5/14/2021

CM3203 Individual Project

By Kyle Swire-Thompson

Supervisor: Neetesh Saxena

Moderator: Hantao Liu

kyle.swirethompson@gmail.com
CARDIFF UNIVERCITY

Acknowledgements

Firstly, I would like to express my profound gratitude to both of my parents to support me throughout my entire education and enable me to take on a project such as this.

Secondly, I would like to thank my supervisor for all of his hard work supporting this project and providing me with guidance on how to complete a task as monumental as this.

Finally, I would like to thank all my other friends and family that have supported me throughout my degree and encouraged my pursuit of computer science.

DECLARATION OF ORIGINALITY

I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree, except where states otherwise by reference or acknowledgement, the work presented is entirely my own.

X _____

PROFORMA

I. Candidate number.
   A. 1887995

II. The Title of the Project.
   A. Cyber-Events Detection in Smart Grid for Situational Awareness

III. The Examination and Year.
   A. 2021-CM3203

IV. Word count for the Dissertation.
   A. 22808

V. Final line count: Number of lines written by the **student** in the final version of their software.
   A. 1005

VI. Project Originator.
   A. Neetesh Saxena

VII. Project Supervisor.
   A. Neetesh Saxena

VIII. The original aims of the project.
   A. The project's original aims were to develop a cyber event detection capability within the context of a smart grid power system. This, in turn, would be used to improve the situational awareness of the system. This was done through the use Of machine learning algorithms in combination with data mining techniques. After analysis and event detection, a report on the said event is then produced detailing all relevant knowledge and guidance on how to proceed. The detection system must be fully customisable to detect better the attacks they encounter in the real world.

IX. At most 100 words summarising the work completed.
   A. The work completed here involves using and improving machine learning algorithm based classifiers to prevent attacks on smart grid power systems. The tool produced alongside this paper completes this task.

TABLE OF CONTENTS

1 CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

1.1.1 Introduction to the topic

With the ever-increasing digitalisation of the modern world, we are becoming more reliant on computer systems than ever. Whether that be financial transactions, communications or managing and controlling physical machines, computer systems tend to be at the centre of these. As these computer systems are critical to everyday tasks, any disruption can have disastrous knock-on effects for the other systems reliant on them. In recent years, cyber operations have been increasingly utilised to further political ambitions and globally affect the balance of power. This is typically called cyberterrorism and involves three basic types of attack, " As a rule, a distinction should be drawn among three basic attack categories: an attack on the gateway of an organisation, mainly its Internet sites, through direct attacks, denial of service, or the defacement of websites; an attack on an organisation's information systems; and finally, the most sophisticated (and complex) category—attacks on an organisation's core operational systems, for example, industrial control systems ". (Babak Akhgar, 2014). This paper will attempt to tackle the third and most sophisticated category of attack, industrial control systems.

Previously the most devastating attacks were only possible by a large organised group with a common goal, but cyber operations do not follow this rule. While terrorists can and do still make use of this technology in order to "reach their objectives, create damage, influence policy, and leverage the disproportional power relation between terrorists and the defending state." (Babak Akhgar, 2013). cyber operations are open to everyone. "Civilians acting alone or as part of a mass uprising can leverage widely available hacking tools and techniques to conduct cyber operations. “ (Vacca, 2017). These crimes cause mass disruption to the community and are estimated to cost $600 billion U.S. dollars annually or 0.8% of the global GDP (cyber security firm McAfee, the Center for Strategic and International Studies, 2020). Following this historic spike in cybercrime, a vast and diverse market for cybersecurity has emerged. This market is expected to reach $248.26 billion U.S. dollars by 2023 (Mlitz, 2018).

