Université d'Ottawa
Faculté de génie

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

ELG5901 Electrical Engineering Project

Final Report

Name: Abdelrhman Gaber Youssef Saad Rezkallah	ID: 300327290
Name: Abdulrahman Muhammad AbdulSalam Ahmed	ID: 300327218
Name: Ali Amin El-Sayed Mahmoud El-Sherif	ID: 300327246
Name: Amir Safwat Halim Youssef	ID: 300327305
Name: Basma Reda Shaban Abd-Elsalam Abd-Elwahab	ID: 300327209

Graduate Program:

Master of Engineering Electrical and Computer Engineering – Artificial Intelligence and Data Science Track (DEBI Program)

Semester to Register:

Spring 2022

Project Title:

Network optimization and analysis using AI

Supervised by:

Dr. Murat Simsek

Table of Contents:

Contents

1. Introduction 9
1.1 Problem Definition 9
1.1.1 Problem Definition 9
1.1.2 Potential Benefits to Sponsor/Users 10
1.1.3 General approach 10
1.1.4 Learning points and accomplishments 10
1.2 Background 10
1.2.1 5G MIMO Data for Machine Learning: Application to Beam-Selection using Deep Learning [4] 10
1.2.2 Comparative Analysis of Network Fault Classification Using Machine Learning. [13] 12
1.2.3 How to Enhance 5G Network Planning Using Analytics for Faster Time to Market. [6] 14
1.2.4 Scalable Mobile Private 4G and 5G Network Services on AWS from Deloitte.[14] 15
1.2.5 Optimization of 5G Networks for Smart Logistics.[7] 16
1.3 Project Context 19
Project Guidance 19
Project Dependencies 19
2. Design Overview 20
2.1 Requirements 20
2.2 Detailed Design 21
2.2.1 Binary classification 21
2.2.2 multi-class classification 21
2.2.3 Cascaded modeling 22
2.2.4 One vs all modeling 23
2.2.5 Clustering modeling 23
2.2.6 Beamforming selection 27
2.3 Implementation 29
2.3.1 Binary classification 29
2.3.2 Multi-class classification 33
2.3.3 Cascaded modeling 35
3.4 One vs all modeling 35
2.3.5 Clustering modeling 36

2.3.6 Beamforming selection 41
2.4 Testing 44
2.4.1 Data Plan 44
2.4.2 Validation & Verification 44
3. Overall Results and Analysis 50
3.1 Binary classification 50
3.2 Multi-class classification 52
3.3 Cascaded modeling 56
3.4 One vs all modeling 61
3.5 Clustering modeling 64
3.5.1 Agglomerative model 64
3.5.2 DBSCAN model 65
3.5.3 GMM model 66
3.5.4 K-means model: 69
3.6 Beamforming selection 73
4. Deployment Plan 77
5. Conclusions and Future Works 83
6. References 85

Table of Figures:

Figure 1Block diagram of failure detection and classification problem 9
Figure 2 Block diagram of optimization using beamforming selection 9
Figure 3 5G spectrum bands 14
Figure 4 Solution building blocks 14
Figure 5 Coverage analysis 15
Figure 6 Private 4G/5G on AWS, Deloitte Private Networks Lab Houston. 16
Figure 7 Comparison between old and smart logistics 17
Figure 8 The proposed system using 5G technology provides a good solution for the problem 18
Figure 9 Approaches taken using tabular data 20
Figure 10 The approach taken using images data 20
Figure 11 Flow chart of binary classification 21
Figure 12 Flow chart of multi-class classification 22
Figure 13 Flow chart of cascaded modeling 22
Figure 14 Flow chart of one vs all modeling 23
Figure 15 Flow chart of clustering modeling 23
Figure 16 Flow chart of agglomerative modeling 24
Figure 17 Flow chart of DBSCAN modeling 25
Figure 18 Flow chart of GMM modeling 26
Figure 19 Flow chart of K-means modeling 26
Figure 20 Flow chart of beamforming selection 28
Figure 21 Architecture of federated learning 29
Figure 22 Accuracies of different number of features using ANOVA approach on XGboost with test dataset 30
Figure 23 Accuracies of different number of features according to gradient boosting applied on XGboost 30
Figure 24 Pearson ranking on the best 22 features in binary classification 31
Figure 25 Spearman correlation heatmap in binary classification 32
Figure 26 Pearson ranking on the best 33 features in multi-class classification 34
Figure 27 Average linkage of agglomerative clustering 36
Figure 28 Ward linkage of agglomerative clustering 36
Figure 29 Number of clusters vs silhouette score using ward linkage 37
Figure 30 Number of clusters vs kappa score using average linkage 37
Figure 31 The best hyperparameters for DBSCAN to be used with silhouette score 38
Figure 32 The best hyperparameters for DBSCAN to be used with kappa score 38
Figure 33 GMM components vs silhouette scores 39
Figure 34 GMM components vs kappa scores 39
Figure 35 Number of clusters vs silhouette scores for k-means model 40
Figure 36 Number of clusters vs kappa scores for k-means model 40
Figure 37 Architecture of the third model of beamforming selection 43
Figure 38 Architecture of the second model of beamforming selection 43
Figure 39 Architecture of the first model of beamforming selection 43
Figure 40 Cluster distribution for training in agglomerative modeling 45
Figure 41 Cluster distribution for testing in agglomerative modeling 45
Figure 42 Cluster distribution for testing in DBSCAN modeling (1st) 46
Figure 43 Cluster distribution for training in DBSCAN modeling (1st) 46
Figure 44 Cluster distribution for training in DBSCAN modeling (2nd) 47

Figure 45 Cluster distribution for testing in DBSCAN modeling (2nd) 47
Figure 46 Cluster distribution for testing in GMM modeling at components = 15 48
Figure 47 Cluster distribution for training in GMM modeling at components = 15 48
Figure 49 Cluster distribution for testing in k-means modeling at components = 11 48
Figure 48 Cluster distribution for training in k-means modeling at components = 11 48
Figure 53 Cluster distribution for testing in k-means modeling at components = 10 49
Figure 52 Cluster distribution for training in k-means modeling at components = 10 49
Figure 50 Cluster distribution for testing in k-means modeling at components = 11 49
Figure 51 Cluster distribution for training in k-means modeling at components = 11 49
Figure 54 Classification report of XGboost (binary classification) using training dataset 50
Figure 55 Classification report of XGboost (binary classification) using testing dataset 50
Figure 56 Confusion matrix of XGboost (binary classification) using testing dataset 51
Figure 57 Confusion matrix of XGboost (binary classification) using training dataset 51
Figure 58 T-SNE plot on training with actual data 51
Figure 59 T-SNE plot on training with predicted labels 51
Figure 60 T-SNE plot on testing with predicted labels 51
Figure 61 T-SNE plot on testing with actual data 51
Figure 62 Classification report of Logistic regression (multi-class classification) using testing dataset 52
Figure 63 confusion matrix of Logistic regression (multi-class classification) using testing dataset 52
Figure 64 Classification report of naïve bayes (multi-class classification) using testing dataset 53
Figure 65 confusion matrix of naïve bayes (multi-class classification) using testing dataset 53
Figure 66 Classification report of random forest (multi-class classification) using testing dataset 53
Figure 67 confusion matrix of random forest (multi-class classification) using testing dataset 54
Figure 68 Classification report of XGboost (multi-class classification) using testing dataset 54
Figure 69 confusion matrix of XGboost (multi-class classification) using testing dataset 54
Figure 70 T-SNE for XGboost as baseline model 55
Figure 71 whole time of random forest and XGboost in multi-class classification 55
Figure 72 Classification report of hyperparameter tuned XGboost (multi-class classification) using testing dataset 55
Figure 73 confusion matrix of hyperparameter tuned XGboost (multi-class classification) using testing dataset 56
Figure 74 T-SNE of hyperparameter tuned XGboost model 56
Figure 75 Classification report of binary model using training dataset 57
Figure 76 Confusion matrix of binary model using training dataset 57
Figure 77 Classification report of binary model using testing dataset 57
Figure 78 Confusion matrix of binary model using testing dataset 57
Figure 79 T-SNE plot of binary model using training dataset 58
Figure 80 T-SNE plot of binary model using testing dataset 58
Figure 81 Classification report of multi-class model using training dataset 58
Figure 82 Confusion matrix of multi-class model using training dataset 59
Figure 83 Classification report of multi-class model using testing dataset 59
Figure 84 Confusion matrix of multi-class model using testing dataset 59
Figure 88 classification report of class one model in training 60
Figure 87 Confusion matrix of class one model in testing 60
Figure 85 T-SNE plot of cascaded model using training dataset 60
Figure 86 T-SNE plot of cascaded model using testing dataset 60
Figure 89 classification report of class one model in testing 61
Figure 90 Confusion matrix of class one model in training 61
Figure 91 Classification report of class two model in training 61

Figure 92 Classification report of class two model in testing 61
Figure 93 Confusion matrix of class two model in testing 61
Figure 94 Confusion matrix of class two model in training 61
Figure 96 Classification report of class three model in training 62
Figure 95 Classification report of class three model in testing 62
Figure 97 Confusion matrix of class three model in testing 62
Figure 98 Confusion matrix of class three model in training 62
Figure 99 Classification report of class four model in training 62
Figure 100 Classification report of class four model in testing 62
Figure 101 Confusion matrix of class four model in training 62
Figure 102 Confusion matrix of class four model in testing 62
Figure 103 Classification report of class five model in testing 63
Figure 104 Classification report of class five model in training 63
Figure 105 Confusion matrix of class five model in testing 63
Figure 106 Confusion matrix of class five model in training 63
Figure 107 Classification report of aggregation in one vs all using testing dataset 63
Figure 108 Confusion matrix of aggregation in one vs all using testing dataset 63
Figure 109 Predicted agglomerative TSNE plot for training 64
Figure 110 Predicted agglomerative TSNE plot for testing 64
Figure 111 Actual agglomerative TSNE plot for testing 64
Figure 112 Actual agglomerative T-SNE plot for training 64
Figure 113 Predicted agglomerative T-SNE plot for training 64
Figure 114 Actual agglomerative T-SNE plot for training 64
Figure 115 Actual agglomerative T-SNE plot for testing 65
Figure 116 Predicted agglomerative T-SNE plot for testing 65
Figure 117 Predicted DBSCAN T-SNE plot for testing 65
Figure 118 Predicted DBSCAN T-SNE plot for training 65
Figure 119 Actual DBSCAN T-SNE plot for testing 65
Figure 120 Actual DBSCAN T-SNE plot for training 65
Figure 121 Actual DBSCAN T-SNE plot for testing 66
Figure 122 Predicted DBSCAN T-SNE plot for testing 66
Figure 123 Predicted DBSCAN T-SNE plot for training 66
Figure 124 Actual DBSCAN T-SNE plot for training 66
Figure 125 Cluster distribution for testing 67
Figure 126 Cluster distribution for training 67
Figure 127 T-SNE for predicted testing data 67
Figure 128 T-SNE for predicted training data 67
Figure 129 T-SNE plot for actual training data 67
Figure 130 T-SNE plot for actual testing data 67
Figure 131 Cluster distribution for testing 68
Figure 132 Cluster distribution for training 68
Figure 133 T-SNE plot for actual training data 68
Figure 134 T-SNE for predicted training data 68
Figure 135 T-SNE for predicted testing data 68
Figure 136 T-SNE plot for actual testing data 68
Figure 137 Cluster distribution for training 69
Figure 138 Cluster distribution for testing 69
Figure 139 T-SNE plot for actual testing data 69
Figure 140 T-SNE for predicted testing data 69

Figure 141 T-SNE plot for actual training data 69
Figure 142 T-SNE for predicted training data 69
Figure 143 Cluster distribution for testing 70
Figure 144 Cluster distribution for training 70
Figure 145 T-SNE plot for actual training data 70
Figure 146 T-SNE for predicted training data 70
Figure 147 T-SNE for predicted testing data 70
Figure 148 T-SNE plot for actual testing data 70
Figure 149 Training classification report 71
Figure 150 Training confusion matrix 71
Figure 151 Testing classification report 71
Figure 152 Testing confusion matrix 71
Figure 153 Training confusion matrix 71
Figure 154 Training classification report 71
Figure 155 Testing confusion matrix 72
Figure 156 Testing classification report 72
Figure 157 Training classification report 72
Figure 158 Training confusion matrix 72
Figure 159 Testing classification report 72
Figure 160 Testing confusion matrix 72
Figure 161 Testing classification report 72
Figure 162 Combined confusion matrix (cascaded modeling) 73
Figure 163 Combined confusion matrix (Clustering + classification modeling) 73
Figure 164 Experiment 1 top k accuracies vs number of epochs 73
Figure 165 Experiment 1 training loss vs number of epochs 74
Figure 166 Experiment 1 training learning rate vs number of epochs 74
Figure 167 Experiment 2 top k accuracies vs number of epochs 74
Figure 168 Experiment 2 training learning rate vs number of epochs 75
Figure 169 Experiment 2 training learning rate vs number of epochs 75
Figure 170 Experiment 3 top k accuracies vs number of epochs 76
Figure 171 Experiment 3 training learning rate vs number of epochs 76
Figure 172 Experiment 3 training learning rate vs number of epochs 76
Figure 173 Experiment 3 accuracies of different top k accuracies using all datasets 77
Figure 174 Top k-pairs visualization between transmitters and receivers 77
Figure 175 Results after aggregation in federated learning 77
Figure 176 First two deployments on AWS 78
Figure 177 Third and fourth deployments on AWS 79
Figure 178 Screenshot for deployment 2 (contact us) 80
Figure 179 Screenshot for deployment 2 (about) 80
Figure 180 Screenshot for deployment 2 (home) 80
Figure 181 Screenshot of deployment 4 (home) 81
Figure 182 Screenshot of deployment 4 (contact us) 81
Figure 183 Screenshot of deployment 4 (about) 81
Figure 184 Confusion matrices of cascaded modeling and baseline model 83
Figure 185 Confusion matrices of cascaded modeling and clustering modeling 83

Table of Tables:

Table 1 Project Guidance 19
Table 2 Shapes of the used data in beamforming selection 28
Table 3 Values of used parameters of XGboost in binary classification 33
Table 4 Parameters of LR as baseline model in multi-class classification 34
Table 5 Random state of Random Forest model in multi-class classification 35
Table 6 Random state of XGboost of gradient boosting in multi-class classification 35
Table 7 Values of used parameters of XGboost in multi-class classification 35
Table 8 Values of used parameters of binary XGboost in cascaded modeling 35
Table 9 Values of used parameters of multi-class XGboost in cascaded modeling 35
Table 10 Best hyperparameters of DBSCAN at the highest silhouette score 38
Table 11 Best hyperparameters of DBSCAN at the highest kappa score 38
Table 12 Parameters used in the first experiment of beamforming selection 41
Table 13 Parameters used in the second experiment of beamforming selection 42
Table 14 Parameters used in the third experiment of beamforming selection 42
Table 15 Parameters used in federated learning 44
Table 16 Kappa and silhouette scores for agglomerative modeling (ward) 45
Table 17 Kappa and silhouette scores for agglomerative modeling (average) 46
Table 18 Kappa and silhouette scores for DBSCAN modeling (1st) 46
Table 19 Kappa and silhouette scores for DBSCAN modeling (2nd) 47
Table 20 Kappa and silhouette scores for GMM modeling at components = 11 47
Table 21 Kappa and silhouette scores for GMM modeling at components = 15 48
Table 22 Kappa and silhouette scores for k-means modeling at components = 11 48
Table 23 Kappa and silhouette scores for k-means modeling at components = 10 49
Table 24 Experiment 1 accuracies of different top k accuracies using all datasets 74
Table 25 Experiment 2 accuracies of different top k accuracies using all datasets 75
Table 26 Deployment considerations 82
Table 27 Deployment cost per month for both applications 82

Acronyms

Acronym	Full term
AWS	Amazon web services
Amazon EC2	Amazon elastic compute cloud
Amazon ECR	Amazon elastic container registry
Amazon ECS	Amazon elastic container service
Amazon S3	Amazon simple storage service
ANOVA	Analysis of variance
CNN	Convolutional neural network
CSP	communications service providers
DBSCAN	Density-based spatial clustering of applications with noise
EDA	Exploratory data analysis
FL	Federated learning
GaussianNB	Gaussian Naïve bayes
GMM	Gaussian mixture model
IoT	Internet of things
KPI	Key performance indicator
MIMO	Multiple-Input Multiple-Output
MSP	Managed service provider
MTC	Machine type communication
SGD	Stochastic gradient descent
SVM	Support vector machine
T-SNE	t-distributed stochastic neighbor embedding
vCPU	Virtual central processing unit
XGboost	Extreme gradient boosting

1. Introduction

1.1 Problem Definition

1.1.1 Problem Definition

5G network is a bit more complex than the previous generations and in need of regular enhancement as it communicates with a much higher number of devices. There are two main features to work on when applying artificial intelligence, firstly, network failure detection and classification (the failure and its type) to provide lower maintenance cost and minimalize maintenance work for better network establishments including lower cost and lower use of devices, nevertheless, higher provided quality of service for the users. Secondly, 5G network optimization using beamforming selection from LiDAR to increase data rate to provide lower maintenance cost and avoid unnecessary maintenance work. To settle these problems five aspects of 5G which are mobility, spectrum, user's peak data rate, user's experience data rate and network energy efficiency will need improvement through the following points:

- Network failure detection and classification (the failure and its type) to provide lower maintenance cost and minimalize maintenance work.

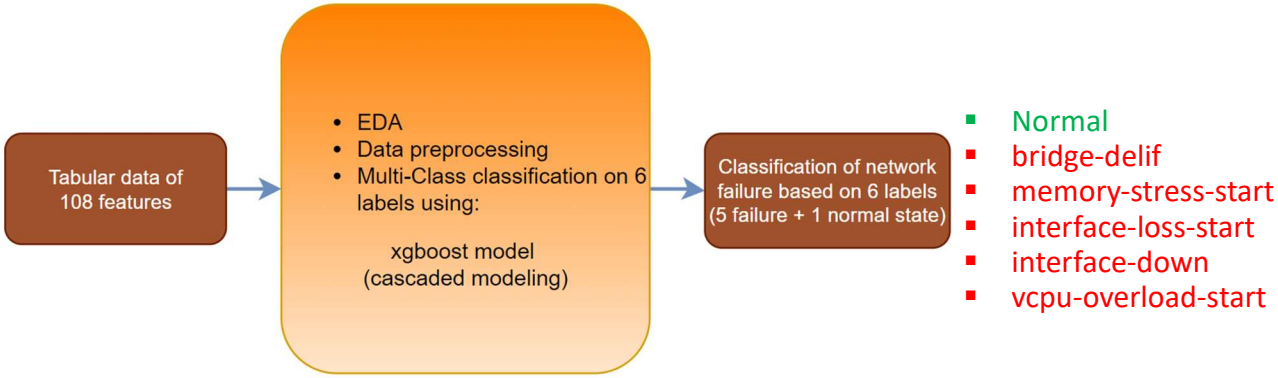

Figure 1Block diagram of failure detection and classification problem

- 5G network optimization using beamforming selection from LiDAR to increase data rate.

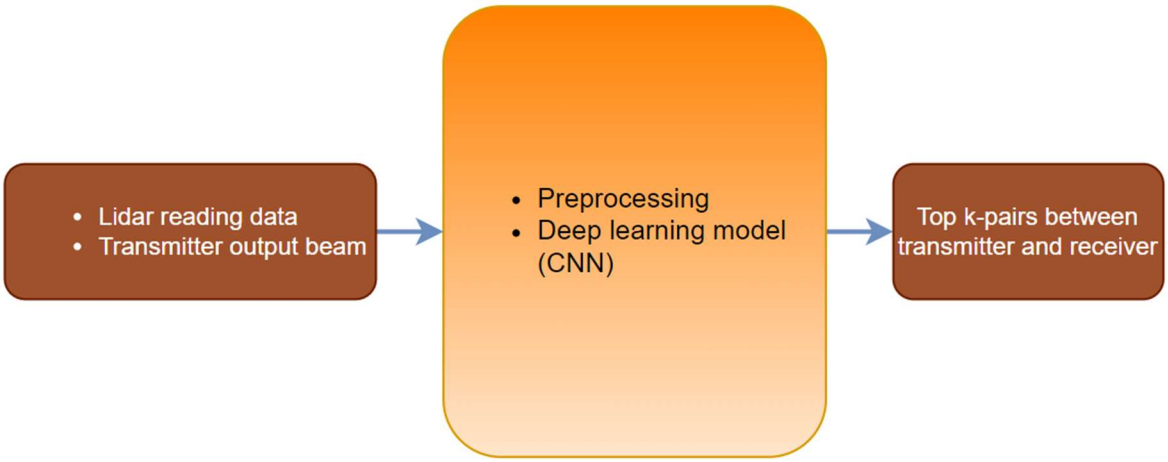

Figure 2 Block diagram of optimization using beamforming selection

1.1.2 Potential Benefits to Sponsor/Users

- AWS Private 5G is currently available in the following AWS Regions: US East (Ohio), US East (N. Virginia), and US West (Oregon).[8]
- AWS has a service category in augmented reality (AR) and virtual reality (VR) fields that require 5G technology.
- Optimization of beam selection efficiency will lead to the increase of spectrum efficiency which is beneficial to the company, nevertheless, users can experience convenience through AWS or any other network applied to it.

1.1.3 General approach

To achieve the mentioned aims, 5G optimization will be worked on using, multi-input multi-output structures (MIMO), classical and deep machine learning techniques to detect the capabilities of the network to obtain a positive outcome, deep learning techniques.

1.1.4 Learning points and accomplishments

- Beamforming selection using deep learning.
- Optimization for 5G network.
- Failure detection forecasting.

1.2 Background

1.2.1 5G MIMO Data for Machine Learning: Application to Beam-Selection using Deep Learning [4]

Research goal:

5G technologies require complex algorithms to reduce the failure of the network at both receivers and transmitters; this problem is called beamforming selection. To increase the efficiency and availability of networks, machine learning models can be used. But according to the size, large data files, and model complexity, deep learning is the most suitable technique for such a problem. This paper presents a solution to generate a channel realization that represents 5G scenarios with the mobility of both receivers and transmitters. This paper objective is to make it easier to investigate machine learning model based on the problems of 5G mm-waves.

Methodology:

For generating the data, challenges of mm-waves channel modeling must be considered to deal with issues while generating the data, which depends on the mm-wave MIMO modeling. It can be split into configuration and simulation methods. For the configuration methodology, a ray-tracing simulator and a traffic simulator have been used. For the simulation methodology, python orchestration code has been used. Three approaches have been used to select the best beamforming, the first one is convolutional drop-based classification, the input is transmitters and receivers that had 4*4 uniform planar antenna arrays and the output is the best beam pair index. For classification, linear SVM, AdaBoost, Decision Tree, Random Forest, and Deep Neural Networks have been used.

The second approach is Conventional drop-based multivariate regression. This problem is considered as a multivariate regression problem. Shallow neural network and Deep neural network have been used.

The third approach is Deep reinforcement learning, which learned from the previous experience and take a reward if it takes the right action, it maximizes the reward over time. It applied a cascaded model of two convolutional deep neural networks. Finally, comparing it with the dynamic programming.

Results and evaluation:

On all of the data, linear SVM model achieved 33.2% accuracy, AdaBoost model achieved 55% accuracy, Decision Tree model achieved 55.5% accuracy, Random Forest model achieved 63.2% accuracy, and Deep Neural Networks achieved 63.8% accuracy.

For regression, Elementary regression modeling and AZI regressor have been used for evaluating the results. Shallow neural network achieved 6.5 at departure (TX), 7.9 at arrival (RX) with ELE regressor, and with AZI regressor, 137.4 at departure (TX), and 180.6 at arrival (RX).

Deep neural network achieved 4.8 at departure (TX), 6.2 at arrival (RX) with ELE regressor, and with AZI regressor, 49.9 at departure (TX), and 102.8 at arrival (RX).

The deep reinforcement learning achieved Root-Mean-Square error on average 0.074 and 67.5% accuracy. In case of dynamic programming, the reward increases to 0.891, it indicates that the agent learned from the experience and rewards.

Conclusion:

Beamforming selection is an important problem to be considered to determine the best key pair of transmitters and receivers. There are many ways to generate and collect a LiDAR data such as generating a data from a simulation. After applying classical machine learning models, deep neural networks, reinforcement learning and dynamic programming, the high results are the reinforcement learning, as the

agent can take a reward if it takes the right action. To enhance the reinforcement learning environment, dynamic programming can be used.

Future work:

Apply the reinforcement learning environment with a real data, and try different approaches for model validation. As the computational cost of deep reinforcement learning is very high, a tuned deep convolutional neural network will be applied to minimize the model's cost.

1.2.2 Comparative Analysis of Network Fault Classification Using Machine Learning. [13]

Research Goal:

Creating a framework that can classify 5G network failure into three types using simulation data generated by a dataset generator that can purposefully inject several types of failure into network performance data, and analyze the effect of data balancing and feature selection and reduction on each algorithm used in this framework.

Methodology:

The methodology consisted of three main components:

The first component was the dataset generator which purposefully injected three failure types into network performance data (node-down, interface-down and CPU Overload), each of these failure types was caused by a different root cause in a real world scenario, Where for example, node-down is a power supply related failure category, Interface-down is an unstable connection related failure category which may be caused due to loose cables, And CPU Overload happens due to a distributed denial of service attack on the network, each failure that happened in the network was represented by a scenario, and each scenario was executed using command line interface, and then performance network data throughout the scenario was stored in a data storage.

The second one is the preprocessor which separated Network data into two categories, device data, and interface data, in each of those categories, Text data was converted into vectors using bag of words, difference was applied to values of cumulative nature , and values of instantaneous nature were not changed such as CPU Usage, because of the large number of features, two types of datasets are formed, one for the device data, and one for the interface data, and a model was trained for each type of dataset.

The last one is the evaluator which evaluates the three used algorithms Which are MLP, SVM, and Random Forest based on three criteria, precision, recall and F1-Score as the simulated data reflects the true ratio of normal vs abnormal (failure) data in a real world scenario which represents severe imbalance case,

that's why F1-score is needed, as well as an improvement step which involved experimenting with different data balance ratios(normal/abnormal data ratio), and number of features selected.

Results and Evaluation:

Under these conditions of data imbalance, and no feature reduction, SVM was not able to achieve any result (0% F1 Score), MLP achieved poor performance, while random forest had the best performance with over 95% F1 Score.

When increasing the number of data examples, Random Forest result was not affected, as it's still the best classifier, SVM result was improved, while MLP result did not improve.

The next two experiments on feature reduction and dataset balance were done only on the MLP and SVM because based on the above-mentioned results, only those two algorithms need the improvement. SVM showed huge improvement when using feature selection by choosing the best 10 features based on feature selection done by the random forest, giving results almost equal while MLP result only improved in the Classification of CPU Overload failure category but decreased in other categories.

Regarding the data balance case, the results showed that the random forest was not affected by the severe imbalance in the data while both MLP and SVM were clearly affected by this experiment which made them more suitable classifiers for predicting the normal case.

Conclusion:

The reason of getting the above result which specified the random forest as the best place for this case is because of the isolated nature of the simulated performance data, while through the experiments it was shown the SVM and the MLP can improve the results based on different training conditions such as feature reduction or a carefully considered the balance between normal and abnormal cases.

Future Work:

Verify the efficiency of random forest algorithm on industrial networks as well as look into its ability in the analysis of complex failure categories.

Try using a boosting algorithm such as Xgboost, as it has demonstrated efficiency in extreme data imbalance cases like this one.

1.2.3 How to Enhance 5G Network Planning Using Analytics for Faster Time to Market. [6]

Challenges:

Building a 5G network involves many considerations, including Radio Propagation and Device Characteristics. All of this makes network planning a daunting task. 5G New Radio will use different frequency bands, making 5G network planning a complex and ongoing task.

Communication service providers spend a great deal of time and effort managing multiple network planning tools that cannot handle large geospatial data and heterogeneous data sets, and too little time is spent on predictive dynamic analysis for critical initiatives such as 5G deployments. It is time consuming. Network planners must have additional data and analytical tools to make accurate site placement and network optimization decisions in order to secure a large capital investment.

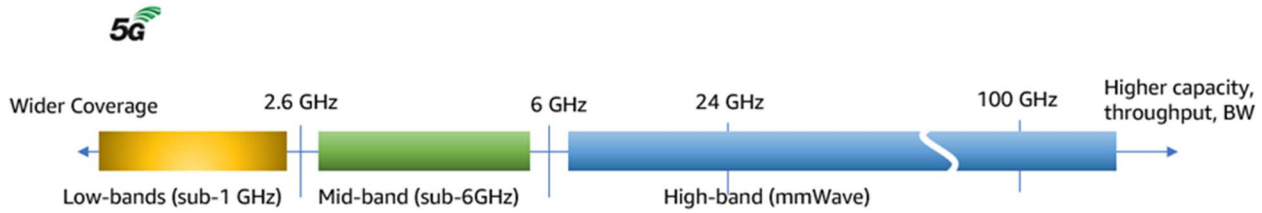

Figure 3 5G spectrum bands

To reduce the capital investment to determine how many sites the vendors need to cover this area also in the 5G network we have mm wave this cover short distances This makes the spatial analysis in 5G more complex, requiring more powerful processing capabilities.

Solution:

Kinetica built a solution on AWS to Communications service providers (CSPs) can significantly accelerate 5G planning with interactive and predictive modeling of network coverage at scale.

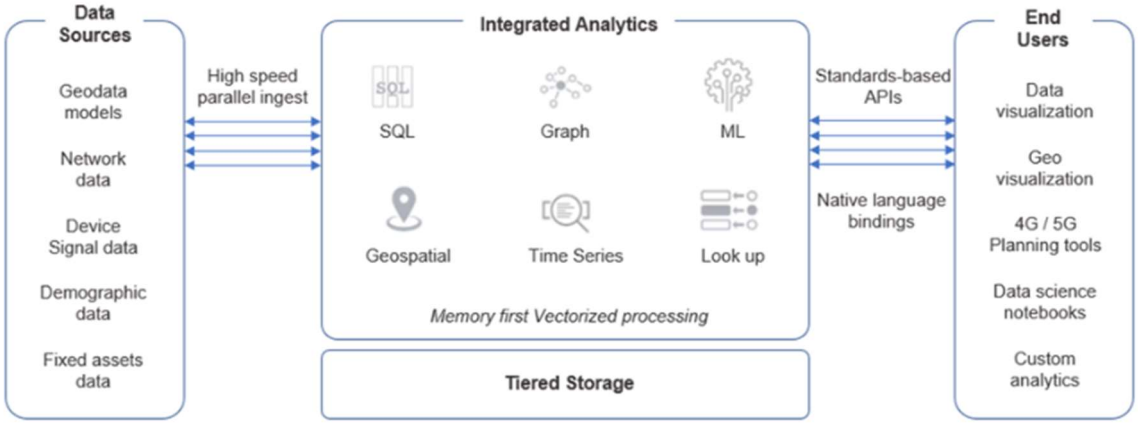

Figure 4 Solution building blocks

Kinetica on AWS allows CSPs to quickly develop and deploy complex RF propagation models against business and location datasets for visual and interactive scenario analysis.

