The Computer Network Faults Classification Using a Novel Hybrid Classifier

By

Karwan Hussein Qader

Supervisor

Dr. Mo Adda

This thesis is submitted in partial fulfilment of
the requirements for the award of the degree of
Doctor of Philosophy of the University of Portsmouth.

February, 2019

I would like to dedicate this thesis to my loving family ...

The soul of my Mother-in-law, *whose memories persist in being forever*

my Mother, *whom I owe my life*

my Father, *who sacrificed his life for us*

my Father-in-law, *who who was being my guardian during my PhD course*

my loving Wife "Asmaa" and Daughters "Dya & Lya, *who are my eternal gratitude*

List of publications

Published articles

Qader, K., Adda, M., Allaf, Zirak. (2019). A Proposed Subtractive Fuzzy Probabilistic Neural Network Classifier (SFPNNC) to Classify The Different Scenarios of The Traffic Faults in Computer Network System. Journal of IEEE Transactions on Fuzzy Systems. Special Issue (under review)

Qader, K., Adda, M. (2019, July). DOS and Brute Force attacks faults detection using An Optimised Fuzzy C-means. In IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)(under review)

Qader, K., Adda, M., Al-Kasassbeh, M. (2017). Comparative analysis of clustering techniques in network traffic faults classification. International Journal of Innovative Research in Computer and Communication Engineering, 5(4), 6551-6563.

Qader, K., Adda, M. (2014, September). Fault classification system for computer networks using Fuzzy Probabilistic Neural Network Classifier (FPNNC). In International Conference on Engineering Applications of Neural Networks (pp. 217-226). Springer, Cham.

Qader, K., Adda, M. (2013). Network Faults Classification Using FCM. In Distributed Computer and Communication Networks: Control, Computation, Communications (DCCN-2013) (pp. 66-73).

Qader, K., Adda, M. (2013). A survey of network faults classification using clustering techniques. International Journal of Advanced Research in Computer and Communication Engineering, 2(10), 4028-4032.

Declaration

Whilst registered as a candidate for the above degree, I have not been registered for any other research award. The results and conclusions embodied in this thesis are the work of the named candidate and have not been submitted for any other academic award

April 2019

Acknowledgements

Firstly, I would like to thank almighty God for guiding me through my PhD and granting me the strength to complete it. The last few years of my PhD have been some of the hardest, but also most inspiring and rewarding in my life. I can say that the lessons I have learnt throughout this period of time have been both professional and personal. There are a number of people whom I must mention and give thanks to.

First and foremost, I must offer my appreciation to my supervisor, **Mo Adda**, who has been a beacon of inspiration and guidance for me throughout my PhD journey. The professional and personal knowledge and support I received I will always remember.

Secondly, my dearest and closest friend and colleague **Zirak Allaf**. I am forever indebted to him for the endless encouragement and direction he was always to pleased to give me.

My parents, sisters and brothers have also offered me warm encouragement and support, for which I am also grateful.

Finally, and most importantly, I must thank my wonderful and loving wife, **Asmaa** and my two beautiful daughters **Dya** and **Lya** who always remind me that life should always be enjoyed.

Abstract

The increasing importance and complexity of networks led to the development of network fault management as a distinct field, providing support for network administrators with quality services and ensuring that networks work appropriately. Fault diagnosis is a central aspect of network fault management. Since faults are unavoidable in communication systems, their quick detection and isolation are essential for the robustness, reliability and accessibility of the system. In large and complex communication networks, automating fault classification is critical. Because of many factors, including the volume of network information, it is hard to solve network fault problems with traditional tools, rendering intelligent analysis a critical method in the process of network fault diagnosis.

This work stated how the common traffic faults in a computer network system could be classified efficiently and improved the performance of the network which is lead to lessening the cost and the time consumed amount. Thus the network administrators will avail to diagnosis the network issues instantly. The experimental work used the datasets of IF-MIB variables which is detected by a researcher in two different scenarios that captured in the router and the server.

To address these issues, this research has conducted several significant contributions, which is relate to the concern of fault management in the computer network system. One of the main tasks of classifying faults is refining the existing datasets. Usually datasets obtained from network analysis tools and experiments are prone to noise, ambiguities and inaccuracies. Thus to produce cleansed datasets that are free from any noises, and inconsistent, two different filtering techniques (normalisation and standard scaling) are proposed.

The optimised fuzzy clustering means (subtractive fuzzy clustering means), which is hybridised with the subtractive clustering is also developed to indicate the optimal number of clusters efficiently. Other researchers did not address the efficiency of their clustering methods.

Further, based on the advantages of the PNN classifier, the SFCM is consolidated with it, and a new proposed classifier model is developed which is defined as a subtractive fuzzy probabilistic neural network classifier (SFPNNC).

The newly generated datasets have been tested through different algorithms to output clusters first then labelling each group of the fault traffic. The experimental analysis revealed that the SFCM achieved immensely higher performance and more accurate results than K-means and Expectation Maximisation (EM).

Ultimately, the empirical outcomes showed that our proposed hybrid classifier model (SFPNNC) outperforms the existing classifier (PNN) and achieved higher precisions and performances.

Overall conclusion, it is validated that the proposed SFCM outperformed the K-means and EM which the partition coefficient (PC) reached to (0.96). The (SFPNNC) classifier outcomes showed that the proposed method classified the faults very efficiently with an accuracy of 94% for light traffic workloads.

Table of contents

List of figures xiii

List of tables xvi

Nomenclature xvii

1 Introduction 1

1.1 Motivations . . . 3
1.2 Research questions . . . 4
1.3 Original contributions and benefits of this Work . . . 5
1.4 Research aim and objectives . . . 6
1.5 Research methods . . . 7
1.6 Thesis Organization . . . 8

2 Background and Literature Review 10

2.1 Network Fault Classification: An Overview . . . 10
2.2 Network Faults . . . 10
2.3 Network Fault Management . . . 11
2.4 Current Types of Network Fault Classification . . . 12
2.4.1 Clustering Methods . . . 13
2.4.1.1 Hierarchical Clustering . . . 14
2.4.1.2 Partitioning Clustering . . . 15
2.4.1.3 Density Based Clustering . . . 16
2.4.1.4 Model-Based Clustering Methods . . . 18
2.4.1.5 Grid-Based Clustering Methods . . . 18
2.4.1.6 Soft Computing Methods . . . 19
2.4.2 Classification Methods . . . 24
2.4.2.1 Nearest Neighbour Classifier . . . 25

2.4.2.2 Decision Tree Classifiers . . . 26
2.4.2.3 Statistical Learning Methods . . . 26
2.4.2.4 Instance-Based Learning Methods . . . 27
2.4.2.5 Support Vector Machine . . . 28
2.4.2.6 Neural Networks . . . 29
2.5 Fault Classification Challenges . . . 32
2.6 Prior Works on Fault Management and Classification . . . 32
2.7 Summary and Research Gaps . . . 39

3 Methodology and Framework Design 41
3.1 Fuzzy Theory . . . 42
3.2 Fuzzy C-Means . . . 43
3.2.1 Initialization . . . 46
3.2.2 Volume of Clusters . . . 49
3.2.3 Fuzziness Degree . . . 49
3.2.4 Euclidean Distance . . . 51
3.3 K-means . . . 51
3.4 Expectation Maximization . . . 52
3.5 Neural Networks for Fault Classification . . . 52
3.6 Theory of Neural Networks . . . 53
3.7 Probabilistic Neural Network . . . 55
3.7.1 Classification Theory of PNN . . . 55
3.7.2 The Architecture of Probabilistic Neural Networks . . . 57
3.7.2.1 Input layer . . . 59
3.7.2.2 Hidden layer . . . 59
3.7.2.3 Summation layer . . . 61
3.7.2.4 Decision layer . . . 62
3.7.3 Radial Basis Function . . . 63
3.7.4 The smoothing parameter, σ . . . 64
3.8 Frame work design . . . 65
3.9 Chapter Summary . . . 67

4 Data visualisation and preprocessing 71
4.1 Network Traffic Fault Datasets Characteristics and Types . . . 72
4.1.1 Server crash and link failure . . . 74
4.1.2 Broadcast Storm . . . 74

4.1.3 Babbling node . . . 75
4.2 Data Visualisation . . . 75
4.3 Data Pre-processing (Filtering) . . . 86
4.3.1 Feature Scaled . . . 86
4.3.1.1 Standard Scaler . . . 87
4.3.1.2 Data Normalisation . . . 89
4.4 Chapter Summary . . . 90

5 Results discussions and evaluations 92
5.1 Introduction . . . 92
5.2 Comparative Experimental Results of SFCM, K-means and EM . . . 92
5.3 Classification of Evaluations . . . 98
5.3.1 Receiver Operating Characteristic (ROC) curve . . . 99
5.3.2 Macro-average precision . . . 100
5.3.3 Micro-average precision . . . 103
5.4 Experimental Results of Subtractive Fuzzy Probabilistic Neural Network Classifier (SFPNNC) . . . 104
5.4.1 SFPNNC experiment results and evaluations for light scenario of router traffic faults . . . 104
5.4.2 SFPNNC experiment results for light scenario of server traffic faults 106
5.4.3 SFPNNC experiment results for heavy scenario of router traffic faults 108
5.4.4 SFPNN experiment results for heavy scenario of server traffic faults 110
5.5 Comparative Analysis of the Existing Classifier (PNN) with the Proposed Method (SFPNNC) for all Datasets . . . 112
5.6 Chapter Summary . . . 114

6 Conclusions And Future Works 116
6.1 Conclusions and Research Summary . . . 116
6.2 Summary of Research Questions . . . 118
6.3 Limitations . . . 120
6.4 Future Work . . . 121

References 123

Appendix A The Router Heavy workloads data visualisation 135

Appendix B The Router Heavy workloads data visualisation 140

Appendix C Standard Scale **145**

Appendix D Normalisation filtering **146**

List of figures

2.1 Machine learning – supervised vs. unsupervised 12
2.2 Supervised machine learning 13
2.3 Workflow of unsupervised learning 14
2.4 Summary of clustering algorithms 15
2.5 Illustrates high and low density points 16
2.6 Differentiating core points, border points, and outliers 17
2.7 Data mining paradigms including classification methods 24
2.8 The functioning of kNN 25
2.9 The process underlying the decision tree algorithm C4.5 27
2.10 General statistical learning methods 28
2.11 Instance-based learning 29
2.12 Architecture of PNN 30

3.1 Distribution of mono-dimensional data 45
3.2 k-means membership degree 46
3.3 Fuzzy c-means membership level 47
3.4 Optimal number of clusters 49
3.5 Representation of a typical simple neural network 54
3.6 Standard PNN structure 58
3.7 Input aspect of the PNN 60
3.8 PNN Pattern layer 61
3.9 Summation unit of the PNN 62
3.10 Decision unit of the PNN 63
3.11 Smoothing Parameter 65
3.12 The framework design of the proposed method 69
3.13 The main proposed flowchart of SFPNNC 70

4.1 The normal traffic of the router in the light scenario 76
4.2 The link failure and server crash traffic of the router in the light scenario . . 77
4.3 The broadcast storm traffic of the router in the light scenario 78
4.4 The babbling node traffic of the router in the light scenario 80
4.5 The normal light workloads in a server 82
4.6 The light workloads of link failure or server crash faults in a server 83
4.7 The light workloads of the broadcast storm faults in a server 84
4.8 The light network workloads of babbling node faults in a server 85

5.1 Performance Comparison of K-means, SFCM and EM implementations . . 93
5.2 The PC of the proposed SFCM for nine different tests 95
5.3 The PC of K-means for nine different tests 96
5.4 The PC of EM for nine different tests 97
5.5 The PC curves of the SFCM, K-means and EM 98
5.6 ROC curve sample 100
5.7 The confusion matrix 101
5.8 The ROC to the light router workloads 105
5.9 The proposed method performance in classifying the light traffic of the router 106
5.10 The ROC to the light server workloads 107
5.11 The proposed method performance in classifying the light traffic of the server 108
5.12 The ROC curves to the heavy router workloads 109
5.13 The proposed method performance in classifying the heavy traffic of the router 110
5.14 The ROC curves to the heavy server workloads 111
5.15 The proposed method performance in classifying the heavy traffics of the server 112
5.16 The ROC curves to the various workloads of the PNN 113

A.1 The router normal traffic of the heavy workloads 136
A.2 The router heavy server crash or link failure traffic workloads 137
A.3 The router heavy broadcast storm workloads 138
A.4 The router heavy babbling node workloads 139

B.1 Normal traffic of the heavy workloads in server 141
B.2 The heavy traffic of server crash or link failure workloads in server 142
B.3 The heavy broadcast storm workloads in server 143
B.4 The heavy babbling node workloads in server 144

C.1 The standard scaler filtering code in Pyhton 145

D.1 The normalisation filtering code in Pyhton 146

List of tables

3.1 Fuzziness degree 50

4.1 Parameters of network involved in router anomalies 73
4.2 Parameters of network involved in server anomalies 74
4.3 Datasets details 74
4.4 A non-normalized and non-scaled data of heavy server traffics 87
4.5 Means of heavy server network traffic features 88
4.6 The data sample after standard scaling 89
4.7 The Normalised dataset 91

5.1 The sample class prediction of the heavy network workloads of the server . 102
5.2 Precision and macro-average of the sample 102
5.3 Individual class performance for SFPNNC and PNN classifiers 114
5.4 Classifiers performance comparison 114

Nomenclature

Abbreviations

AANN Approaches such as Auto-associative Neural Networks

AFLC Adaptive Fuzzy Leader Clustering

AI IArtificial Intelligence

AIRS Artificial Immune Recognition System

ANFIS Adaptive Network-Based Fuzzy Inference System

ANN Artificial Neural Networks

AUC Area Under Curve

BPN Back Propagation Network

CART Classification and Regression Trees

CFBPN Cascaded Correlation Feed Forward Network

CoG Centre of Gravity

DAG Directed Acyclic Graph

DBSCAN Density Based Spatial Clustering of Applications

DNN Deep Neural Network

DPM Diritchlet Process Mixture

DT Decision Trees

DWT Discrete Wavelet Transform

EC Evolutionary Computing

ECR Extra-Cellular Recordings

EEG Electro Encephalo Graphy

EM Expectation Maximisation

EMD Empirical Mode Decomposition

EMD Empirical Model Decomposition

FAIRS Fuzzy Artificial Immune Recognition System

FCM Fuzzy Cluster Means

FDA Fisher Discriminant Analysis

FDI Fault detection and isolation

FK-NN Fuzzy K-Nearest Neighbour

FL Fuzzy Logic

FMRI Functional Magnetic Resonance Imaging

GANFIS Genetic Adaptive Neuro-Fuzzy Inference System

GNFS Genetic Neuro-Fuzzy Systems

HC Hard Computing

HIF High Impedance Faults

ICA Independent Component Analysis

IDS Intrusion Detection System

IMFs Intrinsic Mode Functions

IOCM Input-Output Classification Mapping

LIF Low Impedance Faults

LTE Long Time Evaluation

MA Mobile Agent

MIB management information bases

ML Machine Learning

MLE Maximum Likelihood Estimation

MST Minimal Spanning Tree

NN Nearest Neighbour

NNC Nearest Neighbour Classifier

OPTICS Ordering Points to Identify the Clustering Structure

PC partition coefficient

PCA Principal Component Analysis

PDF Probability Distribution Function

PLS Partial Lease Squares

PNN Probabilistic Neural Networks

PNNC Probabilistic Neural Network Classifier

QoS Quality of Service

RBFN Radial Basic Function Network

RL Reinforcement Learning

ROC Receiver Operating Characteristic

SC Soft computing

SFCM Subtractive Fuzzy Cluster Means

SFPNNC Subtractive Fuzzy Probabilistic Neural Network Classifier

SFPNNC Subtractive Fuzzy Probabilistic Neural Networks Classifier

SNMP Simple Network Management Protocol

SOM Self Organising Map

SR Stochastic Resonance

SSE Sum of Squared Error

STING Statistical Information Grid

SVM Support Vector Machines

TICNN Convolution Neural Networks with Training Interference

WPA Wavelet Packet Analysis

Chapter 1

Introduction

Computer networks have become integral aspects of modern society. They are now utilised for everyday activities by vast proportions of the populations and they are increasingly invaluable in the business world. Regrettably, these networks are frequently abused as a result of the unsanctioned dissemination of copyrighted material, the distribution of fraudulent communication and online attacks on other customers, among others. Such actions can present significant risks to online users, businesses or creative professionals, which emphasises the need to determine methods that can provide protection. As each of these threats has unique characteristics, it is necessary to develop different strategies. For instance, online defence could take the form of an appropriately structured network firewall, regular updates to operating systems and applications, legal constraints, scanning of network traffic and the filtering of undesired applications. Parallel with the increased magnitude of networks of communication in the real world, it has become a necessity to enhance the efficiency of network management. As the condition and security of networks are essential factors in the maintenance of performance, systems should implemented that that control networks in terms of attainability, scalability, adaptability and tolerance of faults (Qader et al., 2017). In order to achieve this, it has become a necessity that measures are developed to control and categorise network errors, with the goal of applying preventative approaches in addition to developing methods of crisis management and plans for all eventualities.

A network error is defined a defect that occurs within a communication network. This means that it is the unintended deficiency of hardware, software or any other aspect of the particular network. These errors can be caused by different factors, such as failure of a router, disruption to fibres or various other reasons. Furthermore, each instance of service disruption may not be caused by an error and vice versa (Cho et al., 2003). It was suggested by Gill et al. (2011) that network faults are generally caused by either the breakdown of links of

devices, which both lead to traffic losses. Hence, tolerance of faults is a critical attribute of such networks (Angjeli et al., 2013). It also represents the capability of a network to maintain functionality regardless of the emergence of errors within the system. Both tolerance for errors and strong reliability are coveted attributes of all networks (Bistouni and Jahanshahi, 2015). Faults that occur in networks can lead to packet loss and network faults frequently occur in load balancers (Gill et al., 2011).

Applications designed for monitoring networks are becoming increasingly important as they provide benefits for network security as well as for diagnosing errors in the network. The concept of managing network faults is not new. The detection and categorisation of errors is a critical component of the maintenance of network performance as well as the prevention of any system breakdowns. It can facilitate the process of network administration and enables the implementation of appropriate actions that guarantee network functionality and the provision of service quality. One of the most frequently used methods of data mining is clustering, which is employed in applications designed to manage faults for the purpose of grouping objects and resolving issues related to increased dimensionality. Fuzzy Cluster Means can be utilised in combination with Probabilistic Neural Networks (PNN) in order to categorise network errors, with the goal of enhancing the performance and improving the precision of classification (Qader and Adda, 2014). The object groups that are determined by this procedure enable network administrators to make informed decisions that will provide protection for data communications throughout the network. There are numerous existing methods that are employed for the process of clustering and classification in relation to data mining.

The objective of the present study is to devise a method of classifying network issues by utilising datasets that incorporate the aspects of management information bases (MIB) variables in the local nodes, as mentioned in Chapter 4 and 5. This will be accomplished through the application of machine learning approaches that are predominantly founded on a semi-supervised technique that uses Subtractive Fuzzy Cluster Means (SFCM) and the Probabilistic Neural Classifier (PNN), which have the capability to classify the irregular traffic activity in the extant datasets. Furthermore, this project will provide an explanation regarding the preparation and refinement of noisy data into pure data for the purposes of classification. Subsequently, a model will be proposed that is dependent on the partition of datasets into subsets by utilising FCM, while the process of clustering will be beneficial for improving the precision of the outcomes. When data is tested without being clustered, there is a significant amount of noise and this can lead to reduction in the precision of the classifiers. The aim here is to augment the precision of the current classifier through

the use of hybrid techniques, with the combination and introduction of various supervised and unsupervised algorithms. Resultantly, the acquired results will facilitate the process of network administrators making effective decisions, which can prevent network system malfunctions.

1.1 Motivations

One of the most significant difficulties in the modern world lies in how the performance of computer network structures can be calculated when different kinds of networks are combined. Recently, data-focused networks have transformed into converged frameworks in which the importance of real-time traffic continues to expand. These structures consist of conventional or more advanced fibre connections as well as mobile and wireless networks (Al-Kasassbeh and Adda, 2008). Many different techniques are currently being used for the classification of errors, which determine the measurements for the network.

Recently conducted studies have employed various different approaches in this field, although they have predominantly revealed new problems; for example, some do not deal with issues related to the performance of the classifier, while others only concentrate on single faults without considering the principle source involvements such as the router or server, which could be the primary targets of such events. Furthermore, network administrators are cognisant of how the errors are isolated, which will reveal information on the precise origins of the issue. There are several explanations as to why service providers, and network administrators in general, have declared a profound interest in having the ability to determine network traffic, which include: i) tracking application tendencies; ii) the capability to implement policies based on the class of traffic, such as enabling improved quality of service for voice traffic; iii) comprehension of the applications being utilised facilitates the understanding of end users and provides a beneficial input for numerous research studies, ranging from experience quality to marketing forecasts.

This endeavour presents various challenges as a result of the rapid expansion of networks and the transformation of software that can generate various forms of traffic. Additionally, some applications attempt to obscure traffic in order for it not to be detected, as can be observed in peer-to-peer clients. Therefore, a contest can be observed between the techniques of obfuscation and detection that have similarities to those involved in computer security. Thus far, although there has been substantial research in this field, there are a number of factors that require further investigation. In the present study, comprehensive investigation will reveal problems that have previously been ignored, such as classifier portability problems

that use a Probabilistic Neural Network Classifier. Various techniques are also proposed that can be used to design a model to classify and address faults through the development of a hybrid classifier based on machine learning.

Hence, it is of utmost important that an efficient structure is developed for the classification of such abnormalities as well as to determine the different types of fault. It can be challenging to differentiate certain faults from conventional traffic as they have similar characteristics. This problem can be resolved through the application of the proposed hybrid Subtractive Fuzzy Cluster Means (SFCM), which has the ability to precisely identify different data points that are contained within groups of traffic. Systems of monitoring, which do not possess flexibility, can be challenging in terms of implementation or create acceptable overheads are not an appropriate means of accomplishing sensitive measurements.

1.2 Research questions

This part of the thesis determines and provides an explanation for the research questions that are necessary to generate an answer to the proposed hypothesis. **Hypothesis:** *The creation of an efficient model in a complex computational environment that has the capability to classify both light and heavy network workloads with accuracy and precision.* The network delay, bandwidth, loss of packets and additional parameters will be considered as the effects that can generate quality services and network application. The suggested hybrid technique for the classification of traffic should be able to resolve certain problems and should aim to sustain the quality of the network, which generates the following questions:

1. How should network traffic in light and heavy loads be classified in real-time, ensuring that the cost of the determined devices (server, router and the network bandwidth) remains affordable? Additionally, how should traffic faults that are identified and gathered in various scenarios be resolved?

2. How can the traffic that emanates from different devices be characterised and is it light or heavy traffic?

3. How does the clustering augment the outcomes of the classification of traffic faults?

4. What effect does pre-processing have on the performance of the system?

5. To what extend does the proposed hybrid method improve the outcomes in comparison to the current classifier?

In order to validate the hypothesis and address the questions listed above, the findings derived from the experiments will generate definitive answers in the conclusion of the research.

1.3 Original contributions and benefits of this Work

The aim of this thesis to provide a bridge between the actual requirements of the network industry and the research conducted in the field of network traffic classification. Although the majority of the current methods that have been developed for classifying network traffic have reported high precision, they generally have not dealt with the practical problems in relation to how such systems can be incorporated into actual environments.

This thesis presents some contributions in the field of computer faults management and classification. In this research, two different filtering methods (Normalisation and Standard Scaler) are designed which they are used to purify and refine the existing datasets. Thus new datasets will be constructed which is free from the noises, removing any misrelated features and inconsistency, which makes the data to be efficiently adaptable to the classifier model and improves its accuracy results, see Appendix C and D.

Next, Fuzzy Cluster Means (FCM) has used to avoid the classifier from any misclassification when possibly all faults may conceive in the system. To the best of our knowledge, no other research applied FCM to the IF-MIB variables to make the data grouping. There are few rare works that used FCM, but they applied the algorithm to a different environments and diverse nature of the data as explained in chapter 2. To overcome the central issues of the FCM which is known commonly for its slow performance, we combined a new algorithm (Subtractive Clustering- SC), and new FCM is proposed named as Subtractive FCM (SFCM). The SFCM as the fuzzy clustering compared with K-means the hard clustering and EM the hard and soft clustering method has achieved higher efficiency plus performance depending on the PC metric see section 5.2 in Chapter 5.

Furthermore, the new hybrid method which is a semi-supervised approach is developed to classify the potential traffics and faults that may occur in different conditions and resources of the computer network system. The proposed system combines the SFCM with the PNN classifier to build a (Subtractive Fuzzy Probabilistic Neural Network Classifier- SFPNNC). Then the obtained classification results in two different classifiers -the proposed one and the existing PNN classifier- are validated based on their performances and accuracy as reported in see section 5.5 Chapter 5. The results shown that the proposed method (SFPNNC) outperforms the existing classifier (PNN)

The majority of past studies and projects have only focused on the classification of faults within various environments rather than computer networks. Although some researchers have investigated the process of detecting faults, they have not extended this research to the classification of such faults. While some studies have discussed fault classification, they have only concentrated on a minimal number of faults with a disregard for other factors.

Some previous studies have not taken the MIB variables into consideration. As these variables are critical factors in the process of detecting and classifying faults, it is important that this area receive more attention. Some studies in the literature have concentrated on the benefits of certain methods when applied for the purpose of fault management. However, the limitations were not investigated by examining different types of faults observed in different network situations. The findings of other studies have been founded on the current analytical tools and their characteristics instead of assuming a more holistic approach to the identification and classification of network faults.

As has been emphasised, all of the issues mentioned above will be addressed by this project.

1.4 Research aim and objectives

The principal aim of this thesis is the development of a new technique and a proposed semi-supervised technique that will combine Probabilistic Neural Network (PNN) and proposed Subtractive Fuzzy Clustering Means (FCM) that will have the capability to classify different categories of faults. The development of a logical classification system would have particular benefits for the process of network management, could prevent system failures through the earlier isolation of faults, could forecast abnormal occurrences and determine the type of fault in order for efficient repair.

Additionally, the ensemble model will amalgamate the strengths and weaknesses of the individual members and effectively employ the fusion strategy to obtain augmented outcomes. By combining the outcomes of each individual model, this will result in enhanced overall performance.

The key objectives of this project are:

1. To classify network faults by utilising a semi-supervised technique in the form of a hybrid technique through the development of Fuzzy Clustering Means (FCM) with Probabilistic Neural Network (PNN) based on artificial intelligence in order to precisely classify network faults.

2. To investigate the necessity for a new network fault management technology that can address the contemporary and future network specifications and needs

3. To study the utilisation of ensemble machine learning instruments in the classification of network faults.

4. To study statistical techniques, predominantly in the pre-processing and refinement of data, based on various approaches, including Feature Scaling, min-max scaling and normalisation, and use them in the process of purifying actual sensitive faults in the network

5. To provide guidance to network managers so that they can take the appropriate measures to ensure that network functionality is sustained and service quality is provided, while the groups of objects generated are beneficial for network managers when making informed decisions to ensure protection of data communications through the network.

6. To demonstrate the significance of MIB attributes that have an effect on the achievement of enhanced classification performance

1.5 Research methods

We have attempted to resolve the deficiencies related to the past studies and projects in the literature. The principle value of this study is that it proposes new techniques beginning with the initial stages of filtering data utilising different statistical methods through to the refinement and acquisition of adequately prepared sample data. Furthermore, semi-supervised techniques such as Fuzzy Clustering Means will be utilised, which are important in the scenario where the sample datasets have different faults simultaneously, which leads to traffic within the network as a result of its magnitude.

In order to prevent this scenario, the recommendation is that it will be distributed as segments of the datasets or subsets. Moreover, the clustering process breaks down any big data into smaller data that makes the proposed classifier to be a valuable application to process big data efficiently, and this is considered one of the main contributions of the research. In order to address this issue, unsupervised machine learning is employed to cluster the datasets that emanate from distinct segments of the traffic in the network.

In this context, FCM is considered to have certain advantages over previous techniques for the process of clustering. This can be explained by its capability to allow an object to belong to multiple clusters with a certain level of probability. This type of soft computing

has strong potential for application in real-world systems where the particular focus is on precision.

An additional algorithm defined as PNN, which is a feed-forward neural network, has significant benefits for the network fault classification. It is another frequently used algorithm that employs the Probability Distribution Function (PDF) for each of the classes when making decisions related to classification.

In this work, a hybrid system for managing network faults is proposed which is called a Subtractive Fuzzy Probabilistic Neural Network (SFPNNC), which makes use of the fuzzy clustering potential of FCM and the efficient classification of PNN in order to generate more precise and efficient network fault classification.

The reasoning behind the consideration of FCM and combined with the Subtractive Clustering in this context is detailed below:

1. The FCM can resolve problems where the boundaries of clusters are not exact, allowing the opportunity for soft clustering decisions to be made

2. It can be beneficial for problems of high dimensionality.

3. It is an iterative algorithm that is used for the optimisation of objects into different clusters. It is appropriate for the examination of network traffic, particularly the management of faults.

4. The SC support the FCM to overcome its limitation and improves the performance of the method see 5.2 in chapter 5.

Hence, the ideal attributes of the proposed hybrid system would incorporate training that is more efficient than in BP networks, assurance of improved convergence in comparison to Bayesian networks, and the facilitation of incremental training with uniform speed and durability when addressing sample noise.

1.6 Thesis Organization

This thesis is arranged as follows:

1. **Chapter Two** presents relevant background information regarding the area of network traffic classification and provides an in-depth explanation of supervised and unsupervised techniques utilised in the classification of faults. In the literature, the main gaps of the research papers are pointed out.

2. **Chapter Three** the utilisation of supervised and unsupervised methods for the classification of network traffic is presented, along with the proposal of a new method founded on the semi-supervised technique (a hybrid method-SFPNNC). The context of the network situations render this approach ideal for a real-time traffic classification solution. After this, the framework structure of the proposed system will be shown in a diagram, in which each stage will be illustrated accompanied by an explanation.

3. **Chapter Four** the data visualisation for all data and scenarios will be presented. The processes of data manipulation and pre-processing based on two different data filtering that are administered to the unprocessed data are explained

4. **Chapter Five** the outcomes of the that conducted by employing the proposed technique will be presented in this chapter. First the FCM is validated in performance and effectiveness with the K-means and EM. Following the outcomes of the hybrid method including three different algorithms, namely fuzzy clustering means, subtractive clustering and the probabilistic neural network is demonstrated. Then, the outcomes from the proposed method and the PNN classifier will be discussed, followed by a comparison of the experimental findings in order to assess the individual performance of each approach.

5. **Chapter Six** presents a conclusion of the thesis along with recommendations for future research and the author's opinion regarding how the findings of this paper could be advanced in future projects.

Chapter 2

Background and Literature Review

The identification and classification of network faults is crucial in ensuring the health of various types of global communication networks. Further research in this area is therefore of paramount importance. This chapter reviews the literature on network fault classification and elucidates two important aspects of this research: the clustering and classification methods involved, and their utility in network fault classification. It also reviews current research on fault classification and management.

2.1 Network Fault Classification: An Overview

There is an increased need to manage communication networks efficiently as they continue to grow. Network security and the health of services play a vital role in sustained performance and it is therefore important to have mechanisms in place to manage the availability, scalability, flexibility, and fault tolerance of networks. Measures are therefore needed to manage and classify network faults and take preventive steps while developing strategies to support disaster recovery and contingency plans.

2.2 Network Faults

A network fault is a malfunction in a communications network caused by an unplanned software or hardware failure. Such faults may occur for various reasons such as router failure, a fibre cut, and so on (Bathula et al., 2018). Jhawar and Piuri (2017) state that network failures are due either to link failure or device failure, both of which result in loss of traffic. Fault tolerance is therefore a very important characteristic of a network as it denotes its

ability to continue functioning despite the presence of faults in the system (Wang et al., 2018). Fault tolerance and high reliability are therefore desirable qualities for a network (Siewiorek and Swarz, 2017). Network faults cause a loss of packets and many faults are experienced by load balancers (Roy et al., 2017).

2.3 Network Fault Management

Fault management refers to the process of diagnosing faults and taking corrective measures. It is conducted in two phases: fault management tuning for fault management configuration, and the action taken when faults are encountered (Paradis and Han, 2007). Analysing network traffic and faults is therefore an essential part of network management. The classification of internet traffic can help network administrators and security personnel understand traffic demands, diagnose faults, and develop strategies to build better models (Auld et al., 2007). Network faults or malicious traffic are therefore a key element of traffic classification. One such class of traffic is known as attack traffic (Moore and Papagiannaki, 2005). The characteristics of network faults affect the nature of fault diagnosis which is both non-linear and complicated. This is the rationale underlying the need for a form of network classification that offers more utility regarding fault management (Li et al., 2008). Network reliability and fault tolerance can be enhanced by considering different dimensions. These include the identification and characterisation of network faults, estimation of their possible impact, and an analysis of the effectiveness of network redundancy (Palanikumar and Ramasamy, 2018).

Different parameters of the network are associated with faults, such as unknown protocol rate, utilising rate, error rate of input, error rate of output, drop rate of input, drop rate of output, interface operating status, and interface admin status. The network fault patterns with which these characteristics are associated are incompatible protocol, insufficient network bandwidth, traffic jams, insufficient interface buffers, physical faults in the cables, incorrect configuration of interfaces, and physical faults in an interface (Li et al., 2008). Fault detection and isolation (FDI) play an important role in fault diagnosis in sensor networks. Principal Component Analysis (PCA) is one of the techniques used to detect and isolate faults. Sensor fault diagnosis involves the identification and classification of faulty sensors. Approaches such as Auto-associative Neural Networks (AANN), Partial Lease Squares (PLS), and Fisher Discriminant Analysis (FDA) are also used for the classification of fault patterns (Sharifi and Langari, 2011).

2.4 Current Types of Network Fault Classification

Network faults can be of different types. Machine Learning (ML) is therefore needed to automate the classification of faults so that they can be managed and counter measures or implement contingency plans employed to improve the health of networks. Machine learning provides Artificial Intelligence (AI) that enables computers to learn and implement well informed decisions. ML focuses on two aspects: improving through experience and dealing with the laws that govern learning systems (Jordan and Mitchell, 2015).

In supervised learning approaches, new data is classified based on already labelled data or training data (Classification and Regression, while unsupervised learning approaches find structure in unlabelled data (Clusters) and groups of data objects based on their similarity with each other (Lin, 2015) see Figure 2.1. K-Means is an example of unsupervised learning (Velmurugan, 2014). Clustering and classification are the two machine learning approaches widely used in classifying network faults. Clustering is the process of grouping of similar objects. All objects in a cluster should have high intra-cluster similarity and minimal inter-cluster similarity. Clustering is a form of unsupervised learning that does not require a training dataset. Classification, on the other hand, is used to predict labels for unlabelled objects. It is a form of supervised learning that uses a training dataset to train the classifier (Qiu and Sapiro, 2015). These two machine learning approaches are frequently used in the classification of network faults. The supervised machine learning process is depicted in Figure 2.2.

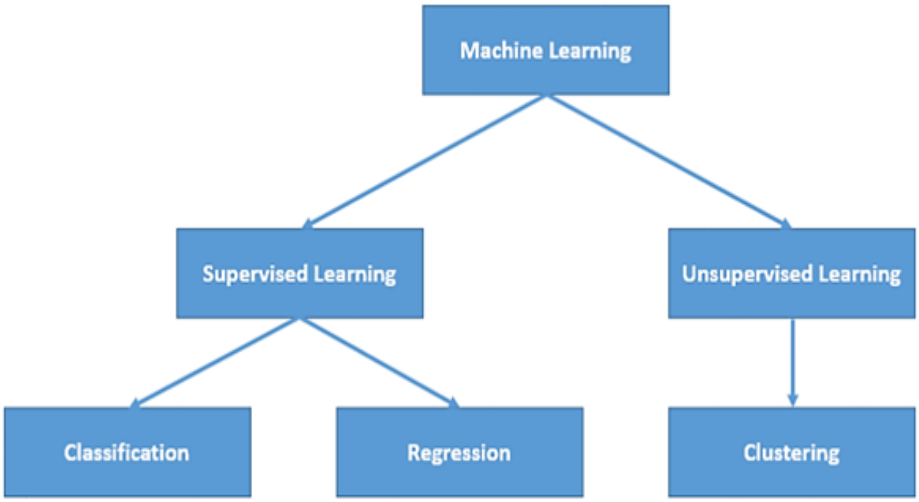

Fig. 2.1 Machine learning – supervised vs. unsupervised

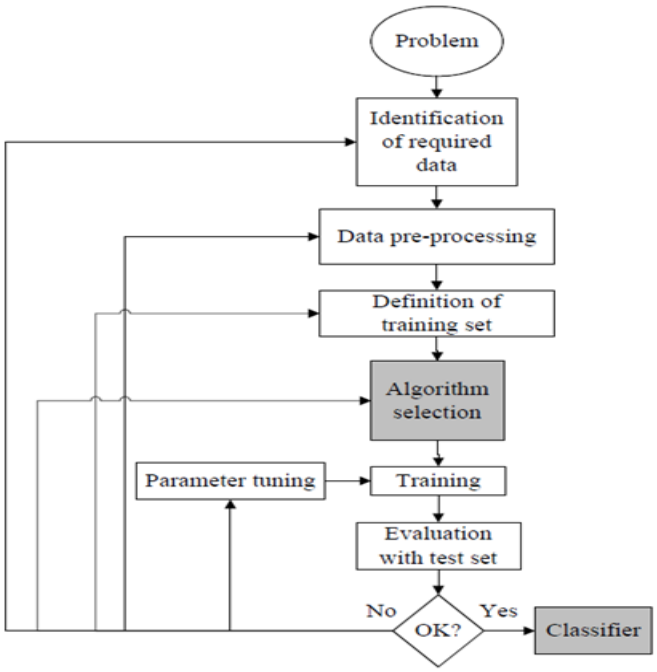

Fig. 2.2 Supervised machine learning

So the outcome of the supervised machine learning is a classifier that can be used to classify network faults or any other objects. The classifier is evaluated prior to further use. Similarly, unsupervised learning is used to group objects with high similarity. This is achieved without using any labelled data or a training dataset. It is difficult to find latent patterns based on the features of data and data objects are grouped together. Fuzz C Means is another example of unsupervised learning (Kasim et al., 2015). The workflow of unsupervised learning is presented in Figure 2.3 In unsupervised learning, data are analysed and a model then constructed that is composed of different clusters based on their features and similarity. Once the model is built, it continues to be updated when new data are added to reflect the new clusters. This phenomenon is known as unsupervised machine learning (Hastie et al., 2009). Subsequent sections will review literature on the clustering and classification methods used in the classification of network faults.

2.4.1 Clustering Methods

A cluster is a group of objects with similarities and clustering is one of the most widely explored areas in data mining. Many definitions of clustering exist in the literature and there is no one precise definition (De et al., 2016). A plethora of techniques for clustering are therefore available. These can be classified into different types, as shown in Figure

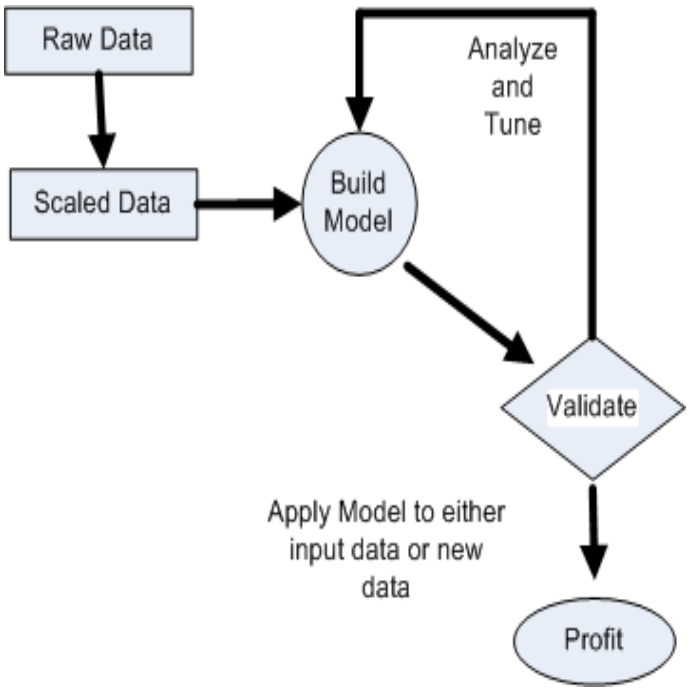

Fig. 2.3 Workflow of unsupervised learning

2.4, namely hierarchical clustering methods, partition clustering methods, density-based clustering methods, model-based clustering methods, grid-based clustering methods, and soft-clustering methods.

Each category of clustering methods has a different approach depending on their modus operandi and utility. More details on hierarchical and partitioning clustering methods are given in subsequent sections.

2.4.1.1 Hierarchical Clustering

This is a deterministic approach to clustering given data objects. It is not dependent on defining the number of clusters beforehand (De et al., 2016; Wang et al., 2015). Hierarchical clustering is a method that can be subdivided into many other clustering methods including agglomerative, divisive, single link, complete link, and average link. These approaches differ in their determination of the cluster formation. The most widely used clustering scheme is agglomerative hierarchical clustering. This is mostly used for embedded classification schemes, as discussed by (Kumar et al., 2014). Hierarchical clustering is a recursive approach in which clusters are constructed in a top-down or bottom-up fashion. The resultant clusters then form a hierarchy. The two main types of hierarchical clustering are agglomerative and divisive.

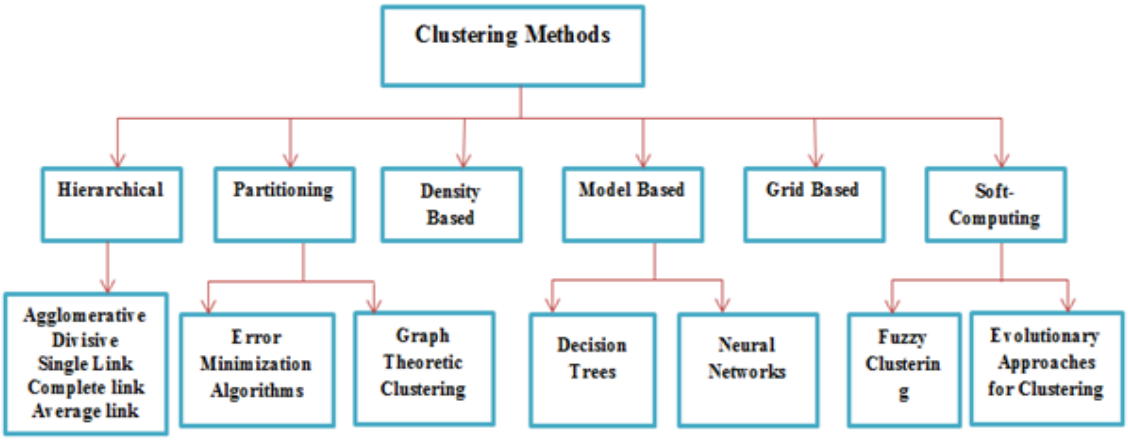

Fig. 2.4 Summary of clustering algorithms

In the agglomerative approach, each object is initially treated as a cluster. Based on their similarity, different objects are then merged into one or more clusters. An important criterion is that similar objects are grouped into a cluster. In the divisive hierarchical approach, all objects are initially formed into a single cluster. They are then verified for similarity and divided into different clusters. The result of divisive clustering is a nested group of objects. However, different measures of similarity can be employed. One such measure is used to determine whether two objects are similar. Depending on the utility of the similarity measure, hierarchical clustering can be divided into three different types: average-link clustering, complete-link clustering, and single-link clustering (Chang et al., 2011). Versatility and multiple partitions are the strengths of hierarchical partitioning. However, Wu et al. (2008) notes that the downside of hierarchical clustering methods are less scalability and a lack of backtracking capability. Agglomerative hierarchical clustering was used by Guralnik and Foslien (2009) to automatically classify faults in the known abnormal behaviour of model-based processes.

2.4.1.2 Partitioning Clustering

Partitioning clustering is an approach in which similar objects are grouped by relocating objects among clusters. This follows the process of initial clustering (Äyrämö and Kärkkäinen, 2006; Hung et al., 2005). This means that the user needs to provide information to the clustering method about the number of clusters. Partitioning clustering algorithms are broadly classified into two types; graph-theoretic clustering and error minimisation algorithms. As the name implies, error minimisation algorithms reduce the error rate while clustering. These algorithms employ the Sum of Squared Error (SSE) to achieve error minimisation. Graph

theoretic methods, on the other hand, use graphs when producing clusters. Important examples of graph theoretic methods are the Minimal Spanning Tree (MST) which was described by Zsebők et al. (2018), and Limited Neighbourhood Sets, which was discussed by Chang et al. (2011). A fault classification model was built by Enrico et al. (2008) using fuzzy logic and was based on the partitioning clustering method.

