UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
MESTRADO EM ENGENHARIA ELÉTRICA E DE TELECOMUNICAÇÕES

FAULT DETECTION AND DIAGNOSIS OF SOLAR-POWERED WIRELESS MESH NETWORKS THROUGH MACHINE LEARNING

VINICIUS CORRÊA FERREIRA

NITERÓI

2016

Universidade Federal Fluminense

Vinicius Corrêa Ferreira

Fault Detection and Diagnosis of Solar-Powered Wireless Mesh Networks through Machine Learning

Master thesis proposal submitted to the Electrical and Telecommunications Engineering Graduate Program of the Universidade Federal Fluminense as a partial requirement for obtaining the title of Master of Electrical and Telecommunications Engineering in Telecommunications Systems.

Supervisor:
Prof. Ricardo Campanha Carrano, D.Sc.

NITERÓI

2016

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

F383 Ferreira, Vinicius Corrêa
Detecção e diagnóstico de falhas em redes em malha sem fio alimentadas por energia solar utilizando aprendizado de máquinas / Vinicius Corrêa Ferreira. – Niterói, RJ : [s.n.], 2016.
73 f.

Dissertação (Mestrado em Engenharia de Telecomunicações) - Universidade Federal Fluminense, 2016.
Orientador: Ricardo Campanha Carrano.

1. Sistemas de telecomunicação móvel. 2. Rede sem fio. 3. Aprendizado de máquina. I. Título.

CDD 621.38456

VINICIUS CORRÊA FERREIRA

DETECÇÃO E DIAGNÓSTICO DE FALHAS EM REDES EM MALHA SEM FIO ALIMENTADAS POR ENERGIA SOLAR UTILIZANDO APRENDIZADO DE MÁQUINAS

Dissertação de Mestrado apresentada ao Programa de Pós Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense como requisito parcial para a Obtenção do Grau de Mestre em Engenharia Elétrica e de Telecomunicações.
Área de concentração: Sistemas de Telecomunicações

BANCA EXAMINADORA

Prof. Dr. Ricardo Campanha Carrano - Orientador
Universidade Federal Fluminense - UFF

Prof. Dr. Natalia Castro Fernandes
Universidade Federal Fluminense - UFF

Prof. Dr. Diego Gimenez Passos
Universidade Federal Fluminense – UFF/Instituto de Computação

Prof. Dr. Miguel Elias Mitre Campista
Universidade Federal do Rio de Janeiro - UFRJ

Niterói
(abril/2016)

Agradecimentos

Agradeço aos meus pais, Fernando e Sueli, que fizeram da minha vida o projeto de vida deles. Todo o meu amor e gratidão.

À minha namorada, Flavia, pelo companheirismo, suporte e incentivo em todos os momentos.

Aos meus irmãos na amizade, que estão sempre presentes na minha essência e formação.

Ao meu orientador Ricardo Carrano, amigo e mentor, por generosamente compartilhar seu conhecimento e pela sua paciência e dedicação.

À equipe do Laboratório MidiaCom, é um enorme prazer aprender e trabalhar com vocês.

A todos que direta ou indiretamente participaram da minha formação, o meu muito obrigado.

Abstract

This thesis proposes a machine learning based automated fault detection and diagnosis (FDD) on solar-powered wireless mesh networks (SWMN). The inherent complexity of WMNs makes it difficult to find a fixed model solution for FDD. In addition, manual inspection is extremely costly and requires a highly skilled workforce, thus becoming impractical as the problem scales. A predefined dictionary of failures, based on previous experiences, was used to solve those problems (modeling and scalability) and turn the fault detection and diagnosis task into a feasible pattern classification problem. The knowledge discovery in databases methodology was evaluated to solve the problem. Each steps of the methodology, as database population, problem characterization, data processing, a study of classification algorithms suitability for the task and results interpretation were performed and evaluated. At the end of this quest, the Support Vector Machine presented the best results and was selected to be implemented and tested in our future works.

Resumo

Este trabalho descreve uma solução para um módulo de detecção e diagnóstico de falhas autônomo para redes em malha sem fio utilizando técnicas de aprendizado de máquina. A complexidade inerente às redes em malha sem fio dificultam a solução do problema atravs de modelos matemáticos ou probabilísticos. Além disto, inspeções manuais são extremamente caras e requerem uma força de trabalho qualificada, se tornando inviáveis conforme o problema escala. Utilizou-se a experiência prévia da equipe do projeto ReMoTe para definir um dicionário de falhas e transformar o problema de detecção e diagnóstico de falhas em um problema de classificação de padrões. Desta forma simplificou-se tanto o problema de modelagem quanto o de escalabilidade da solução. Os passos da metodologia de descoberta de conhecimento em base de dados foi seguida e avaliada, como a populao da base dados, a caracterizao do problema, o estudo de algoritmos de classificação mais adequados a tarefa e a interpretao dos resultados. Ao fim desta procura, a Máquina de Vetor Suporte apresentou o melhor resultado e foi escolhida para ser implementada e testada em trabalhos futuros.

Keywords:

1. Network Management.
2. Wireless Mesh Networks.
3. Machine Learning.
4. Fault Detection and Diagnosis.

Abbreviations

AI	:	Artificial Intelligence
ARFF	:	Attribute-Relation File Format
BACnet	:	Building Automation and Control Networks
CF	:	Confidence Factor
CRs	:	Cognitive Radios
DTM	:	Decision Table Majority
ELM	:	Extreme Learning Machines
EM	:	Expectation-Maximization
FDD	:	Fault Detection and Diagnosis
IDS	:	Intrusion Detection Systems
IP	:	Internet Protocol
KDD	:	Knowledge Data Discovery
KKT	:	Karush-Kuhn-Tucker
k-NN	:	k-Nearest Neighbors
M2M	:	Machine-to-Machine
MAC	:	Medium Access Control
Mac OS	:	Macintosh Operating System
MAD	:	*Módulo Autonômico de Diagnósticos*
MANETs	:	Mobile Ad Hoc Networks
ML	:	Minimum Loss
NLOS	:	Non-Line-of-Sight
ODM	:	Oracle Data Mining
OSLR	:	Optimized Link State Routing
PC	:	Personal Computer
PDA	:	Personal Digital Assistant
QoS	:	Quality of Service
RFID	:	Radio Frequency Identification
SVM	:	Support Vector Machine
SWMN	:	Solar-powered Wireless Mesh Networks

VANETs	:	Vehicular Ad Hoc Networks
WEKA	:	Waikato Environment for Knowledge Analysis
WLAN	:	Wireless Local Area Network
WMNs	:	Wireless Mesh Networks
WSNs	:	Wireless Sensor Networks

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last twenty years, mobile communication systems and its services were of great relevance to the general population and experienced continuous growth. In 2014, mobile data traffic accounted for 4% of total IP traffic and it is estimated that it will grow three times faster than IP traffic from non-mobile devices, reaching 14% of total IP traffic in 2019 [10].

The need for broadband network connectivity gets a further boost with the growth of mobile Internet-connected devices, ranging from smartphones, tablets, and laptops, to sensor networks and machine-to-machine connectivity (M2M). This growing demand requires a greater investment in network infrastructure.

As various wireless networks evolved into the next generation to provide better services, a new technology has emerged: the Wireless Mesh Networks (WMNs). WMNs are wireless networks that interconnect a fixed set of nodes able to pass data packets to each other, directing them to their destination through multiple hops using wireless links.

The WMN Infrastructure/Backbone can be built using various types of radio technologies, in addition to the mostly used IEEE 802.11 technologies. The mesh routers form a network of independent self-configuring, self-healing links. With gateway functionality, mesh routers can be connected to the Internet. This approach, also referred to as infrastructure meshing, provides backbone for client nodes and enables integration of WMNs with existing wired networks, through gateway/bridge functionality in mesh routers. A possible WMN architecture can be observed in Figure 1.1.

WMNs' advantages are: the topologies are self-organized and self-configured, with nodes automatically establishing and maintaining mesh connectivity among themselves. This feature brings many advantages to WMNs such as low implementation cost, fault tolerance, and reliable service coverage [3]. Due to these characteristics, WMNs are being

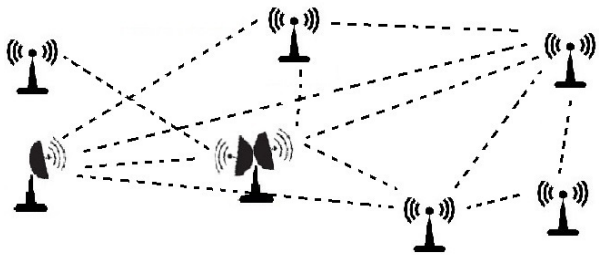


Figure 1.1: Wireless Mesh Networks.

used in a wide range of applications, such as: broadband home networking, community and neighborhood networking, enterprise networking, metropolitan area networking, vehicular networks, automation and control systems, health and medical systems, security and surveillance systems, etc [3].

But these advantages are balanced by the difficulty of its management. This is because a network is a complex system with many inter-dependent elements that affect its behavior. These elements include network protocols, traffic flows, hardware, software and, most importantly, the interactions between them. Troubleshooting a multihop wireless network is even more difficult due to unreliable physical medium, fluctuating environmental conditions, complicated wireless interference, and limited network resources. There's no heuristic or theoretical technique that captures these interactions and explains the behavior of such networks yet [15].

In addition to wireless medium uncertainties, some WMNs are installed in isolated sites. This makes physical access to nodes difficult and requires an alternative electric powering system, increasing system's complexity. One example of a highly complex mesh network set in isolated sites is the solar powered WMN deployed for the ReMoTe project [45]. It consists of 41 nodes for supervision and communication purposes installed along the power transmission line that connects the hydroelectric plant in Machadinho to a substation near the city of Campos Novos, in the south of Brazil.

The high complexity of the network itself, coupled with the difficulty of access resulted in the search for automated management methods in order to improve reliability and reduce costs and downtime.

To provide automated management of systems some features must be developed. These features serve to indicate when the system enters in undesired or unpermitted states and also to take the appropriate actions in order to maintain the system's operational state, avoiding or reducing damage and accidents. These features would be responsible

for three tasks [28]:

- **Monitoring:** Measurable variables are checked with tolerance intervals, and alarms are generated for the operator.
- **Automatic Protection:** In the case of a dangerous state, an automated control function initiates an appropriate counteraction.
- **Supervision with Fault Diagnosis:** The Monitoring task supplies the network manager with raw data. In the Supervision task this is transformed in information by processing. Features are calculated and if any significant change in these features is detected, a fault diagnosis is performed and decisions for counteractions are made.

In the ReMoTe project, some of these function were already implemented in two modules:

The monitoring function is performed by the MeshAdmin tool [13], an integrated platform for WMN management, that includes modules for collecting, storing and displaying nodes' management data. Its interface allows the network manager to observe the evolution of the collected data graphically. MeshAdmin also provides a warning module, which detects and indicates the occurrence of failures. Although MeshAdmin facilitates the detection of failures, the actual diagnosis must be done manually.

For the automatic protection function, a watchdog software was developed. This function stands on a local microprocessor, which oversees dangerous energy states and enables/disables the power supply, protecting the node. The watchdog check the nodes' responsiveness through serial communication, rebooting the node if it is unresponsive.

The supervision with fault detection and diagnosis (FDD) is not yet carried out by MeshAdmin. Thus, the intention to develop an autonomous module for that task, the *Módulo Autonômico de Diagnóstico* (MAD), a module that is able to detect and diagnose faults automatically and its methodology could be applied not only to the ReMoTe network, but also to similar WMNs.

Fault detection and diagnosis is a widely studied field in engineering. This problem is typically solved using one of three methods: the analytical solution, the application of a statistical model or the use of artificial intelligence (AI) [14]. Given the difficulties to model a WMN, their behavior, usage, and possible changes to the system, both the analytical solution and the application of a particular statistical model becomes impractical. Therefore, the AI approach was selected for the development of the module.

For five years since deployment, ReMoTe nodes presented different modes of failure that could only be diagnosed through costly and sporadic physical inspections, which often require the shutdown of the transmission line along which the nodes were installed. With those modes of failure and all those years of measured data stored by MeshAdmin in mind we focused on a machine learning approach.

In order to develop MAD, it was necessary to define a method and a specific AI technique to perform the task of FDD. This works proposal is to use machine learning technique. To this end, the knowledge discovery in databases methodology was discussed. This methodology requires an understanding of the problem, of its most critical variables and an adequate data representation, compatible with classification algorithms inputs. These steps must be carefully followed to achieve a satisfactory accuracy. After those steps, a group of classification algorithms were tested and evaluated.

To achieve this goal several trials and tests were performed as part of this process [18]. This work describes the employed methodology as well as the difficulties faced during the development of MAD, and how they were overcome.

The text is organized as follows: In Chapter 2 an overview of related works is given; We review WMN management and monitoring and also provide a description of the SWMN nodes used in the ReMoTe project in Chapter 3; We provide an introduction to the process of knowledge data discovery (KDD) and the problem of fault detection and diagnosis in Chapter 4; In Chapter 5, each traditional KDD step taken is discussed in the context of developing MAD, including populating the database, data treatment and preprocessing, and the description of a set of algorithms to perform the knowledge extraction; We then provide an objective comparison of the performance of each algorithm in providing accurate diagnosis in Chapter 6; Finally, conclusions and future work are presented in Chapter 7.

Chapter 2

Related Work

There is a wide range of applications and areas of research that uses machine learning techniques in computer networks [20][1][5]. Some traditional network functions are being implemented using machine learning seeking to improve network performance.

Usual tasks in network management and operations take advantage of machine learning tools in their solution[1]. A function in which machine learning use is widespread is network security, more specifically intrusion detection systems (IDS). In [6] IDSs are presented, both following a centralized and a distributed approach. Some approaches are presented to model a system's intrusion, these are: game theory/reinforcement learning, rule-based algorithms and statistical analysis of network data.

Classification techniques, clustering and state machines solved through reinforcement learning are shown in [1] and [20] as solutions for tasks as: routing, data aggregation, objects location and pointing, event detection and queue processing, link layer development, security and intrusion detection, as well as QoS functions, data integrity and fault detection.

Ad hoc networks, specifically Wireless Sensor Networks (WSNs) and Mobile Ad hoc Networks (MANETs) may be used to monitor dynamic environments, as in Vehicular Ad hoc Networks (VANETs). These scenarios, in which system's response change rapidly over time, due to external factors or by the system's design, may benefit from machine learning techniques to adapt to these changing conditions, thus eliminating system's redesign. Machine learning also inspires many practical solutions that maximizes resource use and increases network's life [1].