There are many critical elements to cybersecurity, but this paper will focus on detecting and minimising attacks on industrial control systems, specifically a power-based smart grid. In order to detect and minimise these breaches, this paper will use both data mining techniques and machine learning algorithms applied to data produced by the smart grids infrastructure. Such a task has four core elements: data mining and processing, machine learning classification, smart grid and finally situational awareness. The first and arguably most straightforward in this context is data mining. Data mining is the process of turning raw data into useful information. This topic encompasses the entire paper as the overall goal is to take unreadable data and produce useful information. A machine learning algorithm is used to accomplish this goal. A machine learning algorithm is a prediction method that uses automated analytical model building and is a branch of Artificial Intelligence. The use of machine learning algorithms is appropriate here as these algorithms specialise in turning an overwhelming amount of data with uncomprehending complexity for the human mind and finding patterns and correlations that otherwise would have been missed. The models that these algorithms build based on the patterns and correlations can then be used to make predictions on unknown data.

Attacks do not occur by themselves and must happen on a targeted computer system. A smart grid was chosen as the target computer system, specifically a power system smart grid. A smart grid is an electrical network that enables the two-way flow of both data and electricity. It also makes use of digital communications technology to better detect and react to any changes that may happen within the smart grid. Smart grids also typically have self-healing capabilities and integrate innovative tools and technologies to improve the grids' automation and efficiency. A power systems smart grid aims to deliver power safely and securely to a paying customer. Detection of attacks will improve the situational awareness of the smart grid. Situational awareness is the understanding that a person of authority has of the current state of the computer system they oversee. The greater the situational awareness, the greater likelihood that the effects of an attack are minimised. Smart grid situational awareness is provided through a "monitoring system composed of advanced monitoring equipment, which can be more intuitive and predictable in the control of power grid." (Dong, et al., 2017).

Attacks inflict damage to their target in many different methods. For instance, a well-researched type of attack used on electric supply systems is the Aurora attack. According to mark Zeller at the DistribuTECH Conference in San Diego, California, an Aurora vulnerability attack is "to intentionally open a breaker and close it out of synchronism to cause damage to connected power system equipment, such as generators, motors, and transformers. When an out-of-synchronism close is initiated, the resulting high electrical current and torque translate to stress on the mechanical shaft of rotating equipment. This stress reduces the life of the rotating equipment and can possibly destroy it." (Zeller, 2011).

The first-ever know example of a cyber warfare tool differs from the context slightly as it occurred against a nuclear power plant instead of a power grid smart system. Never the less the theory behind the attack is the same, and it is far more dramatic with the stakes of a nuclear meltdown. This example is called Stuxnet, a malicious computer first discovered in 2010 but thought to have been developed in the mid-2000s. It is widely accepted that it was developed by the intelligent agencies of both the United States and Israel under the code name operation Olympic Games. Stuxnet Reportedly has destroyed numerous central fuses within the Iranian uranium enrichment facility. This damage was done by exploiting multiple previously unknown windows 0-day vulnerabilities to infect other computers on the system and spread. The centrifuges themselves were destroyed by the worm, slowly but surely changing the centrifuge values to the point where it was unstable and caused burnout, thus reducing the country's nuclear capabilities. Since being discovered, it faded from the popular consciousness until around 2016 when a Microsoft security Intelligence Report identified it among the exploit related malware families detected in the second half of 2015. since then, other groups have reportedly modified it to target other facilities, including water treatment plants, power plants, and gas facilities. Stuxnet's specific goals differed from previous malicious software as, " Stuxnet wasn't about industrial espionage: it didn't steal, manipulate, or erase information. Rather, Stuxnet's goal was to physically destroy a military target-not just metaphorically, but literally." (Langner, 2011).

This project aims to produce two things. A dissertation of in-depth research which will demonstrate a thorough investigation of the topic and a machine learning tool that can be applied to a computer system which will take inputted data and process said data in order to create a report on the current situation facing the smart grid and overall, will improve the situational awareness of the smart grid.

1.1.2 Motivation

There are many types of applications where this project would be helpful. A network security project might need to test the effectiveness and hiding capabilities of certain types of attacks against custom multi-algorithm machine learning detection methods. Both the tool and accompanying paper would help a researcher create a prediction methodology using multiple algorithms and test the attacks. The flexibility of the application is paramount to quickly change in and change out algorithms to produce the most effective prediction methodology. This project will also easily find the advantages and disadvantages of each machine learning algorithm against each of the types of attacks inflicted on the smart grid. Finding out these advantages and disadvantages will allow the researcher to place algorithms that complement each other together. There are many methods to combining these predictions, all of which will improve the single classifiers based prediction. The increased level of accuracy offered in this project will help to inform any users next step when an event has occurred.

