

COMMUNICATION CHANNEL FAILURE PREDICTION IN 5G  
NETWORKS

by

Mohammad Ariful Islam

Submitted in partial fulfillment of the requirements  
for the degree of Master of Computer Science

at

Dalhousie University  
Halifax, Nova Scotia  
February 2022

© Copyright by Mohammad Ariful Islam, 2022

# Table of Contents

<b>List of Tables</b> . . . . .	<b>iv</b>
<b>List of Figures</b> . . . . .	<b>v</b>
<b>Abstract</b> . . . . .	<b>vii</b>
<b>List of Abbreviations Used</b> . . . . .	<b>viii</b>
<b>Acknowledgements</b> . . . . .	<b>ix</b>
<b>Chapter 1 Introduction</b> . . . . .	<b>1</b>
1.1 Contribution . . . . .	3
1.2 Thesis Outline . . . . .	5
<b>Chapter 2 Background and Data Description</b> . . . . .	<b>6</b>
2.1 Background . . . . .	6
2.1.1 Data Pre-processing . . . . .	6
2.1.1.1 The Curse of High Dimensionality . . . . .	8
2.1.2 Dataset Balancing . . . . .	10
2.1.3 Learning Algorithms . . . . .	11
2.1.3.1 Ensemble Learning Method. . . . .	11
2.1.3.2 Deep Learning based Models . . . . .	14
2.2 Data Explanation . . . . .	18
<b>Chapter 3 Literature Review</b> . . . . .	<b>22</b>
3.1 Related Works . . . . .	22
3.1.1 Topology Based . . . . .	23
3.1.2 Machine Learning Based . . . . .	24
<b>Chapter 4 Research Methodology and Evaluation</b> . . . . .	<b>29</b>
4.1 Data Pre-Processing . . . . .	30
4.1.1 Distance correlation. . . . .	30
4.1.2 Data Combination . . . . .	33
4.1.3 Missing Data . . . . .	34
4.1.4 Categorical Feature Encoding . . . . .	37
4.1.5 Feature Selection . . . . .	38
4.1.6 Principal Component Analysis . . . . .	39
4.2 Data Preparation . . . . .	41

4.2.1	Data Segmentation . . . . .	43
4.2.2	Data Normalization . . . . .	45
4.3	Model Training . . . . .	45
4.3.1	Neural Network based learning Methods . . . . .	45
4.3.2	Supervised Ensemble Learning Method . . . . .	52
4.4	Evaluation . . . . .	54
4.4.1	Performance metrics . . . . .	55
4.4.2	Parameter Tuning . . . . .	56
4.4.3	Model Comparisons . . . . .	69
<b>Chapter 5</b>	<b>Conclusion and Future work . . . . .</b>	<b>72</b>
5.1	Conclusion . . . . .	72
5.2	Future Work . . . . .	72
<b>Bibliography</b>	. . . . .	<b>74</b>

## List of Tables

2.1	Summary of the weather data. . . . .	19
2.2	Summary of radio tower data. . . . .	20
2.3	Dataset table description . . . . .	21
3.1	A comparison of existing solutions for Link Failure / Anomaly Detection. . . . .	27
4.1	Principal Components VS Retained Variance. . . . .	41
4.2	Hyper-parameters of FCNN. . . . .	48
4.3	Hyper-parameters of LSTM-AE. . . . .	49
4.4	Hyper-parameters of ensemble learning. . . . .	55
4.5	Evaluation result for fully connected convolution model. . . . .	59
4.6	Prediction results for different encoding techniques (PCA applied). . . . .	62
4.7	Prediction results for different numbers of hidden layers. . . . .	65
4.8	Prediction results for different amounts of days in the past data being used with one hot encoded + PCA and 4 hidden layers of the model. . . . .	66
4.9	Evaluation result for supervised Ensemble Model. . . . .	68
4.10	Comparison of prediction results with state of the art approach. . . . .	70

## List of Figures

2.1	Example of Binary encoding operation. . . . .	7
2.2	Example of one hot encoding operation. . . . .	8
2.3	Synthetic Minority Oversampling. . . . .	11
2.4	Random Forest Classifier . . . . .	12
2.5	Light Gradient Boosting Machine leaf-wise growth. . . . .	13
2.6	LSTM Autoencoder Architecture . . . . .	17
4.1	The workflow of the proposed approach. . . . .	29
4.2	Correlation established based on the optimal distance $d$ . . . . .	31
4.3	Correlation heuristic flowchart. . . . .	32
4.4	Outlier weather stations. . . . .	32
4.5	Data Structure . . . . .	34
4.6	Data Quality report of the Continuous features . . . . .	35
4.7	Data quality report of the categorical features. . . . .	35
4.8	Binary encoding to categorical features. . . . .	37
4.9	One hot encoding to Categorical Features. . . . .	38
4.10	Feature Coefficient scores. . . . .	39
4.11	Data balancing operations using synthetic approach. . . . .	42
4.12	Expanding window Time series segmentation. . . . .	44
4.13	Fully connected CNN architecture and workflow. . . . .	46
4.14	LSTM - Autoencoder Architecture. . . . .	50
4.15	Ensemble method and its prediction strategy. . . . .	53
4.16	Train loss of FCNN for binary encoded and one hot encoded with PCA data. . . . .	57
4.17	ROC curve generated from different parameters tuning. . . . .	58
4.18	Reconstruction error of different encoding methods. . . . .	61

4.19	ROC curve representation for one hot encoded data with PCA of 2019. . . . .	62
4.20	Training reconstruction error of different structures of LSTM-AE.	63
4.21	ROC curve representation for one hot encoded + PCA data of 2019 with 4 hidden layers for LSTM-AE. . . . .	64
4.22	Reconstruction error for different numbers of lag steps. . . . .	65
4.23	ROC curve representation for one hot encoded data of 2019 with 4 hidden layers and five days past data observation. . . . .	67

## Abstract

5G networks enable emerging latency and bandwidth critical applications like industrial IoT, AR/VR, or autonomous vehicles in addition to supporting traditional voice and data communications. In the 5G infrastructure, Radio Access Networks (RANs) consist of radio base stations that communicate over wireless radio links. This communication, however, is prone to environmental changes, such as the weather. These links can suffer from radio link failure and subsequently interrupt ongoing services, severely impacting the above-mentioned applications. One way to mitigate such service interruption is to proactively predict failures and reconfigure the resource allocation accordingly. Existing works focused on such failure prediction (e.g., using supervised ensemble learning) do not considering the spatio-temporal correlation of radio communication and weather changes. In this work, we propose a communication link failure prediction model based on the LSTM autoencoder, i.e., considering both the spatio-temporal correlation of radio communication as well as weather changes. We implement and evaluate the proposed scheme over a huge volume of real radio and weather data. The results confirm that the proposed scheme performs better than the state-of-the-art solution.

## List of Abbreviations Used

<b>KPI</b>	Key Performace Indicator
<b>AODV</b>	Ad Hoc On-Demand Distance Vector
<b>IoT</b>	Internet of Things
<b>mmWave</b>	millimeter wave
<b>RLF</b>	Radio Link Failure
<b>ITU</b>	International Telecommunication Union
<b>BBE</b>	Background Bit Error
<b>DSDV</b>	Destination-Sequenced Distance-Vector Routing
<b>OLSR</b>	Optimized Link State Routing Protocol
<b>WiMAX</b>	Worldwide Interoperability for Microwave Access
<b>MANET</b>	Mobile Ad hoc network
<b>RFE</b>	Recursive Feature Elimination
<b>SMOTE</b>	Synthetic Minority Oversampling Technique
<b>KNN</b>	K-Nearest Neighbour
<b>PCA</b>	Principal Component Analysis
<b>LGBM</b>	Light Gradient Boosting Machine
<b>XGBoost</b>	Extreme Gradient Boosting Machine
<b>FCNN</b>	Fully Connected Feed-Forward Convolution Neural Network
<b>RNN</b>	Recurrent Neural Network
<b>LSTM</b>	Long- Short Term Memory Network
<b>MAE</b>	Mean Absolute Error
<b>GAN</b>	Generative Adversarial Network
<b>LSTM-AE</b>	Long- Short Term Memory Network-Autoencoder
<b>OCSVM</b>	One-Class Support Vector Machine
<b>ROC</b>	Receiver Operating Characteristic



## Acknowledgements

First of all, I would like to humbly thank my supervisor, Dr. Israat Haque, for guiding and supporting me during the MCS study and research despite her busy work schedule. Her continuous feed back and constant motivation driven me to achieve my desired output from the research. She carefully guided me throughout every step of the thesis to carry out extensive research and provided effort and knowledge as much as possible. I feel really honored to have worked under her supervision and thank her for continuous support during my study at Dalhousie.

I would also like to thank all of my fellow lab members from the Programmable and Intelligent Network (PINet) research lab for their help and opinion regarding my work. I would specially thank Dipon Saha, Hisham Sidiqqe, Miheer Kulkarni, Meysam Shojaee, Conrado Boeira, Kazi Hasan who are my lab mates, for their feedback on my work and help to complete my research. I am also thankful to Dr. Sageev Oore for his guidance and valuable feedback.

Finally, I want to thank my parents for their love, support, and selfless sacrifices that they made to send me to such a prestigious university for my further education. Without their support, I would not have been able to finish my work.

# Chapter 1

## Introduction

Emerging networking applications include Industry 4.0, intelligent transportation, smart health system, AR/VR, etc., that demand high network bandwidth, high reliability, and low communication time [1]. Mobile and wireless devices from these applications usually communicate over radio links and various types of networks like ad hoc, sensor, mesh, or cellular [2–7]. Fifth-generation (5G) cellular networks aim to support the above emerging applications out of these aforementioned networks. Unlike 4G networks with large and high-power cell towers to reflect signals over long distances, a 5G network consists of cells with a small coverage. This is due to the usage of millimeter-wave (mmWave) spectrum (between 24GHz and 100GHz) [8] in 5G that can travel over short distances, but is susceptible to interference from the weather and physical obstacles (e.g., buildings). There are various weather conditions like rain, fog, precipitation, and smog that cause these waves to attenuate during propagation. Extreme conditions may lead to the total link failure of the network. The mmWave, however, enables high-speed communications. Some estimates indicate that 5G wireless broadband connections can access data at speeds of 20 gigabits per second (Gbps) or higher [9]. It is expected that 5G services and networks will be deployed over the next several years to meet the growing demand for mobile devices and internet connectivity.

However, the use of mmWave has a number of disadvantages, besides the fact that the frequency is affected by the weather conditions. In addition to passing through the atmosphere, the mmWave radio frequency suffers from the frequency dependent absorption and dispersion effects which cause distortions in the amplitude and phase of signals [10]. Various properties of insulated media, such as the refraction, absorption and scatter, influence the regular propagation of mmWave [8]. This includes attenuation while propagating through water and the atmosphere. It can affect the parameters of the signals from the radio antenna that ultimately causes

the unavailability of the link transmission. Likewise, the wind plays an important role in the transmission of the mmWave waves as strong winds cause atmospheric vibration, which is responsible for attenuating the key performance indicator (KPI) parameters [11].

Emerging applications deployed over 5G demand reliable and fast communications, wherein the system must be robust against any radio link failure. Thus, in this thesis we focus on learning-based radio link failure prediction. We use the radio station (cellular tower) and real-world weather data sourced from Turkish networking company ITU-Turkcell [12]. The radio station data contains KPI parameters for the radio sensors collected on a daily basis in 2019, as well as the spatial information of each radio tower. In addition, the weather data from the weather stations scattered throughout the same region were provided by the Turkish National Meteorological Institute. The weather data contains forecast data for the next five days from any date of 2019, collected in the mornings and evenings of each day. This data also contains the real weather report of the region captured hourly everyday. The ITU-Turkcell network operators set two thresholds for the unavailability of a link based on time (in seconds) and background bit error (BBE). A radio link is declared failed if both these thresholds are exceeded.

Previously proposed solutions can be divided into two main domains. The first domain encompasses the physical layer or topology based solutions [11, 13, 14], while the other domain consists of software based solutions that use Artificial Intelligence or Machine Learning, or Deep Learning algorithms [15–17]. Our research focuses on the software-based solutions for the radio link failure prediction, which leverages deep learning technologies.

An ensemble learning model for high prediction accuracy was proposed by Kotagiri *et al.* [18], which uses a combination of Random Forest, Light Gradient Boosting Machine and Extreme Gradient Boosting Machine. They compute the average failure probability of the individual model and classify the link failure by selecting an optimal threshold. However, the solution does not maintain the sequential property of the data which is an important aspect for the model to recognize the feature patterns in the new sequence of data. The existing approaches are limited to the supervised model training, where the scalability of the solution is computationally complicated. The

traditional feature extraction method may affect the overall result in larger network settings.

The goal of this research is to use a time-series deep-learning solution to utilize improved feature mapping on the pre-processed data and, ultimately, perform the link failure prediction. The proposed model learns the high-level features from the sequential data in an incremental manner, which reduces the complex data pre-processing and allows the model to perform prediction over a large amount of data. The experiment shows improved performance over the traditional pre-processing and supervised models and provide an improved scope for scalability in large network settings.

## 1.1 Contribution

In this work, we first analyzed both the radio and weather data sets to understand the weather features and the KPI features. This was then used to identify feature correlations and pattern matching, which were imperative for testing our initial hypothesis and assumptions. Following this, we investigated the existing state-of-the-art solution from NEC corporation [18] on the same data and identified its limitations. Specifically, in data pre-processing steps, the authors did not remove the outliers from long distance weather stations. Additionally, there was no cross-validation of time series data for the segmentation for training and validation. The training and testing was performed on the standard *scikit-learn* train test split with only the 2019 data.

We calculated the optimal distance between the radio towers and the weather stations within a specific region, following which we detected and removed the outlier connections with those stations that were far away from the radio towers. A recalibration of the data was accomplished using the synthetic approach, *smote* [19], to strengthen the learning performance of the proposed model. The sequential property of the data has been taken into account and applied to a customized time series cross validation to perform a sequence based training of the proposed model. We developed an Auto-encoder with the LSTM units, where a series of gates was used to control the sequence of the data. This was then used as an input for training the time-specific prediction. The final step consisted of testing our model with data collected over the new time period and evaluating the performance of our solution, which did not exist in the state-of-the-art solution [18].

The performance of our proposed solution is better than the existing one in terms of the accuracy, precision, recall, and F1 score. The major contribution in the development of our solution was taking the sequential nature of the data into account for the implementation of the model. This allows a well-trained model to reconstruct any sequence of the data while calculating a higher reconstruction error than normal data. Based on the evaluation of our proposed solution, we have achieved score of **88%** accurate classification (F1 Score ), which is **10%** higher than the current state-of-the-art solution [18], along with **14%** higher precision and **3%** higher recall indicating a significant improvement of our proposed deep-learning based solution over the existing ensemble-learning based one. A more detailed overview of our contributions have been provided below:

- Built correlations based on the distance between the radio tower site and the weather station data. A heuristic to find the correlation between these two based on the distance matrix was also developed. Outlier weather stations were subsequently detected and removed.
- Applied the principal component analysis of the data, where the dimensions were reduced so as to contain the maximum information in the data..
- Proposed a **time-series based deep learning model** which provides the hyper-parameters to tune the training for different settings and properties of the data.
- Evaluated the performance of our proposed solution with the standard evaluation metrics for prediction and compared the result with the existing state-of-the-art solution from NEC corporation [18]. We also discussed the limitations of the existing solutions and proposed improvements to them.
- Tested the solution with a new time span of data, performing **10%** better in terms of **prediction accuracy**.
- Released the code for the above solution in a public repository [20].

## 1.2 Thesis Outline

The rest of this thesis is organized as follows: Chapter 2 presents two sections: Section 2.1 introduces the necessary background for understanding the thesis, including the overview of data pre-processing steps, ensemble models and sequence-based deep learning models. Section 2.2 describes, in detail, the features from the available data. Chapter 3 discusses research work relevant to this thesis, including their findings and the limitations. Chapter 4 presents the design methodology and evaluation of the overall thesis. This is divided into two parts. In the first three sections, we explain the data pre-processing steps of both existing state-of-the-art solution and our proposed methods as well as the learning model training and implementation details. After that, Section 4.4 represents the explanation of the evaluation metrics, the results of the training and the testing of both existing solution and our proposed solution alongside the findings. Chapter 5 discusses the conclusion and future research directions of our work.

## Chapter 2

### Background and Data Description

#### 2.1 Background

In this section, we briefly review the machine learning algorithms required to build the proposed solution. This includes the pre-processing methods such as encoding and feature selection as well as learning models like ensemble learning, the Long Short-Term Memory (LSTM) networks and the Autoencoder networks.

##### 2.1.1 Data Pre-processing

When working with a sequential classification problem using deep learning methods like LSTM neural networks, categorical data cannot be input directly. It must be converted to numerical data first, called *encoding*, which can be achieved in a number of ways. We have applied the binary encoding and the one-hot encoding method separately in our research for the categorical encoding, evaluating and observing the performance of each in our proposed model. This experimentation helped us to choose the best encoding method for our data. The following is a brief description of the working procedure of the mentioned two encoding approach.

**Binary Encoding Method.** Binary encoding is a straightforward means of converting data into binary format where categorical features are first numbered and then uniquely assigned as columns in the output.

We first convert each category into an integer numerical order beginning with 1. Then, the binary code is generated by converting each integer to binary. Thereafter, the digits of the binary number are divided into separate columns according to their features. Binary encoding produces  $\log_2 n$  features if there are  $n$  unique categories [21].

The Fig. 2.1 shows an example of binary encoding operation. The first table contains the weather related categorical features. First, ordinal integer numbers are assigned for each of the categorical features. Then, the numbers are converted to the

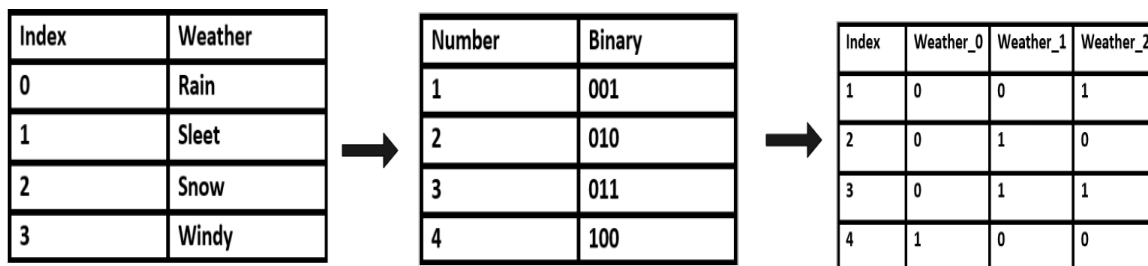


Figure 2.1: Example of Binary encoding operation.

binary representations. Finally, the binary values are split into the different weather columns. Usually, the binary encoding operation works best for the categorical features with ordinal correlations among them. The categorical features with no such correlation as like shown in the example figure 2.1 creates a false sense of ranking in the dataset among the features. The next paragraph discusses how the one hot encoding was used to avoid such bias in the data.

**One-Hot Encoding Method.** One-hot encoding performs well for data that has little or no relationship within it. Categorical data without any ordinal properties are best encoded using this encoding method [22]. Machine learning algorithms consider the order of a set of numbers to be a significant trait. Higher numbers are perceived as better or more important than the lower numbers [23]. Although the binary encoding algorithm works well for most ordinal situations, data which doesn't have ranked categories causes problems in both with prediction and performance [24].

Using categorical values as input, we create new categorical columns and assign binary values to each one in the one hot encoding method. This creates binary vectors that represent the integer values. Binary variables are mapped to each category containing either 0 or 1. Here, 0 represents the absence and 1 represents the presence of that category. An example of one hot encoding method in Fig. 2.2 clarifies the heuristic of this encoding method where Table 1 contains a column with categorical weather data that has no ordinal relation among them.

The second table in the Fig. 2.2 includes dummy variables that correspond to a specific weather category. 1 represents the presence of each category, while 0 represents its absence.



The diagram illustrates the hot encoding process. On the left, a table with two columns, 'Index' and 'Weather', shows four rows: (0, Rain), (1, Sleet), (2, Snow), and (3, Windy). An arrow points to the right, where a larger table with five columns, 'Index', 'Weather\_Rain', 'Weather\_Sleet', 'Weather\_Snow', and 'Weather\_Windy', shows the resulting binary matrix. Each row in the second table corresponds to an index and has a '1' in the column corresponding to the weather type and '0' in all other columns.

Index	Weather
0	Rain
1	Sleet
2	Snow
3	Windy

Index	Weather_Rain	Weather_Sleet	Weather_Snow	Weather_Windy
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

Figure 2.2: Example of one hot encoding operation.

### 2.1.1.1 The Curse of High Dimensionality

Categorical data encoding can create data overfitting, since it adds substantially more dimensions to the data, thereby impacting how the learning model trains and performs for the test data. A few dimensionality reduction techniques have been employed in this study, which have been briefly discussed in the following sections.

**Feature Selection.** Predictive models are designed to be developed with a small number of input variables. Feature selection is the process of reducing the number of input variables or dimensions. A reduction in the number of input variables can reduce computational costs as well as enhance model performance. In feature selection methods, which use statistical methods as part of their evaluation process, input variables are evaluated in relation to target variables; then, those that are most closely related to target variables are chosen [25]. Data types play a critical role in determining which statistical measures are applied to input and output variables. The methods can be fast and effective depending upon the type of data. The objectives of feature selection techniques include:

1. Simplification of models to make them easier to interpret by researchers/users.
2. Shorter training times.
3. Avoiding the curse of dimensionality.
4. Enhanced generalization by reducing overfitting (formally, reduction of variance).

