POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Machine-Learning-Assisted Failure Prediction in Microwave Networks based on Equipment Alarms

Master Thesis in
Telecommunication Engineering - Ingegneria delle Telecomunicazioni

Author: **Francesco Lateano**

Student ID: 913035
Advisor: Prof. Francesco Musumeci
Co-advisors: Prof. Massimo Tornatore, Dr. Omran Ayoub
Academic Year: 2020-21

Acknowledgment

I would like to thank Prof. Massimo Tornatore to give me the opportunity to work with him, and to trust me and my abilities, allowing me to work to this thesis project. A special thanks go to Dr. Omran Ayoub and Prof. Francesco Musumeci, that guided me in the project path, helped me in writing my thesis and were a big source of positiveness and motivation. Work with all of them was an honour and a great opportunity for growth.

A thank goes to my friends to always support and bear me. Thanks also to my course-mates that shared with me this learning path.

A special thanks to my family that always believes in me, supports me and made this path possible.

Abstract

Communication services on microwave networks are designed to cope with strict Quality of Services (QoS) requirements such as low latency and high bandwidth. As failures can affect the network availability, failure forecast, detection and identification are crucial for the service maintenance and must be executed in short time. In our work, we study new solutions for failure management on microwave networks. Starting with monitored measurement data collected from network equipment, in particular alarms, we first perform some preparatory data wrangling, then we mainly focus on the failure prediction problem, composed of three steps: alarms forecasting, failure detection and fault identification.

Nowadays microwave fault management is mainly handled reactively, waiting for the failure event, and only then restoring service. In this study, we are proposing a modular framework to handle faults in a proactive way, where the occurrence of a failure could be detected and managed in advance, potentially avoiding the actual failure event. Today alarms forecasting is not performed, while the failure detection and fault identification procedure is carried out by human experts, who typically analyse radio-power measures and equipment alarms related to the failure event and, based on their filed experience, identify possible root causes of the failure and proper countermeasures to restore the service. Usually, the amount of data related to the failure events that must be analysed is huge, and new ultra reliable low latency communications imposes a stringent time constraints to restore the service after a failure. This constraints can be satisfied using data-analysis techniques that allow to handle huge quantity of data in short time. In our work, we opted for techniques from machine learning discipline.

Our proposed workflow, based on a modular implementation, takes as input a set of the available highly unbalanced data. First it performs data pre-processing based on human experience, then alarms are forecasted with a deep learning model for time-series prediction. Based on these alarm forecasts, our framework predicts if there will be failure or not using ensemble classification methods, and, eventually ensemble classification ML models are used to predict also the failure root-cause.

Finally, we propose an alternative single-step approach - taking as input alarms statistics

and leveraging on classification techniques - with lower specificity on alarms forecasting and more on fault root-cause prediction based on previously defined knowledge, enabling farther prediction horizon.

Numerical experiments on real data from SIAE Microelectronics have shown very promising results, with performance metrics as accuracy, precision and recall above 95% and execution time a lot lower than the equivalent hand-made processing.

In conclusion a trade-off between the two proposed approaches must be considered: the first, short-term, more accurate but with computational constraint limiting prediction horizon and the latter, long-term, able to achieve longer predictions though losing accuracy and the ability of short-term predictions.

Abstract in lingua italiana

I servizi di comunicazione su reti millimetriche sono progettati per far fronte ai requisiti di alta qualità dei servizi (QoS) come bassa latenza e ampia larghezza di banda. Poiché la presenza di guasti può influire sulla disponibilità della rete, la predizione degli allarmi, il rilevamento dei malfunzionamenti e l'identificazione della causa è cruciale per la manutenzione del servizio e deve essere eseguita in tempi brevi. Nel nostro lavoro, studiamo la gestione dei guasti nelle reti a onde millimetriche. Partendo con le misure monitorate dagli apparati di rete, in particolare lo storico degli allarmi, prima eseguiamo alcuni processi di trasformazione sui dati, poi ci concentriamo principalmente sul problema della previsione dei guasti, composto da tre fasi: previsione degli allarmi, rilevamento dei malfunzionamenti e identificazione delle cause.

Oggi la gestione dei guasti delle onde millimetriche viene gestita principalmente in modo reattivo, attendendo l'evento di malfunzionamento e solo allora ripristinando il servizio. In questo studio, stiamo proponendo un framework modulare per gestire i guasti in modo proattivo, dove l'avvenimento di malfunzionamento può essere rilevato e gestito in anticipo, evitando potenzialmente l'effettivo guasto. Oggi la previsione degli allarmi non viene eseguita, mentre la procedura di rilevamento e identificazione del guasto è svolta da esperti umani, che tipicamente analizzano le misure di potenza radio e gli allarmi delle apparecchiature relativi all'evento di guasto, che sulla base dell'esperienza, identificano le possibili cause alla causa del guasto e le contromisure adeguate per ripristinare il servizio. Solitamente, la quantità di dati relativi agli eventi di guasto che deve essere analizzata è enorme, mentre il nuovo schema di comunicazione ultra affidabile a bassa latenza impone vincoli di tempo stringenti per ripristinare il servizio dopo un guasto. Questi vincoli possono essere soddisfatti utilizzando tecniche di analisi dei dati che permettono la manipolazione di enormi quantità di dati in breve tempo. Nel nostro lavoro abbiamo optato per tecniche della disciplina del machine learning.

Il flusso di lavoro da noi proposto, basato su un'implementazione modulare, elastica e flessibile, prende come input un insieme dei dati disponibili altamente sbilanciati. Prima esegue la preparazione dei dati, quindi avviene la previsione degli allarmi con il modello di deep learning per la predizione di serie temporali. Basandoci su queste predizioni di

allarmi, la nostra struttura predice se ci saranno o meno malfunzionamenti utilizzando metodi di classificazione d'insieme del campo machine learning e, infine, i modelli vengono utilizzati per predirre la causa del malfunzionamento.

Infine viene presentato un flusso di lavoro alternativo a singolo passo - prendendo come dati iniziali le statistiche degli allarmi e facendo leva sulle tecniche di classificazione - con una specificità minore sulla predizione degli allarmi, quanto più sulla causa di malfunzionamento basata sulla conoscenza precedente e abilitando una maggiore visione nel futuro.

Gli esperimenti numerici su dati reali forniti da SIAE Microelectronics hanno mostrato risultati molto promettenti, con metriche prestazionali come accuracy, precision e recall oltre il 95% e tempi di esecuzione molto inferiori al rispettivo processo manuale.

In conclusione, un compromesso va considerato tra i due approcci: il primo a corto raggio, più accurato ma con un limite computazionale sull'orizzonte temporale, mentre il secondo a lungo raggio capace di raggiungere predizioni più lontane, tuttavia perdendo in accuratezza e in capacità di predizioni a corto raggio.

Contents

1 | Introduction

Today's communication services and applications are designed to cope with the strict requirements of high Quality of Services (QoS) such as low latency and high bandwidth and availability. As the presence of failures can affect the network availability of the network, failure prediction is a crucial element for the service maintenance. Existing works on failure management in microwave networks have shown how to perform failure detection and root cause identification, but these procedure might take a long time and might not be enough to avoid the loss of communication. Hence, a solid mechanism for failure forecasting (i.e. anticipate failure event discovery), in addition to failure detection (i.e., recognize anomalies due to failure occurrences), localization (i.e., identify where the failure occurred in the network), and identification (i.e., understand the actual cause of the failure) is crucial, as it may be used by operators to perform traffic re-routing and rapid failure recovery. In our work, we propose a framework for failure forecasting based on alarm forecasting, failure detection and identification problems in microwave networks.

Our work is part of a research project, supported by SIAE Microelettronica, an Italian manufacturing company in telecommunication field, specialized in microwave networks, precisely in radio links. The microwave networks, composed by radio links, use a communication technology that travels in the free space. The fact that the communication in the microwave network is not shielded like in the wired technology, makes that the power measure of the link can change during time, due to physical impairments and changes in the environment. The failures in microwave networks can be temporary or permanent. Both temporary and permanent failures can be caused by events like deep fading, extra attenuation, interference, low margin, self-interference and Hardware failures, and in these cases the received power measure receives a severe attenuation due to some impairments (i.e., atmospheric events, physical obstacles). The duration of these events can provide better intuition on its root-cause, and help in the identification of the proper countermeasures. Failures in the network can occur due to errors in the design phase, providing a permanent effect on the link performances and on the power measures; they do not let the link work properly, causing a great number of failure events giving rise to unavailability period. This type of failure causes can be related to the wrong set up of the radio link, or

to the interference provided by other links in the same region. Other failures can be caused by the ageing of the devices or by some hardware failure, in these cases the link can work with degraded performances or stop to work respectively, providing permanent failures that must be solved by direct intervention on the radio link devices. Failures affecting the radio link, have a clear effect on the several monitored parameters as the received and transmitted powers, that can provide a hint on the failure causes to an expert. However, different failure related to the hardware components can provide the same behaviour in the power measure, so, to have a clear description of the failure cause, more information must be taken into account. So, in order to discriminate hardware failures, one has to consider the alarms present on the network elements composing the link. Nowadays, the failure identification procedure is carried out by human experts, who analyse alarms, and then based on experience, identify the possible root causes of the failure can be identified, and proper countermeasures can be adopted to restore the service. The number of links to be analysed in these networks can be enormous, leading to an increase in the time needed to restore the service. To use machine learning methods to solve the failure identification problem, part of the data must have a label. In our problem, this means that failure causes must be directly associated to network alarms describing a failure event.

Despite this, the application of machine learning on real data typically requires to hand-label data. Such labelling procedure is extremely costly as it must be carried by dedicated human expert. Hence a trade-off arises between data availability and the cost of obtaining new labelled data. So in this thesis we use as ground truth a previously developed method to automate labelling.

Under these circumstances, in this thesis we first provide short-term multi-step approach composed by a reliable alarm forecaster and subsequently by a reliable failure detector and identificator: specifically, it must be able to take as input network alarms, and provide as output the root cause that provides the future failure. Combining the two steps above (alarm prediction and failure detection and identification) a novel long-term single-step prediction approach is provided in this thesis, that by leveraging ground truth data structure and machine learning classification techniques, it detects faults without using alarms forecasting module, resulting in a complexity reduction.

1.1. Thesis Contribution

The work of this thesis is mostly related to failure forecast in microwave networks. The main contributions of this work can be listed as follows:

- We generate various set of alarms according to alarm set provided by SIAE;

- We propose two frameworks for failure prediction: First, we use a predictive deep learning model to forecast alarms in the network, in order to feed an existing detection and identification framework; Second, we use machine learning models to identify faults in a single step.

In conclusion a trade-off between the two proposed approaches must be considered: the first, short-term, more accurate but with computational constraint limiting prediction horizon and the latter, long-term, able to achieve longer predictions though losing accuracy and the ability of short-term predictions.

1.2. Thesis Outline

The remainder of the thesis is organized as follows:

In Chapter 2, we discuss related works, in which machine learning techniques are implemented to a broader set of networking use cases. Then, an overview of previous work is presented. We first describe the state of the art work related to the machine learning-based failure management in different types of networks and usage alarm logs for failure management.

In Chapter 3, the background knowledge of the thesis is presented. First, a technology background on microwave networks is provided. An overview of the hardware and software components is given; moreover the channel model, the hardware failures and alarm usage are explained. Second, a methodological background on machine learning is provided.

In Chapter 4, a functional definition of the tackled problems is provide

In Chapter 5, we present the data used in our study. In addition, we describe specific equipment type and their respective set of alarms forming our dataset.

In Chapter 6, our proposed methodologies to solve the problem are presented. Initially we describe data retrieval. Then, we discuss how we perform machine learning training and cross validation and we describe the algorithms used to perform alarms forecasting. We also present machine learning classifiers suited for both failure detection and failure identification. Finally we describe our two proposed methodologies leveraging developed ML models for failure identification: respectively, long-term single-step and short-term multi-step.

In Chapter 7, we provide numerical evaluations of failure forecast, comparing the different ML algorithms in terms of prediction metrics and complexity.

Chapter 8 concludes the research and explains the further possible studies.

2 | Related Works

This chapter summarizes the recent developments and the ongoing research works, first related to machine learning application in networking field, second to failure management, identification, localization and prediction using alarm logs in communication networks.

Fault management is an important but challenging area of telecommunication network management. A variety of techniques for failure management were developed in different networking contexts.

Work [1] discuss a real time monitoring and operation system called netdoctor which is used in mobile networks. This system exploit the collected performance data for duplex radio channel at base station subsystem. This data helps to operational personnel to solve problems.

While [2] with the continuous expansion of data availability in many large-scale, complex, and networked systems, such as surveillance, security, Internet, and finance, it becomes critical to advance the fundamental understanding of knowledge discovery and analysis from raw data to support decision-making processes. Although existing knowledge discovery and data engineering techniques have shown great success in many real-world applications, the problem of learning from imbalanced data (the imbalanced learning problem) is a relatively new challenge that has attracted growing attention from both academia and industry. The imbalanced learning problem is concerned with the performance of learning algorithms in the presence of underrepresented data and severe class distribution skews. Due to the inherent complex characteristics of imbalanced data sets, learning from such data requires new understandings, principles, algorithms, and tools to transform vast amounts of raw data efficiently into information and knowledge representation. In this paper, we provide a comprehensive review of the development of research in learning from imbalanced data. Our focus is to provide a critical review of the nature of the problem, the state-of-the-art technologies, and the current assessment metrics used to evaluate learning performance under the imbalanced learning scenario. Furthermore, in order to stimulate future research in this field, we also highlight the major opportunities and challenges, as well as potential important research directions for learning from

imbalanced data.

Predicting future performance curve and mining the top-K influential KPIs are two important tasks for Database Management System (DBMS) operations. In this paper [3], is proposed a multi-task sequence learning approach to address the two tasks in a uniform framework. The proposed approach adopts a Long Short-Term Memory (LSTM) based deep neural network model that uses multilevel discrete wavelets transform and LSTM-based Seq2Seq forecaster to capture the features in both time and frequency domains from high-dimensional time series, and achieves multi-step performance prediction and top-K KPI mining concurrently. The performance of the proposed multi-task sequence learning approach is evaluated based on two real-world DBMS datasets, which shows that the proposed approach achieves the lowest mean absolute error and root mean squared error in predicting performance scores, and significantly outperforms the state-of-the-art algorithms in both learning tasks.

In this work [4] , deep graph convolutional neural networks (DGCNN) are applied for estimating the quality-of- transmission (QoT) of unseen network states in elastic optical networks (EONs) in the presence of physical layer impairments (PLIs), including inter- and intra-channel crosstalk (XT). The objective is to find a DGCNN-QoT model that accurately estimates network state feasibility. A network state is considered feasible if the QoT of the in-service lightpaths and of the lightpath under provisioning is sufficient; that is, the DGCNN does not only infer about the feasibility of an unestablished lightpath but also whether the feasibility of the in-service lightpaths will be affected by the establishment of a new lightpath due to XT. As DGCNN model generalization over unseen graphs is known to be negatively affected by the number of possible graphs and their dimensionality, problem uncertainty and complexity is reduced by formulating the QoT estimation problem over sub- network states, capturing only the spatio-temporal correlations that are relevant to the unestablished lightpath at decision time. DGCNN model accuracy is compared to a state-of-the-art deep neural network (DNN) model trained only over per-lightpath information. It is shown that DGCNN achieves accuracies above 92%, while DNN performs poorly with accuracies as low as 77%, as it fails to infer about the feasibility of in-service connections; an indicator of the importance of explicitly considering during the QoT model training, not only the lightpath patterns, but also the network-state patterns capturing the XT effect. Importantly, it is demonstrated that deep graph learning is a promising approach towards accomplishing this objective.

The authors of [5] present an advanced method to solve the multiple failure localization problem, using the network topology information and services information; the presented approach is based on a Hopfield neural network (HNN), which receives as input a bipartite

graph with the candidate failure link and the alarm set. The HNN is used to optimize the uncertainty between the failure and alarms, in order to evaluate the exact position of the failures.

The aim of this paper [6] is to develop data-driven models by exploring the consequential relationships in the alarm and event-log database of industrial systems. Our motivation is twofold: (1) to facilitate the work of the operators by predicting future events and (2) analyse how consequent the event series is. The core idea is that machine learning algorithms can learn sequences of events by exploring connected events in databases. First, frequent sequence mining applications are utilised to determine how the event sequences evolve during the operation. Second, a sequence-to-sequence deep learning model is proposed for their prediction. The long short-term memory unit-based model (LSTM) is capable of evaluating rare operation situations and their consequential events.

In [7], the authors provide a framework divided in two parts able to solve the failure detection and identification problems in an optical link. Their framework takes as input the BER measures from the network, the authors compare various machine learning algorithms in order to identify the best solution for both problems.

The authors of [8] and [9] describe a system for discovering regularities in the alarms called Telecommunication Alarm Sequence Analyzer, which uses data mining algorithms for locating frequently-occurring episodes from sequential data and then finds confidences of rules. This application is one of the starting points in knowledge discovery and analysing network log data with frequent pattern mining. The authors of [10] and [11] proposed a clustering algorithm technique which in latter research they called i Logcluster, to determine frequent alarm pattern which represents a previously-known failure. Also this technique find outliers which can be correspond an unknown failure patterns.

Alarm correlation is widely used to detection and preprocessing parts of failure identification. Work [12] is one of the primary works using alarm correlation. In paper [13], association rules mining techniques which based on the time window preprocessing and the weighted frequent tree structure are used to find correlation of alarms. Author of [14] propose a technique based on statistical sampling to predict future sequences of alarms. In this tecnique, after calculating correlation between alarms sequences, prediction is made based on the probabilities of reoccurrence of these alarm sequences.

The authors in [15] develop a method for failure management in optical transport networks (OTN). Difference in OTN is that alarms rise not only from the optical transport plane but also from service plane and control plane, so that the large amount of alarms from different planes often interact with each other. As a result, if the association rules are directly mined

from the original alarms, the performance of the algorithm will be degraded significantly. First alarm transaction are extracted with a method combine time series segmentation and time sliding windows in pre-processing phase, then to evaluate importance of each alarm they use K-means and back propagation neural network in alarm correlation analysis. Their latter research [16] focus on failure forecasting based on alarm analysis using similar steps with combining SVM and LSTM to predict device failure in advance with observing device performances and physical parameters.