1.2 Scope and Context

To understand why this work is critical, we must first understand the context surrounding smart grids, cyberattacks and machine learning algorithms. Doing this will give us a greater understanding of any problems or issues we may face later on in the project. However, these subjects are extensive, meaning that any attempt to understand the entire subject requires much study and would be far too cumbersome to produce a credible paper and tool. We overcome this by defining a scope that sets limits for the extent that the project can reach. If the scope were too broad, then the work would be far too great to finish, but if it were too small, not enough work would be covered to make this paper relevant, so a correct balance has to be found between the two.

1.2.1 The scope

Smart grids are typically very interlinked, and as such, one action occurring on one end of the smart grid may affect a component on the other end. This dependency results in a need to correctly understand the layout of our specific smart grid and how each component interacts with each other. This is especially important for understanding what part of the smart grid an attacker is attempting to infiltrate or affect and how changes in the data affect the real-life smart grid. While we do not need to understand every aspect of the smart grid, we need to understand how the data we are processing is gathered and transferred.

Many different cyberattacks can be inflicted upon a smart grid, but we will use attacks that have most notably been used against smart grids for this project. These attacks include remote trip attacks, command injections attacks, relay setting change attacks and data injection attacks. No other types of attack will be considered, although multiple subtypes are belonging to each different attack category.

Finally, we come to machine learning algorithms. This is the most prominent topic covered in this paper as the whole detection method is based on using multiple machine learning algorithms. As such, we need to have a thorough understanding of each algorithm's advantages and disadvantages, including how each algorithm interacts when used in a voting methodology for a final prediction. In order to create an accurate prediction method to inform the tool, we must understand how each aspect and type of machine learning algorithm works.

1.2.2 Context of the problem

Previously these types of attacks would only occur against large infrastructure projects such as nuclear power plants or other critical high-value targets. However, while these were the first to be attacked, they were also the first to adopt cutting edge cybersecurity tools and techniques. Even though the rewards for successfully attacking these targets will be high, the difficulty is ever increasing. Unfortunately, this has led to hackers typically targeting smaller projects with substantially less cybersecurity or targeting countries that do not have the same cybersecurity capabilities as other more wealthy countries.

Attacks that physically damage the smart grid can be devastating to the grids' correct functionality and significantly impact any infrastructure or people relying on the smart grid. It Is challenging to detect when these attacks are happening as hackers have only become better and better at hiding themselves on computer systems from typical antivirus or anti-ransomware software. A human looking at the voltage data which is reported from a smart grid will not be able to tell if an attack is happening even if they know the correct parameters for what the voltage is meant to be, the amount of data they will be receiving and the number of voltages they will have to monitor will be far too cumbersome and overwhelming. Hackers may also decide to camouflage how quickly they increase the voltage by spreading this over a long period such as weeks or even months until the elements are overused till destruction. Disruption to essential services such as water, electricity and natural gas can have detrimental effects on the nation's stability. A lack of these essential services erodes trust in the populist that the government can provide for its citizens.

Attacks do not necessarily have to physically damage the smart grid in order to cause a significant impact. Suppose an attack was interfering with the signals that the smart grid passes onto the control room, which will result in energy being diverted to areas that do not need resulting in vast amounts of money being wasted. Thus is devastating both to the power company as well as the environment as unnecessary pollution occurs.

1.3 PROBLEM STATEMENT

In the early days of smart grids, they would typically be protected due to their isolation from any other network. This means that there was no risk of being attacked since there was no way for anyone from outside of the grid to gain access. This model for a smart grid, however, is not possible in the modern day. Nowadays, smart grids real-time sensing and measuring technologies enable the system to automatically detect and respond to any changes or problems within the grid, thus reducing any outrages and maintenance requirements. Smart grids like these are essential to prevent outages by correctly implementing various automation tools and techniques that require less human input or maintenance.