**Recursive Feature Elimination.** In this work, we used the *Recursive Feature*

*Elimination (RFE)* technique of dimensionality reduction [26]. RFE involves recursively evaluating smaller sets of features. In this method, a feature importance or coefficient of variation (*coeff*) attribute is calculated to determine the importance of each feature before the estimator is trained on the initial set of features. Thereafter, the least important features are pruned from the current set of features. The *coeff* represents the contribution of a feature to the prediction task in the dataset, where negatively scored features are responsible for negatively classified class predictions, and positively scored features are responsible for positively classified class predictions. The features with scores close to zero are eliminated until the performance of the estimator begins to deteriorate. This process is repeated recursively on the pruned set until the desired number of features is eventually selected.

**Principal Component Analysis.** Principal Component Analysis is an unsupervised statistical analysis method for component analysis that constructs relevant features through linear or non-linear combination of the original features. In order to construct the most relevant features, the correlated features are linearly transformed into smaller uncorrelated components [27]. The original data is projected into the reduced space by using the eigenvectors of the covariance matrix, which is also known as the *principal components* [28]. This operation reduces the overall dimension of the data. The equation 2.1 refers the computation of covariance matrix  $\sigma_{jk}$  for variable  $j$  and  $k$ .

$$\sigma_{jk} = \frac{1}{n-1} \times \int_{i=1}^n (x_{ij} - x'_j)(x_{ik} - x'_k). \quad (2.1)$$

Here,  $\sigma_{jk}$  represents the covariance matrix,  $x_{ij}$  represents the specific number of sample from original feature space  $x_j$ ,  $x'_j$  represents the mean over all of the samples,  $x_{ik}$  represents the specific number of sample from the original feature space  $x_k$  and  $x'_k$  represents the mean over all of the samples.

Despite being undefined by definition, dimensionality reduction tends to simplify data sets at the expense of some accuracy. The analysis of small datasets is simpler and machine learning algorithms can analyze them without having to bother with extraneous data. Essentially, PCA ensures that data set information is preserved while reducing the number of features within the dataset [28]. Along with RFE, the motivation of using the PCA with the high dimensional encoded data is to retrieve the maximum information after the dimension reduction operation. We have trained

our proposed model with both PCA and RFE methods applied on the data and subsequently observed the prediction performance of the model, which is represented and discussed in detail in Chapter 4.

A few separate tasks need to be performed in the PCA in order to achieve the reduction in dimension. The first step is to investigate the correlation between the features to determine the information, i.e., the change in mean from each feature to the other. This involves calculating the matrix of the covariance, which is a  $p \times p$  symmetric matrix ( $p$  is the number of dimensions) where the covariances are associated with all possible pairs of the initial variables.

The next step is to determine the principal components of the covariance matrix by computing their eigenvectors and eigenvalues. The eigenvectors of a variability matrix are the directions of the axes that contain the most variance. The eigenvalues are simply the coefficients attached to the eigenvectors, which are indicative of the variance in each Principal Component. Following that, principal components are computed by ranking eigenvectors according to their eigenvalues. As a final step, the feature vector is calculated by adding the highest significant values from the previous steps. The purpose of this step is to remove the features with low significant values, which is a passive operation when reducing the dimensions [29].

### 2.1.2 Dataset Balancing

An important aspect of data preprocessing is balancing minority and majority classes within the data.

Resampling data is one of the most common approaches to deal with the unbalanced datasets. This can be achieved in two different ways: 1) Undersampling and 2) Oversampling. Usually, oversampling techniques tend to be preferred over undersampling techniques [19].

Synthetic minority oversampling technique (SMOTE) is an oversampling technique where the synthetic samples are generated for the minority class, which aids in overcoming the over-fitting problem caused by random oversampling. By interpolating between positive instances that lie together, it generates new instances based on the feature space [19].

In the initial step, a total of  $N$  oversampling observations are determined. An

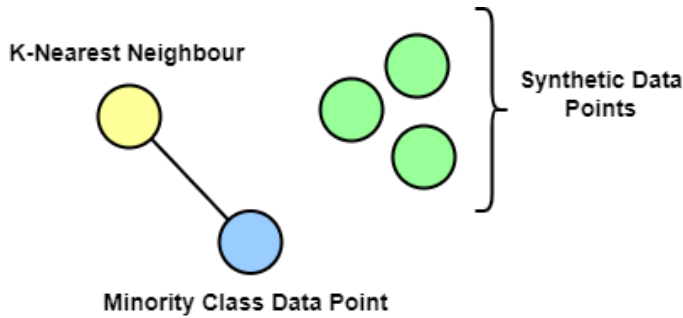


Figure 2.3: Synthetic Minority Oversampling.

iteration starts with the selection of the first random instance of the class. The next step is to retrieve the K-Nearest Neighbour ( KNN ) (5 by default) for the instance as shown in Fig. 2.3. A new synthetic instance is generated by interpolating N instances from these K instances. It utilizes a distance metric in order to calculate the difference between the feature vector (selected minority class sample) and its neighbors. A random value between  $[0, 1]$  is multiplied by this matrix that calculates the difference between the matrix and the feature vector [30]. As many synthetic examples of the minority class can be created using this procedure as needed.

### 2.1.3 Learning Algorithms

#### 2.1.3.1 Ensemble Learning Method.

Ensemble learning is a general meta approach to machine learning that seeks better predictive performance by combining the predictions from multiple models. It is a popular machine learning paradigm where multiple models (often called “weak learners”) are trained to solve the same problem and combined to get better results [31]. The main hypothesis is that when weak models are correctly combined we can obtain more accurate and/or robust models.

The existing state-of-the-art solution [18] used a combination of three models to prepare the Ensemble learning model. The three individual models are: Random Forest, Light Gradient Boosting Machine and Extreme Gradient Boosting Machine. Our approach reproduced the prediction performance in the same way, then analyzed the details further to discover limitations and other details of the solution. We will be discussing the theoretical details in this section and the implementation details in

Chapter 4.

**Random Forest.** A random forest consists of many individual decision trees that work together as an ensemble. A random forest is a method of predicting classes based on the number of votes each tree provides and the class with the most votes becomes the model prediction as shown in Fig. 2.4. One of the most fundamental concepts underlying random forest is the idea of the wisdom of crowds [32]. Considering the impact of correlated and uncorrelated models, the likelihood is that the results will be better when their number increases and they work as a group.

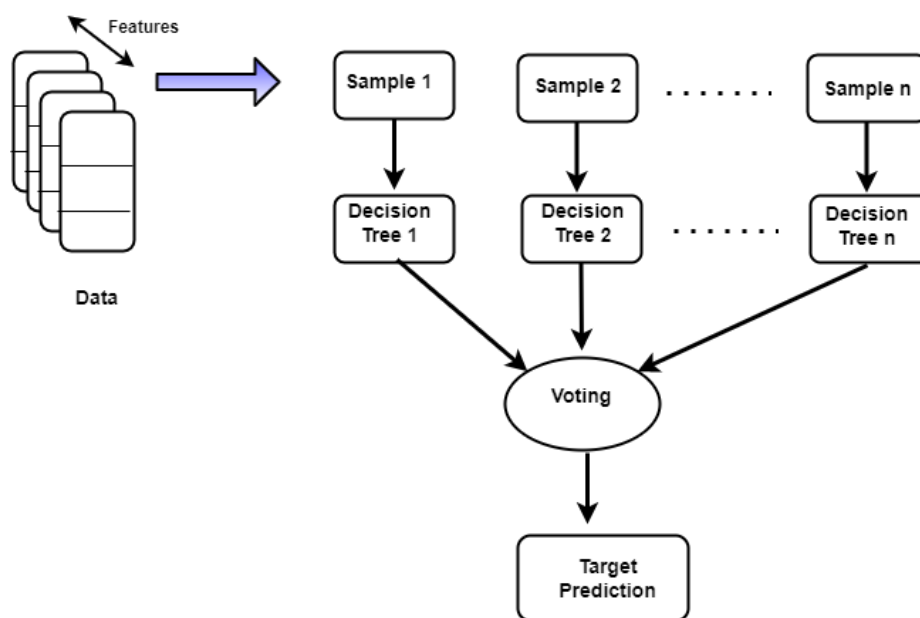


Figure 2.4: Random Forest Classifier

Initially a randomly split dataset is transformed into a decision tree ensemble (based on the divide-and-conquer strategy) [33]. On the basis of selection indicators like information gain or Gini index, a decision tree is generated. The trees are generated from a sample of random data. The best class is chosen as the final outcome for a classification problem when each tree votes.

**Light Gradient Boosting Machine (LGBM).** Light Gradient Boosting Machine (LGBM) is a gradient boosting framework that uses a tree-based learning algorithm. LGBM grows the tree horizontally while other algorithms grow trees vertically. In other words, Light GBM grows the tree leaf-wise horizontally while other algorithm grows level-wise [34]. The figure 2.5 below shows the horizontal growth of the tree.

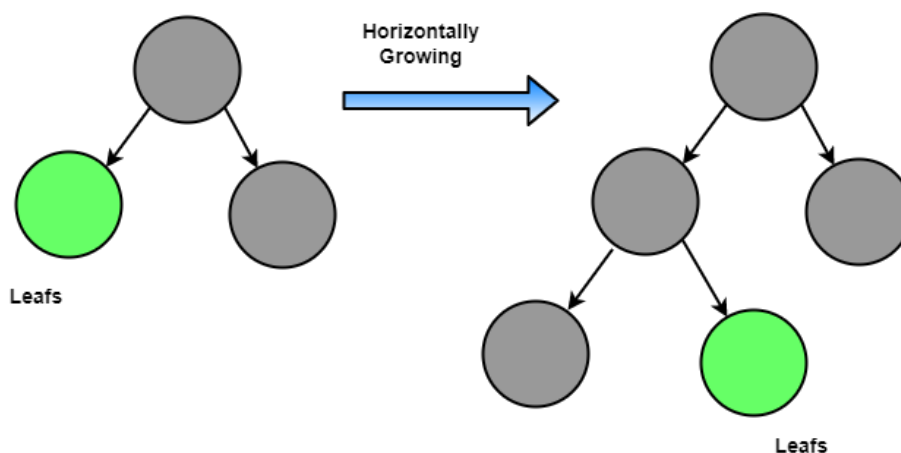


Figure 2.5: Light Gradient Boosting Machine leaf-wise growth.

It chooses the leaf with max delta loss to grow. Leaf-wise algorithms can reduce more loss than level-wise algorithms when growing the same leaf. As the data size grows, traditional data science algorithms face difficulty delivering results faster [35]. Data can be handled by Light GBM with less memory usage and bigger data sizes. The fact that Light GBM focuses on accuracy is another reason for its popularity. Data scientists are increasingly using LGBM to develop data science applications because it supports GPU learning.

**Extreme Gradient Boosting Machine.** Extreme Gradient Boosting Machine (XGBoost) is an implementation of gradient boosted decision trees designed for speed and performance, and is a competitive machine learning technique [36].

Boosting also works on the ensemble principle. It is a method that utilizes the combined strength of weak learners so that prediction accuracy can be improved. The model outcome at any instant,  $t$ , is weighed according to the model outcome at the previous instance,  $t - 1$ . Both correctly predicted outcomes and incorrectly predicted outcomes are assigned weights. A weak learner is slightly more accurate than random guessing. In a boosting algorithm, the mis-classification error created by the previous model creates a stronger model by using the evidence from the previous model to create a new, stronger model [37]. As mentioned previously, XGBoost is algorithm that combines a decision-tree-based ensemble machine learning framework with gradient boosting to achieve an ensemble prediction.

### 2.1.3.2 Deep Learning based Models

**Fully Connected Feed-Forward Convolution Neural Network.** The fully connected convolution neural networks are deep learning algorithms that use an image or a data series as input, capture the spatial patterns with programmable filters, and then assign importance using trainable weights. In contrast with other classification algorithms, the processing required for convolutional neural networks is much lower. Convolutional Neural Networks have the capability of learning filters in contrast to many other methods that require additional feature engineering to design them. A Convolutional Neural Network is composed of the following three different layers.

1. Convolutional Layer.
2. Pooling Layer.
3. Fully-Connected Layer.

Usually several Convolutional Layers and Pooling Layers are alternated before the Fully - Connected Layer.

**Convolution Layer.** The convolution operation is the fundamental building block of this type of network. It performs a convolution of feature maps with a filter matrix to obtain, as output, a different series of feature maps, with the goal to extract high-level features. The result of the convolution between one input feature map and one filter is the ordered feature map obtained applying the filter across the width and height of the input feature map [38].

**Stride.** Stride controls how the filters slides to one input feature map. In particular, the value of stride indicates how many units must be shifted at a time.

**Padding.** Padding indicates how many extra columns and rows to add outside an input feature map, before applying a convolution filter. All the cells of the new columns and rows have a dummy value, usually 0.

**Pooling Layer.** The purpose of the pooling operation is to achieve a dimension reduction of the aforementioned feature maps, preserving as much information as possible. It is useful for extracting dominant features which are rotational and positional invariant. Its input is a series of feature maps and the output is a different series of feature maps with lower dimensions.

**Fully - Connected Layers.** A fully-connected Layer is designed to learn non-linear combinations of the high-level features represented by the output of the convolution layer and the pooling Layer. Fully connected layers are usually implemented using Multi Layer Perceptrons (MLPR) [39]. A series of feature maps are generated from the original input series after convolution and pooling operations. A column vector represents all the feature maps flattened into one, representing the original multivariate input. As the input series is flattened, the single column is connected to the MLPR, whose neurons are divided equally between classes of the series. Back-propagation is applied to the data during training. Over a series of epochs, the model is able to distinguish the input series thanks to their dominant high-level features and to classify them.

**Long Short Term Memory Network (LSTM).** An LSTM is a type of Recurrent Neural Network (RNN) which allows the network to retain long-term dependencies between data at a given time from previous timesteps [40]. The LSTM network consists of a series of repeating modules, each containing three control gates: the forget gate, the input gate, and the output gate. The cell state of the network transitions horizontally from one end to another. The network control gates regulate the information for passing to the cell state during the transition. A sigmoid neural network layer and a pointwise multiplication operation are used in the control gates. The sigmoid layer outputs numbers in 0 and 1, which represents which information should be let through and which one should be forgotten. In the LSTM model, the vectors are read from an input vector  $x = [x_1, x_2, \dots, x_t, \dots]$ , where  $x_t \times Rm$  represents a m-dimensional vector of readings for m variables at the time-instance  $t$  [41].

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2.2)$$

where  $h_{t-1}$  is the output in state  $t - 1$ ,  $W_f$  and  $b_f$  are the weight matrices and the bias of the forget gate. Then,  $x_t$  is processed before storing in cell state [42]. The value is determined in the input gate along with a vector of candidate values  $C_t$ . The activation function simultaneously updates it in the new cell state  $C'_t$ ,

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2.3)$$



$$C'_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (2.4)$$

The old state is multiplied by the factor  $f_t$  from Eq. 2.2 and the  $i_t \times C'_t$  is added which is represented in Eq. 2.5. This is the new candidate state value, scaled by the amount decided upon to update each state value.

$$C_t = (f_t \times C_{t-1}) + (i_t \times C'_t) \quad (2.5)$$

where  $(W_i, b_i)$  and  $(W_c, b_c)$  from Eq. 2.3 and Eq. 2.4 are the weight matrices and the biases of input gate and memory cell state, respectively. Finally, the output gate, which is defined by,

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (2.6)$$

where  $W_o$  and  $b_o$  are the weight matrix and the bias of output gate. The output is the filtered version generated from the cell state  $C_t$ . Then, The cell state is converted to output values from  $-1$  to  $1$  and multiplied by the sigmoid gate output from Eq. 2.6.

$$h_t = o_t \times \tanh(C_t) \quad (2.7)$$

The  $h_t$  output from each LSTM unit at the end of the transition is generated by its cell state.

**LSTM - Autoencoder.** Autoencoders are neural networks that learn the best encoding and decoding schemes from data. They consist of an input layer, an output layer, an encoder neural network, a decoder neural network, and a latent space. In order to feed the data to the network, decoders decompress the encoded representation into the output layer, while encoders compress them in the latent space. After the encoded-decoded output has been compared with the initial data, errors are propagated back through the architecture in order to update the network weights [41]. In particular, given the input  $x \in \mathbb{R}^m$ , the encoder compresses  $x$  to obtain an encoded representation  $z = e(x) \in \mathbb{R}^n$ . The decoder reconstructs this representation to give the output  $\hat{x} = d(z) \in \mathbb{R}^m$ . The autoencoder is trained by minimizing the reconstruction error.

$$L = |x - x'| \quad (2.8)$$

where  $x'$  represents the reconstructed output and  $x$  represents the ground truth. An autoencoder does not simply copy input to output. The constraint, i.e.,  $n \times m$ , forces the autoencoder to learn the most salient features of the training data. Alternatively, one of the most important properties of an autoencoder is the ability to reduce data dimensions while retaining the structure of the data [17].

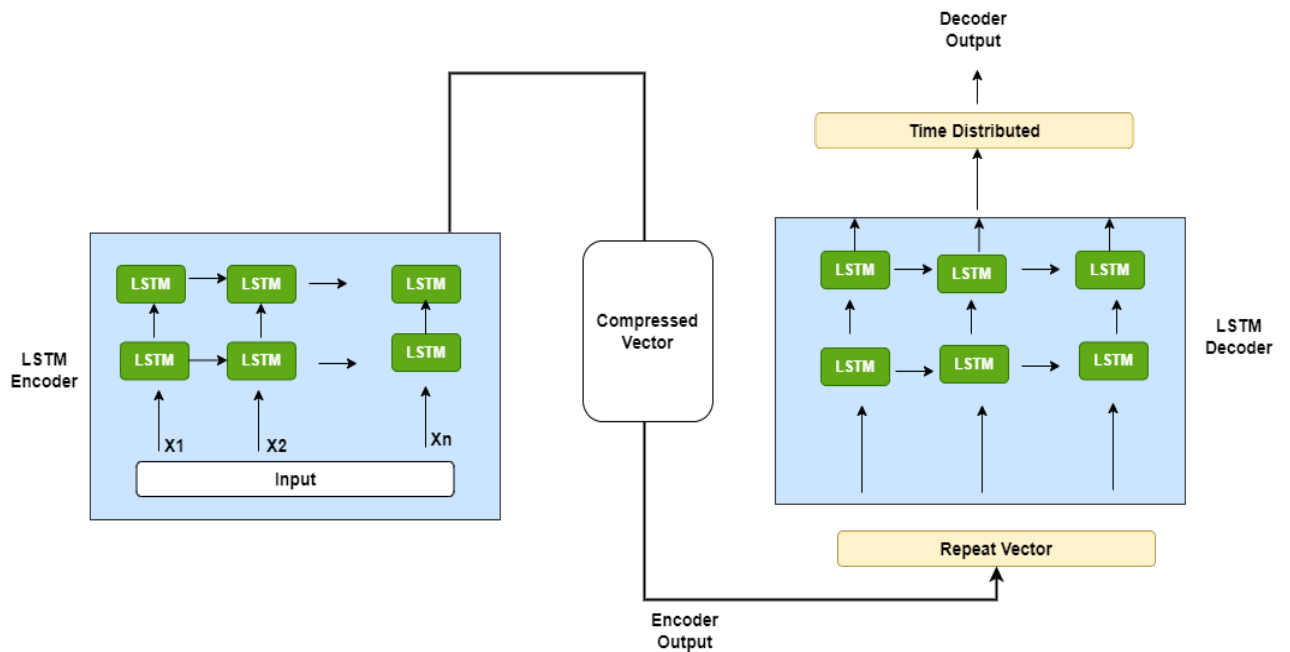


Figure 2.6: LSTM Autoencoder Architecture

Encoder-decoder LSTMs are implemented as LSTM Autoencoders using the encoder-decoder architecture. LSTM encoders are capable of decoding, encoding, and recreating sequences from a given time-series data. Evaluation of the model is based on its capacity to recreate the input sequence.

The Fig. 2.6 represents the architecture of an LSTM-Autoencoder network. The Encoder part (LSTM Encoder) of the network transforms a given input sequence into a fixed-length vector, or a compressed sequence. The vectors with a fixed length are called context vectors. Decoders (LSTM Decoders) start from the context vector as input, while predicting output sequences from the final encoder state as an initial decoder state. A repeat vector layer is used to repeat a context vector generated by

the encoder so that the decoder can use it. Each forecasting step is repeated for  $n$  steps ( $n$  is the number of future steps to be taken). The output from the decoder is mixed according to the time step. Each time step is applied to a full dense layer with a separate output for each time step. The *Time Distributed* layer is a wrapper that enables the application of layers to every window of a time interval.

There are several types of autoencoders that have been proposed in the literature, such as vanilla autoencoders, convolutional autoencoders, regularized autoencoders, and LSTM - autoencoders. Among them, LSTM - autoencoders refer to autoencoders where both the encoder and the decoder are constructed with LSTM units. LSTMs have the ability to detect temporal patterns in data with a large number of observations, which makes them a good choice for forecasting time series or identifying anomalies/ failures [40]. The operation of using an Autoencoder with LSTM involves the reconstruction of the data while maintaining the temporal dependency of the features. In this research, an autoencoder model with multiple LSTM layers is trained with only the normal sequences which can be used to detect link failures in the time series test data. During training, the encoder-decoder only encountered normal instances/ non-failed links and learned to reconstruct them. A failed link which is not learnt by the model during training, may not be easy to re-construct, resulting in higher reconstruction errors. In this thesis, we explored the capabilities of a well-trained model that can show the difference between anomalous data, such as failed links, since such anomalous events are rare in a real world scenario.