CSPs can quickly understand and analyze different scenarios for 5G coverage, while planners can interact with the data in real-time. The next figure shows a sample of analysis solution:

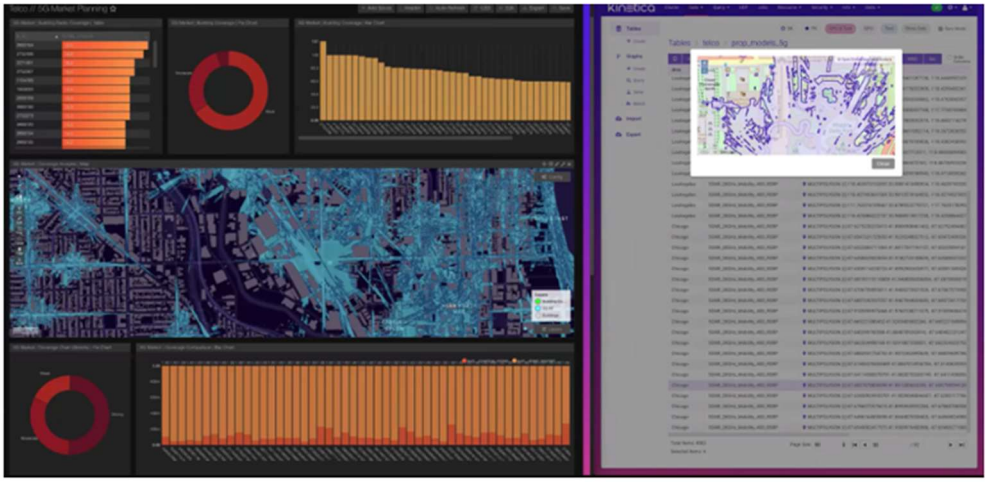

Figure 5 Coverage analysis

Conclusion:

This solution help vendors (CSPs) to reduce the capital investment by applying the analysis planning to decide the right locations and how many sites they need for covering the specific area.

1.2.4 Scalable Mobile Private 4G and 5G Network Services on AWS from Deloitte.[14]

Challenge:

As digital transformation and the Internet of Things (IoT) become business imperatives, private 4G and private 5G networks must connect everything from robots, cameras, signage, and machinery to virtual reality applications. This huge kind of data need security, high performance and low latency while doing preprocessing for this kind of data in real time to extract business value from this data.

Solution:

Amazon Web Services (AWS) and Deloitte, an AWS Premier Consulting Partner and Managed Service Provider (MSP), enable modern enterprises to build their own private 4G or private 5G networks. This solution provides security, High performance, and low latency for real time preprocessing. The next figure shows the build solution on AWS services:

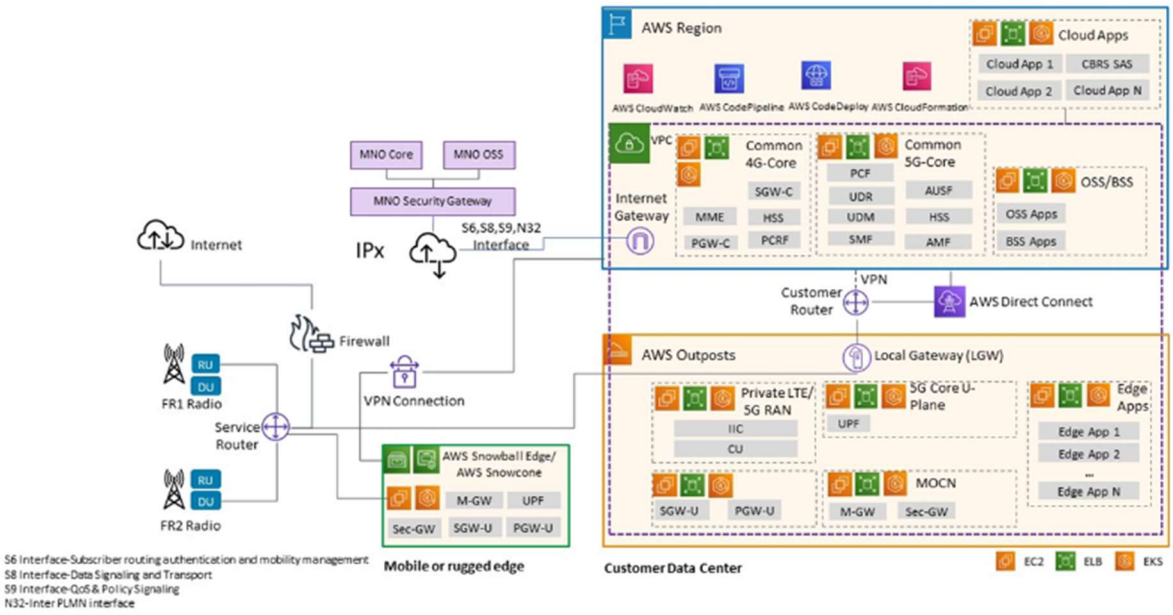

Figure 6 Private 4G/5G on AWS, Deloitte Private Networks Lab Houston.

Use Cases for Private Networks:

For example, in manufacturing, private networks allow factory automation to leverage camera analytics, AI/ML, and automation to check product quality in real time. Similarly, a private network will allow retailers to provide a virtual mirror to their in-store customer, leveraging high-bandwidth, low-latency connections to deliver his interactive AR (Augmented Reality) experience.

Conclusion:

This private network solution enhances the user experience in real time preprocessing and provide high security because this is private network, high performance and Low latency.

1.2.5 Optimization of 5G Networks for Smart Logistics.[7]

Consumer expectations have shifted in recent years as a result of the effect that the Internet and mobile communications have had on our everyday lives. Users buy and sell things online in tiny quantities, with shipment times of a few days or even hours. As a result, logistics supply networks must be extremely flexible and efficient. Wireless communication is becoming more common in manufacturing and logistics as part of Industry 4.0. Novel technologies are utilized in Smart Logistics to provide dynamic supply chains with lower management, energy, and storage costs. This study presents a framework for using 5G networks' application-specific networking capabilities. Machine-Type Communications (MTC) are found in many devices and serve a wide range of operations in an industry that is heavily reliant on data. Massive Machine Type Communication is one of the primary use cases considered in the 5G architecture. This study presents a 5G network management system that may be used more efficiently for an IoT-enabled supply chain.

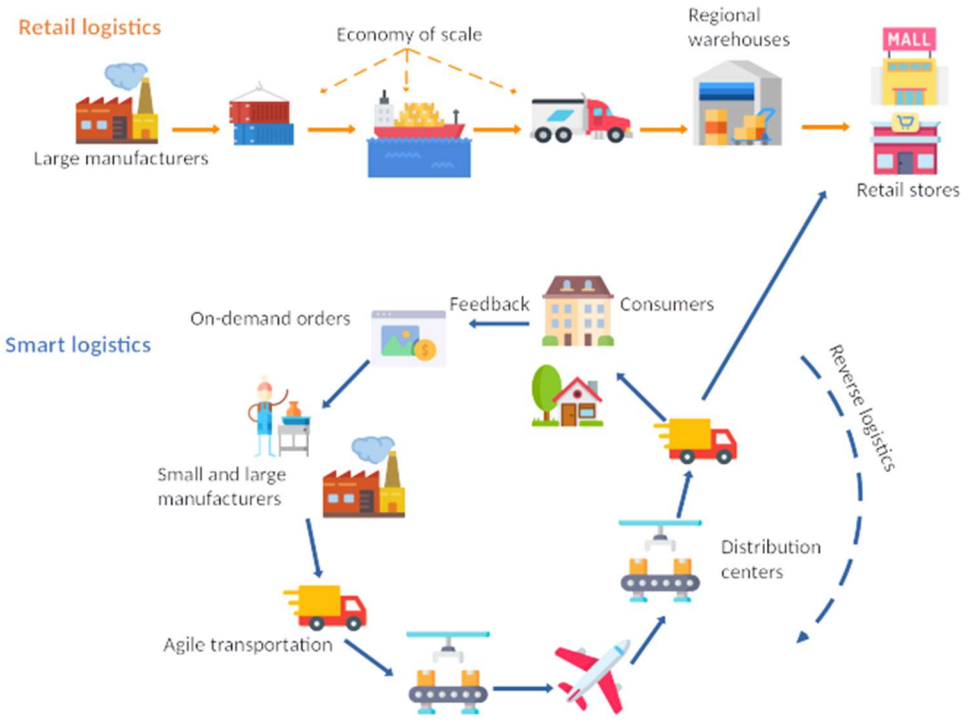

Figure 7 Comparison between old and smart logistics

Problem:

The lack of demand and slow distribution of items is one of the most notable aspects of the internet market. A single product may have an extremely uneven distribution of consumers around the world, and sales may come and go at random times. On the other hand, product sellers might be big or small producers that create and/or transport their goods when there is no demand. In conventional retail logistics, goods are produced and shipped in bulk, transported to local warehouses, and then held there until they are distributed to retailers. By tracking sales growth and making future projections, retail establishments can drive demand in a more or less predictable manner. This system works best for commodities that don't vary over time and have a localized consumer base. However, this plan falls short when it comes to the customized goods mentioned before or goods that sell slowly and cannot be kept in warehouses.

Solution:

This system's goal is to meet the demands of the supported Industry 4.0 applications while optimizing the allocation of the available resources into the slices for various types of traffic. To forecast the workload and determine which apps and how many instances will be run inside a network element's service area, big data analytics will be employed. The applications must be modelled as a first step. The resources in the network will then be set with NS using the prediction and application models. The major goal of the suggested system is to establish an adaptive and predictive NS focused on the demands of logistics and

allow 5G for usage in this vertical. The ability to outsource communications management is the primary benefit of 5G in logistics (while keeping confidentiality thanks to end-to-end encryption). This will substitute a reduced service cost for ownership expenses, equipment and administrative costs for communications infrastructure, security updates, etc. Second, 5G is the first "universal" connection option that, when properly tuned, can accommodate all the traffic profiles now in use. Utilizing a single technology makes purchasing and managing radio equipment much easier, while also lowering expenses. Finally, a network that has been properly optimized will be able to meet the needs of some Industry 4.0 applications. The efficient use of industrial resources, energy, etc. will be made possible by enabling the stable functioning of such services. The availability of a network that can, for example, provide coverage inside of delivery vehicles would allow applications like package tracking in real time without the need for specialized equipment in the logistics field.

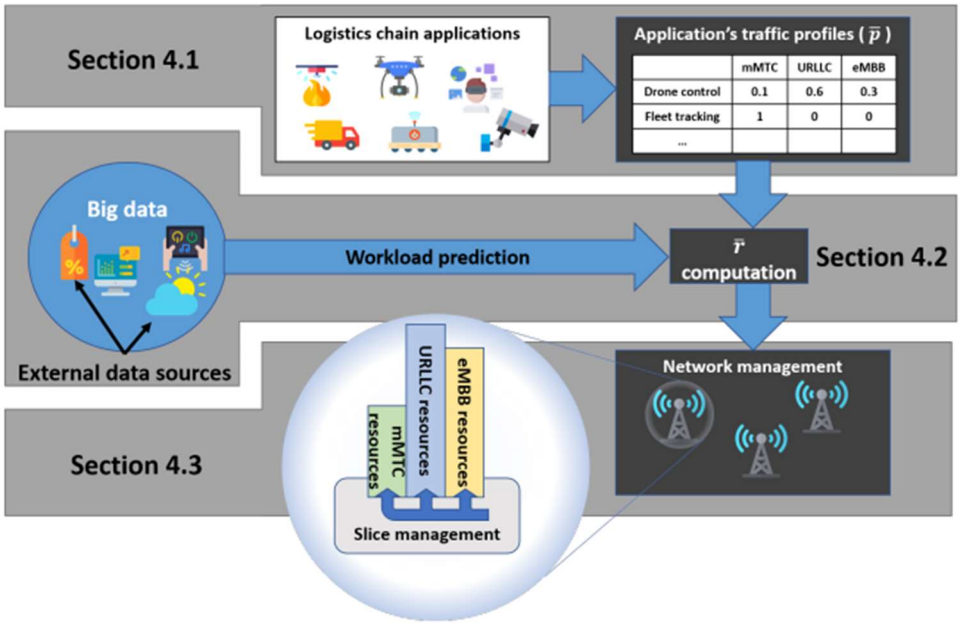

Figure 8 The proposed system using 5G technology provides a good solution for the problem

1.3 Project Context

Project Guidance

	uOttawa Supporter (Dr. Murat Simsek)	Egypt Mentor (Dr. Anas Youssef)	Project Sponsor - AWS (Dr. Sayed Eldawy)
Summary of expertise and guidance	Knowledge and technical guidance and paperwork recommendations	Knowledge and technical guidance and paperwork recommendations	Knowledge and technical guidance and assistance with previous expertise in the field
LinkedIn	https://www.linkedin.com/in/murat-simsek-32a7b220/	https://www.linkedin.com/in/anas-youssef-62137370/	https://www.linkedin.com/in/sayedeldawy
Job title	Engineer in residence and Research Associate	Research Assistant & Application Developer at University of Calgary	Sr. Solutions Architect at AWS
Type of interaction	Online meeting / Email	Online meeting / Email	Online meeting / Email
Frequency of interaction	Every two weeks or when knowledge supervision needed	Every two weeks	Every two weeks or when technical or knowledge assistance needed

Table 1 Project Guidance

Project Dependencies

In order to build, run or store python notebooks and the used data, the following tools were used:

- Anaconda Navigator
- Google colab
- Google drive
- Visual studio code
- Amazon Sagemaker
- Amazon S3

In deployment, the following AWS services were used (will be illustrated more in “Deployment Plan”) :

- Amazon EC2
- Amazon ECR
- Amazon ECS
- Docker
- AWS Fargate
- Amazon CLI V2
- PuTTY

2. Design Overview

In this project, 2 types of data were used (tabular data and images), so the project is mainly split into two parts. The following figures show the basic structure for each part that will be described in details later on.

In the first part that is using tabular data, the four main approaches are marked in blue which are binary classification, multi-class classification, cascaded modeling and clustering modeling and they are represented as shown:

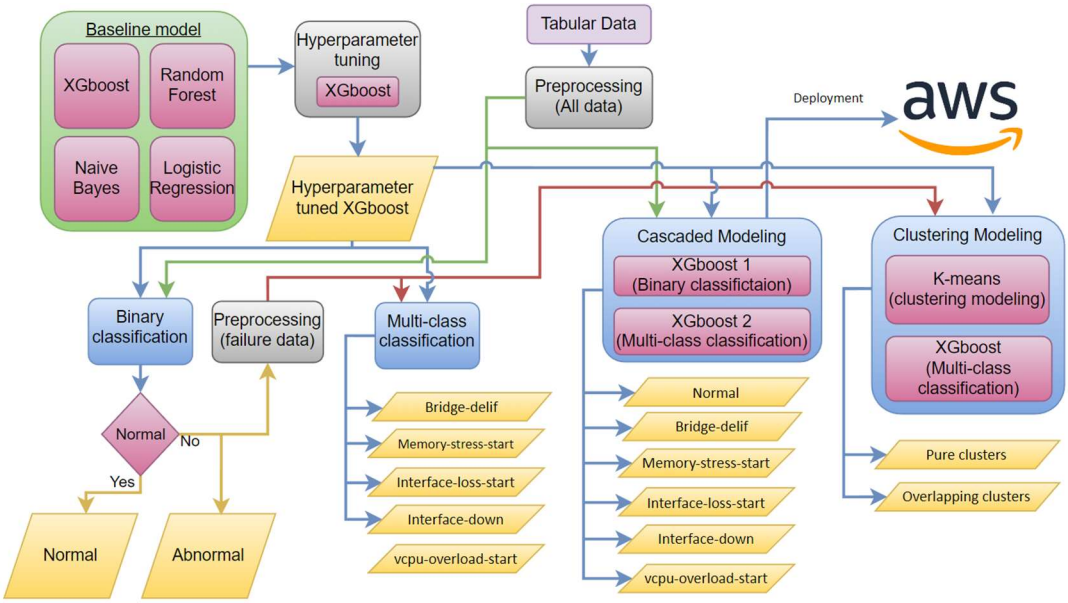

Figure 9 Approaches taken using tabular data

In the second part, a deep learning model was used in training that will be illustrated in details in the following sections. The following figure is the overall approach of the second part of the project:

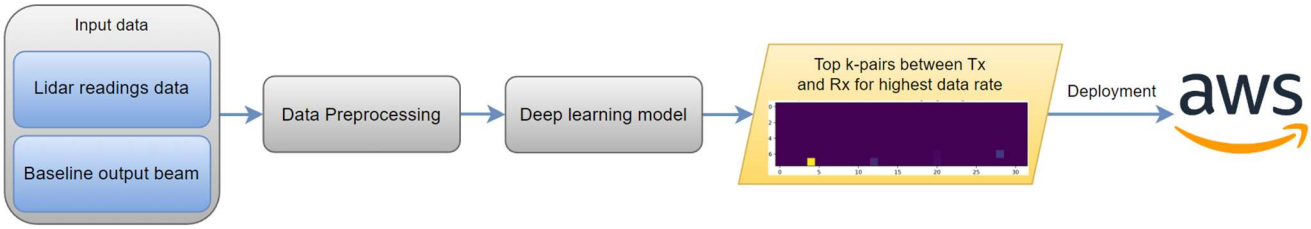

Figure 10 The approach taken using images data

2.1 Requirements

- At the end of this project the user can get the current state of the network by using a deployed version of the project by assigning the specific network's readings in a webpage.
- Nevertheless, the user can get the top K-pairs between transmitters and receivers to know the best possible connection in current state to reach the highest data rate.

2.2 Detailed Design

In this section, parts 2.2.1, 2.2.2, 2.2.3, 2.2.4 and 2.2.5 are using the tabular data and 2.2.6 is using the images data.

2.2.1 Binary classification

Network classification is an important method to predict if the network will fail or not by classifying it into normal and abnormal states. This technique has been applied by performing data preprocessing, feature selection and classification using XGboost model. XGboost model has been applied and tuned to predict if the network will fail (abnormal) or not (normal) as shown in the following flow chart:

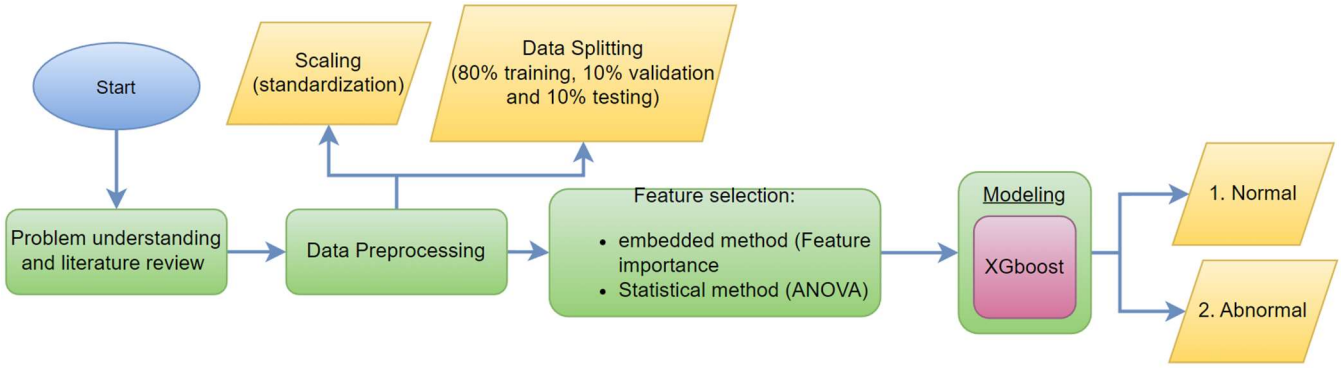

Figure 11 Flow chart of binary classification

2.2.2 multi-class classification

Because of the imbalanced data, the binary class classification was not enough for network failure classification, so applying multiclass classification based on failure data would help to classify failure types. The goal of multiclass classification is to classify the network into normal and different failure types which are:

- bridge-delif
- memory-stress-start
- interface-loss-start
- interface-down
- vcpu-overload-start

By performing data preprocessing, feature selection on failure data, then apply XGBOOST, Random Forest Naïve Bayes, and logistic regression to choose a champion model among them, which turned out to be XGboost. Finally, tuning and evaluating the results of that champion model. The following figure shows the structure of multi-class classification:

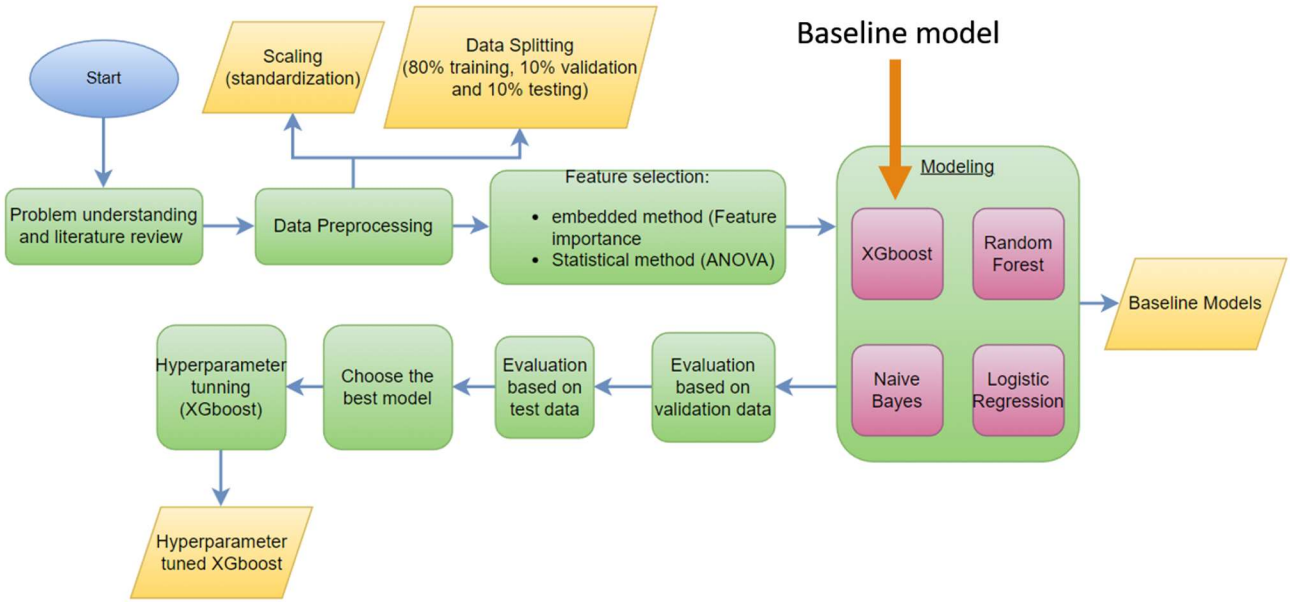

Figure 12 Flow chart of multi-class classification

2.2.3 Cascaded modeling

The goal of the cascaded model is to enhance the model prediction when predicting the failure types and classify the network into normal and abnormal failure types. This can be done by applying the best model from binary class classification and multiclass classification which is XGboost model.

The main idea of cascaded modeling is ensemble learning based on concatenation between several classifiers using all of the information collected from the output of a classifier and added it to the next classifier model.

So, the binary classifier model using XGBOOST had reached some results that have been added to the next classifier for the multiclass classification model. The following figure shows the structure of the cascaded model with the outputs marked with red numbers:

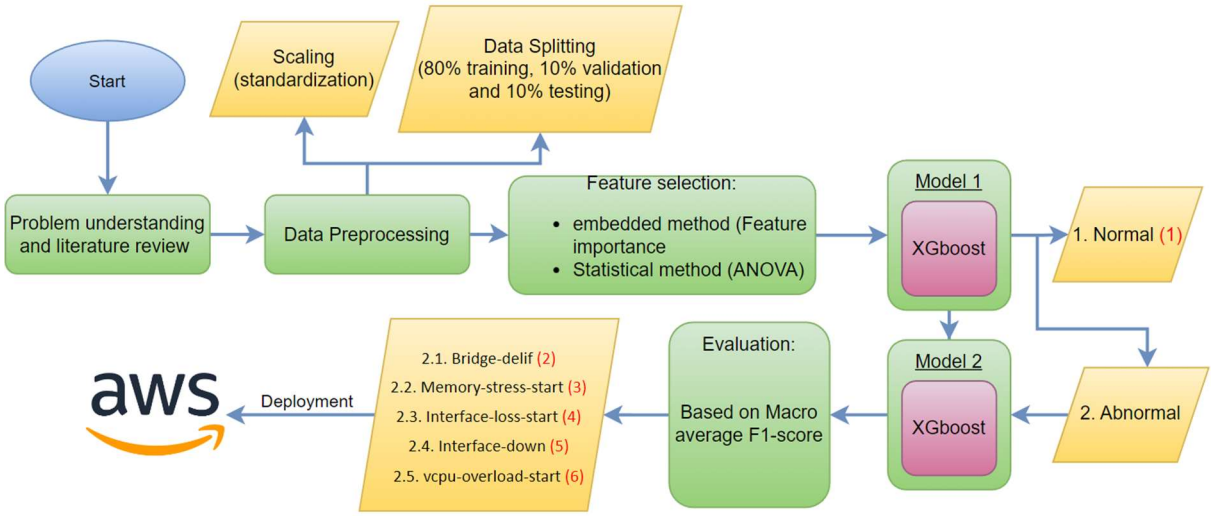

Figure 13 Flow chart of cascaded modeling

2.2.4 One vs all modeling

The goal of the one vs all method is to enhance the Cascaded model's prediction using failure data to try and improve the prediction of the failure types that were known to be overlapping classes from previous EDA and would be confirmed later in clustering part.

One vs all modeling uses a binary classification model specifically for ea ch class and consider the possibility coming from each model when it comes to aggregating the predictions to know the final classification of all data points combined. The following figure shows the structure of one vs all modeling:

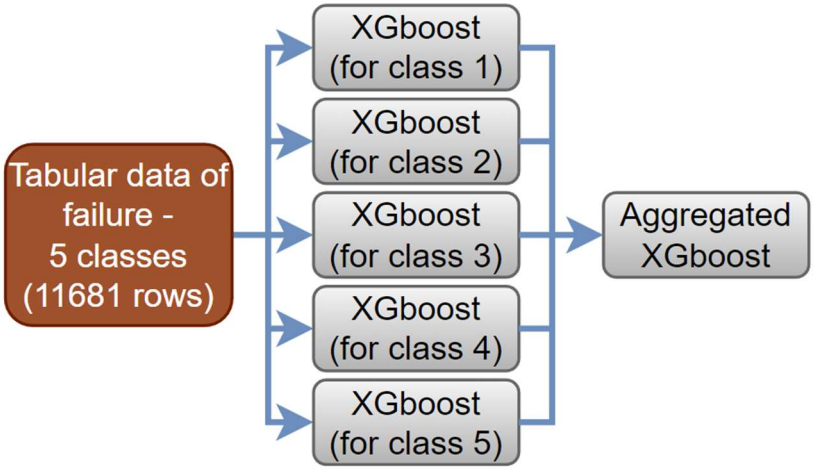

Figure 14 Flow chart of one vs all modeling

2.2.5 Clustering modeling

The following figure illustrates the clustering system architecture with the outputs marked with red numbers as shown:

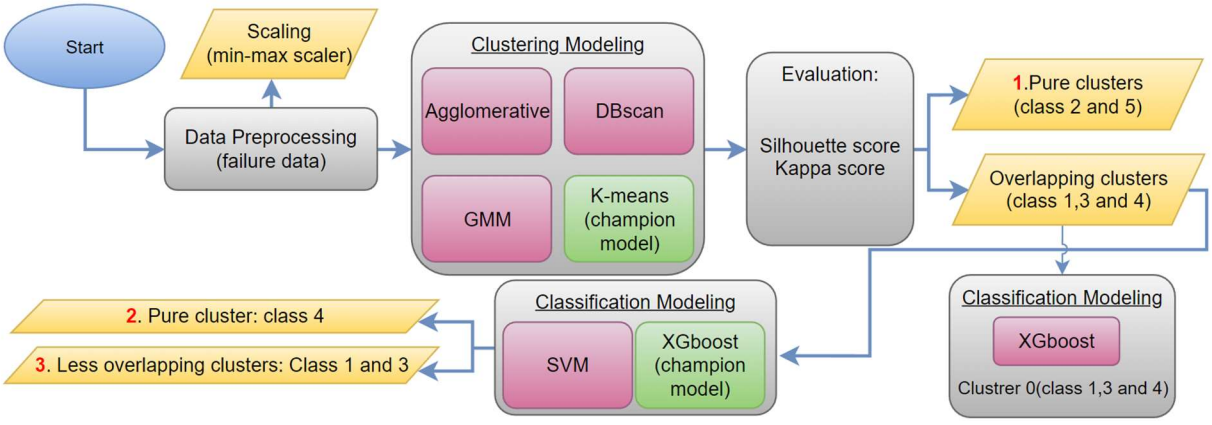

Figure 15 Flow chart of clustering modeling

A- Scaling techniques:

Different scaling techniques have been tried like Robust scaler which removes the median and scales the data according to the quantile ranges. As a result of this scaling technique, a huge overlapping between clusters was noticed among all the classes which was not beneficial for this scenario, so, min-max normalization technique was utilised which had slightly separated the overlapping between classes two and five.

B- Clustering modeling:

Applying Clustering techniques through different approaches that results into pure or overlapping clusters using the abnormal data of five failure scenarios as input, the best number of clusters for each model was deducted based on the highest silhouette and kappa scores, the champion model of each clustering technique was then chose based on the comparison of cluster distribution of training and testing datasets in each clustering model.

C- Classification approach:

Two different classification approaches were used based on the champion model of the clustering technique and these approaches were using two different algorithms which are SVM model and K-means model. K-means had succeeded to isolate the highest number of pure classes, first, a classification model was built with only the overlapping clusters from the cluster distribution of the champion clustering model, secondly, it was figured out which classes represent the distribution of impure clusters and a classification model was built with all the data of these classes as training data to the model.

2.2.5.1 Agglomerative model

This is the agglomerative modeling architecture through different approaches showing the best model in green which is based on Cluster distribution.

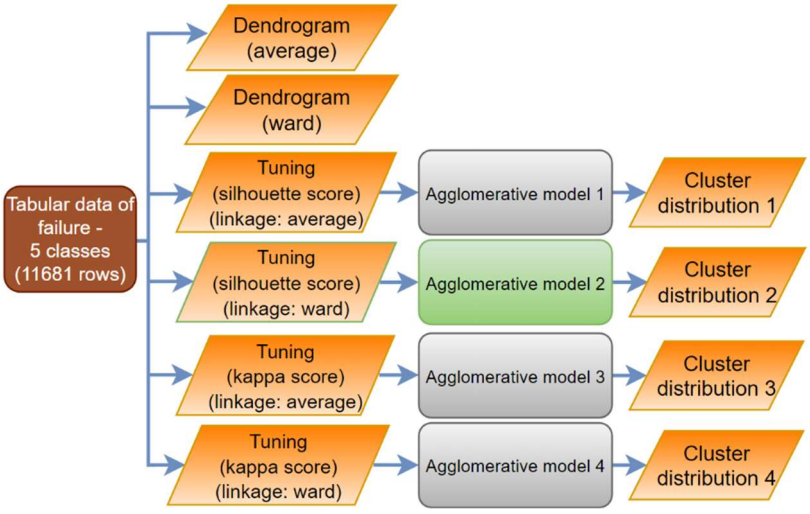

Figure 16 Flow chart of agglomerative modeling

A. Input:

All abnormal data of five failure scenarios as input which are 11681 rows.

B. Cluster modeling:

Applying agglomerative clustering modeling through different approaches that resulted pure or overlapping clusters with the abnormal data of five failure scenarios as input, the best number of clusters of each agglomerative model was deducted based on the highest silhouette and kappa score, and also the two types of linkages (ward-average),we chose the champion model of each agglomerative clustering technique based on comparing the cluster distribution of training and testing datasets in each of agglomerative clustering models.