2.4.1.3 Density Based Clustering

This is a type of clustering where a cluster is formed with a maximal set of density connected points. Thus, clusters are nothing but dense regions in data space and are separated by regions that exhibit low object density. This method therefore discovers randomly shaped clusters from a given dataset. An example of density-based clustering is the Density Based Spatial Clustering of Applications with Noise (DBSCAN) which groups together points that are close to one another. Points in low-density regions are therefore considered outliers. This algorithm requires two arguments which are the minimum number of points denoted as minPts and the maximum radius of the neighbourhood from the given point p denoted as ε. Density is then computed based on these two parameters (Patwary et al., 2012). This is illustrated in Figure 2.5 where it is assumed that the value of minPts is 4. The density of point p is considered high

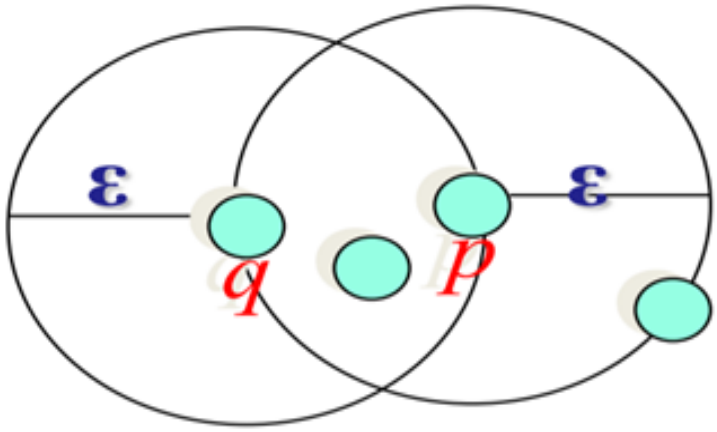

Fig. 2.5 Illustrates high and low density points

as there are a minimum of four points in its neighbourhood. Similarly, point q is considered low density because there are only 3 points in its neighbourhood. For DBSCAN, every point in space can be classified into three types: core points, border points, and noise points. Core points are those that satisfy the minPts parameter in the neighbourhood while border points have fewer points than the minPts parameter. A noise point does not fit the criteria for core

or border points (Ertöz et al., 2003). Assuming the core points is set as 10 and minPts as 4, the functionality of DBSCAN is shown in Figure 2.6.

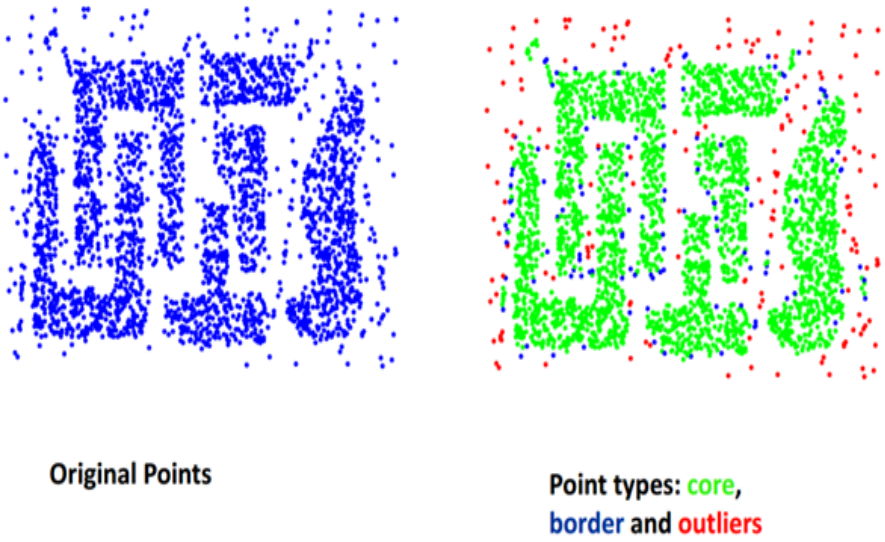

Fig. 2.6 Differentiating core points, border points, and outliers

Another concept associated with the DBSCAN algorithm is density-reachability. An object A is directly density-reachable from object B if B happens to be the core object and A is in the neighbourhood of B. Similarly, in Figure 1 the point q is considered directly density-reachable from p. However, p is not directly density-reachable from q. Therefore, density-reachability is asymmetric in nature. DBSCAN is resistant to noise and effective at handling clusters of different sizes and shapes. However, it is sensitive to parameters and it is thus difficult to find the most suitable values for these parameters (He et al., 2011).

Wang et al. (2017) proposed a density-based clustering approach for the detection of communities. The algorithm takes community structures as input and uses a heuristic approach to group points and identify communities. Zhang et al. (2016a) proposed a density-based clustering algorithm that exploits a manifold distance measure. This measure captures both local and global spatial information while making clustering decisions. The algorithm was found to be effective with datasets containing noise overlap, multi-scale, and diverse density features. Hahsler et al. (2017) explored the use of both DBSCAN and Ordering Points to Identify the Clustering Structure (OPTICS) in the density-based clustering of objects. They employed a KNN search mechanism on a KD-tree data structure to improve performance.

2.4.1.4 Model-Based Clustering Methods

In model-based clustering, M clusters are created. An object thus belongs to one of the M clusters, although it is difficult to determine which. Clustering is based on the formal model that has been built which needs to be updated for every new object. Expectation Maximisation (EM) is one example of model-based clustering. This is an iterative clustering method that determines the maximum likelihood of parameters. It creates a random initial model and then converges this into a more meaningful model. EM is a widely used clustering approach in data mining, computer vision, and machine learning (Moon, 1996). It can also be used to solve parameter estimation problem.

Hasnat et al. (2015) proposed an evolutionary clustering method that is also model based as it makes use of multinomial mixture models. It can be used to study different parametric models as well as allowing for the temporal evolution of resultant clusters. Singh et al. (2016) explored differences in performance among various clustering methods, including those which are model based. Model based methods provide the maximum likelihood with respect to statistical evolution. Sanse and Sharma (2015) claim that model-based clustering methods optimise the best fit between a mathematical model and given data. In such methods, Maximum Likelihood Estimation (MLE) is used to determine the parameters of the model.

Sanse and Sharma (2015) state that EM uses a statistical, mixture density model while neural network models like the Self Organising Map (SOM) adopt a "winner takes all" approach to model based clustering. Krueger et al. (2018) chaired a conference in which authors came up with a solution to achieve model-based clustering using a sequential dirichlet process mixture (DPM) where multivariate skew t-distributions was employed.

2.4.1.5 Grid-Based Clustering Methods

Grid-based clustering refers to a grid of rows and columns used to organise objects in cells based on the density of each cell. Cells whose density is less than the threshold are eliminated. Clusters are formed from contiguous groups of dense cells. There are many advantages of this approach, for example, distance computations are not required, complexity is reduced, and clustering is performed on summaries rather than on individual objects. It is also easier to locate nearby clusters. Grid-based clustering methods make use of a multi-resolution grid data structure. The number of grid cells can have an influence on clustering complexity (Joshi and Kaur, 2013). There are numerous algorithms that utilise this approach, including the Statistical Information Grid (STING) approach proposed by Wang et al. (1997) and CLIQUE which was proposed by Agrawal et al. (1998).

STING divides the spatial area into rectangular cells, each of which is at a different level according to their resolution. Each cell can also contain further partitions and is equipped with statistical information stored a priori to facilitate faster processing of queries. The parameters of low level cells are used to compute the parameters of high level cells with ease. The parameters required by STING include min, max, count, mean, and s, as well as distribution parameters such as uniform and normal. Spatial data queries are answered using a top-down approach. Query processing with STING is an iterative process that, while making clustering decisions, starts from a pre-selected layer with a small number of cells until the bottom layer is reached. Cloud boundaries cannot be diagonal but are either vertical or horizontal. CLIQUE is another grid-based clustering algorithm that can automatically identify high dimensional data placed for better clustering. This algorithm can be considered both a grid-based and density-based algorithm. It divides each dimension into the same number of equal length intervals and partitions m-dimensional space into rectangular, non-overlapping units. The clusters formed using CLIQUE are a maximal set of connected dense units. The accuracy of CLIQUE may, however, be degraded although its approach is simple (Khan et al., 2014).

Chang et al. (2009) proposed a grid-based clustering method known as the Axis-Shifted Grid-Clustering (ASGC) algorithm. This combines the features of density and grid-based approaches and employs an axis-shifted partitioning strategy. This strategy is used to find high density within a given input space. Amini et al. (2011) explored grid-based clustering algorithms that make use of density-based concepts in the clustering process, especially in the context of data streams. Among the algorithms they explored were DUCStream, D-Stream I, DD-Stream, D-Stream II, and PKS-Stream. A useful survey of clustering algorithms, including those which are grid-based, was recently conducted by Kaur et al. (2016).

2.4.1.6 Soft Computing Methods

Soft computing (SC) involves the fusion of different methodologies and is a computing phenomenon in which a guiding principle such as tolerance to partial truth, imprecision, and uncertainty is used to achieve a low-cost and robust solution with tractability. The diversified techniques used in soft computing include Bayesian statistics, fuzzy logic (FL), artificial neural networks (ANNs), machine learning (ML), and evolutionary computing (EC). Methodologies are fused so that they can be used to solve real world problems that are too difficult to solve without SC. In contrast, conventional computing is known as hard computing (HC). SC is fundamentally an optimisation solution, which can refer to either

maximisation or minimisation depending on the context in which optimisation is performed (Gan and Tao, 2015).

- **Fuzzy Clustering Means (FCM)**

  This is a clustering technique derived from the K-Means algorithm. The main difference between FCM and K-Means is that the latter refers to hard clustering while the FCM is used for soft clustering. FCM is therefore a type of fuzzy clustering that is referred to as soft computing. It produces soft clusters in which an object may belong to more than one cluster with a different probability. This approach makes use of a distance measure that operates on objects placed in an n-dimensional vector (Yang, 1993). It is an algorithm that is widely used in the data mining domain, especially for pattern recognition. Dunn (1973) originally developed this algorithm which was then improved later by Bezdek (1981).

  FCM solves real world problems due to its fuzziness and the intelligent control it has over the process concerned. It has a wide range of applications, including the classification of network faults and patterns. It uses Euclidean distance which is a distance measure used to make similarity decisions. The algorithm initially creates cluster centres denoted as $SC_0 = Cj_{(0)}$ and set $p = 1$. The algorithm then computes cluster centres and updates their membership. This is achieved using the following equation.

$$u_(i,j = [(d_i j)\frac{1}{m} - 1 \sum_{i=1}^{k} [\frac{1}{d_{ij}}]\frac{1}{m} - 1] \tag{2.1}$$

  It then computes a cluster centre to form new cluster representatives using the following equation.

$$C_j(p) = \frac{\sum_{i=1}^{N} u_{ij} m X_i}{\sum_{i=1}^{N} u_{ij} m} \tag{2.2}$$

  Finally, the algorithm checks condition such as whether $||C_{j(p)} - C_{j(p-1)}|| <$ for $j = 1$ to k otherwise set $p+1 \rightarrow p$ and then moves on to the second step. Steps 2 and 3 of FCM involve major computational complexity. Irfan et al. (2009) claimed that FCM can be optimised further when it uses a pre-processing step involving Canopy Clustering as this can also can reduce computational complexity. With canopy clustering, FCM can speed up the clustering process based on canopies already generated in the pre-processing phase. McCallum et al. (2000) observed a 25% reduction in computational time when FCM was preceded by canopy clustering.

Chang et al. (2011) conducted experiments with FCM and derived two more algorithms to evaluate performance. These algorithms are CDFKM and MCDFKM. Newton et al. (1992) developed a hybrid network named Adaptive Fuzzy Leader Clustering (AFLC) by combining a neural network and fuzzy clustering. This involves embedding FCM into the control structure of a neural network to make AFLC. The empirical results showed that the AFLC algorithm was able to produce arbitrary data patterns. Watanabe and Imaizumi (2001) performed fuzzy modelling using hyperbolic Fuzzy K-Means while Kau et al. (2006) performed lossless image coding, also using Fuzzy K-Means. Kau et al. (2006) found that a non-linear predictor exhibited superior performance to a linear predictor. Liao et al. (2009) employed FCM for the characterisation of biomedical samples which resulted in good decision making in medical diagnoses. Yu et al. (2006) then proposed a new sampling technique and used FCM based on clonal optimisation to improve clustering performance over enormous databases. Du et al. (2008) used a fuzzy clustering approach for performance fabric clustering which has associated physical and mechanical properties. Li et al. (2006) proposed fuzzy K-Modes and compared these with Fuzzy K-Means to show they exhibit superior clustering performance. Chakrabarty and Roy (2018) proposed a new algorithm based on FCM known as Agglomerative FKM which facilitates the automatic selection of cluster members as well as improving the accuracy and time complexity of the clustering process to detect plagiarism effectively. Hoang et al. (2014) proposed a fuzzy clustering method in the form of a protocol for a Wireless Sensor Network (WSN). This algorithm was used to make the network energy more efficient. Sulaiman and Isa (2010) employed an FCM for image segmentation which enhanced the visual quality of the resultant images. Sivarathri and Govardhan (2014) explored the utility of FCM over K-Means and found FCM was better for image segmentation. Rajini and Bhavani (2011) improved FCM using Kernelized Fuzzy C Means to examine MRI brain images with segmentation. They found that their kernelized version of FCM exhibited a superior performance. Park (2009) employed Kernel Fuzzy K-Means clustering for speech feature extraction which exhibited improved performance in terms of speech recognition. Zhang et al. (2016b) conducted extensive experiments with FCM and derived new clustering algorithms such as Rationale Fuzzy C-Means, None Euclidean Relational Fuzzy C-Means, and Any Rational Fuzzy C-Means. They therefore improved the sensitivity and efficiency of the clustering process. de Vargas and Bedregal (2011) then proposed CKMeans which is a hybrid algorithm. This algorithm combines the features of K-Means and Fuzzy K-Means to improve the

accuracy of effective clustering. Zhang and Xu (2015) proposed an Agglomerative Fuzzy C Means to enhance effective classification using underlying decision cluster classifiers. Rong et al. (2011) clustered Wikipedia articles in a cloud environment using both K-Means and Fuzzy C-Means by utilising Apache Mahout which provides algorithms for distributed programming. Honda et al. (2012) proposed a Fuzzy C-Lines algorithm from K-Means with an improved distance measure. This improved the efficiency of the clustering process. Svetlova et al. (2013) proposed a fuzzy clustering algorithm known as the Minkowski Metric Fuzzy Weighted K-Means to handle high dimensional data while performing the fuzzy clustering of objects.

- **Similarity Measures Used For Clustering**

  In every clustering approach it is imperative to have a similarity measure. This section explores the different similarity measures available. Wu et al. (2008) grouped these measures into distance measures and similarity measures. The distance between two objects denoted as x_i and x_j is represented as $d(x_i, x_j)$. Distance measures are used for ordinal attributes, binary attributes, numeric attributes, nominal attributes, and mixed-type attributes. According to (Han et al., 2001), the Minkowski metric is best used with numeric attributes where the distance is computed as follows:

$$d(x_i, x_j) = \left(\left|x_{i1-x_{ji}}+\right| g \left|x_{i2} - x_{j2}\right| g + \ldots + \left|x_{ip} - x_{jp}\right| g\right) \frac{1}{g} \tag{2.3}$$

  With respect to binary attributes, a contingency table is used for the distance measure. A simple matching coefficient is used when attributes are symmetric, as presented in Equation 2.4, while asymmetric attributes make use of the Jaccard coefficient, as shown in Equation 2.5.

$$d(x_i, x_j) = \frac{r+s}{q+r+s+t} \tag{2.4}$$

$$d(x_i, x_j) = \frac{r+s}{q+r+s} \tag{2.5}$$

  In the case of nominal attributes, a simple matching approach is used, as shown in Equation 2.6. In the case of original attributes, mapping is conducted with numeric attributes, as shown in Equation 2.7.

$$d(x_i, x_j) = \frac{p-m}{p} \tag{2.6}$$

$$Z_{i,n} = \frac{r_{i,n^{-1}}}{M_n^{-1}} \tag{2.7}$$

In the case of mixed type attributes, the measure shown in Equation 2.8 is used, which combines multiple distance measures.

$$d(x_i, x_j) = \frac{\sum_{n=1}^{p} \delta_{i,j}^{(}n) d_{i,j}^{(}n)}{\sum_{n=1}^{p} \delta_{i,j}^{(}n)} \tag{2.8}$$

Similarity measures, as the name implies, are used to find the similarity between objects. These measures include the dice coefficient measure, the extended Jaccard measure, the Pearson correlation measure, and the cosine measure. The similarity between two objects ranges between 0.0 and 1.0. One indicates 100% similarity while zero indicates no similarity. Cosine similarity is computed as shown in Equation 2.9.

$$S(x_i, x_j) = \frac{x_i^T . x_j}{\|x_i\| . \|x_j\|} \tag{2.9}$$

The Pearson correlation is computed as shown in Equation 2.10.

$$r = \frac{n(\sum xy) - (\sum x)()\sum y}{\sqrt{[n\sum x^2 - (\sum x)^2]\,[n\sum y^2 - (\sum y)^2]}} \tag{2.10}$$

The Extended Jaccard similarity measure is computed as shown in Equation 2.11.

$$s(x_i, x_j) = \frac{x_i^T . x_j}{\|x_i\| 2 + \|x_j\| 2 - x_i^T . x_j} \tag{2.11}$$

The Dice coefficient measure is computed as shown in Equation 2.12.

$$s(x_i, x_j) = \frac{2x_i^T . x_j}{\|x_i\| 2 + \|x_j\| 2} \tag{2.12}$$

These measures have a distinct utility when making clustering decisions. However, Bonner (1964) notes that it is difficult to derive a universal definition that reflects good clustering. Nevertheless, determining the quality of clusters is possible, and this can

be of two types: external (structure) and internal (compactness). The internal criteria include edge cut metrics, C-Criterion, category utility metric, Condorcet's criterion, minimum variance criteria, scatter criteria, and the Sum of Squared Error (SSE). The external criteria include the rand index, precision, recall, and mutual information-based measures Bonner (1964).

2.4.2 Classification Methods

Classification is the process of predicting the class of an unlabelled object based on training data in which the membership of objects is known. As shown in Figure 2.2, classification is a supervised learning method and therefore a form of machine learning. There are numerous supervised learning algorithms, including logic-based algorithms, perception-based algorithms, statistical learning algorithms, instance-based learning algorithms, and Support Vector Machines (SVMs). Specific examples of learning-based algorithms include Decision Trees (DT), Neural Networks (NN), Naive Bayes, kNN, SVM and rule learners (Kotsiantis et al., 2007) see Figure 2.7.

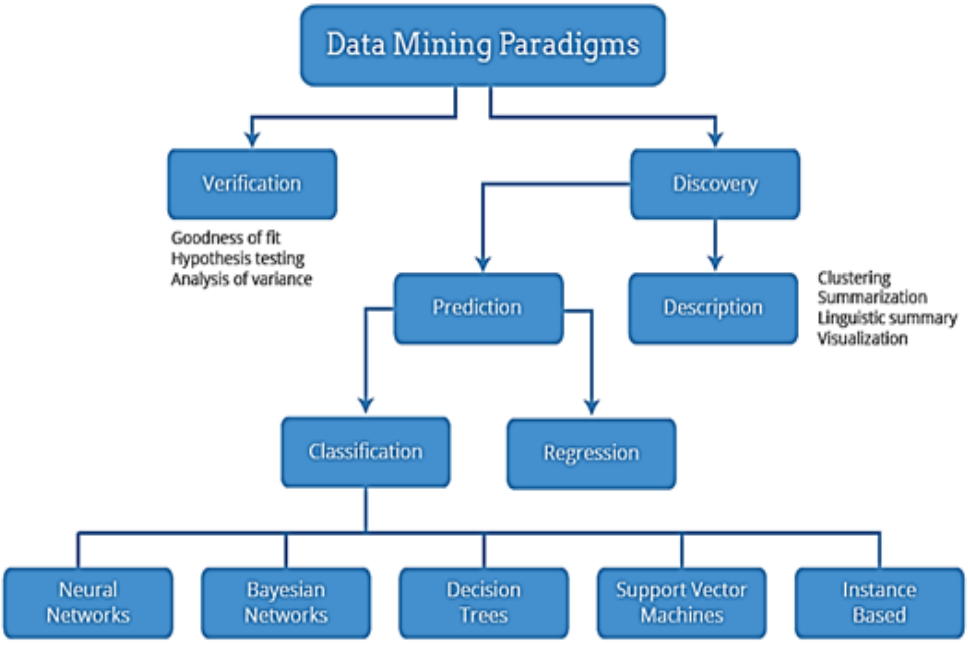

Fig. 2.7 Data mining paradigms including classification methods

A neural network is a computational model that simulates the connections between axons in the human brain. It consists of numerous units known as artificial neurons and employs

a machine learning approach to solving complex problems. A Bayesian network, on the other hand, is a classification technique that employs a directed acyclic graph (DAG). It is a statistical model where a set of random variables are involved in the process of classification. The decision tree is a tool for supporting decision-making. It is represented as a tree-like graph that shows all the decision variables and all the potential courses of action. SVM is a classification technique that makes use of learning algorithms. Instance-based learning methods are associated with a set of learning algorithms involved in memory-based learning.

Classification techniques are widely used in various domains. For example, Tuan et al. (2015) employed supervised learning in the general virology domain to classify viruses. Subsequent subsections review the following supervised learning methods: kNN, DT, statistical learning algorithms, and instance-based learning methods.

2.4.2.1 Nearest Neighbour Classifier

The nearest neighbour classifier is one of the machine learning algorithms used to predict the labels of objects given for testing. Nikhath et al. (2016) used the kNN classifier for text categorisation. Text categorisation refers to the process of inputting texts or documents to different categories based on their underlying content. It involves two phases: training and classification. In the training phase, trained documents are provided upon which KNN constructs a model that can be used to classify test documents, as illustrated in Figure 2.8.

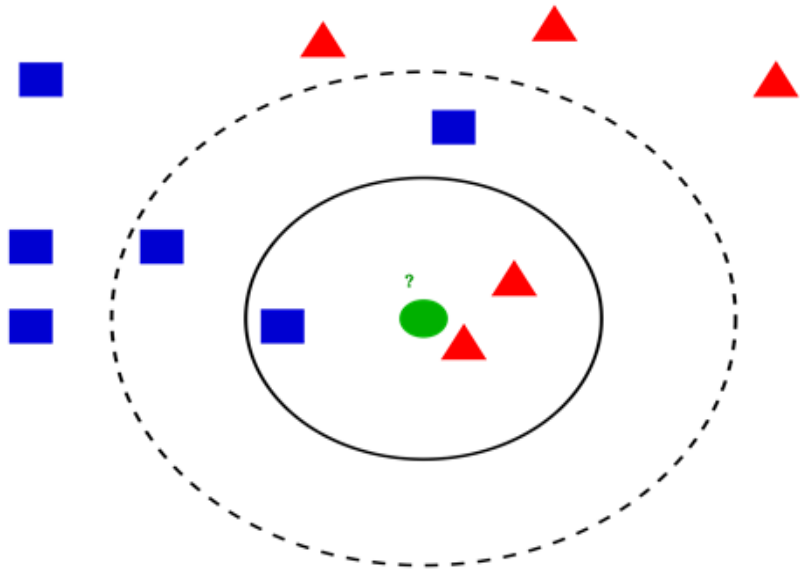

Fig. 2.8 The functioning of kNN

Anava and Levy (2016) claimed that the k-Nearest Neighbour is a non-parametric algorithm that can be widely used for machine learning and pattern recognition. They employed kNN and used optimal weights to efficiently find neighbours in the given dataset. A variant of kNN, weighted kNN, uses three ingredients; the weight vector, number of nearest neighbours, and a distance metric. Bhamre et al. (2016) performed image classification using multiple pipelined methods, including kNN, where the Mahalanobis distance measure is used to compare images. Hoffer and Ailon (2016) explored the use of metric embedding for deep learning using a semi-supervised approach. Their approach can be used to construct classification models. They also used nearest neighbour classification to predict class labels based on trained labelled samples.

2.4.2.2 Decision Tree Classifiers

The Decision Tree is a widely used classification technique. It employs a series of intelligent questions that are used to build a classification model. The outcome of the model is a decision tree, which is used to understand trends in the data and make well-informed decisions. Ashari et al. (2013) investigated the utility of the decision tree and compared its performance with other classifiers. They found that DT was faster than kNN and Naive Bayes. Examples of DT classifiers are ID3, C4.5, and CART. Gallé (2018) proposed an ensemble method to form random forests with the help of decision trees. They thus exploited the power of decision trees in making automated decisions.

The decision tree algorithm C4.5, shown in Figure 2.9, is an iterative process with a condition inside. The condition verifies the criteria in given instances. If the condition is satisfied, a new decision path tree is constructed. If the condition is not satisfied, it adopts child-link values for the completion of the same decision path. Decision trees provide utility, resource costs, event outcomes, and possible consequences.

2.4.2.3 Statistical Learning Methods

Statistical learning is a method that can be used to find predictive functions from data. Understanding and predicting are the two important goals of statistical learning methods. Liu et al. (2017) employed statistical learning methods to solve a learning problem where a random variable Y is used to predict another random variable X and its observations. They used a hypothesis to evaluate the quality of the predictions. With the help of SVMs, Joachims (2001) proposed a statistical learning model that would be able to conduct text classification.

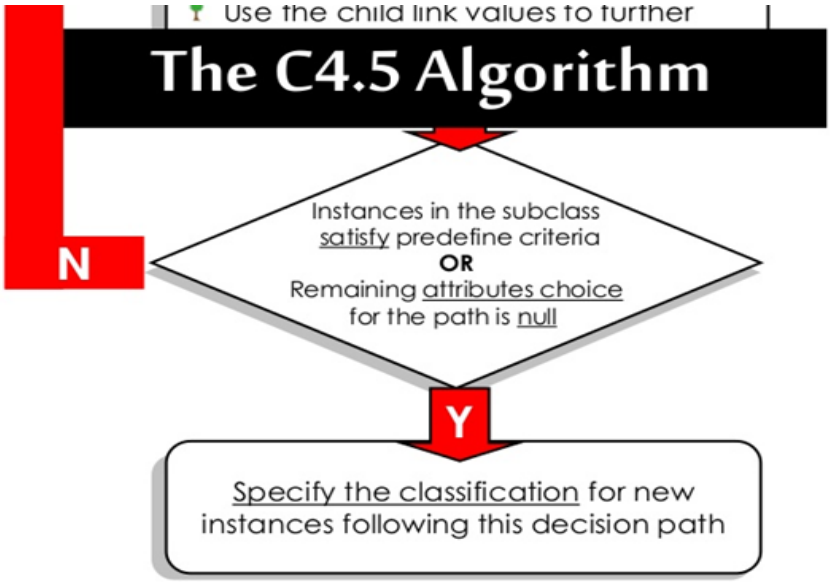

Fig. 2.9 The process underlying the decision tree algorithm C4.5

This involved handling features of text classification such as high-dimensional feature space, sparse document vectors, the heterogeneous use of terms, and a high level of redundancy.

As shown in Figure 2.10, statistical methods are generic and can be used with datasets from different domains. Pre-processing steps such as Extra-Cellular Recordings (ECR), functional Magnetic Resonance Imaging (fMRI), and Electroencephalography (EEG) are performed before utilising statistical methods such as feature sensitivity, feature selection, and classification.

2.4.2.4 Instance-Based Learning Methods

Instance-based learning is also known as memory-based learning. Instead of explicit generalisation, it encompasses a family of algorithms that are used to compare new instances with instances that are seen in training. Brighton and Mellish (2002) investigated instance-based learning algorithms and exploited technical advances in the process of instance selection. They developed ways to identify noisy instances and prune them to improve the performance of applications where instance-based learning methods are employed. New advances in the selection of instances include selection as removal and the structure of instance space. They found that structure of class plays an important role in instance-based learning algorithms. Figure 2.11 illustrates the concept of instance-based learning.

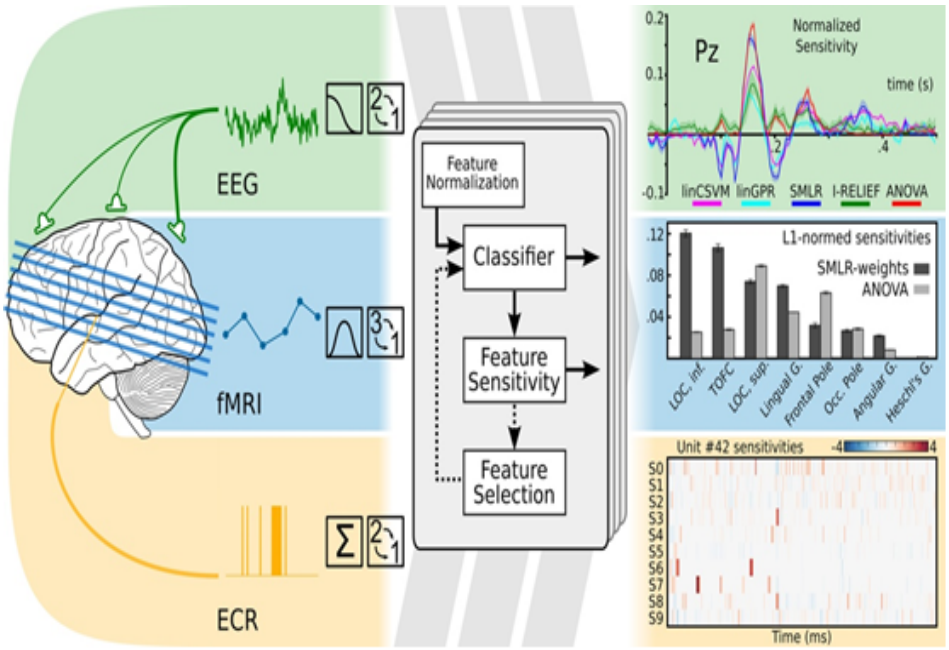

Fig. 2.10 General statistical learning methods

Toussaint and Berzan (2012) contended that instance-based learning algorithms are among the best machine learning algorithms used for pattern classification. However, these algorithms suffer from the curse of dimensionality. This problem is overcome by using proximity-graph based instance-based learning which exploits minimum spanning trees (MST), relative neighbourhood graphs, and Gabriel graphs. Li et al. (2016) employed instance-based learning for image processing, where it comprised a form of reinforcement learning (RL). Gershman and Daw (2017) used reinforcement learning paradigms with the help of instance-based learning. This involves decisions based in experience and RL plays a vital role in such decision making.

2.4.2.5 Support Vector Machine

The Support Vector Machine (SVM) is one of the supervised learning methods and is therefore a discriminative classifier. It uses labelled training data to classify new samples that are unlabelled and constructs a hyperplane which is designed to have regression or classification. Some of the applications of SVM include text categorisation, classification of images, image segmentation, character recognition, identification features, and making predictions. Ghamisi and Hofle (2017) proposed a framework for composite kernel SVM to classify LiDAR data with high accuracy and reduced resource consumption. Nakayama et al. (2017) employed SVM in high-dimension settings for having bias-corrected SVM for

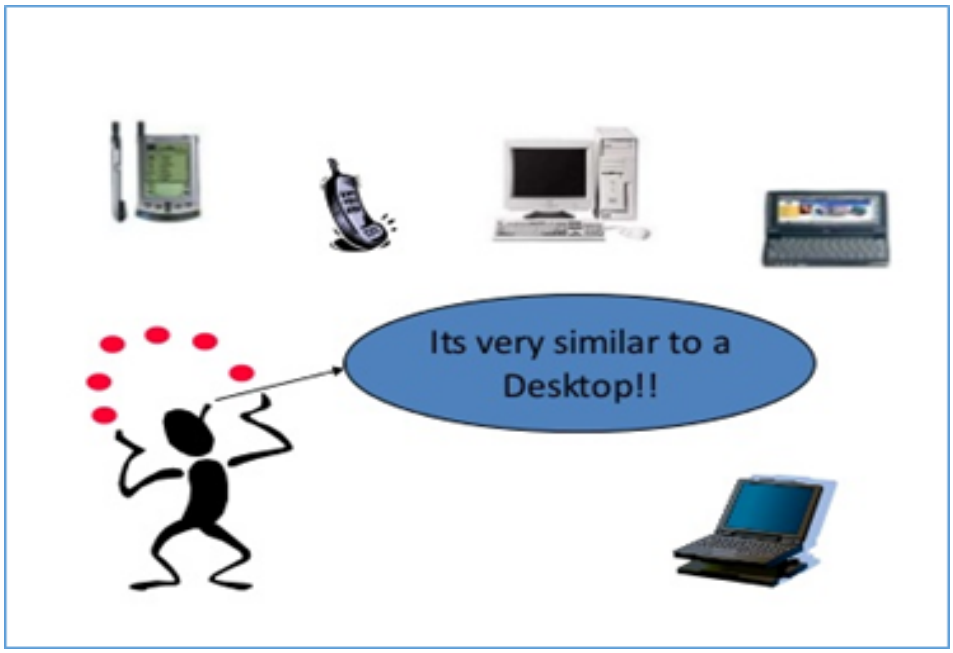

Fig. 2.11 Instance-based learning

classification. Chakrabarty et al. (2017) used SVM to construct a feasible non-linear model predictive controller that provides guaranteed stability. Nandi et al. (2017) employed SVM for the classification of data in flood management.

2.4.2.6 Neural Networks

A neural network is a computational model with interconnected nodes that resemble the neurons in the human brain. The network contains multiple layers including an input layer, a hidden layer, and an output layer. The computational model proposed by McCulloch and Pitts (1943) paved the way for research on neural networks. Goudar and Buonomano (2018) explored how neural networks could be used to convert spoken words into hand-written text. Masood et al. (2017) studied the utility of such networks in facilitating the automatic identification of vehicle colour, make, and model. Poria et al. (2017) employed convolution neural networks to investigate human emotions and thoughts extraction from video content. Ferrari et al. (2017) also used convolution neural networks and bacterial colony counting for digital microbiology imaging.

- **Probabilistic Neural Network**

  The Probabilistic Neural Network (PNN) is based on the estimation of Probabilistic Density Function (PDF) and Bayesian classification. PNN was first introduced by

Specht (1990), since then, it has become widely used to solve numerous classification problems based on its underlying statistical foundations in Bayesian estimation theory. PNN is largely similar to the Parzen Window PDF estimator. For each class, the sub-network present in PNN is nothing but a Parzen Window PDF estimator. Therefore, PNN relies on Parzen window classifiers that are associated with a non-parametric procedure. Each classifier is able to make a decision after computing the PDF for each class based on the training examples given. One such classifier decision is as follows.

$$P_k f_k > P_j f_j \tag{2.13}$$

Where P_k represents the prior probability of examples occurring from a given class denoted as k and the estimated PDF is represented as f_k. It is known as a "neural network" because it contains two-layer feed-forward network mapping.

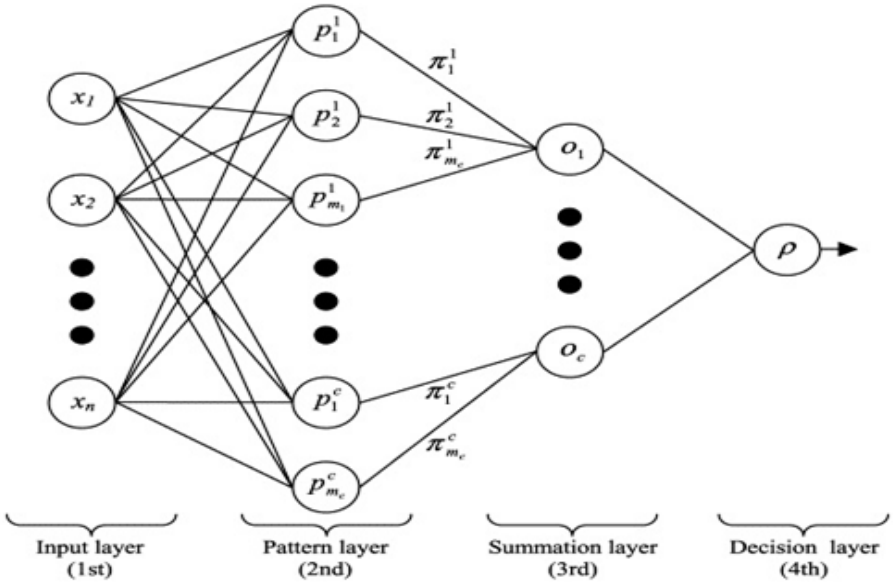

Fig. 2.12 Architecture of PNN

As shown in Figure 3.6, the PNN contains four layers: input layer, pattern layer, summation layer, and decision layer. The PNN recognises the number of classes represented by c. Input features are represented by the first layer with n features. The nodes present in the pattern layer represent the instance in the training set. The first layer is fully connected to the second layer; it is responsible for distributing inputs to the neurons and does not perform any sort of computation. The next layer is

the summation layer which is semi-connected to the pattern layer. A set of training instances is connected to one single node in the summation class. This means that the nodes in the summation class contain the sum of inputs derived from the selected training patterns. PNN creates a set of multivariate probability densities. These are derived from the inputs or training vectors given to the network. Any input instance whose category is not known is sent to the pattern layer. The nodes in the pattern layer receive input and the output of each node in the layer is calculated as follows.

$$\pi_i^c = \frac{1}{(2\pi)^{n/2}\sigma^n} exp[-\frac{(x-x_{ij})^T(x-x_{ij})}{2\sigma^2}] \tag{2.14}$$

where π denotes the features of instance x which is the input. The smoothing parameter is represented by σ and the training instance for a given category c is represented as x_{ij}. The neurons in the summation layer compute the likelihood of x which is classified as c. The computation for this is as follows.

$$P_i(x) = \frac{1}{(2\pi)^{n/2}\sigma^n} \frac{1}{N} \sum_{i=1}^{N_i} exp[-\frac{(x-x_{ij})^T(x-x_{ij})}{2\sigma^2}] \tag{2.15}$$

where N_i represents the number of samples present in c. If the probabilities and losses for incorrect decisions for each class are same, the pattern x is classified by the decision layer according to the rules derived from the output of the summation layer. This is calculated as follows.

$$C(x) = argmax[p_i(x)], i = 1,2,...,c \tag{2.16}$$

where C(x) represents the envisaged class of pattern x and the total number of classes in the given training set is represented by m. If the probabilities and losses associated with an incorrect decision for each class are different, the result of the summation layer is calculated as follows.

$$C(x) = argmax[p_i(x)cost_i(x)apro_i(x)], i = 1,2,...,c \tag{2.17}$$

where the misclassification of input vector is denoted as $cost_i(x)$. The probability of the occurrence of prior patterns in class c is denoted as $apro_i(x)$.

2.5 Fault Classification Challenges

Fault classification is an essential activity in network fault management for understanding faults and taking well-informed decisions. However, Rozaki (2015) found that this activity presents certain challenges that need to be addressed, including the dynamic and real time classification of faults, fault handling, and achieving compliance with better Quality of Service (QoS) under enforced QoS constraints. Multi-layer fault localisation is also another challenging area to be addressed. The proliferation and usage of certain technologies are insufficient in this respect. Moreover, faults are bulky which means data mining techniques need to be employed. One possible solution to overcome these problems is to couple a data mining technique with other approaches to deal with the size and complexity of data when performing fault localisation, detection, and classification. Some of the specific challenges pertaining to fault classification are as follows.

- It is by no means -trivial to have to test all MIB variables and the extent of their influence on exact fault identification.
- There is a plethora of machine learning techniques. It is difficult to determine which is most suitable for fault detection and management.
- The reliability of research on fault classification depends on the quality of datasets. It is an immense challenge to obtain datasets that provide scope for meaningful interpretations through machine learning techniques.
- The fact that a large number of MIB variables need to be investigated in the network fault management system leads to the curse of dimensionality problem. Moreover, MIB variables belong to different groups. Overcoming the high dimensionality problem is therefore another challenge.
- Automation of fault isolation is difficult for certain faults due to a lack of established procedures.

2.6 Prior Works on Fault Management and Classification

Al-Kasassbeh and Adda (2009) proposed a method that used a Wiener filter-based agent for network fault detection. The software component with intelligence that they used was the Mobile Agent (MA). MA can move across networks and obtain Management Information Base (MIB) variables from different nodes. Abnormal changes in the behaviour of MIB

variables can then be studied. Their method operates on four different datasets captured from two network scenarios. The network manager node is provided with a high level of information, including conclusions and recommendations. This level of knowhow can help the node make expert decisions. It is achieved by identifying and classifying network faults. The use of multiple mobile agents with a filtering process makes this approach efficient and scalable. Cross correlation of MIB variables is also conducted to check abnormalities in the traffic.

Al-Kasassbeh (2011a) exploited MIB variables to build change detection algorithms. Change detection can provide useful insights into any problem that might have occurred in the network. In network traffic it may detect the presence of a possible anomaly. Various change detection methodologies were explored by Al-Kasassbeh (2011a). Those applications that are of interest and related to network change management include network performance management, security management, and network fault management. However, research on the adaptive detection of change or the utility of any hybrid approach that provides better results has yet to be conducted.

Al-Kasassbeh (2011b) also investigated network attacks and explored the need for an intrusion detection system (IDS) to enable the automatic detection of network attacks. He argued that commercial IDSs are generally centralised and are not scalable. To overcome scalability issues, a distributed model was introduced based on the concept of a mobile agent. This is an autonomous program that collects information from different places in the network. A Weiner filter is used to investigate issues with network data and detect anomalous behaviour. Statistical methods and the Weiner filter can be combined to enhance the detection of network attacks. The algorithm was tested for its effectiveness in both lightweight and heavyweight scenarios. Its main drawback was that it did not work well with heavy traffic. Further investigation is therefore required to find an IDS that can perform well in heavy traffic scenarios.

Wang et al. (2008) studied faults in aero-engines and proposed a method for the detection and classification of faults that would work in conjunction with self-recovery. When combined with other approaches this method has made a significant contribution to fault classification. These methods include Support Vector Machine (SVM), Wavelet Packet Analysis (WPA), and Stochastic Resonance (SR). Building multiple fault classifiers paved the way for ensuring effective fault identification and classification.

Zhang et al. (2011) investigated network-based nonlinear systems in the presence of random missing measurements and communication constraints. They proposed a fault detection filter to improve the quality of fault identification. Both fault detection and filtering

are achieved by addressing detection and filtering problems using a stochastic switched fuzzy time-delay system. Yu et al. (2011) also proposed a method for the diagnosis and classification of faults. Their method exploits two approaches known as Independent Component Analysis (ICA) and Empirical Model Decomposition (EMD) which is used for decomposing intrinsic mode functions. These functions are subjected to both aliasing and correlation. Thus, information redundancy is eliminated through better fault disclosure and fault diagnosis. The ICA method is also involved in fault classification. In this process a vector is created to provide enough information to make classification effective. The information contained in the vector includes entropy, entropy of frequency, and correlation coefficients which are subjected to the regression neural network. Youssef (2009) proposed a new approach for the real time detection and classification of faults by exploiting the effectiveness of SVM and integrating it with online wavelet-based pre-processing. They found that the good generalisation in SVM was extremely useful for the classification of faults in power transmission systems.

Yeo et al. (2003) proposed a new algorithm for classifying faults in transmission lines. They classified faults into High Impedance Faults (HIF) and Low Impedance Faults (LIF). Their algorithm is based on the Adaptive Network-based Fuzzy Inference System (ANFIS) which takes current signals into the system and identifies as well as classifies faults. A neural network structure that classifies catastrophic faults in integrated circuits is another approach used to ensure effective fault detection and classification. Many earlier techniques assumed that the variances of faults in circuits are the same. This assumption is released and found that neural networks to be slow. However, the heteroscedastic probabilistic neural network exhibited better performance in terms of fault detection and classification (Yang et al., 2000). ANN with back propagation was used by Tayeb and Rhim (2011) to locate, detect, and classify faults in transmission line system. Back propagation is one of the training approaches in ANN that optimises the results.

Saravanan and Rathinam (2012) explored fault location and classification techniques for double circuit transmission lines. These are useful for detecting different faults such as LG, LLG, and LLLG and protecting networks from such faults. They proposed neural network solutions with three feed forward variations which were the Radial Basic Function Network (RBF), Back Propagation Network (BPN), and Cascaded Correlation Feed Forward Network (CFBPN). The proposed method extracts a hidden relationship from input patterns to locate, detect, and classify faults. This approached worked well with different system parameters. RBF was found to be the most effective of these variations.

Dasgupta et al. (2012) proposed an algorithm for the classification of network faults in underground cables. They used a hybrid approach that combines Probabilistic Neural

Network (PNN) and Discrete Wavelet Transform (DWT) and improved the accuracy with which faults were classified. The different types of fault they classified were the three-phase fault, line to line fault, double line to ground fault, and the single line to ground fault.

Xu and Chow (2008) studied fault diagnosis. They proposed a hybrid algorithm for fault diagnosis based on artificial immune systems and fuzzy classification. This contains features such as computation of support and confidence, rule cleaning, and classification. Their method is known as the Fuzzy Artificial Immune Recognition System (FAIRS). FAIRS performed better than the E-algorithm and Artificial Immune Recognition System (AIRS).

Nagaraja et al. (2003) explored the performance and availability of cluster-based services. They employed analytical modelling and fault injection techniques to evaluate the performance and availability of networks that use protocols, such as VIA and TCP. They conducted empirical research on different fault loads and found that VIA based networks showed the greatest availability. The faults that were injected were network hardware faults, node faults, application faults, and memory exhaustion faults. With both protocols they found 99% to 99.8% availability on cluster-based services. However, they acknowledged that their research requires further validation.

Jaudet et al. (2005) proposed temporal classification as a method for predicting fault-predication in telecommunications networks. Using this method, alarm messages are analysed and fault-predication made using temporal classification. To ensure temporal classification, temporal rules must be defined and used. The temporal version of the Decision Tree (DT) classifier was compared with the non-temporal DT classifier. The results showed that the temporal DT classifier performed better than its non-temporal counterpart. However, this method needs to be improved further to consider the downtime caused by each fault in the multi-vendor environment.

