping between the input and output layer. The hidden layer output is treated as NLPCs. The generated NLPCs are fed to MLR for prediction in Phase-II. Here the number of hidden nodes, a user-defined parameter, is less than the number of input nodes.

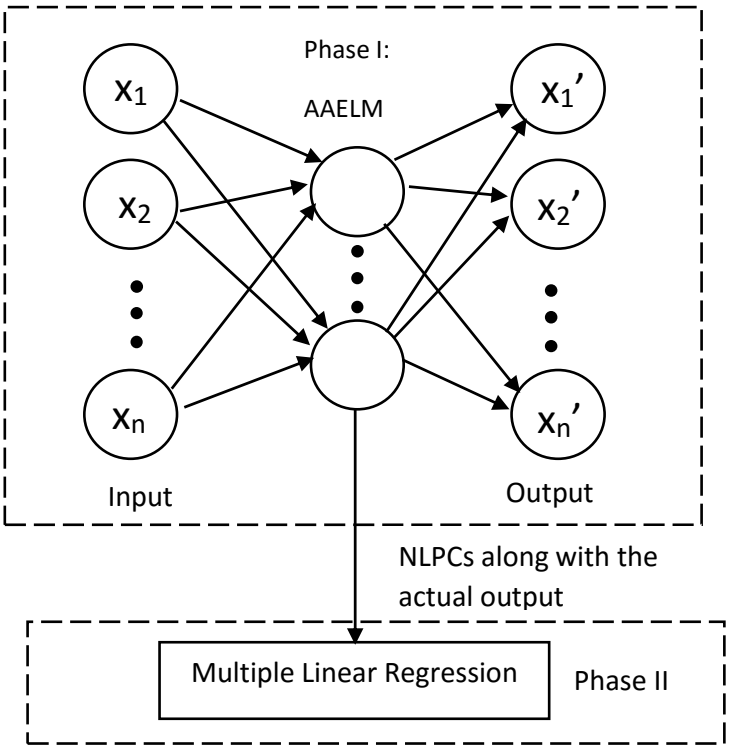

Figure 3.1: *Architecture of AAELM+MLR hybrid model*

3.2.4 Algorithm for the Hybrid model

Training algorithm with training dataset

Phase-I

1. Standardize the data matrix comprising only the features and not the dependent variable.

2. The standardized data matrix (X) is then converted into typed bytes format (sequence format) using Mapreduce [420], [421].

3. Choose the number of hidden nodes, h, a value lying between 3 and d-1, where d is the feature dimension of dataset.

4. *Random feature mapping of input matrix into a non-linear sigmoid matrix using MapReduce*:

   The matrix from step 2, X is multiplied with a random matrix, whose entries are uniformly distributed between -1 and 1. Then, h bias node values are generated using uniform distribution U(-1, 1) and are added to each row of the resultant matrix. After that, the sigmoid activation function is applied to each entry of the matrix, resulting in the matrix H, which signifies the output of the hidden nodes. The generated random matrix and bias values are saved for the test data.

   In this step, X is divided into smaller blocks, where a block refers to a matrix having the same number of columns as X matrix but having very fewer rows than the X. The number of rows in a block can be specified by the user. Mappers (MapReduce) perform multiplication of block with random matrix and then add bias values to each row of the resultant matrix and subsequently, apply sigmoid function g(), to each entry of matrix (see Figure 3.2). The output of Mappers is fed to the identity reducer, which produces the final output.

5. Then, $H^T H$ and $H^T X$ are computed using MapReduce.

   Computation of $H^T H$:

   $H^T H = \sum_{i=1}^{n} H_i^T H_i$, where H is divided into n small matrices H_i, having the same number of columns as H but smaller number of rows. Mapping operation is performed as $H_i^T H_i$. This mapping operation produces a matrix with h×h dimension. Corresponding to this h×h matrix, h key - value pairs of form (row_id, row_array) will be produced where row_id varies from 0 to h-1 and row_array is the vector corresponding to the row_id. Corresponding

to each row_id there is a reduce operation, which adds the rows element by element corresponding to that row_id [421] (see Figure 3.3). Code for matrix multiplication is available at GitHub [420].

Similarly, $H^T X$ *is computed as follows*:

$$H^T X = sum_{i=1}^{n} H_i^T X_i$$

Here Mappers distribute data to reducers. Reducers perform the task of block multiplication and addition. The summation method is similar to $H^T H$.

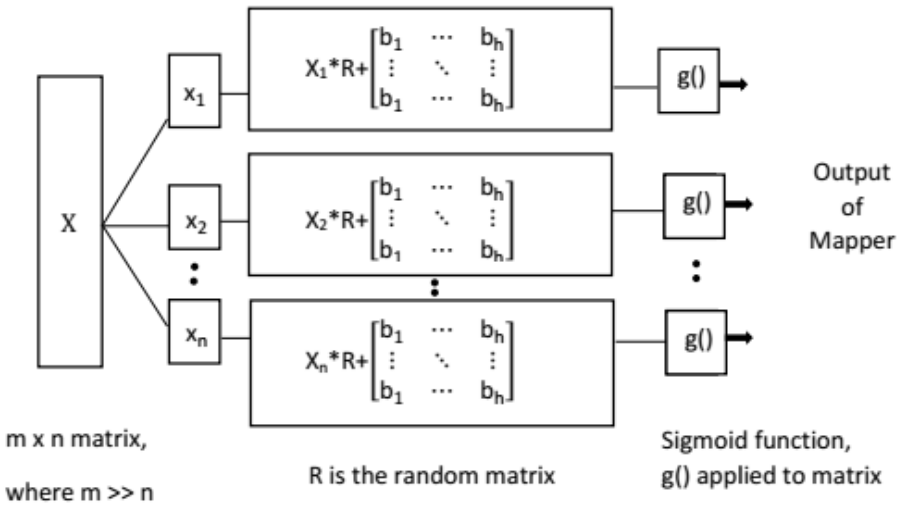

Figure 3.2: *Step 4 of Training Algorithm*

6. The result obtained in the previous step is multiplied locally in a node in the following way:
   $\beta = \left(H^T H\right)^{-1} H^T X$. As $\left(H^T H\right)$ is of order h×h and $H^T X$ is of order h×d so that this computation can be easily done on a single node.

7. Compute H * β using MapReduce. H * β is the matrix comprising the predicted values of the output variable of the AAELM, which is the same as the input variables themselves, as AAELM is auto-associative.

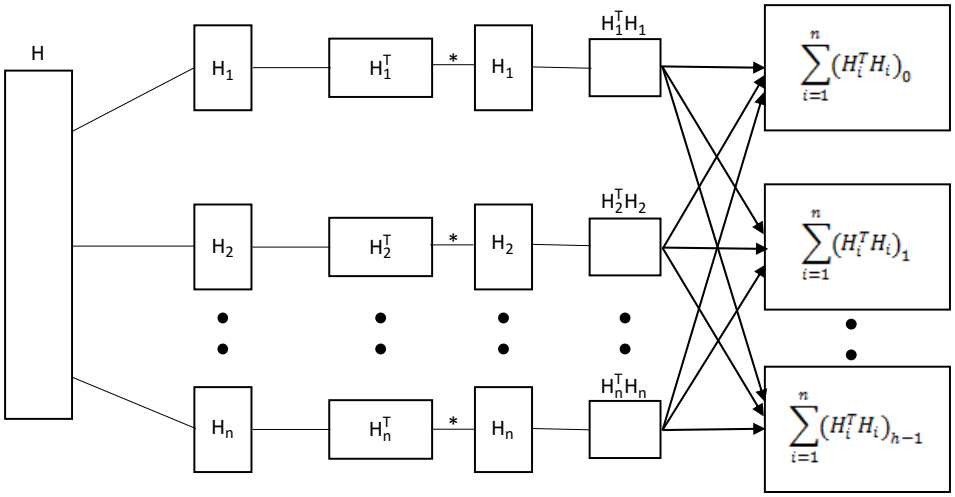

Figure 3.3: *Multiplication of $H^T H$*

8. Compute mean square error (MSE) for the predicted data.

9. Repeat the steps from 4 to 10 for the different number of hidden nodes. The number of hidden nodes should vary from h = 3 to h = l - 1, where l is the dimension.

10. Choose the number of hidden nodes for which the model gives the minimum MSE.

Phase-II

11. Compute H by fine tuning the number h, in order to obtain minimum RMSE. Compute $H^T H$ and $H^T Y$, where Y represents the target vector corresponding to training dataset. Above computations are done using MapReduce similarly as demonstrated matrix computations in phase I.

12. Compute $\alpha = \left(H^T H\right)^{-1} H^T Y$ locally.

Algorithm for testing the trained model with a test set

1. Standardized test data matrix is multiplied with the random matrix, and then bias values are added to each row of the result. Subsequently, the sigmoid

function is applied to each entry of the resultant matrix. Let us denote it as H_test.

MapReduce is applied to the above step, similar to step 4 of the training of the hybrid model. Random matrix, bias node values, and the number of hidden nodes are the same as used in step 11 of the training algorithm.

2. Multiply H_test with α obtained in step 12 of the algorithm for training hybrid model to obtain the predicted result for test data.

3.3 Experimental Setup

Hadoop version 2.6.0 is used for the experiments. A cluster of 7 nodes is connected through a 100 megabit network using Ubuntu 14.04 platform. The replication factor is 3, while the block size is 64 MB. MapReduce is used for parallelizing the process. The detailed hardware description of cluster is presented in Table 3.1.

Python 2.7.6 is used with Dumbo API 0.21.36 for Hadoop Streaming. It is a python framework for working with Hadoop. It is mature, concise, and easy to debug. It is suitable for complex jobs that may involve complex keys and in a situation where multiple MapReduce steps are involved. It carries out the task of serialization using typed bytes. It quickly reads sequence files. Dumbo programs can be executed very efficiently using a single command line.

Table 3.1: Hardware specification of the cluster

Node type	Node count	CPU	Clock speed	RAM
Master	1	Intel(R)Core(TM)i5-4200M	2.5GHz	8 GB
Slave	7	Intel(R) Core(TM) 2 Duo CPU E7500	2.93GHz	4 GB
Slave	1	Intel(R) Core(TM) 2 Duo CPU E7500	2.20GHz	8 GB

3.4 Results and Discussion

In order to demonstrate the proposed approach, the hybrid model has analyzed two datasets. The details of the datasets, results generated after analyzing the datasets are presented in further sections.

3.4.1 Datasets Analyzed

The effectiveness of AAELM+MLR hybrid model is tested on the following datasets,

(i) the data from an array of gas sensors involving different gas mixtures, and

(ii) Airline flight dataset.

The gas sensor dataset [422] was collected from UCI ML repository [423]. It has 4208262 numbers of samples. The data corresponding to the Ethylene-CO mixture has been used in this chapter. The input data size is 1.071GB after converting each entry to a floating-point variable. There are 16 features in this dataset, which denote the value of sensor readings. The sensors respond to gas concentration, with a sensitivity that is different for each gas type. Hence, the response of the sensor is indeed correlated with the input gas concentration. These 16 sensor readings are used as independent variables to predict the concentration of Ethylene gas. The other variable, which is the concentration of carbon monoxide, is treated as an environment variable.

The Airline flight dataset is collected from http://statcomputing.org/dataexpo/2009/. Six years of data from the year 2000 to 2005 has been considered for analysis. Seven attributes have been selected for the work, which is as follows: (i) CRS departure time, (ii) CRS arrival time, (iii) CRS elapsed time, (iv) Distance, (v) Taxi-out time, (vi) Departure-delay, (vii) Arival-delay.

Arrival-delay is the dependent variable, and the rest are independent variables. The original data is transformed so that the CRS departure time and CRS arrival time are converted into the minute format. The negative arrival and departure delay

are excluded from the data. Data is standardized before using the proposed methodology. The total number of instances is 10024916. The input file size is 1.1 GB after converting each entry to a floating-point variable.

3.4.2 Analysis of Results

The experiments are conducted using a 10-fold cross-validation framework. The results of the experiments are presented as average MSE and MAPE in Table 3.2 and Table 3.3, respectively. The sensitivity analysis is performed by studying the influence of the number of hidden nodes of the AAELM on the overall accuracy of the AAELM+MLR hybrid. The optimal number of hidden nodes is 13 and 4 in the case of Gas sensor dataset and air flight dataset, respectively.

Table 3.2: Average MSE Values over 10-Fold Cross Validation

Dataset	MLR	Hybrid	t-statistic
Gas sensor array	0.001	0.0008	283.484
Airline flight	1.88	1.0000003	145.828

Table 3.3: Average MAPE Values over 10-Fold Cross Validation

Dataset	MLR	Hybrid	t-statistic
Gas sensor array	4.707	4.263	14.208
Airline flight	8001.799	107.437	168.699

It is clear from Table 3.2 and Table 3.3 that the hybrid model outperformed the MLR model on both datasets in terms of both MSE and MAPE. This is because standalone MLR cannot account for the non-linear relationships among the input features. The proposed hybrid model not only models the non-linearity in the data but also generates NLPCs in the form of hidden nodes outputs, which are, in turn, fed as input to MLR. The t-test performed at a 1% level of significance to figure out if the results are statistically significant, indicating that the hybrid is indeed statistically superior to MLR (see Table 3.2 and Table 3.3).

3.5 Conclusions

The proposed work is a novel application of AAELM towards performing NLPCA and then non-linear principal component regression with the help of MLR for Big Data regression analysis under the MapReduce paradigm. The experiments conducted and the statistical significance test performed demonstrates that the hybrid AAELM+MLR model outperforms the standalone MLR model in terms of both MSE and MAPE.

Chapter 4

Parallel Distributed One-class Classifier

This chapter presents the proposed parallel distributed one-class classifier. The next section of the chapter presents an introduction to the current work. Section 4.2 presents the proposed approach. The experimental setup is described in Section 4.3. Results and discussion are presented in Section 4.4. The chapter is concluded in Section 4.5.

4.1 Introduction

A credit card is a payment card provided by every bank to eligible customers (cardholders) to make day-to-day transactions. A cardholder can pay for goods and services without having money in his account at that particular moment and can pay back to banks at a later point in time. The legitimate transactions made by the cardholder provide a pattern of his/her expenditures. If a card is stolen or accessed by some fraudsters, the transactions show an abnormal expenditure pattern, and such a transaction is called a fraudulent transaction. But, compared to large voluminous legitimate transactions, these types of transactions are relatively rare. Therefore, the identification of such fraudulent transactions is quite a complex task, and it is a part of fraud analytics. Due to the complexity involved in fraud analytics, identification of fraudulent transactions always has been an interesting research problem for

banking and financial industries, research communities, and academia. Fraud analytics can be achieved using different data mining tasks like classification, outlier detection, etc.

Classification can be performed in various ways viz. binary, multi-, and One-Class Classification (OCC). Binary classification is the process of classifying a set of samples into two classes. Similarly, the multi-class classification is used to classify a sample into three or more classes. In the case of OCC, there is a sufficient amount of samples available for one class, whereas samples for other classes will be rare. In this case, some rare samples do not belong to any of the known classes. Some examples of rare events are a failure of a nuclear plant, credit card fraud, network intrusion, etc. Hence, whenever the normal/regular samples are present in abundance, and the targeted event is scarce, a one-class classification approach can be employed to detect such a rare event.

In this study, OCC has been employed for credit card fraud detection in the Big Data framework. Big Data can be attributed using 4 V's viz. Volume, Velocity, Variety, and Veracity [424]. Volume refers to a massive amount of data. Velocity implies how fast data are generated. Variety attributed to various formats of data. Finally, Veracity represents the accuracy of the data. In the case of credit card transactions, every bank deals with a huge amount of credit card transactions every hour. Here, a huge amount refers to volume, and the number of transactions per hour implies velocity. So, credit card transactions can be attributed to two V's of Big Data, i.e., volume and velocity. Here the study has relied upon one-class classification. Since there is a scarcity of fraudulent credit card transactions and a binary classification approach needs sufficient amount of historical data for both classes like legitimate and fraudulent, the work has extended one-class classification model proposed by Paramjeet et al. [425] in Big Data environment using Apache Spark framework for credit card fraud detection. Henceforth, "Apache Spark" is referred to as "Spark" only.

In this chapter, the parallelization of a hybrid architecture involving Particle Swarm Optimization (PSO) and Auto-Associative Neural Network (AANN) has been proposed, which is referred to as PSOAANN architecture. The PSOAANN architecture was proposed by Paramjeet et al. [425]. In this chapter, the AANN is implemented in a parallel manner over a Spark standalone cluster for one-class

classification. The weight update steps using PSO is implemented in a parallel manner in all the partitions of data where the update of particles takes place in a parallel manner.

4.2 Proposed Approach

In this work (PSOAANN) is implemented in a parallel, distributed manner where the weights are updated by particle swarm optimization. The next two section presents an overall introduction to PSO and AANN, followed by the proposed architecture.

4.2.1 Particle Swarm Optimization (PSO)

PSO, developed by Eberhart and Kennedy [426] is a population-based optimization algorithm. It is a bio-inspired algorithm that mimics the behavior of a flock of birds or a school of fish in search of food. PSO implementation is easy due to the tweaking of a few parameters. PSO generates a solution by sharing knowledge mutually among all the particles present in the population to achieve the goal.

The process of PSO involves two basic steps. First, an update of the velocity of a particle. Second, the update of the position with the help of updated velocity of the particle. The two equations are given below:

$$V_d^{i+1} = w * V_d^i + c_1 * r_1^i * \left(P_d^g - x_d^i\right) + c_2 * r_2^i * \left(P_d^l - x_d^i\right) \tag{4.1}$$

$$x_d^{i+1} = x_d^i + V_d^{i+1} \tag{4.2}$$

where V_d^i is the velocity of the particle at the instant of i$^{\text{th}}$ iteration i.e. old velocity, V_d^{i+1} is the velocity of the particle at the instant of (i+1)$^{\text{th}}$ iteration i.e. new velocity, P_d^l is the best position travelled through a particle, P_d^g is the best position travelled through all the particles, x_d^i and x_d^{i+1} are the positions of a particle at the instant of i$^{\text{th}}$ iteration and (i+1)$^{\text{th}}$ iteration respectively. The subscript d is the d^{th} dimension of the data object. The variable w is the inertia weight value, c_1 and c_2 are

two predefined positive constants, r_1 and r_2 are random numbers generated through uniform distribution U(0, 1). The flowchart for PSO is depicted in Figure 4.1.

The equaiton for velocity update i.e., Equation (4.1) in PSO has three components, (i) inertia, (ii) personal influence, and (iii) social influence. The different components are shown in following equation.

$$V_d^{i+1} = \underbrace{w * V_d^i}_{inertia} + \underbrace{c_1 * r_1^i * \left(P_d^g - x_d^i\right)}_{social\ influence} + \underbrace{c_2 * r_2^i * \left(P_d^l - x_d^i\right)}_{personal\ influence}$$

- **inertia**: It ensures the particle to move in the same direction and with the same velocity.
- **social influence**: It persuades the particle to follow the best neighbor's direction.
- **personal influence**: It improves the individual particle in the search space. It makes the particle return to a previous position, better than the current position if it happens to be. It is a conservative approach by the particle to find the best position.

The above three components influence the direction of movement of the particles towards the optimal position. The particles move step-wise towards the optimum and converge to it. The Figure 4.2 shows how the particles are influenced by the different factors while merging towards the optimal value.

4.2.2 Auto-Associative Neural Network (AANN)

AANN, developed by Kramer [407], is a variant of the neural network where the number of nodes in the input layer is the same as the output layer. Thus, it is named as auto-associative. The proposed model of Kramer contains three hidden layers viz. mapping layer, bottle-neck layer, and de-mapping layer. The AANN is a supervised model where the inputs are compared against the outputs of the model. Thereupon, the error between inputs and outputs is optimized.

[1] https://www.slideshare.net/rkmohammadi/particle-swarm-optimization-pso/

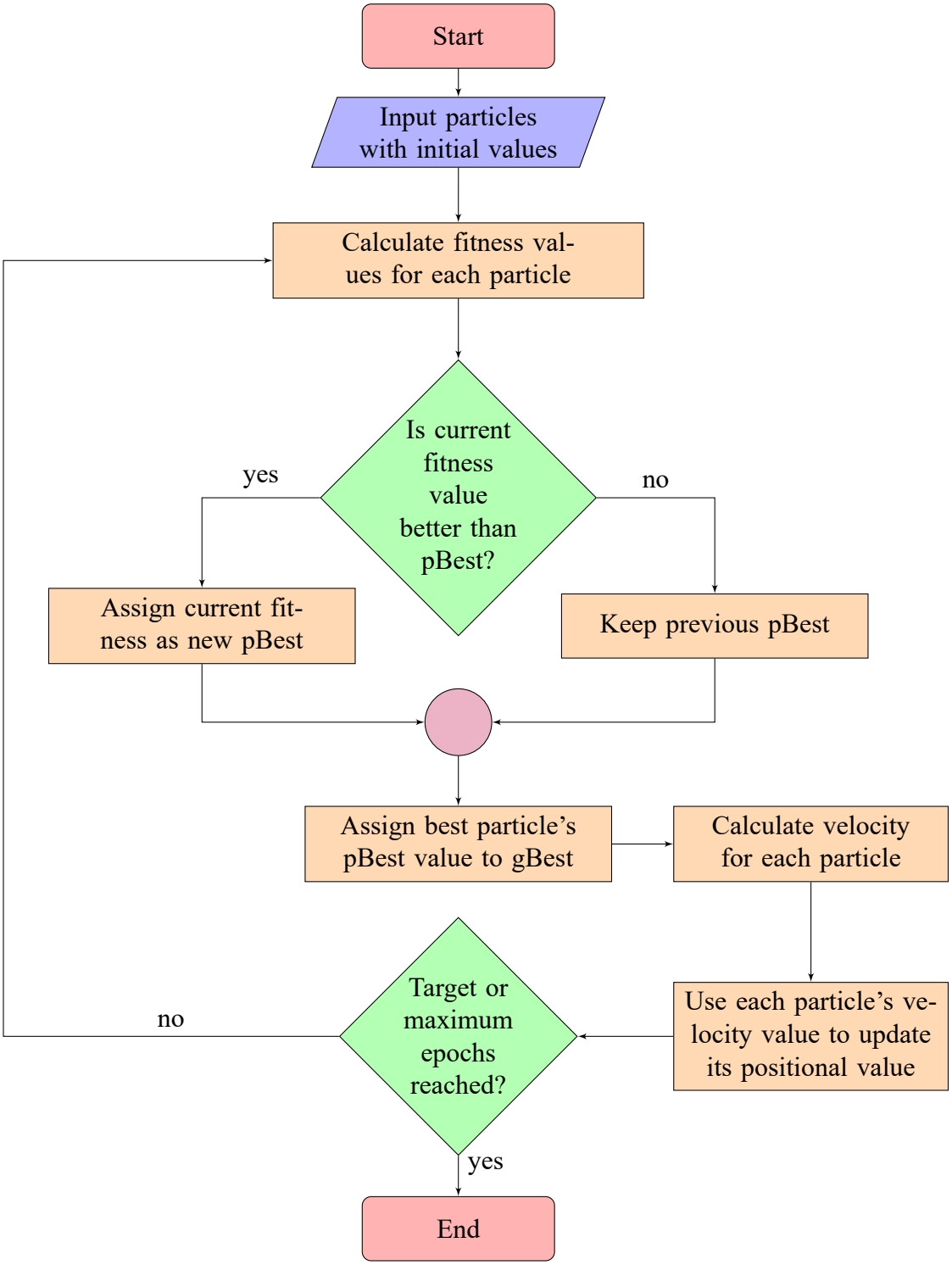

Figure 4.1: Flowchart for PSO

4.2.3 PSOAANN: Proposed architecture

Paramjeet et al. [425] proposed PSOAANN, which comprises three layers, i.e., input, hidden, and an output layer. The input and output layers contain an equal number of nodes that is to represent the same variables in the input as well as the output layer. The number of hidden nodes is specified by the user. Each input node of the input layer is connected to all the nodes of the hidden layer. Similarly, each hidden node, in turn, is connected to all nodes of the output layer. The Sigmoid activation function is used in the hidden and output layers [425].

At the end of the training phase, the nodes in the output layer contain the perturbed values of the original input variables. In other words, the PSOAANN is non-linearly transforming the original set of input variables into a set of perturbed values. There are well-known drawbacks in the back-propagation algorithm, such as slow convergence and entrapment in local minima. To overcome these drawbacks, we have used a swarm intelligence technique, PSO, which is an evolutionary approach for weight update, instead of the back-propagation.

Figure 4.3 presents the architecture of PSOAANN, which depicts the training and test modules of PSOAANN. The mean squared error (MSE) is considered as the error function. The advantages of using a three-layered AANN for one-class classification over the five-layered architecture of AANN [407] are decreasing computation time and the associated complexity.

The working procedure for PSOAANN is as follows:

Step 1: *Training phase of PSOAANN algorithm*:

Specify the required number of hidden nodes. Initialize randomly the weight values between the input and the hidden layers (W) and also between the hidden and the output layers (W′) using uniform distribution in the range $[5,5)$. The PSOAANN model is trained with negative class or majority class samples (i.e. legitimate credit card transactions) only. The input nodes take the normalized input which is calculated as below. The output nodes contain the input variables as the target variables thereby bringing in the auto association concept.

$$x_{dk}^{normalized} = \frac{x_{dmax} - x_{dk}}{x_{dmax} - x_{dmin}} \tag{4.3}$$

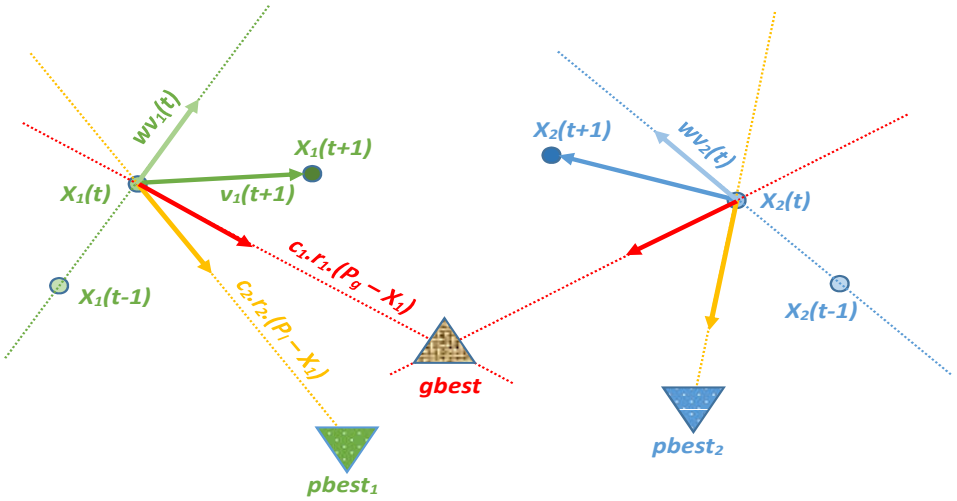

Figure 4.2: *Movement of the particles under the influence of inertia, social and personal influence in PSO*[1]

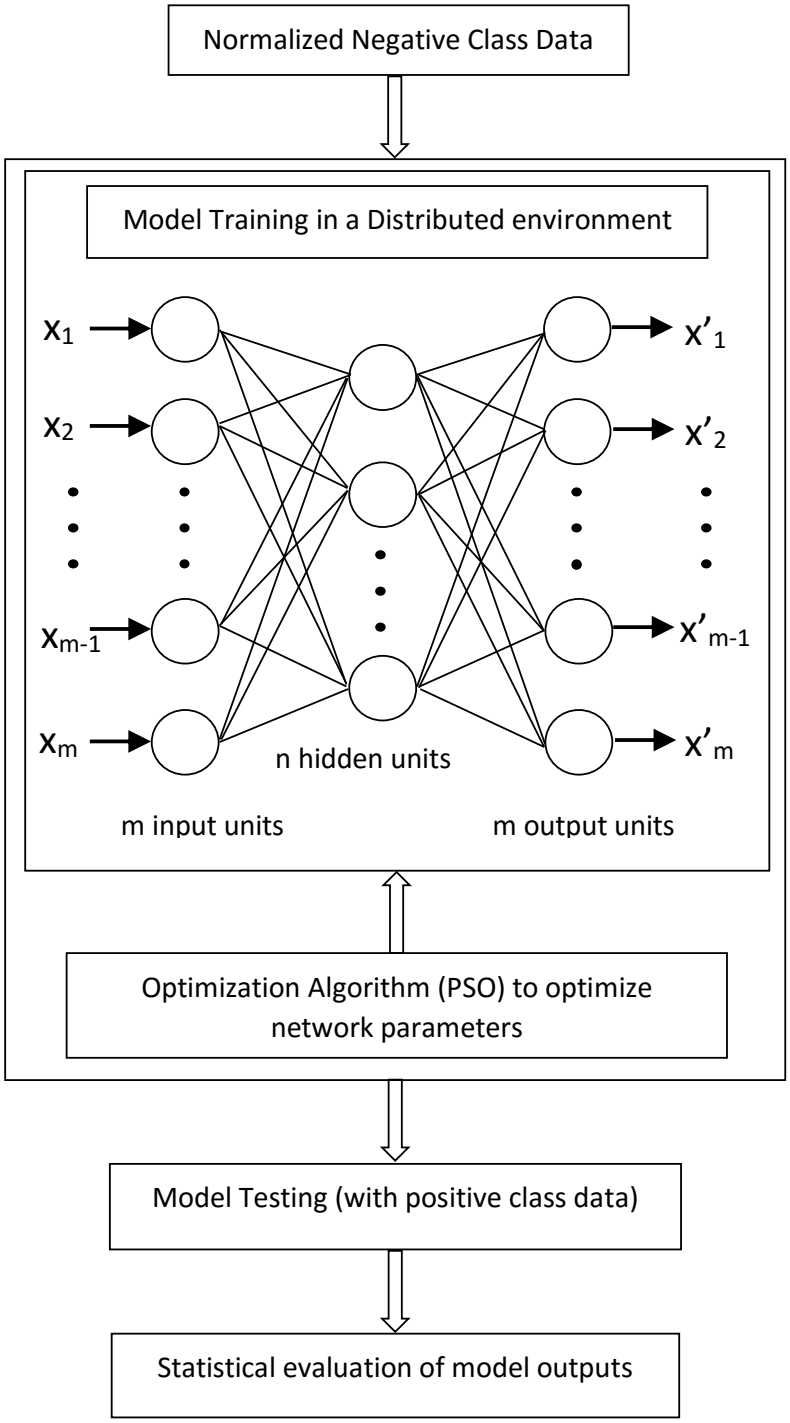

Figure 4.3: *Architecture of hybrid model of PSOAANN*

where, x_{dmax}, x_{dmin} are the maximum and minimum value of the dimension 'd' in the input data object and x_{dk} is the input value of dimension 'd' of k^{th} input object.

Step 2: Compute $\hat{x}_{dk}$ as follows: x_{dk} is the actual input and x_{dk} is the predicted input. Let *'m'* be the number of nodes in the both input and output layer; *'n'* be the number of nodes in the hidden layer. The predicted output is calculated as follows.
Computation at Hidden layer for each hidden node:

The input value is multiplied by the weight values 'W' between the input and hidden layer is given by (H_j):

$$H_j = \sum_{k=1}^{m} x_{dk} * w_{kj} \tag{4.4}$$

where, j = 1 to n, for n hidden nodes; x_{dk} is the input value of dimension 'd' of k^{th} input object and w_{kj} is the weight from k^{th} input node to j^{th} hidden node.

Then sigmoid function is applied to H_j, given as below.

$$S\left(H_j\right) = \frac{1}{1 + e^{-H_j}} \tag{4.5}$$

where, H_j is given by Equation (4.4).
Computation at Output layer for each output node:

The output of sigmoid function at hidden layer $(S(H_j))$ is multiplied by the weight values W′ between the hidden and output layer is given by (O_i):

$$O_i = \sum_{j=1}^{n} S\left(H_j\right) * w_{ji} \tag{4.6}$$

where, i = 1 to m, for, m output (= input) nodes, $S(H_j)$ is given by Equation (4.5) and w_{ji} is the weight from j^{th} hidden node to i^{th} output node.

Then sigmoid function is applied to O_i, given as below:

$$S(O_i) = \frac{1}{1 + e^{-O_i}} \tag{4.7}$$

where, O_i is given by Equation (4.6).

As the sigmoid function results in a predictive value, $\hat{x}_{dk}$ in [0, 1], which is compared with the normalized input, x_{dk} that results into the Mean Squared Error (MSE).

Step 3: Compute error measure (MSE) E as follows:

$$E = \frac{1}{mn} \sum_{k=1}^{m} \sum_{d=1}^{n} (\hat{x}_{dk} - x_{dk})^2 \tag{4.8}$$

Step 4: The MSE is optimized using PSO by updating all weights. The user-defined parameters in Equation (4.1) are considered as, inertia weight = 0.9, c_1 = c_2 = 2.

Step 5: Repeat Steps 2 to 4 until convergence is achieved or the max number of iterations is completed.

Step 6: *The Testing phase of PSOAANN algorithm*:
The PSOAANN model was trained on negative samples. The model learns the characteristics of samples belonging to the majority class. The objective is to minimize the MSE. When the model was properly trained, it was tested on positive samples, i.e., minority class or fraudulent transactions. The model will produce larger MSEs at the testing time compared to the training phase. Pattern classification is achieved at the test phase by computation of relative error for each of the features present in the input and output data. In this case, the outputs are approximations of inputs. If the relative error is having a larger value than a threshold value for all the input features, then a sample is considered to belong to the positive class. Here, the threshold value is specified by the user, and we have taken it as 0.05. Otherwise, the sample is identified to belong to the negative class.

$$Relative\ Error = \frac{|\hat{x}_{dk} - x_{dk}|}{x_{dk}} \tag{4.9}$$

The classification rate is computed as

$$Classification\ rate = \frac{No.\ of\ transactions\ classified\ as\ Fraud}{No.\ of\ transactions\ classified\ as\ Fraud} * 100 \quad (4.10)$$

The training of PSOAANN falls under both unsupervised and supervised learning. It is unsupervised because we do not provide the class or output variable information to the AANN during training. The input variables are mapped onto themselves. It is supervised because the weights of AANN are updated by PSO, where the fitness function of the particles viz., MSE, is minimized. Therefore, the PSO-based training algorithm provides supervised learning. The concise algorithm for PSOAANN is presented in Algorithm 1.

4.3 Experimental Setup

The experimental setup consists of a standalone Spark cluster using the local file system as the storage system and Apache Zeppelin as an editor. The Spark cluster comprises 9 worker nodes and a master node running the driver program. All the 10 nodes had the same configuration, i.e., Intel® Core™ i7-6700 CPU @ 3.40GHz with 8 logical cores. We allocated 24 GB of memory to worker nodes and 28GB of memory to the master node. Out of 8 logical cores, we allocated 6 logical cores in all ten nodes.

The model was executed in Spark 1.6.1 and Apache Zeppelin 0.5.6. The best execution time was achieved by tweaking the memory used by the executors in each worker node with the right number of data partitions. The data locality was achieved by using the local file system, thus improving the performance in execution time.

4.4 Results and Discussion

The number of credit card transactions is growing day-by-day rapidly. This problem of credit card fraud detection can be considered as a Big Data problem because of the following reasons. This growth leads to a high Volume of data. The speed of credit card transactions results in a high Velocity of the generation of credit card transaction data. In general, credit card transaction data is structured. Still, we

Algorithm 1: Algorithm for PSOAANN

Input: Specify the number of hidden nodes. Initialize random weight values 'W' between the input and the hidden layers and also between the hidden and the output layers 'W′' using a uniform distribution. The min-max normalized data from the majority class is used for the training of the model. The model is tested with normalized data from minority class. For PSO, the user-defined parameters are as inertia weight = 0.9, $c_1 = c_2 = 2$.

Output: 'Classification rate' which is the number of transactions identified to be fraudulent out of all the fraudulent transactions.

Data: data distributed in partitions

1 *Training phase of PSOAANN algorithm*:
The normalized training data is fed to the nodes in the input layer of AANN, where the number of nodes in the input layer is the same as the number of features, which again is the same as the number of nodes in the output layer.

2 The normalized input data is multiplied by the weight values 'W' and its summation is calculated at each hidden node. The value computed is given by H_j where, j = 1 to n, for n hidden nodes.
Then sigmoid function is applied to H_j.
The output of sigmoid function at hidden layer, $S(H_j)$ is multiplied by the weight values W′ is given by (O_i), where, i = 1 to m, for, m output (= input) nodes.
Then sigmoid function is applied to O_i
The sigmoid function results in a predictive value, $\hat{x}_{dk}$ in [0, 1].

3 Compute error measure MSE by comparing $\hat{x}_{dk}$ with the normalized input, x_{dk}.

4 The MSE is optimized using PSO by updating all weights. The PSO was executed with specified user-defined parameters.

5 Repeat Steps 2 to 4 until convergence is achieved or the maximum number of iterations is completed.

6 *The test phase of PSOAANN algorithm*:
Input the normalized test data to the trained model. Compute MSE. Compute relative error for each of the features present in the input and output data.

7 If the relative error is having a larger value than a threshold value for all the input features, then a sample is considered to belong to the positive class. Otherwise, the sample is identified to belong to the negative class.

8 The performance of the classifier is measured by computing classification rate.

can make a better detection of a fraudulent transaction if we include the profile information of the cardholder and the transaction into the fraud detection model. The inclusion of profile information incorporates the Variety feature of the Big Data. The portfolio data are not always complete, which can play a significant role in the prediction. The Veracity dimension refers to the biases, noise, and uncertainty in data. The credit card transactions dataset is a highly biased one, as a fraudulent transaction is a rare occurrence. Thus, resulting in the veracity or uncertainty in the data.

The details of the datasets, results generated after analyzing the datasets are presented in further sections.