C. Validation:

Comparing the cluster distribution of train and test data for each of the four agglomerative models.

2.2.5.2 DBSCAN model

This is the DBSCAN modeling architecture through two different approaches both after applying Bayesian optimization, one of them used silhouette score as a reference and the other used kappa score, both followed by DBSCAN model and the results were compared using the output cluster distributions. The following figure shows the structure of DBSCAN modeling:

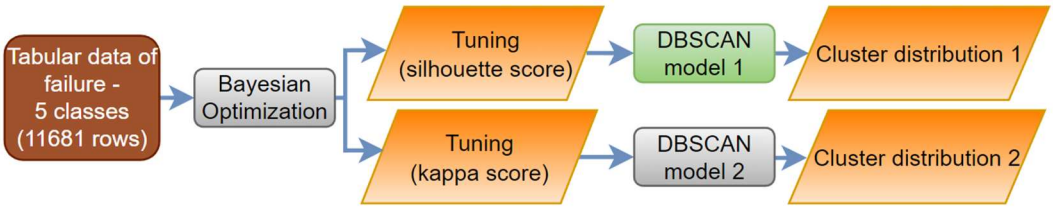

Figure 17 Flow chart of DBSCAN modeling

A. Input:

All abnormal data of five failure scenarios as input which are 11681 rows.

B. Modeling and Evaluation:

Two objective functions were designed to be used by Bayesian optimization which is an advanced hyperparameter tuning technique. The first objective function tried to maximize the silhouette score based on the best epsilon and minimum sample parameters of DBSCAN model. The second objective function tried to maximize the Kappa score based on the best epsilon and minimum sample parameters of DBSCAN model.

C. Validation:

Comparing the cluster distribution of train and test data for each of the two DBSCAN models one for best silhouette and one for best kappa score.

2.2.5.3 Gaussian mixture model (GMM)

This is the GMM modeling architecture through different techniques, one of them used silhouette score as a reference and the other used kappa score, both followed by GMM model. The following figure shows the structure of GMM modeling:

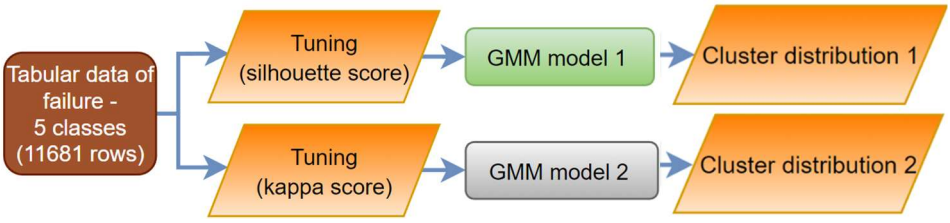

Figure 18 Flow chart of GMM modeling

A. Input:

All abnormal data of five failure scenarios as input which are 11681 rows.

B. Cluster Modeling:

Applying GMM cluster modeling with the abnormal data of five failure scenarios as input, range of values have been tried using GMM and silhouette and kappa scores were used to deduce the number of clusters and GMM components. The champion model of each GMM clustering technique has been chose based on cluster distribution that had the least number of GMM components.

C. Validation:

Comparing the cluster distribution of training and testing datasets for each of the two GMM models.

2.2.5.4 K-means model

In K-means modeling, the architecture was built through two different approaches, one of them used silhouette score as a reference and the other used kappa score, both followed by K-means model. The following figure shows the structure of K-means modeling:

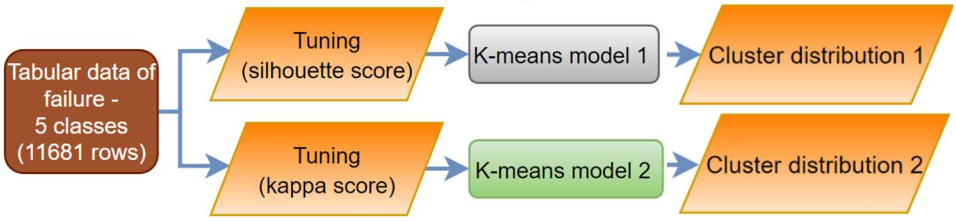

Figure 19 Flow chart of K-means modeling

A. Input:

All abnormal data of five failure scenarios as input which are 11681 rows.

B. Cluster Modeling:

Applying k-means cluster modeling with the abnormal data of five failure scenarios as input, a range of values using k-means and used silhouette and kappa scores have been used to deduce the number of clusters, the champion model of each k-means clustering technique was

chosen based on k-number that separates the largest number of classes on both training and testing datasets.

C. Validation:

Comparing the cluster distribution of training and testing datasets for each of the two k-means models.

2.2.6 Beamforming selection

A promising method for overcoming communication is mm-wave communications (30-300GHz). Sub-6GHz spectrum bottlenecks caused by overuse and higher frequency spectrum can be tapped by Key performance indicators (KPIs) for 5G appear to be within reach thanks to mm-waves, which open up a large and unused frequency band. However, there are a lot of technical issues with mm-wave communications including threats resulting from hostile physical phenomena at high frequencies, including: increased atmospheric attenuation, penetration losses, severe blockage, and many others. More antennas can be packed into a given space thanks to shorter wavelengths. Shifting communication toward massive multi-input multi-output (MIMO) connections, more at the transmitter and receiver ends. Eventually, higher beamforming gains result from this technical advantage. Mm-waves become a workable alternative for future generation by making up for the propagation losses. communications. However, there are still plenty of unsolved problems to solve before the can be released. One significant way that mm-wave communications can be used to their full potential is by minimizing the overhead that is introduced. by the beam selection (or alignment) processes.
The objective is selecting the transmitter and receiver beam combinations that maximize the beamformed channel.

A simulated dataset that represents a vehicle to infrastructure communication using mm-wave was provided which uses a LiDAR. The covariates were the latter and roadside cameras to forecast the beamformed channels gains applied during the beam alignment stage. The suggested solution primarily uses LiDAR receivers and maps out of the two data modalities: actual data from transmitter antennas and information gathered by LiDAR, both are used in maximizing the data rate between transmitter and receiver antennas. due to the fact that it contains locations of, becomes a very important piece of information in the beam selection process. The receiver's vicinity could have reflectors or obstructions. A CNN model was used to get the top k-pairs between transmitters and receivers. The following figure shows the structure of the beamforming selection including the deployment on AWS:

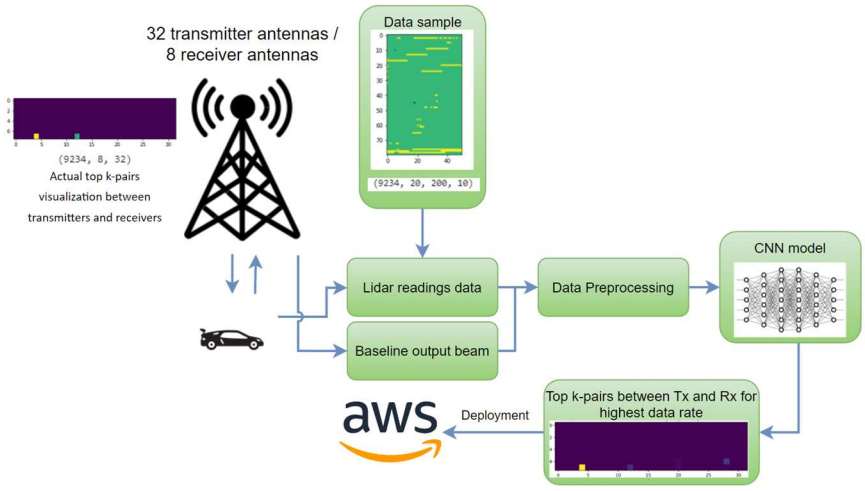

Figure 20 Flow chart of beamforming selection

In this problem we have 32 transmitter antennas on the base station and 8 receivers. The aim is to maximize the gain to get higher data rate by finding the best pairs between transmitters and receivers. There is input data from two LiDARs and output beams from transmitters the next table will explain all the detailed shape about the dataset.

Data Set	Train	Validation	Test
LiDAR Data	(9234, 20, 200, 10)	(1960, 20, 200, 10)	(9638, 20, 200, 10)
Output Beam	(9234, 8, 32)	(1960, 8, 32)	(9638, 8, 32)

Table 2 Shapes of the used data in beamforming selection

Another approach has been used which is federated learning, is a technique for training AI models without letting anyone else access or see your data, providing a way to make data available for new AI applications.

Why Federated Learning is important in 5G networks?

Recently, the idea of federated learning (FL) has been put forth as a way to preserve participant privacy while developing machine learning models on distributed training datasets that are locally maintained and stored on various devices in 5G networks. To update a global model in FL, the central aggregator combines local updates uploaded by participants. The poisoning attack and the membership inference attack are two serious security risks, though. The failure to construct global models or the privacy leakage of FL models may be caused by these attacks carried out by malicious or unreliable participants. As a result, FL must develop security measures of defense. In this article, we suggest a secure FL framework built on blockchain that uses smart contracts to keep malicious or unreliable participants out of FL. By automatically executing smart contracts to prevent poisoning attacks, the central aggregator can identify

malicious and unreliable participants. Additionally, we defend against membership inference attacks on smart contracts by using local differential privacy techniques. The proposed framework can successfully fend off poisoning and membership inference attacks, according to numerical findings, enhancing the security of federated learning in 5G networks.

In the next figure we will introduce the design of implementing Federated learning

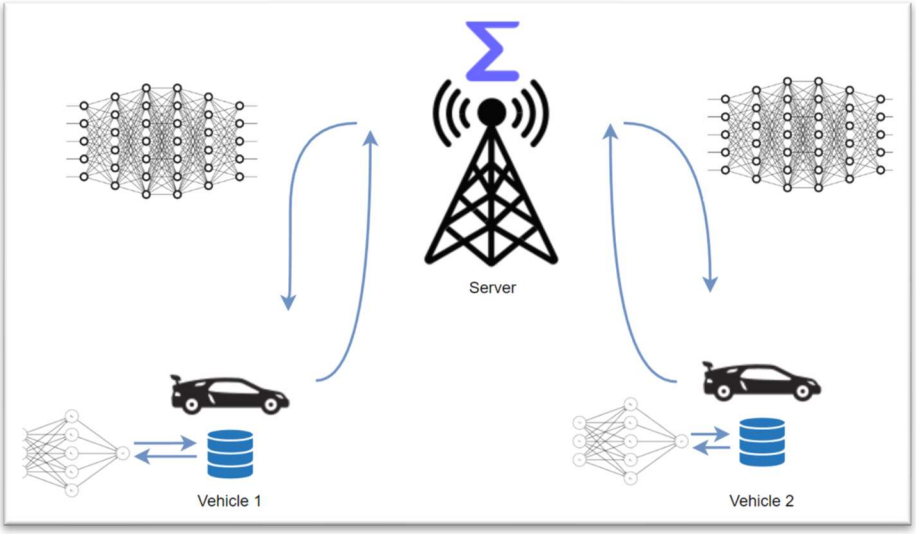

Figure 21 Architecture of federated learning

We use train data for training a local model on the vehicle 1 server and get the weight of the training model also use test data for training a local model on the vehicle 2 server we aggregate the final weight of both training model on centralized server this concept will increase the trustworthy of the data for each vehicle.

2.3 Implementation

2.3.1 Binary classification

Binary classification data for network failure contains 849994 rows and 108 columns, and the data label is 0 and 1 (normal and abnormal). Dealing with null values is very important, and also feature selection because of the large amount of data and columns.

2.3.1.1 Data preprocessing

Two techniques were followed in data preprocessing and they are:

1- Scaling and removing outliers: standardization is a way to perform data scaling to center the values around the mean using standard deviation.

2- Data splitting: the data has been split into training, testing and validation data, with percentages of 80% for training, 10% for testing and 10% for validation.

2.3.1.2 Feature selection

This is an important stage to select the most important features from the data, so two methods have been applied for feature selection which are analysis of variance (ANOVA) feature selection method and feature importance based on an XGboost model.

1- ANOVA feature selection:

It is a statistical method to select the most important features from the data based on the data distribution. The following figure represents the achieved accuracies using different number of features using ANOVA applied with a XGboost model using test dataset:

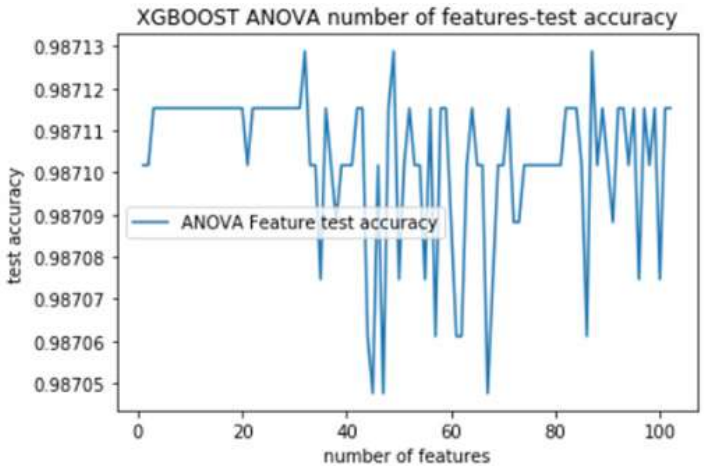

Figure 22 Accuracies of different number of features using ANOVA approach on XGboost with test dataset

2- Feature importance based on XGboost model:

Gradient boosting techniques can be used to retrieve the importance scores for each attribute from the dataset, the higher scores indicate that the feature is so important. Here XGBOOST model has been applied to select the most important features as shown in the figure:

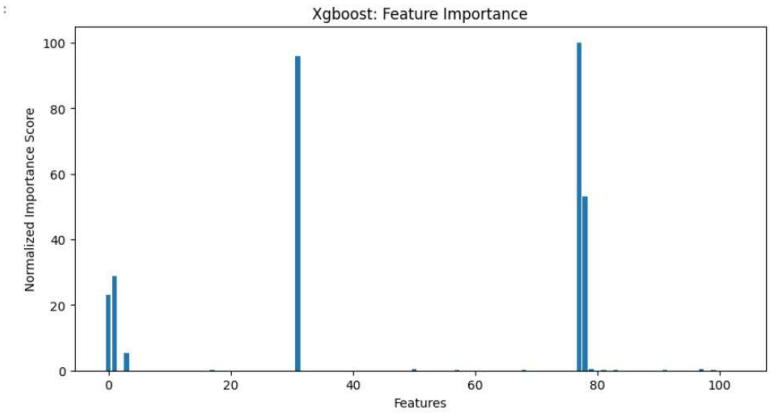

Figure 23 Accuracies of different number of features according to gradient boosting applied on XGboost

2.3.1.3 Performing exploratory data analysis

Exploratory data analysis (EDA) is an important stage to analyze the data and visualize them to explore the nature of the data. According to the large amount of the data, EDA has been performed on the most important features, Pearson ranking correlation and spearman correlation heatmap have been applied to explore the data.

1- Pearson ranking on the best 22 features:

Pearson ranking represents the strength and the direction of the relationship between two variables. Values of Pearson ranking are between 1 and -1. If the rank is 1, it means that the two features are highly correlated. If it's equal to zero, it means that there is no correlation. If it's equal to negative one, it's a negative correlation. Pearson ranking is shown in figure (23).

2- Spearman correlation heatmap:

Also known as rank correlation, it examines the relationships between IQ scores and test scores. It can be used to check whether the relationship between two variables if linear or nonlinear. The following two figures show the Pearson ranking and spearman correlation between the selected 22 features. Spearman correlation heatmap is shown in figure (24).

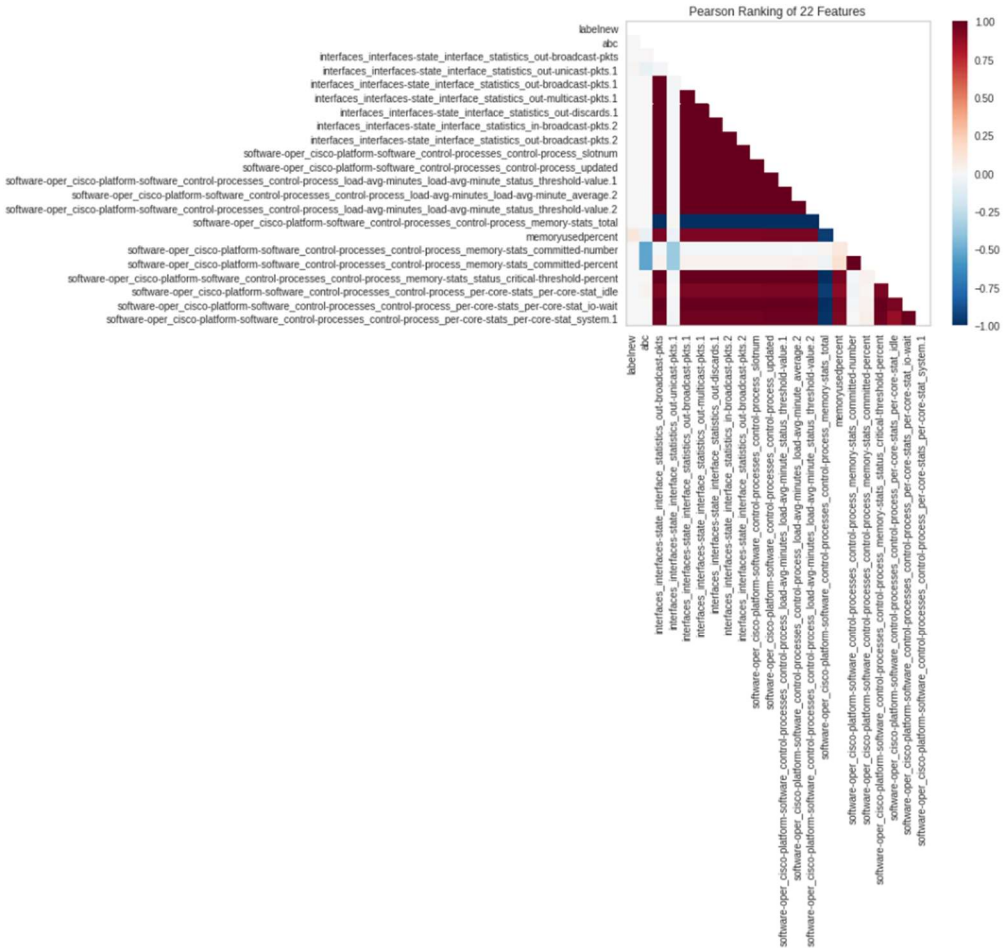

Figure 24 Pearson ranking on the best 22 features in binary classification

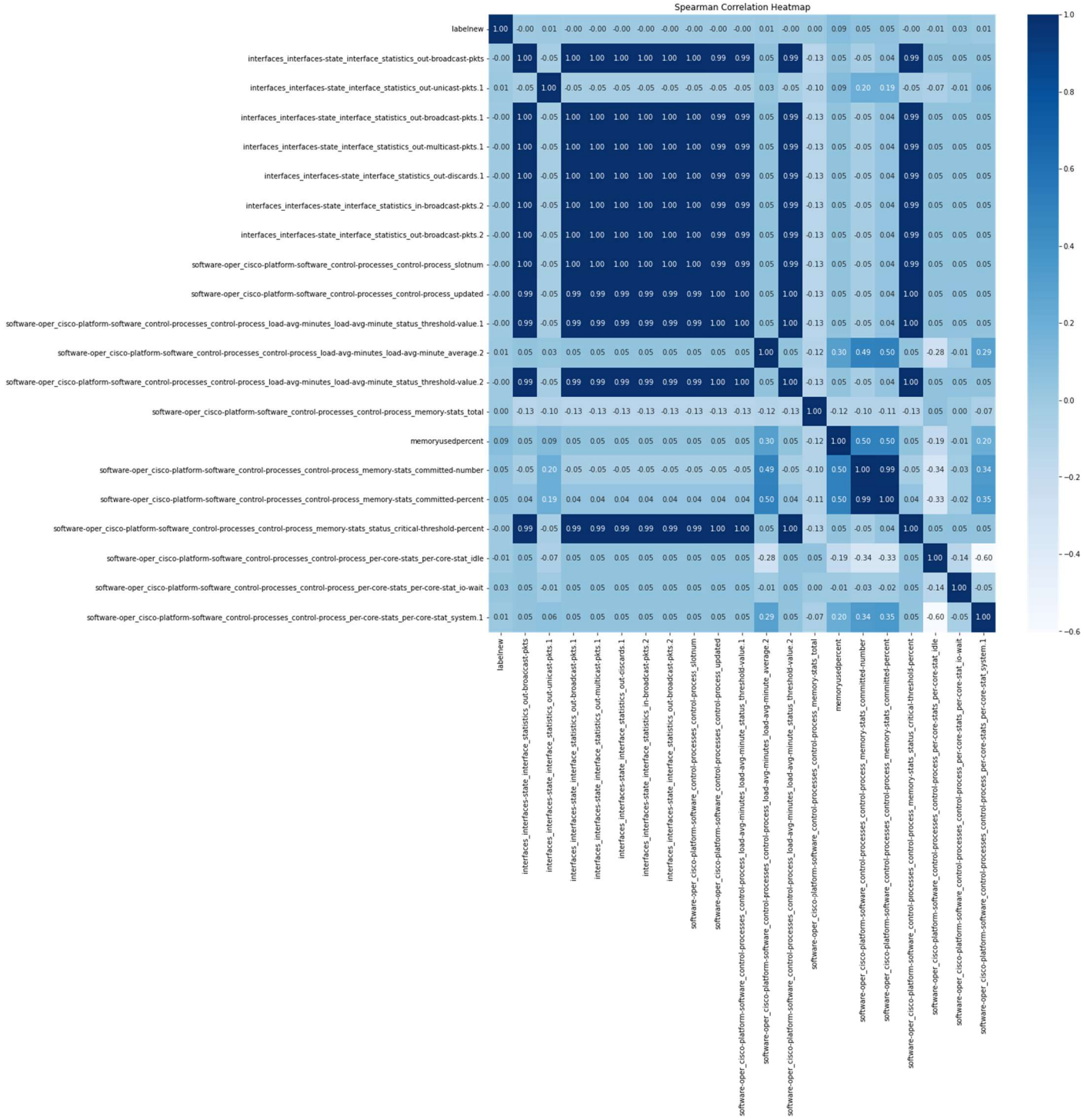

Figure 25 Spearman correlation heatmap in binary classification

2.3.1.4 Applying XGboost

XGboost is one of the gradient boosting techniques for supervised learning in machine learning. The following table shows the values of used model parameters in training process:

eta	max_depth	min_child_weight	subsample	colsample_bytree	seed	eval_metric	scale_pos_weight
0.3	15	6	0.9	0.7	0	auc	0.09

Table 3 Values of used parameters of XGboost in binary classification

- Eta is the learning rate of XGboost algorithm, the default value is 0.3.
- Max_depth of the tree, the default value is 6. By increasing the value of max_depth, the complexity increases. Here the max_depth value has been set to 15.
- Min_child_weight: the default value is 1. The more of min_child_weight, the more conservative algorithm. Here min_child_weight has been set to 6.
- Subsample: is the ratio of the training instances. The default value is 1, here the subsample value has been set to 0.9 to prevent overfitting.
- Colsample_bytree: is a parameter for subsampling the columns. The default value is 1. Here the value has been set to 0.7.
- Seed: is the random number seed. The default value is 0.
- Eval_metric: is an evaluation metric for validation data. Here area under the curve (auc) as the problem is a binary classification problem.
- Scale_pos_weight: to control the balance between positive and negative weights. The default value is 1. Here the value has been set to 0.09.

2.3.2 Multi-class classification

After applying binary classification, studying the reasons of network failure, it was reasonable to add more complexity and challenge to the undergoing problem by making it a multi-class classification problem, Standardization also applied to center the values around the mean using standard deviation, and based on the domain expert the most important 33 features are shown in the following figure:

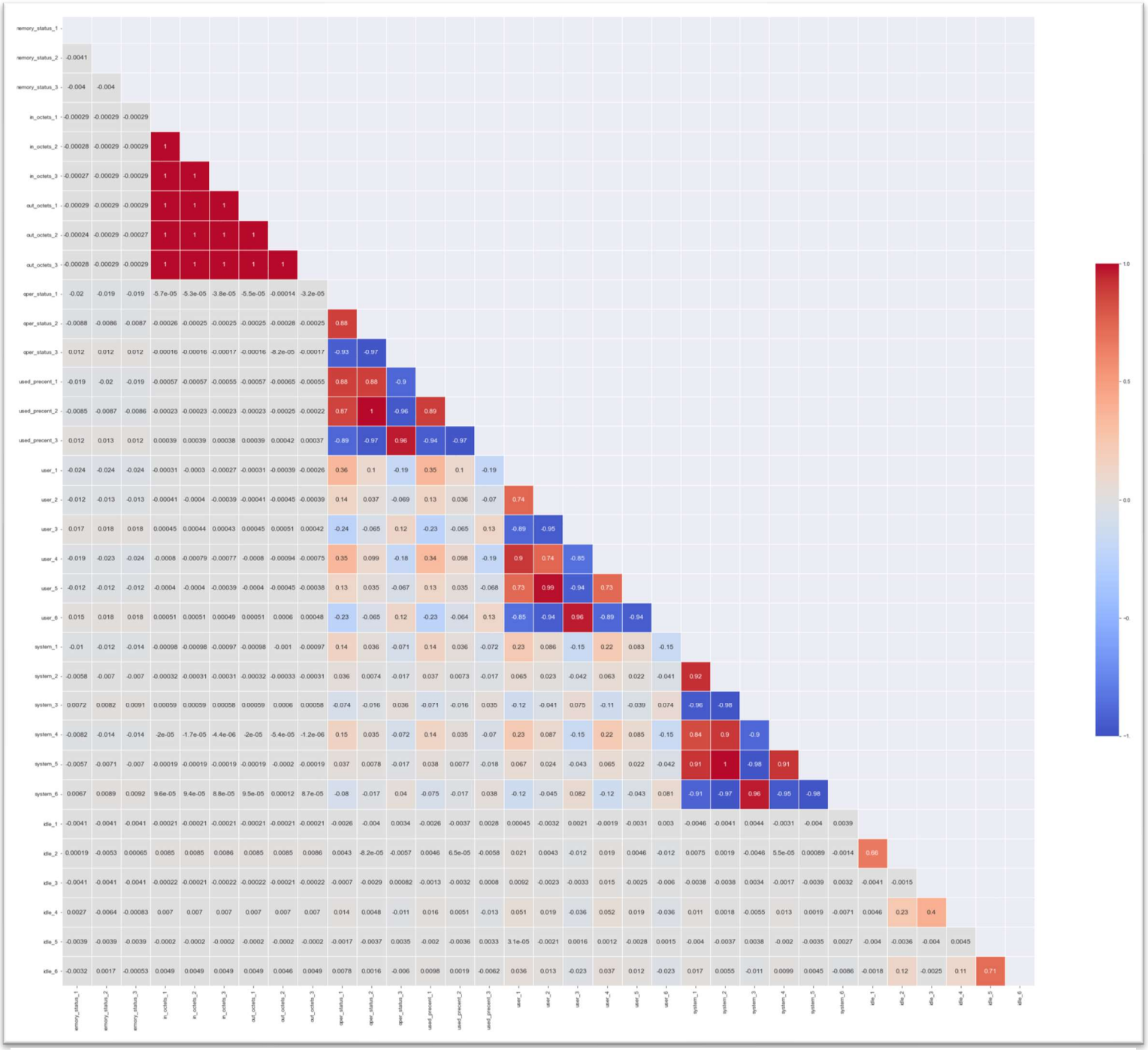

Figure 26 Pearson ranking on the best 33 features in multi-class classification

1- Apply logistic regression model:

Logistic regression model is one of supervised techniques of machine learning that provides the probability of predicting the class label and to ensure that our data was linearly separable or not. Parameters used to train logistic regression as a baseline model:

multi_class	random_state
auto	42

Table 4 Parameters of LR as baseline model in multi-class classification

2- Applying naïve bayes model:

Naïve bayes model is one of the supervised techniques in machine learning which is based on the Bayes theorem that has an effective probabilistic of class labels and to ensure that the data was uncorrelated.

The kernel used here is GaussianNB that can predict continuous values.

3- Applying Random Forest model:

Random Forest model is one of the supervised learning techniques that is based on tree. Testing takes 1:07 minutes.

Random_state	42

Table 5 Random state of Random Forest model in multi-class classification

4- Applying XGboost model:

XGBOOST model is one of the gradient boosting techniques for supervised learning in machine learning. Testing takes 39 seconds.

Random_state	42

Table 6 Random state of XGboost of gradient boosting in multi-class classification

5- Apply hyperparameter tuning on XGBOOST model:

As the results of XGBOOST model as a baseline, it has been chosen for tuning.

Parameters of XGBOOST model:

eta	max_depth	min_child_weight	subsample	colsample_bytree	seed	eval_metric
0.3	15	6	0.9	0.7	0	mlogloss

Table 7 Values of used parameters of XGboost in multi-class classification

2.3.3 Cascaded modeling

XGBOOST was tuned in the case of binary class classification and multiclass classification.
Parameters used in tuning binary class classification:

eta	max_depth	min_child_weight	subsample	colsample_bytree	seed	eval_metric	scale_pos_weight
0.3	15	6	0.9	0.7	0	auc	0.09

Table 8 Values of used parameters of binary XGboost in cascaded modeling

Parameters used in tuning multiclass classification:

eta	max_depth	min_child_weight	subsample	colsample_bytree	seed	eval_metric
0.3	15	6	0.9	0.7	0	mlogloss

Table 9 Values of used parameters of multi-class XGboost in cascaded modeling

3.4 One vs all modeling

As a way of trying to improve the result of the cascaded model using failure data, a binary XGboost model has been created for each type of failure with another XGboost model for aggregating of all of them. The system works as follows:

1- **Model specified for class one**: it was done by using binary labels, which means specifying the label for class one to one and the other four types of failure were given a label of zero.
2- **Model specified for class two**: it was done by using binary labels, which means specifying the label for class two to one and the other four types of failure were given a label of zero.
3- **Model specified for class three**: it was done by using binary labels, which means specifying the label for class three to one and the other four types of failure were given a label of zero.
4- **Model specified for class four**: it was done by using binary labels, which means specifying the label for class four to one and the other four types of failure were given a label of zero.
5- **Model specified for class five**: it was done by using binary labels, which means specifying the label for class five to one and the other four types of failure were given a label of zero.

Aggregation:

The five models were used to predict the probability of each point with the labels of one or zero, then the probabilities of one were collected from all the five models and an argmax function was used to determine the index of the highest probability that a certain datapoint belongs to which refers to a specific class.

2.3.5 Clustering modeling

2.3.5.1 Agglomerative model

Agglomerative clustering: is a hierarchical clustering that is used to group objects in clusters based on their similarity. The result of the agglomerative clustering is a tree-based representation of the objects which called dendrogram.

Dendrogram has been used to see the best number of clusters using both average and ward linkages.