## 2.2 Data Explanation

The dataset used throughout this thesis was collected from the regional weather stations and the radio tower sensors by the organization network experts [12]. The dataset can be found here [43]. The first set of data contains samples from January 2019 to December 2019. Additionally, we collected data from January 2020 to June 2020. We trained our model with the 2019 data only, while the data from the first six months of 2020 was incorporated in the testing phase. We utilized the information from both radio towers and the weather stations from the same region and prepared our solution accordingly to predict the network link failure.

There are six subsets in the original data set that represents three parts for different network and weather operations. A brief description of the parts and subsets are provided below:

1. Weather Data :

- **Met - Forecast Data.** The weather forecast data contains forecast reports for the next five days. The features are Date-time, station\_id, min-max temp, weather type, wind speed, humidity, report time, wind direction.
- **Met - Station.** This data-set contains the spatial features of the weather stations. The features are weather stations, ground height (distance of the weather station from the sea level), and the clutter class which indicates the type of environment of the weather station location.
- **Met - Real.** This data contains the real time weather report with of each hour of the day. The features are station-no, date time, measure-date, measure-hour, temp (max,min,current), wind-dir (max,min), wind-speed (max,min), humidity,precipitation, precipitation-coeff, pressure, pressure-sea-level.

A summary of the weather data is given in Table 2.1.

Subset	Total Features	Total Samples
Met-Forecast	9	39370
Met-Real	32	629217
Met-Stations	3	117

Table 2.1: Summary of the weather data.

2. Radio Link Data :

- **RL - Sites.** This data-set contains spatial features of the Radio link (RL) sites. The features are site-no, ground height (distance of the radio link site from the sea level) and the clutter class which indicates the type of environment of the site location.

- **RL - KPI.** Each sample in the Key Performance Indicator (KPI) data represents the features that are used to measure the performance of the radio links. The features are type, date-time, tip, mlid, mw-connection id, site no, polarization, card type, adaptive modulation, frequency band, link length, severely error seconds, error seconds, unavailable seconds, available time, bbe (background bit error), power level, scalability score, capacity, modulation, radio link failure. A summary of the radio link data is given in Table 2.2.

Subset	Total Features	Total Samples
RL-Sites	3	1674
RL-KPI	21	979063

Table 2.2: Summary of radio tower data.

3. Distance Matrix : The exact location of the radio towers and the weather station base are kept secret due to the security reasons. But the geo-relation between the radio tower and the weather stations is represented in the form of the distance between them. The distance matrix represents the distance between each of the weather station and the radio link tower site. Each sample is in terms of distance in kilometers for each weather station and site id. The distance matrix is the only connected relation between the weather stations and the radio link tower site.

A brief summary of the contents of each table in the dataset is summarized and can be seen in Table 2.3.

<b>Table</b>	<b>Contents</b>
Weather Station Table	Spatial Properties and environment type of station sites
Real Weather Table	Data on the weather condition on real time, e.g. wind speed and temperature
Weather Forecast Table	Forecast data for the next 5 days after measurement
Radio Link Site Table	Spatial Properties and environment type of RL sites
Radio Link KPIs Table	RL performance indicators, e.g. background bit error count (BBE) and unavailable seconds
Distance Table	Distance among the weather stations and RL sites

Table 2.3: Dataset table description

## Chapter 3

### Literature Review

#### 3.1 Related Works

Fifth generation (5G) wireless networks promise to provide faster and more expansive data connectivity, exceeding thresholds from previous fourth generation (4G) technology. The need for high bandwidth data access to end users is always increasing, thanks to the growing number of users and devices who rely on mobile computing, as well as the constantly increasing sophisticated applications. Weather interruptions are particularly damaging to high frequency communications. In particular, Millimeter wave transmissions suffer from significant depletion owing to precipitation. As a result, during rainstorms, connection availability and dependability suffer considerably [14].

Weather forecasts can be used to predict whether connection disruptions will occur ahead of time, thereby saving significant recovery costs and potential downtime. Many existing solutions are proposed and implemented that take into account the impacts of the weather on network links. These can be divided into two types of literature found in the context of radio link failure prediction. Some of these works focused on the network topology and the solutions implemented at the physical layer. Different routing protocols are employed to detect the unavailability of packets in topology-based works. Another approach is the use of machine learning algorithms in order to detect the failed link [11].

The proposed solutions utilize machine learning algorithms to classify the normal frequency links from the affected links with the collected mmWave radio frequency data and meteorological weather forecast data. As opposed to the topology-based solutions, which are complex and expensive to migrate or scale for a new network situation, machine learning-related solutions allow for dynamic scaling regardless of data size in a new region or time-frame. In this section, we present and compare the existing solutions. We thoroughly reviewed the literature and previously proposed

solutions as well as pointed out the gaps and limitations in those solutions.

### 3.1.1 Topology Based

Topology-based reliability service is common both in wired [44–48] and wireless [49–55] networks. However, these works did not consider weather impact. Nauman Javed *et al.* [56] proposed a predictive routing model for wireless mesh networks which operate at millimeter-wave bands with directional links, that uses in-network parameter prediction to make the network adaptive, as opposed to using meteorological weather information from external sources, such as weather radars. The approach is validated through simulations based on real-world weather events, observed through a network of weather radars, and compared with approaches that do not make use of predictions but may use the link quality as a parameter in routing decision making. The results showed 8% better performance in terms of entire link failure detection in the mm-Wave predictive mesh network model.

A new transmission technique for wireless network was proposed by Jacek Rak *et al.* [57], which mitigates the link quality degradation or complete failure particularly in the heavy rain storm conditions. The proposed methodology in his research is the first one to use information on forecast attenuation of links based on radar measurements and days ahead weather forecast to perform in advance the periodic updates of a network topology that is used for further accurate prediction of the link failure.

Another study proposed a reinforcement learning approach to prevent the mmWave based connection disruption. The authors discussed about the radio link signal failure (RLF) in the closed environment context such as the elevator. The metallic wall of the elevator interrupts the radio connection between the radio tower and the mobile device [16].

A Reinforcement learning model approach called Intelligent Elevator Detection and Network Adaptation is proposed where the user will upload the failure report, link stats, to a cloud server which can be then utilized by other users to acquire prior knowledge about the RLF [16]. The algorithm switches to the stronger signal (3g/4g/5g/wifi) seamlessly without losing connectivity. The learning model uses the accelerometer and gravity sensor to detect user entrance and exit into the elevator and detects the elevator by the nearby WiFi MAC addresses. Then, it classifies the



signal strength of the available cellular network in the elevator. The proposed solution in this study leans into the recovery aspect from the UE and does not provide much scaling convenience.

An RF prediction model has been formulated analytically from the rain echo maps statistics in order to calculate signal attenuation based on the radar-based rain echo map [58]. Through the predictive routing protocol, a method of predicting the background bit error (BBE) rate and the likelihood of detecting backhaul failure has been used to predict the future. While heavy rain weather features are one of the most common weather features, using the only one in the dataset to feed into the model also results in incorrect predictions.

Another type of research has been conducted that uses the various weather features from the weather station that are situated near network sites [59]. The prediction model in the methodology is built by taking several weather features into account. The study also proposed the model that raises an alarm before predicting and registering a link failure if the radio link signal quality is below the threshold. The methodology has a limitation of working with the distance between weather stations and radio link sites, as long-distance weather station data suffers from time synchronization problems with network link site data.

### 3.1.2 Machine Learning Based

The available solutions that employed machine learning techniques to classify network connections, are divided into two categories. Some of the existing solutions utilized the decision tree based supervised learning model where the correlations of the radio network frequency data features and the weather forecast data features were calculated and classified the failed links to normal links by forming conditional decisions. The other type of the existing solutions used the feature compression and latent representation of the most important features properties of the deep learning algorithms.

The existing state-of-the-art solution from NEC Corporation [18] worked with the same provided data such as ours, using an ensemble model based on the decision tree algorithm. The authors designed the approach to have three individual decision tree based models, which are trained and then perform classification for each

link. The final prediction result was calculated from the average prediction from each model prediction score. The detail of the workflow and learning model approach is described and compared with our proposed solution in later sections.

Another decision tree based approach was proposed in [60] where a low depth model, *DecisionTreeClassifier*, was able to capture the feature correlations between the radio frequency KPI data and the weather forecast data. The solution relied heavily on the random iterative feature elimination to prevent the model overfitting from the high dimensional combined data, whereas we used systematic neural network based dimension reduction in our proposed solution. *DecisionTreeClassifier* was unsatisfactory in classifying the failed link compared to the aforementioned ensemble model due to the use of random feature elimination and data imbalance. There exist other works on failure cause analysis and detection of failures in radio networks [61, 62].

The authors of [61] used causality graphs to apply feature correlations and link failure detection to classify sequential network data. Another work of supervised as well as semi-supervised ML based models was proposed by Francesco *et al.* [63] where high classification accuracy was achieved from a supervised model. The solution performed worse on the real network deployment with little or no data labelling. Later, the author performed the experiment with deep learning instead and achieved higher classification accuracy than the supervised solution. However, unlike our approach, the solution did not process the sequential training of the model where models are trained and validated by time sequence.

A supervised learning technique for predicting failures was suggested by Vanerio *et al.* [64], such that any potential microwave connection problems may be repaired immediately. The decision trees were utilised to detect anomalies in synthetically created data derived from genuine microwave network statistics. The proposed work used the synthetic data of various disruption events such as weather, geo-location and so on to train the supervised model and classify the anomalous links. Nevertheless, unlike our proposed solution, the authors did not enlighten that the high dimensions of the data due to encoding may degrade the model training performance in a real network setting.

Some existing studies utilized deep learning methods for anomaly detection in

radio link features, which is nearly similar to detecting/forecasting the failed links. M. Furdek *et al.* [65] proposed an unsupervised learning technique which predicts malicious attacks at the optical layer as anomalies by having the knowledge of the mmWave radio link features.

A density-based unsupervised clustering technique for grouping and labelling points into potential classes was proposed by X. Chen *et al.* [66], detecting outliers as network failures, and then training a neural network to identify failures via binary classification using the already labelled data. These proposed solutions did not put emphasis on maintaining the data sequence during training, unlike our proposed solution in terms of time series data.

Decomposition and modelling approaches are two types of strategies for detecting abnormal records in time-series data. Only suited for univariate time series, decomposition techniques split a time series into level, trend, seasonality, and noise components, monitoring the noise components to capture anomalous records [40]. Modeling approaches depict a time series as a linear/non-linear function that relates each current value with its previous values, forecasts the value of a record at a given time, and reports records whose prediction error goes outside of a threshold as anomalies. Moving Average (MA) is a stochastic modelling method. Autoregressive Integrated Moving Average (ARIMA) [67], and Holt-Winters (HW) [68] used statistical measures to calculate the correlation between the data records. We utilized the time series aspect of the deep learning model due to the sequential property of the data. One of the main aspects of our work is to maintain the sequence according to the date of the weather forecast and radio link performance indicators.

A deep learning model was proposed by H.D. Nguyen *et al.* [15], for forecasting the trend in the supply-chain management system while maintaining the sequential properties of the data. Similarly, Mahmoud *et al.* [41] proposed an anomaly detection technique for malicious attack detection. By training the models with just examples of normal classes, the Long Short Term Memory (LSTM) autoencoder and One-class Support Vector Machine (OC-SVM) were used to identify anomalies-based attacks in an imbalanced dataset. The LSTM-autoencoder was trained to learn the normal traffic pattern and the compressed representation of the input data (i.e. latent features), after which it was fed to an OC-SVM method for classification. We took

the inspiration from the aforementioned anomaly detection solutions and built our own solution to predict failures in the radio link in the context of weather impact on mmWave radio link. One of the main advantages of the aforementioned techniques is the ability to handle high-dimensional data sets with high performance, which is nearly impossible or requires extra efforts and time for supervised models. Moreover, the amount of weather and radio link data is increased along with the time where supervised models might fail to classify due to the high dimensionality in the real world setup. However, the existing techniques for anomalous sequence detection split the data into multiple subsequences, typically based on a fixed sized windows, where we have explored the dynamic window size for the past failure events as well as systematic feature reduction to improve model performance. An LSTM-Autoencoder is a time series autoencoder that captures long-term temporal correlations between data recordings in the form of complicated equations that are incomprehensible to humans.

Approach	Scaling	New Data Testing	Sequence Maintain	Recovery
P-WARP	✓	Not Tested	No	Yes
Ensemble Model	×	Not Tested	No	No
DecisionTree	×	Poor	No	No
LSTM-AE-OCSVM	✓	Not Tested	Yes	No
ConvAE	×	Not Tested	No	No
Proposed LSTM-AE	✓	Good	Yes	No

Table 3.1: A comparison of existing solutions for Link Failure / Anomaly Detection.

In summary, we can observe that most of the machine learning based solutions do not focus on the data sequences during the model training, in terms of radio link failure prediction. Decision Tree based solutions offer less computation at the cost of low prediction accuracy. The solutions that utilize the reconstruction property of an Autoencoder with LSTM layers maintain the sequence of the training data but there are no experiments for the new sequence of data coming from radio tower sensors in real world. This greatly limits the ability of these models to predict in a new setting. On the other hand, the Convolutional Autoencoder models do not maintain the historical data during model training. In our proposed model, we addressed the above limitations. Furthermore, we find that our model is very successful on the new data. Even though we do not provide the recovery of the radio link, which is

the future direction of this research, our model performance is better than previously discussed machine learning based solutions.

## Chapter 4

### Research Methodology and Evaluation

This section introduces the approaches for both the current state-of-the-art solution from NEC corp [18] and our proposed solution for predicting the radio link failures. In this thesis, we address the issue of mmWave radio link failure in radio base-station towers due the weather effect. Radio towers operating in mmWave radio signals produce Key Performance Indicators (KPIs) of sensors for a certain time span, which are used to correlate weather forecast data. Geographical characteristics of radio towers and weather stations are revealed by the spatial information from the area.

The objective of this study is to use the temporal dependency of the radio links and weather features and predict a link failure for a particular day. The deep learning strategies were utilized for the improvement of feature mapping, learning feature correlations and accurately predict link failures. The workflow of the proposed methodology is given below in the Fig. 4.1.

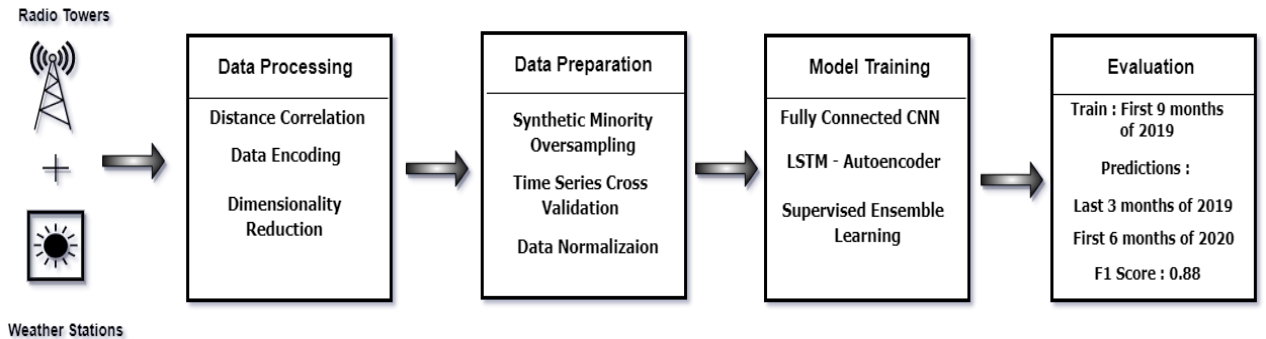


Figure 4.1: The workflow of the proposed approach.

## 4.1 Data Pre-Processing

### 4.1.1 Distance correlation.

In order to determine whether the failure of the radio links was related to the weather condition, we had to use the weather forecast report associated with the radio links. The distance from a radio base tower to the weather station is the only primary parameter found in the data. The dataset contained a distance matrix between radio towers and weather stations, along with the radio tower id and the weather station id, which provided the necessary correlation parameter for the features between the radio tower and weather station data. However, the region from which the data originates extends to a radius of more than 100 km. Radio towers located in far away areas cannot all be correlated with weather stations. The failure of a far radio tower link in the mmWave environment would not be impacted by the weather forecast because of the short-range properties of mmWave [9]. So, using the distance information between radio towers and weather stations, we devised a step-by-step heuristic to extract correlations between them. We used the following heuristic to find the best distance radius to corroborate the radio towers with the weather stations.

1. First, the closest radio link tower from each of the weather stations is found. This ensures that at least one radio link tower is associated with a weather station.
2. Next, among the above set of all radio link tower distances, we found the maximum distance between that radio tower and its corresponding nearest weather station. For example, lets assume, RL1, RL2 and RL3 ( Radio link tower) are connected with their nearest weather stations WS1, WS2 and WS3 (Weather station), respectively. RL3, and its distance  $d_3$ , was found to be the largest of the three connections.  $d_3$  is thus set as the radius for other weather stations to be connected to radio towers, and denoted as the *optimal distance*  $\mathbf{d}$ . The computed *optimal distance*  $\mathbf{d}$  was 4.2 km.
3. Finally, all of the radio link towers were taken into account which are within the *optimal distance*  $\mathbf{d}$  for each weather station and formed the correlation between RL and WS as shown in Fig. 4.2. As we can see in the figure, if weather station

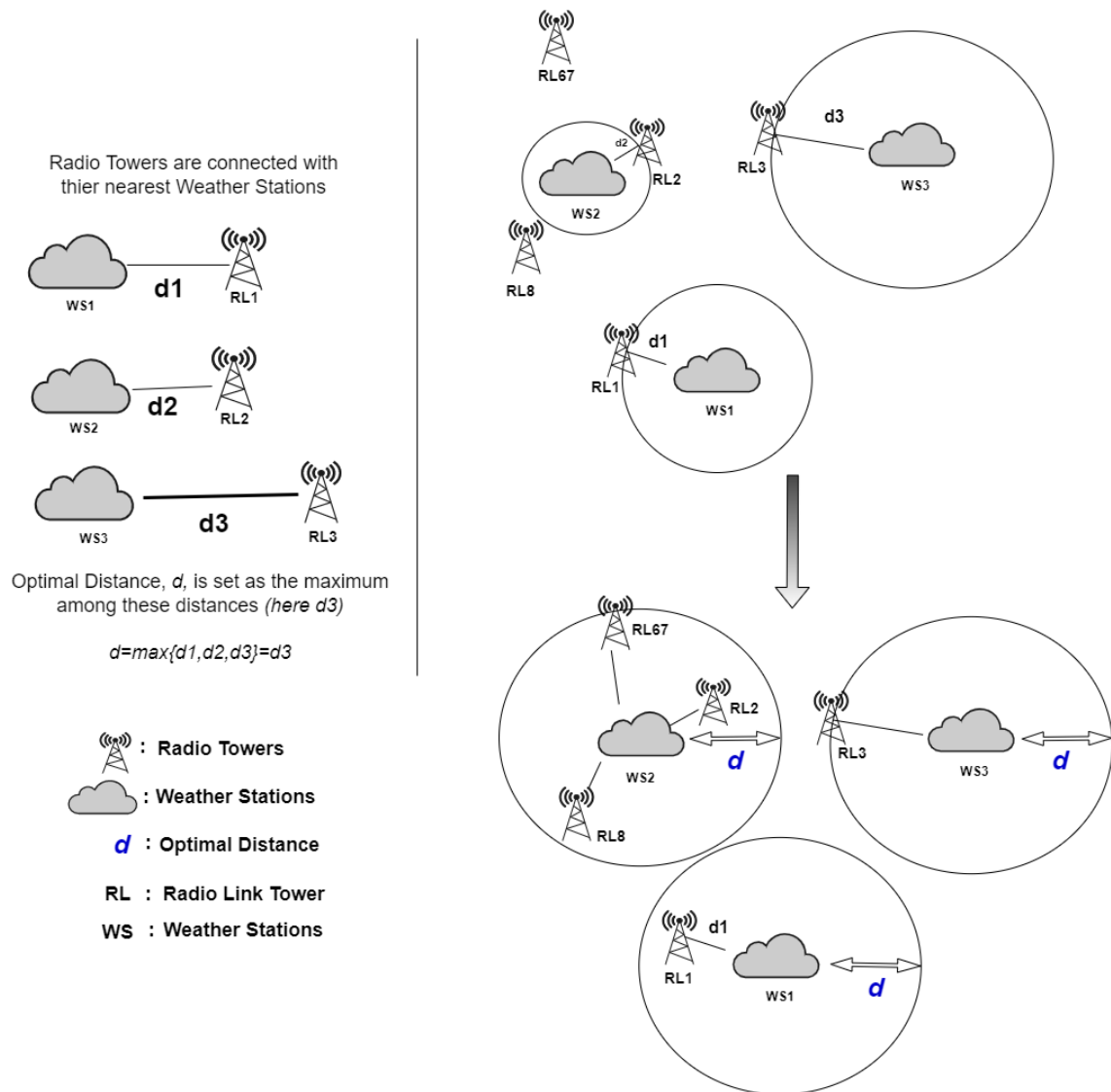


Figure 4.2: Correlation established based on the optimal distance  $d$ .

WS2 has radio link towers RL2, RL8 and RL67 within the *optimal distance*  $d$ , then the correlation is established between this weather station and all three radio link towers

However, after calculating the shortest distance between RL and WS, it was revealed that, some of the WS have a very high distance from the RL sites. For example, compared to most radio towers, which are only 2 or 3 km away from their closest weather stations, a small number of radio towers are located approximately 100 km away from their closest weather stations. The data from these weather stations do not have the same impact on the radio link key performance of the corresponding radio



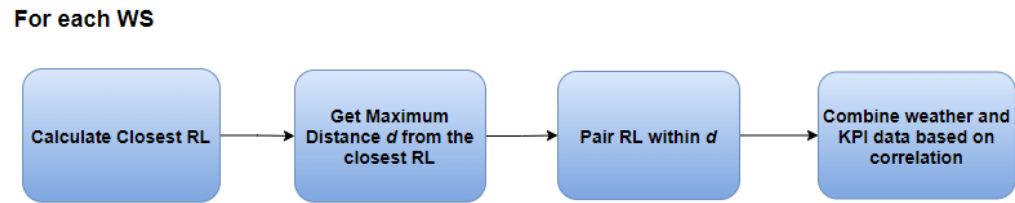


Figure 4.3: Correlation heuristic flowchart.

tower in the mmWave network environment, and thus designated as outliers. These WS and their corresponding closest radio link tower were removed from the formed correlation. The outlier radio towers are shown in Fig. 4.4. The links associated with these towers are also not considered [10].