In [17], authors utilize a different mining technique. To identify important network alarms and the causes, they build a directed acyclic graphs that connect alarms with causality from a set of time series data. These graph extract related alarms.

In [18] failure management plays a role of capital importance in optical networks to avoid service disruptions and to satisfy customers' service level agreements. Machine Learning (ML) promises to revolutionize the (mostly manual and human-driven) approaches in which failure management in optical networks has been traditionally managed, by introducing automated methods for failure prediction, detection, localization and identification. This tutorial provides a gentle introduction to some ML techniques that have been recently applied in the field of optical-network failure management. It then introduces a taxonomy to classify failure-management tasks and discusses possible applications of ML for these failure management tasks. Finally, for a reader interested in more implementative details, it provides a step-by-step description of how to solve a representative example of a practical failure-management task.

An important problem of monitoring a large scale network is scalability. To solve this problem authors of [19] offer a technique based on graph theory and information theory to automatically eliminate noisy alarms from important alarms. They suggest that nodes contributing most to the entropy of a graph are the nodes most likely to generate incidents when alarms occur. This event elimination yields significant savings as time and computational complexity.

One of the main area of machine learning algorithms in networks is providing better performance for network traffic as resources optimization and metrics reduction like latency and jitter. As mentioned in survey [20] and [21], large increase in network traffic needs sustainable network performance and reliable network security solutions. Flags are used to classify different protocols and applications in network traffic routing, routers perform services corresponding to the flags of packets. In these surveys, different classification algorithms target packet flags discussed. In [22], it is mentioned that old traffic classification relied on the port-based approach where each application was identified by its

registered and known port, defined by the Internet Assigned Numbers Authority (IANA). This approach became unreliable and inaccurate due to the proliferation of new applications with unregistered or random generated ports. Another approach that gained a lot of popularity in this field is called Deep Packet Inspection (DPI). DPI performs a matching between the packet payload and a set of stored signatures to classify network traffic. However, DPI fails when privacy policies and laws prevent accessing to the packet content, as well as the case of protocol encapsulation. In [23] semi-supervised algorithm is used to detect unknown protocol on application layer. In the proposed system deep neural network choose necessary protocol flow features and classify unknown protocols unlike the traditional ways.

Software defined network (SDN) based network planning frameworks are used for monitoring network, control and automatic resource adaptation. Authors of [24] work on automatically classify real network traffic data collected from ONOS (Open Network Operating System) platform. Combining this information with SDN enables the networks to be efficiently and autonomously coped within SDN/NFV environment. Works [25] and [26] propose a SDN based centralized and programmable system to monitor and collect real-time data from network on top of network management. This framework provides control and adapt network elements when it is needed and ML based performance prediction, so system can maximize link capacity based on predicted link performance. As author of [26] said traditional network planning is not scalable and can't be use resources efficiently.

To achieve high QOS by automatically, machine learning algorithms are widely used. In [27], a supervised learning algorithm is proposed to choose the best logical network on the physical network to guarantee QoS. A classification method automatically determine a policy of resource selection for virtual networks and adaptively updates the resources based on performances of virtual networks.

It is mentioned in [28] that using deep neutral network with gradient descent method from huge amount of data can cause to stuck in local optimum. They propose a deep neural evolution network (DNEN) to solve that fault location problem. DNEN generates a large number of network model populations initially, and then the initial population screened by loss function generate new models by crossover and mutation, and so on, until the network models meet the accuracy requirement.

Authors of [29] utilize unsupervised clustering algorithm to decrease latency with determining which low power nodes should upgraded to fog nodes to provide fastest communication in 5G networks.

Another use of machine learning for communication networks is security. In survey [30], different algorithms are discussed and compared to solve attacks based on their target, location of attacker and targeted layer of the protocol. In [31], authors propose using recirculation neural network to discover unknown attack types in real-time, while emphasizing old signature and rule based methods can be insufficient to detect new attack because of the lateness of the signature databases. In [32], machine learning is used both prevent security problem and solve network congestion caused from anomalies in the network traffic. Work compare naive bayes and k-means algorithms to extract of network behaviour.

Moreover machine Learning is used to forecast QoT parameters and alarms. The performance of optical devices can degrade because of aging and external causes like, for example, temperature variations. Such degradation might start with a low impact on the Quality of Transmission (QoT) of the supported lightpaths (soft-failure). However, it can degenerate into a hard-failure if the device itself is not repaired or replaced, or if an external cause responsible for the degradation is not properly addressed. In work [33] is proposed a comparison between the QoT measured in the transponders with the one estimated using a QoT tool. Those deviations can be explained by changes in the value of input parameters of the QoT model representing the optical devices, like noise figure in optical amplifiers and reduced Optical Signal to Noise Ratio in the Wavelength Selective Switches. By applying reverse engineering, the value of those modeling parameters can be estimated as a function of the observed QoT of the lightpaths. Experiments reveal high accuracy estimation. In the era of the fifth-generation fixed network (F5G), optical networks must be developed to support large bandwidth, low latency, high reliability, and intelligent management. Studies have shown that software-defined optical networks (SDON) and artificial intelligence can help improve the performance and management capabilities of optical networks. Inside a large-scale optical network, many types of alarms are reported that indicate network anomalies. Relationships between the alarms are complicated, making it difficult to accurately locate the source of the fault(s). In work [34] is proposed a knowledge-guided fault localization method, using network alarm knowledge to analyze network abnormalities. This method introduces knowledge graphs (KGs) into the alarm analysis process. It also proposes a reasoning model based on graph neural network (GNN), to perform relational reasoning on alarm KGs and locate the network faults. It develops an ONOS-based SDON platform for experimental.

Finally, time series prediction problems are a difficult type of predictive modeling problem. Fortunately, LSTM models can solve hard long time lag problem [35]. In paper [36], is proposed a time series prediction method based on a variant long short-term memory

(LSTM) recurrent neural network. In the proposed method, first the memory module of the LSTM recurrent neural network is improved by merging its forget gate and input gate into one update gate, and using Sigmoid layer to control information update. Using improved LSTM recurrent neural network, a time series prediction model is developed. In the proposed model, the parameter migration method is used model update to ensure the model has good predictive ability after predicting multi-step sequences. Experimental results show, compared with several typical time series prediction models, the proposed method have better performance for long-sequence data prediction.

In [37] classification problems, the class imbalance problem will cause a bias on the training of classifiers and will result in the lower sensitivity of detecting the minority class examples. The Mahalanobis-Taguchi System (MTS) is a diagnostic and forecasting technique for multivariate data. MTS establishes a classifier by constructing a continuous measurement scale rather than directly learning from the training set. Therefore, it is expected that the construction of an MTS model will not be influenced by data distribution, and this property is helpful to overcome the class imbalance problem. To verify the robustness of MTS for imbalanced data, this study compares MTS with several popular classification techniques. The results indicate that MTS is the most robust technique to deal with the classification problem on imbalanced data. In addition, this study develops a "probabilistic thresholding method" to determine the classification threshold for MTS, and it obtains a good performance. Finally, MTS is employed to analyze the radio frequency (RF) inspection process of mobile phone manufacturing. The data collected from the RF inspection process is typically an imbalanced type. Implementation results show that the inspection attributes are significantly reduced and that the RF inspection process can also maintain high inspection accuracy.

Last but not least, in work [38], an automatic labelling model for failure root-causes which are divided to six categories, namely, Deep fading, Extra attenuation, Interference, Low Margin, Self-Interference and Hardware failures, is implemented with using power measurements.

So, we are proposing a first solution for fault root-causes prediction via machine learning techniques based on alarms of microwave links.

3 | Background

3.1. Microwave Network Technologies

Microwave is a wireless communication technology that uses high frequency beams of radio waves to provide high-speed wireless connections that can send and receive voice, video, and data information.

Microwave links are widely used for point-to-point communications because their small wavelength allows conveniently-sized antennas to direct them in narrow beams, which can be pointed directly at the receiving antenna. This allows nearby microwave equipment to use the same frequencies without interfering with each other, as lower frequency radio waves do.

3.1.1. Hardware Components

The microwave communication structure includes a microwave radio at each site, connected to a directional antenna via a transmission line. In this section we discuss these three fundamental elements and their impact on the communication quality. The essential building blocks of a microwave communication system are shown in 3.1

Radio

Each end of the connection has its radio unit, commonly with both transmission (TX) and receiving (RX) capabilities. On the transmitter site (TX) it generates the signal to transmit and code it, then it aggregates and compresses the signal in a relatively small radio channel This procedure is called modulation. After modulation, the radio up-converts the radio channel to the right microwave frequency to be transmitted. The radio on the receiver site (RX) operates the opposite procedure, i.e., it down-converts the radio channel from the microwave frequency, and then demodulates the signal to be sent again on the cable communication.

There are three basic radio configuration depend on where the active components are

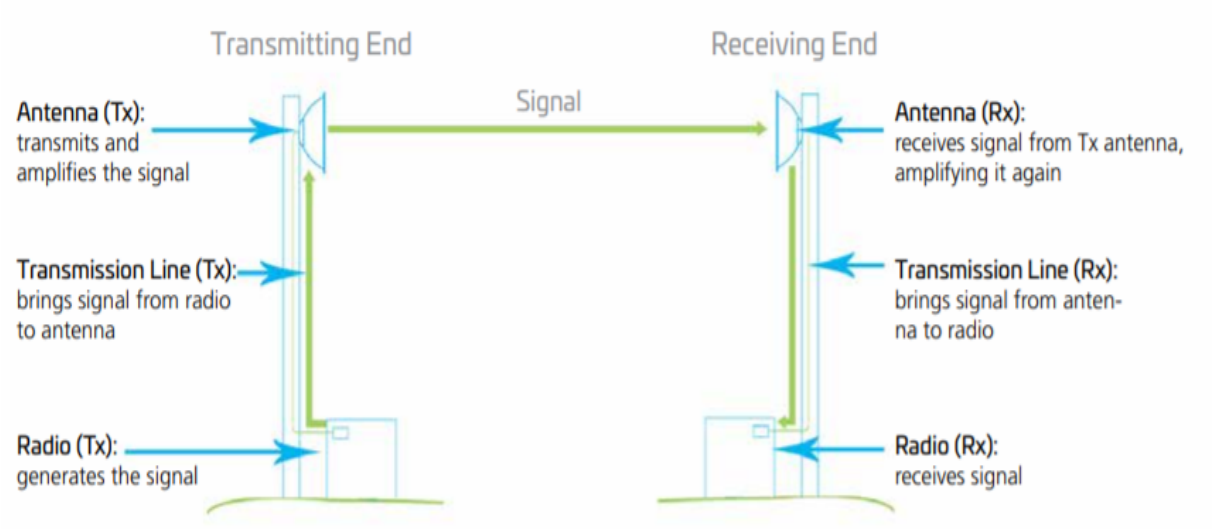

Figure 3.1: Main scheme of microwave communication [39]

located which are 'All indoor', 'All Outdoor', 'Split-Mount'[40]. Maintenance and upgrades are easier for Indoor units (IDU) than outdoor units(ODU), but transmission line losses can be caused because of distance from IDU to antenna in this configuration. As a trade-off option Split-mount configuration can be used.

Transmission Line:

The transmission line is a connector that transmits energy from the radio to directional antenna. While transmitting or while receiving, the energy transfer has to be done effectively, without power wastage. To achieve this, certain important parameters are resistance, inductance, capacitance and conductance.

Transmission line type can be determined by frequencies in use because of the amount of signal loss they can introduce [39]. Coaxial cables are more suitable for lower frequencies. Above 2GHz range, some signal loss can be encountered. Waveguides type can support frequencies up to around 40 GHz.

Antenna

A microwave antenna is designed to radiate and receive electromagnetic waves of certain frequencies. A directional antenna in a microwave system is typically parabolic in shape, as this permits the greatest focus of energy possible in a single beam. They are usually polarized, vertical or horizontal, based on the location of their feed connection. There are various sorts of antennas, chosen by the connection link characteristics and assessing some particular antenna parameters, i.e. radiation pattern, frequency of operation, half-power

beamwidth, gain and return loss.[39]

3.1.2. Channel Characterization

The electromagnetic waves are radiated in free space and suffer the channel effects. These affects the transmission of microwave communication unlike wired network where electromagnetic waves are guided, so their attenuation is provided mostly by internal effects. Not only it is susceptible to noise, interference, and other channel impediments, but these impediments change over time in unpredictable effects of environment dynamics.

Different channel effects can be explained as:

Free Space Path Loss

Path loss, or path attenuation, is the reduction in power density of an electromagnetic wave as it propagates through space. Path loss is a major component in the analysis and design of the link budget of a telecommunication system.

Multi-path fading

In the propagation between transmitter and receiver, signal can follow different paths due to total or partial reflection over obstacles. The behaviour of the waves when interacting with objects depends on their frequency and the characteristics and size of the objects.

Shadow fading

Besides free-space attenuation, other attenuations may be present due to atmosphere (depending on the frequency, fog, rain, etc.) and obstacles. Moreover, propagation near the earth surface has some fundamental differences with respect to free-space.

Interference

Because the radio waves travel in narrow beams confined to a line-of-sight path from one antenna to the other, they do not interfere with other microwave equipment, so nearby microwave links can use the same frequencies.

3.1.3. Diversity

Both multi-path fading and shadow fading usually induce a high penalty on signal quality. There are different techniques to mitigate the effects of these fading. One of them is

diversity combining of independently fading signal paths. Diversity combining exploits the fact that independent signal paths have a low probability of experiencing deep fades simultaneously. Thus, the idea behind diversity is to send the same data over independent fading paths. These independent paths are combined in such a way that the fading of the resultant signal is reduced. Thus the basic idea of diversity is repetition or redundancy of information. In virtually all the applications, the diversity decisions are made by the receiver and are unknown to the transmitter.

The following diversity schemes can be used in microwaves links:

- Space diversity: Independent signal paths can be combined together with multiple antennas.
- Time Diversity: The information signal is transmitted repeatedly in time at regularly intervals.
- Polarization diversity: Polarization diversity is using either two transmit antennas or two receive antennas with different polarization (e.g., vertically and horizontally polarized waves).
- Angle diversity: Directional antennas provide angle diversity by restricting the receive antenna beam width to a given angle.
- Frequency diversity: Frequency diversity is achieved by transmitting the same narrow band signal at different carrier frequencies, where the carriers are separated by the coherence bandwidth of the channel.

3.1.4. Adaptive Modulation and Coding

The AMC goal is to improve the efficiency of the radio link by increasing network capacity over the existing infrastructure, while reducing sensitivity to environmental interferences by adapting modulation and coding to the channel characterization.

3.1.5. Hardware Failures

Microwave communication is affected by environment, interference with other links, broken equipment, etc. These can reduce the reliability of the link and the loss of huge amount of data with temporary unavailability on the link. Determining the underlying reason for failures should be immediate to restore the link back.

Network failures can be divided into two broad categories, namely, hardware failures and

propagation-related failures. Propagation problems are caused by environmental reasons, while hardware failures are directly related to equipment in the microwave network. In this thesis, we concentrate on hardware failures, which relate to equipment malfunctioning due to, e.g., fan failure, power supply issues, overheating.

3.1.6. Alarms

An alarm is a message raised by a network element when there is an irregularity on the link which contains information about a specific problem. The information contents of alarm messages are very diverse e.g., in the case of alarms of power supplies or cable failures. Each alarm has some features that indicate the time, location, and equipment in which the alarm was raised together with some measurements related to the problem.

The sharp increase in the number of equipment and links would lead to more frequent failures as well as massive alarm information in communication networks, which increases the difficulty of highly accurate fault localization. The alarm data contains hidden valuable knowledge about the network behavior, particularly considering information retrieved by combining multiple alarm sources. This knowledge can be used to filter redundant alarms, locate problems in the network, and possibly predict severe faults.

A monitoring system collects all flows of alarms from the telecommunication network. This introduces a huge amount of data, which makes the procedure of analyzing much more complex. Even though alarms have many information about the related failure, that may not provide adequate information to identify the real cause. One failure might result in several different alarms arising from several network elements. Different alarms should be considered together to find the root cause of the problem. All this information must be handled carefully when evaluating alarms occurrences and the correlation between different alarms. To analyze log files separately and manually each link-related log afterward is seldom practical and highly time-consuming.

3.1.7. Performance Metrics

Recommendations G.826 and G.828 [41] define how to measure the performance of radio links in terms of availability and errors with the metrics determined by ITU-T. The estimations refer to the block which is a set of consecutive bits associated with the path, where each bit has a place with one and only one block. The recommendations G.826 and G.828, as introduced in [42], contain the accompanying error counters that are characterized as follows.

- Errored Block (EB): A block in which one or more bits are in error.
- Errored Second (ES): One-second with one or more errored blocks or at least one defect; where the defects are different errors from an EB defined according to the technology.
- Severely Errored Second (SES): One-second period that contains 30% errored blocks or at least one defect.
- Background Block Error (BBE): An errored block not occurring as part of an SES.
- Severely Errored Period (SEP): A period during which at least three but not more than 9 consecutive severely errored seconds (SES) occur.
- Errored Second Ratio (ESR): The ratio of ES out of total time in seconds in available time during a fixed measurement interval.
- Severely Errored Second Ratio (SESR): The ratio of SES out of total time in seconds in available time during a fixed measurement interval.
- Background Block Error Ratio (BBER): The ratio of Background Block Errors (BBE) out of the total number of blocks in available time during a fixed measurement interval.
- Severely Errored Period Intensity (SEPI): The number of SEP events in the available time, divided by the total available time in seconds.
- The previous metrics are related to the errors on the communication but none of them defines directly how to measure the unavailability. The system is defined unavailable, when ten consecutive severely errored seconds (SES) are measured, and it becomes available again after ten consecutive seconds that are not severely errored; to measure the unavailability the “Unavailable Seconds (UAS)" measure is defined, it contains the number of seconds when the system is unavailable in a measurement interval; the UAS of a measurement interval is computed as the sum of all the time interval containing at least ten consecutive severely errored seconds in at least one direction of transmission.

3.1.8. Equipment Type

AGS20 is the universal Microwave aggregation platform to address all the new requirements of the different radio access network (RAN) generation over a common wireless transport infrastructure like increasing throughput, low latency performances, higher capacity, introducing SDN and automation in the higher management layers of the network.