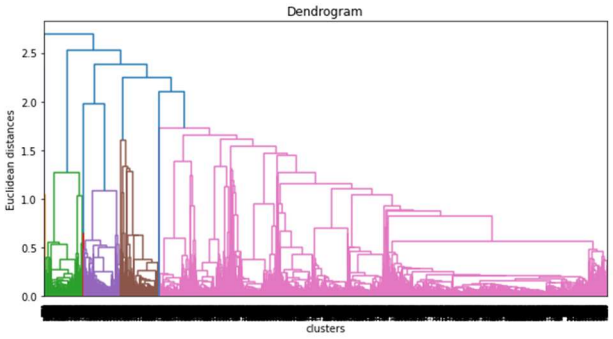

Figure 27 Average linkage of agglomerative clustering

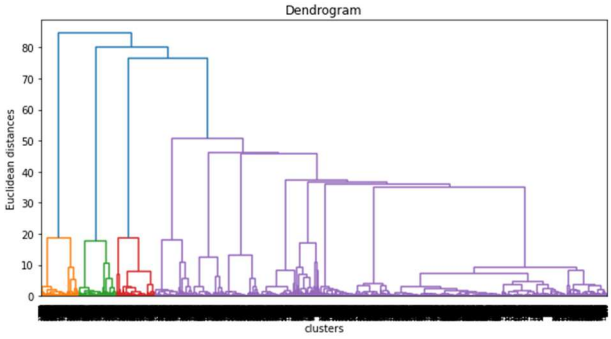

Figure 28 Ward linkage of agglomerative clustering

To reach the best results, two experiments have been applied, one for the best silhouette score and the other for the best kappa score.

The first experiment was based on the highest silhouette score. The main idea is tuning agglomerative clustering model using ward linkage method which gave the highest silhouette score. The following figure indicates that, the best number of clusters equaled 11:

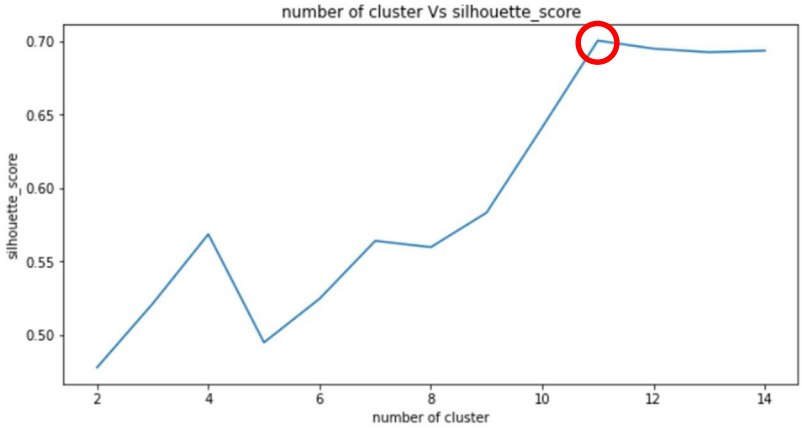

Figure 29 Number of clusters vs silhouette score using ward linkage

The second experiment was based on the highest kappa score by tuning agglomerative clustering model using average linkage method. The following graph shows that the best number of clusters is 11:

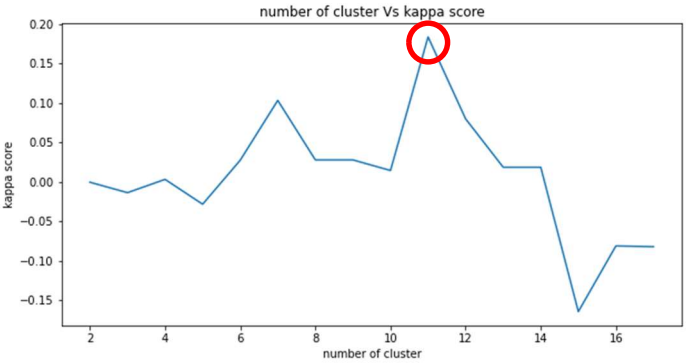

Figure 30 Number of clusters vs kappa score using average linkage

2.3.5.2 DBSCAN model

DBSCAN is one of the unsupervised clustering techniques that can divide the data points into specific groups that contain the similar points. Two experiments have been applied by Bayesian optimization one to get the highest silhouette score and the other is to get the highest kappa score. The first experiment was tuning DBSCAN using Bayesian optimization to get the best value for the parameter epsilon and minimum samples based on the highest silhouette score.

After tuning, the best number of clusters was 11, so a DBSCAN model has been built with the best hyperparameters shown in table (10). The following figure shows the best hyperparameters for DBSCAN model at the highest silhouette score of 0.690042:

iter	target	eps	min_sa...
114	0.6866	0.8386	45.33
115	0.6892	0.6067	47.03
116	0.6887	0.6485	65.12
117	0.5675	1.217	68.38
118	0.685	0.9818	11.98
119	0.5666	1.066	42.08
120	0.5666	1.074	61.81
121	0.6865	0.8962	32.34
122	0.6864	0.9112	62.68
123	0.5666	1.153	8.828
124	0.5666	1.033	74.33
125	0.6865	0.6846	29.57
126	0.5675	1.227	72.51
127	-2.0	4.979	51.0
128	0.686	0.8216	75.62
129	0.6866	0.8445	52.61
130	0.5663	1.014	53.52
131	0.5666	1.155	56.69
132	0.5666	1.171	57.85
133	0.686	0.9544	59.4
134	0.5666	1.051	60.34
135	0.5666	1.156	49.64
136	0.6863	0.9767	55.67
137	0.5666	1.018	54.78
138	0.457	0.1079	37.16
139	0.6865	0.7163	37.82
140	0.5666	1.171	58.73
141	0.5666	1.076	37.25
142	0.6867	0.8874	38.98
143	0.6902	0.5862	71.43
144	0.6885	0.6398	56.09

Figure 31 The best hyperparameters for DBSCAN to be used with silhouette score

n	Min_sample	Epsilon
250	71	0.586222874

Table 10 Best hyperparameters of DBSCAN at the highest silhouette score

The second experiment was tuning DBSCAN clustering model using Bayesian optimization to get the best parameter epsilon and minimum samples based on the highest kappa score. After tuning, the best number of clusters was 17. Nevertheless, that DBSCAN model has been built with the best hyperparameters shown in table (11). The following figure shows the best hyperparameters for DBSCAN model at the highest kappa score of 0.3078:

iter	target	eps	min_sa...
1	0.02014	1.349	8.511
2	-2.0	2.437	61.83
3	-2.0	3.744	75.37
4	0.02016	1.211	11.77
5	-2.0	4.994	10.51
6	0.3078	0.2929	10.01
7	0.1755	0.1	3.26
8	-0.1117	0.1	18.74
9	-0.114	0.1	27.33
10	-2.0	5.0	33.42
11	-2.0	4.995	22.9
12	-2.0	4.436	2.098
13	-0.05201	0.1	5.985
14	0.1562	0.1926	15.04
15	0.04376	0.1	47.26
16	-2.0	5.0	48.77
17	0.04447	0.1	42.88
18	0.05798	0.2006	38.71
19	-2.0	4.316	41.02
20	0.04345	0.1	52.85
21	0.04536	0.1	31.59
22	0.04454	0.1	35.27
23	-0.004643	0.1664	79.89
24	-0.05777	0.199	56.19
25	-2.0	5.0	55.56
26	0.04274	0.1	69.04

Figure 32 The best hyperparameters for DBSCAN to be used with kappa score

n	Min_sample	Epsilon
350	10	0.292886058969

Table 11 Best hyperparameters of DBSCAN at the highest kappa score

2.3.5.3 GMM model

A range of values must be taken to get the best number of GMM components using silhouette and kappa scores. The first experiment was tuning the GMM model using a number of components to get the highest silhouette score. The following graph shows that the best number of components was 11 using the silhouette scores:

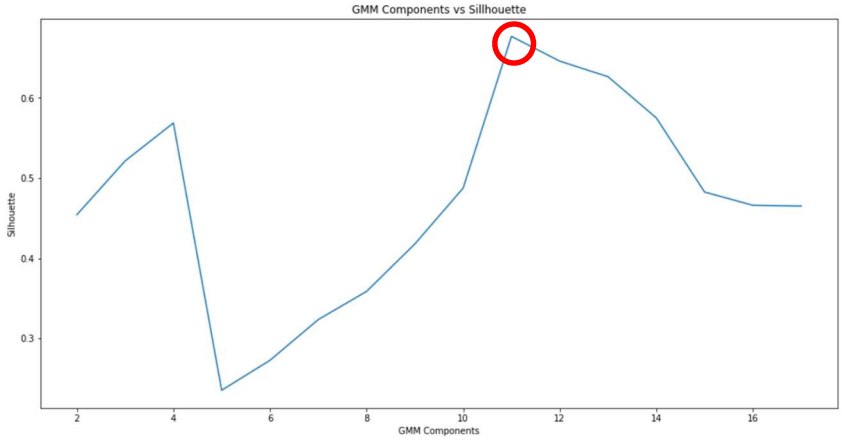

Figure 33 GMM components vs silhouette scores

The second experiment was tuning the GMM model using the number of components that gives the highest Kappa score. The following figure shows that, the best number of components was 15 using the kappa scores:

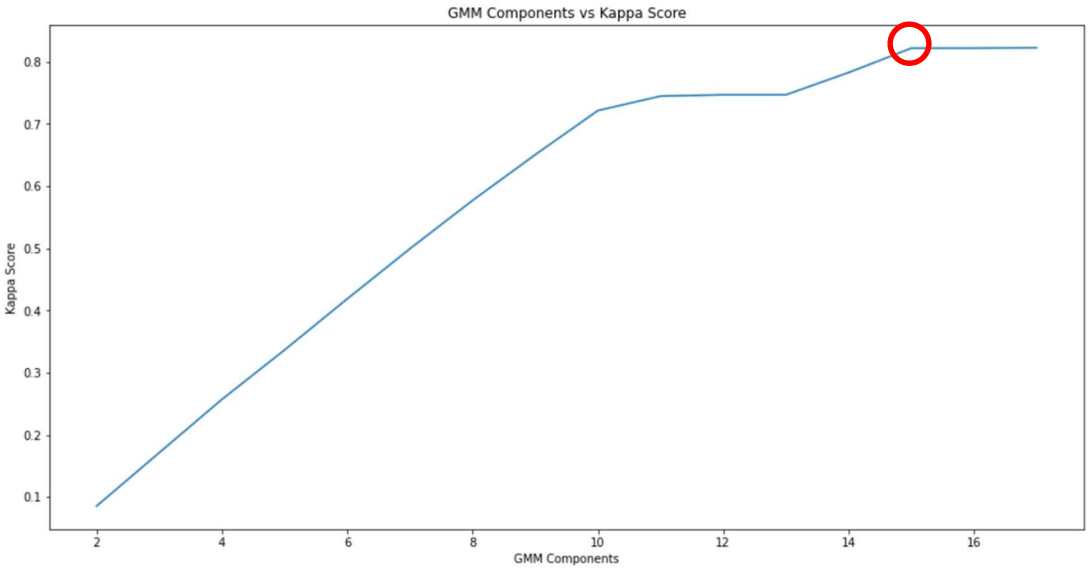

Figure 34 GMM components vs kappa scores

2.3.5.4 K-means model

The first experiment was tuning K-means clustering model using the number of clusters that give the highest silhouette score. This experiment had indicated that the best number of clusters was when k = 11 as the following figure shows:

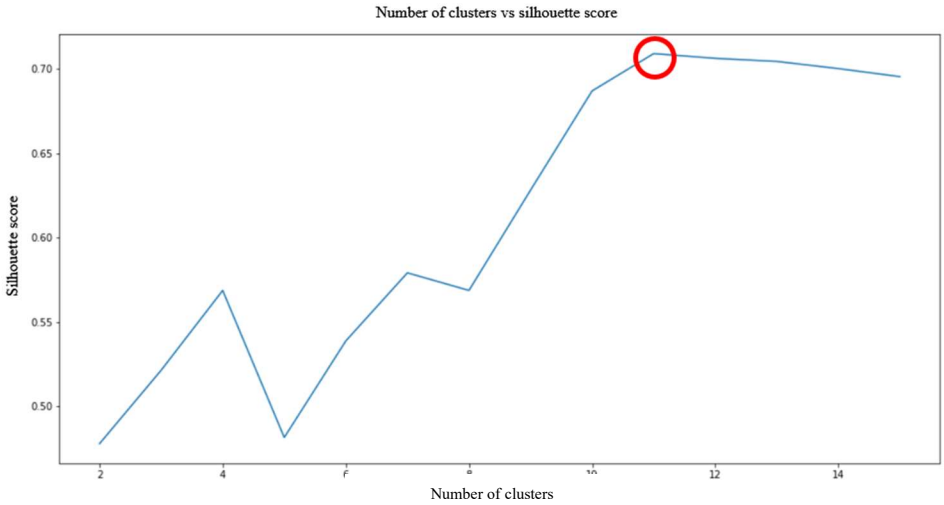

Figure 35 Number of clusters vs silhouette scores for k-means model

The second experiment was tuning k-means model using the number of clusters that give the highest kappa score. This experiment had indicated that the best number of clusters was when k = 10 as the following figure shows:

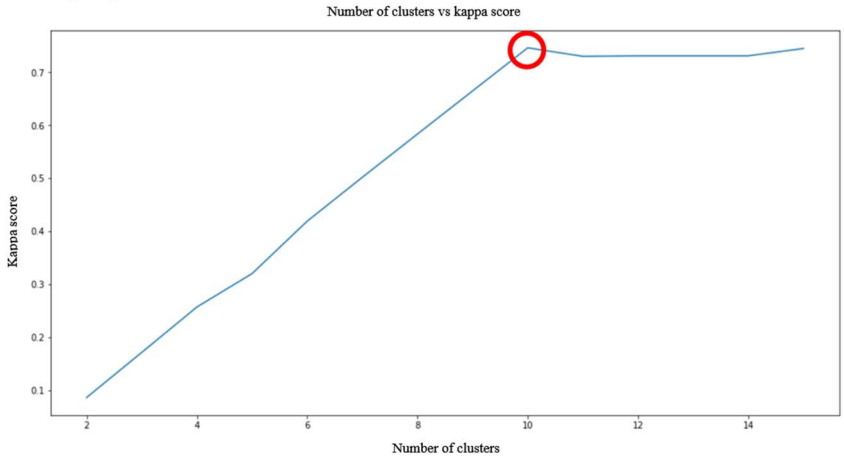

Figure 36 Number of clusters vs kappa scores for k-means model

The third experiment was classification using k-means output of cluster zero based method. As cluster zero is the only impure cluster on the test distribution, only the data points of cluster zero on the training dataset has been used to create a model specifically for this impure cluster, and this model was also tested on the testing data points belonging to cluster zero.

The fourth experiment was classification using k-means output of classes 1, 3 and 4 based Method. As k-means cluster distribution and cluster numbers are consistent across training and testing datasets, and the algorithm can perfectly isolate classes 2 and 5, all the data points of classes 1, 3 and 4 have been used with the training dataset to create a model specifically those, and this model has been tested on the testing data points belonging to those 3 classes, two models were tried in this experiment, SVM and XGboost.

The fifth experiment was classification using k-means output for all of the classes. Combining all of the results of k-means clustering and classify them with XGboost model to reach better results and be able to identify network failure types.

2.3.6 Beamforming selection

After loading the data, preprocessing on LiDAR data has been applied to convert LiDAR data from 3D to 2D, then, all the data have been fed to Convolution neural network (CNN) to extract the most important features to predict the top k-pairs that maximize the gain which gives the highest data rate.

The figure (41) shows in detail the CNN model architecture of experiment one.

In the mentioned architecture, batch normalization has been used. To prevent overfitting ReLU activation functions have been used after each convolution layer, these activation functions remove all negative values after applying fully connected layers with a dense layer of 512 neurons and an output layer of 256 neurons with softmax activation function that results the probability between the top k-pairs between transmitter and receiver antennas.

When the probability is close to one, it means that it is the best match between transmitter and receiver antennas which gives the maximum gain (highest data rate). The next table shows the training parameters for the first experiment:

Optimizer	Schedular	Callback	Loss function	Batch size	Epochs
SGD(lr=1e-3)	0.690042	tf.keras.callbacks.LearningRateScheduler	lambda y_true, y_pred: - tf.reduce_sum(tf.reduce_mean(y_true[y_pred>0] * tf.math.log(y_pred[y_pred>0]), axis=0))	16	25

Table 12 Parameters used in the first experiment of beamforming selection

In the second experiment that is showed in figure (40), batch normalization was used to prevent overfitting ReLU activation functions have been used after each convolutional layer to remove all negative values. Fully connected layers have been applied with dense layers of 16 neurons for each and an output layer of 256 neurons with softmax activation function that gives the between the top k-pairs between transmitter and receiver antennas.

The figure (40) shows in detail the CNN model architecture of second experiment. The following table shows the used parameters in the second experiment:

Optimizer	Schedular	Callback	Loss function	Batch size	Epochs
RMSPROP (lr=1e-3)	0.690042	tf.keras.callbacks.LearningRateScheduler	lambda y_true, y_pred: - tf.reduce_sum(tf.reduce_mean(y_true[y_pred>0] * tf.math.log(y_pred[y_pred>0]), axis=0))	32	25

Table 13 Parameters used in the second experiment of beamforming selection

In the second experiment that is showed in figure (39), batch normalization was used to prevent overfitting ReLU activation functions have been used after each convolutional layer to remove all negative values. Fully connected layers have been applied with dense layers of 16 neurons for each and an output layer of 256 neurons with softmax activation function that gives the between the top k-pairs between transmitter and receiver antennas.

The figure (39) shows in detail the CNN model architecture of second experiment. The following table shows the used parameters in the second experiment:

Optimizer	Schedular	Callback	Loss function	Batch size	Epochs
Adam (lr=1e3, epsilon=1e-8)	0.690042	tf.keras.callbacks.LearningRateScheduler	lambda y_true, y_pred: - tf.reduce_sum(tf.reduce_mean(y_true[y_pred>0] * tf.math.log(y_pred[y_pred>0]), axis=0))	16	25

Table 14 Parameters used in the third experiment of beamforming selection

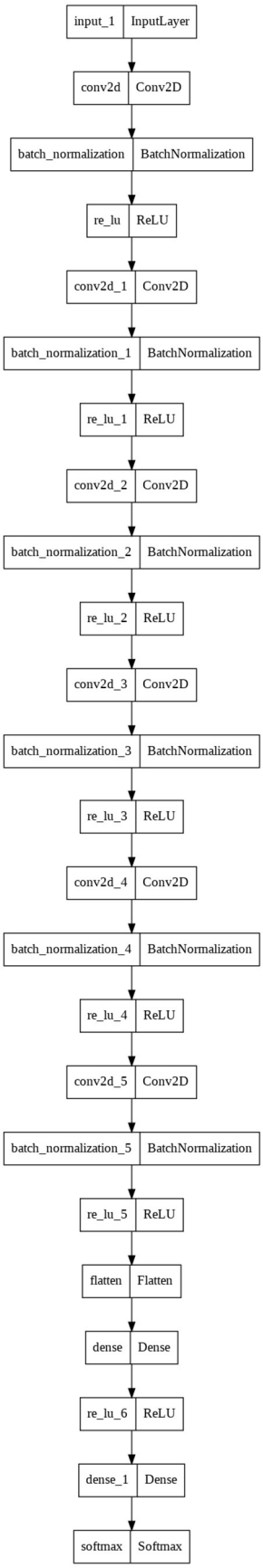

Figure 39 Architecture of the first model of beamforming selection

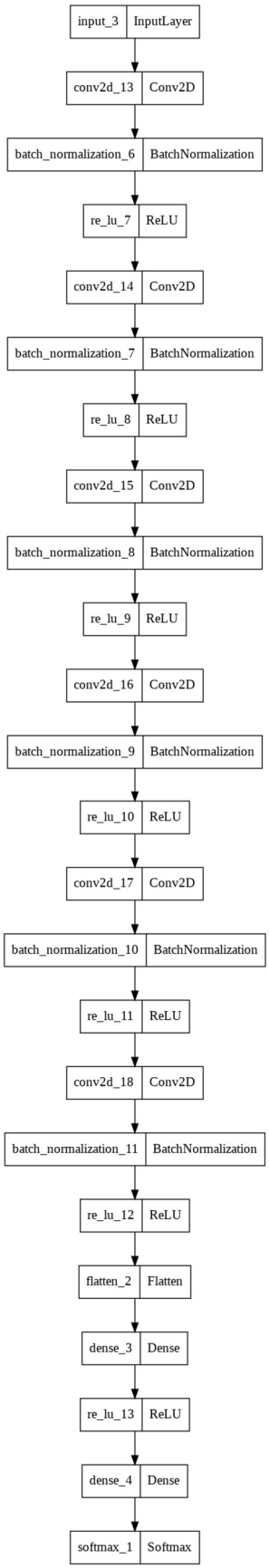

Figure 38 Architecture of the second model of beamforming selection

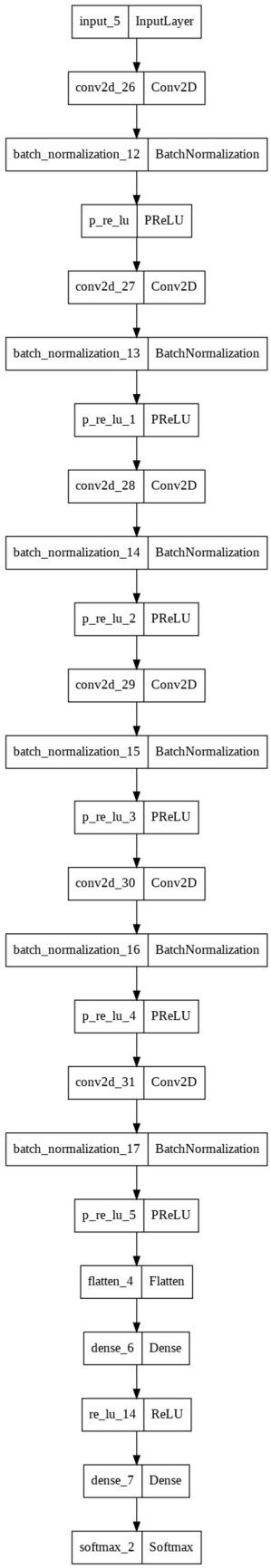

Figure 37 Architecture of the third model of beamforming selection

In federated learning, the next table will introduce the training parameters on the client server and centralized server:

Training Parameter	Value
Client optimizer	Adam(lr=5e-3)
Server optimizer	SGD(lr=.25)
Loss	CategoricalCrossentropy

Table 15 Parameters used in federated learning

2.4 Testing

2.4.1 Data Plan

The dataset was provided by Para-State university as a challenge that the university has posted based on a simulation made by its staff, any team around the globe can compete in this challenge.[11][10] Two sets of data have been used, the first one is tabular data that was created in an NFV-based test environment simulated for 5G communication. This dataset is classified into normal and abnormal labels, it was split into three fragments which are training, validation and testing with the splitting percentages of 80%, 10% and 10% respectively. The second dataset is named raymobtime that consists of LiDAR data and the part of it that was used has an input shape of (9234, 20, 200, 10) and the proposed system gives an output shape of (9234, 8, 32) which represents 8 receiver antennas and 32 transmitter antennas. The shape of validation shape is (1960, 20, 200, 10) and that of testing is (9638, 20, 200, 10).[16]

2.4.2 Validation & Verification

2.4.2.1 Binary classification

Confusion matrix and macro average F1-Score have been used to evaluate the results of binary classification problem. To meet the design requirements, the model can classify the network into normal and abnormal with high scores, so the problem is so simple and achieved the design requirements, so apply multiclass classification can make the problem more complex and classify them to normal and their failure types to know if the network fails and the reason of the failure.

2.4.2.2 Multi-class classification

Confusion matrix and F1-Score have been used to evaluate the results of multi-class classification problem. To meet the design requirements, the model can classify the network into normal and failure types of abnormal with high scores.

2.4.2.3 Cascaded modeling

Confusion matrix and F1-Score have been used to evaluate the results of cascaded model problem. The model meets the design requirements to classify the network status into normal and its failure types, then deploying the cascaded model on AWS.

2.4.2.4 One vs all modeling

Confusion matrix and F1-Score have been used to evaluate the results of one Vs All models to examine if each class was easy to separate and if the aggregation model would be better than the cascaded model in classifying failure types.

2.4.2.5 Clustering modeling

2.4.2.5.1 Agglomerative model

For the first experiment, the following table contains the evaluation metrices of using silhouette and kappa scores for both training and testing datasets at k=11 for ward linkage agglomerative clustering:

	Evaluation metric	Value
Training	kappa score	0.02303368
	Silhouette score	0.7005552
Testing	kappa score	0.00511996
	Silhouette score	0.63134602

Table 16 Kappa and silhouette scores for agglomerative modeling (ward)

The following cluster distribution shows that the algorithm can isolate class five perfectly on both training and testing datasets, but the train and test distributions were not consistent.

pred	0	1	2	3	4	5	6	7	8	9	10
true											
1	0	0	0	143	0	0	0	0	2313	0	0
2	0	0	0	72	0	0	0	752	0	729	708
3	0	0	0	188	0	0	1	0	2089	0	2
4	0	0	0	91	765	667	726	0	80	0	0
5	787	787	781	0	0	0	0	0	0	0	0

Figure 40 Cluster distribution for training in agglomerative modeling

pred	0	1	2	3	4	5	6	7	8	9	10
true											
1	0	0	0	0	22	0	0	0	0	0	286
2	0	0	0	0	8	0	0	98	92	89	0
3	0	0	0	0	50	0	0	0	0	0	230
4	0	97	93	0	4	0	91	0	0	0	6
5	98	0	0	99	0	97	0	0	0	0	0

Figure 41 Cluster distribution for testing in agglomerative modeling

For the second experiment, the following table contains the evaluation metrices of using silhouette and kappa scores for both training and testing datasets at k=11 for average linkage agglomerative clustering:

	Evaluation metric	Value
Training	kappa score	0.183672
	Silhouette score	0.49738
Testing	kappa score	-0.021478
	Silhouette score	0.54609

Table 17 Kappa and silhouette scores for agglomerative modeling (average)

2.4.2.5.2 DBSCAN model

For the third experiment, the following table contains the evaluation metrices silhouette and kappa scores using both training and testing datasets, as the Bayesian optimization has been utilized to get the highest silhouette score regardless of the kappa score.

	Evaluation metric	Value
Training	kappa score	0.159561
	Silhouette score	0.690042
Testing	kappa score	-0.061771
	Silhouette score	0.5762

Table 18 Kappa and silhouette scores for DBSCAN modeling (1st)

The following cluster distributions show that, cluster 1 contains the majority of classes 1 and 3, and that outlier cluster (-1) contains less overlapped minority of all classes on the training and the testing datasets, because cluster 0 in testing contains overlapped classes of 1 and 3 same for cluster 1 in training dataset.

pred	-1	0	1	2	3	4	5	6	7	8	9
true											
1	3	0	2453	0	0	0	0	0	0	0	0
2	36	0	0	741	741	743	0	0	0	0	0
3	5	0	2275	0	0	0	0	0	0	0	0
4	128	0	14	0	0	0	744	0	751	692	0
5	57	762	0	0	0	0	0	772	0	0	764

Figure 43 Cluster distribution for training in DBSCAN modeling (1st)

pred	-1	0	1	2	3	4	5	6	7	8
true										
1	8	300	0	0	0	0	0	0	0	0
2	16	0	0	0	91	89	91	0	0	0
3	8	272	0	0	0	0	0	0	0	0
4	136	4	80	0	0	0	0	0	0	71
5	66	0	0	74	0	0	0	74	80	0

Figure 42 Cluster distribution for testing in DBSCAN modeling (1st)

For the fourth experiment, the following table contains the evaluation metrices silhouette and kappa scores using both training and testing datasets, as the Bayesian optimization has been utilized to get the highest kappa score regardless of the silhouette score.

	Evaluation metric	Value
Training	kappa score	0.3078
	Silhouette score	0.638944
Testing	kappa score	-0.021478
	Silhouette score	0.54609

Table 19 Kappa and silhouette scores for DBSCAN modeling (2nd)

The following cluster distributions show that, cluster 1 has an overlapping of all points of classes 1 and 3 and also cluster -1 contains some points of all classes and there are no pure class and the same thing for testing.

pred	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
true																	
1	3	0	2453	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	27	0	0	744	746	0	0	0	0	0	0	744	0	0	0	0	0
3	7	0	2273	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	28	0	8	0	0	742	0	765	689	0	44	0	0	22	19	0	12
5	43	747	0	0	0	0	781	0	0	754	0	0	20	0	0	10	0

Figure 44 Cluster distribution for training in DBSCAN modeling (2nd)

pred	-1	0	1	2	3	4	5	6	7	8	9	10	11	12
true														
1	0	0	308	0	0	0	0	0	0	0	0	0	0	0
2	32	0	0	0	0	0	0	0	89	81	85	0	0	0
3	8	0	272	0	0	0	0	0	0	0	0	0	0	0
4	52	0	0	82	61	14	0	0	0	0	0	0	65	17
5	54	84	0	0	0	0	69	17	0	0	0	70	0	0

Figure 45 Cluster distribution for testing in DBSCAN modeling (2nd)

2.4.2.5.3 GMM model

For the first experiment, the following table contains evaluation metrices of silhouette and kappa scores for both training and testing datasets at GMM Components = 11:

	Evaluation metric	Value
Training	kappa score	0.7446
	Silhouette score	0.67
Testing	kappa score	0.73
	Silhouette score	0.61

Table 20 Kappa and silhouette scores for GMM modeling at components = 11

For the first experiment: The following figures show the cluster distribution which indicates that, the algorithm can isolate class5 well on the training dataset but is not able to do the same on testing dataset, and the train and test distributions are not consistent.

For the second experiment, the following table contains the evaluation metrices silhouette and kappa scores for both training and testing datasets at GMM Components = 15:

	Evaluation metric	Value
Training	kappa score	0.821
	Silhouette score	0.482
Testing	kappa score	0.822
	Silhouette score	0.419

Table 21 Kappa and silhouette scores for GMM modeling at components = 15

For the second experiment: The following figures show the cluster distribution which indicates that, the algorithm can isolate class5 well on the training dataset but is not able to do the same on testing dataset, and the train and test distributions are not consistent.

pred	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
true															
1	551	3	0	0	0	0	0	0	0	0	0	0	0	0	1902
2	0	135	0	0	0	0	723	0	0	678	725	0	0	0	0
3	1256	7	0	0	0	0	0	1	0	0	0	0	0	0	1016
4	79	0	0	0	0	715	0	455	472	0	0	293	0	307	8
5	0	0	787	384	787	0	0	0	0	0	0	0	397	0	0

Figure 47 Cluster distribution for training in GMM modeling at components = 15

pred	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
true															
1	85	1	0	0	0	0	0	0	0	0	0	0	0	0	222
2	0	33	0	0	0	0	84	0	0	81	89	0	0	0	0
3	179	2	0	0	0	0	0	0	0	0	0	0	0	0	99
4	7	0	0	0	0	97	0	47	52	0	0	39	0	46	3
5	0	5	94	48	98	0	0	0	0	5	0	0	44	0	0

Figure 46 Cluster distribution for testing in GMM modeling at components = 15

2.4.2.5.4 K-means model

For the first experiment, the following table contains the evaluation metrices silhouette and kappa scores for both training and testing datasets at k-means Components = 11:

pred	0	1	2	3	4	5	6	7	8	9	10
true											
1	5	2451	0	0	0	0	0	0	0	0	0
2	21	0	0	0	0	0	747	0	0	746	747
3	48	2231	0	0	0	0	0	1	0	0	0
4	53	71	0	0	0	703	0	748	754	0	0
5	0	0	787	781	787	0	0	0	0	0	0

Figure 49 Cluster distribution for training in k-means modeling at components = 11

pred	0	1	2	3	4	5	6	7	8	9	10
true											
1	1	307	0	0	0	0	0	0	0	0	0
2	10	0	0	0	0	0	90	0	0	95	92
3	14	266	0	0	0	0	0	0	0	0	0
4	13	9	0	0	0	93	0	90	86	0	0
5	10	0	94	92	98	0	0	0	0	0	0

Figure 48 Cluster distribution for testing in k-means modeling at components = 11

	Evaluation metric	Value
Training	kappa score	0.730
	Silhouette score	0.70
Testing	kappa score	0.731
	Silhouette score	0.65

Table 22 Kappa and silhouette scores for k-means modeling at components = 11

For the second experiment, the following table contains the evaluation metrices silhouette and kappa scores for both training and testing datasets at k-means Components = 10:

	Evaluation metric	Value
Training	kappa score	0.7464
	Silhouette score	0.68
Testing	kappa score	0.753
	Silhouette score	0.62

Table 23 Kappa and silhouette scores for k-means modeling at components = 10

pred	0	1	2	3	4	5	6	7	8	9
true										
1	2456	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	752	0	0	755	754
3	2278	0	0	0	0	0	2	0	0	0
4	80	0	0	720	0	0	763	766	0	0
5	0	781	787	0	787	0	0	0	0	0

Figure 51 Cluster distribution for training in k-means modeling at components = 10

pred	0	1	2	3	4	5	6	7	8	9
true										
1	308	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	93	0	0	96	98
3	280	0	0	0	0	0	0	0	0	0
4	7	0	0	97	0	0	94	93	0	0
5	0	97	98	0	99	0	0	0	0	0

Figure 50 Cluster distribution for testing in k-means modeling at components = 10

For the third experiment, the following figures show the cluster distribution which indicates that the algorithm cannot isolate any class perfectly on both training and testing datasets, but the train and test distributions are consistent.

