# Chapter 1

# Abstract

**Keywords**

Powerline communications,Supervised machine learning, Appliances Signatures.

**Abstract**

With the current available high data transmission in Powerline systems, a remarkable attention has been shifted to use Powerline communication systems. It is considered to be a preferred choice over Ethernet or Wireless Home communication systems. The later is due to the fact that Powerline wires and AC outlets have always been installed in each home. So its advantages are various as it can also be a low-cost, reliable, secured and efficient option comparing to Ethernet or wireless hardware. Powerline Communication (PLC) systems are a potential alternative to use Internet, home Automation, and electricity consumption by the customers. These advantages and more benefit both the consumers and energy providers.

Power line communication is an emerging technology in the field of communications that aims to use Powerline as a medium to bi-directionally transfer data. However, the communication still faces different distortions which cause inefficient data transmission. The sources of these distortions or noise are caused by existing home appliances and devices. Therefore, in our thesis project, several experiments with powerline noise sources (appliances) are conducted with reporting the novelty of detection. Results and performance are presented based on the findings of the current acquired dataset and Machine Learning (ML) algorithms. Existing Powerline appliances High Frequency Noise and Modulation schemes have been analyzed in order to find the best Machine Learning algorithm to classify the noise sources.

Detecting anomalies in frequency-series data is a problem of great practical interest in powerline communication and signal processing applications. This thesis serves as a general practical and theoretical reference on Power-Line Communication (PLC) systems. It serves with a comprehensive presentation and

analysis on the current up-to-date powerline communication anomalies, modulations, Machine Learning Algorithms and associated development in PLC communication systems. The development in this subject has been relatively scarce, and information to gather datasets were very limited. Therefore, this research represents a collective information from different research papers and scientific patents.

This paper is based on finding the best ML model based on Powerline Communication anomalies traffic patterns caused by appliances' noise. Several tests are performed using various Machine Learning Algorithm models and the results have been documented to help assess future PLC researches and communication systems breakthroughs. The thesis concludes with the best model that first trained to recognize all N different appliances' Signal-To-Noise (SNR) frequencies with the best performing accuracy; then be able to predict for new appliances' SNR frequencies and other analogous signals residing in the powerline. This thesis leverages the trend towards more efficient powerline communication for customers using their electrical powerline infrastructure as a communication medium. The later introduces many benefits to PLC technology providers to harden their communication technology against various PLC disturbances of typical electric loads, or predict PLC network failures or likability to fail. The research also underlines the limitations and challenges that have been witnessed during implementation and testing. Other important factors with PLC data acquisition have been also briefly presented.

# Chapter 2

# Introduction

This chapter serves as a general introduction to the following chapters. This chapter is divided into three parts. In part 2.1, the general introduction describes the overall overview of powerline networks with a brief description of its current advantages and disadvantages. This chapter also highlights the basic concepts and PLC technologies that are related to the thesis. In part 2.2, the project specification clarifies the main objectives and the scope of this master thesis. Finally, in part 2.3, the organization of this master thesis is briefed into multiple main well-guided chapters.

## 2.1 General Introduction

In communication technology, there are two different types of communication infrastructure. One is a wired and the other is wireless communication technology. In the present time of high speed of technology advancement, the current focus in both fields of communication and Machine Learning have been massively vocalized. Communication in several mediums are being used which include telephone wires, Ethernet cabling, wireless and satellite communication technologies. However, each of these mediums has its limitations. In this thesis, it will focus solely on wired communication technology; namely Powerline wired communication in home. Powerline wires have been used solely for electric transmission in most of homes globally. However, the recent emergence of the various communication technologies which focus on reliability and cost efficiency, communication industries steered the attention to powerline communication. Wired Powerline Communication - PLC technology can simultaneously transmit electrical power and data for short and long distances[1]. PLC is also known as power-line digital subscriber line (PDSL), powerline carrier, powerline telecommunications and power-line networking (PLN). PLC provides a new communication path that does not allow obstacles such as buildings, hills and basements that usually block wireless communications. As PLC is getting more attention these days since it is used in Internet of Thing (IoT), new green

energy, smart grid devices and smart cities, different purposes in which PLC could enhance communication and networks performance.

One of the further advantages of using Powerline cabling as a data transmission medium is the fact that each home owner has already installed and connect the powerline cables to the power grid. Powerline Communication uses the existing AC electrical wiring that each appliance in the house uses to power its circuits. Powerline cables provides a high speed communication medium almost everywhere in the AC outlet [1]. It is reliable and cost efficient using the existing electrical wiring than trying to install new Ethernet wires in the house. The potentials of PLC of delivering not only electricity, but also signals, high-speed data and multimedia content [1]. The development in PLC has been conducted in the recent researches, but the information gathered is still dispersed due to the lack of collective knowledge that illustrates clearly the existing technology.

In this thesis, the terms appliances and devices both indicate home electrical devices plugged into powerline AV outlet. Furthermore, the term noise and disturbance both indicate the same concept. In addition to Broad Band Communication - BB, or Narrow Band Communication - NB, all refer to in-home low voltage electrical network [2]. In short, these concepts comprises to the Powerline Communication network attached to the acquired dataset we have gathered .

## 2.2 Project Specification

Activity of sensing home appliances has a various important applications which include home automation, smart-meters energy monitoring, entertainment and even in health-care devices. Many studies have been conducted to detect specific occupant activities in powerline, but a limited ones were conducted regarding powerline communication systems. Such activities to be detected require multiple types of sensors, extensive hardware installation or invasive vision systems. Disclosed, in this master thesis, is an approach that uses Powerline communication signals to detect the source of the electrical noise on the residual in-home powerline networks. These electrical noise signals are created by a certain appliance while in operation. As a result, the noise signal creates unwanted distributions in the powerline communication systems which fail to transmit data in an efficient way. Therefore, these noise events need to be classified using Machine Learning techniques, and then isolated while in communication. Machine Learning (ML), in this thesis, is concerned to take empirical appliance data (such as PLC modulation signals induced with appliances' High Frequency noise), and recognize the source of noise in the modulation signals. In other words, ML algorithms should be able to recognize the Signal-to-Noise ratio in powerline modems' modulations caused by the electric noisy events such as switching on/off a particular appliance in the house. Different Machine Learning Algorithms have been tested to evaluate the best system performance over time with different in-house appliances.There are several steps and algorithms which ML follows to reach the best pattern recognition performance accuracy

such as Linear or Non-Linear Regression, Classification, Clustering and Optimization. Therefore, in this master thesis, the findings are based on a specific algorithms that are known to be the best performing ML model for detecting discrete classification problems such as powerline modulation Signal-to-Noise Ratio frequency-series. These results can be used for PLC technology providers to harden their communication technology against various PLC disturbances of typical electric loads, or predict PLC network failures or likability to fail.

## 2.3 Organization of the Thesis

In this thesis, it is attempted to follow a feasible approach to meet the ends of this master thesis project. As it can be visualized, one end is gathering the required dataset, and the other end is training the dataset using machine learning. Therefore, the remainder of this master thesis is organized as follows: Chapter 3 highlights the overview of recent related works that are based this master thesis on. This chapter explains the existing Home network techniques and technologies that have been utilized in creating the dataset. It also explains appliances' High Frequency data acquisition, signatures, and classification models used in the previous researches. This will provide a basic background of this thesis path and the current scenario of powerline anomaly detection. Chapter 4 highlights the related notions to this master thesis. This chapter will give a brief description of the types of appliances' noise in the powerline, powerline modulations, and machine learning algorithms which will be used in this project. Chapter 5 will highlight the concepts for this project which include Data Acquisition, Signal to Noise ratio calculations, data processing, Data partitioning and Machine Learning classification models and methods used in this project. Chapter 6 will describe the implementation of the project along with the main parts of the *MATLAB* and *R* code documentations and explain the libraries and function used for achieving the results in the project. Chapter 7 will discuss the data processing results and eventually leading to the Machine Learning results. It also highlights the remarkable major and manor findings of each step. Furthermore, this chapter will draw a detailed comparison of the related work results and the findings. Furthermore, in this chapter, the constrains and challenges faced during each method will be briefed in each section. It is also important to highlight the unexpected and intriguing findings; in addition to the limitations witnessed during the implementation of each method. These results could be significant, and they can be helpful to discuss the potential impact associated with powerline communication in the future. Chapter 8 discusses the advantages and disadvantages of using the methods and compare them to the related recent works. It will also discuss the master thesis future directions as it should yield to new scientific results that could prompt further or new studies in Powerline Communication systems, sensors or Machine Learning.

# Chapter 3

# Related Work

## 3.1 Detecting actuation of electrical devices using electrical noise over a power line

This patent presented methods of detecting electrical appliances by using their electrical noise in the powerline. The researchers developed their solutions to be cost-effective without some withdraws such as having to install and sustain multiple sensors which require time and arduous effort to maintain across the entire powerline infrastructure [3, 4, 5]. Their hardware senses electrical noise from a single point, with less sensors deployed within a home. Some former other research projects approaches sensed powerline electrical power activities using the current flow or power consumption to gather appliances' noise high frequency signals [6]; however, due to changes in the current flow instrumented with each appliance, such approaches required multiple current sensors to be installed for each appliance or at least around its power plug.[3, 7, 8] This patent approach did no require much of hardware installation as the consumer had to simply attach their sensor in one of the home plugs, and sense appliances activities. This solution provided a low-cost and easy-to-install approach that was also capable to identify electrical events. Their approach also had many applicable implications which were used in home automation, health, and energy usage information. The authors disclosed a set of methods to identify electrical appliances actuation which were coupled in the powerline infrastructure. The system was embodied in a single computer which was coupled to a plug into the electrical powerline. The computer identified electrical noise signatures associated with each electrical appliance. Furthermore, it comprised software that records electrical noise signatures for each appliance that had been switched on/off and their related energy loads when in operation. The computer also then identified the toggled/switched appliances that had been actuated by comparing the electrical noise signatures transmitted over the powerline with the recorded electrical noise signatures. Their approach to detect appliances actuation was based on the wired 60 Hz electrical powerline infrastructures; however, it was

proved that it could be extended to 50 Hz electrical powerline infrastructures. Their approach relied on the abruptly switching (human-initiated or automatic) electrical loads which were produced in the broadband spectrum. The patent used two approaches to examine and classify the home wired electrical powerline infrastructure. The first approach was to classify new electrical noise high frequency signals in the home which exclude sensors' electrical noise in home. The second approach encompassed the existing appliances in the home infrastructure. They deployed their system in several homes in order to show the stability of their system, and the ability to detect and classify electrical events in different homes. [9]

### 3.1.1 Data collection

The patent approach used the existing powerline infrastructure that provides home appliances localization. It leveraged each home existing powerline infrastructure to support their activity to detect appliances. Comparing to the previous approaches to actively sense tagged objects, appliances in this patent were considered as an information source which they could observe different electrical events passively. The advantage of their approach is that it requires to install the powerline interface sensor only in a single point in the home which is also connected to the USB data acquisition Oscilloscope to calculate the Fast Fourier transform (FFT) of the noise and store it to the embedded personal computer. As the graph below shows the electrical device actuation used in their data collection and testing.

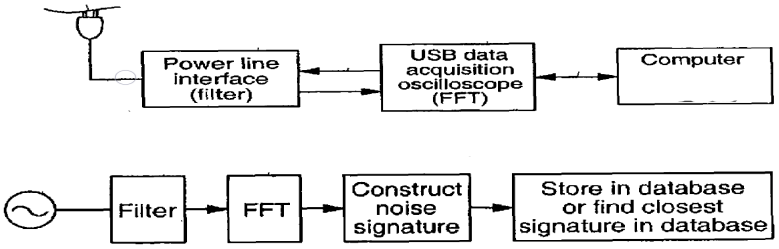



Figure 3.1: device actuation detection [9]

As the figure shows, the powerline interface (PLI) which filtered electrical output in the powerline. The filtered signals are the output of the powerline FFT that are initially voltage transients. In other words, they are frequency verses voltage versus time data which are all associated with appliances noise when they are switched on/off. The FFT was then processed in the embedded system where the noise signatures were constructed and associated to each appliance noisy events. As the inventors collected the voltage transient noise, they also processed and stored appliances energy load signatures of each appliance. Furthermore, the embedded system recordings were transferred to a central

database where it ran a software application which simultaneously collected and analyzed electrical noisy events when they were present in the powerline. It was also configured to further process streamed FFT data from the Oscilloscope and learn the appliances certain characteristics by switching them on/off. The embedded system then could classify and predict each appliance actuation noise signals based on their learned characteristics.

### 3.1.2 Hardware details

To better understand their methods of data collection and the concept of detecting the noise and ultimately learning various electrical noise events, the hardware prototypes were constructed in a way that comprises powerline interface modem. This PLI modem has three outputs as illustrated in the following figure.

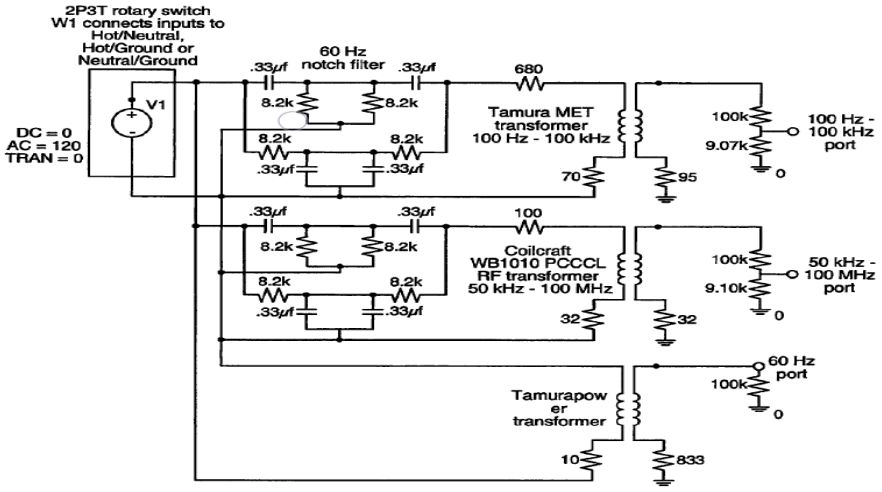



Figure 3.2: detailed schematic of an exemplary power line interface device; [9]

One output was designed for the standard AC 60 Hz powerline signal which was used in their exploratory phase [9]. The second output was the attenuated PL output which filtered the bandpass with the passband frequency signals ranging from 100 Hz to 100 Khz. The last output was identical to the filtered bandpass, but with a higher passband frequency ranging from 50 kHz – 100 MHz.[9]

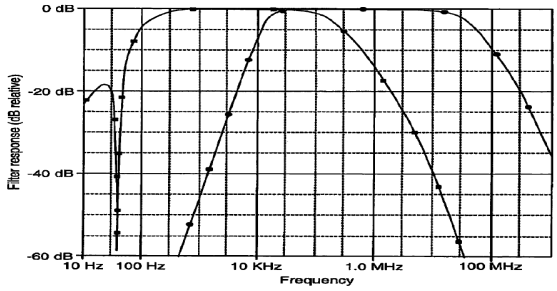



Figure 3.3: detailed schematic of an exemplary power line interface device; [9]

As the graph above shows the frequency response curves and covered frequency spectrum of the powerline data collection. The reason for having different filtered outputs was to have a broader and flexible range of frequency spectrum. All of the filtered outputs had a 60 Hz filter in the front of the bandpass filters in order to discard frequencies accompanied with the AC power; in addition to improve the sampled data dynamic ranges [9]. PLI hardware components were built to monitor powerline characteristics such as hot and ground, hot and neutral, neutral and ground. The normal mode, according to the patent, was the noise between hot and neutral due to the fact that several loads caused by small appliances. For example, lamps don't possess a ground connection. The authors connected the PLI module to a single point plug in the home with 120 V branch of the powerline electrical infrastructure. This is typically a standardized voltage level in USA which has a single-phase or a split single-phase powerline electrical infrastructure [7, 8]. The latter indicates that there are two different 120 volts electrical branches in the home electrical infrastructure which supplies 240 V electrical appliances [7, 8, 9]. As the authors indicated that both branches were in phase; so appliances generated noise connected to the other branch were also coupled to the electrical branch that PLI was connected to; therefore, the noise were all detectable by their system. The authors suggested installing a coupler into the 240 V or more outlet in order to assure a direct access to both branches. All the PLI outputs eventually were streamed to the oscilloscope through a USB connection. Each input had a full voltage of 1 Volt with a 10-bit resolution. Each bit represented a voltage level of 4 millivolts. The oscilloscope had a real-time sampling rate that reaches 100 million samples per second. [9]

### 3.1.3 Software details and Machine Learning Processing

The patent has constructed two different software applications. The first application was a C++ based codes that were constructed to sample Oscilloscope streamed data. The application also conducted Fast Fourier Transform (FFT) to each incoming signal which separated the frequency components for analysis. The software then constructed a frequency domain versus time domain waterfall plot that we also used in this thesis(see chapter 7 figure 7.1). Their

application could analyze data in real-time and record the data for post processing purposes. The second software application was a Java-based code that was constructed to perform ML algorithms and ran a UI for the users. The first FFT application was connected to ML application through a TCP connection where the later could read the streamed data coming from FFT application in real-time. The authors used *Weka* ML toolkit to classify and predict electrical events. The second software used ML techniques to identify and classify these patterns when each appliance electrical event started or ended. Their approach included the physical human-initiated events of turning on/off appliances; in addition to automatic events of appliances. For example, Fridge fan system which automatically turns on/off according to the temperature inside the system. Their system recognized electrical appliances which could be detectable in the home. The authors installed their system in a singled fixed location in the home through data acquisition process. The data was collected in both low and high frequencies that ranged between 100 Hz – 100 KHz and between 50 kHz – 100 MHz, respectively. Initially, as the authors indicated, none of the electrical appliances was on operation during the data collection phase; the reason for that was to measure the existing transient noise in the powerline infrastructure. Then by turning each electrical appliances, each noise signatures were observable. It was observed that electric loads that reached less than 0.25 amps were barely detectable relative to the protruding electrical transient noise or continuous noise (see chapter 4 section 4.2 for types of noise). The latter were caused by the dynamic range of the data collection Interface which was limited to 10 bits resolution. Furthermore, it was observed that a delay occurred when the appliances were switched on/off that reached 500 Milliseconds for the data collection noise impulse detection. The latter was also related to their data collection system sampling rate and processing latencies. [9]

**Detecting transient pulses**

This patent attempted to classify low frequency transient noise by filtering high frequency noise through their powerline interface. They also included the broadband noise at low amplitudes. Low frequencies ranging from 100 Hz to 100 KHz were designed for transient noise detection. For example, motor-based appliances, such as fans, produce transient noise pulses when they are switched On/Off **??**. However, the authors could not detect continuous noise as expected from the electromechanical motor brushes. The reason for that was related to the 60 Hz AC filter which initially blocked the 60 Hz frequencies.[9] The authors created a sliding window algorithm to detect transient pulses at both the switching on/off times. As the authors indicated that these transient noise signals only lasted abruptly for a few microseconds. The sliding window was made to coup up with the short periods of the transient noise signals. It acquired 1 microsecond sample that was averaged from the data after acquiring FFT of the streamed data. Each sample had frequency components and amplitudes in a vector form. As the authors indicated that each vector included amplitude values for frequencies covering low frequencies between 0 and 50 KHz. The

authors used Euclidean distance to calculate the vectors in their time sliding window. The Euclidean distance was set to a certain threshold value that could mark the transient noise when it was detected. The authors believed that it was better to make an adaptive threshold at each home in order to successfully identify transient noise signals.[9] After extracting the transient noise from the electrical powerline infrastructure, the authors defined N for the vectors of length L. N stood for the pulse width in 1 microsecond; while L was the frequency component number that extended to 2048-points vectors. New vectors were then added with length of L+1 and by averaging the corresponding N pulse width values of each frequency components. AS the authors indicated that the (L+1)th value was the N pulse width of the transient noise pulse. These values were considered to be the feature vectors for their particular transients.[9]

**Learning transient pulses**

The authors used *Weka* toolkit which included Support Vector Machine (SVM) Machine learning algorithm. SVM employs feature classification by constructing multi-dimensional hyperplane which separates data into multiple classes [65, 67, 69, 70] (See chapter 4 and chapter 5). SVM separates the chosen data with the largest distance to the hyperplane with the nearest negative and positive examples [10]. Therefore, as the author indicated that it was the most suitable classification to test the data that was almost similar to the training data. SVMs are known with their large space classification which can categorize multiple categories [65, 67, 69, 70]. As the authors explained that SVM prevented overfitting in their learning process; thus, they had the ability to handle larger amount of features. It was not clear which SVM model was used, but in this thesis implementation, we will show the results of both the Linear SVM and Radial SVM models (See chapter 6 and chapter 7).