The AGS20 is a family of microwave aggregators with multiple mechanical arrangements specifically designed to fit different needs and deployment scenarios in a modern mobile backhaul infrastructure. It has high performance Carrier Ethernet 2.0 and IP/MPLS engine supporting L2/L3 services, 4 to 2048 QAM modulations while maintaining full support of the legacy traffic E1 and STM-1 services. It has fully flexible architecture with multiple variants. AGS20 is the next Generation Indoor Unit for split Mount Radio, capable of addressing traditional split mount application with multiple IF radio, and aggregation of multiple all out door radio solutions. [43]

3.2. Machine Learning Methodologies

Machine learning is the study of computer algorithms that improve automatically through experience and by the use of data. Machine learning algorithms build a model based on sample data in order to make predictions or decisions without being explicitly programmed to do so.

It involves computers learning from data provided in order to carry out specific tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to create the required algorithms manually. In practice, it can turn out to be more effective to help the machine develop its algorithm, rather than having human programmers specify every needed step.

ML approaches are traditionally divided into three broad categories, depending on the feedback available to the learning system:

Supervised learning: There are input variables and an output variable in supervised learning, and the goal is to learn the mapping function from the input to the output. The algorithm iteratively makes predictions on the training data and is corrected by the training dataset.

Unsupervised learning: Unsupervised learning is where there are only input data and no corresponding output variables. No labels are given to the learning algorithm, leaving it on to find its own structure in its input. The goal for unsupervised learning is to model the underlying structure or distribution in the data to learn more about the data.

Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a particular goal. As it navigates its problem space, the program

is provided feedback that's analogous to rewards, which it tries to maximize.

Other approaches, such as dimensionality reduction, have been developed, which don't fit neatly into this three-fold categorization. Sometimes, more than one is used by the same machine learning system.

Recently, deep learning has become the dominant approach for much ongoing work in the field of machine learning.

The workflow of a ML project includes all the steps required to build the proper machine learning project from scratch [44]. In figure 3.2, the steps of a standard workflow of ML-based projects is shown:

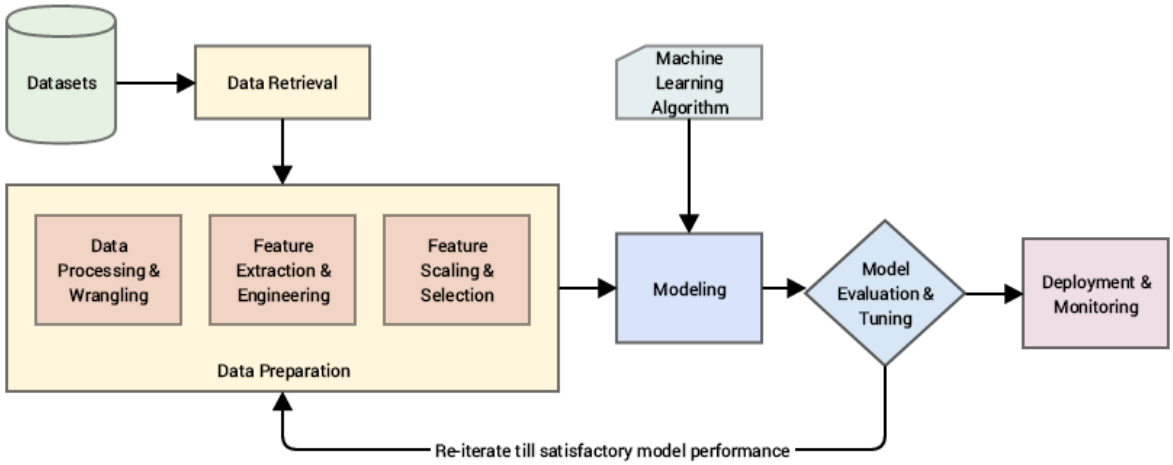

Figure 3.2: A standard machine learning pipeline[45]

1. Data Gathering: The process of gathering data depends on the type of project we want to make. The data set can be collected from various sources such as a file, database and many other such sources but the collected data cannot be used directly for performing the analysis process as there might be a lot of missing data, extremely large values, unorganized text data or noisy data. Therefore, to solve this problem Data Preparation is done.

2. Data Pre-processing: Data pre-processing is one of the most important steps in machine learning that helps building models more accurately. Data pre-processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn't feasible for the

analysis. In this process missing, noise, inconsistent data and the outliers should be handled.

3. Model Choosing: The main goal is to train the best performing model possible, using the pre-processed data. The most suitable model should be chosen according to the goal of the project, constraints and input data. First higher category of model can be defined such as regression or classification then best algorithm can be determined in terms of the target.

4. Training and testing: For training a model generally data initially is split into 3 three sections which are 'Training data', 'Validation data' and 'Testing data'. Model is trained using 'training data set', tune the parameters using 'validation set' and then test the performance of your model on unseen 'test data set'. An important point to note is that during training the model only the training and/or validation set is available. The test data set must not be used during training the classifier. The test set will only be available during testing the classifier.

5. Evaluation: Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. To improve the model we might tune the hyper-parameters of the model and try to improve the accuracy and also looking at the confusion matrix to try to increase the number of true positives and true negatives.

3.2.1. Clustering

In this section, we explain the mathematical concept behind the clustering algorithms used in our work to compute ground truth. One of this work's main objectives is to group data on failed microwave links based on alarm information to identify different root causes corresponding to different patterns of alarms occurring together and/or with similar characteristics.

Clustering is one of the main areas of unsupervised learning. It is the task of dividing the data points into several groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. This can be achieved by various algorithms that differ significantly in understanding what constitutes a cluster and how to find them efficiently. Popular notions of clusters include groups with small distances between cluster members, dense areas of the data space, intervals, or particular statistical distributions.

The appropriate clustering algorithm and parameter settings depend on the individual

data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data preprocessing and model parameters until the result achieves the desired properties.

K-Means

K-means is one of the centroid-based unsupervised learning algorithms. The procedure follows a simple way to group a given set of data points $(x1, x2, ..., xn)$, where each data points is a d-dimensional vector, into a given (pre-determined) k clusters $C = \{C1, C2, ..., Ck\}$ with minimizing the within-cluster sum of squares error (SSE) [46].

The algorithm begins with the initialization of k centroids. Each clusters are defined with a centroid which is the location representing the center of the cluster. Then each data point is assigned to the closest centroid by computing the sum of the squared distance between data points and all centroids. At this point, new centroids are re-calculated by taking the average of all data points that belong to each cluster. These steps are kept iterating until there is no change in the SSE, which is computed as:

$$SSE = \sum_{i=1}^{k} \sum_{x \epsilon C_i} \|x - \mu_i\|^2 \tag{3.1}$$

where k is the total cluster number, μ_i is the centroid d dimensional vector of the cluster C_i.

The goal of clustering is not just to make clusters, but to reach meaningful clusters. Clusters of K-means are acceptable when the data points within a cluster are close together, and far from other clusters. Disadvantages of K-means is that algorithm does not guarantee convergence to the global optimum. The result may depend on the initial clusters. As the algorithm is usually fast, it is common to run it multiple times with different initial points.

In figure 3.3, a 2D example of k-means clustering can be seen, each color represent a different cluster and the red points are the centroids of each cluster.

Two commonly-used metrics to evaluate clustering quality are described below:

Inertia: Inertia tells how far away the points within a cluster are. It is the sum of squared

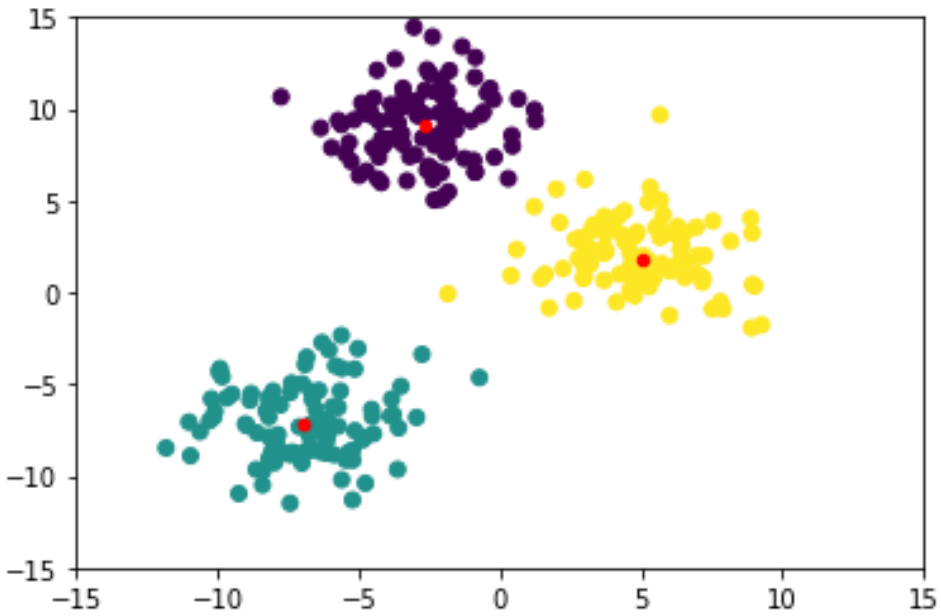

Figure 3.3: Example of 2D K-Means clustering

distances of samples to their closest cluster center. It can be calculated as formula 2.1. Therefore, a small value of inertia is aimed for. The range of inertia's value starts from zero and goes up. The value of inertia decreases as the number of clusters increase. Therefore, it is a good idea to plot the graph of inertia as the number of clusters increase. Elbow method is used in order to find the optimal number of clusters. As depicted in the following diagram 3.4, the number of clusters to be chosen over there can be equal 5 or 6 as after that curve reaches a plateau.

Silhouette Coefficient: Silhouette score tells how far away the data points in one cluster are, from the data points in another cluster. The coefficient varies between -1 and 1. A value close to 1 implies that the instance is close to its cluster is a part of the right cluster. Whereas, a value close to -1 means that the value is assigned to the wrong cluster.

Silhouette-Coefficient of data point i is calculated as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{3.2}$$

where a is average distance to all other points within same cluster as that of point i while b is minimum of average distance to all other points from all other clusters.

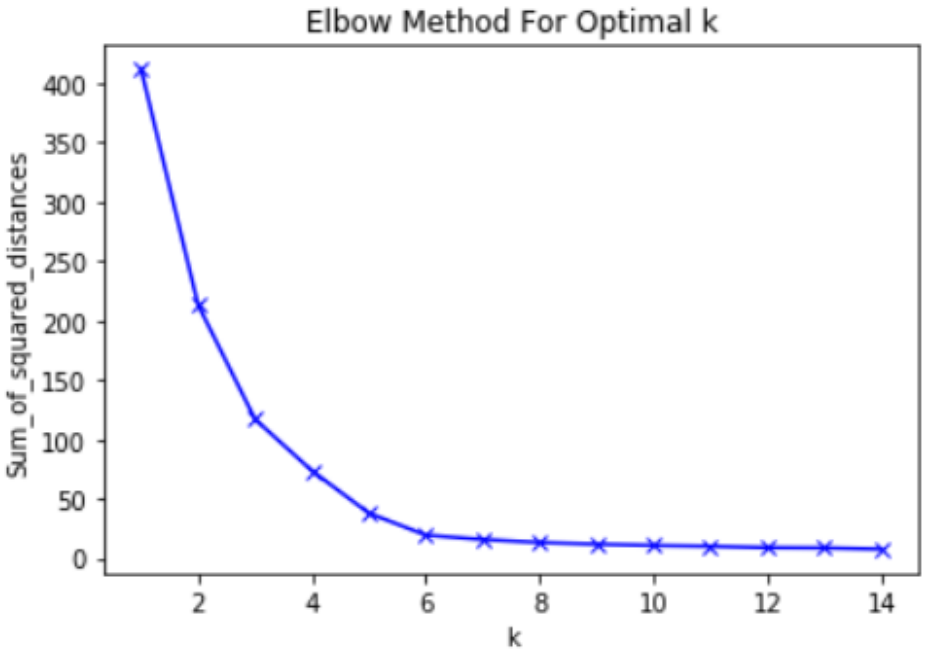

Figure 3.4: Selection of number of cluster with elbow method

3.2.2. Classification

One main focus of this work is to provide an automated program that take alarm logs as input and provide the root cause of failure in the microwave link as output. We model this task as a ML classification problem, where our target labels is produced by human experts with evaluation of clustering results.

Classification is a supervised learning algorithm because there is a labelled training data. An algorithm that implements classification, especially in a concrete implementation, is known as a classifier. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.

In this section we will describe from a mathematical point of view the supervised learning algorithms used in our analysis. In case the output of the prediction is a continuous value, we are facing a regression problem. Conversely, when a discrete value is predicted, we are solving a classification problem. The challenges to get good performance from these algorithms are the number of samples, that in some cases must be huge, and the number of samples per class, that normally must be more or less balanced. In our case we are facing a classification problem because we are interested in predicting which problem is affecting our radio link.

Among the several algorithms, some algorithms have a lower computational complexity,

they provide different representations and allow the specification of prior knowledge. If one algorithm seems to outperform another in a certain situation, it is a consequence of its fit to the particular problem, not the general superiority of the algorithm. When confronting a new problem, we should focus on the aspects that matter most prior information, data distribution, amount of training data and cost.

There are three main type of classifiers:

- Binary classification: It refers to classification tasks that have two mutually-exclusive class labels. e.g a mail can be a spam or not spam
- Multi-class classification: It refers to classification tasks that have more than two mutually-exclusive class labels. e.g. an animal photo can be a dog, cat or bird

  Algorithms that are designed for binary classification can be adapted for use for multi-class problems. This involves using a strategy of fitting multiple binary classification models as [47]

  - One-vs-Rest: The strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. Only $n_{classes}$ classifiers are needed.
  - One-vs-One: Fit one binary classification model for each pair of classes, then the class which received the most votes is selected. Because $n_{classes} * (n_{classes} - 1)/2$ classifiers are needed, this method is usually slower than one-vs-the-rest.
- Multi-label classification: Classification tasks are where one or more class labels may be predicted for each sample. e.g. a document can have content on sports, finance, and politics.

Support Vector Machine

The objective of the support vector machine algorithm is to find a hyperplane which linearly divide the n-dimensional data points in two component. The hyperplane should be positioned with the maximum distance to the data points. Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. The data points with the minimum distance to the hyperplane are called Support Vectors. Due to their close position, their influence on the exact position of the hyperplane is bigger than of other data points.

Given data points $\{x_i, y_i\}$ where $i = 1 \cdots N, y_i \in \{-1, 1\}$ Any hyperplane can be written

as the set of points x satisfying:

$$\mathbf{w}^T\mathbf{x} - b = 0 \tag{3.3}$$

where w is the normal vector to the hyperplane. The parameter $\frac{b}{\|w\|}$ determines the offset of the hyperplane from the origin along the normal vector w. Figure 3.5 shows the optimal hyperplane for a 2D data. Data points which are on/above $\mathbf{w}^T\mathbf{x} - b = 1$ are assigned to label 1 and data points which are on/below $\mathbf{w}^T\mathbf{x} - b = -1$ are assigned to label -1.

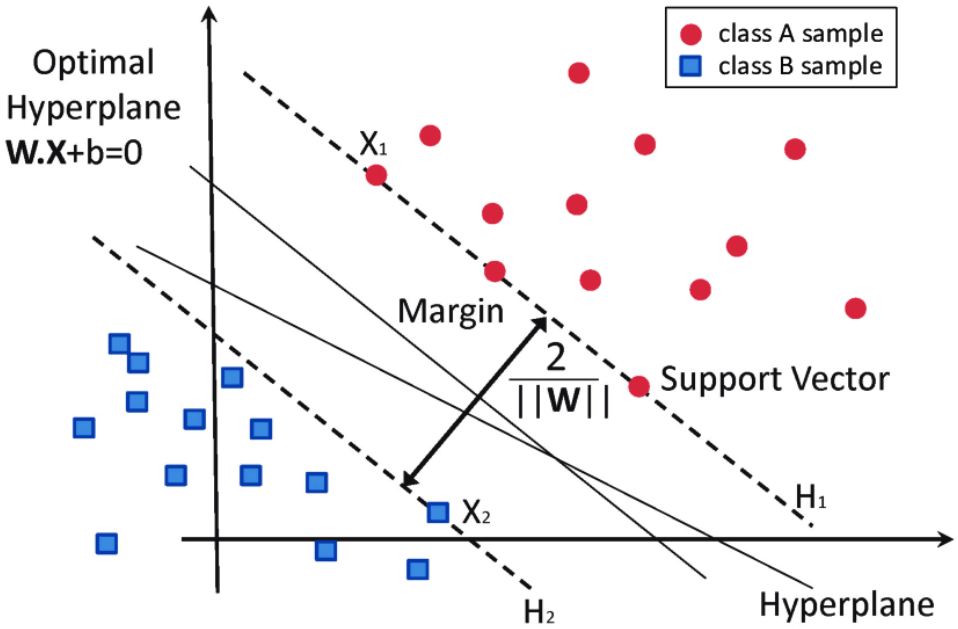

Figure 3.5: Optimal hyperplane of 2D data provided by SVM [48]

To extend linear SVM to cases in which the data are not linearly separable, we will allow some margin violation to occur. This type of classification are called as soft margin classification. We will relax the constrains of the equation slightly to allow the margin violation to occur with the help of positive slack variable.

A way to create non-linear is suggested classifiers by applying the kernel trick to maximum-margin hyperplanes. The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be non-linear and the transformed space high-dimensional; although the classifier is a hyperplane in the transformed feature space, it may be non-linear in the original input space. Some common kernels are polynomial, Gaussian radial basis function. [49]

Random Forest

The another supervised algorithm presented in this section is random forest (RF), which are a combination of decision tree predictors; in this section we first analyse the building block of RF, the decision tree, then we go deep in how the individual trees are combined to create the random forest structure.

Decision trees are very popular among ML algorithms because their intelligibility and simplicity. A decision tree is constructed from observations about an item to conclusions about the item's target value which are represented in the leaves. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. At each node, available attributes are evaluated based on separating the classes of the training examples using either a purity or impurity measure. Given a data set S contains examples from n classes, the classification task the measurements are calculated as:

$$Gini(S) = 1 - \sum_{j=1}^{n} p_j^2 \tag{3.4}$$

$$Entropy(S) = \sum_{j=1}^{n} -p_j \log(p_j) \tag{3.5}$$

where p_j is the relative frequency of class j in S.