For the fourth experiment, the following figures show the cluster distribution which indicates that the algorithm can isolate class 2 and 5 well on both training and testing datasets and the train and test distributions are consistent.

pred	0	1	2	3	4	5	6	7	8	9	10
true											
1	2354	0	0	0	0	0	0	102	0	0	0
2	0	0	0	0	0	0	742	33	743	0	743
3	2187	0	0	0	1	0	0	92	0	0	0
4	76	0	0	0	724	669	0	148	0	712	0
5	0	781	768	787	0	0	0	19	0	0	0

Figure 53 Cluster distribution for training in k-means modeling at components = 11

pred	0	1	2	3	4	5	6	7	8	9	10
true											
1	297	0	0	0	0	0	0	11	0	0	0
2	0	0	0	0	0	0	91	9	93	0	94
3	262	0	0	0	0	0	0	18	0	0	0
4	7	0	0	0	80	88	0	37	0	79	0
5	0	97	94	98	0	0	0	5	0	0	0

Figure 52 Cluster distribution for testing in k-means modeling at components = 11

For the fifth experiment, confusion matrix and F1-Score based on the classification report for cluster zero that has been used to verify the model and meet the design requirements.

For the sixth experiment, confusion matrix and F1-Score based on the classification report for the clusters 1, 3 and 4 that have been used to verify the model and meet the design requirements.

For the seventh experiment, confusion matrix and F1-Score based on the classification report for the classification that has been used to verify the model and meet the design requirements and be able to classify all the classes.

2.4.2.6 Beamforming selection

The evaluation metrics to evaluate the training process Top K accuracies in 3.6 the training accuracy curve will be shown using top 1,5,10 accuracies.

3. Overall Results and Analysis

3.1 Binary classification

The classification report of training XGBOOST model shows that the model achieved 98% accuracy and 90% macro average F1-Score.

evaluation on training

	precision	recall	f1-score	support
0	0.98	1.00	0.99	175339
1	1.00	0.69	0.81	11681
accuracy			0.98	187020
macro avg	0.99	0.84	0.90	187020
weighted avg	0.98	0.98	0.98	187020

Figure 54 Classification report of XGboost (binary classification) using training dataset

The classification report on testing XGBOOST model shows that the model achieved 98% accuracy and 90% macro average F1-Score.

evaluation on testing

	precision	recall	f1-score	support
0	0.98	1.00	0.99	21900
1	0.93	0.72	0.81	1460
accuracy			0.98	23360
macro avg	0.95	0.86	0.90	23360
weighted avg	0.98	0.98	0.98	23360

Figure 55 Classification report of XGboost (binary classification) using testing dataset

According to the training and testing confusion matrices, the model misclassifies some values of class 1.

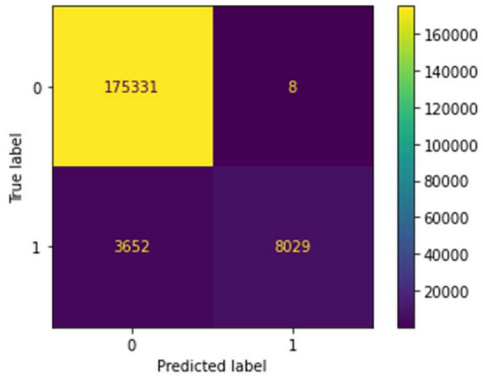

Figure 57 Confusion matrix of XGboost (binary classification) using training dataset

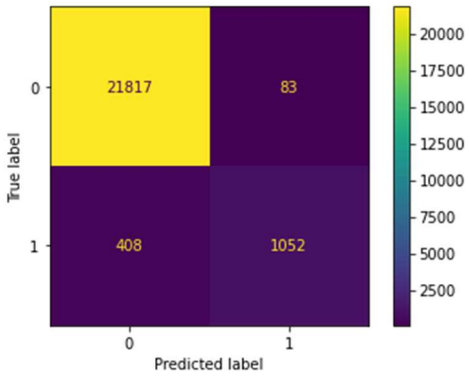

Figure 56 Confusion matrix of XGboost (binary classification) using testing dataset

The T-SNE plot shows the distribution of the data:

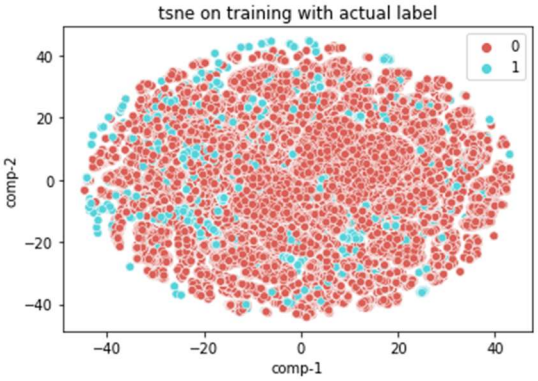

Figure 58 T-SNE plot on training with actual data

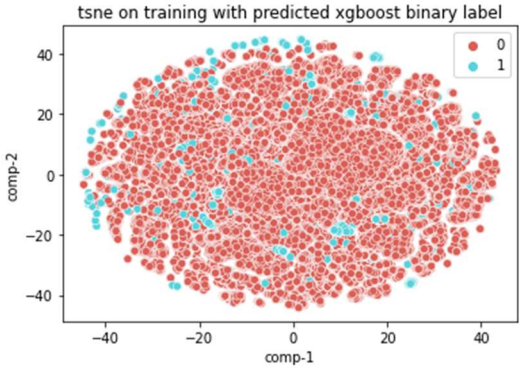

Figure 59 T-SNE plot on training with predicted labels

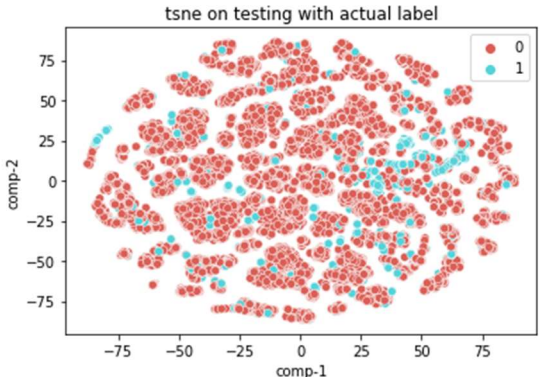

Figure 61 T-SNE plot on testing with actual data

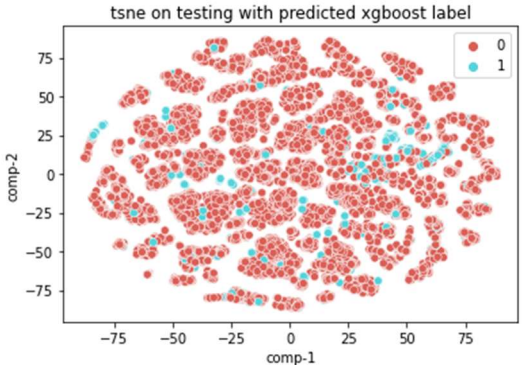

Figure 60 T-SNE plot on testing with predicted labels

According to the results of T-SNE plots on training and testing with XGBOOST model, the model misclassified some values of class 1 and classify them as class 0, as the number of zeros is 840126 and number of ones is 9868, the data is imbalanced data, so the solution is to make this problem more complex by applying multiclass classification. The main idea of multiclass classification is to classify the abnormal status into failure types based on domain expert and studying the main reasons of network failure.

3.2 Multi-class classification

1- Results of logistic regression model:

Classification report of testing logistic regression model:

	precision	recall	f1-score	support
0	0.94	1.00	0.97	21900
1	0.00	0.00	0.00	308
2	0.30	0.06	0.10	287
3	0.39	0.07	0.12	280
4	0.00	0.00	0.00	291
5	0.00	0.00	0.00	294
accuracy			0.94	23360
macro avg	0.27	0.19	0.20	23360
weighted avg	0.89	0.94	0.91	23360

Figure 62 Classification report of Logistic regression (multi-class classification) using testing dataset

According to the F1-Score, the model can't predict class 1, class 4 and class 5 and misclassify some points in class 2 and class 3. Also, according to the confusion matrix, the model can't predict class1, class4 and class 5.

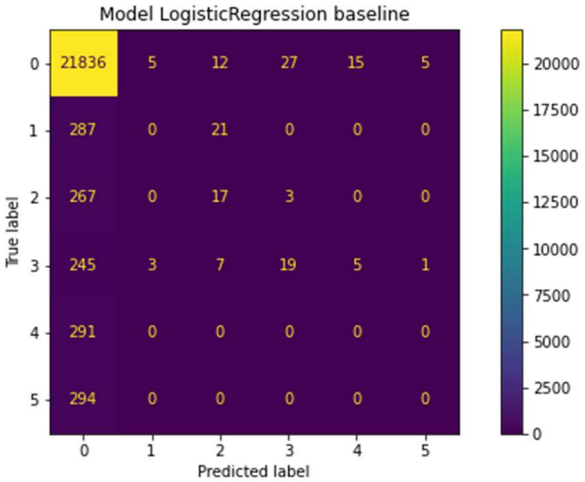

Figure 63 confusion matrix of Logistic regression (multi-class classification) using testing dataset

2- Results of Naïve Bayes model:

Classification report on testing Naïve Bayes model:

	precision	recall	f1-score	support
0	0.91	0.01	0.02	21900
1	0.00	0.00	0.00	308
2	0.02	0.88	0.03	287
3	0.03	0.15	0.05	280
4	0.01	0.22	0.03	291
5	0.01	0.05	0.01	294
accuracy			0.03	23360
macro avg	0.16	0.22	0.02	23360
weighted avg	0.85	0.03	0.02	23360

Figure 64 Classification report of naïve bayes (multi-class classification) using testing dataset

According to the F1-Score in the classification report of Naïve Bayes model can't predict all of the classes well and it shown in the following confusion matrix:

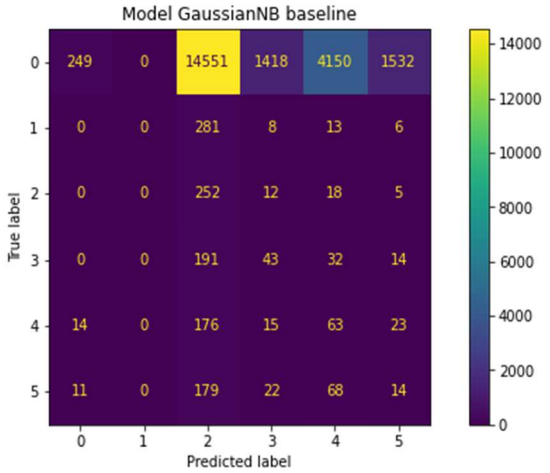

Figure 65 confusion matrix of naïve bayes (multi-class classification) using testing dataset

These results are obvious evidences that the problem is a non-linear problem, so boosting techniques is the best choice to deal with this problem. Random Forest and XGboost will be applied.

3- Results of Random Forest model:

Classification report on testing Random Forest model:

	precision	recall	f1-score	support
0	0.99	0.98	0.98	21900
1	0.83	0.57	0.67	308
2	0.36	1.00	0.53	287
3	0.93	0.25	0.40	280
4	1.00	1.00	1.00	291
5	1.00	1.00	1.00	294
accuracy			0.96	23360
macro avg	0.85	0.80	0.76	23360
weighted avg	0.97	0.96	0.96	23360

Figure 66 Classification report of random forest (multi-class classification) using testing dataset

According to the F1-Score in the classification report of Random Forest model, the model can predict class 0, class 4 and class 5 well, but the model misclassified class 1, class 2 and class 3 and the confusion matrix shows that:

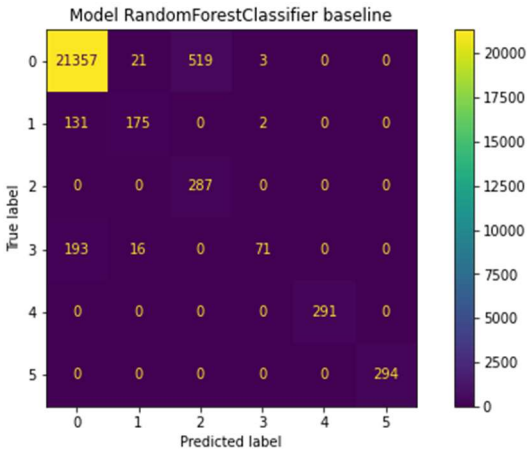

Figure 67 confusion matrix of random forest (multi-class classification) using testing dataset

4- Results of XGboost model:

Classification report on testing XGboost model:

	precision	recall	f1-score	support
0	0.99	0.97	0.98	21900
1	0.81	0.71	0.76	308
2	0.35	1.00	0.52	287
3	0.84	0.38	0.53	280
4	1.00	1.00	1.00	291
5	1.00	0.98	0.99	294
accuracy			0.96	23360
macro avg	0.83	0.84	0.80	23360
weighted avg	0.98	0.96	0.97	23360

Figure 68 Classification report of XGboost (multi-class classification) using testing dataset

According to the F1-Score in the classification report of XGboost model, the model can predict class0, class 4 and class 5 well, but the model misclassified some values of class 1, class 2 and class 3 and the confusion matrix shows that:

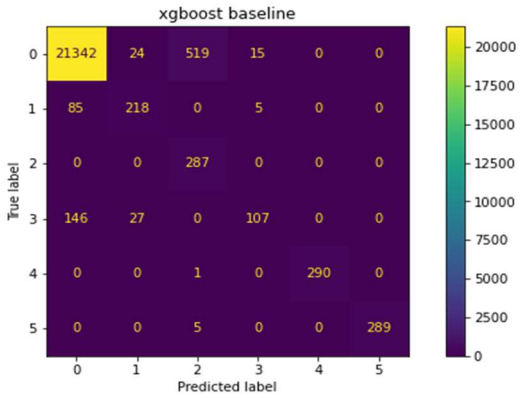

Figure 69 confusion matrix of XGboost (multi-class classification) using testing dataset

After comparing the results of the 4 baseline models, the champion model is XGboost. T-SNE plot of XGboost as a baseline model:

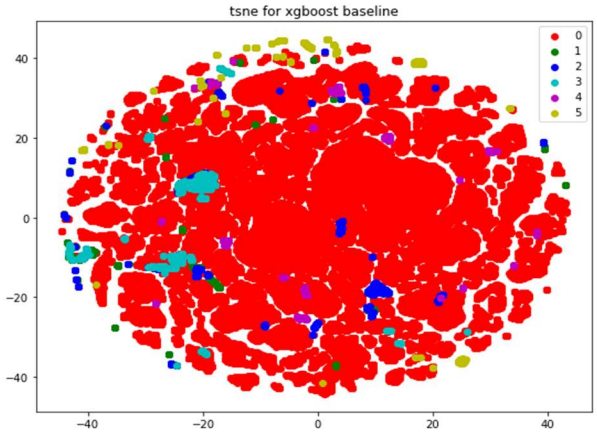

Figure 70 T-SNE for XGboost as baseline model

According to the results of Random Forest and XGBOOST model, they are nearby from each other, the time taken to train each model will be an important factor in comparison and is shown below.

	RandomForest	XGBOOST
whole time	1min 7s	43.2 s

Figure 71 whole time of random forest and XGboost in multi-class classification

XGBOOST model take less time than Random Forest model, so it's the champion model.

5- Results of XGboost model after hyperparameter tuning:

Classification report on testing hyperparameter tuned XGboost model:

	precision	recall	f1-score	support
0	0.99	0.97	0.98	21900
1	0.83	0.72	0.77	308
2	0.36	1.00	0.53	287
3	0.88	0.42	0.57	280
4	1.00	1.00	1.00	291
5	1.00	1.00	1.00	294
accuracy			0.97	23360
macro avg	0.84	0.85	0.81	23360
weighted avg	0.98	0.97	0.97	23360

Figure 72 Classification report of hyperparameter tuned XGboost (multi-class classification) using testing dataset

According to the F1-score in the classification report of tuned XGboost model, the model can predict class 0, class 1, class 4 and class 5 well, but the model misclassified some values of class 2 and class 3 and the confusion matrix shows that:

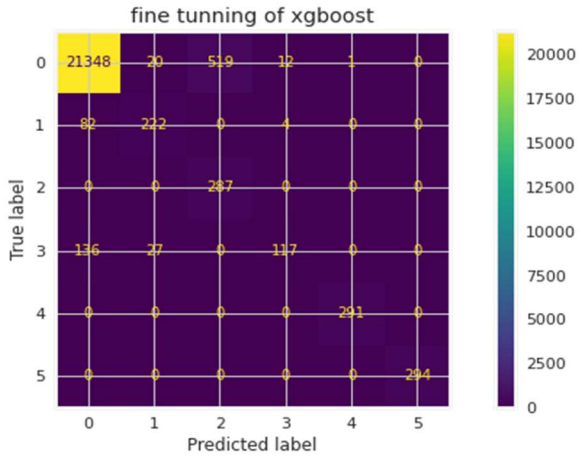

Figure 73 confusion matrix of hyperparameter tuned XGboost (multi-class classification) using testing dataset

T-SNE plot of hyperparameter tuned XGboost model:

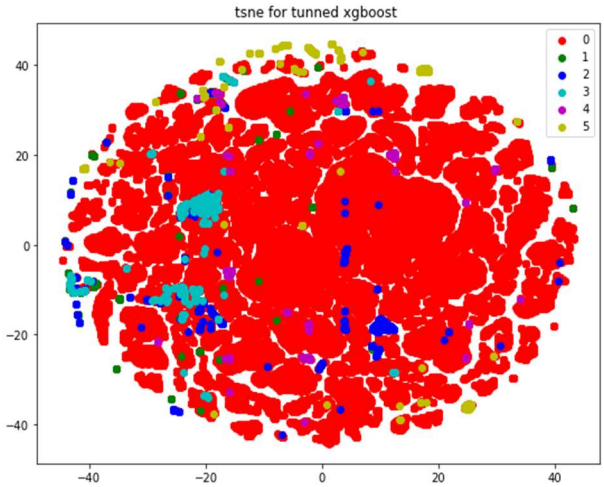

Figure 74 T-SNE of hyperparameter tuned XGboost model

According to the T-SNE plot of tuned XGboost model, the majority class is class 0 which has the red color and the minority classes are class 1, class 2, class 3, class 4, and class 5. Some of the classes are overlapped, so the solution to overcome overlapping is applying Cascaded model to apply a binary class classification model then apply multiclass classification model and concatenate between the results to make the model able to detect failure types if the network status is abnormal.

3.3 Cascaded modeling

For the binary class classification, the classification report on training shows that the model trained well on the training data.

evaluation on training

	precision	recall	f1-score	support
0	1.00	1.00	1.00	175339
1	1.00	1.00	1.00	11681
accuracy			1.00	187020
macro avg	1.00	1.00	1.00	187020
weighted avg	1.00	1.00	1.00	187020

Figure 75 Classification report of binary model using training dataset

The following confusion matrix on training show that:

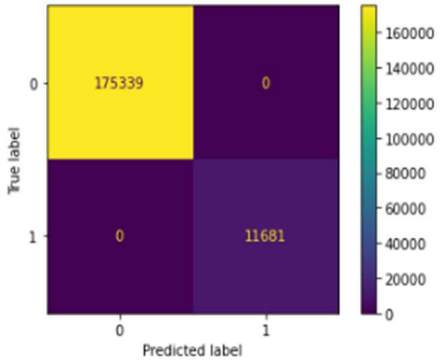

Figure 76 Confusion matrix of binary model using training dataset

And also, according to the testing classification report, the model can predict the classes well as shown in the following classification report:

evaluation on testing

	precision	recall	f1-score	support
0	0.98	1.00	0.99	21900
1	0.93	0.72	0.81	1460
accuracy			0.98	23360
macro avg	0.95	0.86	0.90	23360
weighted avg	0.98	0.98	0.98	23360

Figure 77 Classification report of binary model using testing dataset

And the following confusion matrix prove that:

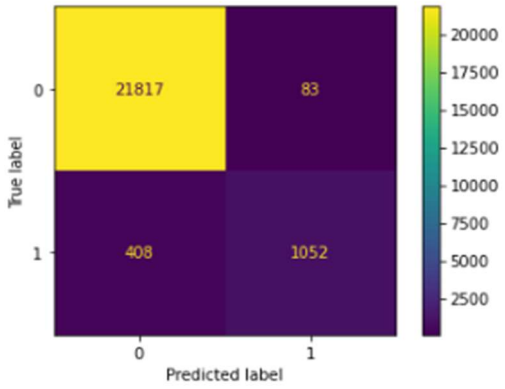

Figure 78 Confusion matrix of binary model using testing dataset

The T-SNE plot on the training with predicted label show that the majority class is class 0 and minority class is class 1.

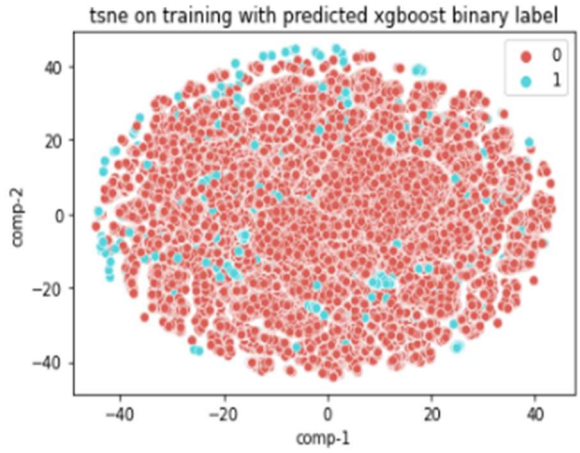

Figure 79 T-SNE plot of binary model using training dataset

The T-SNE plot on testing dataset prove that the multiclass classification must be implemented:

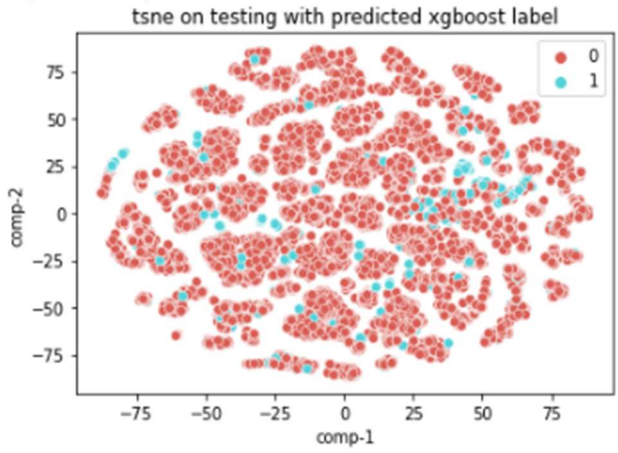

Figure 80 T-SNE plot of binary model using testing dataset

For the multiclass class classification, the classification report on training shows that the model trained well on the training data.

evaluation on training

	precision	recall	f1-score	support
1	1.00	1.00	1.00	2456
2	1.00	1.00	1.00	2261
3	1.00	1.00	1.00	2280
4	1.00	1.00	1.00	2329
5	1.00	1.00	1.00	2355
accuracy			1.00	11681
macro avg	1.00	1.00	1.00	11681
weighted avg	1.00	1.00	1.00	11681

Figure 81 Classification report of multi-class model using training dataset

According to the F1-Score of all of the classes, the model trained well. The following confusion matrix on training show that:

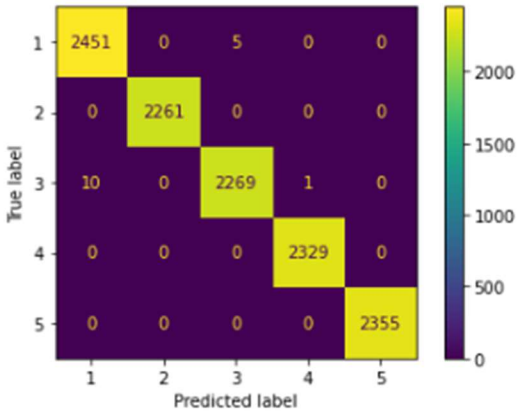

Figure 82 Confusion matrix of multi-class model using training dataset

For the multiclass class classification, the classification report on testing shows that the model fits well on the testing data.

evaluation on testing

	precision	recall	f1-score	support
1	0.80	0.91	0.85	308
2	1.00	1.00	1.00	287
3	0.88	0.75	0.81	280
4	1.00	1.00	1.00	291
5	1.00	1.00	1.00	294
accuracy			0.93	1460
macro avg	0.94	0.93	0.93	1460
weighted avg	0.93	0.93	0.93	1460

Figure 83 Classification report of multi-class model using testing dataset

According to F1-Score, the model can predict class2, class4 and class5 with 100%, class1 with 85% and class3 with 81%. The following confusion matrix prove that:

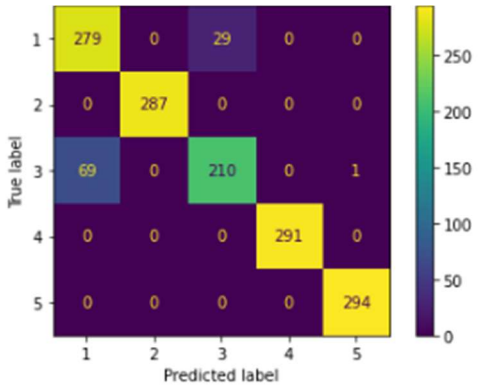

Figure 84 Confusion matrix of multi-class model using testing dataset

T-SNE plot for training data on cascaded model:

evaluation on training

	precision	recall	f1-score	support
0	1.00	1.00	1.00	9225
1	0.98	0.98	0.98	2456
accuracy			0.99	11681
macro avg	0.99	0.99	0.99	11681
weighted avg	0.99	0.99	0.99	11681

Figure 85 classification report of class one model in training

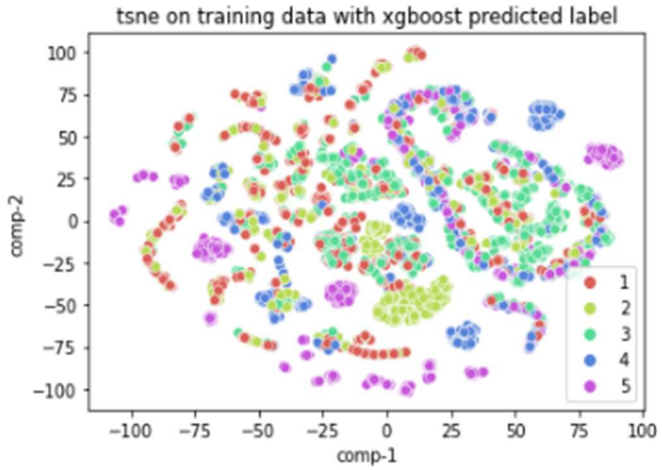

Figure 87 T-SNE plot of cascaded model using training dataset

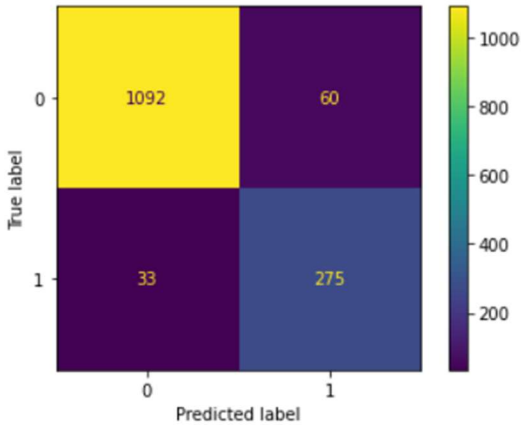

Figure 86 Confusion matrix of class one model in testing

This TSNE plot indicates that there is some small overlapping between some of the classes, but the model can predict and classify the classes well.

The TSNE plot on testing data refers to that the model can classify the classes, and the cascaded model meets the design requirements well.

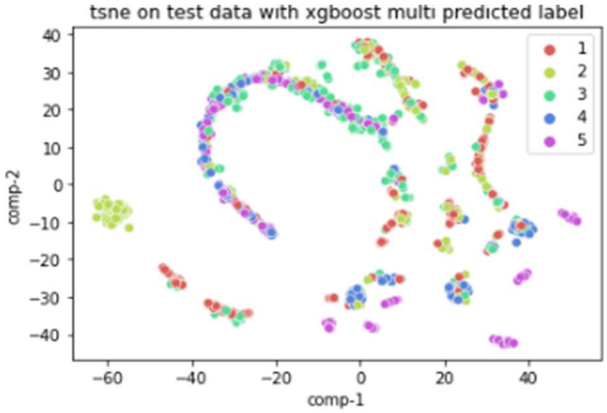

Figure 88 T-SNE plot of cascaded model using testing dataset

3.4 One vs all modeling

Model specified for class one outputs:

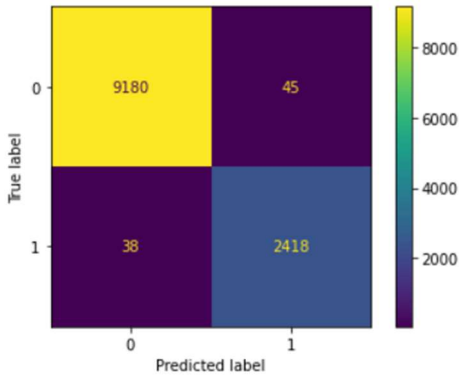

Figure 90 Confusion matrix of class one model in training

evaluation on testing

	precision	recall	f1-score	support
0	0.97	0.95	0.96	1152
1	0.82	0.89	0.86	308
accuracy			0.94	1460
macro avg	0.90	0.92	0.91	1460
weighted avg	0.94	0.94	0.94	1460

Figure 89 classification report of class one model in testing

These results show that class 1 is relatively hard to identify, and such information was known from EDA and would be later on confirmed using clustering techniques.

Model specified for class 2 outputs:

evaluation on training

	precision	recall	f1-score	support
0	1.00	1.00	1.00	9420
1	1.00	1.00	1.00	2261
accuracy			1.00	11681
macro avg	1.00	1.00	1.00	11681
weighted avg	1.00	1.00	1.00	11681

Figure 91 Classification report of class two model in training

evaluation on testing

	precision	recall	f1-score	support
0	1.00	0.99	0.99	1173
1	0.95	1.00	0.97	287
accuracy			0.99	1460
macro avg	0.98	0.99	0.98	1460
weighted avg	0.99	0.99	0.99	1460

Figure 92 Classification report of class two model in testing

These results show that class 2 is easy to identify, so the model can identify mostly all of them, and such information would be confirmed later on using clustering techniques.