4.4.1 Datasets Analyzed

We experimented with credit card fraud dataset i.e., ccFraud dataset [209]. The ccFraud dataset has a high volume of transactions. It has been analyzed with a parallel distributed processing environment of Apache Spark. This dataset is a snapshot at a particular instant of time for processing, as there is non-availability of credit card dataset having a real-time inflow of transactions in the public domain. The ccFraud dataset is a highly unbalanced dataset with only 5.96% of fraudulent transactions, rendering the veracity in the data.

The ccFraud dataset contains ten million samples with 9 features. In the proposed model, 7 features have been considered to train the PSOAANN model. The feature "custID" has been discarded since it contains unique values in all samples, which will disturb the generalization of patterns. The class variable "fraudRisk" is also not provided at the time of training a PSOAANN since the model is to be trained with only one class, i.e., negative samples.

4.4.2 Analysis of Results

The current literature does not have any experimental result with credit card data set in the Apache Spark or MapReduce distributed environment. So our result could not be compared with any other result to confront.

The dataset is having 94.04% of legitimate or genuine credit card transactions and only 5.96% of fraudulent transactions. Hence the dataset is highly unbalanced,

yielding to the complexity involved in the classification task. Without going for undersampling or oversampling of the biased data, we have experimented with one-class classification using legitimate transactions only and later tested the performance of the model by the fraudulent transactions.

The training of the PSOAANN model was completed with 30 runs where each run is of 20 iterations. The computational time for each iteration is about 51sec on an average. The model was dependent on the evolutionary algorithm, PSO, which uses a random seed. Every run is dependent on the random seed, thus producing varying results for different runs. Hence, in order to nullify the effect of randomness caused, we ran the model for 30 times. The execution time for each run is observed to be, on average, about 17m 05sec.

The AANN had three-layered architectures with six nodes in the hidden layer. The weights between input and hidden layer, as well as hidden and output layers are initialized using uniform distribution in the range of $[-5, 5)$. The MSE calculation resulting from AANN was minimized by PSO. The objective function was to minimize the MSE between the actual input and predicted output. PSO was used to minimize the MSE. The convergence plot for mean MSE value over 30 runs versus 20 iterations is depicted in Figure 4.4.

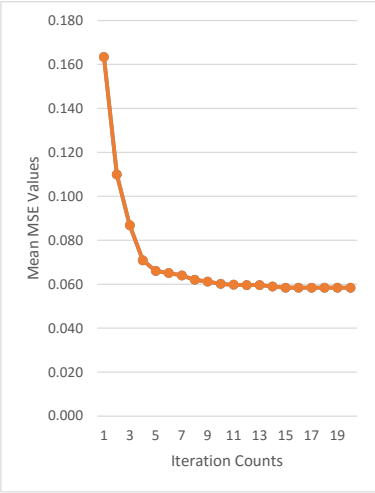

Figure 4.4: *Mean MSE convergence plot*

We had conducted several experiments with different PSO parameters and found that the inertia weight w = 0.9 gives the best result. We also varied the values of c_1 and c_2 so that their summation equals 4 and with several experiments, it is found that $c_1 = c_2 = 2$ yields a better result. Incidentally, many research papers e.g., Bansal et al. [427], Maheshkumar et al. [428] support the values of c_1 and c_2 to be 2 for producing a superior result. Bansal et al. [427] describe that for minimizing error, the best strategy for inertia weight is to take a constant value for it. In our case, it is found to be the value 0.9, which produces a better result.

The classification rate of the above runs was in the range of 85% to 95%. The statistical inferences are as follows. The minimum value of Classification Rate (CR): 85.89%, maximum value of CR: 95.312%, arithmetic mean of CR: 89.16%, median of CR: 88.645%, mode of CR: 87.16%, and standard deviation: 2.67%.

The median of CR being 88.64% indicates that for 50% of the runs, CR lies above 88.64%. The arithmetic mean lies on the right side of the median at 89.16%, indicating that the data is right-skewed. The mode of CR at 87.15% indicates more chances to get 87% of the classification rate. The standard deviation of 2.67% clearly shows that the observations for CR are closely placed, and hence the algorithm yielded a stable result.

The proposed parallel approach uses Apache Spark for the parallelization of datasets in a distributed clustered computation. In addition to that, we have implemented the parallelization of the algorithm for the AANN. The implementation involved parallelization of computations in the AANN in training as well as the test phase of the model. Each instruction in AANN is carried out in a parallel manner over multiple worker nodes in the Spark cluster. The weight update scheme in AANN using PSO is also parallelized in all the worker nodes. The whole model could not be constructed in a single system, as the transaction volume is large enough to fit in the memory of a single system. Hence, the speedup and efficiency measure could not be computed due to the lack of serial computation of the algorithm on a single machine.

4.5 Conclusions

In this chapter, a hybrid architecture has been employed involving particle swarm optimization (PSO) and auto-associative neural network (AANN) to get a solution for one-class classification (OCC) in a Big Data paradigm in a SPARK cluster. In this work, we parallelized AANN and PSO with the achievement of an average of 89% true classification of the credit card fraud transactions. By the inclusion of portfolio data, the variety and veracity can be incorporated, which is identified to be a problem area.

Chapter 5

Parallel Distributed Incremental One-pass Clustering Method

This chapter presents our proposed parallel distributed incremental clustering approach. The next section of the chapter presents an introduction to the current work. Section 5.2 presents the proposed approach. The experimental setup is described in Section 5.3. Results and discussion are presented in Section 5.4. The chapter is concluded in Section 5.5.

5.1 Introduction

Clustering is the process of assembling objects having similarity in some aspect in the same group. There are several types of clustering methods viz. Centroid-based (partition based) clustering methods [374], Connectivity-based clustering (Hierarchical clustering) Methods, Agglomerative Approach [429], Divisive Approach [430], Density-based Clustering Method [431], Grid-based Clustering Method [432] and Model-based method [433].

The algorithms belonging to these methods are iterative in nature, i.e., making several passes over the data samples, thus rendering it unsuitable for Big Data

Analytics (BDA).

In centroid-based clustering methods, clusters are delineated by a central vector, called the centroid of the cluster, which may not inevitably be an element of the dataset. The distance of the data points are calculated from the centroids of all the clusters, and the data point belongs to the cluster with the minimum distance. These are iterative methods, which constitute several passes over the data.

The density-based clustering method searches for the dense regions in the data space. It differentiates different density regions, and the data points within a given locality are considered to belong to the same cluster.

The grid-based clustering method divides the data space into a specific number of cells to construct a grid-like structure. Then, selecting dense regions from the cells in the grid structure results in the clusters.

All the clustering approaches mentioned above have shortcomings to handle large-sized data with faster execution. Either they have the curse of dimensionality, or they are multi-pass algorithms, which render them inefficient for large datasets. Being online, the Evolving Clustering Method (ECM) can handle a stream of data while the clustering process goes on. Thus, the clustering process evolves with the incoming data points. Being one-pass and evolving in its approach, the ECM is suitable for large-sized dataset [375].

The afore-mentioned feature has provided a firm motivation for implementing a parallel version of ECM to handle massive datasets efficiently.

5.2 Proposed Approach

The proposed Parallel Evolving Clustering Method (PECM) is implemented in Apache Spark computational framework with distributed data storage in HDFS. Thus, the PECM algorithm executes in a parallel, distributed manner.

5.2.1 The Evolving Clustering Method (ECM) Algorithm

The algorithm of ECM is a distance-based clustering technique, which uses D_{thr}, a clustering parameter, which is the threshold value for similarity index. A sample point belonging to a cluster which is the farthest from the cluster center should be

less than or equal to D_{thr}, the threshold value. D_{thr} influences the count of clusters that the algorithm yields. The ECM is presented below:

Step 1: Initialize the first cluster C_1 with the first data point from the input dataset and its position is considered as the cluster center Cc_1, for the cluster C_1 and the radius of the cluster C_1 is Ru_1, which is set to a value 0.

Step 2: If the analysis of entire samples from the dataset is completed then the clustering process comes to an end. Else, the present input instance, x_i, is considered and the normalized Euclidean distances *d(i, j)*, between this instance and the cluster centers Cc_j of all n already existing clusters,

$$d(i,j) = \left\| x_i - Cc_j \right\| \tag{5.1}$$

where j = 1, 2, 3, …, n, are evaluated. Here *d(a, b)* is the normalized Euclidean distance between the two q-dimensional vectors a and b and are defined as follows:

$$\|a - b\| = \left(\sum_{i=1}^{q} |a_i - b_i|^2 \right)^{1/2} \Big/ q^{1/2} \tag{5.2}$$

where $a, b \in R^q$.

Step 3: The distance *d(i, m)* between a sample x_i and a cluster center Cc_m where Cc_m is the center of a cluster C_m with radius Ru_m is defined as follows:

$$d(i,m) = \min_j d(i,j) = \min_j \left(\left\| x_i - Cc_j \right\| \right) \tag{5.3}$$

where, $j = 1, 2, \ldots, n$ and if $d(i,m) \leq Ru_m$.

Then the current sample x_i is a member of the cluster C_m. On this occasion, no new cluster is formed. Also, no existing cluster is modified. The algorithm then goes back to Step 2. Else,

Step 4: The extended distance *s(i, j)* between sample x_i and cluster center Cc_j is evaluated by adding the distance value *d(i, j)* and radius Ru_j of cluster C_j.

$$s(i,j) = d(i,j) + Ru_j \tag{5.4}$$

where $j = 1, 2, \ldots, n$ and then choosing the cluster C_a with the minimum value *s(i,*

a):

$$s(i,a) = d(i,a) + Ru_a = \min_j s(i,j) \tag{5.5}$$

where $j = 1,2,\ldots,n$.

Step 5: If $s(i,a) > 2D_{thr}$, the sample x_i cannot be a member of any existing clusters. A new cluster is formed as mentioned in Step 1. The algorithm then returns to Step 2. Else,

Step 6: If $s(i,a) \leq 2D_{thr}$, the cluster C_a is modified by relocating its center, Cc_a, and enlarging its radius value, Ru_a. The modified radius Ru_a^{new} is assigned the value equal to $s(i,a)/2$ and the new center Cc_a^{new} is positioned on the line joining input vector x_i and the former cluster center Cc_a so that the sample point x_i is at a distance of Ru_a^{new} from the newly formed center Cc_a^{new}. The algorithm then returns to Step 2.

5.2.2 The PECM Algorithm

The PECM algorithm is a parallel implementation of the ECM algorithm. The PECM algorithm follows below:

Required: The dataset D has to be uploaded to HDFS for distributed storage. The dataset is normalized using min-max normalization.

Step 1: The data is divided into a specified number of partitions in the distributed storage to make the parallel execution efficient.

Step 2: The ECM algorithm is executed in all the partitions of the dataset, i.e., the number of instances of ECM is the same as the number of partitions, running in parallel. The different partitions of the dataset produce clusters the same as the number of classes for a given D_{thr} value.

Step 3: The sub-clusters are collected at the master node from each partition of data and merged in a parallel manner. The merging process is executed over the worker nodes in a parallel manner with the partitioning of sub-clusters, i.e., the sub-clusters are divided into partitions, and each partition produces clusters the same as the number of classes. The merging process is carried out with a parallel merging threshold. These clusters are collected and merged in the master node, which we call the serial merging with a serial merging threshold, thereby producing the required number of clusters.

The workflow of PECM is depicted in Figure 5.1.

The map-reduce operation in Spark is executed with transformation and action operations. The PECM is executed in a parallel manner by mapping it to all the partitions of the dataset by a map operation e.g.

```
val clusterCentersRDD = partitionedData.mapPartitions(ECM)
```

where, the partitioned data is represented with *partitionedData* and *ECM* is the function passed to the *mapPartitions* function.

Then the reduce operation collects all the sub-clusters formed in each partition of the dataset. The reduce operation is carried out with collect action operation e.g.

```
val clusterCenters = clusterCentersRDD.collect()
```

Here, all the cluster centers are collected in *clusterCenters*.

According to Step 1, initially, the dataset is divided into a number of partitions across the worker nodes, i.e., the data is stored in the HDFS spanning over the worker nodes. The partition number is chosen to be high enough using a trial and error method to reduce the latency. The count of ECM instances running in parallel is the same as the number of partitions of the dataset. In Step 2, depending on the D_{thr} value, a different number of sub-clusters are formed in each partition, which is the output of each instance of ECM.

Then, the sub-clusters are collected in the master node. The high number of partitions entails a high number of sub-clusters resulting from a high number of ECM instances running in parallel. Therefore, the situation warrants a parallel merging phase.

In the parallel merging phase, firstly, the collected sub-cluster centers are partitioned at the master node and are distributed over the worker nodes. Secondly, a merging process is executed on each partition of sub-cluster centers as follows. Two sub-clusters are merged if the distance between their centers is within a pre-specified merging threshold value. This process results in a new group of sub-cluster centers which are collected at the master node.

These new sub-cluster centers are then merged at the master node (which is called the serial merging phase) so that the final desired number of clusters is achieved.

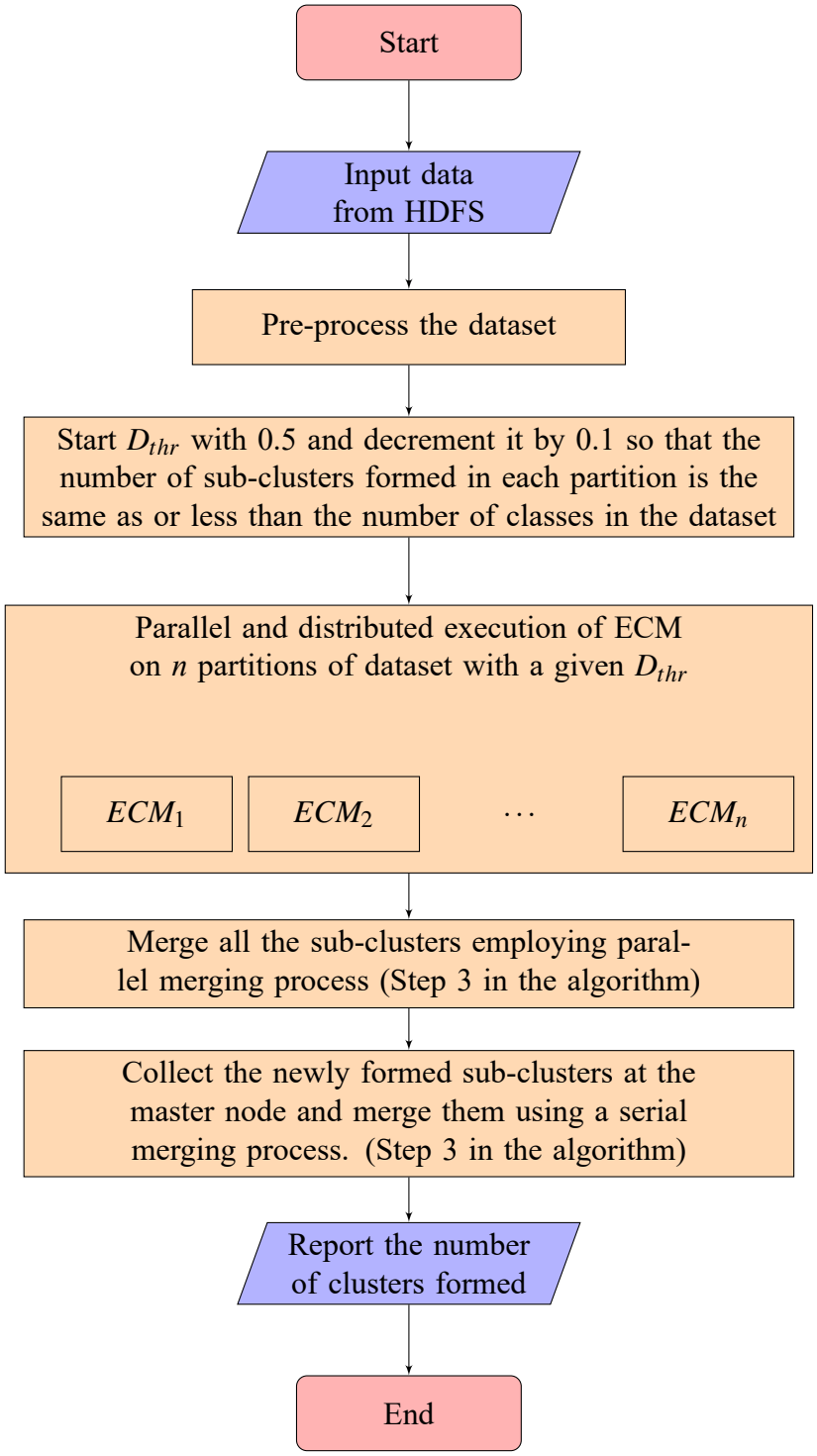

Figure 5.1: Execution flowchart for PECM

In summary, the proposed PECM comprises (i) running as many instances of ECM in parallel as the number of partitions effected on the data (ii) a merging process involving a parallel phase at worker nodes and a serial phase at the master node in tandem. We observed that, without the invocation of the merging process, the parallel and distributed version of ECM could not be developed. In other words, the merging process helps PECM to reduce the huge number of sub-clusters formed of the data at various worker nodes. We believe that the proposed novel merging process is an essential contribution to the literature of parallel incremental clustering techniques.

It may be noted that the total count of the clusters produced by the proposed PECM algorithm is the same as the number of class variables. If the class variable is not present in the dataset, then we can start with a random D_{thr} value and find the Dunn Index [434] of the clusters formed in each partition. Then, the sub-clusters from each partition with the highest Dunn Index value obtained by varying the D_{thr} value will be collected at the master node for the serial merging process. Another way to resolve this problem is to visualize the data either in a multi-dimensional space using the method of parallel coordinates [435] or a 3-dimensional space using principal component analysis plot [436]. Finally, while solving problems related to the business domain, one can take the help of a domain expert to determine the number of clusters.

5.3 Experimental Setup

The experimental setup consists of a standalone Spark cluster using the HDFS as the storage system and Apache Zeppelin 0.7.1 as an editor. The Spark cluster comprises a master node running the *driver program* and ten *worker nodes* including one worker node running on the *master node*. All the ten nodes have the same configuration, i.e., Intel® Core™ i7-6700 CPU @ 3.40GHz with 8 logical cores with 32GB RAM and Ubuntu 14.04 LTS Operating System (see Table 5.1). We allocated 28 GB of memory in nine worker nodes. In each of the worker nodes, four executors with 7GB memory and two cores are configured. The worker in the master node is configured with three executors with 6GB memory and two cores each. The driver process was allocated 10 GB of memory (see Table 5.2).

The PECM was executed in Spark 2.1.0 cluster with Hadoop 2.7.3 as distributed storage using Scala 2.11.8 programming language (see Table 5.2). The best execution time was achieved by tweaking the memory used by the executors in each worker node with the optimal number of data partitions.

Table 5.1: System description

CPU	Intel® Core™ i7-6700 CPU @ 3.40GHz with 8 logical cores
Memory	32GB
Operating System	Ubuntu 14.04 LTS

Table 5.2: Parallel distributed computational environment configuration

Configuration details	**Node Type**	
	Master	**Slaves**
Driver memory	10GB	-
Number of workers	3	4
Worker memory	6GB	7GB
Number of executors	3	4
Executor memory	6GB	7GB
Number of cores/executors	2	2
Total executor memory	18GB	28GB
Total memory utilized(out of 32GB)	28GB	28GB
Computational framework	Apache Spark 2.1.0	
Distributed storage system	HDFS (Hadoop 2.7.3)	
Editor for code development	Apache Zeppelin 0.7.1	
Language used for coding	Scala 2.11.8	

5.4 Results and Discussion

The proposed PECM is implemented in a parallel distributed computational environment. The ccFraud dataset and Higgs dataset have been analyzed to verify the performance of the proposed approach. The details of the datasets, results generated after analyzing the datasets are presented in further sections.

5.4.1 Datasets Analyzed

The first dataset that has been analyzed is the credit card fraud dataset i.e., ccFraud dataset [209]. The credit card transaction count is growing day-by-day with leaps and bounds. There is a large volume of data generated through credit card transactions. The data is also generated at a high velocity. These conditions make the problem amenable to the application of BDA.

This dataset is a snapshot at a particular instant of time for processing, as there is non-availability of credit card dataset having a real-time inflow of transactions in the public domain.

The details of the ccFraud dataset can be found at Appendix C.2. In the proposed model, seven features have been considered for the clustering process of PECM. We discarded the "custID" since it contains unique values in all samples, which will disturb the similarity calculation of the patterns. The class variable "fraudRisk", is also not included in the process of clustering to perform unsupervised learning.

The other dataset, we analyzed is the Physics dataset generated from particle detectors in the accelerator, the HIGGS dataset [437]. The data is produced using Monte Carlo simulations. The details of the dataset can be found at Appendix C.4. We have utilized 7 high-level features for clustering purpose those are extracted from 21 low-level features for the PECM to cluster it into two clusters. The resulting dataset is 1.4GB in size.

5.4.2 Analysis of Results

The ccFraud dataset contains two values for the class variable viz., legitimate transaction, and fraudulent transaction. The PECM produced the clusters for each partition of the dataset present in the distributed storage system. These sub-clusters are then merged in a parallel as well as a serial manner to produce the final clusters. In the parallel merging process, the sub-clusters produced by PECM were merged using the Spark cluster with a parallel merging threshold. Then the result of the parallel merging process was submitted to the serial merging process with its merging threshold. The selection of merging thresholds was automated to produce the clusters, which are the same as the number of class values present in the dataset.

The ccFraud and Higgs dataset are binary classification datasets, hence contains two class values. Table 5.3 and Table 5.4 presents the different combinations of parallel and serial merging thresholds those produced the desired clusters for ccFraud, and Higgs dataset, respectively. It may be noted that we can not present classification accuracy for the dataset because PECM is a clustering algorithm, which cannot output the accuracy of prediction.

Table 5.3: Parallel and serial merging thresholds for ccFraud dataset

Merging Threshold for parallel merging	Merging Threshold for serial merging	No. of Clusters formed
0.1	0.08	2
0.15	0.13	2
0.2	0.06	2
0.25	0.17	2
0.35	0.05	2

Table 5.4: Parallel and serial merging thresholds for Higgs dataset

Merging Threshold for parallel merging	Merging Threshold for serial merging	No. of Clusters formed
0.15	0.05	2
0.2	0.18	2
0.25	0.05	2
0.3	0.08	2
0.4	0.06	2

The ccFraud dataset is having 94.04% of legitimate credit card transactions and only 5.96% of fraudulent transactions. Hence the dataset is highly unbalanced, yielding to the complexity involved in the clustering task.

The Higgs dataset is having 53% of positive samples and 47% of negative samples. Hence, the dataset is a balanced one leading to the simplicity of the clustering process.

The PECM has a dramatic improvement over the serial ECM in terms of execution time. The ECM completed the execution of the ccFraud dataset in 74s, whereas PECM completed with 28s. The ECM completed execution of Higgs dataset 77s whereas PECM completed with 10s (See Figure 5.2).

Though the Higgs dataset is quite larger than the ccFraud dataset, the execution of the Higgs dataset is much faster than the ccFraud dataset. This discrepancy is attributed to data distribution. The Higgs dataset is balanced, whereas the ccFraud dataset is a highly unbalanced one.

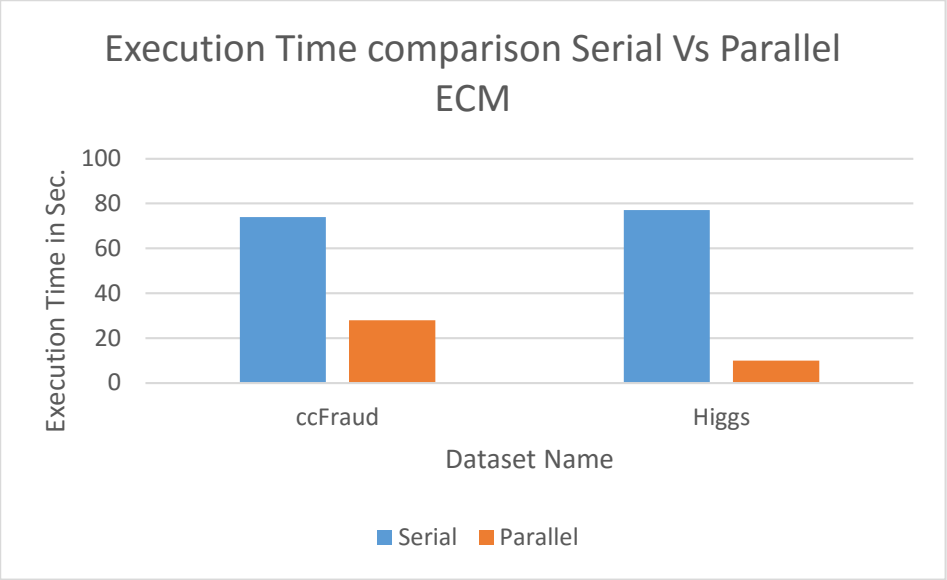

Figure 5.2: *Execution time comparison ECM Vs. PECM for ccFraud and Higgs dataset*

5.5 Conclusions

The PECM is implemented with the Apache Spark with distributed storage of data points using HDFS. The PECM has achieved clustering of the large-sized dataset with a single go and thus is faster than any other clustering method. The PECM is implemented with the Scala programming language. The ccFraud and Higgs datasets are analyzed.

An essential contribution of the current work is the simple and innovative merging of sub-clusters involving both parallel and serial phases that is central to the distributed and parallel implementation of the ECM.

The performance of PECM is found to be 2.6x faster in the ccFraud dataset and 7.7x faster in the case of the Higgs dataset.

Chapter 6

Parallel distributed one-pass regression model

This chapter presents our proposed parallel distributed one-pass regression model. The next section of the chapter presents an introduction to the current work. Section 6.2 presents the proposed approach. The experimental setup is described in Section 6.4. Results and discussion are presented in Section 6.5. The chapter is concluded in Section 6.6.

6.1 Introduction

Regression is one of the most fundamental tasks of data mining, having a wide variety of applications in many domains. Numerous techniques from a diverse family of techniques viz., statistics and machine learning (subsuming decision tree regression, support vector regression, and feed-forward neural networks) have been proposed for this task. Under neural networks category, various architectures were proposed for regression, viz., MLP, RBFN [438], WNN [439], GRNN [376], and GMDH [440], Counter Propagation Neural Network (CPNN).

MLP has been used for regression in several research works, viz., Kusakunniran et al. [441] presented a gait recognition system based on motion regression using MLP, Agirre-Basurko et al. [442] suggested MLP to forecast gas levels, Gaudart et al. [443] performed a regression for epidemiological data employing MLP.

RBFN has been used for regression, viz., Mignon and Jurie [444] proposed a face reconstruction algorithm based on RBF-regression in Eigenspace, Hannan et al. [445] presented heart disease diagnosis employing RBF, Taki et al. [446] employed RBF for energy consumption prediction for wheat production.

WNN is used for regression models for forecasting reservoir inflow [447], Vinaykumar et al. [448] implemented WNN for the estimation of the cost of software development, Chauhan et al. [449] employed DE trained WNN for bankruptcy prediction in banks, Rajkiran and Ravi [450] employed WNN for predicting the reliability of software.

So also GMDH is employed for regression, viz., Astakhov and Galitsky [451] presented tool life prediction in gundrilling, Elattar et al. [452] employed GMDH for short-term load forecasting, Srinivasan [453] employed GMDH for energy demand prediction, Ravisankar and Ravi [454] implemented GMDH for bankruptcy prediction in banks, Mohanty et al. [381] employed GMDH for prediction of software reliability, Reddy and Ravi [455] proposed kernel GMDH for regression.

Ravisankar and Ravi [454] employed CPNN for financial distress prediction in banks. Chandan and Ravi [456] implemented CPNN for data imputation. CPNN can be applied to pattern recognition, function approximation, statistical analysis, and data compression [457].

Multi-Layer Perceptron (MLP) have vast variations of architectures and have been implemented in different application domains, viz., wireless networks [458], robot manipulator control [459], wind energy systems [460], cancer prediction in healthcare [461], crop production [462], business [463].

Among these neural network architectures, GRNN is preferable for regression, as it uses a single pass for learning and produces reasonably good results. Further, GRNN does exploit non-parametric estimation, which often produces good results. While single-pass learning is a desirable property in analyzing big datasets, GRNN is not scalable due to its vast memory requirement. Hence, this chapter focusses on scaling up GRNN in the Apache Spark environment.

It is worthwhile to mention that the primary objective of this chapter is to present a distributed and parallel version of the traditional GRNN to handle big datasets. Therefore, it is not compared with the parallel versions of any of its competitors.

In this chapter, we have proposed a parallel distributed hybrid version of GRNN for prediction in the Big Data paradigm (GRNN++). Further, we have extended GRNN++ reported in Kamaruddin and Ravi [117] and is henceforth will be termed as EGRNN++. The GRNN++ has been extended to EGRNN++ with the following modifications:

- GRNN++ implemented GRNN with K-Means||, whereas the EGRNN++ has been extended with the implementation of another parallel clustering method of Bisecting K-Means for prediction of data in a Big Data paradigm. The embedding of an efficient parallel clustering approach in the traditional GRNN architecture is an innovative contribution to the work.
- The GRNN++ was implemented with the Gaussian activation function, which has been extended with rigorous experimentation with two more additional activation functions viz., Logistic, and Cauchy in EGRNN++.
- The GRNN++ had analyzed one gas mixture dataset, whereas the EGRNN++ includes two more datasets, viz., another additional gas mixture dataset which is of numeric type and Amazon movie reviews data which is of the text data type.
- The EGRNN++ analyzes the utility of the model for both structured and unstructured format of data, where the unstructured data comes from the social media, which adds up the complexity due to its voluminous size.

Finally, the EGRNN++ has been implemented with two clustering methods, each of which implementing three activation functions and analyzing three sets of datasets.

6.2 Proposed Approach

GRNN is not able to handle large-sized datasets without efficient clustering. It falters with memory issues and increased execution time. These limitations attribute to the incapability of the standard clustering approach to handle large-sized dataset. In the proposed method, EGRNN++, these drawbacks are addressed.

The training data is clustered with K-Means|| or Bisecting K-Means||. Thus, the cluster centers represent the data samples belonging to the cluster most effectively. Then the Dunn-like Index (a modified Dunn Index), a cluster validity index, is used for measuring the cluster validity.

The Dunn Index, proposed by Dunn [434], presents the approach for measuring the cluster quality so that the best clusters were selected based on the high compactness of clusters and high separation among them, i.e., compact and well-separated clusters. The compactness with good separation is found by the ratio of minimum intra-cluster distance to the maximum inter-cluster distance. The Dunn Index can be presented as:

$$Dunn\ Index(D) = \min_{1 \leqslant i \leqslant M} \left\{ \min_{i+1 \leqslant j \leqslant M} \left(\frac{dist\left(c_i, c_j\right)}{\max_{1 \leqslant l \leqslant M} \left(diam\left(c_l\right)\right)} \right) \right\} \tag{6.1}$$

where, M is the total number of clusters under consideration, $dist(c_i, c_j)$ is the inter-cluster distance and $diam(c_l)$ is the maximum intra-cluster distance or the largest diameter among all the clusters.

As the Dunn Index performs a calculation of maximum intra-cluster distance, i.e., the diameter of a cluster under consideration, it is not suitable for large-scale dataset as it will involve large computational overhead. Also, it is affected by noisy data and outliers. So, we have employed a Dunn-like index [464] for cluster validity measurement. In this approach, the diameter, i.e., the denominator of Equation (6.1) is calculated by finding the radius using the distance from each sample present in the cluster to its cluster center, i.e., $\bar{c_l}$, where $x \in c_l$ cluster and $\bar{c_l}$ is its center. The $|c_l|$ is the count of samples present in c_l cluster. This Dunn-like index is not affected by noise or outlier present in the data. The Dunn-like index can be presented as:

$$Dunn-like\ Index(DL) = \min_{1 \leqslant i \leqslant M} \left\{ \min_{i+1 \leqslant j \leqslant M} \left(\frac{dist\left(c_i, c_j\right)}{\max_{1 \leqslant l \leqslant M} \left(2 * \frac{1}{|c_l|} \Sigma_{x \in c_l} d\left(x, \bar{c_l}\right)\right)} \right) \right\} \tag{6.2}$$

The cluster centers form the clusters generated by the clustering algorithm and validated by Dunn-like Index, now represent the pattern nodes of EGRNN++. Then, each test sample is passed on to the neurons present in the Pattern layer, where one of the Gaussian, Logistic, or Cauchy activation functions results in higher value if the neuron has a strong similarity to the test sample. Now, the summation layer represents the numerator, denominator, and the output layer presents the prediction similar to GRNN. The training algorithm for EGRNN++ with the Gaussian activation function is presented below. The algorithms for the other variants of EGRNN++, i.e., with Logistic or Cauchy activation function, can be presented by replacing the Gaussian activation function with Logistic or Cauchy activation function. The different activation functions are discussed below.

Algorithm 2: Training Algorithm for EGRNN++

Input: σ value iterated from 0.001 to 1.0 with an increment of 0.001
Output: Average MSE in 10-FCV setup
Data: data distributed in partitions

1 Perform data normalization with the range [0,1].
2 Divide the data to perform 10-Fold cross-validation.
3 Take the training set and cluster them with either K-Means|| or Bisecting K-Means|| with the k = 2 to 10.
4 Compute Dunn-like Index by varying the value of k and find the k, which produces optimal clustering.
5 Consider the cluster centers represented by the optimal k as the neurons in the pattern layer of EGRNN++ and pass the training or test data, as the case may be, to the pattern layer nodes with a Gaussian activation function.
6 The product of A_i, i.e., the sum of the output values of the samples present in the i^{th} cluster and the result of activation function for i^{th} neuron in the pattern layer F_i† i.e., $\sum_{i=1}^{n} A_i * F_i$ forms the numerator term in the summation layer (say **Num**).
7 The product of B_i, i.e., count of the samples present in the i^{th} cluster and the result of activation function for i^{th} neuron in the pattern layer F_i i.e., $\sum_{i=1}^{n} B_i * F_i$ forms the denominator term in the summation layer (say **Denom**).
8 In the output layer, the prediction is computed as $\hat{Y}(x) = \frac{Num}{Denom}$.
9 The accuracy of prediction is measured by computing MSE.
10 Steps 3 to 9 are repeated for the 10 folds of data.
11 Average MSE for 10-FCV is computed and reported.

† F_i is the activation function for i^{th} neuron and it can be PDF of Gaussian, Logistic or Cauchy.

The probability distribution function (PDF) for Gaussian distribution can be represented as:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)} \tag{6.3}$$

where, $x \in (-\infty, +\infty)$, μ is the mean of the distribution and $\sigma^2 \geq 0$ is the variance. The EGRNN++ prediction with Gaussian activation function can be presented as:

$$\hat{Y}(X) = \frac{\sum_{i=1}^{n} A_i * e^{\left(\frac{-(x_i-\mu)^2}{2\sigma^2}\right)}}{\sum_{i=1}^{n} B_i * e^{\left(\frac{-(x_i-\mu)^2}{2\sigma^2}\right)}} \tag{6.4}$$

where, $(x_i - \mu)^2$ represents the squared distance between the test sample and the cluster center, A_i represents the sum of the target values of the samples present in the i^{th} cluster, B_i represents the count of the samples present in the i^{th} cluster. These symbols carry their meanings in the prediction equation represented with Logistic (5.6) and Cauchy (5.8) activation function. The constant part $\frac{1}{\sqrt{2\pi\sigma^2}}$ in Equation (6.3) is cancelled out from the numerator and denominator of the prediction Equation (6.4).

The PDF for Logistic distribution is presented as:

$$f(x) = \frac{e^{\left(\frac{-(x-\mu)}{\sigma}\right)}}{\sigma\left(1 + e^{\left(\frac{-(x-\mu)}{\sigma}\right)}\right)^2} \tag{6.5}$$

where, $x \in (-\infty, +\infty)$, $\sigma \geq 0$ is standard deviation and μ is the mean of the distribution. The EGRNN++ prediction with Logistic activation function can be presented

as:

$$\hat{Y}(X) = \frac{\sum_{i=1}^{n} A_i * \frac{e^{\left(\frac{-(x_i-\mu)}{\sigma}\right)}}{\sigma\left(1+e^{\left(\frac{-(x_i-\mu)}{\sigma}\right)}\right)^2}}{\sum_{i=1}^{n} B_i * \frac{e^{\left(\frac{-(x_i-\mu)}{\sigma}\right)}}{\sigma\left(1+e^{\left(\frac{-(x_i-\mu)}{\sigma}\right)}\right)^2}} \tag{6.6}$$

where the σ present in the denominator of the PDF Equation (6.5) is canceled out from the numerator and denominator of the prediction Equation (6.6).

The PDF for Cauchy distribution is presented as:

$$f(x) = \frac{1}{\pi\gamma\left[1+\left(\frac{x-\mu}{\gamma}\right)^2\right]} \tag{6.7}$$

where, $x \in (-\infty, +\infty)$, $\gamma > 0$ is a scale variable and μ is the median of the distribution. The EGRNN++ prediction with Cauchy activation function can be presented as:

$$\hat{Y}(X) = \frac{\sum_{i=1}^{n} A_i * \frac{1}{2\pi\left[1+\left(\frac{x_i-\mu}{2}\right)^2\right]}}{\sum_{i=1}^{n} B_i * \frac{1}{2\pi\left[1+\left(\frac{x_i-\mu}{2}\right)^2\right]}} \tag{6.8}$$

where, γ is considered to be 2.

The architecture of EGRNN++ is the same as the traditional GRNN except for the presence of the clusters (obtained through K-Mean‖ or Bisecting K-Means‖) occupying the pattern layer in place of the training patterns themselves (see Figure B.1). The 3^{rd} and 4^{th} layer of EGRNN++ are identical to that of GRNN. The architecture of EGRNN++ is depicted in Figure 6.1.