**Testing and performance evaluation**

After training SVM algorithm on their given data, it was then tested in six different homes that had different powerline infrastructure. For example, the size, style and location of rooms were different. The authors tried to test the model ability to detect and predict transient noise in every home with a period ranging from one to six-weeks. The reason for testing for 6 weeks was to evaluate the classification accuracy for different types of electrical appliances, and how the system could be retrained overtime. The other reason to test their system in 6 different homes was to determine the transient noise signatures characteristics and how stable or similar they were in the other homes.[9]

**Transient noise evaluation**

The authors evaluated the feasibility of their transient detection approach by collecting the data from a single home for period of four hours. Their software isolated the transient noise signals of each appliance. During the data collection

period, they actuated many electrical devices and recorded their timestamps. Almost a 100 significant electrical events were gathered. [9]

**Transient events Classification**

The authors detected appliances electrical transient noise in the home either by human-initiated or automatic events. Events ,such as the lights inside refrigerators when they were turned on or microwave when it was turned on, were all detectable.The authors included the transient noise despite the unpredictability of these frequency noise signals comparing to continuous noise signals (check chapter 4 for more details). Thus, they only considered the duration of the transient noise and its frequency noise components. For instance, they only considered the individual on/off events which were enough to recognize the same pattern over time.[9]

For the testing period, their data collection PLI was configured in a single point electrical outlet for each home. As for home 1, they collected the data and labeled the electrical appliances for three times a week with a period of 6 weeks. For each toggled appliance, they labeled the toggled (On/Off) events manually. [9] Data was similarly collected in the other 5 homes, but with a shorter time period of one week. The authors trained the data in the beginning of the week, then they collected the test data in the end of the week. 2 to 5 event instances were collected for each appliance in the training dataset. The reason for that was to have equal distribution for each appliance and preventing bias towards other appliances. Moreover, in order to provide the best classification performance, the authors decided to set a minimum number of instances per event as the training process gets laborious with huge data. [9] The classification accuracies ranged around 85% for house 1 with a period of 6 weeks as the following table shows

| **Training set size/instances per event** | **Week 1 (%)** | **Week 2 (%)** | **Week 3 (%)** | **Week 4 (%)** | **Week 5 (%)** | **Week 6 (%)** |
|---|---|---|---|---|---|---|
| 164/2 | 83 | 82 | 81 | 79 | 80 | 79 |
| 246/3 | 86 | 84 | 83 | 84 | 82 | 83 |
| 328/4 | 88 | 91 | 87 | 85 | 86 | 86 |
| 410/5 | 90 | 92 | 91 | 87 | 86 | 87 |

As the results above illustrate that a small number of training dataset size resulted with a lower SVM classification accuracies. The results accuracies were initially based on the correctly identified toggled events in the test dataset. The authors noted that by increasing the number of the training instances, as the table above shows, it increased the accuracy of the SVM classification. The more training samples were provided, the higher the accuracy got on the test data. Other homes results were also given during their single week of deployment for both training and testing.

| Home | Distinct Events | Training set (events) | Test set (events) | Accuracy | Majority Classif. (%) |
|---|---|---|---|---|---|
| 2 | 32 | 328 | 100 | 87 | 4 |
| 3 | 48 | 129 | 96 | 88 | 6 |
| 4 | 76 | 304 | 103 | 92 | 3 |
| 5 | 64 | 256 | 94 | 84 | 3 |
| 4 | 38 | 152 | 80 | 90 | 8 |

As the results above show the accuracies of correctly identified appliances on the test dataset. It can be observed that only home 5 had the lowest classification accuracy, and that was due to the interfering existing powerline frequency noise with transient noise events.

## 3.2 ElectriSense: Single-Point Sensing Using EMI for Electrical Event Detection and Classification in the Home

ElectriSense used home's electric devices' Electromagnetic Interferences (EMI) signatures, which are also considered to be continuous noise, to classify and detect the electronic devices during operation in a home powerline wiring network. The research used similar methods of a single-point electric events detection used in the previous related work in section 3.1. The authors detected various class of devices that; specifically, use a Switched Mode Power Supplies (SMPS) which are commonly used in current consumer's homes. SMPS are also commonly used in modern consumers in Europe due to their efficiency and low costs.[11] . ElectriSense research was based on section 3.1 approaches which related some of their EMI frequency analysis to the voltage transients from activation to deactivation of the electrical loads' inductance and resistance. Unlike the previous research, ElectriSense research gathered the continuous EMI signals generated by home devices. It was found out that the continuous noise signals were repeatable frequency-domain signatures during devices' operation. As indicated, in current Powerline-Infrastructure events sensing researches, they involved multiple sensing techniques in home which sensed tagged objects, appliances in this patent were also considered as an information source which they could observe different electrical events passively. [6, 12, 13, 14]

ElectriSense data collection system consisted of a single powerline interface (PLI) module which functioned as a front-end powerline modem and its internal schematic was similar to figure 3.3. It could be plugged in any home electrical outlet. The PLI module filtered the AC line 60 Hz frequency so that the spectrum analyzer, or the analog-to-digital USRP device was not overloaded by the 60 Hz frequency components. The PLI was connected to a Universal Software Radio Peripheral (USRP) which was known for its high speed data acquisition that digitized powerline analog signals with 1 MHz sampling rate through the software. The USRP then streamed the data over an embedded USB connection to a PC.

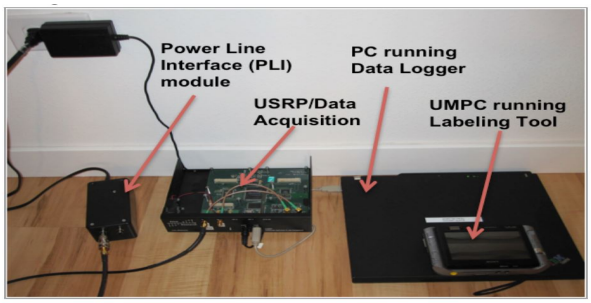



Figure 3.4: ElectiSense Data Acquisition Prototype System [15]

The incoming streamed data was in time-domain that was eventually fed into the PC in real-time. The streamed data was buffered with 2048-point vectors. The FFTs of these vectors were already computed to obtain the frequency-domain signals. These 2048-point vectors were spread over 500 KHz frequency spectrum which eventually yield a 244 Hz resolution per FFT bin. In other words, the FFT frequency vector was computed 244 times per second which was fed into ElectiSense's event detection and extraction.

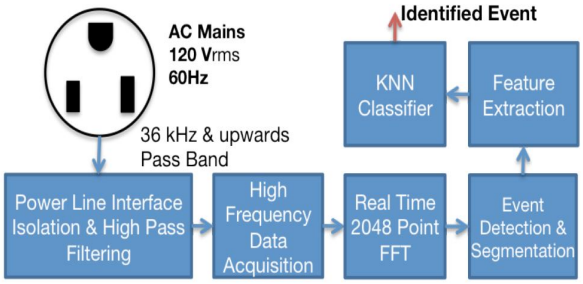



Figure 3.5: ElectiSense System component block diagram [15]

The incoming streamed data captured the continuous noise from -100 dBm to -10 dBm across 36 kHz – 500 kHz. The following figure shows the frequency spectrum across 36 Khz – 133 Khz for different appliances when they were turned on/off. [15]

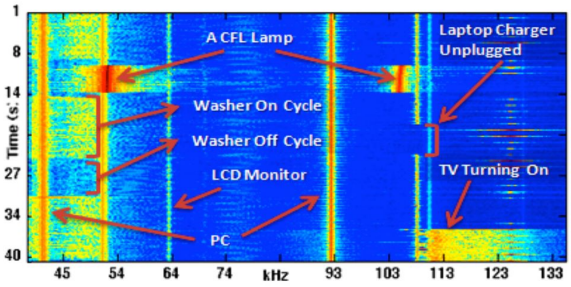



Figure 3.6: Frequency spectrum showing device actuation [15]

As the figure shows the appliances actuation when they were in operation. We can see the narrow-band continuous noise signature of each mentioned device that lasted during each appliance's operational period. It was noticeable that some of the devices noise center deviated in intensity and extended to either lower or higher frequencies. For instance, the CFL lap in the figure shows that its strongest intensities were on both 54 Khz and 105 Khz, and then it decayed around these frequency points. This behavior in some devices was due to the error tolerance of the switching circuit core components, in addition to the characteristics of the power supply's load. Some devices' switching core components, like the TV in the figure, we notice a single narrow-band signal peak at the switching frequency. Electisense averaged the incoming frequency vector over time to determine a steady baseline frequency since it was found that most appliances generated frequencies above the baseline. The baseline noise varied across the entire spectrum with a period of Hertz. As the following figure shows, ElectriSense used an average sliding window with a size of 25 frequency vectors.

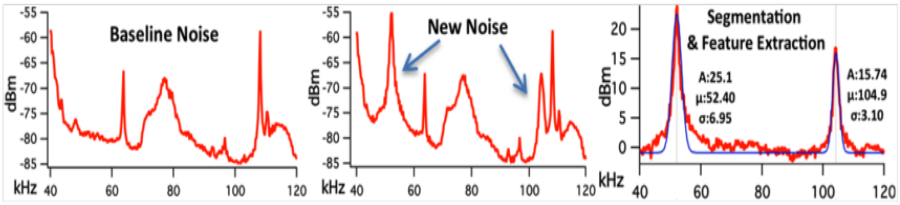



Figure 3.7: Frequency spectrum showing device actuation [15]

As the figure shows, on the left is the observable baseline Background noise in the powerline without any appliance was turned on. It was captured after 25 window size sliding window. The center graph shows when an appliance was actuated and introducing new continuous or transient noise in the frequency domain. The last graph on the right shows the difference vector by using the Gaussian fit which segmented the amplitude, mean and variance of the noise. With this approach, ElectriSense could define a global threshold that could work

in every home. The threshold was defined in around 8 dB above the baseline noise .[**?** ]. However, as the study showed, the baseline threshold could be drastically different from one home to another due to electrical infrastructure coupling. Therefore, ElecriSense required a pre-training process to learn and extract the exact parameters of the baseline in each home separately. The latter required the user to actuate each appliance at least once so the algorithm could learn the exact behavior of each appliance noise signal. Another reason ElectriSense required pre-training process was that most homes, whether big or small size homes, had different strong signal coupling and crosstalk across phases. Therefore, it was required to install more than a PLI sensor to detect the electrical events at each phase. However, multiple similar electrical signatures from the same appliance were captured in this process.

In powerline communication, accumulated noise of identical appliances always occur in a home .[12, 13, 16]. For example, having multiple CFL bulbs, TVs from the same or a different manufacturing brands, knowing that each manufactured appliance brand uses their own electric circuitry that suits the operation of the appliance; thus, generating different EMI signals. Therefore, as the authors indicated, multiple similar devices caused problems as they could not be classified in the same group. ElectriSense approached this problem in two different ways:

First, it was observed that the center of the noise frequency shifted due to the tolerance in the appliance's components switching circuitry, the appliances from the same brand introduced enough variability in switching frequency in a way that the mean of the mentioned Gaussian fit shifted in the frequency spectra[15]. A case in point, the CFL lamps shifted, as observed by the authors, as the following graph shows.

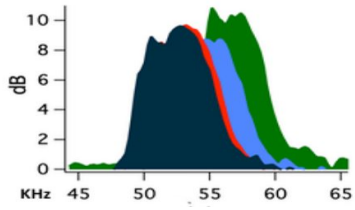



Figure 3.8: identical appliances noise [15]

To overcome this shortcoming, ElectriSense used a higher ADC resolution and a larger FFT bins to increase the frequency resolution. Eventually, such frequency behaviors could be detectable.[15]

Second, it was observed that EMI of multiple appliances of the identical brand was affected in many ways. To clarify, EMI signal was attenuated as a function of the line inductance between the sensing point and the noise source. Therefore, two or more identical appliances could always generate the identical EMI signal; however, they may look different depending on the sensing source distance from the noise across the powerline.

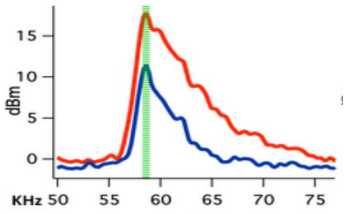



Figure 3.9: identical appliances noise from different locations [15]

The figure shows the EMI frequency spectrum of two identical appliances with relative to distance from the sensing source. These appliances were plugged in two different locations in home. As it was observed that there was a difference in amplitude in respect of the frequency. Therefore, with this observation, it could be a deterministic way to find the fixed number of existing identical appliances in the home. [15] In powerline, accumulated noise with the different appliances always occur simultaneously in a home [12, 13, 16]. For that, ElectriSense feature extraction method could find the peaks of other appliances using the difference vector which fit with a Gaussian function and then extracted the mean, variance, and amplitude parameters as in the figure above shows. Other peaks could also be present due to the transient noise and continuous noise harmonics. However, as the study used KNN algorithm, it could still detect and classify the simultaneous existing appliances' events. [15] ElectriSense used K-Nearest Neighbour (KNN) machine learning algorithm; namely using Euclidean distance which will also be used in this thesis (see chapter 5). The Euclidean distance has an inverse weighting that suits or fits the data dimensionality [28]. The authors used 10-fold cross validation for each house, and observed the average accuracy which reached 91.75%.

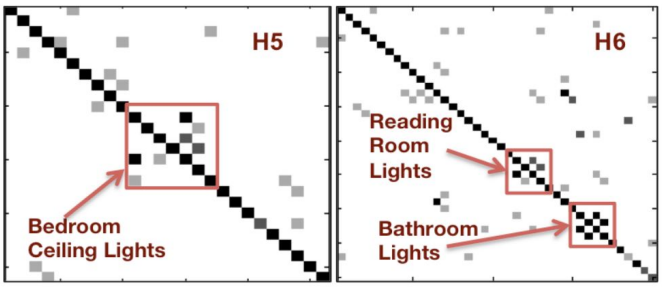



Figure 3.10: Confusion matrix misclassification [15]

Their approach could detect 2576 different electrical noise events. Their implementation detected simultaneous electrical noise as close as 102 milliseconds. This was dependent to the sampling frequency and the fixed averaging window size. However, the shortcoming of this approach was that if there were multiple events occurring within 102 ms, they were detected as a single event.

| House | Accuracy | Kappa |
|---|---|---|
| 1 | 93,4% | 96,2% |
| 2 | 83,7% | 89,6% |
| 3 | 93,5% | 96,4% |
| 4 | 93,8% | 97,2% |
| 5 | 90,5% | 85,3% |
| 6 | 84,6% | 84,4% |
| 7 | 85,1% | 92,8% |

Table 3.1: Comparison of different houses' results in KNN

ElectriSense classification approach could not separate these combined features to identify the exact appliances.

# Chapter 4

# Related Notions

This chapter focuses on the basic related notions to this thesis. It starts by explaining the background of powerline communication and its modulation schemes. It also explains the types of powerline residual noise signals and their sources which ultimatly affect powerline communication modulation schemes. This chapter explains briefly each machine learning models used in this thesis classification problem to classify powerline noise sources.

## 4.1 Powerline Communication (PLC)

Powerline networks are considered to be a difficult medium for transmission of information. Communication over the AC powerline is difficult due to the unpredictable noise signals caused from electrical sources which cause bit errors while communicating. Compared to Ethernet networks which have clean transmission and consistent characteristics,, powerline networks are not controlled over time. Powerline networks have multiple appliances and devices toggled in at any time. The constant toggling of various appliances causes the powerline characteristics to vary in transmission. All of the noise signals are originated from different appliances connected to the powerline network, and their characteristics strongly depend on each noise source appliance internal particular design. [1]

Powerline networks are considered to be harsh environment for communication. Powerline modems efficiency vary in time, location and load changes. Therefore, they require sophisticated modulation schemes. Conventional modulation schemes such as PSK, FSK are not used in powerline communication due to the hostile behavior of the powerline channel. These modems receive corrupted data due to powerline residual noise signals although using robust different modulation schemes for communication that could adapt to handle unknown attenuation and unknown phase shifts in the powerline [1]. To illustrate, these schemes performance are flexible according to the strength of the noise attenuation in the powerline infrastructure. [20, 21]

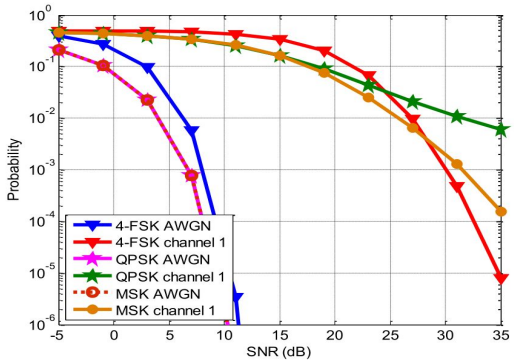



Figure 4.1: BER versus SNR curve for modulation schemes under powerline attenuation [20]

The figure above illustrates the bit error rate (BER) variation to the signal-to-noise (SNR) for different modulation schemes. The signal-to-noise ratio (SNR) is a key parameter to estimate a communications system performance.Signal-to-noise ratio (SNR) is a measure used in communication that compares the level of a modulating signal to the level of background noise. As it can be observed, the modulation performances depend on the noise attenuation in the powerline channel. The impact of these noise signals is impaired with the modulation performance. [20]

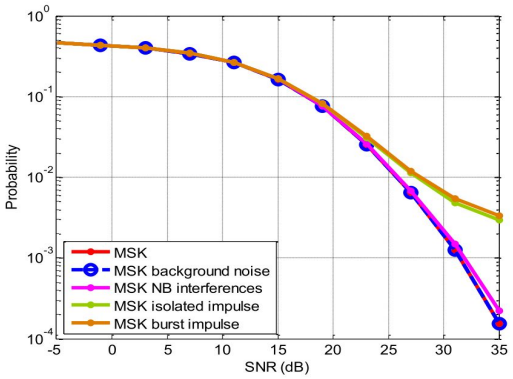



Figure 4.2: BER versus SNR curve for MSK under powerline attenuation and different noise classes [20]

As the figure shows the different noise signals effect on MSK modulation scheme. The same effect can occur to different modulation schemes within powerline modems which end up distorting bits reception. Each modulation scheme has a different BER vs SNR curves and their performance loss is dependent to the noise classes. [20]

In this thesis, we believe that by classifying the noise sources can improve the signal-to-noise ratio which also can improve powerline communication modems

to efficiently transfer data. The cost of implementing such models is very cheap as PLC modems manufacturers can install it as a software embedded to their hardware.

## 4.2 Powerline Noise Characteristics on the Residential Power Circuit

There have been many approaches which tried to detect electrical appliances activation from home outlets (without heavy in-line installation) [16]. They leveraged transients generated by mechanically switched incandescent, electric loads, heating to detect and classify electrical events. However, continuous and transient High Frequency noise signals were only shortly mentioned. Home electrical devices or appliances generate two types of noise or Electromagnetic Interferences - EMI: transient and continuous noise. It is presumed that the noise behavior of any appliance depends on the device embedded components and the interconnected powerline transmission line behavior. In any home electrical appliances, three general types of electrical embedded components are considered to be the noise source. Resistive loads source, inductive load source produced by motors where voltage noise is generated from the mechanical continuous switching, and inductive load caused by the internal oscillators of a solid state switches. [9, 15, 17]

* Resistive loads usually create low frequencies which are barely detectable while in operation. A case in point is the electrical stoves and lamps when they are switched on. They do not make a detectable amount of noise. In this example, the appliances create transient noise due to the arcing in their mechanical components. [9]

* There are appliances that modeled to produce both resistive and inductive loads simultaneously. For instance, the embedded motor, inside the fans or blenders, is considered to be both an inductive and resistive load source. The fact that these motors brushes create a voltage noise due to the continuous connecting and breaking in their components. This noise is synchronous to the AC power that ranges from 60-120 Hz.[9]

* Appliances with a solid state switching such as TRIACs in the light dimmers, and MOSFETs in the computer power supplies. These devices produce noise that is different due to its synchronous internal oscillators. Therefore, they are considered to be transient noise generated by their internal switching mechanisms. [9] The following table, which refers to the patent mentioned in the related work chapter 3.1, lists different devices with both transient and continuous noise signatures are detectable during their on/off or automatic initiated states. [9, 17]

| Device class/type | Devices observed | On to Off transition noise? | Off to On transition noise? | Continuously on noise? |
|---|---|---|---|---|
| Resistive | Incandescent lights via wall switch | Y | Y | N |
| | Microwave door light | Y | Y | N |
| | Oven light/door | Y | Y | N |
| | Electric stove | Y | Y | N |
| | Refrigerator door | Y | Y | N |
| | Electric oven | Y | Y | N |
| Inductive | Bathroom exhaust fan | Y | Y | N |
| (Mechanically | Ceiling fan | Y | Y | N |
| Switched) | Garage door opener | Y | Y | N |
| | Dryer | Y | Y | N |
| | Dishwasher | Y | Y | N |
| | Refrigerator compressor | Y | Y | N |
| | HVAC/Heat pump | Y | Y | N |
| | Garbage disposal | Y | Y | N |
| Inductive | Lights via a dimmer wall switch | Y | Y | Y |
| (Solid State | Fluorescent lights via wall switch | Y | Y | N |
| Switched) | Laptop power adapter | Y | N | N |
| | Microwave oven | Y | Y | Y |
| | Television (CRT, plasma, LCD) | Y | Y | N |

Transient noise is characterized with the short duration noise in which can be observed in nanoseconds to milliseconds. [18] It occupies both the narrowband and broadband frequency spectrum. Transient noise is usually generated when a device switch circuit is turned on/off due the tiny arcs generated in the switch. These arcs excite the step response of the powerline electrical network transfer function. Capturing such noise has been an expensive process that requires a continuous capture and analysis for every transient noise event in the powerline, knowing that there exists some transient noise in home even when there is not any device in operation[9, 17]. Transient events are difficult to comprehend as they randomly occur on unpredictable behavior [9]. Furthermore, they are also known with their broadband distribution on the High Frequencies; therefore, it is very challenging to capture such noise signals without proper equipment. [9, 15] Transient noise signals have extrinsic properties due to the mechanical switching loads, which make their behavior unpredictable;thus, the latter requires a sophisticated unsupervised training algorithm for each physical appliance. Therefore, High Frequency(HF) signatures gathered from one appliance can not be applied to another even if they have similar HF noise magnitudes. The latter possesses a challenge to gather appliances' transient noise. Some appliances transients may also change once it is moved from one room to another in the house [9, 12, 15]. Therefore, the dataset gathered in this thesis research was leveraged signals that manifested from a particular appliance circuit design and the components that the appliance uses. Transient noise signals can be generated from the abrupt switching on/off events. It is characterized to be rich in high amplitude within the lower frequency spectrum which ranges from 100 Hz to 5 KHz. It is also deducted that inductive loads with a solid state switches generally generate continuous noise in the higher frequency spectrum which ranges from 5 KHz to 1 MHz [9, 12, 15, 18] . Furthermore, inductive loads featuring mechanical switches produce noise close to 60 Hz, but in the

data provided in the implementation, the data was filtered by the PLI module as mentioned in both related works in chapter 3. Therefore, the collected data is divided into two parts. The higher frequencies spectrum ranging from 5 KHz to 1 MHz which is characterized with continues noise events which are produced by appliances TRIACs and switching power supplies. The lower frequencies spectrum ranging from 100 Hz to 5 KHz which is characterized with transient noise events which are mostly generated by wall switches. Transient noise impulses typically last for a few milliseconds; the latter include a low frequency spectrum components which ranges between 10 Hz to 100 kHz. As the example, the CFL lamps produce a short bursts when they first turned on. These bursts comes from circuitry ignition that a CFL lamps requires to warm up for operation.[15**?** ]

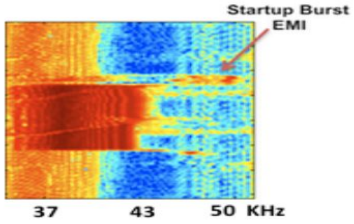



Figure 4.3: Startup transient noise burst signal produced by CFL lamps on ignition [15]

Transient noise frequency spectrum can also change slightly in an unpredictable way as the following graphs show two identical events of switching on/off a switch in two different times.