In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, i.e. have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model. [50]

Random forest is an ensemble method which means that combining the prediction of multiple classifiers to achieve better performance thane base classifiers. Decision structure can be seen in 3.6.

Given training set D, set of variables to test for split F, each tree k_i depends on the values of a random sub datasets D_i sampled from training set independently and with the same distribution for all trees in the forest. Each tree learn from its sub datasets using at each

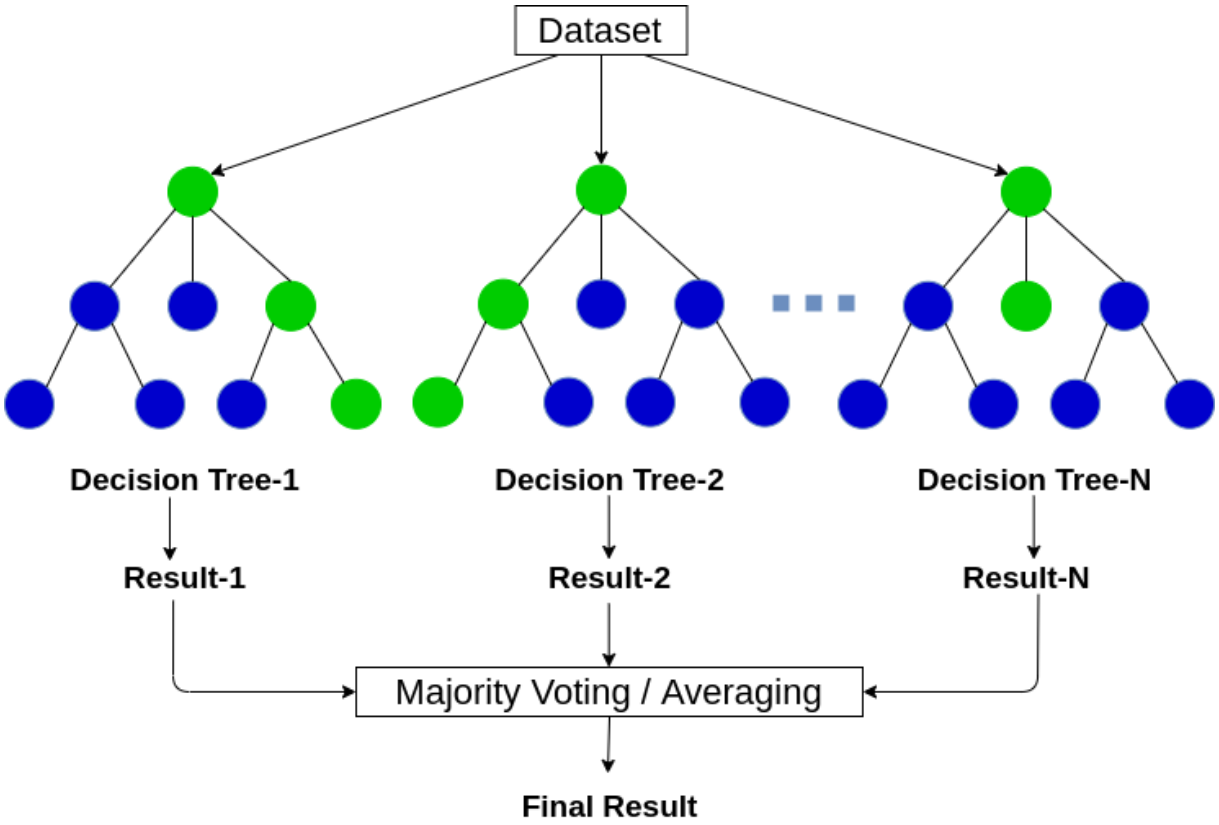

Figure 3.6: Decision scheme of random forest [51]

node only a subset F_i. Output is computed as the majority voting (for classification) or average (for prediction) of the set of k generated trees. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest (how good are they on average) and the correlation between them (how much group-think is there in the predictions).

K-Nearest Neighbours

In statistics, the K-nearest neighbors algorithm (k-NN) is used for classification and regression. In both cases, the input consists of the K closest training examples in data set. The training dataset is searched for the instances that are more similar to the unlabeled instance. No explicit training step is required. Similarity is defined according to a distance metric between two data points. A popular choice is the Euclidean distance but other measures can be more suitable for a given setting and include the Manhattan, Chebyshev and Hamming distance.

Given a positive integer K, an data points x and a similarity metric d, KNN classifier compute d between x and each training observation. Then it estimates the conditional

probability for each class and input x gets assigned to the class with the largest probability [52]:

$$P(y = j|X = x) = \frac{1}{K} \sum_{i \in A} I(y^{(i)} = j) \tag{3.6}$$

Where I(x) is the indicator function which evaluates to 1 when the argument x is true and 0 otherwise, Set A consist the closest K points in the training set.

Figure 3.7 shows how to knn classification works. In this example if k=2, assigned class will be red because of the majority but if k=3, it will be blue. The output depends on value of K. If it is too large, neighborhood may include quite dissimilar examples or if it is too small, classification might be sensitive to noise points.

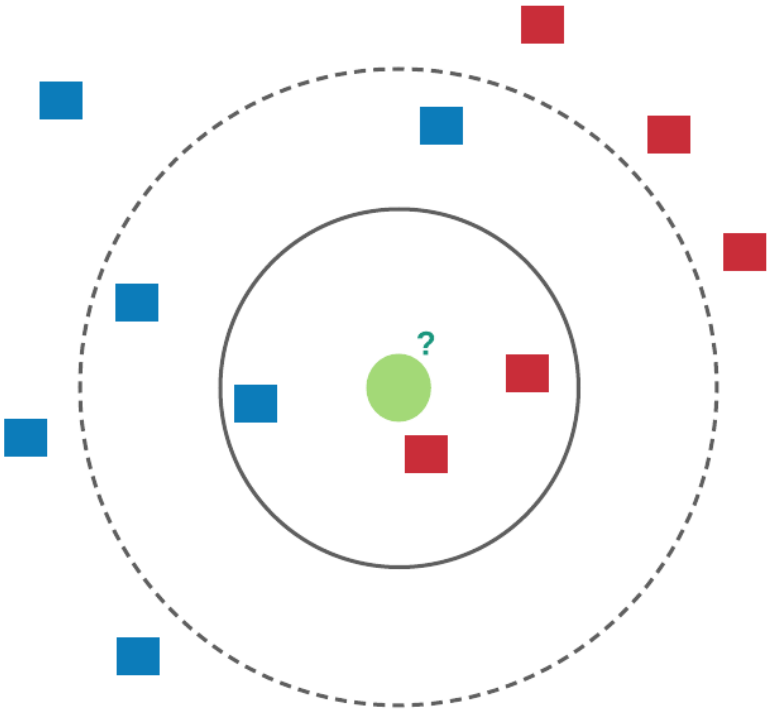

Figure 3.7: Decision scheme of K-Nearest Neighbours

Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor.

In linear scan classification time for a single distance depends on the number of data points and the number of variables. Instead of it, search can be speeded up by using hierarchically techniques like KD-Trees, Ball-Trees.

Artificial Neural networks

An Artificial Neural Network (ANN) is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. These structures are used for broad deep learning algorithms which are popular part of ML to solve hard cases and extracting hidden pattern in the problems. Their areas are increasing with big data and computational complexity. In this section we focus on the feed forward neural networks used for the classification task.

Neural nets consists of an artificial network of functions, called parameters, which allows the computer to learn, and to fine tune itself, by analyzing new data. Each parameter, sometimes also referred to as neurons, is a function which produces an output, after receiving one or multiple inputs. The inputs can be the feature values of a sample of external data, such as images or documents, or they can be the outputs of other neurons. The outputs of the final output neurons of the neural net accomplish the task, such as recognizing an object in an image.

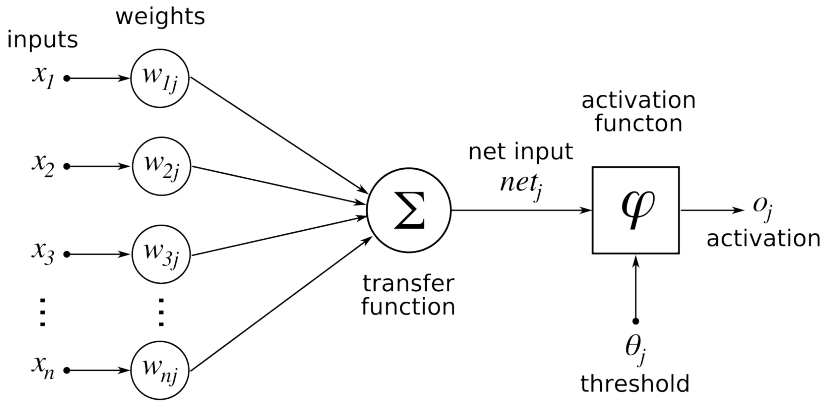

Figure 3.8: Structure of a single neuron

To find the output of the neuron, first we take the weighted sum of all the inputs, weighted by the weights of the connections from the inputs to the neuron. We add a bias term to this sum. A bias value allows you to shift the activation function curve up or down. This weighted sum is then passed through a (usually nonlinear) activation function which major types of are sigmoid, tanh, relu and softmax, to produce the output. The initial inputs are external data, such as images and documents. The difficulty lies in determining the optimal value for each bias term, as well as finding the best weighted value for each pass in

the neural network. To accomplish this, one must choose a cost function. A cost function is a way of calculating how far a particular solution is from the best possible solution. There are many different possible cost functions each with advantages and drawbacks, each best suited under certain conditions. The ultimate outputs accomplish the task, such as recognizing an object in an image. Output of single neuron can be formulated as;

$$a = \sigma(b + \sum_{i=1}^{n} x_i w_i) \tag{3.7}$$

where x_i is each input value, w_i is the weight of each input, b is bias, σ is the activation function and a is the output.

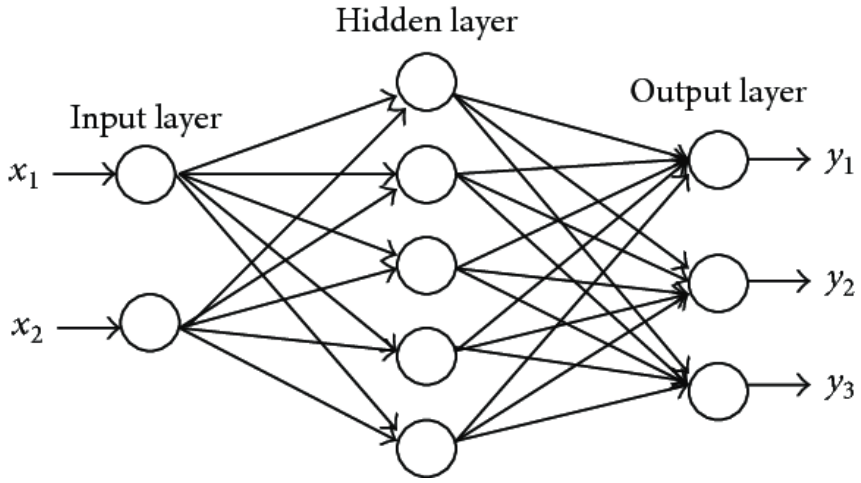

Figure 3.9: Three layer feed-forward neural network

In multilayer networks, outputs of each layer take the output of previous layer as input. Figure 3.9 shows a three-layer network. Outputs are calculated as:

$$a^0 = Input \tag{3.8}$$

$$a_j^l = \sigma\left(b_j^l + \sum_k w_{jk}^l a_k^{l-1}\right) \tag{3.9}$$

$$a^L = Output \tag{3.10}$$

where l and j respectively stand for layer and node.

A binary classification problem may have a single output neuron and use a sigmoid activation function to output a value between 0 and 1 to represent the probability of predicting

a value for the class 1. If the sum of the input signals exceeds a certain threshold, it either outputs a signal or does not return an output. This also called perceptron.

In multi-class classification, the neural network has the same number of output nodes as the number of classes. Each output node belongs to some class and outputs a score for that class. Scores from the last layer are passed through a softmax layer. The softmax layer converts the score into probability values. At last, data is classified into a corresponding class, that has the highest probability value.[53] The softmax function turns a vector of K real values into a vector of K real values that sum to 1.[54]. Softmax activation function can be formulated as follow:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.11}$$

where K is the number of classes (number of output node) and z_i of the input vector z.

Measuring Classifier Performance

A confusion matrix is a tabular way of visualizing the performance of classifier. Each entry in a confusion matrix denotes the number of predictions made by the model where it classified the classes correctly or incorrectly.

	True class= 1	True class=0
Predicted class = 1	TP	FP
Predicted class = 0	FN	TN

Table 3.1: Statistics of equipments

A binary classification problem has only two classes to classify, preferably a positive and a negative class. True Positive (TP) refers to the number of predictions where the classifier correctly predicts the positive class as positive. True Negative (TN) stands for the number of predictions where the classifier correctly predicts the negative class as negative. False Positive (FP) refers to the number of predictions where the classifier incorrectly predicts the negative class as positive and False Negative (FN) is opposite of FP. [55]

Accuracy: It gives the fraction of the total samples that were correctly classified by the classifier:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.12}$$

Precision: It tells what fraction of predictions as a positive class were actually positive:

$$Precision = \frac{TP}{TP + FP} \tag{3.13}$$

Recall: It calculates what fraction of all positive samples were correctly predicted as positive by the classifier. It is also known as True Positive Rate (TPR), Sensitivity, Probability of Detection:

$$Recall = \frac{TP}{TP + FN} \tag{3.14}$$

F1-score: It combines precision and recall into a single measure:

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} \tag{3.15}$$

A good model has a accuracy, precision and recall closer to 1. Accuracy suffers the presence of imbalanced classes, due to the fact that predicting the most representative class will provide high value even if the classification is completely wrong. To deal with that sensitive of the Accuracy, Precision and Recall are calculated jointly, analyzing them alone can provide to misleading results, this because classifying only one instance correctly yields 100% precision, but a very low recall, and respectively classifying all instances as positive yields 100% recall, but a very low precision. In order to avoid these situations, the F1-score was defined.

Unlike binary classification, there are no positive or negative classes in multi-classes classification. TP, TN, FP and FN for each individual class should be considered. Metrics should be calculated for each class individually and then takes unweighted mean of the measures. This individual metrics called macro metrics. If the metrics are calculated globally, which are called micro metrics, all the measures become equal as Precision = Recall = F1-score = Accuracy [56]

Validation Methods

Cross validation (CV) is one of the technique used to test the effectiveness of a ML models and generalized the models, it is also a re-sampling procedure used to evaluate a model if the data is limited. To perform CV, a portion of the data must be kept aside on which is not used to train the model, later used sample for testing/validating.[57]

- Holdout: The data set is randomly separated into two sets, called the training set and the testing set. It is very simple and easy to implement. However, the evaluation may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made.
- K-fold cross validation: The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set k-1 times. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times.
- Leave-one-out cross validation: Leave-one-out is one extreme case of K-fold cross-validation where given a dataset of N instances, only one instance is left out as the validation set and training uses the N-1 instances. We then get N separate pairs by leaving out a different instance at each iteration.
- Stratified k-fold cross validation: In Stratified k-fold cross-validation, the dataset is partitioned into k groups or folds such that the validation data has an equal number of instances of target class label. This ensures that one particular class is not over present in the validation or train data especially when the dataset is imbalanced.

3.2.3. Time Series Forecasting

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs.

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks,

LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or intrusion detection systems.

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell as shown in 3.10.

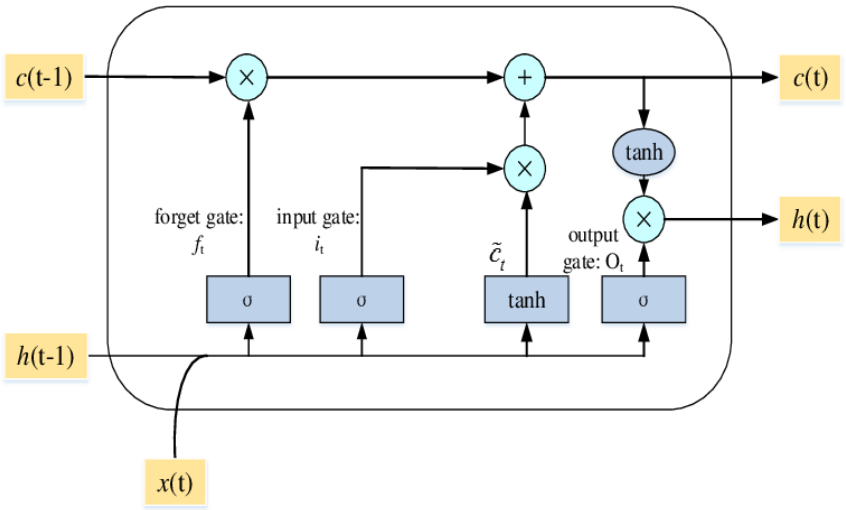

Figure 3.10: Structure of an LSTM cell [58]

The input gate decides what new information will be stored in the long-term memory. Previous hidden state and current input are passed into a sigmoid function. That decides which values will be updated by transforming the values to be between 0 and 1. Therefore, it has to filter out the information from these variables that are not useful.

The cell state takes the output from the input gate and do a pointwise addition which updates the cell state to new values that the neural network finds relevant. That gives new cell state.

Forget gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. The closer to 0 means to forget, and the closer to 1 means to keep.

Lastly the output gate decides what the next hidden state should be. First, previous hidden state and the current input are passed into a sigmoid function, then the newly modified cell state into the tanh function. We multiply the tanh output with the sigmoid output to decide what information the hidden state should carry. The output is the

hidden state. The new cell state and the new hidden is then carried over to the next time step.

The equations for the gates in LSTM are:

$$i_t = \sigma\left(w_i\left[h_{t-1}, x_t\right] + b_i\right) \tag{3.16}$$

$$f_t = \sigma\left(w_f\left[h_{t-1}, x_t\right] + b_f\right) \tag{3.17}$$

$$o_t = \sigma\left(w_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{3.18}$$

where i_t, f_t, o_t represent respectively input, forget and output gates. h_{t-1} is the output of the previous lstm block, x_t is current timestamp, w_x and b_x are weight and biases for the respective gates(x).