Model specified for class 3 outputs:

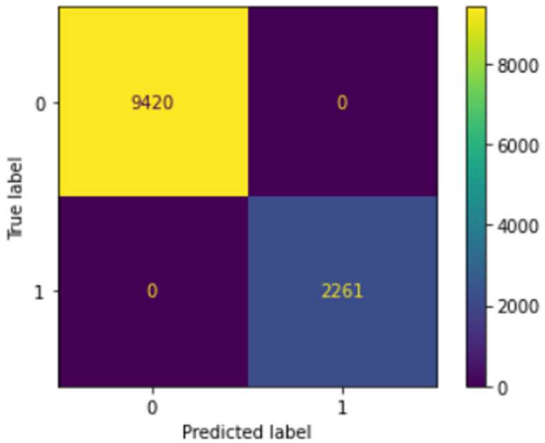

Figure 94 Confusion matrix of class two model in training

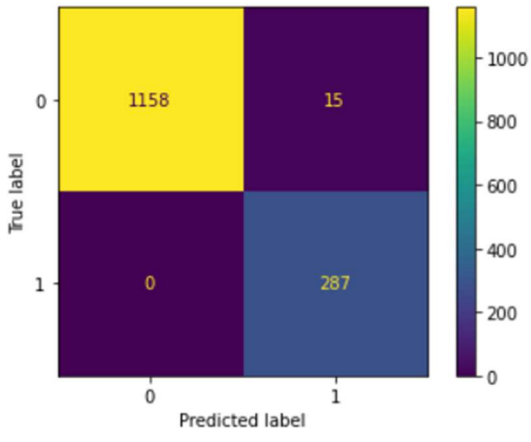

Figure 93 Confusion matrix of class two model in testing

evaluation on training

	precision	recall	f1-score	support
0	0.99	1.00	0.99	9401
1	0.99	0.96	0.98	2280
accuracy			0.99	11681
macro avg	0.99	0.98	0.99	11681
weighted avg	0.99	0.99	0.99	11681

Figure 95 Classification report of class three model in training

evaluation on testing

	precision	recall	f1-score	support
0	0.94	0.98	0.96	1180
1	0.89	0.73	0.80	280
accuracy			0.93	1460
macro avg	0.92	0.85	0.88	1460
weighted avg	0.93	0.93	0.93	1460

Figure 96 Classification report of class three model in testing

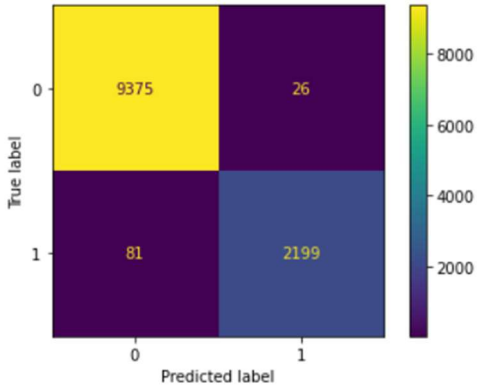

Figure 98 Confusion matrix of class three model in training

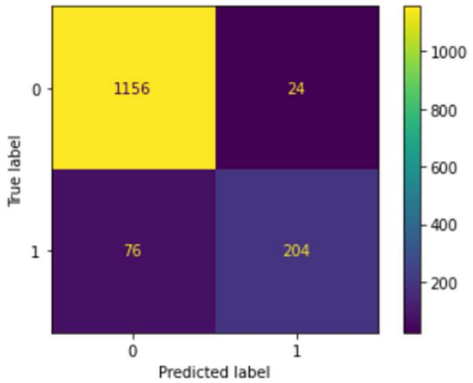

Figure 97 Confusion matrix of class three model in testing

These results show that class 3 is relatively hard to identify, and such information was known from EDA and would be confirmed later on using clustering techniques.

Model specified for class 4 outputs:

evaluation on training

	precision	recall	f1-score	support
0	1.00	1.00	1.00	9352
1	1.00	1.00	1.00	2329
accuracy			1.00	11681
macro avg	1.00	1.00	1.00	11681
weighted avg	1.00	1.00	1.00	11681

Figure 99 Classification report of class four model in training

evaluation on testing

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1169
1	1.00	1.00	1.00	291
accuracy			1.00	1460
macro avg	1.00	1.00	1.00	1460
weighted avg	1.00	1.00	1.00	1460

Figure 100 Classification report of class four model in testing

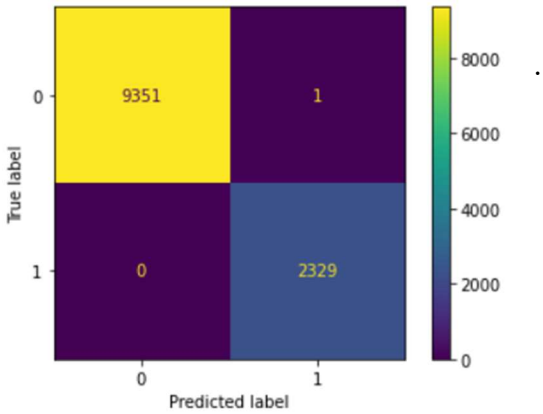

Figure 101 Confusion matrix of class four model in training

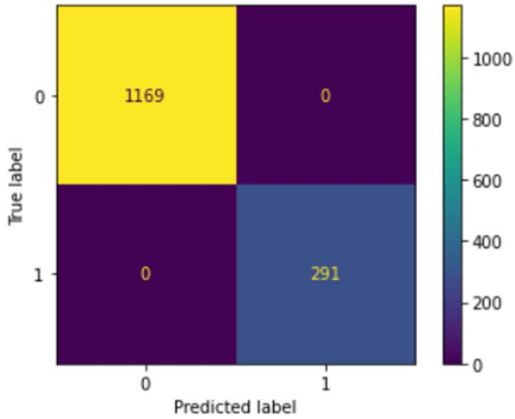

Figure 102 Confusion matrix of class four model in testing

These results show that class 4 is easy to identify, so the model can perfectly identify all of them.
Model specified for class 4 outputs:

evaluation on training

	precision	recall	f1-score	support
0	1.00	1.00	1.00	9326
1	1.00	1.00	1.00	2355
accuracy			1.00	11681
macro avg	1.00	1.00	1.00	11681
weighted avg	1.00	1.00	1.00	11681

Figure 104 Classification report of class five model in training

evaluation on testing

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1166
1	1.00	1.00	1.00	294
accuracy			1.00	1460
macro avg	1.00	1.00	1.00	1460
weighted avg	1.00	1.00	1.00	1460

Figure 103 Classification report of class five model in testing

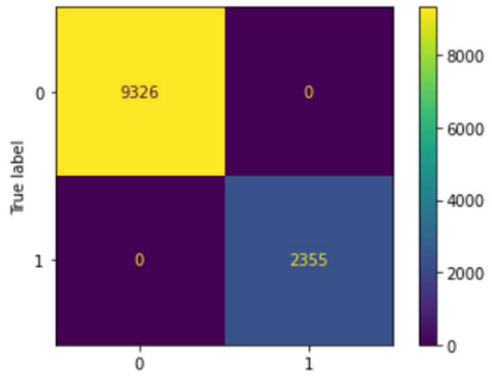

Figure 106 Confusion matrix of class five model in training

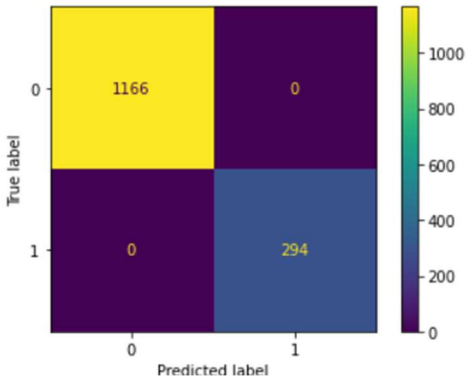

Figure 105 Confusion matrix of class five model in testing

These results show that class 5 is easy to identify, so the model can perfectly identify all of them, and such information would be confirmed using clustering.

Aggregation model results:

It is shown from the following classification report & confusion Matrix that aggregating the predictions formed a performance that is not comparable enough to the cascaded model, so this experiment didn't improve the performance, and didn't get the expected results.

evaluation on testing

	precision	recall	f1-score	support
1	0.80	0.92	0.86	308
2	1.00	1.00	1.00	287
3	0.89	0.75	0.81	280
4	1.00	1.00	1.00	291
5	1.00	1.00	1.00	294
accuracy			0.93	1460
macro avg	0.94	0.93	0.93	1460
weighted avg	0.94	0.93	0.93	1460

Figure 107 Classification report of aggregation in one vs all using testing dataset

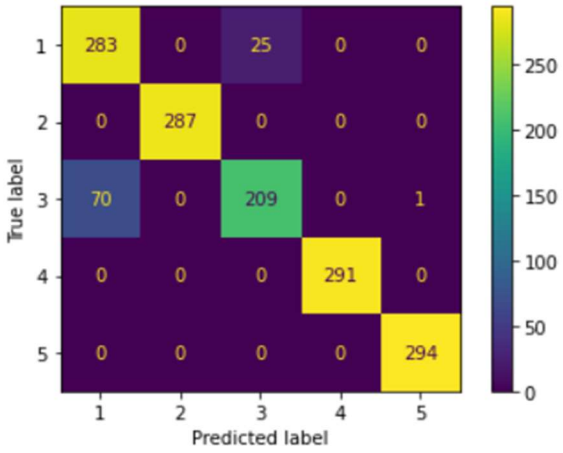

Figure 108 Confusion matrix of aggregation in one vs all using testing dataset

3.5 Clustering modeling

3.5.1 Agglomerative model

For the first experiment:

The following figures show a comparison between TSNE actual labels and TSNE with predicted label of agglomerative clustering model with linkage ward with k=11.

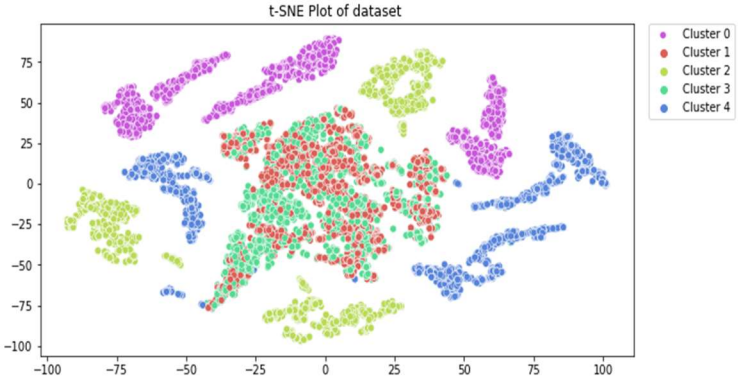

Figure 112 Actual agglomerative T-SNE plot for training

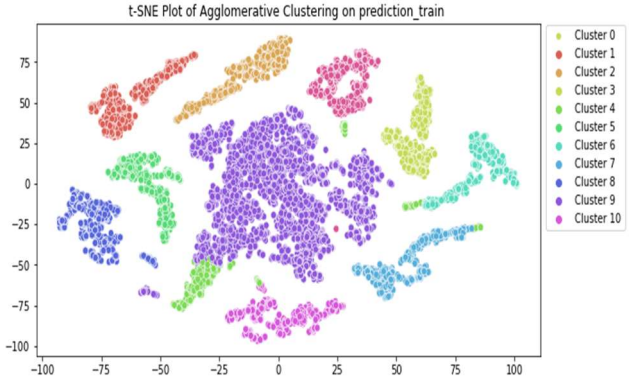

Figure 109 Predicted agglomerative TSNE plot for training

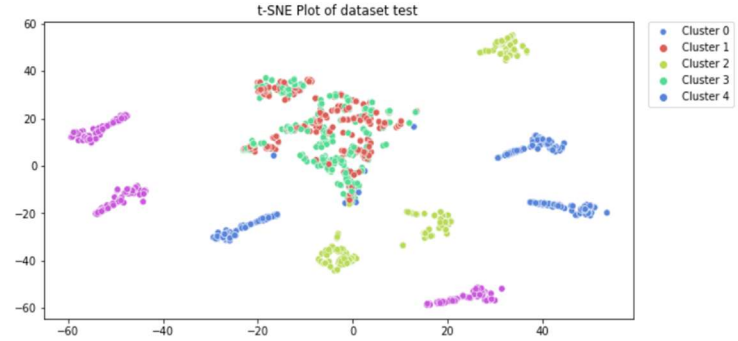

Figure 111 Actual agglomerative TSNE plot for testing

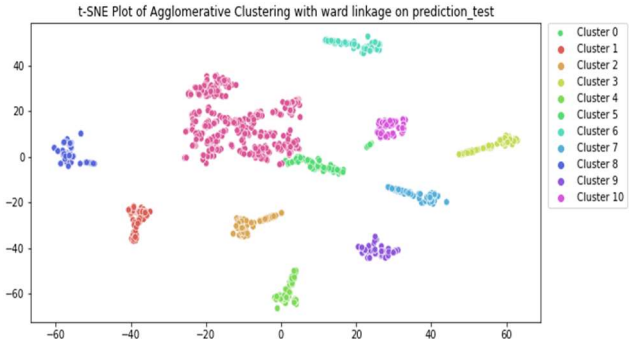

Figure 110 Predicted agglomerative TSNE plot for testing

From T-SNE plot on training, cluster 8 that contains most of class 1, class 3 and minority of class 4, so class 5 is well separated. From T-SNE plot on testing, cluster ten that contains most of class 1, class 3 and minority of class 4, and class 5 is well separated.

For the second experiment:

The following figures show a comparison between TSNE actual labels and TSNE with predicted label of agglomerative clustering model with linkage average with k=11

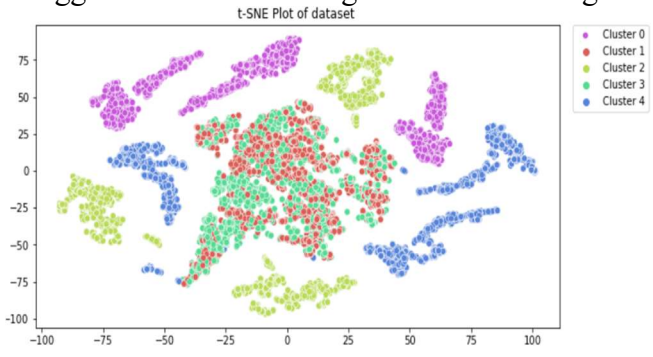

Figure 114 Actual agglomerative T-SNE plot for training

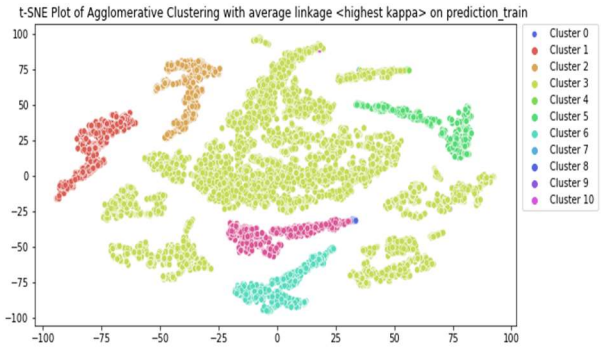

Figure 113 Predicted agglomerative T-SNE plot for training

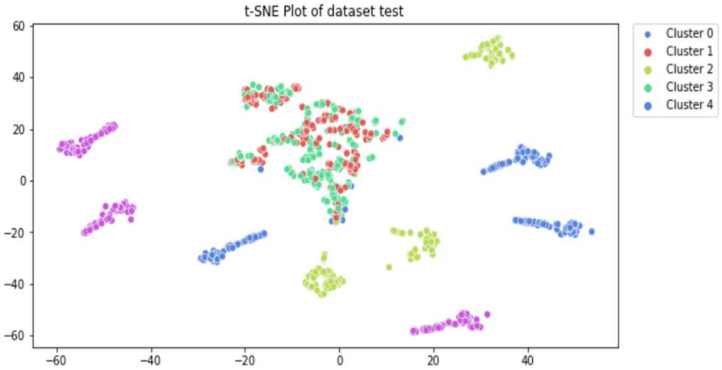

Figure 115 Actual agglomerative T-SNE plot for testing

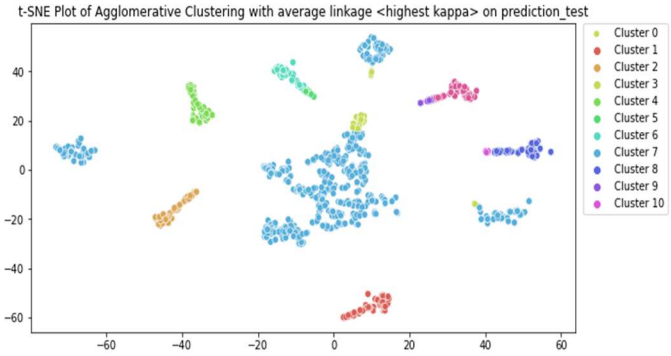

Figure 116 Predicted agglomerative T-SNE plot for testing

From the TSNE plot on training, the predicted training cluster 2 contains most of the classes points except class5 which is well isolated and also well separated class5.

As we see the cluster 2 contains less overlapped points from four classes except class5, which is well isolated and well separated, train and test distribution are not consistent as cluster 6 on training contains nearly no points, while cluster 6 on the test contains most of the classes points except class5.

3.5.2 DBSCAN model

For the first experiment:

The following figures show a comparison between TSNE actual labels and TSNE with predicted label of DBSCAN clustering model with k=11.

Figure 120 Actual DBSCAN T-SNE plot for training

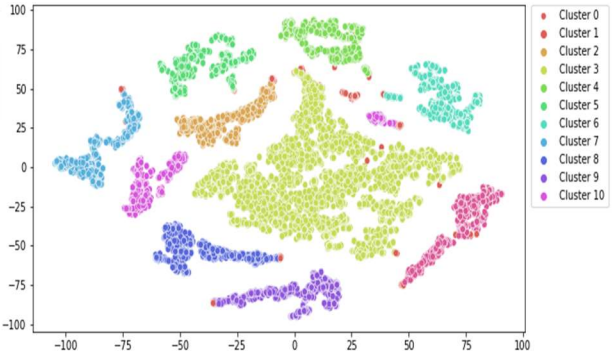

Figure 118 Predicted DBSCAN T-SNE plot for training

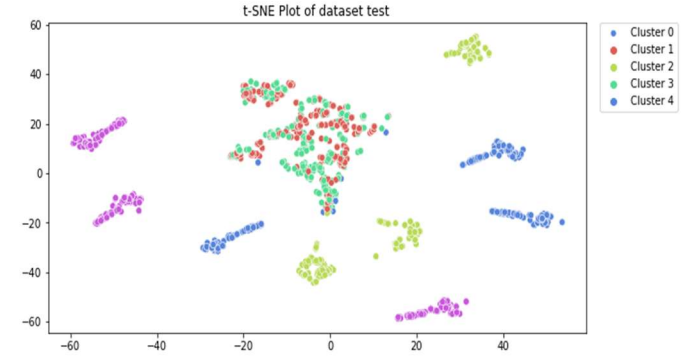

Figure 119 Actual DBSCAN T-SNE plot for testing

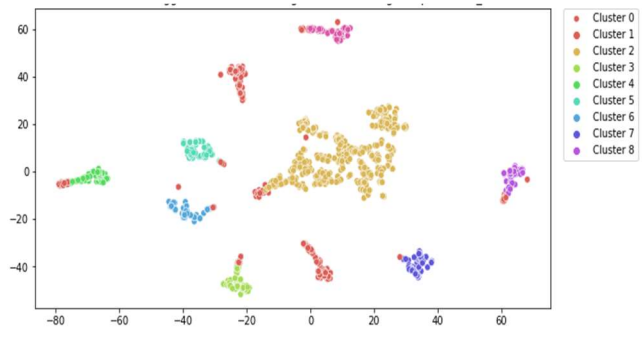

Figure 117 Predicted DBSCAN T-SNE plot for testing

From the T-SNE plot on training, the predicted training cluster 3 contains all values of classes 1, 3 and also cluster 0 contain some points of all classes as outliers.

Cluster 2 contains all points of classes 1 and 3 and cluster -1 contains some points of all clusters, so the train and test distribution are not consistent.

For the second experiment:

The following figures show a comparison between TSNE actual labels and TSNE with predicted label of DBSCAN clustering model with k=17.

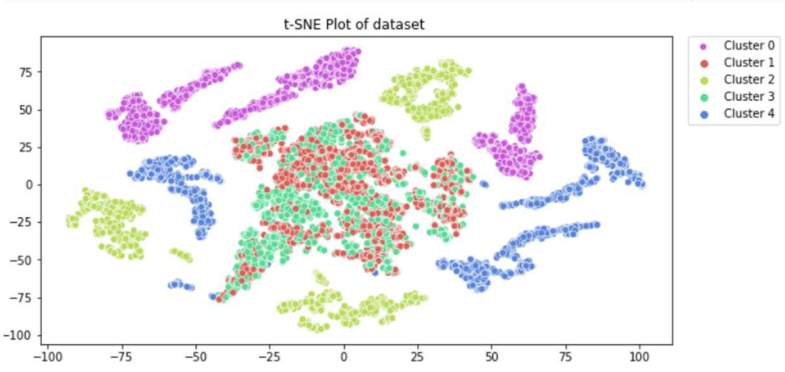

Figure 124 Actual DBSCAN T-SNE plot for training

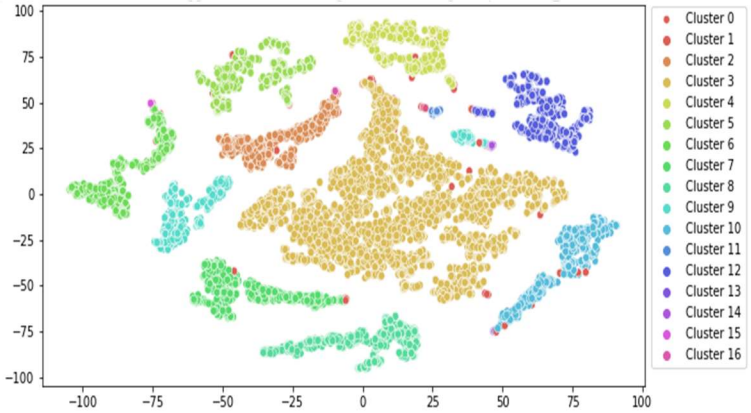

Figure 123 Predicted DBSCAN T-SNE plot for training

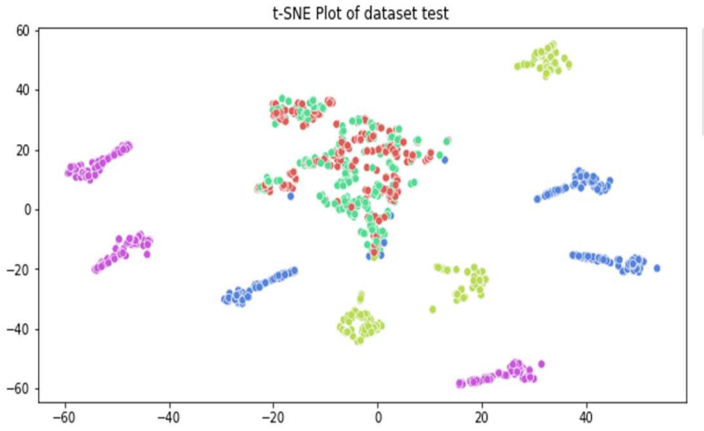

Figure 121 Actual DBSCAN T-SNE plot for testing

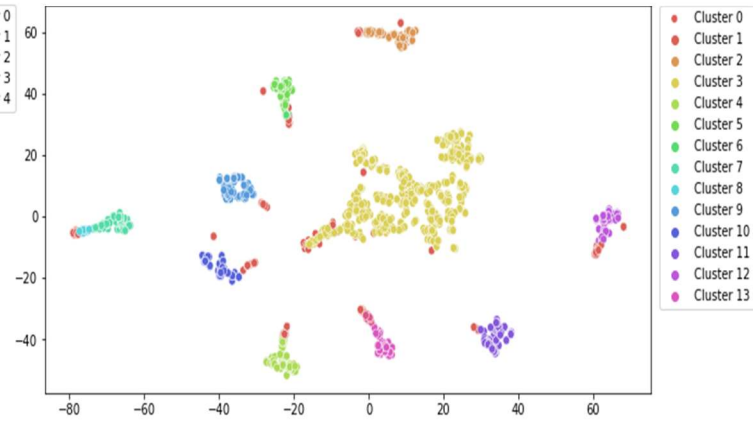

Figure 122 Predicted DBSCAN T-SNE plot for testing

From the T-SNE plot on training and testing, cluster 3 contains all classes 1,3 there are not purely classes.

3.5.3 GMM model

For the first experiment:

The following figures show the cluster distribution which indicates that, the algorithm can isolate class5 well on the training dataset but is not able to do the same on testing dataset, and the train and test distributions are not consistent.

pred	0	1	2	3	4	5	6	7	8	9	10
true											
1	5	2451	0	0	0	0	0	0	0	0	0
2	21	0	0	0	0	0	747	0	0	746	747
3	48	2231	0	0	0	0	0	1	0	0	0
4	53	71	0	0	0	703	0	748	754	0	0
5	0	0	787	781	787	0	0	0	0	0	0

Figure 126 Cluster distribution for training

pred	0	1	2	3	4	5	6	7	8	9	10
true											
1	1	307	0	0	0	0	0	0	0	0	0
2	10	0	0	0	0	0	90	0	0	95	92
3	14	266	0	0	0	0	0	0	0	0	0
4	13	9	0	0	0	93	0	90	86	0	0
5	10	0	94	92	98	0	0	0	0	0	0

Figure 125 Cluster distribution for testing

The following figures show the TSNE plot for GMM model on training and testing with actual labels and with predicted labels with number of components = 11:

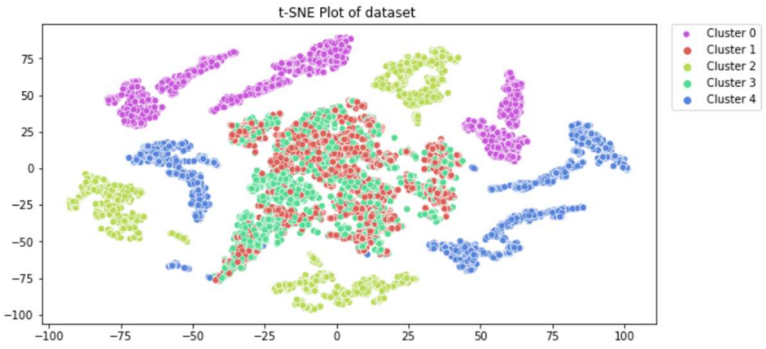

Figure 129 T-SNE plot for actual training data

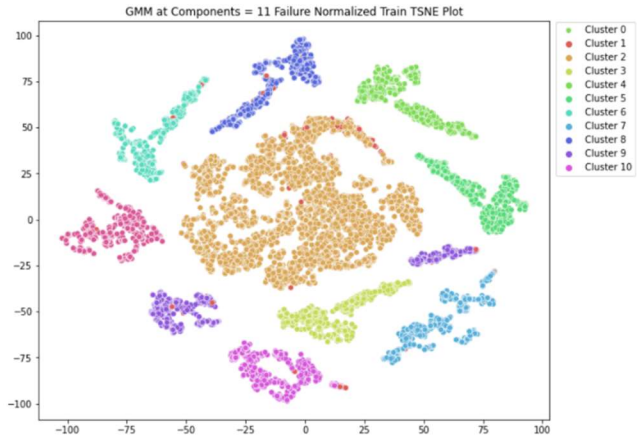

Figure 128 T-SNE for predicted training data

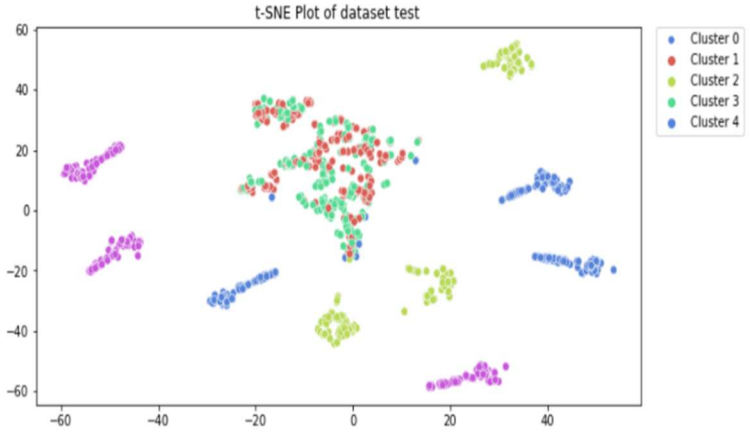

Figure 130 T-SNE plot for actual testing data

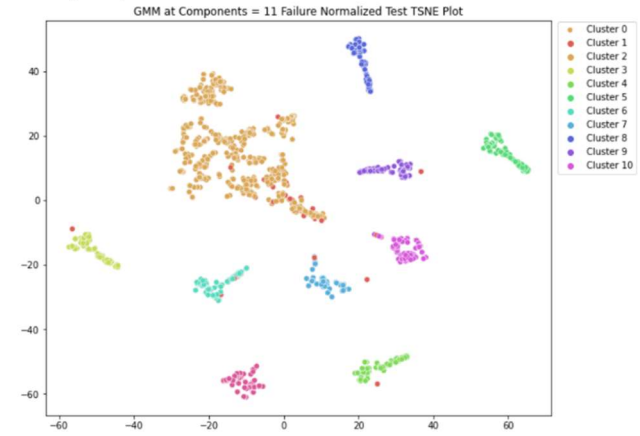

Figure 127 T-SNE for predicted testing data

From the TSNE plot on training, cluster1 contains most of class1, class3 and minority of class 4, but class5 is well separated.

For testing, the following figures show that cluster zero contains some points of class5, so it's not well separated. But from the true TSNE some of class5 points are close to class1 and class3.