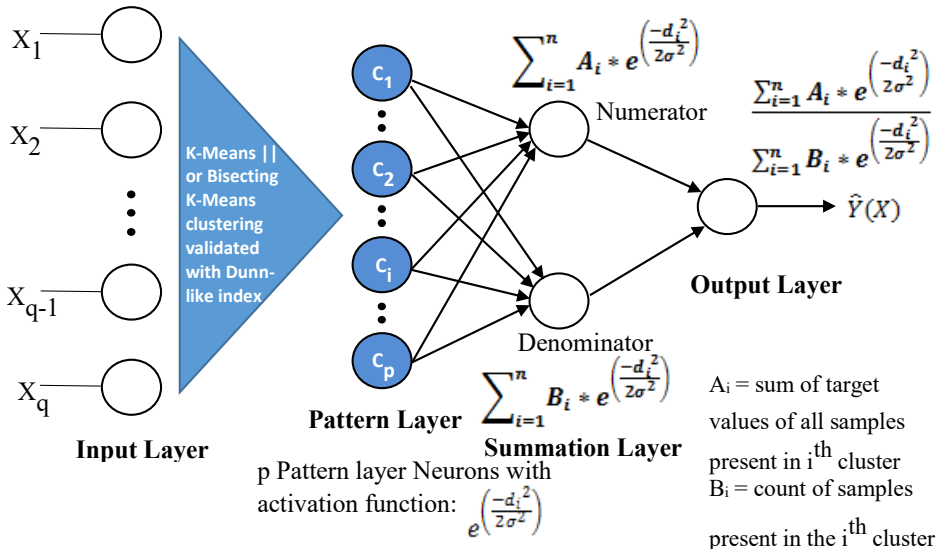

Figure 6.1: *EGRNN++: Architecture*

6.3 Social Media Analytics

Social media analytics is the procedure of collecting data from social media websites or social networks viz., Facebook, Instagram, LinkedIn, and Twitter, and analyzing that data using social media analytics tools to make business decisions. Social media analytics also defined as "the art and science of extracting valuable hidden insights from vast amounts of semi-structured and unstructured social media data to enable informed and insightful decision making" [465]. The utilization of social media generates an enormous amount of data on a daily basis, including customer reviews, preferences, and sentiments towards brands, products, and services that are valuable to business [466].

The presence of social media has changed the way customers interact with the products, services, businesses, and brands. They have started to rely on the information on social media to support their decision-making [467]. Reviews, preferences, and sentiments towards products or services may be either positive or negative and may affect attitudes, perceptions, and purchase decisions of customers [468].

The tremendous growth in the use of social media has led to an increasing accumulation of data, which is known as Social Media Big Data. Social media platforms

offer a variety of data formats, which includes images, audios, videos, text data, and geolocations. Generally, this data can be categorized into unstructured data and structured data [469]. In social networks, the text content is the unstructured data, while the follower/followed by relationship is the structured data.

Whenever, there is a post on social media, social data will be generated. Data created by consumers - comments, sentiment, likes, shares, retweets, etc. These social data help in leading the social media marketing campaigns to huge success.

Following are the few reasons why we need social media analytics:

- **Understand the audience -** One should know when the targeted audience is online, it will be futile to post if they are snoozing. The post should be in multiple languages based on the geographical location of the audience. Figuring out the interest of the audience will help to deliver more relevant content to them. Knowing the details of the audience will help with customer segmentation, which may help for targeted campaigns.

- **Selection of the social platform -** One should find out which social platform the target audience is online. The social platform that is suitable for the product should be chosen, e.g., a visually attractive product or product part that can be campaigned on an image-based network like Instagram. The performance of the campaign can be figured out using the platforms' inbuilt analytics tools.

- **Competitive benchmarking -** It allows to analyze competitor's data to find what leads to success and what doesn't. One can follow the path of successes, and the risks can be avoided.

- **Creation of better content -** The content of a campaign can be identified, which has resulted in the campaign success. The content can be the inspiration for future campaigns. One should figure out which form of content is more approachable to the audience - text, links, images, videos.

- **Brand reputation** = The organization should get informed about how the brand is being accepted by the consumers. This information is available across social networks, blogs, and forums. A good social media analytics

platform can provide the media monitoring data along with social media analytics data.

- **Protection of brand assets -** Image recognition can be utilized to figure how brand assets such as logos are being used online. It will help to identify potential misuse/trademark infringement of the logo, tagline, etc.

- **Tracking of Social campaigns -** Once the social campaign is launched, it can be monitored to check whether it is working or not. Is there a pocket to improve the campaign? Is it performing as per expectation or not? Should it be stopped? These questions should be addressed. The sentiment analysis can be used to figure out how consumers feel about the campaigns, brand, and products.

The most prevalent use of social media analytics is to extract consumer sentiments to support and enhance marketing and post sale customer service activities. Sentiment analysis is about extracting consumer attitudes, emotions, experiences. In general, sentiment analysis aims to figure out the attitude presented by the text writer or speaker in relation to the topic or the overall contextual polarity of a document [470]. Pang and Lee [471] provided a detailed review on the fundamentals and the methods of sentiment classification and extraction.

Sentiment analysis can be divided into the following specific subtasks:

- **Sentiment context —** To mine opinion or sentiments, one needs to know the 'context' of the text, which can vary considerably from specialist review portals/feeds to general forums where opinions can cover a spectrum of topics [472].

- **Sentiment level —** Text analytics can be conducted at the document, sentence, or attribute level.

- **Sentiment subjectivity —** Deciding whether a given text expresses an opinion or is factual.

- **Sentiment orientation/polarity —** It indicates whether an opinion in a text is positive, neutral, or negative.

- **Sentiment strength -** It presents the 'strength' of an opinion in a text: weak, mild, or strong.

The insights obtained through SMA can lead to multi-facet brand operations. Following are some of the examples:

- **Increase Customer Acquisition -** By delivering the content the customer wants, an organization can increase its customer base.
- **Protect Brand Health -** Using social media, any organization can produce contents reflecting the positive sides of its products and thus can enhance the brand health.
- **Lower Customer Care Costs -** The consumer wishes and complaints can be addressed using the social media platform. Thus, customer care costs can be cut down while performing efficient and effective customer management.
- **Maximize Product Launches -** By providing the content and product to the right influencers in the social network, the product launch can be maximized.
- **Boost Campaign Performance -** By smartly using influencers in a social network, the campaign performance can be increased.
- **Improve Crisis Management -** Social monitoring can be used to understand public misperception, and it can be utilized to respond to potential crises.

6.4 Experimental Setup

The system configuration for carrying out experimental work consists of standalone Spark cluster 2.2.0 as the computational framework on top of HDFS, which is the distributed storage system. The program development environment used is Apache Zeppelin 0.7.3, with Scala 2.11.8 as the coding environment. The Spark cluster consists of a system as a master node and seven systems as worker nodes. The driver program resides in the master node, along with two instances of worker daemons. Each of the worker nodes executes four instances of worker daemons. All the seven systems have a similar configuration, (refer to Table 6.1). In a stand-alone cluster

system, each worker daemon has one executor. The details of resources allocated to worker daemon or executor, driver process, and programming environment details are presented in Table 6.2.

Table 6.1: System configuration description

Central Processing Unit	Intel® Core™ i7-6700 CPU @ 3.40GHz
No. of Cores	4 Physical cores or 8 logical cores
Primary Memory	32GB
OS	Ubuntu 16.04 LTS

Table 6.2: Details of Resource allocation in the Cluster and coding environment

Allocated resource details	**Node Type**	
	Master	**Worker**
Memory allocated for Driver process	14GB	-
Count of worker daemons	2	4
Memory allocated to worker daemon	7GB	7GB
Number of executors per node	2	4
Memory allocated to each executor	7GB	7GB
Cores allocated per executor	2	2
Memory allocated to all executors	14GB	28GB
Total memory utilized(out of 32GB)	28GB	28GB
Framework for computation	Apache Spark 2.2.0	
Distributed storage system	HDFS (Hadoop 2.7.3)	
Coding interface	Apache Zeppelin 0.7.3	
Programming language	Scala 2.11.8	

The GRNN++ was implemented with the above cluster configuration and programming environment. We found the best execution time by adjustment of the amount of memory allocation to the executors along with the optimum data partitions.

6.5 Results and Discussion

In order to demonstrate the proposed approach, we presented two variants of EGRNN++, viz. EGRNN$_1$++ and EGRNN$_2$++ with two activation functions, viz.

Logistic and Cauchy activation function. The proposed approaches are compared with the traditional Gaussian activation function. Both the variant of EGRNN++ were analyzed with six datasets. The details of the datasets, results generated after analyzing the datasets are presented in further sections.

6.5.1 Datasets Analyzed

The EGRNN++ has analyzed

(i) the data from an array of gas sensors involving different gas mixtures, and

(ii) Amazon movie review (AMR) dataset that includes movie reviews from different users.

The gas sensor dataset [422] was collected from UCI ML repository [423]. The dataset contains the readings gathered from sixteen sensors exposed to varying concentrations of gas mixtures. The dataset contains the sensor readings of mixtures of Ethylene with Methane, and Ethylene with CO gas in the Air. We have analyzed the dataset for the prediction of Ethylene, Methane, and CO concentration in Air. The two mixtures of gases present two datasets with 19 features. The details of the features for Ethylene-Methane is in Table C.2 and for Ethylene-CO is in Table C.1.The Ethylene-CO dataset contains 4.2 million samples and is of 643MB. The Ethylene-Methane dataset contains 4.17 million samples and is of 638 MB.

For our regression work, we have dropped the first feature being the time series values. Out of the second and third features, one is selected as the dependent variable. We have selected the features from the fourth to the last as independent variables for our experimentation.

The AMR dataset [473] available on the Stanford Network Analysis Platform (SNAP) data repository [474]. It contains 7,911,684 reviews on 253,059 movies provided by 889,176 users, which were collected from August 1997 to October 2012. The reviews include product and user information, ratings, and a plaintext review. The ratings are in five levels of the Likert scale. The detailed structure of each review is depicted in Table C.6. The dataset is containing social media reviews, and it is used here for social media analytics for predicting the sentiment score of the movie reviewers.

In the experimental work, we have selected the review score or ratings as our dependent variable. The plain text review underwent a preprocessing phase whereby the special characters, HTML tags, emojis, the short form of texts were removed. Then, the stop words were removed, and the text was tokenized. After the preprocessing of the text was over, the tokens were converted to hashing Document Term Matrix (DTM) with feature-length of 20 and 100. Then, its TF-IDF value was calculated. Thus, two datasets were generated one with 20 TF-IDF values and the other with 100 TF-IDF values. These TF-IDF values were considered as independent variables for the EGRNN++ model.

6.5.2 Analysis of Results

The research work conducted comprises the analysis of the datasets mentioned above with the proposed approach. The results of the carried out work cannot be compared with any other technique as we could not find any work carried out in the same domain and using such hybrid architectural techniques.

The experiment was carried out with ten-fold cross-validation (10-FCV) of the min-max normalized dataset. The K-Means|| or Bisecting K-Means|| was carried out with k value ranging from 2 to 10, and then the Dunn-like Index was taken into account for cluster validity for finding out the optimal cluster centers. Then, with the selected cluster centers, EGRNN++ was carried out with the sigma (σ) value ranging from 0.001 to 1.0 with an increment of 0.001.

The performance of EGRNN++ is measured as MSE. Three activation functions used in EGRNN++ viz., Gaussian, Logistic, and Cauchy function, which captures the features of a test data and correspondingly presents the prediction value. The traditional GRNN employs the Gaussian activation function. We have proposed the Logistic and Cauchy activation function to be implemented. The Table 6.3 and Table 6.4 lists the average MSE of 10-FCV for the two implementations of EGRNN++ i.e., $EGRNN_1$++ (K-Means|| embedded in GRNN) and $EGRNN_2$++ (Bisecting K-Means|| embedded in GRNN), respectively.

The results shown in bold in Table 6.3 and Table 6.4 are the better values of mean MSE when the results of $EGRNN_1$++ and $EGRNN_2$++ are compared against

Table 6.3: Mean MSE for 10-FCV of EGRNN$_1$++ (K-Means|| embedded in GRNN)

Dataset	Prediction of	Gaussian AF*	Logistic AF	Cauchy AF
Gas sensor dataset	Ethylene Conc. from Ethylene-CO dataset	**0.061347399**	**0.066123807**	**0.074654456**
	CO Conc. from Ethylene-CO dataset	0.07796598	0.082468302	0.08819724
	Ethylene Conc. from Ethylene-Methane dataset	0.066086447	0.067908875	0.07305021
	Methane Conc. from Ethylene-Methane dataset	0.07049286	**0.056855938**	**0.06295538**
Amazon movie review dataset	review ratings with TF-IDF values of 20 features	1.63009754	**1.630151108**	1.63021239
	review ratings with TF-IDF values of 100 features	**1.611014908**	**1.611124609**	**1.611218718**

[$\star$]AF $\Rightarrow$ Activation Function

Table 6.4: Mean MSE for 10-FCV of EGRNN$_2$++ (Bisecting K-Means|| embedded in GRNN)

Dataset	Prediction of	Gaussian AF*	Logistic AF	Cauchy AF
Gas sensor dataset	Ethylene Conc. from Ethylene-CO dataset	0.06134848	0.066124431	0.074654607
	CO Conc. from Ethylene-CO dataset	**0.077965799**	**0.082468171**	**0.088197179**
	Ethylene Conc. from Ethylene-Methane dataset	**0.065585107**	**0.067437855**	**0.072853175**
	Methane Conc. from Ethylene-Methane dataset	**0.054793796**	0.057044785	0.06306113
Amazon movie review dataset	review ratings with TF-IDF values of 20 features	**1.63008182**	1.630153355	**1.63021059**
	review ratings with TF-IDF values of 100 features	1.611014999	1.611124657	1.61121873

[$\star$]AF $\Rightarrow$ Activation Function

each other. As the results do not present a clear winner, we proceeded for a t-test and calculated the p-value. The results are presented in Table 6.5.

6.6 Conclusions

We propose GRNN++, which is the parallel version of GRNN with Gaussian AF, and later it has been extended to EGRNN++, where apart from Gaussian AF Logistic and Cauchy AF have been included and for unsupervised learning parallel Bisecting K-Means has been added. The two variants of parallel and distributed version of EGRNN++ viz., $EGRNN_1$++, $EGRNN_2$++ in Apache Spark have been presented, where HDFS takes care of the distributed data storage. These are implemented in the Scala programming language. Both variants of EGRNN++ could perform Big Data regression in a single pass and yielded good results under 10-fold cross-validation. The hallmark of the proposed architectures is the invocation of (i) the parallel distributed K-Means++ aka K-Means|| (in case of $EGRNN_1$++) and (ii) Bisecting K-Means|| (in case of $EGRNN_2$++) into the architecture of the traditional GRNN.

The experiment involved structured numeric data as well as unstructured textual data. Both models have performed equally well, which is the inference from the statistical significance test and hypothesis test. Here, the architecture presents its superiority and efficiency in handling both structured and unstructured data. In this real-world, a high volume of unstructured data is generated and with significant speed. The movie review dataset presents the unstructured voluminous social network data. The experiment presents the efficacy of the EGRNN++ to handle both structured and unstructured large-sized data.

The innovation of the proposed method resulted in a drastic reduction in the number of pattern nodes in EGRNN++. This feature overcame the major shortcoming of GRNN, which is the computational overhead.

Table 6.5: t-test value and p-value ($EGRNN_1$++ Vs. $EGRNN_2$++)

Dataset	Prediction of		Gaussian AF*			Logistic AF			Cauchy AF		
			MSE	t-test value	p-value	MSE	t-test value	p-value	MSE	t-test value	p-value
Gas sensor dataset	Ethylene Conc.[1]	M1†	0.061347	0.029	0.977	0.066123	0.024	0.98	0.0746544	0.006	0.995
		M2‡	0.061348			0.066124			0.0746546		
	CO Conc.[2]	M1	0.0779659	0.002	0.998	0.0824683	0.001	0.998	0.0881972	0.0006	0.999
		M2	0.0779657			0.0824681			0.0881971		
	Ethylene Conc.[3]	M1	0.066	2.686	0.015	0.0679	2.867	0.01	0.073	2.069	0.053
		M2	0.065			0.0674			0.072		
	Methane Conc.[4]	M1	0.07	2.566	0.019	0.056	2.285	0.034	0.062	1.204	0.243
		M2	0.054			0.057			0.063		
Amazon movie review dataset	TF-IDF20[5]	M1	1.63009	0.004	0.996	1.630151	0.0006	0.999	1.630212	0.0005	0.999
		M2	1.63008			1.630153			1.63021		
	TF-IDF100[6]	M1	1.6110149	6.96884 E-06	1.0	1.6111246	3.63346 E-06	1.0	1.61121871	9.47175E-07	1.0
		M2	1.61101499			1.61112465			1.61121873		

[a] $EGRNN_1$++ ⇒ K-Means‖ embedded in GRNN
[1] Ethylene Conc. from Ethylene-CO dataset
[3] Ethylene Conc. from Ethylene-Methane dataset
[5] review ratings with TF-IDF values of 20 features
[†] $EGRNN_1$++
[⋆]AF ⇒ Activation Function
[b] $EGRNN_2$++ ⇒ Bisecting K-Means‖ embedded in GRNN
[2] CO Conc. from Ethylene-CO dataset
[4] Methane Conc. from Ethylene-Methane dataset
[6] review ratings with TF-IDF values of 100 features
[‡] $EGRNN_2$++

Chapter 7

Parallel distributed one-pass classifier

This chapter elucidates our proposed parallel distributed one-pass classifier. The next section of the chapter presents an introduction to the current work. Section 7.2 presents the proposed approach. The experimental setup is described in Section 7.3. Results and discussion are presented in Section 7.4. The chapter is concluded in Section 7.5.

7.1 Introduction

Classification is the most fundamental task of data mining, having a wide variety of applications in many domains. Numerous techniques from a diverse family of techniques viz., statistics, and machine learning (subsuming decision tree, fuzzy classification, Support vector machine, and feed-forward neural networks) have been proposed for this task. The popular neural network architectures for classification are Multi-layer Perceptron (MLP), Radial Basis Function Network (RBFN) [438], Wavelet Neural Network (WNN) [439], Probabilistic Neural Network (PNN) [386] and Group Method of Data Handling (GMDH) [440].

RBFN has been used for classification in many applications, viz., Lin [475] has employed RBFN for facial expression classification, Radha, and Nallammal [476] have implemented face recognition using RBFN. WNN also has been employed for

classification tasks viz., Subasi et al. [477] employed WNN for electromyographic (EMG) signal classification, Geethanjali, and Priya [478] implemented WNN for fault detection and classification in transmission lines. Similarly, several works have been carried out in classification using GMDH viz., Baig et al. [479] implemented GMDH for intrusion detection, El-Alfy, and Abdel-Aal [480] employed GMDH for spam detection.

Among these architectures, PNN is preferable as it uses a single pass for learning and produces reasonably good results. The architecture of PNN in detail is presented at Appendix B.5. While single-pass learning is a desirable property in analyzing sets, PNN is not scalable due to its huge memory requirement. Hence, this paper focusses on scaling up PNN in the Apache Spark environment. It is worthwhile to mention that the primary objective of this article is to present a distributed and parallel version of the traditional PNN to handle big datasets.

7.2 Proposed Approach

PNN is not able to handle large-sized datasets without efficient clustering. It falters with memory issues and increased execution time. These limitations attribute to the incapability of the standard clustering approach to handle large-sized dataset. In the proposed method PNN++, this drawback has been addressed. The PNN++ is implemented with Apache Spark and HDFS. It has two components, viz.,

(i) clustering component implemented with K-Means|| clustering (Appendix B.2) or parallel bisecting K-Means (BK-Means||) (Appendix B.3), and

(ii) PNN (Appendix B.5) for classification.

The training data is clustered with K-Means|| or Bisecting K-Means||. Thus, the cluster centers represent the data samples belonging to the cluster most effectively. Then the Dunn-like Index [464] (a modified Dunn Index [434]), a cluster validity index, is used for measuring the cluster validity.

The cluster centers generated by the clustering algorithm and validated by Dunn-like Index, now represent the nodes in the pattern layer of PNN++. Then, each test

sample is passed on to the neurons present in the pattern layer with one of the Gaussian, Logistic, or Cauchy activation function that results in a higher value if the neuron has a strong similarity to the test sample. Now, the summation layer represents the Probability Density Functions (PDFs) of each class involved in the classification process, and the output layer presents the classifier's prediction as in the PNN (Appendix B.5). The training algorithm for PNN++ with the Gaussian activation function is presented below. The algorithms for the other variants of PNN++, i.e., with Logistic or Cauchy activation function, can be presented by replacing the Gaussian activation function with Logistic or Cauchy activation function.

The architecture of PNN++ is the same as the traditional PNN except for the presence of the cluster centers (obtained through K-Mean|| or Bisecting K-Means||) occupying the pattern layer in place of the training patterns themselves (see Figure B.2). The 3rd and 4th layers of PNN++ are identical to that of PNN. The architecture of PNN++ is depicted in Figure 7.1.

After the PNN++ model is trained in a parallel and distributed environment, testing the network also happens in a parallel distributed computational environment, and the overall Accuracy or AUC is computed at the master node.

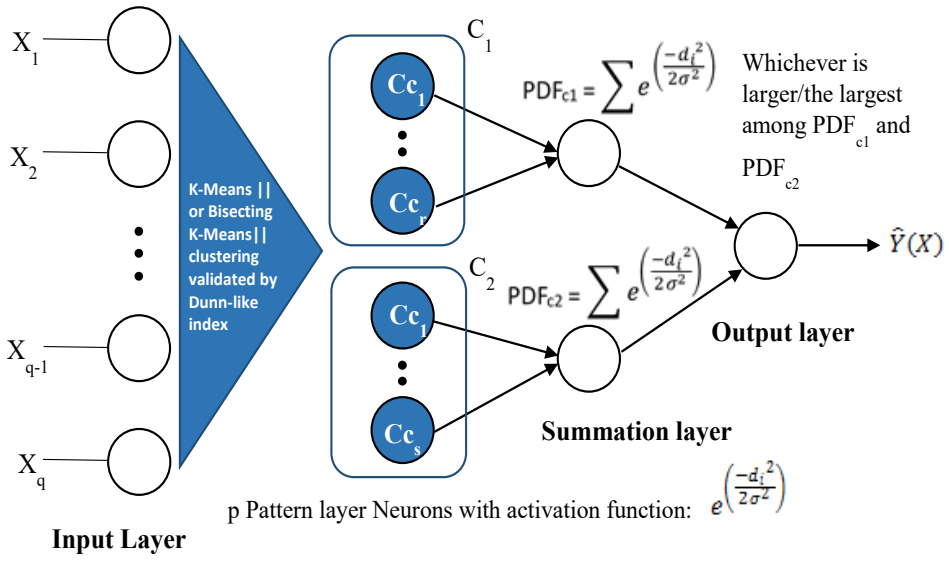

Figure 7.1: *Architecture of PNN++*

Algorithm 3: Training Algorithm for PNN++

Input: σ value iterated from 0.001 to 1.0 with an increment of 0.001
Output: Average AUC, Sensitivity and Specificity in 10-FCV setup
Data: data distributed in partitions

1 Perform data normalization with the range [0,1].
2 Divide the data to perform 10-Fold cross-validation.
3 Take the training set and cluster them with either K-Means|| or Bisecting K-Means|| with the k = 2 to 10.
4 Compute Dunn-like Index by varying the value of k and find the k, which produces optimal clustering.
5 Consider the cluster centers represented by the optimal k as the neurons in the pattern layer of PNN++ and pass the training or test data, as the case may be, to the pattern layer nodes with a Gaussian activation function.
6 The sum of the result of activation function F_i†, for i^{th} neuron (for class C_j)in the pattern layer over n samples of that class i.e., $\sum_{i=1}^{n} f_i$ forms the predicted probability value of the class C_j in the summation layer (say P_j), where j value ranges upto 2 for binary classification or to the number of classes for the multiclass classification.
7 In the output layer, the classifier's prediction is computed as $\hat{Y}(x)$, where $\hat{Y}(x)$ represents the class label of the larger probability for the binary classification or the largest one among all the probabilities, in case of multiclass classification.
8 The performance of the classifier is measured by computing AUC, Sensitivity, and Specificity.
9 Steps 3 to 8 are repeated for the 10 folds of data.
10 Average AUC, Sensitivity, and Specificity for 10-FCV are computed and reported.

† F_i is the activation function for i^{th} neuron and it can be PDF of Gaussian, Logistic or Cauchy.

7.3 Experimental Setup

The system configuration for carrying out experimental work consists of standalone Spark cluster 2.2.0 as the computational framework on top of HDFS, which is the distributed storage system. The program development environment used is Apache Zeppelin 0.7.3, with Scala 2.11.8 as the coding environment. The Spark cluster consists of a system as a master node and seven systems as worker nodes. The driver program resides in the master node, along with two instances of worker daemons.

Each of the worker nodes executes four instances of worker daemons. All the seven systems have a similar configuration (refer to Table 7.1). In a stand-alone cluster system, each worker daemon has one executor. The details of resources allocated to worker daemon or executor, driver process, and programming environment details are presented in Table 7.2.

Table 7.1: System configuration description

Central Processing Unit	Intel® Core™ i7-6700 CPU @ 3.40GHz
No. of Cores	4 Physical cores or 8 logical cores
Primary Memory	32GB
OS	Ubuntu 16.04 LTS

Table 7.2: Details of Resource allocation in the Cluster and coding environment

Allocated resource details	**Node Type**	
	Master	**Worker**
Memory allocated for Driver process	14GB	-
Count of worker daemons	2	4
Memory allocated to worker daemon	7GB	7GB
Number of executors per node	2	4
Memory allocated to each executor	7GB	7GB
Cores allocated per executor	2	2
Memory allocated to all executors	14GB	28GB
Total memory utilized(out of 32GB)	28GB	28GB
Framework for computation	Apache Spark 2.2.0	
Distributed storage system	HDFS (Hadoop 2.7.3)	
Coding interface	Apache Zeppelin 0.7.3	
Programming language	Scala 2.11.8	

The PNN++ was implemented with the above cluster configuration and programming environment. We found the best execution time by adjustment of the amount of memory allocation to the executors along with the optimum data partitions.

7.4 Results and Discussion

In order to demonstrate the proposed approach, two variants of PNN++, viz. PNN_1++ and PNN_2++ with two activation functions, viz. Logistic and Cauchy activation function has been presented. The proposed approaches are compared with the traditional Gaussian activation function. Both the variant of PNN++ were analyzed with seven datasets. The details of the datasets, results generated after analyzing the datasets are presented in further sections.

7.4.1 Datasets Analyzed

The current work has analyzed

(i) the data from HEPMASS dataset and HIGGS dataset involving data from Physics experiment,

(ii) ccFraud dataset involving credit card fraud data, and

(iii) Amazon Movie Review (AMR) dataset that includes movie reviews from different users.

The HEPMASS and HIGGS dataset was collected from UCI ML repository. The HEPMASS dataset [481] contains the high energy physics experiments data to search for the presence of exotic particles. The dataset contains 28 attributes and 10500000 number of instances. The dataset is 7.8 GB, which has been used in the experiment. Here, we have analyzed the dataset to classify the particles producing collisions from a background source. The details of the attributes of the dataset are depicted in Table C.4.

The HIGGS dataset [482] has been produced using Monte Carlo simulations. The dataset contains 28 attributes and 11000000 number of instances. The dataset is of 8.0 GB. Here we have considered 7 high-level features out of 28, which were derived from 21 remaining low-level features. Hence, the dataset we have analyzed is 1.4 GB. Here, we have analyzed the dataset to classify the signal data from the background. The attribute details are in Table C.5.

The ccFraud dataset [209] contains credit card fraud data. The dataset contains 9 features and 10000000 number of instances. The negative class (genuine transactions) has 9,403,986 samples. The positive class (fraudulent transactions) has 596,014 samples. The dataset is 292 MB, which has been analyzed. This dataset is quite imbalanced, with a ratio of 94:6 genuine to fraudulent transactions. Here, we have analyzed how the classifier handles imbalanced numerical data. The details of the attributes are mentioned in Table C.3.

The AMR dataset [473] available on the Stanford Network Analysis Platform (SNAP) data repository [474]. It contains 7,911,684 reviews on 253,059 movies provided by 889,176 users, which were collected from August 1997 to October 2012. The reviews include product and user information, ratings, and a plaintext review. The ratings are in five levels of the Likert scale. The detailed structure of each review is depicted in Table C.6. The dataset is used here for sentiment classification. Here the sentiments are classified as positive and negative. The details of sentiment analysis, which is a part of social media analytics, can be found at Section 6.3.

In the experimental work, we have modified the 5-level review score to two class labels. We have made all the review scores less than 3 (< 3) to be labeled as 1.0, and the rest review scores, i.e., review scores 3, 4, and 5, are labeled as 0.0. Here the class we are interested in (i.e., positive class) is the negative review score which, we have labeled as 1.0. The dataset is imbalanced, with a ratio of 86:14 positive to negative reviews. The plain text review underwent a preprocessing phase whereby the special characters, HTML tags, emojis, the short form of texts were removed. Then, the stop words were removed, and the text was tokenized. After the preprocessing of the text was over, the tokens were converted to hashing Document Term Matrix (DTM) with TF-IDF values. We have considered the top 20 and 100 frequency terms from the hashing DTM for generating two different datasets. Also, a dataset was created with TF-IDF weights based DTM having a dimension of 100 features. The doc2Vec transformation was also made to generate features for analysis. Thus, four datasets were generated for analyzing the classifier.

7.4.2 Analysis of Results

The research work conducted comprises the analysis of the datasets mentioned above with the proposed approach. The results of the carried out work cannot be compared with any other technique as we could not find any work carried out in the same domain and using such hybrid architectural techniques.

The experiment was carried out with ten-fold cross-validation (10-FCV) of the min-max normalized dataset. The K-Means|| or Bisecting K-Means|| was carried out with k value ranging from 2 to 10, and then the Dunn-like Index was taken into account for cluster validity for finding out the optimal cluster centers. Then, with the selected cluster centers, PNN++ was carried out with the sigma (σ) value ranging from 0.001 to 2.0 with an increment of 0.001. The performance is measured as Average AUC, Sensitivity, and Specificity of 10-FCV.

Three activation functions have been used in PNN++ viz., Gaussian, Logistic, and Cauchy function, which captures the features of a test data and correspondingly classifies it. The traditional PNN employs the Gaussian Activation Function. We have proposed the Logistic and Cauchy activation function to be implemented.

We have reported average Accuracy of 10-FCV obtained with HEPMASS and HIGGS dataset, as both the datasets are balanced dataset, for the two implementations of PNN++, i.e., PNN_1++ (K-Means|| embedded in PNN) and PNN_2++ (Bisecting K-Means|| embedded in PNN) in the Table 7.3. But we have reported the average AUC, Sensitivity and Specificity of 10-FCV generated with ccFraud and AMR dataset, as both the datasets are imbalanced dataset, for PNN_1++ and PNN_2++ in Table 7.4 and Table 7.5, respectively.

Table 7.3: Mean Accuracy for 10-FCV with PNN_1++ and PNN_2++

Dataset	†PNN_1++			‡PNN_2++		
	Gaussian AF⋆	**Logistic AF**	**Cauchy AF**	**Gaussian AF**	**Logistic AF**	**Cauchy AF**
HEPMASS	69.577	**72.911**	56.126	73.704	**76.361**	67.950
HIGGS	62.042	**62.056**	62.013	62.034	**62.052**	62.022

† K-Means|| embedded in PNN ‡Bisecting K-Means|| embedded in PNN
⋆ AF = Activation Function

Table 7.4: Mean AUC, SV, SP for 10-FCV of PNN_1++ (K-Means|| embedded in PNN)

Dataset	Gaussian AF			Logistic AF			Cauchy AF		
	AUC	SV	SP	AUC	SV	SP	AUC	SV	SP
ccFraud	**0.792**	0.774	0.810	0.791	0.771	0.810	0.763	0.681	0.846
AMR DTM100	**0.6159**	0.505	0.727	0.6157	0.510	0.721	0.554	0.198	0.910
AMR hDTM20	0.545	0.717	0.374	**0.547**	0.785	0.308	0.504	0.974	0.035
AMR hDTM100	0.568	0.718	0.417	**0.570**	0.771	0.369	0.507	0.975	0.038
AMR D2V100	0.91217	0.907	0.916	**0.9122**	0.91168	0.913	0.537	0.782	0.293

AF = Activation Function, SV = Sensitivity, SP = Specificity, DTM = Document Term Matrix, hDTM = hashing DTM, D2V = Doc2Vec

Table 7.5: Mean AUC, SV, SP for 10-FCV of PNN_2++ (Bisecting K-Means|| embedded in PNN)

Dataset	Gaussian AF			Logistic AF			Cauchy AF		
	AUC	SV	SP	AUC	SV	SP	AUC	SV	SP
ccFraud	**0.79336**	0.756	0.830	0.79335	0.756	0.831	0.7926	0.756	0.829
AMR DTM100	**0.616**	0.505	0.727	0.6157	0.510	0.721	0.554	0.198	0.910
AMR hDTM20	0.545	0.719	0.371	**0.547**	0.790	0.303	0.504	0.974	0.035
AMR hDTM100	0.568	0.721	0.414	**0.570**	0.770	0.369	0.507	0.975	0.039
AMR D2V100	0.919	0.865	0.973	**0.920**	0.866	0.973	0.919	0.867	0.971

AF = Activation Function, SV = Sensitivity, SP = Specificity, DTM = Document Term Matrix, hDTM = hashing DTM, D2V = Doc2Vec

The results are shown in bold in Table 7.3 are the better values of mean Accuracy when the results of PNN_1++ and PNN_2++ are compared against each other along with three AFs. In Table 7.4 and Table 7.5, the bold figures represent the better values of mean AUC when the results are compared among the three AFs. As the results do not present a clear winner, we proceeded for a t-test and calculated the p-value. The results are presented in Table 7.6. The Mean Accuracy/AUC column

in Table 7.6 represents the average of 10-Fold Accuracy for the balanced datasets viz., HEPMASS and HIGGS dataset and it represents the average of 10-Fold AUC for the imbalanced datasets viz., ccFraud and AMR datasets.

The t-test is conducted to detect whether any model is statistically significant. The t-test was conducted at a 1% level of significance with 18 degrees of freedom. The t-test value is found to be less than 2.83 (which is the t-table value at 10+10-2 = 18 degrees of freedom) for all models except one case, which is for AMR D2V dataset with Cauchy Activation function. Hence, all the models are statistically equivalent except the one aforementioned. The model with the Bisecting K-Means|| and Cauchy Activation function is found to be statistically significant with AMR D2V dataset. Also, we performed a p-value test with a confidence level of 1%. The results present that both the models are statistically the same except the case of BK-Means|| with Cauchy AF for AMR D2V dataset.

Our results are compared to those of Ravi et al. [483]. They implemented Logistic regression (LogR), Decision Tree (DT), Random Forest (RF), Gradient Boosted Tree (GBT), MLP, and Support Vector Machine (SVM) for the classification of the reviews. Our proposed variants of PNN++ for DTM with 100 features outperformed DT, RF, GBT, and SVM. Then, both variants of PNN++ with hashing DTM with 20 features as input, surpassed LogR, DT, RF, and SVM. However, both variants of PNN++ have not fared well compared to all the methods for the hashing DTM with 100 features. Finally, both variants of PNN++ outperformed LogR and DT for the Doc2Vec dataset with 100 features.

The experiment involved structured numeric data as well as unstructured textual data. With the help of a t-test at 18 degrees of freedom, both models turned out to be statistically no different at a 1% level of significance.

We know the K-Means|| chooses the first center at random following a uniform distribution. Then, the next centers are chosen non-uniformly with a given probability, which is stochastically biased by the already chosen centers [402]. Similarly, the Bisecting K-Means algorithm bisects a cluster into two and proceeds with the cluster having higher Within Set Sum of Squares (WSSE) in each bisecting step. Thus, at each bisecting step, two clusters are formed [403]. Both the algorithms deal with a new cluster center at each step. Though both the clustering methods produced numerically different cluster centers, the PNN_1++ and PNN_2++ turned

Table 7.6: t-test on PNN_1++ Vs. PNN_2++

Dataset	Model	Gaussian AF			Logistic AF			Cauchy AF		
		⋆Mean Accuracy/ AUC	t-test value	p-value	*Mean Accuracy/ AUC	t-test value	p-value	*Mean Accuracy/ AUC	t-test value	p-value
HEPMASS	PNN_1++	69.577	1.7284	0.101032	72.911	1.862	0.079011	56.126	1.9953	0.061373
	PNN_2++	73.704			76.361			67.950		
HIGGS	PNN_1++	62.042	0.3187	0.753622	62.056	0.1543	0.87909	62.013	0.3092	0.76072
	PNN_2++	62.034			62.052			62.022		
ccFraud	PNN_1++	0.792	0.9271	0.36614	0.791	1.0892	0.290434	0.763	1.0248	0.319034
	PNN_2++	0.793			0.793			0.793		
AMR DTM100	PNN_1++	0.616	0.0	1.0	0.616	0.0	1.0	0.554	0.0	1.0
	PNN_2++	0.616			0.616			0.554		
AMR hDTM20	PNN_1++	0.545	0.0	1.0	0.547	0.0	1.0	0.504	0.0	1.0
	PNN_2++	0.545			0.547			0.504		
AMR hDTM100	PNN_1++	0.568	0.0	1.0	0.570	0.0	1.0	0.507	0.0	1.0
	PNN_2++	0.568			0.570			0.507		
AMR D2V100	PNN_1++	0.912	1.4532	0.16338	0.912	1.6149	0.123725	0.537	10.1985	<0.00001
	PNN_2++	0.919			0.920			0.919		

* Mean Accuracy considered for balanced datasets, viz., HEPMASS & HIGGS dataset; and Mean AUC for unbalanced datasets, viz., ccFraud & AMR datasets

out to be statistically the same. But, we recommend the PNN_1++ as it is found to be faster in terms of execution time.

Interestingly, numerically, both Gaussian and Logistic activation functions performed almost the same, while Cauchy activation function did not perform that well.