One of the critical issues related to maintenance and failure prevention in WMNs is their management. Some of these issues are addressed in [15], in the context of ReMesh, a pioneering WMN project. It presents methods to capture different statistics and metrics

from network nodes, and visual tools to aid in failure detection.

The MeshAdmin tool [13] is an integrated platform for WMN management, developed for the ReMoTe project. The platform includes modules for collecting, storing and displaying nodes' management data. Its interface allows the network manager to observe the node's behavior in a graphic form, based on collected data. MeshAdmin also provides a warning module, which detects and indicates the occurrence of failures. Although MeshAdmin facilitates the detection of failures, the actual diagnosis must be done manually.

An automated FDD technique using simulation models is proposed in [40]. The work uses the monitored network metrics to feed a simulator and compares the performance of the simulated model to the network's gathered data. If there is a significant difference, the proposal systematically injects faults in the simulated environment. When the performance of the simulator, having been inserted an specific fault, gets close to the network's performance, the system assumes that particular fault as the diagnose. The downside is a possible lack of information in a failure mode. This technique depends on a complete set of network metrics. It also relies on the accuracy and efficiency of the simulator.

Sympathy [42] also deals with FDD in sensor networks. The proposed solution relies on connectivity metrics, packet flows and node operation information. The collected data is fed to a decision tree. This decision tree was generated based on empirical knowledge of an expert to diagnose the cause of the problem, without machine learning usage, to emulate the experts' steps.

Other studies employ machine learning to perform diagnosis. Among them, [50] uses information from the node's routing table and the RIPPER algorithm [11], a classifier based on rules for intrusion detection.

Due to the limited availability of the spectrum and its inefficient use it's desirable to develop a new paradigm of communication that explores the wireless spectrum in an opportunistic way. Thus, dynamic spectrum access was proposed to solve these inefficiency problems using intelligent radios, known as Cognitive Radios (CRs). The CRs take advantage of unused frequencies, often referred to as holes in the spectrum or blanks, both in time and in space [33]. CR has a great potential in research for machine learning applied to spectrum management and for achieving efficient spectral use.

A CR must manage the spectrum properly, taking into account the communication requirements of its users. The spectrum management is divided in three stages: spectrum sensing, spectrum characterization and spectrum decision. Machine learning techniques

are used in all stages [5].

Several sensing techniques were developed and machine learning techniques were applied to increase spectrum sensing (also called spectral consciousness) performance. The spectrum usage history is used to predict its future availability and the sensing schedule is modeled as a Markovian Process. Reinforcement learning is used to predict the spectrum usage and check only the most likely to be available, reducing the spectral sensing search space [49].

Besides spectrum availability, other parameters are defined and used to characterize a channel, such as: interference, path loss, wireless channel loss, link layer delay, and the spectrum sensing and characterization delay *per se*. Neural Networks and Support Vector Machine are used to classify the channel as suitable or not to use, according to a given Quality of Service (QoS) requirement. The final step, the spectrum decision, is also modeled as a Markovian Process and the optimal policy found using reinforcement learning [5].

Differently from previous proposals [40][42][50], the solution presented in this work employs a completely autonomic process for fault detection and diagnosis on an individual node level, based on machine learning techniques. The MeshAdmin monitored data will be used to feed MAD and a set of defined failures, as the battery fault for example, can be diagnosed without human aid. This approach has the advantage of not requiring an analytical model or static computational approaches. It also seeks an increased diagnosis accuracy, reducing the costs and workforce required for network maintenance.

Chapter 3

Wireless Mesh Networks

As stated in the Introduction, Wireless Mesh Networks (WMNs) are networks that interconnect a set of nodes that can transmit data packets to each other, through multiple hops via wireless links.

In these networks there are two types of nodes: mesh routers and mesh clients. Other than the routing capability for gateway/repeater functions as in a conventional wireless router, a wireless mesh router contains additional routing functions to support mesh networking. To further improve the flexibility of mesh networking, a mesh router is usually equipped with multiple wireless interfaces built on either the same or different wireless access technologies.

Compared with conventional wireless routers, wireless mesh routers in a WMN can achieve the same coverage with much lower transmission power through multi-hop communications. Optionally, the medium access control (MAC) protocol in a mesh router is enhanced with better scalability in a multi-hop mesh environment and energy management, specially in when there is energy constraints as in a solar powered WMN, achieved through scheduling strategies [7][8].

In spite of those differences, mesh and conventional wireless routers are usually built on a similar hardware platform. Mesh routers can be built on dedicated computer systems (e.g., embedded systems) and look compact, or they can also be built on general-purpose computer systems (e.g., laptop/desktop PC).

Mesh clients also have essential functions for mesh networking, and thus, can also work as a router. However, gateway or bridge functions do not exist in these nodes. In addition, mesh clients usually have only one wireless interface. As a consequence, the hardware platform and the software for mesh clients can be much simpler than those for mesh routers. Mesh clients have a higher variety of devices compared to mesh routers.

They can be a laptop/desktop PC, pocket PC, PDA, IP phone, RFID reader, BACnet (building automation and control networks) controller, and many other devices.

3.1 WMNs Architecture

The architecture of WMNs can be classified into three main groups based on the functionality of the nodes:

- **Infrastructure/Backbone WMNs:** This type of WMNs consists of mesh routers forming an infrastructure for clients that connect to them. The WMN infrastructure/backbone can be built using various types of radio technologies. The mesh routers form a mesh of self-configuring, self-healing links. Featuring gateway functionality, mesh routers can be connected to the Internet. This approach, also referred to as infrastructure meshing, provides backbone for conventional clients and enables integration of WMNs with existing networks. An example of this type of architecture can be observed in Figure 3.1. Infrastructure/Backbone WMNs are the most commonly used type. For example, community and neighborhood networks can be built using infrastructure meshing. Typically, two types of radios are used in the routers, i.e., for backbone communication and for user communication, respectively. The mesh backbone communication can be established using long-range communication techniques including directional antennas.

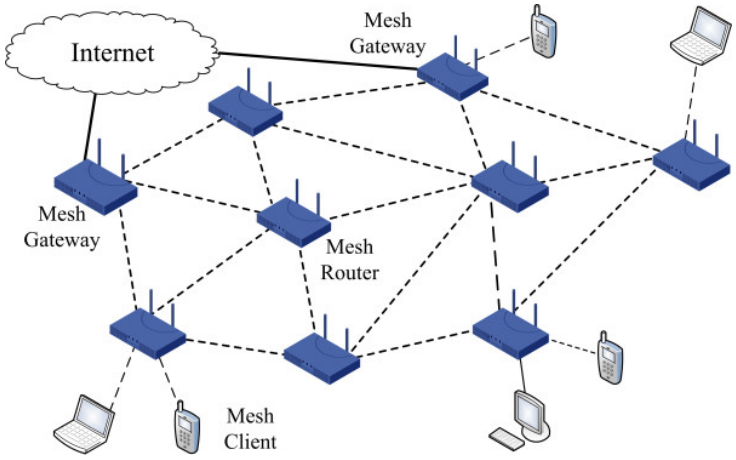


Figure 3.1: Infrastructure WMN.

- **Client WMNs:** Client meshing provides peer-to-peer networks among client devices. In this type of architecture, client nodes constitute the actual network to performing routing and configuration functionalities as well as providing end-user

applications to customers. Hence, a mesh router is not required for these types of networks. In Figure 3.2 there is an example of this architecture. In Client WMNs, a packet destined to a node in the network hops through multiple nodes to reach the destination. Client WMNs are usually formed using one type of radios on devices. Moreover, the requirements on end-user devices is increased when compared to infrastructure meshing, since, in Client WMNs, the end-users must perform additional functions such as routing and self-configuration.

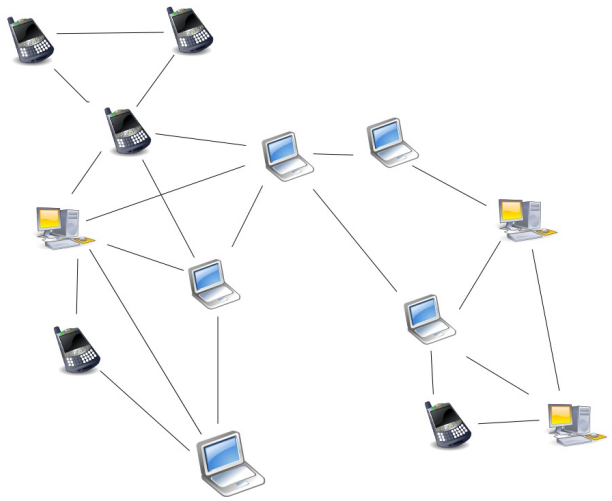


Figure 3.2: Client WMN.

- **Hybrid WMNs:** This architecture is the combination of infrastructure and client meshing, as in Figure 3.3. Mesh clients can access the network through mesh routers as well as directly meshing with other mesh clients. While the infrastructure provides connectivity to other networks such as the Internet, Wi-Fi, WiMAX, cellular, and sensor networks, the routing capabilities of clients provide improved connectivity and coverage inside the WMN.

3.2 Characteristics

The WMNs have some basic characteristics that are pointed out bellow:

- **Multi-hop wireless network.** The use of multiple hops to achieve a destination is a basic characteristic of WMNs that enables an extended coverage of the network and provides non-line-of-sight (NLOS) connectivity. The multi-hop wireless characteristic imposes the use of a routing protocol suitable for the dynamic environmental

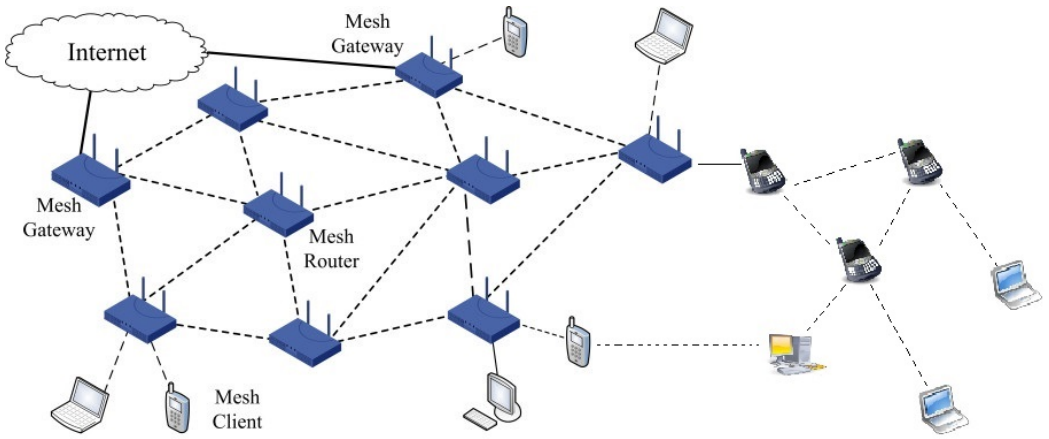


Figure 3.3: Hybrid WMN.

changes and links stability. Therefore, a more sophisticated and complex routing protocol must be used which incurs in a higher management complexity.

- **Support for ad hoc networking.** The flexibility of network architecture as viewed in Section 3.1 provides easy deployment and configuration. This characteristic is responsible for the low upfront cost and gradual growth of a WMN. From the management point-of-view this flexibility increases even more the network complexity. Various wireless links can be created and vanished with a node deployment, generating connectivity problems and highly dynamic routes. Also, a more generic protocol must be used to deal with different architectures and devices in the network.

- **Dedicated routing and configuration.** Despite the ad hoc networking characteristics, Infrastructure and Hybrid WMNs have dedicated mesh routers. It decreases the load on end-users, which provides high-end application capabilities, client mobility and less energy constrains.

- **Interoperability and integration.** Through gateway or bridge functions and different interface technologies use WMNs enables clients to use its infrastructure as a backhaul and access to existing networks and services as the Internet. Here again the network complexity is increased to support the different services requested by the users. The network protocol has to be interoperable with these different technologies.

- **Multiple radios.** Mesh routers can be equipped with multiple radios in order to separate the access and backbone traffic in the wireless domain. This improves network capacity and facilitates the management.

- **Mobility.** Mesh clients might be mobile devices, which imposes an additional challenge to routing and configuration in client and hybrid topologies.
- **Power-consumption constraints.** As stated before, WMNs can be used as access networks in isolated sites. Therefore nodes' energy supply depends on batteries and alternative power sources increasing system's complexity and challenges in the network management.

3.3 ReMoTe's Network

The ReMoTe project uses an infrastructure based on WMN. While the ultimate goal is to deploy MAD at the production network of the ReMoTe project, in the development phase prototypes were evaluated in a WMN testbed located on one of the campi of UFF, hereinafter referred to as External Mesh Network.

Both the production network and the External Mesh Network are infrastructure WMNs and have similar characteristics as the energy constraints, the use of a solar power system and multiple radios with the same technologies. Therefore, the methodology and the scenarios used in the development phase have immediate applicability for this work's purpose.

Their topologies are depicted in Figure 3.4. The External Mesh Network will be detailed in this section.

In the External Mesh Network, each mesh router is composed of three modules: the communication module, the sensing module and the power module.

The communication module consists of a router with two wireless interfaces, a client access interface consisting of an IEEE 802.11g radio, and an interface for communication between nodes (backbone), consisting of an IEEE 802.11a radio. The backbone radio is connected through a RF splitter to two directional antennas pointed towards specific nodes.

The network protocol used is the Optimized Link State Routing (OSLR), a link state based protocol designed for ad hoc networks. In the ReMoTe's network an OSLR variation is used, the OLSR-ML. This variation uses as cost function the Minimum Loss (ML) metric [39], which results in routes with minimum error probability in end-to-end communication.

The power module is formed by a solar power system that comprises a 40W solar panel, a charge controller and a bank of three 12V/7Ah lead-acid sealed batteries connected in

(a) ReMoTe project network

(b) External Mesh Network

Figure 3.4: Topologies of the Production network and the External Mesh Network.

parallel, resulting in a voltage of 12 V and total rated capacity of 21 Ah.

The sensing module used for site supervision contains two LM35 temperature sensors, one LDR 5mm light sensor and voltage and current sensors for the solar panel, batteries and primary load (the communication module).

The sensing module, allows monitoring the following physical aspects of the network:

- Solar Panel Voltage: the open circuit voltage is 21.6 V. The range is 0 - 21.6V.
- Solar Panel Current: the short circuit current is 2.67 A.
- Battery Voltage: the cycle use voltage regulation is 14.40 - 14.70V.
- Communication's Module Voltage: the router power rating is 10.5V up to 24V.
- Communication's Module Current: the nominal power consumption is 6W, given the power rating operational range the nominal current is 250mA - 570mA.
- External (ambiance) Temperature: the temperature range of the sensor goes from -55 to 150 degrees.