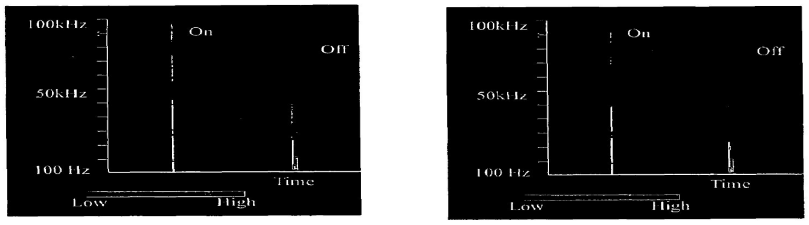



Figure 4.4: frequency domain graphs of a light switch that is toggled at different times [9]

As the two graphs indicate frequency amplitudes at two different frequency levels for the same switching device; however, the second switching event was taken 2 hours later. It is also proven that different light switches can produce different signatures due to their mechanical embedded components of each switching device.

Continuous noise has repeatable frequency-domain signatures during devices' operation as indicated in current Powerline-Infrastructure sensing researches which involved multiple sensing techniques in home [9, 12, 15]. A case in point, Fluorescent light bulbs generate continuous noise over the pow-

erline. Some home's appliances' noise signals distribution propagate widely on the powerline wiring since home's powerline wiring network is interconnected in parallel. [6, 12, 13, 14, 16] To properly capture continuous noise, electrical switches should be isolated from the powerline residual noise and tested. It was observed that continues noise can be detectable at the higher frequencies ranging from 50 KHz-100 MHz. Comparing to lower frequencies ranging from 100 Hz tp 5 KHz, continuous noise can be barely recognizable, but strong transient noise signals are detectable within that frequency range.[15] Continuous noise is always interconnected with the appliances' operation and their internal designed electronics. Appliances like hairdryers, grinders and fans, use the mechanical embedded motors that create voltage noise. These noise signals are synchronous to AC power frequency (50 Hz in Europe) and its harmonics (250 Hz, 350 Hz and so on) due to the continuous electrical contact by breaking and making by the motor bushes [19]. Other devices also create such continuous noise signals due to their internal oscillators. Dimmers, for instance, produce continuous noise due to their internal TRIAC switches. However, dimmers are characterized with their large frequency spectrum which spans hundreds of kHz in both the broadband and narrow-band noise. Furthermore, due to the various dim levels in the dimmers, it is hard to group them in one category device.[6, 14, 15, 16]

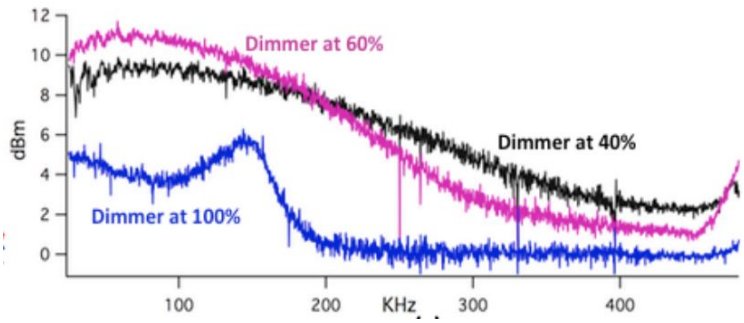



Figure 4.5: Band limited Noise generated by a dimmer shown at various dim levels [15]

There are appliances that neither emit continuous noise nor transient noise. For example, appliances such as electric stoves and dryers do not generate such noise due to their large resistive loads. Washing machines also generate very low frequency noise patterns which reach roughly to 0.1 Hz when they are in the wash cycle. [15]

Some appliances generate both transient and continuous noise when they are in operation. For instance, motor-type appliances like fans create transient noise pulses when they are off; however, when they are turned on, they also produce continuous noise signals while in operation. Mechanical switches also produce transient noise such as a light switch which produces transient electrical noise[9, 16]. Another example is dimmer switches generate rich mixed high frequency and low frequency noise when they are in operation. Moreover,

microwaves produce broadband continuous noise in the powerline while in operation. These devices incline to generate detectable continuous noise which reaches 1 MHz. [6, 14, 15, 16] It is also claimed that all devices and appliances generating continuous noise are always bounded by transient events in their components.[9, 15] Therefore, it is often extremely difficult to analyze, classify and predict transient noise because ordinary switches are not globally standardized to be well characterized during their connecting-and-breaking times.

## 4.3 Machine Learning

Machine learning (ML) is a category in computer science that allows software applications to learn about a specific problem and become more accurate in predicting the same problem with different inputs. The basic idea of ML is to utilize algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available. Machine Learning learns by analyzing the given data and learn from that data to make some prediction or classification depending on the structure of the data. [22]

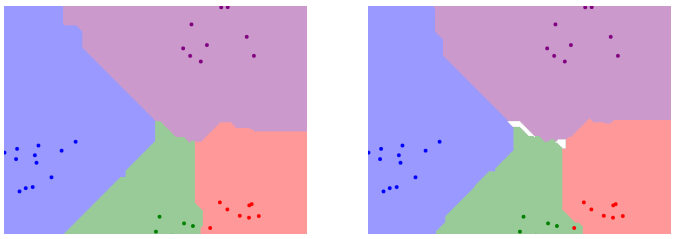
Machine learning algorithms are categorized as either supervised or unsupervised. Supervised algorithms, on one hand, require a researcher to provide both input and desired output, in addition to perfecting predictions accuracy during training. The goal is to map all inputs across dataset to the desired output and learn the general rule of this mapping. Researchers determine which variables, or features, the model should analyze and use to develop predictions. Once training is complete, the algorithm will apply what was learned to new data or testing data in the mentioned examples.[23] Unsupervised algorithms, on the other hand, do not need to be trained with desired outcome data. In other words, the algorithm does not have concrete features and the output for most of the problems is unknown. In such problem cases, ML algorithms are completely blind of how the output should be. The algorithm has to find identical patterns in the input data.[22, 23] Instead, researchers use an iterative approach called deep learning to review data and arrive at conclusions. Neural networks (NN), for instance, are considered to be both supervised and unsupervised machine learning model [24]. They are used mostly in unsupervised ML purposes as they are used for more complex processing tasks than supervised learning systems. In this master thesis, it is bounded, along with machine learning models, to be a supervised machine learning models. Such algorithms have only become feasible in the age of big data, as they require massive amounts of training data. There are limitless applications that ML has been used in. They range from the fairly simple tasks to the highly complex tasks. Here are a few of the supervised models used in the implementations:

### 4.3.1 k-nearest neighbors algorithm (KNN)

KNN algorithm is one of the simplest, lazy learning, supervised classification algorithm [26]. Though of its simplicity, it can give highly accurate results, as the previous related work had used it(see chapter 3 section 3.2). This model classifies a specified number of data points into hyperplane based on their similarities characteristics. KNN is considered to be a multi-label and non-parametric pattern recognition method which is used for classification problems[26, 27]. In this thesis and in the previous related work - ElectriSense 3.2, KNN has been used in appliances High frequency classification problem. KNN offers three important aspects:

1. Ease to interpret outputs
2. Cheap run-time complexity
3. Predictive Power

In KNN classification method, the output is a class to be predicted in the testing dataset. An input class is classified first by the neighbors votes in the training phase. k is an integer number of neighbors "voting" on the class. If k = 4, then the object class is assigned to the class of that single nearest neighbor. [27, 28, 29] To illustrate how K changes the boundaries in KNN. A simple example is when given 4 different classes, with a given K value. KNN makes boundaries of each class based on the default k vote, K = 1 by default. These boundaries will segregate appliance classes. The same way, with attempting to observe the effect of value "K" on the class boundaries. The following graphs show the different separating boundaries for the two classes with different values of K[30]



(a) K = 1 boundary (b) K = 3 boundary

(c) K = 5 boundary (d) K = 7 boundary [27]

As the graphs show,, there is a margin finding the best integer number for K as the training error rate increases. Therefore, the validation error rate will determine the best local minimum K-value with the minimal error rate, and this is integrated with *Caret* library in R as shown in the results chapter 7 section 7.2.[31]

### 4.3.2 Linear Discriminant Analysis (LDA)

Linear discriminant analysis (LDA) is a method used in statistics, pattern recognition which finds a linear combination of features that segregates multi-class of events [32]. It is known with its well-established ML classification method for predicting categories. LDA has several advantages comparing to other ML classification algorithms. LDA is a model that is interpretable and that predictions are simplistic. Its run-time performance is better than other ML algorithms such as Neural Networks or Support Vector Machine algorithms as shown in the results based on the classification problem in section 7.3. Usually, it is slightly as fast as KNN algorithm in terms of run-time complexity and prediction power. [32, 33] The goal of LDA is to detect a set of classes or objects and group them with the closest mean value. Every class has a mean vector evaluated from the characteristic of other objects' vectors which belong in that group. The closeness is determined by a distance metric. [31, 34] In this case since more than two classes are given, the analysis used in the derivation of LDA can be extended to find a subspace which contains all of the class variability and separate them with a linear line. Suppose that each class has a mean $\mu_i$ and the same covariance $\Sigma$ . So the segregation between these classes variability may be defined by the sample covariance of all classes Cs means

$$\Sigma_b = \frac{1}{C}\sum_{i=1}^{C}(\mu_i - \mu)(\mu_i - \mu)^T \tag{4.1}$$

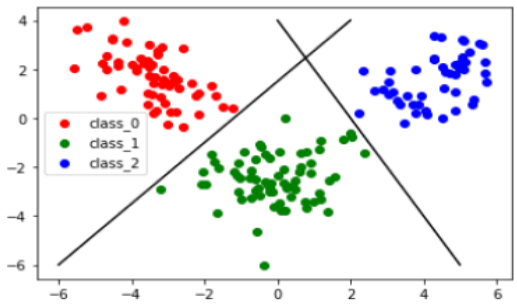



Figure 4.6: LDA linear separation [32]

### 4.3.3 Support Vector Machine (SVM)

Support vector machine (SVM) are a large margin classifier learning model with associated different learning algorithms that analyzes data used for classification and regression analysis [35]. It determines the optimal hyperplanes that maximize margin between classes [35, 36]. SVM is used for classification purposes given the dataset of training appliances Signal-to-Noise (SNR) Ratio, each marked as belonging to one or the other of two classes. An SVM training algorithm constructs a model that assigns new appliances to one class or the other, making it a non-probabilistic and statistical binary linear and/or non-linear classifier. SVMs always give global optimum, comparing to the previous mentioned ML models (it is impossible to obtain a local optimum because of how its math works). Therefore, they require higher computational and processing time. [35, 37, 38] An SVM models appliances SNR in the case as points in space, mapped so that the appliances of the separate classes are divided by a clear gap that is as wide as possible. Gap to class separation represents probability of sample being of that class. The higher the gap is, the higher the probability. New appliances are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.[35, 37, 38]

Not only SVM performs linear classification, but it can also perform a non-linear (Radial) classification efficiently using what is called the kernel trick. The latter implicitly maps the dataset inputs into high-dimensional feature spaces [39]. SVM kernel selection can be very challenging, and it is always dependent to the training dataset (see chapter 5 section 5.5.3 for SVM kernel tuning parameters used in the implementation)

#### Linear SVM

With the arbitrary appliances dataset, the best tuning parameters for the linear SVM kernel can't be determined easily. So, it is required to start with the simplest hypothesis space. For instance, given that the SVMs do not know much about the input data, then they work their way up towards the more complex hypothesis spaces. So, when the linear kernel works the best to the dataset it linearly separates classes. A linear kernel allows to use linear functions to separate classes, which are really impoverished (maximumly tuned) comparing to the Radial kernel. Once the order of the linear kernel is increased, the size of the linear function class increases [39, 40]. For an explanatory and visualization purposes of how Linear SVM works, let's assume the dataset contains of 2 dimensional classes only. Below is a graph plotted the decision regions of a radial SVM on 2 features space dataset:

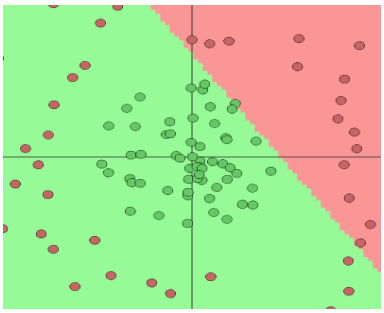


Figure 4.7: 2 dimensional Linear SVM decision regions [41]

**Radial SVM**

The appliances dataset can be also radially separable with more accuracy, a linear kernel isn't going to separate it precisely; therefore, Radial SVM is also used in this thesis study, for comparison purposes too. The Radial SVM kernel defines a function space that is a lot larger than that of the linear kernel. Therefore, it will require more processing time to separate classes. [39]

For an explanatory and visualization purposes of how Radial SVM works, similarly to the Linear SVM graph 4.7, it is assumed that the dataset contains of 2 dimensional classes only. Below is a graph plotted the decision regions of a radial SVM on 2 features space dataset:

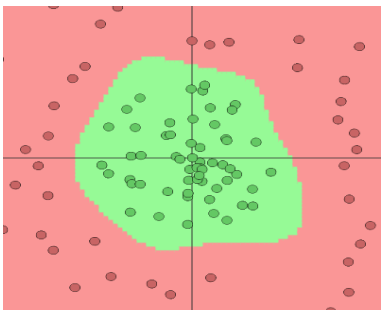


Figure 4.8: 2 dimensional Radial SVM decision regions [41]

In conclusion, the Radial SVM kernel is generally more robust than the Linear kernel, which can model a whole lot more classes with its function space, but it can be very expensive in run time complexity and results can not be guaranteed perfectly describe the classification problem.

### 4.3.4 Artificial Neural Network (ANN)

Artificial Neural Networks (ANNs) are biologically derived Machine Learning algorithms. They are designed to function similar to the human brain when processing information. ANNs collect their knowledge by finding the patterns

and relationships in the input data; then learn through experience. Usually, an ANN is formulated from tens to hundreds of single units which called artificial neurons. The neurons are connected with coefficients (weights) which construct the neural unit structure and simultaneously are ordered in layers [24]. The computational power for ANN depends on the connecting neurons within the network. Each single unit has a weighted inputs, transfer function and ending with its outputs. The results of an ANN is determined by its neurons transfer function when learning. ANN weights are parameters that can be adjusted in the neural network system which later can adapt to predict future datasets [24, 42]. The weighted summation of the inputs triggers the neuron activation; then the activation signal passes through the transfer function which eventually produce a single neuron output. Transfer function creates a non-linear boundaries to the network when training. The hidden layers unit connections are optimized till the predictions errors are at their minimum and the network reaches its global maximum accuracy. [24, 42, 43]

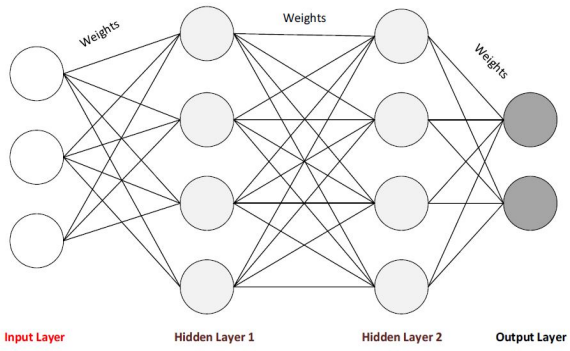



Figure 4.9: Artificial Neural Network [43]

Once ANN is trained, it can be tested by feeding it with a new input in order to predict the output. There are different types of ANNs and still new ANNs are emerging recently, so each ANN has its specifications and different applications that they are used with. Every ANN is described by their neurons' transfer functions which are considered the learning connection formulas for ANNs. ANN has been the best modeling algorithm these days; especially for non-linear relationships within the datasets. Thus, this paper will solely explain the ANN that has been used during the implementation and testing. So with their various applications, ANNs are used mostly in classification and pattern recognition problems Their prediction capabilities are very powerful when they are tested. Therefore, its potential prediction is considered in this research. [24, 42, 43]

ANN neuron is the main building element in the ANN. It is designed to imitate the functionality of the biological brain neuron. The incoming signals are inputs which are multiplied by their weights. These weights are first summed

and then passed in the transfer function which eventually produces the neuron's output. The activation function is the neuron's input weighted sum. The well-known used transfer function is the sigmoid function.[24, 42, 43]

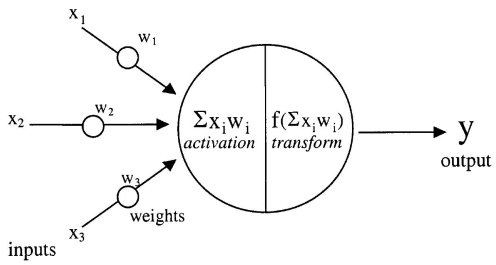



Figure 4.10: A simple Artificial Neuron [44]

The neurons' connection to each other is very significant in ANN. Just like human neurons, the neurons are either excited or inhibited by the given inputs. If the neuron is excited by the input, the neuron will trigger the summing mechanism of the following neuron to add; however, if the neuron is inhibited, it will trigger subtraction. All neurons can excite or inhibit other neurons within the same layer. The network will choose eventually the highest excited probability and inhibit all other probabilities [45]. There are two types of connections where the output of one layer routes in the network other layers:

1. *Feed Forward Neural Networks (FFNN)*:

   FFNNs are the default neural network connection route. The connection binds with multiple hidden layers which allow the signals to travel in one direction starting from input and ending to the output. In other words, FFNNs architecture does not have a reverse connection from the output to the input neurons; hence, it does not retain information of the previous output values. [43, 44, 46]

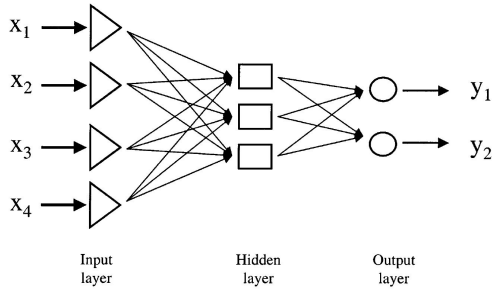



Figure 4.11: Feed Forward Neural Network [44]

Initially when training, the network consists of mapped pairs of input and

output neurons. When the network is trained, and the weights are determined for the connection between the neurons, it classifies a new dataset (test set) based on the previous learning [46]. When the new dataset is being classified, the matching inputs go through the network to determine the activation function for all the outputs values. Each input neuron has its activation function value which is represented with a few features. Each input neuron sends its activation value to its connected hidden layer. These hidden neurons determine their own activation value and then pass their activation value to the output layer neurons. The sum function in the neurons sums all the weights of the connections including their activation values. The sum is then adjusted by setting the value between 0 - 1 for the activation sum, or unless the parameter is manually set to a certain threshold which should be met when the activation value is set to 0.[43, 44] Each input layer neuron is associated to all of the neurons in the hidden layer(s) which are also connected to the output layer neurons. FFNN has a drawback when estimating the size of the hidden layers because it can result with overfitting behavior due to an overestimation of neuron numbers; therefore, the prediction accuracy will decrease dramatically. To overcome this problem FFNNs are better trained with back propagation technique.[43, 44]

2. *Back propagation Neural Networks (BPNN)*:

   BPNN architecture has the reverse connections comparing to FFNN. Backpropagation is also called "the backward propagation of errors," since an error is calculated at the output and distributed backwards throughout the network's layers. Each neuron has an additional weight since the input allows the additional freedom degree when attempting to minimize the error of the training.[43, 44, 47]

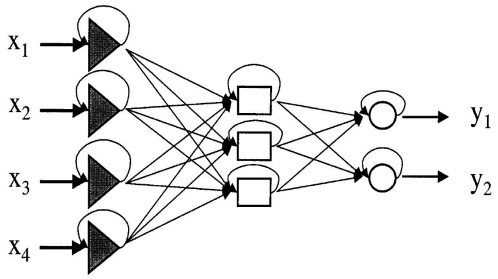



Figure 4.12: Back Propagation in a Neural Network [44]

BPNN is considered a method used to train an ANN to calculate a gradient that is needed in the calculation of the neurons' weights to be used in the network. Based on the error signal, the weights are calculated and updated iteratively [47]. Initially, the network weights are assigned randomly to

small values. The values are updated when the NN process every number in the training set. When training ANNs, it is common to use "weight decay," after each update. Weight decay is a term used in the weight update rule that causes the weights to exponentially decay to zero, if no other update in the network is scheduled. To Illustrate in details, the algorithm compares the output result with the desired output, then it computes the error value which will be updated. ANN keeps updating the network weights to decrease the error. The error, after each update, is backpropagated through the layers. At each iteration with each layer, back propagation minimizes the error signal. To decrease the error of the network the weights, the connection are updated until its global minimum value is reached. [43, 44, 47]

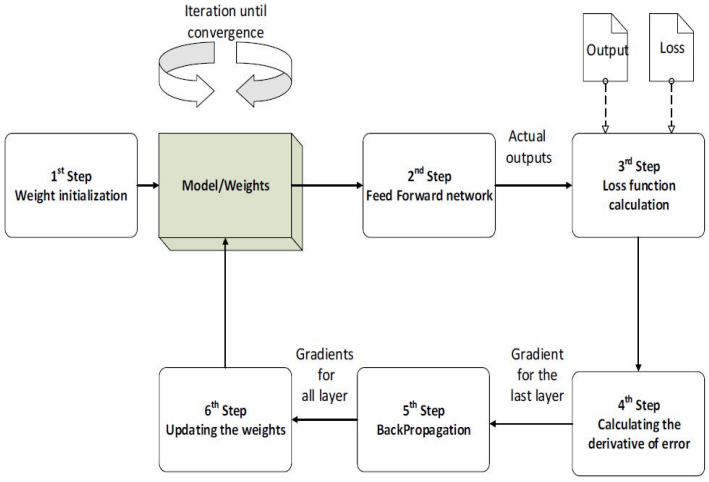



Figure 4.13: Steps of NN with FFNN and BPNN [48]

BPNN has drawbacks also which result in over-fitting. This case can happen when the learning is conducted with an insufficient training dataset, then the network may just set random features which have no association to the output data. Over-fitting can also occur when NN creates many hidden layers. It is also difficult to tell that the network has found the best accurate classifying results [43, 44]. This problem can be approached by setting up the parameters for the resampling methods such as Repeated Cross-Validation method which will be discussed later.