7.5 Conclusions

We propose two variants of parallel and distributed versions of PNN, namely PNN_1++ and PNN_2++ in Apache Spark, where HDFS takes care of the distributed data storage. These are implemented in the Scala programming language. Both variants of PNN++ could perform Big Data classification in a single pass and yielded good results under 10-fold cross-validation. The hallmark of the proposed architectures is the invocation of

(i) the parallel distributed K-Means++ aka K-Means|| (in the case of PNN_1++) and

(ii) Bisecting K-Means|| (in the case of PNN_2++) into the architecture of the traditional PNN.

The innovation of the proposed method resulted in a drastic reduction in the number of pattern layer nodes of PNN. This feature overcame the major shortcoming of PNN, which is the computational overhead.

Chapter 8

Parallel Distributed Versatile Architecture for Regression and Classification

This chapter presents a versatile network for both classification and regression problems in parallel distributed computational framework. The next section of the chapter presents an introduction to the current work. Section 8.2 presents the proposed approach. The experimental setup is described in Section 8.3. Results and discussion are presented in Section 8.4. The chapter is concluded in Section 8.5.

8.1 Introduction

Classification is one of the most fundamental tasks of data mining, which has a wide variety of applications in many domains. Similarly, another significant task of data mining is regression. Various techniques from a diverse family of methods viz., statistics, and machine learning (subsuming decision tree, fuzzy classification, support vector machine, and feed-forward neural networks) have been proposed for these tasks. Under neural networks category, various architectures were proposed for classification, viz., MLP, RBFN [484], WNN [439], PNN [386], and GMDH [440]. Also, for regression, a number of architectures were proposed, viz., MLP,

RBFN, WNN, GRNN [376], and GMDH.

There are several neural network architectures that can be called versatile architecture in the way that they can handle multiple data mining tasks. Such architectures are MLP, WNN, GMDH, and RBFN. Out of them, we have considered RBFN to implement in the parallel distributed computational environment as we did not come across any literature where the versatility of the architecture has been explored over parallel distributed computational environment.

RBFN has been used for regression in several domains in literature. Mignon and Jurie [444] proposed a face reconstruction algorithm, based on RBF-regression in Eigenspace. Hannan et al. [445] presented a heart disease diagnosis employing RBF. Taki et al. [446] employed RBF for energy consumption prediction for wheat production. Nikolaos [485] has employed RBFN for financial status evaluation of corporations. Feng and Chou [486] implemented RBFN for prediction in time-series data. Also, several implementations of RBFN have been used for classification. Lin [475] has employed RBFN for facial expression classification. Radha and Nallammal [476] have implemented face recognition using RBFN. Ravi et al. [487] have implemented RBFN for bankruptcy prediction for banks. Naveen et al. [488] have employed differential evolution trained RBFN for bankruptcy prediction in banks. Acır [489] implemented a modified RBFN to classify EEG signals for epileptiform pattern detection.

It is worthwhile to mention that the primary objective of this chapter is to present a distributed and parallel version of the traditional RBFN to handle big datasets. Therefore, it is not compared with the parallel versions of any of its competitors.

8.2 Proposed Approach

The proposed methodology is the parallel implementation of RBFN, where unsupervised learning employs a parallel implementation of either K-Means ++, which is known as K-Means|| or parallel Bisecting K-Means. The supervised learning involves the parallel implementation of Least Square Estimation (LSE) using the outer product of matrices. An overview to the K-Means++, K-Means||, and parallel Bisecting K-Means is presented in Appendix B: K-Means++, Appendix B: K-Means||,

and Appendix B: Bisecting K-Means||, respectively. The following subsections present an overview of RBFN and parallel RBFN (PRBFN).

8.2.1 RBFN: An Overview

RBFN [484] is a feed-forward neural network that implements a combination of unsupervised and supervised learning. The architecture can be used for classification as well as regression.

The topology of RBFN involves three layers of neurons, viz., input, hidden, and output. The hidden layer comprises 'k' training neurons, which are k samples obtained from the training dataset with random sampling. The test sample is fed to the input layer. The d_i is the distance between the training sample present as a neuron in the hidden layer and the data point from the test set. The output of the hidden layer is calculated with the Gaussian activation function. The weights from the hidden layer to the output layer are calculated with LSE. The output of each hidden node is multiplied with a corresponding weight w_i present between the hidden and the output node. Then, the sum of all the above products is calculated in the output node. The output with a proper threshold is classified into one of the two classes (for binary classification). In the case of a regression problem, the architecture works similarly. Only, in the last step, the above-said sum in the output layer is the predicted value. The architecture of RBFN is presented in Figure 8.1.

8.2.2 PRBFN: The Proposed Method

The training data is clustered with K-Means|| or parallel Bisecting K-Means. Thus, the cluster centers represent the data samples belonging to the cluster most effectively. Then the Dunn-like Index [464] (a modified Dunn Index [434]), a cluster validity index, is used for measuring the cluster validity.

The cluster centers generated by the clustering algorithm and validated by Dunn-like Index, now represent the nodes in the hidden layer of PRBFN. Then, each test sample is passed on to the neurons present in the hidden layer with activation function (i.e., Gaussian or Logistic activation function). The activation function results in higher value, i.e., closer to 1, if the neuron has substantial similarity to the test

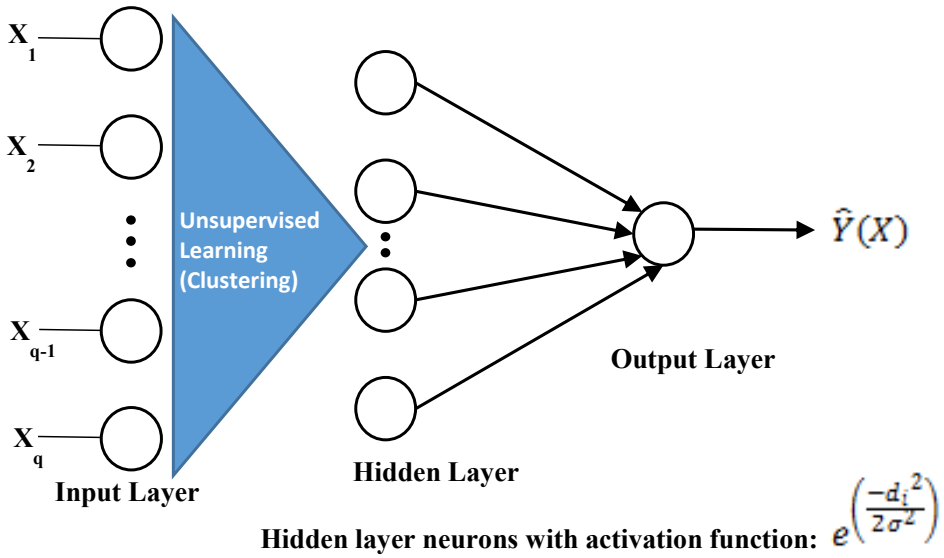

Figure 8.1: *Architecture of RBFN*

sample else it is closer to 0. We have considered the Gaussian and Logistic activation function in the hidden layer nodes. The vanilla version of RBFN implements the Gaussian activation function, and we have selected the Logistic activation function in addition to it. The reason for the selection of the Logistic activation function is that it has a similar distribution to the Gaussian activation function. The Gaussian activation function can be represented as:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)} \tag{8.1}$$

where, $x \in (-\infty, +\infty)$, μ is the mean of the distribution and $\sigma^2 \geq 0$, is the variance.

The PDF for Logistic distribution is presented as:

$$f(x) = \frac{e^{\left(\frac{-(x-\mu)}{\sigma}\right)}}{\sigma\left(1+e^{\left(\frac{-(x-\mu)}{\sigma}\right)}\right)^2} \tag{8.2}$$

where, $x \in (-\infty, +\infty)$, $\sigma \geq 0$, is the standard deviation, and μ is the mean of the

distribution.

The architecture of PRBFN is the same as that of the traditional RBFN except for the presence of the cluster centers (obtained through either by K-Means|| or parallel bisecting K-Means) occupying the hidden layer. The hidden and output layers of PRBFN are identical to that of RBFN. The architecture of PRBFN is depicted in Figure 8.2.

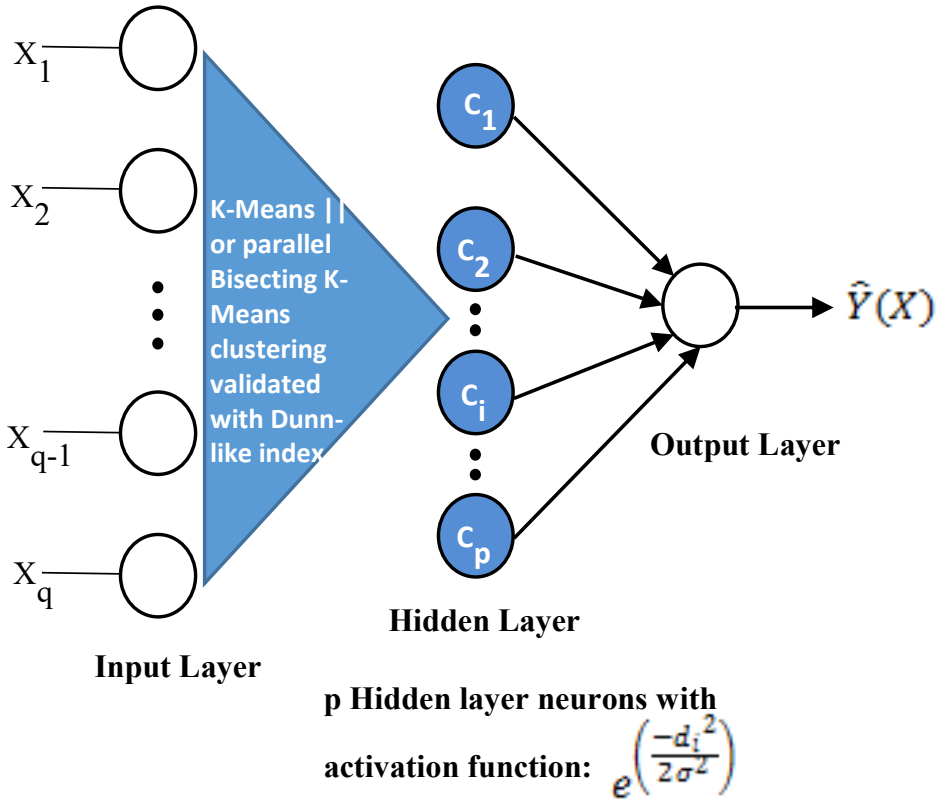

Figure 8.2: *Architecture of PRBFN*

The supervised learning in RBFN is achieved through LSE. A parallel version of LSE is implemented through the outer product of matrices, which can handle large-sized dataset.

The proposed methodology has implemented a parallel clustering approach of K-Means|| and parallel bisecting K-Means for unsupervised learning, then the cluster validity is measured with a parallel implementation of Dunn-like index, and

finally, the supervised learning of LSE is implemented with a parallel version of the outer product of matrices. Thus, the proposed method is an end-to-end parallel implementation of RBFN, i.e., PRBFN.

8.3 Experimental Setup

The system configuration for carrying out experimental work consists of standalone Spark cluster 2.2.0 as the computational framework on top of HDFS (Hadoop 2.7.3), which is the distributed storage system. The program development environment used is Apache Zeppelin 0.7.3, with Scala 2.11.8 as the coding environment. The Spark cluster consists of a system as a master node and seven systems as worker nodes. The driver program resides in the master node, along with two instances of worker daemons. The driver program is allocated a memory of 14 GB. Each of the worker nodes executes four instances of worker daemons. Each worker daemon is assigned a memory of 7 GB. All seven systems have a similar configuration. In a stand-alone cluster system, each worker daemon has one executor. Thus, the total memory utilized in each of the systems is 28 GB.

The PRBFN was implemented with the above cluster configuration and programming environment. We found the best execution time by adjustment of the amount of memory allocation to the executors along with the optimum data partitions.

8.4 Results and Discussion

The research work conducted comprises the analysis of the datasets mentioned in the next subsection with the proposed approach. The results of the carried out work cannot be compared with any other technique as we could not find any work carried out in the same domain and using such hybrid architectural techniques.

8.4.1 Datasets Analyzed

8.4.1.1 Datasets for classification

The PRBFN analyzed HEPMASS dataset [481] and HIGGS dataset [482] (collected from UCI ML repository), ccFraud dataset [209], and Amazon Movie Review (AMR) dataset [473] for classification problem. The HEPMASS dataset (refer Appendix C) contains the high energy physics experiments data to search for the presence of exotic particles. The dataset contains 28 attributes and 10500000 number of instances. The dataset is 7.8 GB, which has been used in the experiment. Here, we have analyzed the dataset to classify the particles producing collisions from a background source.

The HIGGS dataset (refer Appendix C) has been produced using Monte Carlo simulations. The dataset contains 28 attributes and 11000000 number of instances. The dataset is of 8.0 GB. Here we have considered 7 high-level features out of 28, which are derived from 21 remaining low-level features. Hence, the dataset we have analyzed is 1.4 GB. Here, we have analyzed the dataset to classify the signal data from the background.

The ccFraud dataset (refer Appendix C) contains credit card fraud data. The dataset includes 9 features and 10,000,000 number of instances. The negative class (genuine transactions) has 9,403,986 samples. The positive class (fraudulent transactions) has 596,014 samples. The dataset is 292 MB, which is analyzed. This dataset is quite imbalanced, with a ratio of 94:6 genuine to fraudulent transactions. Here, we have analyzed how the classifier handles imbalanced numerical data.

The AMR dataset (refer Appendix C) is available on the SNAP data repository [474]. It contains social media data, and the dataset is also included for analysis as it comes under the purview of Big Data Analytics (BDA). Why the dataset is considered and how it satisfies the conditions for BDA are addressed in Section 6.3. It contains 7,911,684 reviews on 253,059 movies provided by 889,176 users. The reviews include product and user information, ratings, and a plaintext review. The ratings are in five levels of the Likert scale. The detailed structure of each review is present in Appendix C.

In the experimental work, the 5-level review score is modified to two class labels for the classification problem. The details can be found at Ch. 7 Section 7.4.1.

8.4.1.2 Datasets for regression

The PRBFN analyzed (i) the data from an array of gas sensors involving different gas mixtures, and (ii) Amazon movie review (AMR) dataset that includes movie reviews from different users for the regression problem.

The gas sensor dataset [422] was collected from UCI ML repository [423]. The dataset contains the readings gathered from sixteen sensors exposed to varying concentrations of gas mixtures. The original dataset includes readings of two different mixtures of gas in Air, viz., Ethylene with Methane, and Ethylene with Carbon monoxide (CO) in two different datasets. We have analyzed the datasets for the prediction of Ethylene, Methane, and CO concentration in Air.

The 'Ethylene and CO mixture in Air' dataset, as well as 'Ethylene and Methane mixture in Air' dataset, contain 19 features. Here, the first feature represents the time of sensor reading, the second feature represents CO and Methane concentration in ppm in former and later datasets respectively, the third feature represents the Ethylene concentration in ppm. The next 16 features represent the sensor readings from 16 chemical sensors (refer Appendix C).

For our regression work, we have dropped the first feature being the time series values. Out of the second and third features, one is selected as the dependent variable. We have selected the features from the fourth to the last as independent variables for our experimentation.

The AMR dataset contains the movie reviews that include product and user information, ratings, and a plaintext review. The ratings are in five levels of the Likert scale. The detailed structure of each review is depicted in Appendix C. The more details regarding why the dataset has been considered for analysis and how it fits into the Big Data paradigm is discussed at Section 6.3.

The details of the dependent and independent variable for the regression model used in the experimental work can be referred at Section 6.5.1.

8.4.2 Analysis of Results

The experiment was carried out with ten-fold cross-validation (10-FCV) of the min-max normalized dataset. The unsupervised learning was carried out with the K-Means|| (employing k value ranging from 2 to 10) and parallel Bisecting K-Means

then the Dunn-like Index was taken into account for cluster validity for finding out the optimal clusters. Then, with the selected cluster centers, PRBFN was carried out with sigma (σ) values ranging from 0.01 to 1.0 with an increment of 0.01.

The performance of PRBFN for classification is measured as average accuracy of 10-FCV where the dataset is balanced (e.g., HEPMASS and Higgs) and AUC of 10-FCV where the dataset is imbalanced (e.g., ccFraud and AMR). The performance for regression is measured with an average Mean Squared Error (MSE) of 10-FCV. For both the classification and regression problem, the architecture employed the Gaussian and Logistic activation functions in the hidden layer of PRBFN that captures the features of a test data. Hence, it enables the architecture to classify the data or find the predicted value.

We have reported an average accuracy of 10-FCV of the balanced datasets in Table 8.1. The Table 8.2 and Table 8.3 report the average AUC of 10-FCV of the imbalanced datasets for $PRBFN_1$ (i.e., K-Means|| + PRBFN) and $PRBFN_2$ (i.e., parallel Bisecting K-Means + PRBFN), respectively. The average MSE of 10-FCV for $PRBFN_1$ is presented in Table 8.4, and the average MSE of 10-FCV for $PRBFN_2$ is presented in Table 8.5.

Table 8.1: Mean Accuracy for 10-FCV with $PRBFN_1$ and $PRBFN_2$

Dataset	$PRBFN_1$ (K-Means|| + PRBFN)		$PRBFN_2$ (Bisecting K-Means|| + PRBFN)	
	Gaussian Activation Function (AF)	Logistic AF	Gaussian AF	Logistic AF
HEPMASS	0.50031	**0.50038**	**0.50049**	0.50038
HIGGS	**0.57314**	0.5731	**0.57317**	0.57311

Table 8.2: Mean AUC for 10-FCV of $PRBFN_1$ (K-Means|| + PRBFN)

Dataset	Gaussian Activation Function (AF)	Logistic AF
ccFraud	0.50000025	**0.50000027**
AMR DTM100	**0.50031**	0.5003
AMR hDTM20	**0.5001**	0.50002
AMR hDTM100	0.50011	**0.50012**
AMR D2V100	**0.50005**	0.50002

The Table 8.1 shows the result for the balanced dataset, and here for $PRBFN_2$ the Gaussian activation function is a winner and in $PRBFN_1$ also the Gaussian AF

Table 8.3: Mean AUC for 10-FCV of $PRBFN_2$ (Bisecting K-Means|| + PRBFN)

Dataset	Gaussian Activation Function (AF)	Logistic AF
ccFraud	**0.500002**	0.500001
AMR DTM100	**0.50031**	0.5003
AMR hDTM20	**0.5001**	0.50002
AMR hDTM100	0.50011	**0.50012**
AMR D2V100	**0.50004**	0.50002

wins in case of Higgs dataset. When we refer at imbalanced datasets in Table 8.2 and Table 8.3, we can observe in $PRBFN_1$ Gaussian AF is a winner in three out of five cases. If we observe $PRBFN_2$ Gaussian AF is again the winner in four out of five cases. We have found the Gaussian AF is performing better in the case of the classification problem.

When we observe at the regression results, we figured out the Logistic function is a clear winner in the case of AMR dataset in both $PRBFN_1$ and $PRBFN_2$. The same is the winner in for gas sensor dataset in three out of four cases. Here, we figured out the Logistic regression as a winner.

8.5 Conclusions

In the proposed parallel distributed architecture of RBFN (PRBFN) where the unsupervised learning component is employed through a parallel clustering approach of either K-Means|| or parallel Bisecting K-Means which is validated through parallel Dunn-like index. The supervised learning component of PRBFN, i.e., LSE, is implemented with a parallel version of the outer product of matrices. Thus, a complete parallel version of RBFN, i.e., PRBFN, is implemented with a parallel distributed computational environment of Apache Spark, where HDFS takes care of the distributed data storage. The PRBFN is implemented in the Scala programming language. We have implemented binary classification and regression with PRBFN amenable for Big Data. The PRBFN performed semi-supervised learning and yielded results under 10-fold cross-validation. The experimental results presented the Gaussian activation function as the best performer for the classification

Table 8.4: Mean MSE for 10-FCV of $PRBFN_1$ (K-Means$\|$ + PRBFN)

Dataset	Prediction of	Gaussian AF*	Logistic AF
Gas sensor dataset	Ethylene Conc. from Ethylene-CO dataset	0.07146	**0.064223**
	CO Conc. from Ethylene-CO dataset	**0.0721126**	0.0735805
	Ethylene Conc. from Ethylene-Methane dataset	0.079	**0.0728**
	Methane Conc. from Ethylene-Methane dataset	0.0604	**0.0554**
Amazon movie review dataset	review ratings with TF-IDF values of 20 features	1.634948746	**1.6296084**
	review ratings with TF-IDF values of 100 features	1.64273374	**1.6329117**

[$\star$]AF $\Rightarrow$ Activation Function

Table 8.5: Mean MSE for 10-FCV of $PRBFN_2$ (BK-Means|| + PRBFN)

Dataset	Prediction of	Gaussian AF*	Logistic AF
Gas sensor dataset	Ethylene Conc. from Ethylene-CO dataset	0.07147	**0.064224**
	CO Conc. from Ethylene-CO dataset	**0.0721127**	0.0735807
	Ethylene Conc. from Ethylene-Methane dataset	0.078	**0.072**
	Methane Conc. from Ethylene-Methane dataset	0.0606	**0.0555**
Amazon movie review dataset	review ratings with TF-IDF values of 20 features	1.63494874	**1.6296085**
	review ratings with TF-IDF values of 100 features	1.64273379	**1.6329115**

[$\star$]AF $\Rightarrow$ Activation Function

problem and the Logistic activation function for the regression problem. The hallmark of the proposed architectures is the invocation of (i) the parallel distributed K-Means|| or the parallel Bisecting K-Means as unsupervised learning, (ii) parallel version of Dunn-like index, and (iii) parallel implementation of the outer product of matrices for LSE as the supervised learning of PRBFN.

Chapter 9

Conclusions and Future Directions

9.1 Conclusions

The thesis is mainly aimed at proposing some data mining algorithms scaled in a parallel distributed programming environment for Big Data Analytics. The proposed scaled parallel distributed algorithms include:

1. We proposed a comprehensive review of machine learning techniques implemented in a parallel, distributed computational framework of Apache Spark. These machine learning techniques are grouped into data mining tasks, viz. association rule mining, regression, classification, clustering, outlier detection. The review will help research communities to know the current developments and issues in the state-of-the-art.

2. We proposed a novel application of AAELM for performing non-linear principal component analysis in a parallel, distributed environment. We hybridized it with MLR for the purpose of Big Data regression under the Hadoop MapReduce paradigm.

3. We proposed PSOAANN in a parallel, distributed environment of Apache Spark for one-class classification.

4. We proposed an incremental one-pass clustering method ECM in parallel, distributed computational framework of Spark. The clustering method is amenable for the Big Data environment where PECM is capable of handling the velocity and the volume aspects of Big Data.

5. We proposed a parallel distributed one-pass regression model EGRNN++ which makes GRNN scalable for Big Data by invoking K-Means|| (parallel, distributed version of K-Means++) or parallel Bisecting K-Means in the pattern layer of EGRNN++.

6. We proposed a parallel distributed one-pass classifier PNN++, which is capable of handling Big Data for binary as well as multiclass classification problems. To make it amenable for Big Data K-Means|| or parallel Bisecting K-Means is employed in the pattern layer of PNN++.

7. We proposed a parallel, distributed implementation of Radial Basis Function Network (PRBFN) where the unsupervised learning part of RBFN was implemented with either K-Means|| or Bisecting K-Means|| and the supervised learning part(Least Square Estimation) was implemented with a parallel version of the outer product of matrices. The versatility of the RBFN continues to be a feature in PRBFN also, and hence we solved regression and classification problems.

9.2 Future Directions

The budding researchers are suggested to explore the following research areas in parallel distributed computation paradigm:

1. The quantile regression and elastic net regression in a parallel distributed environment can be explored.

2. The parallel, distributed version of Kernel density estimation can be employed.

3. The rough set approximation can be implemented with horizontal scaling.

4. The presence of less number of articles in various NN architectures such as QRNN, WNN, GMDH, Functional Link Neural Network (FLNN), Single Multiplicative Neural Network (SMNN), Sigma-Pi-Sigma Neural Network (SPSNN), Multi-Layer Morphological Neural Network (MLMNN) present a fertile ground to be explored.

5. Though a lot of work is reported in clustering, classification, and association rule mining, they are carried out with the Hadoop MapReduce framework. Therefore, there is also excellent scope for conducting the research in the Apache Spark environment, i.e., with the in-memory computational framework, too, in order to reap the benefits of the Spark environment.

6. The outlier detection/intrusion detection, and Recommendation present an open area to conduct more research work.

7. The unbalanced datasets and high dimensional datasets pose significant challenges in Big Data paradigm and thus present themselves as a rich area to be explored.

8. Streaming data analytics, Social Network Analysis (SNA), and Social Media Analysis (SMA) also pose challenges. The real-time or quasi-real-time streaming analysis are open areas for future work.

9. Scalable Advanced Massive Online Analysis (SAMOA) has not been explored at all for streaming data analytics.

10. Soft computing hybrid models and their applications are also attractive to explore.

11. Evolutionary algorithms present an open area for future research.

12. A few papers were found on fuzzy logic-based techniques covering fuzzy classification and fuzzy clustering. Their combination with optimization techniques can be an area to explore further.

13. Self-organizing Map (SOM) is a fascinating topology for exploration.

14. Deep learning is now a growing research area that can handle complex non-linear data with high dimensionality. It can be applied to solve Data Mining (DM) tasks. Deep learning architectures are computationally expensive due to the presence of multiple hidden layers, where there is a scope of parallelization. They can be explored with in-memory computing utilizing Apache Spark.

15. The Relevance Vector Machine (RVM), Independent Component Analysis (ICA), Singular Value Decomposition (SVD), Latent Semantic Indexing (LSI), One-Class SVM (OCSVM), Class Association Rule Mining (CARM) techniques are not explored at all, thus, resulting into an open space for exploration.

16. The different DM tasks on spatial and temporal data can be explored with their inherent challenge of high dimensionality.

17. It is also found that General-purpose Graphics Processing Unit (GPGPU) based vertical parallelization using MapReduce and Apache Spark is another fertile ground for budding and future researchers.

18. The hybridization of horizontal parallelization with vertical parallelization, i.e., a cluster of GPGPU based machines, is also an area to explore.

19. Big data visualization using MapReduce or Apache Spark is an area to be researched.

20. There is a vast scope to explore the optimization algorithms in Hadoop MapReduce or Apache Spark frameworks.

21. The research in the application domain of BDA involving the banking, insurance and the finance sector is an attractive area to explore.

22. AAELM driven NLPCA can be implemented for logistic regression in Big Data classification.

23. The Parallel evolving clustering method (PECM) can be scaled up to solve big data analytics problems with streaming data.

24. EGRNN++ or PNN++ can be extended to suit online streaming data mining by invoking the parallel evolving clustering method (PECM) in the clustering phase of EGRNN++ or PNN++.

25. The PRBFN can perform both binary and multiclass classification with some modification to the architecture. The multiclass classification can be an avenue to explore.

References

[1] 175 Zettabytes By 2025, [Online]. Available: https://www.forbes.com/sites/tomcoughlin/2018/11/27/175-zettabytes-by-2025/%7B%5C#%7D5b2adab45459 (visited on 04/03/2020).

[2] What Is Big Data? - Gartner IT Glossary - Big Data, [Online]. Available: https://research.gartner.com/definition-whatis-big-data?resId=3002918%7B%5C&%7DsrcId=1-8163325102 (visited on 07/24/2017).

[3] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. Byers, "Big data: The next frontier for innovation, competition, and productivity," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209,

[4] D. Singh and C. K. Reddy, "A survey on platforms for big data analytics," *Journal of Big Data*, vol. 2, no. 1, p. 8,

[5] M. Saecker and V. Markl, "Big Data Analytics on Modern Hardware Architectures: A Technology Survey," in *Business Intelligence*, ser. Lecture Notes in Business Information Processing, A. Marie-Aude and Z. Esteban, Eds., vol. 138, Brussels, Belgium: Springer, Berlin, Heidelberg, pp. 125–149.

[6] M. Chen, S. Mao, and Y. Liu, "Big Data: A Survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209,

[7] X. Liu, N. Iftikhar, and X. Xie, "Survey of real-time processing systems for big data," in *Proceedings of the 18th International Database Engineering & Applications Symposium on - IDEAS '14*, Porto, Portugal: ACM Press, pp. 356–361.

[8] S. Sharma, U. S. Tim, J. Wong, S. Gadia, and S. Sharma, "A Brief Review on Leading Big Data Models," *Data Science Journal*, vol. 13, no. 0, pp. 138–157,

[9] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha, "Efficient Machine Learning for Big Data: A Review," *Big Data Research*, vol. 2, no. 3, pp. 87–93,

[10] S. Sagiroglu and D. Sinanc, "Big data: A review," in *2013 International Conference on Collaboration Technologies and Systems (CTS)*, IEEE, pp. 42–47.

[11] P. Aruna Sri and M. Anusha, "Big Data-Survey," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 4, no. 1, pp. 74–80,

[12] N. Khan, I. Yaqoob, I. A. T. Hashem, Z. Inayat, W. K. M. Ali, M. Alam, M. Shiraz, and A. Gani, "Big data: survey, technologies, opportunities, and challenges.," *TheScientificWorldJournal*, vol. 2014, p. 18,

[13] H. Ekbia, M. Mattioli, I. Kouper, G. Arave, A. Ghazinejad, T. Bowman, V. R. Suri, A. Tsou, S. Weingart, and C. R. Sugimoto, "Big data, bigger dilemmas: A critical review," *Journal of the Association for Information Science and Technology*, vol. 66, no. 8, pp. 1523–1545,

[14] C. L. Philip Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Information Sciences*, vol. 275, pp. 314–347,

[15] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel data processing with MapReduce: a survey," *ACM SIGMOD Record*, vol. 40, no. 4, pp. 11–20,

[16] J. S. Ward and A. Barker, "Undefined By Data: A Survey of Big Data Definitions," *arXiv preprint arXiv:1309.5821*, arXiv: 1309.5821.

[17] A. S. Shirkhorshidi, S. Aghabozorgi, T. Y. Wah, and T. Herawan, "Big Data Clustering: A Review," in *Computational Science and Its Applications – ICCSA 2014*, ser. Lecture Notes in Computer Science, M. Beniamino, M. Sanjay, M. A. C. R. Ana, T. Carmelo, G. R. Jorge, I. F. Maria, T. David, O. A.

Bernady, and G. Osvaldo, Eds., vol. 8583, Guimarães, Portugal: Springer, Cham, pp. 707–720.

[18] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras, "A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 267–279,

[19] E. Y. Gorodov and V. V. Gubarev, "Analytical Review of Data Visualization Methods in Application to Big Data," *Journal of Electrical and Computer Engineering*, vol. 2013, pp. 1–7,

[20] H. Zhang, G. Chen, B. C. Ooi, K.-L. Tan, and M. Zhang, "In-Memory Big Data Management and Processing: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 7, pp. 1920–1948,

[21] M. R. Karim, M. A. Hossain, M. M. Rashid, B.-S. Jeong, and H.-J. Choi, "A MapReduce Framework for Mining Maximal Contiguous Frequent Patterns in Large DNA Sequence Datasets," *IETE Technical Review*, vol. 29, no. 2, pp. 162–168,

[22] M. R. Karim, C. F. Ahmed, B.-S. Jeong, and H.-J. Choi, "An Efficient Distributed Programming Model for Mining Useful Patterns in Big Datasets," *Iete Technical Review*, vol. 30, no. 1, pp. 53–63,

[23] M. Lichman. (2013). {UCI} Machine Learning Repository, [Online]. Available: http://archive.ics.uci.edu/ml.

[24] M. A. Bhuiyan and M. Al Hasan, "An iterative MapReduce based frequent subgraph mining algorithm," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 608–620, arXiv: 1307.5894.

[25] UCSB Computer Science, [Online]. Available: http://www.cs.ucsb.edu/xyan/dataset.htm (visited on 07/24/2017).

[26] The PubChem Project, [Online]. Available: https://pubchem.ncbi.nlm.nih.gov/ (visited on 07/24/2017).

[27] Universität Trier: Informatikwissenschaften, [Online]. Available: http://www.informatik.uni-trier.de/%E2%88%BCley/db/ (visited on 07/24/2017).

[28] J. Cheng, Y. Ke, W. Ng, and A. Lu, "Fg-index: towards verification-free query processing on graph databases," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data - SIGMOD '07*, Beijing, China: ACM Press, pp. 857–872.

[29] S. Hill, B. Srichandan, and R. Sunderraman, "An iterative MapReduce approach to frequent subgraph mining in biological datasets," in *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine - BCB '12*, New York, New York, USA: ACM Press, pp. 661–666.

[30] Y. Xun, J. Zhang, and X. Qin, "FiDoop: Parallel Mining of Frequent Itemsets Using MapReduce," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 3, pp. 313–325,

[31] S. Salah, R. Akbarinia, and F. Masseglia, "A highly scalable parallel algorithm for maximally informative k-itemset mining," *Knowledge and Information Systems*, vol. 50, no. 1, pp. 1–26,

[32] SNAP: Web data: Amazon movie reviews, [Online]. Available: https://snap.stanford.edu/data/web-Movies.html (visited on 11/10/2018).

[33] English Wikipedia Articles, [Online]. Available: https://dumps.wikimedia.org/enwiki/latest/ (visited on 07/23/2017).

[34] The ClueWeb09 Dataset, [Online]. Available: http://www.lemurproject.org/clueweb09.php/ (visited on 07/23/2017).

[35] J. Breier and J. Branišová, "A Dynamic Rule Creation Based Anomaly Detection Method for Identifying Security Breaches in Log Records," *Wireless Personal Communications*, vol. 94, no. 3, pp. 497–511,

[36] MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation, [Online]. Available: http://www.ll.mit.edu/ideval/data/ (visited on 07/23/2017).

[37] Snort - Network Intrusion Detection & Prevention System, [Online]. Available: https://www.snort.org/ (visited on 07/23/2017).

[38] X. Yan, J. Zhang, Y. Xun, and X. Qin, "A parallel algorithm for mining constrained frequent patterns using MapReduce," *Soft Computing*, vol. 21, no. 9, pp. 2237–2249,

[39] V. Leroy, M. Kirchgessner, A. Termier, and S. Amer-Yahia, "TopPI: An efficient algorithm for item-centric mining," *Information Systems*, vol. 64, pp. 104–118,

[40] M. K. Sohrabi and N. Taheri, "A haoop-based parallel mining of frequent itemsets using N-Lists," *Journal of the Chinese Institute of Engineers*, vol. 41, no. 3, pp. 229–238,

[41] Frequent itemset mining dataset repository, [Online]. Available: http://fimi.ua.ac.be/data (visited on 07/22/2017).

[42] Z. Liu, L. Hu, C. Wu, Y. Ding, Q. Wen, and J. Zhao, "A novel process-based association rule approach through maximal frequent itemsets for big data processing," *Future Generation Computer Systems*, vol. 81, pp. 414–424,

[43] S. Singh, R. Garg, and P. K. Mishra, "Performance optimization of MapReduce-based Apriori algorithm on Hadoop cluster," *Computers & Electrical Engineering*, vol. 67, pp. 348–364,

[44] F. Zhang, M. Liu, F. Gui, W. Shen, A. Shami, and Y. Ma, "A distributed frequent itemset mining algorithm using spark for big data analytics," *Cluster Computing*, vol. 18, no. 4, pp. 1493–1501,

[45] Y. Chen and A. An, "Approximate Parallel High Utility Itemset Mining," *Big Data Research*, vol. 6, pp. 26–42,

[46] SPMF: A Java Open-Source Data Mining Library, [Online]. Available: http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php (visited on 07/24/2017).

[47] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" In *Proceedings of the 19th international conference on World wide web - WWW '10*, New York, New York, USA: ACM Press, p. 591.

[48] Grocery shopping datasets - RecSysWiki, [Online]. Available: http://recsyswiki.com/wiki/Grocery%7B%5C_%7Dshopping%7B%5C_%7Ddatasets (visited on 07/24/2017).

[49] M. R. Karim, M. Cochez, O. D. Beyan, C. F. Ahmed, and S. Decker, "Mining maximal frequent patterns in transactional databases and dynamic data streams: A spark-based approach," *Information Sciences*, vol. 432, pp. 278–300,

[50] IBM Quest Synthetic Data Generator, [Online]. Available: https://sourceforge.net/projects/ibmquestdatagen/ (visited on 08/27/2018).

[51] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets, "Using association rules for product assortment decisions," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '99*, New York, New York, USA: ACM Press, pp. 254–260.

[52] D. Martín, M. Martínez-Ballesteros, D. García-Gil, J. Alcalá-Fdez, F. Herrera, and J. Riquelme-Santos, "MRQAR: A generic MapReduce framework to discover quantitative association rules in big data problems," *Knowledge-Based Systems*, vol. 153, pp. 176–192,

[53] LIBSVM Data: Classification, Regression, and Multi-label, [Online]. Available: https://www.csie.ntu.edu.tw/%7B~%7Dcjlin/libsvmtools/datasets/ (visited on 07/24/2017).

[54] GraphGen - a synthetic graph generator, [Online]. Available: http://www.cse.ust.hk/graphgen/ (visited on 07/24/2017).

[55] Z. Farzanyar and N. Cercone, "Efficient mining of frequent itemsets in social network data based on MapReduce framework," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks*

Analysis and Mining - ASONAM '13, Niagara, Ontario, Canada: ACM Press, pp. 1183–1188.

[56] R. Agrawal and R. Srikant, "Quest synthetic data generator," *IBM Almaden Research Center*,

[57] Z. Zheng, R. Kohavi, and L. Mason, "Real world performance of association rule algorithms," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, San Francisco, California: ACM Press, pp. 401–406.

[58] Z. Farzanyar and N. Cercone, "Accelerating Frequent Itemsets Mining on the Cloud: A MapReduce -Based Approach," in *2013 IEEE 13th International Conference on Data Mining Workshops*, IEEE, pp. 592–598.

[59] W. Mao and W. Guo, "An Improved Association Rules Mining Algorithm Based on Power Set and Hadoop," in *2013 International Conference on Information Science and Cloud Computing Companion*, IEEE, pp. 236–241.