Figure 3.5: ReMoTe's testbed node - External Mesh Network node.

- Internal (sealed box) Temperature: the temperature range of the sensor goes from -55 to 150 degrees.
- Incident Light Intensity: The LDR 5mm had its resistance adjusted to output values in a range from 0 to 5V.
- Bytes in/out for each Network Interface: there is no specific limit.
- Available Flash Memory: the total flash memory is 4MB.
- Available RAM memory: the total RAM memory is 16MB.
- CPU Load Average: there is no specific limit.
- Link Quality: the ML metric scale goes from 0.0 to 1.0.

All data is collected, stored and managed by MeshAdmin [13]. For the purposes of this work, MeshAdmin was configured to take samples of the monitored parameters in each network node every 10 minutes. With that configuration set in a network as large as

the REMOTE project's one, the network database scales up to 1GB of sensed data per year.

Having described the network and it's available data, a discussion over AI methods and techniques to perform the fault detection and diagnosis follows.

Chapter 4

Machine Learning and Fault Detection and Diagnosis

Along with the development of the Internet, multimedia technology and research in Artificial Intelligence (AI), a series of new questions were aroused. AI has attracted increasing attention in many disciplines, being used to emulate the human brain. Started as a computer science branch, these systems have shown intelligence and human behavior characteristics. AI expert systems are used in practical applications in various fields of knowledge, as well as many aspects of social life, and also in the development of the learning as a concept [48].

4.1 Machine Learning

Machine learning is a branch of artificial intelligence that focuses on the study of systems that develop learning through data analysis. Its classic definition is [35]:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"

In other words, machine learning-based programs are those that change their behavior with the experience for a better future performance.

Thus, it can be said that machine learning focuses on the prediction of behavior based on previously known properties. This prior knowledge that assists in this prediction is acquired by what is commonly named "training".

There are three well established approaches for training in machine learning: supervised learning, unsupervised learning and reinforcement learning [32]:

- Supervised learning: is one that requires some sort of supervision in the learning process. The supervisor gives typical examples of each concept that is to be learnt. An input and expected output is given and it is replicated for new inputs.
- Unsupervised learning: there is no supervisor to give typical examples. In this case, the learning process is autonomous. Using mostly statistic properties of the data a set of examples is determined and used to evaluate new inputs.
- Reinforcement learning: learns by the interaction with a dynamic environment in which a certain goal must be accomplished (such as driving a vehicle). A target function is used to determine the degree of success of a strategy.

The most common tasks in machine learning can be divided into classification, regression, clustering and association [32]. For a slightly more comprehensive description of the concepts involved in learning and these activities, the following subsections will describe the main points regarding these learning methods from the perspective of algorithms.

4.1.1 Supervised learning

To solve the learning problem many different algorithms were developed in order to get useful results. Some of these algorithms use data generalization, where an universe can be represented from a data set with particular characteristics.

Labeling all possible answers of a data input set, a search within this set of previously known responses can be used as a solution when new unlabeled data is presented. When we classify all possible answers this way, it is said that the created algorithm is a supervised algorithm. And the data group used to adjust the machine is called the training set.

In the search for generalization there are three important decisions to make in machine learning systems [32]:

1. The concept description language.
2. The order in which the search for answers is performed.
3. How to avoid overtraining (overfitting) for a set of training data.

The three properties that emerge from the decisions above are called “bias”, respectively: language bias, search bias and bias in estimator.

In the case of language bias, data representation should be carefully chosen, as this directly affects what can be extracted from it. The choice of "best response" is made from some criteria defined by the researcher. The final result should be the best from the quantitative point of view and also from goodness of fit point of view. The goodness of fit is a measure of how well the model fits the data set, observing the error between the model response and the expected value.

Due to the large number of response possibilities, it is not usual to perform a search through all response space to ensure that the result is the absolute best. Therefore, the search result is found most often in an heuristic way, thus there is always the possibility to enhance the solution.

Regarding the prevention of excessive training, despite not being totally unrelated to the search bias, it constitutes a special problem. This problem of choosing the training data set will influence what may or may not be learned. One should start training with the most basic forms of representation and only after its validation use a more complex representation form.

Thus, the search is always in favor of simpler descriptions, using as stop criteria a response considered complex enough. This method is called forward pruning.

There is no best algorithm or learning paradigm. For each problem you must find the best way to associate descriptions and biases [47].

4.1.1.1 Classification and Regression

The main task of supervised learning, classification, aims to identify to which particular class a given entry belongs. Consider the following example: a user of a social network has features like access time, home city of access and other relevant points that make up his/her profile. These features are associated with a user profile class and can be used to trigger specific responses to user's actions. Thus, the user in question could fit into profiles such as sports enthusiast and that information used to display sports related news and advertisement.

Regression is similar to classification. However, in classification, records are identified by categorical values and grouped into classes. In the regression, records are identified by a numerical value. This makes possible a value estimation of a given variable analyzing a known record. As an example, a set of records containing the values of monthly energy consumption of a residence can be considered. After being analyzed, these records can

provide a forecast of future expenses for the same household.

Examples of algorithms: Decision Trees, k-Nearest Neighbors, Linear Regression, Naive Bayes, Neural Networks, Support Vector Machine [22].

4.1.2 Unsupervised Learning

In unsupervised learning properties and probability density functions should be inferred from the data set without the aid of previous information about the response space.

In general, the data set is larger and more complex than the used in a supervised learning [32]. Therefore, it is hard to determine the validity of inferences from most unsupervised learning algorithms.

Appealing to heuristic arguments is usual, not only to motivate algorithms, as is often the case in supervised learning, but also to measure the quality of obtained results. This led to the proliferation of proposed methods, since efficacy can not be verified directly [23].

4.1.2.1 Clustering

The clustering problem is: given a database, group (cluster) objects in a way that similar objects are in the same cluster among the clusters created [36].

There are several areas that makes use of it, from grouping categories of people to more general objects such as software functions.

Examples of algorithms: k-Means, DBSCAN, Expectation-Maximization (EM) [22].

4.1.2.2 The Association Problem

The methods of learning shown so far were part of a predictive analysis, in which you want to predict the unknown value of a particular variable from the historical analysis of the data stored in the database (training base). Other tasks as data description, that extracts patterns and rules that describe important characteristics of the data application domain, may also be solved through machine learning. The association problems are the most common example.

To associate is to identify relations in data. These relations refer to objects frequencies in an experiment. One object frequency is related to another one and if this

occurrence is recurrent through experiments there is an association between these two objects frequencies.

Although it looks simple to the point of view of data mining techniques, it gives good results, especially when standard profiles are analyzed.

A classic example is a supermarket customer who has a consumer profile, this profile was generated analyzing several costumers shop lists. Consumed items and their frequencies are related, facilitating the organization and arrangement of the items displayed to the costumer and also provide information for personalized marketing.

Examples of algorithms used: Apriori, Eclat, FT-Growth.

4.1.3 Reinforcement Learning

Reinforcement learning is the problem faced by an agent who must learn how to behave in a dynamic environment through trial and error interactions. There is a strong resemblance to a research paradigm with the same name in psychology. Reinforcement learning is considered as a class of problems, rather than as a set of techniques to solve a generic problem [29].

There are two main strategies for solving reinforcement learning problems. The first is to look in the space of behaviors in order to find one that performs well in the environment. This approach was taken by work on genetic algorithms and genetic programming, as well as some newer research techniques [44]. The second is the use of statistical techniques and dynamic programming methods to estimate the utility of taking actions in states of a defined universe.

4.1.4 Other Learning Methods

Some alternative learning methods are emerging and are mostly techniques that try to bypass the traditional methods' problems. Examples of these techniques are:

- Extreme Learning Machines (ELM): These are techniques that seek to solve the problems of slowness in learning, difficulties to interpret the operations of the machines and low computational scalability. The ELM is based in using artificial neural networks with random intermediate layers and make the training only in the outer, extreme, layers. This technique has shown good potential in solving regression and classification problems [25].

- Semi-Supervised Learning: It assumes that there are a few labeled data and the effort to obtain more copies of labeled is too high, for getting labeled data is not a simple task. So it tries to make use of both supervised learning with the labeled data and unsupervised learning with the unlabeled data and produces a decision boundary analyzing the responses of both models [51].

- Transfer Learning: In all methods mentioned training and test data are taken from the same space and have the same distribution attributes. The Transfer Learning seeks to solve a given problem adapting the solution of a similar one. Thus, the technique enables the learning transfer in different attribute spaces and problems with small distribution differences [38].

4.1.5 Available Tools

Several tools were developed for the machine learning purpose. Many of them are in the form of libraries and modules for programing languages, as scikit-learn, used in Python and MatLab toolboxes, and there are also specific programing languages used for statistical purposes, as R [46]. There are also software suites that offer the whole process of learning, from organizing and preparing the data to create graphical representations that help interpret the results. Some these software suites are presented in the subsequent sections.

4.1.5.1 IBM SPSS Modeler

Originally called Clementine by ISL upon its release in 1994, it was later acquired by SPSS in 2010, which in turn merged with IBM in 2010. Since then it has been called the IBM SPSS Modeler. With its latest version launched in 2015, as IBM SPSS Modeler 17.0.

SPSS Modeler is a text analysis and data mining software, widely used to create predictive models. This tool has a graphical interface that allows users an easy way to visualize the data, using flow charts [26].

This software supports several algorithms, such as neural networks, decision trees, Bayesian learning, among many others [22].

4.1.5.2 Oracle Data Mining

Oracle Data Mining (ODM) offers the use of machine learning techniques supported by native features of Oracle Database [37].

The main idea behind ODM is to enable Oracle Database users to transparently consult the database using data mining techniques. This is intended to facilitate the database administrator to find new results and analyze the data from a new perspective.

In conjunction with the ODM, other supplements are presented, including Oracle Data Miner, which assists you in generating ODM flow chart. The graphical interface of Oracle Data Miner is very similar to the IBM SPSS Modeler. ODM implements several algorithms in the categories of classification, regression and association.

4.1.5.3 Weka

Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can be applied directly to a dataset or called from Java code. The Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well suited for developing new machine learning systems [21].

Weka was developed at the University of Waikato in New Zealand, and its name means Waikato Environment for Knowledge Analysis (WEKA). The system is written in Java and distributed under the terms of the GNU General Public License. The tool runs on many platforms and has been tested on Linux, Windows and Macintosh Operating System (Mac OS). It provides a uniform interface to many different learning algorithms, along with methods of pre and post processing and evaluating the result of learning systems in any given data set.

Weka is available from the website www.cs.waikato.ac.nz/ml/weka. It can be downloaded as a platform-specific installer or a jar file, a Java executable, which runs in the usual way if Java is installed. It is a free tool. There are free online courses for those interested in data mining and machine learning field.

4.2 Fault Detection and Diagnosis

With the database of network's monitored data and the knowledge of failures that most affected the mesh routers, we decided to model our FDD problem as a classification

problem. Therefore, it should be sufficient to define if the node is under regular operation or in failure mode, and identify that failure, based on the data generated by the sensing module.

To set the classes of the problem, we defined a fault dictionary. A set of the common flaws observed on the network nodes was defined, i.e. the fault dictionary, based on the previous experience of the project team. Each fault of the dictionary will be detailed later, they are:

- high processor usage;
- high RAM memory consumption;
- battery failures;
- low efficiency of the solar panel;
- misalignment of the antennas;
- defects on the RF cable connectors.

Having defined the set of classes in this fault dictionary, and the possible sensed data used for the classification task, we had to evaluate the classification algorithms for the task.

4.2.1 Classification algorithms

We had to evaluate a series of classification algorithms, and choose a group of these to perform the tests. For the construction of the classifiers and performance tests, the Weka tool [21] was used.

The sensing module generated data containing several numerical attributes, and we have added one nominal attribute to the problem, the class. Different classification models can be used to solve problems with such characteristic. These models are: statistical, linear, rule-based, instance-based and divide-and-conquer [47].

Before discussing the models, we will introduce some assumptions and notations. Let $\mathbf{X} = (X_1, ..., X_n)$ be a vector of observed random variables, called *features*, where each feature,X_i, takes values from its *domain* D_i. The set of all feature vectors is denoted $\omega = D_1 \times ... \times D_n$, in our problem it represents all possible samples of sensed data.

Let C be an unobserved random variable denoting the class of a given sample, where $C = c|c \in \{0, .., m-1\}$. Capital letters such as X_i, will denote variables, while lower-case letters, such as x_i, will denote their values; boldface letters will denote vectors. A classifier is defined by an hypothesis function $h : \omega \rightarrow C$, that assigns a class to any given example.

4.2.1.1 Statistical Classifiers

Statistical models are used to study the probability that an instance belongs to a certain class based on the value of its attributes and distribution of each attribute per class. The most common statistical models are based on Bayes Theorem of conditional probability. In this work, the Naive Bayes [43] algorithm was chosen to represent this group of algorithms.

The Naive Bayes algorithm is a simplification of Bayesian Classifiers. The Bayes Optimal Classifier $h^*(\mathbf{x})$ use as hypothesis a discriminant function $f_i(\mathbf{x}), i \in C$, the class posteriori probabilities given a feature vector, i.e. $f_i(\mathbf{x}) = P(C = i|\mathbf{X} = \mathbf{x})$ and selects the class with maximum discriminant function on a given example: $h^* = argmax_{i \in C}\{f_i(\mathbf{x})\}$.

Applying the Bayes' Theorem:

$$f_i(\mathbf{x}) = \frac{P(\mathbf{X} = \mathbf{x}|C = i)P(C = i)}{P(\mathbf{X} = \mathbf{x})}$$

The value of P($\mathbf{X}$=$\mathbf{x}$) is identical for all classes, and therefore can be ignored. This yields Bayes discriminant functions to:

$$h^*(\mathbf{x}) = argmax_{i \in C}\{P(\mathbf{X} = \mathbf{x}|C = i)P(C = i)\}$$

Direct estimation of $P(\mathbf{X} = \mathbf{x}|C = i)P(C = i)$ can be hard in high-dimensional feature spaces. Therefore, approximations are commonly used. In Naive Bayes, it is assumed that all features are independent given class, that is, $P(\mathbf{X}|C) = \prod_{i=1}^{n} P(X_i|C)$. Consequently, we have the Naive Bayes classifier defined as:

$$h(\mathbf{x}) = argmax_{i \in C}\{\prod_{j=1}^{n} P(X_j = x_j|C = i)P(C = i)\}$$

4.2.1.2 Linear Models

Linear models are divided into two subgroups: those using linear regression and linear transformation to create a class membership function, and those that work with the hypothesis of linear separability of classes, and thus try to find hyperplanes that divide the classes in a vector space. For this work, the Support Vector Machine [9] was used.