# Chapter 5

# Concept

This chapter explores the approaches that this master thesis is based on starting from data acquisition and ending up with Machine Learning classification. The predictions are based on several Machine Learning supervised classification algorithms on appliances' Signal to Noise Ratio (SNR) frequencies induced in powerline communication. This section is split into several parts, each part explores the phases that this project has been based on that eventually led to appliances' SNR classification as it will be shown in the results chapter 7. In this chapter first part, it will explain the attained dataset structure and how it is acquired based on the previous related works 3. It will also explain the challenges faced during data acquisition and how they are solved solely. In the second part of this chapter, the dataset will be visualized and the steps to process and manipulate the data to the form as appliances' SNR are illustrated. Lastly in the last part of this chapter, it will discuss the used Machine Learning concepts on how the data should be trained and modeled. This part will brief how the models are fed with the processed data, or how further the data could be preprocessed.

## 5.1 Data Acquisition

Data acquisition is a key phase in any Machine Learning project. The data acquisition phase is based on the existing powerline infrastructure that Powerline modems use as a medium to transfer data. The data should be calculated in terms of powerline modems modulators' Signal frequencies. While each appliance noise High frequencies are presented as the noise in SNR. The later should be trained and classified in Machine Learning. The initial dataset is acquired based on the related works that was mentioned in both related works 3 which are recorded using the same hardware, but with different software. The data took both researches almost 6 months to be acquired **??**. The dataset is large enough to train and then test with Machine Learning Algorithms. It was provided by the two researches which are available publicly; so this the-

sis implementation based the data collection to their hardware and software to acquire the proper data .[9, 15]. However, comparing to their data structure, the collected data in this thesis is slightly different in terms of broadband and narrow-band frequencies coverage. Furthermore, comparing to the previous related works' approaches, this thesis data has been captured in a more robust way, which it will be further discussed in the following sections. For instance, in the previous data acquisitions, researchers used the simplistic hardware and methods to capture powerline noise and either trained transient noise or continuous noise; but in this thesis provided data contains both of the transient noise and continuous noise with a higher frequency spectrum. The higher frequencies spectrum ranging from 5 KHz to 1 MHz, which is characterized with continues noise events which are produced by appliances TRIACs and switching power supplies. The lower frequencies spectrum ranging from 100 Hz to 5 KHz which is characterized with transient noise events which are mostly generated by wall switches.

.

### 5.1.1 Hardware

The hardware is not much different from ElectriSense research and its previous study that is based on [9, 15]. The Powerline Interface (PLI) is plugged in a single point in the home to acquire appliances' various frequencies. The PLI plug-in filters the AC 60 Hz power frequency and digitizes the analog Powerline signals to the USRP device, see figure 3.4 on page 14. The USRP, in this thesis case, demodulated the signals from analog to digital with a sampling rate or sampling frequency of 2 MHz, then it streams the collected data over a USB connection to a PC where data is collected and each appliance is labeled with a UNIX timestamp by the UMPC tool.[9, 15]

### 5.1.2 Software

Comparing to the previous related works 3 as their frequency ranges maximum to 2048-point vectors; in this thesis dataset, the incoming stream from the USRP is buffered in time domain signal with over 4096-point vectors. The Fast Fourier Transform, FFTs of these vectors are already computed to obtain the frequency-domain signals of the appliances noise. These 4096-point vectors cover both the narrowband and broadband frequency spectrum. The 4096 points are spread equally over 1 MHz spectral width which yields with a resolution of 244 Hz per FFT bin. In other words, the FFT frequency vector is computed 244 times every 1.06 seconds which is fed. [9, 15].

### 5.1.3 Data Structure

The dataset is provided from four different homes which consists of both training and testing datasets. The goal of this thesis is to utilize the training datasets to learn how each appliance SNR in each home behaves from a machine-learning

perspective and build a model(s) which can be applied to the test datasets to make the future predictions. The data has been labeled with different appliances which are all recorded in four different homes for a period of 6 months. For each home, the various recorded appliances' names and details of the households are shown in table 5.1

The collected dataset for this project is provided as MATLAB file extensions. This is the language and storage structure used by the prototype and its researchers in both ElectriSense in chapter 3.2 and its previous based research in chapter 3.1.

1. *AllTaggingInfo.mat*: This file consists of 4xN columns; where N is the number of the labeled appliances in a particular home. The 4 columns respectively include the *appliance ID, the Appliance Name, the UNIX time* the appliance is turned on, and *the UNIX time* the appliance is turned off.

2. *Training.mat*: These files are the actual dataset needed for this project. Initially, the data is required to be further pre-processed before feeding it to ML training and prediction. The embedded given information when the appliance is turned On/Off eased the data preprocessing work.

3. *The training.mat*: These files contain different elements, but the UNIX timestamps and High Frequency spectrogram are required to be extracted for each appliance. The High Frequency spectrogram contains a 4096xN captured noise. N here stands for the number of FFT vectors with each being computed at roughly 1.07 seconds. [9, 15]

## 5.2 Signal to Noise Ratio (SNR) calculation

The references to the Signal vary in electrical signals, but in this thesis implementation, it is referred to the powerline communication modulation signals(see chapter 4 for PLC modulations behavior). When a series of bits are digitized on a modulation carrier through the powerline infrastructure, the number of bits used to represent the measurement determine the maximum possible SNR. The noise level is non-linear and signal-dependent;therefore, different appliances cause different noise behavior in the powerline infrastructure. The theoretical maximum SNR assumes a perfect modulation signal that receives the same bit series sent[49, 50, 51]. Powerline channels always introduce attenuation and noise to any modulation carriers. The attenuation affects the received bits in the same way as the transmitted modulated signal. It scales the modulation signal by the same amount. Therefore, appliances' noise should be filtered from the received signal before demodulation [52, 53, 54]. Hence, calculating SNR for each appliance is processed to enhance the performance of modulation systems. The modulated signals are always exposed to the channel effects caused by appliances' noise. The desired modulation signal is defined as a constant number. However, for future further implementation, it is desirable to use modulation schemes' modulation signals.

Table 5.1: Appliances used in each of the 4 houses of the acquired dataset [15].

| House 1: | House 2: | House 3: | House 4: |
|---|---|---|---|
| Outside Over Garage Lights | Kitchen Lights | Back Porch Lights | Stove |
| Outside Front Door Lights | Kitchen Table Lights | Bedroom 1 LCD TV | Microwave |
| Downstairs Hallway Lights | Stairway Lights | Bedroom 1 & 2 Lights | Toaster Oven |
| Stairway Lights | Hallway Lights | Bonus Room Blueray/DVD | Bread Maker |
| Downstairs Bathroom Lights | Front Room Lights | Bonus Room LED TV | Kettle |
| Downstairs Bathroom Fan | Laundry Room Lights | Bonus Room Lights | Dishwasher |
| Downstairs Bathroom Fan Lights | Living Room Lights | Bonus Room Wii | Mixer |
| BR1 &2 Lights | Office Lights | Computer | Sandwich Maker |
| BR1 &2 Closet Light | Bedroom 1 & 2 Lights | Dining Room Lights | Kitchen Lights with Dimmer |
| Backyard Light | Master Bedroom Lights | Foyer Lights | Kitchen Counter Lights |
| GR Lights | Master Bath Lights | Garage Door Opener | Oven |
| Dining Room Lights | Master Closet Lights | Garage Lights | Vaccum |
| Kitchen Lights | Tea Kettle | Garbage Disposal | Hairdryer |
| Kitchen Under Cabinet Lights | Toaster | Guest Bath Fan | Bedroom Lamp 1 & 2 |
| Over Sink Light | Washer | Guest Bath Lights | PS3 |
| Powder Rm Lights | Dryer | Hair Dryer | Electric Furnace |
| Upstairs Hallway Lights | Outdoor Lights | Kitchen Lights | Livingroom Lamp 1 & 2 |
| Master BR Entry Lights | Front Hall Lights | Laundry Room Lights | Bose iPhone Dock |
| Master BR Lights | Dishwasher | Living Room Audio-DVR-TV | Forced-air Heater |
| Master BR Walk-in Closet Lights | Master Bath Fan | Living Room Lights | PC |
| Master BA Lights | Hair Dryer | Front Porch Lights | Computer Desk Lamp |
| Master BA Heat Lamp | Straightening Iron | Master Bath Fan | Den Baseboard Heater |
| Balcony Lights | Garbage Disposal | Master Bath Lights | Den Overhead Light |
| Dishwasher | Microwave | Master Blueray/DVD | LivingRoom overhead halogen |
| BR1 DVD | Refrigerator | Master Closet Lights | Entry light |
| GR LCD TV | Phone Charger | Master LCD TV/DVR | Hallway overhead Light |
| GR PS4 | Vacuum | Master Lights | Deck Light |
| Trash Compactor | Dining Room Lights | Microwave | Toilet Halogen |
| Coffee Maker | Laptop Charger | Office Lights | Toilet Exhaust |
| Garage Lights | Crockpot | Oven | Closet Light |
| Garage Door Opene | Coffee Maker | Powder Room Lights | Apple Macbook Pro 13 |
| Washer | TV/DVR | Toaster | HP Elitebook Laptop |
| Dryer | Computer | Upstairs Hallway Lights | N/A |
| Toaster | Printer | Dishwasher | N/A |
| Portable Vacuum | N/A | Dryer | N/A |
| Central Vacuum | N/A | Washer | N/A |

| Appliance ID | Appliance Name | Appliance UNIX On time | Appliance UNIX Off time |
|---|---|---|---|
| 32 | Stove | 1343331240 | 1343331270 |
| 25 | Microwave | 1343331360 | 1343331420 |
| 33 | Toaster Oven | 1343331540 | 1343331600 |
| 6 | Bread Maker | 1343331720 | 1343331840 |
| 19 | Kettle | 1343331960 | 1343332200 |
| 26 | Mixer | 1343332380 | 1343332380 |
| 31 | Sandwich Maker | 1343332560 | 1343332620 |
| 20 | Kitchen Counter Lights | 1343332830 | 1343332890 |

Table 5.2: Example of the data format of the AllTaggingInfo.mat file.

## 5.3 Data Processing and Visualization

Data processing is a data science technique that transforms raw data into an understandable format that can be fed into Machine Learning algorithms[55]. Note that the concepts *"Data Processing"* and *"Data pre-processing"* are used interchangeably in this thesis, which both indicate the transformation and preparation of the acquired raw data ; so it can eventually feed it to the machine learning models. The sequence of the used *Data processing* methods are arbitrary due the fact that two different programming languages are used, which both provide some different tools that were used in each step.

The following certain steps in data processing are followed:

1. *Data selection*: The exact data must be defined and visualized in order to proceed with Machine Learning.

2. *Data cleaning*: Extrapolation/interpolation are used to the missing values in the dataset. This process also involves removing unwanted outliers and resolving the dataset's inconsistencies.

3. *Data transformation*: The Signal to Noise Ratio is calculated for each appliance high frequency noise.

4. *Data Validation and Optimization*: correlated features have to be also removed, as this may worsen the model prediction accuracies.

5. *Data integration*: The whole mentioned calculations' results are integrated into one file that can be either used directly in Machine learning, or further pre-processed the data.

## 5.4 Data Partitioning

Once the data is processed, the next phase is to partition the data before feeding into Machine Learning models. The purpose of supervised ML, in particular classification, is to define a model that accurately assigns data to separate predefined classes. In order to test the generality of a learned model, the model is typically applied to an independent test data, and the accuracy of the prediction will inform us about the quality of the classifier ML model. [56] Different parameters are set, which validate the training accuracy before testing. These parameters have to be carefully chosen because too simple parameters lead to under-fitting, or over-fitting in the opposite case. For instance, the model will not able to predict the complexity of the data. However, too complex parameters at the same time lead to over-fitting; for example, the model is too expensive and complex.[57, 58] To test all of these assumptions and simultaneously optimize the bias-variance problem [59] ,in machine learning, it is a quite common practice to divide the dataset into three parts:

1. A training set

2. A validation set used for optimizing the parameters of the chosen ML model. [60]

3. A separated test set to validate the generalization performance of the final ML model classifier.

### 5.4.1 Cross Validation (CV)

Cross-validation (CV) is a way used in Machine Learning that makes near-optimal use of the available data by repeatedly training and testing classifiers on different subsets of data [60]. It typically uses a large training dataset segment for training and uses a small segment validation set for testing in each iteration [60, 61]. In K-fold CV, where k = n (the number of observations) and the k-fold CV is exactly the leave-one-out cross-validation. For instance, in 10-fold CV, 80% of the data are used for training, while 20% for the model validation, and in the next iteration another 20% of data are chosen as a test set, etc. This process is repeated ten times until all data have served as validation data once. CV is repeated with different parameters, and once the best parameters are found, the model is then trained with the chosen parameters on all data that have previously been used for CV and applied to the separate test set. [60, 61]


Figure 5.1: "Cross-validation and testing" approach [62]

## 5.5 Machine Learning Classification Models

Classification of appliances, which is based on their SNR, is achieved by using one of the multiple classification algorithms used in tgis thesis research. All of the selected Machine Learning Models are used to solve multiple classification problems, but it is required to ensure the best performing model to the dataset. The training is going to be supervised due to the fact that the desired outputs from the given inputs are already defined and the data has been already labeled. The dataset is an SNR frequency-series data as it is recorded with respect to appliances' noise time-frequency domain. After preprocessing and organizing the data, it is feed into the classification models. The models use SNR features as inputs in a multi-dimensional space to classify each input separately.

### 5.5.1 k-nearest neighbors algorithm (KNN)

KNN classification is used because KNN is non-parametric and cheap in run time. It makes no assumption about the data distribution. When it computes the distance between SNR points, it usually uses Euclidean distance by default or any other distance space generalizations, depending on the dataset structures [28]. For instance, the used Euclidean distance in this thesis, which *Caret* in R language uses, is measured in an n-dimensional space and discrete classification problems, the distance (d) from p to q, or vice versa is given by

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}. \quad (5.1)$$

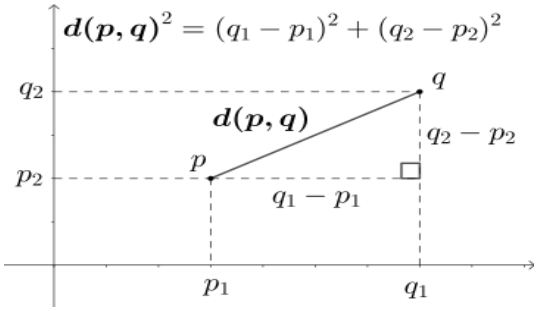



Figure 5.2: Euclidean distance in two dimensions [63]

### 5.5.2 Linear Discriminant Analysis (LDA)

LDA can be used either for dimensionality reduction or classification[32]. However, as in this research, classification is required. Therefore, there are a number of alternative techniques that have been used in this domain. For example, when different classes are partitioned in the dataset, LDA classifies each partition. With each class, LDA uses "one class against the rest" technique where the vectors from one class are put in one group, and everything else in another group. This will result in a single non-dimensional hyperplane, whose results are combined [64]. LDA2, in this thesis, uses the same technique, but with classes are classified in a multi-dimensional hyperplane.[31] The more dimensions LDA2 creates, the better accuracy results it produces. Both of LDA and LDA2 are integrated with *Caret* library in R.[31] Both LDA and LDA2 algorithm use the data to divide the predictor space variables into areas. The areas are labeled by classes and have linear boundaries[31]. The model predicts the classes of a new testing case according to which area it lies in. The model predicts that with all cases within an area that it belongs to the same class. These linear boundaries are the result of assuming that the predictor variables for each class have the similar multivariate Gaussian distribution [64]. Mathematically, LDA and LDA2 use the input data to derive the scoring function coefficients for each class. Every function takes as arguments the numeric predictor variables for each

case. Afterwards, it scales each variable according to its category-specific coefficients and outputs a score. The LDA/LDA2 model looks at the score from each function and uses the highest score to allocate a case to a class prediction.[34]

### 5.5.3 Support Vector Machine (SVM)

SVM models are parameterizable to best adapt a given problem. By selecting a ML algorithm for a classification problem, one cannot know which model's parameters will be best describing a problem [65]. Therefore, the best practice to do is to investigate empirically with controlled experiments.

*caret* package in R was designed to find the optimal parameters for high complex ML models such as Artificial Neural Networks and SVM [31]. It provides data scientists with different optimization methods. For instance, a grid search method for searching parameters, combined with different methods for determining the performance of the SVM model. In this thesis implementation with SVM, two Grid Search parametrization cases are used to find the optimal parameters for the problem using the *caret* R package. [31, 66]

#### Automatic Grid Search

The default way to tune any algorithm in R is by allowing the system to conduct it automatically. This can be done by defining the tune Length value to indicate the number of different values to try for SVM algorithm parameter[65, 67]. This tune length can be only for categorical and integer algorithm parameters; however, the model will make a crude guess as to what values to initialize, but it can get running very quickly.[68]

#### Manual Grid Search

The other way to tune any algorithm is to specify a tune grid manually. In the grid, SVM algorithm parameters can be defined as a vector of possible values. These vectors define almost all the possible combinations to tune the algorithm. There are two parameters for the SVM kernels namely by adjusting the C parameters and sigma parameter which both have an impact on the decision boundary for all appliances SNR dataset [69, 70]. The parameters are defined as follows:

1. Linear SVM : Linear SVM requires to adjust solely the C parameters since it is a linear classification method [71]. The C parameter directs the SVM optimization to avoid misclassifying each appliance SNR. These parameters have to be chosen carefully. For instance, if C parameter is large, the model will choose a smaller-margin hyperplane if that hyperplane does a good job in classifying all the appliances' points correctly. However, if C parameter is very small, the model will cause the optimizer to look for a larger-margin separating hyperplane. Even if the hyperplane misclassifies more points in the space dimension. [65, 67, 69, 70]

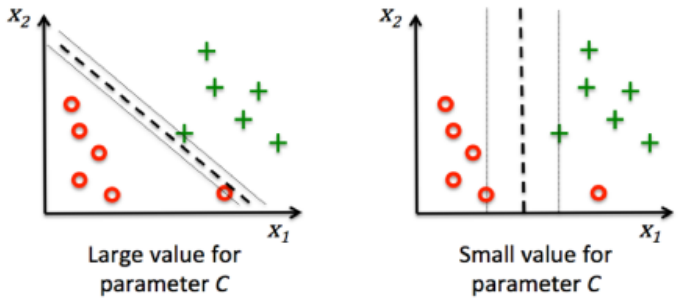
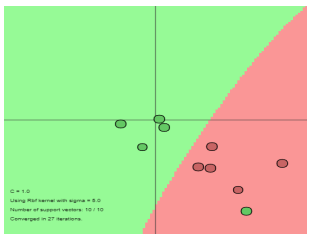
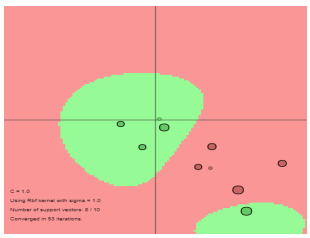



Figure 5.3: C Parameter classification influence [72]

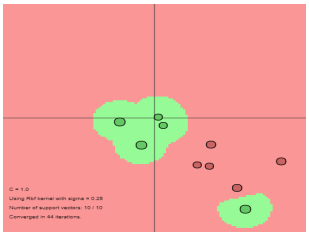
2. Radial SVM : It requires to adjust both the C and sigma parameters since it is a Radial classification method. Sigma is considered, as in a Gaussian Distribution, standard deviation. It defines the width of the Gaussian distribution around classes.[69, 70]



(a) sigma=5 Radial boundary



(b) sigma=1 Radial boundary



(c) sigma=0.25 [73]

The figures above illustrate the Radial SVM decision boundary defined between 0.25 and 5. It is observable that for a larger sigma, the decision tends to be linear; thus making misclassification while predicting, but it also avoids over fitting. However, for a smaller sigma, the Radial decision boundary tends to be sharp, but it tends to over-fit. [65, 67, 69, 70]

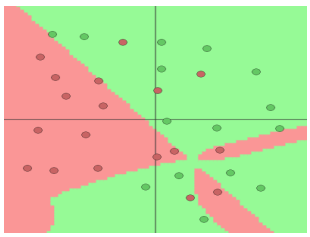
### 5.5.4 Artificial Neural Network (ANN)

Similarly to SVM models, ANNs are also parameterizable to best adapt this thesis problem. It is desired to investigate empirically the usable parameters with the controlled experiment. A multi-layer ANNS is used to compare the

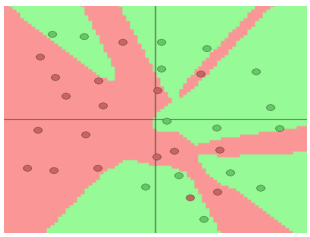
results of a range of hidden layers to the training and testing. As discussed in chapter 4 of how multi-layer ANNs form themselves and how the connections are connected, but this section will discuss in detail at the way in which the ANN neurons' weights and ANN hidden layers, which can be optimized to solve this thesis classification problem. There are definitely many considerations involved with learning such appliances' SNR signals with ANNs, and some are considered here. In the implementation with ANN, the manual Grid Search parameterization case is used to find the optimal parameters for this problem using the *caret* R package.

1. Hidden layers parameterization:

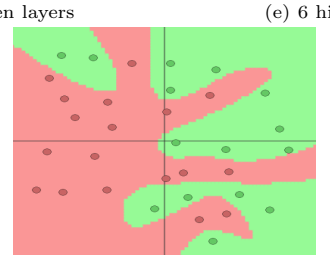
   ANN can get stuck in local minimum [74]; therefore, some optimization should be considered in this case. It is advised to consider constructing larger networks by increasing the hidden layers in the network. It is observable that the more hidden layer the network has, the more accurate the model becomes [74]. For instance, let's assume having two classes of Appliances. Three different ANNs can be visualized in which each has more hidden layers to obtain the following classification results:



(d) 3 hidden layers



(e) 6 hidden layers

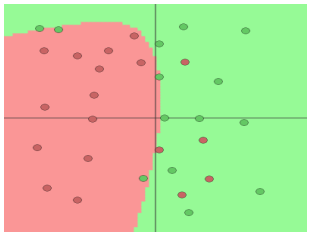


(f) 20 hidden layers [75]

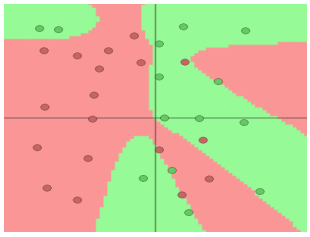
   As the diagrams show that ANNs are better with more hidden layers, but as it is observed, the model can over-fit the dataset with more layers. For instance, with 20 hidden layers which can totally fit all the appliances classes but with a cost of processing time and over-fitting problem. In contrast, the model with 3 hidden layers has a problem of under-fitting as it can lead to misclassification of some data points. Therefore, the best practice is to find the number of the hidden layers which could describe the training data in which could lead to better generalizations on the test dataset.