The equations for the cell state c_t, candidate cell state $\tilde{c}_t$ and the final output:

$$\tilde{c}_t = tanh\left(w_c\left[h_{t-1}, x_t\right] + b_c\right) \tag{3.19}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{3.20}$$

$$h_t = o_t * tanh\left(c_t\right) \tag{3.21}$$

notations are same as the previous formulation.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

4 | Problems Statement

An automatic framework to predict failures can increase the reliability of communication networks. In this work, we extract the related alarms groups from the alarm logs to determine different root-causes of hardware failures in the communication network.

Although each alarm in the data set contains many information, to reduce the complexity of working with a large data set, we can only consider the set of important features of these alarms determined by domain experts. There is a vast diversity in the information provided by these alarms. So, each required detail must be considered to analyze the data. This process should be done very carefully, a wrong selection can cause incorrect or missing root-cause patterns.

Using historical knowledge of alarms, a time series model can be implemented to predict future failures. This information can help to prevent or fix troubles corresponding to this predicted failure as quickly as possible.

In this chapter, we formalize the description of the three different problems that need to be solved to perform failure forecasting; namely, Failure-cause clustering, Failure-cause detection and identification, and Failure-cause forecasting itself. Before delving in the description of the three problems, we provide some information of the characteristics of the input data to our problems. The following figure 4.1 shows prediction underlying logic and problems interconnection.

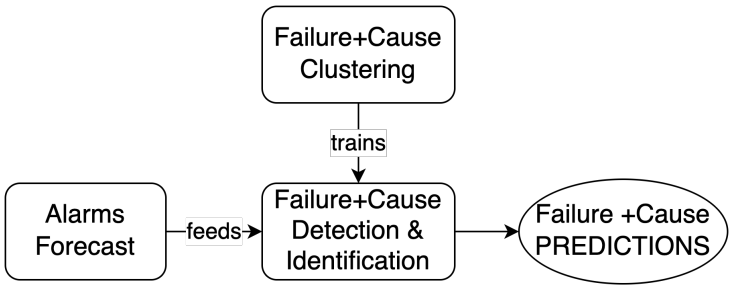

Figure 4.1: Functional prediction logic

4.1. Failure Cause Clustering

Since the available data provides information on alarms behavior, but not on specific failure-causes, which can be even unknown for network operators, the objective of this problem is to group data in order to extract significant information from alarms data and identify distinct failure-causes associated to alarms behavior, their frequency, correlation among different alarms, etc. To accomplish this task, we use ML clustering algorithms. Clustering algorithm takes in input $T_U = 163.174$ with-UAS windows data and divide them into clusters, which will then be associated to different labels identifying the various failure-causes, thus leading to a labelled dataset which will be used in a subsequent problem.

More formally,

- given unlabeled with-UAS windows as input X_U,
- the output of the clustering algorithm will be a set of clusters C.

Used implementation compares and updates different set and parameters to achieve more correlated and self-descriptive clusters and then evaluate the clustering solutions by means of numerical metrics like silhouette and inertia. These clusters are examined by domain experts and a label y is assigned to each cluster c through their ability in associating alarm patterns to failure-causes.

We evaluate the outputs of clustering by means of numerical metrics like silhouette and inertia and then they are validated by domain experts' ability in associating alarm patterns to failure-causes.

4.2. Alarms Forecasting

Failure prediction is carried out considering a time series dataset, which includes the evolution of the states of each alarm for each link, which we call *bit-sequence*. In this data type, we extract the alarm states which can be in ON or in OFF, with 1-second granularity for each link. The aim of this block is to predict next alarms statuses.

4.3. Failure Cause Detection and Identification

Once a labeled dataset is obtained, this is considered as a ground-truth dataset used to develop ML-based classification algorithms for the detection of forecasted failures and

identification of the forecasted failure-causes in microwave links. We use our labelled data from clustering step to solve a multi-class classification problem, where the classifiers take labeled data in input and are trained to learn a mapping between input features and the labels, i.e., the failure-causes. Then, the classifiers can be applied to new data points (i.e., microwave links observed in a certain window) to classify failure-causes based on the observed features. The aim of this analysis is to compare various classifiers in terms of accuracy and complexity, expressed in terms of algorithm training duration.

More formally, given in input a labeled dataset $[X_L, y]$, in which at each $x_{ftl} \in X_L$ related to a UAS-window and associated a label y, we provide as output a classifier able to take as input network measures X associated to failure events, and provides as output a predicted failure-cause y, maximizing its classification accuracy.

5 | Dataset description and pre-processing

In this chapter, we describe the datasets provided by SIAE Microelettronica that have been used for our analysis and we discuss only the characteristics of AGS20 equipment type.

5.1. Input Data

The input data consists of collected measurements and alarm logs which are used to analyze network failures, obtained by an Italian microwave network of 10841 radio links. Network measurements and plots are typically used for detecting propagation failures. In this work, we use alarm logs to determine hardware failures.

As mentioned in section 3.1.6, alarm is a message that reports the information when there is a problem on the link. All the arisen alarms are monitored and collected by a centralized system that creates a database, where the alarm logs include information regarding alarm name, equipment location (i.e., link ID and site), timestamp, equipment type and other information related to the faults. We analyze network alarms for one week period starting from the beginning of 27/09/2019 to the end of 4/10/2019.

We generate various datasets with different alarm structure, which are discussed in the following sections.

Alarm History - Bitsequence

Bitsequence dataset consists of a log with information on alarms ON/OFF status, also including reference equipment and site, and other information related to the alarm. This dataset is collected with a granularity of one second. Only some of the attributes that are useful for extracting alarm relations are considered, as described in the following:

- EQ_ADDRESS: IP address of the network element that compose a link. Each site

EQ_ADDRESS	EQ_MOC	TRAP_MSG	DATE_SET	DATE_CLEAR
10.191.121.7	29	ODU A Rx Active Status	27.02.2018 11:53	1.10.2019 22:20

Figure 5.1: Example log of Alarm History Bitsequence dataset

of network element has an IP address, this attribute helps to determine the site of the link.

- EQ_MOC: Equipment number of the hardware equipment. 29 refers to AGS20.
- TRAP_MSG: The alarm name which describes the type of failure for example: Rx/Tx power alarms, modulation related alarms, IDU/ODU alarms.
- DATE_SET: Timestamp of when alarm switches ON.
- DATE_CLEAR: Timestamp of when alarm switches OFF. If alarm is still ON when database examined, this value is set as "null" by the system.

As an example, we show in Fig. 5.1 one entry which indicates the date and time for switch-ON and Switch-OFF of an alarm named "ODU A Rx Active Status" on an equipment with type "29" (i.e., "AGS20") with IP address 10.191.121.7. The exact time when the alarm switched ON is mentioned in the DATE_SET column, and the exact time when the alarm goes OFF is mentioned in the DATE_CLEAR column. With these tow columns is possible to determine duration of the active alarm.

Alarm Statistics - Window

Alarm statistics windows includes all links and all the existing alarms in the system, computing occurrences with a 15 minutes granularity. More specifically, alarm statistics database shows how many times an alarms switches from OFF state to ON state in 15-minutes windows. The attributes that we consider are given below:

EQ_ADDRESS	DATE_	ALARM_DESCRIPTION	FREQUENCY
10.199.230.178	5.08.2019 00:00	Ethernet Traffic Concatenation fail on radio link	5

Figure 5.2: Example log of Alarm Statics file

- EQ_ADDRESS: The IP address of the network element
- DATE_: Starting time of the 15-minutes window
- ALARM_DESCRIPTION: The alarm name which describes the type of failure.

- FREQUENCY: It shows how many times the alarm given in column ALARM_DESCRIPTION is switched ON in the 15-minutes slot identified in DATE_.

In Fig 5.2, we show an example where alarm "Ethernet Traffic Concatenation fail on radio link" was switched-ON 5 times from 5.08.2019 00:00 to 5.08.2019 00:15 on the link that has an IP address 10.199.230.178. Note that, information provided by alarm statistics database does not provide any indication on whether the alarm is ON at the beginning or at the end of the 15-minutes window, but provides how many time an alarm occur in that specific window.

Alarm Superset

Alarm Superset consists in all available alarm names separately for every equipment type in the network. It provides consideration of all alarms even though some rare alarms which never occur in the considered data period.

This alarm set is reduced during the work to a limited number of alarms handpicked by SIAE domain experts and it is used to filter irrelevant alarms for our fault identification problem.

Table 5.1 shows the number of alarms for each equipment type in the corresponding supersets:

	Number of alarms	**Number of links**
AGS20	231	1025

Table 5.1: Statistics of equipments

A network link consists of two network elements which are at two different locations, namely, site A and site B. Therefore, for every alarm, there exist two distinct alarms of type 'Alarm_A' and 'Alarm_B'. So considering both sites of a microwave link, the number of alarms is doubled.

Windows with at least one UAS are define with-UAS windows, they are relevant since UAS is the main cause of the unavailability of the network. So they are calculated from the total number of windows for the different equipment types. The with-UAS windows for AGS20 equipment type are 49899 compared to a total number of 787200 windows which is 6.34%.

5.2. Alarms Analysis

Alarms analysis aims at reducing the amount of features and consequently the complexity of the developed ML algorithms, so alarms occurrences and alarms correlation analysis are performed as described in the following sections.

Occurrence Analysis

Now we evaluate, for all the links in our dataset, the number of times each alarm occurs for the considered time period. Indeed, if any alarm occurred in all the (or a very large percentage of) windows or, alternatively in none (or a very small percentage of) windows, we would conclude that such alarm does not provide significant information and thus we would discard it from our alarm set.

Highly occurring alarms can be discarded because they will appear in all the alarm lists of clusters. So we remove very rarely occurring alarms since they are just causing computation and time complexity. Finally we can discard alarms which are not related to hardware problems which are defined by cooperated company, since we only focus hardware failures in this work. Eventually a list of relevant alarms to be considered as input of developed workflow is validated and given by SIAE.

Correlation Analysis

The objective of correlating alarms occurrence is to remove redundant alarms. As an example, if two given alarms always occur together, including both alarms in the features set may introduce redundancy and so add unnecessary computational complexity.

The correlation coefficient is a statistical measure of the strength of the relationship between two vectors. A positive correlation means that, the variables move together by the same direction and negative correlation shows they move by the opposite directions. This analysis can be used for extract the positively related alarms. Given two vectors X and Y, which in our case represent alarm vectors in UAS-windows.

If correlation coefficient equals to 1 for two given alarms, this means that these two alarms are always in ON and OFF states in the same 15-minutes windows. By using this correlation, instead of using highly related alarms together, they can be decreased to one alarm and others can be neglected. SIAE domain experts choose which alarm is used to represent these set of highly related alarms, in some cases they can choose to use all/some of the set.

Correlation coefficient lower than 1 identifies pairs of alarms where two alarms are in opposite states in at least one windows. We still interested in positively related alarms, where one of the alarm is in the same at least selected percentage of the other alarm state. For example, considering alarms X and Y which occur simultaneously in 1000 windows, and Alarm Y occur in additional 100 windows. In general, a choice shall be made to select the value of correlation above which we can consider that two alarms provide redundant information. In this analysis, again SIAE domain experts choose which alarm is used to represent these set of related alarms.

5.3. Data Wrangling

Any intelligent system basically consists of an end-to-end pipeline starting from pre-process raw data in order to process and select relevant features and attributes from this data. Then, data is used to feed statistical or machine learning models which extract useful information in an unsupervised and/or supervised manner. Finally, the developed models can be deployed in a real-field environment to solve the problem at hand.

Even though there are a lot of newer methodologies coming in, like deep learning and meta-heuristics which aid in automated machine learning, each problem is domain specific, and selecting the set of features to use is often a decisive factor impacting the performance of the system.

The input data used in our work are obtained on an real microwave network and consists of SIAE Microelettronica equipment. As the nature of the collected alarm dataset are not suitable to be handle directly by machine learning algorithms, we perform data preprocessing before using them in our analysis. In the following, we detail the various data preprocessing steps which have been taken before developing ML algorithms for prediction, clustering and classification of failures in microwave networks.

Data Cleaning

Raw data includes all the alarms information, for both with-UAS and without-UAS windows on more than 10000 links, and consists of around 2076654 kbytes of Alarm History data and 515072 kbytes of Alarm Stats data. These datasets are used in our analysis but need to be changed in size in order to reduce computational complexity, so data cleaning procedures is applied to raw data.

We consider one week's data from beginning of 27 September 2019 to end of 04 October 2019.

First, since alarm data report a lot of unnecessary information beside the important ones, we discarded all the unnecessary information in the alarm data.

Then each bitsequence is statistically analyzed to have unbalance measurements. Links with low number of alarmed seconds are eliminated and just relevant alarms identified by iterations with SIAE are took into consideration.

Subsequently, data is split into training, validation and test sets with ratio 0.7, 0.2 and 0.1, respectively, in order to optimize ML models hyperparameters.

Finally, data is sliced into time slices format to feed the deep learning forecasting model.

Windows Formation

Information about which 15-minutes windows are with-UAS or without-UAS with the corresponding link ID are provided besides the alarm dataset. Recall that "Unavailable seconds (UAS)" metric is defined as the number of seconds when the system is unavailable in a given measurement interval. In this work we refer to failures considering the with-UAS windows, hence we discard the without-UAS windows from our analysis. Table 5.2 shows the number of windows for both equipment types. It can be seen that majority of the windows are without-UAS, elimination of these windows left us only failure related windows.

	AGS20
With UAS	49899
With and Without UAS	787200

Table 5.2: Windows Information for Equipment Types

As shown in Fig. 5.3, we divided the set of all 15-minutes windows into various groups; first we identify with-UAS windows and without-UAS windows. The UAS windows are further divided into "transitory" and "non-transitory" windows. Transitory windows are the with-UAS windows following a without-UAS windows. The necessity of transitory window is observing how the alarms status change as soon as UAS occur. Transitory windows can be seen as starting time of the failures and can help to see the differences between various types of failures.

In Tab. 5.3, we show an example of several windows, also detailing transitory and non-transitory UAS windows.

All the windows are on the same link. Window 10/30/2019 12:45 is a with-UAS window, while the previous window 10/30/2019 12:30 is without-UAS window, a with-UAS

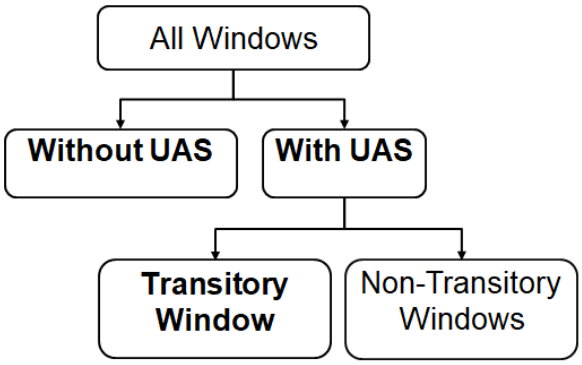

Figure 5.3: Division of 15-minutes windows

Without-UAS - 10/30/2019 12:30	
With UAS - 10/30/2019 12:45	Transitory window
With-UAS - 10/30/2019 13:00	Non-Transitory window
With-UAS - 10/30/2019 13:15	Non-Transitory window

Table 5.3: Transitory windows example

windows is following a without-UAS window, therefore, window 10/30/2019 12:45 is a transitory window. However, the last two windows are non-transitory windows because they are all preceded by other with-UAS windows.

Occurrence Metrics

Alarms have various behaviours, for instance some of them can stay in ON state for multiple windows while others can go in OFF state after few seconds. These behaviours should be extracted from the input datasets to identify associated alarms. For this reason, we create and compare different occurrence metrics.

In our work, we define the following occurrences of an alarm:

- Binary value representing if an alarm is ON state in the window as m_1
- Number of seconds an alarm is in ON state in the window, represented by an integer number as m_2
- Number of times an alarm switches to ON state in the window, represented by an integer number as m_3

We consider both alarm history and alarm stats datasets for extracting occurrence metrics. Alarm stats provide the frequency of the all the alarms for each 15-minutes windows which is basically same metric with the m_3. However, no information is included on how

much time the alarms stays in ON state. However all the occurrence metrics can be extracted from history log dataset because this dataset is recorded with a granularity of one second. It gives information about when an alarm switches to ON and OFF state and how many seconds stays in ON state. Disadvantages of using history log dataset are the computational and time complexity.

The following two illustrations explain how occurrence metrics are extracted from datasets. The illustration in Fig. 5.4 shows an alarm that goes in ON state only once (up arrow) in a 15-minutes window. Therefore the frequency value is 1. If the alarm goes ON, goes OFF, and again goes ON in a 15-minutes window then the frequency value is 2. This alarm stays ON more than one windows but the value of subsequent windows is 0 because alarm doesn't goes ON in these windows.

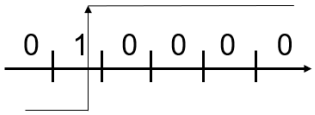

Figure 5.4: Illustration of computing occurrence metric m_3

Fig. 5.5 depicts an alarm this alarm is in ON state and in how many windows an alarm is present (ON). Unlike the previous metric, the value of subsequent windows is not 0. If an alarm is present in a window, its value is 1 for binary metric, but for integer metric, how many seconds the alarm is on should be calculated which can be up to 900 seconds. In the case an alarm switched in a window multiple times, each period of seconds should be added up.

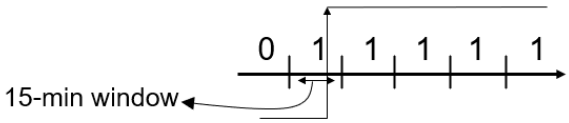

Figure 5.5: Illustration of computing occurrence metric m_2

Considering only one of the occurrence metrics as a feature can cause loss of information about alarm behaviour.

m_3 is used, it cannot provide any information about if the alarm is permanent or transient: two different alarms can switch to ON state once in the same window but one of them can stay in ON state for an entire window, while other one can go off after few seconds.

m_2 cannot give any information about when alarm switched on and how frequent these switches. The problem is that two different alarms can stay on for the same amount of

time, but one alarm can switch ON and OFF many times in the same window.

m_1 is a binary feature, it does not provide any behavioural value on alarms, it is a state variable representing alarm presence in the window. We use it for bitsequence forecasting.

5.4. Data Output

Our work takes in input data collected from network equipments, it is then transformed and at the end a dataset is released. This dataset has for all scenario the same structure and it consists of windows as previously defined with an additional feature, fault label, identifying fault for each window.