These smart grids are always critical to whoever relies on them and, as such, are heavily protected with cybersecurity. However, these cybersecurity tools rely on finding compromise indicators, whether that be packet analysis, system log entries/files, etc. If an attacker can hide from the tool's methods to detect an intruder, no action will be taken against the intruder due to these tools' automated nature. Even if the attacker starts to damage parts of the smart grid, these cybersecurity tools will still not detect them as they cannot see what effect the intruder is having on the smart grid. This is the knowledge gap this paper and tool will be able to fill. Instead of relying on pieces of forensic data found on the smart grid system, it will instead use the data gathered directly from the smart grid.

Therefore our problem statement is to increase the situational awareness of a smart grid by monitoring the physical changes on the smart grid for any indicators of compromise or potentially dangerous situations while providing a report on any incident that occurs. This report will contain guidance on the correct path going forward to take. Previous literature has attempted to solve this problem but not to any degree of accuracy that would make it relevant to a real-world context

1.4 AIMS AND OBJECTIVES

To have a clear idea of what we want to accomplish, we first must set an aim and a set of objectives that we need to complete by the end of this project.

1.4.1 The Aim

Many large-scale smart grid systems will typically have a specialised team of professionals on call to deal with incidents that may occur, but many medium or small smart grids rely on automated security tools to do this job for them. This means that they are vulnerable to any attack that these automated tools do not recognise and cannot stop. Automated cybersecurity tools rely on the attacker leaving behind or producing forensic data that can then be collected and processed to determine if an attack occurs. If an attacker can circumvent these automated tools, any smart grid that relies on these tools is therefore vulnerable. To overcome this, the project aims to create an easy to use tool that uses machine learning algorithms to detect a hidden attack even when the attacker can hide their presence from automated cybersecurity tools.

1.4.2 Objectives

The work that this paper will aim to cover will do the following objectives :

1. Collect giving data about the smart grid and use it to build and train multiple machine learning classifiers to a high degree of accuracy to perform predictions about the state of this smart grid from live data inputted by the user.
2. Provide the user with numerous other options for predictions based on the single classifiers' predictions and produce a report based on those predictions that details all relevant knowledge about the incident and guidance on how to proceed.
3. The tool with a degree of certainty always tells the difference between the three categories of events: no events, natural events and attack events. However, this level of precision is not enough, and the tool must also correctly classify each scenario found in the data correctly.

1.5 Building on previous work

Previous work with machine learning algorithms to make predictions about the state of a smart grid has focused primarily on being able to classify whether an attack is occurring correctly or not. This is an insufficient level of precision, and for a user wishing to make use of this work, it is not enough. This project will significantly improve the precision elements of the previous machine learning tools while using them to create a practical application that can improve an individuals' situational awareness of their smart grid.

1.6 Assumptions

The business or individuals that use the tool will be familiar with the topics covered, with only some need to explain technical concepts. The data will be inputted in .ARFF file type and will be of the same format regardless of the data set, although error checking for this will still occur within the tool. The tool will not require any security to prevent the attacker from manipulating the tool. The user has fully downloaded all appropriate libraries, and any required tools are up to date. Any test instant treated like live data has the same attribute information as the data set each classifier would is trained on.

1.7 Report Organisation

In the next chapter, there will be background material about the topics discussed in this section. This background knowledge is imperative to understand both the solution and the results produced after that chapter follows a detailed description of the approach and how this paper will overcome the problems described. After the approach will come a chapter presenting the solution idea and how it was implemented. A breakdown of all the functionality within the tool is also included. Next follows the results and evaluation chapter. This part of the paper will detail any results from experimentation done in conjunction with producing the tool and proving this paper has achieved the goals set out. Once the results have been presented, they will then be evaluated to provide critical context to the results and highlight any key elements or takeaways. The Final three research chapters are based around conclusions of this work and any future work that could make use of this project.

2 CHAPTER 2: BACKGROUND

2.1 OVERVIEW

In this chapter, both the terms and context needed to understand the project will be explained thoroughly. This includes any constraints that have used in order to simplify the scope of the project.

2.2 TERMS

Here will be defined some standard terms found in these topics.