Mohamed and Basir (2010) studied computer networks and proposed a distributed alarm correlation technique that used a fusion-based approach for fault identification. Their work is significant as thousands of alarms are generated by networking systems. When alarms are correlated using the fusion-based approach, it is possible to have accurate fault identification. An intelligent agent is used to correlate local alarms globally. By fusing local alarms into global alarms and finding the representative faults, it is possible to reduce fault data to ensure better fault diagnosis and classification. Mohamed and Basir conducted their experiments in noisy network environments. Multiple intelligent agents were used to correlate local alarms produced by different domains in a managed network. The detection rate for this method was higher than for its predecessors.

Barakat et al. (2010) proposed a method for fault detection and identification for large

scale industrial plants. The method is known as Input-Output Classification Mapping (IOCM). It makes use of RBF neural networks where the output of the input layer is mapped to the input of the output layer. After decomposing the data, both techniques are used to ensure the accurate identification of faults. The IOCM was compared with RBF and IOCM+DWD systems and both IOCM and IOCM based systems exhibited higher performance than RBF systems.

ElMadbouly et al. (2010) proposed a fault detection method based on Fuzzy Cluster Means (FCM) and Centre of Gravity (COG) defuzzification. For the signal separation, FCM is used while defuzzification is used for fused signal generation. The components included in the system are sensor measurements, FCM algorithm, fusion engine, a fault detector unit, and a performance index calculator. Due to fuzzy membership in the clustering process, this method exhibited superior performance in terms of fault detection.

Khalid et al. (2011) proposed a hybrid mechanism for fault detection and classification in two tank process industries. They used techniques such as Genetic Neuro-Fuzzy Systems and the Kalman Filter. Their hybrid approach was named the Genetic Adaptive Neuro-Fuzzy Inference System (GANFIS). The novelty of this approach lay in discovering critical information that indicates the presence or absence of a fault at the earliest possible time. Additionally, it provides fine-grained details of the fault that is detected. This makes fault diagnosis more accurate and reliable. The error rate of the method is considerably less than that of ANFIS, ANN and Fuzzy methods.

Benkaci et al. (2011) explored a novel approach for feature selection named fuzzy-ARTMAP whose efficiency lies in the classification of faults in an automotive application. It is a neural network approach for pattern classification that utilises both conflict intersection and classification for the isolation and classification of faults and enhances performance. The discriminative feature is employed for conflict intersection which determines the actions taken for classification and fault isolation. Although the efficiency of the system was demonstrated through simulations, it needs to be applied to a real-time scenario to fully demonstrate its benefits.

Harmouche et al. (2012) proposed a hybrid approach based on Kullback-Leibler Divergence and Principal Component Analysis (PCA) for fault detection and diagnosis which is applied in Hoteling T2 system. In this approach, a diagnosis criterion is employed from Kullback-Leibler Divergence while the PCA is conducted through control charts. In terms of performance, this method was able to detect small faults that went undetected by a Hostelling test. Once again, although the effectiveness of the method was shown through simulations it needs to be tested on a real time system.

Malik and Mishra (2014) employed a data mining tool known as RapidMiner to extract the most influential features. These were then subjected to PNN for fault diagnosis. The feature selection with PNN showed highest classification performance in terms of accuracy when compared with FL, IEC, and AFC. The data mining approach has therefore been shown to be efficient and will be further enhanced to provide online fault diagnosis in power transformers.

Wang et al. (2014) proposed a method for fault detection and classification at the multi-sensor feeder level using two techniques known as SVM and PCA. The method yielded high accuracy in both fault detection and classification. Because it was applied to the sensor networks, it needs to be improved further by determining sampling frequency, the optimal number of sensors, and reliability by increasing the number of sensors used in empirical testing.

Flores-Martos et al. (2015) studied mobile networks for Key Performance Indicators (KPI) and alarms. They explored Long Time Evaluation (LTE) networks because the size and complexity of mobile networks is continually increasing. The Bayesian networks technique was used for fault diagnosis. Their system comprised two phases. In the first phase, Expectation Maximization (EM) was used for the discretisation of input data while in the second phase call status was identified. Faults in LTE networks were then diagnosed. The results were evaluated and compared with different discretisation methods such as hierarchical classification, K-means clustering, and EM algorithm. A high success rate was achieved with the EM algorithm.

Zhang et al. (2016c) proposed a Fuzzy K-Nearest Neighbour (FK-NN) algorithm for locating and classifying faults in transmission lines. They found that fuzzy K-NN exhibited better performance than traditional K-NN. However, further improvement is required because, as network faults grow larger in size, traditional computational power will be insufficient. It is therefore essential to use online big data analytical tools such as Spark and Storm to ensure comprehensive classification results.

Aggarwal et al. (2016) proposed a method for classifying network faults in transmission lines. They considered ten categories of shunt faults and employed a combination of two techniques to form a hybrid mechanism. The two techniques were Probabilistic Neural Network (PNN) and Empirical Mode Decomposition (EMD). Intrinsic Mode Functions (IMFs) were used to model faults with the help of EMD to form an intelligent fault detection model. Higher classification accuracy was then achieved.

The article by Chen et al. (2017), describes research carried out by the authors into an efficient intrusion detection system for mobile ad hoc networks (MANETs) implementation

into cloud-based storage environments. The paper proposes an FCM technique in the form of an efficient fuzzy clustering-based algorithm which seeks to detect possible attacks. The authors concluded that the new model successfully identified malicious had improved performance over that of previously used models.

The patent filed by Cardente et al. (2017), describes and illustrates their invention; a computing system which delivers a classification technique to correctly identify and manage network types. The invention is illustrated in several embodiments of how the system can be implemented in a network. In one case it in used to monitor the activity of a network and classifying and identified unknown endpoint type. The embodiment of the invention in another case is used to monitor behaviour of an endpoint and examine the endpoint against its expected behaviour. Additional embodiments of the system include, but are not limited to; computer networks, apparatus and articles of manurfacture.

The research paper by Swain and Khilar (2017), describes the study carried out by the authors on fault diagnosis in wireless sensor networks (WSNs). A new neural network-based fault diagnosis algorithm is presented in this paper, which aims to help WSNs to manage a composite fault environment. The algorithm seeks to achieves this through four stages which include; clustering, communication, fault detection and classification. Conclusions from the paper show that's the new protocol is superior in comparison to existing algorithms in case of fault detection accuracy, false alarm rate and false positive rate.

Oliveira et al. (2017), delivered in their research their findings on Fault Detection Diagnosis (FDD) built on Weightless Neural Networks (WNN) and their use in univariate and multivariate dynamic systems. The paper presents a new system which executes three stages on the input; time series mapping of data, detection and diagnosis, and finally passing through a clustering filter. The research outlines findings from two tests the system was put through which involved two simulated cases; monitoring temperature of a sales gas compressor for natural gas and against data from an industrial plant. Conclusions from the research show that the proposed system solved the problem of Fault Detection and Diagnosis in the cases of multivariate and univariate dynamic systems.

A research article by Khokhar et al. (2017), presents findings of the authors around the subject of successful classification of Power Quality Disturbances (PQDs). The paper presents an algorithm using a Probabilistic Neural Network (PNN) based classifier simultaneously with an Artificial Bee Colony (ABC) optimization technique, which the researchers sought to provide a new method to automatically classify both single and hybrid PQDs and ultimately recognise the origin of disturbances. Findings from the simulation carried out in the research concluded, that the innovative PNN-ABC approach successfully singled out both single and

multiple PDQs. Thereby proving itself to be a superior method of classification against methods used previously.

In their research article, Zhang et al. (2018) offer their findings around the topic of variable performance of intelligent fault diagnosis methods in relation to the challenges of changing workloads and work environment noise. The paper proposes a new model built on deep learning which the researchers aim to combats both problems with the additional aid of data augmentation. The paper concludes that the model, named Convolution Neural Networks withTraining Interference (TICNN), was found to deliver great accuracy on normal datasets when exposed to changing workloads and noisy environment. This differed from to that of a Deep Neural Network (DNN) model which deteriorated under the same conditions.

2.7 Summary and Research Gaps

This chapter has comprehensively reviewed current literature on the cutting-edge techniques used for network fault management. It has defined network faults, network fault management, and different fault management techniques, in the process elucidating various aspects of network faults and their classification. The review focused on machine learning approaches such as supervised learning and unsupervised learning methods. It explored different clustering methods that fall under the rubric of unsupervised learning. These include hierarchical clustering, partitioning clustering, density-based clustering, model-based clustering, grid-based clustering, soft computing methods, and Fuzzy Clustering Means (FCM). Each of these clustering techniques needs to use different measures to find the similarity between objects and make clustering decisions. The different measures discussed include the cosine similarity measure, Pearson correlation, extended Jacquard, and the coefficient measure. The review then covered supervised methods, otherwise known as classification methods. These include Nearest Neighbour (NN), Decision Tree (DT) classifier, statistical learning methods, instance-based learning methods, neural networks, SVM, and PNN. The challenges involved in network fault classification and prior work focusing on network classification were also discussed. Various insights from the literature were presented and research gaps were identified. Some of the specific research gaps were as follows.

- Most of the research articles found in the literature focused on fault classification in a range of environments that were not necessarily computer networks.
- Some researchers explored the detection of faults but not their classification. Similarly,

articles addressing fault classification focused on one or two types of faults while ignoring many others.

- Some of the articles did not take MIB variables into consideration, yet these play a vital role in the detection and classification of faults.
- Some techniques were presented solely in terms of their advantages regarding in fault management. Their limitations were not explored in relation to different types of faults captured from different network scenarios.
- Some articles provided research insights based on existing analytical tools and their features rather than taking a holistic approach to the identification and classification of network faults.

FCM is considered better than its predecessors for clustering. The rationale behind this claim centres the ability of an object to belong to multiple clusters with a certain degree of probability. This type of soft computing has high utility in real-world applications where accuracy is afforded great importance. Another widely used algorithm, known as the PNN algorithm, is a feed forward neural network that is also very useful for classifying network faults. It makes use of the Probability Distribution Function (PDF) of each class to make classification decisions. Although the PNN can classify network faults, it needs an efficient pre-processing step that can only be provided by FCM. In this thesis a hybrid approach is therefore proposed for network fault management. The proposed hybrid technique is Subtractive Fuzzy Probabilistic Neural Network (SFPNNC). This exploits the soft clustering power of FCM and the classification efficiency of PNN to provide a more accurate and efficient classification of network faults. The rationale for considering FCM is as follows.

- FCM can handle issues arising from unclear boundaries (big data) between clusters break down into smaller segments that provide more chances to implement soft clustering decisions.
- It can shed light on the high dimensionality problem.
- It is an iterative algorithm used to optimise the clustering of objects into different clusters. It is suitable for network traffic analysis, especially fault management.

The desired features of the propose hybrid approach therefore include: faster training than BP networks, guaranteed convergence (better than Bayesian networks), facilitation of incremental training with consistent speed and the robustness to deal with sample noise.

Chapter 3

Methodology and Framework Design

It is possible to acquire an effective performance classifier through the combination of separate algorithms in order to produce classification outcomes with superior quality. In recent years, researchers into community classification have increased their focus on the classification of fault systems. In general, ensemble learning can be explained as a machine-based learning system that comprises a group of separate learner models that employs a decision fusion approach to amalgamate each of their results and ultimately generates one answer to a specific issue. In this context, the fundamental concept involves the combination of different experts and the effective use of the outcomes generated by each of these experts within the group. Hence, the resultant ensemble model should be capable of amalgamating the positive and negative attributes of each of the individual components and then effectively applying the fusion strategy to obtain augmented outcomes. Through the process of consolidating the outcomes of each individual model, it will be possible to increase the overall performance level. In recent years, numerous scholars have chosen to utilise a mixture of different classification algorithms in their work with the goal of improving the performance in comparison to single classifiers As a result of diverse factors in the process of classification, including insufficient training data, the disadvantages of feature extraction, the filtering of data and learning algorithms, the performance quality can be constrained. Various researchers have demonstrated that the precision of ensembling classifiers exceeds that recorded by the optimal individual classifier (Ho et al., 1994; Kittler et al., 1998; Xu et al., 1992).

This chapter will present the fundamental concepts involved in the combination of distinct supervised (Probabilistic Neural Network Classifier- PNNC) and unsupervised (Fuzzy Clustering Means- FCM) algorithms required to generate classification outcomes with higher quality. Additionally, in terms of unsupervised algorithms, a further technique (Subtractive Clustering- SC) is injected with FCM in order to enhance the iteration stages

and to determine the ideal cluster centres as well as the initial cluster identification of the faults. An additional development is administered in order to generate optimal training data founded on the sensitive features that are affected by the fault traffic.

3.1 Fuzzy Theory

The majority of researchers on modern projects are dependent on "yes or no" instead of "more-or-less" decisions. However, it is not always the case that problems have to take the form of "yes or no" or "true or false". Furthermore, it is sometimes observed that there are instances where the problem is fuzzy and obscure. These problems can be resolved by applying fuzzy theory (Bezdek, 1981). A fuzzy theory system has the capability to produce more accurate outcomes for any classes that lack precision by using 33 mathematical explanations for vagueness. The central functionality of a fuzzy system is based on "IF-THEN" rules, which is a system based on knowledge. The fuzzy theory has developed over time after the initial concept was effectively proposed by Zadeh (1965). The primary tenet of fuzzy logic is found on various basic components that have significant importance in fuzzy modelling. Each component or observation in a fuzzy system has a connection with a membership degree, meaning that each component is assigned a membership value.

The principle objective of the membership function is to transform the group of real data values into vague datasets, in which every individual datapoint will be connected with a proper degree of every input variable within the fuzzy control system. This can be achieved by a set that is formed as a mixture of (A, m), where A represents the set and m the membership degree, $m : A \rightarrow [0, 1]$. In order to explain the ambiguity of a data set to the greatest extent possible, the design of systems based on fuzzy theory is dependent on discrete or continuous membership functions that use IF-THEN rules.

Within computer traffic faults data, there is uncertainty and inaccurate information that emerge from the different variables related to Management Information Base (MIB) groups, which are defined as the group of network entities that can be controlled through the use of Simple Network Management Protocol (SNMP). Methods of hard clustering, including k-means and Self Organizing Maps (SOM), are not compatible with the examination of such data, as the traffic faults clusters are frequently observed to intersect as a result of the involvement of multiple MIB variables in each distinct flaw. There are various positive aspects of fuzzy theory in terms of its approach to managing data that contains a degree of uncertainty. Fuzzy clustering methods are compatible with the theory of fuzzy sets, which takes uncertainty into account when the data features of Simple Network Management

Protocol (SNMP) are examined. Hence, every feature vector within the dataset will be assigned a membership value with each of the groups, thus implying the degree to which it belongs to the cluster (Bezdek, 1981). The aim of fuzzy clustering is to provide a definition for every cluster by determining its membership function, and it is beneficial for resolving the issue of high dimensionality. It is regarded as an iteratively optimal algorithm that has greater flexibility, particularly for the examination of traffic.

3.2 Fuzzy C-Means

Fuzzy C-Means (FCM), which is an adaptation of Hard C-Mean clustering, was originally proposed by Bezdek (1981). It is defined as an unsupervised machine learning clustering approach that can be applied to numerous diverse problems, including the design of classification, clustering and feature analysis. FCM is predominantly applied by considering the distance between different data points, while the generation of every cluster is designed based on the measurement of each distinct from the cluster centres. Thus, there will be a centre associated with every group and through optimisation of its location, it will be possible to determine the optimal amount of clusters. Similar to other clustering methods, FCM is predominantly dependent on calculating the distance between different data points and employs the measure of Euclidean Distance to determine similar features among objects. FCM is one approach to fuzzy logic that enables each datum to be associated with multiple data groups. Calculation of the distance between data points facilitates the process of the algorithm making decisions regarding the creation of sets for each of the data points based on their similarities and dissimilarities. Those data points that are considered to have similarities are maintained in a group defined as a cluster. The value of the similarity measures ranges between 0.0 and 1.0, where the value of 0.0 indicates strong dissimilarity and a value of 1.0 is an indication of the strongest similarity between the relevant objects.

The primary goal of FCM is to reduce Jm, as shown in the equation below with the aim of obtaining the optimal number of clusters. As this approach is iterative, it is important that the minimisation is improved at each iteration.

The Euclidean Distance Function, which is defined as the objective function J_m, is utilised in FCM to obtain the C partition $A = A_1, A_2, A_3, \ldots, A_n$ for the specific dataset $A = X_1, X_2, X_3, \ldots X_n$ as well as the volume of clusters, defined as c.

$$J_m(\mu, V : X) = \sum_{i=1}^{c} \sum_{j=1}^{n} (\mu_{ij})^m \left\| X_j - V_i \right\|^2 \tag{3.1}$$

Where μ_{ij} represents the membership function of the data point in the i cluster, m represents the fuzzifier that serves as the fuzziness controller value; in general, m can be a real number over 1. This parameter acts to manage the degree of vagueness of every obtaining clusters that has a membership represents the centre of the i cluster, while $||.||$ indicates the Euclidean distance between the prototype and the vector.

Equation 3.1 is employed to calculate the value of $J_m(U,V)$ as well as to ascertain the criterion function depending on a given threshold. Objects are denoted by the letter n, while clusters are represented by the letter c, the i cluster centre denoted by V_i and the ith cluster of the membership of an object by Xi is denoted by

$$0 \leq u_{ij} \leq 1 and \sum_{i=1}^{c} \mu_{ij} = 1 \tag{3.2}$$

$\left\|X_j - V_i\right\|^2$ represents the Euclidean distance between the prototype Vi and the vector X_i. The fuzziness of the subsequent partition is constrained by the parameter m. This denotes the level of distribution of the membership of each of the faults considered in the present study, which are separated into (F1, F2, F3 and F4) via the clusters. In this scenario, the letter F depicts the separate traffic in each of the scenarios with diverse sources. Further explanation of this will provided in the following chapter of this thesis. Consequently, the FCM data develops into a hard cluster. Additionally, the outcome is that the cluster j are used by the prototypes V_i, thus ensuring that the fuzziness of the partition is optimised. Users are required to determine the minimal change values in the objective function in addition to the parameter m in order to enable termination and to achieve the maximum number of iterations.

The two partial derivatives are assigned the value of zero with the purpose of identifying the closed form formulas for updates. This facilitates the acquisition of pattern prototypes and the fuzzy partition matrix after the iteration has converged (Bezdek, 1981). In order that the outcomes of each iteration in the process are optimised, both the fuzzy membership and the fuzzy cluster centroids are updated by applying the following equations:

$$\sum_{i=1}^{c} \mu_{ij} = 1 \tag{3.3}$$

The total of all the memberships in each of the clusters must equate to 1, as indicated by the equation shown above. Subsequently, the cluster centres and membership level will be updated at each iteration, which will enable the optimisation of the number of clusters using the following equations:

$$V_i = (\sum_{j=1}^{n} (\mu_{ij})^m X_j)/(\mu_{ij})^m \tag{3.4}$$

$$\mu_{ij} = [\sum_{k=1}^{c} (\left\|X_j - V_i\right\|^2)/(\left\|X_j - V_k\right\|^2)^{(\frac{1}{(m-1)})}]^{-1} \tag{3.5}$$

In FCM, data are attached to each cluster through a membership function, which depicts the algorithm's fuzzy behaviour. The algorithm constructs a suitable matrix U containing matrix components that range between 0 and 1, which denote the level of membership that exists among the data and cluster centres. Figure 3.1 is an illustration of a mono-dimensional example, depicting a one-dimension dataset that is spread along the x-axis.

The above data set can be conventionally grouped into multiple clusters. By determining a threshold at a point on the x-axis, the data can be largely divided into two separate clusters; although it is also possible to have three or four clusters, the optimum number is two. In the figure, the subsequent clusters have been highlighted by red circles. In this scenario, each data point that belongs to the data set is assigned a membership coefficient of either 1 or 0, which are illustrated in the figure by the addition of the y-axis. In regard to the k-means

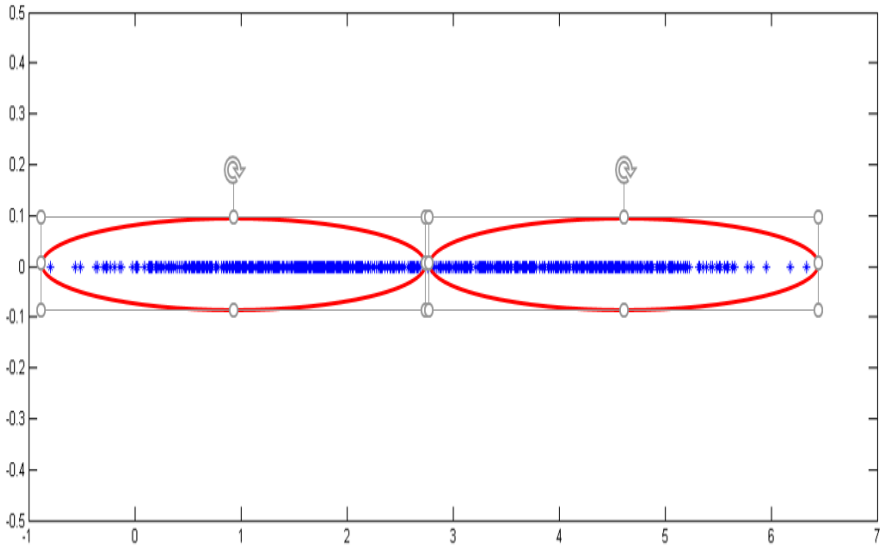

Fig. 3.1 Distribution of mono-dimensional data

algorithm, each of the datum is linked with a certain centroid. A pair of clusters (Cluster 1 and Cluster 2) can be determined in the vicinity of the two concentrations of data. Figure 3.2 depicts the membership degree for k means, which is defined as hard clustering.

In the fuzzy clustering process, every data point can be a member of different clusters. If the definition of the membership coefficients is not specifically constrained to 1 or 0, it

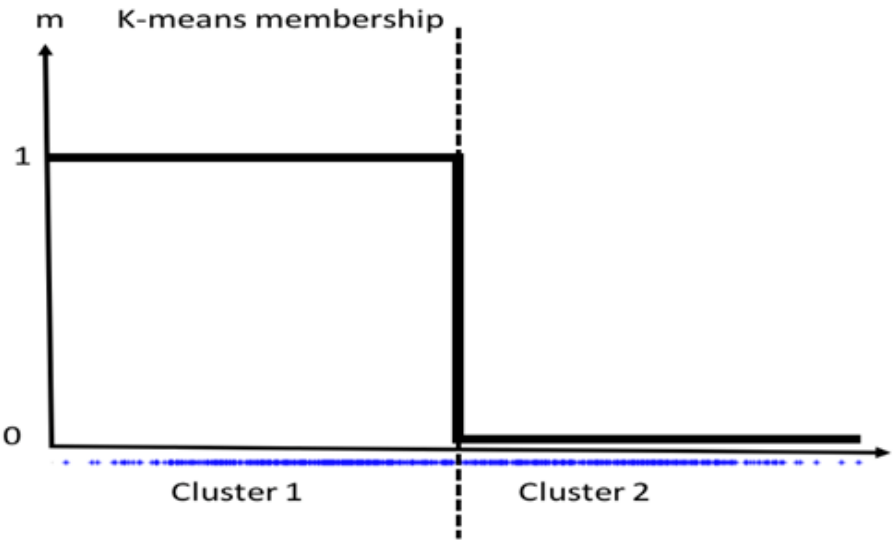

Fig. 3.2 k-means membership degree

is possible for the coefficients to have any value between 1 and 0. In Figure 3.3, the data from the aforementioned clustering is shown, although in the case, fuzzy c-means clustering has been used. Firstly, the two clusters can now be re-defined by the creation of a different threshold value. Subsequently, each data point will be assigned new membership coefficients that are based on the clusters' centroids, in addition to the distance of the data point from the centroid.

As demonstrated in Figure 3.2 the central data point belongs to both Cluster 1 and Cluster 2. The data point's membership coefficient for Cluster 1 is calculated as 0.5.

3.2.1 Initialization

In FCM, disparate initialisations can cause diverse outcomes as it only will only converge to local minima. The traditional method applied to prevent FCM being constrained in local minima is to apply the FCM with various initialisations and the outcome with the lowest value function is selected. Nevertheless, such a process can be lengthy with a certain level of instability. In recently conducted studies, FCM has been combined with optimisation algorithms, including Subtractive Clustering, Self Adaptive Clustering, the Genetic Algorithm, Particle Swarm Optimization, and Ant Colony Optimization (Ghosh and Acharya, 2011; Loginov et al., 2011; Mehdizadeh et al., 2008).

The subtractive clustering technique is an extrapolation of the mountain clustering approach that uses data sets generated by utilising the mountain function, thus forming

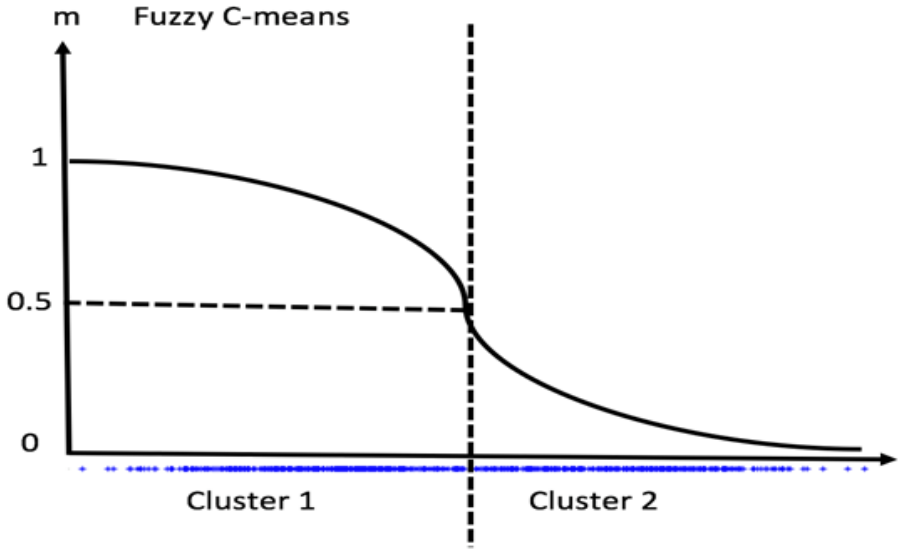

Fig. 3.3 Fuzzy c-means membership level

cluster centres. The functionality of this method involves the assemblage of n data points in an m-dimensional space and begins with the establishment of a data point xi as a possible cluster centre, with the defined potential as a function of the Euclidean distances of the entire data points. Thus, the potential at data point x_i is identified by the function through the following equation:

$$P_i = \sum_{j=1}^{n} e^{-\alpha \left\| x_i - x_j \right\|^2} \tag{3.6}$$

Where α is a parameter given by:

$$\alpha = 4/r_a^2 \tag{3.7}$$

In Equation 3.7, r_a represents the cluster radius, which determines a hypersphere of data points that can be regarded as neighbours based on their influence on the cluster centre potential. The value of r_a has a significant impact on the number of clusters that are induced. It is important to note three observations in relation to this value: a data point that that exists outside the vicinity of radius r_a only has a minimal impact on the potential of the relevant neighbour centre data point; a higher value of ra generally leads to the creation of a reduced amount of clusters and therefore the model will be overly generalised; however, a low value of r_a could suggest that there will be an excessive volume of clusters, thus leading to the scenario where the model is not adequately generalised. In general, r_a values are selected to

ensure that that there is a sufficient volume of clusters, which are linked with the subsequent amount of fuzzy rules. It is possible to fit the r_a parameter depending on the complexity of the targeted model and its generalisation capability. After the potentials of all data points have been determined, through Expression 3.6, the subtractive clustering technique denotes the first cluster centre to be the data point that has the greatest potential. Subsequently, it is possible to re-assess the potential of this data point by utilising the assignment provided by:

$$P_i \Leftarrow P_i - P_1^* e^{-\beta \left\| X_i - X_1^* \right\|^2} \tag{3.8}$$

In Equation 3.8 P_1^* represents the potential value of the first cluster centre, X_1^* is the location of the centre and β is a parameter calculated as follows:

$$\beta = 4/(\delta r_a)^2 \tag{3.9}$$

In Equation 3.9, δ represents the squash factor that determines the neighbourhood of data points that can assert a considerable calculable diminishment in the potential value. Generally, a value of $\delta = 1.5$ represents a suitable option. After determination of the initial cluster centre and each of the possible data points has been re-assessed by employing the process detailed by assignment 3.8, the data point that is revealed to have the greatest potential is selected as the second cluster centre. Normally, after acquiring the k^{th} cluster centre, the potential of every data point is reassessed through the following assignment:

$$P_i \Leftarrow P_i - P_k^* e^{-\beta \left\| X_i - X_k^* \right\|^2} \tag{3.10}$$

In assignment 3.10, P_k^* represents the potential value of the k^{th} cluster centre, whereas xk^* is the centre's location. Consequently, fuzzy rules are established as a result of the selected cluster centres. The cluster approximations acquired from subclust function can be applied to initiate clustering techniques iterative optimisation (fcm) as well as model identification procedures (such as anfis). The clusters are discovered by the subclust function through the use of the subtractive clustering method. The Sklearn.cluster in the python function expands the subclust function in order to generate a rapid, one-pass algorithm to gather input-output training information. In this context, the radius of the subcluster that is acquired by integrating a new sample and the nearest sublcuster must be shorter than the threshold. If this is not the case, then a new subcluster is initiated. If this value is particularly low, then splitting will be promoted, while the opposite also holds.

3.2.2 Volume of Clusters

In the process of clustering data in the absence of any a priori knowledge of the structure of the data, it is normally necessary to make certain suppositions about how many clusters there are. For example, in Figure 3.4, it is demonstrated that it is possible to divide the data in question into five clusters while it is also feasible that the data can be grouped into four clusters. The clustering outcome is significantly dependent on the optimal amount of clusters, as different numbers of clusters can generate distinct explanations.

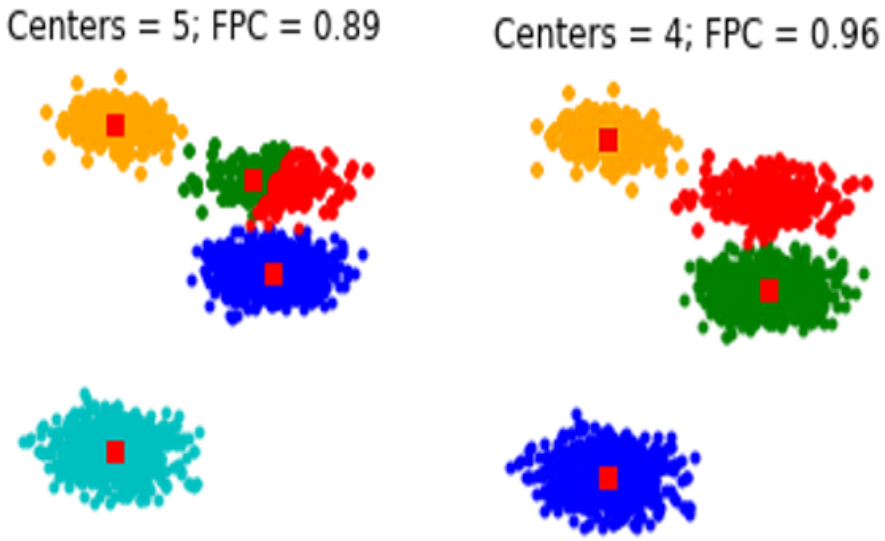

Fig. 3.4 Optimal number of clusters

3.2.3 Fuzziness Degree

The fuzziness exponent m is a parameter that has significant importance as it dictates the impact of noise on the analysis of the clusters (Bezdek, 1981). When $m = 1$, FCM develops into hard clustering and the FCM algorithm is the same as the k-means clustering. In this case, the membership values can only be zero or one. When calculating the cluster centre, each traffic fault is treated in the same way, Elevating the parameter m diminishes the impact of traffic that has a lower membership value. Fault vectors with high dimensional features are usually observed to have reduced membership values, as the associated traffic fault is not adequately represented by an individual cluster and is therefore partly allocated to multiple clusters. As m approaches infinity, all memberships μ_{ij}/j, the FCM will reach their fuzziest level and each of the clusters will dissolve. The default setting for the fuzziness exponential

Table 3.1 Fuzziness degree

Fuzziness Parameters	Fuziness Value	Cluster optimality number
M1	1.25	2
M2	1.5	3
M3	1.69	4
M4	1.85	5
M5	1.94	6
M6	2.05	7
M7	2.2	8
M8	2.45	9

is established as $m = 2$, which is the most common value used be relevant applications. In the present study, in order to optimise the partition by updating the cluster centres and the degree of membership that are reliant on this parameter, it is necessary to test different values of m, as demonstrated in Table 3.1 The data in the table shows that the optimal cluster solution for the dataset traffic is obtained with the fuzziness parameter M3 with a value of 1.69, whereas the extant data managed four separate traffics.

The ideal values for m can be different in each dataset. Various researchers have proposed methods for selecting the m values, but these approaches frequently involve a lengthy process. In empirical terms, the FCM algorithm can determine the optimal clustering outcomes through the minimisation of the objective function.

$$\frac{\partial J}{\partial m} = \sum_{i=1}^{c} \sum_{j=1}^{n} (\mu_{ij})^m ln(\mu_{ij}) \left\| X_j - V_i \right\|^2 \tag{3.11}$$

Based on Formula 3.11, the objective function is monotonically decreased as the value of m increases. It is feasible that the value of m at the point where the clustering outcome is minimised. The objective function has a minimum point in relation to the partial derivative of the objective function with regard to parameter m (Dembele and Kastner, 2003).

$$m^* = \left\{ m | (\frac{\partial j}{\partial m}) \right\} = 0 \tag{3.12}$$

In terms of fuzzy clustering, the point of inflection of the objective function is only associated with the lowest value of its derivative. Consequently, it is possible to select the ideal weighted index m^* by applying the formula shown below:

$$m^* = arg \left\{ min \left\{ \frac{\partial j}{\partial m} \right\} \right\} \tag{3.13}$$

Based on the experimental results presented in this thesis, the fuzzy exponent m for the datasets utilised is empirically selected using the equation shown in Table 3.1. Further explanation will be provided in Chapter 5.

3.2.4 Euclidean Distance

When applying clustering methods, it is generally necessary to define the distance or similar features of data in order to determine which traffic faults or samples have corresponding expression profiles. The determination of the measurement of similarity is essential for the clustering outcome, and it is dependent on the distinct attributes of the particular data framework. The distance metrics that are most frequently employed in the analysis of traffic faults expression are Pearson Correlation coefficients and Euclidean distance. The Euclidean distance calculates the difference by using the value of absolute expression, which is formulated as:

$$E(x,y) = \sqrt{\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{n}} \tag{3.14}$$

In Equation 3.14, the expression profile y_i is subtracted directly from x_i, which means that it is therefore necessary to confirm that the data faults are suitable normalised when the Euclidean distance approach is applied.

The Euclidean distance has a level of sensitivity to both outliers and noise. For example, an individual noise has the potential to change the Euclidean distance into an unrestricted value. The measurement can easily be influenced in the circumstance that the levels of expression are not evenly spread along the expression pattern. For instance, where a pair of expression patterns have a single high calculated value with the same condition, an increased correlation coefficient score will be returned, irrespective of the expression values of the remaining traffic conditions. Likewise, a significant disparity in the level of an individual expression will generate a large Euclidean distance, irrespective of the levels of the additional expressions.

3.3 K-means

K-means algorithm has been used for many years to perform hard clustering, which it performs of objects into clusters which there is similarity between them, while the dissimilar objects belong to another cluster. It is based on similarity measure, used to ensure intra-cluster faults are highly similar while inter-cluster faults are highly dissimilar. Its general

flow is as shown below:

$$j_m = \sum_{i=1}^{k} \sum_{j=1}^{n} \left\| x_i^{(j)} - c_j \right\| \tag{3.15}$$

The specific steps of the algorithm are: 1) identify initial group centroids; 2) use distance measure to identify objects that are closer to a group and assign it; 3) once all objects are assigned, recomputed the position of centroids; and 4) repeat the second and third steps until convergence.

3.4 Expectation Maximization

It is more than of an approach to deriving an algorithm for approximately obtaining maximum likelihood or maximum or maximum a posterior estimates of parameters when some of the data is missing. The general form of Expectation maximisation is built as written below:

$$p(X_i|\Theta) = \sum_{c=1}^{n} P(X_i|t_i = c, \Theta) P(t_i = c|\theta) \tag{3.16}$$

It has many benefits, including its optimisation for a large number of variables simultaneously and it gives reasonable estimates for missing values. It supports both hard and soft clustering.

3.5 Neural Networks for Fault Classification

The application of neural networks creates the potential to diminish or completely remove the necessity for complicated mathematical system that require the utilisation of significant resources and time. It is normal for data to be acquired for neural networks in real time, which can be subsequently examined using time series methods. There are certain significant drawbacks to neural networks that that are trained via back-propagation algorithms. The decision in regard to the appropriate network framework is frequently conducted on an experimental basis, whereby the amount of levels and nodes in the concealed layer are significantly dependent on manual testing. The training procedure can take a considerable amount of time, particularly if a significant volume of training data is considered. Probabilistic neural networks, which have adopted the majority of the beneficial aspects of neural networks, have also been adapted for problems involving classification. They incorporate a comparable feed-forward framework that is established in a straightforward manner based on the volume of training samples and the output classes. As the training is essentially a one-pass process, it

is a particularly efficient process. A PNN has the capability to make classification decisions based on strategy rules determined by Bayesian strategy and can additionally calculate both the reliability and probability of every classification. When probabilistic neural networks are trained ith an adequate number of training samples, their performance is comparable to neural networks that have been trained via back-propagation methods. The important benefits of PNNs are that training only necessitates a single pass, while it is guaranteed that the decision surfaces will converge on the Bayes optimal decision boundaries proportional to the increased volume of training samples. Additionally, the form of the decision surfaces can be rendered as complicated or as straightforward as required; it is normally significantly more efficient that the commonly used back-propagation approach for scenarios where the progressive adaptation of the back-propagation represents a considerable proportion of the overall calculation time (Specht, 1990).

3.6 Theory of Neural Networks

A neural network has certain similarities to a biological neural network, which facilitates connections between neurons with the goal of creating a complicated learning and interaction structure. Those NNs that are applied to applications including fault diagnosis are defined as ANNs. The fundamental structure of the network has similarities to other biological neural networks, in which there are clear levels and neuron layers between the inputs and outputs. The complexity of the complete neural network model is generally established by the amount of layers as well as the organisation of the neurons in addition to the inputs and outputs. The fundamental concept of a neural network is that a connectionist approach is utilised in the resolution of problems that occur in the real world, which enables the neural network to interact and learn during the process of its development. The general framework of the neural network is constantly changing and generally evolves as new information is learned (Montavon and Müller, 2012). Explained simply, a neural network can be considered to be the combination of different mathematical functions that can be delineated from the input X to the output Y as $f : X \rightarrow Y$.

In this scenario, the distribution could be over X or a combination of both X and Y. As well as the representation utilising functions, it is important to note that such functions could be subject to certain rules that facilitate learning. Additionally, a neural network model comprises a group of such functions that map the input to the output. It is possible to obtain these functions and the group of functions by changing the inputs and outputs to ascertain the reaction of the complete neural network model. One of the crucial aspects of any ANN

model is that the network itself forms the connection between various nodes and neurons. Dissimilar to other more simple networks, the components/neurons in a neural network possess weighted links with each other that are continually modified as new information is learned by the network. The simplest neural network comprises three clear layers that are linked by a network of neurons. The outer layer is comprised of neurons that only accept inputs and then subsequently transfers them to the next layer of neurons. This second layer of neurons is frequently described as the hidden layer, as its arrangement has not been accurately determined. This layer is followed by a third layer of neurons, which serve as the outputs of the neural network framework. Such a system is demonstrated in diagram form in Figure 3.5.

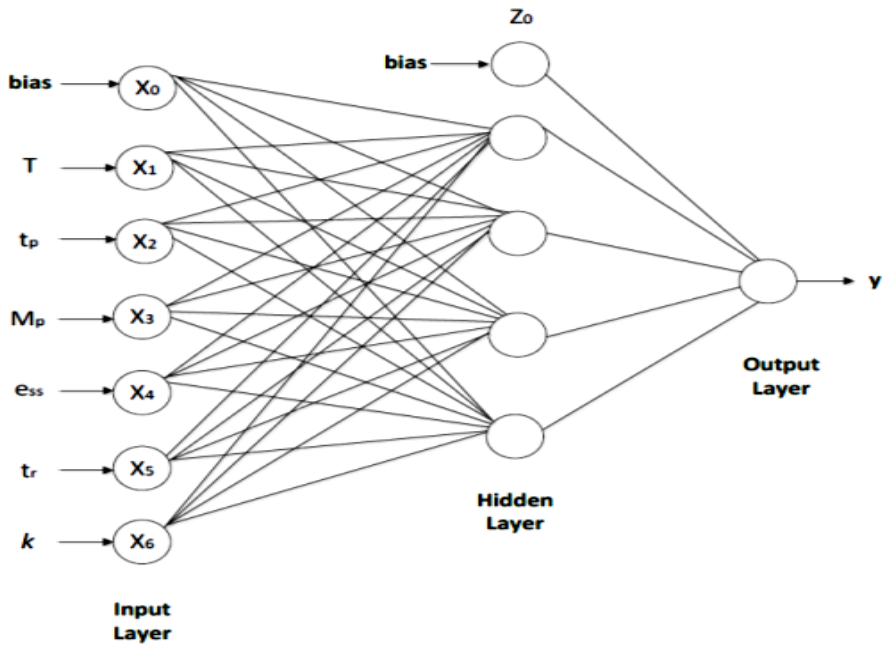

Fig. 3.5 Representation of a typical simple neural network

It is important to note that when dealing with more complicated neural networks, the second layer as depicted in the figure will contain many different levels. Consequently, the general complexity of the neural network in question will be determined by how many levels there are as well as how they are arranged. Mathematical functions are the fundamental tools used to explain neural networks in greater detail. The function of any neuron network is generally reliant on additional functions, which themselves are dependent on other functions. Such dependencies among different functions facilitate the generation of a network framework. The typical framework of a neural network can be represented in the mathematical form shown below:

$$f(x) = K(\sum w_i g_i(x)) \tag{3.17}$$

x represents the input variable (x) represents the output function $g_i(x)$ represents a function in the hidden layer that is reliant on other functions $w_i(x)$) represents the weight of $g_i(x)$ K represents the activation function

3.7 Probabilistic Neural Network

A specific application of Probabilistic Neural Networks is represented by PNNs that are fundamentally feed-forward neural networks that consist of a pattern layer as well as additional layers. A PNN is dependent on inputs that are not associated with a specific pattern. Specht was the first to develop PNNs, exemplified by his proposed high-efficiency PNN that was utilised to resolve problems with classification (Specht, 1988). The fundamental principle of a PNN is that estimations are generated in order to calculate the probability density function associated with the different categories that are determined in a classification problem. By using the Bayes Decision Theory as the foundation, PNNs are developed from ANNs, which facilitates the emergence of a probabilistic approach. Supervised training is frequently employed in conjunction with PNNs so that the probability density functions that are utilised in the pattern layer can be developed. The application of PNNs provides the specific benefits of faster training, an innate parallel organisation that facilitates more simple convergence in addition to increased flexibility that permits the deletion and introduction of training samples without the necessity for comprehensive training work (Revett et al., 2005). A PNN's input layer is followed by its pattern layer and finally, the category layer. The system is dependent on the variation between the training input and the input vector when calculating how close one is to the other. Conversely, the PNNs second layer generally combines the different inputs in order to generate the probability of each connection occurring. These probabilities are subsequently screened by using a transfer function that generally operates by distinguishing the highest probability, which is then assigned the value "1" and the remaining probabilities are assigned the value "0", which therefore determines which possibility has a greater likelihood of occurring in comparison to the others.