SVM is based on the search for the optimal hyperplane to discriminate linear separable problems. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class, so-called *functional margin* (ρ), since in general the larger the margin the lower the generalization error of the classifier, as can be seen in Figure 4.1.

Let $g(\mathbf{x})$ be the discriminant function and $C = \{-1, 1\}$ for the closest points of each class to the discriminant function, called *support vectors*. We'll have w_0 and b_0 as parameters to be optimized in terms of ρ to find the discriminant function with maximum functional margin.

$$g(\mathbf{x}) = \mathbf{w}_0^T.\mathbf{x} + b_0 \text{ (discriminant function)}$$
$$g(\mathbf{x}_i) = \mathbf{w}_0^T.\mathbf{x}_i + b_0 = -1 \text{ (lower margin hyperplane)}$$
$$g(\mathbf{x}_i) = \mathbf{w}_0^T.\mathbf{x}_i + b_0 = +1 \text{ (upper margin hyperplane)}$$

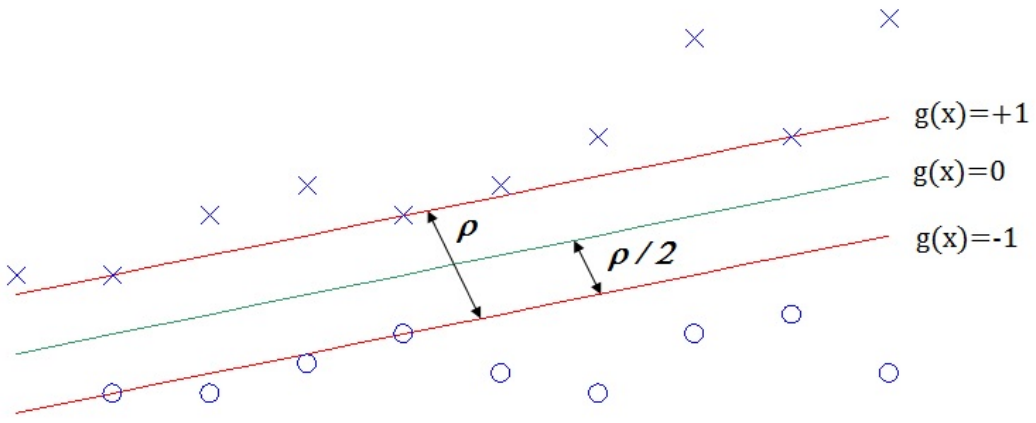


Figure 4.1: SVM example.

We know from analytic geometry that the distance between a given point and a line is:

$$r = \left| \frac{g(\mathbf{x}_i)}{\mathbf{w}_0^T} \right|$$

Therefore, the distance between the *support vectors* and the discriminant function is:

$$r = \frac{1}{|\mathbf{w}_0^T|} \implies \rho = \frac{2}{|\mathbf{w}_0^T|}$$

As we also have to prevent examples from falling into the margin, we add another limitation:

$$|g(\mathbf{x})| \geq 1, \forall \mathbf{x}_i$$

The optimization problem presented can be by convenience reduced to: $argmin_{(\mathbf{w},b)} \frac{1}{2}\|\mathbf{w}\|^2$, subject to $y_i(\mathbf{w}.\mathbf{x}_i - b) \geq 1, \forall i = 1, 2, ..., N$. Which corresponds to the Karush-Kuhn-Tucker (KKT) conditions [19] and can be solved through quadratic programming. For linearly separable problems we have our hypothesis:

$$h(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

From the KKT Conditions:

$$\lambda_i[d_i(\mathbf{w}^T.\mathbf{x}_i + b) - 1] = 0, \forall i = 1, 2, ..., N$$

$$\lambda_i \geq 0, \forall i = 1, 2, ..., N$$

For non-linear separable problems the Cover's Theorem [12]:

"A complex pattern-classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated."

Therefore, for non-linear separable problems, a non-linear transformation of the feature space is performed and the problem is solved as a linear one. The hypothesis for the non-linear separable case is:

$$h(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i y_i K(\mathbf{x}, \mathbf{x}_i) + b$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0, \forall i = 1, 2, ..., N$$
$$0 \leq \lambda_i \leq C, \forall i = 1, 2, ..., N$$

Where C is a parameter to be set and $K(\mathbf{x}, \mathbf{x}_i)$ is a kernel function of the transformation. The most common kernels are:

- Gaussian: $K(\mathbf{x}, \mathbf{x}_i) = e^{-\frac{1}{2\sigma}\|\mathbf{x}-\mathbf{x}_i\|^2}$
- Polinomial $K(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x}^T\mathbf{x}_i + 1)^p$
- Sigmoidal $K(\mathbf{x}, \mathbf{x}_i) = tanh(\beta_0\mathbf{x}^T\mathbf{x}_i + \beta_1)$

4.2.1.3 Ruled-Based Models

Rule-based models focus on trying to find rules which best separate the given class from the others. These models start from the observation of a specific class to create a rule that separates it from the others as accurately as possible. At the end, a set of rules is created and evaluated in succession to determine the class of a given instance. The rule-based algorithm tested in this work was the Decision Table Majority [30].

The Decision Table Majority (DTM) has two basic components:

- A *schema*, which is a set of features, $\{\mathbf{S} \subset \omega\}$.
- A *body*, a multiset of labelled instances. The instances consists of an example, with defined values of features in the schema and a label, the class, $\{\mathbf{I} \in \mathbf{X}|C = c\}$.

When a new instance $\mathbf{X}_i$ is presented, the respective value of c is assigned by searching in the *body* for a set I of exact matches for the features $\mathbf{S}$, being all other features ignored. If there's no match, it returns the majority class in the DTM; otherwise it returns the majority class in I.

The problem is finding the optimal subset $\mathbf{S}$ for the classification task. An usual approach is to take each class in turn and seek a way of describing it with *if-then* tests and cover all of its instances. This is called a *covering* approach because at each stage you identify a rule that "covers" some of the instances. It ranks the tests performed in the rule based on the information gain of the test:

$$p\left[log\left(\frac{p}{T}\right) - log\left(\frac{P}{T}\right)\right]$$

Where p and t are the number of positive instances and the total number of instances covered by the new rule, and P and T are the corresponding number of instances that satisfied the rule before the new test was added.

To illustrate, consider Figure 4.2 with an ω space of instances, with two features x and y and classes red and blue. To cover the red class we add the following rule to the

table: $if(x \leq 1.2) \implies c = red$. But it doesn't cover extensively all red instances. Therefore another test is performed in the rule, $if(x \leq 1.2 \vee y \leq 1.0) \implies c = red$, which gives us information gain. After exhausting tests, if there are still instances of a class to be covered, a new rule might be added to the table. And the process continues until all instances in the space are covered by a rule.

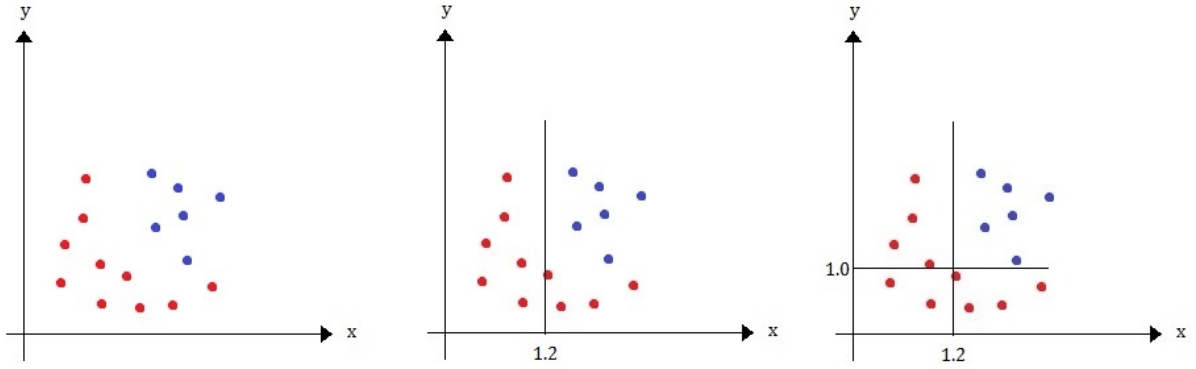


Figure 4.2: DTM example.

4.2.1.4 Instance-Based Models

Instance-based models use distance functions to determine a group of instances at the training database closest to the evaluated instance. The majority class of this group is chosen to label the targeted instance. Under this class of algorithms, we used the k-Nearest Neighbors (k-NN) [2].

The k-NN algorithm uses a distance function to define the k nearest instances in the space ω to a given instance of test. The outcome is the majority class of this k group of instances. As can be seen in the Figure 4.3 a proper set of the parameter k is of great relevance.

To define this k-group different distance functions might be used, usually they are:

- Euclidean distance: $D(\mathbf{X}_1, \mathbf{X}_2) = ((x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + ... + (x_{1n} - x_{2n})^2)^{\frac{1}{2}}$
- Manhattan distance: $D(\mathbf{X}_1, \mathbf{X}_2) = |x_{11} - x_{21}| + |x_{12} - x_{22}| + ... + |x_{1n} - x_{2n}|$
- Minkowski distance:$D(\mathbf{X}_1, \mathbf{X}_2) = (|x_{11} - x_{21}|^q + |x_{12} - x_{22}|^q + ... + |x_{1n} - x_{2n}|^q)^{\frac{1}{q}}$

4.2.1.5 Divide-and-Conquer Models

Finally, the divide-and-conquer models, usually represented as decision trees, recursively analyze each attribute and possible cutoff points that result in a subgroup of in-

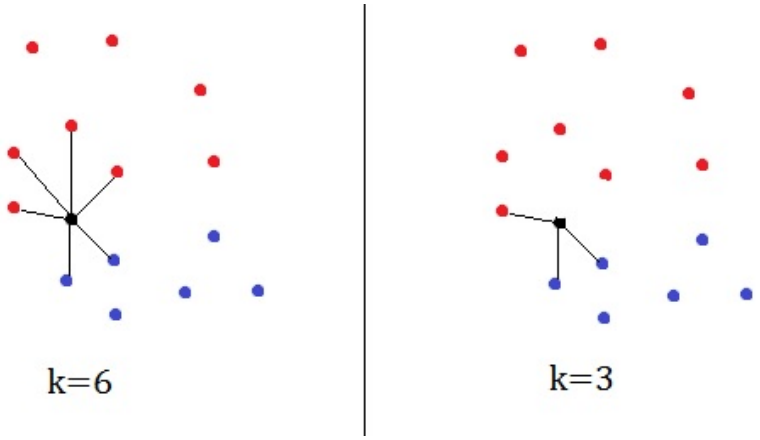


Figure 4.3: k-NN example.

stances that better separate classes, with higher level of purity of the majority class in each subgroup. The algorithm C 4.5 [41] was selected.

The C4.5 is a decision tree created based on the concept of information entropy. It is based in ID3, a previous decision tree generator also developed by Quinlan. ID3 looks in each feature for splitting points that separate the space of features ω in subgroups with greater class purity, i.e. information gain ratio in splitting, calculated as:

$$G = E(\mathbf{S}) - [E(\mathbf{S}_1) + E(\mathbf{S}_2)]$$

$$E(\mathbf{S}) = -\sum_{i=0}^{m-1} p_i log_2(p_i)$$

G is the information gain, $E(\mathbf{S})$ the entropy of a set of instances in a subspace $\mathbf{S} \in \omega$ based on classes distribution, $\mathbf{S}_1$ and $\mathbf{S}_2$ the subsets after splitting $\mathbf{S}$ and p_i is the ratio of instances in S of the i-th class. The algorithm calculates recursively the information gain for splitting points in each attribute and then generates the decision tree. The C4.5 algorithm takes one step further, it prunes the tree for better generalization.

4.2.2 Algorithm's Performance Evaluation

In order to evaluate the classification algorithms there are three usual steps:

- **Train.** The train step uses a training database to fit the hypothesis function's constants.
- **Test.** The test step uses a pre-labeled test database to adjust the algorithm's parameters obtained in the training. The adjustments are used to perform a new

training, and this train-test cycle is executed until the algorithm attains its best performance with the given dataset.

- **Validation.** After training the algorithm to fit the hypothesis' constants, finding the best parameters with train-test cycles, the classifier is supposedly with the higher performance, but this might be an overfitted solution. A third step, the validation, is needed to verify the classifier's generalization. A validation database is used with completely new data for that purpose.

To illustrate the performance measurements made in the test and validation steps let's use an example of a two classes problem. Take the white dots and black dots represented in Figure 4.4 and suppose the classifier divided the classes with the circle, classifying all dots inside the circle as white dots and outside the circle as black dots.

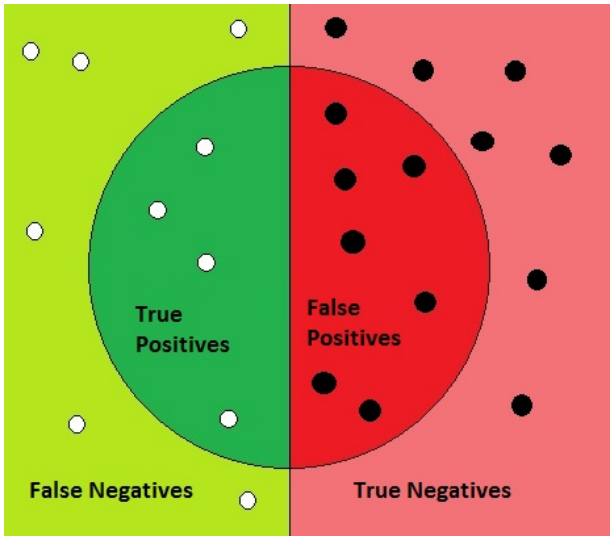


Figure 4.4: Algorithm Performance Measurement Example.

In this example, we use the white dots as our classification target: the white dots correctly classified as white dots are the true positives, represented by the dark green area; the white dots wrongly classified as black dots are the false negatives, the light green area; the black dots wrongly classified as white dots are the false positives, the dark red area; and the black dots correctly classified as black dots are the true negatives, the light red area.