For the second experiment:

The following figures show the cluster distribution which indicates that, the algorithm can isolate class5 well on the training dataset but is not able to do the same on testing dataset, and the train and test distributions are not consistent.

pred	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
true															
1	551	3	0	0	0	0	0	0	0	0	0	0	0	0	1902
2	0	135	0	0	0	0	723	0	0	678	725	0	0	0	0
3	1256	7	0	0	0	0	0	1	0	0	0	0	0	0	1016
4	79	0	0	0	0	715	0	455	472	0	0	293	0	307	8
5	0	0	787	384	787	0	0	0	0	0	0	0	397	0	0

Figure 132 Cluster distribution for training

pred	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
true															
1	85	1	0	0	0	0	0	0	0	0	0	0	0	0	222
2	0	33	0	0	0	0	84	0	0	81	89	0	0	0	0
3	179	2	0	0	0	0	0	0	0	0	0	0	0	0	99
4	7	0	0	0	0	97	0	47	52	0	0	39	0	46	3
5	0	5	94	48	98	0	0	0	0	5	0	0	44	0	0

Figure 131 Cluster distribution for testing

The following figures show the TSNE plot for GMM model on training and testing with actual labels and with predicted labels with number of components = 15:

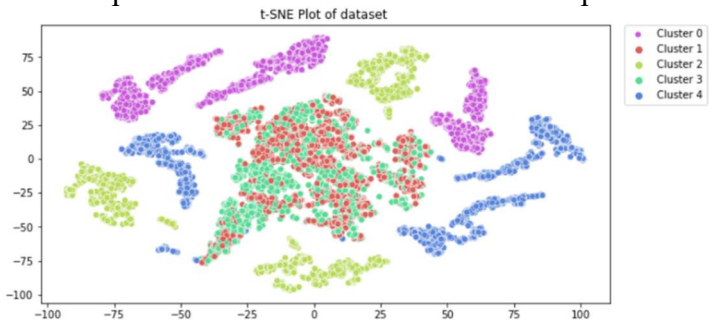

Figure 133 T-SNE plot for actual training data

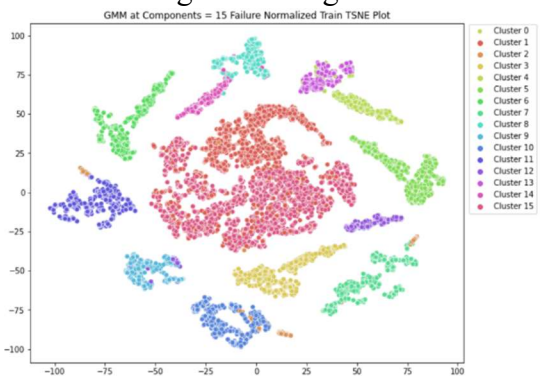

Figure 134 T-SNE for predicted training data

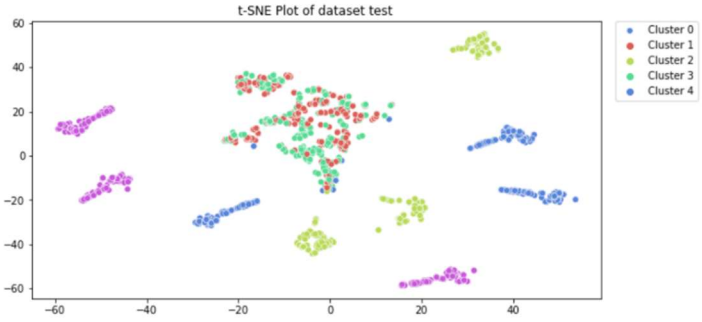

Figure 136 T-SNE plot for actual testing data

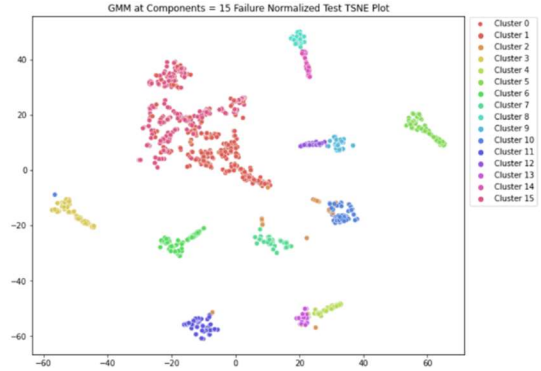

Figure 135 T-SNE for predicted testing data

From the T-SNE plot on the predicted training data, some of the classes are overlapped, but class 5 is well separated.

For testing, the following figures show that cluster 1 contains some points of class 5, so it's not well separated.

3.5.4 K-means model:

For third experiment:

The following figures show the cluster distribution which indicates that, the algorithm cannot isolate any class perfectly on both training and testing datasets, but the train and test distributions are consistent.

pred	0	1	2	3	4	5	6	7	8	9	10
true											
1	2354	0	0	0	0	0	0	102	0	0	0
2	0	0	0	0	0	0	742	33	743	0	743
3	2187	0	0	0	1	0	0	92	0	0	0
4	76	0	0	0	724	669	0	148	0	712	0
5	0	781	768	787	0	0	0	19	0	0	0

Figure 137 Cluster distribution for training

pred	0	1	2	3	4	5	6	7	8	9	10
true											
1	297	0	0	0	0	0	0	11	0	0	0
2	0	0	0	0	0	0	91	9	93	0	94
3	262	0	0	0	0	0	0	18	0	0	0
4	7	0	0	0	80	88	0	37	0	79	0
5	0	97	94	98	0	0	0	5	0	0	0

Figure 138 Cluster distribution for testing

The following figures show the T-SNE plot for k-means model on training and testing with actual labels and predicted labels with number of components = 11:

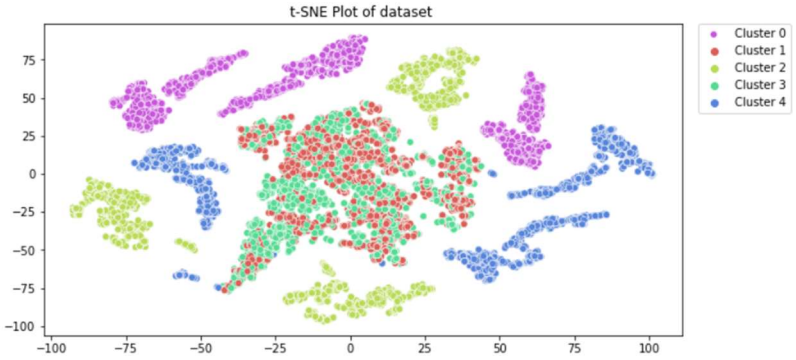

Figure 141 T-SNE plot for actual training data

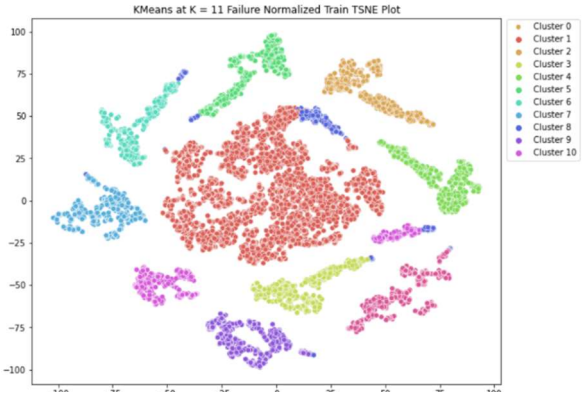

Figure 142 T-SNE for predicted training data

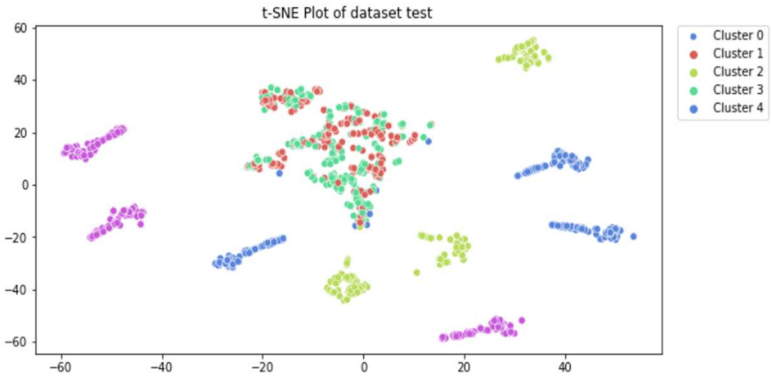

Figure 139 T-SNE plot for actual testing data

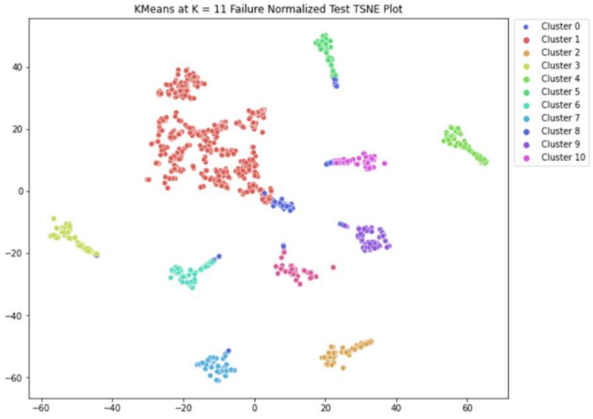

Figure 140 T-SNE for predicted testing data

From the training T-SNE plot, Cluster seven is distributed across all the true classes and causes no class to be well separated.

From the testing T-SNE plot, Cluster seven is distributed across all the true classes and causes no class to be well separated also k-means is providing consistent cluster distribution and cluster numbers across the test dataset.

For the fourth experiment:

The following figures show the cluster distribution which indicates that, the algorithm can isolate class 2 and class 5 well on both training and testing datasets and the train and test distributions are consistent.

pred	0	1	2	3	4	5	6	7	8	9
true										
1	2456	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	752	0	0	755	754
3	2278	0	0	0	0	0	2	0	0	0
4	80	0	0	720	0	0	763	766	0	0
5	0	781	787	0	787	0	0	0	0	0

Figure 144 Cluster distribution for training

pred	0	1	2	3	4	5	6	7	8	9
true										
1	308	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	93	0	0	96	98
3	280	0	0	0	0	0	0	0	0	0
4	7	0	0	97	0	0	94	93	0	0
5	0	97	98	0	99	0	0	0	0	0

Figure 143 Cluster distribution for testing

The following figures show the T-SNE plot for k-means model on training with actual labels and training with predicted labels with number of components = 10:

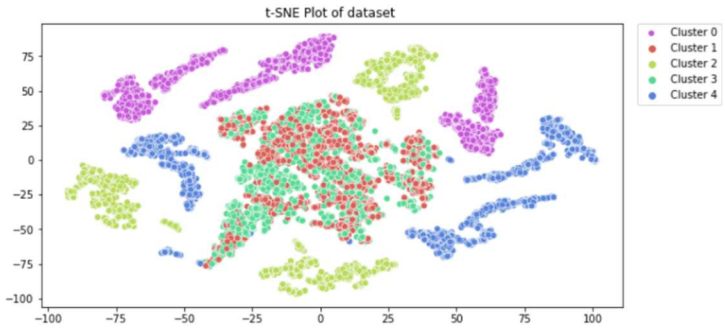

Figure 145 T-SNE plot for actual training data

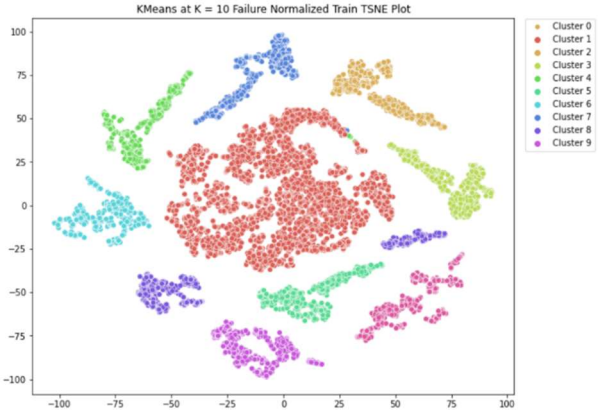

Figure 146 T-SNE for predicted training data

From the training TSNE plot, Cluster zero is nearly the only impure cluster containing the two overlapping class1, class3 and a minority of class 4.

From the testing TSNE plot, Cluster zero is nearly the only impure cluster containing the two overlapping class1, class3 and a minority of class 4, also KMeans is providing consistent cluster distribution and cluster numbers across the test dataset.

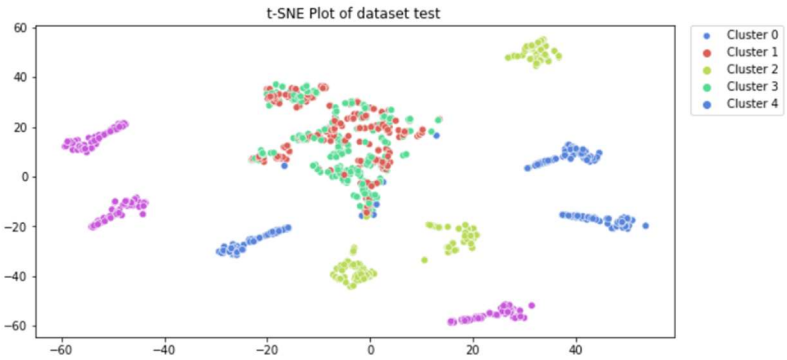

Figure 148 T-SNE plot for actual testing data

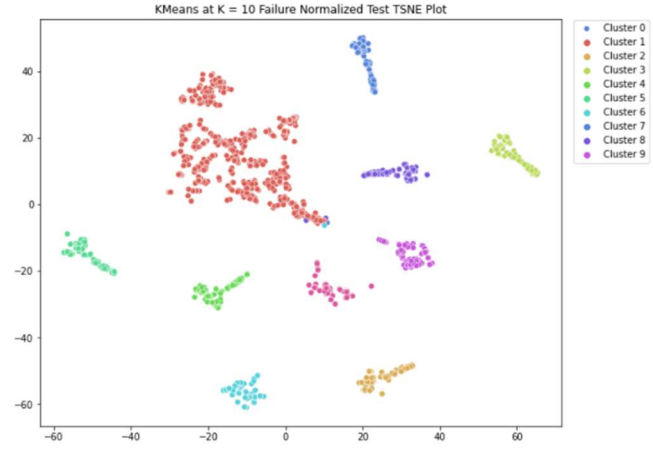

Figure 147 T-SNE for predicted testing data

For the fifth experiment:

Cluster Zero Trained XGBoost Train Classification_report

	precision	recall	f1-score	support
1	1.00	1.00	1.00	2456
3	1.00	1.00	1.00	2278
4	0.99	1.00	0.99	80
accuracy			1.00	4814
macro avg	1.00	1.00	1.00	4814
weighted avg	1.00	1.00	1.00	4814

Figure 149 Training classification report

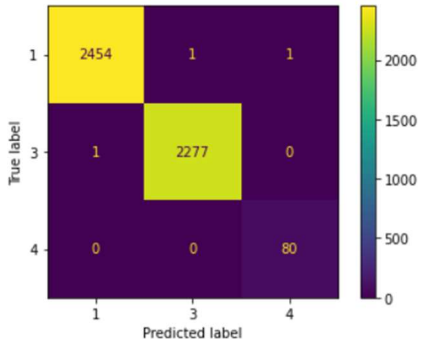

Figure 150 Training confusion matrix

Cluster Zero Trained XGBoost Test Classification_report

	precision	recall	f1-score	support
1	0.80	0.93	0.86	308
3	0.89	0.76	0.82	280
4	0.00	0.00	0.00	7
accuracy			0.84	595
macro avg	0.56	0.56	0.56	595
weighted avg	0.83	0.84	0.83	595

Figure 151 Testing classification report

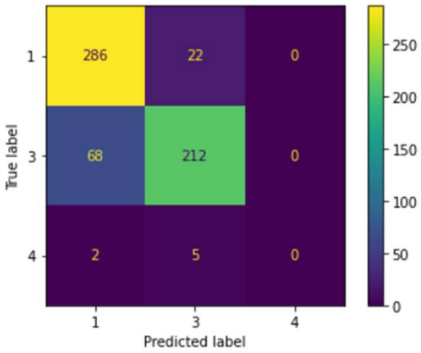

Figure 152 Testing confusion matrix

Based in F1-Score, the model trained well to classify the overlapped classes, class 1, class 3 and class 4. But according to the classification report of testing, based on F1-Score, the model can't classify class 4 anymore.

According to the confusion matrix class 1 and class 3 forms most of the data points in this cluster, which are the two target classes, that have been used to reduce the overlapping in this cluster, but class four points were misclassified as they are a minority in this class.

For the sixth experiment:

Based on F1-Score, SVM model Cannot classify any of the classes correctly, even though it is thought to be good at separating the overlapped data in the higher dimension due to the kernel trick and the confusion matrix prove that.

evaluation on training

	precision	recall	f1-score	support
1	0.48	0.94	0.64	2456
3	0.80	0.39	0.52	2280
4	0.76	0.39	0.51	2329
accuracy			0.58	7065
macro avg	0.68	0.57	0.56	7065
weighted avg	0.68	0.58	0.56	7065

Figure 154 Training classification report

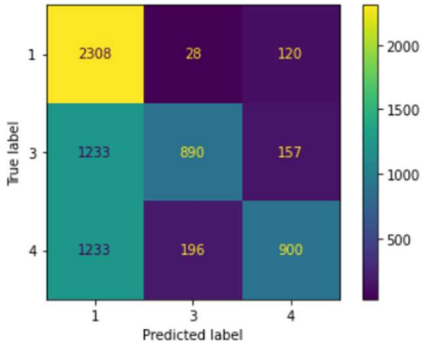

Figure 153 Training confusion matrix

evaluation on testing

	precision	recall	f1-score	support
1	0.51	0.95	0.66	308
3	0.87	0.42	0.57	280
4	0.79	0.46	0.58	291
accuracy			0.62	879
macro avg	0.72	0.61	0.60	879
weighted avg	0.72	0.62	0.60	879

Figure 156 Testing classification report

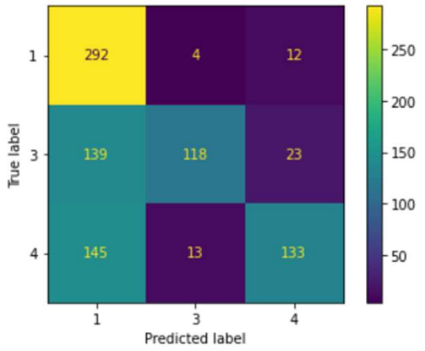

Figure 155 Testing confusion matrix

Based in F1-score, the XGboost model trained well to classify the overlapped classes, class 1, class 3 and class 4. But according to the classification report of testing, based on F1-score, the model can classify all the classes and the confusion matrix prove that.

evaluation on training

	precision	recall	f1-score	support
1	1.00	1.00	1.00	2456
3	1.00	1.00	1.00	2280
4	1.00	1.00	1.00	2329
accuracy			1.00	7065
macro avg	1.00	1.00	1.00	7065
weighted avg	1.00	1.00	1.00	7065

Figure 157 Training classification report

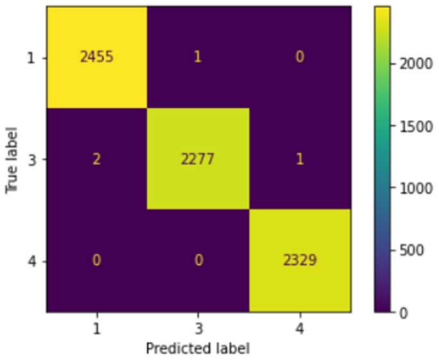

Figure 158 Training confusion matrix

evaluation on testing

	precision	recall	f1-score	support
1	0.83	0.93	0.88	308
3	0.91	0.79	0.85	280
4	1.00	1.00	1.00	291
accuracy			0.91	879
macro avg	0.91	0.91	0.91	879
weighted avg	0.91	0.91	0.91	879

Figure 159 Testing classification report

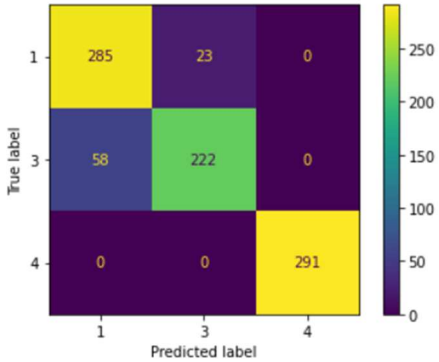

Figure 160 Testing confusion matrix

For the seventh experiment:

From the following classification report, the combined model can separate between the classes well and the F1-Score indicates that.

	precision	recall	f1-score	support
1	0.83	0.93	0.88	308
2	1.00	1.00	1.00	287
3	0.91	0.79	0.85	280
4	1.00	1.00	1.00	291
5	1.00	1.00	1.00	294
accuracy			0.94	1460
macro avg	0.95	0.94	0.94	1460
weighted avg	0.95	0.94	0.94	1460

Figure 161 Testing classification report

From the classification report for the methodology of using clustering in combination with classification has improved the accuracy by about 1.5% as well as increased the precision and recall for the overlapping classes 1 and 3. The following comparison proved that:

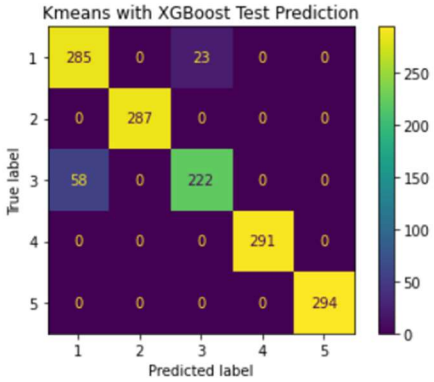

Figure 163 Combined confusion matrix (Clustering + classification modeling)

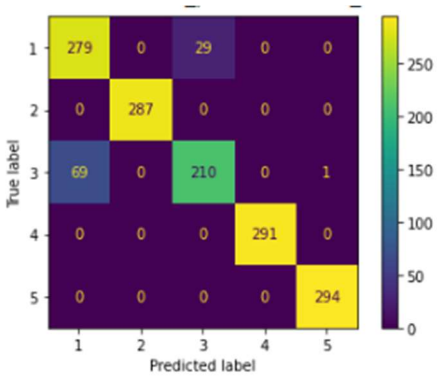

Figure 162 Combined confusion matrix (cascaded modeling)

These results in network failure classification part meet the requirements of the project as it achieved a very good accuracy by being able to completely identify 3 failure types out of five as well as took a systematic approach to reduce the overlap between the other two types and improve the result.

3.6 Beamforming selection

For the first experiment:

The following graph shows the top 10 accuracies which give the best result, but the overall result is very poor, so hyperparameter tuning must be applied to achieve high results.

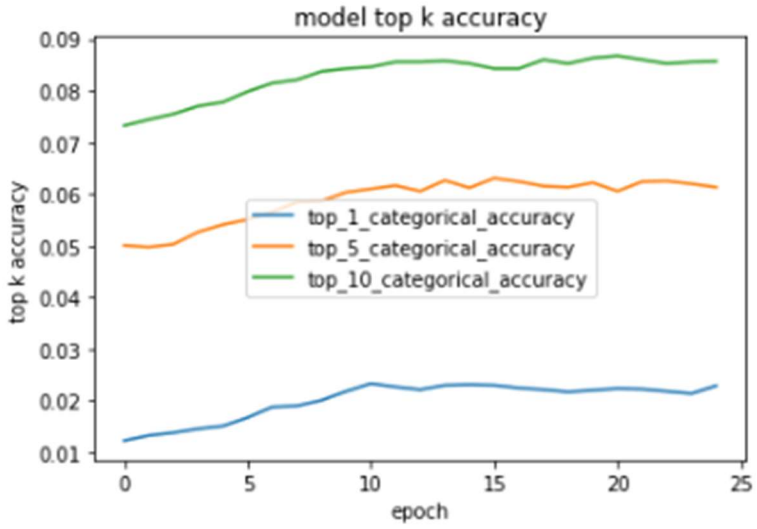

Figure 164 Experiment 1 top k accuracies vs number of epochs

The following curve shows that, there is no learning in this model and the weight almost constant

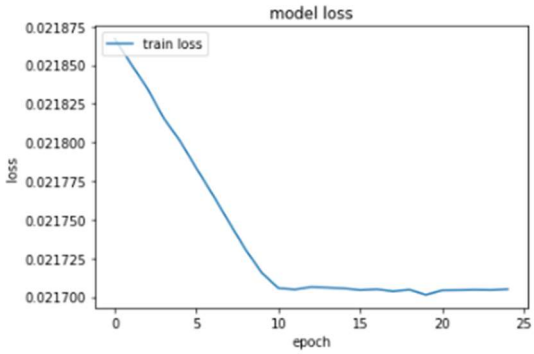

Figure 165 Experiment 1 training loss vs number of epochs

The following figure shows how the learning rate changes after 10 epochs to prevent exploding gradient problem.

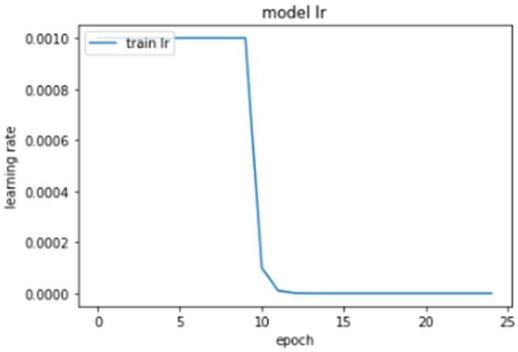

Figure 166 Experiment 1 training learning rate vs number of epochs

The following table shows that, this model doesn't meet the expectation and there is underfitting problem due to the result between training, validation and testing very poor.

Top K Accuracy	Training data	Validation data	Testing data
Top 1	0.0229	0.0291	0.0170
Top 5	0.0613	0.0735	0.0518
Top 10	0.0857	0.0893	0.0737

Table 24 Experiment 1 accuracies of different top k accuracies using all datasets

For the second experiment:

The following graph shows the top 10 accuracies curve on 25 epochs which achieves high result.

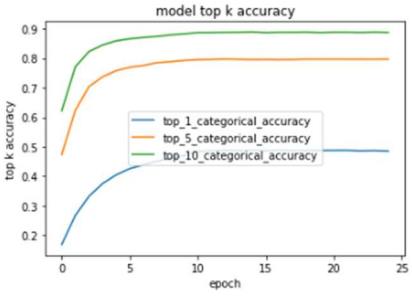

Figure 167 Experiment 2 top k accuracies vs number of epochs

The following curve shows that, after epoch number10, the loss function is almost constant.

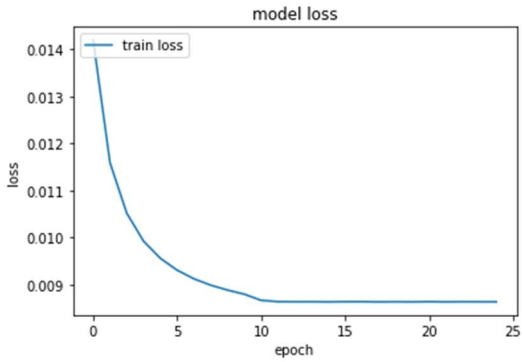

Figure 168 Experiment 2 training learning rate vs number of epochs

The following figure shows how the learning rate changes after 10 epochs to prevent exploding gradient problem.

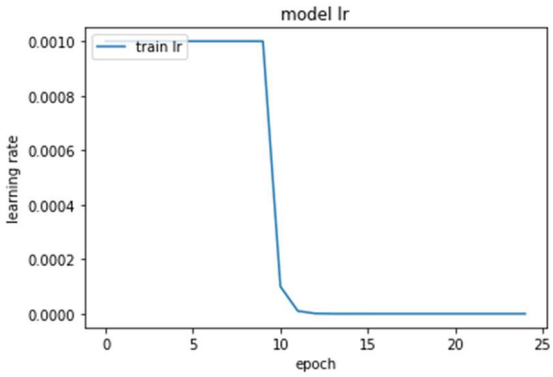

Figure 169 Experiment 2 training learning rate vs number of epochs

The following table shows that, this model doesn't meet the expectation and there is underfitting problem due to the result between training, validation and testing very poor.

Top K Accuracy	Training data	Validation data	Testing data
Top 1	0.4845	0.6316	0.5076
Top 5	0.7966	0.8985	0.8337
Top 10	0.8870	0.9327	0.8961

Table 25 Experiment 2 accuracies of different top k accuracies using all datasets

For the third experiment:

The following graph shows the top 10 accuracies curve on 25 epochs which achieves high result.

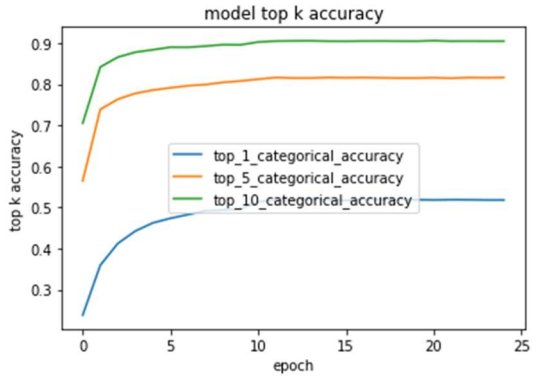

Figure 170 Experiment 3 top k accuracies vs number of epochs

The following curve shows that, after epoch number10, the loss function is almost constant.

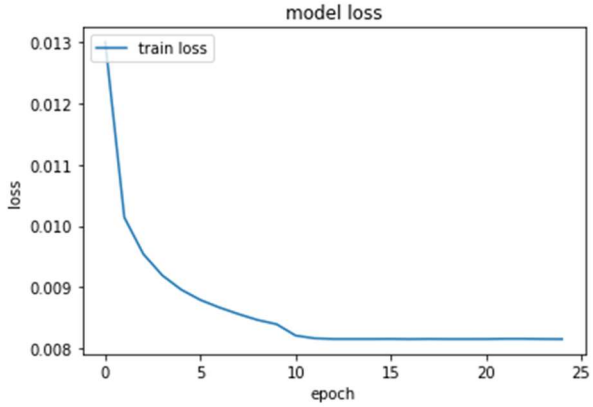

Figure 171 Experiment 3 training learning rate vs number of epochs

The following figure shows how the learning rate changes after 10 epochs to prevent exploding gradient problem.

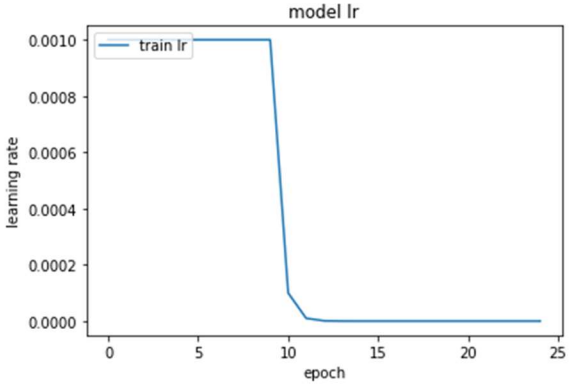

Figure 172 Experiment 3 training learning rate vs number of epochs

The following table shows that, this is the champion model which meets the design requirements, as there is no overfitting problem due to the result between training, validation and testing almost closed.

Top K Accuracy	Training data	Validation data	Testing data
Top 1	0.4845	0.6316	0.5076
Top 5	0.7966	0.8985	0.8337
Top 10	0.8870	0.9327	0.8961

Figure 173 Experiment 3 accuracies of different top k accuracies using all datasets

The next figure shows the test sample to predict the top transmitters and receiver that maximize the gain.

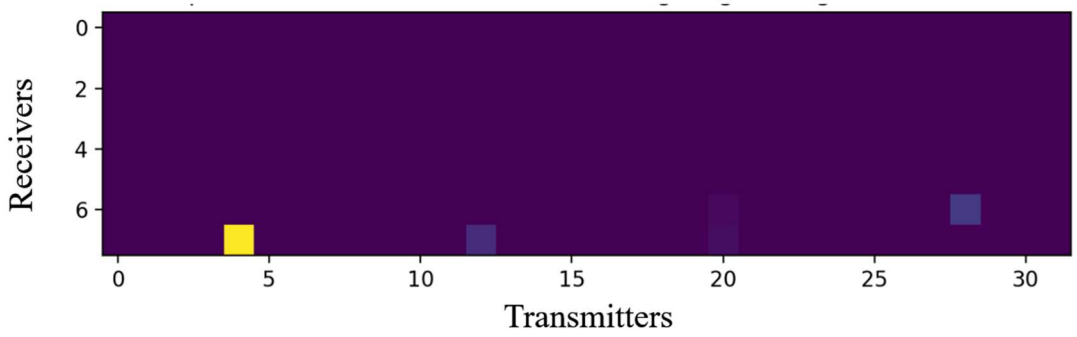

Figure 174 Top k-pairs visualization between transmitters and receivers

For the federated learning:

Result after aggregation models on Centralized server side:

```
[('top 1 categorical accuracy', 0.51634806), ('top 10 categorical accuracy', 0.9015544)
```

Figure 175 Results after aggregation in federated learning

4. Deployment Plan

The two types of the used data in this project were deployed twice each. Regarding the tabular data, the cascaded model described in 2.2.3 was deployed using amazon web services (AWS) using streamlit in two successful deployments. For the beamforming selection part, the CNN model from 2.2.6 was also deployed on AWS in two successful deployments. Firstly, the used services are: Amazon EC2 instances, Amazon ECR, Amazon ECS, AWS Fargate and other less impact features.