3.7.1 Classification Theory of PNN

One of the primary aspects of PNNs is classification, which enables PNNs to function and learn more efficiently than different kinds of ANNs. Furthermore, classification facilitates a

centralised scheme that transforms the inputs into outputs as a result of learning acquired from the training sets. It is possible to increase the understanding of classification for a standard PNN by considering it from a mathematical viewpoint. Consider vector x, such that it possesses m dimensions for a PNN's input, it is assumed that it is possible to classify it into one of two categories. These categories can be classified as C1 and C2, meaning that the vector x cannot be classified into any other category. Similarly, it is possible to label the probability density functions for the respective categories as $F_1(x)$ and$F_2(x)$. In this scenario, L1 and L2 represent the cost function or the losses connected to inappropriate classification of the input vectors associated with categories C1 and C2, respectively. In a similar manner, P1 and P2 are the prior likelihoods that x is associated with categories C1or C2, respectively. By applying the Bayes decision rule, it is possible to conclude that x can only be associated with category C1 if:

$$\frac{F_1(x)}{F_2(x)} > \frac{L_1 P_2}{L_2 P_1} \tag{3.18}$$

Conversely, x could only belong to the category C_2 if:

$$\frac{F_2(x)}{F_1(x)} > \frac{L_1 P_2}{L_2 P_1} \tag{3.19}$$

It is also important to note that in a variety of different PNN scenarios, the loss function and prior probability are either the same or almost the same. By applying this condition to the above mentioned classification processes, it is evident that classification is for the most part purely dependent on probability density functions. Therefore, it is possible to conclude that probability density functions sourced from the training patterns can be successfully implemented to classify the input vectors for a particular PNN (Goh, 2002). The Parzen window approach is utilised in PNNs with the aim of calculating the class reliant probability functions. It should be noted that these estimations have a non-parametric nature. Such class-reliant probability functions that are generated in this way are subsequently applied in the classification of present categories based on the Bayes decision rule. This approach enables the verification of whether it is possible to categorise an input vector pattern into one of the determined categories. This data is subsequently added to the relevant frequency of every category in order that the PNN can choose the category that best corresponds to the given input vector pattern. It is important to consider that both the Bayes decision rule and the Parzen window techniques have been utilised for many years and are well regarded, which means that there is high reliability. In mathematical terms, it is possible to express

the Parzen window estimate for the probability distribution function for category C1 as the following (Goh, 2002):

$$f_1(x) = \frac{1}{(2\pi)^{\frac{m}{2}}\sigma^{mn}} \sum_{j=1}^{n} e\left[-\frac{(x-x_j)^T(x-x_j)}{2\sigma^2}\right] \tag{3.20}$$

n represents the different data training sets utilised m represents the dimension of the input vector x j represents the pattern being used w represents the smoothing parameter being utilised σ= smoothing parameter

3.7.2 The Architecture of Probabilistic Neural Networks

Fundamentally, a PNN is a feed-forward kind of ANN. The most simple application of PNN is founded on non-parametric probability density functions in addition to the Bayes classification rules. The model approach to training in PNNs involves a single pass through each of the given training data sets. As this is more rapid and efficient, it generally means that the training process in a PNN is quicker in comparison to alternative neural networks; however, this generally necessitates increased memory in order to allow for the increased number of training patterns and their associated data sets. The development of more cost effective and efficient computer memory in recent years has resulted in reduced focus on increasing the data storage capacity when implementing modern PNNs (Tripathy et al., 2010). PNNs are feed-forward neural networks that are comprised of four layers with a design that enables them to determine the most appropriate classifier. Gaussian activation functions are frequently used, meaning that the classified pattern is either located in region with an output of "1" or a different region that is classified as "0". PNN output functions are only limited to these two possibilities. Furthermore, PNNs have no connection in terms of form or behaviour to any normal methods of distribution. A typical example of a PNN is displayed in Figure 3.6 below:

PNNs function in a non-linear manner using non-parametric techniques to form a pattern-recognition algorithm. A PNN normally defines a probability density function for the different classes of data by utilising information collected from the training sets in addition to the kernel width. The outputs of a given PNN can be regarded as Bayesian probabilities, which determine that the inputs are members of the output class based on a certain probability. Thus, all PNNs are purely based on Bayes classification method and can be expressed in mathematical terms as the following:

$$h_i c_i(x) > h_j c_j f_j(x) \tag{3.21}$$

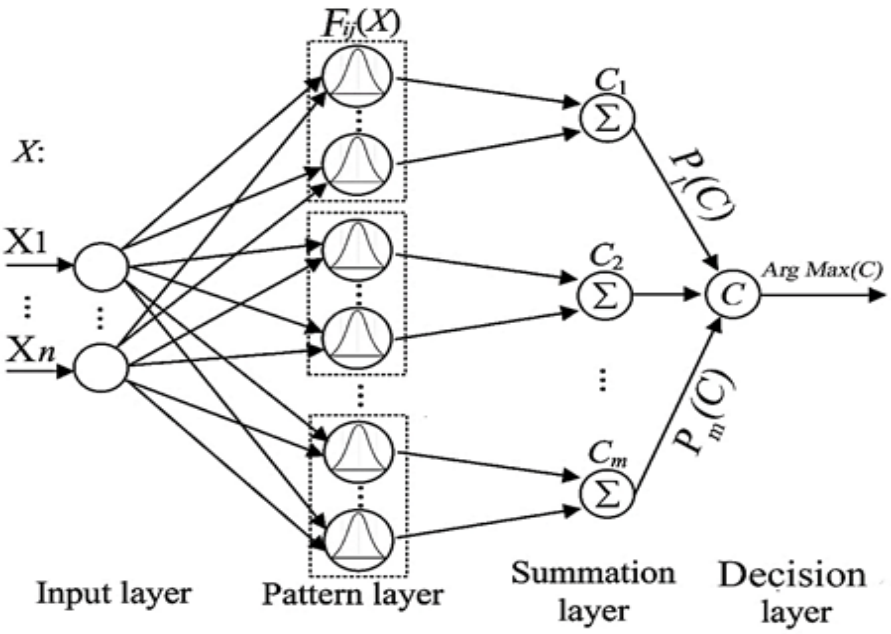

Fig. 3.6 Standard PNN structure

Where: h_i and h_j represent the preceding probabilities c_i and c_j represent the costs of improper classification f_i and f_j represent the probability density functions

In a given PNN, the preceding probabilities cannot be determined, as it is difficult to ascertain whether the input sample is sourced from the data in the training set that is utilised to develop the PNN in question. The most effective method of addressing this problem is to employ the training set data to calculate such probabilities. This kind of assessment is founded on Parzen's method of probability density function estimation (Shaffer and Rose-Pehrsson, 1999). A probabilistic neural network is usually comprised of layers, including the input layer, the pattern layer, the summation layer and the output layer, and is appropriate for problems of classification see Figure 3.6. The amount of input layer units corresponds to an identical number dimension of input pattern vectors, the amount of pattern layer units is identical to the amount of training samples, and the amount of output layer units is identical to the amount of classes that exist in the training samples.

The one-pass training procedure incorporates the memorisation of all patterns that exist in the training set, in which every Gaining pattern is allocated to the matching node. For instance, the weight, W_{ij}, between the i^{th} input node and the j^{th} second layer pattern node is allocated by establishing that $W_{ij} = X_{ij}(i = l, 2, ..., p)$ and p is the dimension of the input pattern, X_{ij} is the ith input in the jth training pattern). As part of the examination procedure, every input pattern vector is distributed to each pattern unit and a dot product is generated

at every pattern unit and then conducted via a nonlinear function (typically a multivariate Gaussian function). The summation units calculate the total of the outputs of the associated pattern units that are in identical classes. It is possible to interpret the output value as the probability distributions of different classes. Hence, the output layer has the capability to determine to which class a pattern belongs by utilising a conventional decision-making technique, such as the Bayes decision strategy. It is possible to modify two different types of weights of the PNN in order to ameliorate its effectiveness, namely the coefficient C_k, which is related with class k, and the smoothing parameter, a, in the activation functions, which is a multivariate Gaussian function that is applied to calculate the class probability density function.

3.7.2.1 Input layer

The input layer represents the initial layer that consists of m input variables that comprise the input vector x. Every neuron that exists in the input layer corresponds to a predictor variable. Neurons that exist in the initial layer serve to spread the different inputs of vector x throughout the next layer. There are equivalent numbers of neurons and parameters (variables) within the system. There is a complete connection between the neurons in the first layer and the neurons in the second layer, as illustrated in the red rectangle in Figure 3.7.

In regard to categorical variables, N-1 neurons are utilised in the scenario where the number of categories is equal to N. The range of values is standardised through the process of subtracting the median and then dividing by the interquartile range. Subsequently, the input neurons supply the values to each of the neurons located in the hidden layer. In the hidden layer, the Input vector $X = (X_a, X_b, \ldots X_n)$ where $a, b, \ldots n$ represent the data points that are fed from the FCM vectors of the MIB features and Weights of Edges = Input Vector.

3.7.2.2 Hidden layer

The next layer, defined as the pattern layer, has an equivalent number of neurons to the input layer. The patterns in this layer can be best represented by stating that a neuron exists for each input in the input layer. Neurons that exist in the layer are allocated different weights based on the various training sets that are used.

The amount of units present in this layer equate to the amount of samples present in the training set. The radial units are direct copies from the training set and there is only one for each case. Each of them models a Gaussian function that is centred on a training sample. An output unit exists for every class, which has an association to each of the hidden units that

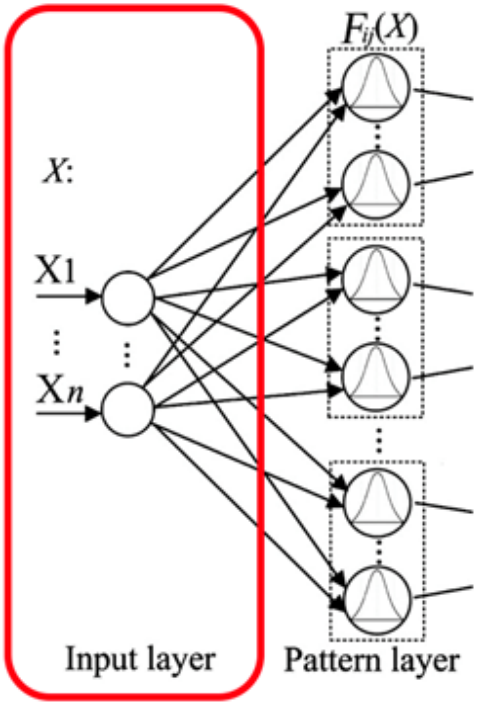

Fig. 3.7 Input aspect of the PNN

belong to this class, but has no associated with any other units, as illustrated in Figure 3.8. In this layer, the activation function listed below is used:

$$f(x) = e^{-\frac{(w_i - x_i)^t (w_i - x_i)}{2\sigma^2}} \tag{3.22}$$

Where w_i represent the weights, xi represent the model's variables and σ is a smoothing parameter, which is the only network parameter that must be set at the start of the training process. Every training sample corresponds to a specific pattern unit. The pattern unit output is calculated by determining the Gaussian Distance between every sample and the corresponding unit, as follows:

$$F(x) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{j} \sum_{i=1}^{j} e^{-\frac{(x - x_{ij})^t (x - x_{ij})}{2\sigma^2}} \tag{3.23}$$

Where: X= unknown (input) X_{ij} = ith training sample from category σ = smoothing parameter j = number of training samples d= length of vector C= Class

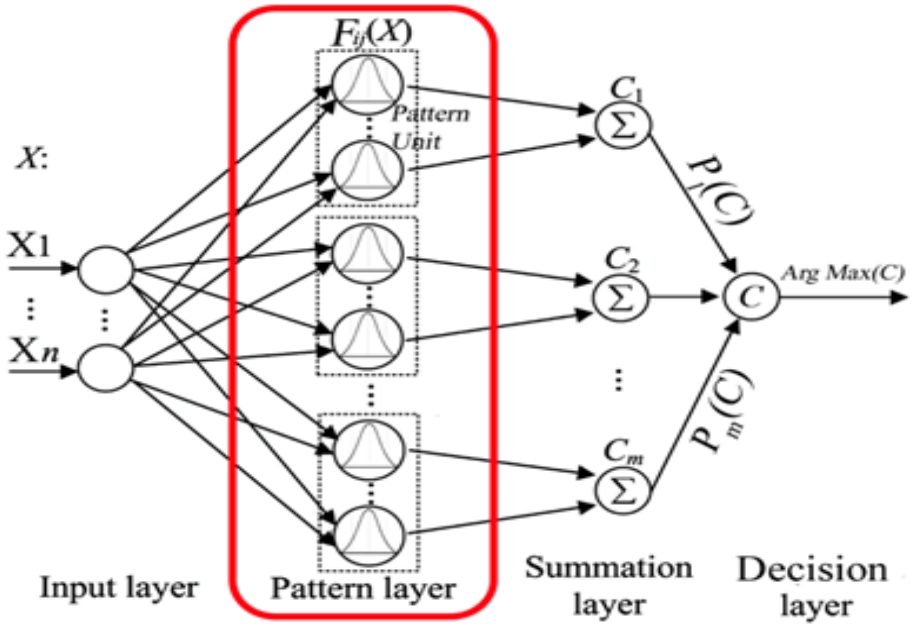

Fig. 3.8 PNN Pattern layer

3.7.2.3 Summation layer

At the summation layer, the amount of neurons is equivalent to the amount of data classes. Every neuron has a connection to each of the neurons within the pattern layer that are members of the class that that this unit represents. The process of summing the Parzen window estimate's exponential terms is completed by the summation layer, where the amount of neurons is lower in comparison to the prior layer, as illustrated in Figure 3.9 . Rather, the amount of neurons in the summation layer is in fact based on the number of categories, as every neuron generally represents each category.

It should be noted that the connections in the summation layer are basically restricted to 1, in order that the pattern layer outputs are only summed together. The resulting values are subsequently transferred to the last output layer, in which there is only one neuron component that implements the classification. Each neuron's output in this layer is an indication of the likelihood that the input vector X will belong to class C_i, which can be represented as:

$$P(x|C_i) = P_i = \sum_{j=1}^{|C_i|} f(x) \tag{3.24}$$

Where, C_i represents the amount of training patterns from class C.

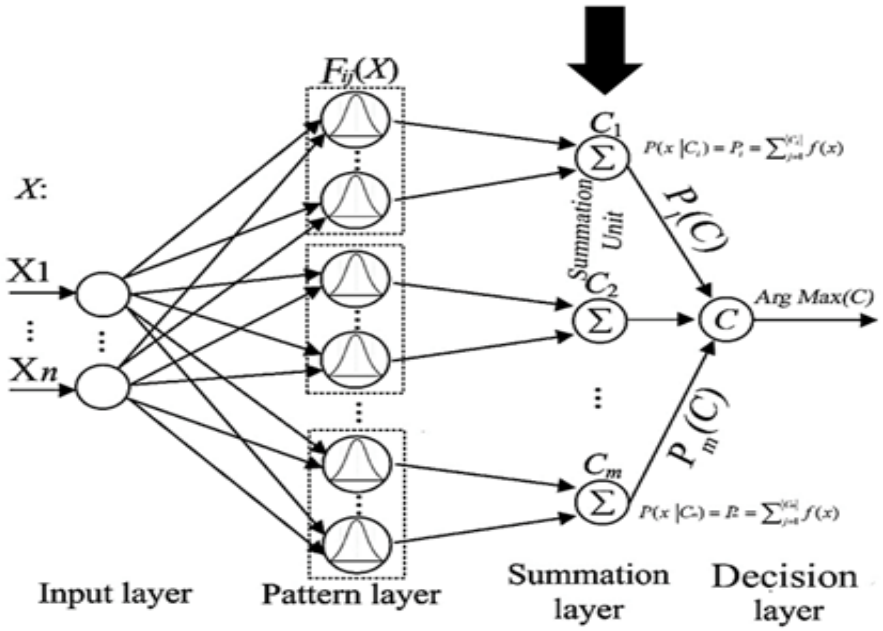

Fig. 3.9 Summation unit of the PNN

3.7.2.4 Decision layer

In general, the neuron present in the output layer of the PNN generates a binary value that is a result of the probability density function that has the highest value, as illustrated in Figure 3.10. Essentially, the greatest value acquired by using this mechanism is an indication of the optimal classification that the PNN can give for the given input pattern (Bolat and Yildirim, 2003).

The output value is regarded as the probability distributions for the different classes. Thus, the output layer is capable of determining which class the input pattern belongs to through the application of a conventional decision-making approach, such as the Bayes decision strategy. There is one neuron in the output layer, which facilitates the classification decision related to the input vector X based on the Bayes decision rule. The output can be calculated through the following equation:

$$C(x) = argmax_{i=1\rightarrow m} \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{j} \sum_{i=1}^{j} exp^{-\frac{(x-x_{ij})^t (x-x_{ij})}{2\sigma^2}} \tag{3.25}$$

m represents the number of classes presented in the system(Specht, 1990).

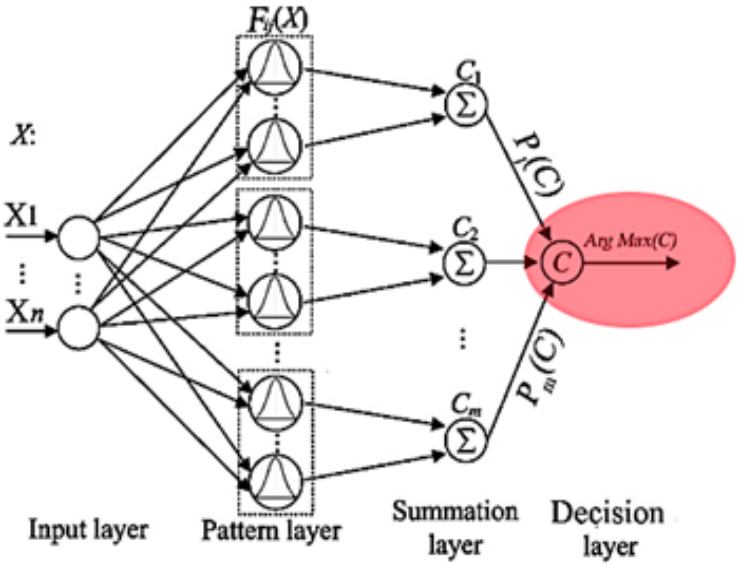

Fig. 3.10 Decision unit of the PNN

3.7.3 Radial Basis Function

Once the training is over, the probabilistic neural network parameter adaption process is complete. Subsequently, the testing samples are classified. In this process, a test sample is assigned to the class in which the output nodes in the PNN have the highest output based on the Bayes rule, which assigns a test sample x to class A when:

or if only (class A and B involved)

$$f_A(x) \geqslant L(h_A/h_s) \tag{3.26}$$

where h_A represents the a priori likelihood of the incidence of the test samples from class A, lA represents the loss factor linked to incorrect categorisation, $L = l_B/l_A$ represents the loss ratio and $f_A(x)$ is the probability density function associated with class A. The coefficient $C_k, (k = l, 2...n)$ shown in 3.9 is calculated by $C_k = h_k l_k/n_k$. In the scenario where no information is available concerning the a priori likelihood of the incidence of each one of the classes, h_k can be ascertained based on the incidence of the classes in the training samples; for example, $h_k = n_k/n$, where n_k represents the amount of training samples in class k and n is the overall amount of training samples. I_k can be selected based on the user's own experiences or specific approach. The PNN classified determined by Specht (1990) utilises an equation that is similar to the Parzen probability density function estimator, which has a strong association with the interpolation technique of the radial basis function:

$$F_k(x) = \frac{1}{(2\pi)^{p/2}\sigma^p}\frac{1}{n_k}\sum_{i=1}^{n_k} exp\left[-\frac{(x-x_{ki})^T(x-x_{ki})^2}{2\sigma}\right] \tag{3.27}$$

where i = pattern number, U_k = number of training samples in class k, X_{ki} = ith training pattern from class k, σ = smoothing parameter, p = the dimension of input vector space. Every training sample in class k- generates a Gaussian distribution centred at itself, and $f_k(x)$ represents the total of these small Gaussian distributions.

3.7.4 The smoothing parameter, σ

The smoothing parameter,σ, is utilised in the process of modifying the significance of each pattern unit found in the second layer. If the value of σ is small, each of the training patterns will exhibit a thin Gaussian distribution surrounding itself. There are numerous different modes involved in the class probability density function. An input pattern defined as x that has a strong similarity to a particular training pattern, for example X_{ki}, will have a high value of $exp[(x-x_{ki})t(x-Xki)/2\sigma^2]$. Therefore, the value of $f_k(x)$ is predominantly determined by X_{ki} instead of alternative class k patterns that are situated at a greater distance from x, which can be detrimental to the generalisability. If the value of σ is large, numerous wider Gaussian distributions are inclined to intersect each other, and $f_k(x)$ must be determined by each of the patterns in class k. In such a scenario, there is a higher level of interpolation among the training patterns in the PNN, and it exhibits tolerances to incorrect training samples, as the class pdf is largely determined by most of the class patterns. Nevertheless, if the value of σ is excessively large, this will significantly reduce the importance of the roles of individual training patterns, which means that it will not be possible to generate a model that will effectively solve the problem. The selection of σ will determine how the complex the decision surface is and therefore makes a significant contribution to the performance level of the PNN. (Specht, 1990) conducted an experiment that revealed the outcomes of using various different values of σ where electrocardiograms were categorised as either abnormal or normal. Superior diagnostic precision can be obtained with a specific wide range of σ that is relatively easy determine. In Figure 3.11, the overall association between the smoothing parameter, σ, and the incorrect classification flaw in the training and test samples is demonstrated.

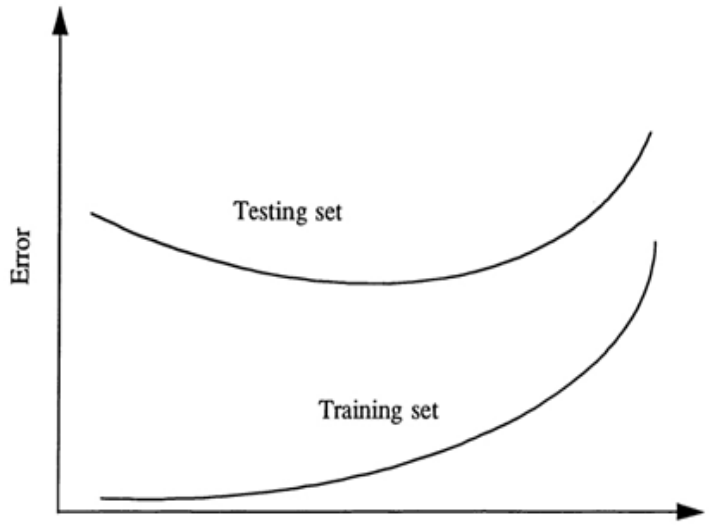

Fig. 3.11 Smoothing Parameter

3.8 Frame work design

The conceptual framework in this study is founded on various machine techniques and algorithms, beginning with the cleansing and then loading of data into the system, culminating in the final step of generating output classes. The principle conceptual framework of this research can be clearly seen in Figure 3.12, which commences with the process of data purification.

As there are an excessive amount of problems related to the data, such as variability, repeated values, data deficiencies and noise, a data pre-processing algorithm has been employed in the preparation of the data. The pre-processing of data is considerably important to ensure high precision and performance. Hence, two key methods are utilised for normalisation and standard scaler in order to enhance the model precision. In the following stage, processing of data occurs and various equations are utilised to generate data that is suitable for clustering and classification in a manner that will ensure superior outcomes. Further explanations of these specific procedures are given in Chapter 4. Subsequently, the purified data is transferred into the unsupervised clustering system. In this stage, one of the most widely used clustering methods is Fuzzy c-means, as its performance exceeds the process of clustering traffic fault data. Nevertheless, this algorithm has certain drawbacks in that the process of ascertaining the number of clusters and clusters at the start can be relatively lengthy. Furthermore, an excessive amount of iterations are required in the optimisation of the initial and optimal cluster numbers. As the performance of FCM is reduced in the process of attempting to resolve these

problems, an additional algorithm defined as subtractive clustering (SC) is introduced into the FCM in order to resolve the aforementioned problem. The primary objective of utilising FCM is its contribution to both grouping and segmentation; however, in the scenario where there are different datasets that have multiple faults simultaneously, this can lead to network traffic as a result of its magnitude. In order to prevent this issue, unsupervised machine learning is applied to cluster the different datasets that originate from diverse segments in the network traffic. Following this, each of the clusters that occurs is optimised and the output feature vectors are moved to the subsequent stages so that all faults that exist in the system can be classified and labelled appropriately. FCM is comprised of various mathematical equations that are applied that ultimately function to minimise the objective functions, which can be achieved through the optimisation of the degree of membership and cluster centres at each iteration. Furthermore, the FCM technique is beneficial for managing vague cluster boundaries. It is essential that the problem of high dimensionality is resolved in such a scenario. FCM is an algorithm that is iteratively optimised and is believed to have greater flexibility, particularly for the analysis of traffic.

Additionally, superior outcomes can be achieved by using FCM for overlapped datasets in comparison the K-means algorithm. As opposed to k-means, in which each data point is required to only belong to one cluster centre, in FCM, the datapoint is allocated to cluster centre, which means that it could belong to multiple cluster centres.

In this stage, various mathematical equations are applied to process the data, as a result of which four separate clusters are created based on the attributes of the extant datasets. Subsequently, the final outputs that have been generated are stored in various feature vectors proceed to the classifier.

Following this, for every specific vector in the input datasets, a matching target vector is generated in another matrix in order to create the same magnitude as the data vector and it is comprised of values 1,2,3 and 4 that match with the primary erroneous components in the data vector. These correspond to the light and heavy cases in the router and server that are utilised in the PNN training process.

Each of the data sets has an identical amount of samples taken from standard and faulty scenarios. The data set and target vector are both separated into equally sized subsets by taking all the other values of the vectors, where one is intended for PNN training and the other for testing the network after training is complete. This means that a total of two data sets are generated. The first subset from the time dimension is separated into two subsets that are equal. As the sample data utilised for training the PNNs in this case is allocated at a ratio of 70% for training and 30% for testing,. Subsequently, data simulation implemented and by

using a trial and error process, the spread parameter of the PNN is calculated as 0.06. Lastly, the classification outcomes are observed by plotting the figures that have different features. The overall suggested flow chart for this research project is displayed in Figure 3.13, which reveals all the steps in the process as well as the principle calculations involved in each stage.

The execution processes listed above are explained in greater detail in Chapter 4 and 5 along with the result discussions for every step that are based on the dataset utilised in this research project.

Eventually the new developed algorithm is proposed to classify the faults in the network system as depicted in the below:

Algorithm 1 SFPNNC Classfier

1: **procedure** PREPROCESSING(DATA)
2: $m = degreeoffuzziness$
3: $X = Load(data)$
4: $V, Xinit = Estimate(X)$
5: $Cluster, Center, member_s hip = Euclidean(V, X)$
6: $U = Update, membership(ClusterCenter, membership)$
7: $V = Update, cluster, center(U, membership)$
8: **if** optimal(V) then **then**
9: $Dset = Jm(V, X, U)$
10: $Classifier = PPN(Dset)$
11: $Evaluate(Classifier)$
12: **else**
13: $goto : 4$
14: **end if**
15: **end procedure**

3.9 Chapter Summary

The methodology and design architecture that are proposed to be used for this project are presented in this chapter, including all parameters for the techniques, such as the supervised and unsupervised algorithms and the other methods. The suggested model is designed based on the benefits of each of the models utilised. In order to enhance the outcomes of the process, it begins by refining the noisy data by pre-processing the data, thus purifying an effective sample that that can be readied for the system.

The suggested system of classification as depicted in the framework design shown in Figure 3.12 commences with the filtered data being loaded into the Fuzzy Cluster Means,

in order to group all clusters that occur in each sample. To prevent excessive time being taken in the process of cluster initialisation and identification of the optimal cluster centre based on the nature of the existing datasets, a new algorithm is introduced to the FCM called Subtractive Clustering. The primary objective of utilising FCM is its importance in the process of grouping and segmentation, while in the scenario where the sample datasets have different faults simultaneously, network traffic can be generated as a result of its magnitude. Furthermore, the ideal clustering technique is Fuzzy Cluster Means because of its strong attributes that allow it to resolve problems involving network fault diagnosis, including dealing with vague cluster boundaries, high dimensionality issues as well as its level of flexibility, particularly in the analysis of traffic, as it is an iterative optimal algorithm (Qader and Adda, 2014). Similarly, PNN exhibits superior attributes, including the increased accuracy of the non-linear algorithm. The analogous weight values of the PNN represent the sample's distribution in the model, while it is not necessary to train the network, which means that it is suitable for real-time processing. Additionally, its robust attributes such as sample noise, rapidity of the training data and the classification precision rate enable outcomes with increased quality to be achieved. Based on the beneficial aspects of FCM and PNN, a hybrid method is developed and proposed called the Subtractive Fuzzy Probabilistic Neural Network Classifier (SFPNNC), which is explained in greater detail in the main proposed flowchart of SFPNNC, see 3.13.

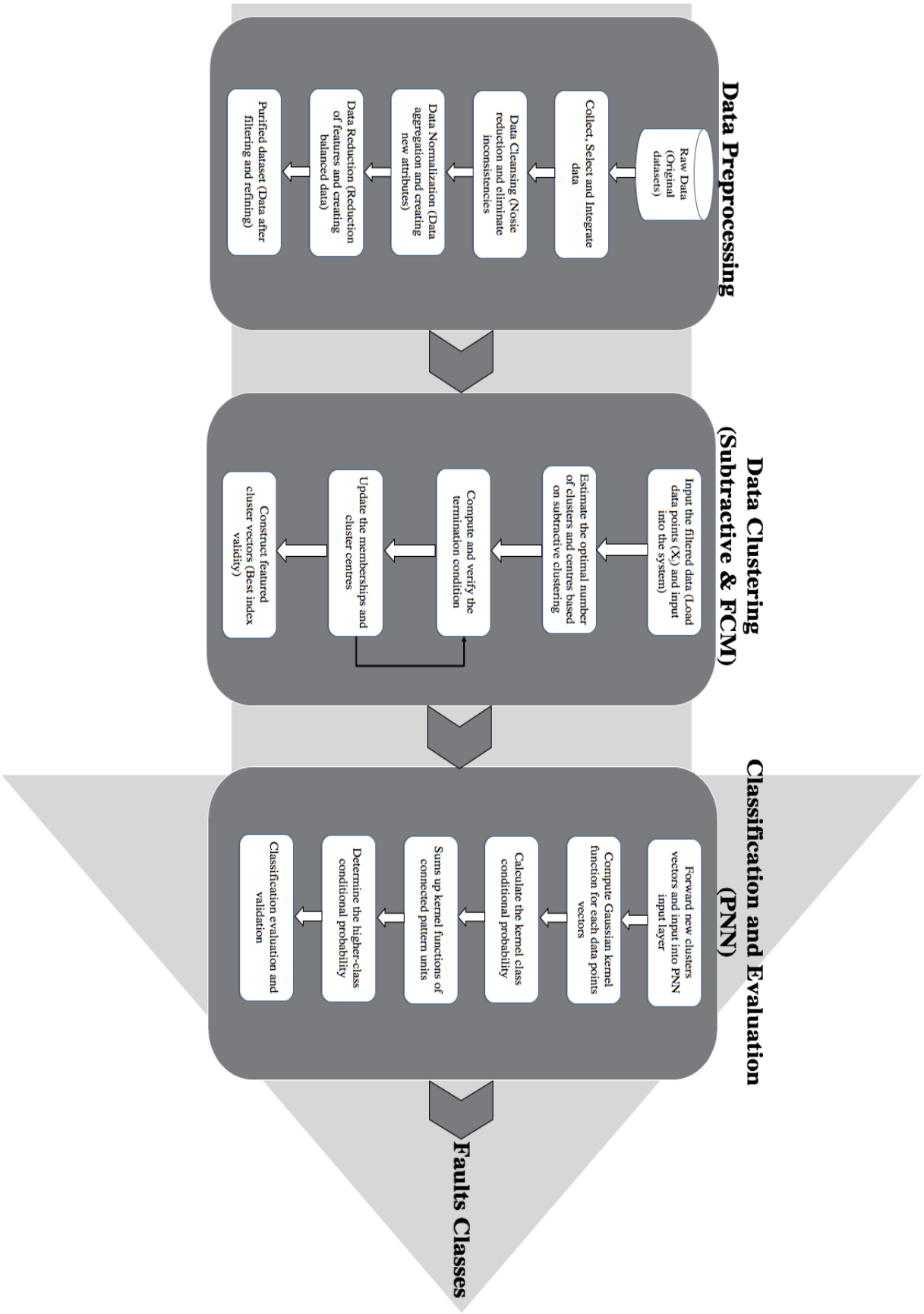

Fig. 3.12 The framework design of the proposed method

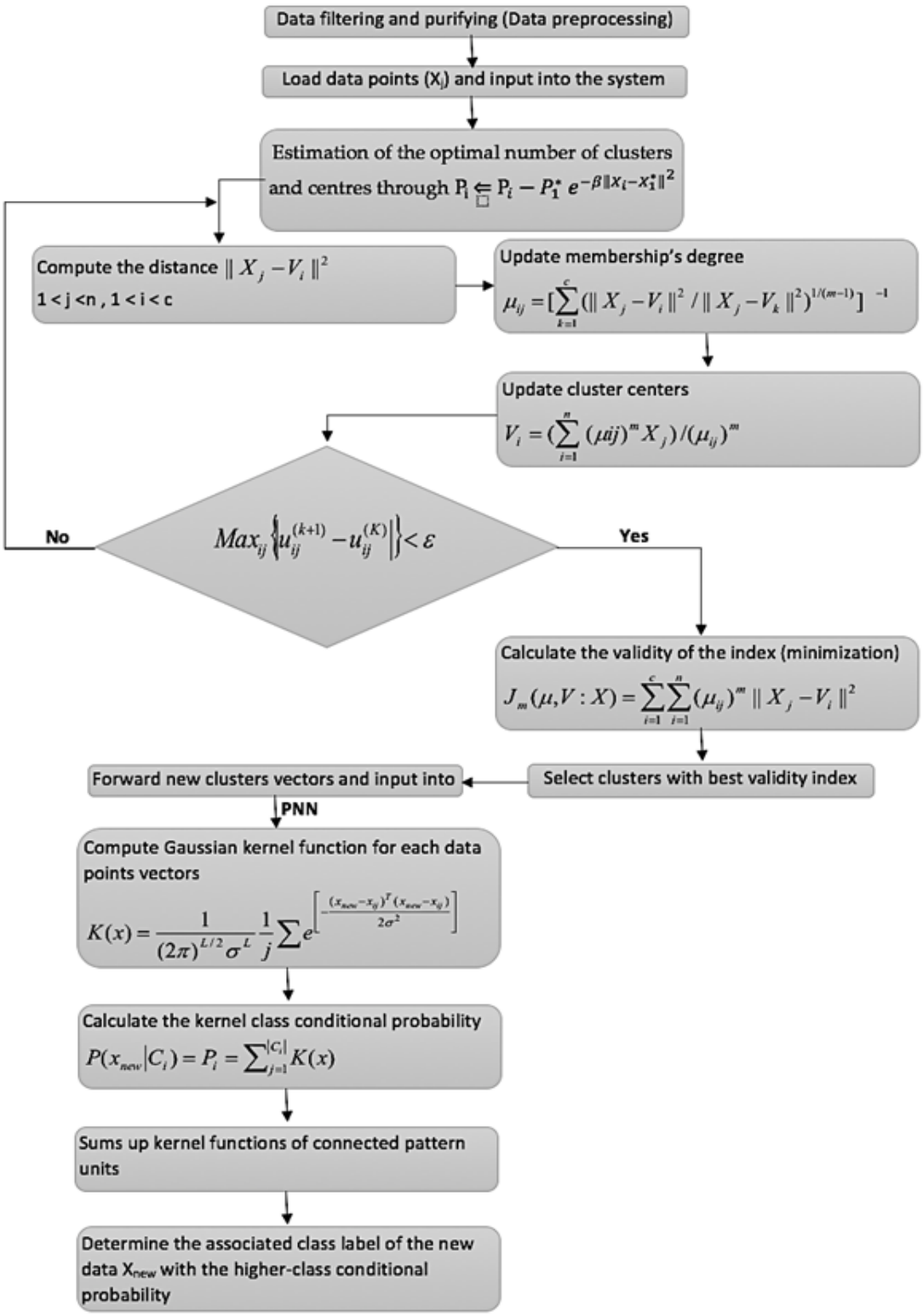

Fig. 3.13 The main proposed flowchart of SFPNNC

Chapter 4

Data visualisation and preprocessing

This chapter will present the datasets that have been utilised during this project along with an explanation of all the data as well as analysis of its specific features. Furthermore, as a result of considerable noise and inconsistencies in the data, the principal stages of the data pre-processing procedure will be demonstrated. In order to comprehend the data, each of the datasets will be visualised individually. The initial stage in the preparation of data for the purpose of machine -learning is data pre-processing, the objective of which is to purify the data so that it can be classified. Additionally, this chapter will examine the important role that data pre-processing plays as well the manner in which the form of the data will be transformed from the public and defective shape into pure data after the eradication of its flaws. Hence, it is imperative that the datasets have good quality in order to ensure that effective results can be obtained from the analysis in the research. It is often necessary to make modifications to the format of the datasets to enable them to be fitted for classification purposes. For instance, the datasets in this research incorporate both small and large values, which means that a normalisation process must be implement to generate consistent and beneficial format data.

In the subsequent parts of this chapter, the different types of data characteristics will be presented, including noise as well as irrational and deviation feature values. Based on the extant datasets, it will be possible to determine which of the frequently used statistical methods can be applied to refine the data. Ultimately, the main aim of this chapter is to enhance the data quality in order to ensure that all measurements are precise, accurate and exact, while also providing as complete data as possible

4.1 Network Traffic Fault Datasets Characteristics and Types

Generally, errors can materialise naturally and are not created. One of the difficulties presented in this project involves the process of introducing errors into the Testbed. As suggested by Carr (1990), Leinwand and Conroy (1993), Durvy et al. (2003) and Comer (2006), errors can be grouped into five different network problems, namely:

- cable issues
- connectivity issues
- collisions within the network
- software issues
- replicated IP addresses

These errors can materialise for a variety of reasons, including transmission link collapse, failure of a network component, software malfunction, protocol malfunction, hostile network activity, unwarranted network growth, disconnection of the interface and misconfiguration. In this research, four frequently occurring faults including (cable issues, connectivity issues, collisions within the network and replicated IP addresses) have been selected in order to assess the outputs of the IFAgents and IPAgents. All errors have been stimulated under conditions of both light and heavy traffic.

The datasets for this experiment were extracted from a study performed by Al-Kasassbeh and Adda (2009) on the diagnosis of errors. In their study, the researchers utilised diverse instruments in varied situations as well two distinct sources, namely a router and server. They predominantly utilised MIB variables and from the 12 distinct groups of these variables, they chose the most appropriate characteristics of the network errors from the IP and IF groups according to the work environment. Nevertheless, in the final stages of the process, they only concentrated on 6 for data gathering from the server and 12 features were captured from the router, while all features were found in the IF group. The datatypes all consist of numerical values, ranging from zero to billions, which means that there is significant inconsistency in the data. Nonetheless, the datasets were utilised by Al-Kasassbeh (2011b), which resulted in significant noise and incorrect classification.

A simple network management protocol (SNMP) is utilised to gather and arrange information regarding the instruments pertaining to an Internet protocol (IP), including routers, switches, hubs, printers and servers. An SNMP employs a management information base

Table 4.1 Parameters of network involved in router anomalies

Network Parameter	Parameter Name
P1	IfInNUcastPkts22
P2	IfInNUcastPkts33
P3	IfInOctets22
P4	ifInUcastPkts22
P5	ifInNUcastPkts22
P6	ifInUcastPkts33
P7	ifOutNUcastPkts22
P8	ifOutNUcastPkts33
P9	ifOutOctets22
P10	ifOutOctets33
P11	ifOutUcastPkts22
P12	ifOutUcastPkts33

(MIB) to acquire this information, which incorporates numerous octets or packets that are distributed, delivered, received and discarded. In total, there are 178 MIB variables, from which 12 variables belonging to the interface (IF) group were included in our experimental research to represent router traffic, while an additional 6 variables from this category (IF) represented server traffic. These variables demonstrate higher levels of sensitivity in regard to Internet traffic behaviour. The datasets examined in Al-Kasassbeh (2011b) incorporate 12 parameters that will facilitate the diagnosis of network faults in the router and 6 parameters for the server, as illustrated in Tables 4.1 and Table 4.2.

Network errors can be grouped under different categories of problems, including IP conflicts, connectivity issues, cable issues, system malfunctions and software issues. There are a number of factors that could cause such malfunctions. For example, transmission connection malfunction, failure of a network component, software malfunction, hostile activity, unwarranted traffic growth, disconnection of the interface and misconfiguration. The MIB variables exhibited in Tables 4.1 and 4.2 belong to the interface category. Additionally, further details on the datasets and given in Table 4.3.

In this research, four frequently occurring errors have been chosen. The datasets were collected under two different conditions, namely lightweight and heavyweight (in terms of the volume of traffic) in both the router and server Al-Kasassbeh (2011b). Explanations of these common traffic faults are provided in the following sections.

Table 4.2 Parameters of network involved in server anomalies

Network Parameter	Parameter Name
P1	ifInNUcastPkts22
P2	ifInOctets22
P3	ifInUcastPkts22
P4	ifOutNUcastPkts22
P5	ifOutOctets22
P6	ifOutUcastPkts22

Table 4.3 Datasets details

Data Name	Number of Instances	NUmber of Attributes
Heavy scenario in the router	960	12
Light scenario in the router	960	12
Heavy scenario in the server	960	6
Light scenario in the router	960	6

4.1.1 Server crash and link failure

This is a form of network error that can lead to server collapse and link malfunction. MIB variables can provide beneficial insights in regard to sudden change in the event that the client machines begin to overwhelm the server with a disproportionate volume of requests. This barrage of FTP requests will ultimately lead to server crash. The MIB variables gathered within the environment with 5PCs are utilised to simultaneously inundate the server. For the purposes of inundation, the DoSHTTP instrument was utilised, which employs 5 PCs to distribute packets to the server on a continuous basis. This attack was initially implemented for 7 minutes, followed by a second attack lasting 9 minutes. This caused the server to crash and cease functioning. Two errors were introduced into the experiment, namely server malfunction and link failure.

4.1.2 Broadcast Storm

This is different for of identified error. This error materialises in network conditions in which one of the PCs sends messages to other PCs requesting services and information. The PCs that receive these messages then begin to reply, which leads to unwarranted network load. This error in the network reduces performance and can also spread to different network areas.

4.1.3 Babbling node

This is also an example of a fault in the network. It is likely the outcome of an issue generated by a network protocol or potentially the PC's Ethernet card. This fault is observed in the event that a PC continuously distributes packets to the network. It is important to note that the experimental design has the capacity to capture errors with significant precision in the light conditions in comparison to the heavy traffic conditions.

4.2 Data Visualisation

For the majority of machine learning, effective learning algorithms are created in order to improve the comprehension of the datasets and to determine a method in which the data can be visualized. In order to understand the essence of each dataset and to investigate the relationships between the features, this section will involve visualization of all the datasets as well as analysis to determine any redundant, irregular or unintegrated data. In this part of the research, a sample of light traffic taken from the router dataset will be shown with a comparison of each of the features to ascertain the significance of their role in the classification. The remainder of the samples will be presented in the appendix and a declaration is emphasised in total for each. As stated by the researchers Al-Kasassbeh (2011b), for each of the distinct situations and instrument sources, normal traffic and three separate errors are captured. Figures 4.1 , 4.2, 4.3 and 4.4 illustrate the visualisation of the four distinct types of traffic in the light conditions that have been gathered from the router, incorporating the normal traffic data in comparison to the traffic with three kinds of fault (server malfunction and link failure, broadcast storm and babbling node), respectively.

In the comparison of Figure 4.1, which depicts normal light traffic in the router, with Figure 4.2, which illustrates the traffic in the light conditions with server malfunction or any disruption in the link to the router, it can be seen in the event that this error materializes, it will have an impact on all extant features. The first feature (iflnNUcastPkts22) is the only one that that is not completely affected by the error, although a small modification still occurs in the shape of the curve, which cannot be utilized as one of the primary features on which the classifier relies. Apart from this, the remaining 11 features are noteworthy and there are significant alterations in their behaviour when the hostile attack or error materialises.