The performance measures taken from such classification are:

- Accuracy (ACC) - The ratio of correct classifications for each class. It represents an overview of the classifier. In the example we have the dots in the dark green area

and light red area as correct classifications over the whole amount of dots.

$$ACC = \frac{11}{24} = 0.46$$

- Precision (P) - The ratio of correct classifications of a given prediction. It represents how accurately the classifier separated a class from another, giving an intuition about how satisfactory is the decision threshold achieved for the observed class. For the white dots it would be the dots in the dark green area over the dots in the circle, dark green and dark red areas;

$$P_{white} = \frac{4}{11} = 0.36$$

- Recall or Sensivity (R) - The ratio of correct classifications of a given class. It represents the classifier's difficulty of adjusting a decision threshold for an observed class. For the white dot class it would be the dark green area dots over all dots in the light and dark green area;

$$R_{white} = \frac{4}{10} = 0.4$$

- F-1 measure (F) - Precision and Recall have related results and even similar interpretations in some cases. They both concern in comparing how a given class actually is and how the classifier perceive it, the first observing the data complexity compared to the classifier's decision threshold and the other the classifier's ability to create complex decision threshold compared to the data. Therefore a single metric for this matter was created taking on account either points of view, given by the harmonic mean of Precision and Recall;

$$F_{white} = \frac{2}{\frac{1}{P} + \frac{1}{R}} = 0.38$$

As more complex problems take place, it is not possible to represent the classifier prediction in an image as in our example. There might be too many classes or several dimensions to represent. A proper form to represent the relationship between predictions and real classes is using the confusion matrix.

In the confusion matrix you represent the predicted classes in the columns and actual classes in rows. In our example we would have the Table 4.1. In the white dots column and white dots row we have the dark green area represented; in the white dots column and black dots row, the light green area; in the black dots column and white dots row the dark red area; and in the black dots column and black dots row the light red area.

	white dots	black dots
white dots	4	7
black dots	6	7

Table 4.1: Confusion Matrix Example

Given these classification algorithms and the performance measurements, the knowledge discovery in databases steps could be started. All the classification algorithms are to be tested, the accuracy and F-1 measure used to evaluate and the confusion matrix will be the representation used to show the results of the best algorithm for the task.

Chapter 5

Methodology

As previously stated, the ReMoTe project team defined the scope of work for MAD, setting a fault dictionary. After this step, this work's proposal was to solve the FDD problem as a classification one, using the machine learning approach. In Chapter 4 some machine learning techniques and performance evaluation methods were discussed.

The machine learning approach usually follows some traditional steps — the so-called five steps for extracting knowledge from a database [17]. As shown in Figure 5.1 these steps are: attribute selection, preprocessing (data cleaning and enrichment), data transformation (if needed), data mining and result evaluation.

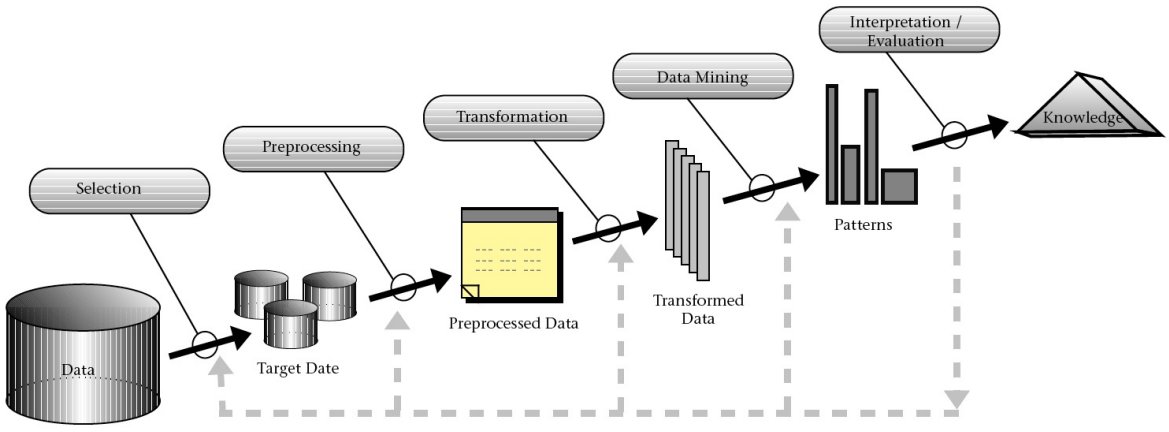


Figure 5.1: An Overview of the Steps That Compose the KDD Process [17].

These steps were created to maximize the algorithm's performance. Raw data usage for training and testing a classifier can lead to a poor result, therefore an attribute selection, preprocessing and transformation might be needed.

In those steps a specialist knowledge is extremely useful, since the specialist can relate attributes' behavior to specific classes and vice versa. The ReMoTe engineers team knowledge was extensively used in those steps and several tests were performed.

5.1 Populating the Database

After the fault dictionary was defined, it was necessary to obtain samples of each of these events occurrences to generate a proper training database. At this stage, controlled tests were performed in the External Mesh Network to inject faults at the network.

Six tests were performed, inflicting faults and monitoring their effects in the collected data. Each test had an observable influence over the parameters. In the following paragraphs, we define these tests and explain how they have influenced the monitored parameters when compared to a node at regular operational state. The tests were:

- **High processor usage -** The objective was to create high demand for CPU usage in the node. To emulate the effect of a large and constant flow of processing requests, a random byte stream was compressed while the average CPU Load was monitored by MeshAdmin (among all other collected node information). As an effect we can observe in Figure 5.2 the constant high requirement of the processor by these processes as soon as the test started (June 30), increasing the CPU Load Average values from below 0.5 to above 2.0 and sustaining it this high.

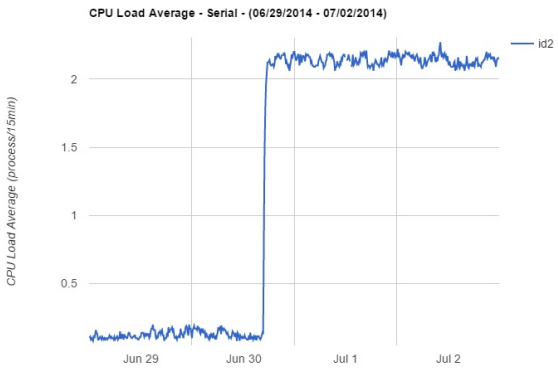


Figure 5.2: CPU Load Average during High Processor Usage test.

- **High RAM memory consumption -** The network nodes used in this work have a reduced amount of available flash memory, used mostly for the operating system installation. When the node is on, most file activities occur in RAM, including

writing into temporary files. Therefore, any faulty process that consumes large amounts of memory may affect core functionalities. On extreme cases, the lack of available RAM may cause node unresponsiveness, which triggers a watchdog to reboot it. Hence, in this experiment we employed a process that would create large temporary files in order to verify how this behavior would be manifested in the data collected by MeshAdmin.

In Figure 5.3 the test result was the available memory decrease. The memory consumption stayed below 1 MB most of the time, with peaks occurring when the node restarted. Usually, at least 1.6 MB of RAM is available under normal conditions, as can be seen in 5.3, from the end of day 22 on, when the test had been ceased.

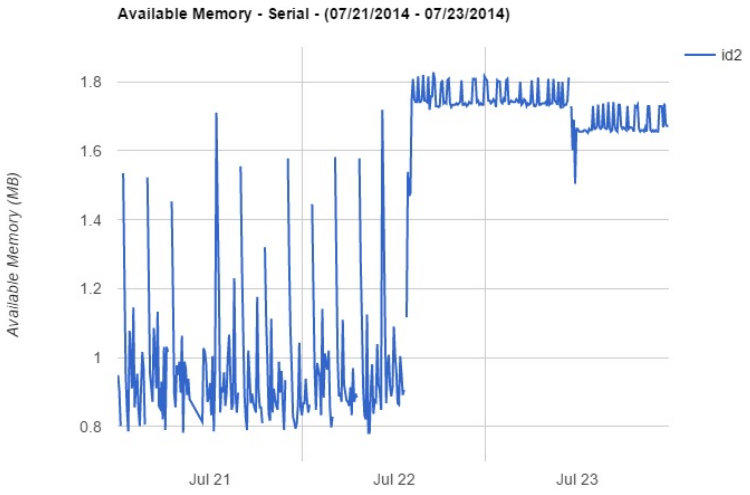


Figure 5.3: Available Memory during High RAM Memory Consumption test.

- **Battery Failure -** The battery pack is comprised of three batteries in parallel. Initial tests consisted of replacing one or more of the good batteries in the pack for defective batteries. Subsequently, additional tests were performed with an incomplete battery bank, with batteries removed gradually. In these tests, a reduction of the node autonomy could be observed, resulting in a shutdown during certain periods of the day. As expected, the node autonomy varied according to the number of defective or missing batteries, but also according to the intensity of sunlight during the period, which affects battery charging during the day.

  In Figure 5.4 there is a plot of two nodes, one with a perfect battery pack (id2) and another with defective batteries (id1). As the sun sets the communication module consumes the batteries' energy and the defective pack discharges almost immediately

with a complete shutdown before 6 PM.

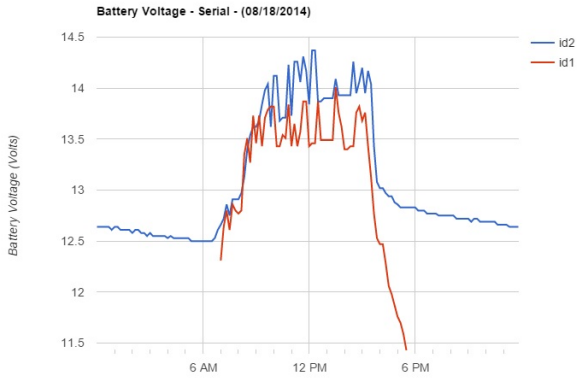


Figure 5.4: Battery Voltage in the Battery Failure test.

- **Low Solar Panel's Efficiency -** Several different tests on the solar panel were performed, all aiming to reduce its efficiency. This was accomplished by casting full and partial shadows over the panel and also by varying the shadow incidence angle. Most tests, indeed, resulted in a lower efficiency of the solar panel, also reducing the node's autonomy.

  The current provided by the solar panel was heavily affected in this process as seen in Figure 5.5. In July 31st, a regular operational day, the current provided by the solar panel had peaks above 2000 mA. As soon as the test started in August 1st, a bulkhead was placed in front of the solar panel and the current provided wasn't much higher than 500 mA, despite the regular solar incidence throughout the day.

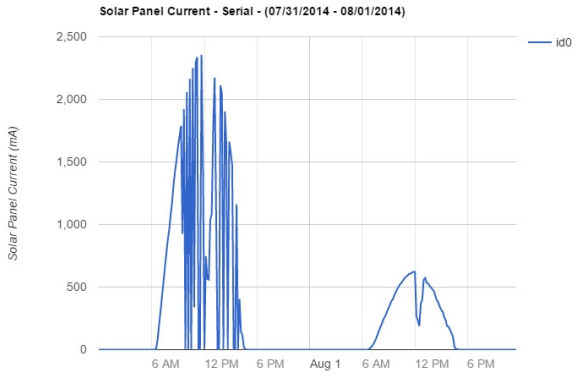


Figure 5.5: Solar Panel Current in the Low Solar Panel's Efficiency test.

- **Antennas Misalignment -** The directional antennas were misaligned to cause a drop on the received signal strength. To register this link quality drop, the ML metric was monitored. During the misalignment tests, the quality level of the tested

link varied, as expected. It was also noted that climate variations during the test had an impact over the metric.

This test affected the ML metric. In Figure 5.6 it is possible to see that before Sep 17th, when the test started, the link quality level was above 0.8 during most of the day. After the antennas were misaligned there's a drop in the quality metric, which stayed under 0.8, during most of the time. Also the ML metric standard deviation was clearly affected, with a more unstable link.

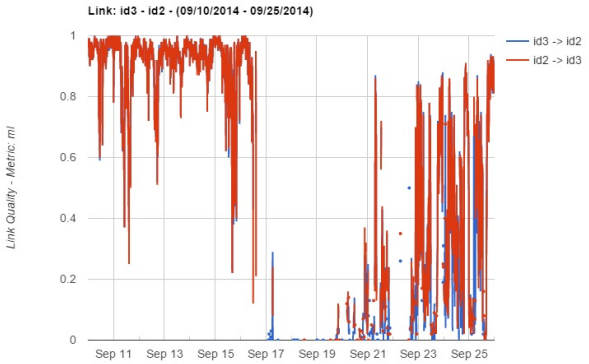


Figure 5.6: ML-Metric in the Antenna Misalignment test.

- **RF Cable Connectors Defect -** The antennas are connected to the router via an N-type male connector. The connector was brought to a poor contact condition, which caused instability in the link quality. In Figure 5.7 it is possible to see that the ML metric average was slightly impacted by the test, but the major effect was an increased standard deviation of the metric when compared to the same link before the tests began in Figure 5.6.

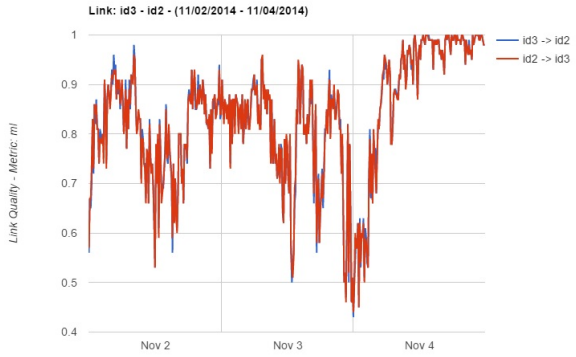


Figure 5.7: ML-Metric in the RF Cable Connectors Defect test.

Those were the performed tests. All network's nodes were being monitored by the MeshAdmin during the whole process and, from that point on, the training and testing databases could be filled.

5.1.1 Databases

After 5 months of tests, 25393 entries were added to the database. The class distribution of the database is shown in Table 5.1 where F is the frequency of the class and Fr the Relative Frequency. Figure 5.8 shows a pie chart with the database composition.

CLASS	F	Fr
Regular Operational State	9432	0,37
High Processor Usage	1010	0,04
High Memory Consumption	1000	0,04
Battery Failure	4032	0,16
Low Solar Panel's Efficiency	3835	0,15
Antennas Misalignment	4215	0,17
RF Cable Connector defected	1869	0,07
TOTAL	25393	1

Table 5.1: Database Class Distribution.