Amazon EC2 (elastic compute cloud), is a service of renting and accessing virtual computers that can have different specifications and amazon machine images (AMIs) which are basically their running software. Amazon ECR (elastic container registry), is a fully managed docker container registry that makes it easy to store, share, and deploy container images. Amazon ECS (elastic container service) is a container orchestration service, generally, a created cluster by Amazon ECS can easily run an entire application since it supports docker containers and can manage a cluster of Amazon EC2 instances that can be connected to different storage services on AWS as well. Finally, AWS Fargate is a serverless compute engine that has pay-as-you-go feature that the user only pays for what is being used, it can run Amazon EC2 containers without to manage them or their clusters.

Both of the first and second deployments have the same output which is the application running on AWS with the use of streamlit. In addition to the python file that uses binary class classification model file and multi-class classification model file which are both the cascaded modeling, nevertheless, a docker file and requirements file are being used so that the docker image can be created correctly and without errors when it run in a container, so, the Amazon EC2 created in deployment 1 is for creating the docker image, test it by running a local container and push that docker image to Amazon ECR repository to be used in deployment 2 later on.

The following figure illustrates the first two deployments which are using the cascaded modeling and its extracted models as reference:

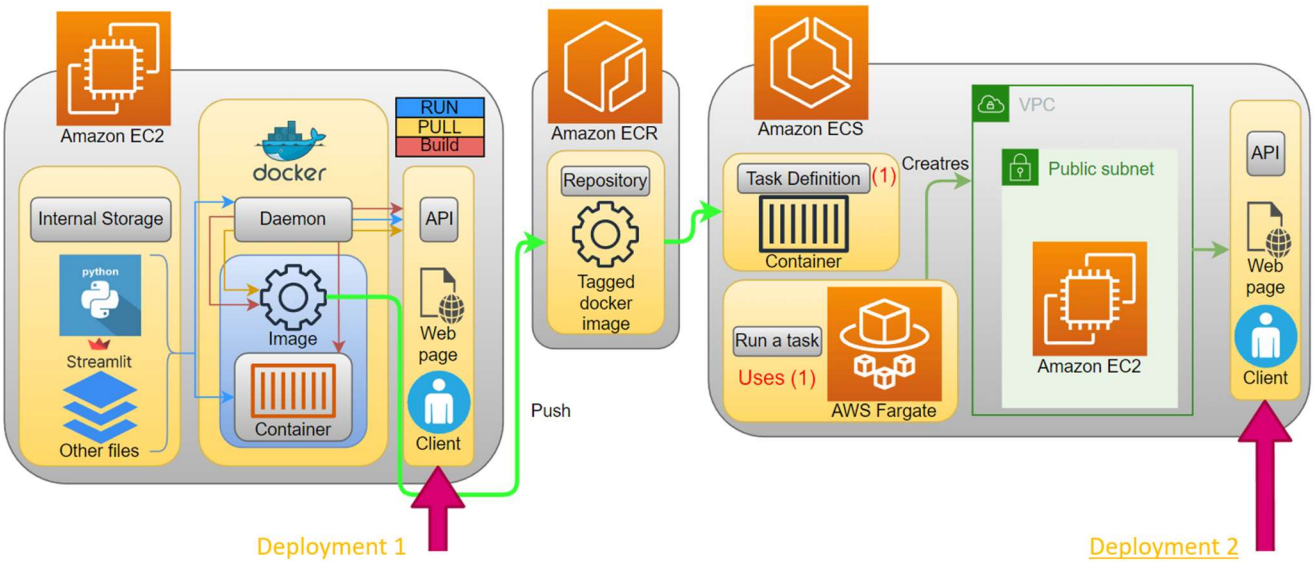

Figure 176 First two deployments on AWS

In deployment 1:

This is the initial deployment of cascaded model, firstly, using a t2.micro Amazon EC2 instance that runs on ubuntu 22.04, from the cascaded model files stored on local storage that uses the XGboost model all was sent from local PC to the EC2 instance earlier, a docker image and container have been created using the daemon that listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. Finally creating an API that allows the client to use the webpage.

In deployment 2:

Secondly, in addition to the first deployment until the creation of a docker image, the created docker image was pushed to a repository hosted by Amazon ECR. Then a cluster has been created using Amazon ECS which is a container orchestration service that uses AWS Fargate to run Amazon EC2 instance of the same type (t2.micro) without managing it. That created cluster uses a task definition option that has a previously created container. That task accessing a virtual private cloud (VPC) that contains the created Amazon EC2 instance to create the API and that's the full deployment for the tabular dataset.

Both of the third and fourth deployments have the same output which is the application running on AWS with the use of streamlit. In addition to the python file that uses the beamforming selection model, nevertheless, a docker file and requirements file both specified for these deployments and are being used so that the docker image can be created correctly and without errors when it run in a container, so, the Amazon EC2 created in deployment 3 is for creating the docker image, test it by running a local container and push that docker image to Amazon ECR repository to be used in deployment 4 later on.

The following figure illustrates the third and fourth deployments which are using the beamforming selection model and its extracted model as reference:

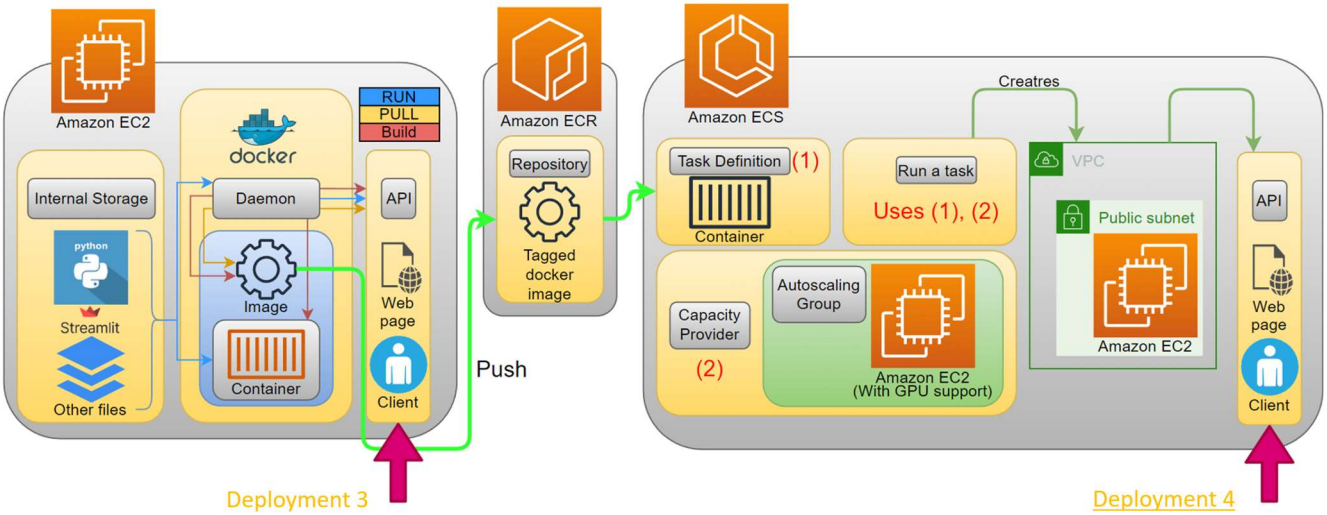

Figure 177 Third and fourth deployments on AWS

In deployment 3:

This deployment is almost typical to deployment 1 but the Amazon EC2 instance type is different which is g5.xlarge which h supports a GPU, also, the AMI is very specific which is ubuntu 20.04 but a version that supports deep learning and already has TensorFlow to be used by the instance as it is used by the created application. A docker image and docker container were created locally and created an API that allows the client to use the webpage.

In deployment 4:

This deployment is relatively close to deployment 2 but there are some differences. The created docker image in deployment 3 was pushed to a repository on Amazon ECR. The differences are in Amazon ECS, which in running a task, additionally to using a task definition, a capacity provider option is used that choses an autoscaling group that maintains a selected number of instances to be created to perform a specific task and it was fixed to 1 EC2 instance of the same type (g5.xlarge), the task then access a virtual private cloud (VPC) that contains the created Amazon EC2 instance which runs the docker image by a container previously created in task definition to create the API and that's the full deployment for the beamforming selection dataset.

Here are screenshots below for deployment 2 (similar to deployment 1 since they have the same output):

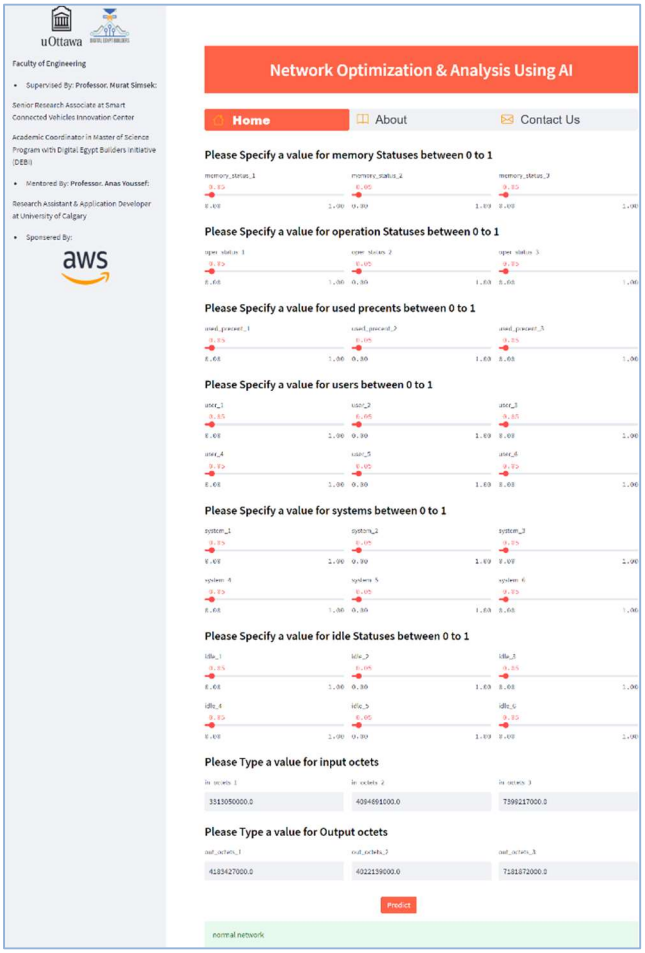

Figure 180 Screenshot for deployment 2 (home)

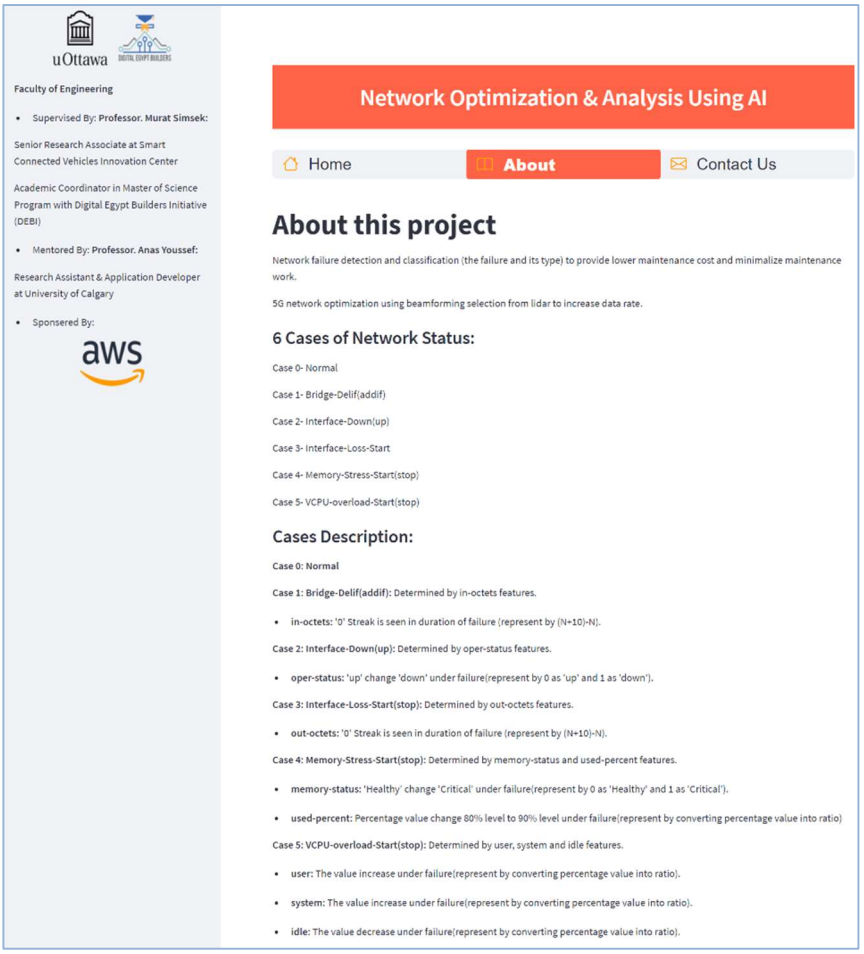

Figure 179 Screenshot for deployment 2 (about)

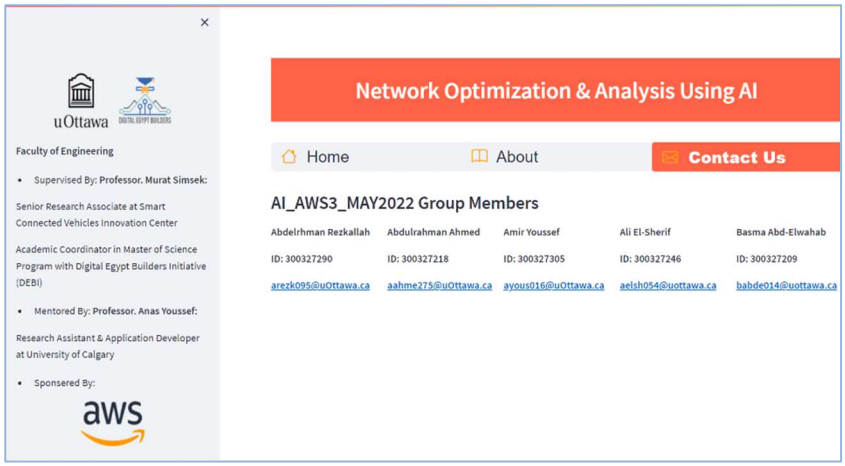

Figure 178 Screenshot for deployment 2 (contact us)

As the "home" tab showing, the model can predict whether the network is normal or there is one of the five types of failure. "about" tab shows the description and all the failing conditions. "Contact us" tab shows the contributors' contact information.

Here are screenshots below for deployment 4 (similar to deployment 3 since they have the same output):

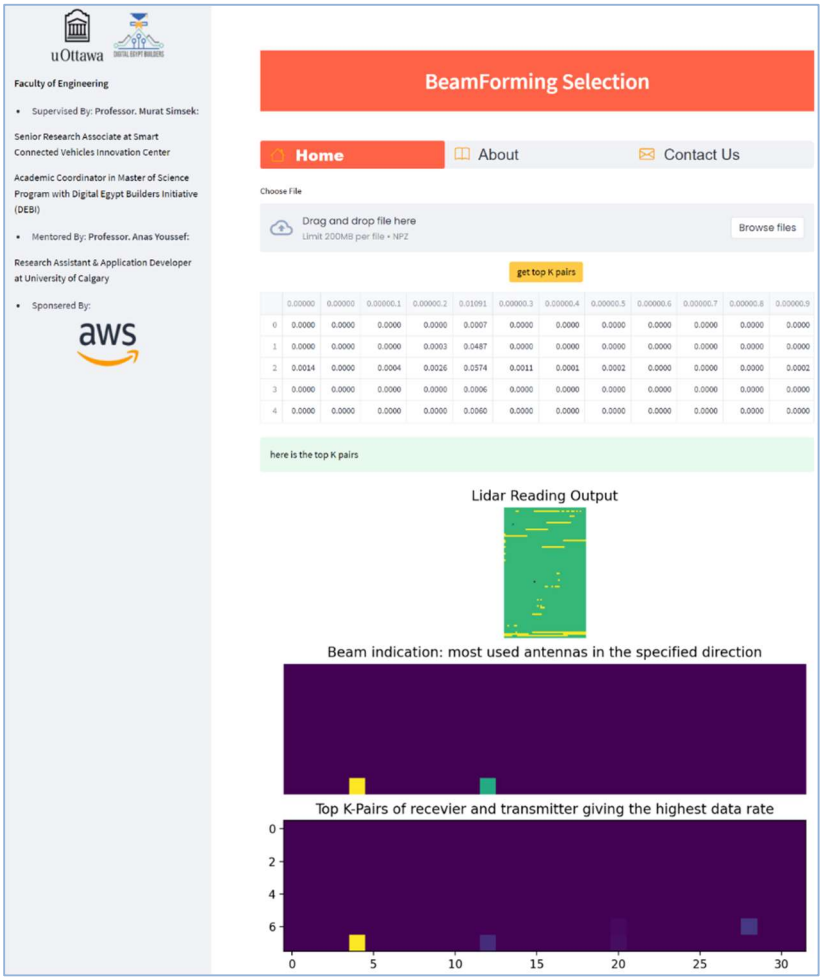

Figure 181 Screenshot of deployment 4 (home)

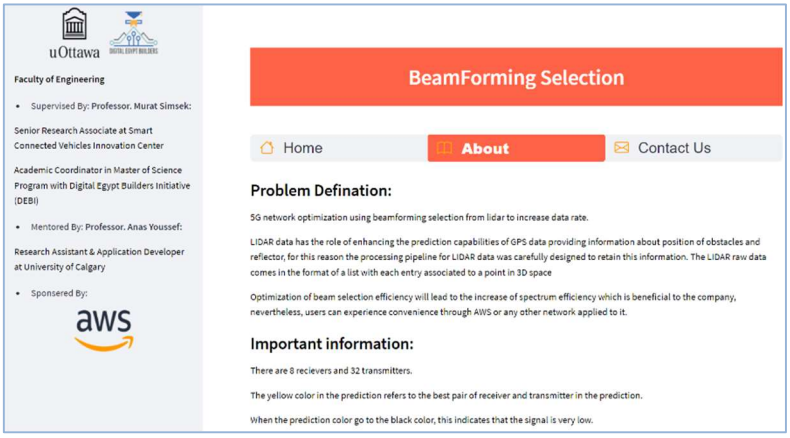

Figure 183 Screenshot of deployment 4 (about)

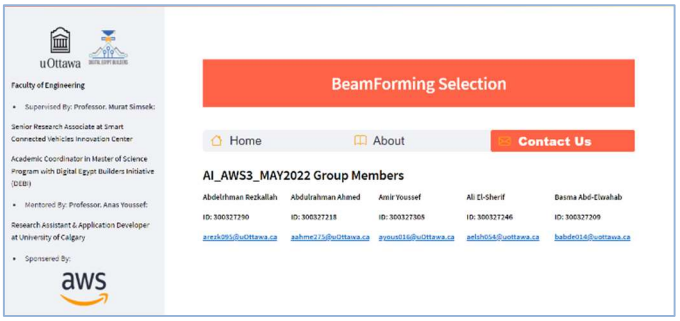

Figure 182 Screenshot of deployment 4 (contact us)

In “home” tab, when pressing on “get top k pairs”, the network webpage shows the LiDAR reading output and the original beam indication and the one with the highest data rate according to the top k-pairs between transmitters and receivers. “about” shows the problem definition and other information about this part. “Contact us” shows the contact information.

Deployment Considerations:

	Cascaded model	Beamforming selection model
Amazon EC2 instance	T2.micro	G5.xlarge
Amazon EC2 AMI	ubuntu-jammy-22.04-amd64-server-20221201	Deep Learning AMI GPU TensorFlow 2.11.0 (Ubuntu 20.04) 20221220
Docker version	20.10.22, build 3a2c30b	20.10.22, build 3a2c30b
Used models	Xgboost(v0.90)	tensorflow.keras.Sequential Tensorflow(v2.9.0)
Dependencies (EC2s and docker images)	Numpy, pandas, streamlit, streamlit_option_menu, joblib, scikit-learn, matplotlib, pillow	Streamlit, streamlit_option_menu, pillow, joblib, matplotlib, tensorflow_hub, numpy, python-time, pandas, protobuf(v3.20.1)
Amazon ECR	✓	✓
Amazon ECS	✓	✓
Capacity provider	×	✓
AWS Fargate	✓	×
Amazon CLI v2	V2	V2
PuTTY (EC2 access)	✓	✓

Table 26 Deployment considerations

Deployment cost (production scale on us-east-1 region of AWS):

		Cascaded model		Beamforming selection model	
		Deployment 1	Deployment 2	Deployment 3	Deployment 4
EC2 (T2.micro)		0.0116$/hour, 8.47$/month	0.0116$/hour, 8.47$/month	×	×
EC2 (G5.Xlarge)		×	×	1.006$/hour, 734.38$/month	1.006$/hour, 734.38$/month
Elastic IP		3.65$/month	3.65$/month	3.65$/month	3.65$/month
Elastic block store (EBS)		2.4$/month	2.4$/month	2.4$/month	2.4$/month
Elastic container registry (ECR)		69.82$/month	69.82$/month	144.75$/month	144.75$/month
Total (per month)	969.52$		84.34$		885.18$

Table 27 Deployment cost per month for both applications

The previous table shows the deployment monthly cost of applying deployment 2 and 4 in a real-world scenario.

5. Conclusions and Future Works

Comparison between cascaded modeling and baseline model:

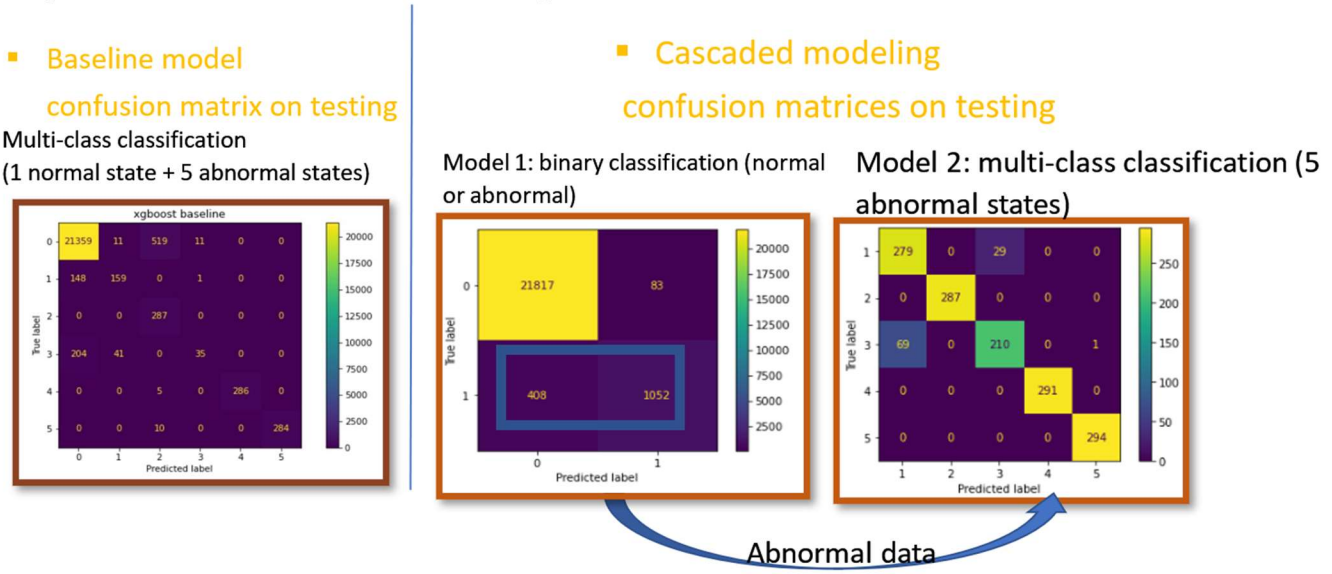

Figure 184 Confusion matrices of cascaded modeling and baseline model

While training XGboost model with all of the data that contains normal and abnormal cases it achieved good result on normal cases but, when it comes to failure cases it achieves slightly good results that is why cascaded approach has been applied. The main idea of the cascaded approach is to separate the data into normal and abnormal, then apply XGBOOST binary classifier model, if the result is normal the model is stopping here, if the result is abnormal, XGBOOST multiclass classification will be trained to give a result with one of five failure types, which achieves good results on predicting the abnormal failure types.

Comparison between cascaded modeling and Clustering approach:

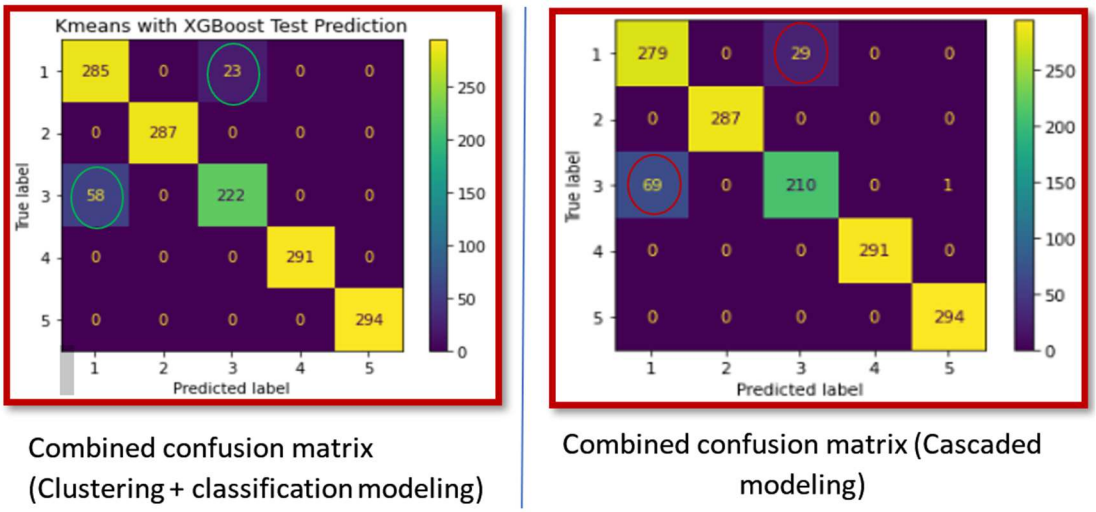

Figure 185 Confusion matrices of cascaded modeling and clustering modeling

After applying clustering approach to separate the pure classes and detect impure classes then classification approach using XGBOOST model has been applied to the impure clusters and the k-means clustering model can separate classes 2 and 5 clearly, cluster 0 is overlapped between class 1, 3 and 4. After applying XGBOOST classification model on classes 1, 3 and 4 that are included in cluster 0 and this approach achieves best result than the cascaded approach. The confusion matrices above show that there is an improvement in class 1 and class 3, this improvement increase the accuracy with 1.5 percent.

Future work:

- Deploying federated learning on AWS.
- Applying beam forming selection part through a real scenario with the cooperation of an industrial company.

6. References

[1] M. H. Abidi *et al.*, "Optimal 5G network slicing using machine learning and deep learning concepts," *Computer Standards & Interfaces*, vol. 76, p. 103518, Jun. 2021, doi: 10.1016/j.csi.2021.103518.

[2] M. Z. Asghar, M. Abbas, K. Zeeshan, P. Kotilainen, and T. Hämäläinen, "Assessment of Deep Learning Methodology for Self-Organizing 5G Networks," *Applied Sciences*, vol. 9, no. 15, p. 2975, Jan. 2019, doi: 10.3390/app9152975.

[3] M. Dias, A. Klautau, N. González-Prelcic, and R. W. Heath, "Position and LIDAR-Aided mmWave Beam Selection using Deep Learning," *IEEE Xplore*, Jul. 01, 2019. https://ieeexplore.ieee.org/document/8815569 (accessed Aug. 10, 2022).

[4] A. Klautau, P. Batista, N. González-Prelcic, Y. Wang, and R. W. Heath, "5G MIMO Data for Machine Learning: Application to Beam-Selection Using Deep Learning," *IEEE Xplore*, Feb. 01, 2018. https://ieeexplore.ieee.org/document/8503086 (accessed Aug. 10, 2022).

[5] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang, and L.-C. Wang, "Deep Reinforcement Learning for Mobile 5G and Beyond: Fundamentals, Applications, and Challenges," *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, pp. 44–52, Jun. 2019, doi: 10.1109/MVT.2019.2903655.

[6] "How to Enhance 5G Network Planning Using Analytics for Faster Time to Market | AWS Partner Network (APN) Blog," *aws.amazon.com*, Nov. 09, 2021. https://aws.amazon.com/blogs/apn/how-to-enhance-the-5g-network-planning-using-analytics-for-faster-time-to-market/

[7] E. J. Khatib and R. Barco, "Optimization of 5G Networks for Smart Logistics," *Energies*, vol. 14, no. 6, p. 1758, Mar. 2021, doi: 10.3390/en14061758.

[8] "AWS Private 5G is now generally available," *Amazon Web Services, Inc.*, Aug. 11, 2022. https://aws.amazon.com/about-aws/whats-new/2022/08/aws-private-5g-now-generally-available/#:~:text=AWS%20Private%205G%20simplifies%20the

[9] "baseline_data," *LASSE Nextcloud*, 2022. https://nextcloud.lasseufpa.org/s/zqeeyDtbbPt7nKz?path=%2F

[10] "Datast Network Failure.zip," *Google Docs*. https://drive.google.com/file/d/16LG1j9Cv7YzalF5Qke22eGyfJxE0QdTc/view?usp=sharing

[11] "ML5G_challenge," *GitHub*, Dec. 11, 2020. https://github.com/ITU-AI-ML-in-5G-Challenge/PS-012-ML5G-PHY-Beam-Selection_BEAMSOUP (accessed Sep. 14, 2022).

[12] "New – AWS Private 5G – Build Your Own Private Mobile Network | AWS News Blog," *aws.amazon.com*, Aug. 11, 2022. https://aws.amazon.com/blogs/aws/new-aws-private-5g-build-your-own-private-mobile-network/ (accessed Jan. 15, 2023).

[13] J. Kawasaki, G. Mouri, and Y. Suzuki, "Comparative Analysis of Network Fault Classification Using Machine Learning," *IEEE Xplore*, Apr. 01, 2020. https://ieeexplore.ieee.org/document/9110454 (accessed Jan. 15, 2023).

[14] "Scalable Mobile Private 4G and 5G Network Services on AWS from Deloitte | AWS Partner Network (APN) Blog," *aws.amazon.com*, Sep. 01, 2021. https://aws.amazon.com/ar/blogs/apn/scalable-mobile-private-4g-and-5g-network-services-on-aws-from-deloitte (accessed Jan. 15, 2023).

[15] "ITU AI Challenge," *challenge.aiforgood.itu.int*. https://challenge.aiforgood.itu.int/match/matchitem/57?_ga=2.38058467.1554067894.1673330808-1369953310.1673330808 (accessed Jan. 15, 2023).

[16] "RAYMOBTIME – LASSE – Núcleo de P&D em Telecomunicações, Automação e Eletrônica." https://www.lasse.ufpa.br/raymobtime/ (accessed Jan. 15, 2023).