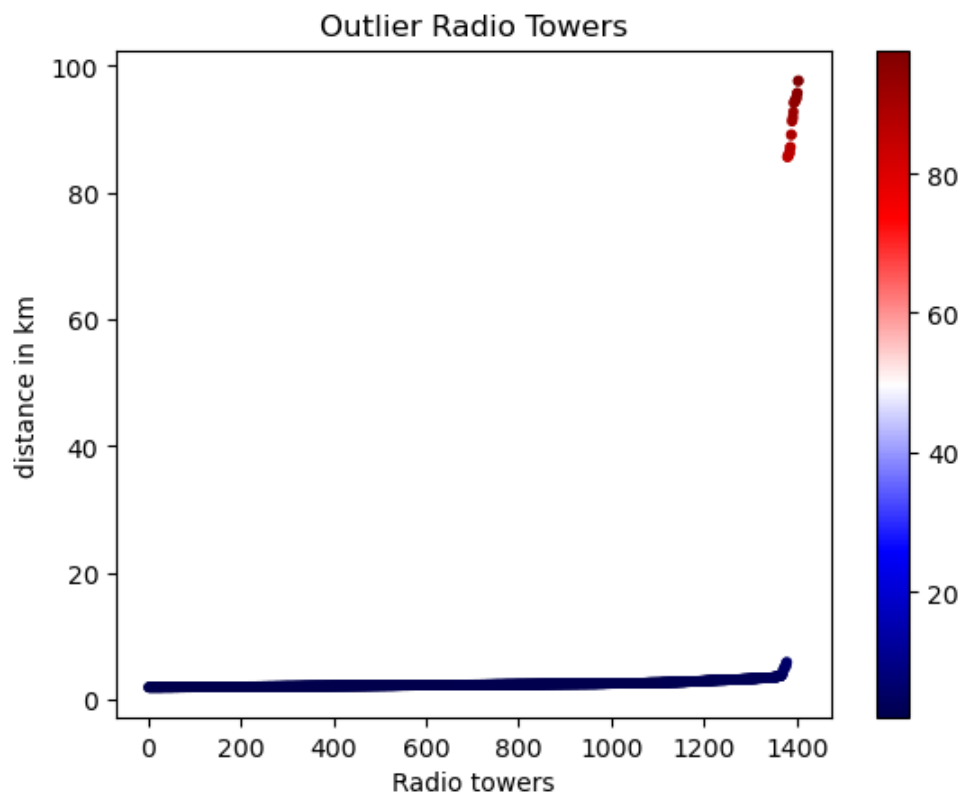


Figure 4.4: Outlier weather stations.

### 4.1.2 Data Combination

The radio link KPI data and the weather forecast data need to be brought together for the classification of the radio links. The combination operation was performed based on the correlation formed between RL site id and WS weather station id, through the optimal distance  $\mathbf{d}$ . As the weather forecast affects radio tower link features differently during different times of the year, the final data achieved sequential properties by combining it with the weather forecast data. This is a major contribution of the work; maintaining the link sequence with forecasts in order to accurately identify the link failure at various times during the year. This is not available in the current state-of-the-art solution [18].

Before performing the merging operation, the weather forecast data required more processing. As mentioned in Section 2.2, the weather report dataset contains the forecast data for the next five days for each particular date. This report includes temperature, humidity, wind direction, speed and weather type. In order to predict the link failures for the next one day, we extracted only the forecast data for the next day from the dataset. Also, we only extracted the morning forecast data as the radio tower KPI samples were collected in the morning only. The spatial data (ground height and clutter class) for each stations were added to the modified forecast dataset.

Then, we combined the radio KPI data with the forecast data. For each radio link failure prediction, we needed the weather forecast data for the next day in our prediction. As mentioned before, this weather forecast data is contained in the current date. So, the alignment was made such that the relationship between this particular date and the KPI data would be preserved in the model.

The combination between the radio link KPI data and the modified weather forecast data was done based on the correlation formed by optimal distance  $\mathbf{d}$ . As there were no weather forecast reports for the first entry in the dataset, '01/01/2019', the KPI dataset also had its entry removed for the same date to ensure the correct alignment, which was then designated as modified KPI data. However, we must also note that the KPI data is also used has historical data in our model. Thus the final dataset was combined such that it contained the KPI data and the weather report of the same day for each radio link, as well as the previous day KPI data to use as historical data.

However, we only kept the target feature for the modified radio KPI data as the failure prediction utilized the ML classification task and not the regression task.

Also, we parsed through the 'date-time' feature of the final combined dataset and extracted the month as an integer number. Then we added a 'month' feature to the modified combined data which contained the integer month value [69]. The temporal attribute of the data was tracked using this feature and the corresponding index. We used this feature as a reference for the further semi-supervised pre-processing operations such as dimension reduction, data-balancing and so on.

The final dataset was structured as follows,

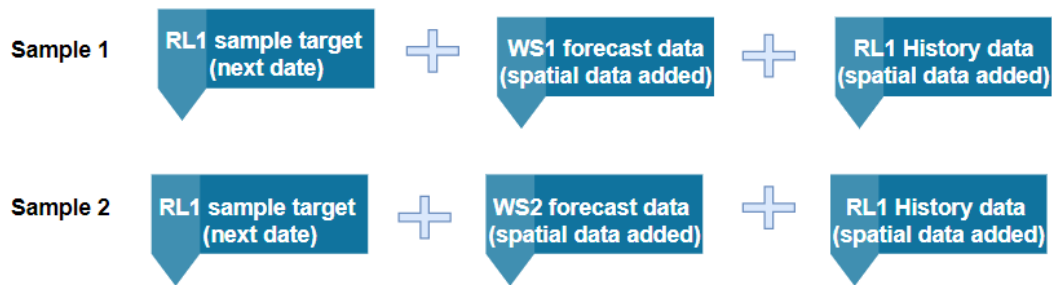


Figure 4.5: Data Structure

### 4.1.3 Missing Data

A data quality report was produced from the final combined data to find out the data quality issues and plan to handle the issues. The data quality report was produced separately for the continuous features of the data and the categorical features of the data [70]. This approach helped in identifying the specific issues in the separate types of the data. The data quality metrics for the continuous features are:

1. Number of samples.
2. Percentage of missing data of each features.
3. Cardinality.
4. Maximum and minimum value.
5. First and third quartiles of the data.

6. Mean and Median.

7. Standard deviation.

FEATURE_NAME	Count	Miss %	Card.	Min	1st Qrt.	Mean	Median	3rd Qrt	Max	Std. Dev.
mw_connection_no	838254	0	1496	143594	223922	8295861	348495	1385238	430874.4	
link_length	74227	91.2	1208	30	1236	4805.34	2835	6600	43222	5210.84
severaly_error_second	838254	0	305	0	0	0.34	0	0	8679	26.47
error_second	838254	0	1088	0	0	5.71	0	0	86400	354.96
unavail_second	838100	4	3107	0	0	168.33	0	0	172798	3513.86
avail_time	838254	0	2301	1	86400	82156.31	86400	86400	172800	15885.12
bbe	838254	0	4061	0	0	107.44	0	0	3403779	8108.34
rxlevmax	838254	0	594	-99.9	-40.5	-38.66	-39.5	-34.7	0	8.45
scalibility_score	386670	53	10586							
capacity	838254	0	68	8	154	260.66	203	406	495	135.31
temp_max_day1	838254	0	50	-7	16	22.76	23	30	43	8.75
temp_min_day1	838254	0	42	-13	7	12.4	12	18	29	6.93
humidity_max_day1	866235	8	70	26	72	80.45	84	91	96	13.71
humidity_min_day1	838254	0	92	5	36	51.29	52	67	96	20.3
wind_dir_day1	838254	0	361	0	89	186.96	195	273	360	105.61
wind_speed_day1	838254	0	51	2	8	11.68	11	14	53	5.44

Figure 4.6: Data Quality report of the Continuous features

The data quality metrics for categorical features are:

1. Number of samples.
2. Percentage of missing data of each features.
3. Cardinality.
4. First Mode value, Mode Frequency and Mode percentage.
5. Second Mode value, Model Frequency and Mode percentage.

FEATURE_NAME	Count	Miss %	Cardinality	Mode	Mode Freq	Mode %
type	838254	0	2	ENK	564455	67.34
tip	838254	0	2	NEAR	422365	50.39
direction	386670	53	1	Near End	386670	100
polarization	69264	91.7	2	Vertical	53569	77.34
card_type	838254	0	6	cardtype4	353585	42.18
adaptive_modulatio	838254	0	2	Enable	773906	92.32
freq_band	832628	2	5	f3	417020	50.08
modulation	838254	0	19	512QAM	320582	38.24
weather_day1	838254	0	20	scattered	3638	21.65

Figure 4.7: Data quality report of the categorical features.

In order to solve this issue of data quality, we devised a plan, wherein the first step was to prepare a solution for the missing values in both continuous and categorical features. The continuous features which contained missing values are the following: ‘link-length’, ‘scalability-score’, ‘min-temperature’, ‘max-temperature’, ‘humidity-max’, ‘humidity-min’, ‘wind-speed’, ‘error-seconds’. The solution applied for the missing values in the continuous features are,

**The mean.** The mean of the values were calculated and placed in the missing cell. This method was applied on min-temperature, max-temperature, humidity-max, humidity-min and wind-speed, and link-length.

**The median.** The median was calculated based on the previous and the next value of the missing value cell [71]. This method was applied on the ‘scalability-score’ feature which determines the scaling ability of the frequency modulation and ranges from 0 to 1. In order to keep the value unbiased, the median was calculated and filled in the missing cell.

**Removing and zero filling.** This approach was applied on the feature *neid*. The *neid* determines the id of the each radio link sample. Filling the cells with any value would introduce a bias in the dataset. The sample with missing *neid* had been removed from the dataset. The ‘0’ is placed in the missing cell for the ‘error-second’ feature as it determines how many seconds the error links are up before reporting the failure. Filling it with zero was the most optimal solution for this feature.

The categorical features which contain missing values are the following: ‘polarization’, ‘frequency-band’, ‘direction’.

**Mode.** The missing values in the categorical features are filled with the Mode value of the particular features i.e., Frequency band and direction [71].

However, in case of a missing value for the feature ‘polarization’, the sample was entirely removed from the dataset [72]. ‘Polarization’ contained only two values in the dataset which are *enable* and *disable*. Filling out with either of these values would introduce bias in the dataset. Removing the sample assured the purity of the dataset in terms of noise and bias.

#### 4.1.4 Categorical Feature Encoding

The categorical features needed to be encoded with binary vectors to train a machine learning algorithm. The existing solution applied the binary encoding method for categorical data. In binary encoding, the categorical features were initially converted to numerical ones using an ordinal encoder. The numbers were converted to binary numbers. After that, the binary values were divided into columns. The binary encoding heuristic of the categorical data was performed as follows,

1. The categories were first converted to numeric order starting from 1 (order is created as categories appear in a dataset and do not mean have any ordinal significance)
2. The integers were converted into binary code, so for example, 3 becomes 011, 4 becomes 100 and so on.
3. Then the digits of the binary number were set as separate columns.

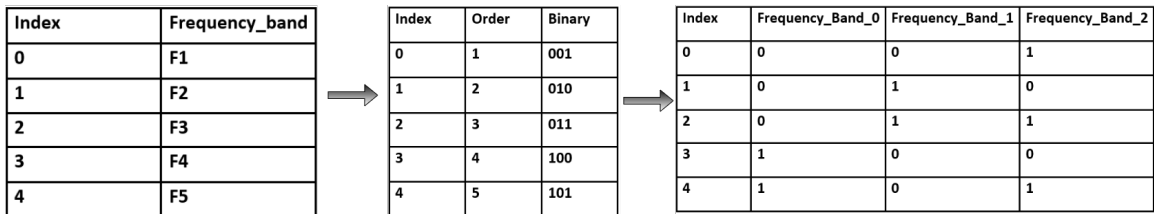



Figure 4.8: Binary encoding to categorical features.

The number of features in the dataset after binary encoding was 67. The models were trained with binary encoded data, reconstruction errors were calculated and classification task for failure detection was performed. Due to the ordinal properties of the binary encoding method which can impact the overall model performance, we applied another encoding method called one hot encoding to observe the model performance difference. The one hot encoding system was applied and the data was prepared for model training separately.

One hot encoding specifically works with categorical data that is nominal [23]. In our dataset, the categorical data did not have any ordinal properties. So one hot encoding was applied on the categorical features in order to eliminate the ordinal

properties from the encoded data. In a binary classification task, the numerical labels produced from the binary encoding may affect the loss calculation of the model training especially in a real world failure event scenario. A small bias during the training period may degrade the model's feature learning ability, which ultimately adversely affects prediction accuracy. This statement is further justified in Section 4.4 by multiple evaluations and comparisons of prediction results. The heuristic of the one hot encoding in the dataset is as follows, In one hot encoding, for each level of a categorical feature, we created a new variable. Each category was mapped with a binary variable containing either 0 or 1. Here, 0 represented the absence, and 1 represented the presence of that category. These newly created binary features were known as 'dummy variables'. The number of dummy variables depends on the levels present in the categorical variable [23]. The number of features in the original data before encoding was 29, which subsequently increased to 1377 after encoding.



Index	Weather_types
0	Heavy_Rain
1	Snow
2	Windy

Index	Weather_type_Heavy_Rain	Weather_type_Snow	Weather_type_Windy
0	1	0	0
1	0	1	0
2	0	0	1

Figure 4.9: One hot encoding to Categorical Features.

#### 4.1.5 Feature Selection

Feature selection was used for reducing the high dimensional feature count. The reduction of feature count improved the performance of the ensemble learning model of the existing solution [18]. Features were removed in an iterative way by using the feature importance property of a supervised classifier. This was calculated for each feature of the dataset where a higher score represented the importance of the feature in classification operation. A linear algorithm (Logistic Regression) was used to find a set of coefficients to use in the weighted sum in order to make the prediction. These coefficients were used directly as a crude type of feature importance score [26].

The feature coefficient scores were both negative and positive as shown in Fig.

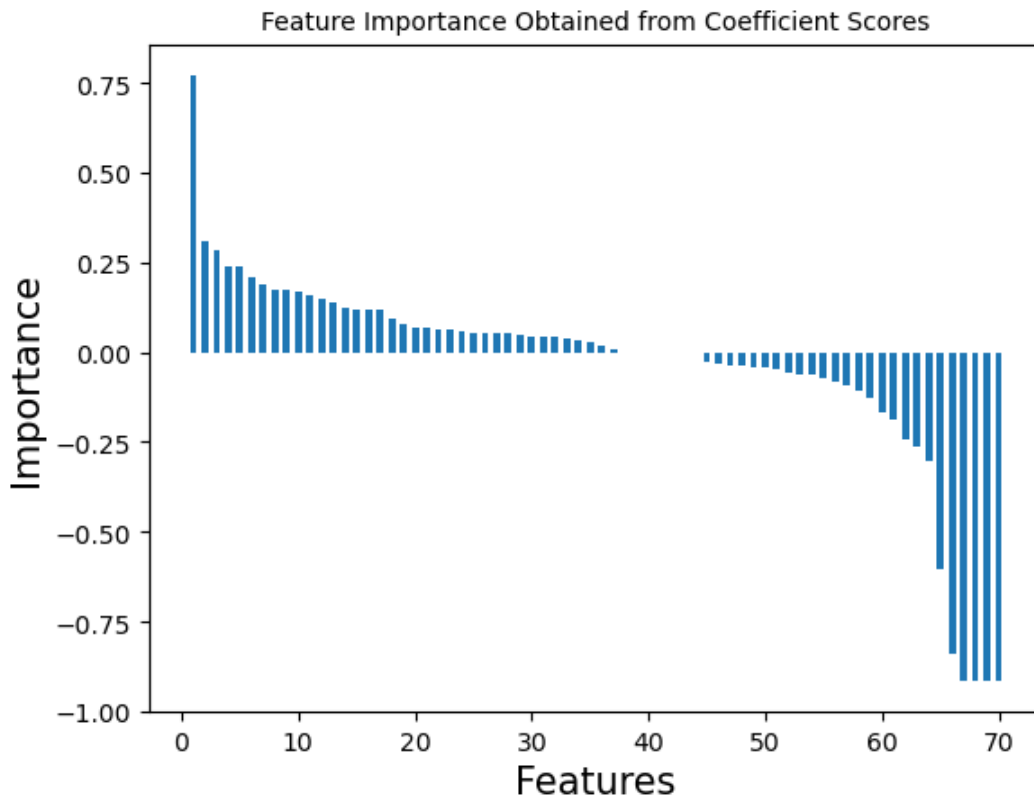


Figure 4.10: Feature Coefficient scores.

4.10. The feature name was changed to numbers only for the better plot interpretation. Here, the positive scores indicated the importance of classifying the class 1 ( failed link ) and negative scores indicated the importance of classifying the class 0 ( normal link ). The features with absolute less scores were iteratively removed until the F1 score of the linear classifier was reduced. The number of features before applying the feature selection method was 67. which dropped to 51 after applying the feature selection.

#### 4.1.6 Principal Component Analysis

The high dimensionality of the data after the categorical encoding operation may lead to the model over-fitting during the training as well as cause features to appear equally spaced from each other, which is one of the reasons why a model cannot learn proper distribution [29]. We employed an unsupervised approach called Principal



Component Analysis (PCA) to tackle this issue regarding high dimensionality. Generally, dimensionality reduction involves reducing the complexity of a model as well as decreasing the computation cost. In contrast, losing the dimensions may negatively affect model performance, as it may contain critical information about the feature distribution. There are two methods for reducing the dimensionality of a dataset: feature selection and feature extraction. The feature selection algorithm selects a subset of the original features, whereas the feature extraction algorithm identifies information from the original feature set to construct a new feature subspace [73]. The PCA algorithm involves feature extraction by constructing new components for a new feature space from the original feature set. The PCA attempts to determine the directions of the maximum variance in high-dimensional data and project it into a subspace with fewer dimensions than the original one. In other words, PCA can be defined as the linear projection that minimizes the mean squared distance between the data points and their projections. Moreover, due to the transformation of the original  $d$ -dimensional data onto the new  $k$ -dimensional subspace (usually  $k \ll d$ ), all information and variance associated with the original feature distribution are included in the first few principal components. In case of the binary representation, PCA considers the internal geometrical interpretations of the data points [74,75]. For example, if  $[x_1, x_2, \dots, x_n]$  are  $n$  variables, consider a  $k$ -dimensional ( $k \ll d$ ) linear space spanned by the orthogonal binary bases  $[b_1, b_2, \dots, b_k]$  with the shift vector  $\mu$ , then PCA minimizes the new feature space reconstruction error  $E_b$  by employing the following equation 4.1:

$$E_b = \sum_{i=1}^n \| x_i - (\mu + a_{i1}b_1 + \dots + a_{ik}b_k) \|^2 \quad (4.1)$$

The PCA algorithm also ranks components in descending order in the new data space according to the variance/information, so the model learns about the distribution of the features mainly from the top-ranked components. PCA considers the linear correlation of the features and projects the feature map accordingly [76]. The feature mapping for linear representation depends on the internal correlation and geometrical interpretation among the features. However, PCA did not perform well on binary encoded data due to the ordinal structure of the binary encoding approach. This also affected the prediction results. The total principal components was 57 from

the original 63 binary encoded features, which indicates poor performance of feature compression for the binary encoded data. Thus, we applied a new encoding method called one hot encoding to the original data. Training a model with lower rank components has little to no impact on the learning of feature distribution of the model [73]. This property of the PCA algorithm allowed us to remove the lower-ranking components without losing any information, thus reducing the overall dimensions of the data.

We applied the Principal Component Analysis in our data such that the retrieved variance was emphasized and the final data dimensions were chosen accordingly. The model was tested for different variances in an iterative process, and a threshold for the number of components was selected based on its performance and computation cost. Our study determined that the conditions of 90% variance retention, with 728 components for one hot encoded data, produced the best results, as seen in Table 4.1. The iterative experimentation with various number of principal components based on the final prediction score and computation cost allowed us to select the most optimal setting for our model training. The features of the one hot encoded data before applying the PCA were 1377, which was transformed into less dimensional components after applying PCA.

Principal Components	Retained Information	F1 Score	Training Time
1377	100%	0.71	310 mins
1136	96%	0.85	230 mins
<b>728</b>	<b>90%</b>	<b>0.88</b>	150 mins
665	85%	0.82	150 mins
413	75%	0.77	120 mins
169	65%	0.61	30 mins

Table 4.1: Principal Components VS Retained Variance.

## 4.2 Data Preparation

The failure event depends on the weather condition of specific regions. The recorded failure in the data is rare due to the low fluctuation of weather condition. The KPI data contains only 0.31% of total samples which resulted radio link failure, which translates to a ratio of 1:320 for failure to regular features.