2. Weight decay parameterization:

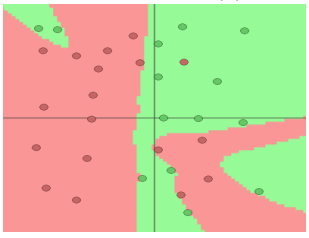
   As with the network neurons, the information in the network is stored in the weights [24, 42, 43]. Therefore, the weights need to be optimized to best categorize the training dataset. A range of weight decays is set; so the network neurons can update the weights to exponentially decay to zero; in a sense that there is no further update is scheduled for the weight. ANN then will show the best weight decay describing the dataset in which it will show the global minimum of weight decays [24, 43, 74]. To reiterate, setting a range of weight decays is the preferred way to control the over-fitting or under-fitting of ANN. The following graphs show the results achieved by three different weight decay settings:



(g) ANN 0.5 weights decay



(h) ANN 0.1 weights decay



(i) ANN 1e-3 weights decay [75]

### 5.5.5 Classification Metrics

The dataset is split into training, cross-validation and test dataset. This thesis classification models should result with two outputs. The first output is the model's accuracy percentage retrieved from confusion metrics results, and the second output is the model's Kappa percentage.

#### Confusion Metrics

A confusion matrix is a table that is used to visualize the performance of a classification model accuracy on a set of test data in which the true values are unknown. It is also called the error matrix which shows the false values in prediction. [76]

Each column, in this simple example, represents a prediction or an instance in a predicted class and each row of the matrix represents the actual class in a predicted class(vice versa). Two possible predicted classes are given: "vacuum"

| **Actual Vacuum** | **50** | **10** |
|---|---|---|
| **Actual Hair Dryer** | **5** | **100** |
| **SNR Total Values = 165** | **Predicted: Vacuum** | **Predicted: Hair Dryer** |

Table 5.3: A simple Confusion Matrix for two appliance classes

and "Hair Dryer". If we were predicting the presence of the Vacuum SNR in the channel, for example, "Vacuum" would mean the SNR noise of the vacuum exists in the channel, and "Hair Dryer" would mean the SNR noise of the Hair Dryer exists in the powerline channel. The classifier made a total of 165 different predictions where 165 SNR values were being tested for the presence of that appliance).

Out of these 165 values, the classifier predicted the "Hair Dryer" 110 times, and "Vacuum" 55 times. while, in reality, 105 SNR values are Hair Dryer and 60 SNR values are for the Vacuum. In confusion Metrics they are defined as follows:

* True positives (TP) = 100 . TP relates to the SNR values in which the model predicted for the Hair Dryer, and they are indeed for the Hair Dryer.

* True negatives (TN) = 50 . TN relates to the SNR values in which it predicted for the Vacuum, and it is also the Vacuum SNR values.

* False negatives (FN) = 5 . FN relates to the predicted SNR values for the Vacuum, but they don't actually belong to the Vacuum. This type is known as the "error rate".

* False positives (FP) = 10 . FP relates to the predicted SNR values for the Hair Dryer, but they actually don't belong to the Hair Dryer

The accuracy of a certain model can be determined from the equation [77]:

$$Accuracy = \frac{\sum TP + \sum TN}{\sum (TP + FN + TN + FP)} \tag{5.2}$$

The models will show accuracy percentage based on the values on the model performance during training and testing [78]. So, regarding the mentioned example with Vacuum/Hair Dryer SNR values:

$$Accuracy = \frac{100 + 50}{165} = 0,91$$

True predictive value for the predicted class is calculated as precision. In other words, it is considered to be the ratio of the actual positive prediction from the total amount of positive predicted observation in one class. If the precision

is high then the number of false positive predictions are very low.[78]

$$Precision = \frac{\sum TP}{\sum(TP) + \sum(FP)} \tag{5.3}$$

True Positive rate for the actual class is calculated as Recall. In other words, it is considered to be the true positive ratio of the positive predicted observations with the total number of observations in one class.[78]

$$Recall = \frac{\sum TP}{\sum(TP) + \sum(FN)} \tag{5.4}$$

True Negative Rate for the actual rate is calculated as specificity. In other words, it is considered as the true negative rate which is measured on the proportion of actual negatives that are correctly identified.[78]

$$Specifity = \frac{\sum TN}{\sum(TN) + \sum(FP)} \tag{5.5}$$

**Kappa Metrics**

*caret* is used to evaluate the models performance in R. *Kappa* or as also Cohen's Kappa metrics is set to measure accuracy of the Cross-validation dataset in this thesis classification problem. Kappa metrics is used to evaluate algorithms on multi-class classification datasets in caret. Kappa is normalized accuracy at the baseline of random chance on the dataset. [31, 66, 79] The formula to calculate kappa for two Cross-Validation repeats is:

$$\kappa \equiv \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e}, \tag{5.6}$$

where $p_o$ is the relative observed accuracy or agreement; while, and $p_e$ is the hypothetical probability of chance agreement. If the cross-validation ratings are in a complete agreement then $\kappa = 1$ [79, 80]. In contrast, if the cross-validation ratings are in disagreement depending on $p_e$ , then $\kappa = 0$.

Let's consider the mentioned predicted and actual classes of the Vacuum and Hair Dryer in table 5.3. The observed proportionate agreement is:

$$p_o = \frac{\sum TP + \sum TN}{\sum(TP + FN + TN + FP)} = \frac{100 + 50}{165} = 0.91$$

So now, it is required to calculate $p_e$ as the probability of random agreement. It can be observed that:

The model classifies "Vacuum" to 55 SNR values and "Hair Dryer" to 110 SNR values during Cross-Validation. Therefore, this model classifies "Vacuum" 33% of the time. Furthermore, the same model classifies "Hair Dryer" to 110 SNR values and "Vacuum" to 55 SNR values during another Cross-Validation. Therefore, this model classifies "Vacuum" 36% of the time.

$$p_{\text{Vacuum}} = \frac{\sum TN + \sum FN}{\sum(TP + FN + TN + FP)} \cdot \frac{\sum TN + \sum FP}{\sum(TP + FN + TN + FP)}$$

$$= \frac{50+5}{(165)} \cdot \frac{50+10}{(165)}$$

$$p_{\text{Vacuum}} = 0.33 \cdot 0.36 = 0.12$$

Similarly with the model's classification ratings of Hair Dryer:

$$p_{\text{Hair Dryer}} = \frac{100+5}{(165)} \cdot \frac{100+10}{(165)}$$

$$p_{\text{Hair Dryer}} = 0.63 \cdot 0.66 = 0.42$$

The overall random agreement probability will be the probability that both classifiers agreed on either Vacuum or Hair Dryer, and it is calculated as follows:

$$p_e = p_{\text{Vacuum}} + p_{\text{Hair Dryer}} = 0.12 + 0.42 = 0.54$$

So now when applying the formula of Cohen's Kappa, the results accuracy will be:

$$\kappa = \frac{p_o - p_e}{1 - p_e} = \frac{0.91 - 0.54}{1 - 0.54} = 0.80$$

# Chapter 6

# Implementation

This chapter explores the approaches of Powerline Anomaly Detection research using Signal-to-Noise Ratio calculations implemented on the acquired datasets and trained in Machine Learning. The implementation illustrates all followed steps with short relative code documentations, starting from data processing and concluding with Machine Learning classification. The implementation was developed in two different programming languages which are *MATLAB* and *R*.

*R* programming language provides more robust tools to further process the data and provide a various range of Machine Learning Algorithms that could help find the best fitting algorithm for our classification problem. Different libraries were used in R that could serve to fully understand and visualize the outcomes of the training and testing phases. The libraries used in our project are:

- ***Plyr*** package is a set of tools used for splitting, applying and combining data. The exact usage of this tool will be described in the following descriptions.
- ***Corrplot*** package is a graphical display tool of a correlation matrix and display of confidence interval. It also contains some algorithms to do matrix reordering.
- ***Caret*** (Classification And Regression Training) package has a set of functions that ease the streamline of the process for creating predictive models. The package tools are used for:
  - Data pre-processing
  - Splitting Data into (training, testing)
  - models tuning ( used for neural network and SVM models)
- ***doMC*** package provides a parallel backend computational functionality using the multicore functionality of the parallel package.
- ***Zoo*** package orders the observations, which include irregular time series – HF SNR frequency series in our case.

- ***Devtools*** is a package basically used for Artificial Neural Network visualization purposes.

In this chapter, it will demonstrate the Concepts 5 and the implementation of the Signal-to-noise Ratio calculations to the appliances' high frequency noise signals. Furthermore, this chapter will demonstrate the tuning parameters of our selected machine learning algorithms and how we will maximize our learning models and show the results in the coming chapter. Therefore, in these regards, this chapter is split into five parts. The first part explains data processing and visualization using *MATLAB*. This part will also explain data selection and data transformation. The second part highlights our Machine Learning methods that involved data cleaning and data optimization. The third part will discuss briefly our method of data partitioning and how many cross-validation partitions that are used. The fourth part will brief the used machine learning classification models. Lastly, the fifth part highlights the models' comparison as we will show the models' results in the coming chapter.

## 6.1 Data Processing and Visualization

Data processing is an important step in this thesis project scope. This section will explain the process of selecting the desired data values and concurrently eliminating missing values in the acquired dataset. Our data preparation steps include cleaning, instance selection, normalization, transformation. The product of data preprocessing is initial training set that will be further pre-processed according to the requirements of our Machine Learning Algorithms.

### 6.1.1 Data Selection

The acquired data is typically messy, undefined, too large and includes several missing and useless values. For example, the data contains a long period when there is no appliance activity exists. Therefore, it is required to select appliances' activities only and discard all none activities(baseline) in the data. Initially, the data provided shows each appliance toggling On/Off timestamps. The provided UNIX timestamps will used to define when appliances' activities were initiated and when ended. *MATLAB* is used in this phase as our dataset were provided in MATLAB file extension. Moreover, *MATLAB* provides us with tools to visualize and analyze the transformation of our dataset in an efficient way. The following scripts help to extract the desired data.

* *Load files.m*:In this script file, as it is a self-explanatory script, it parses the *MATLAB* data files, and load appliances' Raw Data from the directory, then store them in a *MATLAB* buffer variable.

* *ProcessHFRawData.m*: This method extracts only the High Frequency data along with UNIX timestamps, and each appliance tagging information in case they were recorded in the dataset. The fact that there are some *MATLAB* data files which contain no tagging information.

* *HF Plot.m*:This script visualizes the High Frequency time domain and time-frequency domain spectrogram.[81, 82]. This script is called to visualize the existing powerline High Frequency along with the appliances' On/Off tagging.

* *Data processing.m*: This script is split in two data processing methods:

  (a) Cutting 85% of the baseline High Frequency data and extract only the timestamps when the appliances' event first start and end.

```
% UNIX time when appliance activities happen
min_ts = min(cellfun(@(x)x(1), Buffer.TaggingInfo(:,3))
    );
max_ts = max(cellfun(@(x)x(1), Buffer.TaggingInfo(:,4))
    );

% HF Indexes where activities happen
start_idx_HF = min(find(int64(Buffer.HF_TimeTicks(:,1))
    == int64(min_ts) ));
stop_idx_HF = max(find(int64(Buffer.HF_TimeTicks(:,1))
    == int64(max_ts) ));

% Cutting HF where the appliance where On/Off events
    occur
ProcessedData.HF_TimeTicks = Buffer.HF_TimeTicks(start_
    idx_HF:stop_idx_HF);
ProcessedData.HF = Buffer.HF(:, start_idx_HF:stop_idx_
    HF);
ProcessedData.TaggingInfo = Buffer.TaggingInfo;
```

Listing 6.1: Defining when appliances' activities happen [15]

  (b) *Extrapolation*: Extrapolation algorithm is used to calculate the outlier points in the dataset. It is observed that some values are missing or misrepresenting each High frequency values within the 4096-points vector. It is found that these patterns repeat themselves in the whole dataset. Therefore, by extrapolating the missing values, our dataset will be more representative for each appliance High frequency points.

```
output = interp1(x, A, xi,'nearest' ,'extrap');
```

Listing 6.2: extrapolating method in MATLAB.

The extrapolation calculates the missing values on the basis of their relationship with other nearest values that are more representative.

### 6.1.2 Signal-to-Noise Ratio (SNR) data transformation

The theoretical concept of how to use SNR and Machine Learning to classify appliances' SNR used in powerline modulation in a household was explained in chapter 4 and chapter 5. This section will focus on the code documentation calculations of the theoretical concept.

Despite the various types of modulation schemes used in powerline modems, the modulation signal was defined as a constant value. However, the noise

reference here in SNR indicates appliance X frequency noise in the acquired dataset (FFT vectors during t time) as the following formula defines

$$SNR = \frac{signal\ energy}{noise\ energy}$$

* *Data procssing2.m*:This script contains the vital calculations where each appliance On/Off timestamps along with their High frequency noise are extracted and SNR is calculated. It also includes the SNR calculation of the baseline High Frequency noise, or when there is no appliance activity occurring. The baseline is added as a class to be later classified as the baseline SNR. For each class, the maximum UNIX timestamp was defined to be 30 seconds as we noticed that appliances' activities ranged roughly between 30-500 seconds.

```
% Calculating SNR for each appliance event
constant = 255 * ones(4096,1); % 255 dbVm
SNR = 10*log10(constant./double(Buffer.HF(:, i)));
```

Listing 6.3: SNR calculation

After calculating the appliances' SNR for each appliance Noise High frequencies, the data is then transformed and stored in a *.txt* file to be used in Machine Learning steps.

### 6.1.3 Data cleaning

In R, missing values usually exist, and they are represented by the symbol *NA* (not available). Therefore, in this step, after cropping the data according to first and last appliances' SNR activities, any missing values in the dataset are removed.

### 6.1.4 Data Validation and Optimization

In common usage in machine learning, two variables or more are having a linear relationship with each other. The purpose of using feature correlation is to decide which of the SNR features to include in the final training subset and which to ignore. If there are n SNR features, there are $2^n$ possible subsets for training[83]. The only method to find out the best possible subset would be trying all of them. The latter can be very expensive in terms of computational power and time complexity. Therefore, feature correlation search strategies are used in this dataset to search the best subset space in a reasonable computational time. The R function caret :: findCorrelation calculates a matrix of feature-class and feature-feature correlations, and returns a vector of integers, ranges between -1 and 1, corresponding to each variable which, if removed, would reduce pair-wise correlations among the remaining variables.

The amount of features are deduced, but the dataset still keeps a large portions of the information contained in the data. Less features are obtained,

which still represent the same information. Feature correlation technique is essential for four reasons:

1. Simplification for ML models
2. Shorter processing time for training and testing
3. Avoiding appliances complex multi-dimensionality
4. Enhancing models' generalization to the testing dataset by reducing data overfiting problem

```
# feature correlation as plot
corrplot(cor(appliance$data[,2:100]), tl.cex = 0.3)
```

Listing 6.4: Feature correlation plot method in R

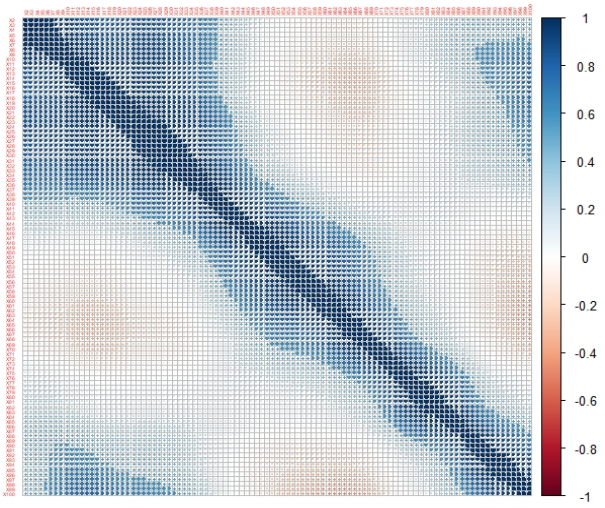



Figure 6.1: Feature Correlation matrix between one appliance SNR variables in the dataset

```
# remove correlated variable using findCorrelation
foundCorIndexes <- findCorrelation(cor(appliance$data))
# foundCorIndexes
corrplot(cor(appliance$data[,-foundCorIndexes]), tl.cex = 0.3)
```

Listing 6.5: Feature correlation method in R

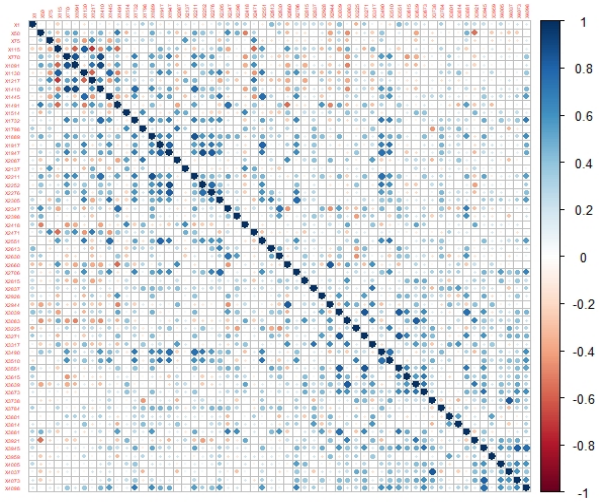



Figure 6.2: Feature Correlation matrix after finding the unique features

This graph 6.2 indicates one appliance SNR as features. When there is no correlation between two SNR features, the threshold comes roughly around 0 and the color is white. The blue circles indicate that there is a perfect positive correlation between the two features, while the red circles indicate that there is a perfect negative correlation.

When evaluating the correlation among all SNR features, the The caret :: findCorrelation method includes the correlation of each feature with itself, which is always 1 represented in blue circle, so this type of graphs always have the blue diagonal from the upper left to the lower right. Other than the diagonal, the rest of the squares show correlation between different features, making it really easy to find the peaks that are highly correlated in each appliance SNR values.

## 6.2 Data Partitioning

Any supervised ML method usually requires splitting data into multiple chunks that are used for training, cross-validating, and finally testing the classifiers. For finding the best parameters of ML classifiers, training and validation are usually carried out with cross-validation. 75% of the data was partitioned for training and cross-validation; while the remaining 25% of the data was used for testing the accuracy of the classifiers. Regarding cross-validation, 20 cross-validation repeats were used and split the 75% data in 10 cross-validation partitions where 8 of these partitions are randomly chosen for training and the remaining 2 are also randomly chosen for cross-validation.

```
trControl <- trainControl(
method = 'repeatedcv',
```

```
number = 10, # number of Cross-Validation partitions
repeats = 20, # number of partitioning repetitions
returnResamp = 'final', # (return Cross0Validation partition
    results for best model)
```

Listing 6.6: Cross-validation method in R

## 6.3 Machine Learning Classification Models

This section demonstrates classification of appliances based on their SNR values using different classification algorithms. All of the selected Machine Learning Models are used to solve multiple classification problems. Each model training is going to be supervised due to the fact that the desired outputs from the given inputs are already defined and the data has been already labeled. The dataset is an SNR frequency series data as it is recorded with respect to appliances' SNR values. After preprocessing and organizing the data, it is time to feed the dataset into the classification models. The models use each appliance SNR features as inputs in a multi-dimensional space to classify each input separately.

### 6.3.1 k-nearest neighbors algorithm (KNN)

KNN is used as it is non-parametric and cheap. Therefore, the result should not take longer than 5 minutes to compute, unless none-preprocessing steps were conducted before feeding the data into KNN.

```
models$knn <- train(training,
factor(training_SNR),
method = 'knn',
metric = 'Kappa',
trControl = trControl)
models$knn
```

Listing 6.7: KNN method in R

As it can be observed from the code above, *caret* is used to evaluate the KNN model. *Kappa* metrics were also selected to measure kappa accuracy of the Cross-validation dataset in equation 5.6, and *Confusion* metrics to determine the overall accuracy of the model according to equation 5.2 in chapter 5 section 5.5.5. KNN should result with a final accuracy, which best classify our dataset with the best K neighbor integer. So, in order to test the effectiveness of the KNN model, the testing dataset is passed to demonstrate the classifier accuracy.

```
predicted <- predict(models$knn, newdata = testing)

# to ensure, that also when one level is not predicted, the
    results can be displayed
u = union(predicted, testing_SNR)
t = table(factor(predicted, u), factor(testing_SNR, u))
conf <- confusionMatrix(t)
```

```
levelplot(sweep(conf$table, MARGIN = 2, STATS = colSums(conf$table
    ), FUN = '/'), col.regions = gray(100:0/100), aspect="fill",
    scales=list( x=list(rot=45)))
```

Listing 6.8: Tesing KNN model with the testing dataset

Confusion Matrix is used to state the amount/rate of samples being correctly/not correctly classified. The confusion matrix shows each appliance actual and predicted classes with their associated accuracies.

### 6.3.2 Linear Discriminant Analysis (LDA/LDA2)

LDA and LDA2 are two classic ML classifiers, with two dimensional linear boundary or multi-linear boundary classification, respectively. These classifiers are cheap in computation because they have linear solutions that can be easily computed. As they are known to be inherently linear multi-class classifiers, they have proven to work well in practice, and have no tuning hyperparameterizations.

```
models$lda <- train(training,
factor(training_SNR),
method = 'lda',
metric = 'Kappa',
trControl = trControl)
models$lda
```

Listing 6.9: LDA method in R

LDA and LDA2 are also included in *caret* library. *Kappa* metrics were used to measure kappa accuracy of the Cross-validation dataset in equation 5.6, and *Confusion* metrics to determine the overall accuracy of the model according to equation 5.2 in chapter 5 section 5.5.5. LDA should result with a final maximum accuracy which best classifies the dataset. It stops when it finds the first results, that is why it is cheap in computation. Similarly to code in listing 6.8, Confusion matrix is used after demonstrating the model accuracy with the testing dataset.

```
predicted <- predict(models$lda, newdata = testing)
```

Listing 6.10: Tesing LDA model with the testing dataset

Meanwhile with LDA2, it results with a final global maximum accuracy which best classifies our dataset in a multi-dimensional space.

```
models$lda2 <- train(training,
factor(training_SNR),
method = 'lda2',
metric = 'Kappa',
trControl = trControl
)
models$lda2
```

Listing 6.11: LDA2 method in R

It is more expensive comparing to LDA as it continues finding other spaces accuracies and returns with the best accuracy given. Similarly to code in listing 6.8, Confusion matrix is used to demonstrate the model classification accuracy with the testing dataset.

```
predicted <- predict(models$lda2, newdata = testing)
```

Listing 6.12: Tesing LDA2 model with the testing dataset

### 6.3.3 Support Vector Machine (SVM)

As it was described in chapter 5 and chapter 4, SVM is considered to be a large margin classifier which determines the optimal hyperplanes that maximize the margin between appliances' SNR classes. Therefore, SVM is very expensive in computation; especially Radial SVM. It could take hours of training which best classifies the dataset, but it is definitely more over-fitting.