It is important to note that the majority class is the regular operational state class, with 37% of instances. Hence, a simple classification method, which always responds with the majority class (in this case, regular operational state class) would have an accuracy of 37%. Thus, this value will be used as a baseline for other classifiers.

As already stated, tests are used for evaluation purposes, therefore a training database and a test database are both needed. Thus, we had to split the database in two. For that purpose we used the cross-validation method to analyze the training results. This method separates the training database in k groups, of which k-1 groups are used in the algorithm training and the remaining group as a test group to measure the classifier performance. These groups are interchanged between training and testing until each of the k groups is used once for testing.

Using cross-validation may result in a biased result [31]. Thus, the use of stratified sampling to form a test database will diminish this problem and also help to make a proper reproduction of the original database and its characteristics.

As more data are used to train, the results will be closer to the original classifier (the one using the whole available data). Therefore there is a trade-off between the

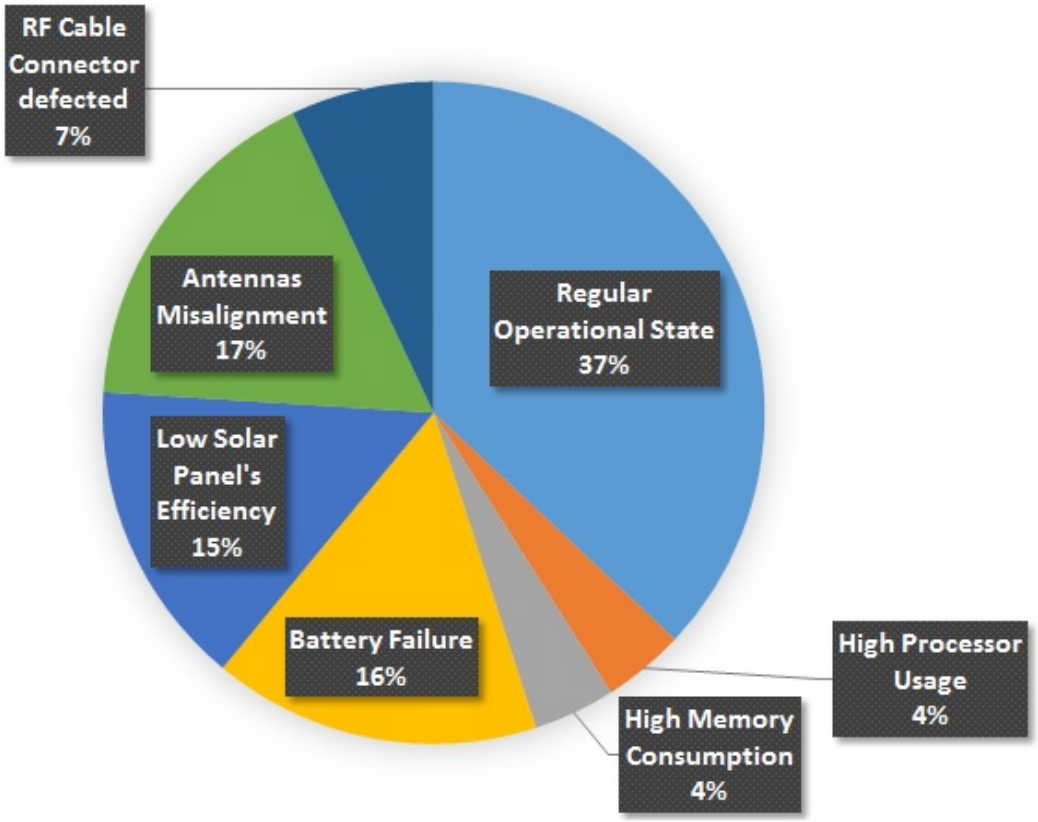


Figure 5.8: Database class distribution.

amount of data used for the test without decreasing the classifier's performance and how representative is your test group. There's no easy answer for that question, but there is a consensus that the more data one has the smaller the test partition can be without loosing generalization. Using 10% of the database for test usually gives good results [34][4], therefore 10-Fold cross-validation will be used to split the database into training and test.

For the Validation database, the network was monitored without any intervention for one month after all tests occurred. In this month it was possible to see the regular operation of the network and one fault incident. One of the mesh nodes had a battery fault during the second half of the month. This node's data will be used to validate the classifier with the best performance measures during test step.

5.2 Selecting Data

The task of learning is preceded by the definition of concepts, as, for example, the concept of node state. The node states defined in this work are: in regular operational state; or in any of those failure states set in the fault dictionary. This concept definition

task can be divided into two subtasks: decide which features will be used to describe it and deciding how to combine these features. Therefore, the selection of relevant attributes is a central problem. It simplifies a model, reduces training time and enhances generalization [21].

In Section 5.1, before the training and test databases definition, the impact of each of the nodes' states in the the monitored parameters were evaluated. A comparison of the node behavior and it's measured parameters before and during tests was performed. This way, it was possible to infer the most relevant parameters using the MeshAdmin graphical analysis tools. This process resulted in an initial selection of a group of relevant parameters: average CPU load, available flash memory, solar panel current, battery voltage and link quality.

In order to evaluate if other parameters not considered could affect the diagnosis the following methodology was employed: The parameters were progressively removed from the group, forming a new candidate group, if the parameter removal didn't have an impact over the accuracy, the parameter was definitely discarded. By repeating this process until all available parameters were either discarded or preserved, a final group of relevant parameters could be obtained.

5.3 Preprocessing and Data Transformation

After selecting the parameters, the steps of preprocessing and data transformation were performed. Long periods of node inactivity, resulting in unusable information, spurious data and other known issues, such as infrastructure network problems affecting the server, were removed from the training database.

After the primary data cleaning was performed, some basic recommended steps were taken, as scaling data [24]. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. The data was then normalized.

The data enrichment and transformation process was based on the description of a time continuous phenomena, like battery charge and discharge. Notice that the gathered data is composed of instantaneous samples of the monitored parameters, while and nodes' failures progress over time. Hence, it was necessary to define a long enough observation interval comprising several samples so that the tuples in the training database were representative of the failures.

To this end, the mean and standard deviation of the parameters were evaluated in the following intervals: 1 hour, 2 hours, 4 hours, 8 hours, 12 hours and 24 hours. To evaluate which interval was more adequate, we used the same methodology employed in the selection step.

The final transformation step was taken to enable the use the Weka software in the algorithm evaluation step. The entry file for Weka has a standard format: the Attribute-Relation File Format (ARFF), which is comprised of an ASCII text file that describes a list of instances sharing a set of attributes.

ARFF files have two distinct sections. The first section is the Header information, which is followed by the Data information. The Header of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types. The Data section of the file contains the data declaration line and the actual instance lines.

An example of an ARFF file dataset used for training looks like this:

```
% 1. Title: Autonomic Diagnose Module Database
%
% 2. Source:
%      (a) Author: Vinicius Ferreira
%      (b) Owner: MidiaCom Laboratory (contato@midiacom.uff.br)
%      (c) Date: December, 2015
%
@RELATION mesh_nodes_measurements

@ATTRIBUTE date DATE ‘‘HH:mm’’
@ATTRIBUTE rebooted {yes, no}
@ATTRIBUTE cpu_load_average NUMERIC
@ATTRIBUTE battery_voltage NUMERIC
@ATTRIBUTE class {Regular_State, High_CPU_Usage, Battery_Failure}

@DATA
03:50, no, 0.14, 11.87, Regular_State
08:30, no, 0.23, 14.35, Regular_State
10:50, yes, 2.11, 12.24, High_CPU_Usage
18:50, yes, 0.14, 11.63, Battery_Failure
```

```
06:30, yes, 0.12, 11.57, Battery_Failure
```

Chapter 6

Results

The decisions made in Chapter 5, such as: the baseline set in about 40% of accuracy; the search for a definitive group of selected attributes; the attribute transformation from an instantaneous observation to a defined time observation; were all evaluated with the algorithms' performance.

The results presented in this chapter are divided in three parts (sections):

- **Problem Description:** Comprises the attribute selection and transformation. The result of this part is the determination of the best way to use the data and represent the real problem.
- **Algorithm Evaluation:** With data representation set, each classifier could be created using the selected algorithms and then tested using the described evaluation method.
- **Results Interpretation:** The best algorithm will be further detailed, its confusion matrix shown, its results validated and the expectations for future work detailed.

6.1 Problem Description

The first investigation was about the observation interval, and the Decision Table was used for that purpose. This choice was based in the algorithm simplicity — it doesn't have any parameter adjustments —, and also because it has an attribute selection embedded. Therefore, the lack of a previous attribute selection doesn't affect its results as it would affect other algorithms. Later, when the attribute selection step takes place, each attribute will already be in its best representation.

The observation intervals were: 1h, 2h, 4h, 8h, 12h and 24h. The Decision Table accuracy results were depicted in Figure 6.1. As the observation interval increased, the algorithms accuracy also increased, but they didn't result in a significant change of the accuracy. The tests stopped with 24 hour observation period as the highest accuracy result.

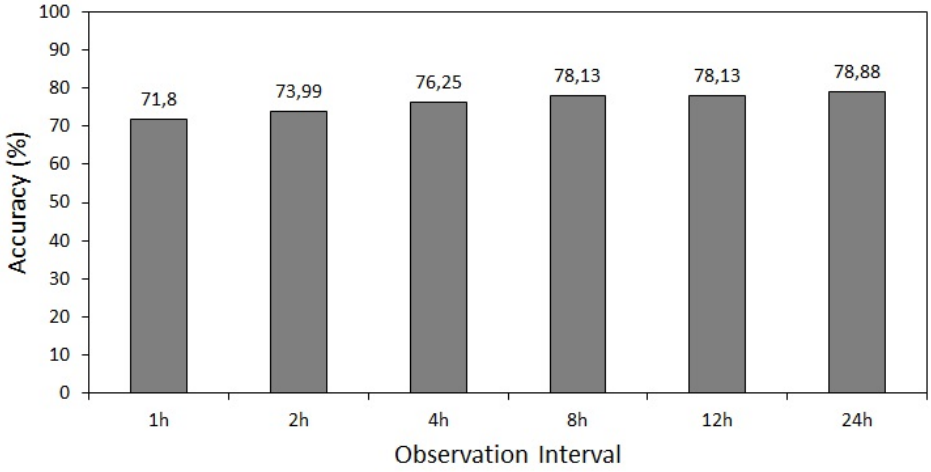


Figure 6.1: Decision Table accuracy per observation interval.

This result is consistent with the nature of the problem, since it accounts for a full day (e.g., a full cycle of battery recharge due to sunlight is comprised, as well as a full cycle of discharge at night). Some of the faults only manifest themselves in certain periods of the day. For example, a battery failure is probably not detectable during day, while the solar panel is capable of supplying energy to the system. However, during the night, the failure becomes apparent with a faster discharge of the batteries. Hence, the use of a 24-hour period ensures that the relevant period of the day is represented in the database instances.

In summary, each instance of the training database is comprised of the average and standard deviation of the samples during the past 24-hour window. Each instance also contains the timestamp of the last sample from the observed period.

The next step is the attribute selection. Several candidate groups were tested and the C4.5 algorithm was used to evaluate this test. The C4.5 is a tree algorithm that, as described, evaluates the potential class entropy reduction for each attribute. Therefore, it can be seen as an attribute selection algorithm embedded method, since it already assesses all attributes and only uses the relevant ones. The C4.5 was used with the confidence factor (CF) set to its default value, 0.5.

Monitored parameter	24-hours standard deviation	24-hours average
Solar Panel Voltage	No	No
Solar Panel Current	No	Yes
Battery Voltage	Yes	Yes
Communication Module Voltage	No	No
Communication Module Current	No	No
External Temperature	No	No
Internal Temperature	No	Yes
Incident Light Intensity	No	Yes
Available Memory	No	Yes
CPU Load	No	Yes
Link Quality	Yes	Yes
Network Interfaces Traffic	No	No

Table 6.1: Selected attributes for algorithm evaluation.

Initially, the group of all attributes were: date, uptime, the average and standard deviation over the last 24h of: battery voltage, solar panel voltage, solar panel current, router voltage, router current, internal (sealed-box) temperature, external (ambiance) temperature, incident light intensity, cpu load, disk space, available memory, each network interface input and output, and link quality.

All these attributes are listed in Table 6.1. The selected attributes (the 24-hour standard deviation and 24-hour average of each monitored parameter) are marked as Yes, while the discarded attributes are marked as No.

The final relevant group to represent each failure and node state were: date, uptime, average and standard deviation of battery voltage, average of solar panel current, average internal temperature, average of incident light intensity, average of cpu load, average of available memory, average and standard deviation of link quality.

With the description problem set, the attributes selected and properly represented, the search for the most suited classifier for the task can be made.

6.2 Algorithm Evaluation

The selected algorithms were applied to the training database and evaluated. The k-NN, C4.5 and SVM algorithms have parameters to be set while the Decision Table and Naive Bayes do not have any parameters. For k-NN, the k parameter corresponds to the size of the group which will be used to define the class of the target instance. For the C4.5 algorithm, the parameters are the minimum number of instances per leaf of the created

Class	F-1 Measure (%)
Regular Operational State	52,25
High Processor Usage	52,41
High Memory Consumption	50,87
Battery Failure	44,09
Low Solar Panel's Efficiency	43,83
Antennas Misalignment	51,03
RF Cable Connector defected	51,67

Table 6.2: Naive Bayes F-1 Measure.

decision tree and the confidence factor (CF), a parameter used to prune the tree. For the SVM the C parameter, indicating the complexity and limits for the λ_i solution, and also a definition of the kernel function were used.

The first algorithms to be evaluated were the non-parametric ones, Naive Bayes and Decision Table. After these algorithms the three parameter based algorithms were tested. This way it was possible to find the best settings for them and compare it with the other classifiers. The first evaluation metric used was the accuracy to define the best classifier. The F-1 Measure was used in the result interpretation step.

6.2.1 Naive Bayes

To use the Naive Bayes algorithm one more step had to be taken: data discretization. This step was performed using the discretization algorithm embedded in Weka [27], which uses an information entropy minimization heuristic as a tree algorithm. The Naive Bayes obtained an accuracy of 49.50%. This result was only slightly better than the baseline (37%) and it will be used for comparison purposes.

In Table 6.2 the F-1 Measure of each class is presented and it is possible to observe a higher complexity in detecting the Battery Failure and Low Solar Panel's Efficiency while the other classes have a slightly better result.