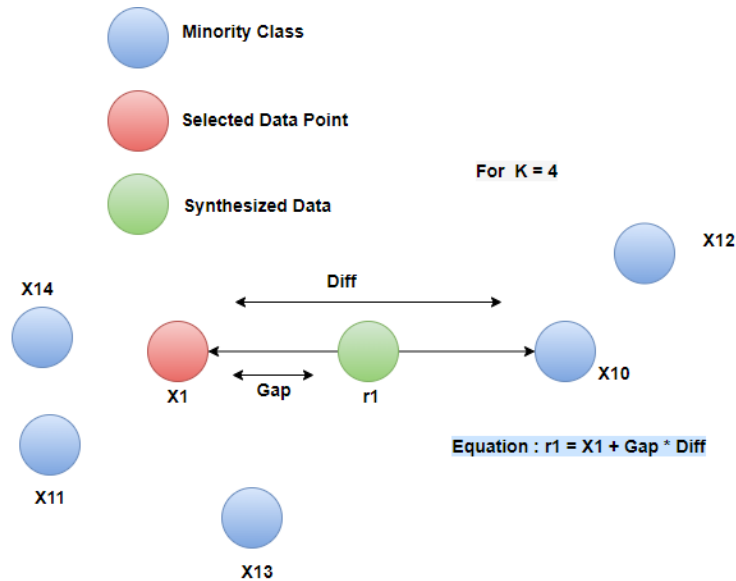


Figure 4.11: Data balancing operations using synthetic approach.

Imbalanced data classification has the drawback of having too few samples of the minority class for a model to learn the decision boundary successfully. The imbalanced data can be balanced using oversampling and under sampling techniques. Existing solutions incorporated the oversampling technique SMOTE ( Synthetic Minority Oversampling Technique) [19], which enhances the minority class in a dataset by synthesizing new samples in the feature space. The method selects the samples that are close to the feature space, then draws a line between the samples in the feature space and adds a new sample at the point along the line. Thus, the new sample is added to the dataset without bias and noise.

Based on the dataset, the total number of majority  $N^+$  and minority  $N^-$  samples were determined. Our next step was to determine the threshold degree of class imbalance, generating the optimal number of samples accordingly. We calculated KNN's for each minority sample  $x1$  using Euclidean distance, and normalize  $r1$  as  $rx \leq r1 / \sum r1$  based on the ratio  $r1$  calculated for  $\delta i/k$ . The total number of synthetic samples for each data point were  $G = rx \times \text{Gap}$ . For our implementation of the synthetic oversampling, the total number of oversampling observations,  $N$ , is set up. Generally, this was selected such that the class distribution ratio was as close possible. Then, we started the iteration by first selecting a positive class instance. We selected  $k = 4$  for the nearest neighbors. The difference in distance between the

feature vector and its neighbors was calculated using the Euclidean distance equation which is shown in the Fig. 4.11. This calculated difference was multiplied by any random value in  $[0, 1]$  to determine the gap and new instances were synthesized accordingly. This generated new positive instances, which were merged into the data. The final minority to majority ratio after the re-balancing operation improved from 1:320 to 1:10.

#### 4.2.1 Data Segmentation

The data requires segmentation into train data and validation data for the model to calculate the performance. Using timed forecast data in conjunction with KPI data from the radio tower, we achieved the sequential properties as mentioned in the previous section. The segmentation required a systematic approach in order to maintain the sequence of the data. The time series segmentation needs to be aware of the fact that, the train set has to be always behind the validation set sequentially so that the learning model does not incorrectly use future train data instead past data for predictions.

The existing solution applied 70:30 segmentation of the entire dataset where first nine months were considered as the train set and the rest three months are considered as the test set [77]. The model was trained on the first nine months of 2019 and the prediction score was calculated on the last three months of the test set.

However, we used expanding window time series cross validation method for the data segmentation into the train set, validation set and test set [78]. The idea for time series splits was to divide the training set into two folds at each iteration on the condition that the validation set is always ahead of the training split [79]. The splitting was performed with k-fold Cross Validation with 5 iterations where  $k = 2$ . A ‘*TimeBasedCV*’ custom splitter function was updated and modified from the default *TimeSeriesSplit* from the *scikit-learn* module, that enabled us to control the time window of the data sequence and the method was imported from a open source git repository [80]. Then it was used as a main *splitter()* function with the *GridSearchcv* model for performing the cross validation of the data segmentation [81]. Every cross-validation iteration used one fold as a validation set and  $k - 1$  folds as a training set. Following that, the standard deviation was calculated by averaging all folds. At the

first iteration, we trained the proposed candidate model on the *RLF* from January to April and validated on May data, and for the next iteration, trained on data from January to May, and validated on June data, and so on to the end of the training set. The October, November and December 2019 data was selected for testing the model which is shown in Figure 4.12. The training and testing of the proposed model performance with the segmented data is described in a later section. As mentioned, we utilized the *GridSearchcv* statistical model to perform the cross validation [82]. The cross validation accuracy of segmentation was computed by averaging over the validation sets, which is 0.9954123658.

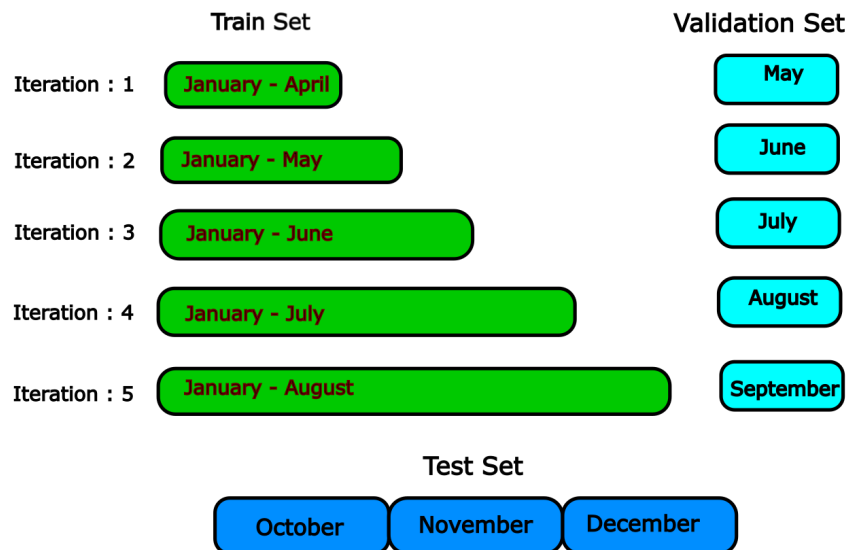


Figure 4.12: Expanding window Time series segmentation.

We also collected the data from 2020 and tested the model with the 2020 data to gain better standing with the results about the proposed solution. The prepared new test data from 2020 contains weather forecast data from weather stations and key performance data from radio tower sensors of January, 2020 to June, 2020 . The data from 2020 was pre-processed in the similar way as the 2019 dataset and prepared accordingly before testing.

### 4.2.2 Data Normalization

We applied a normalization algorithm on the data before using it for model training. The *scikit-learn* library module provided *StandardScaling* method to scale the data in range  $[0, 1]$ . The gradient descent of the neural network helped optimize the training and this optimization requires the data to be scaled. Scaling also helped the activation function to determine the output for the neurons of the network where close to 1 leads to the positive classification result and 0 to the negative one. This was an important step of data preparation before training the learning models.

## 4.3 Model Training

### 4.3.1 Neural Network based learning Methods

**Fully Connected Convolution Neural Network.** We used a fully connected Convolution Neural Network ( FCNN ) as our baseline model for deep learning based solutions. The main goal of using FCNN is to observe the data properties and behaviour on a neural network environment. Convolutional Neural Networks are a type of deep learning model, which can capture spatial patterns through training filters and assign weights to them. The model was trained with the first 9 months and tested with the last 3 months of 2019. The goal of the FCNN implementation is to observe and improve upon the existing solution by employing the spatial feature pattern instead of the decision-tree boundary, as in the NEC implementation [18]. The model was implemented such that it worked in three consecutive stages: data transformation, local convolution and full convolution [83, 84]. The model workflow is described below.

**Data Transformation:** Transformation of the data was applied to the original set in three aforementioned branches. We transformed the original train set differently with each successive iteration. This transformation mapped the identity of the original series so that it preserved the integrity. Following the splitting of the input series, a moving average was calculated. A variety of new input series were created with varying levels of smoothness. Coefficient values were used to down sample the original sets.

**Local Convolution:** Following this, a 1-D convolution with different filter sizes

was applied by passing each convolution layer through a global max pooling layer as shown in Fig. 4.13. We applied the formula  $n/p$ , where  $n$  is the length of the input series in terms of  $p$ , the pooling factor by which we determined the size of the pooling kernel. This algorithm used the original input series as input along with two down-sampled versions of it (half sampled and quarter sampled). The corresponding convolutional kernel was  $[64 \times 728]$  as the convolutional kernels always have the same width as the input series with varied length of data. In the second branch, the input series are down sampled and the filter size used here was  $[32 \times 728]$ . Then, the third branch of the transformation processed the shorter version than previous version of the input series, with a filter size of  $[8 \times 728]$ .

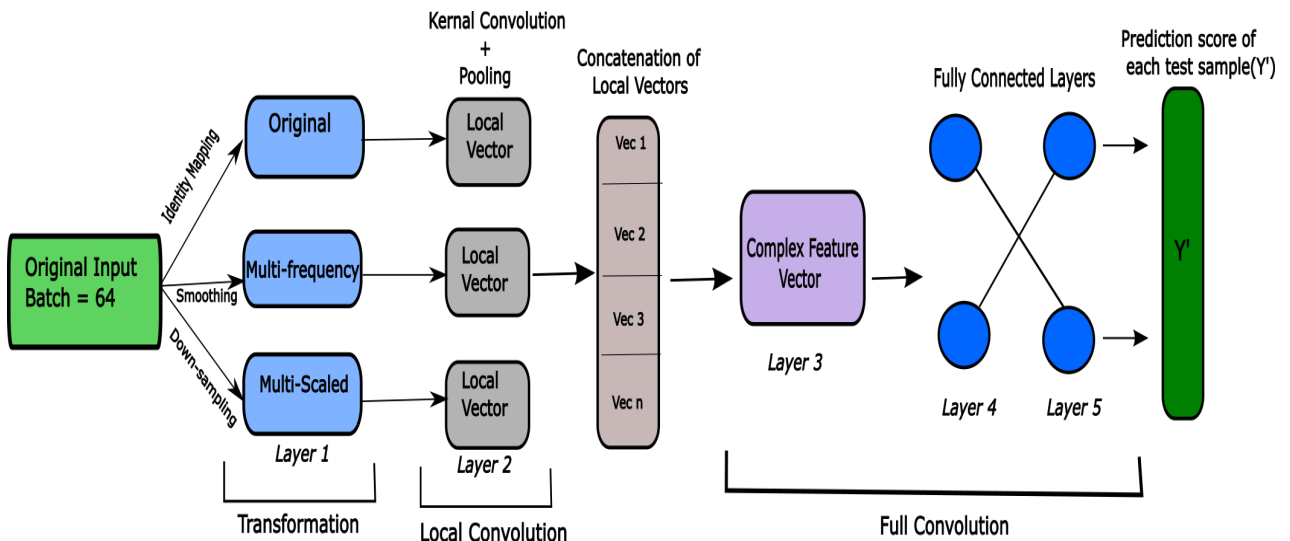


Figure 4.13: Fully connected CNN architecture and workflow.

**Fully Connected Convolution.** At this stage, the output from the aforementioned three branches were concatenated which is represented in Figure 4.13. We converged the 1-D convolutional outputs and added more max-pooling layers. A compressed vector resulted from full convolution, which captured train-set data on a wide range of frequencies. In the subsequent layers, this vector was used as an input into fully connected layers with the softmax function. A dropout regularization layer was added after convolution and global pools, and all the outputs are concatenated. The last fully connected layer outputs the prediction values for the samples.

We used Sigmoid activation function for each layers as we were calculating the

probability distribution for prediction. During each training epoch, the model distinguished the input based on their high-level features and classified them according to the feature mapping [85]. After the training was complete, we fit the test data to evaluate the model performance. All of the high-level features learned by the local convolution layers were flattened into a one-dimensional local vector. A final vector (same size as test input data) containing the probability distribution score was created by concatenating all of the flattened vectors produced by the local convolutions. It was calculated using dot multiplication of the feature weights. As a result, the final output was a one-dimensional vector of the probability scores that were obtained from the high-level feature mapping of the test data in the convolution layers. Then, we selected an optimal threshold based on the probabilities generated from the test input data in order to identify the failed links. The classification details are discussed elaborately in section 4.4. A summary of the training process of the FCNN is given below.

1. At every epoch, the output was computed using the present input vector.
2. The output was a vector where the elements are the estimated probability scores of the classes of the target feature.
3. The prediction error was computed using a cost function. The cost function was recorded and the cost value converged to a solution at the final epoch.
4. The weights were updated in a backward pass to propagate the error by using gradient descent.

The hyper-parameters are listed in Table 4.2.

**LSTM - Autoencoder.** An Autoencoder with an LSTM layer was applied to observe and improve the previous FCNN solution by utilizing the reconstruction property of the Autoencoder and the sequence maintaining property of the LSTM. We planned to improve upon the previous neural network predictions by utilizing the sequential properties along with the spatial properties of the features. Depending on the input data properties, we applied the Recurrent Neural Network (RNN)-based LSTM nodes coupled with Autoencoder architectures, where sequential correlations of radio link features are used during the model training. We used a feed-forward



<b>Hyper-Parameter</b>	<b>Selected Value</b>
Epochs	50
Total hidden layers	5
Batch size	64
Dropout	0.3
Activation function	Sigmoid
Computation	CUDA, Cluster GPU
Loss function	Mean Absolute Error (MAE)

Table 4.2: Hyper-parameters of FCNN.

neural network-based approach to solve this problem because RNNs consider both the previous output and current input at each epoch.

The reason that we decided to use the Autoencoder in our proposed model for failure classification is the fact that the Autoencoder tries to learn the best parameters to reconstruct the input at the output layer. Moreover, we adapted the LSTM algorithm for our model to solve the issues of the standard RNN technique, such as vanishing and exploding gradient problems. The LSTM - Autoencoder model training and validation heuristic is described below.

1. The train and validation data are separated into two parts : zero labeled (Normal Link) and mixed labeled (both normal and failed links).
2. The zero labeled data was the normal link state of the sample. A normal link state is when the link does not face any failure.
3. We ignored the mixed labeled data, and trained an Autoencoder on only zero labeled data.
4. This Autoencoder learned the features of the normal link state.
5. A well-trained Autoencoder is able to predict any new data that is coming from the normal link state as it will have the same statistical distribution. Thus, the reconstruction error is small.
6. In case of mixed labelled data, the autoencoder struggles to reconstruct the data from failure event and generate high reconstruction error for those mixed labelled samples. The samples with high reconstruction error are classified as the failure samples.

**Additional data preparation for LSTM layers.** The input required more preprocessing to fit into the LSTM layers. The input data to an LSTM neuron usually is a 3-dimensional array. The shape of the array is  $[samples \times lookback \times features]$ .

*samples:* This is the number of samples or the number of data points to fit into encoder. We used it to train and validate our model and observed the performance.

*lookback:* LSTM layers were mainly used to maintain the sequence of the data, as the model requires looking at the past data points. The LSTM processes the data up to (t-lookback) at time  $t$ . In this case, the time sequence represented each date of the year.

*features:* It is the number of features in the input data.

The Hyper-parameters are listed in Table 4.3.

Hyper-Parameter	Selected Value
Epochs	50
Learning rate	0.001
Hidden layers	4
Batch size	16
Optimizer	Adam
Dropout	0.3
Activation function	ReLU
Computation	CUDA, Cluster GPU
Loss function	Mean Absolute Error (MAE)

Table 4.3: Hyper-parameters of LSTM-AE.

**Implementation Details:** Our model used LSTM with autoencoder to learn the representations of the network dataset in a semi-supervised fashion. It contained multiple layers of encoder and decoder stages and each stage consists of multiple LSTM units [15]. The final architecture of our proposed model includes 4 layers for each encoder and the decoder including the input layer and the output layer. The operation is explained for one day time lag and one hot encoded data with PCA applied for dimension reduction. The lookback time for LSTM units is the hyper-parameter of the model which was tuned for various previous timesteps (number of past dates) during the experimentation to achieve the best prediction results.

**Encoder.** The input data  $X(n)$  was encoded via the encoder block to generate a fixed range feature vector  $Z(n)$ . As shown in the figure 4.14, the input data  $X(n) \in$

$\mathbb{R}^{728 \times 1 \times 16}$  was the initial features from the dataset. The encoder block sequentially reduces the dimension of the 728-dimension initial features. The dimensions were reduced to 364, 91, 45, and 22, after the first, second, third and fourth layers of the encoder respectively. The final encoded feature vector  $Z(n) \in \mathbb{R}^{22}$  represented the compressed latent input data. *RepeatVector*, replicated the feature vector with input shape of  $[22 \times 1 \times 16]$  for the decoder. The *RepeatVector* layer acted as a bridge between the encoder and decoder modules [17]. The Decoder layer was designed to unfold the encoding. As mentioned, for encoder-decoder, the input was reduced into a compressed feature vector, to regenerate output as the same dimension as the original input. The *RepeatVector* converted the feature vector tensor to retrieve the output of same the dimension. *RepeatVector* connects the encoder to decoder by retrieving the output from the last layer of encoder.

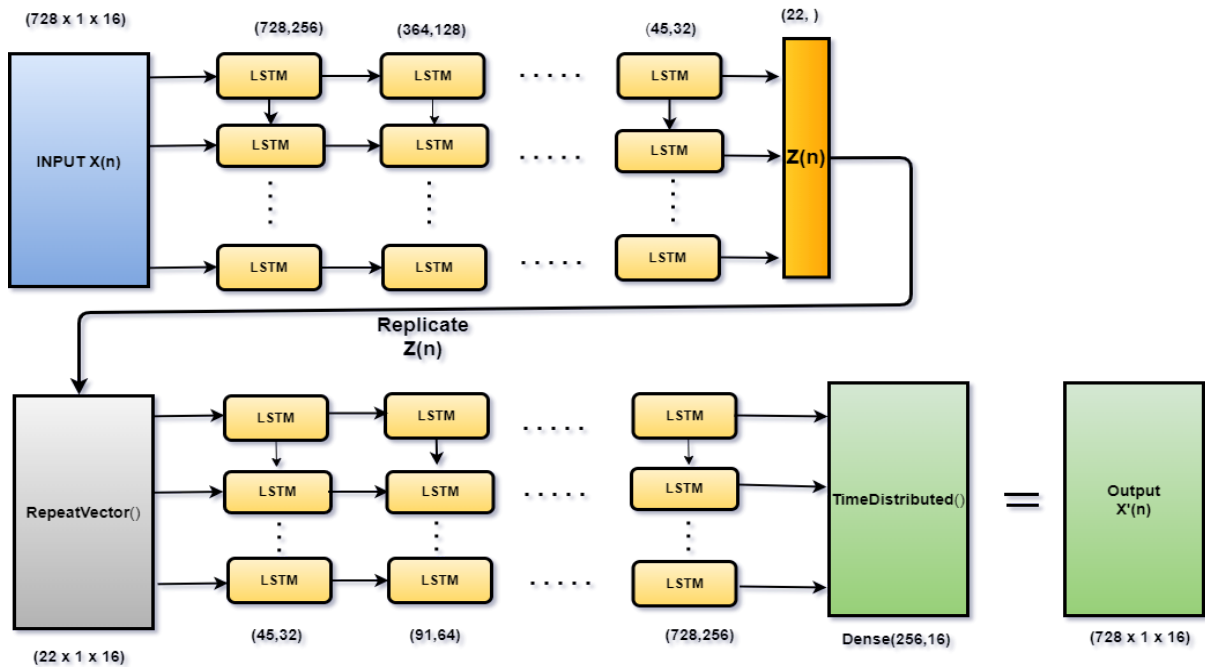


Figure 4.14: LSTM - Autoencoder Architecture.

**Decoder.** Similar to the encoder block, the layers in the decoder block were arranged in reverse order. The encoded features  $Z(n)$  were sent through a series of LSTM blocks to generate the output feature vector  $X'(n)$ . Each subsequent layer of the decoder increases the dimensions again, in reverse order.

The encoded feature vector was fed into the decoder block for generating the

output feature. We represented the input feature vector of the decoder block as  $Z(n)$ . LSTM blocks were then used to generate an output feature vector  $X'(n)$ , based on the encoded features  $Z(n)$ . The dimensions were increased in reverse order of the encoder. *TimeDistributed(Dense())* was added in the end to get the output which is shown in the model diagram figure 4.14. The *TimeDistributed* layer created a vector of length equal to the number of features output from the previous layer. In this network, the final decoder layer outputted 728 features. Therefore, the *TimeDistributed* layer created a 728 long vector and duplicated it 16 (mini-batch) times. The output of the last layer was a  $728 \times 256$  array that we denoted as U and the output from the dense layer was  $256 \times 16$  array which was denoted as V. Multiplication of U and V yielded a result  $728 \times 16$  matching the input size and dimensions.

The output feature vector  $X'(n)$  was reconstructed to be as similar to the input feature vector  $X(n)$  as possible. An estimation error between input and output data was calculated using the mean absolute error (MAE). LSTM - Autoencoder network was used to model normal traffic data based on the discriminatory features presented in the data. The reconstruction error for normal sample data is supposed to be less when compared to that of failed sample data. Due to this behavior, the corresponding error value for the failed link will be significantly higher, making it easier for one to detect the failure.

First, we checked our model performance with a smaller set of the data. In a single training and validation iteration, we separated  $1/5^{th}$  of the data which is the first iteration of the time series splitting. It contained train data with no failure from January 2019 to April 2019 and the validation data with mixed classes (normal and positive) of May 2019. The train data had  $427080 \times 728$  examples (122 timesteps) with no failure links. The validation data had  $106770 \times 728$  (30 timesteps) examples with mixed labels. We trained and validated our model and recorded the reconstruction loss. The train loss converged to a solution after some time of the training which indicates the proper learning of the model. The reconstruction loss for the validation set were a bit higher than the train loss which indicated that our model struggled to reconstruct the samples with failure event.