In relation to the second error, namely the broadcast storm depicted in Figure 4.3, it can be observed that abnormalities begin to occur at the time of the traffic fault as well as that all features apart from (iflnNUcastPkts22) and (ifoutNucastPkts22) exhibit considerable

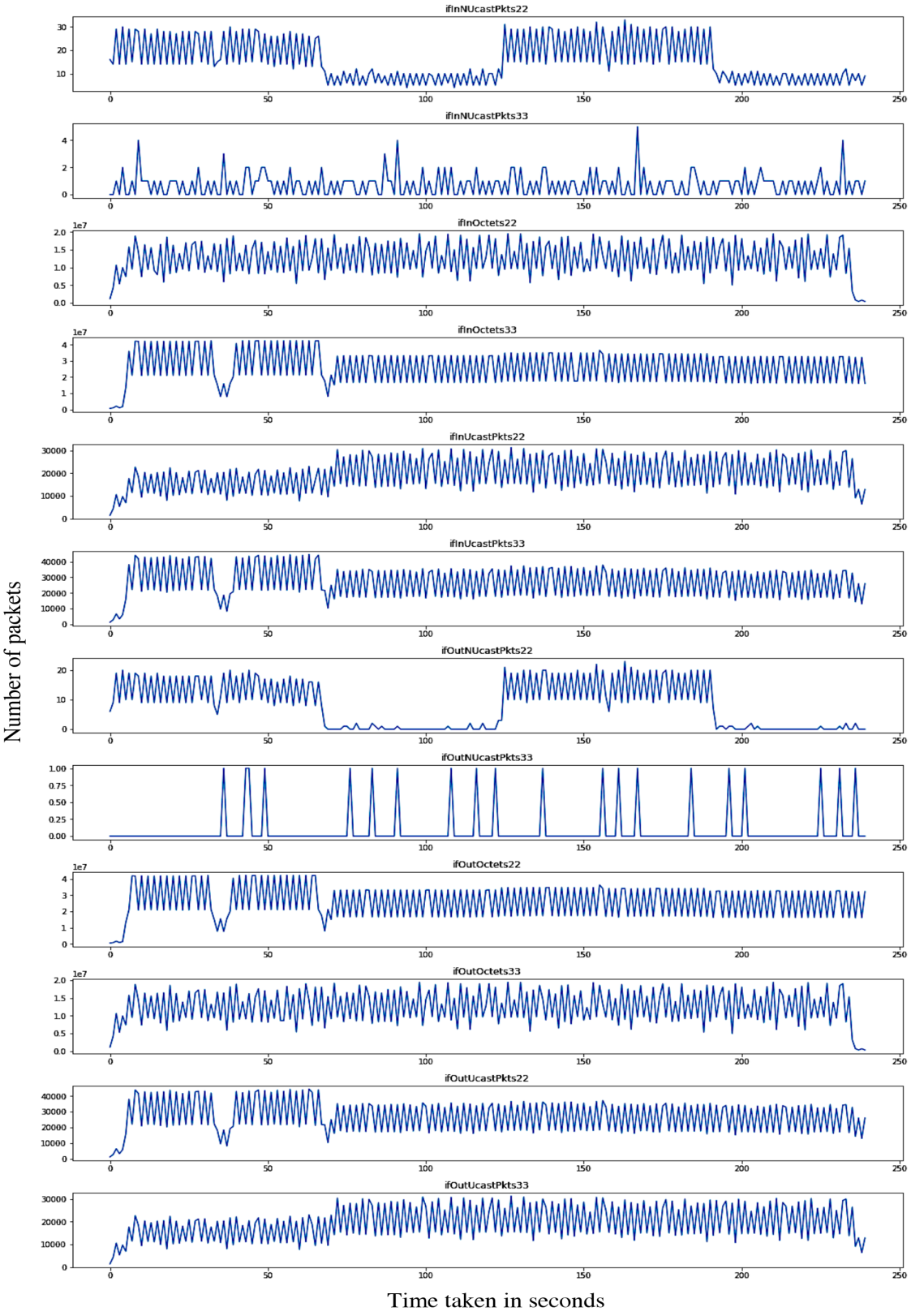

Fig. 4.1 The normal traffic of the router in the light scenario

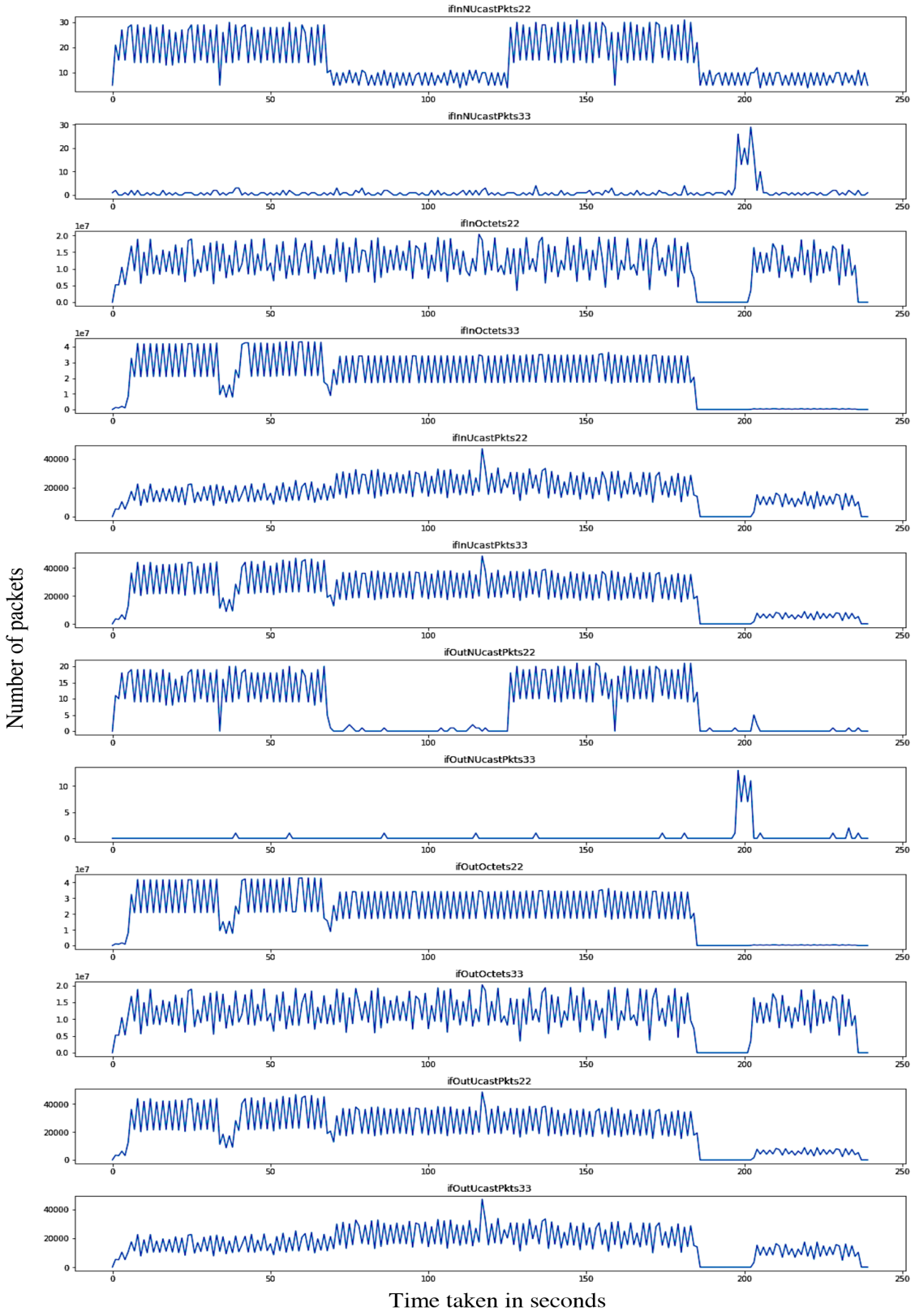

Fig. 4.2 The link failure and server crash traffic of the router in the light scenario

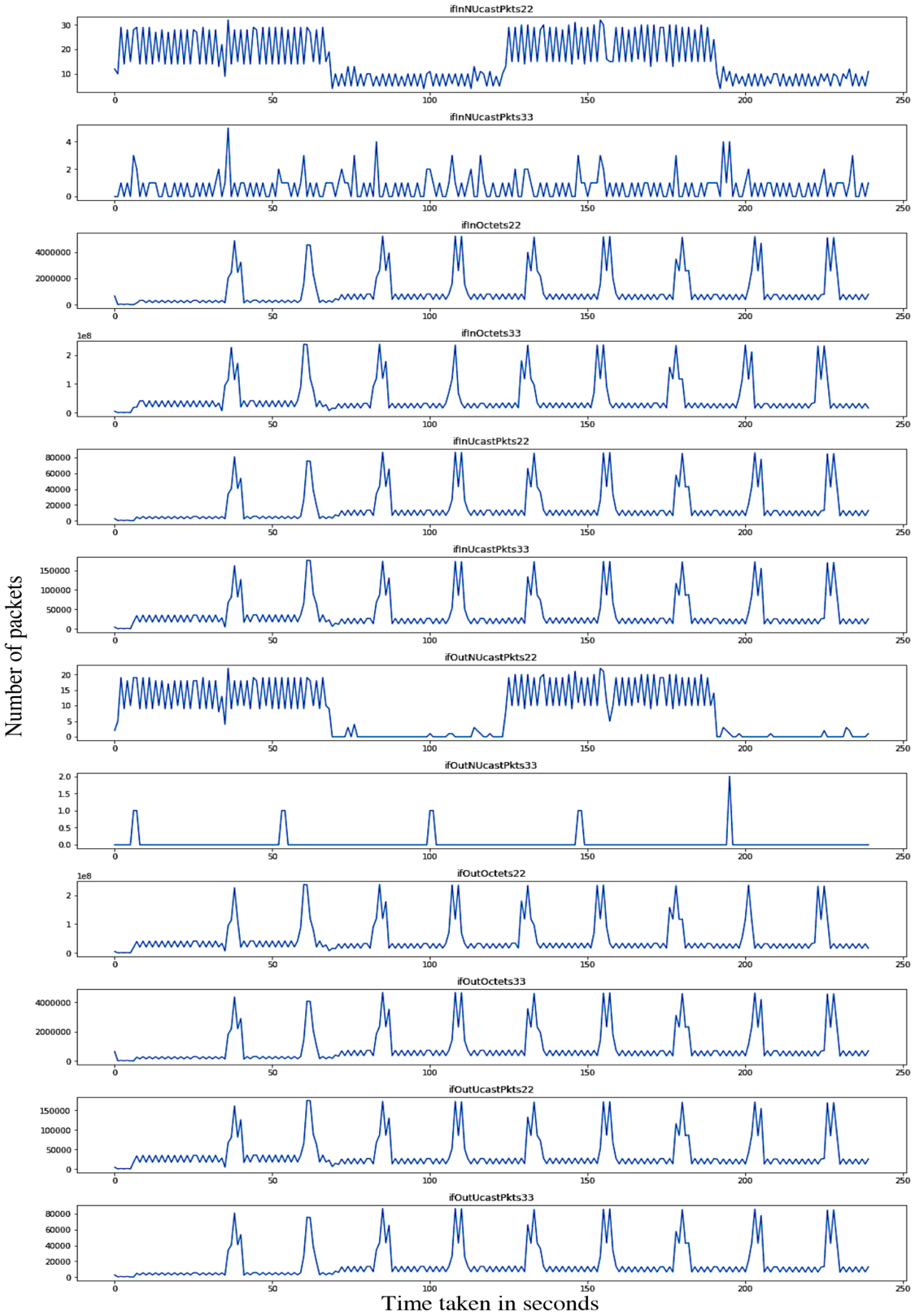

Fig. 4.3 The broadcast storm traffic of the router in the light scenario

sensitivity in the event that a PC begins to distribute messages to all PCs in the network and requests information and services on a continuous basis. However, a comparison of the curves of (iflnNUcastPkts22) and (ifoutNucastPkts22) with the normal traffic indicates a marginal difference, although not obvious, which means that they cannot be utilised by the classifier as the primary distinguishing features.

Furthermore, as depicted in Figure 4.4, in the final error, a clear difference can be observed in terms of all the utilised features when comparing to the scenario where there is no traffic fault. It is therefore apparent that the babbling node attack will affect all of the IF variables apart from (iflnNUcastPkts22) and (ifoutNucastPkts22) – similar to the broadcast storm – as these features are not impacted by the fault in the same manner.

It can ultimately be surmised based on the investigation of the sample's features, including the 12 IF variables, that managing the data will be a complex endeavour as the data is balanced. Each of the features is not able to act independently, while aggregation does not function effectively on this type of data. Rather than creating a sample of aggregation within each individual features, it is possible to employ under sampling. This technique involves utilising the entirety of the smaller class and repeatedly applying a random selection process to choose the same amount from the majority class in order to generate numerous datasets followed by combination of all the classification outcomes. This will enable classification of data that is imbalanced, which is a classification issue that arises where each of the classes does not have equal representation due to the aforementioned reasons. As defined, the issue can be resolved through the process of aggregating all the relevant features in the error to generate a sample in order to create a new dataset.

The process of utilising the entire smaller class and randomly choosing the same amount from the majority class on a repeated basis in order to generate several data sets followed by a combination of all classification outcomes in order to provide balance is the most effective method that can applied without the risk of losing data.

As this project will involve the creation of a balanced dataset and a 50x50 distribution will be performed, it is possible to test via classification of the whole dataset by utilising the optimal model determined after the conventional testing procedure on the 50x50 balanced file. If numerous balanced files are generated using the previously described method, it is then possible to average the outcomes and the select the optimal model.

We next proceed to investigate the outcomes of the visualisation of the sample datasets gathered under light conditions, although the assigned resource is the server in this case and not the router. In this situation, when the hostile attack or error commences, 6 features are detected and all are associated with the IF group, as shown in Table 4.2.

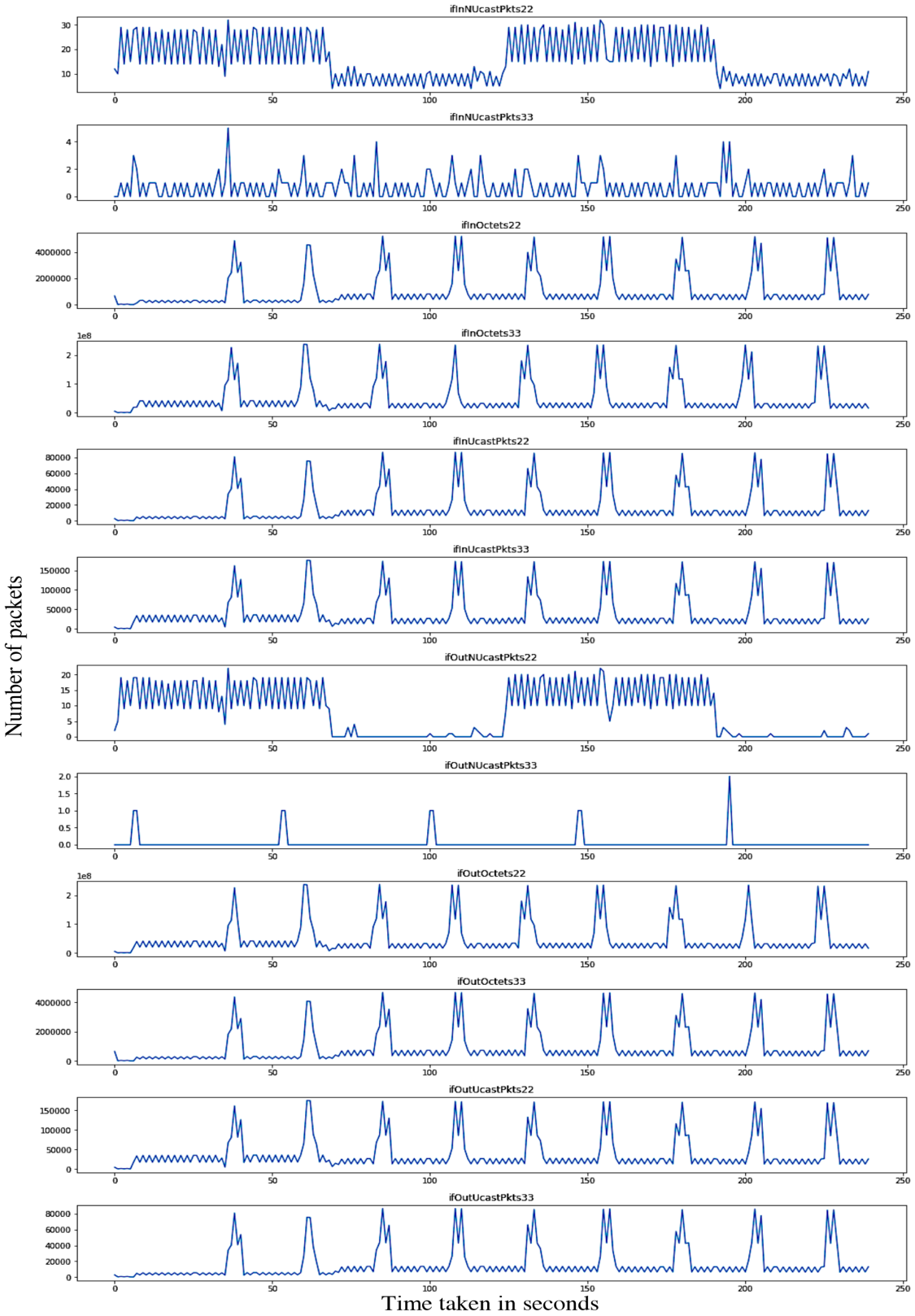

Fig. 4.4 The babbling node traffic of the router in the light scenario

In order to assess each of the features individually where the target of the attack or fault is a server, Figures 4.5, 4.6, 4.7 and 4.8 illustrate the scenarios in which light traffic has been captured from a server and the figures depict standard traffic, server malfunction and connection failure, broadcast storm and babbling node, respectively.

Figures 4.5 and 4.6 illustrate the behaviour of the network under two distinct traffic conditions that are directed towards the server. The first figure shows the network workloads under normal traffic conditions with no attack and it is observed how this effect all of the six extant MIB features. However, in Figure 4.6, the scenario is depicted in which the server experiences a problem or there is connection malfunction that leads to a server crash, and demonstrates that all features behave with a specific scale. It is observed that each of the six features is modified when the error commences. Hence, the specific characteristics of this case can be completely detected and this will be beneficial in the process of the classifier distinguishing these abnormalities.

In the scenario where the broadcast storm error occurred, a weak sensitive response was recorded from (ifOutOctets22), as illustrated in Figure 4.7. Although marginal alterations are observed in this feature's behaviour, these changes are minimal in comparison to the other features which exhibit significant modifications in their response. This indicates that these considerable changes that emerge when a fault happens will be critical in the process of the classifier recognising the unique features and situations. In the investigation, the final error features related to the babbling node error under light conditions where the server was function was repeatedly compared with the normal scenario. It was evident that traffic abnormalities were observed in each of the features apart from the first (iflnNUcastPkts22) in which there was only a marginal alteration that was not clearly obvious.

As can be observed in the aforementioned examples under light conditions from both the server and the router, the behaviour of the traffic was almost completely altered. These transformations in the traffic behaviour will be regarded as real features that can be utilised by the classifier in the diagnosis of each individual error in terms of the situation and the resources that are assigned.

Appendix A and B depict the traffic behaviour when the situation changes to include heavy workloads. These alterations are instigated by modifying the sizes of the packets and enlarging the bandwidth, which has an immediate impact on all of the features and also decreases the network performance.

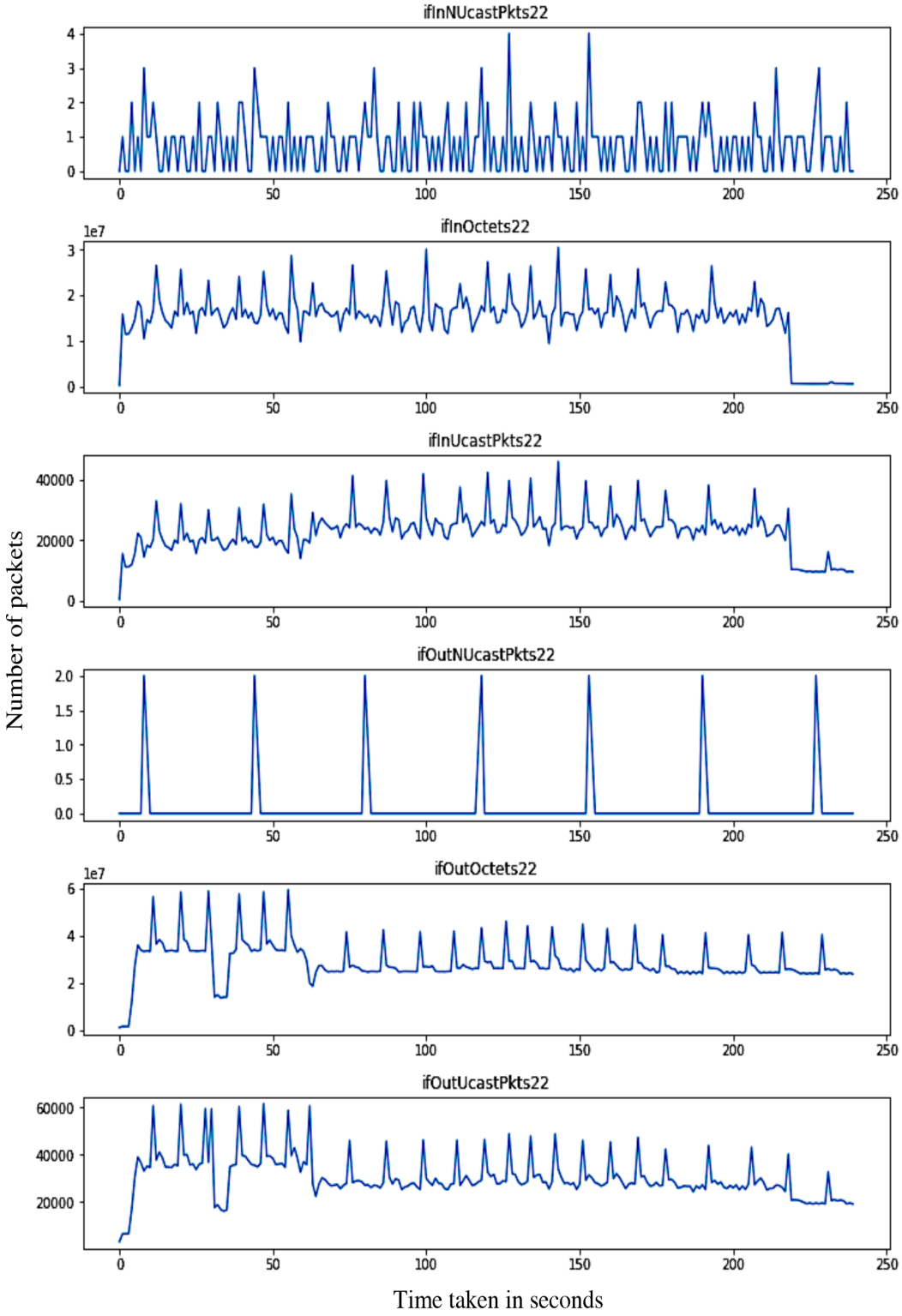

Fig. 4.5 The normal light workloads in a server

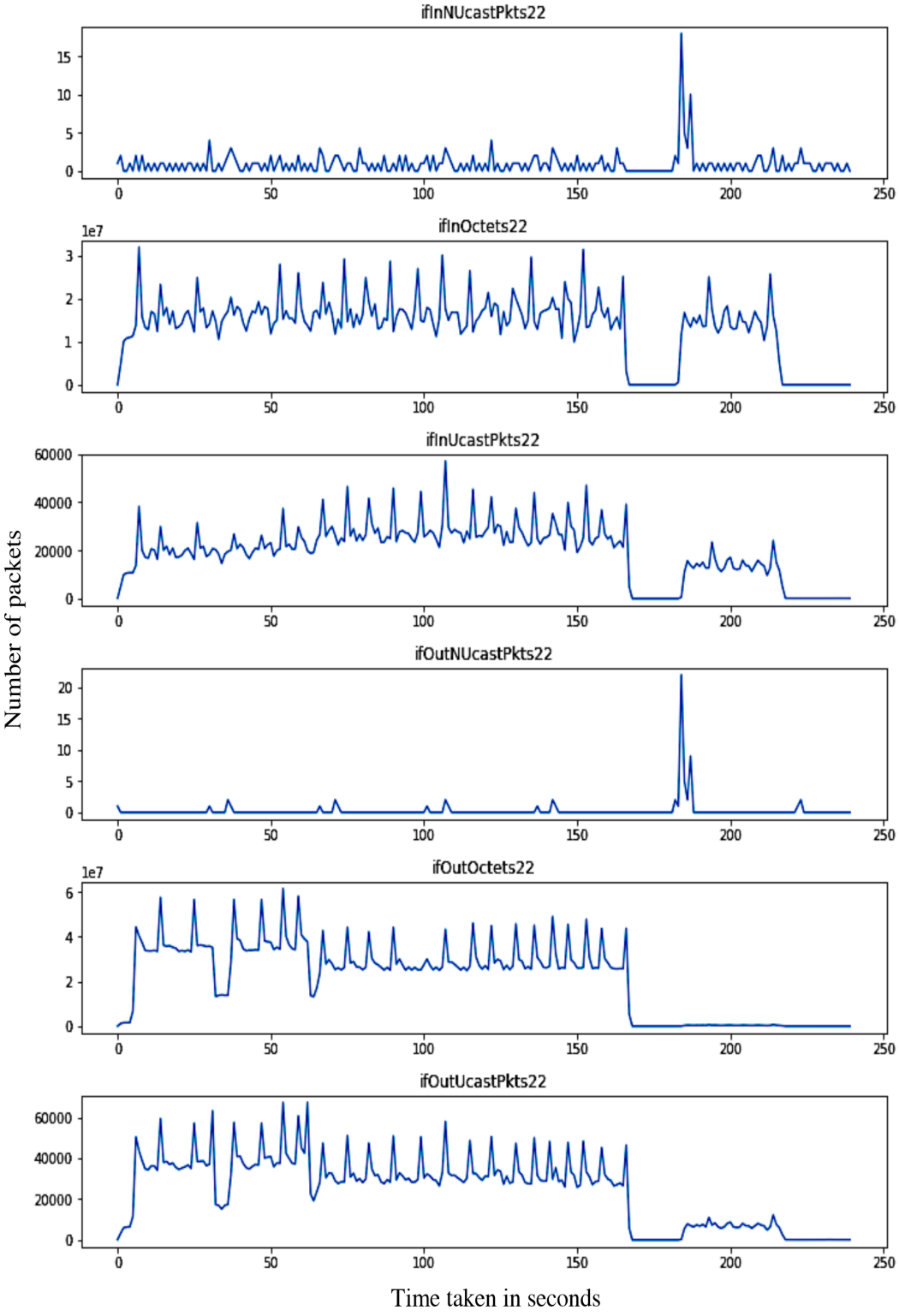

Fig. 4.6 The light workloads of link failure or server crash faults in a server

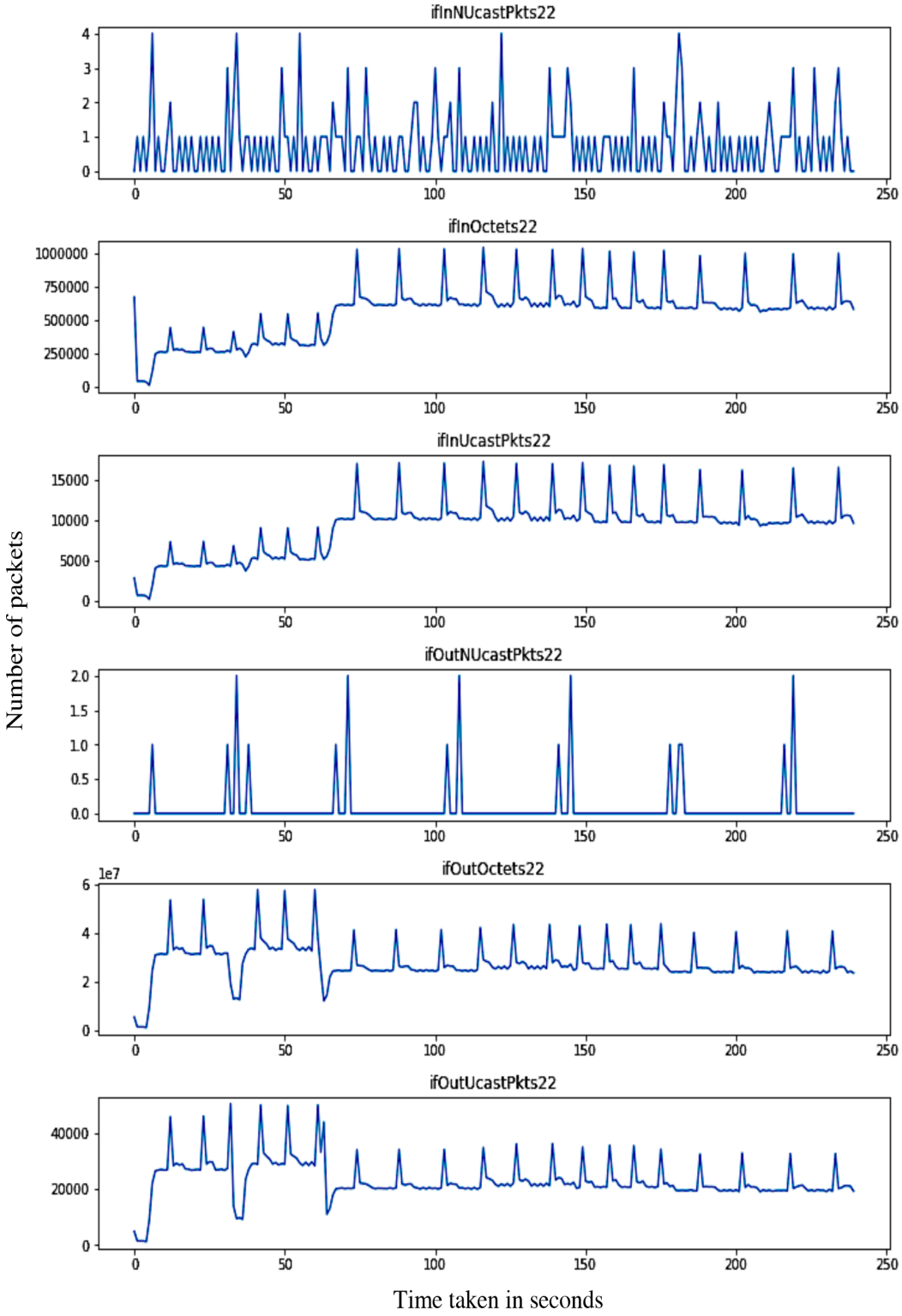

Fig. 4.7 The light workloads of the broadcast storm faults in a server

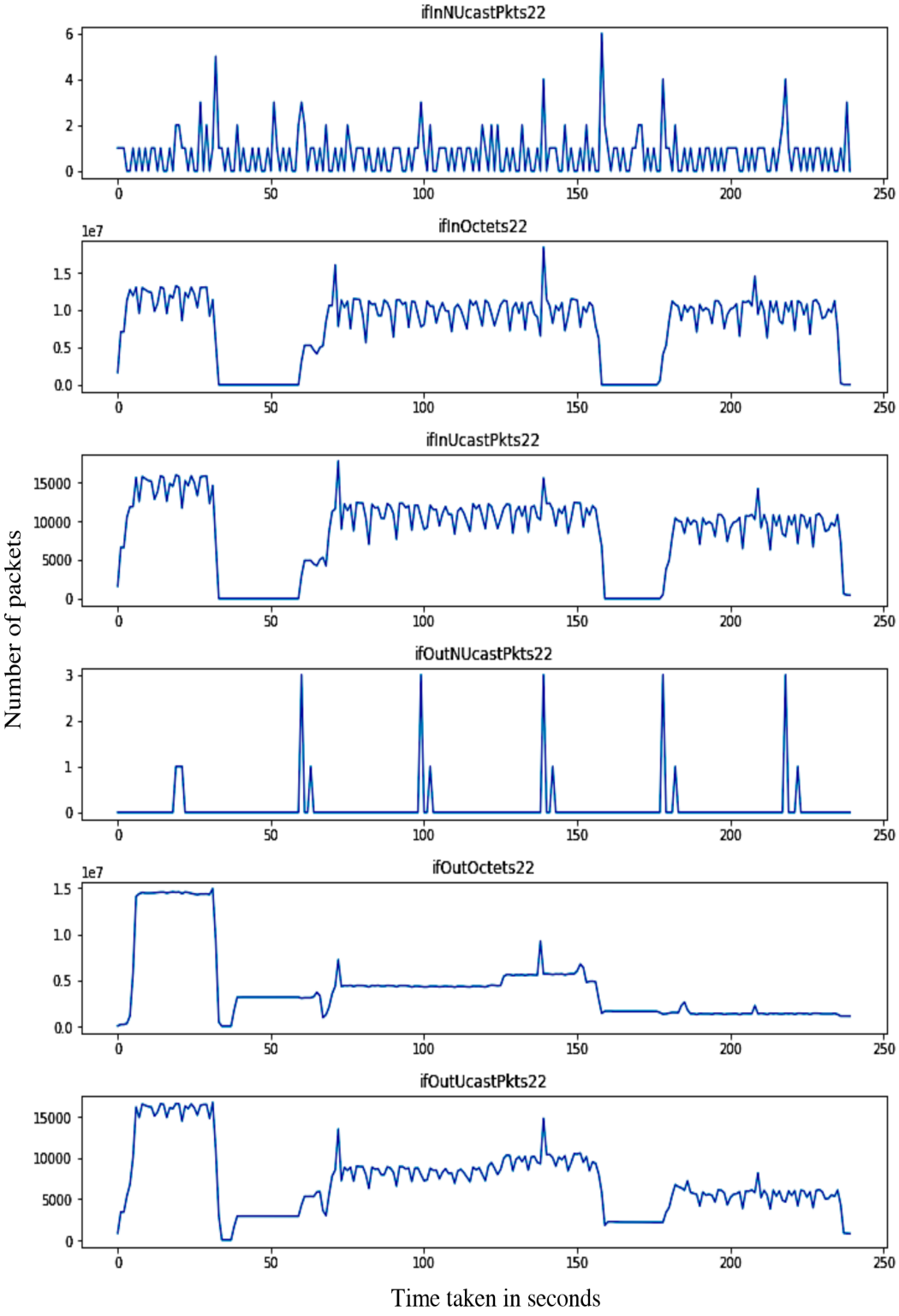

Fig. 4.8 The light network workloads of babbling node faults in a server

4.3 Data Pre-processing (Filtering)

In machine learning, the data is fed into the algorithm and, based on the kind of project being implemented and the kind of learning, it can be supervised or unsupervised. Nevertheless, the data should be in a particular format and in reality, it is frequently in the form of raw data that is impure and has not been processed. Moreover, in order to make the data ready, it must be pre-processed before it can be introduced into the suggested model. This can be an arduous task that takes considerable time, but it is necessary to ensure that the data has been pre-processed in an appropriate manner. If this is not accomplished, the algorithm may not operate effectively and the data in the algorithm would be meaningless. Hence, based on the scope of the project and the specific data as well as additional variables, the pre-processing techniques used can include binarisation, scaling, mean removal, normalisation, among others. Thus, pre-pre-processing is a necessary step to ensure precision as the majority of algorithms only accept standardised data or at a minimum, numerical data. In the event that there is inconsistency in the data or excessive noise, it must be processed and rendered compatible with the classification algorithms.

In the dataset in this case, there are two separate groups of data with 6 and 12 features respectively, with values ranging from 0 to billions. This project designed standard scaling and normalisation filtering by employing the Scikit Learn and NumPy in Python in order to import the primary functions for the pre-processing (See Appendix C and D). This will ultimately reveal that pre-processing facilitates increased efficiency in terms of the data usage and machine learning.

4.3.1 Feature Scaled

This section demonstrates the feature scaling that utilises the Scikit-Learn and NumPy libraries in Python. It is firstly necessary to understand why such feature scaling is necessary. Analysis of Table 4.4, which shows a sample extracted from scenario with light server network workloads, indicates that there are 6 columns and 16 rows in total, with each column representing a unique feature. Comparing the different features, it can be seen that the values range between 0 and 195,813,476, which means that that there is significant disparity among the individual features.

In the event that this non-normalised data is entered into the classification algorithm and is not scaled, it will not conform to the expectations and will result in erroneous outcomes. Hence, this explains the necessity behind pre-processing as, if it was not scaled within a certain range, it will have an impact on the results.

Table 4.4 A non-normalized and non-scaled data of heavy server traffics

F1	F2	F3	F4	F5	F6
3	16077868	24028	0	660253	12202
0	12540604	14981	0	412072	7604
0	5169871	11724	0	325122	5967
2	8870	4895	0	137752	2524
0	7722	139	0	10402	139
1	8248	117	0	7766	117
0	8034	131	0	8522	131
1	7486	122	0	8036	122
1	7593	116	1	8688	115
3	8402	113	2	8974	111
1	7256	130	0	7020	130
1	9092	114	0	7604	114
0	195813476	164366	0	135829492	158493
1	123571346	163731	0	150383466	157262
0	90400182	142205	0	135431872	155354
1	123870120	263151	0	147443638	269078

If there is no scaling, no common range exists into which the data can fall. In order to generate a range with high similarity in addition to a similar mean or standard deviation, feature scaling will conduct a statistical equation for each feature within the data. Different kinds of scaling functionality can be utilised, including MinMaxScaler and Standard Scalar, which were both employed for the purposes of this research. Nevertheless, there are numerous additional scalars in the Scikit-learn library, like Normalize Data and Binarize Data, but based on the nature of the datasets in this project, only two were utilised.

4.3.1.1 Standard Scaler

Based on the data in this project, itn is necessary to implement this scalar, as the primary objective of this method is to render the means of the individuals similar or approximate to each other, while the features of the original data generate different means, as indicated in Table 4.5.

It is acknowledged that the features associated with a certain error have significantly different means. The objective of this project is to considerably increase the similarities between the features by utilising zero mean and unit variance throughout the extant data. After the project data has been imported into Python, the pre-processing library should

Table 4.5 Means of heavy server network traffic features

	F1	F2	F3	F4	F5	F6
	3	16077868	24028	0	660253	12202
	0	12540604	14981	0	412072	7604
	0	5169871	11724	0	325122	5967
	2	8870	4895	0	137752	2524
	0	7722	139	0	10402	139
	1	8248	117	0	7766	117
	0	8034	131	0	8522	131
	1	7486	122	0	8036	122
	1	7593	116	1	8688	115
	3	8402	113	2	8974	111
	1	7256	130	0	7020	130
	1	9092	114	0	7604	114
	0	195813476	164366	0	135829492	158493
	1	123571346	163731	0	150383466	157262
	0	90400182	142205	0	135431872	155354
	1	123870120	263151	0	147443638	269078
Mean	**0.9375**	**35469760.6**	**49378.9375**	**0.1875**	**35668167.4**	**48091.4375**

also be imported to facilitate the process of declaring and importing the conventional scaler function.

Subsequent to the creation of the standard scaler function and the application of the imported data, the StandardScaler makes the assumption that the data is distributed normally within every feature and will then apply scaling to ensure that the distribution is centred around 0, with a variance of 1. Both the mean and standard deviation are determined for the feature and the feature is then scaled based on:

$$S(x_i) - mean(x)/stdev(x) \tag{4.1}$$

In terms of the data in this project, after the standard scaler is applied utilising Equation 4.1, the range alterations of each of the features will directly influence the means and will enable the means to be distributed in a more uniform manner in comparison to the original, as demonstrated in Table 4.6.

This form of scaling is defined as immersing, which enhances the system by generating improved data. This is the perfect type of scale for datasets where the feature ranges have significant differences, as it scales down the ranges to establish means across the features that have strong similarities. It is also called the standard score or Z-score, in which the resulting value fluctuates around zero and generally ranges between minus three and positive three,

Table 4.6 The data sample after standard scaling

	F1	F2	F3	F4	F5	F6
	2.4	-4.5	-0.9	0.7	-2.4	0.8
	0.8	2.7	4.5	-0.7	-3.6	0.8
	0	-4.5	-3.6	2.1	2.4	2.4
	-0.8	-6.3	1.8	1.4	1.2	2.4
	1.6	1.8	2.7	-0.7	1.2	1.6
	1.6	2.7	0	-1.4	-2.4	2.4
	-0.8	-0.9	4.5	0	2.4	-2.4
	0	-5.4	4.5	-0.7	2.4	-2.4
	-2.4	3.6	-3.6	-1.4	3.6	-2.4
	1.6	1.8	0.9	0	2.4	-0.8
	0.8	-5.4	4.5	-2.1	-2.4	0
	-2.4	1.8	1.8	-0.7	-1.2	-0.8
	1.6	-2.7	-1.8	0	1.2	-2.4
	-1.6	0.9	-1.8	0	2.4	0.8
	-3.2	0	-0.9	-1.4	1.2	-0.8
	0	-0.9	-3.6	0	2.4	0.8
Mean	**-0.05**	**-0.95625**	**0.5625**	**-0.30625**	**0.675**	**-5.55E-17**

but could also be higher or lower. After this process, additional scaler can be applied if it is determined that the data requires additional enhancement in order to generate improved classification results.

4.3.1.2 Data Normalisation

Normalisation is an additional technique utilised for pre-processing data, which is applied to each vector's value in order to convert them on a standard scale, generally between 0 and 1. If the application of normalisation leads to loss of information in relation to minimum and maximum, this is not the ideal scenario; however, it is particularly applicable to machine learning algorithms that will generally converge at a more rapid pace on normalised data on algorithms such as Fuzzy C-Means and Probabilistic Neural Network Classifiers, which are the two methods employed in the model proposed in this study.

The normalisation of data is a method with significant importance in the pre-processing of data. Analysis of the dataset in this research indicates that the feature values have a wide range from 0 to billions, as shown in Table 4.4. Hence, it is necessary to normalise these values in order to ensure that there is consistency in the range of values. This process can facilitate statistical analysis of the traffic. By increasing the consistency of the ranges between the variables, normalisation allows an equal comparison among features that are

highly dissimilar, which ensures that their impact is the same. This also has importance in the computational sense.

Considering that this sample was extracted from part of our dataset, which was comprised of six features with some of the features ranging between 0 and 3, while other features ranged between 7,256 and 195,813,476, it can be observed that the second feature is one billion times larger than the first feature, which implies that these features have completely different ranges. Based on additional analysis, such as the classification suggested in this study, the attribute of the second feature will have a greater intrinsic effect as a result of the larger values. Nevertheless, this does not automatically imply that it is useful in the classification process. Hence, due to the nature of the data biases, the second feature has increased weight in the classifier model than the first. In order to prevent this scenario, the variables can be normalised from individual features in values ranging between zero to one. In machine learning, the most frequently used normalisation techniques are L_1 and L_2 normalisation. L_1 represents the least absolute deviations and when applied to a dataset, it guarantees that the sum of the absolute values equates to 1 for each row. While this kind of normalisation exhibits no sensitivity to outliers, this is taken into consideration in L2 during training and it is concerned with the least squares where the sum of the squares equals 1. In this case, the L1 norm is applied to the datasets as it has the least impact from outliers; this is because outliers are commonly found in raw data. The $L_1 - norm$ is also called least absolute deviations (LAD) and least absolute errors (LAE. Fundamentally, it minimises the sum of the absolute differences (L_1) between the estimate values $f(x_i)$ and the target value (X_i):

$$L_1 norm = \sum_{i=1}^{n} |x_i - f(x_i)| \tag{4.2}$$

After this normalisation process is applied using Python Scikit-learn, the data is normalised to values ranging between 0 and 1, as shown in Table 4.7.

Fundamentally, Scikit-learn in Python enables the application of normalisation and the pre-processing module to all individual data values. After normalisation has been completed, the values will scale between 0 and 1.

4.4 Chapter Summary

The aim of this chapter was to provide clarification on the datasets and to elucidate on the common faults on which our proposed classifier was applied. This chapter demonstrates what the data features represent and from where they are sourced. Subsequently, the nature

Table 4.7 The Normalised dataset

F1	F2	F3	F4	F5	F6
0.52938231	0.14731704	0.0737471	0.14277039	0.04748989	0.28674061
0.09325844	0.36822565	0.1943704	0.22890467	0.3305028	0.08926456
0.60152493	0.27750246	0.01664518	0.13990671	0.0065761	0.05452355
0.0456765	0.0512256	0.35745235	0.13490805	0.16355517	0.52246469
0.06653934	0.0604762	0.05661	0.92719481	0.03929044	0.68902484
0.45159713	0.12695724	0.2736517	0.01928027	0.27707422	0.00485554
0.51760907	0.36234016	0.04674098	0.06143215	0.10362904	0.09609106
0.51803926	0.07216584	0.40747458	0.07895334	0.08273807	0.02579091
0.08192784	0.24081209	0.68134269	0.04110054	0.00315156	0.17336194
0.0890278	0.13269894	0.22379859	0.1695778	0.33018482	0.2191833
0.0707298	0.03745926	0.03597818	0.05920655	0.41356511	0.50688935
0.21525665	0.1552106	0.03113147	0.02469973	0.36500207	0.32312341
0.78685471	0.25719348	0.0782765	0.09157331	0.05578857	0.04189171
0.12709372	0.17237019	0.06465674	0.68556646	0.18637117	0.00112451
0.10402913	0.09587855	0.1033978	0.09864662	0.39991911	0.49832797
0.76331491	0.12596821	0.00054514	0.17456929	0.006482	0.08271442

of each of the individual faults is stated as well as how the owner of the data created the scenario. The case study is explained along with how the parameters for each dataset were captured by the prior researcher.

Next, in order to comprehend and obtain a broader perspective on all the datasets and the features contained within them based on the scenarios and allocated resources, the features of the datasets were visualised and by using graphical representations, the sensitivities of each feature are observed as well as how their behaviour is altered when the scenario or device is changed.

Furthermore, the final section of this chapter focuses on the manipulation and pre-processing of the data. In order to refine the data and remove any noise from the datasets, specifically to prepare data that is consistent, two different filtering is created to normalise and scale the data points as shown in appendix C and D. If the raw data is applied to the model without pre-processing, it will lead to confusion and deteriorate the overall performance as the ranges of the datasets will have considerable differences. Finally, the chapter is intended to provide clarification and demonstrate the nature of the datasets as well the specific methods that projects require for the purposes of data filtering in order to generate optimal sample data before it is sent to the proposed classifier.

Chapter 5

Results discussions and evaluations

5.1 Introduction

In this chapter, the classification results of the four different types of traffic in different scenarios and with different allocated resources, such as server and router, are presented. The different sections show the explanation and results of different types of computer network faults that detected heavy or light network loads. The chapter also discusses the aim of using the unsupervised fuzzy cluster means technique to determine its influence in increasing the performance of the proposed model.

Further, an evaluation of the obtained results of different machine learning algorithms, including the proposed hybrid model SFPPNC and PNN classifier, will be presented. Experiment results from the proposed hybrid system and the existing algorithms will be analysed in detail. In the light of the experiment results, the evaluation for each scenario and specific faults will be considered and explained. The last section presents a detailed comparative classification result evaluation between the existing classifier, PNN, and the proposed semi-supervised method, SFPNNC.