Class	F-1 Measure (%)
Regular Operational State	80,49
High Processor Usage	78,99
High Memory Consumption	71,29
Battery Failure	75,36
Low Solar Panel's Efficiency	74,92
Antennas Misalignment	80,94
RF Cable Connector defected	81,64

Table 6.3: Decision Table F-1 Measure.

6.2.2 Decision Table

As already depicted in Section 6.1, the Decision Table obtained an accuracy of 78.88%. The result was well above the accuracy of the baseline and differently from the Naive Bayes result, this one was considered a good option for the desired task.

The F-1 Measure is presented in the Table 6.3. In this classifier the Battery Failure and Low Solar Panel's Efficiency are also slightly lower than the others but the lower point in F-1 Measure is the High Memory Consumption.

This was an unexpected result since the detection of a Battery Fault is considered the most complex problem due all variables involved, as the battery natural charge/discharge cycle, the climate influence in the charging cycle and node's energy consumption in the discharge cycle. While the High Memory Consumption is one of the simplest, possibly diagnosed with just one attribute, the available memory average.

6.2.3 k-NN

Another algorithm tested was the the k-NN. In Figure 6.2 the best accuracy result for this algorithm was found when using k=1.

Despite being superior to the baseline, the 1-NN result was considered much lower than expected when compared to the other algorithms. The 1-NN results were similar to the Naive Bayes, hence, there was no further investigations using this classifier and its accuracy will be used just for comparison purposes.

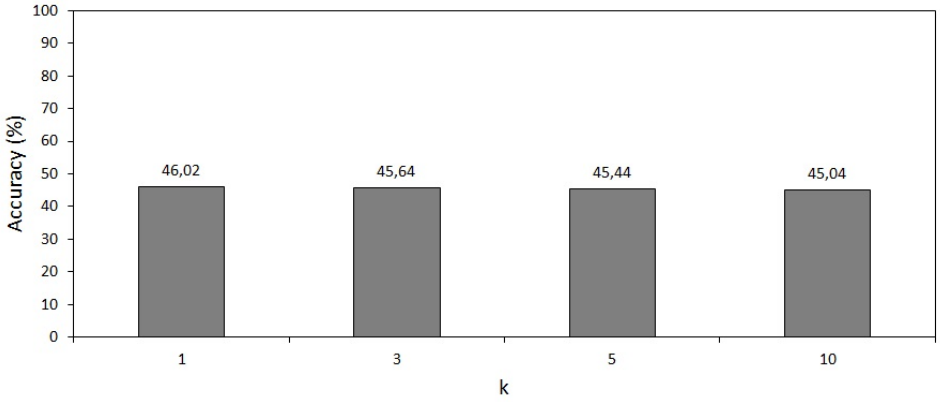


Figure 6.2: k-NN accuracy per k parameter.

6.2.4 SVM

For the SVM there are two settings to be made: the kernel function and the C parameter. Initially the C parameter was set to 1 and the kernel functions were evaluated.

The kernel variation results, in Figure 6.3, were in favor of a linear kernel function. Therefore this was the kernel used to find the C parameter with best performance.

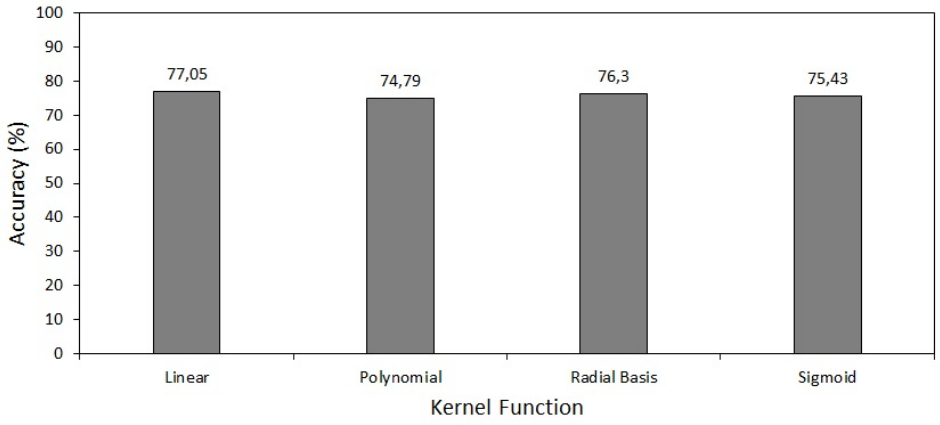


Figure 6.3: SVM accuracy per kernel function with C = 1.

With the kernel function set with a linear function, the search for the parameter C took place. After C=100 the results didn't improve significantly while the processing time suffered a large increase. The search was stopped in C=100, which represents the result, and the accuracy increased with the C parameter as in Figure 6.4. The best accuracy of 90.59%, achieved with linear kernel function and C=100, will be used in the overall classifiers comparison. Observing the F-1 Measure in Table 6.4 it is possible to verify the high complexity in diagnosing the Battery Fault. Therefore this might be a weakness of this classifier.

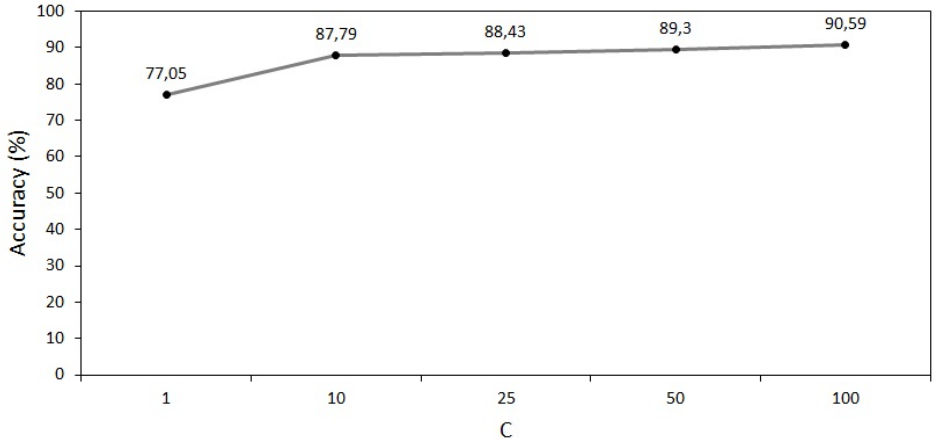


Figure 6.4: SVM accuracy per C parameter with linear kernel function.

Class	F-1 Measure (%)
Regular Operational State	94,9
High Processor Usage	99,9
High Memory Consumption	93,6
Battery Failure	87,6
Low Solar Panel's Efficiency	93,1
Antennas Misalignment	90,9
RF Cable Connector defected	90,7

Table 6.4: SVM F-1 Measure.

6.2.5 C4.5

The C4.5 algorithm minimum instances per leaf was set in 2, a small value, and the confidence factor was varied. The tests have shown that the confidence factor variation did not have a significant impact over the accuracy, as in Figure 6.5.

As observed, the C4.5 algorithm presented a good result. The accuracy of 88.45% will be used as the C4.5 score, with a set of CF=0.5 and 2 instances per leaf. The F-1 Measure result in Table 6.5 shows the same weak points as the majority of the classifiers, the Battery Failure and Low Solar Panel's Efficiency.

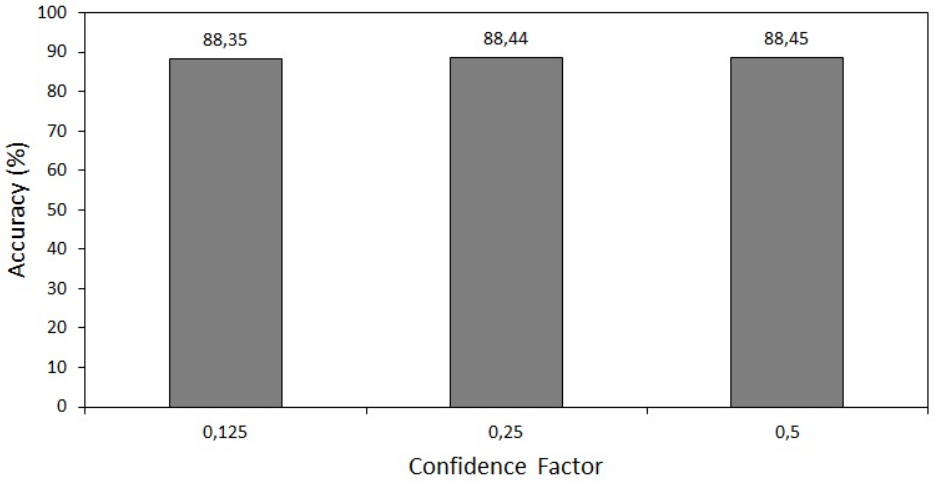


Figure 6.5: C4.5 accuracy per confidence factor (CF).

Class	F-1 Measure (%)
Regular Operational State	88,19
High Processor Usage	88,28
High Memory Consumption	85,98
Battery Failure	81,47
Low Solar Panel's Efficiency	81,91
Antennas Misalignment	87,92
RF Cable Connector defected	87,57

Table 6.5: C4.5 F-1 Measure.

6.3 Results Validation and Interpretation

The final step is to compare all the classifiers and choose the best ones to validate. The best classifiers generated with each algorithm are now compared in Figure 6.6. The highest accuracies were achieved by SVM and C4.5 algorithms with marks of 90.59% and 88.45% respectively.

The C4.5 classifier was the first to be validated with the data for node id0, collected during January. This validation database had two distinct phases, the first fortnight, in which the node appeared to be in perfect state, and the second fortnight, the moment the battery fault occurred.

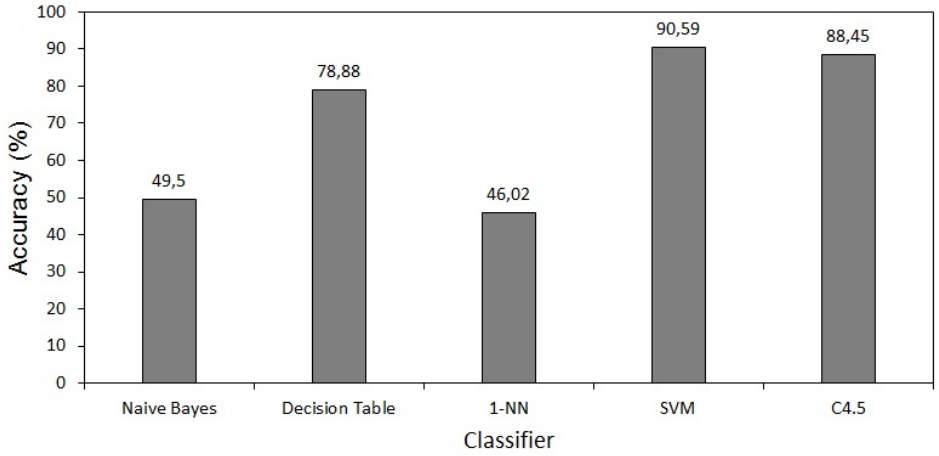


Figure 6.6: Overall classifier comparison.

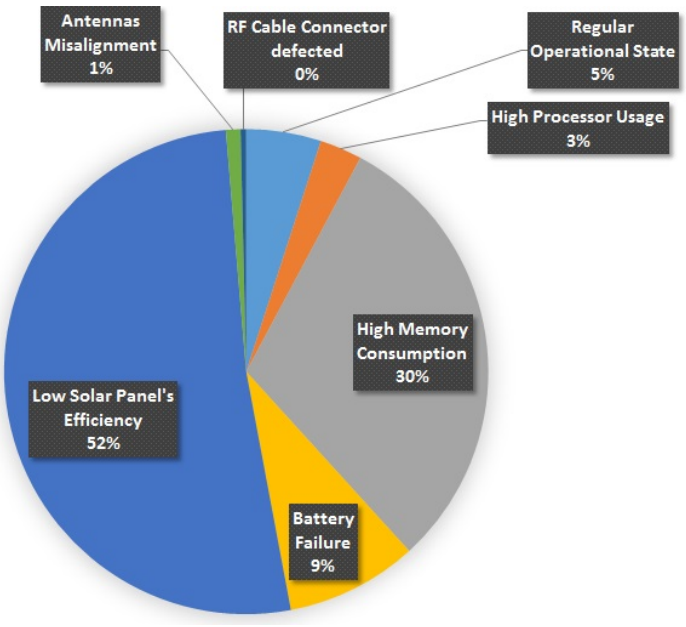
The results were not as expected, as shown in Figure 6.7. The real accuracy was under 65%, which indicates an overfitting to the training data.

The SVM algorithm responded as expected in the first fortnight, classifying 85% of the entries as regular operational state. For the second fortnight there was a problem, 40% of the classifications were indeed for battery fault and 37% were for low solar panel's efficiency. As previously presumed during SVM's F-1 measure analysis, the distinction between battery fault and low solar panel's efficiency was proved a weak point.

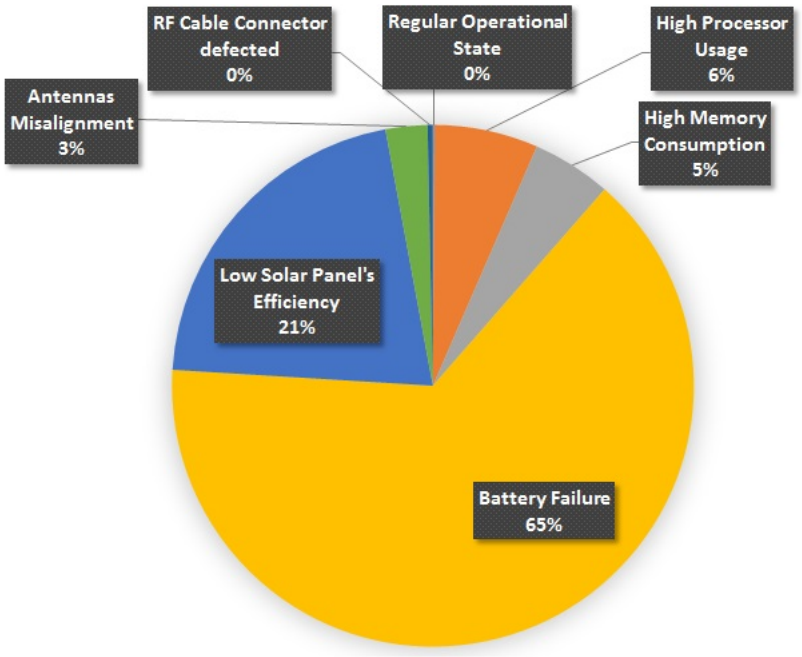
As already expected by the F-1 Measure observation, the Battery Fault is a state where the classifier tend to perform poorly. Another indicator of this result was the confusion matrix for the SVM classifier presented in Table 6.6. In this table, the columns are the real classes, while the rows are the classifier predictions.