2.2.1 Machine learning algorithm

A machine learning algorithm is an algorithm that turns a set of data into a model and uses this model to make predictions on new data. This prediction technique outperforms other statistical methods because of the non-linear and complexity of the data set we are using; machine learning algorithms will always surpass other algorithms in terms of accuracy metrics.

2.2.2 Smart-Grid

A smart grid is a power solution that employs an extensive range of information technology resources to detect and respond appropriately to any changes in local power usage. The use of a smart grid will reduce electricity waste and overall energy costs.

2.2.3 Situational Awareness

Situational awareness within this context is how well the user understands what is currently happening within their smart grid.

2.2.4 WEKA

Weka is a tried and tested open-source machine learning software that can be accessed through a graphical user interface, standard terminal applications, or a Java API. It is widely used for teaching, research, and industrial applications, contains a plethora of built-in tools for standard machine learning tasks, and additionally gives transparent access to well-known toolboxes such as scikit-learn, R, and Deeplearning4j. (Machine Learning Group at the University of Waikato., n.d.)

2.2.5 Voting Methodology

A voting methodology is a process that is applied to previous predictions made by single classifiers to make the overall prediction more accurate.

2.2.6 Industrial control systems

An industrial control system uses control systems and associated measurement and control instrumentation for industrial process control. These systems cover an extensive range, including electric power generation, chemical manufacturing, oil refineries, water and waste treatment. Specifically, in our context, the control system is a smart grid, and the process that it controls is the generation and distribution of power.

2.2.7 Artificial Intelligence

Artificial Intelligence is the theory and development of computer systems that can perform tasks typically off-limits to other computer systems due to human intelligence requirements. There are many examples of these, such as speech recognition, translation between languages, visual perception and decision-making.

2.2.8 Cross-validation

Cross-validation is a process we can use to evaluate how well a classifier performs. First, in this process, the number of folds is defined. In this example, we will use 10. Imagine there were instances of data on which you wish to perform ten-fold cross-validation. First, These instances would be divided into ten equal-sized sets. Within each of these ten sets, they are further divided down into two groups. 90% of each of equal size divisions is used for training, whereas 10% is used for testing. The classifier chosen to be tested is then trained on the training data and is tested on the test data for the first equal-sized data set. This process is repeated for all nine other chunks of data, which produces nine more classifiers. The performance from each of these ten classifiers is then averaged to produce a result.

2.2.9 Supervised data

Supervised data is data in which each instance has been labelled and belongs to a specific scenario. This label means that any algorithm which uses supervised data knows which scenario each of the instances within the training data belongs to. The opposite of this is unsupervised data which can look the same as supervised data but would be missing the last attribute called the class attribute, which defines the scenarios each of the instances belongs to.

2.2.10 Classifiers

A classifier is a machine learning algorithm that automatically orders or categorises supervised data in one or more sets of classes.

2.3 Related Work

This next section will detail previous work done in this field of research and demonstrate how it is related to this project.

The work that is arguably the most closely related to this topic is entitled machine learning power system and cyber-attack discrimination and classification of disturbances (Raymond C. Borges Hink & Mark A. Buckner, 2014). In this work, the authors theorised that the machine learning algorithms in the open-source Waikato environment for knowledge analysis (WEKA) could "leverage non-linear complex relationships between power system measurements and be able to discriminate between malicious, non-malicious, and normal disturbances" (Raymond C. Borges Hink, 2014). To test this, they used the same data set as the one used in this paper.

When testing this theory, they used ten-fold cross-validation and a 90/10 training test split. They then applied a series of machine learning algorithms to the data set, including OneR, NNGe, Random Forests, Naïve Bayes, Support Vector Machines (SVM), JRipper, and Adaboost+JRipper. In order to judge the accuracy of these algorithms, they recorded metrics including accuracy, recall, precision, and F measure. Overall the results concluded that JRipper combined with an aggregated boost, a single classifier enhancer method, had the highest accuracy across all metrics at approximately 95% correct classification (Raymond C. Borges Hink, 2014). However, upon closer inspection of the results, a problem arises.