5.2 Comparative Experimental Results of SFCM, K-means and EM

The main aim of using the unsupervised method in this research is to facilitate the classification process by segmenting each different pattern of the faults and forming groups for any different detected traffic. Thus, this process will enrich the research to provide an application that could overwhelm any Big Data as one of the common research concerns is in seeking to

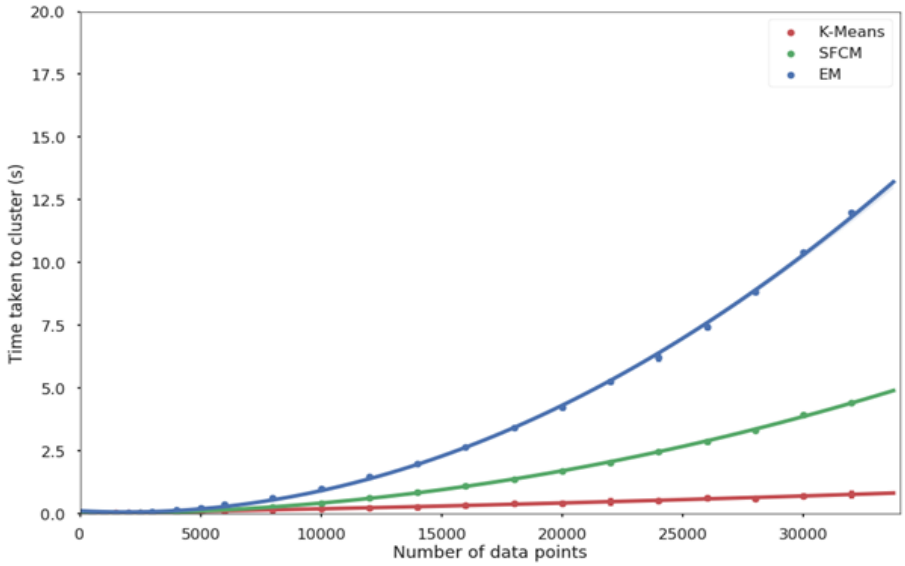

Fig. 5.1 Performance Comparison of K-means, SFCM and EM implementations

find a suitable solution for handling the problems Big Data presents. According to previous studies, numerous clustering algorithms exist to perform this procedure, depending on the dataset's nature, that ideally could be adapted, as highlighted in Chapter Three.

In accordance with previous research in the literature, Fuzzy C-means, K-means and Expectation Maximisation (EM) are the three most common cluster algorithms and have been selected and applied to a sample of our datasets. We have noticed that, due to its fuzziness nature, FCM achieved optimal clustering results by which each data point can be very precisely indicated to be a member of whichever group of clusters.

We have already solved the performance issue as the main limitation of FCM, and improved the performance results by overcoming the initial cluster centres, as approved by consolidating a subtractive clustering method with the FCM. We generated data from our datasets consisted of 32,000 data points and used different tests to compare how each method outperformed. Figure 5.1 shows the performance of three different cluster algorithms (Subtractive Fuzzy Clustering Means [SFCM], K-means and EM) involved in the experiment. As is shown on the graph, our optimised model (SFCM) competed with the K-means by achieving roughly the same performance over the clustering of the first 10,000 data points, while the K-means is known for its best performance. After the data scaled out, the SFCM became slower than the K-means, by which the time taken slightly increased and, for the total of the data points implementations, consumed around four seconds. Further, the result showed that SFCM accomplished a better performance than EM, which needed 14 seconds to complete the implementation.

Thus, among the existing clustering methods, the hybrid SFCM, K-means and EM are used to apply to the datasets so as to select the optimal number of clusters.

Having discriminated each group with a specific feature following feature analysis in Chapter Four, it can be seen very clearly that the features of ifOutOctet22 and ifNUcastPkts22 fall into a distinct area for each particular scenario considering their allocated resources. These two features mainly distinguish that each traffic pattern will be grouped into a different range. While the server only takes six variables from MIB groups, the router holds more features from MIB, and consists of 12 MIB features. However, to unify all network loads that have the same features in their traffic, the above two stated features could be considered to form an optimal clusterisation.

This process will optimise the classification process wherein the output of the clustering procedure represents a distinct vector of features and each vector comprises the data points of the specific traffic, including abnormal and the normal traffic, according to their scenario and devices. The clustering plays a vital role in the classification process, and its impact will be seen in the next sections of this chapter. The performance of the classification shows how it is developed through the data simulated through the clusterisation and how the performance will be when the data are directly forwarded into the classifier.

According to one of our samples of the dataset that included all different traffic patterns from all different scenarios and distinct devices, such as server and the router, using the several libraries in Python, such as Numpy, Pandas and Scikit, learning for data analysis, we clustered our set of data. It should reach its optimality when having four clusters, several times, with between two and nine clusters, as shown in Figure 5.2. The partition coefficient (PC) is defined on the range from 0 to 1, with one being best. It is a metric which tells us the purity by which a specific model describes our data and is calculated using the equation below:

$$V_{pc} = \frac{1}{N} \sum_{i=1}^{c} \sum_{j=1}^{N} (\mu_{ij})^2 \tag{5.1}$$

where N represents the number of datapoints, c is the cluster numbers and uij pictures the vector of the membership function degrees.

The experiment result in Figure 5.2 showed that PC for the SFCM is the greatest, which is 0.96, when obtaining four different clusters; as such, it can be stated that SFCM achieved an optimal value of clusters.

For the same number of clusters using K-means and EM methods, as depicted in Figures 5.3 and 5.4, the PC was minimised and reached to 0.67 and 0.73, respectively. In this case,

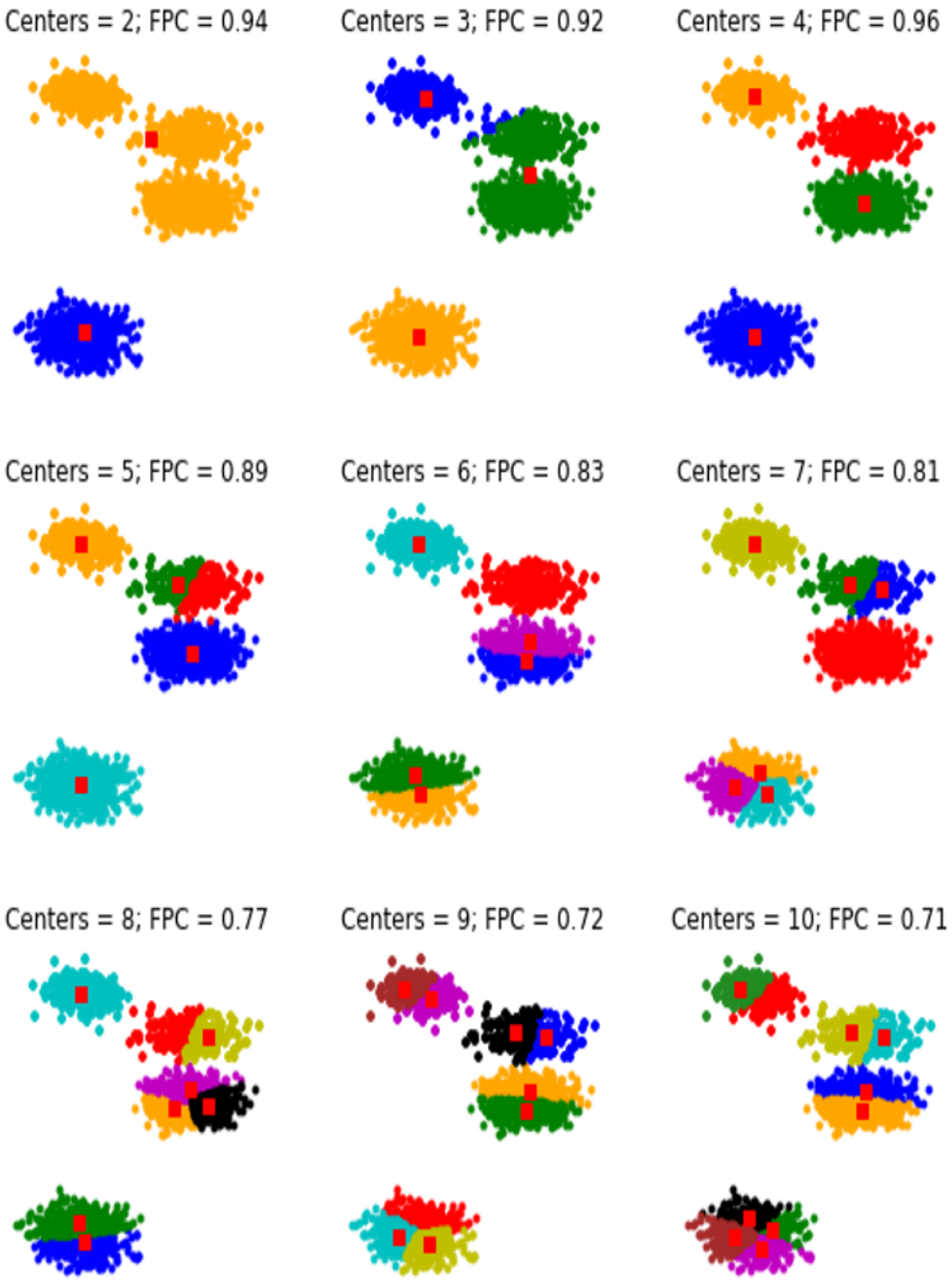

Fig. 5.2 The PC of the proposed SFCM for nine different tests

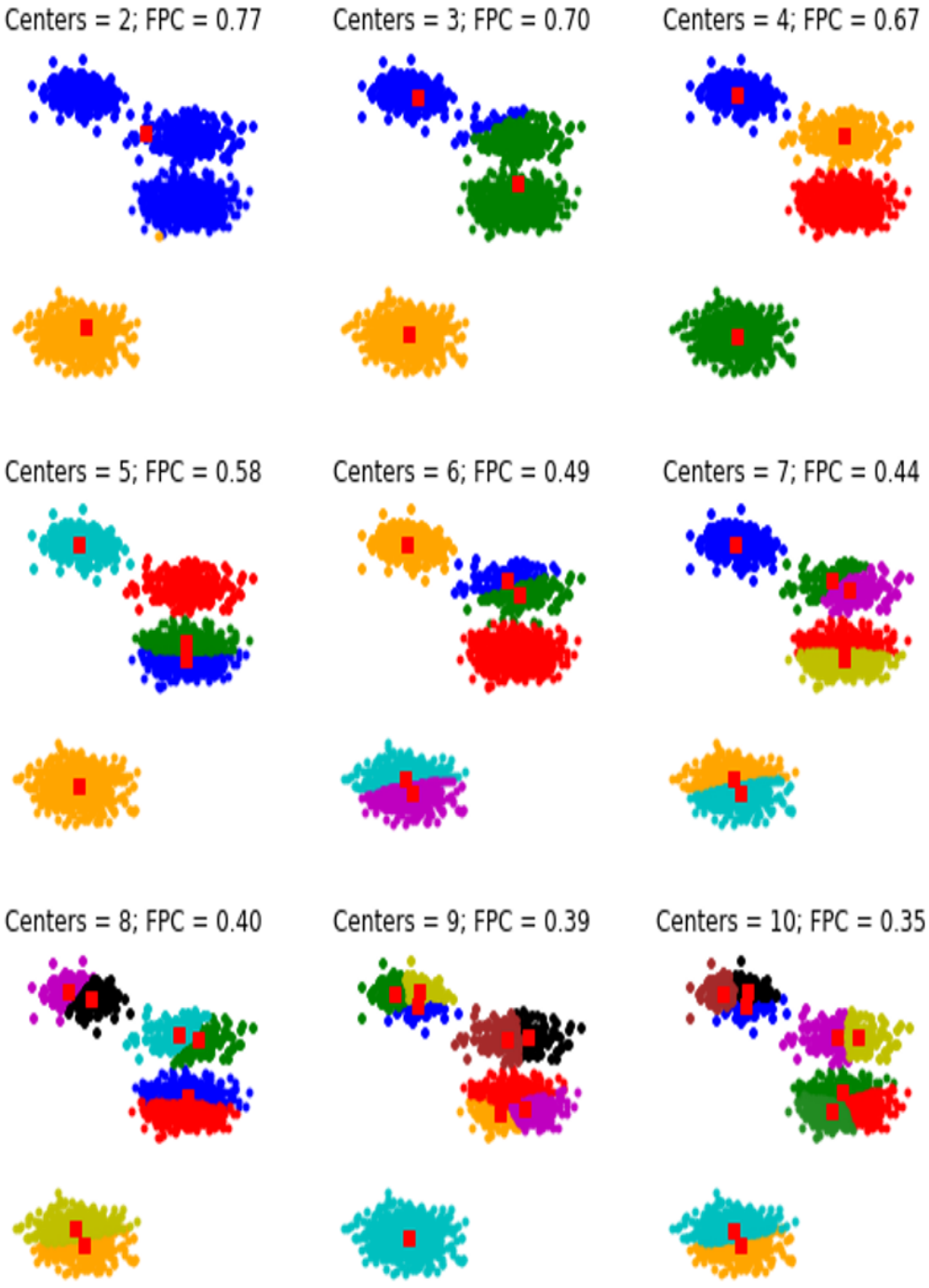

Fig. 5.3 The PC of K-means for nine different tests

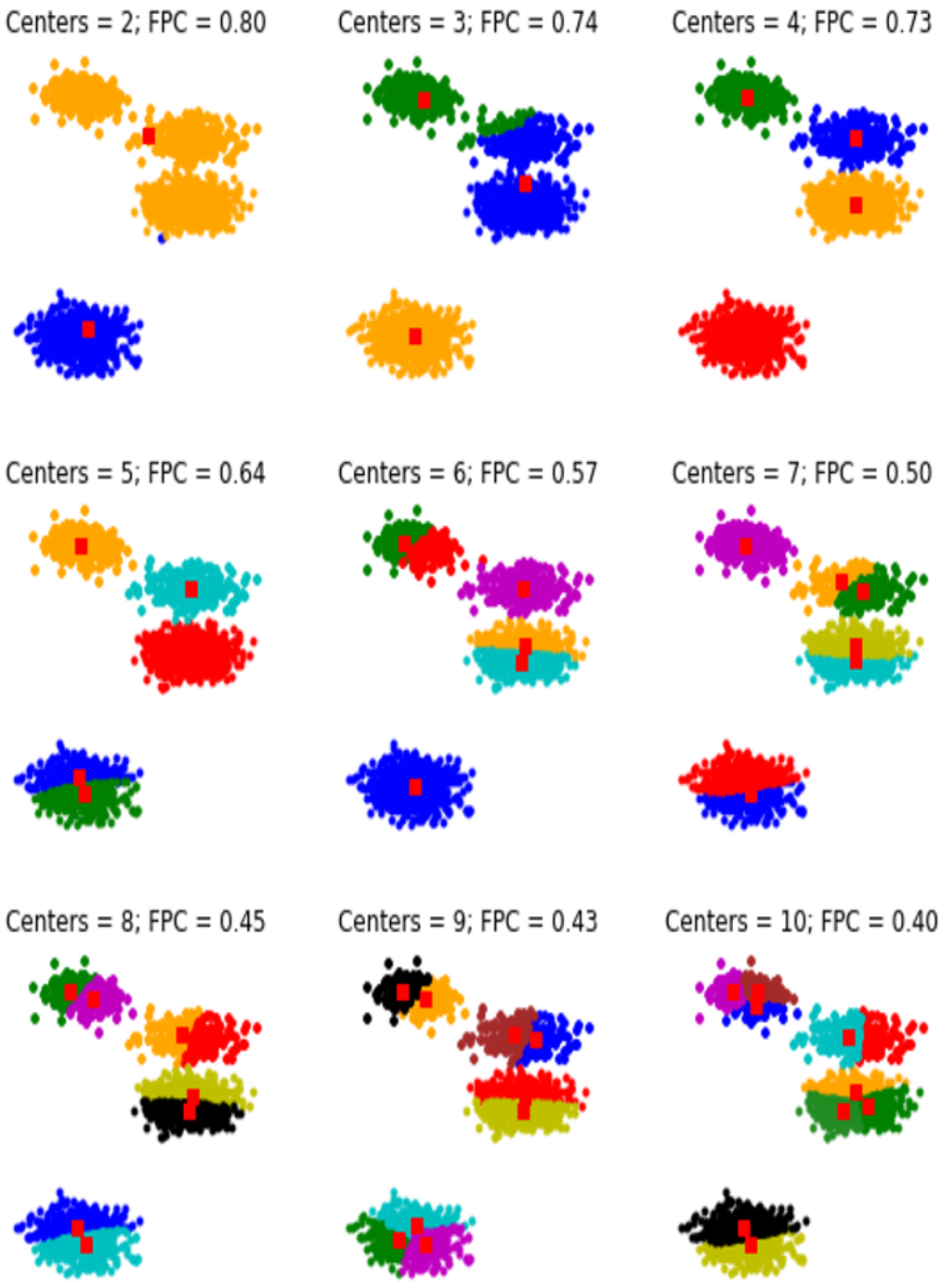

Fig. 5.4 The PC of EM for nine different tests

applying K-means and EM proved that these two methods are not suited for this kind of data to achieve well-compacted clusters.

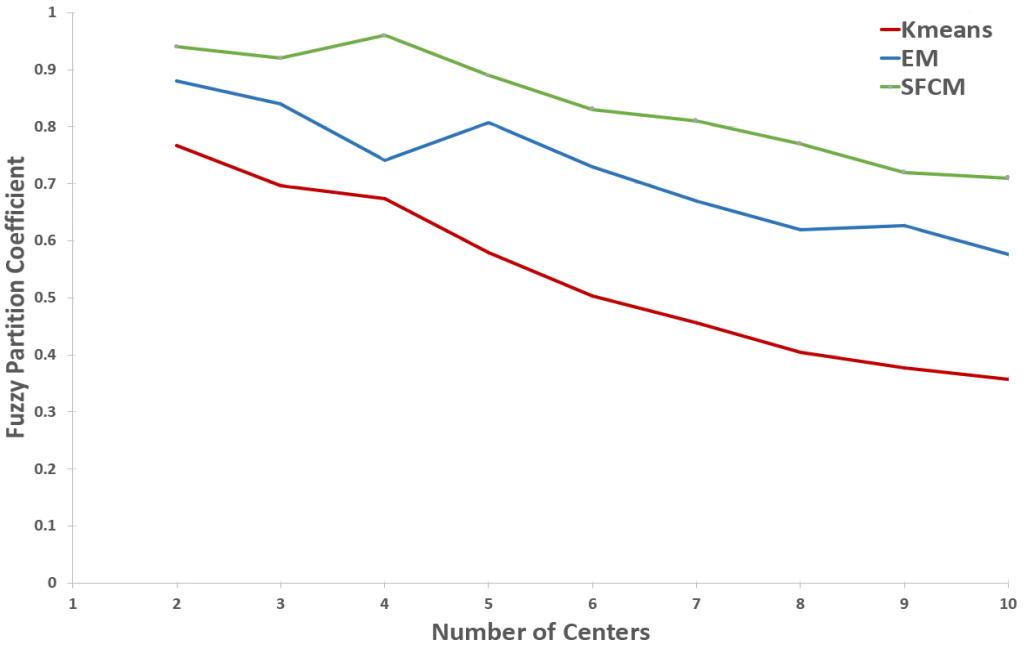

Fig. 5.5 The PC curves of the SFCM, K-means and EM

Each of the above clusters shown in Figure 5.2, 5.3 and 5.4 represents the particular scenario with its abnormal and normal traffic. We will then show the results of the clustering and plot the partition coefficient (PC) for each used method. When the PC is maximised, our data are described best and, according to this sample, the optimal cluster number reaches the highest when it has four clusters based on SFCM and, contrarily, approaches poor outcomes when K-means and EM are applied, as shown in Figure 5.5.

Hence, each of the above clusters shown in Figure 5.2, 5.3 and 5.4 represents the particular scenario with its abnormal and normal traffic.

5.3 Classification of Evaluations

At the beginning of our research, we focused on the different types of scenarios that the traffic captured. This first step was made to show from which devices the traffic faults come and in what condition they were created. There are several types of faults, and the most dangerous and commonly appeared in the network belong to the server or router. Mainly, the connection is lost when there is a crash or link failure or problems in the Ethernet or freezing of the system due to broadcast storming messages.

Each network application and service have different requirements according to bandwidth, delay, packet loss, and other network parameters. Therefore, in order to assess the quality of the particular session, we need to know what application or service is associated with the examined network flow. To assess the quality in real-time, the involved traffic classification techniques must fulfil some crucial requirements. Therefore, considering the retaining performance of the network and its transparent way to users is essential.

In this research, the classification results of various traffic faults are analysed, and different metrics are used to show how the performance will be affected when there is a change in the scenarios or allocated resources. Before starting the visualisation of any results, there are some main factors which should be declared to grasp the core of the problem, such as all metrics that have been used in this study to assess the quality of the work, for example, ROC curve, or in what circumstances and for what aim micro and macro are used. These are explained in the following sections.

5.3.1 Receiver Operating Characteristic (ROC) curve

ROC curve stands for Receiver Operating Characteristic and is a commonly used way to visualise the performance of a binary classifier in a graphical plot and can also be used to compare models and classifiers, as shown in Figure 5.6. In other words, mathematically, an ROC curve can be defined as a relationship between the true positive rate and the false positive rate. True positive rate means sensitivity, which means how many classes truly have these types of faults or abnormalities, while false positive means what was thought to be true positive was, in fact, not.

The dotted line shown in Figure 5.6is used as a base line, so a bad classifier will have performance that is random, possibly even worse than or slightly better than the random classification dotted line. A reasonably good classifier will give an ROC curve that is consistently better than random across all decision threshold choices, as drawn with blue, red and green solid lines, while an excellent classifier would be one such as the orange line shown in the above figure. Binary classification is the task of classifying the members of a given set of objects into two groups on the basis of whether they have some property or not. There are four possible outcomes from a binary classifier, see Figure 5.7.

- True positive (TP): predicted to be positive and the actual value is also positive
- False positive (FP): predicted to be positive, but the actual value is negative
- True negative (TN): predicted to be negative and the actual value is also negative

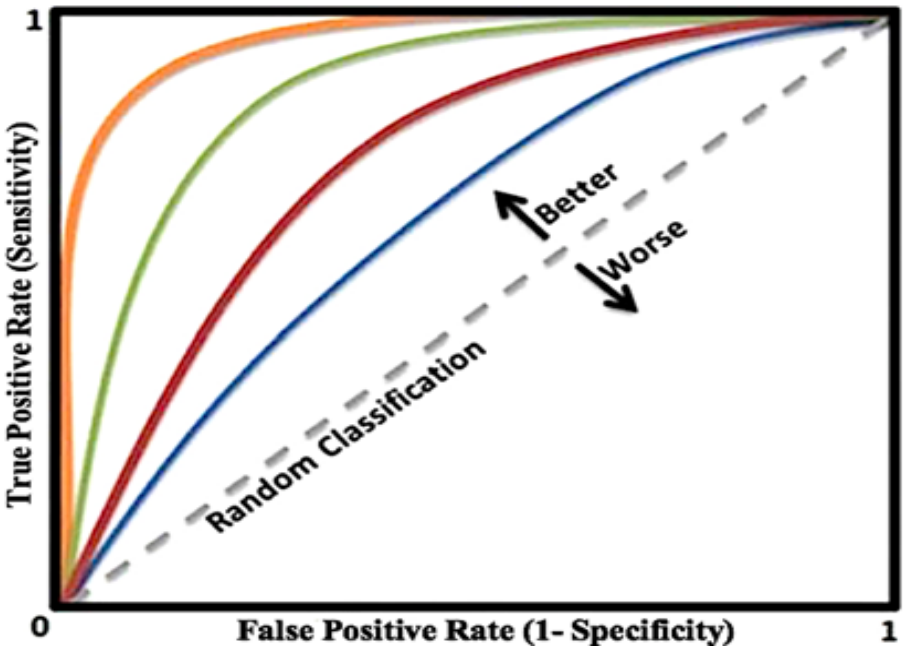

Fig. 5.6 ROC curve sample

- False negative (FN): predicted to be negative, but the actual value is positive

From the above numbers, the following can be calculated:

$$Sensitivity = TruePositiveRate : TPR = \frac{positive\ correctly\ classified}{True\ Positive + False\ Negative} = \frac{TP}{TP+FN} \quad (5.2)$$

$$Specificity = TrueNegativeRate : TNR = \frac{negatively\ incorrectly\ classified}{True\ Negatives + False\ Positives} = \frac{TN}{TN+FP} \quad (5.3)$$

The receiver operating characteristic (ROC) curve is a two-dimensional graph in which the false positive rate is plotted on the X axis and the true positive rate is plotted on the Y axis. The ROC curves are useful to visualise and compare the performance of classifier methods.

5.3.2 Macro-average precision

Macro-averages will measure slightly different things to micro-average precision, and, thus, their description differs. A macro-average will calculate the metric independently for each class and then take the average (hence, treating all classes equally), whereas a micro-average will aggregate the contributions of all classes to compute the average metric. In a multi-

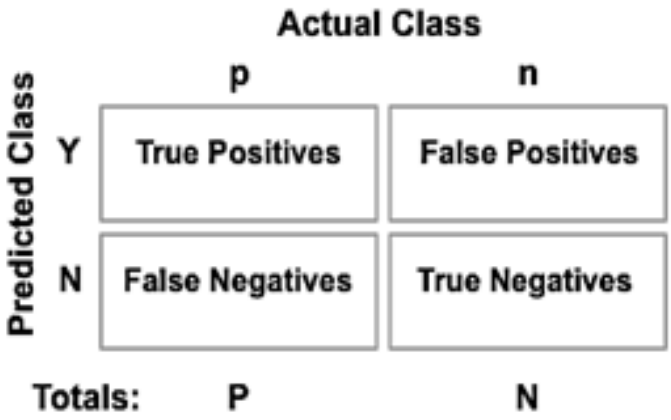

Fig. 5.7 The confusion matrix

class classification setup, micro-average is preferable if you suspect there might be a class imbalance.

In this research, we are dealing with a multi-class classification problem wherein we have four classes for each specific scenario and device consideration.

The critical aspect of macro-average precision is that each class has equal weight. There are two steps to compute the macro-average precision. The first one is to compute the metric within each class to calculate classes correctly predicted by the classifier, then taking the average of all corrected classes. To see what exactly macro-average means and how to compute it, we are going to explain with an example of our dataset. Table 5.1 shows a sample dataset that has been extracted from our heavy scenario traffic in the server.

In this table, we have three columns where the first is the actual class of an example, the second is the predictive class from some classifier where the last is a binary variable that denotes whether the predicted class matches the actual class. Our study deals with a multi-class classification problem in which we have four classes. We have several instances in this sample of our data, where the five first instances are server crash, four instances for babbling node traffic, two instances of broadcast storm and the last two instances are normal traffic, as shown in Table 5.1.

First, we will compute the macro-average precision, which mainly calculates the performance of each class, then average them using the equation below:

$$Macro - Average - Precision = \frac{\sum P_i}{N} \tag{5.4}$$

where P is the precision of each class and N represents the number of classes in the

Table 5.1 The sample class prediction of the heavy network workloads of the server

Class (Network Traffic)	Predicted Class	Correct Prediction
Server Crash	Broadcast Storm	0
Server Crash	Broadcast Storm	0
Server Crash	Server Crash	1
Server Crash	Server Crash	1
Server Crash	Normal	0
Babbling Node	Babbling Node	1
Babbling Node	Babbling Node	1
Babbling Node	Babbling Node	1
Babbling Node	Broadcast Storm	0
Broadcast Storm	Broadcast Storm	1
Broadcast Storm	Broadcast Storm	1
Normal	Normal	1
Normal	Normal	1
Normal	Normal	1

Table 5.2 Precision and macro-average of the sample

Class	Precision
Server Crash	2/5 = 0.40
Babbling Node	3/4 = 0.75
Broadcast Storm	2/2 = 1.00
Normal	3/3 = 1.00
Macro Average Precision:	
(0.40 + 0.75 + 1 + 1) / 4 = 0.78	

sample. This type of the precision will be suitable for cases in which each class has equal weight, where each of these classes will contribute one-fourth weight toward the final macro-average precision value. As we highlighted at the beginning of this section, in order to compute the macro-average we are going to compute the precision within each class. Table 5.2 illustrates the precision of each class and then calculating the macro-average precision.

As shown in Table 5.2, there are five total examples in the server crash class, and only two of them were predicted correctly by the classifier, which leads to a precision of 0.40. For the second class, the babbling node class, there are a total of four instances, and three of them were predicted correctly, which leads to 0.75 for the babbling node class. In the last two classes, broadcast storm and the normal traffic class, where there were two instances of the broadcast storm and three of normal, the classifier predicted both of these correctly and led to 1.00 precision for both classes. In the final step, we simply average across these four

to produce the final result, the ultimate macro-average precision, by which we can simply compute 0.40, 0.75, 1.00 and 1.00; we then get the final macro-average precision for this set of results of 0.78. We noticed here that, no matter how many instances there were in each class, because we computed the position within each class first, each class contributes equally to the overall macro-average precision.

5.3.3 Micro-average precision

As mentioned in the previous section, that micro-average precision is computed a little differently to the macro-average. It gives each instance in the data results to have equal weight. In micro-average precision, we do not compute for each class separately. We treat the entire dataset as the entire set of results, as in the example given in Table 5.2, which has an aggregate outcome and the largest classes have the most influence. Thus, to compute the micro-average precision, we have to look at how many of all the examples, which are 14 in total, can be calculated based on this equation:

$$P = \frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FP_c} \tag{5.5}$$

where c is the class label, TP represents the true positives and FP states the false positives. Thus, the micro-average precision for the same sample of the data shown in Table 5.2 will be the precision of all examples regardless of class in the set of results. Thus, we have found that the classifier predicted 10 of them correctly out of 14.

$$\text{Micro-average precision} = 10/14 = 0.71$$

In this case, we can see that, if there were a million instances of the server crash class, with micro-average precision, because each instance has an equal weight, that will lead to the server crash class, contributing many more instances to our overall micro-average precision. Therefore, the effect of micro-average precision is to give classes with greater instances much more influence. The average would have been influenced much more by the million server crash examples than by the other three (babbling node, broadcast storm and normal) examples.

Hence, that is the difference between micro-average and macro-average precision. If the classes have about the same number of instances, the macro and the micro-average will be about the same. In a case where some classes are much larger, including more instances than

others and we want to weight the metrics toward the largest ones, micro-averaging precision is more preferable. Contradictory, to weight the metrics toward the smallest classes, we would use macro-averaging.

In samples where micro-average is much lower than the macro-average, we then examine the lager classes for poor metric performance. Conversely, if the macro-average is much lower than the micro-average, then we should examine the smaller classes to see why we are having poor metric performance. Principally, the datasets that we used in our research have almost the same weight for each specific class; therefore, there would not be too many differences between these two metrics.

5.4 Experimental Results of Subtractive Fuzzy Probabilistic Neural Network Classifier (SFPNNC)

This section will explain the results of the classifications based on the proposed method (SFPNNC). As declared in the previous chapters and sections, the datasets were obtained after processing through different steps, such as manipulations and preprocessing and then applying the FCM on them. The classifier receives the outputs stored in vectors in which each of these vectors contains the features, including the normal and abnormal traffic, of a specific scenario considering their allocated resources, such as a server or a router. The segmentation process, which is regularly performed by one of the clustering methods (such as FCM) helps the classifier to quickly identify that each group belongs to a distinct traffic by considering their scenario and the devices the data collected from it.

5.4.1 SFPNNC experiment results and evaluations for light scenario of router traffic faults

Depicting ROC curves is an excellent way to visualise and compare the performance of various traffic types in the computer network system. The features of light network workload for the router are illustrated in Table 1 in Chapter Four, which are the MIB variables of IF group that were extracted from MIB variables databases. Figure 5.8 is generated by performing 2D to show the ROC curves to state the performance comparability of an individual class of the network traffic captured in a light scenario where the victims were the routers. Using the proposed method classifier (SFPPNC), depending on the four types of network workloads (normal, broadcast storm, babbling node and server crash), the datasets

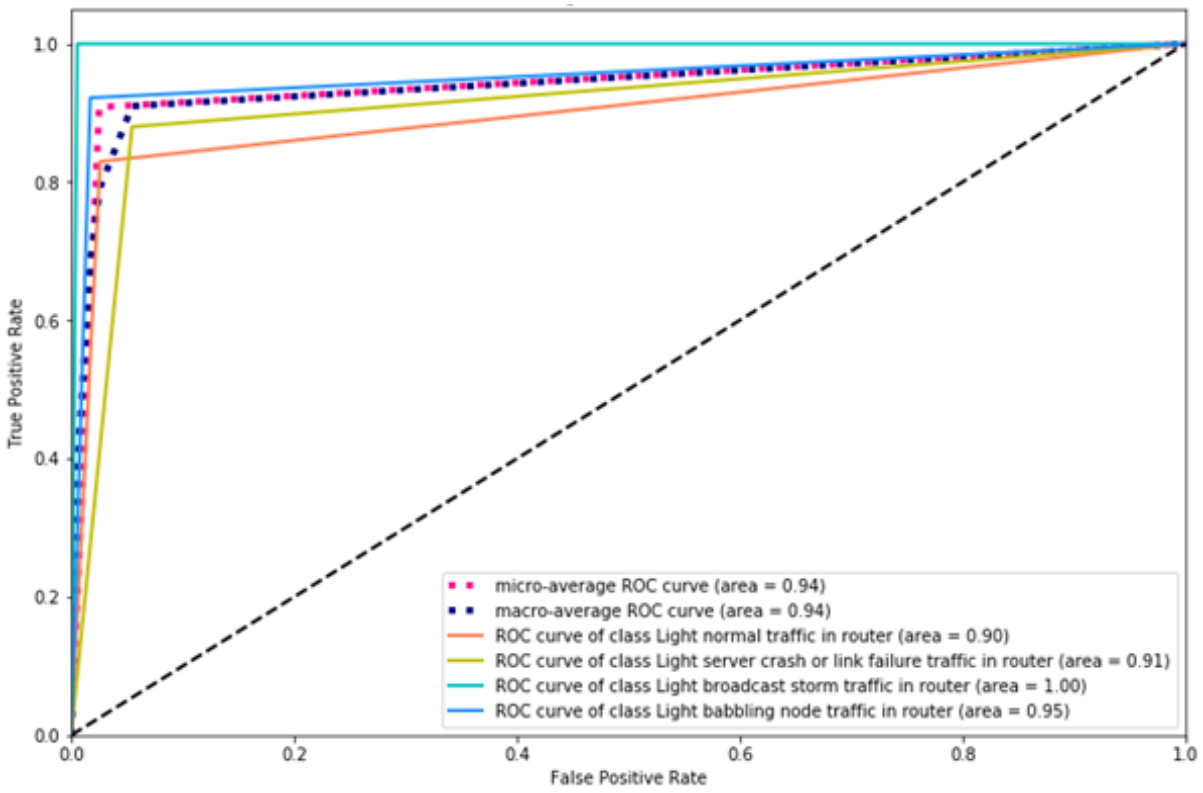

Fig. 5.8 The ROC to the light router workloads

are classified into four different classes. As it can be seen clearly on the graph, broadcast storm class maximised the true positive rate to the highest rate and scored 1.00 while the false positive rate for this class is minimised to zero. In this case, all classes for the broadcast storm were correctly predicted. As we see on the graph, the goodness of classifying this type of traffic can be qualified by looking at how much area there is underneath the curve. Where the area under the random dotted classifier line is regularly 0.5, then the area above the random classifier dotted line, in this case, approaches the top left corner. The area underneath the curve becomes larger and larger, approaching 1.0, which shows that the classifier achieved 100% accuracy in classifying this fault.

The experiment results revealed that the other three classes of network traffic, normal, server crash or the link failure and the babbling node, reached an excellent area of performance and obtained accuracy of 90%, 91% and 95%, respectively. Therefore, the accuracy of such degrees represents that the proposed classifier predicted most of the classes correctly. As the AUC is maximised almost to its highest rate for all four individual classes, this is an indication of the proposed classifier capability in absolutely performing classifying the faults under the light scenario where the attack source was the router.

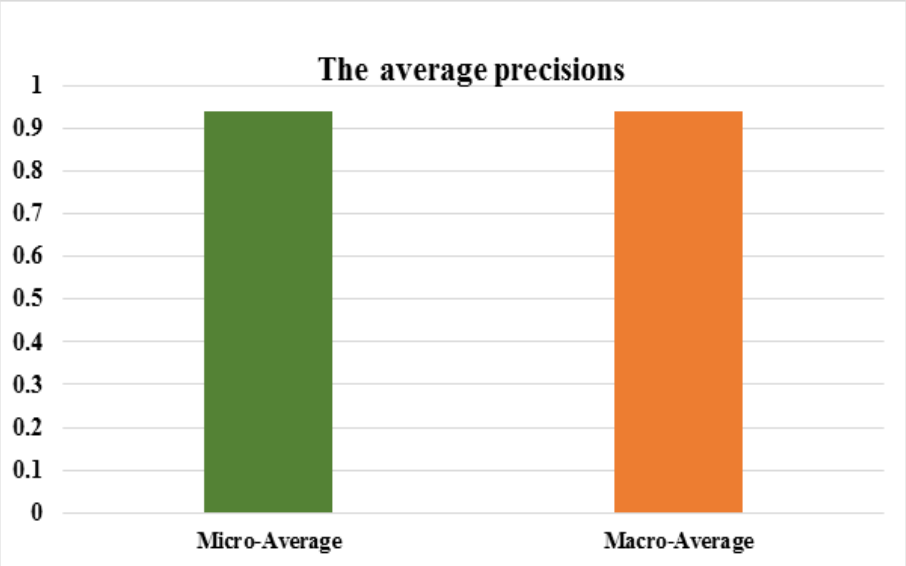

Fig. 5.9 The proposed method performance in classifying the light traffic of the router

By looking at the graph, we can clearly see that the micro-average and macro-average precisions gained the same scale for this case. Although we previously highlighted that there is a slight difference between micro-average and macro-average precision, since the classes have about the same number of instances and the same weight in the classification process, macro and the micro-average will be equivalent, as disclosed in this example, and both achieved 94%, see Figure 5.9. For each specific case, we calculated independently how the proposed classifier would have behaved regarding the accuracy and the performance of the classifier, as depicted in Figure 5.9.

The graph reveals that the proposed classifier performance accomplished 94% accuracy when applied in classifying the faults of light traffic loads of the router. The classifier performance is one of the vital factor considerations in this research to assess how the classifier would act when applied to the datasets with distinct scenarios and devices, and, further, to make a comparison of the performance between the proposed method and the existing classifier.

5.4.2 SFPNNC experiment results for light scenario of server traffic faults

This section shows the classification results of four different classes where the server is being attacked by distinct light network traffic. To explore how the classes predicted and to compute the total true positive rate and the false positive rate, the ROC curve is drawn

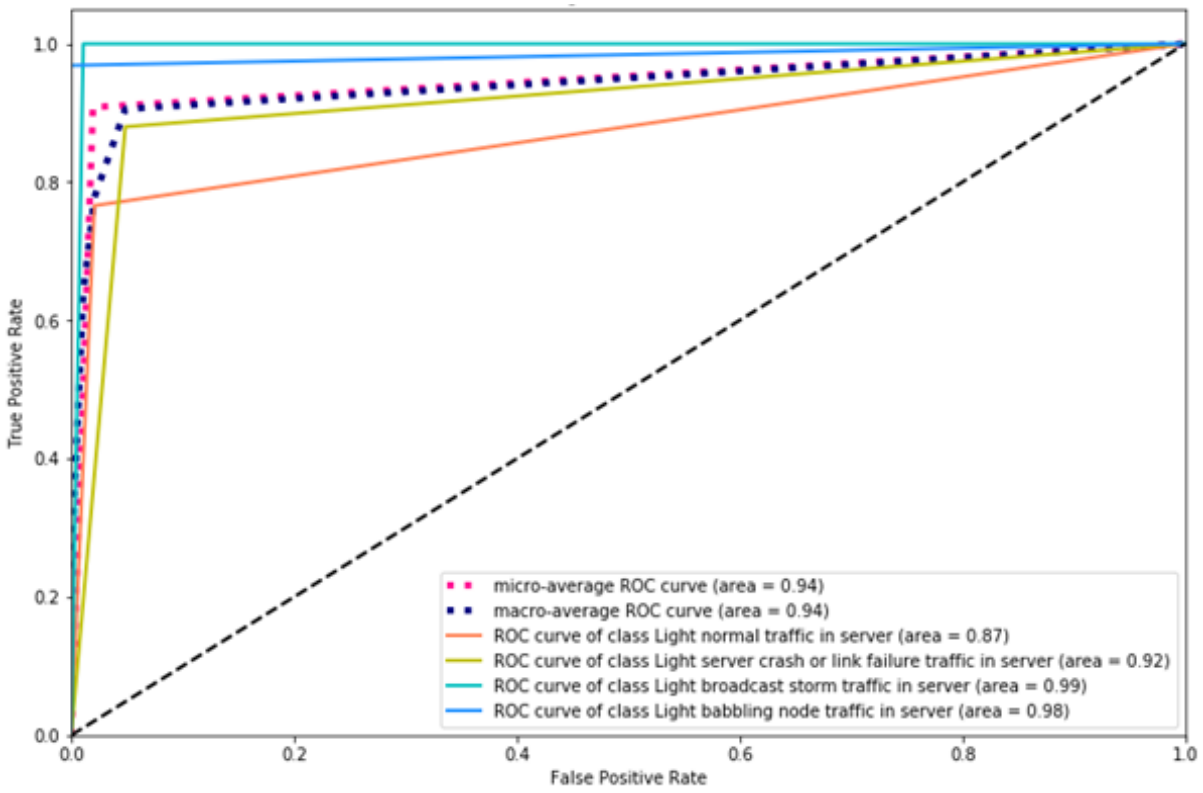

Fig. 5.10 The ROC to the light server workloads

and reveals how the proposed classifier behaved once applied to such dataset. Figure 5.12 shows the results of evaluations of the light traffic that collected in a server and classifies them into four different classes, as depicted on the graph, as to normal, server crash or link failure, broadcast storm and the babbling node classes.

As indicated on the graph, the proposed classifier almost totally classified the broadcast storm and the babbling node traffic, wherein the accuracy reached to 99% and 98%, respectively. For these two classes, the error rate is minimal, which means that, out of 100 classes for both, it correctly predicted 99 classes for the broadcast storm class and 98 for the babbling node class. Providentially, the server crash is classified with a precise ratio, which is most of the classes diagnosed as the actual classes, and the accuracy scored 92%, which is considered as an excellent estimation. Compared to the light traffic in a router, in the accuracy of the normal traffic class there is a slight downward turn approaching 87%. Overall, the proposed classifier (SFPNNC) classified the faults and traffics for this particular case outstandingly. According to these two cases, where the traffic was collected in a light scenario wherein the targeted allocated resource was a server or a router, the proposed classifier performed an outstanding classification. Due to having the equivalent weight for the existing class in

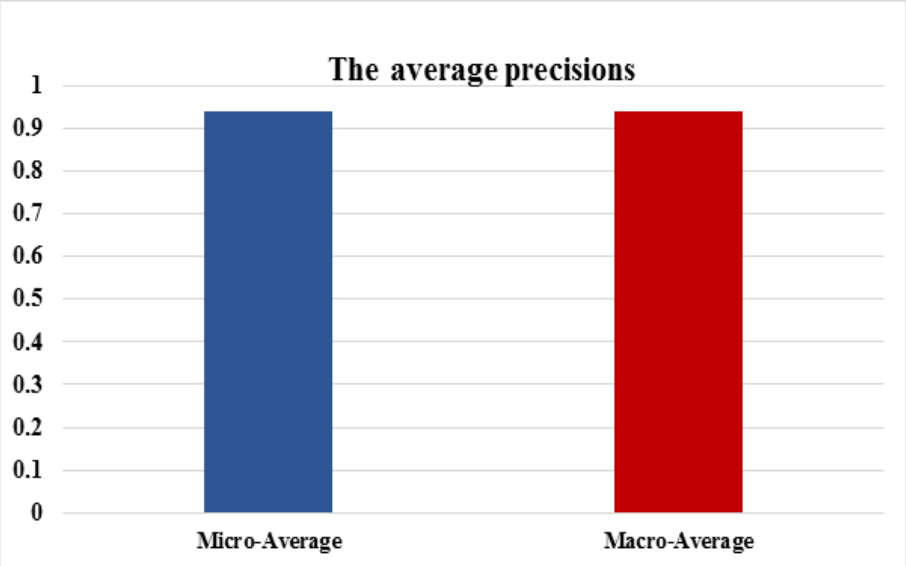

Fig. 5.11 The proposed method performance in classifying the light traffic of the server

the simulation, the macro-average and the micro-average precision lead to achieving the same average as the proposed classifier obtained for the previous case of light traffic in a router, which is 94%, as shown in Figure 5.11. This score of the precision proved that the classifier is outperforming the existing classifier, which will be evidenced and discussed in the following sections. The graph addresses this matter and reveals that the proposed method performance is repeatedly the same in obtaining 94% average precisions for both micro and macro in which the error rate is minimised to a very small ratio.

Repeatedly, this confirms that the research is grounded on the right path in combining of different machine learning methods based on using supervised and unsupervised techniques to develop a new proposed method that classifies the occurrence faults precisely when the traffic workloads are collected in the light scenario.