[60] S. Natarajan and S. Sehar, "A Noval Algorithm for Distributed Data Mining in HDFS," *Proceedings - 2013 5th Int. Conf. on Adv. Computing (ICoAC)*, pp. 93–99,

[61] Z. Rong, D. Xia, and Z. Zhang, "Complex statistical analysis of big data: Implementation and application of Apriori and FP-Growth algorithm based on MapReduce," in *2013 IEEE 4th International Conference on Software Engineering and Service Science*, Beijing, China: IEEE, pp. 968–972.

[62] S. Moens, E. Aksehirli, and B. Goethals, "Frequent Itemset Mining for Big Data," in *2013 IEEE International Conference on Big Data*, Silicon Valley, CA, USA: IEEE, pp. 111–118.

[63] H. Chen, T. Y. Lin, Z. Zhang, and J. Zhong, "Parallel mining frequent patterns over big transactional data in extended mapreduce," in *2013 IEEE International Conference on Granular Computing (GrC)*, Beijing, China: IEEE, pp. 43–48.

[64] C. K.-S. Leung and F. Jiang, "A Data Science Solution for Mining Interesting Patterns from Uncertain Big Data," in *2014 IEEE Fourth International Conference on Big Data and Cloud Computing*, IEEE, pp. 235–242.

[65] R. Agrawal, R. Srikant, *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, pp. 487–499.

[66] C. K.-S. Leung, M. A. F. Mateo, and D. A. Brajczuk, "A Tree-Based Approach for Frequent Pattern Mining from Uncertain Data," in *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, T. Washio, E. Suzuki, K. Ting, and A. Inokuchi, Eds., vol. 5012, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 653–661.

[67] R.-M. Yu, M.-G. Lee, Y.-S. Huang, and S.-X. Chen, "An efficient frequent patterns mining algorithm based on MaPreduce framework," in *International Conference on Software Intelligence Technologies and Applications & International Conference on Frontiers of Internet of Things 2014*, Hsinchu, Taiwan: Institution of Engineering and Technology, pp. 1–5.

[68] Q. Yu, H. Huo, X. Chen, H. Guo, J. S. Vitter, and J. Huan, "An efficient motif finding algorithm for large DNA data sets," in *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, pp. 397–402.

[69] X. Zhou and Y. Huang, "An improved parallel association rules algorithm based on MapReduce framework for big data," in *2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, IEEE, pp. 284–288.

[70] ARtool Project, [Online]. Available: http://www.cs.umb.edu/%7B~%7Dlaur/ARtool/index.html (visited on 07/24/2017).

[71] F. Fumarola and D. Malerba, "A parallel algorithm for approximate frequent itemset mining using MapReduce," in *2014 International Conference on High Performance Computing & Simulation (HPCS)*, Bologna, Italy: IEEE, pp. 335–342.

[72] S.-H. Liu, S.-J. Liu, S.-X. Chen, and K.-M. Yu, "IOMRA - A High Efficiency Frequent Itemset Mining Algorithm Based on the MapReduce Computation Model," in *2014 IEEE 17th International Conference on Computational Science and Engineering*, Chengdu, China: IEEE, pp. 1290–1295.

[73] H. Cao, M. Phinney, D. Petersohn, B. Merideth, and C.-R. Shyu, "MRSMRS: Mining repetitive sequences in a MapReduce setting," in *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Belfast, UK: IEEE, pp. 463–470.

[74] C. Liu and Y. Li, "Non-iteration Parallel Algorithm for Frequent Pattern Discovery," in *2014 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, Xian Ning, China: IEEE, pp. 127–132.

[75] H.-Y. Chang, Y.-J. Tzang, J.-C. Lin, Z.-H. Hong, T.-Y. Chi, and C.-Y. Huang, "A Hybrid Algorithm for Frequent Pattern Mining Using MapReduce Framework," in *2015 First International Conference on Computational Intelligence Theory, Systems and Applications (CCITSA)*, IEEE, pp. 19–22.

[76] D. Sun, V. C. Lee, F. Burstein, and P. D. Haghighi, "An efficient vertical-Apriori Mapreduce algorithm for frequent item-set mining," in *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, IEEE, pp. 108–112.

[77] F. Ferrucci, P. Salza, M.-T. Kechadi, and F. Sarro, "A parallel genetic algorithms framework based on Hadoop MapReduce," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC '15*, New York, New York, USA: ACM Press, pp. 1664–1667.

[78] F. Jiang, K. Kawagoe, and C. K. Leung, "Big Social Network Mining for "Following" Patterns," in *Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering - C3S2E '15*, Yokohama, Japan: ACM Press, pp. 28–37.

[79] Stanford Large Network Dataset Collection, [Online]. Available: http://snap.stanford.edu/data/ (visited on 07/24/2017).

[80] S. Salah, R. Akbarinia, and F. Masseglia, "Fast Parallel Mining of Maximally Informative k-Itemsets in Big Data," in *2015 IEEE International Conference on Data Mining*, Atlantic City, NJ, USA: IEEE, pp. 359–368.

[81] J. Liu, Y. Wu, S. Xu, Q. Zhou, and M. Xu, "MapReduce-based efficient algorithm for finding large patterns," in *2015 Seventh International Conference on Advanced Computational Intelligence (ICACI)*, Wuyi, China: IEEE, pp. 164–169.

[82] D. Apiletti, E. Baralis, T. Cerquitelli, P. Garza, P. Michiardi, and F. Pulvirenti, "PaMPa-HD: A Parallel MapReduce-Based Frequent Pattern Miner for High-Dimensional Data," in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, Atlantic City, NJ, USA: IEEE, pp. 839–846.

[83] Kent Ridge Breast Cancer dataset, [Online]. Available: http : / / mldata . org / repository / data / viewslug / breastcancer - kent - ridge - 2 (visited on 07/24/2017).

[84] Index of /PaMPa-HD, [Online]. Available: http://dbdmg.polito.it/PaMPa-HD/ (visited on 07/24/2017).

[85] E. Baralis, L. Cagliero, P. Garza, and L. Grimaudo, "PaWI: Parallel Weighted Itemset Mining by Means of MapReduce," in *2015 IEEE International Congress on Big Data*, New York, NY, USA: IEEE, pp. 25–32.

[86] C. K. Leung, F. Jiang, H. Zhang, and A. G. Pazdor, "A Data Science Model for Big Data Analytics of Frequent Patterns," in *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, IEEE, pp. 866–873.

[87] Y. Gonen and E. Gudes, "An Improved MapReduce Algorithm for Mining Closed Frequent Itemsets," in *2016 IEEE International Conference on Software Science, Technology and Engineering (SWSTE)*, IEEE, pp. 77–83.

[88] S. Thakare, S. Rathi, and R. R. Sedamkar, “An Improved PrePost Algorithm for Frequent Pattern Mining with Hadoop on Cloud,” *Procedia Computer Science*, vol. 79, pp. 207–214,

[89] H.-Y. Chang, Z.-H. Hong, T.-L. Lin, W.-K. Chang, and Y.-Y. Lin, “IPARBC: An Improved Parallel Association Rule Based on MapReduce Framework,” in *2016 International Conference on Networking and Network Applications (NaNA)*, Hakodate, Japan: IEEE, pp. 370–374.

[90] M. Sheshikala, D. R. Rao, and R. V. Prakash, “Parallel Approach for Finding Co-location Pattern – A Map Reduce Framework,” *Procedia Computer Science*, vol. 89, pp. 341–348,

[91] F. Gui, Y. Ma, F. Zhang, M. Liu, F. Li, W. Shen, and H. Bai, “A distributed frequent itemset mining algorithm based on Spark,” in *2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, IEEE, pp. 271–275.

[92] L. Deng and Y. Lou, “Improvement and Research of FP-Growth Algorithm Based on Distributed Spark,” in *2015 International Conference on Cloud Computing and Big Data (CCBD)*, Shanghai, China: IEEE, pp. 105–108.

[93] S. Rathee, M. Kaul, and A. Kashyap, “R-Apriori: An Efficient Apriori based Algorithm on Spark,” in *Proceedings of the 8th Workshop on Ph.D. Workshop in Information and Knowledge Management*, Melbourne, Australia: ACM, pp. 27–34.

[94] P. A. Utama and B. Distiawan, “Spark-gram: Mining frequent N-grams using parallel processing in Spark,” in *2015 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, Depok, Indonesia: IEEE, pp. 129–136.

[95] R. Joy and K. K. Sherly, “Parallel frequent itemset mining with spark RDD framework for disease prediction,” in *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, Nagercoil, India: IEEE, pp. 1–5.

[96] Q. He, T. Shang, F. Zhuang, and Z. Shi, "Parallel extreme learning machine for regression based on MapReduce," *Neurocomputing*, vol. 102, pp. 52–58,

[97] J. Luts, "Real-Time Semiparametric Regression for Distributed Data Sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 2, pp. 545–557,

[98] Real-time Semiparametric Regression, [Online]. Available: http://realtime-semiparametric-regression.net/ (visited on 07/24/2017).

[99] M. Naimur Rahman, A. Esmailpour, and J. Zhao, "Machine Learning with Big Data An Efficient Electricity Generation Forecasting System," *Big Data Research*, vol. 5, no. February, pp. 9–15,

[100] C. P. Saranya and N. Nagarajan, "Efficient agricultural yield prediction using metaheuristic optimized artificial neural network using Hadoop framework," *Soft Computing*, pp. 1–11,

[101] J. Chen, K. Li, Z. Tang, and K. Bilal, "A Parallel Patient Treatment Time Prediction Algorithm and Its Applications in Hospital Queuing-Recommendation in a Big Data Environment," *IEEE Access*, vol. 4, pp. 1767–1783,

[102] I. Rodríguez-Fdez, M. Mucientes, and A. Bugarín, "S-FRULER: Scalable fuzzy rule learning through evolution for regression," *Knowledge-Based Systems*, vol. 110, pp. 255–266,

[103] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework," *J. of Mult.-Valued Logic & Soft Computing*, vol. 17, pp. 255–287,

[104] A. Galicia, J. Torres, F. Martínez-Álvarez, and A. Troncoso, "A novel spark-based multi-step forecasting algorithm for big data time series," *Information Sciences*,

[105] R. Talavera-Llames, R. Pérez-Chacón, A. Troncoso, and F. Martínez-Álvarez, "Big data time series forecasting based on nearest neighbours distributed computing with Spark," *Knowledge-Based Systems*,

[106] Y. Xu, H. Liu, and Z. Long, "A distributed computing framework for wind speed big data forecasting on Apache Spark," *Sustainable Energy Technologies and Assessments*, vol. 37, p. 100 582,

[107] D. Ding, J. Li, P. Tu, H. Wang, T. Cao, and F. Zhang, "Electric vehicle charging warning and path planning method based on spark," *IEEE Access*, vol. 8, pp. 8543–8553,

[108] W. Yin, Y. Simmhan, and V. K. Prasanna, "Scalable regression tree learning on Hadoop using OpenPlanet," in *Proceedings of third international workshop on MapReduce and its Applications Date - MapReduce '12*, Delft, The Netherlands: ACM Press, p. 57.

[109] V. Tejasviram, H. Solanki, V. Ravi, and S. Kamaruddin, "Auto associative Extreme Learning Machine based non-linear principal component regression for big data applications," in *2015 Tenth International Conference on Digital Information Management (ICDIM)*, Jeju, South Korea: IEEE, pp. 223–228.

[110] M. A. Rehab and F. Boufares, "Scalable Massively Parallel Learning of Multiple Linear Regression Algorithm with MapReduce," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 2, Helsinki, Finland: IEEE, pp. 41–47.

[111] P. K. Chavda and J. S. Dhobi, "Web users browsing behavior prediction by implementing support vector machines in MapReduce using cloud based Hadoop," in *2015 5th Nirma University International Conference on Engineering (NUiCONE)*, Ahmedabad, India: IEEE, pp. 1–6.

[112] G. Xu, M. Liu, F. Li, F. Zhang, and W. Shen, "User behavior prediction model for smart home using parallelized neural network algorithm," in *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, Nanchang, China: IEEE, pp. 221–226.

[113] S. Vluymans, H. Asfoor, Y. Saeys, C. Cornelis, M. Tolentino, A. Teredesai, and M. De Cock, "Distributed fuzzy rough prototype selection for Big Data regression," in *2015 Annual Conference of the North American Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conference on Soft Computing (WConSC)*, Redmond, WA, USA: IEEE, pp. 1–6.

[114] HCUP-US SID Overview, [Online]. Available: https://www.hcup-us.ahrq.gov/sidoverview.jsp (visited on 07/24/2017).

[115] Medical Expenditure Panel Survey, [Online]. Available: https://meps.ahrq.gov/mepsweb/ (visited on 07/24/2017).

[116] L. Oneto, E. Fumeo, G. Clerico, R. Canepa, F. Papa, C. Dambra, N. Mazzino, and D. Anguita, "Delay Prediction System for Large-Scale Railway Networks Based on Big Data Analytics," in *Advances in Big Data*, ser. Advances in Intelligent Systems and Computing, P. Angelov, Y. Manolopoulos, L. Iliadis, A. Roy, and M. Vellasco, Eds., vol. 529, Springer, Cham, pp. 139–150.

[117] S. Kamaruddin and V. Ravi, "GRNN++: A Parallel and Distributed Version of GRNN Under Apache Spark for Big Data Regression," in *Advances in Intelligent Systems and Computing*, N. Sharma, A. Chakrabarti, and V. Balas, Eds., vol. 1042, Kuala Lumpur, Malaysia: Springer, Singapore, pp. 215–227.

[118] Q. He, C. Du, Q. Wang, F. Zhuang, and Z. Shi, "A parallel incremental extreme SVM classifier," *Neurocomputing*, vol. 74, no. 16, pp. 2532–2540,

[119] M. Janaki Meena, K. R. Chandran, A. Karthik, and A. Vijay Samuel, "An enhanced ACO algorithm to select features for text categorization and its parallelization," *Expert Systems with Applications*, vol. 39, no. 5, pp. 5861–5871,

[120] G. Caruana, M. Li, and Y. Liu, "An ontology enhanced parallel SVM for scalable spam filter training," *Neurocomputing*, vol. 108, pp. 45–57,

[121] Z. H. You, J. Z. Yu, L. Zhu, S. Li, and Z. K. Wen, "A MapReduce based parallel SVM for large-scale predicting protein-protein interactions," *Neurocomputing*, vol. 145, pp. 37–43, arXiv: arXiv:1311.3144v2.

[122] X.-Y. Pan, Y.-N. Zhang, and H.-B. Shen, "Large-Scale Prediction of Human Protein−Protein Interactions from Amino Acid Sequence Based on Latent Topic Features," *Journal of Proteome Research*, vol. 9, no. 10, pp. 4992–5001,

[123] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests," *Information Sciences*, vol. 278, pp. 488–497,

[124] CAIDA Data - Overview of Datasets, Monitors, and Reports, [Online]. Available: http://www.caida.org/data/overview/ (visited on 07/24/2017).

[125] T. Chen, X. Zhang, S. Jin, and O. Kim, "Efficient classification using parallel and scalable compressed model and its application on intrusion detection," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5972–5983,

[126] M. Kumar and S. Kumar Rath, "Classification of microarray using MapReduce based proximal support vector machine classifier," *Knowledge-Based Systems*, vol. 89, pp. 584–602,

[127] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring," *Science*, vol. 286, no. 5439,

[128] GEO DataSets - NCBI, [Online]. Available: https://www.ncbi.nlm.nih.gov/gds/ (visited on 07/24/2017).

[129] D. H. Han, X. Zhang, and G. R. Wang, "Classifying Uncertain and Evolving Data Streams with Distributed Extreme Learning Machine," *Journal of Computer Science and Technology*, vol. 30, no. 4, pp. 874–887,

[130] KDD Cup 1999 Data, [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (visited on 07/24/2017).

[131] M. Barkhordari and M. Niamanesh, "ScaDiPaSi: An Effective Scalable and Distributable MapReduce-Based Method to Find Patient Similarity on Huge Healthcare Networks," *Big Data Research*, vol. 2, no. 1, pp. 19–27,

[132] V. López, S. del Río, J. M. Benítez, and F. Herrera, "Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data," *Fuzzy Sets and Systems*, vol. 258, pp. 5–38,

[133] J. Xin, Z. Wang, L. Qu, G. Yu, and Y. Kang, "A-ELM*: Adaptive Distributed Extreme Learning Machine with MapReduce," *Neurocomputing*, vol. 174, pp. 368–374,

[134] D. Xia, H. Li, B. Wang, Y. Li, and Z. Zhang, "A MapReduce-Based Nearest Neighbor Approach for Big-Data-Driven Traffic Flow Prediction," *IEEE Access*, vol. 4, pp. 2920–2934,

[135] Data platform _ large data trading platform, [Online]. Available: http://www.datatang.com/ (visited on 07/24/2017).

[136] A. Bechini, F. Marcelloni, and A. Segatori, "A MapReduce solution for associative classification of big data," *Information Sciences*, vol. 332, pp. 33–55,

[137] M. Kumar, N. K. Rath, and S. K. Rath, "Analysis of microarray leukemia data using an efficient MapReduce-based K-nearest-neighbor classifier," *Journal of Biomedical Informatics*, vol. 60, no. March, pp. 395–409,

[138] J. Chen, H. Chen, X. Wan, and G. Zheng, "MR-ELM: a MapReduce-based framework for large-scale ELM training in big data era," *Neural Computing and Applications*, vol. 27, no. 1, pp. 101–110,

[139] FCUP, [Online]. Available: http://www.dcc.fc.up.pt/*ltorgo/Regression/DataSets.html (visited on 07/24/2017).

[140] S. Huang, B. Wang, J. Qiu, J. Yao, G. Wang, and G. Yu, "Parallel ensemble of online sequential extreme learning machine based on MapReduce," *Neurocomputing*, vol. 174, pp. 352–367,

[141] J. Zhai, X. Wang, and X. Pang, "Voting-based instance selection from large data sets with MapReduce and random weight networks," *Information Sciences*, vol. 367-368, pp. 1066–1077,

[142] S. Huang, B. Wang, Y. Chen, G. Wang, and G. Yu, "An efficient parallel method for batched OS-ELM training using MapReduce," *Memetic Computing*, vol. 9, no. 3, pp. 1–15,

[143] Automatic Flower Classification Based on Large Data Sets, [Online]. Available: http://www.datatang.com/data/13152 (visited on 07/24/2017).

[144] Z. Li, W. Lu, Z. Sun, and W. Xing, "A parallel feature selection method study for text classification," *Neural Computing and Applications*, vol. 28, no. S1, pp. 513–524,

[145] J. Zhai, S. Zhang, and C. Wang, "The classification of imbalanced large data sets based on MapReduce and ensemble of ELM classifiers," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 3, pp. 1009–1017,

[146] J. Jedrzejowicz, R. Kostrzewski, J. Neumann, and M. Zakrzewska, "Imbalanced data classification using MapReduce and relief," *Journal of Information and Telecommunication*, vol. 2, no. 2, pp. 217–230,

[147] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou, "A novel ensemble method for classifying imbalanced data," *Pattern Recognition*, vol. 48, no. 5, pp. 1623–1637,

[148] X. Zhang, Q. Song, G. Wang, K. Zhang, L. He, and X. Jia, "A dissimilarity-based imbalance data classification algorithm," *Applied Intelligence*, vol. 42, no. 3, pp. 544–565,

[149] M. Elkano, M. Galar, J. Sanz, and H. Bustince, "CHI-BD: A fuzzy rule-based classification system for Big Data classification problems," *Fuzzy Sets and Systems*, vol. 348, pp. 75–101,

[150] R. K. Thakur and M. V. Deshpande, "Kernel Optimized-Support Vector Machine and Mapreduce framework for sentiment classification of train reviews," *Sadhana - Academy Proceedings in Engineering Sciences*, vol. 44, no. 6,

[151] Rajdhani Express Review, Reviews - 41 to 60 - MouthShut.com, [Online]. Available: https://www.mouthshut.com/product-reviews/Rajdhani-Express-reviews-925004322-page-3 (visited on 10/22/2019).

[152] Movie Review Data, [Online]. Available: http://www.cs.cornell.edu/people/pabo/movie-review-data/ (visited on 10/22/2019).

[153] J. Maillo, S. Ramírez, I. Triguero, and F. Herrera, "kNN-IS: An Iterative Spark-based design of the k-Nearest Neighbors classifier for big data," *Knowledge-Based Systems*, vol. 117, pp. 3–15,

[154] J. Arias, J. A. Gamez, and J. M. Puerta, "Learning distributed discrete Bayesian Network Classifiers under MapReduce with Apache Spark," *Knowledge-Based Systems*, vol. 117, pp. 16–26,

[155] Y. Liu, L. Xu, and M. Li, "The Parallelization of Back Propagation Neural Network in MapReduce and Spark," *International Journal of Parallel Programming*, vol. 45, no. 4, pp. 760–779,

[156] J. Chen, K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng, and K. Li, "A parallel random forest algorithm for big data in a spark cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 919–933,

[157] W. Shi, Y. Zhu, T. Huang, G. Sheng, Y. Lian, G. Wang, and Y. Chen, "An Integrated Data Preprocessing Framework Based on Apache Spark for Fault Diagnosis of Power Grid Equipment," *Journal of Signal Processing Systems*, vol. 86, no. 2-3, pp. 221–236,

[158] W. Lin, Z. Wu, L. Lin, A. Wen, and J. Li, "An Ensemble Random Forest Algorithm for Insurance Big Data Analysis," *IEEE Access*, vol. 5, pp. 16 568–16 575,

[159] L. R. Nair, S. D. Shetty, and S. D. Shetty, "Applying spark based machine learning model on streaming big data for health status prediction," *Computers & Electrical Engineering*, vol. 65, pp. 393–399,

[160] J. Gonzalez-Lopez, S. Ventura, and A. Cano, "Distributed nearest neighbor classification for large-scale multi-label data on spark," *Future Generation Computer Systems*, vol. 87, pp. 66–82,

[161] L. Venkataramana, S. G. Jacob, and R. Ramadoss, "A Parallel Multilevel Feature Selection algorithm for improved cancer classification," *Journal of Parallel and Distributed Computing*, vol. 138, pp. 78–98,

[162] J. Magnusson and T. Kvernvik, "Subscriber classification within telecom networks utilizing big data technologies and machine learning," in *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining Algorithms, Systems, Programming Models and Applications - BigMine '12*, Beijing, China: ACM Press, pp. 77–84.

[163] P. Erdos and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60,

[164] V. N. Khuc, C. Shivade, R. Ramnath, and J. Ramanathan, "Towards building large-scale distributed systems for twitter sentiment analysis," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12*, Trento, Italy: ACM Press, pp. 459–464.

[165] D. Wang, X. Liu, and M. Wang, "A DT-SVM Strategy for Stock Futures Prediction with Big Data," in *2013 IEEE 16th International Conference on Computational Science and Engineering*, IEEE, pp. 1005–1012.

[166] J. Han, Y. Liu, and X. Sun, "A scalable random forest algorithm based on MapReduce," in *2013 IEEE 4th International Conference on Software Engineering and Service Science*, IEEE, pp. 849–852.

[167] J. Chen, G. Zheng, and H. Chen, "ELM-MapReduce: MapReduce accelerated extreme learning machine for big spatial data analysis," in *2013 10th IEEE International Conference on Control and Automation (ICCA)*, Hangzhou, China: IEEE, pp. 400–405.

[168] X. Liu, X. Wang, S. Matwin, and N. Japkowicz, "Meta-learning for large scale machine learning with MapReduce," in *2013 IEEE International Conference on Big Data*, Silicon Valley, CA, USA: IEEE, pp. 105–110.

[169] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18,

[170] P. Lubell-Doughtie and J. Sondag, "Practical distributed classification using the Alternating Direction Method of Multipliers algorithm," in *2013 IEEE International Conference on Big Data*, Silicon Valley; CA; USA: IEEE, pp. 773–776.

[171] N. Al-Madi and S. A. Ludwig, "Scaling Genetic Programming for data classification using MapReduce methodology," in *2013 World Congress on Nature and Biologically Inspired Computing*, Fargo, ND, USA: IEEE, pp. 132–139.

[172] M. Kiran, A. Kumar, and B. R. Prathap, "Verification and validation of Parallel Support Vector Machine algorithm based on MapReduce Program model on Hadoop cluster," in *2013 International Conference on Advanced Computing and Communication Systems*, Coimbatore, India: IEEE, pp. 1–6.

[173] X. Wang, X. Liu, and S. Matwin, "A distributed instance-weighted SVM algorithm on large-scale imbalanced datasets," in *2014 IEEE International Conference on Big Data (Big Data)*, IEEE, pp. 45–51.

[174] S. H. Park and Y. G. Ha, "Large Imbalance Data Classification Based on MapReduce for Traffic Accident Prediction," in *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Birmingham, UK: IEEE, pp. 45–49.

[175] V. Lopez, S. del Rio, J. M. Benitez, and F. Herrera, "On the use of MapReduce to build linguistic fuzzy rule based classification systems for big data," in *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Beijing, China: IEEE, pp. 1905–1912.

[176] V. Kolias, C. Kolias, I. Anagnostopoulos, and E. Kayafas, "RuleMR: Classification rule discovery with MapReduce," in *2014 IEEE International Conference on Big Data (Big Data)*, Washington, DC, USA: IEEE, pp. 20–28.

[177] A. G. Kakade, P. K. Kharat, A. K. Gupta, and T. Batra, “Spam filtering techniques and MapReduce with SVM: A study,” in *2014 Asia-Pacific Conference on Computer Aided System Engineering (APCASE)*, South Kuta, Indonesia: IEEE, pp. 59–64.

[178] P. P. Anchalia and K. Roy, “The k-Nearest Neighbor Algorithm Using MapReduce Paradigm,” in *2014 5th International Conference on Intelligent Systems, Modelling and Simulation*, Langkawi, Malaysia: IEEE, pp. 513–518.

[179] X. Ke, H. Jin, X. Xie, and J. Cao, “A distributed SVM method based on the iterative MapReduce,” in *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*, IEEE, pp. 116–119.

[180] V. Kolias, I. Anagnostopoulos, and E. Kayafas, “A Covering Classification Rule Induction Approach for Big Datasets,” in *Proceedings - 2014 International Symposium on Big Data Computing, BDC 2014*, pp. 45–53.

[181] J. Maillo, I. Triguero, and F. Herrera, “A MapReduce-Based k-Nearest Neighbor Approach for Big Data Classification,” in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 2, IEEE, pp. 167–172.

[182] G. Chatzigeorgakidis, S. Karagiorgou, S. Athanasiou, and S. Skiadopoulos, “A MapReduce based k-NN joins probabilistic classifier,” in *2015 IEEE International Conference on Big Data (Big Data)*, IEEE, pp. 952–957.

[183] S. Wang, J. Zhai, S. Zhang, and H. Zhu, “An Ordinal Random Forest and Its Parallel Implementation with MapReduce,” in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, pp. 2170–2173.

[184] T. Cui and H. Zhao, “A novel gender classification method based on MapReduce,” in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, IEEE, pp. 742–745.

[185] F. Yuan, F. Lian, X. Xu, and Z. Ji, “Decision tree algorithm optimization research based on MapReduce,” in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China: IEEE, pp. 1010–1013.

[186] R. Wakayama, R. Murata, A. Kimura, T. Yamashita, Y. Yamauchi, and H. Fujiyoshi, "Distributed forests for MapReduce-based machine learning," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Kuala Lumpur, Malaysia: IEEE, pp. 276–280.

[187] I. Triguero, M. Galar, S. Vluymans, C. Cornelis, H. Bustince, F. Herrera, and Y. Saeys, "Evolutionary undersampling for imbalanced big data classification," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, Sendai, Japan: IEEE, pp. 715–722.

[188] E. Zdravevski, P. Lameski, A. Kulakov, B. Jakimovski, S. Filiposka, and D. Trajanov, "Feature Ranking Based on Information Gain for Large Classification Problems with MapReduce," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 2, Helsinki, Finland: IEEE, pp. 186–191.

[189] Competition: AAIA'14 Data Mining Competition: Key risk factors for Polish State Fire Service, [Online]. Available: https://knowledgepit.fedcsis.org/contest/view.php?id=83 (visited on 07/24/2017).

[190] M. Kumar, N. K. Rath, A. Swain, and S. K. Rath, "Feature Selection and Classification of Microarray Data using MapReduce based ANOVA and K-Nearest Neighbor," *Procedia Computer Science*, vol. 54, pp. 301–310,

[191] R. Bhukya and J. Gyani, "Fuzzy associative classification algorithm based on MapReduce framework," in *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, Davangere, India: IEEE, pp. 357–360.

[192] J. Arias, J. A. Gamez, and J. M. Puerta, "Scalable Learning of k-dependence Bayesian Classifiers under MapReduce," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 2, Helsinki, Finland: IEEE, pp. 25–32.

[193] M. V. Sukanya, S. Sathyadevan, and U. B. U. Sreeveni, "Benchmarking Support Vector Machines Implementation Using Multiple Techniques," in *Advances in Intelligent Informatics*, ser. Advances in Intelligent Systems and Computing, E.-A. El-Sayed M., T. Sabu M., T. Hideyuki, P. Selwyn, and H. Thomas, Eds., vol. 320, Springer, Cham, pp. 227–238.

[194] W. Yang, Y. Fu, and D. Zhang, “An Improved Parallel Algorithm for Text Categorization,” in *2016 International Symposium on Computer, Consumer and Control (IS3C)*, IEEE, pp. 451–454.

[195] H. Peng, D. Liang, and C. Choi, “Evaluating parallel logistic regression models,” in *2013 IEEE International Conference on Big Data*, Silicon Valley, CA, USA: IEEE, pp. 119–126.

[196] H. Tao, B. Wu, and X. Lin, “Budgeted mini-batch parallel gradient descent for support vector machines on Spark,” in *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, Hsinchu, Taiwan: IEEE, pp. 945–950.

[197] C.-Y. Lin, C.-H. Tsai, C.-P. Lee, and C.-J. Lin, “Large-scale logistic regression and linear support vector machines using spark,” in *2014 IEEE International Conference on Big Data (Big Data)*, Washington, DC, USA: IEEE, pp. 519–528.

[198] S. Wang, P. Wu, T. Liu, and W. M. Kong, “P-WLPA algorithm research on parallel framework Spark,” in *2014 IEEE 7th Joint International Information Technology and Artificial Intelligence Conference*, Chongqing, China: IEEE, pp. 437–441.

[199] A. Roy, “Automated online feature selection and learning from high-dimensional streaming data using an ensemble of Kohonen neurons,” in *2015 International Joint Conference on Neural Networks (IJCNN)*, IEEE, pp. 1–8.

[200] S. Ramirez-Gallego, S. Garcia, H. Mourino-Talin, and D. Martinez-Rego, “Distributed Entropy Minimization Discretizer for Big Data Analysis under Apache Spark,” in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 2, Helsinki, Finland: IEEE, pp. 33–40.

[201] R. C. Bhagat and S. S. Patil, “Enhanced SMOTE algorithm for classification of imbalanced big-data using Random Forest,” in *2015 IEEE International Advance Computing Conference (IACC)*, Banglore, India: IEEE, pp. 403–408.

[202] M. Chandorkar, R. Mall, O. Lauwers, J. A. Suykens, and B. D. Moor, "Fixed-Size Least Squares Support Vector Machines: Scala Implementation for Large Scale Classification," in *2015 IEEE Symposium Series on Computational Intelligence*, Cape Town, South Africa: IEEE, pp. 522–528.

[203] L. Venturini, P. Garza, and D. Apiletti, "BAC: A Bagged Associative Classifier for Big Data Frameworks," in *New Trends in Databases and Information Systems*, ser. Communications in Computer and Information Science, I. Mirjana, T. Bernhard, C. Barbara, S. Klaus-Dieter, K. Mārīte, Š. Petr, D. Ajantha, C. Tania, B. Elena, and M. Pietro, Eds., vol. 637, Springer, Cham, pp. 137–146.

[204] P. Semberecki and H. Maciejewski, "Distributed Classification of Text Documents on Apache Spark Platform," in *Artificial Intelligence and Soft Computing*, ser. Lecture Notes in Computer Science, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, and J. Zurada, Eds., vol. 9692, Springer, Cham, pp. 621–630.

[205] N. Nodarakis, S. Sioutas, A. Tsakalidis, and G. Tzimas, "Large Scale Sentiment Analysis on Twitter with Spark," in *Workshop Proceedings of the EDBT/ICDT 2016 Joint Conference*, Bordeaux, France, pp. 1–8.

[206] R. B. Ray, M. Kumar, A. Tirkey, and S. K. Rath, "Scalable Information Gain Variant on Spark Cluster for Rapid Quantification of Microarray," *Procedia Computer Science*, vol. 93, pp. 292–298,

[207] S. S. Tayde and N. Nagamma Patil, "A Novel Approach for Genome Data Classification Using Hadoop and Spark Framework," in *Emerging Research in Computing, Information, Communication and Applications*, N. R. Shetty, P. N. Hamsavath, and N. Nalini, Eds., vol. 3, Singapore: Springer, Singapore, pp. 333–343.

[208] S. Kamaruddin and V. Ravi, "Credit Card Fraud Detection using Big Data Analytics: Use of PSOAANN based One-Class Classification," in *Proceedings of the International Conference on Informatics and Analytics - ICIA-16*, Pondicherry, India: ACM Press, 33:1–33:8.

[209] ccFraud Dataset, [Online]. Available: https://packages.revolutionanalytics.com/datasets/ (visited on 07/31/2018).

[210] R. L. Talavera-Llames, R. Pérez-Chacón, M. Martinez-Ballesteros, A. Troncoso, and F. Martinez-Álvarez, "A nearest neighbours-based algorithm for big time series data forecasting," in *International Conference on Hybrid Artificial Intelligence Systems*, Springer, pp. 174–185.

[211] S. B. T. Lincy and S. K. Nagarajan, "A distributed support vector machine using apache spark for semi-supervised classification with data augmentation," in *Advances in Intelligent Systems and Computing*, J. Wang, G. Reddy, V. Prasad, and V. Reddy, Eds., vol. 898, Springer, Singapore, pp. 395–405.

[212] Z. Sun, G. Fox, W. Gu, and Z. Li, "A parallel clustering method combined information bottleneck theory and centroid-based clustering," *The Journal of Supercomputing*, vol. 69, no. 1, pp. 452–467,

[213] Million Sequence Clustering, [Online]. Available: http://salsahpc.indiana.edu/millionseq/mina/16SrRNA%7B%5C_%7Dindex.html (visited on 07/24/2017).

[214] Y. Xu, W. Qu, Z. Li, G. Min, K. Li, and Z. Liu, "Efficient k-means++ approximation with MapReduce," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3135–3144,

[215] X. Cui, P. Zhu, X. Yang, K. Li, and C. Ji, "Optimized big data K-means clustering using MapReduce," *The Journal of Supercomputing*, vol. 70, no. 3, pp. 1249–1259,

[216] S. A. Ludwig, "MapReduce-based fuzzy c-means clustering algorithm: implementation and scalability," *International Journal of Machine Learning and Cybernetics*, vol. 6, no. 6, pp. 923–934,

[217] Y. Yang, F. Teng, T. Li, H. Wang, H. Wang, and Q. Zhang, "Parallel semi-supervised multi-Ant colonies clustering ensemble based on mapreduce methodology," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99,

[218] Image database, [Online]. Available: http://research.microsoft.com/enus/projects/msrammdata/ (visited on 07/24/2017).

[219] W. Bi, M. Cai, M. Liu, and G. Li, "A Big Data Clustering Algorithm for Mitigating the Risk of Customer Churn," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 3, pp. 1270–1281,

[220] W. Liu, Y. Shen, and P. Wang, "An efficient MapReduce algorithm for similarity join in metric spaces," *The Journal of Supercomputing*, vol. 72, no. 3, pp. 1179–1200,

[221] Wikipedia, the free encyclopedia, [Online]. Available: https://en.wikipedia.org/wiki/Main%7B%5C_%7DPage (visited on 07/24/2017).

[222] Y. Zhang, S. Chen, and G. Yu, "Efficient Distributed Density Peaks for Clustering Large Data Sets in MapReduce," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, no. 99, pp. 1–14,

[223] Enron Email Dataset, [Online]. Available: http://www.cs.cmu.edu/%7B~%7D./enron/ (visited on 07/24/2017).

[224] M. Alewiwi, C. Orencik, and E. Savaş, "Efficient top-k similarity document search utilizing distributed file systems and cosine similarity," *Cluster Computing*, vol. 19, no. 1, pp. 109–126,

[225] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A New Benchmark Collection for Text Categorization Research," *Journal of Machine Learning Research*, vol. 5, no. Apr, pp. 361–397,

[226] Y. Fang, R. Cheng, W. Tang, S. Maniu, and X. Yang, "Scalable Algorithms for Nearest-Neighbor Joins on Big Trajectory Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 785–800,

[227] Y. Theodoridis, J. R. O. Silva, and M. A. Nascimento, "On the Generation of Spatiotemporal Datasets," in *Advances in Spatial Databases*, ser. Lecture Notes in Computer Science, H. G. Ralf, P. Dimitris, and L. Fred, Eds., vol. 1651, Springer, Berlin, Heidelberg, pp. 147–164.

[228] T. Brinkhoff, “A Framework for Generating Network-Based Moving Objects,” *GeoInformatica*, vol. 6, no. 2, pp. 153–180,

[229] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, “T-drive: driving directions based on taxi trajectories,” in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS ’10*, San Jose, California: ACM Press, pp. 99–108.