After that, we repeated the above phase five times according to the iterations of the data segmentation, i.e., we fed the whole data set of same number of features from

1.2 million samples of training data and 0.53 million samples of the validation data. Although, the model took a long time (approximately 230 minutes) to finish the training and the validation, the loss converged to a solution which indicated the learning of the model. We experimented the training of the model with multiple parameter tuning for achieving the best version of the trained model. The experimentation included different types of encoded training data, varying amounts of historical data for past observation during the feature compression, as well as different numbers of hidden layers and LSTM units. We performed the experiments in a sequence that allowed us to determine the best setting for a specific step, and then used the knowledge to perform the next experiment to achieve even greater prediction accuracy. Evaluations and results of the experiments are discussed in Section 4.4.

Following that, we trained the model with data from the new time span of 2020 in the same environment setting. The data from 2020 was similarly pre-processed and prepared and sent to the model that had been trained with the data from 2019. The model was tested on the new data and we evaluated its data reconstruction performance with the temporal properties presented in the data. The next step was to predict link failures using the standard evaluation metrics, which is discussed in more detail in the next section, using the optimal threshold derived from the reconstruction loss.

### 4.3.2 Supervised Ensemble Learning Method

We also implemented the current state-of-art solution to recreate the prediction result and compared it with our proposed solution. As a general practice in machine learning, ensemble learning seeks to improve predictive performance by combining predictions from multiple models. The existing solution from NEC corp [18] trained a decision-tree based ensemble learning model that includes Random Forest (RF) classifier, Light Gradient Boosting Machine (LGBM) and Extreme Gradient Boosting Machine (XGB).

In order to recreate the state of the art solution, we implemented the ensemble learning solution with the decision tree based models in the pipeline. We particularly used Random Forest (RF), Light Gradient Boosting Machine (LGBM) and Extreme Gradient Boosting Machine (XGB). Here, we selected  $n\_estimator = 1000$ , which

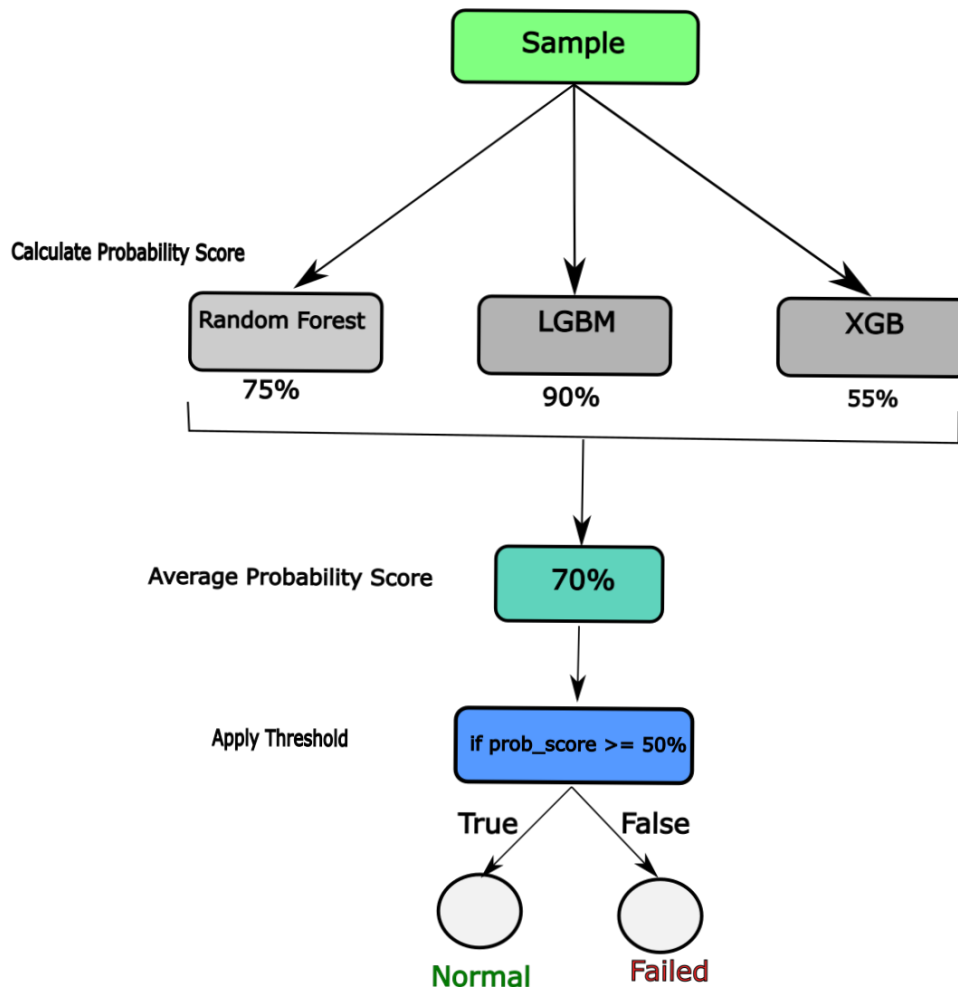


Figure 4.15: Ensemble method and its prediction strategy.

is the number of trees the algorithm built before taking the maximum voting. The maximum tree depth was four. The minimum sample splits were set to two as its a binary class classification. We selected two maximum terminal nodes from each tree which prevented the tree from growing any further to prevent the model overfitting.  $1/5^{th}$  sample were left out for out-of-the-bag sample error estimation which was used to get the unbiased estimate of the classification error as trees were added to the forest iteratively [86]. It was also used to estimate variable importance.

We ran all of the data down each tree, created the pair of classes and computed proximities for each pair. When two cases occupied the same terminal node, their

proximity was increased by one. After that, we normalized the proximities by dividing by the number of trees (set up in the estimator) at the end of the train run. A low-dimensional view of the data was produced by using proximities. For each splitting of the node, the *Gini impurity criterion* for split node was less than the parent node [86], which ultimately decreased the variable count in the node which led to the faster training of the model.

Finally, the model calculated the probability scores for each test sample by utilizing the `oob_score` estimation and output the results. The LGBM and XGB were utilized to boost the training rate of the RF by performing optimal gradient descent. The number of horizontal leaves set to  $2^{\text{max\_depth}}$  where `max_depth` of the tree was set to 2 for the boosting algorithms. Keeping the `max_depth` low helped the model prevent overfitting [86]. The average prediction scores calculated from each three model were determined as the final prediction score of the ensemble model. The ensemble model hyper-parameters are listed in Table 4.4.

The models were trained with similar train data which contained data from January 2019 to September 2019. After training was completed, we tested the models with data from October 2019 to December 2019 and calculated the failure probability score for each sample of the target ‘**RLF**’. A probability score was calculated based on the scores generated from each model in the ensemble. However, the existing solution used a fixed threshold for performing the classification task. During our implementation of the existing solution to re-create similar output, we used both fixed threshold of 0.5 and an optimal threshold of 0.58. The optimal threshold was selected iteratively from achieving the highest prediction results. The prediction results were improved when an optimal threshold was used instead of a fixed one which is a contribution for improvement to the existing solution. Using the threshold value of failure probability of each sample as the denominator, we declared a link failed if the target probability score is higher than the threshold value.

#### 4.4 Evaluation

In this section, we discuss the evaluation process and results of our proposed solution along with the state of the art solution [18], which shows that our method provides better performance in terms of classification score in detecting link failures from the

Hyper-Parameter	Selected Value
Number of trees (RF)	1000
Tree depth (RF)	4
No. of leaves (per node) (LGBM, XGB)	2
Max tree depth (LGBM, XGB)	2

Table 4.4: Hyper-parameters of ensemble learning.

weather forecast and the sensor key performance indicator data.

#### 4.4.1 Performance metrics

The performance metrics of the model are classification accuracy, precision, recall and F1 score for evaluating the existing ensemble learning solution, the baseline convolution neural network solution, and proposed candidate method LSTM - Autoencoder. We also used the receiver operating characteristic curve (ROC) to represent the classification results. The mathematical representation of the metrics are calculated as follows.

$$Precision = \frac{TP}{(TP + FP)} \quad (4.2)$$

$$Recall = \frac{TP}{(TP + FN)} \quad (4.3)$$

$$F1\_Score = 2 \times \frac{(Precision * Recall)}{(Precision + Recall)} \quad (4.4)$$

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FN + FP)} \quad (4.5)$$

where TP (True Positive) represents the number of instances correctly classified as a failure; TN (True Negative) represents the number of instances correctly classified as normal link; FP (False Positive) represents the number of instances incorrectly classified a failed link; FN (False Negative) represents the number of instances incorrectly classified as normal.

However, we did not consider accuracy from Eq. 4.5 as our definitive evaluation metric as like precision, recall and F1 score. Usually, accuracy is a good measure of performance evaluation in the symmetrically balanced data ( ratio 1:1) but often fails to represent the correct results for imbalanced data [87,88]. As our data has the class



ratio of 1:10 for minority:majority classes, we decided not to include the accuracy as an evaluation metric for the prediction results.

**Receiver Operating Characteristic curve.** A receiver operating characteristic curve (ROC) is a graph showing the performance of a model of classifying True Positive Rate (TPR) and False Positive Rate (FPR) for optimal threshold. An ROC curve plots TPR vs FPR at different classification thresholds and points out the result based on the most optimal one [89].

**Testbed.** Our experiments were conducted using a Core I7 10700K processor running at 4.2 GHz, 32 GB DDR4 RAM, and Nvidia RTX 2080 TI GPU SLI. Additionally, Ubuntu 18.04 LTS was used and CUDA version 10.2.9.55 was used to accelerate the training process.

#### 4.4.2 Parameter Tuning

**Fully Connected Convolution Neural Network Model.** The main goal of this research is to apply a deep learning method and utilize the properties to improve upon the existing supervised Machine learning solution. In order to do it, first we trained a fully connected feed-forward convolution neural network and performed the classification based on the prediction score. Then we implemented a time series based neural network model of Autoencoder with LSTM nodes to use the output reconstruction property of the Autoencoder to classify the failures.

We trained the model using differently preprocessed data to assess the performance and obtain the best predictions on test data. Thus, we created different versions of the trained model and tested the new data. Our first training of the model was to use binary encoded data with a recursive feature elimination method and PCA for dimension reduction. Last but not least, we reduced the dimension of the one hot encoded data using PCA and trained the model. We trained the baseline model with different encoded and preprocessed data and observed the training loss in order to determine the best preprocessing techniques for the model and to improve predictions with a more robust solution. The model was trained with first 9 months of the 2019. Then we tested the model performance with the last 3 months of the 2019 and the first 6 months of the 2020 which was completely new data to the model.

The above Fig. 4.16 illustrated the train loss generated during the training of

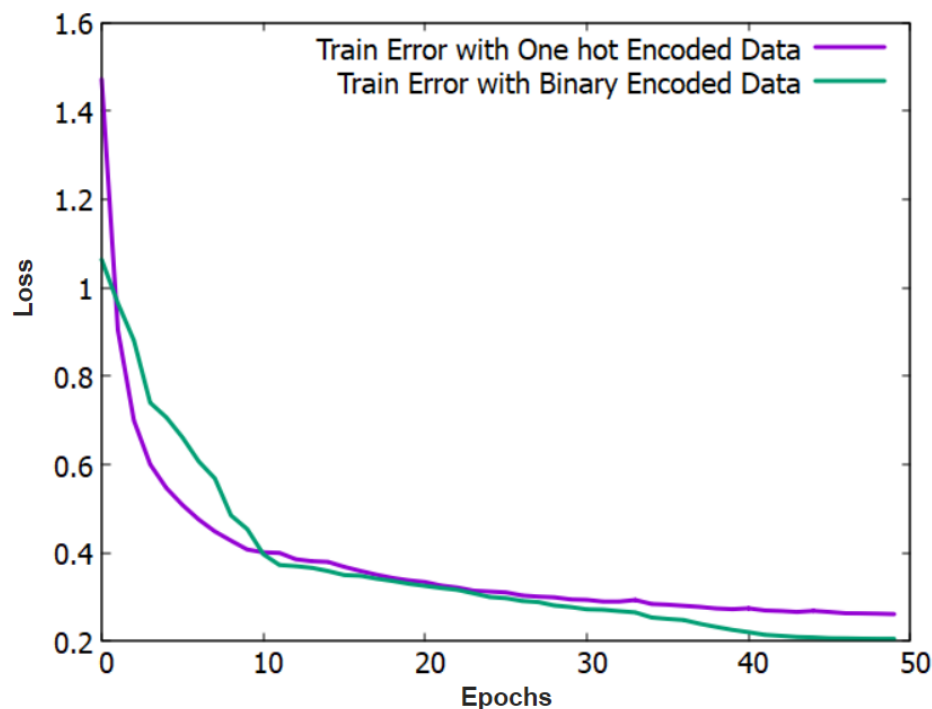


Figure 4.16: Train loss of FCNN for binary encoded and one hot encoded with PCA data.

the baseline model with both binary encoded data and the one hot encoded data. As can be seen from the plot, a one hot encoded data has a very good convergence to a solution while a binary encoded train shows some hiccups along the line. This supports the evidence of better training of the model with one hot encoded data. The claim was solidified with the classification results calculated from the prediction scores which was generated from both binary encoded data and one hot encoded data trained model.

**ROC Representation.** We used ROC method for representing the classification results. [89].

The ROC curve represented the classification of the test target variable. The model was trained and tested with different preprocessed data. The Figure 4.17 represents the plots for different sets of test data. PCA outperformed recursive elimination method with binary encoded data. By maintaining the total information or retained variance at 90%, we were able to improve model training performance which was not the case with the RFE method. Furthermore, we found that one-hot encoded

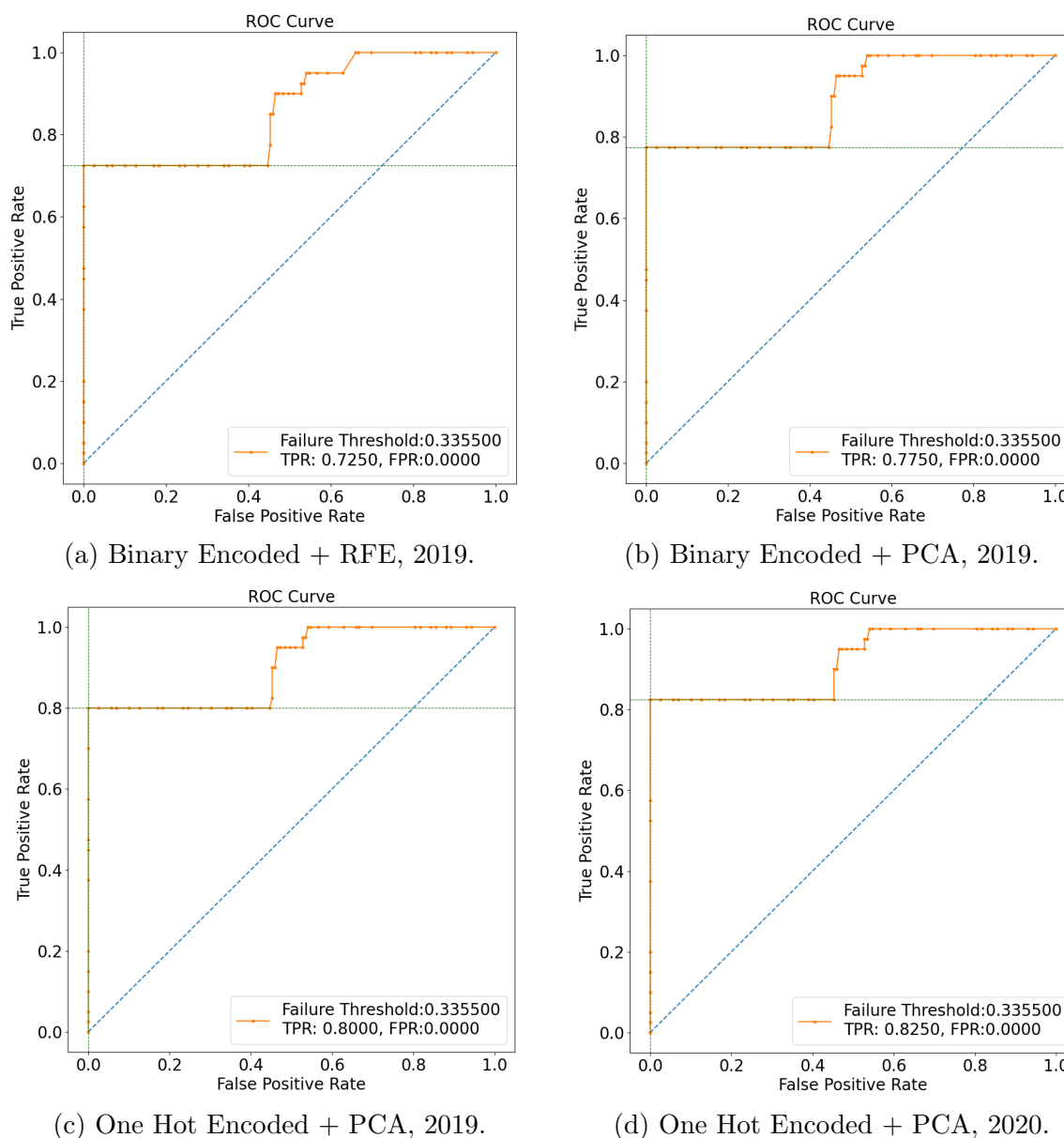


Figure 4.17: ROC curve generated from different parameters tuning.

and PCA combination in training performed better than both previously preprocessed data, indicating that one hot encoding method is better suited for this data due to its non-ranking encoding properties. The model outperformed for computing the prediction distribution using one hot encoding for both test data from 2019 and the new data from 2020, providing a clear indicator of the best encoding method selection based on the data. As a result of the 2020 test data, the model performed even better, with a higher recall, indicating a clear improvement in overall model accuracy for

predicting new data in the future. Nevertheless, the Recursive Feature Elimination method for dimensionality reduction performed significantly worse than the PCA in terms of generating accurate predictions, so we determined that the PCA with one hot encoded data was the best option for further experiments. The classification result of the fully connected convolutional Neural Network is given below.

- Data A = Binary encoded + RFE 2019
- Data B = Binary encoded + PCA applied data of 2019
- Data C = One Hot Encoded + PCA applied data of 2019
- Data D = One hot encoded + PCA applied Data of 2020

<b>Data Setting</b>	<b>Precision</b>	<b>Recall</b>	<b>F1 Score</b>
Data A	0.72	0.67	0.70
Data B	0.8	0.78	0.78
Data C	0.82	0.8	0.81
Data D	0.84	0.82	0.83

Table 4.5: Evaluation result for fully connected convolution model.

The precision score and F1 score of the one hot encoded 2019 test data is 2% higher than the previously state-of-the-art supervised ensemble learning model performance. We utilized the significant spatial properties presented in the data for the key point of training the FCNN. The ability of neural network to learn the feature importance better than the supervised ensemble model is one of the key reason of the result improvements. The supervised models struggles classifying binary classes when the testing sample is significantly less than the training data [90] such our data. The data preparation of our proposed approach helped the FCNN to detect and utilize the important features and resulted 4% better for new test samples than the existing solution [18]. Furthermore, the model performed better with one hot encoded data than binary encoded data. This affirmed the effectiveness of one hot encoding as a pre-processing hyper-parameter. Also, we tuned hyper-parameter such batch size, learning rate, training epoch in iterative way until the most optimal results were produced. This was absent from the existing state of the art solution which led to

produce poor classification score than our proposed model. The model performed well with the new test data of 2020 which also signifies the better training of the model.

**LSTM - Autoencoder.** We implemented the Autoencoder model with LSTM layers to perform and observe the classification of the failure link in 2019 data as well as the new 2020 data. At each epoch, the training process fed the LSTM-AE model with all training sample with normal label/normal links and validated with the validation set for each segments. Both reconstruction error of train and validation set converged after a certain point which determined the proper learning of the model. The training process trying to minimize the reconstruction loss for each epoch. The validation set reconstruction started lower and then eventually it crossed the training error as the failed link reconstruction error is higher in the validation set.

Tests were conducted on the performance of our proposed model in different ways. Several versions of the model were trained with different configurations varying the encoding techniques in preprocessing, number of hidden layers, number of past days were used in training of the model. These experiments allowed us to determine the optimal version of our approach, which we compared with a baseline Full Connected Network (FCN) and the existing state-of-the-art [18] solution.

*Encoding Methods.* As like our baseline FCNN model, We trained the model with binary encoded data and one hot encoded data. As a result of the ranking of the features in the binary encoding heuristic, the baseline model learned poorer with binary encoded data than the one hot encoded data. Our claim can be further justified by training our candidate proposed solution with both encoded data and observing the learning characteristics.