SVM models are both parameterizable to best adapt our classification problem. Different parameters were selected to best classify the dataset.

#### Linear SVM

```
train_model <- function(method, tuneGrid=NULL) {
train(x = training,
y = training_SNR,
method = method,
metric = 'Kappa',
tuneGrid = tuneGrid,
trControl = trControl)}
models$svmLinear <- train_model('svmLinear', tuneGrid = expand.
    grid(C=(-5:5)))
```

Listing 6.13: Linear SVM method with its hyper-parameterizations in R

As it can be seen, the C parameter was adjusted in a range from -5 to 5 linear space margin. These 10-step C parameter will redirect the linear boundary to avoid misclassifying each appliance's SNR. Similarly to code in listing 6.8, Confusion matrix is also used to demonstrate Linear SVM's classification accuracy with our testing dataset.

```
predicted <- predict(models$svmLinear, newdata = testing)
```

Listing 6.14: Tesing Linear SVM model with the testing dataset

#### Radial SVM

```
train_model <- function(method, tuneGrid=NULL) {
train(x = training,
y = training_SNR,
method = method,
metric = 'Kappa',
```

```
tuneGrid = tuneGrid,
trControl = trControl)}
models$svmRadial <- train_model('svmRadial', tuneGrid = expand.
    grid(C=(-5:5), sigma=(-5:5)))
```

Listing 6.15: Radial SVM method with its hyper-parameterizations in R

As it can be seen, two parameters are adjusted with the Radial SVM model:

1. Similar to Linear SVM, the C parameters range from -5 to 5 linear space margin. These 10-step C parameters will redirect the radial boundary.
2. We adjusted the sigma parameter to range between -5 to 5; thus making the model redirecting the radial boundaries and precisely avoid misclassification.

Similarly to code in listing 6.8, Confusion matrix is used to demonstrate Radial SVM's classification accuracy with the testing dataset.

```
predicted <- predict(models$svmRadial, newdata = testing)
```

Listing 6.16: Tesing Radial SVM model with the testing dataset

### 6.3.4 Artificial Neural Network (ANN)

Similarly to SVM models, parametrized ANNs are redirected to best adapt the classification problem. There are many considerations involved with the parameterizations. In this thesis implementation with ANN, the manual Grid Search parameterization case is used to find the optimal parameters for the problem using the *caret* R package. Parameterizations are defined as follows:

#### Hidden layers parameterization

As it was described in chapter 5.5.4, ANN can get stuck in local minimum in classification; therefore, optimization parameters need to be defined. Constructing a larger ANN is implemented with increasing the hidden layers in the network. The more hidden layers in the network, the more accurate ANN model becomes.

```
train_model <- function(method, tuneGrid=NULL) {
train(x = training,
y = training_SNR,
method = method,
metric = 'Kappa',
tuneGrid = tuneGrid,
trControl = trControl)}

nnet_grid <- expand.grid(.decay = c(0.5, 0.1, 1e-2, 1e-3, 1e-4, 1e
    -5, 1e-6, 1e-7), .size = c(3, 5, 10, 20))
```

Listing 6.17: ANN method with its hyper-parameterizations in R

As the code listing 6.17 shows that the size of the hidden layers are defined to reach to 20 maximum, but it is important that the model may overfit the dataset with 20 layers; in contrast, the model can underfit using only 3 hidden layers. However, this range of layers are used to demonstrate the ultimate results of ANN model. Therefore, the best practice is to set a range of the hidden layers. The latter could describe each of the training data separately in which could lead to better generalizations on the test dataset.

### Weight decay parameterization

As it was mentioned regarding the network neurons in chapter 5 and chapter 4, the information in the network is stored in the neurons' weights. Therefore a range of weights parameters are set, as code listing 6.17 shows, to best categorize the training dataset. ANN will keep updating the weights until no further update is scheduled for the weights (weight decay). The results in code list 6.18 will show the best scheduled weights in our ANN model .

```
plot(models$nn, scales = list(x = list(log = 3)))
levelplot(x = Kappa ~ size * decay, data = models$nn$results[
    models$nn$results$decay!=3 & models$nn$results$size != 1,],
    col.regions=gray(100:0/100), scales=list(y=list(log=3)))
```

Listing 6.18: Showing weights decay method in R

Similarly to code in listing 6.8, Confusion matrix is used to demonstrate ANN's classification accuracy with the testing dataset.

```
predicted <- predict(models$nn, newdata = testing)
```

Listing 6.19: Tesing ANN model with the testing dataset

## 6.4 Models' Results Comparison

A comparison of the performance of the six machine-learning classification models should demonstrate the best classifying model for the problem. The results should define the minimum, the mean and the maximum accuracies range as the code listing in 6.20 shows.

```
results <- resamples(models)
summary(results)
bwplot(results)
```

Listing 6.20: Models comparison method in R

# Chapter 7

# Results

This chapter will explore the results of Powerline Communication Anomaly Detection classifiers. It will represent both the major and the minor findings accompanied with each step. The first section of this chapter will focus on the Data Processing and Visualization results with both *MATLAB* and *R* data preprocessing as explained in section 6.1. The second part will explain our Machine Learning models results as described in section 6.3. The last part will give a detailed comparison among all Machine Learning models as described in section 6.4

## 7.1 Data Processing and Visualization

This section will show the results of processing the data by selecting the desired data and concurrently eliminating unwanted data in the dataset. It will also provide a full detailed analysis and visualizations related to the data transformation and preparation.

### 7.1.1 Data Selection

The acquired data is typically messy, undefined, too large and consists of missing and useless values. Prior to visualization and analysis, the raw data needs to be parsed, cleaned, transformed. It is mandatory to define the exact data and visualize it to pre-process it later and train it with Machine Learning. Initially, the data contains all appliances' noise High Frequency signals and their UNIX timestamps datasets including the long periods of no-activity of appliances. The following figure shows House 4 time-frequency domain visualization.

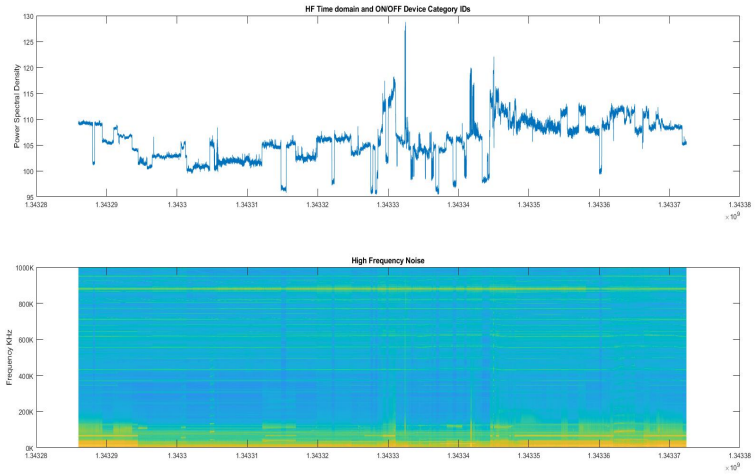



Figure 7.1: Powerline High Frequency in time-frequency domain

The figure above shows powerline time domain actuation and the graph below shows the frequency-time domain spectral density. The figure has no definitions of when exactly the appliances were toggled; thus, making it difficult to extract the exact data for each appliance high frequency noise behavior. The frequency-time domain spectral density graph also shows the 4096xN waterfall spectrogram of the captured powerline high frequency noise. N is the number of FFT vectors, with each being computed every 1.0667 seconds.

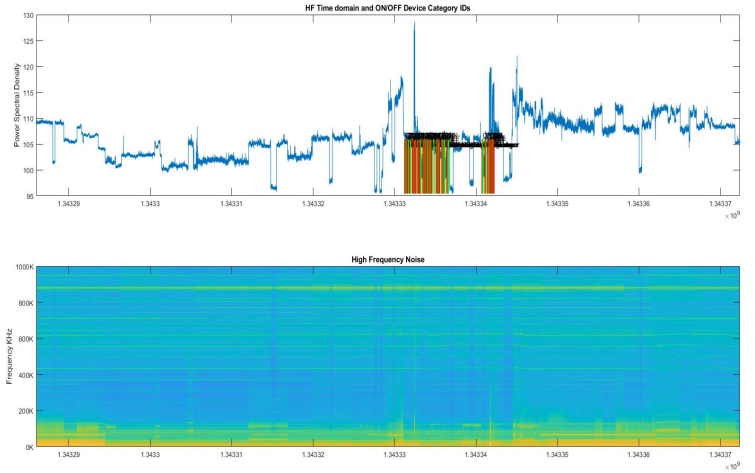



Figure 7.2: Powerline High Frequency On/Off Unix timestamps

After extracting each appliance On/Off timestamps from the provided dataset, it was observed that the appliances toggling UNIX time was present solely in the middle as the graph above shows. It is observable that 85% of the data contains baseline High frequency fluctuations that are huge, unrepresentative and unwanted. Therefore, these signals are required to be eliminated and downsized our data to when first appliance was turned on to the last instance when the last appliance was turned off. The following graph shows appliances' extracted activities.

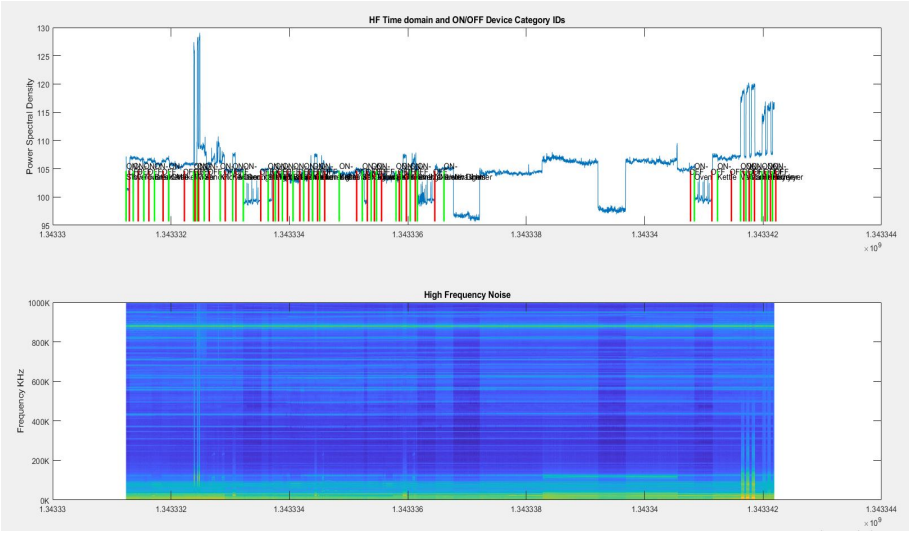



Figure 7.3: Extracted Powerline HF On/Off Unix timestamps

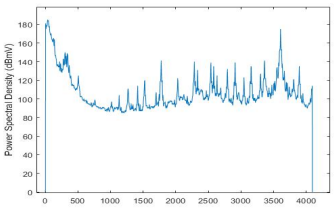
As the figure above shows, only the period when the appliances noise activities first start and end were retrieved. In this dataset, only 14 different appliances HF noise are present. The following table shows the amount of the appliances presented in this dataset which will be used in the training, cross-validation and testing.

| **Appliances** | **On/Off** | **Period** |
|---|---|---|
| Stove | 3 | 30 |
| Microwave | 3 | 30 |
| Toaster Oven | 3 | 30 |
| Bread Maker | 3 | 30 |
| Kettle | 3 | 30 |
| Mixer | 3 | 30 |
| Sandwich Maker | 3 | 30 |
| Kitchen Counter Lights | 3 | 30 |
| Kitchen Lights with Dimmer | 3 | 30 |
| Oven | 3 | 30 |
| Dishwasher | 3 | 30 |
| Vacuum | 3 | 30 |
| Hairdryer | 3 | 30 |
| Baseline | 0 | 29 |

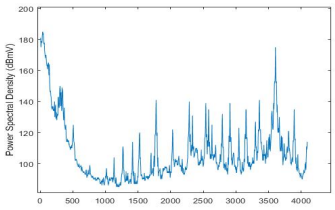
Table 7.1: Appliances presented in our dataset

As table 7.1 shows the existing appliances in the dataset that will be trained and tested using the models. Each appliance was turned on/Off in three different times of the day, and the first 30 seconds of each event were extracted. However, it is observed that the Baseline period could be underrepresented, which could cause the models to mis-classify; however, the effect of this will be observed when training and testing later on.

The data contains unrepresentative HF noise values that need to be extrapolated. For instance, the FFT vector bins for each appliance Noise start at 0 dBmV and suddenly jump to 100 dBmV after 4 FFT bins. However, in the whole dataset, this behavior is repeated in all FFT vector bins with a range of 8 bins out of 4096 FFT bins. 4 FFT bins in beginning of the HF FFT vector and 4 at the end. A *MATLAB* function *interpl* is used to estimate the values of the missing FFT bins on the basis of their relationship with the nearest values of the FFT bins.




(a) Original frequency vector




(b) Extrapolated frequency vector

### 7.1.2 Signal-to-Noise Ratio (SNR) data transformation

The theoretical concept of how to use SNR and Machine Learning to classify the appliances' SNR used in powerline modulation in a household was explained in chapter 4 and chapter 5. This section will focus on the calculations results of the theoretical concept.

The references to the Signal varies in the field of electrical signals as described in chapter 4, but in this thesis implementation, it was referred to it as a powerline communication modulation signals used in Powerline Communication modems. The modulation signal was defined as a constant value above 180 dB across the entire frequency FFT vector bins. In the data itself, it is noticed that the highest spectral density magnitude reaches roughly 127 dB which maps to 255 dB. Therefore, the constant signal is defined at 255 dB across FFT vector bins. The noise reference here in SNR indicates appliance X frequency noise (FFT vectors during t time) as the following formula defines

$$SNR = \frac{signal\ energy}{noise\ energy}$$

All SNR calculations are included, also when there is no appliance activity occurring (Baseline), and it was added as a class to be classified. As mentioned, the maximum UNIX timestamps for each appliance activity was determined to be 30 seconds as it was noticed that appliances' activities ranged between 30-500 seconds as our table 7.1 shows. The reason is to have the least dataset that can be trained later in machine learning. In addition to help the prediction speed to be more accurate within the first seconds of predicting the new testing dataset.

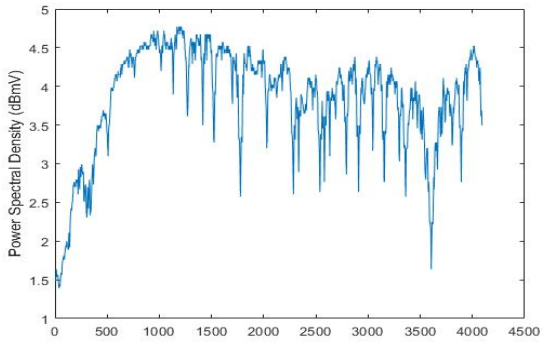



Figure 7.4: Signal to Noise ratio frequency signal for appliance X

After calculating the appliances' SNR including the baseline residual noise, the data is then transformed and stored in a .txt file to be used in Machine Learning steps. In the machine learning implementation, it was decided to conduct the experiment by training and testing using appliances' SNR noise sampled at 1 MHz which covers the whole 4096xN spectrogram. In this implementation,

one preprocessing method was used which is required for the dataset. Data optimization using correlation matrix mentioned in 6.1.4 has been used. The reason it is required to use the preprocessing method is to reduce the features dimensionality in the dataset as it may take weeks to a month to only train the dataset. For instance, running the implementation without this step with the training dataset could take days to a week. Therefore, it is advised to obtain less features, which still represent the same information. The results gathered are still expensive with each model as some of the models took some minutes to a couple of hours for training. We will demonstrate the results and findings of each model here. The following plot shows the first 10 raw appliance X SNR values without any preprocessing technique used.

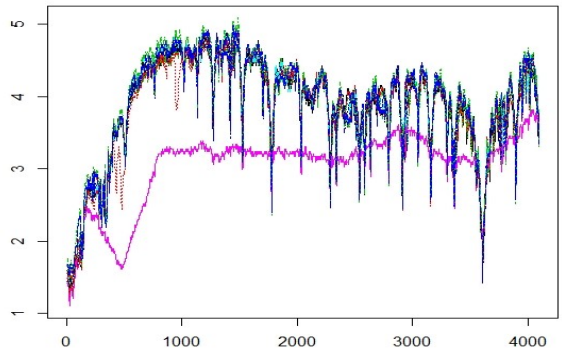



Figure 7.5: Appliance X SNR values over the 1 MHz frequency

### 7.1.3 Data Validation and Optimization

Feature correlation preprocessing technique is used to the existing dataset. It is observed that several SNR values for one appliance have similarities with its other SNR values along the 4096-point vectors, and it is needed to filter all but keep the unique SNR values of each appliance SNR features. caret :: findCorrelation method set a correlation threshold so the exacted features could be linearly correlated for each appliance. The method calculates a matrix of feature-feature correlations, and returns a vector of integers, ranges between -1 and 1 which indicate the strength of each feature correlation. The following graph shows the first 100 SNR values (features) for one appliance and how correlated they are from each other.

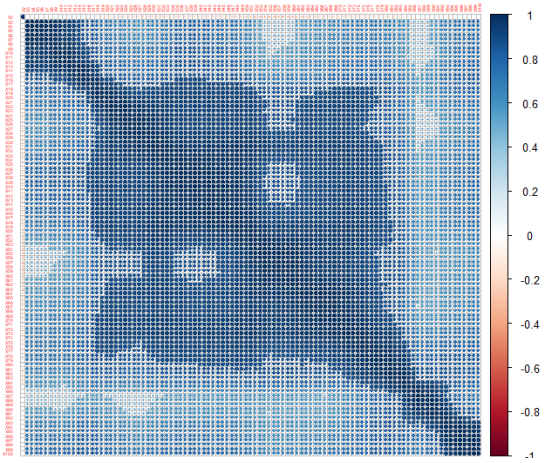



Figure 7.6: feature correlation for different appliances' SNR values

The matrix above shows the first hundred features of one appliance SNR values and how they are correlated. caret :: findCorrelation method performed a heuristic algorithm that reduced the feature-feature and feature-class space and resulted with the best possible subset for our training as the following Feature correlation matrix shows.

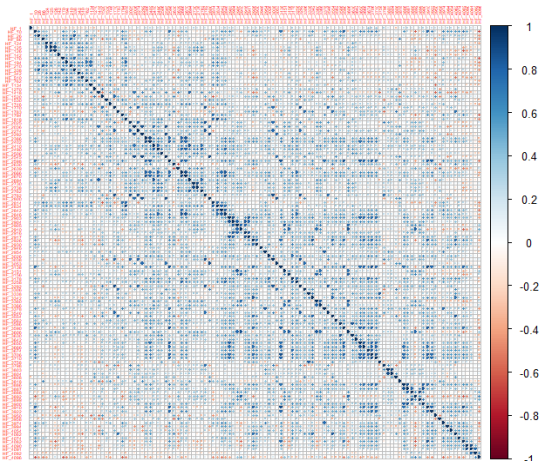
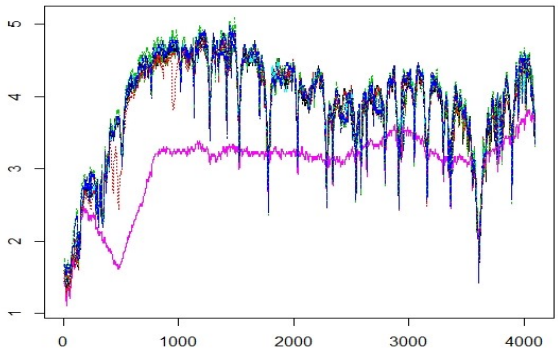
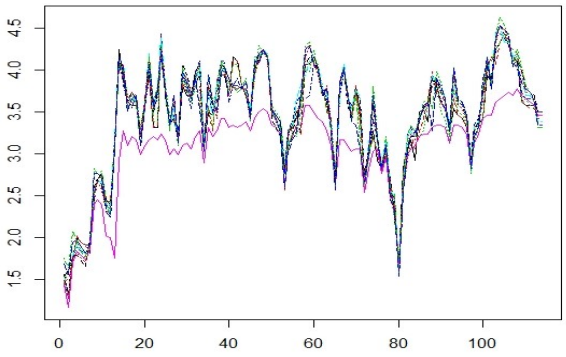



Figure 7.7: feature correlation for different appliances' SNR values

The features selected are expected to be the useful subset for training and predicting each appliance class. The current training subset contains the unique SNR signatures of each appliance which can be trained in our Machine Learning models. The following graph shows the first 10 SNR values before and after feature-feature correlation is taken.




(a) Data before correlation




(b) Data after correlation

It is observed that caret :: findCorrelation method removed the duplicate features in the entire SNR values and left the important features that indicated the possible higher importance of the feature for each class.

## 7.2 Machine Learning Models Results

This section will demonstrate ML classification results of appliances based on their SNR values using different classification algorithms. The dataset was split as 70% training set which had roughly 700x4096 different appliances, and 30% testing set which had roughly 300x4096 different appliances.

### 7.2.1 k-nearest neighbors algorithm (KNN)

KNN performed quick in training as it is a non-parametric ML model, it took us a few seconds for training each appliance SNR values as shown in table 7.16. In this thesis implementation, k was oriented to be k = 10 as the possible nearest neighbors. According to the final results, only 3 neighbors decided the nearest segregating boundaries of each appliance class, but only the best k neighbor is chosen. The following table illustrates the highest k fold accuracies.

| **k** | **Accuracy** | **Kappa** |
|---|---|---|
| 5 | 95,5% | 95,0% |
| 7 | 95,1% | 94,6% |
| 9 | 94,9% | 94,4% |

Table 7.2: Comparison of different k results in KNN

Table 7.2 shows the overall training accuracy and kappa for classifying appliances SNR. Using 10-fold cross validation, it is observed with an overall average kappa of 95,6%. KNN selected the optimal k model with the largest kappa. Therefore, the final value used for KNN model was k = 5.