[230] Z. Wu, G. Gao, Z. Bu, and J. Cao, “SIMPLE: a simplifying-ensembling framework for parallel community detection from large networks,” *Cluster Computing*, vol. 19, no. 1, pp. 211–221,

[231] S. Shahrivari and S. Jalili, “Single-pass and linear-time k-means clustering based on MapReduce,” *Information Systems*, vol. 60, pp. 1–12,

[232] A. Banharnsakun, “A MapReduce-based artificial bee colony for large-scale data clustering,” *Pattern Recognition Letters*, vol. 93, pp. 78–84,

[233] M. Capó, A. Pérez, and J. A. Lozano, “An efficient approximation to the K-means clustering for massive data,” *Knowledge-Based Systems*, vol. 117, pp. 56–69,

[234] B. Tidke, R. Mehta, D. Rana, and H. Jangir, “Topic Sensitive User Clustering Using Sentiment Score and Similarity Measures: Big Data and Social Network,” *International Journal of Web-Based Learning and Teaching Technologies (IJWLTT)*, vol. 15, no. 2, pp. 34–45,

[235] Y. Lu, B. Cao, C. Rego, and F. Glover, “A Tabu search based clustering algorithm and its parallel implementation on Spark,” *Applied Soft Computing*, vol. 63, pp. 97–109,

[236] D. Xia, F. Ning, and W. He, “Research on Parallel Adaptive Canopy-K-Means Clustering Algorithm for Big Data Mining Based on Cloud Platform,” *Journal of Grid Computing*, pp. 1–11,

[237] J. Yuan, “An Anomaly Data Mining Method for Mass Sensor Networks Using Improved PSO Algorithm Based on Spark Parallel Framework,” *Journal of Grid Computing*, pp. 1–11,

[238] M. Ianni, E. Masciari, G. M. Mazzeo, M. Mezzanzanica, and C. Zaniolo, "Fast and effective Big Data exploration by clustering," *Future Generation Computer Systems*, vol. 102, pp. 84–94,

[239] W. Zhao, H. Ma, and Q. He, "Parallel K-Means Clustering Based on MapReduce," in *Cloud Computing*, ser. Lecture Notes in Computer Science, C. R. Martin Gilje Jaatun, Gansen Zhao, Ed., vol. 5931, Beijing, China: Springer, Berlin, Heidelberg, pp. 674–679.

[240] A. Ene, S. Im, and B. Moseley, "Fast clustering using MapReduce," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*, San Diego, California, USA: ACM Press, pp. 681–689.

[241] H. Zongzhen, Z. Weina, Liyue, D. Xiaojuan, and Y. Fan, "A Fuzzy Approach to Clustering of Text Documents Based on MapReduce," in *2013 International Conference on Computational and Information Sciences*, IEEE, pp. 666–669.

[242] Q. Liao, F. Yang, and J. Zhao, "An improved parallel K-means clustering algorithm with MapReduce," in *2013 15th IEEE International Conference on Communication Technology*, IEEE, pp. 764–768.

[243] R. M. Esteves, T. Hacker, and C. Rong, "Competitive K-means: A new accurate and distributed K-means algorithm for large datasets," in *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, vol. 1, Bristol, UK: IEEE, pp. 17–24.

[244] A. Kumar, M. Kiran, and B. R. Prathap, "Verification and validation of MapReduce program model for parallel K-means algorithm on Hadoop cluster," in *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, Tiruchengode, India: IEEE, pp. 1–8.

[245] K. Lin, X. Li, Z. Zhang, and J. Chen, "A K-means clustering with optimized initial center based on Hadoop platform," in *2014 9th International Conference on Computer Science & Education*, IEEE, pp. 263–266.

[246] R. Zhang and Y. Wang, "An enhanced agglomerative fuzzy k-means clustering method with mapreduce implementation on Hadoop platform," in *2014 IEEE International Conference on Progress in Informatics and Computing*, IEEE, pp. 509–513.

[247] A. Bousbaci and N. Kamel, "A parallel sampling-PSO-multi-core-K-means algorithm using mapreduce," in *2014 14th International Conference on Hybrid Intelligent Systems*, IEEE, pp. 129–134.

[248] Clustering benchmark datasets, [Online]. Available: http://cs.joensuu.fi/sipu/datasets/ (visited on 07/24/2017).

[249] D. Garg and K. Trivedi, "Fuzzy K-mean clustering in MapReduce on cloud based hadoop," in *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, Ramanathapuram, India: IEEE, pp. 1607–1610.

[250] P. P. Anchalia, "Improved MapReduce k-Means Clustering Algorithm with Combiner," in *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, Cambridge, UK: IEEE, pp. 386–391.

[251] Y.-t. Zhu, F.-z. Wang, X.-h. Shan, and X.-y. Lv, "K-medoids clustering based on MapReduce and optimal search of medoids," in *2014 9th International Conference on Computer Science & Education*, Vancouver, BC, Canada: IEEE, pp. 573–577.

[252] Y.-M. Choi and H. K.-H. So, "Map-reduce processing of k-means algorithm with FPGA-accelerated computer cluster," in *2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors*, Zurich, Switzerland: IEEE, pp. 9–16.

[253] K. D. Garcia and M. C. Naldi, "Multiple Parallel MapReduce k-Means Clustering with Validation and Selection," in *2014 Brazilian Conference on Intelligent Systems*, Sao Paulo, Brazil: IEEE, pp. 432–437.

[254] V. Melnykov, W.-C. Chen, and R. Maitra, "MixSim : An R Package for Simulating Data to Study Performance of Clustering Algorithms," *Journal of Statistical Software*, vol. 51, no. 12, pp. 1–25,

[255] M. Daoudi, S. Hamena, Z. Benmounah, and M. Batouche, "Parallel diffrential evolution clustering algorithm based on MapReduce," in *2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, Tunis, Tunisia: IEEE, pp. 337–341.

[256] N. Al-Madi, I. Aljarah, and S. A. Ludwig, "Parallel glowworm swarm optimization clustering algorithm based on MapReduce," in *2014 IEEE Symposium on Swarm Intelligence*, Orlando, FL, USA: IEEE, pp. 1–8.

[257] Datasets – MOA Massive Online Analysis, [Online]. Available: https://moa.cms.waikato.ac.nz/datasets/ (visited on 07/24/2017).

[258] R. Orlandic, Y. Lai, and W. G. Yee, "Clustering high-dimensional data using an efficient and effective data space reduction," in *Proceedings of the 14th ACM international conference on Information and knowledge management - CIKM '05*, Bremen, Germany: ACM Press, pp. 201–208.

[259] Y. Jiang and J. Zhang, "Parallel K-Medoids clustering algorithm based on Hadoop," in *2014 IEEE 5th International Conference on Software Engineering and Service Science*, Beijing, China: IEEE, pp. 649–652.

[260] H. Yuan, Y. Ma, F. Zhang, M. Liu, and W. Shen, "A Distributed Link Prediction Algorithm Based on Clustering in Dynamic Social Networks," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, pp. 1341–1345.

[261] R. Shettar and B. V. Purohit, "A MapReduce framework to implement enhanced K-means algorithm," in *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, IEEE, pp. 361–363.

[262] (2014). DEBS-Grand Challenge, 8th ACM International conference on Distributed Event-Based systems, [Online]. Available: http://www.cse.iitb.ac.in/debs2014/?page%7B%5C_%7Did=42 (visited on 07/24/2017).

[263] Q. Yu and Z. Ding, "An improved Fuzzy C-Means algorithm based on MapReduce," in *2015 8th International Conference on Biomedical Engineering and Informatics (BMEI)*, Shenyang, China: IEEE, pp. 634–638.

[264] A. Boukhdhir, O. Lachiheb, and M. S. Gouider, "An improved mapReduce design of kmeans for clustering very large datasets," in *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, IEEE, pp. 1–6.

[265] O. Lachiheb, M. S. Gouider, and L. B. Said, "An Improved MapReduce Design of Kmeans with Iteration Reducing for Clustering Stock Exchange Very Large Datasets," in *2015 11th International Conference on Semantics, Knowledge and Grids (SKG)*, IEEE, pp. 252–255.

[266] Y. Mao, Z. Xu, X. Li, and P. Ping, "An optimal distributed K-Means clustering algorithm based on cloudstack," in *2015 IEEE International Conference on Information and Automation*, IEEE, pp. 3149–3156.

[267] V. Saranya and M. Sumalatha, "Dynamic neighborhood selection (DNS) clustering using MapReduce," in *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, Chennai, India: IEEE, pp. 1–5.

[268] S. Ketu, B. R. Prasad, and S. Agarwal, "Effect of Corpus Size Selection on Performance of Map-Reduce Based Distributed K-Means for Big Textual Data Clustering," in *Proceedings of the Sixth International Conference on Computer and Communication Technology 2015*, Allahabad, India: ACM, pp. 256–260.

[269] DBpedia, [Online]. Available: http://wiki.dbpedia.org/Datasets (visited on 07/24/2017).

[270] DBpedia : 2014 Downloads, [Online]. Available: http://oldwiki.dbpedia.org/Downloads2014 (visited on 07/24/2017).

[271] J. Karimov and M. Ozbayoglu, "High quality clustering of big data and solving empty-clustering problem with an evolutionary hybrid algorithm," in *2015 IEEE International Conference on Big Data (Big Data)*, Santa Clara, CA, USA: IEEE, pp. 1473–1478.

[272] J. Karimov, M. Ozbayoglu, and E. Dogdu, "k-Means Performance Improvements with Centroid Calculation Heuristics Both for Serial and Parallel En-

vironments," in *2015 IEEE International Congress on Big Data*, New York, NY, USA: IEEE, pp. 444–451.

[273] D. Garg, P. Gohil, and K. Trivedi, "Modified Fuzzy K-mean clustering using MapReduce in Hadoop and cloud," in *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, India: IEEE, pp. 1–5.

[274] S. Ling and Q. Yunfeng, "Optimization of the distributed K-means clustering algorithm based on set pair analysis," in *2015 8th International Congress on Image and Signal Processing (CISP)*, Shenyang, China: IEEE, pp. 1593–1598.

[275] G. Tao, D. Xiangwu, and L. Yefeng, "Parallel k-modes algorithm based on MapReduce," in *2015 Third International Conference on Digital Information, Networking, and Wireless Communications (DINWC)*, Moscow, Russia: IEEE, pp. 176–179.

[276] T. N. Phan, M. Jager, S. Nadschlager, and J. Kung, "Range-Based Clustering Supporting Similarity Search in Big Data," in *2015 26th International Workshop on Database and Expert Systems Applications (DEXA)*, Valencia, Spain: IEEE, pp. 120–124.

[277] A. P. Chunne, U. Chandrasekhar, and C. Malhotra, "Real time clustering of tweets using adaptive PSO technique and MapReduce," in *2015 Global Conference on Communication Technologies (GCCT)*, Thuckalay, India: IEEE, pp. 452–457.

[278] C.-C. Chen, T.-Y. Chen, J.-W. Huang, and M.-S. Chen, "Reducing Communication and Merging Overheads for Distributed Clustering Algorithms on the Cloud," in *2015 International Conference on Cloud Computing and Big Data (CCBD)*, Shanghai, China: IEEE, pp. 41–48.

[279] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: A New Data Clustering Algorithm and Its Applications," *Data Mining and Knowledge Discovery*, vol. 1, no. 2, pp. 141–182,

[280] K. Wu, W. Zeng, T. Wu, and Y. An, “Research and improve on K-means algorithm based on hadoop,” in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China: IEEE, pp. 334–337.

[281] W. Gao, X. Li, and D. Li, “Research on fixed traffic bottleneck of K-means clustering based on Hadoop,” in *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, Harbin, China: IEEE, pp. 351–354.

[282] G. V. de Oliveira and M. C. Naldi, “Scalable Fast Evolutionary k-Means Clustering,” in *2015 Brazilian Conference on Intelligent Systems (BRACIS)*, Natal, Brazil: IEEE, pp. 74–79.

[283] V. S. Moertini and L. Venica, “Enhancing parallel k-means using map reduce for discovering knowledge from big data,” in *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, Chengdu, China: IEEE, pp. 81–87.

[284] N. Akthar, M. V. Ahamad, and S. Ahmad, “MapReduce Model of Improved K-Means Clustering Algorithm Using Hadoop MapReduce,” in *2016 Second International Conference on Computational Intelligence & Communication Technology (CICT)*, Ghaziabad, India: IEEE, pp. 192–198.

[285] Y. Zhong and D. Liu, “The application of K-means clustering algorithm based on Hadoop,” in *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, Chengdu, China: IEEE, pp. 88–92.

[286] T. Budhraja, B. Goyal, A. Kilaru, and V. Sikarwar, “Fuzzy Clustering-Based Efficient Classification Model for Large TCP Dump Dataset Using Hadoop Framework,” in *Proceedings of International Conference on ICT for Sustainable Development*, ser. Advances in Intelligent Systems and Computing, S. Suresh Chandra, J. Amit, M. Nilesh, and P. Nisarg, Eds., vol. 408, Springer, Singapore, pp. 427–437.

[287] S. Alshammari, M. B. Zolkepli, and R. B. Abdullah, “Genetic Algorithm Based Parallel K-Means Data Clustering Algorithm Using MapReduce Pro-

gramming Paradigm on Hadoop Environment (GAPKCA)," in *Advances in Intelligent Systems and Computing*, vol. 978, Springer, pp. 98–108.

[288] J. Drechsler, *Synthetic datasets for statistical disclosure control: theory and implementation*. Springer Science & Business Media, vol. 201.

[289] T. Sarazin, H. Azzag, and M. Lebbah, "SOM Clustering Using Spark-MapReduce," in *2014 IEEE International Parallel & Distributed Processing Symposium Workshops*, Phoenix, AZ, USA: IEEE, pp. 1727–1734.

[290] N. Tsapanos, A. Tefas, N. Nikolaidis, and I. Pitas, "Distributed, MapReduce-Based Nearest Neighbor and E-Ball Kernel k-Means," in *2015 IEEE Symposium Series on Computational Intelligence*, Cape Town, South Africa: IEEE, pp. 509–515.

[291] K. Govindarajan, D. Boulanger, V. S. Kumar, and Kinshuk, "Parallel Particle Swarm Optimization (PPSO) clustering for learning analytics," in *2015 IEEE International Conference on Big Data (Big Data)*, Santa Clara, CA, USA: IEEE, pp. 1461–1465.

[292] S. Ketu and S. Agarwal, "Performance enhancement of distributed K-Means clustering for big Data analytics through in-memory computation," in *2015 Eighth International Conference on Contemporary Computing (IC3)*, Noida, India: IEEE, pp. 318–324.

[293] H. Zhu, Y. Guo, M. Niu, G. Yang, and L. Jiao, "Distributed SAR image change detection based on Spark," in *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Milan, Italy: IEEE, pp. 4149–4152.

[294] X.-Y. Peng, Y.-B. Yang, C.-D. Wang, D. Huang, and J.-H. Lai, "An Efficient Parallel Nonlinear Clustering Algorithm Using MapReduce," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, Chicago, IL, USA: IEEE, pp. 1473–1476.

[295] D. Han, A. Agrawal, W.-K. Liao, and A. Choudhary, "A Novel Scalable DBSCAN Algorithm with Spark," in *2016 IEEE International Parallel and*

Distributed Processing Symposium Workshops (IPDPSW), IEEE, pp. 1393–1402.

[296] J. Jędrzejowicz, P. Jędrzejowicz, and I. Wierzbowska, "Apache Spark Implementation of the Distance-Based Kernel-Based Fuzzy C-Means Clustering Classifier," in *Intelligent Decision Technologies 2016*, ser. Smart Innovation, Systems and Technologies, I. Czarnowski, A. Caballero, R. Howlett, and L. Jain, Eds., vol. 56, Springer, Cham, pp. 317–324.

[297] N. Tsapanos, A. Tefas, N. Nikolaidis, and I. Pitas, "Efficient MapReduce Kernel k-Means for Big Data Clustering," in *Proceedings of the 9th Hellenic Conference on Artificial Intelligence - SETN '16*, Thessaloniki, Greece: ACM Press, pp. 1–5.

[298] N. Bharill, A. Tiwari, and A. Malviya, "Fuzzy Based Clustering Algorithms to Handle Big Data with Implementation on Apache Spark," in *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)*, Oxford, UK: IEEE, pp. 95–104.

[299] F. Gouineau, T. Landry, and T. Triplet, "PatchWork, a scalable density-grid clustering algorithm," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing - SAC '16*, Pisa, Italy: ACM Press, pp. 824–831.

[300] S. Zhu, J. Li, J. Huang, S. Luo, and W. Peng, "A mapreduced-based and cell-based outlier detection algorithm," *Wuhan University Journal of Natural Sciences*, vol. 19, no. 3, pp. 199–205,

[301] N. Soltani Halvaiee and M. K. Akbari, "A novel model for credit card fraud detection using Artificial Immune Systems," *Applied Soft Computing Journal*, vol. 24, pp. 40–49,

[302] P. Natesan, R. R. Rajalaxmi, G. Gowrison, and P. Balasubramanie, "Hadoop Based Parallel Binary Bat Algorithm for Network Intrusion Detection," *International Journal of Parallel Programming*, pp. 1–20,

[303] E.-S. M. El-Alfy and M. A. Alshammari, "Towards scalable rough set based attribute subset selection for intrusion detection using parallel genetic algo-

rithm in MapReduce," *Simulation Modelling Practice and Theory*, vol. 64, pp. 18–29,

[304] M. M. Rathore, A. Ahmad, and A. Paul, "Real time intrusion detection system for ultra-high-speed big data environments," *The Journal of Supercomputing*, vol. 72, no. 9, pp. 3489–3510,

[305] NSL-KDD dataset, [Online]. Available: http://www.unb.ca/cic/research/datasets/nsl.html (visited on 07/24/2017).

[306] F. Carcillo, A. Dal Pozzolo, Y.-A. Le Borgne, O. Caelen, Y. Mazzer, and G. Bontempi, "SCARFF: A scalable framework for streaming credit card fraud detection with spark," *Information Fusion*, vol. 41, pp. 182–194,

[307] S. Tanupabrungsun and T. Achalakul, "Feature Reduction for Anomaly Detection in Manufacturing with MapReduce GA/kNN," in *2013 International Conference on Parallel and Distributed Systems*, Seoul, South Korea: IEEE, pp. 639–644.

[308] I. Aljarah and S. A. Ludwig, "MapReduce intrusion detection system based on a particle swarm optimization clustering algorithm," in *2013 IEEE Congress on Evolutionary Computation*, Cancun, Mexico: IEEE, pp. 955–962.

[309] J. Xiang, M. Westerlund, D. Sovilj, and G. Pulkkis, "Using extreme learning machine for intrusion detection in a big data environment," in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop - AISec '14*, Scottsdale, Arizona, USA: ACM Press, pp. 73–82.

[310] R. Sharma, P. Sharma, P. Mishra, and E. S. Pilli, "Towards MapReduce based classification approaches for Intrusion Detection," in *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, Noida, India: IEEE, pp. 361–367.

[311] G. P. Gupta and M. Kulariya, "A Framework for Fast and Efficient Cyber Security Network Intrusion Detection Using Apache Spark," *Procedia Computer Science*, vol. 93, pp. 824–831,

[312] R. Kumari, Sheetanshu, M. K. Singh, R. Jha, and N. Singh, "Anomaly detection in network traffic using K-mean clustering," in *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, IEEE, pp. 387–393.

[313] B. Veloso, F. Leal, H. González-Vélez, B. Malheiro, and J. C. Burguillo, "Scalable data analytics using crowdsourced repositories and streams," *Journal of Parallel and Distributed Computing*, vol. 122, pp. 1–10,

[314] Yelp Dataset, [Online]. Available: https://www.yelp.com/dataset (visited on 08/31/2018).

[315] S. Schelter, C. Boden, and V. Markl, "Scalable similarity-based neighborhood methods with MapReduce," in *Proceedings of the sixth ACM conference on Recommender systems - RecSys '12*, Dublin, Ireland: ACM Press, pp. 163–170.

[316] MovieLens | GroupLens, [Online]. Available: https://grouplens.org/datasets/movielens/ (visited on 07/24/2017).

[317] Flixster dataset, [Online]. Available: http://www.cs.sfu.ca/%E2%88%BCsja25/personal/datasets (visited on 07/24/2017).

[318] P. Ghuli, A. Ghosh, and R. Shettar, "A collaborative filtering recommendation engine in a distributed environment," in *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, IEEE, pp. 568–574.

[319] Y. Shang, Z. Li, W. Qu, Y. Xu, Z. Song, and X. Zhou, "Scalable Collaborative Filtering Recommendation Algorithm with MapReduce," in *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, Dalian, China: IEEE, pp. 103–108.

[320] M. Pozo and R. Chiky, "An implementation of a Distributed Stochastic Gradient Descent for Recommender Systems based on Map-Reduce," in *2015 International Workshop on Computational Intelligence for Multimedia Understanding (IWCIM)*, IEEE, pp. 1–5.

[321] F. Lu, L. Hong, and L. Changfeng, "The improvement and implementation of distributed item-based collaborative filtering algorithm on Hadoop," in

2015 34th Chinese Control Conference (CCC), Hangzhou, China: IEEE, pp. 9078–9083.

[322] V. Subramaniyaswamy, V. Vijayakumar, R. Logesh, and V. Indragandhi, "Unstructured data analysis on big data using map reduce," *Procedia Computer Science*, vol. 50, pp. 456–465,

[323] F. Shen and R. Jiamthapthaksin, "Dimension independent cosine similarity for collaborative filtering using MapReduce," in *2016 8th International Conference on Knowledge and Smart Technology (KST)*, Chiangmai, Thailand: IEEE, pp. 72–76.

[324] Webscope | Yahoo Labs, [Online]. Available: https://webscope.sandbox.yahoo.com/ (visited on 07/24/2017).

[325] S. Panigrahi, R. K. Lenka, and A. Stitipragyan, "A Hybrid Distributed Collaborative Filtering Recommender Engine Using Apache Spark," *Procedia Computer Science*, vol. 83, pp. 1000–1006,

[326] P. Singh, K. Dutta, R. Kaye, and S. Garg, "Music Listening History Dataset Curation and Distributed Music Recommendation Engines Using Collaborative Filtering," in *Proceedings of ICETIT 2019*, P. Singh, B. Panigrahi, N. Suryadevara, S. Sharma, and A. Singh, Eds., vol. 605, Delhi, India: Springer, Cham, pp. 623–632.

[327] L. Kumar, A. Mitra, M. Mittal, V. Sanghvi, S. Roy, and S. K. Setua, "Music Tagging and Similarity Analysis for Recommendation System," in *Advances in Intelligent Systems and Computing*, vol. 999, Springer, pp. 477–485.

[328] J. Zhang, T. Li, D. Ruan, Z. Gao, and C. Zhao, "A parallel method for computing rough set approximations," *Information Sciences*, vol. 194, pp. 209–223,

[329] K. Chen, W. Q. Wan, and Y. Li, "Differentially private feature selection under Map Reduce framework," *Journal of China Universities of Posts and Telecommunications*, vol. 20, no. 5, pp. 85–90,

[330] J. Qian, D. Miao, Z. Zhang, and X. Yue, "Parallel attribute reduction algorithms using MapReduce," *Information Sciences*, vol. 279, pp. 671–690,

[331] K. Yue, Q. Fang, X. Wang, J. Li, and W. Liu, "A parallel and incremental approach for data-intensive learning of Bayesian networks," *IEEE Transactions on Cybernetics*, vol. 45, no. 12, pp. 2890–2904,

[332] Y. Zhang, S. Chen, Q. Wang, and G. Yu, "i2MapReduce: Incremental MapReduce for Mining Evolving Big Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 7, pp. 1906–1919,

[333] BigCross dataset, [Online]. Available: http://www.cs.uni-paderborn.de/en/fachgebiete/agbloemer/research/clustering/streamkmpp (visited on 07/24/2017).

[334] SNAP: Network datasets: Wikipedia talk network, [Online]. Available: http://snap.stanford.edu/data/wiki-Talk.html (visited on 07/24/2017).

[335] K. Yue, H. Wu, X. Fu, J. Xu, Z. Yin, and W. Liu, "A data-intensive approach for discovering user similarities in social behavioral interactions based on the bayesian network," *Neurocomputing*, vol. 219, pp. 364–375,

[336] Y. Wang, Y. Li, Z. Chen, and Y. Xue, "Cooperative particle swarm optimization using MapReduce," *Soft Computing*, vol. 21, no. 22, pp. 6593–6603,

[337] R. Z. Qi, Z. J. Wang, and S. Y. Li, "A Parallel Genetic Algorithm Based on Spark for Pairwise Test Suite Generation," *Journal of Computer Science and Technology*, vol. 31, no. 2, pp. 417–427,

[338] Y. Jia, M. B. Cohen, M. Harman, and J. Petke, "Learning Combinatorial Interaction Test Generation Strategies Using Hyperheuristic Search," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Florence, Italy: IEEE, pp. 540–550.

[339] B. J. Garvin, M. B. Cohen, and M. B. Dwyer, "Evaluating improvements to a meta-heuristic search for constrained interaction testing," *Empirical Software Engineering*, vol. 16, no. 1, pp. 61–102,

[340] C. Eiras-Franco, V. Bolón-Canedo, S. Ramos, J. González-Domínguez, A. Alonso-Betanzos, and J. Touriño, "Multithreaded and Spark parallelization of feature selection filters," *Journal of Computational Science*, vol. 17, pp. 609–619,

[341] S. Ramírez-Gallego, I. Lastra, D. Martínez-Rego, V. Bolón-Canedo, J. M. Benítez, F. Herrera, and A. Alonso-Betanzos, "Fast-mRMR: Fast Minimum Redundancy Maximum Relevance Algorithm for High-Dimensional Big Data," *International Journal of Intelligent Systems*, vol. 32, no. 2, pp. 134–152,

[342] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*, San Diego, California, USA: ACM Press, pp. 69–77.

[343] C. Zhang and J. Sun, "Large scale microblog mining using distributed MB-LDA," in *Proceedings of the 21st international conference companion on World Wide Web - WWW '12 Companion*, Lyon, France: ACM Press, pp. 1035–1042.

[344] L. P. Thompson, W. Xu, and D. P. Miranker, "Fast scalable selection algorithms for large scale data," in *2013 IEEE International Conference on Big Data*, Silicon Valley, CA, USA: IEEE, pp. 412–420.

[345] L. Hu, J. Liu, C. Liang, and F. Ni, "A MapReduce Enabled Simulated Annealing Genetic Algorithm," in *2014 International Conference on Identification, Information and Knowledge in the Internet of Things*, IEEE, pp. 252–255.

[346] Traveling Salesman Problem, [Online]. Available: http : / / www . math . uwaterloo.ca/tsp/index.html (visited on 07/24/2017).

[347] Q. He, X. Cheng, F. Zhuang, and Z. Shi, "Parallel feature selection using positive approximation based on MapReduce," in *2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Xiamen, China: IEEE, pp. 397–402.

[348] G. T. Hilda and R. R. Rajalaxmi, "Effective feature selection for supervised learning using genetic algorithm," in *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, Coimbatore, India: IEEE, pp. 909–914.

[349] A. Kourid and M. Batouche, "A novel approach for feature selection based on MapReduce for biomarker discovery," in *International Conference on Computer Vision and Image Analysis Applications*, IEEE, pp. 1–11.

[350] M. A. Alshammari and E.-S. M. El-Alfy, "MapReduce implementation for minimum reduct using parallel genetic algorithm," in *2015 6th International Conference on Information and Communication Systems (ICICS)*, Amman, Jordan: IEEE, pp. 13–18.

[351] Z. Yu, X. Yu, Y. Chen, and K. Ma, "Distributed Top-k Keyword Search over Very Large Databases with MapReduce," in *2016 IEEE International Congress on Big Data (BigData Congress)*, San Francisco, CA, USA: IEEE, pp. 349–352.

[352] C. Li, T. Wen, H. Dong, Q. Wu, and Z. Zhang, "Implementation of parallel multi-objective artificial bee colony algorithm based on spark platform," in *2016 11th International Conference on Computer Science & Education (ICCSE)*, Nagoya, Japan: IEEE, pp. 592–597.

[353] W. Hu and Y. Tan, "Partitioning Based N-Gram Feature Selection for Malware Classification," in *Data Mining and Big Data*, ser. Lecture Notes in Computer Science, T. Ying and S. Yuhui, Eds., vol. 9714, Bali, Indonesia: Springer, Cham, pp. 187–195.

[354] VX-Heaven: Virus collection, [Online]. Available: https://vxheaven.org/vl.php (visited on 07/24/2017).

[355] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288,

[356] E. A. Horel, "Application of ridge analysis to regression problems," *Chemical Engineering Progress*, vol. 58, pp. 54–59,

[357] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188,

[358] G. McLachlan, *Discriminant analysis and statistical pattern recognition*. John Wiley & Sons, vol. 544.

[359] T. R. Knapp, “Canonical correlation analysis: A general parametric significance-testing system.,” *Psychological Bulletin*, vol. 85, no. 2, pp. 410–416,

[360] R. B. Cattell, “Factor analysis: an introduction and manual for the psychologist and social scientist.,”

[361] H. O. Hirschfeld and J. Wishart, “A Connection between Correlation and Contingency,” *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 31, no. 04, pp. 520–524,

[362] P. E. Green and V. Srinivasan, “Conjoint Analysis in Consumer Research: Issues and Outlook,” *Journal of Consumer Research*, vol. 5, no. 2, pp. 103–123,

[363] J. H. Jr. Ward, “Hierarchical Grouping to Optimize an Objective Function,” *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236–244,

[364] V. Ravi and P. Singh, “Auto-associative extreme learning factory as a single class classifier,” in *2014 IEEE International Conference on Computational Intelligence and Computing Research, IEEE ICCIC 2014*, Coimbatore, India: IEEE, pp. 985–990.

[365] D. M. Tax and R. P. Duin, “Uniform Object Generation for Optimizing One-class Classifiers,” *Journal of Machine Learning Research*, vol. 2, no. Dec, pp. 155–173,

[366] J. Strackeljan, S. Goreczka, and D. Behr, “A fault detection concept for single class problems,” in *The Ninth International Conference on Condition Monitoring and Machinery Failure Prevention Technologies*, London, UK.

[367] J. Mourão-Miranda, D. R. Hardoon, T. Hahn, A. F. Marquand, S. C. Williams, J. Shawe-Taylor, and M. Brammer, "Patient classification as an outlier detection problem: An application of the One-Class Support Vector Machine," *NeuroImage*, vol. 58, no. 3, pp. 793–804,

[368] M. Pandey and V. Ravi, "Phishing Detection Using PSOAANN Based One-Class Classifier," in *2013 6th International Conference on Emerging Trends in Engineering and Technology*, Nagpur, India: IEEE, pp. 148–153.

[369] V. Ravi, N. Nekuri, and M. Das, "Particle Swarm Optimization Trained Auto Associative Neural Networks Used as Single Class Classifier," in *Swarm, Evolutionary, and Memetic Computing*, B. K. Panigrahi, S. Das, P. N. Suganthan, and P. K. Nanda, Eds., vol. 7677, Bhubaneswar, India: Springer, Berlin, Heidelberg, pp. 577–584.

[370] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. B. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, "MLlib: Machine Learning in Apache Spark," Tech. Rep., pp. 1–7.

[371] E. R. Sparks, A. Talwalkar, V. Smith, J. Kottalam, X. Pan, J. Gonzalez, M. J. Franklin, M. I. Jordan, and T. Kraska, "MLI: An API for Distributed Machine Learning," in *2013 IEEE 13th International Conference on Data Mining*, Dallas, TX, USA: IEEE, pp. 1187–1192.

[372] P. Mcneil, S. Shetty, D. Guntu, and G. Barve, "SCREDENT: Scalable Real-time Anomalies Detection and Notification of Targeted Malware in Mobile Devices," *Procedia - Procedia Computer Science*, vol. 83, pp. 1219–1225,

[373] G. Bello-Orgaz, J. J. Jung, and D. Camacho, "Social big data: Recent achievements and new challenges," *Information Fusion*, vol. 28, pp. 45–59,

[374] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on*, vol. 1, Oakland, CA, USA, pp. 281–297.

[375] Q. Song and N. Kasabov, "ECM — A Novel On-line, Evolving Clustering Method and Its Applications," *Foundations of cognitive science*, pp. 631–682,

[376] D. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576,

[377] M. T. Leung, A.-S. Chen, and H. Daouk, "Forecasting exchange rates using general regression neural networks," *Computers & Operations Research*, vol. 27, no. 11-12, pp. 1093–1110,

[378] K. Kayaer and T. Yildirim, "Medical diagnosis on Pima Indian diabetes using general regression neural networks," in *Proceedings of the international conference on artificial neural networks and neural information processing (ICANN/ICONIP)*, pp. 181–184.

[379] C. Li, A. C. Bovik, and X. Wu, "Blind Image Quality Assessment Using a General Regression Neural Network," *IEEE Transactions on Neural Networks*, vol. 22, no. 5, pp. 793–799,

[380] H.-z. Li, S. Guo, C.-j. Li, and J.-q. Sun, "A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm," *Knowledge-Based Systems*, vol. 37, pp. 378–387,

[381] R. Mohanty, V. Ravi, and M. R. Patra, "Software Reliability Prediction Using Group Method of Data Handling," in *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing. RSFDGrC 2009*, ser. Lecture Notes in Computer Science, H. Sakai, M. K. Chakraborty, A. E. Hassanien, D. Ślęzak, and W. Zhu, Eds., vol. 5908, Delhi, India: Springer, Berlin, Heidelberg, pp. 344–351.

[382] D. Pradeepkumar and V. Ravi, "Forecasting financial time series volatility using Particle Swarm Optimization trained Quantile Regression Neural Network," *Applied Soft Computing*, vol. 58, pp. 35–52,

[383] N. Raj Kiran and V. Ravi, "Software reliability prediction by soft computing techniques," *Journal of Systems and Software*, vol. 81, no. 4, pp. 576–583,

[384] V. Kamini, V. Ravi, and D. N. Kumar, "Chaotic time series analysis with neural networks to forecast cash demand in ATMs," in *2014 IEEE International Conference on Computational Intelligence and Computing Research*, Coimbatore, India: IEEE, pp. 1–5.

[385] V. Ravi, D. Pradeepkumar, and K. Deb, "Financial time series prediction using hybrids of chaos theory, multi-layer perceptron and multi-objective evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 36, pp. 136–149,

[386] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109–118,

[387] M. F. Othman and M. A. M. Basri, "Probabilistic Neural Network for Brain Tumor Classification," in *2011 Second International Conference on Intelligent Systems, Modelling and Simulation*, Kuala Lumpur, Malaysia: IEEE, pp. 136–138.

[388] J. Virmani, N. Dey, and V. Kumar, "PCA-PNN and PCA-SVM Based CAD Systems for Breast Density Classification," in *Applications of Intelligent Optimization in Biology and Medicine*, ser. Intelligent Systems Reference Library (ISRL), A. Hassanien, C. Grosan, and M. F. Tolba, Eds., vol. 96, Springer, Cham, ch. 7, pp. 159–180.

[389] W. P. Sweeney, M. T. Musavi, and J. N. Guidi, "Classification of chromosomes using a probabilistic neural network," *Cytometry*, vol. 16, no. 1, pp. 17–24,

[390] J. Lozano, M. Fernández, J. Fontecha, M. Aleixandre, J. Santos, I. Sayago, T. Arroyo, J. Cabellos, F. Gutiérrez, and M. Horrillo, "Wine classification with a zinc oxide SAW sensor array," *Sensors and Actuators B: Chemical*, vol. 120, no. 1, pp. 166–171,

[391] F. Mo and W. Kinsner, "Probabilistic neural networks for power line fault classification," in *Conference Proceedings. IEEE Canadian Conference on Electrical and Computer Engineering (Cat. No.98TH8341)*, vol. 2, Waterloo, Ontario, Canada: IEEE, pp. 585–588.

[392] K. J. Nishanth and V. Ravi, "Probabilistic neural network based categorical data imputation," *Neurocomputing*, vol. 218, pp. 17–25,

[393] P. Ravisankar, V. Ravi, G. Raghava Rao, and I. Bose, "Detection of financial statement fraud and feature selection using data mining techniques," *Decision Support Systems*, vol. 50, no. 2, pp. 491–500,

[394] R. Mohanty, V. Ravi, and M. Patra, "Web-services classification using intelligent techniques," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5484–5490,

[395] V. Ravi, H. Kurniawan, P. N. K. Thai, and P. R. Kumar, "Soft computing system for bank performance prediction," *Applied Soft Computing*, vol. 8, no. 1, pp. 305–315,

[396] P. Ravisankar, V. Ravi, and I. Bose, "Failure prediction of dotcom companies using neural network–genetic programming hybrids," *Information Sciences*, vol. 180, no. 8, pp. 1257–1267,

[397] K. J. Nishanth, V. Ravi, N. Ankaiah, and I. Bose, "Soft computing based imputation and hybrid data and text mining: The case of predicting the severity of phishing alerts," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10 583–10 589,

[398] G. G. Sundarkumar and V. Ravi, "A novel hybrid undersampling method for mining unbalanced datasets in banking and insurance," *Engineering Applications of Artificial Intelligence*, vol. 37, pp. 368–377,

[399] R. Ghosh, K. Ravi, and V. Ravi, "A novel deep learning architecture for sentiment classification," in *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, Dhanbad, India: IEEE, pp. 511–516.

[400] A. C. D. De Souza and M. A. C. Fernandes, "Parallel Fixed Point Implementation of a Radial Basis Function Network in an FPGA," *Sensors*, vol. 14, no. 10, pp. 18 223–18 243,

[401] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium*

on Discrete algorithms, Society for Industrial and Applied Mathematics, pp. 1027–1035.

[402] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable K-Means++," *Proceedings of the VLDB Endowment*, vol. 5, no. 7, pp. 622–633, arXiv: 1203.6402.

[403] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *KDD workshop on text mining*, Boston, vol. 400, pp. 525–526.

[404] Big Data, [Online]. Available: https://www.gartner.com/en/information-technology/glossary/big-data (visited on 01/14/2020).

[405] F. Castells, P. Laguna, L. Sörnmo, A. Bollmann, and J. M. Roig, "Principal component analysis in ECG signal processing," *Eurasip Journal on Advances in Signal Processing*, vol. 2007, pp. 1–21,

[406] N. Le Bihan and S. J. Sangwine, "Quaternion principal component analysis of color images," in *IEEE International Conference on Image Processing*, vol. 1, Barcelona, Spain: IEEE, pp. 809–812.