The Fig. 4.18 illustrates the reconstruction errors from training the model with different encoded data. One hot encoded data resulted in a smoother convergence of training and validation error. These results demonstrate better learning of the feature representation of one hot encoded data which is in line with the baseline solution. Better learning of the train data eventually improved the accuracy of predicting the failed link on the test data. our method involved iteratively selecting the most optimal threshold and categorizing the reconstruction errors for every sample in the last training epoch. Then, we determined the sample is a link failure if the generated error for the corresponding sample is higher than the optimal threshold. We calculated the

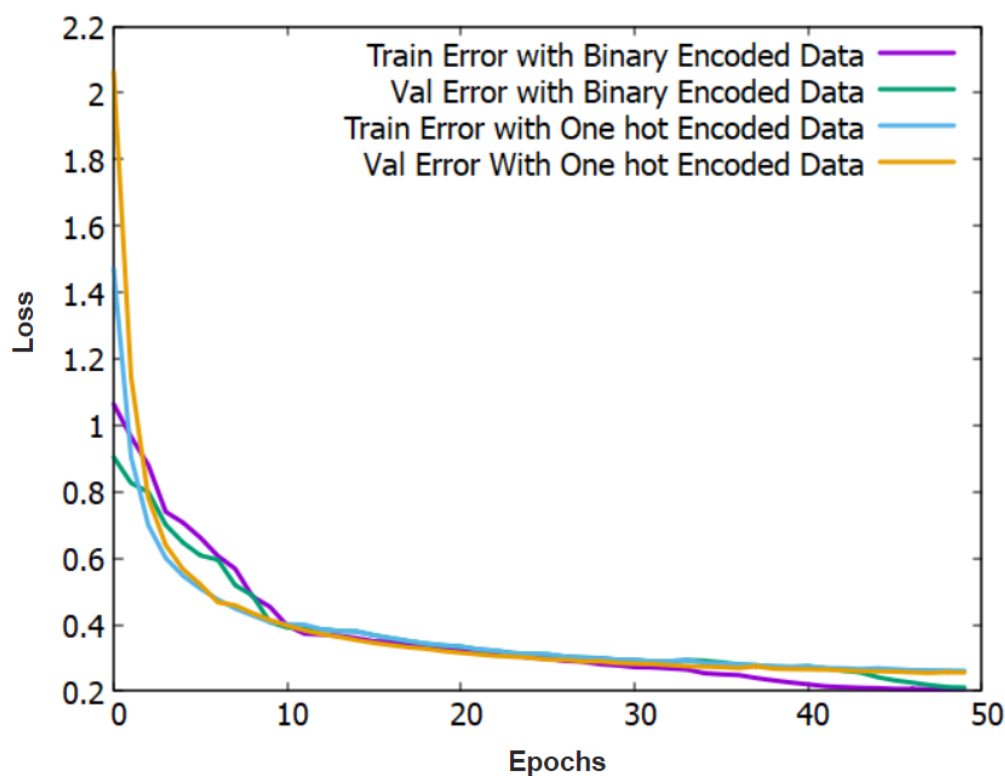


Figure 4.18: Reconstruction error of different encoding methods.

Accuracy, Precision, Recall and F1 score for the test data using the optimal threshold on the reconstruction error. We utilized this heuristic for every experiment for the proposed solution and recorded the prediction results on the test data. Both versions of the models trained with batch size 16 and one day past data lookback for LSTM units.

**The ROC curve representation.** Here we represented the ROC curve for the classification result. The model generated the best results for the one hot encoded data of 2019 and 2020. The model was capable of successfully distinguishing the true positive rate and the false positive rate. We generated ROC curve for each experiments. However, we are only representing one ROC plot containing the best result of one hot encoded data of 2019. The full classification results are given in Table 4.6.

We can see by the results showed in Table 4.6 that the performance of the model improved when using one-hot as the encoding technique. The positive effect can be explained as a result of the categorical features not having any type of rank between

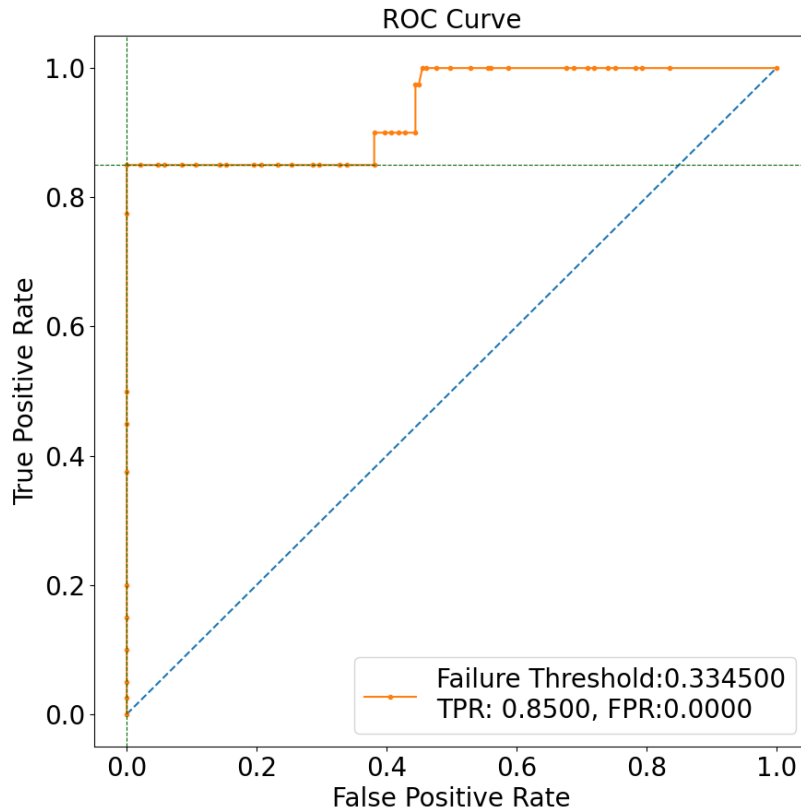


Figure 4.19: ROC curve representation for one hot encoded data with PCA of 2019.

Encoding technique	Precision	Recall	F1 Score
Binary (2019)	0.8	0.81	0.81
Binary (2020)	0.8	0.79	0.78
One-hot (2019)	<b>0.87</b>	<b>0.85</b>	<b>0.86</b>
One-hot (2020)	<b>0.87</b>	<b>0.85</b>	<b>0.86</b>

Table 4.6: Prediction results for different encoding techniques (PCA applied).

them. Besides, the fact that our approach uses PCA to pre-process the data ensures that it does not suffer from handling the higher dimensional data that is generated when using one-hot encoding. So, we selected the one hot encoding method as the preferred encoding method and continued next experiments with one hot encoded data.

*Number of Layers.* As part of training and experimentation with a variety of hidden layer numbers, we fine-tuned the structure of the proposed LSTM - Autoencoder

model. During this test, we used 2, 3, 4 layers with 384, 438, and 480 units respectively, and compared the resulting reconstruction error and classification score based on the errors. We used the one hot encoded data for conducting this experiment. This experiment enabled selection of the best version of the hyper-parameters for the proposed model based on knowledge of the optimal number of hidden layers.

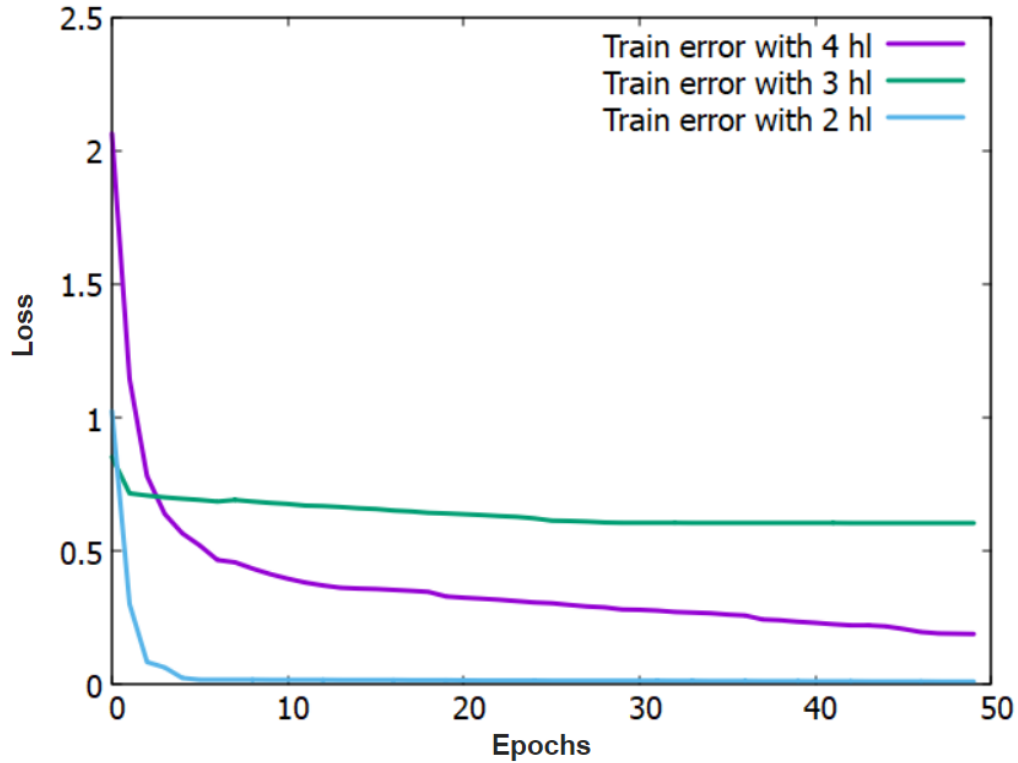


Figure 4.20: Training reconstruction error of different structures of LSTM-AE.

As shown in Fig. 4.20, data compression performance improved when more hidden layers were added. In this case, the model trained with four hidden layers performed better than the other configurations. The low number of hidden layers resulted an under-fitting model because of the high dimensional data and complex correlation between features. We calculated the optimal threshold for different settings in order to classify the reconstruction errors. The table 4.7 shows the classification results for failed links which represents the better performance with more layers. The results were generated for 2019 test data. The test results showed a significant impact for each settings. From the observations, we selected the 4 hidden layers model as the best version of the model due to the highest prediction scores calculated from this



version of the model. However, we also built and trained our model with 5 hidden layers and 556 units which resulted the model over-fitting. This added complexity of the model allowed the model to capture the details and noises of the data to the extent that it negatively impacted the overall performance of the model. Thus, we used only the 4 hidden layers version of the model with one hot encoded data training for next experiment.

**The ROC curve representation.** We represented the ROC curve for the classification result. The model generated the best results for the one hot encoded data of 2019 for 4 hidden layers architecture. The model was able to predict 87% of the true positive class. The ROC plot was generated for each experimentation. The ROC curve containing the best result is shown in Figure 4.21. The classification results are given in Table 4.7.

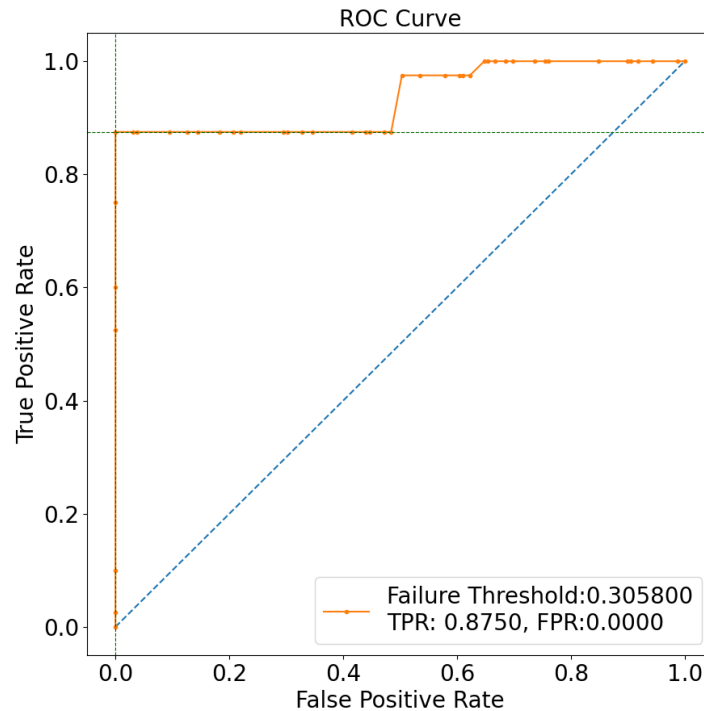


Figure 4.21: ROC curve representation for one hot encoded + PCA data of 2019 with 4 hidden layers for LSTM-AE.

*Usage of Past Sample Measurement.* One of the major differences between our approach and that of the state-of-the-art model is our approach used of temporal

Hidden layers	Precision	Recall	F1 Score
2	0.64	0.4	0.49
3	0.73	0.67	0.7
4	<b>0.90</b>	<b>0.87</b>	<b>0.88</b>

Table 4.7: Prediction results for different numbers of hidden layers.

properties, taking advantage of the LSTM units to understand this type of dependency better. Nevertheless, we were able to fine-tune how much historical information the model receives to reconstruct a sample. In this experiment, we varied the number of days in the past (lookback steps) that the LSTM units receive (1, 5 or 7) in order to forecast a failure. The main reason for using the different amount of historical data is that the weather effects might be different for older data, which may provide a better insight into the weather-radio link correlation and help predict the failure more accurately.

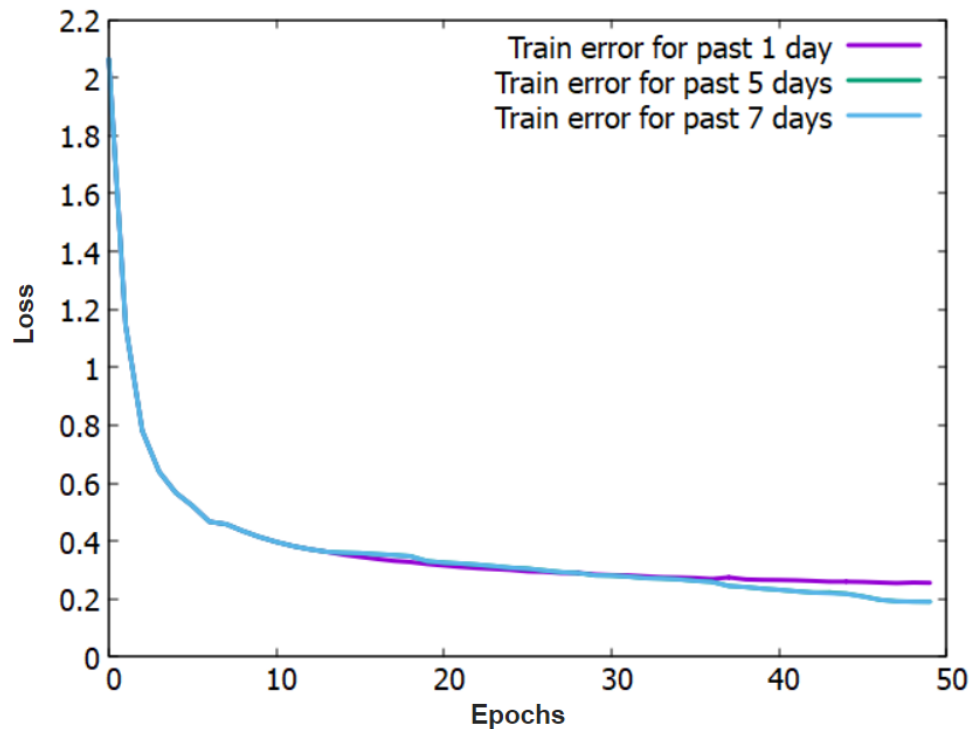


Figure 4.22: Reconstruction error for different numbers of lag steps.

This experiment was conducted using the model with four hidden layers and with one-hot encoded data, as this was the best version of the model according to previous results. The Fig. 4.22 shows the reconstruction error generated from different past

data observations. We can see that increasing the number of lookback steps from one day to five days lead to a improved performance in prediction, which can be seen in Table 4.8 as well. However, that the performance stayed almost the same when we increased the duration from five days to seven days past data observation. We attribute that to the fact that these extra two days are further away in the past end up not having any significant impact in the the day we intended to forecast. Due to this, we determined that five lookback steps would be the optimal configuration for the model, since it provides better results than a single step while using less data than a seven-day time lag. According to previous experiments and observations, we determined the best version of the model is trained and tested using one-hot encoded data, built with four hidden layers, and observe five days past data to produce the best prediction result.

**The ROC curve representation.** We represented the ROC curve for the classification result. The model generated the best results for the one hot encoded with PCA applied data of 2019 for 4 hidden layers architecture and five past days lookback parameters. The model was able to predict 88% of the true positive class. However, we generated the ROC curve for each past date observations (1, 5 and 7). Here we presented the ROC curve containing the best results. The full classification results are given in Table 4.8.

Test Data	Past Days used	Precision	Recall	F1 Score
2019	1	0.87	0.85	0.86
	5	<b>0.90</b>	<b>0.87</b>	<b>0.88</b>
	7	0.90	0.86	0.88
2020	1	0.87	0.85	0.86
	5	0.90	0.86	0.88
	7	0.90	0.86	0.88

Table 4.8: Prediction results for different amounts of days in the past data being used with one hot encoded + PCA and 4 hidden layers of the model.

**State-of-the-art Supervised Model.** We also recreated the current state-of-the-art solution to compare it with our proposal. The existing solution [18] used the ensemble learning method of the supervised models. The ensemble model was trained and tested with different variations of the preprocessed data to evaluate the model performance. The evaluation procedure is given below.

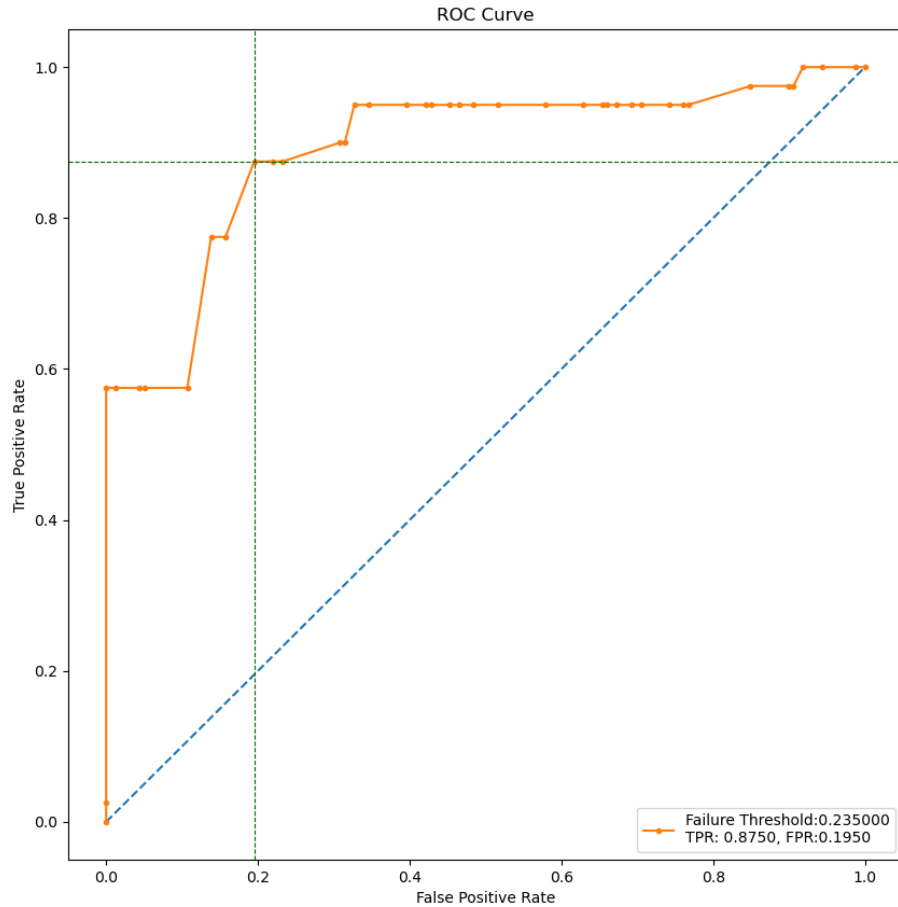


Figure 4.23: ROC curve representation for one hot encoded data of 2019 with 4 hidden layers and five days past data observation.

1. The ensemble model was built with decision-tree based supervised models. Random forest, LGBM and XBM were trained in a pipeline where the final results were improved by utilizing LGBM and XGB boosting algorithm.
2. Trained the model with the one hot encoded with PCA applied, default segmented data (70:30) with applying data rebalancing technique SMOTE.
3. First, we tested the model with the last three months of 2019. Then, we used the new data from 2020 as the test data and calculated the prediction scores for each samples.

We calculated the prediction score of each sample of the target label *Rlf* of the train set and tested the model performance on the test set of 2019 as well as the new data from 2020 with same preprocessing methods. We applied our proposed heuristic

of optimal distance calculation and trained ensemble supervised models due to the lack of information provided by the existing solution regarding distance correlation calculation. We found no significant differences between the F1 score (0.77) of the test result for our proposed distance correlation heuristic and the F1 score(0.78) published for the state-of-the-art NEC solution [18], which supports the effectiveness of the proposed optimal distance heuristic. The predicted class probabilities of an input sample were computed as the softmax of the weighted terminal leaves from the decision tree ensemble corresponding to the provided sample. The average predicted probability for the test data was calculated from the individual model prediction. A fixed threshold (0.5) and an optimal threshold (0.58) was applied on the predicted probabilities to perform the binary classification. The optimal threshold was selected iteratively where maximum performance of the failure prediction was achieved. Final prediction results were calculated from the probability scores derived from the test data.

- Data 1 = Test data of 2019 with fixed threshold
- Data 2 = Test data of 2019 with optimal threshold
- Data 3 = Test data of 2020 with optimal threshold
- Data 4 = Test data of 2020 with fixed threshold

<b>Data Setting</b>	<b>Precision</b>	<b>Recall</b>	<b>F1 Score</b>
Data 1	0.70	0.84	0.76
Data 2	0.79	0.77	0.77
Data 3	0.76	0.84	0.79
Data 4	0.66	0.82	0.73

Table 4.9: Evaluation result for supervised Ensemble Model.