6 | Machine-Learning Failure Prediction

This chapter details the methodological approaches and algorithms used to perform fault prediction in microwave networks. In the previous section, we explained the pre-processing performed on raw input data, which is necessary to develop data augmentation, forecasting, detection and identification using ML. In the following sections we discuss more in detail data augmentation problem to better exploit not-labelled retrieved data. Subsequently we describe two approaches: first we propose a multi-step prediction framework composted by a predictive deep learning model applied to bitsequence time-series to forecast alarm statuses in the future, then failure detection system and fault detection module using machine-learning ensembles is described. Second, we propose a single-step approach for direct fault prediction via machine learning techniques. The overall system is described by the following diagram 6.1, describing the logic workflow of our work in a holistic way.

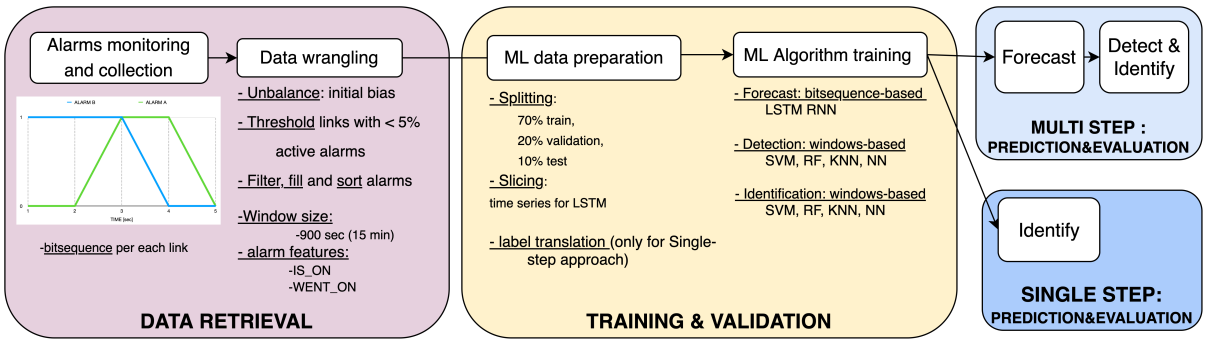

Figure 6.1: Prediction Workflow

1. Data Retrieval phase includes all operation described in previous section, related to data gathering, analysis and wrangling, as describe into section 5.
2. Training and Validation phase considers machine learning operations as model definition, training and validation, including strictly related input data preparation.
3. Prediction is the final phase, where two approach are available: either multi-step or

single-step: the first composed of forecast, detection and identification; the latter just of identification.

6.1. Data Augmentation

The objective of clustering alarms windows is to identify proper number and groups of hardware failures, based on alarm information. It is worth noting that some types of hardware failures are very-well-known as they often occur in a microwave link (e.g., power supplies, overheating) However, a number of unknown failures can be also present, e.g., due to lack of historical information on equipment failures, changing system conditions which lead to unseen equipment behavior, etc. This aspect makes the total number of different failure causes an output of our problem, which we tackle through clustering approach, using a previously developed implementation and in which we evaluate different clustering solutions in terms of inertia and silhouette. Nonetheless, besides a quantitative numerical assessment, we also evaluate the different clustering solutions by analyzing several elements for each of the obtained clusters (i.e., several windows and the corresponding set of features for each cluster) with the help of SIAE domain experts.

One of the main challenges in this process is finding the most reasonable number of clusters identifying different hardware failures. To evaluate the number of clusters, we first evaluate inertia for different cluster numbers. Elbow method applied to the values of inertia obtained for increasing number of clusters is generally a good solution for choosing the best number of clusters. With such a method, based on a predefined threshold T [%], the optimal number of clusters N is the one for which inertia is more than T% lower than case with N-1 clusters. However, in order to avoid neglecting some possibly-relevant failure causes, instead of defining an arbitrary numerical threshold T on inertia values, with the help of SIAE domain experts we perform a detailed inspection of patterns of alarms occurring in the various clusters for different clustering solutions, considering the cases where the number of clusters ranges between 2 and 40.

As a complete evaluation of all the elements in each cluster and for each of the clustering solutions would be time consuming, for each cluster, we only observe, 10 data points randomly picked within the cluster. Besides the 10 random points, we also include in our inspection the medoids for all the clusters, which represent the real points in our dataset which are closest to the cluster centroids.

We used as clustering algorithm K-means. K-means algorithm is tested with different values of K, i.e., different values for clusters number which is an input of the k-means

algorithm to be selected a-priori. Inertia and silhouette plots are used to compare different alarm sets.

After dividing the data into clusters with proper alarm set and parameters/algorithm, human experts label each cluster to a root-causes. These labeled windows associated with a hardware failure are used as input of a classifier which provides the root cause that provides the specific failure.

Results relative to this phase can be found in Section 7.1.

6.2. Short-Term Multi-Step Prediction

In this approach prediction of future fault root-causes is divided into 3 sequential modules: alarm forecasting, failure detection and fault identification.

Figure 6.2 shows the actual high level workflow.

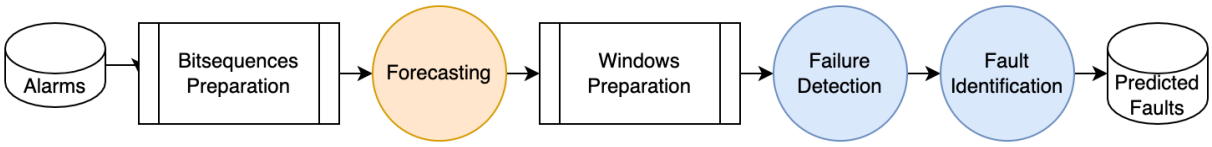

Figure 6.2: Multi-step Prediction Workflow

1. Alarms are collected and organized in a bitsequence per each link;
2. Bitsequence are prepared for forecasting: data is splitted into training, validation and testing set, as defined into following section 7.2 and after that each dataset is prepared for LSTM.
3. Alarms forecast is performed by baseline models and LSTM as described in Section 6.2.1;
4. Windows are prepared from predicted alarms bitsequence as described in Section 5.3
5. Failure detection and identification are performed on windows with classification machine learning models, as defined in Section 6.2.2.

6.2.1. Alarms Forecasting

For alarm we develop a LSTM model to predict future alarms based on alarm bitsequences.

As discussed in Section 5.1, first we generate bit-sequences for each link. We can extract different size of alarm sequences to use as input from these data. Then we divide the bit-sequences for training, validation and test sets.

Long Short Term Memory Forecaster

For training and testing the LSTM network, datasets are divided into slices. First the length of input T_i and output sequences T_o of the model should be chosen. Then LSTM structures takes these both an input sequences $X_{t:(t+Ti)}$ and adjust its weights according to output sequences $X_{(t+Ti):(t+Ti+To)}$. Then same bit sequence is used with 1 time step sliding $t+1$. After training of model, the same approach is used in testing, but instead of adjusting weights, it predicts the value and compares with the output sequence.

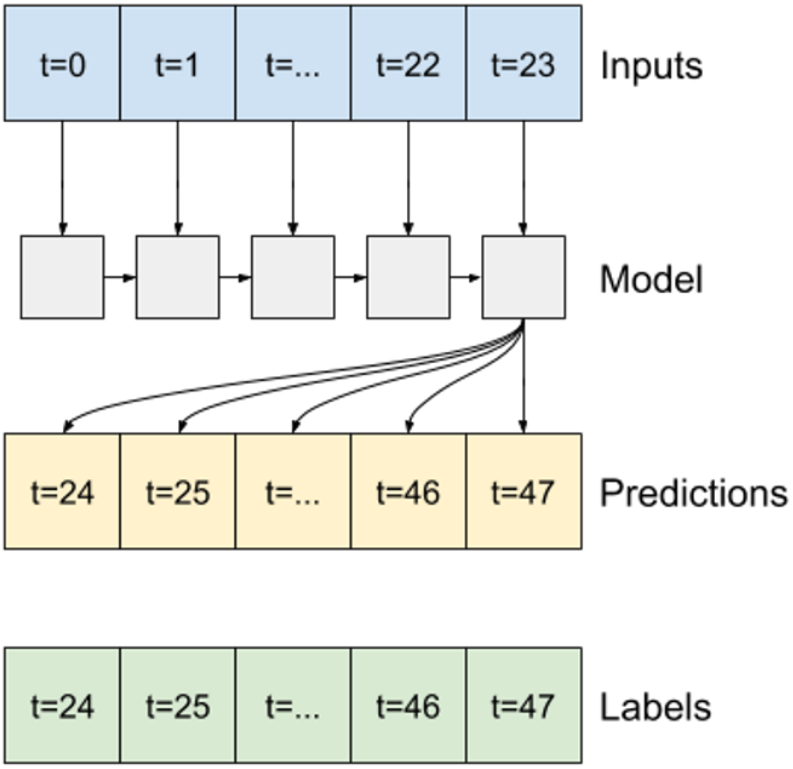

Figure 6.3: LSTM scheme

Then LSTM network should be trained with a percent of these data, and rest of the data is used for testing the performance of model in terms of accuracy, precision, recall and time. Generally speaking, high amount of data provide better result in terms of accuracy but it needs more time for both training and testing parts.

We build our model as multi-multi structure, which takes alarm sequences of multiple

alarms as input and produce alarm sequences of multiple alarms as output. After some comparison of different activation functions such as sigmoid and relu, different number of LSTM units in hidden layer and different input and output length of alarm sequences in terms of training/testing time and accuracy, we train a LSTM model iteratively on each link. In order to avoid overfitting, the earlystopping method is used. Eventually model's parameter are picked as follows:

- LSTM Input layer: time slice input length units (INPUT-STEPS);
- Dense Output layer: time slice output length units using sigmoid activation function;
- Loss function: Binary Cross Entropy;
- Optimizer: Adam;
- Metrics: TruePositives, TrueNegatives, FalsePositives, FalseNegatives, Accuracy, Recall, Precision, AUC and PRC;
- Earlystopping enabled monitoring validation PRC increments, patience = 10
- Single epoch.

Probabilistic Moving Average Forecaster

We developed two baseline models to forecast alarms based on the probability of having an active alarm.

First we forecast using each second of the prediction interval using the positive ratio of all the dataset with the following values:

$$\begin{cases} p_1 = \dfrac{positive[all\ data]}{total[all\ data]}; & (6.1) \\ p_0 = 1 - p_1. & (6.2) \end{cases}$$

Then, we developed an evolved version, using linear combination of the overall positive ratio and the positive ratio of a configurable moving window, considering just last time period with size equal to prediction interval input dimension (10 seconds in our scenario). The following formulas are used:

$$\begin{cases} p_1 = \alpha \cdot \dfrac{positive[all\ data]}{total[all\ data]} + \beta \cdot \dfrac{positive[mov\ win]}{total[mov\ win]}; & (6.3) \\ p_0 = 1 - p_1. & (6.4) \end{cases}$$

Parameters α *and* β could be set to optimize model performance and for our work has been set respectively to 0.5 and 0.1.

6.2.2. Fault Detection and Identification

In this section we focus on the usage of supervised algorithms first for detection of failure windows and after for microwave link failures' root-causes, reusing and adapting a pre-existing implementation that uses our labelled data to solve respectively a single-class and a multi-class classification problem, so the aim of this analysis is to compare various classifiers in terms of accuracy, precision, recall and complexity, expressed as execution duration. So, we develop a failure detector, able to classify if new windows have failure. Then in the input data, each window represents one specific root-cause of the failures out of more than two root-causes. Therefore, in our work the failure identification classifier is a multi-class classifier. Finally, we provide a failure identificator trained for multi-class failure root-cause labeling.

Classifiers are built considering different classification algorithms, namely, SVM, RF, KNN and ANN. For each algorithm, various combinations of hyperparameters are validated to obtain the classifier with highest accuracy. In order to generalize results, k-fold cross validation is performed.

The different algorithms evaluated have different tunable hyperparameters with specific range of values, as detailed in Tab. 6.1

Algorithm	**Parameter**	**Set**
SVM	Regularization parameter	1, 10, 100
	Kernel	rbf, polynomial
	Kernel coefficient	1, 0.1, 0.01
	Decision function	ovo, ovr
RF	Number of trees	3, 4, 5, 10
	Maximum tree length	3,5,10
	Minimum number of split	2,3,5
KNN	Number of neighbors	5,20,50
	Weight function	uniform, distance
	Distance function	manhattan, euclidean
	Searching algorithm	ball_tree, kd_tree, brute
ANN	Number of hidden neuron	3,4,5
	Activation function	linear, sigmoid, tanh, relu

Table 6.1: Hyper-parameters for classifiers

where RBF stands for Radial Basis Function, tanh and relu stand for respectively hyper-

bolic tangent and rectified linear activation function. For ANN, layer size is 3 for all the cases.

K-fold Cross validation

To compare the classification algorithms nested k-fold cross validation is performed. Outer k-fold divide the dataset to k parts. Iteratively, each fold chosen as test set and the rest as train set. For each train set inner k-fold again divide the train set to k folds and use that dataset to find the best hyper-parameters from provided set of parameters. Then outer k-fold take average of all the folds and provide an accuracy of classifiers. Figure 6.4 shows the principle of K-fold Cross validation.

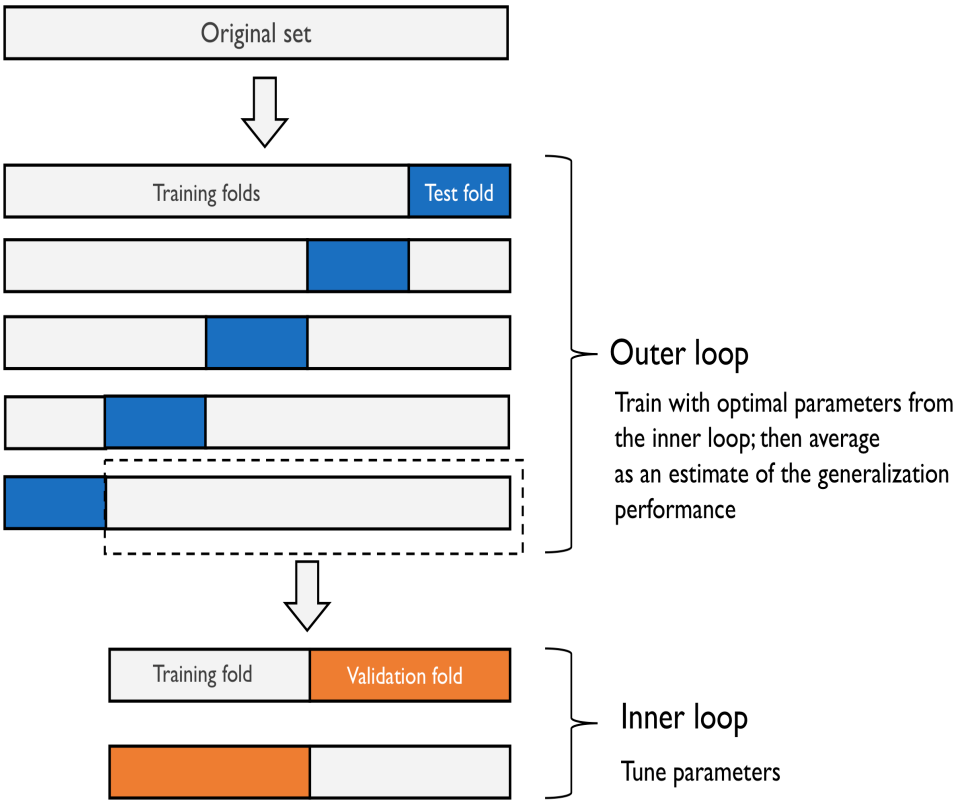

Figure 6.4: Structure of k-fold cross validation

6.3. Long-Term Single-Step Prediction

We propose another approach to perform failure root-cause identification leveraging on data structure characterization. Figure 6.5 shows its workflow diagram.

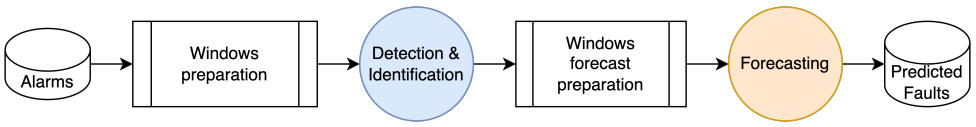

Figure 6.5: Single-step prediction workflow

- Applying data augmentation as describe in section 6.1 to collected windows including no-failure case creating as output a windows dataset with either relative failure root-causes or no-failure.
- Following, future fault labels are attached to each windows considering prediction interval parameter. e.g.: windows 00:00 - 00:15 will have fault label of window 01:00 - 01:15. *1 hour prediction consists of 4 steps traslation, since 1 hour is 4 windows.*
- Finally, forecasting is performed by training models for fault identification as defined in section 6.2.2, so the resulting models could be used for future prediction of faults, predicting fault labels for unlabelled windows.

7 | Results

In this chapter, we provide quantitative analysis of the windows clustering process, aiming at identifying distinct hardware failure causes within the unlabeled data. After, alarms bitsequence forecast is computed using LSTM model and performance metrics are compared with baseline models. Then, we compare performance of various classifiers performing failure cause identification.

Our experiments were conducted on a PC with Processor: Intel Core i7 processor and 16 GB RAM.

In the preprocessing, initially we perform data cleaning. Then different occurrence metrics and features are extracted from input datasets for using in the next steps. Moreover we formed the windows as with-UAS and without-UAS windows.

We start the analysis by defining two baseline sets of alarms according to detailed investigation about alarm occurrences, namely Set 1 and Set 2. Set 1 generated by discarding of alarms only occur in without-UAS windows or are in ON state more than 90% of the total windows, Set 2 is generated from Set 1 with discarding alarms which our more in without-UAS windows than with-UAS windows. Respectively they consist 191 and 149 alarms.

After performing correlation analysis on both occurrence identifiers, i.e., m_2, and m_3 which are defined in section 6.1, we obtain further smaller sets to utilize in the clustering process. First we discard the alarms which have 100% correlation with at least another alarm. Applying this alarms removal to Set 1 and Set 2, we obtain Set 1.A and Set 2.A, respectively. Then we further discard alarms which have more than 90% correlation with at least another alarm and generate Set 1.B and Set 2.B, respectively. In Fig. 7.1, initial sets and their number of alarms for AGS20 can be seen.