[407] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243,

[408] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, no. 1, pp. 53–58,

[409] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113,

[410] ——, "Mapreduce: A flexible data processing tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77,

[411] H.-c. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker, "Map-reduce-merge: Simplified relational data processing on large clusters," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, ACM, pp. 1029–1040.

[412] L. Wang, J. Tao, H. Marten, A. Streit, S. U. Khan, J. Kolodziej, and D. Chen, "MapReduce across distributed clusters for data-intensive applications," in *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2012*, Shanghai, China: IEEE, pp. 2004–2011.

[413] J. Urbani, J. Maassen, N. Drost, F. Seinstra, and H. Bal, "Scalable RDF data compression with MapReduce," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 1, pp. 24–39,

[414] N. K. Alham, M. Li, Y. Liu, and S. Hammoud, "A MapReduce-based distributed SVM algorithm for automatic image annotation," in *Computers and Mathematics with Applications*, vol. 62, Elsevier Ltd., pp. 2801–2811.

[415] X. Deng and Y. Li, "MapReduce based Betweenness Approximation Engineering in Large Scale Graph," *Systems Engineering Procedia*, vol. 5, pp. 162–167,

[416] J. Xin, Z. Wang, L. Qu, and G. Wang, "Elastic extreme learning machine for big data classification," *Neurocomputing*, vol. 149, no. Part A, pp. 464–471,

[417] G. Huang, G. B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Networks*, vol. 61, pp. 32–48,

[418] P. Courrieu, "Fast computation of moore-penrose inverse matrices," *arXiv preprint arXiv:0804.4809*,

[419] NLPCA - nonlinear PCA - auto-associative neural networks - autoencoder bottleneck neural networks - Matthias Scholz, [Online]. Available: http://www.nlpca.org/ (visited on 11/03/2019).

[420] A. R. Benson. Matrix computations using dumbo python, [Online]. Available: https://github.com/arbenson/mrtsqr/tree/master/dumbo (visited on 01/15/2020).

[421] A. R. Benson, D. F. Gleich, and J. Demmel, "Direct qr factorizations for tall-and-skinny matrices in mapreduce architectures," in *2013 IEEE international conference on big data*, IEEE, pp. 264–272.

[422] J. Fonollosa, S. Sheik, R. Huerta, and S. Marco, "Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring," *Sensors and Actuators B: Chemical*, vol. 215, pp. 618–629,

[423] (2015). Gas sensor array under dynamic gas mixtures Data Set, [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+under+dynamic+gas+mixtures (visited on 07/02/2018).

[424] M. Schroeck, R. Shockley, J. Smart, D. Romero-Morales, and P. Tufano, "Analytics: The real-world use of big data: How innovative enterprises extract value from uncertain data, executive report," *IBM Institute for Business Value and Said Business School at the University of Oxford*,

[425] Paramjeet, V. Ravi, N. Nekuri, and C. R. Rao, "Privacy preserving data mining using particle swarm optimisation trained auto-associative neural network: An application to bankruptcy prediction in banks," *International Journal of Data Mining, Modelling and Management*, vol. 4, no. 1, pp. 39–56,

[426] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan: IEEE, pp. 39–43.

[427] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia Weight strategies in Particle Swarm Optimization," in *2011 Third World Congress on Nature and Biologically Inspired Computing*, Salamanca, Spain: IEEE, pp. 633–640.

[428] Y. Maheshkumar, V. Ravi, and A. Abraham, "A particle swarm optimization-threshold accepting hybrid algorithm for unconstrained optimization," *Neural Network World*, vol. 23, no. 3, p. 191,

[429] F. Murtagh, "A Survey of Recent Advances in Hierarchical Clustering Algorithms," *The Computer Journal*, vol. 26, no. 4, pp. 354–359,

[430] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data, An Introduction to Cluster Analysis*, ser. Wiley Series in Probability and Mathematical Statistics. New York: John Wiley & Sons, Inc.

[431] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Kdd*, vol. 96, no. 34, pp. 226–231,

[432] W. Wang, J. Yang, and R. Muntz, "STING: A statistical information grid approach to spatial data mining," *VLDB*, vol. 97, pp. 186–195,

[433] J. D. Banfield and A. E. Raftery, "Model-Based Gaussian and Non-Gaussian Clustering," *Biometrics*, vol. 49, no. 3, pp. 803–821,

[434] J. C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57,

[435] A. Inselberg, "Parallel Coordinates," in *Encyclopedia of Database Systems*, LING LIU and M. TAMER ÖZSU, Eds., Boston, MA: Springer US, pp. 2018–2024.

[436] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, no. 11, pp. 559–572,

[437] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for Exotic Particles in High-Energy Physics with Deep Learning," *NATURE COMMUNICATIONS*, vol. 5, no. 4308, pp. 1–9, arXiv: 1402.4735.

[438] D. S. Broomhead and D. Lowe. (1988). Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks, [Online]. Available: https://apps.dtic.mil/docs/citations/ADA196234.

[439] Q. Zhang and A. Benveniste, "Wavelet networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 889–898,

[440] A. G. Ivakhnenko, "The Group Method of Data of Handling ; A rival of the method of stochastic approximation," *Soviet Automatic Control*, vol. 13, pp. 43–55,

[441] W. Kusakunniran, Q. Wu, J. Zhang, and H. Li, “Multi-view Gait Recognition Based on Motion Regression Using Multilayer Perceptron,” in *2010 20th International Conference on Pattern Recognition*, Istanbul, Turkey: IEEE, pp. 2186–2189.

[442] E. Agirre-Basurko, G. Ibarra-Berastegi, and I. Madariaga, “Regression and multilayer perceptron-based models to forecast hourly O3 and NO2 levels in the Bilbao area,” *Environmental Modelling & Software*, vol. 21, no. 4, pp. 430–446,

[443] J. Gaudart, B. Giusiano, and L. Huiart, “Comparison of the performance of multi-layer perceptron and linear regression for epidemiological data,” *Computational Statistics & Data Analysis*, vol. 44, no. 4, pp. 547–570,

[444] A. Mignon and F. Jurie, “Reconstructing faces from their signatures using RBF regression,” in *Proceedings of the British Machine Vision Conference 2013*, Bristol, UK: British Machine Vision Association, pp. 103.1–103.11.

[445] S. A. Hannan, R. R. Manza, and R. J. Ramteke, “Generalized Regression Neural Network and Radial Basis Function for Heart Disease Diagnosis,” *International Journal of Computer Applications*, vol. 7, no. 13, pp. 7–13,

[446] M. Taki, A. Rohani, F. Soheili-Fard, and A. Abdeshahi, “Assessment of energy consumption and modeling of output energy for wheat production by neural network (MLP and RBF) and Gaussian process regression (GPR) models,” *Journal of Cleaner Production*, vol. 172, pp. 3028–3041,

[447] K. Budu, “Comparison of Wavelet-Based ANN and Regression Models for Reservoir Inflow Forecasting,” *Journal of Hydrologic Engineering*, vol. 19, no. 7, pp. 1385–1400,

[448] K. Vinaykumar, V. Ravi, M. Carr, and N. Rajkiran, “Software development cost estimation using wavelet neural networks,” *Journal of Systems and Software*, vol. 81, no. 11, pp. 1853–1867,

[449] N. Chauhan, V. Ravi, and D. Karthik Chandra, “Differential evolution trained wavelet neural networks: Application to bankruptcy prediction in banks,” *Expert Systems with Applications*, vol. 36, no. 4, pp. 7659–7665,

[450] N. Rajkiran and V. Ravi, “Software Reliability Prediction Using Wavelet Neural Networks,” in *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, Sivakasi, Tamil Nadu, India: IEEE, pp. 195–199.

[451] V. P. Astakhov and V. V. Galitsky, “Tool life testing in gundrilling: an application of the group method of data handling (GMDH),” *International Journal of Machine Tools and Manufacture*, vol. 45, no. 4-5, pp. 509–517,

[452] E. E. Elattar, J. Y. Goulermas, and Q. H. Wu, “Generalized Locally Weighted GMDH for Short Term Load Forecasting,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 3, pp. 345–356,

[453] D. Srinivasan, “Energy demand prediction using GMDH networks,” *Neurocomputing*, vol. 72, no. 1-3, pp. 625–629,

[454] P. Ravisankar and V. Ravi, “Financial distress prediction in banks using Group Method of Data Handling neural network, counter propagation neural network and fuzzy ARTMAP,” *Knowledge-Based Systems*, vol. 23, no. 8, pp. 823–831,

[455] K. N. Reddy and V. Ravi, “Kernel Group Method of Data Handling: Application to Regression Problems,” in *Swarm, Evolutionary, and Memetic Computing. SEMCCO 2012*, ser. Lecture Notes in Computer Science, B. K. Panigrahi, S. Das, P. N. Suganthan, and P. K. Nanda, Eds., vol. 7677, Bhubaneswar, India: Springer, Berlin, Heidelberg, pp. 74–81.

[456] C. Gautam and V. Ravi, “Counter propagation auto-associative neural network based data imputation,” *Information Sciences*, vol. 325, pp. 288–299,

[457] R. Hecht-Nielsen, “Applications of counterpropagation networks,” *Neural Networks*, vol. 1, no. 2, pp. 131–139,

[458] N. Ahad, J. Qadir, and N. Ahsan, “Neural networks in wireless networks: Techniques, applications and guidelines,” *Journal of Network and Computer Applications*, vol. 68, pp. 1–27,

[459] L. Jin, S. Li, J. Yu, and J. He, “Robot manipulator control using neural networks: A survey,” *Neurocomputing*, vol. 285, pp. 23–34,

[460] A. P. Marugán, F. P. G. Márquez, J. M. P. Perez, and D. Ruiz-Hernández, “A survey of artificial neural network in wind energy systems,” *Applied Energy*, vol. 228, pp. 1822–1836,

[461] S. Agrawal and J. Agrawal, “Neural Network Techniques for Cancer Prediction: A Survey,” *Procedia Computer Science*, vol. 60, pp. 769–774,

[462] A. Khoshroo, A. Emrouznejad, A. Ghaffarizadeh, M. Kasraei, and M. Omid, “Sensitivity analysis of energy inputs in crop production using artificial neural networks,” *Journal of Cleaner Production*, vol. 197, Part, pp. 992–998,

[463] M. Tkáč and R. Verner, “Artificial neural networks in business: Two decades of research,” *Applied Soft Computing*, vol. 38, pp. 788–804,

[464] J. C. Bezdek and N. R. Pal, “Some new indices of cluster validity,” *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 28, no. 3, pp. 301–315,

[465] M. Sponder and G. F. Khan, *Digital analytics for marketing*. Routledge.

[466] R. Kohli and V. Grover, “Business value of it: An essay on expanding research directions to keep up with the times,” *Journal of the association for information systems*, vol. 9, no. 1, p. 1,

[467] W. Chamlertwat, P. Bhattarakosol, T. Rungkasiri, and C. Haruechaiyasak, “Discovering consumer insight from twitter via sentiment analysis.,” *J. UCS*, vol. 18, no. 8, pp. 973–992,

[468] W. G. Mangold and D. J. Faulds, “Social media: The new hybrid element of the promotion mix,” *Business horizons*, vol. 52, no. 4, pp. 357–365,

[469] H. Baars and H.-G. Kemper, “Management support with structured and unstructured data—an integrated business intelligence framework,” *Information Systems Management*, vol. 25, no. 2, pp. 132–148,

[470] Y. Mejova, "Sentiment analysis: An overview," *University of Iowa, Computer Science Department*,

[471] B. Pang, L. Lee, *et al.*, "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135,

[472] A. Westerski, "Sentiment analysis: Introduction and the state of the art overview," *Universidad Politecnica de Madrid*, pp. 211–218,

[473] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews," in *Proceedings of the 22nd international conference on World Wide Web*, ACM, pp. 897–908.

[474] SNAP: Web data: Amazon reviews, [Online]. Available: http : / / snap . stanford.edu/data/web-Amazon-links.html (visited on 07/23/2017).

[475] D.-T. Lin, "Facial expression classification using PCA and hierarchical radial basis function network," *Journal of information science and engineering*, vol. 22, no. 5, pp. 1033–1046,

[476] V. Radha and N. Nallammal, "Neural network based face recognition using RBFN classifier," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, San Francisco, USA, pp. 19–21.

[477] A. Subasi, M. Yilmaz, and H. R. Ozcalik, "Classification of EMG signals using wavelet neural network," *Journal of Neuroscience Methods*, vol. 156, no. 1-2, pp. 360–367,

[478] M. Geethanjali and K. S. Priya, "Combined wavelet transfoms and neural network (WNN) based fault detection and classification in transmission lines," in *Control, Automation, Communication and Energy Conservation, 2009. INCACEC 2009, 2009 International Conference on.*, Perundurai, Tamilnadu, India: IEEE, pp. 1–7.

[479] Z. A. Baig, S. M. Sait, and A. Shaheen, "GMDH-based networks for intelligent intrusion detection," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 7, pp. 1731–1740,

[480] E.-S. M. El-Alfy and R. E. Abdel-Aal, "Using GMDH-based networks for improved spam detection and email feature analysis," *Applied Soft Computing*, vol. 11, no. 1, pp. 477–488,

[481] UCI Machine Learning Repository: HEPMASS Data Set, [Online]. Available: http : / / archive . ics . uci . edu / ml / datasets / hepmass (visited on 02/15/2019).

[482] UCI Machine Learning Repository: HIGGS Data Set, [Online]. Available: http://archive.ics.uci.edu/ml/datasets/HIGGS (visited on 02/15/2019).

[483] K. Ravi, V. Ravi, and B. Shivakrishna, "Sentiment Classification Using Paragraph Vector and Cognitive Big Data Semantics on Apache Spark," in *2018 IEEE 17th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, Berkeley, CA, USA: IEEE, pp. 187–194.

[484] J. Moody and C. J. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, vol. 1, no. 2, pp. 281–294,

[485] L. Nikolaos, "Radial basis function networks to hybrid neuro-genetic rbfns in financial evaluation of corporations," *International Journal of Computers*, vol. 2, no. 2, pp. 176–183,

[486] H.-M. Feng and H.-C. Chou, "Evolutional rbfns prediction systems generation in the applications of financial time series data," *Expert Systems with Applications*, vol. 38, no. 7, pp. 8285–8292,

[487] V. Ravi, P. R. Kumar, E. R. Srinivas, and N. K. Kasabov, "A semi-online training algorithm for the radial basis function neural networks: Applications to bankruptcy prediction in banks," in *Advances in Banking Technology and Management: Impacts of ICT and CRM*, IGI Global, pp. 243–260.

[488] N. Naveen, V. Ravi, C. R. Rao, and N. Chauhan, "Differential evolution trained radial basis function network: Application to bankruptcy prediction in banks," *International Journal of Bio-Inspired Computation*, vol. 2, no. 3-4, pp. 222–232,

[489] N. Acır, "Automated system for detection of epileptiform patterns in eeg by using a modified rbfn classifier," *Expert Systems with Applications*, vol. 29, no. 2, pp. 455–462,

[490] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster Computing with Working Sets," *HotCloud*, vol. 10, no. 10-10, p. 95,

[491] Cluster Mode Overview - Spark 2.1.0 Documentation, [Online]. Available: https://spark.apache.org/docs/2.1.0/cluster-overview.html (visited on 09/27/2017).

[492] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, *Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing*.

[493] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mccauley, M. Franklin, S. Shenker, and I. Stoica, "Fast and interactive analytics over hadoop data with spark," *Usenix Login*, vol. 37, no. 4, pp. 45–51,

[494] Big Data Processing with Apache Spark – Part 1: Introduction, [Online]. Available: https://www.infoq.com/articles/apache-spark-introduction (visited on 07/19/2017).

[495] R. R. Patil and A. Khan, "Bisecting K-Means for Clustering Web Log data," *International Journal of Computer Applications*, vol. 116, no. 19, pp. 36–41,

[496] Clustering - RDD-based API - Spark 2.2.0 Documentation, [Online]. Available: https://spark.apache.org/docs/2.2.0/mllib-clustering.html%7B%5C#%7Dbisecting-k-means (visited on 02/14/2019).

[497] M. A. Bhuiyan and M. A. Hasan, "FSM-H: Frequent Subgraph Mining Algorithm in Hadoop," in *2014 IEEE International Congress on Big Data*, Anchorage, AK, USA: IEEE, pp. 9–16.

APPENDICES

Appendix A

Hadoop and Apache Spark: Introduction

A.1 Hadoop MapReduce Vs. Apache Spark

Hadoop MapReduce (MR) [409] is difficult to program and needs abstraction, whereas Spark is easy to program and does not require any abstraction. The Spark has an interactive mode, whereas Hadoop MR does not possess interactive mode except Pig and Hive. Hadoop MR processes data in batch mode and produces reports for answering the queries on historical data. On the other hand, Spark can handle streaming data and allows us to modify the data in real-time. Hadoop MR has more latency as the partial results are stored in the disk. In the case of Spark, it uses primary memory for caching partial results across the memory of distributed workers, which helps in faster execution. Zaharia et al. [490] claimed that Hadoop falls behind Spark by a factor of 10 in iterative machine learning workloads. A machine learning algorithm involves the iterative and interactive mode of execution. In this case, Hadoop falls behind in the latency of execution time in comparison to Spark.

A.2 Apache Spark:Introduction

Apache Spark is an open-source distributed in-memory computational framework for data analytics in the big data paradigm utilizing a cluster of commodity hardware, i.e., a group of affordable low-performance systems. A Cluster is a group of machines connected to LAN and communicating through Secure Shell (SSH). Zaharia [490] developed Spark at UC Berkeley's AMPLab in 2009. It was an academic project at UC Berkley. It provides iterative, interactive, and scalable computational capabilities, especially for machine learning. In the year 2013, the spark project was passed on to the Apache Software Foundation.

It provides high-level APIs in Java, Scala, Python, and R and also interactively can be used from Scala, Python, and R shells. Spark can run on a single machine as well as over several machines with existing cluster managers. Spark has the following options for deployment: (i) Cloud deployment, (ii) Standalone Deploy Mode, (iii) Apache Mesos, and (iv) Hadoop YARN. It can access diverse data sources, including Hadoop Distributed File System (HDFS), Cassandra, HBase, and Amazon S3 (Simple Storage Service) [491].

The Spark is comparably advantageous than other big data analytical technologies like Hadoop and Storm, employing the MapReduce framework. Spark is faster than MapReduce and offers low latency due to reduced disk input and output operation. Spark has the capability of in-memory computation on a distributed environment with fault-tolerance, which makes the data processing faster than MapReduce.

Unlike Hadoop, Spark maintains the intermediate results in memory rather than writing every intermediate output to disk. This operation hugely cuts down the execution time of the job, resulting in faster execution. When data crosses the threshold of the memory storage, it is spilled to the disk.

Spark uses data abstraction through the use of Resilient Distributed Datasets (RDDs) for data processing [492], which is a distributed memory abstraction. RDDs present an influential role in two types of applications viz. iterative algorithms and interactive approach. Other computing frameworks like Hadoop MR does not provide the same. Each RDD is having five pieces of information that can be accessed through a common interface. First, a set of partitions, which are atomic pieces of the dataset. Second, the preferred location for a partition, which is required for faster

access due to data locality. Third, a set of dependencies on parent RDDs, which is needed for computation on RDDs. Fourth, an iterator, which is required for the computation of elements in the partition with the help of iterators of parent RDDs. Finally, the fifth one is metadata about the partitioning scheme and data placement [492]. The concept of RDD provides the parallelization of the algorithm through the partitions of data. Apart from providing in-memory storage, RDDs can also automatically recover from failures. Each RDD tracks the graph of transformations that were used to build it, called its lineage graph. These lineage graphs help in the reconstruction of any lost partitions through the re-execution of operations on base data [493]. Spark doesn't execute the tasks immediately but maintains a chain of operations as meta-data of the job called DAG (Directed Acyclic Graph), which is the lineage graph and is due to the transformation operation. The action on the DAG happens only when an 'action' operation is called on. This process is called lazy evaluation. The lazy evaluation allows optimized execution of the queries on Big Data [494].

A.3 Parallel and Distributed Computation in Spark

Spark has an inherent feature of executing iterative programs and interactive mode of execution for user-friendly data analysis. Figure A.1 depicts the components of the Spark computational environment. The user interacts with the top layer through the computing interface. The top layer provides the usage of different APIs for the Spark application. The Spark applications interact with a cluster manager for accessing the data from the distributed data storage.

A Spark cluster has two main components: *master node* and *worker nodes*. Each machine in the cluster is known as a *node*. There is a single *master node* and more than one *worker nodes* in a cluster. The system, which is the master node, can also serve as a worker node. The *master node* allocates *jobs* to the *worker nodes*. A distributed file system, e.g., HDFS, a cloud storage system, e.g., S3 or a local file system, is used for data storage. Spark can run on a single-node cluster setup having both *master node* and the *worker node* on the same machine. It also runs in a multi-node cluster setup. Spark applications run as independent sets of processes

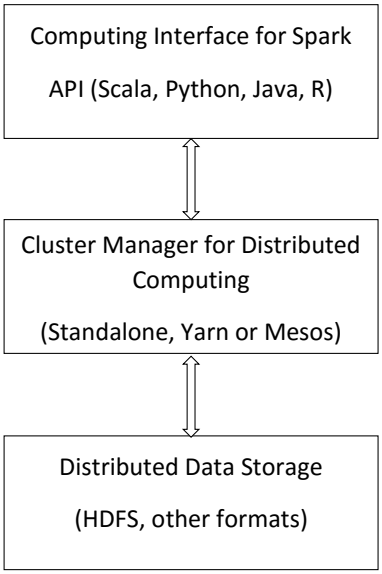

Figure A.1: *Components of Spark Computational Environment*

on a cluster. The Spark applications coordinate among themselves using a SparkContext object in the main program, which is also known as the *driver program*. For running Spark on a cluster, the SparkContext connects to one of the cluster managers. A Spark cluster can have several types of cluster managers (i.e., either Spark's standalone cluster manager, Mesos or YARN), which allocate resources across applications. When the SparkContext is connected with the cluster manager, Spark acquires *executors* on *worker nodes* of the cluster. Here, an *executor* is a process that performs computations and storage operations for an application. Next, it sends application code to the executors. Finally, SparkContext assigns *tasks* to the *executors* to run [491]. The cluster components are presented in Figure A.3.

A *job* is a part of the code in the Spark application which takes inputs from HDFS/local file system, performs computations on them. A *job* writes outputs to an HDFS/local file system. Figure A.2 depicts how a *job* is distributed on a Spark Cluster. A *driver process* runs on the master node and executes a *job* over the Spark Engine. Each *job* is split into a number of *stages* which can be either *map* or *reduce* stages. One *stage* may be dependent on the outcomes of a previous *stage*. So the *stages* are executed sequentially. Each *stage* comprises several *tasks* which can run in parallel. The data is split and spread over the cluster in several *partitions*. The

computations of one *stage* are performed on each *partition* in the form of a *task*. These *tasks* are executed as *processes* running in parallel over the *worker nodes* by the *executor* processes. Only one *task* is performed on one *partition* on each *executor* [298].

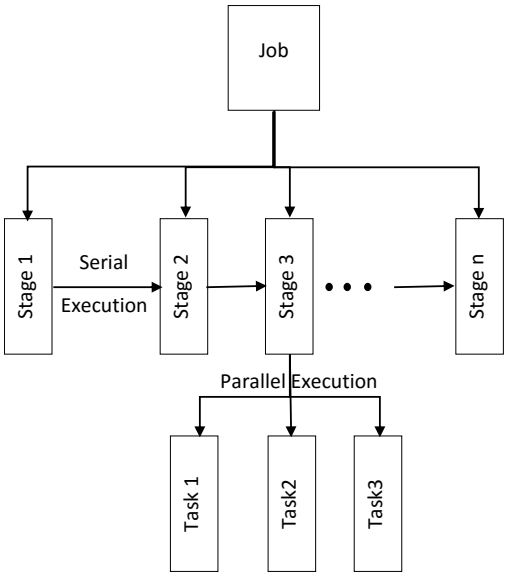

Figure A.2: *Execution of Job, Stage, and Task in Apache Spark*

A.4 Spark Features

Apache Spark has other features, such as:

1. It supports a wide variety of operations compared to Map and Reduce functions.

2. It presents a compact and stable Application Programming Interface (API) in the programming languages such as Scala, Java, and Python.

3. Spark is written in Scala Programming Language and runs in Java Virtual Machine (JVM).

4. An application can be developed employing any of the following programming languages: Scala, Java, Python, and R; in Spark.

5. It provides an interactive programming interface called shell for Scala and Python.
6. It leverages the distributed cluster memory for doing computations for increased speed and data processing.
7. It runs on top of the existing Hadoop cluster and access HDFS; it can also process data stored by HBase structure. It employs three different cluster managers for managing the resources of cluster viz. Yet Another Resource Negotiator (YARN) in Hadoop, Apache Mesos, and standalone mode.
8. Apache Spark can be integrated with various data sources like SQL, NoSQL, S3, HDFS, local file system, etc.
9. It is a good fit for iterative tasks like Machine Learning (ML) algorithms.
10. Apart from MapReduce computational framework, it supports SQL-like queries, streaming data, machine learning, and graph analysis.

A.5 Apache Spark Components and Architecture

Multiple applications run in Spark with independent resources and processes on a cluster. The main program or the driver program contains an object called SparkContext, which coordinates the applications.

For running an application on a cluster, the SparkContext connects to the cluster manager (i.e., either YARN, or Mesos, or standalone). The cluster manager is responsible for resource allocation across applications. But, the standalone cluster can manage a single application only. Once Spark is connected to the cluster manager through the SparkContext, it acquires executors and resources for the executors on the worker nodes in the cluster. The executors are processes that perform computational work and storage of data for the application. SparkContext sends tasks to the executors to run Figure A.3.

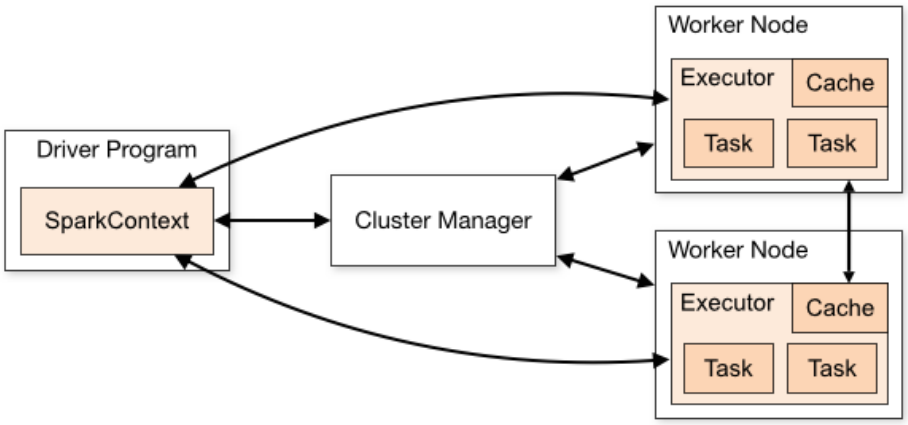

Figure A.3: *Components of Apache Spark Cluster*[491]

Appendix B

Machine Learning Techniques

B.1 K-Means++

The K-Means clustering algorithm chooses a random initial 'K' cluster centers. It updates the cluster centers with iterations until there is no further change in the cluster center and thus finds reasonable solutions quickly. However, it suffers from at least two major drawbacks

i the algorithm has the worst-case execution time which is super-polynomial of input size,

ii the formed clusters may not be optimal with respect to the objective function.

The K-Means++ [401] has addressed the deficiency of optimal clustering by the introduction of a method for cluster center initialization before the execution of the iterations of K-Means algorithm. The K-Means++ clustering algorithm employing an innovative method for initial cluster center selection or seed selection was

proposed by Arthur and Vassilvitskii [401]. The algorithm is as follows:

Algorithm 4: K-Means++ algorithm

Input: dataset D with p points in $\mathbb{R}^q$ and $x \in D$. *d(x)* is the smallest distance from any given data point to its nearest cluster center which has been already selected.

Output: 'K' cluster centers

Data: data distributed in partitions

1 Select first center C_1 uniformly at random from D.

2 For each data point $x \in D$, compute *d(x)* and take a new center C_i, with probability $\frac{d(x)^2}{\sum_{x \varepsilon D} d(x)^2}$.

3 Repeat Step 2 until k centers have been selected all together.

4 Proceed with the iterations of the K-Means.

B.2 K-Means||

The K-Means|| is the parallel implementation of K-Means++. Say, n number of samples are there, and k initial cluster centers are needed. The K-Means++ performs k number of passes to sample one initial cluster center in each pass. But, K-Means|| samples the first center uniformly at random. Then, the next centers will be chosen non-uniformly with a given probability, which is stochastically biased by the already chosen centers. This happens in a parallel way across all partitions. Thus, so obtained $O(k \log n)$ points are finally, re-clustered into k initial centers for the standard K-Means iterations.

B.3 Parallel Bisecting K-Means

Bisecting K-Means [403] is an approach that combines the best features of K-Means and hierarchical clustering, i.e., divisive clustering. Here, instead of partitioning the data into 'k' clusters in each iteration, Bisecting K-Means splits one cluster into two sub-clusters at each bisecting step(by using K-Means) until k clusters are obtained.

Bisecting K-Means is more efficient when 'k' is large. For the K-Means algorithm, the computation involves every data point of the data set and k centroids. But, in Bisecting K-Means, only the data points of one cluster and two centroids are involved in the computation in each bisecting step. Thus, the computation time

is reduced. Bisecting K-Means produce clusters of similar sizes, while K-Means is known to produce clusters of widely different sizes [495].

The algorithm for bisecting K-Means is presented below.

Algorithm 5: Bisecting K-Means algorithm

```
repeat
    Pick a cluster to split.
      /* The following for loop is executed for a fixed `n' number of
     times */
    for i ← 1 to n do
        Find 2 sub-clusters using the basic K-means algorithm (bisecting
          step).
    Take the split that produces the clustering with the highest overall
      similarity.
until the desired number of clusters is reached
```

The Bisecting K-Means has been implemented in a parallel and distributed computational environment by Spark Mllib library [496]. The algorithm starts from a single cluster that contains all points. Iteratively it finds divisible clusters on the bottom level and bisects each of them using K-means until there are 'K' leaf clusters in total or no leaf clusters are divisible. The bisecting steps of clusters on the same level are grouped together to increase parallelism. If bisecting all divisible clusters on the bottom level would result in more than 'K' leaf clusters, larger clusters get higher priority. The parallel version of Bisecting K-Means is referred to as Bisecting K-Means||, in the thesis.

B.4 Generalized Regression Neural Network (GRNN)

GRNN [376] is a feed-forward neural network having roots in statistics. GRNN represents an advanced architecture in the neural networks that implements non-parametric regression with one-pass training.

The topology of GRNN is depicted in Figure B.1 and involves four layers of neurons, viz., input, pattern, summation, and the output.

The pattern layer comprises 'n' training neurons. The test sample is fed to the input layer. The distance, d_i, between the training sample present as a neuron in the

pattern layer and the data point from the test set used for regression, is used to figure out how well each training neuron in pattern layer can represent the feature space of the test sample, X. This probability density is calculated with Gaussian activation function. Thus, the summation of the product of target value and the result of activation function for each neuron i.e., $\sum_{i=1}^{n} Y_i * e^{\left(\frac{-d_i^2}{2\sigma^2}\right)}$ forms the numerator term in the summation layer and the summation of the term $e^{\left(\frac{-d_i^2}{2\sigma^2}\right)}$ i.e., $\sum_{i=1}^{n} e^{\left(\frac{-d_i^2}{2\sigma^2}\right)}$ forms the denominator term in the summation layer. Then, in the output layer, the prediction is calculated as $\hat{Y}(X) = \frac{\sum_{i=1}^{n} Y_i * e^{\left(\frac{-d_i^2}{2\sigma^2}\right)}}{\sum_{i=1}^{n} e^{\left(\frac{-d_i^2}{2\sigma^2}\right)}}$. Here, $d_i^2 = (X - X_i)^T * (X - X_i)$. X is a test sample and X_i is the neuron present in the pattern layer. For big datasets, the equations involved in summation and output layer change to accommodate the information about the cluster centers. For further details please see the training algorithm presented in Algorithm 2.

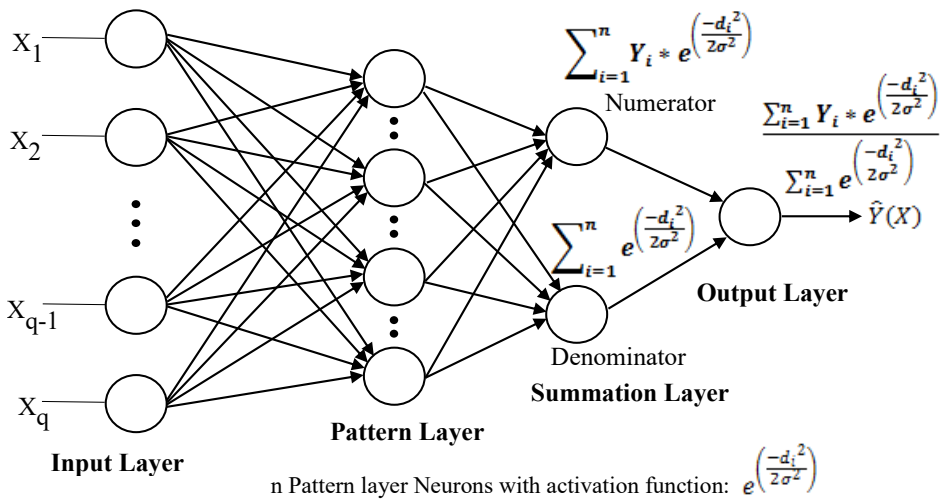

Figure B.1: *Generalized Regression Neural Network: Architecture*

B.5 Probabilistic Neural Network (PNN)

PNN [386] is a feed-forward neural network having roots in statistics. PNN represents an advanced architecture in the neural networks that implements classification with one-pass training.

The topology of PNN is depicted in Figure B.2 and involves four layers of neurons, viz., input, pattern, summation, and the output.

The pattern layer comprises training neurons. The test sample is fed to the input layer. The distance, d_i, between the training sample present as a neuron in the presentation layer and the data point from the test set used for classification, is used to figure out how well each training neuron in presentation layer can represent the feature space of the test sample, X. This probability density is calculated with the Gaussian activation function. Thus, the summation of the value of activation function for each neuron i.e., $\sum_{i=1}^{n} e^{\frac{d_i^2}{2\sigma^2}}$ forms the Probability Density Function (PDF) for one class (say, C_1) in the summation layer and similarly, the summation of the term $e^{\frac{d_i^2}{2\sigma^2}}$ i.e., $\sum_{i=1}^{n} e^{\frac{d_i^2}{2\sigma^2}}$ forms the PDF for the second class (say, C_2) in the summation layer. Then, in the output layer, the classification $\hat{Y}(X)$ is performed by comparing the larger value between the PDFs for both the classes, whichever is larger, the test sample belongs to that class. Here, $d_i^2 = (X - X_i)^T * (X - X_i)$. X is a test sample, and X_i is the neuron present in the pattern layer. If the classification task involves multiple classes, then the PDFs for all the classes are compared, and the largest among them signifies that the sample belongs to that class.

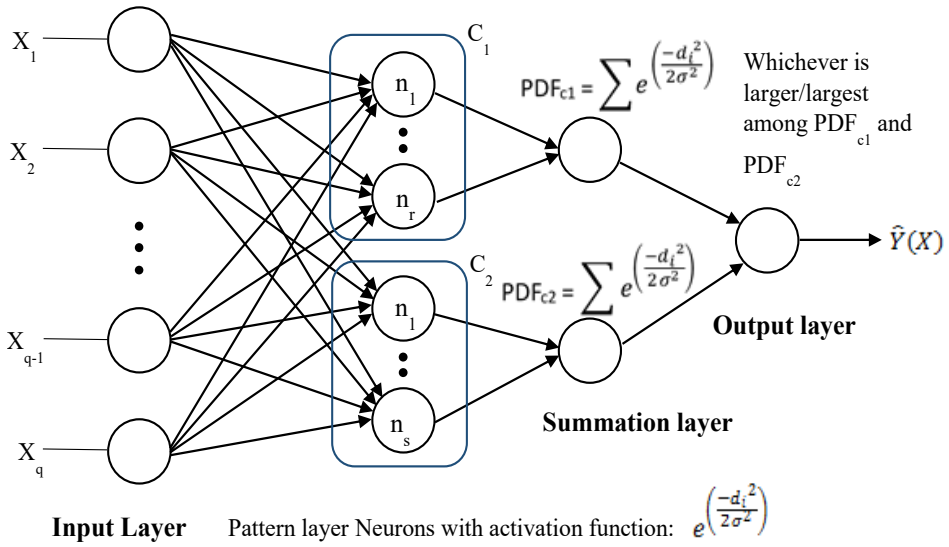

Figure B.2: *Probabilisitic Neural Network: Architecture*

Appendix C

Datasets Analyzed

C.1 Gas Sensor Dataset

The gas sensor dataset contains the readings gathered from sixteen sensors exposed to varying concentrations of gas mixtures. The dataset includes readings of two different mixtures of gas in air, viz., Ethylene with Methane, and Ethylene with CO in two different datasets. The sensor readings were the data from 16-sensor array signals gathered uninterruptedly for about 12 hours.

The Ethylene and CO mixture in Air dataset, and Ethylene and Methane mixture in Air dataset, both contain 19 features. Here, the first feature represents the time of sensor reading; the second feature represents CO and Methane concentration in ppm in former and later datasets respectively, the third feature represents the Ethylene concentration in ppm. The next 16 features represent the sensor readings from 16 chemical sensors. The Ethylene-CO dataset contains 4.2 million samples and is of 643MB. The Ethylene-Methane dataset contains 4.17 million samples and is of 638 MB. The details of the attributes are mentioned in Table C.1 and Table C.2.