The evaluation results are shown in Table 4.9. The ensemble model was trained with the first 9 months of the 2019 and tested with the last 3 months of 2019. We trained and tested with both imbalanced data and the balanced data. The decision tree based models performed better with test data for balanced training. In addition to classifying with fixed thresholds of prediction scores, the optimal threshold was

also selected iteratively. We found a clear maximum accuracy between 0.4 and 0.6 from the generated prediction scores. The exact value of the optimal threshold is 0.5802 which is calculated by using a library called `np.argmax(accuracy)`. A major improvement over the published state-of-the-art solution is that the optimal threshold allowed better prediction of failed links than the previously selected fixed threshold. However, due to the lack of consideration of temporal dependencies of the features in the state-of-the-art solution, we used deep learning neural networks to improve prediction.

#### 4.4.3 Model Comparisons

We used the spatial properties of the features for a convolution neural network to calculate the prediction scores for the target feature RLF. In utilizing the neural network for the most optimal latent features. We experimented with different model hyper-parameter tuning and recorded the best prediction result with one hot encoded data. Then, We proposed a more sophisticated model of Autoencoder with LSTM units considering the temporal dependency and the aforementioned spatial properties both. The proposed LSTM - Autoencoder model provided us with more control over the training of the model with time series data. Through analyzing the temporal dependencies between the features, we conducted multiple experiments on the model training in order to maximize prediction accuracy. We experimented with different hyper-parameter tuning and recorded the best prediction score with one hot encoded data five days past data time lag and built with four hidden layers. Then, we implemented the existing state of the art solution [18] and experimented with different hyper-parameter tuning to recreate and ultimately improve the prediction performance. We used the optimal threshold instead of a fixed threshold to improve the prediction score calculated by the ensemble model and recorded the best prediction result with the balanced and optimal threshold setup. In summary, baseline model FCNN produced the best predictions when training with one-hot encoded data and candidate model LSTM-AE produced the best predictions by using one-hot encoded data, five-day history observation, and four hidden layers. Ensemble model produced the best prediction scores when training with balanced data and using optimal threshold. We compared the best versions of each model from previous experiments

based on prediction scores. We have mentioned the prediction results for 2019 test data in the previous parameter tuning section. Here, Table 4.10 represents the final prediction results for 2020 data.

Method used	Precision	Recall	F1 Score
Baseline model FCNN	0.84	0.82	0.83
Supervised ensemble model	0.76	0.84	0.79
Proposed model LSTM-AE	<b>0.90</b>	<b>0.87</b>	<b>0.88</b>

Table 4.10: Comparison of prediction results with state of the art approach.

Analyzing Table 4.10, we find that the LSTM-Autoencoder approach is able to identify **87%** of the failed links among all of the failed links and false normal links, where **90%** of the identified failed links matches with the original failed links ( True positive) from the test data. Our proposed solution achieved higher precision of **14%** and recall of **3%** and the F1 score of **11%** over the previous method (NEC) [18]. We attribute this improvement to the strategic use of the temporal correlations of the features. By capturing the temporal dependencies of radio stations and weather stations data, our model is able to predict future radio link failure events more accurately. Table 4.10 represents that the proposed LSTM - AE model outperformed other solutions in predicting the one day ahead radio link failure.

The Autoencoder model performs better than the previous FCNN method and the existing supervised ensemble solution. The precision, recall and F1 scores are higher in the Autoencoder model. The proposed neural network model was successful to recognise the important feature patterns and able to classify the failure data from the normal data in both 2019 and 2020 time span. The LSTM - AE model was trained and validated with the time series cross validated data where the lookback window was set for the past data. It helped the encoder to compress the current sample to latent space by observing the past data using the LSTM nodes. Thus, the training quality achieved improvements than other supervised or baseline neural network solution.

The scaling of the data helped the activation functions to optimize the neurons for better reconstruction of the input. There are two feature selection methods are applied on the data and trained the model with different setting to observe the model performance. The recursive elimination method and principal components analysis

methods are applied and tested the performance difference for different dimensionality reduction methods. The PCA as a dimensionality reduction operation performed better than the recursive elimination method. This performance difference happened due to the maximum information kept in the dataset during dimensionality reduction by PCA. Our proposed pre-processing and model solution based on the temporal dependency of the feature set achieved highly accurate prediction results for the one day ahead link failure, which is a major contribution of this work.



## Chapter 5

### Conclusion and Future work

#### 5.1 Conclusion

5G Radio link data from radio tower can often be compromised because of the weather impacts in different regions. In contrast, uninterrupted communication will propel the Internet of Things into the next era of communication. Predicting and preventing failures to ensure this seamless operation will be one of the major objectives for the network operators to provide the desired quality of service.

In this work, we highlighted the limitation of the exiting state-of-the-art solution and discussed the optimal solutions. We proposed and implemented the preprocessing strategy and utilized the reconstruction property of the deep learning algorithm Autoencoder coupled with Long-Short Term Memory neurons. We experimented with tuning different hyper-parameters to achieve the best version of the model which can be scaled and implemented to a real environment. Our solution outperformed the existing solution by utilizing the temporal dependency of the features in terms of the prediction accuracy. It is observed that, the feature recognition ability of our model can be utilized to predict different year data for any region for the similar network setting.

#### 5.2 Future Work

In our proposed model, we have performed the radio link failure prediction for one day ahead of the current date. First, we plan to extend the prediction span to the next five days, instead of one. Then we plan to work on data preparation such that the learning model will be able to predict radio link failure based on all the forecast data available in the future. Furthermore, we have currently applied synthetic minority oversampling technique to mitigate the majority and minority ratio for the current solution. Now, we are developing a Balancing Generative Adversarial Network model

where we will train the model with only minority class for creating more efficient data balancing for the original imbalanced data. The plan is to prepare more balanced data for our proposed model training so that it will perform better for the prediction with new data. To test the performance of our feature recognition property, we are planning to train our model using radio data with different features from various regions.

## Bibliography

- [1] J. Navarro-Ortiz, P. Romero-Diaz, P. A. S. Sendra, J. J. Ramos-Munoz, and J. M. Lopez-Soler, "A survey on 5g usage scenarios and traffic models," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, 2020.
- [2] T. Fevens, I. Haque, and L. Narayanan, "Randomized routing algorithms in mobile ad hoc networks," in *Proceedings of the IFIP International Conference on Mobile and Wireless Communication Networks*, 2004.
- [3] I. Haque, "Non-deterministic geographic forwarding in mobile ad hoc networks," in *IEEE Asia-Pacific Services Computing Conference*, 2008, pp. 1144–1149.
- [4] I. Haque, I. Nikolaidis, and P. Gburzynski, "On the benefits of nondeterminism in location-based forwarding," in *IEEE International Conference on Communications (ICC)*, 2009.
- [5] V. Kolar, I. T. Haque, V. P. Munishwar, and N. B. Abu-Ghazaleh, "Ctvc: Coordinated transport of correlated videos in smart camera networks," in *24th International Conference on Network Protocols (ICNP)*. IEEE, 2016.
- [6] I. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: a survey and taxonomy," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 4, pp. 2713–2737, May 2016.
- [7] M. Kulkarni, M. Baddeley, and I. Haque, "Embedded vs. external controllers in software-defined iot networks," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, 2021.
- [8] "Millimeter wave opens new opportunities for 5g networks," <https://www.globenewswire.com/en/news-release/2020/12/02/2138610/0/en/Millimeter-Wave-Opens-New-Opportunities-for-5G-Networks.html>, December 2020.
- [9] Alexander, "What is 5g network," <https://www.techtarget.com/searchnetworking/definition/5G>.
- [10] JOUR, Nadeem, F.Chessa, S. Leitgeb, E. Zaman, and Safdar, "The effects of weather on the life time of wireless sensor networks using fso/rf communication," in *Radioengineering*, June 2010.
- [11] J. Abdul, "Survivable millimeter-wave mesh networks," *Computer Communications*, vol. 34, no. 16, pp. 1942–1955, 2011.
- [12] NEC, "Itu-challenge p.036," 2020. [Online]. Available: <https://aiforgood.itu.int/event/itu-ai-ml-in-5g-challenge-radio-link-failure-prediction-challenge/>

- [13] Esposito and Christian, “On the disaster resiliency within the context of 5g networks: The recodis experience.” in *European Conference on Antennas and Propagation (EuCAP)*, 2016.
- [14] Drozdy, Árpád, P. Kántor, and J. Bitó, “Effects of rain fading in 5g millimeter wavelength mesh networks,” in *Computer Communications*, June 2018.
- [15] S. Elsayed, “Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management,” in *International Journal of Information Management*, 2021.
- [16] R. G. Karjee, Jyotirmoy and V. Tijoriwala, “A reinforcement learning approach to handle radio link failure in elevator scenario,” in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, 2020.
- [17] Ranjan, “Lstm-autoencoder for extreme rare event classification in keras,” <https://towardsdatascience.com/lstm-autoencoder-for-extreme-rare-event-classification-in-keras-ce209a224cfb>.
- [18] LinkBusters, “Ensemble solution,” 2020. [Online]. Available: [https://www.itu.int/en/ITU-T/AI/challenge/2020/Documents/ITUChallenge\\_LinkBusters.pdf](https://www.itu.int/en/ITU-T/AI/challenge/2020/Documents/ITUChallenge_LinkBusters.pdf)
- [19] Bhagat, “Enhanced smote algorithm for classification of imbalanced big-data using random forest,” in *IEEE International Advance Computing Conference*, 2015.
- [20] “The thesis code repository.” [Online]. Available: [https://github.com/ArifulIslamPreence/RLF\\_Prediction\\_Main\\_Thesis](https://github.com/ArifulIslamPreence/RLF_Prediction_Main_Thesis)
- [21] B. Roy, “Binary encoding,” 2019. [Online]. Available: <https://towardsdatascience.com/all-about-categorical-variable-encoding-305f3361fd02>
- [22] Jie and L. I. A. N. G, “One-hot encoding and convolutional neural network based anomaly detection.” *Journal of Tsinghua University (Science and Technology)*, vol. 8, no. 2, 2011.
- [23] Li and Jia, “Deep convolutional neural network based ecg classification system using information fusion and one-hot encoding techniques.” *Mathematical Problems in Engineering*, vol. 6, no. 3, 2018.
- [24] C. Seger, “An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing,” 2018.
- [25] Chandrasheka, “A survey on feature selection methods.” *Computers & Electrical Engineering*, vol. 40, no. 1, 2014.
- [26] Zhang, “Integrating feature selection and feature extraction methods with deep learning to predict clinical outcome of breast cancer.” *IEEE Access*, no. 6, 2018.

- [27] Wold, Svante, K. Esbensen, and P. Geladi, "Principal component analysis," in *Chemometrics and intelligent laboratory systems*, 2020.
- [28] L. Geladi, Paul and Johan, "Principal component analysis," 2020.
- [29] XIE and XIAOHUI, "Principal component analysis," in *Principal component analysis*, 2019.
- [30] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.
- [31] Dietterich, "Ensemble learning." *The handbook of brain theory and neural networks*, vol. 2, no. 1, 2002.
- [32] —, "Random forest classifier for remote sensing classification." in *International journal of remote sensing.*, vol. 26, no. 1, 2005.
- [33] Alam, "Random forest classification for detecting android malware," in *IEEE international conference on green computing and communications and IEEE Internet of Things and IEEE cyber, physical and social computing*, 2013.
- [34] Documentation, "Light gradient boosting," <https://lightgbm.readthedocs.io/en/latest/>.
- [35] "Light gradient boosting machine," <https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/>, September 2021.
- [36] Saul, "Light gradient boosting machine," <https://towardsdatascience.com/xgboost-extreme-gradient-boosting-how-to-improve-on-regular-gradient-boosting-5c6acf66c70a>, March 2021.
- [37] "Extreme gradient boosting machine," <https://xgboost.readthedocs.io/en/latest/>, 2021.
- [38] Zhao, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, 2017.
- [39] Borovykh, "Conditional time series forecasting with convolutional neural networks." *arXiv preprint*, vol. 1703, no. 4691, 2017.
- [40] Breuel, "Benchmarking of lstm networks." *arXiv preprint*, vol. 1508, 2015.
- [41] S. Elsayed, "Network anomaly detection using lstm based autoencoder," in *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2020.
- [42] N. K. Manaswi, "Rnn and lstm," in *Deep Learning with Applications Using Python*. Springer, 2018, pp. 115–126.

- [43] “RI kpi + ws data,” 2020. [Online]. Available: <https://docs.google.com/spreadsheets/d/1KvFV5WXtRo9dU3Yak40b2G7JG8KCIqrg/edit?usp=sharing&oid=105740223973240568967&rtpof=true&sd=true>
- [44] M. Shojaee, M. C. Neves, and I. Haque, “Safeguard: Congestion and memory-aware failure recovery in SD-WAN,” in *16th International Conference on Network and Service Management, CNSM 2020, Izmir, Turkey, November 2-6, 2020*. IEEE, 2020, pp. 1–7. [Online]. Available: <https://doi.org/10.23919/CNSM50824.2020.9269119>
- [45] M. A. Moyeen, F. Tang, D. Saha, and I. Haque, “SD-FAST: A packet rerouting architecture in SDN,” in *15th International Conference on Network and Service Management, CNSM 2019, Halifax, NS, Canada, October 21-25, 2019*. IEEE, 2019, pp. 1–7. [Online]. Available: <https://doi.org/10.23919/CNSM46954.2019.9012703>
- [46] U. Lekhala and I. Haque, “Piqos: A programmable and intelligent qos framework,” in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2019, Paris, France, April 29 - May 2, 2019*. IEEE, 2019, pp. 234–239. [Online]. Available: <https://doi.org/10.1109/INFCOMW.2019.8845158>
- [47] F. Tang and I. Haque, “Remon: A resilient flow monitoring framework,” in *Network Traffic Measurement and Analysis Conference, TMA 2019, Paris, France, June 19-21, 2019*. IEEE, 2019, pp. 137–144. [Online]. Available: <https://doi.org/10.23919/TMA.2019.8784521>
- [48] I. Haque and M. A. Moyeen, “Revive: A reliable software defined data plane failure recovery scheme,” in *14th International Conference on Network and Service Management, CNSM 2018, Rome, Italy, November 5-9, 2018*, S. Salsano, R. Riggio, T. Ahmed, T. Samak, and C. R. P. dos Santos, Eds. IEEE Computer Society, 2018, pp. 268–274. [Online]. Available: <https://ieeexplore.ieee.org/document/8584938>
- [49] I. Haque and D. Saha, “SoftIoT: A resource-aware sdn/nfv-based iot network,” *The Elsevier Journal of Network and Computer Applications*, vol. 193, Nov 2021.
- [50] D. Saha, M. Shojaee, M. Baddeley, and I. Haque, “An Energy-Aware SDN/NFV architecture for the internet of things,” in *IFIP Networking 2020 Conference (IFIP Networking 2020)*, Paris, France, Jun. 2020.
- [51] I. Haque, M. Nurujjaman, J. Harms, and N. Abu-ghazaleh, “SDSense: An agile and flexible SDN-based framework for wireless sensor networks,” *The IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1866 – 1876, February 2019.

- [52] I. Haque, S. Islam, and J. Harms, "On selecting a reliable topology in wireless sensor networks," in *Proceedings of the 2015 IEEE International Conference on Communications*, ser. ICC '15, 2015.
- [53] I. Haque, C. Assi, and W. Atwood, "Randomized energy-aware routing algorithms in mobile ad hoc networks," in *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, ser. MSWiM '05, 2005.
- [54] I. Haque and C. Assi, "OLEAR: Optimal localized energy aware routing in mobile ad hoc networks," in *Proceedings of the 2005 IEEE International Conference on Communications*, ser. ICC '05, 2005.
- [55] I. T. Haque and C. Assi, "Localized energy efficient routing in mobile ad hoc networks," *The Willey Journal of Wireless and Mobile Computing*, vol. 7, no. 6, pp. 781–793, August 2007.
- [56] N. Javed, E. Lyons, M. Zink, and T, "Adaptive wireless mesh networks: Surviving weather without sensing it," *22nd International Conference on Computer Communication and Networks*, pp. 1–7, 2013.
- [57] R. J, "A new approach to design of weather disruption-tolerant wireless mesh networks," *Telecommun Syst*, vol. 61, 2016.
- [58] J. P. Rohrer, A. Oberthaler, E. K. Cetinkaya, V. Frost, and J. P. G. Sterbenz, "Performance comparison of weather disruption-tolerant cross-layer routing algorithms," *IEEE INFOCOM*, vol. 1703, no. 4691, 2009.
- [59] M. Tornatore, "A survey on network resiliency methodologies against weather-based disruptions," *8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2016.
- [60] P. H. Swain and H. Hauska, "The decision tree classifier: Design and potential," *IEEE Transactions on Geoscience Electronics*, vol. 15, no. 3, pp. 142–147, 1977.
- [61] Kliger, "A coding approach to event correlation," in *International Symposium on Integrated Network Management*. Springer, 1995.
- [62] Wietgreffe, "Using neural networks for alarm correlation in cellular phone networks," in *International Workshop on Applications of Neural Networks to Telecommunications (IWANNNT)*, 1997.
- [63] Weitgreffe, "Supervised and semi-supervised learning for failure identification in microwave networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, 2020.
- [64] Vanerio, "Machine-learning based approaches for anomaly detection and classification in cellular networks," in *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, 2017.

- [65] C. N. M. Furdek, “Experimentbased detection of service disruption attacks in optical networks using data analytics and unsupervised learning,” *Metro and Data Center Optical Networks and Short-Reach Links II and International Society for Optics and Photonics.*, vol. 10946, 2019.
- [66] X. Chen and B. Li, “Self-taught anomaly detection with hybrid unsupervised/supervised machine learning in optical networks,” *Journal of Lightwave Technology*, vol. 37, no. 7, 2019.
- [67] Chen, “Forecasting crime using the arima model.” *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery.*, vol. 5, 2008.
- [68] Kalekar, “Time series forecasting using holt-winters exponential smoothing,” *Kanwal Rekhi school of information Technology*, vol. 4329008, no. 13, 2004.
- [69] “Applied ml failure prediction project by iec,” [https://github.com/ITU-AI-ML-in-5G-Challenge/PS-036.IEC\\_Research\\_RLFPrediction](https://github.com/ITU-AI-ML-in-5G-Challenge/PS-036.IEC_Research_RLFPrediction), December 2020.
- [70] Kahn, “Transparent reporting of data quality in distributed data networks.” *Egems*, vol. 3, no. 1, 2015.
- [71] Kumar, “Python – replace missing values with mean, median and mode,” 2021. [Online]. Available: <https://vitalflux.com/pandas-impute-missing-values-mean-median-mode/>
- [72] Baweja, “how to deal with missing data in python,” <https://towardsdatascience.com/how-to-deal-with-missing-data-in-python-1f74a9112d93>.
- [73] L. Li, “Principal component analysis for dimensionality reduction,” 2019. [Online]. Available: <https://towardsdatascience.com/principal-component-analysis-for-dimensionality-reduction-115a3d157bad>
- [74] H. Shen and J. Z. Huang, “Sparse principal component analysis via regularized low rank matrix approximation,” *Journal of multivariate analysis*, vol. 99, no. 6, pp. 1015–1034, 2008.
- [75] S. Lee, *Principal components analysis for binary data*. Texas A&M University, 2009.
- [76] S. Kolenikov, G. Angeles *et al.*, “The use of discrete data in pca: theory, simulations, and applications to socioeconomic indices,” *Chapel Hill: Carolina Population Center, University of North Carolina*, vol. 20, pp. 1–59, 2004.
- [77] Documentation, “Train test split,” [https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.train\\_test\\_split.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html).
- [78] Bergmeir, “On the use of cross-validation for time series predictor evaluation.” *Information Sciences*, vol. 191, (2012).



- [79] Herman, “How to use convolutional neural networks for time series classification,” <https://towardsdatascience.com/how-to-use-convolutional-neural-networks-for-time-series-classification-56b1b0a07a57>.
- [80] G. link of RLF project, “The time series cross validation code repository.” [Online]. Available: <https://gist.github.com/orhermansaffar/2bd2342c81026de1c09c97d66226eb46>
- [81] K. Miyaki, “Time series split with scikit-learn,” 2019. [Online]. Available: <https://medium.com/keita-starts-data-science/time-series-split-with-scikit-learn-74f5be38489e>
- [82] G. Ranjan, A. K. Verma, and S. Radhika, “K-nearest neighbors and grid search cv based real time fault monitoring system for industries,” in *2019 IEEE 5th international conference for convergence in technology (I2CT)*. IEEE, 2019, pp. 1–5.
- [83] Qian, “Dynamic multi-scale convolutional neural network for time series classification,” *IEEE Access*, vol. 8, 2020.
- [84] Z. Cui, W. Chen, and Y. Chen, “Multi-scale convolutional neural networks for time series classification,” *arXiv preprint arXiv:1603.06995*, 2016.
- [85] Zhao, “Time-series-approach with fully connected neural network,” <https://towardsdatascience.com/time-series-classification-with-deep-learning-d238f0147d6f>.
- [86] “Random forest.” [Online]. Available: [https://www.stat.berkeley.edu/~breiman/RandomForests/cc\\_home.htm](https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm)
- [87] K. RAJ, “Ml classification-why accuracy is not a best measure for assessing,” 2020. [Online]. Available: <https://medium.com/@KrishnaRajParthasarathy/ml-classification-why-accuracy-is-not-a-best-measure-for-assessing-ceeb964ae47c>
- [88] Jake, “Classification evaluation: It is important to understand both what a classification metric expresses and what it hides,” in *Nature Methods*, Aug. 2016.
- [89] Qian, “Time-dependent roc curve analysis in medical research: current methods and applications.” *BMC medical research methodology*, vol. 17, no. 1, 2017.
- [90] Sambit, “why-deep-learning-is-needed-over-traditional-machine-learning.” [Online]. Available: <https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063>