5.4.3 SFPNNC experiment results for heavy scenario of router traffic faults

In this case, the classifier model has been applied to a different type of traffic in a router. The condition of the traffic has been changed from light to heavy by increasing the size of packets and bandwidth. The dataset of the IF group is used, which involves 12 MIB variables. The experimental results revealed that the proposed model diagnosed all the faults and classified them clearly, as shown in Figure 5.12. As we can see from the graph, the performance of the individual classes is lowered when compared to the same faults in the

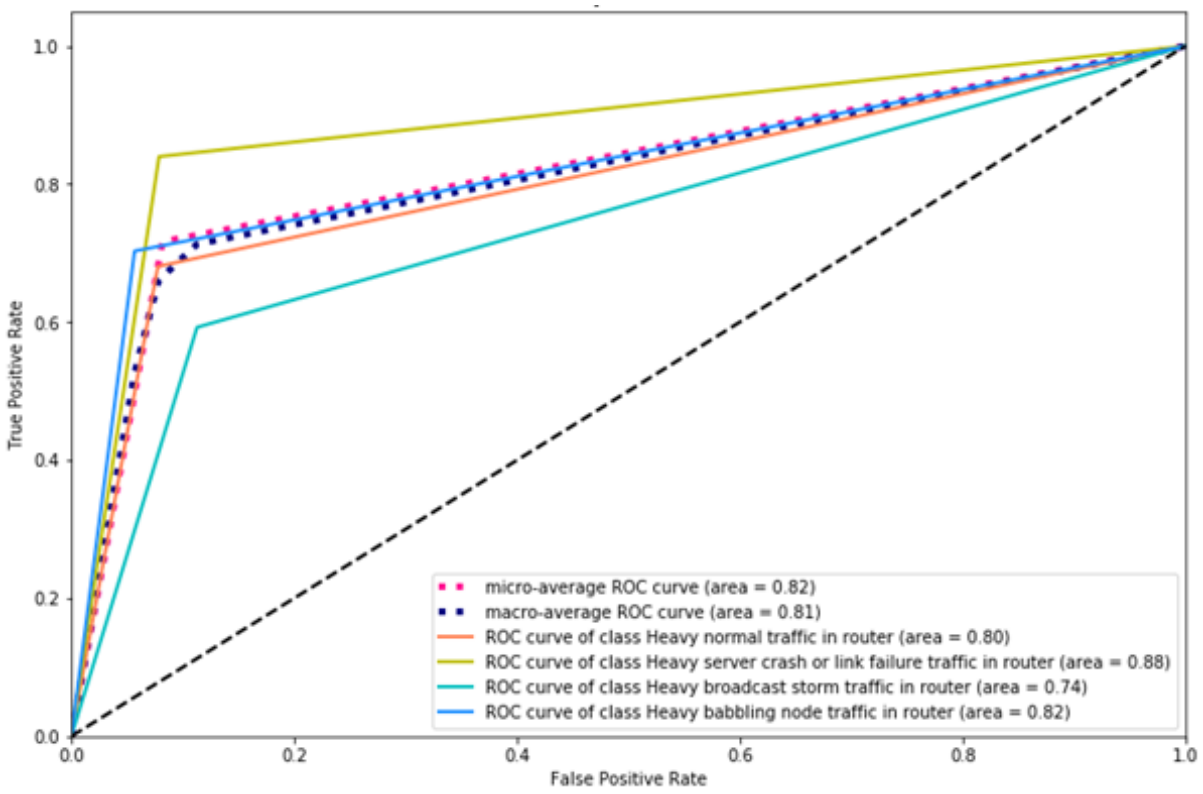

Fig. 5.12 The ROC curves to the heavy router workloads

light scenario. The reason behind decreasing the accuracy of the heavy datasets refers to the existing complications in the original data. As the researcher highlighted that the main sensitive parameters of the IF-MIB were not detected properly as discovered outstandingly in the light condition. Nonetheless, the classification results, besides acquiring an adequate performance rate, still needs more improvement, particularly toward the class of broadcast storm traffic in which there was a high proportion of misleading and the accuracy scored 74%. Thus, the broadcast storm class maximised the area under ROC and, conversely, minimised the precision rate.

The graph demonstrates that the normal and babbling node traffic accomplished the accuracy with 80% and 82%, respectively, wherein both classes have a slight difference in diagnosing the two different traffic. The results confirmed that the class of server crash or link failure is identified more precisely compared to the other three classes and the accuracy approached 88%. Concerning the micro-average and the macro-average precision, as depicted in Figure 5.15, these scored 82% and 81% precision, respectively. The graph shows the performance of the proposed classifier (SFPNNC) when applied to heavy workloads and demonstrates that the classifier performance is moderately performed based on the IF-MIB

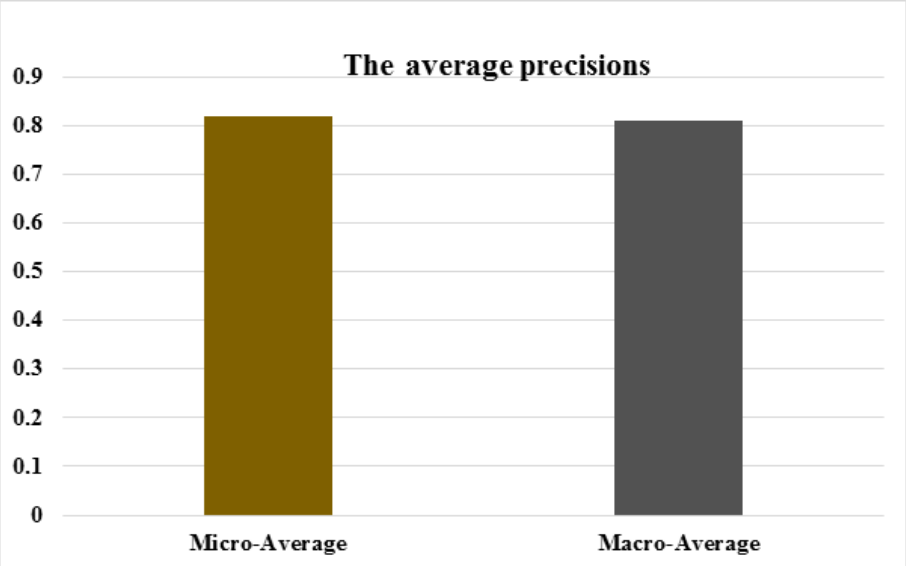

Fig. 5.13 The proposed method performance in classifying the heavy traffic of the router

variables for the router. As is stated in the previous section of this chapter, according to the features involvement, the dataset is considered as imbalanced data, while, according to the class weights, it is almost balanced; therefore, in this case we see that macro-average precision is slightly lower than the micro-average precision. The problem of the diminishing precision of this case is considered as one of the concerns in the research and will be optimised further.

5.4.4 SFPNN experiment results for heavy scenario of server traffic faults

The classifier is applied to a dataset that was collected in the server, where the scenario is switched to heavy, although the features are still the same, which consist of six IF-MIB variables, as highlighted in Table 4.2, Chapter 4. The classifier is relatively executed to classify the traffic as it operated in the heavy scenario for the router. We have found that the heavy scenario in either a router or a server increases the misleading if compared for both resources in the light scenario, the main reasons that caused this is highlighted in section 5.4.3. Each class of the traffic achieved a different accuracy, as shown in Figure 5.14. The broadcast storm class obtained the minimum accuracy and scored 73%. In our research, we have determined that the distinct condition of the data environment will directly influence the classification accuracy of the traffic. The normal traffic class is insignificantly higher than the broadcast storm class and approached 76% accuracy.

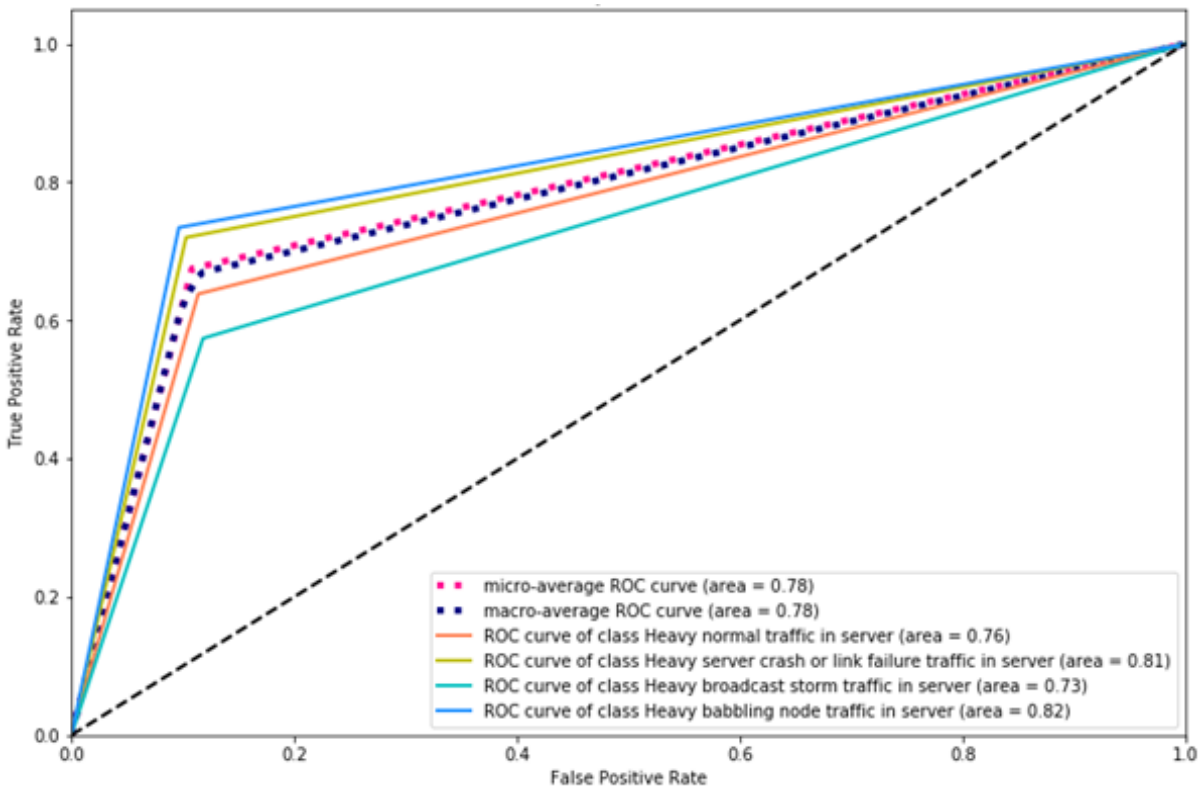

Fig. 5.14 The ROC curves to the heavy server workloads

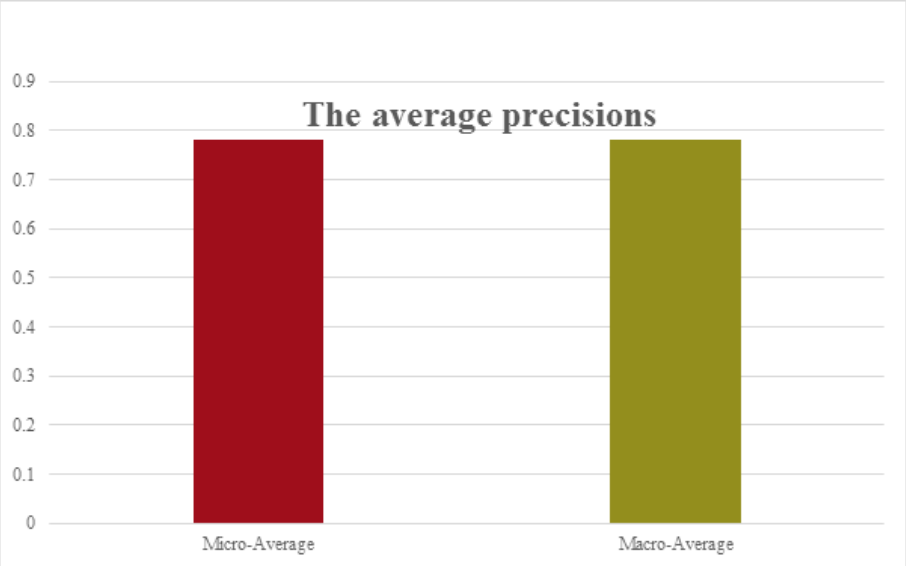

Fig. 5.15 The proposed method performance in classifying the heavy traffics of the server

Further, the accuracy of the server crash and babbling node classes reached to 81% and 82%, respectively, and achieved a better accuracy than the other two classes. The average precisions for both micro-average and macro-average are the same, scoring 78%, as shown in Figure 5.15, due to the classes' weight having equal distribution.

5.5 Comparative Analysis of the Existing Classifier (PNN) with the Proposed Method (SFPNNC) for all Datasets

Figure 5.16 illustrates the use of Receiver Operating Characteristics (ROC) metrics to quantify the PNN classifier's efficiency in distinguishing 16 different activities, which are four different workloads of the normal and abnormal traffic (light workloads in the router, light workloads in the server, heavy workloads in the router and heavy workloads in the server).

The samples of the data, which comprise the traffic from various scenarios and resources (server and router), are delivered to the PNN classifier, which leads to misclassification because the data are not passing through the clustering algorithm. Thus, the combination of our hybrid methods (SFPNNC), which includes supervised and unsupervised methods, indicates the high precision and accuracy of the detection or classification. The results showed that, in some cases, the PNN classifier lowered the classification to below the random classification line, such the heavy server crash or link failure traffic, in which the accuracy

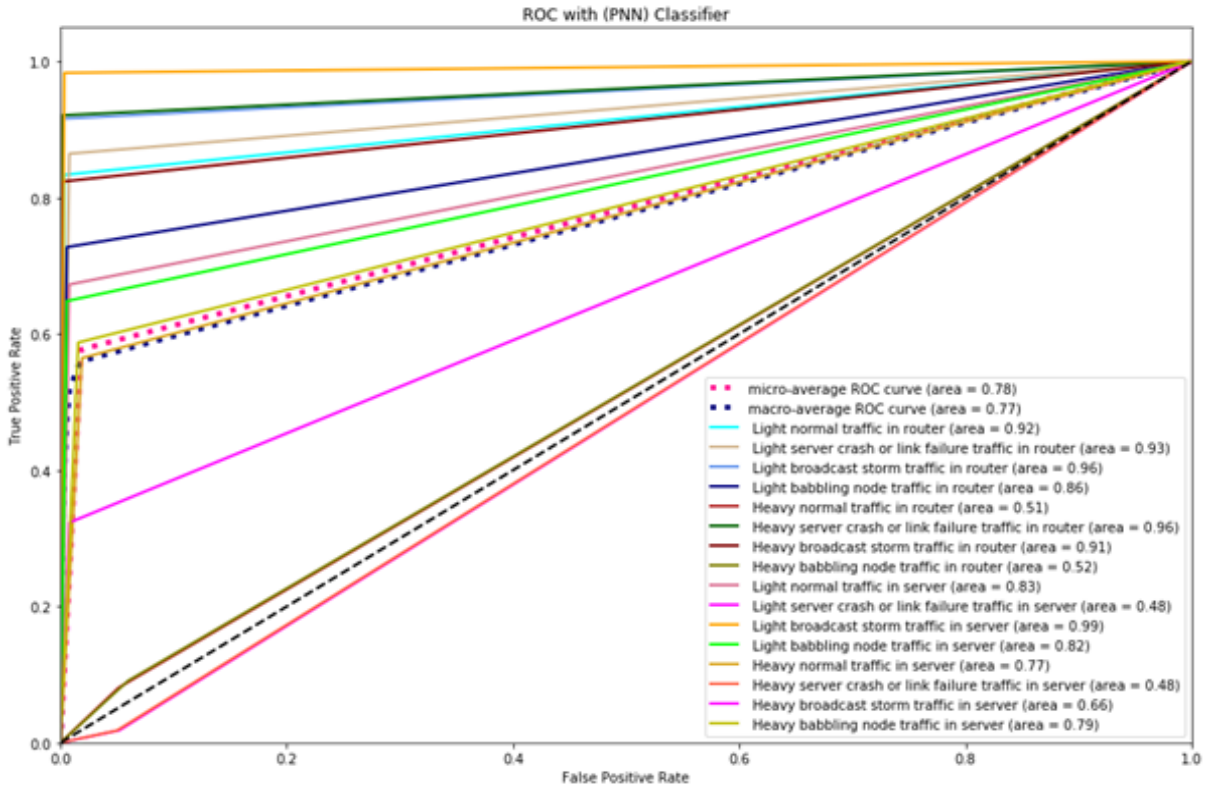

Fig. 5.16 The ROC curves to the various workloads of the PNN

reached 48%, while the minimum scored class accuracy of our proposed model stopped at 73% of the accuracy.

Table 5.3 depicts the achieved accuracy for each particular traffic considering their condition and resources based on our proposed model (SFPNNC) and the existing classifier (PNN). The table shows the leverage of our hybrid proposed method and makes a clear comparison of each particular traffic of the two different models.

Apparently, the results proved that the heavy condition affected the accuracy of both models, while its influences get enormously worse as repeatedly noticed that the accuracy dropped very sharply to 51%, 52%, and 48% for the normal router workload, babbling node traffic and server crash faults in server respectively. Merely in one of the event, the light traffic accuracy based on the proposed model approached to 87%, and all other classes achieved the accuracy above 90%, it is discerned that the PNN classifier accuracy is descending below 80% as demonstrated in table 5.3. The results in Table 5.4 demonstrate the efficiency of the proposed hybrid algorithm SFPNNCS to a build a classifier which very precisely classifies the network traffics in light scenario while the accuracy is adequate for the heavy scenario traffics. The research proved that the hybrid method achieved the average precisions for

Table 5.3 Individual class performance for SFPNNC and PNN classifiers

The proposed Model SFPNNC				PNN classifier			
Traffic types	**AUC score**	**Scenario**	**Allocated Resource**	**Traffic types**	**AUC score**	**Scenario**	**Allocated Resource**
Normal	0.9	Light	Router	Normal	0.93	Light	Router
Server crash or Link Failure	0.91	Light	Router	Server crash or Link Failure	0.96	Light	Router
Broadcast Storm	1	Light	Router	Broadcast Storm	0.86	Light	Router
Babbling Node	0.95	Light	Router	Babbling Node	0.51	Light	Router
Normal	0.87	Light	Server	Normal	0.83	Light	Server
Server crash or Link Failure	0.92	Light	Server	Server crash or Link Failure	0.48	Light	Server
Broadcast Storm	0.99	Light	Server	Broadcast Storm	0.99	Light	Server
Babbling Node	0.98	Light	Server	Babbling Node	0.82	Light	Server
Normal	0.8	Heavy	Router	Normal	0.51	Heavy	Router
Server crash or Link Failure	0.88	Heavy	Router	Server crash or Link Failure	0.96	Heavy	Router
Broadcast Storm	0.74	Heavy	Router	Broadcast Storm	0.91	Heavy	Router
Babbling Node	0.82	Heavy	Router	Babbling Node	0.52	Heavy	Router
Normal	0.76	Heavy	Server	Normal	0.77	Heavy	Server
Server crash or Link Failure	0.81	Heavy	Server	Server crash or Link Failure	0.48	Heavy	Server
Broadcast Storm	0.73	Heavy	Server	Broadcast Storm	0.66	Heavy	Server
Babbling Node	0.82	Heavy	Server	Babbling Node	0.79	Heavy	Server

Table 5.4 Classifiers performance comparison

Classifier	Scenario	Device	Micro-Average Precision	Macro-Average Precision
SFPNNC	Light	Router	0.94	0.94
		Server	0.94	0.94
	Heavy	Router	0.82	0.81
		Server	0.78	0.78
PNN	Light + Heavy	Router + Server	0.78	0.77

micro and macro of 94% in light conditions, while to the micro-average and macro-average of the accuracy approached 82% and 81% respectively for traffics detected in the router in the heavy scenario. The average precisions for the micro and macro are descending to 78% where the traffics detected and classified in the server for the heavy condition. Thus the hybrid proposed method improved the accuracy of the results in three cases out of four, while still for the last status obtained imperceptibly better precision for the macro-average than PNN classifier which is gained 77% and got the same average precision of the micro-average of 78% as depicted in table 5.4.

5.6 Chapter Summary

In this chapter, we have shown that the FCM method achieved optimal clusters due to its observing very intensively to indicate the membership for each data point. The PC metric is used and showed that the FCM got a remarkably higher thanK-means and EM. Next, the new algorithm (subtractive clustering) is combined with the FCM to develop a hybrid SFCM, that improved the performance of the FCM to compete with the best performance classifier which is K-means and achieved considerably better performance than EM. Further, the proposed classifier applied to the datasets, and the ROC curve is used to evaluate each distinct

class. The experiment results confirmed that the proposed method (SFPNNC) classified each different very accurate meanwhile compared to the existing classifier (PNN). The performance of each classifier is computed based on using the micro-average precision and the macro-average precision as the two common metrics. The results of the performance demonstrated that the proposed hybrid method achieved a distinguished higher precision in three different cases out of four than a PNN.

Chapter 6

Conclusions And Future Works

This Chapter presents the conclusions drawn from the utilised methods of the detection and classification system in real-time and demonstrates the potential future work.

6.1 Conclusions and Research Summary

During the past decade, the world has experienced expeditious advancements in various types of networking applications, while network domains have increasingly developed in terms of their degree of heterogeneity, complexity and magnitude. Various barriers including the level of access, flexibility and inadequate scalability have impacted the extant centralised network management mechanisms as the distribution of networks increases and the likelihood of network errors is augmented. The process of administrating manually has become significantly dated and it is imperative that effective management is applied to enable the provision and sustainability of superior quality service within computer networks. It is possible to offer service guarantees with increased rates of reliability and availability for applications that run in real time through a systematic approach for the real-time classification of network errors, which facilitates highly informed (frequently automated) decision-making processes.

This research contributes to a number of the most significant novelties related to classifying traffic within computer network systems. To prepare a cleansed datasets free from any noises and inconsistencies, we created two filtering - normalisation and standard scaler - models refer to see appendix C and D for more details, which is applied to the datasets very efficiently and produced a proper data for the classifier as demonstrated in section 4.3 in chapter 4. As emphasised in the project limitations, it was challenging to obtain a sufficient

volume of data that had not been cleansed to facilitate the analysis; however, the data filtering process ultimate enabled the work simulations to accomplish the required outcomes.

Further, the FCM is utilised to break down the complex data into segments, the experiment results shown in figure 5.2, 5.3 and 5.4 in chapter 5 proved that the FCM (Fuzzy Cluster) achieved more efficient cluster results than K-means (Hard Cluster) and EM (Hard and soft Clusters).

We improved the FCM algorithm and developed Subtractive FCM (SFCM) by combining it with the Subtractive Cluster to enhance the performance of the FCM algorithm, and this practically approved as shown in figure 5.1 in chapter 5. In order to resolve the constraints of FCM in regard to determining the number of starting clusters and the cluster centres, a subtractive clustering algorithm is introduced into the FCM, which directly augments and optimises the cluster outcomes.

The research work is extended, and consequently, a new proposed classifier model developed, we named Subtractive Fuzzy Probabilistic Neural Network Classifier (SFPNNC) to create a hybrid classifier method employing the existing classifier PNN with the SFCM. The obtained results confirmed that the proposed method achieved an outstanding accuracy when compared to the existing classifier PNN, refer to section 5.5 in chapter 5.

In this research, we have presented a methodology for a computer network traffic based on IF-MIB data by applying machine learning algorithms. The goal of the study is to prove the ability and the effectiveness of the proposed method in classifying the network traffic including normal and abnormal traffics of the most common and modern faults that can occur such as Server crash or link failure, broadcast storm and babbling node.

The methodology of the research involved three different treads including (preprocessing, data clustering and faults classification) in proceeding the classification of the computer network traffics that starts with the refining (Preprocess) the existing datasets to prepare them for the clusterisation with hybrid (SFCM clustering) model and forwarding the obtained outputs toward the classifier (PNN Classifier).

The proposed classification algorithm is implemented for each group of the data based on their different scenarios (Light and Heavy) and allocated resources (Server and Router), in order to manifest how the classifier performance and accuracy of each class is affected and to discover which groups of the traffic are the most efficient groups in the classification. The FCM algorithm is used as one of the efficient unsupervised methods to split any complex or big data and group them into distinct clusters. From the results, we found that the performance of this algorithm based on the fuzzy partition coefficient (FPC) approached to (0.96) while with same metric K-means and EM got 0.67 and 0.73 respectively, when the data classified

into four groups and reached to its optimality. The subtractive clustering is injected with the FCM to optimise the model in the initial cluster and cluster centres identification when the algorithm starts the simulation. Afterwards, we computed the classification accuracy of each specific classes within based on the proposed classifier method (SFPPNC). In addition, the average precisions are calculated using the micro-average precision and the macro-average precisions. We found that the performance of the proposed classifier for IF-MIB data for each scenario is different, where the accuracy rate varied between high and low depending on the scenario circumstances and the type of resources used. Using the ROC curve, we noticed that the average precision of (94%) achieved for both of the light scenarios of the router and the server.

It was found that in certain cases, particularly when the scenario was modified to heavy traffic which caused by the limitations in the orginal data as was explained in section 5.4.4 and 5.4.3in Chapter 5, the performance was reduced and the research objectives were not satisfied; nevertheless, these findings will be taken into consideration for future research. From the results, we discovered that the average precision was affected and lowered when the scenario changed to heavy. The average precision for the micro and the macro dropped to (0.82) and (0.81) respectively for the traffic in the router. The average precisions of the heavy workload traffic are more affected and reduced to (0.78), this refers to the existing problem in the origin data where the sensitive IF-MIB parameters were not detected correctly. The achieved results of the proposed method validated with an existing classifier (PNN) and the outcomes showed that the proposed method achieved an outstanding precision over the existing ones for the light workload traffics. The overall conclusion is that the proposed method (SFPNNC) using the IF-MIB datasets is an efficient approach to classify the light traffics and is moderately classifier for the heavy traffic data.

6.2 Summary of Research Questions

This part addresses the research questions presented in Section 1.2 and provides an explanation as to how they have been answered in the thesis.

1. How can network traffic in light and heavy loads be classified in real-time, while ensuring the designated device remain cost-effective (e.g., server, router and the network system's bandwidth)? Or, how can traffic faults identified and gathered in different situations be resolved?

   There are current methods that have already been implemented for the purposes of

classifying traffic, but it is evident that the majority have focused on different kinds of traffic to the present study and have not taken MIB variables into consideration. Furthermore, they have commonly only utilised traditional classifiers and not have not exploited the benefits of newly developed methods based on different machine learning algorithms. Hence, by using the benefits of other machine learning algorithms and incorporating them into the new technique, it was possible to increase the performance while maintaining cost effectiveness, as demonstrated in the experimental findings. This is an optimal method in scenarios where the system receives different features. Showing different output patterns revealed where the scenario captured and in what scenario this fault was created.

2. How can the traffic that emanates from different devices be characterised and is it heavy or light traffic?

   The MIB behaviour under both normal and abnormal conditions was investigated and it was observed that specific changes occur in different situations, leading to the utilisation of certain assigned resources. The easiest way of answering this question is by examining each of the existing features in the datasets and then determining the manner in which they are impacted by the process. It is verified whether the data is balanced or imbalanced. This can be achieved by visualising the features of all datasets.

3. How does clustering influence the classification outcomes?

   In the present study, the faults were grouped into different segments on the basis of the MIB variables that effect the individual situations and assigned resources. Moreover, clustering faults into different features of the vectors are valuable to handle the issues of the big data; it makes the model to be an application to improve the classifier performance where the big data is input to the model. We employed the cluster techniques in order to facilitate this procedure for the classifier. Thus, the utilisation of fuzzy clustering means enhances the classification procedure by accepting different patterns from the clustering outputs. The experimental findings indicated that when the entire data is entered into the classification model, it had a direct effect on the overall system and deteriorated the performance.

4. What effect does pre-processing have on the system performance?

   Different filtering were designed in order to refine the data and prepare it for input into the classification system. The new set of datasets composed after the preprocessing.

The cleansed datasets removed all redundant, inconsistent, and noises from the data. The procedure diminished the time consumed amount and enhanced the accuracy of the results as shown in the results discussion section 4.4 in chapter 4.

5. To what degree does the developed hybrid technique accomplish improved outcomes in comparison to existing classifiers? There are currently used methods that have already been implemented for the purposes of classifying traffic, but it is evident that the majority have focused on different kinds of traffic and have not taken MIB variables into consideration. Additionally, they often utilise existing classifiers without taking advantage of the newly developed techniques based on different machine learning algorithms.

   Machine learning techniques that are dependent on various statistical equations have been employed in the classification of all identified faults and any abnormalities that could materialise. Thus, based on the findings of this study, it can be concluded that the combination of different machine learning techniques, such as supervised and unsupervised in the suggested model, (SFPPNC) represents the perfect option for the real-time classification of traffic within computer networks.

   The experimental results showed that the proposed model improved the performance in extremely high precision for the light traffic loads, while it achieved competent results for the heavy workloads compared to the results that obtained from the existing classifier.

6.3 Limitations

This thesis has several limitations, including that the performance is impacted when the scenario is altered from light to heavy, which is achieved by enlarging the magnitude of the packets and the bandwidth. In a small number of instances, the performance was even impacted in light traffic conditions.

An additional research limitation is the inadequate number of datasets, which had a direct impact on the classification performance outcomes. There are insufficient available datasets to conduct additional tests to resolve the problems of reduced performance, particularly in the heavy load scenario.

Furthermore, there were only 12 MIB variables being utilised for the router and 6 MIB features for the server in the traffic datasets, where all only belonged to the IF group. However,

there are total of 178 MIB features associated with 12 different groups, and some have been reliable connected with particular faults.

An additional limitation is associated with the machine learning algorithms employed; although they certainly have some benefits, which are considered in this thesis, FCM is a time-consuming algorithm in which every data point is analysed in great detail in order to determine the ideal solution for that data point. An FCM with a reduced value of β will generate improved outcomes, but will also lead to an increased number of iterations. In some circumstances, Euclidean distance measures can assign unequal weights to underlying factors.

Hence, although the PNN does have the aforementioned benefits, it also has certain limitations. For instance, it does not have the same level of efficiency as multilayer perceptron networks when classifying new scenarios. The PNN also necessitates extra memory space for model storage in addition to all the trained data sample. A representative training set is needed for every sample.

6.4 Future Work

Even though this thesis has made some important contributions in regard to the problem of classification and the SFPNNC-based ensemble, there are still possibilities for extending the work contained within the thesis. The specific areas that are recommended for future investigation include:

- Enhancing the model performance in the scenario where the data is impacted by the bandwidth and packet magnitude (i.e., when the scenario is changed from light to heavy conditions).

- This study concentrated on the combination of supervised and unsupervised classifiers within an ensemble. Classifiers could be beneficial for enhancing the general performance of the suggested ensemble architecture and thus, the general performance of the overall architecture.

  studies could investigate the utilisation of different algorithms to identify and forecast directly in real time when any errors materialise. One of the recently introduced algorithms is the Spike Neural Network, which could be analysed to determine whether its performance is similar to or exceeds the ensemble method.

- The suggested ensemble technique could be additionally tested on different datasets with numerous classes in order to assess its performance levels.
- Future studies can also incorporate the examination of additional MIB features from the 12 different groups as well as the parameters that are considered to influence the most frequently observed faults.
- The suggest model can be generalised to any fault that can occur in the computer network system in order to automatically observe and classify the faulty immediately. This application could thus be easily adapted to any network system.
- The suggested model could be developed into an application that is capable of detecting and classifying faults in real time. This would isolate and resolve the majority of faults associated with software problems, although it is not feasible for the resolution of any hardware problems.

References

Aggarwal, A., Malik, H., and Sharma, R. (2016). Feature extraction using emd and classification through probabilistic neural network for fault diagnosis of transmission line. In *Power Electronics, Intelligent Control and Energy Systems (ICPEICES), IEEE International Conference on*, pages 1–6. IEEE.

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998). *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM.

Al-Kasassbeh, M. (2011a). Change detection methods for computer network problems. *World Applied Sciences Journal*, 13(11):2364–2371.

Al-Kasassbeh, M. (2011b). Network intrusion detection with wiener filter-based agent. *World Appl. Sci. J*, 13(11):2372–2384.

Al-Kasassbeh, M. and Adda, M. (2008). Analysis of mobile agents in network fault management. *Journal of Network and Computer Applications*, 31(4):699–711.

Al-Kasassbeh, M. and Adda, M. (2009). Network fault detection with wiener filter-based agent. *Journal of Network and Computer Applications*, 32(4):824–833.

Amini, A., Wah, T. Y., Saybani, M. R., and Yazdi, S. R. A. S. (2011). A study of density-grid based clustering algorithms on data streams. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, volume 3, pages 1652–1656. IEEE.

Anava, O. and Levy, K. (2016). k*-nearest neighbors: From global to local. In *Advances in Neural Information Processing Systems*, pages 4916–4924.

Angjeli, A., Cheng, E., and Lipták, L. (2013). Linearly many faults in dual-cube-like networks. *Theoretical Computer Science*, 472:1–8.

Ashari, A., Paryudi, I., and Tjoa, A. M. (2013). Performance comparison between naïve bayes, decision tree and k-nearest neighbor in searching alternative design in an energy simulation tool. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 4(11).

Auld, T., Moore, A. W., and Gull, S. F. (2007). Bayesian neural networks for internet traffic classification. *IEEE Transactions on neural networks*, 18(1):223–239.

Äyrämö, S. and Kärkkäinen, T. (2006). Introduction to partitioning-based clustering methods with a robust example. *Reports of the Department of Mathematical Information Technology. Series C, Software engineering and computational intelligence*, (1/2006).

Barakat, M., Lefebvre, D., Khalil, M., Mustapha, O., and Druaux, F. (2010). Input—output classification mapping for the fault detection, identification and accommodation. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 403–410. IEEE.

Bathula, B. G., Sinha, R. K., Chiu, A. L., and Woodward, S. L. (2018). Routing and regenerator planning in a carrier's core reconfigurable optical network. *Journal of Optical Communications and Networking*, 10(2):A196–A205.

Benkaci, M., Doncescu, A., and Jammes, B. (2011). Fault detection and isolation using fuzzy-artmap classification and conflict intersection. In *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2011 International Conference on*, pages 441–446. IEEE.

Bezdek, J. C. (1981). Objective function clustering. In *Pattern recognition with fuzzy objective function algorithms*, pages 43–93. Springer.

Bhamre, T., Zhang, T., and Singer, A. (2016). Denoising and covariance estimation of single particle cryo-em images. *Journal of structural biology*, 195(1):72–81.

Bistouni, F. and Jahanshahi, M. (2015). Pars network: a multistage interconnection network with fault-tolerance capability. *Journal of Parallel and Distributed Computing*, 75:168–183.

Bolat, B. and Yildirim, T. (2003). Performance increasing methods for probabilistic neural networks. *Pakistan Journal of Information and Technology*, 2(3):250–255.

Bonner, R. E. (1964). On some clustering techniques. *IBM journal of research and development*, 8(1):22–32.

Brighton, H. and Mellish, C. (2002). Advances in instance selection for instance-based learning algorithms. *Data mining and knowledge discovery*, 6(2):153–172.

Cardente, J., Durazzo, K., and Harwood, J. (2017). Classification techniques to identify network entity types and determine network topologies. US Patent 9,621,431.

Carr, J. (1990). Network management. *Network Magazine*.

Chakrabarty, A., Buzzard, G. T., and Żak, S. H. (2017). Output-tracking quantized explicit nonlinear model predictive control using multiclass support vector machines. *IEEE Transactions on Industrial Electronics*, 64(5):4130–4138.

Chakrabarty, A. and Roy, S. (2018). An efficient context-aware agglomerative fuzzy clustering framework for plagiarism detection. *International Journal of Data Mining, Modelling and Management*, 10(2):188–208.

Chang, C.-I., Lin, N. P., and Jan, N.-Y. (2009). An axis-shifted grid-clustering algorithm. , 12(2):183–192.

Chang, C.-T., Lai, J. Z., and Jeng, M.-D. (2011). A fuzzy k-means clustering algorithm using cluster center displacement. *J. Inf. Sci. Eng.*, 27(3):995–1009.

Chen, M., Wang, N., Zhou, H., and Chen, Y. (2017). Fcm technique for efficient intrusion detection system for wireless networks in cloud environment. *Computers & Electrical Engineering*.

Cho, K.-J., Ahn, S.-J., and Chung, J.-W. (2003). A study on the classified model and the agent collaboration model for network configuration fault management. *Knowledge-Based Systems*, 16(4):177–190.

Comer, D. E. (2006). *Automated network management systems*. Prentice-Hall, Inc.

Dasgupta, A., Nath, S., and Das, A. (2012). Transmission line fault classification and location using wavelet entropy and neural network. *Electric Power Components and Systems*, 40(15):1676–1689.

De, T., Fraix Burnet, D., and Chattopadhyay, A. K. (2016). Clustering large number of extragalactic spectra of galaxies and quasars through canopies. *Communications in Statistics-Theory and Methods*, 45(9):2638–2653.

de Vargas, R. R. and Bedregal, B. R. (2011). Interval ckmeans: An algorithm for clustering symbolic data. In *Fuzzy Information Processing Society (NAFIPS), 2011 Annual Meeting of the North American*, pages 1–6. IEEE.

Dembele, D. and Kastner, P. (2003). Fuzzy c-means method for clustering microarray data. *bioinformatics*, 19(8):973–980.

Du, Z., Yu, W., Yu, Y., Chen, L., Ni, Y., Luo, J., and Yu, X. (2008). Fabric handle clusters based on fuzzy clustering algorithm. In *Fuzzy Systems and Knowledge Discovery, 2008. FSKD'08. Fifth International Conference on*, volume 3, pages 546–550. IEEE.

Durvy, M., Diot, C., Taft, N., and Thiran, P. (2003). Network availability based service differentiation. In *International Workshop on Quality of Service*, pages 305–325. Springer.

ElMadbouly, E., Abdalla, A., and ElBanby, G. M. (2010). Sensor fusion and sensor fault detection with fuzzy clustering. In *Computer Engineering and Systems (ICCES), 2010 International Conference on*, pages 265–268. IEEE.

Enrico, Z., Piero, B., and CRENGUTA, P. I. (2008). From fuzzy clustering to a fuzzy rule-based fault classification model [j]. *International Journal of Computational Intelligence Systems*, 1(1):60–76.

Ertöz, L., Steinbach, M., and Kumar, V. (2003). Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of the 2003 SIAM international conference on data mining*, pages 47–58. SIAM.

Ferrari, A., Lombardi, S., and Signoroni, A. (2017). Bacterial colony counting with convolutional neural networks in digital microbiology imaging. *Pattern Recognition*, 61:629–640.

Flores-Martos, L., Gomez-Andrades, A., Barco, R., and Serrano, I. (2015). Unsupervised system for diagnosis in lte networks using bayesian networks. In *Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*, pages 1–5. IEEE.

Gallé, M. (2018). Method and system for assisting users in an automated decision-making environment. US Patent App. 15/407,782.

Gan, J. and Tao, Y. (2015). Dbscan revisited: mis-claim, un-fixability, and approximation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 519–530. ACM.

Gershman, S. J. and Daw, N. D. (2017). Reinforcement learning and episodic memory in humans and animals: an integrative framework. *Annual review of psychology*, 68:101–128.

Ghamisi, P. and Hofle, B. (2017). Lidar data classification using extinction profiles and a composite kernel support vector machine. *IEEE Geosci. Remote Sensing Lett.*, 14(5):659–663.

Ghosh, J. and Acharya, A. (2011). Cluster ensembles. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(4):305–315.

Gill, P., Jain, N., and Nagappan, N. (2011). Understanding network failures in data centers: measurement, analysis, and implications. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 350–361. ACM.

Goh, A. T. (2002). Probabilistic neural network for evaluating seismic liquefaction potential. *Canadian Geotechnical Journal*, 39(1):219–232.

Goudar, V. and Buonomano, D. V. (2018). Encoding sensory and motor patterns as time-invariant trajectories in recurrent neural networks. *Elife*, 7.

Guralnik, V. and Foslien, W. K. (2009). Automatic fault classification for model-based process monitoring. US Patent 7,533,070.

Hahsler, M., Piekenbrock, M., Arya, S., and Mount, D. (2017). dbscan: Density based clustering of applications with noise (dbscan) and related algorithms. *R package version*, pages 1–0.

Han, J., Kamber, M., and Tung, A. K. (2001). Spatial clustering methods in data mining. *Geographic data mining and knowledge discovery*, pages 188–217.

Harmouche, J., Delpha, C., and Diallo, D. (2012). Faults diagnosis and detection using principal component analysis and kullback-leibler divergence. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, pages 3907–3912. IEEE.

Hasnat, M. A., Velcin, J., Bonnevay, S., and Jacques, J. (2015). Simultaneous clustering and model selection for multinomial distribution: A comparative study. In *International Symposium on Intelligent Data Analysis*, pages 120–131. Springer.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer.

He, Y., Tan, H., Luo, W., Mao, H., Ma, D., Feng, S., and Fan, J. (2011). Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 473–480. IEEE.

Ho, T. K., Hull, J. J., and Srihari, S. N. (1994). Decision combination in multiple classifier systems. *IEEE transactions on pattern analysis and machine intelligence*, 16(1):66–75.

Hoang, D. C., Yadav, P., Kumar, R., and Panda, S. K. (2014). Real-time implementation of a harmony search algorithm-based clustering protocol for energy-efficient wireless sensor networks. *IEEE transactions on industrial informatics*, 10(1):774–783.

Hoffer, E. and Ailon, N. (2016). Semi-supervised deep learning by metric embedding. *arXiv preprint arXiv:1611.01449*.

Honda, K., Nakao, S., Notsu, A., and Ichihashi, H. (2012). Alternative fuzzy c-lines and comparison with noise clustering in cluster validation. In *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, pages 1–6. IEEE.

Hung, M.-C., Wu, J., Chang, J.-H., and Yang, D.-L. (2005). An efficient k-means clustering algorithm using simple partitioning. *journal of Information Science and Engineering*, 21(6):1157–1177.

Irfan, D., Xu, X., Deng, S., He, Z., and Ye, Y. (2009). S-canopy: A feature-based clustering algorithm for supplier categorization. In *Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on*, pages 677–681. IEEE.

Jaudet, M., Hussain, A., Sharif, K., et al. (2005). Temporal classification for fault-prediction in a real-world telecommunications network. In *Emerging Technologies, 2005. Proceedings of the IEEE Symposium on*, pages 209–214. IEEE.

Jhawar, R. and Piuri, V. (2017). Fault tolerance and resilience in cloud computing environments. In *Computer and Information Security Handbook (Third Edition)*, pages 165–181. Elsevier.

Joachims, T. (2001). Estimating the generalization performance of a svm efficiently. Technical report, Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

Joshi, A. and Kaur, R. (2013). A review: Comparative study of various clustering techniques in data mining. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(3).

Kasim, A. A., Wardoyo, R., and Harjoko, A. (2015). Fuzzy c means for image batik clustering based on spatial features. *International Journal of Computer Applications*, 117(2).

Kau, L.-J., Lin, Y.-P., and Lin, C.-T. (2006). Lossless image coding using adaptive, switching algorithm with automatic fuzzy context modelling. *IEE Proceedings-Vision, Image and Signal Processing*, 153(5):684–694.

Kaur, S. et al. (2016). Survey of different data clustering algorithms. *International Journal of Computer Science and Mobile Computing*, 5(5):584–588.

Khalid, H. M., Khoukhi, A., and Al-Sunni, F. M. (2011). Fault detection and classification using kalman filter and genetic neuro-fuzzy systems. In *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society, El Paso, TX, USA*, pages 18–20.

Khan, K., Rehman, S. U., Aziz, K., Fong, S., and Sarasvady, S. (2014). Dbscan: Past, present and future. In *Applications of Digital Information and Web Technologies (ICADIWT), 2014 Fifth International Conference on the*, pages 232–238. IEEE.

Khokhar, S., Zin, A. A. M., Memon, A. P., and Mokhtar, A. S. (2017). A new optimal feature selection algorithm for classification of power quality disturbances using discrete wavelet transform and probabilistic neural network. *Measurement*, 95:246–259.

Kittler, J., Hatef, M., Duin, R. P., and Matas, J. (1998). On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, 20(3):226–239.

Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24.

Krueger, R., Vij, A., and Rashidi, T. H. (2018). A dirichlet process mixture model of discrete choice. *arXiv preprint arXiv:1801.06296*.

Kumar, A., Ingle, Y. S., Pande, A., and Dhule, P. (2014). Canopy clustering: A review on pre-clustering approach to k-means clustering. *IJIACS*, 3(5):22–9.

Leinwand, A. and Conroy, K. F. (1993). *Network management: a practical perspective*. Addison-Wesley.

Li, J., Gao, X.-B., and Jiao, L.-C. (2006). A new feature weighted fuzzy clustering algorithm. *Acta Electronica Sinica*, 34(1):89.

Li, P., Wang, B.-H., Sun, H., Gao, P., and Zhou, T. (2008). A limited resource model of fault-tolerant capability against cascading failure of complex network. *The European Physical Journal B*, 62(1):101–104.

Li, X., Uricchio, T., Ballan, L., Bertini, M., Snoek, C. G., and Bimbo, A. D. (2016). Socializing the semantic gap: A comparative survey on image tag assignment, refinement, and retrieval. *ACM Computing Surveys (CSUR)*, 49(1):14.

Liao, Y.-Y., Yeh, C.-K., Tsui, P.-H., and Chang, C.-C. (2009). Classification of benign and malignant breast tumors by the contour analysis and scatterers characterization. In *Ultrasonics Symposium (IUS), 2009 IEEE International*, pages 2488–2491. IEEE.

Lin, C. (2015). Machine learning in microsoft azure.

Liu, T., Lugosi, G., Neu, G., and Tao, D. (2017). Algorithmic stability and hypothesis complexity. *arXiv preprint arXiv:1702.08712*.

Loginov, E., Gomez, L. F., Chiang, N., Halder, A., Guggemos, N., Kresin, V. V., and Vilesov, A. F. (2011). Photoabsorption of ag n (n 6–6000) nanoclusters formed in helium droplets: Transition from compact to multicenter aggregation. *Physical review letters*, 106(23):233401.

Malik, H. and Mishra, S. (2014). Feature selection using rapidminer and classification through probabilistic neural network for fault diagnostics of power transformer. In *India Conference (INDICON), 2014 Annual IEEE*, pages 1–6. IEEE.

Masood, S. Z., Shu, G., Dehghan, A., and Ortiz, E. G. (2017). License plate detection and recognition using deeply learned convolutional neural networks. *arXiv preprint arXiv:1703.07330*.