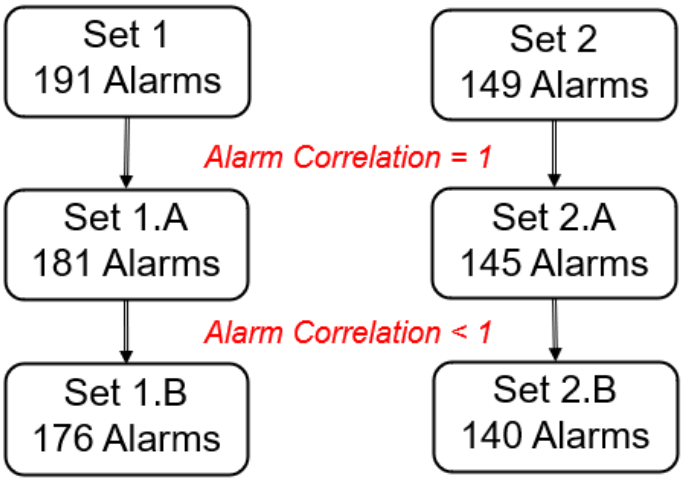

Figure 7.1: Initial AGS20 alarms sets used for the different clustering cases

7.1. Data Augmentation

Augmentation is performed with different feature selections seen in Fig. 7.1. This selection is used to cluster alarm sets for both equipment types. Results are interpreted with inertia, silhouette and feedback from SIAE domain experts.

K-Means

We test kMeans considering a number of clusters ranging from 2 to 40 and show the values of inertia for the different cases in Fig 7.2 in order to define a reasonable number of clusters that identify the different failure-causes.

As expected, increasing number of clusters provides lower values of inertia.

Clustering starts with initial sets (i.e., Set 1 and Set 2) that mentioned in the previous subsection. Same is done for all 6 sets.

There is no notable difference between the different cases in terms of inertia and silhouette. Moreover,for a given number of clusters, alarm sets provide very similar/exact set of medoids in the same number of clusters compared to the other sets. After manual analysis by SIAE domain experts, if an insignificant alarms is suggested by experts, these alarms are discarded or if there is an alarm that helps to find patterns more easily, they are added and new sets are generated, so a new set with 86 alarms is generated.

In conclusion of this analysis, using k-means algorithm, considering clustering solutions up to 40 clusters and after interaction with SIAE domain experts, clusters were labeled

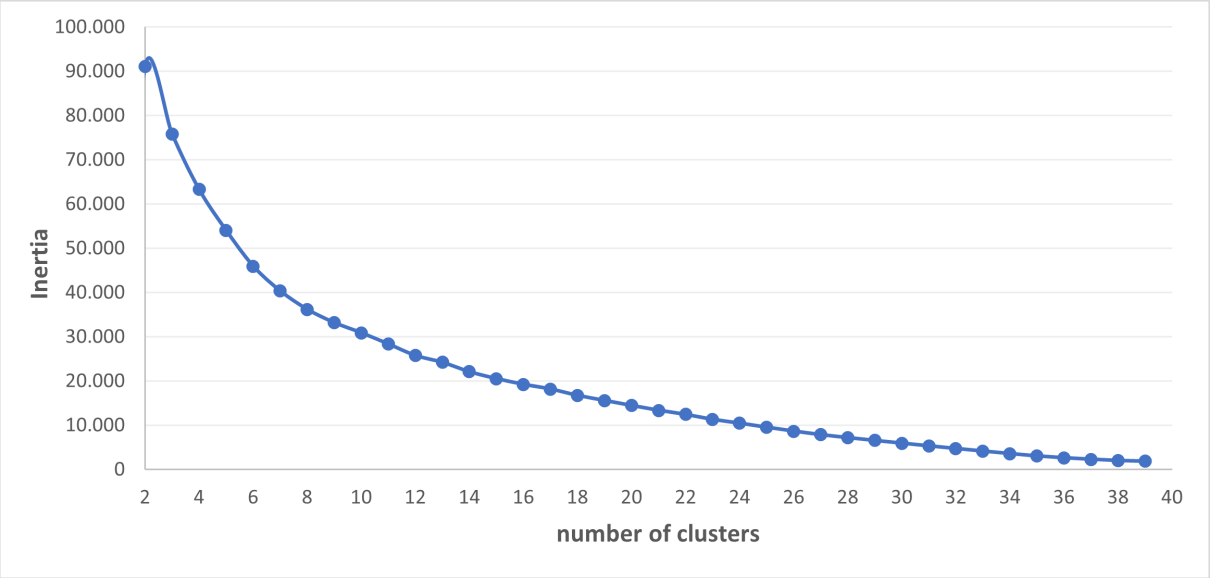

Figure 7.2: Values of inertia obtained for different number of clusters with K-means algorithm using Set 1 for AGS20

using the most frequent alarm patterns in the medoids. As a result, 35 different root-causes are found by this analysis. Figure 7.3 shows the corresponding labels used for the final clustering for AGS20 equipment type.

7.2. Short-Term Multi-Step Prediction

7.2.1. Bitsequence Alarms Forecasting

Different metrics are computed: TruePositives, TrueNegatives, FalsePositives, FalseNegatives, Accuracy, Recall, Precision, AUC and PRC. Using earlystopping which monitors val. PRC increments with patience = 10 to avoid overfitting. LSTM model is trained iteratively with each link with just one epoch.

We divided sequentially bitsequence data to 70% as training set and 20% as validation and 10% as test.

Comparing different time length prediction is not significant for models comparison since model performs extremely well for considered scenario. We can highlight a decrease of performance moving from 2 seconds prediction (considered just for theoretic purpose, and not contemplated in other graphs since irrelevant to final objective) to 10 seconds, while increasing the time horizon till the limit case of 120s does not show any variation on forecast performance, as is shown in figure 7.4

Medoid Labelling		Medoid Description				
Medoid #	Label Proposal	FMP Active	Macro-category #	Sub-category #	Branch #	Site ID
1	HW FAULT BRANCH1 SITE-A/B	NO	2	2b	1	A or B
2	IDU FMP FAULT BRANCH2 SITE-A/B	YES	2	2a	2	A or B
3	IDU FMP FAULT BRANCH2 SITE-B	YES	1	1c	2	B
4	IDU FMP FAULT BRANCH1 SITE-A/B	YES	2	1	1	A or B
5	ODU TX-OFF BRANCH1 SITE-B	NO	6	1	1	B
6	IDU FMP FAULT BRANCH2 SITE-B	YES	1	1b	2	B
7	IF CABLE/ODU FAULT BRANCH1 SITE-B	NO	3	1a	1	B
8	IDU FMP FAULT BRANCH1 SITE-B	YES	1	2	1	B
9	IDU FMP FAULT BRANCH1 SITE-A	YES	1	3a	1	A
10	HW FAULT BRANCH2 SITE-A/B	NO	2	1	2	A or B
11	IDU FMP FAULT BRANCH2 SITE-B	YES	1	1d	2	B
12	IF CABLE/ODU FAULT BRANCH2 SITE-B	YES	3	3	2	B
13	IDU FAULT SITE-B (FAR END)	NO	4	1	1 and 2	B or LINK
14	HW FAULT BRANCH1 SITE-A/B	NO	2	2a	1	A or B
15	IF CABLE/ODU FAULT BRANCH2 SITE-A	YES	3	2	2	A
16	ODU TX-OFF BRANCH2 SITE-A	NO	6	2	2	A
17	IF CABLE/ODU FAULT BRANCH1 SITE-B	YES	3	1b	1	B
18	HW FAULT BRANCH2 SITE-A	NO	1	3	2	A
19	ODU FAULT BRANCH1 SITE-B	NO	5	2a	1	B
20	ODU FAULT BRANCH1 SITE-B	NO	5	2b	1	B
21	HW FAULT BRANCH1 SITE-B	NO	1	2	1	B
22	MANUAL OPERATION	---	1	1	---	---
23	IDU FAULT SITE-A (FAR END)	NO	4	2	1 and 2	A or LINK
24	ODU FAULT BRANCH2 SITE-A	YES	4	1	2	A
25	HW FAULT BRANCH1 SITE-A	NO	1	1	1	A
26	ODU FAULT BRANCH2 SITE-A	NO	5	1	2	A
27	ODU TX-OFF BRANCH1 SITE-A/B	NO	7	1	1	A or B
28	IDU FMP FAULT BRANCH2 SITE-A/B	YES	2	2b	2	A or B
29	IDU FMP FAULT BRANCH2 SITE-B	YES	1	1e	2	B
30	FLEETING/TRANSIENT HW FAULT	---	2	1	---	---
31	IDU FMP FAULT BRANCH2 SITE-A/B	YES	2	2d	2	A or B
32	IDU FMP FAULT BRANCH1 SITE-A	YES	1	3b	1	A
33	IDU FMP FAULT BRANCH2 SITE-A/B	YES	2	2c	2	A or B
34	IDU FAULT SITE-B (FAR-END)	YES	5	1	1 and 2	B
35	IDU FMP FAULT BRANCH2 SITE-B	YES	1	1a	2	B

Figure 7.3: Failure-causes classes for AGS20 obtained after clustering process

So a performance analysis has been performed using a couple of baseline forecasters. First using a naive model and second with a more evoluted probabilistic model with a linear combination of the overall bitsequence unbalance value and the negative ratio of the last time interval, as described in section 6.2.1. The accuracy, precision and recall are compared to LSTM performance in Fig 7.5 showing a great increase in performance value with LSTM against baselines. Metrics are high for LSTM with an high increase compared to baselines. Moreover a boxplots in Fig. 7.6, show the distributions of these metrics, across all microwave links, resulting in a better adherence of LSTM to each link.

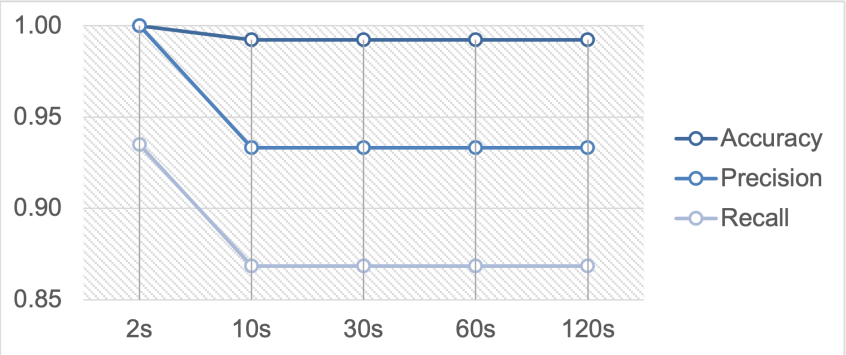

Figure 7.4: LSTM performance varying prediction horizon

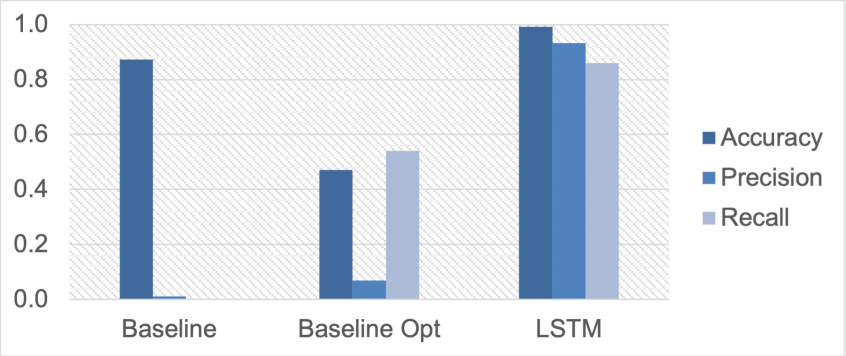

Figure 7.5: Forecast Performance Comparison

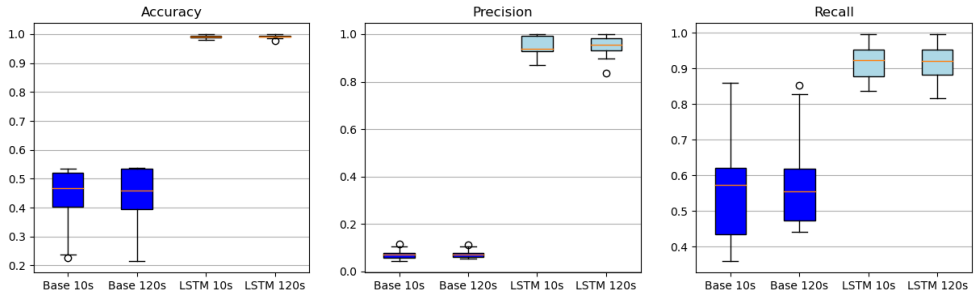

Figure 7.6: LSTM Box Plot

Complexity is linear with regard to bitsequence slice dimensions. Execution times are presented in figure 7.7 .

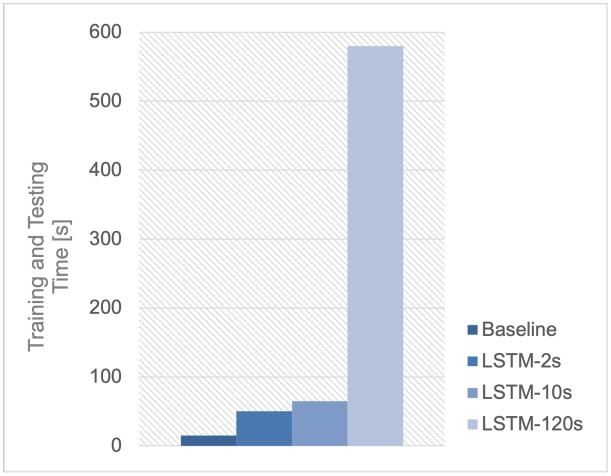

Figure 7.7: Forecast execution time including training and testing

7.2.2. Failure-cause Detection and Identification

The goal of failure-cause detection and identification is first to discriminate between failure and no failure, then the 35 classes which have been identified in the previous section and which refer to 35 distinct failure-causes based on alarms data. We evaluate the performance of four machine learning algorithms, i.e., SVM, RF, KNN and ANN, with the aim of identifying which of them is the most suitable to perform failure-cause identification, in terms of metrics and execution time, considering training and testing time.

The model selection for SVM, RF, KNN and ANN, provide as a result the hyperparameters presented in tables 7.1, 7.2, 7.3 and 7.4 respectively.

Parameter	Value
Regularization parameter	100
Kernel	RBF
Kernel coefficient	0.01
Decision function	One-versus-rest

Table 7.1: Hyperparameters selected for SVM algoritm

Parameter	Value
Number of trees	10
Maximum tree depth	10
Minimum number of split	2

Table 7.2: Hyperparameters selected for RF algoritm

Parameter	Value
Number of neighbors	5
Weight function	Distance
Distance function	Manhattan
Searching algortihm	Kd-tree

Table 7.3: Hyperparameters selected for KNN algoritm

Parameter	Value
Number of hidden neuron	5
Activation function	Linear

Table 7.4: Hyperparameters selected for ANN algoritm

The accuracy, precision and recall for detection classifiers for both scenario are plotted in Fig. 7.8. SVM, RF, KNN and ANN have extremely high metrics over 95%. Farther, all classifiers provide results close to each other.

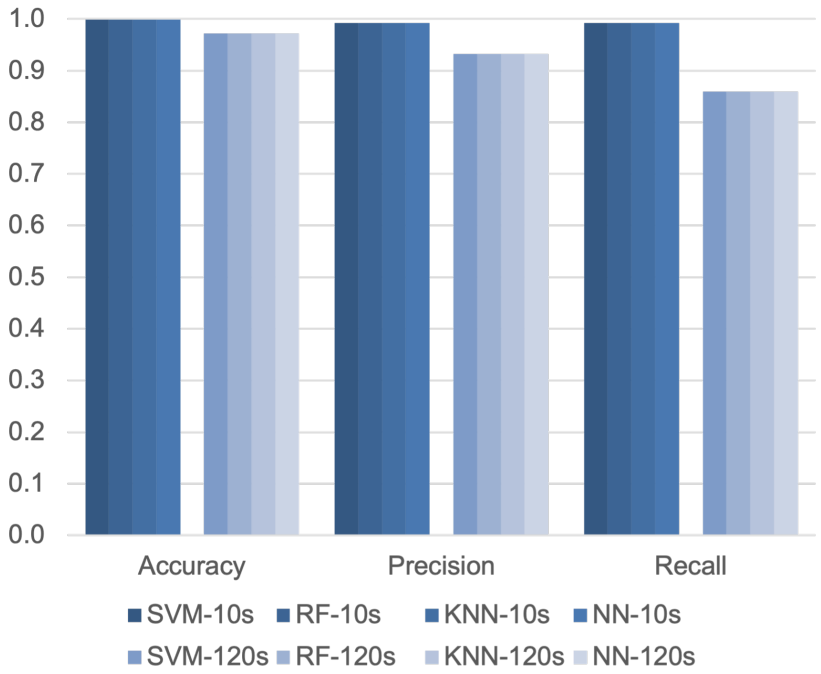

Figure 7.8: Comparison of detection classifiers

Execution times considering both training and prediction are plotted on Fig. 7.9. Here KNN shows the lowest execution time, as expected, considering that no training is performed by the algorithm, followed by RF, while at relevant distance we have SVM and at last NN.

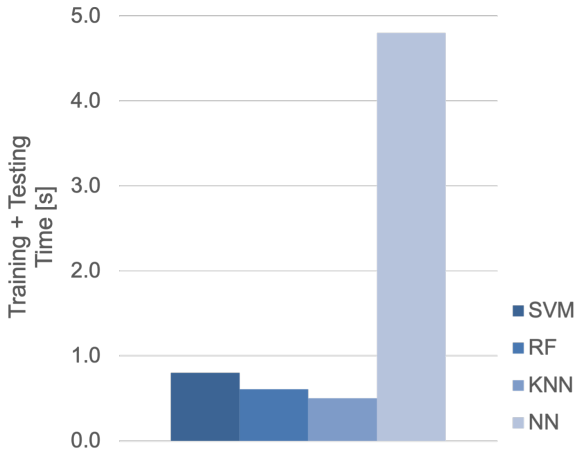

Figure 7.9: Detection classifiers ex. time

Next identification module is considered with accuracy, precision and recall for classifiers and both scenario plotted in Fig. 7.10. Here too SVM, RF, KNN and ANN have extremely high metrics over 95%. Farther, all classifiers provide results close to each other.