Through the confusion matrix, it is possible to note a correlation between battery failure and low efficiency of the solar panel. This correlation between these diagnoses represents a possibility of error, as already seen in the validation. However, the solution presented high reliability indicators, such as high accuracy and F-1 Measure.

The ReMoTe team considered this a good result, considering the high complexity in diagnosing a Battery Fault. A battery fault diagnose usually takes advantage of a battery current sensor as a fundamental asset to verify the battery capacity, instrument not used in this work. Another noticed point was Battery Failure's different causes, such as lower capacity, higher inner resistance, among others [16]. In the database population tests only the high inner resistance was considered, while the problem occurred in this validation was a lower capacity one, which might have led to this misinterpretation.



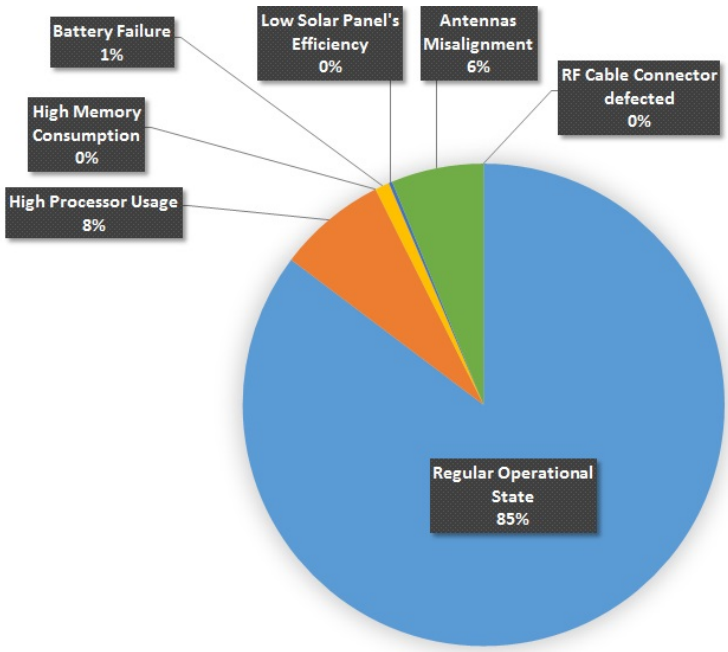
(a) First fortnight classifications



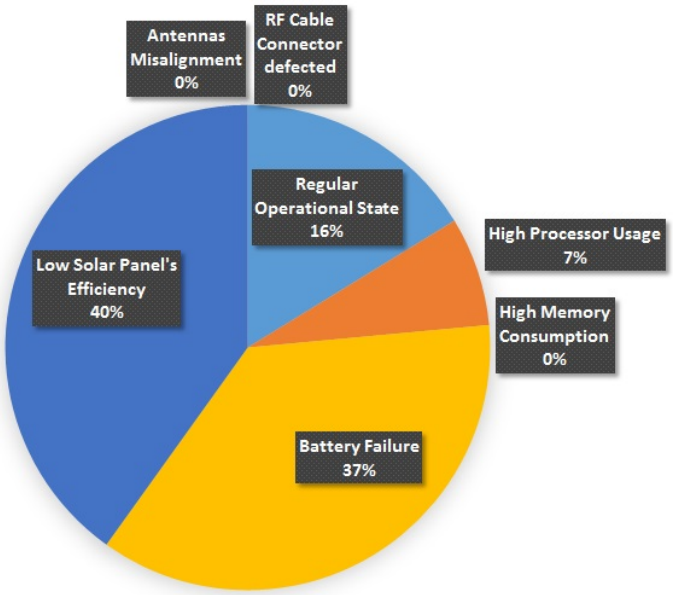
(b) Second fortnight classifications

Figure 6.7: C4.5 Validation Performance.

Analyzing the likelihood of the output of each class by the classifier, it was possible to see that in the Low Solar Panel's Efficiency classification cases the likelihood of this diagnose was around 60% of certainty while, for these same cases, the probability for Battery Fault was 30%. The ReMoTe Team then decided to use as output the two most probable causes of the diagnose and their likelihood as a final result, to workaround this problem.



(a) First fortnight classifications



(b) Second fortnight classifications

Figure 6.8: SVM Validation Performance.

This way the autonomic diagnosis solution for SWMNs is feasible and satisfactory. The generated SVM classifier module will be integrated to MeshAdmin and tested for possible adjustments and deployment in the ReMoTe production network.

	regular operational state	high processor usage	high RAM memory consumption	battery failure	low efficiency of the solar panel	antennas misalignment	defects on the RF cable connectors
regular operational state	8148	0	0	4	5	0	0
high processor usage	4	863	0	0	0	0	0
high RAM memory consumption	17	0	812	3	19	3	0
battery failure	15	0	2	3172	295	5	0
low efficiency of the solar panel	9	0	2	216	3436	1	0
antennas misalignment	0	0	0	6	1	4049	31
defects on the RF cable connectors	0	0	0	0	0	3	1724

Table 6.6: SVM Confusion Matrix

Chapter 7

Conclusions

This work presented a proposal for an autonomic solution for fault detection and diagnosis (FDD) for SWMNs, using artificial intelligence techniques.

To set the artificial intelligence approach a reduced scope of work was determined by the creation of a fault dictionary. As the fault dictionary was created, the problem could be seen as a pattern recognition problem, more specifically, a classification problem. Therefore, machine learning techniques for the classification problem were analyzed.

Several classification algorithms were considered and studied in the process. A performance evaluation method had to be defined and the search for the classifier most suitable for the task has been carried on.

For this purpose, the steps for knowledge discovery in databases were followed. Five months of network tests were made to populate the database generating a set of examples to be worked upon in order to produce training, test and validation databases.

A treatment of the data has followed, in the search for the best way to understand and represent the real phenomena with the gathered data. The data had to be transformed and selected to achieve the best possible results to this work's purpose. As an outcome, of all 38 initially considered attributes to represent the problem, only 11 were considered relevant through a 24 hours observation period of averages and standard deviations.

With the databases formed, we compared a number of well-known classification algorithms for the problem, namely Naive Bayes, Decision Table, k-NN, SVM and C4.5. The result of this evaluation showed that the C4.5 and the SVM algorithms had the best overall prediction performances, with accuracy over 80%. This accuracy levels indicates that an autonomic solution is, indeed, feasible.

Their result was brought to a validation test. In this test, the C4.5 presented over-

fitting characteristics, with poor results when new data was used. While the SVM has shown a good overall performance, but an already expected weak point, the Battery Failure detection. This problem was solved using a multi-classification solution — the two classes with higher likelihood of success are presented to the user. With this adjustment, the classifier presented the correct diagnosis (between the two indicated) in all cases and the work was considered a success.

As future work, there is an intention to integrate this solution to MeshAdmin so that MeshAdmin can provide automatic warnings about detected failures on network nodes before they cause disruptions in the communication services. As a final goal, this solution will be implemented in the production network of the ReMoTe project.

References

[1] Mohammad Abu Alsheikh, Shaowei Lin, Dusit Niyato e Hwee-Pink Tan. Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *Communications Surveys & Tutorials, IEEE*, 16(4):1996–2018, 2014.

[2] David W Aha, Dennis Kibler e Marc K Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.

[3] Ian F Akyildiz, Xudong Wang e Weilin Wang. Wireless mesh networks: A survey. *Computer Networks*, 47(4):445–487, 2005.

[4] Yoshua Bengio e Yves Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *The Journal of Machine Learning Research*, 5:1089–1105, 2004.

[5] Mario Bkassiny, Yang Li e Sudharman K Jayaweera. A survey on machine-learning techniques in cognitive radios. *Communications Surveys & Tutorials, IEEE*, 15(3):1136–1159, 2013.

[6] Ismail Butun, Salvatore D Morgera e Ravi Sankar. A survey of intrusion detection systems in wireless sensor networks. *Communications Surveys & Tutorials, IEEE*, 16(1):266–282, 2014.

[7] Ricardo C Carrano, Luiz Magalhaes, Débora C Muchaluat Saade e Célio VN Albuquerque. IEEE 802.11s multihop MAC: A tutorial. *Communications Surveys & Tutorials, IEEE*, 13(1):52–67, 2011.

[8] Ricardo Campanha Carrano, Diego Passos, Luiz Magalhães e Célio V.N. Albuquerque. A comprehensive analysis on the use of schedule-based asynchronous duty cycling in wireless sensor networks. *Ad Hoc Networks*, 16:142–164, 2014.

[9] Chih-Chung Chang e Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[10] Cisco Visual Networking Index Cisco. Global mobile data traffic forecast update. 2014–2019 (white paper), 2015.

[11] William W Cohen. Fast effective rule induction. In *Proceedings of the twelfth international conference on machine learning*, pp. 115–123, 1995.

[12] Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *Electronic Computers, IEEE Transactions on*, (3):326–334, 1965.

[13] Rafael De Tommaso do Valle e Débora Christina Muchaluat-Saade. Meshadmin: An integrated platform for wireless mesh network management. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pp. 293–301. IEEE, 2012.

[14] Jorge Luís Machado do Amaral. *Sistemas imunológicos artificiais aplicados à detecção de falhas*. PhD Thesis, PUC-Rio, 2006.

[15] Jorge Duarte, Diego Passos, Rafael L Valle, Eunice Oliveira, Débora Muchaluat-Saade e Célio V Albuquerque. Management issues on wireless mesh networks. In *Network Operations and Management Symposium, 2007. LANOMS 2007. Latin American*, pp. 8–19. IEEE, 2007.

[16] Ahmed Fasih. Modeling and fault diagnosis of automotive lead-acid batteries. 2006.

[17] Usama Fayyad, Gregory Piatetsky-Shapiro e Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[18] Vinicius Ferreira, Ricardo Carrano, Joacir Silva, Diego Passos e Célio Vinicius Neves de Albuquerque. Diagnóstico de falhas em redes em malha alimentadas por energia solar através de mineração de dados. *XX Workshop de Gerência e Operação de Redes e Serviços - SBRC*, pp. 53–66, 2015.

[19] Michael C Ferris e Todd S Munson. Interior-point methods for massive support vector machines. *SIAM Journal on Optimization*, 13(3):783–804, 2002.

[20] Anna Förster. Machine learning techniques applied to wireless ad-hoc networks: Guide and survey. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pp. 365–370. IEEE, 2007.

[21] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann e Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[22] J. Kamer Han e J M. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3rd edition, 2011.

[23] Trevor Hastie, Robert Tibshirani, Jerome Friedman e James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[24] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin et al. A practical guide to support vector classification. 2003.

[25] Guang-Bin Huang, Dian Hui Wang e Yuan Lan. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, 2(2):107–122, 2011.

[26] IBM Corporation. IBM SPSS Modeler. `http://www-01.ibm.com/software/analytics/spss/products/modeler/`, 2015. Acessed: 08/17/2015.

[27] Keki B Irani e Usama M Fayyad. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the International Joint Conference on Uncertainty in AI*, volume 334, pp. 1022–1027, 1993.

[28] Rolf Isermann. Supervision, fault-detection and fault-diagnosis methods - an introduction. *Control engineering practice*, 5(5):639–652, 1997.

[29] Leslie Pack Kaelbling, Michael L Littman e Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, pp. 237–285, 1996.

[30] Ron Kohavi. The power of decision tables. In *Machine Learning: ECML-95*, pp. 174–189. Springer, 1995.

[31] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pp. 1137–1145, 1995.

[32] Daniel T Larose. *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.

[33] José Marinho e Edmundo Monteiro. Cognitive radio: survey on communication protocols, spectrum decision issues, and future research directions. *Wireless Networks*, 18(2):147–164, 2012.

[34] Marianthi Markatou, Hong Tian, Shameek Biswas e George M Hripcsak. Analysis of variance of cross-validation estimators of the generalization error. *Journal of Machine Learning Research*, 6:1127–1168, 2005.

[35] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1st edition, 1997.

[36] Luiz Satoru Ochi, Carlos Rodrigo Dias e Stênio S Furtado Soares. Clusterização em mineração de dados. *Instituto de Computação-Universidade Federal Fluminense-Niterói*, 2004.

[37] Oracle Corporation. Oracle Data Mining. `http://www.oracle.com/technetwork/database/options/advanced-analytics/odm/index.html?ssSourceSiteId=otnru`, 2015. Acessed: 08/17/2015.

[38] Sinno Jialin Pan e Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[39] Diego Passos, Célio Vinicius N de Albuquerque, Miguel Elias M Campista, Luís Henrique MK Costa e Otto Carlos MB Duarte. Minimum loss multiplicative routing metrics for wireless mesh networks. *Journal of Internet Services and Applications*, 1(3):201–214, 2011.

[40] Lili Qiu, Paramvir Bahl, Ananth Rao e Lidong Zhou. Troubleshooting wireless mesh networks. *ACM SIGCOMM Computer Communication Review*, 36(5):17–28, 2006.

[41] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[42] Nithya Ramanathan, Kevin Chang, Rahul Kapur, Lewis Girod, Eddie Kohler e Deborah Estrin. Sympathy for the sensor network debugger. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pp. 255–267. ACM, 2005.

[43] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pp. 41–46. IBM New York, 2001.

[44] J Schmidhuber. A general method for multi-agent learning and incremental self-improvement in unrestricted environments. *Evolutionary Computation: Theory and Applications. Scientific Publ. Co., Singapore*, 1996.

[45] Bruno Siqueira, Diego Passos, Debora Christina Muchaluat-Saade e Celio Albuquerque. Libr: Id-based routing for linear wireless mesh networks. In *Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE*, pp. 461–466. IEEE, 2015.

[46] R Core Team. R: A language and environment for statistical computing. r foundation for statistical computing, vienna, austria. 2013, 2014.

[47] I. H. Witten, F. Eibe e M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 3rd edition, 2011.

[48] Ming Xue e Changjun Zhu. A study and application on machine learning of artificial intellligence. In *Artificial Intelligence, 2009. JCAI'09. International Joint Conference on*, pp. 272–274. IEEE, 2009.

[49] Tevfik Yücek e Hüseyin Arslan. A survey of spectrum sensing algorithms for cognitive radio applications. *Communications Surveys & Tutorials, IEEE*, 11(1):116–130, 2009.

[50] Yongguang Zhang e Wenke Lee. Intrusion detection in wireless ad-hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 275–283. ACM, 2000.

[51] Xiaojin Zhu e Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.