An instance was classed as successfully classified if the classifier could differentiate between disturbances and attack. The machine-learning algorithm was incapable of classifying specific faults and attack types. While the research conducted here does explore the classification of the

entire data set and does attempt to categorise them, it only does this in the most basic form of classification and, for our purposes, is not precise enough. It does not explicitly address the classification of any individual scenarios, such as the specific subtypes of attacks found within the data set. This project aims to build on this and create a tool that will correctly classify the type of scenario as shown in previous literature and expand upon it to create a more precise prediction method.

A critical element to this paper is situational awareness in the context of a smart grid. The paper that best demonstrated this is Classification of Disturbances and Cyber-Attacks in Power Systems Using Heterogeneous Time-Synchronized Data as in the words of its author, it "proposes a sequential pattern mining approach to accurately extract patterns of power-system disturbances and cyber-attacks from heterogeneous time-synchronised data, including synchrophasor measurements, relay logs, and network event monitor logs." (Shengyi Pan, 2015). This paper has a strong focus on the topic of situational awareness. This paper branches off into using common path mining algorithms to discover common paths from labelled logs, which will not be covered here, but the common path mining paper focuses on similar concepts as what this paper is beneficial. The author's methodology for producing accurate results is still invaluable, and the paper's focus on situational awareness both provide invaluable knowledge.

Another article related to this work is an evaluation of machine learning methods to detect malicious SCADA communications. This article details a similar approach with many of the same machine learning algorithms except with the addition of J48. These algorithms are used to read from remote terminal units in another ICS environment, a gas pipeline system. (Beaver, et al., 2013). This article included routine operational observations and other observations of similar attacks to the remote command injection attack, including an illegal process ID attack. This attack is when a malicious command is sent to a programmable logic control unit in order to change its performance. This attack also included other types of command injection attacks like manipulating the setpoint of the pipeline pressure valve and a command injection attack that dealt primarily in manipulating outgoing commands within the system to phish for relevant information such as address and function scans. Despite this, the data set structures are not all that similar. With regular operation numbering at 28,086 and command injection attacks numbering at 257, only 49 are similar to the remote trip command injection utilised in our work. The work that we are using has 8737 instances of remote trip command injection attack and 4405 instances of normal operations, a vastly different ratio than in this work. Even though this difference impacts the comparison, the significant takeaways from this work are still relevant. This article reiterates the previous findings that both attack and routine operations only attempting to find the difference between these two categories have a higher classification rate than if a classifier attempts to classify individual scenarios. In this work, they identified the highest accuracy classifiers as the nearest neighbour algorithm, specifically NNGe and decision tree algorithms specifically the random forest algorithm.

Different algorithms will be able to classify different types of attack at different rates. This difference in ability is shown in the paper analysis of remote trip command injection attacks in industrial control systems through statistical and machine learning methods. It was concluded that "specific attacks are classified with higher accuracy through an application of differing ML algorithms and that in this case, RandomForest is a better classifier for remote tripping command injection attacks than JRipper+Adaboost." (Timm, 2018). We can conclude that multiple algorithms should be used to achieve this high classification level across different attacks types.

2.4 EXISTING TOOLS

Here will be a description of three popular existing tools and how they use machine learning algorithms. It is essential to understand what products are already developed and populate the commercial market to make informed decisions about the tool created alongside this paper.

2.4.1 Microsoft Defender for Endpoint

A very well known cybersecurity tool that uses machine learning is the Windows Defender advanced threat protection from Microsoft. “Microsoft Defender for Endpoint is a holistic, cloud-delivered endpoint security solution that includes risk-based vulnerability management and assessment, attack surface reduction, behavioural based and cloud-powered next-generation protection, endpoint detection and response (EDR), automatic investigation and remediation, and managed hunting services. ” (microsoft, 2021). This quote means that this application uses cloud-based AI and multiple clusters of machine learning algorithms to spot threats.