Since the used N-fold CV is generally optimistic and could be not a true measure of the expected classification performance;hence, it is mandatory to run this model results on our testing dataset. The overall accuracy of the model was analyzed with the testing dataset and it reached to 93,3%. With analyzing the confusion matrices of each appliance (see Figure 7.8 for the confusion matrix), it was found that there was some confusion between the baseline SNR values and microwave, toaster oven and Dishwasher. The reason for this confusion is believed to be due to the low appliances SNR amplitude values that reach the same amplitude dB level as the baseline SNR values.

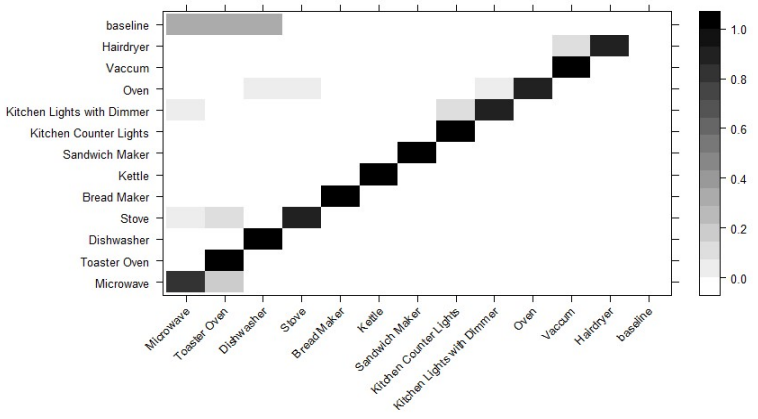



Figure 7.8: Visual confusion matrix highlighting KNN misclassification

From the given confusion matrix of test dataset results, it is observed that the KNN model classifier correctly identified different appliances' SNR values. However, it failed to classify the baseline SNR values and confused them with other appliances SNR low values. The reason for baseline misclassification is due to the baseline period underrepresentation as we mentioned in table 7.1. The following table shows the classes accuracy, precision, specificity, precision and recall percentages for each appliance testing dataset.

| **Class** | **Specificity** | **Precision** | **Recall** | **Accuracy** |
|---|---|---|---|---|
| Microwave | 0.986 | 0.850 | 0.809 | 0.897 |
| Toaster Oven | 0.968 | 0.758 | 1.00 | 0.984 |
| Dishwasher | 0.991 | 0.777 | 1.000 | 0.995 |
| Stove | 0.995 | 0.950 | 0.863 | 0.929 |
| Bread Maker | 1.000 | 1.000 | 1.000 | 1.000 |
| Kettle | 1.000 | 1.000 | 1.00 | 1.000 |
| Sandwich Maker | 1.000 | 1.000 | 1.000 | 1.000 |
| Kitchen Counter Lights | 0.990 | 0.916 | 1.000 | 0.995 |
| Kitchen Lights with Dimmer | 0.995 | 0.950 | 0.863 | 0.929 |
| Oven | 1.000 | 1.000 | 0.863 | 0.931 |
| Vaccum | 1.000 | 1.000 | 1.000 | 1.00 |
| Hairdryer | 1.000 | 1.000 | 1.000 | 1.000 |
| baseline | 1.000 | NA | 0.000 | 0.500 |

Table 7.3: Comparison of different classes results in KNN

### 7.2.2 Linear Discriminant Analysis (LDA/LDA2)

It was observed that both LDA and LDA2 have almost similar results. Both models were less expensive in computation as they more time to classify the training dataset as it is shown in table 7.16. Each LDA and LDA2 had different results, and each is presented as follows:

#### LDA

The dataset was trained with LDA, and it was found that LDA reached its maximum accuracy after a couple of minutes of classification. According to the final results, only one dimensional maximum accuracy decided the boundaries for each appliance class. The following table illustrates the maximum accuracy and kappa resulted from our training. As the used N-fold CV was generally optimistic to measure the expected classification performance for a real world

| Accuracy | Kappa |
|---|---|
| 90,4% | 89,5% |

Table 7.4: LDA local maximum accuracy

SNR values, the performance in prediction was run with this model results on the testing dataset. With analyzing the confusion matrices of each appliance (see Figure 7.9 for the confusion matrix), it was found that LDA had better classified the baseline SNR values with 60% accuracy result, but it also confused them with the toaster oven and kitchen lights. The model also misclassified other appliances' SNR. For example, the toaster oven, vacuum, kitchen light were confused with dimmers and kitchen counter lights. The reason for that could be the possibility of the strong correlation of each appliance SNR values.

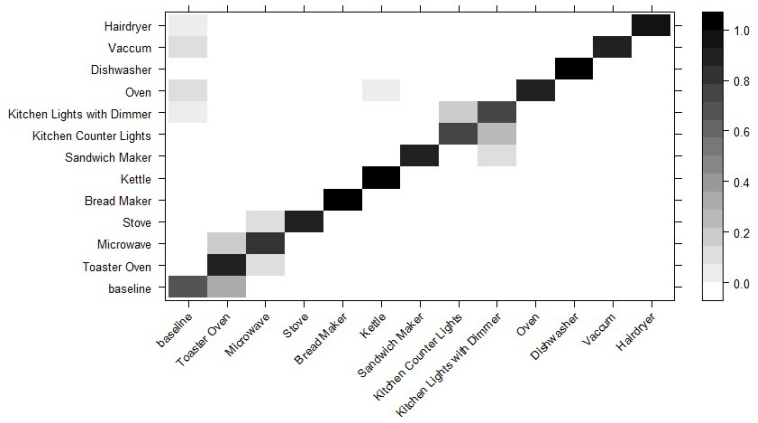



Figure 7.9: Visual confusion matrix highlighting LDA misclassification

From this confusion matrix of the test dataset results, it was observed that the LDA classifier correctly identified all appliances' SNR values with high accuracies. However, it also failed to classify other appliances SNR with low error misclassification rate. The following table shows all classes' accuracy, precision, specificity, precision and recall percentages.

| Class | Specificity | Precision | Recall | Accuracy |
|---|---|---|---|---|
| baseline | 0.974 | 0.250 | 0.666 | 0.820 |
| Toaster Oven | 0.977 | 0.791 | 0.863 | 0.920 |
| Microwave | 0.977 | 0.772 | 0.809 | 0.893 |
| Stove | 1.000 | 1.00 | 0.909 | 0.954 |
| Bread Maker | 1.000 | 1.00 | 1.000 | 1.000 |
| Kettle | 0.995 | 0.956 | 1.000 | 0.997 |
| Sandwich Maker | 1.000 | 1.000 | 0.909 | 0.954 |
| Kitchen Counter Lights | 0.981 | 0.809 | 0.772 | 0.877 |
| Kitchen Lights with Dimmer | 0.968 | 0.809 | 0.772 | 0.870 |
| Oven | 1.000 | 1.000 | 0.86 | 0.931 |
| Dishwasher | 1.000 | 1.000 | 1.000 | 1.000 |
| Vacuum | 1.000 | 1.000 | 0.904 | 0.952 |
| Hairdryer | 1.000 | 1.000 | 0.952 | 0.976 |

Table 7.5: Comparison of different classes results in LDA

As the table shows, the True Negative Rate (specificity) for the actual rate average was 98%. The Positive predictive value (Precision) reached an average of 88% excluding the precision rate for the Baseline which reached 25,5%. The True positive rate (Recall) for the actual classes reached 80% in average as the table above shows. The reason for that was the low recall percentage for the Baseline which reached 66,6%. Finally, with the accuracy average reached 92%.

### LDA2

The dataset is trained with LDA2, and as mentioned in chapter 6 that LDA2 results with multiple final accuracy which best classifies our dataset in a multi-dimensional space. Each accuracy is represented in a multi-dimensional classification as table 7.6 shows. LDA2 is more expensive in terms of computation or run-time complexity as it will be shown in table 7.16. According to the final training results, there were 12 dimensions LDA2 used to decide the boundaries for each appliance class. The following table illustrates the different N dimensions results.

| dimen | Accuracy | Kappa |
|---|---|---|
| 1 | 35,4% | 29,15% |
| 2 | 68,0% | 64,9% |
| 3 | 82,6% | 80,9% |
| 4 | 85,2% | 83,8% |
| 5 | 88,0% | 86,9% |
| 6 | 88,8% | 87,8% |
| 7 | 88,7% | 87,6% |
| 8 | 89,8% | 88,8% |
| 9 | 89,8% | 88,9% |
| 10 | 90,5% | 89,7% |
| 11 | 90,5% | 89,6% |
| 12 | 90,5% | 89,7% |

Table 7.6: Comparison of different dimensions results in LDA2

Table 7.6 shows the overall accuracy of each dimension and kappa for classifying appliances SNR. With more dimensions, LDA2 classifies with better accuracy, until it reaches the highest dimensions where it can no longer classify with better accuracy. The highest dimensionality is considered to be the maximum accuracy LDA2 could reach. Using 10-fold cross validation, it was observed that the overall average kappa reached 86%. LDA2 selected the optimal N dimensions with the largest kappa. Therefore, the final value used for LDA2 model was dimen = 10.

By analyzing the confusion matrices of each appliance (see Figure 7.10 for the confusion matrix), it was found that both LDA and LDA2 resulted with the same confusion between the baseline SNR values with the kettle and kitchen counter lights. It also resulted with a close Kappa and accuracy results with LDA, the reason why the results might have looked similar, but with less confusion with other appliances' SNR values. For instance, the Kitchen Lights with Dimmer and Sandwich maker misclassification in LDA in figure 7.9 had been eliminated in LDA2 as our confusion matrix figure 7.10 shows.

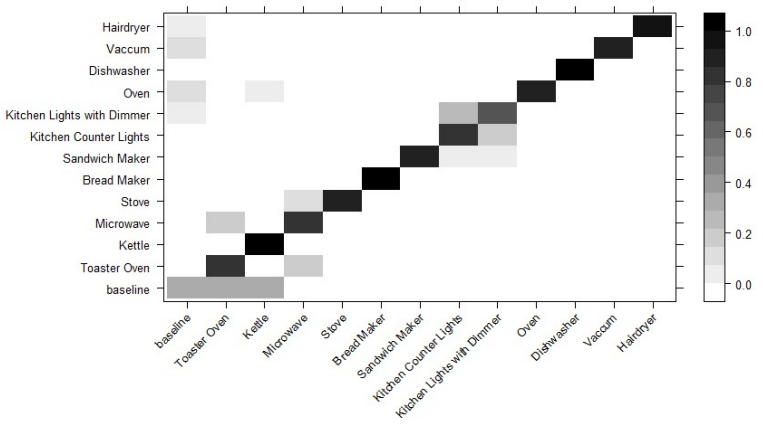



Figure 7.10: Visual confusion matrix highlighting LDA2 misclassification

From the given confusion matrix of test dataset results, it was observed that the LDA2 classifier also correctly identified different appliances' SNR values with better accuracy comparing to LDA1. However, similarly to LDA, it also failed to classify other appliances SNR with low error misclassification rate. The following table shows different classes' accuracy, precision, specificity, precision and recall percentages.

| **Class** | **Specificity** | **Precision** | **Recall** | **Accuracy** |
|---|---|---|---|---|
| baseline | 0.974 | 0.142 | 0.333 | 0.654 |
| Toaster Oven | 0.977 | 0.782 | 0.818 | 0.897 |
| Kettle | 0.990 | 0.916 | 1.000 | 0.995 |
| Microwave | 0.972 | 0.739 | 0.809 | 0.891 |
| Stove | 1.000 | 1.000 | 0.909 | 0.954 |
| Bread Maker | 1.000 | 1.000 | 1.000 | 1.000 |
| Sandwich Maker | 1.000 | 1.000 | 0.909 | 0.954 |
| Kitchen Counter Lights | 0.968 | 0.720 | 0.818 | 0.893 |
| Kitchen Lights with Dimmer | 0.977 | 0.750 | 0.681 | 0.829 |
| Oven | 1.000 | 1.000 | 0.863 | 0.931 |
| Dishwasher | 1.000 | 1.000 | 1.000 | 1.000 |
| Vacuum | 1.000 | 1.000 | 0.904 | 0.952 |
| Hairdryer | 1.000 | 1.000 | 0.952 | 0.976 |

Table 7.7: Comparison of different classes results in LDA2

### 7.2.3 Support Vector Machine (SVM)

SVM is a large margin classifier, it determines the optimal hyperplanes that maximize the margin between appliances' SNR classes with a very expensive computational time; namely Radial SVM is considered the most expensive model that used in this thesis as it will be shown in table 7.16. It is assumed that it is expensive due to one key factor that plays into the complexity of the run-time which is the slack in C and Sigma parameters which were set heuristically. The following results represent our accuracies and kappa percentages as trained and tested in our implementation.

#### Linear SVM

The dataset was trained with SVM Linear with C parameters for model tuning. SVM Linear is more expensive in computation comparing to all previous models as it will be shown in table 7.16; however, the resulting accuracy was similar to LDA and LDA2. As the C parameters were solely adjusted since Linear SVM is a linear classification method, the C parameter redirected the SVM to the best possible results in this classification problem. The following table shows different C parameters accuracies and Kappa percentages.

| **C** | **Accuracy** | **Kappa** |
|---|---|---|
| 4.115226e-03 | 87,7% | 86,6% |
| 1.234568e-02 | 94,5% | 94,0% |
| 3.703704e-0 | 95,9% | 95,5% |
| 1.111111e-01 | 96,6% | 96,3% |
| 3.333333e-01 | 96,8% | 96,5% |
| 1.000000e+00 | 96,8% | 96,5% |
| 3.000000e+00 | 96,8% | 96,5% |
| 9.000000e+00 | 96,8% | 96,5% |
| 2.700000e+01 | 96,8% | 96,5% |
| 8.100000e+01 | 96,8% | 96,5% |
| 2.430000e+02 | 96,8% | 96,5% |

Table 7.8: Comparison of different C Parameters' results in SVM Linear

As it can be observed from the table above that C parameters optimized the model's accuracy to 96,8%. Therefore, the final value used for SVM Linear model was C = 0.3333333 or 3.333333e-01.

With analyzing the confusion matrices of each appliance (see Figure 7.11 for the confusion matrix), it was found that SVM Linear resulted with the best results with less misclassification and more accurate classification for other appliances SNR values; except for the baseline as it still misclassified it with the oven, toaster oven and the stove as the figure 7.11 shows. Furthermore, the model couldn't classify the baseline SNR values in the test dataset. It was

further assumed that it was due to the baseline period underrepresentation as mentioned in table 7.1.

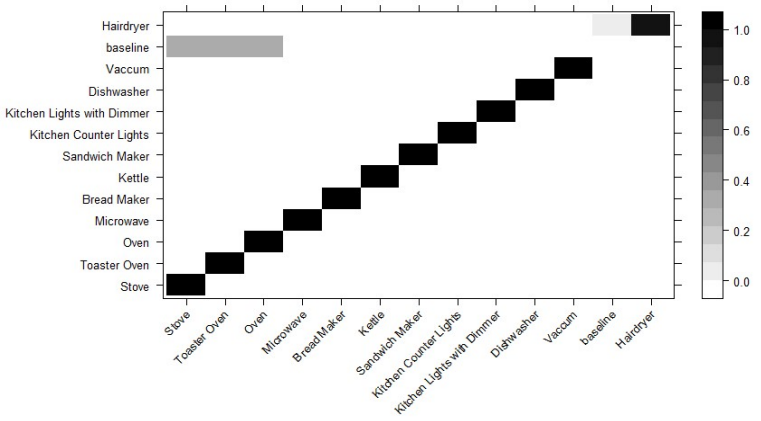



Figure 7.11: Visual confusion matrix highlighting SVM Linear misclassification

From the given confusion matrix of test dataset results, it was observed that SVM linear classifier also correctly identified different appliances' SNR values with better accuracy that reached 98,3%. The following table shows appliances different classes' accuracy, precision, specificity, precision and recall percentages.

| **Class** | **Specificity** | **Precision** | **Recall** | **Accuracy** |
|---|---|---|---|---|
| Stove | 0.995 | 0.956 | 1.000 | 0.997 |
| Toaster Oven | 0.995 | 0.956 | 1.000 | 0.997 |
| Oven | 0.995 | 0.956 | 1.000 | 0.997 |
| Microwave | 1.000 | 1.000 | 1.000 | 1.000 |
| Bread Maker | 1.000 | 1.000 | 1.000 | 1.000 |
| Kettle | 1.000 | 1.000 | 1.000 | 1.000 |
| Sandwich Maker | 1.000 | 1.000 | 1.000 | 1.000 |
| Kitchen Counter Lights | 1.000 | 1.000 | 1.000 | 1.000 |
| Kitchen Lights with Dimmer | 1.000 | 1.000 | 1.000 | 1.000 |
| Dishwasher | 1.000 | 1.000 | 1.000 | 1.000 |
| Vacuum | 1.000 | 1.000 | 1.000 | 1.000 |
| baseline | 0.995 | 0.000 | 0.000 | 0.497 |
| Hairdryer | 1.000 | 1.000 | 0.952 | 0.976 |

Table 7.9: Comparison of different classes results in SVM Linear

As the table shows, the True Negative Rate (specificity) for the actual rate average is 99%. The Positive predictive value (Precision) reaches an average of 98% excluding the precision rate for the Baseline which reached 0,0%. The True positive rate (Recall) for the actual classes reaches 98% in average as the table above shows again with excluding Baseline recall percentage. The reason for the zero percentage for the Baseline is again due to period underrepresentation when we processed it in *MATLAB*.

### Radial SVM

The dataset is trained with SVM Radial, and noticeably SVM Radial has been the most expensive in computation comparing to all the other used ML models as it will be shown in table 7.16. Its results also were not satisfying as it classified 4 different classes among all other appliances given in table 7.1. The reason again due to the C parameter and Sigma parameter optimization as they were set heuristically.

| **C** | **Sigma** | **Accuracy** | **Kappa** |
|---|---|---|---|
| 4.115226e-03 | 4.115226e-03 | 10,9% | 2,4% |
| 1.234568e-02 | 2.700000e+01 | 10,9% | 2,4% |
| 3.703704e-02 | 8.100000e+01 | 10,9% | 2,4% |
| 3.703704e-02 | 2.430000e+02 | 10,9% | 2,4% |
| 1.000000e+00 | 4.115226e-03 | 15,6% | 7,7% |
| 1.000000e+00 | 2.430000e+02 | 15,6% | 7,7% |
| 9.000000e+00 | 3.703704e-02 | 15,7% | 7,8% |
| 8.100000e+01 | 2.700000e+01 | 15,7% | 7,8% |

Table 7.10: Comparison of different C and Sigma Parameters' results in SVM Radial

As the table above shows, both C and sigma parameters reached their optimal model accuracy with only 15,7%. The final value used for SVM Radial model were sigma = 243 or 2.430000e+02 and C = 3 or 3.703704e-02. Although it could have decided for a better sigma and C accuracies, but it stopped at the local minimum kappa and the reason for this could be the multidimensionality that the data contained. In addition, the processing and memory capabilities, given the hardware in this thesis, could not further deal with the huge dataset and the multidimensionality that SVM Radial creates. With analyzing the confusion matrices of each of our 4 appliances (see Figure 7.12 for the confusion matrix), it was found that SVM Radial resulted with the worst results with even more misclassification and error rate for other appliances SNR values.

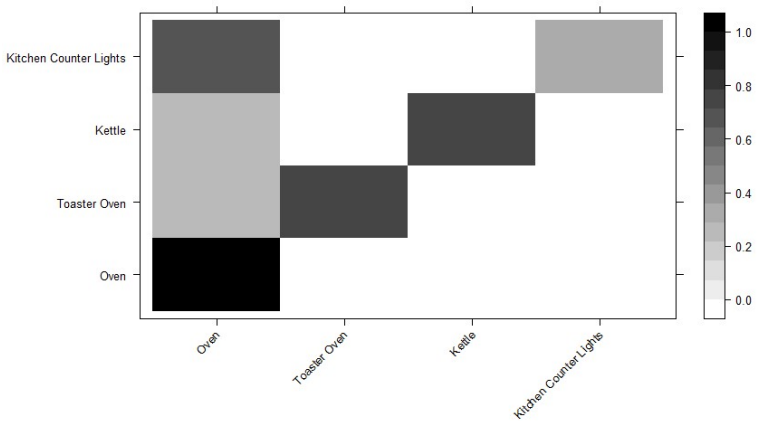



Figure 7.12: Visual confusion matrix highlighting SVM Radial misclassification

From the given confusion matrix of test dataset results, it is observed that SVM Radial classifier also incorrectly identified different appliances' SNR values with worse overall accuracy that reaches 70%. The following table shows different classes' accuracy, precision, specificity, precision and recall percentages.

| **Class** | **Specificity** | **Precision** | **Recall** | **Accuracy** |
|---|---|---|---|---|
| Oven | 0.606 | 0.458 | 1.000 | 0.803 |
| Toaster Oven | 1.000 | 1.000 | 0.727 | 0.863 |
| Kettle | 1.000 | 1.000 | 0.772 | 0.886 |
| Kitchen Counter Lights | 1.000 | 1.000 | 0.318 | 0.659 |

Table 7.11: Comparison of different classes results in SVM Radial

### 7.2.4 Artificial Neural Network (ANN)

The dataset is trained with ANN, and it is found that ANN has been also the best along with SVM linear SVM. According to the final results, ANN used only 5 hidden layers although 10 and 20 more layers were set. ANN stopped after 5 due to the huge dataset and the complexity of the dataset. The following table illustrates the decay size used; in addition to the hidden layer sizes and their corresponding accuracies and kappa.

| decay | size | Accuracy | Kappa |
|---|---|---|---|
| 1e-07 | 3 | 23,1% | 15,8% |
| 1e-07 | 5 | 32,1% | 25,6% |
| 1e-05 | 3 | 25,5% | 18,4% |
| 1e-05 | 5 | 34,1% | 27,8% |
| 1e-03 | 3 | 38,3% | 32,3% |
| 1e-03 | 5 | 46,9% | 41,8% |
| 1e-02 | 3 | 53,0% | 48,5% |
| 1e-02 | 5 | 64,6% | 61,1% |
| 1e-01 | 3 | 76,6% | 74,3% |
| 1e-01 | 5 | 85,3% | 83,9% |
| 5e-01 | 3 | 75,0% | 72,5% |
| 5e-01 | 5 | 87,1% | 85,9% |

Table 7.12: Comparison of different weights decay and Hidden layer sizes Parameters' results in ANN

As the table 7.12 shows, it is observable that the more hidden layers the network has, the more accurate the model becomes. Furthermore, the range of weight decays updated the weights to best classify our dataset with better accuracies and kappa percentages. Kappa is used to select the optimal model using the largest value from table 7.12, and the final values used for ANN model were size = 5 and decay = 0.5 or 5e-01. The following plot illustrates the weights decay updates.