C.2 ccFraud Dataset

The ccFraud dataset contains credit card fraud data. The dataset contains 9 features and 10000000 number of instances. The negative class (genuine transactions) has

Table C.1: Attribute details of Ethylene-CO Gas Sensor dataset

Attribute Name	Attribute details
Time (seconds)	Time measured in seconds for the sensor readings.
CO conc. (ppm)	The concentration of CO measured in ppm.
Ethylene conc. (ppm)	The concentration of Ethylene measured in ppm.
Sensor Readings (16 channels)	Readings from 16 chemical sensors forming the data comprising 16 columns.

Table C.2: Attribute details of Ethylene-Methane Gas Sensor dataset

Attribute Name	Attribute details
Time (seconds)	Time measured in seconds for the sensor readings.
Methane conc. (ppm)	The concentration of Methane measured in ppm.
Ethylene conc. (ppm)	The concentration of Ethylene measured in ppm.
Sensor Readings (16 channels)	Readings from 16 chemical sensors forming the data comprising 16 columns.

9,403,986 samples. The positive class (fraudulent transactions) has 596,014 samples. The dataset is of 292 MB. This dataset is quite imbalanced, with a ratio of 94:6 genuine to fraudulent transactions. The details of the attributes are mentioned in Table C.3.

Table C.3: Attribute details of ccFraud dataset

Attribute Name	Attribute details
custID	customer ID, auto-incrementing integer value.
gender	taking two values either 1 or 2 for male and female.
state	state number given as integer.
cardholder	number of cards per customer with a maximum value of 2.
balance	credit balance.
numTrans	number of transactions made in integer.
numIntlTrans	number of international transactions made in integer.
creditLine	credit limit of a customer in integer.
fraudRisk	whether a given transaction is fraud or not (0 = genuine, 1 = fraud)

C.3 HEPMASS Dataset

The HEPMASS dataset contains the high energy physics experiments data to search for the presence of exotic particles. The dataset contains 28 attributes and 10500000 number of instances. The dataset is 7.8 GB. The details of the attributes of the dataset are depicted in Table C.4.

Table C.4: Attribute details of HEPMASS dataset

Attribute Name	Attribute details
# label	Class label (1 for signal, 0 for background).
f0 – f26	27 normalized features (22 low-level features then 5 high-level features).
mass	Mass of the particle.

C.4 HIGGS Dataset

The HIGGS dataset has been produced using Monte Carlo simulations. The dataset contains 28 attributes and 11000000 number of instances. The dataset is a balanced one with 53% positive samples in the dataset. In this, the first feature is the class variable with two values, i.e., 1 and 0 for representing signal and background, respectively. The dataset contains 28 more features following the class variable, out of which the first 21 are low-level features, the next 7 are high-level features. The low-level features are kinematic attributes measured by the particle detectors in the accelerator. The low-level features are mapped to the high-level features. These high-level features are employed to define the class value. The dataset is of 8.0 GB. The attribute details are in Table C.5.

Table C.5: Attribute details of HIGGS dataset

Attribute Name	Attribute details
“V1”	Class label (1 for signal, 0 for background).
“V2” – “V22”	21 low-level features.
“V23” – “V29”	7 high-level features derived from 21 low-level features.

C.5 Amazon Movie Review (AMR) Dataset

The AMR dataset is available on the Stanford Network Analysis Platform (SNAP) data repository [474]. It contains 7,911,684 reviews on 253,059 movies provided by 889,176 users, which were collected from August 1997 to October 2012. The movie review ratings are on five levels of Likert scale. The detailed structure of the dataset is depicted in Table C.6.

Table C.6: Attribute details of Amazon Movie Review dataset

Attribute Name	Attribute details
product/productId	Unique id of the product about which the review is made.
review/userId	Unique id of the user performing the review.
review/profileName	Profile name of the user performing the review.
review/helpfulness	Fraction of users who found the review helpful.
review/score	Review score or ratings in 5 levels of Likert scale.
review/time	System time when the review was made (unix time).
review/summary	A summary of the review.
review/text	Plain text review of the movie.

Appendix D

Summarized Veiw of Literature Review Chapter

D.1 Table for DM task-wise categorization of articles with ML techniques and Performance measure

Table D.1: DM Task-wise categorization of articles describing ML techniques used in it

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
Association Rule Mining/ Pattern Mining	Hadoop-based Articles			
	[21]	2012	FP mining	Not mentioned
	[22]	2013	FP Growth	Interestingness Measure, all-confidence, $\alpha(X) = Sup(X)/Max\_item\_sup(X)$
	[24]	2015	Frequent Subgraph mining	Not mentioned
	[30]	2016	Frequent itemset mining	Not mentioned

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[31]	2017	Apriori	Not mentioned
	[35]	2017	Dynamic rule creation	True Positive Rate (TPR) = TP/(TP + FN) False Positive Rate (FPR) = FP/(FP + TN)
	[38]	2017	FP Growth	Not mentioned
	[39]	2017	Frequent itemset mining	Not mentioned
	[40]	2018	Frequent itemset mining	Execution time
	[42]	2018	Maximal frequent itemsets mining	Exection time
	[43]	2018	Apriori	Execution time
	[29]	2012	Frequent Subgraph mining	Not mentioned
	[55]	2013	Apriori	Not mentioned
	[58]	2013	Apriori	Not mentioned
	[59]	2013	Apriori	Not mentioned
	[60]	2013	FP Tree	Not mentioned
	[61]	2013	Apriori + FP Growth	Not mentioned
	[62]	2013	Apriori + Eclat	Not mentioned
	[63]	2013	FP mining	Not mentioned
	[64]	2014	FP Growth	Not mentioned
	[67]	2014	Apriori	Not mentioned
	[68]	2014	String mining algorithm using suffix array (SA) and the longest common prefix array (LCP)	Not mentioned
	[69]	2014	Apriori	Not mentioned

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[71]	2014	Frequent itemset mining	Not mentioned
	[497]	2014	Frequent Subgraph mining	Not mentioned
	[72]	2014	Apriori	Not mentioned
	[73]	2014	Repetitive Sequence mining	Not mentioned
	[74]	2014	Frequent pattern mining	Not mentioned
	[75]	2015	Apriori + FP Growth	Not mentioned
	[76]	2015	Apriori	Not mentioned
	[77]	2015	Parallel GA	Not mentioned
	[78]	2015	Follower pattern mining	Not mentioned
	[80]	2015	PHIKS	Not mentioned
	[81]	2015	Vertical mining with breadth-first search	Not mentioned
	[82]	2015	Carpenter algorithm + Depth first search for closed itemset	Not mentioned
	[85]	2015	Weighted itemset mining	Not mentioned
	[86]	2016	Apriori	Not mentioned
	[87]	2016	Mining closed frequent itemsets	Not mentioned
	[88]	2016	Prepost algorithm	Not mentioned
	[89]	2016	Apriori + FP Growth	Not mentioned
	[90]	2016	Grid based partitioning for spatial neighborhood	Not mentioned
	Spark-based Articles			

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[44]	2015	Apriori	Execution time
	[45]	2016	High utility itemset mining	Execution time
	[49]	2018	Maximal Frequent Pattern mining	Not mentioned
	[52]	2018	Quantitative association rule mining	Not mentioned
	[91]	2015	Apriori	Not mentioned
	[92]	2015	FP-Growth	Not mentioned
	[93]	2015	Apriori	Not mentioned
	[94]	2015	Suffix method	Not mentioned
	[95]	2016	Faster-IAPT	Not mentioned
Regression/ Prediction	Hadoop-based Articles			
	[96]	2013	Extreme Learning Machine (ELM)	Speedup, Scaleup, Sizeup
	[97]	2015	Semiparametric regression	Not mentioned
	[99]	2016	BPNN	MAPE
	[100]	2020	MLR	Correlation Coefficient, RMSE
	[108]	2012	Regression Tree	Not mentioned
	[109]	2015	AAELM and MLR	MSE, MAPE, t-test
	[110]	2015	MLR	Not mentioned
	[111]	2015	SVM	Not mentioned
	[112]	2016	BPNN	Execution time, Accuracy
	Spark-based Articles			
	[101]	2016	Random Forest (RF)	Not mentioned
	[102]	2016	Fuzzy rule learning through GA	Not mentioned

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[104]	2018	Linear regression and Regression Tree	Mean Relative Error (MRE)
	[105]	2018	k-Weighted Nearest Neighbour	Not mentioned
	[106]	2020	ELM	MAPE, MAE, RMSE
	[107]	2020		Not mentioned
	[113]	2015	Weighted k-NN regression + prototype selection method based on fuzzy rough set theory (FRPS)	RMSE
	[116]	2016	Stochastic Gradient Descent (SGD) for Extreme Learning Machines (ELM)	Not mentioned
	[117]	2020	GRNN	MSE
Classi-fication	Hadoop-based Articles			
	[118]	2011	SVM	Not mentioned
	[119]	2012	ACO	Precision, recall, accuracy
	[120]	2013	SVM	Speedup, efficiency, accuracy
	[121]	2014	SVM	Accuracy, Sensitivity, Precision, and Matthews correlation coefficient

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[4]	2014	RF	TPR, FPR, precision, recall, Pearson product-moment coefficient
	[125]	2014	Horizontal and vertical compression	Not mentioned
	[126]	2015	SVM	Not mentioned
	[129]	2015	ELM	Not mentioned
	[131]	2015		Execuiton time, accuracy, precision
	[132]	2015	Fuzzy rule-based classification	Not mentioned
	[133]	2016	ELM	Not mentioned
	[134]	2016	k-NN	MAPE, RMSE, mean absolute error (MAE), maximum error (ME)
	[136]	2016	FP-Growth	Not mentioned
	[137]	2016	k-NN	Confusion matrix, Recall, Precision, Accuracy
	[138]	2016	ELM	Speedup, sizeup
	[140]	2016	ELM	Not mentioned
	[141]	2016	Voting based instance selection + random weight networks	Not mentioned
	[142]	2016	ELM	Speedup
	[144]	2016	SVM	Not mentioned
	[145]	2016	ELM	Speedup, scaleup, Precision, Recall, F-measure and G-mean

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[146]	2018	dissimilarity-based imbalance data classification	Not mentioned
	[149]	2018	Fuzzy rule-based classification	Execution time
	[150]	2019	Kernel-optimized SVM	Accuracy, execution time
	[162]	2012	Social Network Analysis (SNA) algorithms	Centrality measures
	[164]	2012	Sentiment classifier	Accuracy
	[165]	2013	DT + SVM	Precision, recall, F1 measure
	[166]	2013	RF	Accuracy
	[167]	2013	ELM	Speedup, sizeup
	[168]	2013	Meta-learning	Accuracy, speedup
	[170]	2013	Logistic regression (LogR)	Not mentioned
	[171]	2013	Genetic Programming	Accuracy, speedup
	[172]	2013	SVM	Not mentioned
	[180]	2014	Classification rule induction	Speedup, scaleup, sizeup
	[173]	2014	SVM	Speedup
	[174]	2014	K-means + LogR	Accuracy, TPR, FPR
	[175]	2014	Fuzzy rule-based classification	Accuracy
	[176]	2014	Classification rule	Accuracy
	[177]	2014	SVM	Not mentioned
	[178]	2014	k-NN	Not mentioned
	[179]	2015	SVM	Not mentioned
	[181]	2015	k-NN	Not mentioned
	[182]	2015	k-NN	Not mentioned

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[183]	2015	RF	Not mentioned
	[184]	2015	SVM, k-NN, Adaboost	Speedup
	[185]	2015	GA optimized DT	Not mentioned
	[186]	2015	RF	Not mentioned
	[187]	2015	Evolutionary undersampling	AUC, gmean
	[188]	2015	Feature ranking	Information gain
	[190]	2015	k-NN	Not mentioned
	[191]	2015	Fuzzy Associative classification	Not mentioned
	[192]	2015	Bayesian classifier	Not mentioned
	[193]	2015	SVM	Accuracy, execution time
	[194]	2016	TF-IDF + cosine similarity + Rocchio algorithm	Recall, Precision, F1 measure, speedup
	Spark-based Articles			
	[153]	2016	k-NN	Accuracy, execution time, speedup
	[154]	2016	Bayesian Classifier	Not mentioned
	[155]	2016	BPNN	Accuracy
	[156]	2017	RF	Accuracy, execution time, speedup
	[157]	2017	LogR, SVM	Accuracy
	[158]	2017	RF	Accuracy, execution time
	[159]	2018	DT	Not mentioned
	[160]	2018	Tree based index	Not mentioned
	[161]	2020	Random Forest	Accuracy, execution time

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[195]	2013	LogR	Accuracy, execution time
	[196]	2014	SVM	Accuracy, execution time
	[197]	2014	LogR, linear SVM	Execution time
	[198]	2014	k-NN	Execution time
	[199]	2015	Kohonen Neuron	Average error rate, Standard Deviation (SD)
	[200]	2015	NB + Entropy minimization discretizer	Accuracy, execution time
	[201]	2015	SMOTE + RF	Sensitivity, specificity, precision, G-mean, F-measure, execution time
	[202]	2015	Least Squares Support Vector Machines	Accuracy
	[203]	2016	Associative classifier	Accuracy
	[204]	2016	Naïve Bayes (NB), DT, RF	Accuracy, sensitivity, specificity
	[205]	2016	k-NN	Accuracy
	[206]	2016	LogR, NB	Accuracy, execution time
	[207]	2016	SVM	Accuracy
	[208]	2016	AANN + PSO	Classification Rate, MSE
	[210]	2016	k-NN	MRE
	[211]	2019	SVM	Accuracy, execution time
	Hadoop-based Articles			
Clustering	[212]	2014	Centroid-based clustering	Not mentioned

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[214]	2014	k-means++	Not mentioned
	[215]	2014	K-means	Davies–Bouldins index
	[216]	2015	Fuzzy c-means	Cluster purity
	[217]	2015	Ant colony clustering	F-measure, Davies-Bouldin index, Dunn index, Speedup, Scaleup, Sizeup
	[219]	2016	Semantic-Driven Subtractive Clustering	Execution time
	[220]	2016	Similarity Join Tree	Not mentioned
	[222]	2016	Density Peaks clustering	Not mentioned
	[224]	2016	top-k similarity using cosine similarity	Accuracy, execution time
	[226]	2016	k-NN	Not mentioned
	[230]	2016	K-means	Not mentioned
	[231]	2016	K-means	Speedup
	[232]	2016	Artificial Bee Colony (ABC)	F- measure, execution time, speedup
	[233]	2016	K-means	Not mentioned
	[234]	2020	User clustering based on sentiment score and cosine similarity	Execution time
	[239]	2009	K-means	Speedup, scaleup, sizeup
	[240]	2011	K-center, K-median	Execution time

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[241]	2013	Text document clustering	Purity, Rand index, F-measure
	[242]	2013	K-means	Execution time
	[243]	2013	K-means	Accuracy, execution time
	[244]	2013	K-means	Execution time
	[245]	2014	K-means	Accuracy, execution time
	[246]	2014	Fuzzy K-means	Not mentioned
	[247]	2014	K-means	Execution time
	[249]	2014	Fuzzy K-mean	Accuracy, execution time
	[250]	2014	K-means	Not mentioned
	[251]	2014	K-medoids	Silhouette coefficient
	[252]	2014	K-means	Execution time
	[253]	2014	K-means	Silhouette coefficient
	[255]	2014	Differential evolution clustering	Purity
	[256]	2014	Glowworm Swarm Optimization (GSO) clustering	Purity, speedup, scaleup
	[259]	2014	K-medoids	Speedup
	[260]	2015	Clustering based on similarity of nodes	Not mentioned
	[261]	2015	K-means	Silhouette coefficient
	[263]	2015	Fuzzy C-Means	Precision, recall, execution time
	[264]	2015	K-means	Execution time
	[265]	2015	K-means	Accuracy, execution time

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[266]	2015	K-means	Accuracy
	[267]	2015	Dynamic neighborhood selection (DNS)	Execution time, speedup
	[268]	2015	K-means	Connectivity, Dunn index, Silhouette measure, execution time
	[271]	2015	Evolutionary Clustering with Fireworks and Cuckoo-search based evolutionary algorithms	Execution time
	[272]	2015	K-means	Execution time
	[273]	2015	Fuzzy K-means	Execution time
	[274]	2015	K-means	Execution time
	[275]	2015	K-modes	Execution time
	[276]	2015	Range-based clustering	Execution time
	[277]	2015	PSO	Precision, recall, F-measure, execution time
	[278]	2015	Density-based, grid-based clustering	Execution time
	[280]	2015	K-means	Execution time
	[281]	2015	K-means	Execution time
	[282]	2015	K-means	Execution time
	[283]	2016	K-means	Execution time
	[284]	2016	K-means	Precision, recall, F-measure, execution time
	[285]	2016	K-means	Execution time, speedup

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[286]	2016	Fuzzy c-means	Execution time
	[287]	2020	K-means	Execution time
	Spark-based Articles			
	[235]	2018	K-means	Not mentioned
	[236]	2020	Canopy K-means	Not mentioned
	[237]	2020	K-means	Accuracy, execution time
	[238]	2020	Hierarchical clustering	Accuracy, scalability
	[289]	2014	SOM clustering	Accuracy, Rand Index, execution time
	[290]	2015	K-means	Execution time, accuracy
	[291]	2015	PSO	Processing time, Acceleration, Intra Cluster Distance, and Inter Cluster Distance
	[292]	2015	K-means	Execution time
	[293]	2015	Fuzzy c-means	Execution time
	[294]	2016	DenPeak	Execution time
	[295]	2016	DBSCAN	Execution time
	[296]	2016	Fuzzy c-means	Execution time
	[297]	2016	K-means	Execution time
	[298]	2016	Fuzzy c-Means	F-measure, Adjusted Rand Index, execution time
	[299]	2016	Density-grid clustering	Not mentioned
Outlier Detection/ Intrusion Detection	Hadoop-based Articles			
	[300]	2014	Cell-based outlier detection	Execution time

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[301]	2014	Artificial Immune Systems	FPR, accuracy, recall
	[302]	2016	Binary Bat Algorithm	Speedup, scaleup, sizeup, Detection rate, FPR
	[303]	2016	Rough set + Genetic Algorithm (GA)	Execution time
	[304]	2016	NB, SVM, conjunctive rule, RF tree, REPTree, and J48 (C4.5 Java implementation)	Accuracy
	[307]	2013	GA + k-NN	Accuracy, execution time
	[308]	2013	PSO clustering	TPR, speedup, execution time
	[309]	2014	ELM	TPR, FPR, speedup, sizeup
	[310]	2016	NB + k-NN	Accuracy, Sensitivity, Specificity, FPR, Precision, F1 measure
	Spark-based Articles			
	[306]	2018	Random Forest	Not mentioned
	[311]	2016	LogR, SVM, RF, Gradient Boosted DT, and NB	Accuracy, sensitivity, specificity, execution time

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[312]	2016	K-means	Not mentioned
Recommen-dation System	Hadoop-based Articles			
	[313]	2018	SVD	RMSE, Target Recall
	[315]	2012	Top-N recommendation	Prediction quality, speedup
	[318]	2014	Item-based and User-based Collaborative Filtering (CF)	Execution time, speedup
	[319]	2014	User-based CF	Accuracy, execution time
	[320]	2015	Stochastic Gradient Descent	Accuracy using RMSE
	[321]	2015	Item-based CF	RMSE, MAE
	[322]	2015	User-based CF	Execution time
	[323]	2016	Cosine similarity, k-NN, CF	MAE
	Spark-based Articles			
	[325]	2016	User-based CF	RMSE, execution time
	[326]	2020	Collaborative filtering	RMSE
	[327]	2020	SVM	Precision, recall, F1 score, and accuracy
Others	Hadoop-based Articles			
	[328]	2012	Rough set approximations	Speedup, scaleup, sizeup
	[329]	2013	Privacy preserving Feature selection	Not mentioned
	[330]	2014	Feature reduction	Not mentioned

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[331]	2015	Bayesian Network	Execution time, speedup
	[332]	2015	Incremental MapReduce	Execution time
	[335]	2016	Bayesian Network	Recall, execution time
	[336]	2016	PSO	Execution time
	[342]	2011	SGD	Not mentioned
	[343]	2012	Latent Dirichlet Allocation	Execution time, speedup
	[344]	2013	Selection algorithms	Execution time
	[345]	2014	Genetic algorithm and simulated annealing	Execution time
	[347]	2014	Positive Approximation	Not mentioned
	[348]	2015	GA + k-NN	Accuracy
	[349]	2015	K-means + SVM	Sensitivity, specificity, accuracy
	[350]	2015	GA	Execution time
	[351]	2016	Steiner tree	Not mentioned
	Spark-based Articles			
	[337]	2016	GA	Execution time, speedup
	[340]	2016	Feature selection using Information Gain, RELIEF-F, Correlation-based Feature Selection (CFS), Support Vector Machine Recursive Feature Elimination (SVM-RFE)	Not mentioned

Continued on next page…

DM Task	Article Reference Number	Year	Machine Learning Techniques	Performance Measure
	[341]	2016	minimum-redundancy-maximum-relevance (mRMR)	Not mentioned
	[352]	2016	artificial bee colony	Not mentioned
	[353]	2016	N-Gram Feature selection	Information Gain
End of Table				

D.2 Table for Dataset-wise distribution of Papers

Table D.2: Distribution of Papers: Dataset wise

Type of Dataset	Hadoop MapReduce Papers		Spark Papers		Total Count
	Reference Number	Count	Reference Number	Count	
Real-world/ Benchmark Datasets	[21] [31] [35] [38] [39] [40] [59] [61] [62] [67] [68] [71] [73] [74] [77] [78] [80] [81] [85] [96] [97] [108] [109] [111] [112] [118] [119] [120] [121] [4] [125] [126] [129] [131] [132] [134] [136] [137] [138] [144] [145] [146] [149] [150] [164] [165] [166] [167] [171] [173] [174] [175] [177] [179] [181] [182] [184]	123	[44] [52] [92] [94] [95] [101] [102] [104] [105] [113] [116] [153] [154] [155] [156] [157] [158] [159] [160] [196] [197] [199] [200] [201] [202] [204] [205] [206] [207] [208] [210] [211] [235] [268] [289] [290] [292] [293] [296] [297] [298] [306] [311] [312] [325] [340] [341] [352] [353] [117] [106] [161] [236] [237] [238] [326]	56	179

Continued on next page…

Type of Dataset	Hadoop MapReduce Papers		Spark Papers		Total Count
	Reference Number	Count	Reference Number	Count	
	[186] [187] [188] [190] [191] [192] [193] [194] [212] [216] [217] [219] [220] [222] [224] [230] [241] [243] [249] [252] [255] [259] [260] [263] [264] [265] [266] [267] [271] [272] [273] [274] [275] [276] [280] [281] [283] [284] [285] [286] [301] [302] [303] [304] [307] [308] [309] [310] [313] [315] [318] [320] [321] [322] [323] [331] [332] [335] [342] [343] [345] [347] [348] [349] [350] [100]				
Synthetic Datasets	[58] [60] [63] [69] [72] [88] [89] [90] [133] [178] [183] [232] [233] [239] [240] [247] [250] [253] [278] [300] [344] [287] [234]	23	[91] [294] [295]	3	26
Both Real and Synthetic Datasets	[22] [24] [29] [30] [42] [43] [55] [64] [497] [76] [82] [86] [87] [110] [140] [141] [142] [162]	33	[49] [93] [195] [198] [203] [299] [337]	7	40

Continued on next page…

Type of Dataset	Hadoop MapReduce Papers		Spark Papers		Total Count
	Reference Number	Count	Reference Number	Count	
	[168] [180] [176] [214] [215] [226] [231] [246] [256] [261] [282] [319] [328] [329] [330]				
Not Mentioned	[75] [99] [170] [172] [185] [242] [244] [245] [251] [277] [336] [351]	12	[45] [291] [107] [327]	4	16
End of Table					

Appendix E

List of publications

[1] **S. Kamaruddin** and V. Ravi, "EGRNN++ and PNN++: Parallel and Distributed Neural Networks for Big Data Regression and Classification," in ***Applied Soft Computing, Elsevier***. [**Communicated**]

[2] **S. Kamaruddin** and V. Ravi, "Parallel and Distributed Radial Basis Function Network for Regression and Classification in Big Data Paradigm," in **Handbook on Big Data Analytics**, V. Ravi, Ch. Aswani Kumar, Eds. Institution of Engineering and Technology, **IET, UK**, 2020. [**Accepted**]

[3] **S. Kamaruddin** and V. Ravi, "Architectures of Big Data Analytics: Scaling out Data Mining Algorithms using Hadoop-MapReduce and Spark," in **Handbook on Big Data Analytics**, V. Ravi, Ch. Aswani Kumar, Eds. Institution of Engineering and Technology, **IET, UK**, 2020. [**Accepted**]

[4] **S. Kamaruddin** and V. Ravi, "A Parallel and Distributed Radial Basis Function Network for Big Data Analytics," in **TENCON 2019, Kochi, Kerala, India**. IEEE, 2019, pp. 393-397.

[5] **S. Kamaruddin** and V. Ravi, "GRNN++: A Parallel and Distributed version of GRNN under Apache Spark for Big Data Regression," in **Data Management, Analytics and Innovation, ICDMAI**, vol. 1042, N. Sharma, A. Chakrabarti, and V. Balas, Eds. **Kuala Lumpur, Malaysia**: Springer, Singapore, 2020, pp. 215–227.

[6] V. Ravi and **S. Kamaruddin**, "Big Data Analytics Enabled Smart Financial Services: Opportunities and Challenges," in **Big Data Analytics, BDA 2017**, vol. 10721, P. Reddy, A. Sureka, S. Chakravarthy, and S. Bhalla, Eds. Hyderabad, India: Springer, Cham, 2017, pp. 15–39.

[7] **S. Kamaruddin**, V. Ravi, and P. Mayank, "Parallel Evolving Clustering Method for Big Data Analytics Using Apache Spark: Applications to Banking and Physics," in **Big Data Analytics, BDA 2017**. vol. 10721, P. Reddy, A. Sureka, S. Chakravarthy, and S. Bhalla, Eds. Hyderabad, India: Springer, Cham, 2017, pp. 278–292.

[8] **S. Kamaruddin** and V. Ravi, "Credit Card Fraud Detection using Big Data Analytics: Use of PSOAANN based One-Class Classification," in **Proceedings of the International Conference on Informatics and Analytics - ICIA-16**, 2016, pp. 33:1-33:8.

[9] V. Tejasviram, H. Solanki, V. Ravi, and **S. Kamaruddin**, "Auto associative Extreme Learning Machine based non-linear principal component regression for big data applications," in 2015 **Tenth International Conference on Digital Information Management (ICDIM)**, 2015, pp. 223–228.

A Parallel and Distributed Radial Basis Function Network for Big Data Analytics

2 Author(s) S.K. Kamaruddin ; Vadlamani Ravi All Authors

Abstract:
Radial Basis Function Network (RBFN) is one of the versatile neural network architectures, in that it can be used to solve regression as well as classification tasks. The proposed work implements RBFN in a parallel and distributed environment of Apache Spark. The new version of RBFN is referred to as PRBFN. The main contribution in this work is embedding of K-Means|| (a parallel distributed clustering method) in between input and hidden layer and implementation of parallel least square estimation employing outer product of matrices. The effectiveness of the PRBFN was tested on HEPMASS, HIGGS, ccFraud, and Amazon Movie Review (for sentiment classification) datasets under 10 Fold Cross Validation (10-FCV) setup. It is emphasized that the chief objective is to present a distributed and parallel version of the RBFN to handle big datasets.

Published in: TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)

Date of Conference: 17-20 Oct. 2019 **INSPEC Accession Number:** 19250345

Date Added to IEEE *Xplore*: 12 December 2019 **DOI:** 10.1109/TENCON.2019.8929442

ISBN Information: **Publisher:** IEEE

ISSN Information: **Conference Location:** Kochi, India, India

I. Introduction

Classification is one of the most fundamental tasks of data mining which has a wide variety of applications in many domains. Various techniques from a diverse family of methods viz., statistics and [illegible]sion tree, fuzzy classification, support vecto[illegible]l networks) have been

Data Management, Analytics and Innovation pp 215-227 | Cite as

GRNN++: A Parallel and Distributed Version of GRNN Under Apache Spark for Big Data Regression

Authors | Authors and affiliations

Sk. Kamaruddin, Vadlamani Ravi

Conference paper
First Online: 25 October 2019

Part of the Advances in Intelligent Systems and Computing book series (AISC, volume 1042)

Abstract

Among the neural network architectures for prediction, multi-layer perceptron (MLP), radial basis function (RBF), wavelet neural network (WNN), general regression neural network (GRNN), and group method of data handling (GMDH) are popular. Out of these architectures, GRNN is preferable because it involves single-pass learning and produces reasonably good results. Although GRNN involves single-pass learning, it cannot handle big datasets because a pattern layer is required to store all the cluster centers after clustering all the samples. Therefore, this paper proposes a hybrid architecture, GRNN++, which makes GRNN scalable for big data by invoking a parallel distributed version of K-means++, namely, K-means||, in the pattern layer of GRNN. The whole architecture is implemented in the distributed parallel computational architecture of Apache Spark with HDFS. The performance of the GRNN++ was measured on gas sensor dataset which has 613 MB of data under a ten-fold cross-validation setup. The proposed GRNN++ produces very low mean squared error (MSE). It is worthwhile to mention that the primary motivation of this article is to present a distributed and parallel version of the traditional GRNN.

Keywords

Apache Spark | Big data regression | GRNN | HDFS | K-means++ | K-means||

International Conference on Big Data Analytics
BDA 2017: Big Data Analytics pp 15-39 | Cite as

Big Data Analytics Enabled Smart Financial Services: Opportunities and Challenges

Authors | Authors and affiliations

Vadlamani Ravi ✉, Sk Kamaruddin

Conference paper
First Online: 25 November 2017

Part of the Lecture Notes in Computer Science book series (LNCS, volume 10721)

Abstract

Of late, the financial services industry is fast moving away from the traditional paradigm to the sophisticated digital way of dealing and the customer. Both the facets of the financial service industry, viz., the financial service provider and the customer are going through a digital evolution. In particular, banking industry has evolved from just journal and ledger entry paradigm to data and analytics driven banking operations, which subsumes online as well as offline customer behavior. This paper discusses various scenarios in baking, finance services and insurance (BFSI) areas, where big data analytics is turning out to be paramount. The paper also highlights the potential benefits, of the new-age technologies viz., Internet of Things (IoT), Blockchain, Chatbots and robotics.

Keywords

Big data analytics | Digital banking | Financial services | IoT | Chatbot | Insurance | Hadoop | Spark

International Conference on Big Data Analytics
BDA 2017: Big Data Analytics pp 278-292 | Cite as

Parallel Evolving Clustering Method for Big Data Analytics Using Apache Spark: Applications to Banking and Physics

Authors | Authors and affiliations

Sk Kamaruddin, Vadlamani Ravi ✉, Pritman Mayank

Conference paper
First Online: 25 November 2017

Part of the Lecture Notes in Computer Science book series (LNCS, volume 10721)

Abstract

A novel parallel implementation of the Evolving Clustering Method (ECM) is proposed in this paper. The original serial version of the ECM is the clustering method which computes online and with a single-pass. The parallel version (Parallel ECM or PECM) is implemented in the Apache Spark framework, which makes it work in real time. The parallelization of the algorithm aims to handle a dataset with large volume. Many of the extant clustering algorithms do not involve a parallel one-pass method. The proposed method addresses this shortcoming. Its effectiveness is demonstrated on a credit card fraud dataset (with size 297 MB), and a Higgs dataset was taken from Physics pertaining to particle detectors in the accelerator (with size 1.4 GB). The experimental setup included a cluster of 10 machines having 32 GB RAM each with Hadoop Distributed File System (HDFS) and Spark computational environment. A remarkable achievement of this research is a dramatic reduction in computational time compared to the serial version of the ECM. In future, the PECM shall be hybridized with other machine learning algorithms for solving large-scale regression and classification problems.

Keywords

ECM | Clustering | Big data analytics | Apache spark | Credit card fraud | Large Hedron Collider

RESEARCH-ARTICLE

Credit Card Fraud Detection using Big Data Analytics: Use of PSOAANN based One-Class Classification

Authors: Sk. Kamaruddin, Vadlamani Ravi Authors Info & Affiliations

Publication: ICIA-16: Proceedings of the International Conference on Informatics and Analytics • August 2016
• Article No.: 33 • Pages 1–8 • https://doi.org/10.1145/2980258.2980319

ABSTRACT

Banking and financial industries are facing severe challenges in the form of fraudulent transactions. Credit card fraud is one example of them. In order to detect credit card fraud, we employed one-class classification approach in big data paradigm. We implemented a hybrid architecture of Particle Swarm Optimization and Auto-Associative Neural Network for one-class classification in Spark computational framework. In this paper, we implemented parallelization of the auto-associative neural network in the hybrid architecture.

Feedback
ICIA-16: Proceedings of the International...
Credit Card Fraud Detection using Big...
Pages 1–8
← Previous Next →
Feedback
ABSTRACT
References

Auto associative Extreme Learning Machine based non-linear principal component regression for big data applications

Publisher: IEEE

4 Author(s) V. Tejasviram ; H. Solanki ; V. Ravi ; Sk. Kamaruddin **All Authors**

Abstract:
In this paper, we propose a hybrid model that combines the Auto Associative Extreme Learning Machine (AAELM) with Multiple Linear Regression (MLR) (AAELM+MLR) for performing big data regression. It works using Hadoop Mapreduce parallel computing model which is implemented in Python using Dumbo API. It works in two phases. In the first phase, three-layered AAELM is trained. The output of the hidden nodes of AAELM is treated as NLPCs. In the second phase, MLR model is fitted using these NLPCs as input variables. Effectiveness of AAELM+MLR model is demonstrated on two large datasets viz., airline flight delay dataset and gas sensor array dataset, taken from the web. It is observed that AAELM+MLR outperformed MLR model by yielding less average mean squared error (MSE) and MAPE values under the 10 fold cross-validation framework. A statistical test confirms its superiority at 1% level of significance.

Published in: 2015 Tenth International Conference on Digital Information Management (ICDIM)

Date of Conference: 21-23 Oct. 2015

INSPEC Accession Number: 15704571

Date Added to IEEE *Xplore*: 14 January 2016

DOI: 10.1109/ICDIM.2015.7381854

ISBN Information:

Publisher: IEEE

Conference Location: Jeju, South Korea

I. Introduction

With the emergence of SMAC (Social, Mobile, Analytics & Cloud) stack in Information Technology, we are confronted with huge volumes of data coming at a high velocity and from a variety of sources in various forms in almost all branches of science, engineering, finance and business. This has led to the crystallization of the definition of what is known as [redacted]. Later this definition has been extended to 7 V's signifying: Volume, Velocity, Variety, Variability, Veracity,

Scaling out some of the Data Mining Algorithms for Big Data Analytics

The sources bearing numbers 1, 2, 3, 4, 5, 6, 12 and 15 belong to Student's works. ∴ 35 − 27 = 8% Similarity Index

ORIGINALITY REPORT

35%	7%	34%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Dr. V. Ravi, M.Sc., M.S., Ph.D,
Professor
Head, Center of Excellence on CRM & Analytics
INSTITUTE FOR DEVELOPMENT AND RESEARCH IN BANKING TECHNOLOGY
(Established by Reserve Bank of India)
Castle Hills Road No. 1, Masab Tank, Hyderabad - 500 057, (A.P.), India

Dr. A.S. RAMASASTRI
DIRECTOR
INSTITUTE FOR DEVELOPMENT AN
RESEARCH IN BANKING TECHNOLOC
(Established by Reserve Bank of India
Castle Hills, Road No.1, Masab Tank,
Hyderabad - 500 057

1	Sk. Kamaruddin, Vadlamani Ravi. "Credit Card Fraud Detection using Big Data Analytics", Proceedings of the International Conference on Informatics and Analytics - ICIA-16, 2016 Publication	6%
2	S.K. Kamaruddin, Vadlamani Ravi. "A Parallel and Distributed Radial Basis Function Network for Big Data Analytics", TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019 Publication	6%
3	"Big Data Analytics", Springer Science and Business Media LLC, 2017 Publication	5%
4	V. Tejasviram, H. Solanki, V. Ravi, Sk. Kamaruddin. "Auto associative Extreme Learning Machine based non-linear principal component regression for big data applications", 2015 Tenth International Conference on Digital Information Management (ICDIM), 2015 Publication	4%

5	"Data Management, Analytics and Innovation", Springer Science and Business Media LLC, 2020 Publication	4%
6	Sk Kamaruddin, Vadlamani Ravi, Pritman Mayank. "Chapter 19 Parallel Evolving Clustering Method for Big Data Analytics Using Apache Spark: Applications to Banking and Physics", Springer Science and Business Media LLC, 2017 Publication	1%
7	www.springerprofessional.de Internet Source	1%
8	link.springer.com Internet Source	<1%
9	www.edureka.co Internet Source	<1%
10	Lecture Notes in Computer Science, 2015. Publication	<1%
11	"Encyclopedia of Social Network Analysis and Mining", Springer Science and Business Media LLC, 2018 Publication	<1%
12	Sk. Kamaruddin, Vadlamani Ravi. "Chapter 16 GRNN++: A Parallel and Distributed Version of GRNN Under Apache Spark for Big Data	<1%

	Regression", Springer Science and Business Media LLC, 2020 Publication	
13	journalofbigdata.springeropen.com Internet Source	<1%
14	Submitted to Erciyes Ãniversitesi Student Paper	<1%
15	idrbt.ac.in Internet Source	<1%
16	dl.acm.org Internet Source	<1%
17	scholarbank.nus.edu.sg Internet Source	<1%
18	Submitted to University of Hyderabad, Hyderabad Student Paper	<1%
19	www.talkwalker.com Internet Source	<1%
20	Amin Shokrzade, Fardin Akhlaghian Tab, Mohsen Ramezani. "ELM-NET, a closer to practice approach for classifying the big data using multiple independent ELMs", Cluster Computing, 2019 Publication	<1%
21	Submitted to Caledonian College of Engineering	