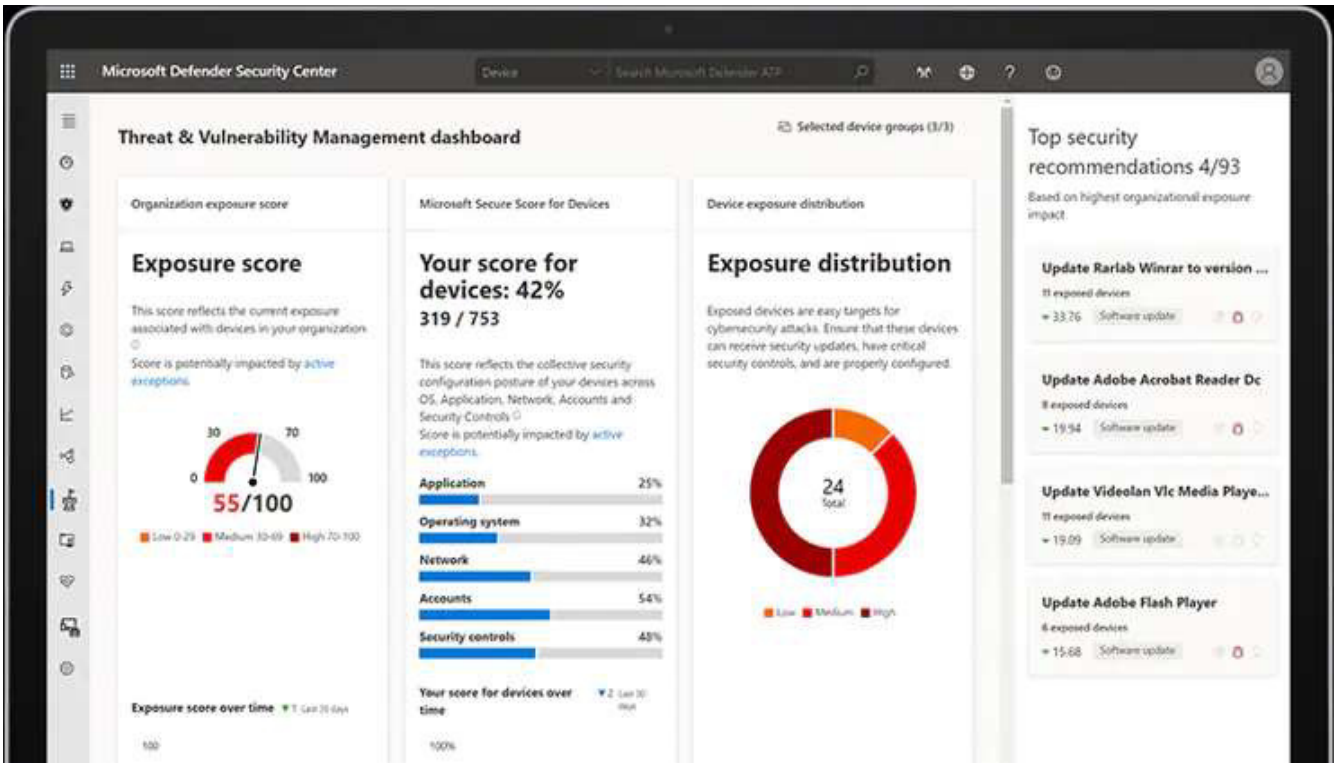


Figure 1. screenshot of Microsoft Defender for Endpoint

2.4.2 Chronicle

Chronicle is a cybersecurity company that branched off from Google's parent company Alphabet. Its first designed a product that analyses large amounts of security data like internal network traffic or suspected malware files, and a machine learning algorithm condenses them into more easily digestible insights into the network.

2.4.3 SPLUNK

This tool has various applications covering a range of areas such as IT operations, analytics and cybersecurity. This software was designed to be used as an automated breach investigator, malware defender, and identifying a client's digital weak points. They make use of machine learning algorithms to detect threats so they can be quickly eliminated.

.