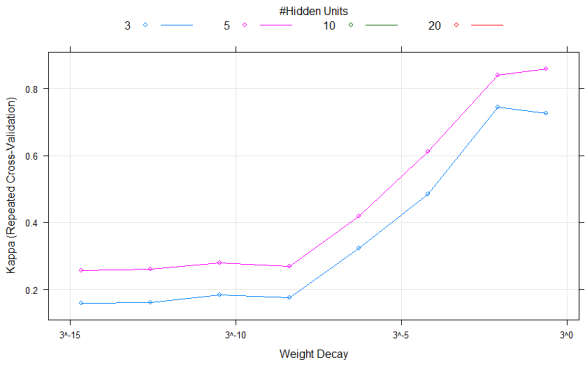



Figure 7.13: ANN Weights decay and number of used hidden layers

With analyzing the confusion matrices of each appliance (see Figure 7.15 for the confusion matrix), it was found that ANN gives the best results with less misclassification and more accurate classification for other appliances SNR values.

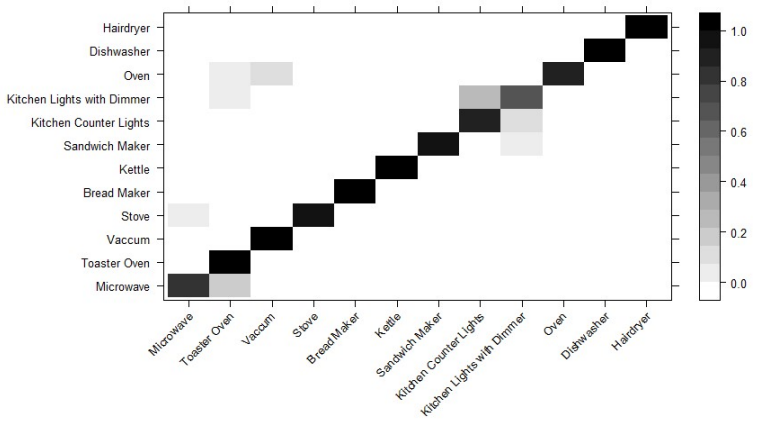



Figure 7.14: Visual confusion matrix highlighting ANN misclassification

From the given confusion matrix of test dataset results, it was observed that ANN classifier correctly identifies different appliances' SNR values. However, it also fails to classify other appliances SNR with low error misclassification rate. The following table shows different classes' accuracy, precision, specificity, precision and recall percentages.

| **Class** | **Specificity** | **Precision** | **Recall** | **Accuracy** |
|---|---|---|---|---|
| Microwave | 0.995 | 0.944 | 0.809 | 0.902 |
| Toaster Oven | 0.972 | 0.785 | 1.000 | 0.986 |
| Vaccum | 0.990 | 0.913 | 1.000 | 0.995 |
| Stove | 1.000 | 1.000 | 0.954 | 0.977 |
| Bread Maker | 1.000 | 1.000 | 1.000 | 1.000 |
| Kettle | 1.000 | 1.000 | 1.000 | 1.000 |
| Sandwich Maker | 1.000 | 1.000 | 0.954 | 0.977 |
| Kitchen Counter Lights | 0.972 | 0.760 | 0.863 | 0.917 |
| Kitchen Lights with Dimmer | 0.981 | 0.789 | 0.681 | 0.831 |
| Oven | 1.000 | 1.000 | 0.863 | 0.931 |
| Dishwasher | 1.000 | 1.000 | 1.000 | 1.000 |
| Hairdryer | 1.000 | 1.000 | 1.000 | 1.000 |

Table 7.13: Comparison of different classes results in ANN

## 7.3 Models' Results Comparison

It is essential to compare the performance of the used multiple different machine learning algorithms to decide what is best describing this dataset. Each model identifies each SNR signatures associated with each electrical appliances. The models also proved to identify the new appliances' SNR signatures by comparing the appliances' SNR signatures transmitted over the powerline with the recorded appliances' SNR signatures that were trained.

It is observed that each model has a different performance characteristics. Starting from training and ending with testing, each algorithm performs differently. By using the cross-validation re-sampling method, an estimation was given of how accurate each model could perform on unseen data (tesing dataset in this case). It was essential to use these estimations to select one or two best models from the suite of models used in this master thesis. A number of different methods are used in this section, which will evaluate the estimated accuracies of our machine learning algorithms to choose the one or two best models.

The first way to compare the results is to create a table where the minimum, mean and maximum accuracy or kappa properties of the distribution of our model accuracies are given. The following tables show all the required parameters to evaluate the models performances.

| **model** | **Min.** | **Mean** | **Max** |
|---|---|---|---|
| KNN | 0.881 | 0.955 | 1.000 |
| LDA | 0.805 | 0.904 | 1.000 |
| LDA2 | 0.791 | 0.905 | 0.986 |
| SVM Linear | 0.905 | 0.968 | 1.000 |
| SVM Radial | 0.077 | 0.157 | 0.310 |
| ANN | 0.432 | 0.871 | 0.986 |

Table 7.14: Results comparison of different models accuracies

| **model** | **Min.** | **Mean** | **Max** |
|---|---|---|---|
| KNN | 0.870 | 0.9508 | 1.000 |
| LDA | 0.787 | 0.895 | 1.000 |
| LDA2 | 0.772 | 0.897 | 0.985 |
| SVM Linear | 0.896 | 0.965 | 1.000 |
| SVM Radial | 0.000 | 0.0782 | 0.238 |
| ANN | 0.377 | 0.859 | 0.985 |

Table 7.15: Results comparison of different models Kappa

By looking at the tables, it can be observed that SVM Linear followed by KNN are considered the best performing models according to our classification problem. These models demonstrate a high level of classification for each appliance SNR. They also demonstrate the results with less misclassification errors, which conclude the effectiveness of these algorithms to detect appliances' noise in powerline communication modulation schemes. The following Box-plot shows the spread of the accuracy scores across each test classification for each algorithm.

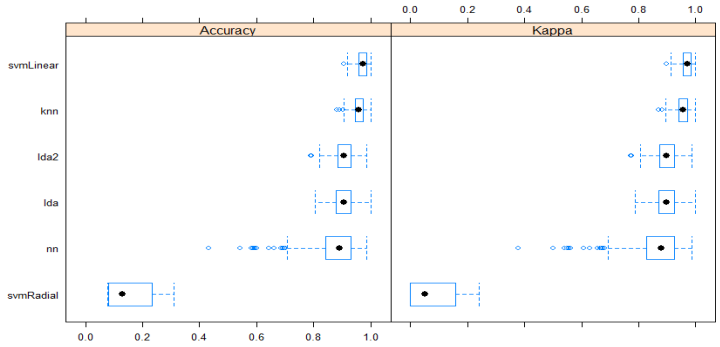



Figure 7.15: Boxplot Comparison of each Machine Learning Algorithms

The computational complexity of each model based on the training is presented in our table 7.16. As the table shows, $n$ is considered to be the number of training samples, $p$ is the number of SNR features which are 4096 every second in this case, $p_d$ is the number of dimensions used in LDA2, $n_s c$ is the number of support vectors, and finally $nl_i$ is the number of neurons at layer $i$ in ANN. The following run-time complexity approximations are given.

| **model** | **training in seconds** | **Training Complexity** | **Prediction Complexity** |
|---|---|---|---|
| KNN | 105.08 | $O(np)$ | $O(np)$ |
| LDA | 36.38 | $O(p)$ | $O(p)$ |
| LDA2 | 41.63 | $O(p_d)$ | $O(p_d)$ |
| SVM Linear | 440.25 | $O(n_c p + n_c)$ | $O(n_c p)$ |
| SVM Radial | 37174.61 | $O(n_{sc}^2 p + n^3)$ | $O(n_{sc}^2 p)$ |
| ANN | 2953.45 | $O(pnl_1 + nl_1 nl_2 + ...)$ | $O(pnl_i)$ |

Table 7.16: Comparison of each model running complexity

From the table above, it can be concluded that KNN, LDA and LDA2 have the cheapest running complexity. However, SVM Radial model has the most expensive running-time complexity as it takes roughly 10 hours to classify this classification problem, yet with the least accuracy. From these results, they

suggest that both LDA and LDA2 are the cheapest performing models to such kind of classification problems; however, their accuracies are not as efficient as KNN and SVM Linear. Therefore, it can be concluded that the best performing running-time Machine Learning model is LDA/2 and followed by KNN. However, the best performing ML model in terms of accuracy is SVM Linear and followed by KNN.

# Chapter 8

# Conclusion

We have presented a significant new findings in appliances' SNR anomaly detection for the home electrical powerline communication events. This thesis leverages the trend towards more efficient powerline communication for customers using their electrical powerline infrastructure as a communication medium. The types of noise generated by these appliances could allow powerline modems to isolate and simultaneously classify the occurrences of these electrical events while communication. The results validate the effectiveness of this thesis approach which could provide a basis for future analyses and advancement. The new results show also the significant difference comparing to the previous related work findings and results. The techniques used in this thesis approach can provide homeowners with a better powerline communication medium, provide them with information of which appliances are in operation, and further a full conduct of home automation purposes with lower costs. .The proposed techniques of appliances recognition can be also utilized by powerline modem manufacturers to improve their communication modems to adapt to the types of residual noise in the powerline.

In this thesis, both of Detecting actuation of electrical devices using electrical noise over a power line, and ElectriSense data acquisition methods were used to acquire this thesis dataset. However, this thesis used approaches in processing the data and classification which serve the scope of this master thesis. Five different Machine Learning models were used to train and test each appliance SNR based on their High Frequency noise behavior. The models are able to recognize new appliances' SNR values with the highest accuracies that reach 96%. The models time complexity to predict on new dataset is linear depending on the amount of appliances and their features. However, training the models on a new house may take a quadratic time with some models due to their complexity to classify patterns.

# Bibliography


[1] K. H. Zuberi, "Powerline carrier (plc) communication systems," *Stockholm, Royal Institute of Technolog*, 2003.

[2] S. Kasthala and P. V. GKD, "Evaluation of channel modeling techniques for indoor power line communication," in *Progress in Advanced Computing and Intelligent Engineering*, pp. 577–587, Springer, 2018.

[3] E. M. Tapia, S. S. Intille, and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," in *International conference on pervasive computing*, pp. 158–175, Springer, 2004.

[4] C. Beckmann, S. Consolvo, and A. LaMarca, "Some assembly required: Supporting end-user sensor installation in domestic ubiquitous computing environments," in *International Conference on Ubiquitous Computing*, pp. 107–124, Springer, 2004.

[5] T. Hirsch, J. Forlizzi, E. Hyder, J. Goetz, C. Kurtz, and J. Stroback, "The elder project: social, emotional, and environmental factors in the design of eldercare technologies," in *Proceedings on the 2000 conference on Universal Usability*, pp. 72–79, ACM, 2000.

[6] S. N. Patel, K. N. Truong, and G. D. Abowd, "Powerline positioning: A practical sub-room-level indoor location system for domestic use," in *International Conference on Ubiquitous Computing*, pp. 441–458, Springer, 2006.

[7] E. M. Tapia, S. S. Intille, L. Lopez, and K. Larson, "The design of a portable kit of wireless sensors for naturalistic data collection," in *International Conference on Pervasive Computing*, pp. 117–134, Springer, 2006.

[8] D. H. Wilson and C. Atkeson, "Simultaneous tracking and activity recognition (star) using many anonymous, binary sensors," in *International Conference on Pervasive Computing*, pp. 62–79, Springer, 2005.

[9] S. N. Patel, T. M. Robertson, G. D. Abowd, and M. S. Reynolds, "Detecting actuation of electrical devices using electrical noise over a power line," Jan. 10 2012. US Patent 8,094,034.

[10] M. Gashler, "Waffles: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2383–2387, 2011.

[11] R. Redl, "Power electronics and electromagnetic compatibility," in *Power Electronics Specialists Conference, 1996. PESC'96 Record., 27th Annual IEEE*, vol. 1, pp. 15–21, IEEE, 1996.

[12] G. Cohn, S. Gupta, J. Froehlich, E. Larson, and S. N. Patel, "Gassense: Appliance-level, single-point sensing of gas activity in the home," in *International Conference on Pervasive Computing*, pp. 265–282, Springer, 2010.

[13] J. E. Froehlich, E. Larson, T. Campbell, C. Haggerty, J. Fogarty, and S. N. Patel, "Hydrosense: infrastructure-mediated single-point sensing of whole-home water activity," in *Proceedings of the 11th international conference on Ubiquitous computing*, pp. 235–244, ACM, 2009.

[14] S. N. Patel, M. S. Reynolds, and G. D. Abowd, "Detecting human movement by differential air pressure sensing in hvac system ductwork: An exploration in infrastructure mediated sensing," in *International Conference on Pervasive Computing*, pp. 1–18, Springer, 2008.

[15] S. Gupta, M. S. Reynolds, and S. N. Patel, "Electrisense: single-point sensing using emi for electrical event detection and classification in the home," in *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pp. 139–148, ACM, 2010.

[16] S. N. Patel, T. Robertson, J. A. Kientz, M. S. Reynolds, and G. D. Abowd, "At the flick of a switch: Detecting and classifying unique electrical events on the residential power line (nominated for the best paper award)," in *International Conference on Ubiquitous Computing*, pp. 271–288, Springer, 2007.

[17] K. Nakamura, S. Yoshimura, and T. Usami, "Drive apparatus for pwm control of two inductive loads with reduced generation of electrical noise," May 10 2005. US Patent 6,891,342.

[18] H. P. Longerich, S. E. Jackson, and D. Günther, "Inter-laboratory note. laser ablation inductively coupled plasma mass spectrometric transient signal data acquisition and analyte concentration calculation," *Journal of Analytical Atomic Spectrometry*, vol. 11, no. 9, pp. 899–904, 1996.

[19] C. F. Stevens, "Inferences about membrane properties from electrical noise measurements," *Biophysical journal*, vol. 12, no. 8, p. 1028, 1972.

[20] S. Najjar, F. Rouissi, H. Gassara, and A. Ghazel, "Comparative study of modulation techniques performances for communication over narrowband powerline channel characteristics," in *Multimedia Computing and Systems (ICMCS), 2016 5th International Conference on*, pp. 788–792, IEEE, 2016.

[21] A. Llano, I. Angulo, P. Angueira, T. Arzuaga, and D. de la Vega, "Analysis of the channel influence to power line communications based on itu-t g. 9904 (prime)," *Energies*, vol. 9, no. 1, p. 39, 2016.

[22] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[23] E. Alpaydin, *Introduction to machine learning*. MIT press, 2009.

[24] J. M. Zurada, *Introduction to artificial neural systems*, vol. 8. West publishing company St. Paul, 1992.

[25] R. J. Schalkoff, *Artificial neural networks*, vol. 1. McGraw-Hill New York, 1997.

[26] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE transactions on systems, man, and cybernetics*, no. 4, pp. 580–585, 1985.

[27] Stanford, "K-nearest neighbors demo."

[28] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[29] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 4, pp. 325–327, 1976.

[30] scikit learn, "Nearest neighbors."

[31] M. Kuhn *et al.*, "Caret package," *Journal of statistical software*, vol. 28, no. 5, pp. 1–26, 2008.

[32] M. Li and B. Yuan, "2d-lda: A statistical linear discriminant analysis for image matrix," *Pattern Recognition Letters*, vol. 26, no. 5, pp. 527–532, 2005.

[33] J. Ye, R. Janardan, and Q. Li, "Two-dimensional linear discriminant analysis," in *Advances in neural information processing systems*, pp. 1569–1576, 2005.

[34] G. Martos, "Discriminant analysis in r."

[35] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[36] N. Cristianini, J. Shawe-Taylor, *et al.*, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[37] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[38] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[39] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[40] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[41] Python, "Index of /python/scikit-learn/images/svm."

[42] R. Lippmann, "An introduction to computing with neural nets," *IEEE Assp magazine*, vol. 4, no. 2, pp. 4–22, 1987.

[43] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[44] S. Agatonovic-Kustrin and R. Beresford, "Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research," *Journal of pharmaceutical and biomedical analysis*, vol. 22, no. 5, pp. 717–727, 2000.

[45] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[46] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *IEEE Potentials*, vol. 13, no. 4, pp. 27–31, 1994.

[47] A. T. Goh, "Back-propagation neural networks for modeling complex systems," *Artificial Intelligence in Engineering*, vol. 9, no. 3, pp. 143–151, 1995.

[48] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[49] K. Razazian, M. Umari, A. Kamalizad, V. Loginov, and M. Navid, "G3-plc specification for powerline communication: Overview, system simulation and field trial results," in *Power Line Communications and Its Applications (ISPLC), 2010 IEEE International Symposium on*, pp. 313–318, IEEE, 2010.

[50] Y. Ma, P. So, and E. Gunawan, "Performance analysis of ofdm systems for broadband power line communications under impulsive noise and multipath effects," *IEEE Transactions on Power Delivery*, vol. 20, no. 2, pp. 674–682, 2005.

[51] A. Majumder *et al.*, "Power line communications," *IEEE potentials*, vol. 23, no. 4, pp. 4–8, 2004.

[52] E. Biglieri, "Coding and modulation for a horrible channel," *IEEE Communications Magazine*, vol. 41, no. 5, pp. 92–98, 2003.

[53] Y.-F. Chen and T.-D. Chiueh, "Baseband transceiver design of a 128-kbps power-line modem for household applications," *IEEE Transactions on Power Delivery*, vol. 17, no. 2, pp. 338–344, 2002.

[54] B. Shrestha, "Power line carrier data transmission systems having signal conditioning for the carrier data signal," Feb. 9 1999. US Patent 5,870,016.

[55] C. Wu, K. Chau, and Y. Li, "Predicting monthly streamflow using data-driven models coupled with data-preprocessing techniques," *Water Resources Research*, vol. 45, no. 8, 2009.

[56] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.

[57] M. Birattari and M. Dorigo, "The problem of tuning metaheuristics as seen from a machine learning perspective," 2004.

[58] J. M. Bernardo and A. F. Smith, "Bayesian theory," 2001.

[59] H. Trevor, T. Robert, and F. JH, "The elements of statistical learning: data mining, inference, and prediction," 2009.

[60] G. H. Golub, M. Heath, and G. Wahba, "Generalized cross-validation as a method for choosing a good ridge parameter," *Technometrics*, vol. 21, no. 2, pp. 215–223, 1979.

[61] A. M. Molinaro, R. Simon, and R. M. Pfeiffer, "Prediction error estimation: a comparison of resampling methods," *Bioinformatics*, vol. 21, no. 15, pp. 3301–3307, 2005.

[62] GNOME, "Cross-validation explained."

[63] GNOME, "Distance metric learning for large margin nearest neighbor classifications."

[64] S. Balakrishnama and A. Ganapathiraju, "Linear discriminant analysis-a brief tutorial," *Institute for Signal and information Processing*, vol. 18, pp. 1–8, 1998.

[65] C.-L. Huang and J.-F. Dun, "A distributed pso–svm hybrid system with feature selection and parameter optimization," *Applied soft computing*, vol. 8, no. 4, pp. 1381–1391, 2008.

[66] M. Kuhn, "A short introduction to the caret package," *R Found Stat Comput*, pp. 1–10, 2015.

[67] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple svm parameters," *Neurocomputing*, vol. 64, pp. 107–117, 2005.

[68] S. Grauer-Gray, L. Xu, R. Searles, S. Ayalasomayajula, and J. Cavazos, "Auto-tuning a high-level language targeted to gpu codes," in *Innovative Parallel Computing (InPar), 2012*, pp. 1–10, IEEE, 2012.

[69] C.-H. Wu, G.-H. Tzeng, Y.-J. Goo, and W.-C. Fang, "A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy," *Expert systems with applications*, vol. 32, no. 2, pp. 397–408, 2007.

[70] I. Lizarazo, "Svm-based segmentation and classification of remotely sensed data," *International Journal of Remote Sensing*, vol. 29, no. 24, pp. 7277–7283, 2008.

[71] F. Chauchard, R. Cogdill, S. Roussel, J. Roger, and V. Bellon-Maurel, "Application of ls-svm to non-linear phenomena in nir spectroscopy: development of a robust and portable sensor for acidity prediction in grapes," *Chemometrics and Intelligent Laboratory Systems*, vol. 71, no. 2, pp. 141–150, 2004.

[72] stack exchange, "the influence of c in svms with linear kernel."

[73] Standford, "Support vector machine in javascript."

[74] A. Cochocki and R. Unbehauen, *Neural networks for optimization and signal processing*. John Wiley & Sons, Inc., 1993.

[75] Standford, "Anns in javascript."

[76] D. School, "Simple guide to confusion matrix terminology."

[77] K. Markham, "Simple guide to confusion matrix terminology," *data school*, 2014.

[78] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[79] R. E. Kennedy, Z. Yang, and W. B. Cohen, "Detecting trends in forest disturbance and recovery using yearly landsat time series: 1. landtrendr—temporal segmentation algorithms," *Remote Sensing of Environment*, vol. 114, no. 12, pp. 2897–2910, 2010.

[80] J. W. Palmer, "Web site usability, design, and performance metrics," *Information systems research*, vol. 13, no. 2, pp. 151–167, 2002.

[81] S. Güzelgöz, H. B. Çelebi, T. Güzel, H. Arslan, and M. K. Mihçak, "Time frequency analysis of noise generated by electrical loads in plc," in *Telecommunications (ICT), 2010 IEEE 17th International Conference on*, pp. 864–871, IEEE, 2010.

[82] H. Meng, Y. L. Guan, and S. Chen, "Modeling and analysis of noise effects on broadband power-line communications," *IEEE Transactions on Power delivery*, vol. 20, no. 2, pp. 630–637, 2005.

[83] M. A. Hall, "Correlation-based feature selection of discrete and numeric class machine learning," 2000.