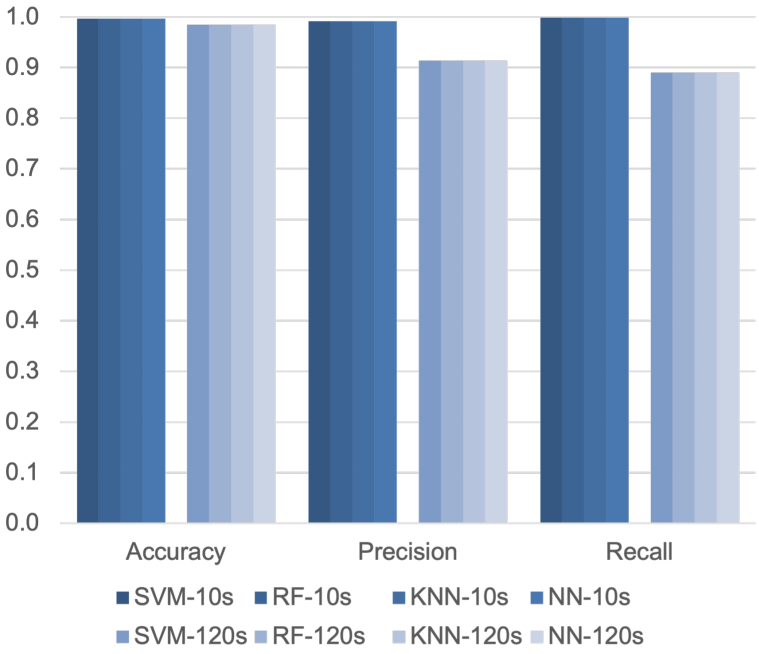

Figure 7.10: Comparison of identification classifiers

Execution times considering both training and prediction are plotted on Fig. 7.11. Here KNN shows the lowest execution time, as expected, considering that no training is performed by the algorithm, followed by RF, third is at SVM and at last with remarkable delay is NN.

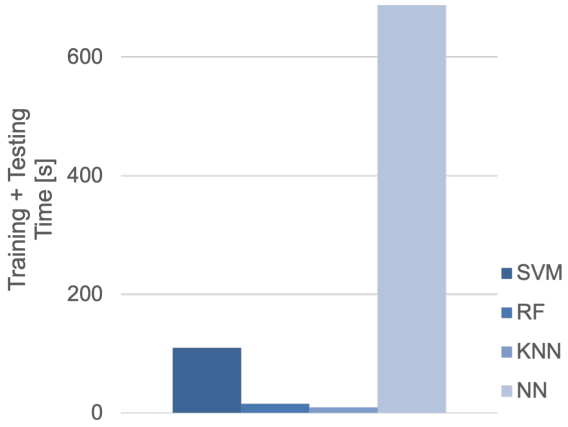

Figure 7.11: Identification classifiers ex. time

7.3. Long-Term Single-Step Prediction

This approach is tested using the multi step structure previously described in 7.2.2.

Three prediction scenarios are compared with respect to prediction interval: 1 hour, 24 hours and 48 hours.

Accuracy, precision and recall are respectively plotted in Figures 7.12, 7.13, 7.14 pointing out a general trend of an accuracy that is inversely proportional to prediction interval. A similar pattern appears for all metrics, showing off similar results for all models except in the 48 hours scenario where KNN performance plumbs of 0.2 points respect to other classifiers.

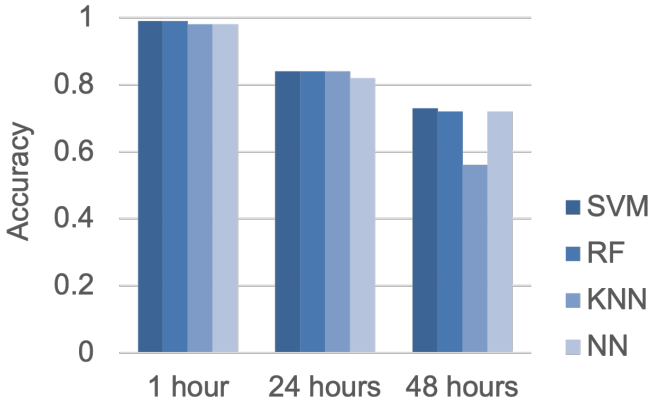

Figure 7.12: Single-step Accuracy

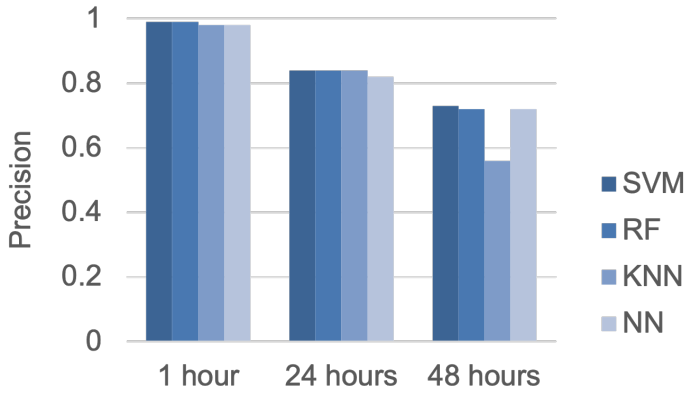

Figure 7.13: Single-step Precision

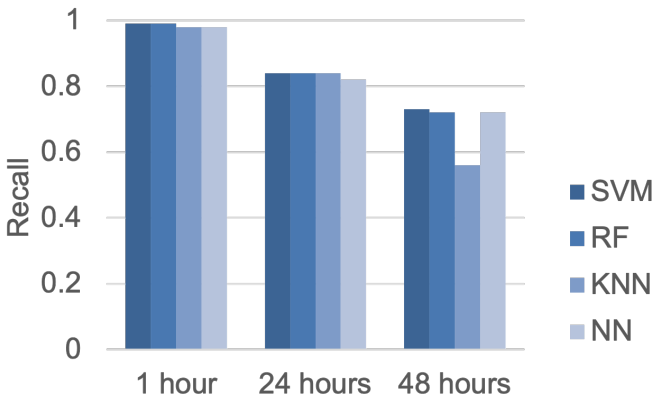

Figure 7.14: Single-step Recall

Execution times considering both training and prediction are plotted on Fig. 7.15. Also here Here KNN shows the lowest execution time, as expected, considering that no training is performed by the algorithm, RF is second, at considerable distance SVM and after another time gap, NN finishes.

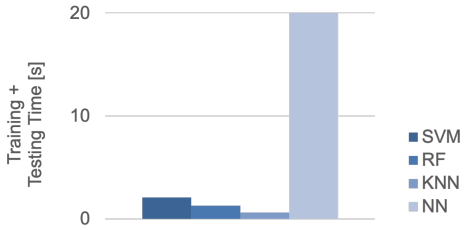

Figure 7.15: Single-step Ex. Time

8 | Conclusion and Future Works

In this thesis, we introduced a failure prediction methodology in microwave networks using machine learning techniques. We use real data from a SIAE's microwave networks.

We proposed two solution: a short-term multi-step and a long-term single-step. Both of them require as a first step, a data preparation module leveraging clustering technique. Then, we developed two different workflows: the first, short-term multi-step that is composed of a alarms forecast module followed by detection and identification. The forecast module is built with a LSTM model to best predict future alarms, improving performance compared to probabilistic baseline forecasters. After, the first classification module detects failure windows that are fed to the final root-cause identification ensemble module. Several scenario are simulated, varying the time horizon from 2 to 120 seconds (which is a bottleneck value imposed by computer memory). During the identification phase, different classification models (i.e. SVM, RF, KNN, ANN) are used and trade-offs in terms of metrics and time complexity are identified. Performance are very good for all of them, however computational time is different, where the following standing stands, starting from the first classified: KNN, RF, SVM and last NN.

The long-term single-step workflow leverages windows data structure avoiding the use of an alarms forecast module and enabling long term prediction in time horizon with granularity of hours instead of seconds. In this case, forecast accuracy decreases for longer time horizon. Different prediction algorithms (e.g. SVM, RF, KNN, ANN) have similar performance and execution times, but a performance degradation of 20% is encountered for kNN with big prediction horizon.

Overall, the framework results in very high forecast accuracy and reasonable linear time complexity, suggesting these approaches are suitable to solve this problem, despite strong dataset unbalance.

Future works could be the integration of a graph neural network for fault identification and localization. With better hardware short term multi-step prediction horizon could be extended and compared with long-term single-step approach. Furthermore alternative forecast models could be benchmarked.

Bibliography

[1] A. Rigallo, A. L. Stringa, and F. Verroca, "Real-time monitoring and operational assistant system for mobile networks," in *NOMS 2002. IEEE/IFIP Network Operations and Management Symposium.'Management Solutions for the New Communications World'(Cat. No. 02CH37327)*, pp. 899–901, 2002.

[2] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[3] C. Wan, W. Li, W. Ding, Z. Zhang, Q. Lu, L. Qian, J. Xu, J. Lu, R. Cao, B. Ye, and S. Lu, "Multi-task sequence learning for performance prediction and kpi mining in database management system," *Information Sciences*, vol. 568, pp. 1–12, 2021.

[4] G. Savva, T. Panayiotou, I. Tomkos, and G. Ellinas, "Deep graph learning for qot estimation of unseen optical sub-network states: Capturing the crosstalk impact on the in-service lightpaths," 2021.

[5] B. Wang, H. Yang, Q. Yao, A. Yu, T. Hong, J. Zhang, M. Kadoch, and M. Cheriet, "Hopfield neural network-based fault location in wireless and optical networks for smart city iot," in *2019 15th IEEE International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 1696–1701, 2019.

[6] G. Dorgo and J. Abonyi, "Learning and predicting operation strategies by sequence mining and deep learning," *Computers & Chemical Engineering*, vol. 128, pp. 174–187, 2019.

[7] S. Shahkarami, F. Musumeci, F. Cugini, and M. Tornatore, "Machine-learning-based soft-failure detection and identification in optical networks," in *2018 Optical Fiber Communications Conference and Exposition (OFC)*, pp. 1–3, 2018.

[8] A. Binczewski, M. Stroinski, and R. Szuman, "Web-based approach to the network physical topology management," in *IEEE Workshop on IP Operations and Management*, pp. 103–107, 2002.

[9] K. Hatonen, M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen, "Tasa:

Telecommunication alarm sequence analyzer or how to enjoy faults in your network," in *Proceedings of NOMS'96-IEEE Network Operations and Management Symposium*, vol. 2, pp. 520–529, 1996.

[10] R. Vaarandi, "A data clustering algorithm for mining patterns from event logs," in *Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003)(IEEE Cat. No. 03EX764)*, pp. 119–126, 2003.

[11] R. Vaarandi and M. Pihelgas, "Logcluster-a data clustering and pattern mining algorithm for event logs," in *2015 11th IEEE International conference on network and service management (CNSM)*, pp. 1–7, 2015.

[12] G. Jakobson and M. Weissman, "Alarm correlation," *IEEE network*, vol. 7, no. 6, pp. 52–59, 1993.

[13] L. Tong-yan and L. Xing-ming, "The study of alarm association rules mining in telecommunication networks," in *2008 IEEE International Conference on Communications, Circuits and Systems*, pp. 1030–1034, 2008.

[14] A. S. Kazmi, "Application of statistical sampling to predict faults from real time alarm data," in *2011 IEEE 14th International Multitopic Conference*, pp. 290–295, 2011.

[15] D. Wang, L. Lou, M. Zhang, A. C. Boucouvalas, C. Zhang, and X. Huang, "Dealing with alarms in optical networks using an intelligent system," *IEEE Access*, vol. 7, pp. 97760–97770, 2019.

[16] M. Zhang and D. Wang, "Machine learning based alarm analysis and failure forecast in optical networks," in *2019 24th IEEE OptoElectronics and Communications Conference (OECC) and 2019 International Conference on Photonics in Switching and Computing (PSC)*, pp. 1–3, 2019.

[17] S. Kobayashi, K. Otomo, K. Fukuda, and H. Esaki, "Mining causality of network events in log data," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 53–67, 2017.

[18] F. Musumeci, C. Rottondi, G. Corani, S. Shahkarami, F. Cugini, and M. Tornatore, "A tutorial on machine learning for failure management in optical networks," *Journal of Lightwave Technology*, vol. 37, no. 16, pp. 4125–4139, 2019.

[19] P. Tee, G. Parisis, and I. Wakeman, "Towards an approximate graph entropy measure for identifying incidents in network event data," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 1049–1054, 2016.

[20] R. Deebalakshmi and V. Jyothi, "A survey of classification algorithms for network traffic," in *2016 Second International IEEE Conference on Science Technology Engineering and Management (ICONSTEM)*, pp. 151–156, 2016.

[21] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[22] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2018.

[23] P. Zhu, S. Zhang, H. Luo, and Z. Wu, "A semi-supervised method for classifying unknown protocols," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pp. 1246–1250, IEEE, 2019.

[24] J. Kwon, D. Jung, and H. Park, "Traffic data classification using machine learning algorithms in sdn networks," in *2020 International IEEE Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1031–1033, 2020.

[25] S. Yan, F. N. Khan, A. Mavromatis, D. Gkounis, Q. Fan, F. Ntavou, K. Nikolovgenis, F. Meng, E. H. Salas, C. Guo, *et al.*, "Field trial of machine-learning-assisted and sdn-based optical network planning with network-scale monitoring database," in *2017 IEEE European Conference on Optical Communication (ECOC)*, pp. 1–3, 2017.

[26] D. Rafique, "Machine learning based optimal modulation format prediction for physical layer network planning," in *2018 20th International Conference on Transparent Optical Networks (ICTON)*, pp. 1–4, 2018.

[27] T. Miyazawa and H. Harai, "Supervised learning based automatic adaptation of virtualized resource selection policy," in *2016 IEEE 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*, pp. 170–175, 2016.

[28] X. Zhao, H. Yang, H. Guo, T. Peng, and J. Zhang, "Accurate fault location based on deep neural evolution network in optical networks for 5g and beyond," in *Optical Fiber Communication Conference*, pp. M3J–5, Optical Society of America, 2019.

[29] E. Balevi and R. D. Gitlin, "Unsupervised machine learning in 5g networks for low latency communications," in *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–2, 2017.

[30] M. Mamdouh, M. A. Elrukhsi, and A. Khattab, "Securing the internet of things and

wireless sensor networks via machine learning: A survey," in *2018 IEEE International Conference on Computer and Applications (ICCA)*, pp. 215–218, 2018.

[31] P. Kachurka and V. Golovko, "Neural network approach to real-time network intrusion detection and recognition," in *Proceedings of the 6th IEEE international conference on intelligent data acquisition and advanced computing systems*, vol. 1, pp. 393–397, 2011.

[32] K. Limthong and T. Tawsook, "Network traffic anomaly detection using machine learning approaches," in *2012 IEEE Network Operations and Management Symposium*, pp. 542–545, 2012.

[33] S. Barzegar, M. Ruiz, A. Sgambelluri, F. Cugini, A. Napoli, and L. Velasco, "Soft failure detection, localization, identification, and severity prediction by estimating qot model input parameters," *IEEE eTransactions on network and service management*, 2021.

[34] Z. Li, Y. Zhao, Y. Li, S. Rahman, F. Wang, X. Xin, and J. Zhang, "Fault localization based on knowledge graph in software- defined optical networks," *Journal of Lightwave Technology*, 2021.

[35] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," *Advances in neural information processing systems*, pp. 473–479, 1997.

[36] X. Wang, Y. Zhang, and M. Zhang, "Time series prediction method based on variant lstm recurrent neural network," *Neural Processing Letters*, vol. 52, no. 2, pp. 1485–1500, 2020.

[37] C.-T. SU and Y.-H. HSIAO, "An evaluation of the robustness of mts for imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 10, pp. 1321–1332, 2007.

[38] F. Musumeci, L. Magni, O. Ayoub, R. Rubino, M. Capacchione, G. Rigamonti, M. Milano, C. Passera, and M. Tornatore, "Supervised and semi-supervised learning for failure identification in microwave networks," *IEEE Transactions on Network and Service Management*, 2020.

[39] Commscope, *Microwave communication basics*. 2017.

[40] "Microwave odu." `http:https://www.microwave-link.com/microwave/microwave-odu/`. Accessed: 2021-03-20.

[41] "Understanding itu-t error performance recommendations." https://www.julesbartow.com/Pictures/ITS/ITU-T_Errors_ApplicationNote2.pdf.

[42] "Understanding itu-t error performance recommendations." https://www.julesbartow.com/Pictures/ITS/ITU-T_Errors_ApplicationNote2.pdf.

[43] "Ags20." https://www.siaemic.com/index.php/products-services/telecommunication-systems/microwave-product-portfolio/ags20.

[44] "Workflow of a machine learning project." https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94.

[45] D. Sarkar, R. Bali, and T. Sharma, "Practical machine learning with python," *A Problem-Solvers Guide To Building Real-World Intelligent Systems. Berkely: Apress*, 2018.

[46] M. J. Zaki and W. Meira Jr, *Data Mining and Machine Learning: Fundamental Concepts and Algorithms*. Cambridge University Press, 2019.

[47] "Multiclass and multioutput algorithms." https://scikit-learn.org/stable/modules/multiclass.html.

[48] E. García-Gonzalo, Z. Fernández-Muñiz, P. J. Garcia Nieto, A. Sánchez, and M. Menéndez, "Hard-rock stability analysis for span design in entry-type excavations with learning classifiers," *Materials*, vol. 9, p. 531, 06 2016.

[49] B. Boser, I. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *COLT '92*, 1992.

[50] S. Lazic, "The elements of statistical learning: Data mining, inference, and prediction, 2nd edn."

[51] "Decision tree vs. random forest – which algorithm should you use?." https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/.

[52] "A complete guide to k-nearest-neighbors with applications in python and r." https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/.

[53] N. Buduma and N. Locascio, *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. " O'Reilly Media, Inc.", 2017.

[54] "Softmax function." https://deepai.org/machine-learning-glossary-and-terms/softmax-layer.

[55] A. Tharwat, "Classification assessment methods," *Applied Computing and Informatics*, 2020.

[56] "Evaluation measures for multiclass problems." `http://gabrielelanaro.github.io/blog/2016/02/03/multiclass-evaluation-measures.html`.

[57] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[58] X. Yuan, L. Li, and Y. Wang, "Nonlinear dynamic soft sensor modeling with supervised long short-term memory network," *IEEE Transactions on Industrial Informatics*, vol. PP, 02 2019.

List of Figures

List of Tables