McCallum, A., Nigam, K., and Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Mehdizadeh, E., Sadi-Nezhad, S., and Tavakkoli-Moghaddam, R. (2008). Optimization of fuzzy clustering criteria by a hybrid pso and fuzzy c-means clustering algorithm. *Iranian Journal of Fuzzy Systems*, 5(3):1–14.

Mohamed, A. A. and Basir, O. (2010). Fusion based approach for distributed alarm correlation in computer networks. In *Communication Software and Networks, 2010. ICCSN'10. Second International Conference on*, pages 318–324. IEEE.

Montavon, G. and Müller, K.-R. (2012). Deep boltzmann machines and the centering trick. In *Neural Networks: Tricks of the Trade*, pages 621–637. Springer.

Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60.

Moore, A. W. and Papagiannaki, K. (2005). Toward the accurate identification of network applications. In *International Workshop on Passive and Active Network Measurement*, pages 41–54. Springer.

Nagaraja, K., Krishnan, N., Bianchini, R., Martin, R. P., and Nguyen, T. D. (2003). Evaluating the impact of communication architecture on the performability of cluster-based services. In *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*, pages 229–240. IEEE.

Nakayama, Y., Yata, K., and Aoshima, M. (2017). Support vector machine and its bias correction in high-dimension, low-sample-size settings. *Journal of Statistical Planning and Inference*, 191:88–100.

Nandi, I., Srivastava, P. K., and Shah, K. (2017). Floodplain mapping through support vector machine and optical/infrared images from landsat 8 oli/tirs sensors: Case study from varanasi. *Water resources management*, 31(4):1157–1171.

Newton, S. C., Pemmaraju, S., and Mitra, S. (1992). Adaptive fuzzy leader clustering of complex data sets in pattern recognition. *IEEE Transactions on Neural Networks*, 3(5):794–800.

Nikhath, A. K., Subrahmanyam, K., and Vasavi, R. (2016). Building a k-nearest neighbor classifier for text categorization. *IJCSIT) International Journal of Computer Science and Information Technologies*, 7:254–256.

Oliveira, J. C. M., Pontes, K. V., Sartori, I., and Embiruçu, M. (2017). Fault detection and diagnosis in dynamic systems using weightless neural networks. *Expert Systems with Applications*, 84:200–219.

Palanikumar, R. and Ramasamy, K. (2018). Effective failure nodes detection using matrix calculus algorithm in wireless sensor networks. *Cluster Computing*, pages 1–10.

Paradis, L. and Han, Q. (2007). A survey of fault management in wireless sensor networks. *Journal of Network and systems management*, 15(2):171–190.

Park, D.-C. (2009). Classification of audio signals using fuzzy c-means with divergence-based kernel. *Pattern Recognition Letters*, 30(9):794–798.

Patwary, M. A., Palsetia, D., Agrawal, A., Liao, W.-k., Manne, F., and Choudhary, A. (2012). A new scalable parallel dbscan algorithm using the disjoint-set data structure. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 62. IEEE Computer Society Press.

Poria, S., Cambria, E., Bajpai, R., and Hussain, A. (2017). A review of affective computing: From unimodal analysis to multimodal fusion. *Information Fusion*, 37:98–125.

Qader, K. and Adda, M. (2014). Fault classification system for computer networks using fuzzy probabilistic neural network classifier (fpnnc). In *International Conference on Engineering Applications of Neural Networks*, pages 217–226. Springer.

Qader, K., Adda, M., and Al-Kasassbeh, M. (2017). Comparative analysis of clustering techniques in network traffic faults classification. *International Journal of Innovative Research in Computer and Communication Engineering*, 5(4):6551–6563.

Qiu, Q. and Sapiro, G. (2015). Learning transformations for clustering and classification. *The Journal of Machine Learning Research*, 16(1):187–225.

Rajini, N. H. and Bhavani, R. (2011). Enhancing k-means and kernelized fuzzy c-means clustering with cluster center initialization in segmenting mri brain images. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, volume 2, pages 259–263. IEEE.

Revett, K., Gorunescu, F., Gorunescu, M., El-Darzi, E., and Ene, M. (2005). A breast cancer diagnosis system: a combined approach using rough sets and probabilistic neural networks. In *Computer as a Tool, 2005. EUROCON 2005. The International Conference on*, volume 2, pages 1124–1127. IEEE.

Rong, C. et al. (2011). Using mahout for clustering wikipedia's latest articles: A comparison between k-means and fuzzy c-means in the cloud. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 565–569. IEEE.

Roy, A., Zeng, H., Bagga, J., and Snoeren, A. C. (2017). Passive realtime datacenter fault detection and localization. In *NSDI*, pages 595–612.

Rozaki, E. (2015). Network fault diagnosis using data mining classifiers. *Eleni Rozaki International Journal of Data Mining & Knowledge Management Process*.

Sanse, K. and Sharma, M. (2015). Clustering methods for big data analysis. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 4(3).

Saravanan, N. and Rathinam, A. (2012). A comparitive study on ann based fault location and classification technique for double circuit transmission line. In *Computational intelligence and communication networks (CICN), 2012 fourth international conference on*, pages 824–830. IEEE.

Shaffer, R. E. and Rose-Pehrsson, S. L. (1999). Improved probabilistic neural network algorithm for chemical sensor array pattern recognition. *Analytical chemistry*, 71(19):4263–4271.

Sharifi, R. and Langari, R. (2011). Isolability of faults in sensor fault diagnosis. *Mechanical systems and signal processing*, 25(7):2733–2744.

Siewiorek, D. and Swarz, R. (2017). *Reliable Computer Systems: Design and Evaluatuion*. Digital Press.

Singh, A., Fernando, A. E., and Leavline, E. J. (2016). Performance analysis on clustering approaches for gene expression data. *International Journal of Advanced Research in Computer and Communication Engineering*, 5(2):196–200.

Sivarathri, S. and Govardhan, A. (2014). Experiments on hypothesis" fuzzy k-means is better than k-means for clustering". *International Journal of Data Mining & Knowledge Management Process*, 4(5):21.

Specht, D. F. (1988). Probabilistic neural networks for classification, mapping, or associative memory. In *IEEE international conference on neural networks*, volume 1, pages 525–532.

Specht, D. F. (1990). Probabilistic neural networks. *Neural networks*, 3(1):109–118.

Sulaiman, S. N. and Isa, N. A. M. (2010). Adaptive fuzzy-k-means clustering algorithm for image segmentation. *IEEE Transactions on Consumer Electronics*, 56(4).

Svetlova, L., Mirkin, B., and Lei, H. (2013). Mfwk-means: Minkowski metric fuzzy weighted k-means for high dimensional data clustering. In *Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on*, pages 692–699. IEEE.

Swain, R. R. and Khilar, P. M. (2017). Composite fault diagnosis in wireless sensor networks using neural networks. *Wireless Personal Communications*, 95(3):2507–2548.

Tayeb, E. B. M. and Rhim, O. A. A. A. (2011). Transmission line faults detection, classification and location using artificial neural network. In *Utility Exhibition on Power and Energy Systems: Issues & Prospects for Asia (ICUE), 2011 International Conference and*, pages 1–5. IEEE.

Toussaint, G. T. and Berzan, C. (2012). Proximity-graph instance-based learning, support vector machines, and high dimensionality: An empirical comparison. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 222–236. Springer.

Tripathy, M., Maheshwari, R., and Verma, H. (2010). Improved transformer protection using probabilistic neural network and power differential method. *International Journal of Engineering, Science and Technology*, 2(3):29–44.

Tuan, N. M., Nhan, H. T., Chau, N. V. V., Hung, N. T., Tuan, H. M., Van Tram, T., Le Da Ha, N., Loi, P., Quang, H. K., Kien, D. T. H., et al. (2015). Sensitivity and specificity of a novel classifier for the early diagnosis of dengue. *PLoS neglected tropical diseases*, 9(4):e0003638.

Velmurugan, T. (2014). Performance based analysis between k-means and fuzzy c-means clustering algorithms for connection oriented telecommunication data. *Applied Soft Computing*, 19:134–146.

Wang, J., Wang, J., Ke, Q., Zeng, G., and Li, S. (2015). Fast approximate k-means via cluster closures. In *Multimedia data mining and analytics*, pages 373–395. Springer.

Wang, N., Aravinthan, V., and Ding, Y. (2014). Feeder-level fault detection and classification with multiple sensors: A smart grid scenario. In *Statistical Signal Processing (SSP), 2014 IEEE Workshop on*, pages 37–40. IEEE.

Wang, W., Yang, J., Muntz, R., et al. (1997). Sting: A statistical information grid approach to spatial data mining. In *VLDB*, volume 97, pages 186–195.

Wang, X., Liu, G., Li, J., and Nees, J. P. (2017). Locating structural centers: A density-based clustering method for community detection. *PloS one*, 12(1):e0169355.

Wang, Y., He, Q., Ye, D., and Yang, Y. (2018). Formulating criticality-based cost-effective fault tolerance strategies for multi-tenant service-based systems. *IEEE Transactions on Software Engineering*, 44(3):291–307.

Wang, Z., Jiang, H., and Xu, Y. (2008). Early fault classification identification and fault self-recovery on aero-engine. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pages 1935–1939. IEEE.

Watanabe, N. and Imaizumi, T. (2001). Fuzzy k-means clustering with crisp regions. In *Fuzzy Systems, 2001. The 10th IEEE International Conference on*, volume 1, pages 199–202. IEEE.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37.

Xu, L. and Chow, M.-Y. (2008). Power distribution system fault diagnosis using hybrid algorithm of fuzzy classification and artificial immune systems. In *Soft Computing Applications in Industry*, pages 357–372. Springer.

Xu, L., Krzyzak, A., and Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE transactions on systems, man, and cybernetics*, 22(3):418–435.

Yang, M.-S. (1993). A survey of fuzzy clustering. *Mathematical and Computer modelling*, 18(11):1–16.

Yang, Z. R., Zwolinski, M., Chalk, C. D., and Williams, A. C. (2000). Applying a robust heteroscedastic probabilistic neural network to analog fault detection and classification. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 19(1):142–151.

Yeo, S.-M., Kim, C.-H., Hong, K., Lim, Y., Aggarwal, R., Johns, A., and Choi, M. (2003). A novel algorithm for fault classification in transmission lines using a combined adaptive network and fuzzy inference system. *International journal of electrical power & energy systems*, 25(9):747–758.

Youssef, O. A. (2009). An optimised fault classification technique based on support-vector-machines. In *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES*, pages 1–8. IEEE.

Yu, H., Li, P., and Fan, Y. (2006). Sampling fuzzy k-means clustering algorithm based on clonal optimization. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 2, pages 6102–6105. IEEE.

Yu, Y., Congming, L., Tingyu, W., and Xing, Z. (2011). Fault diagnosis and classification for bearing based on emd-ica. In *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, volume 5, pages 2715–2718. IEEE.

Zadeh, L. A. (1965). Information and control. *Fuzzy sets*, 8(3):338–353.

Zhang, D., Yu, L., and Wang, Q.-G. (2011). Fault detection for network-based nonlinear systems with communication constraints and missing measurements. In *Control Conference (ASCC), 2011 8th Asian*, pages 181–186. IEEE.

Zhang, J., Pechenizkiy, M., Pei, Y., and Efremova, J. (2016a). A robust density-based clustering algorithm for multi-manifold structure. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 832–838. ACM.

Zhang, L., Lu, W., Liu, X., Pedrycz, W., and Zhong, C. (2016b). Fuzzy c-means clustering of incomplete data based on probabilistic information granules of missing values. *Knowledge-Based Systems*, 99:51–70.

Zhang, W., Li, C., Peng, G., Chen, Y., and Zhang, Z. (2018). A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load. *Mechanical Systems and Signal Processing*, 100:439–453.

Zhang, X. and Xu, Z. (2015). Hesitant fuzzy agglomerative hierarchical clustering algorithms. *International Journal of Systems Science*, 46(3):562–576.

Zhang, Y., Chen, J., Fang, Q., and Ye, Z. (2016c). Fault analysis and prediction of transmission line based on fuzzy k-nearest neighbor algorithm. In *ICNC-FSKD*, pages 894–899.

Zsebők, S., Herczeg, G., Blázi, G., Laczi, M., Nagy, G., Török, J., and Garamszegi, L. Z. (2018). Minimum spanning tree as a new, robust repertoire size comparison method: simulation and test on birdsong. *Behavioral Ecology and Sociobiology*, 72(3):48.

Appendix A

The Router Heavy workloads data visualisation

The Figure A.1,A.2, A.3 and A.4 Represents the features visualisation of the IF-MIB for the traffics that detected when the normal and abnormal traffics were captured in the router in a heavy condition. The graphs visualisation Depict the traffic behaviour when the situation changes to include heavy workloads. These changes are stimulated by adjusting the sizes of the packets and growing the bandwidth, which has an immediate influence on all of the features and also minimises the network performance.

According to the Figure A.1, A.2 and A.3, it is revealed that the behaviour of the features in the case of the server crash and broadcast storm compared to the normal traffic features are exclusively affected, and there is a high sensitivity connection between the parameters.

The main problem in this scenario is noticed that in the last fault when the babbling malfunction is injected into the event as shown in the Figure A.4, there is not any considerable sensitivity between the normal traffics features with the babbling node parameters. This case directly had impacted the classification performance results when the main sensitivity parameters were not detected.

The results showed that the same features in the normal case are copied into the babbling node data. Therefore, in this circumstance, the classifier performance is lowered when the scenario changed from the light to the heavy.

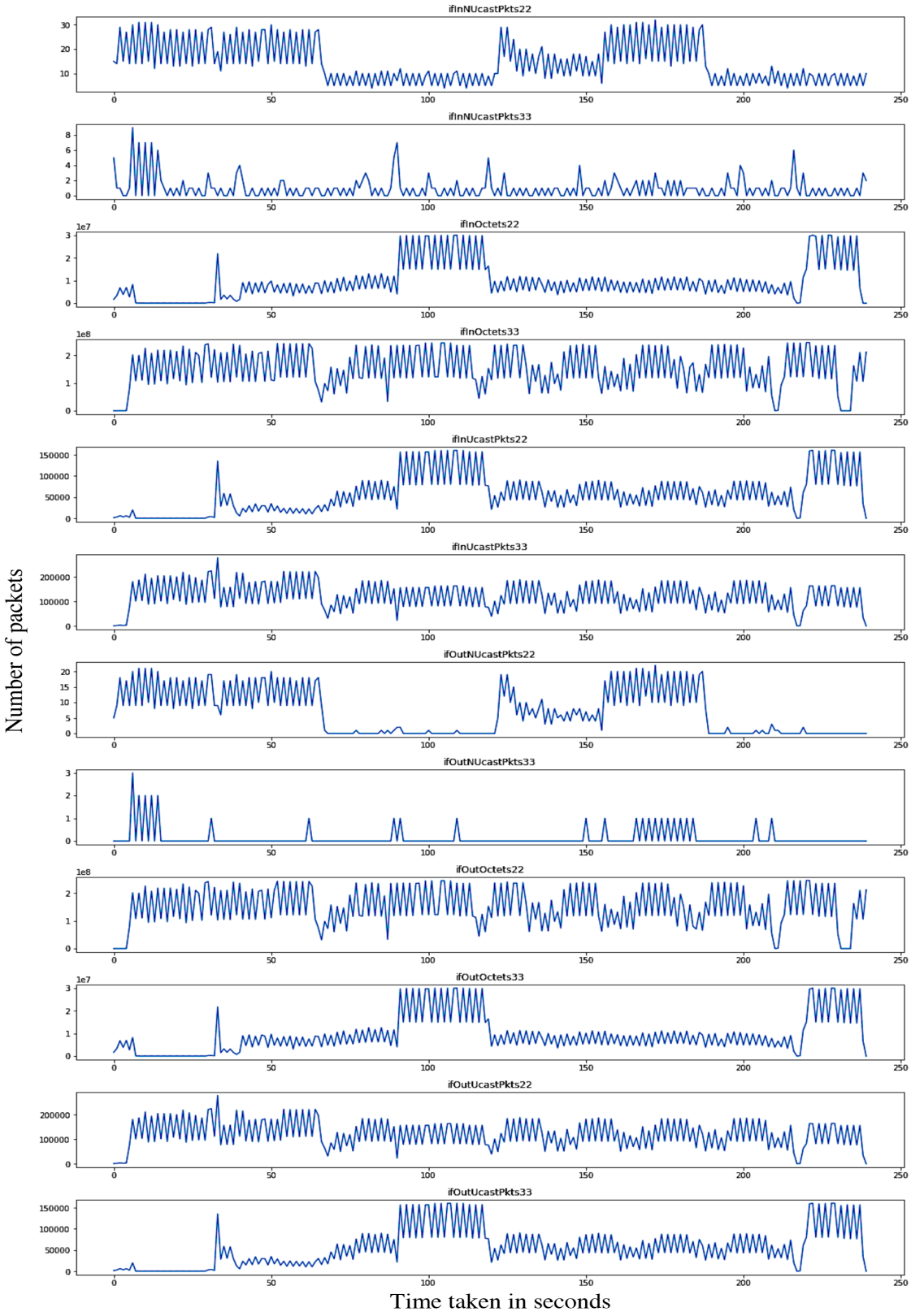

Fig. A.1 The router normal traffic of the heavy workloads

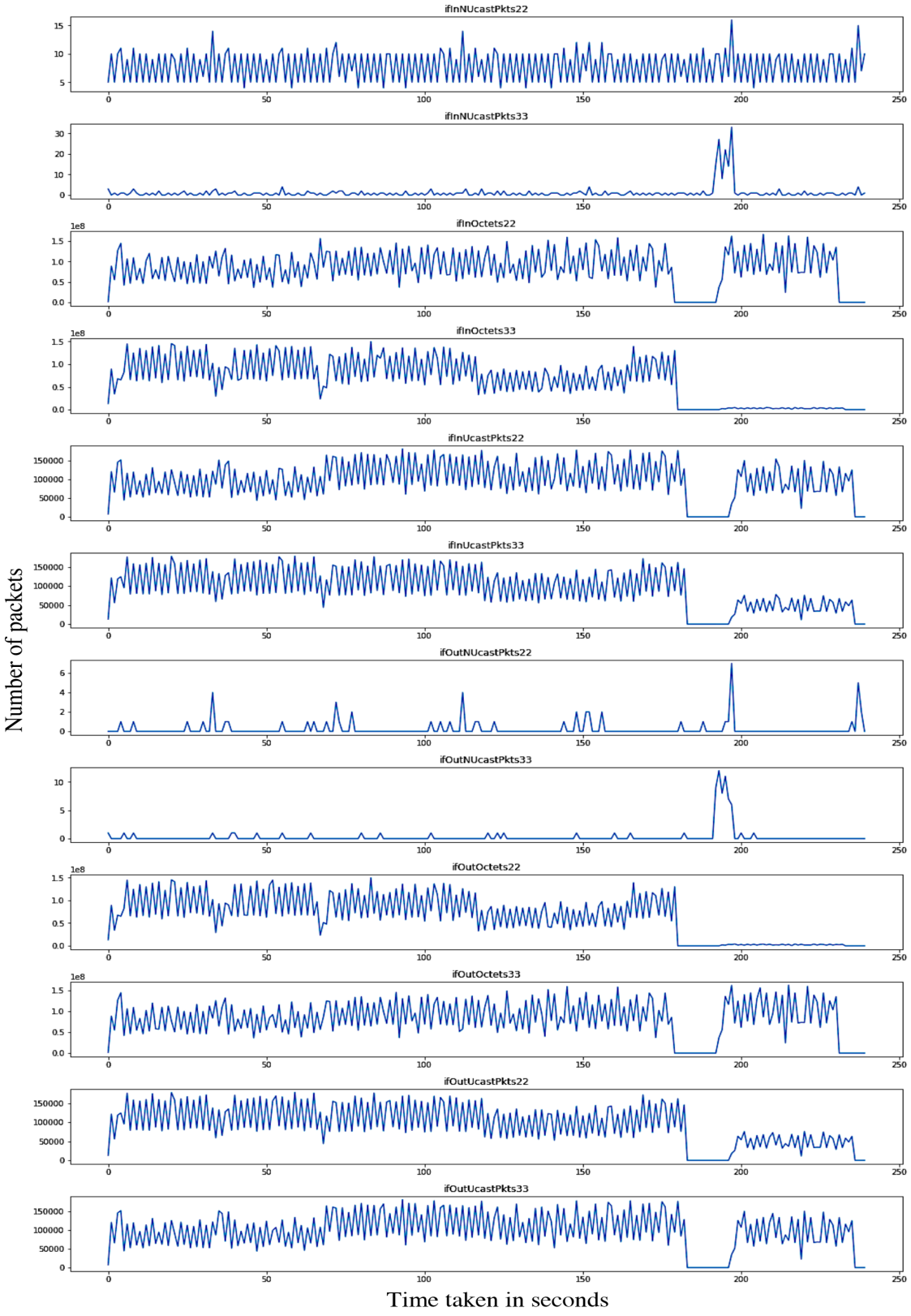

Fig. A.2 The router heavy server crash or link failure traffic workloads

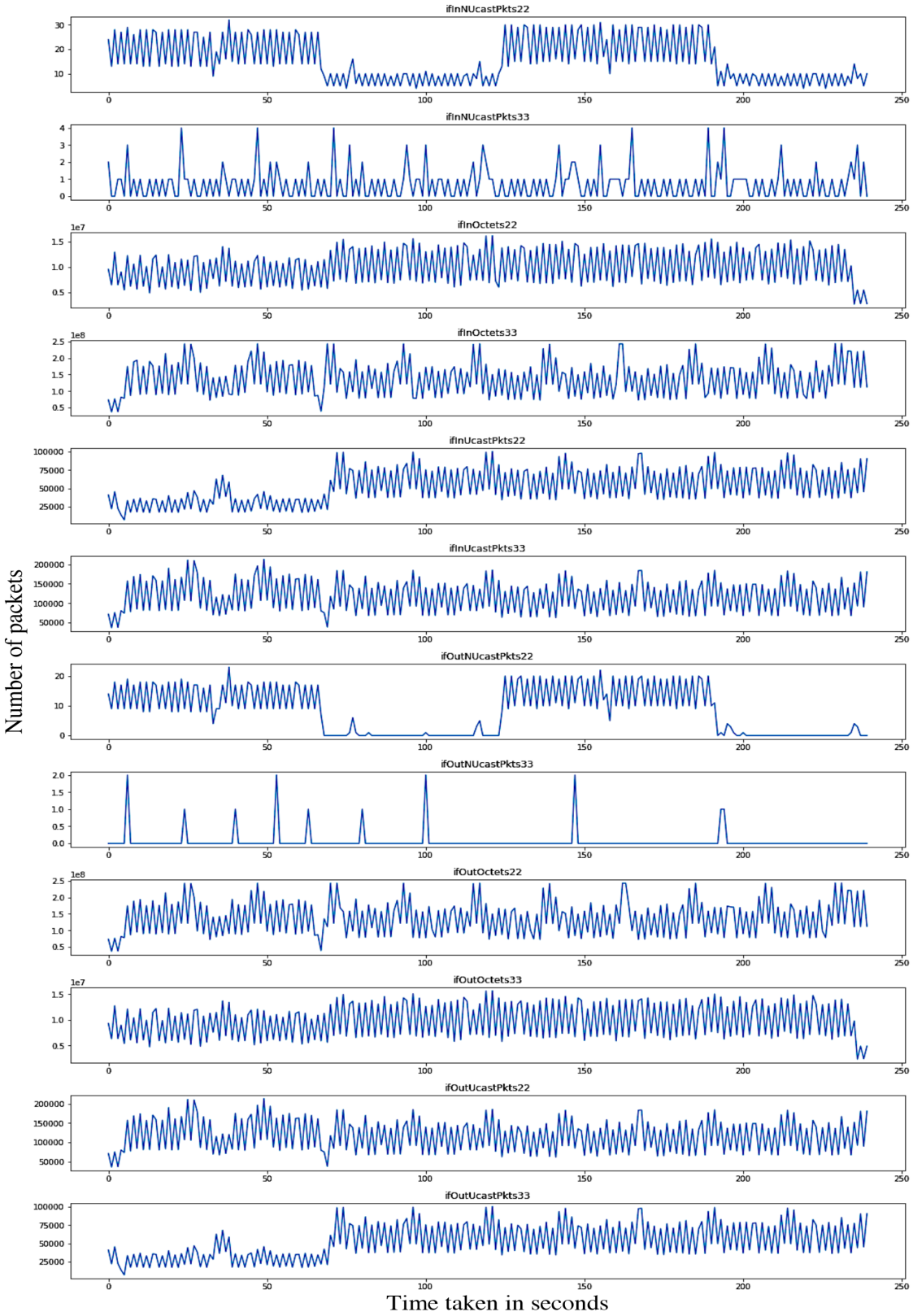

Fig. A.3 The router heavy broadcast storm workloads

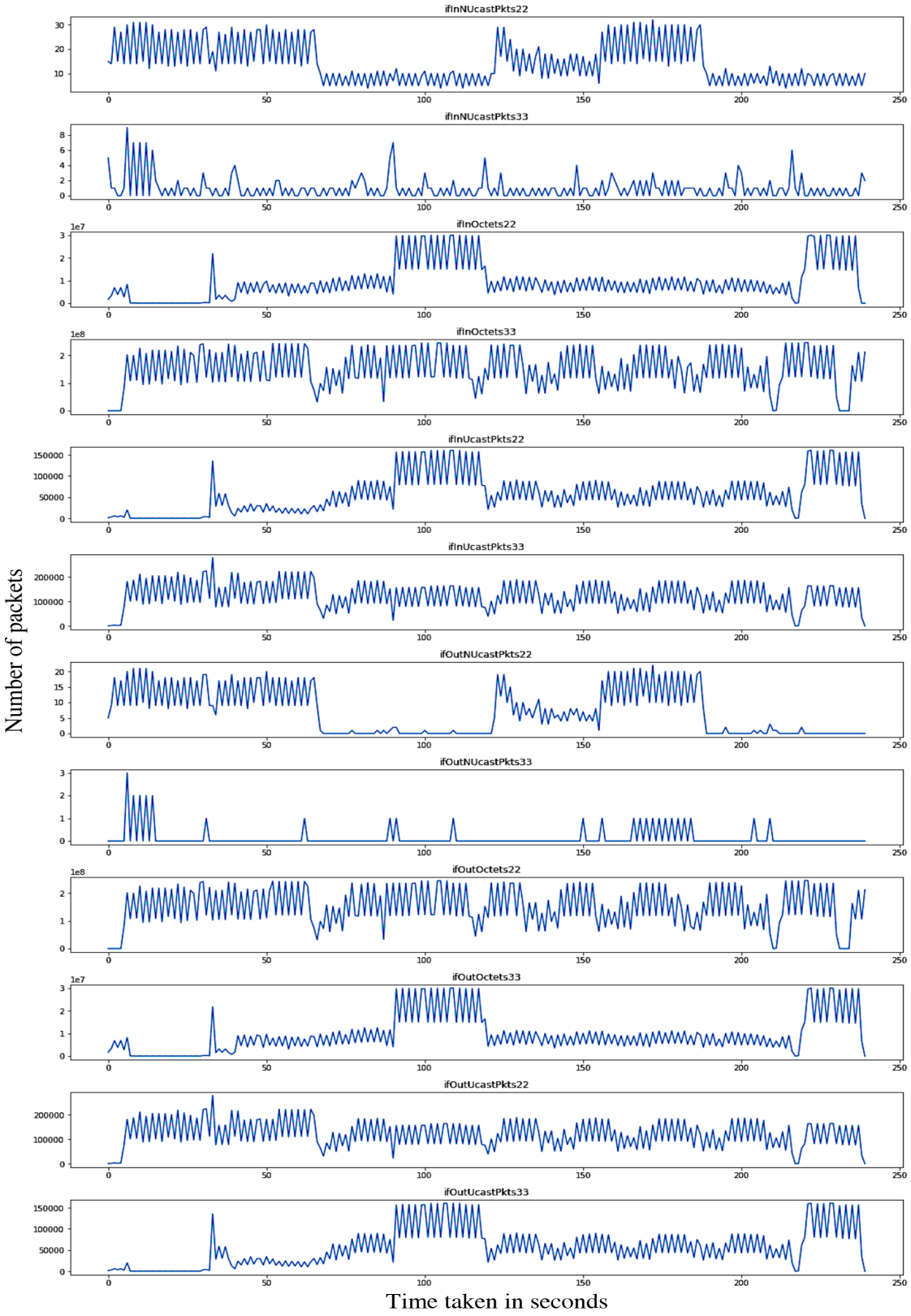

Fig. A.4 The router heavy babbling node workloads

Appendix B

The Router Heavy workloads data visualisation

The Figure B.1,B.2, B.3 and B.4 Represents the features visualisation of the IF-MIB for the traffics that detected when the normal and abnormal traffics were captured in the server in a heavy condition. The graphs visualisation Depict the traffic behaviour when the situation changes to include heavy workloads. These changes are stimulated by adjusting the sizes of the packets and growing the bandwidth, which has an immediate influence on all of the features and also minimises the network performance.

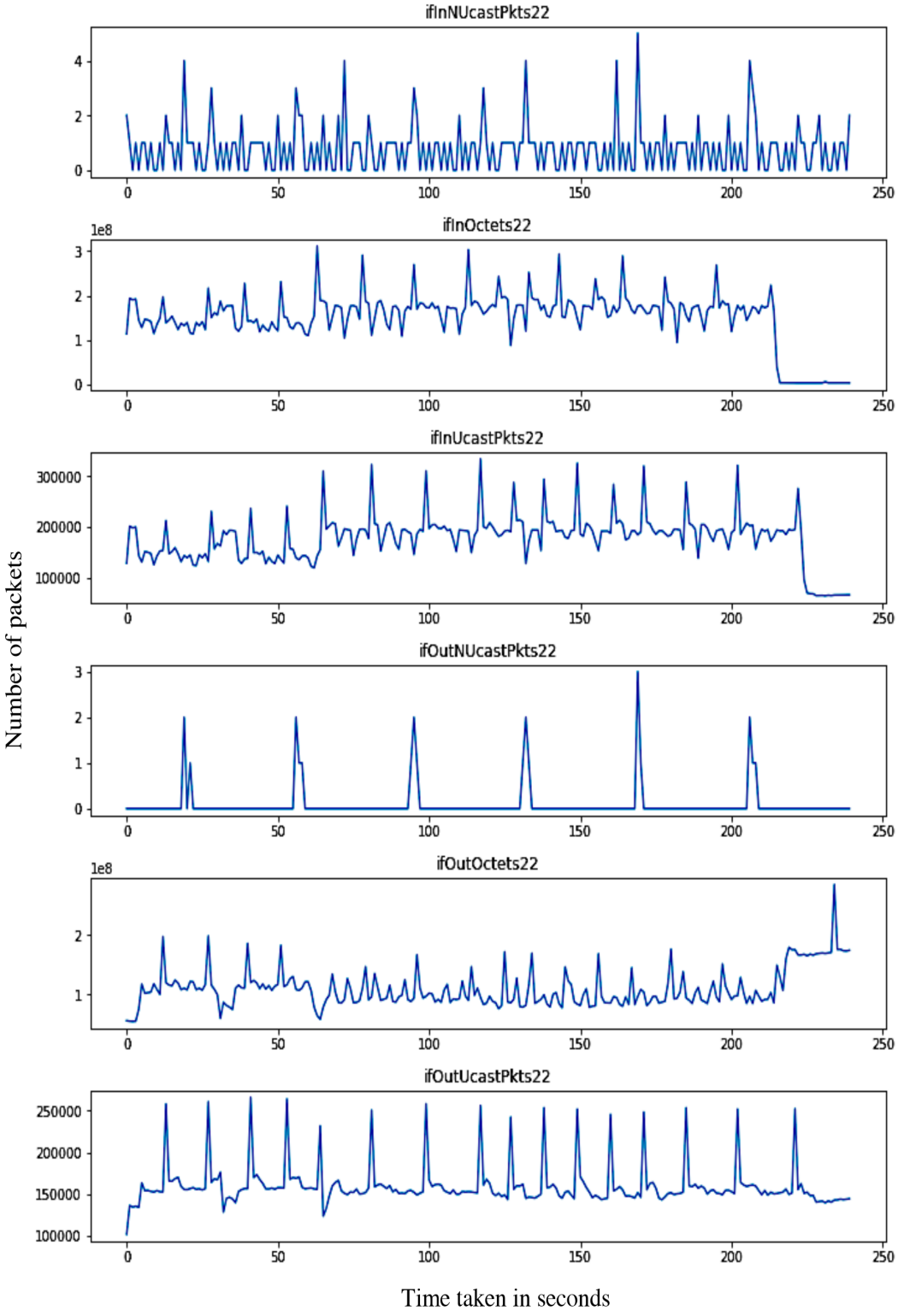

Fig. B.1 Normal traffic of the heavy workloads in server

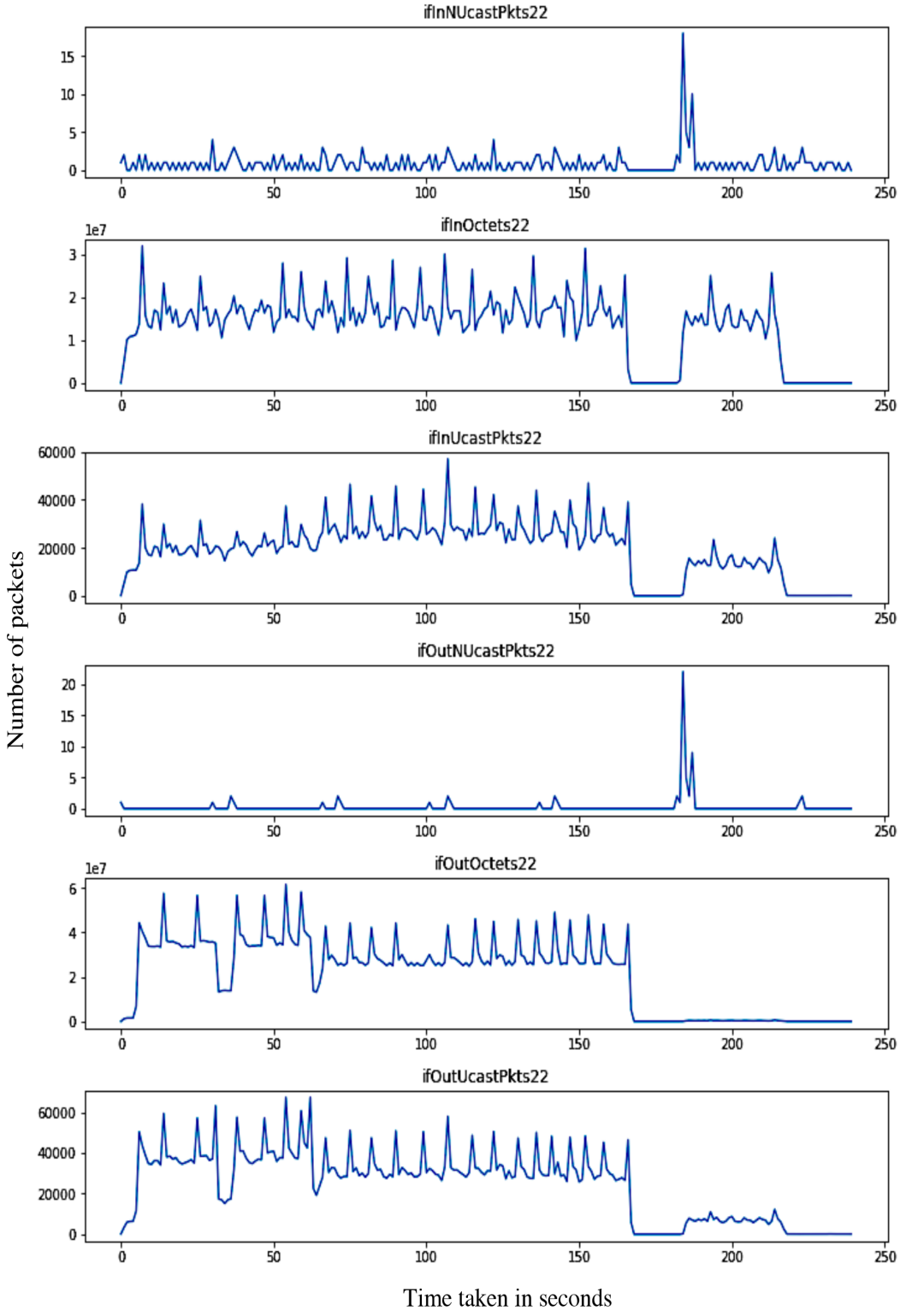

Fig. B.2 The heavy traffic of server crash or link failure workloads in server

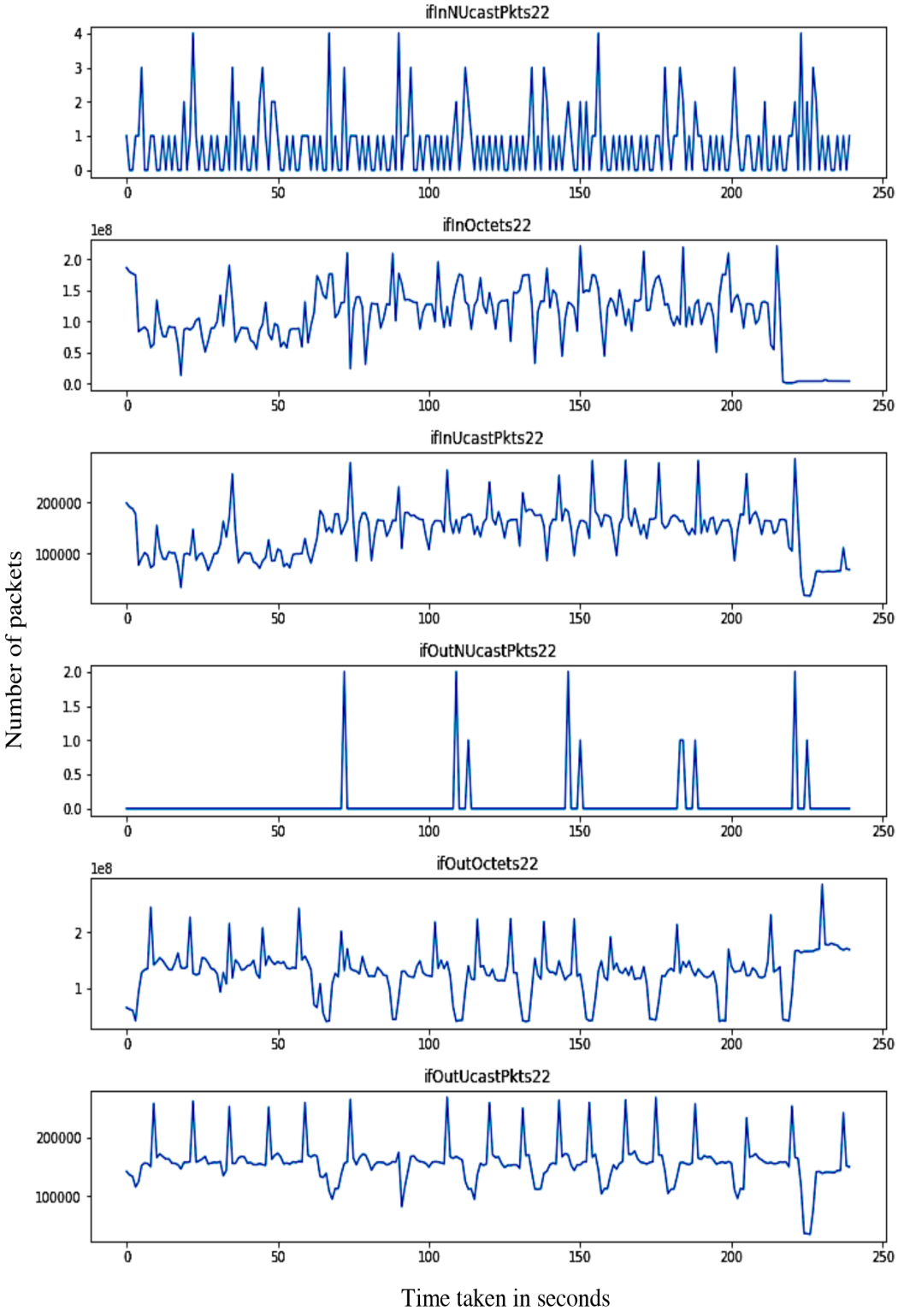

Fig. B.3 The heavy broadcast storm workloads in server

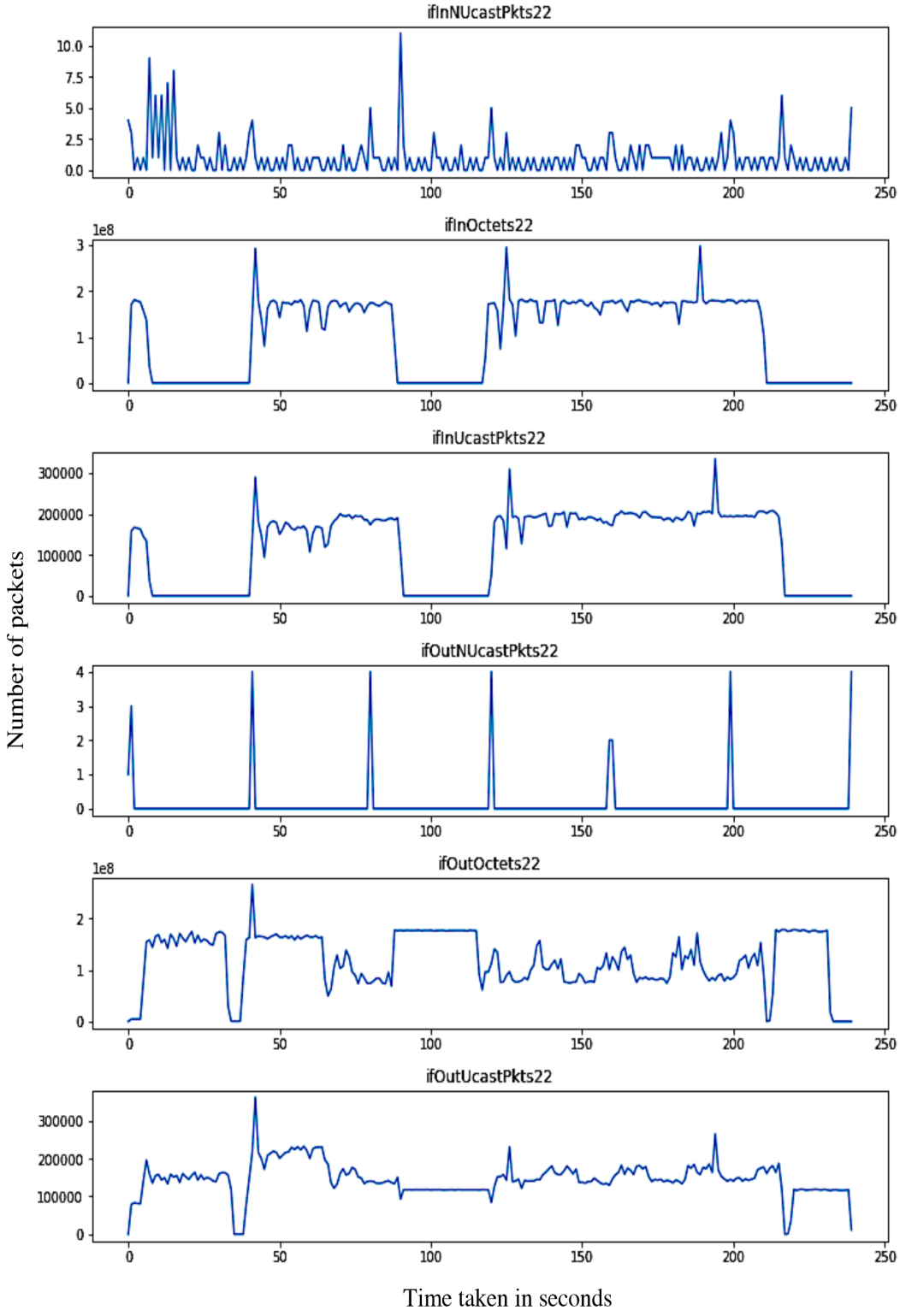

Fig. B.4 The heavy babbling node workloads in server

Appendix C

Standard Scale

This section shows the code written in Python as shown in Figure to purify the noisy data-sets to produce a new sample of data free from any in consistence and unrelated values as shown in Figure C.1.

```
#the function requires a dataframe parameter from pandas library
def standardscaler(df):
    column_names = list(df.columns)
    df_zscore = pd.DataFrame(columns = column_names)
    for column in column_names:
        df_zscore[column] = (df[column] - df[column].mean())/df[column].std(ddof=0)

    return df_zscore;
```

Fig. C.1 The standard scaler filtering code in Pyhton

Appendix D

Normalisation filtering

The written modified code of the normalisation is represnted in this appendix and the detailed code is demonstrated in Figure D.1.

```
def normalisation(x):
    #define the current min, max and mean
    current_min = min(x)
    current_max = max(x)
    current_mean = current_max - current_min

    #define the new min, max and mean
    updated_min = 0.0
    updated_max = 1.0
    updated_mean = updated_max - updated_min

    #check whether the value is constant or not
    if current_mean ==0:
        if current_min < updated_min:
            new_x = updated_min
        elif current_min > updated_max:
            new_x = updated_max
        else:
            new_x = current_min
        normalised_value = [new_x for val in x]
    else:
        scale = updated_mean / current_mean
        normalised_value = [(val - current_min) * scale + updated_min for val in x]

    return normalised_value
```

Fig. D.1 The normalisation filtering code in Pyhton