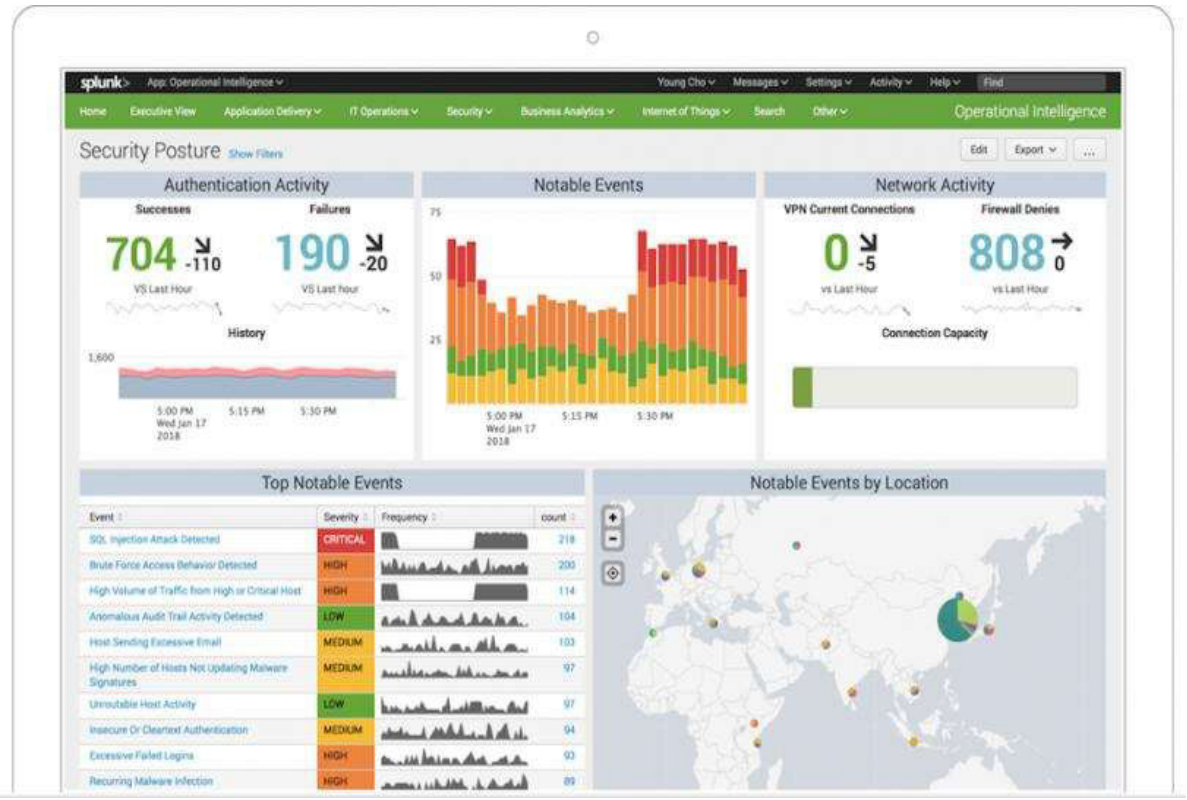


Figure 2. Screenshot from Splunk

2.5 DATA CHARACTERISTICS

In order to train and test the attack detection model, example data is required. Such example data must cover many possible scenarios that may occur within a smart grid so all functionality of the detection model can be tested. In doing research, a power system attack data set published by the Mississippi State University and Oakland Ridge National Laboratory in 2014 was found. These data sets were produced in collaboration with Justin Beaver and Raymond Borge of Oakland Ridge National Laboratory. The raw data logs were provided to Justin by the MSU team, and the Oakland Ridge National Laboratory team formatted these logs into appropriate data sets.

The data they produced contains three sub-datasets; binary, three class and multi-class data sets. Upon investigating the differences between these three, the only change between them is the amount of detail describing what type of scenario each instance of data belongs to. The least detailed is the binary data set which only distinguishes between an attack event and routine operations. The three-class data set improves this by distinguishing between attacks, a naturally occurring event such as line maintenance or standard electrical faults and no events. Finally, the most detailed is the multi-class data set. Its marker attribute distinguishes between individual scenarios and not just the type of scenario the instance belongs to, for example, the same attack occurring multiple times but being executed on different intelligent electronic drivers. For these reasons and the fact that the.ARFF format that the multi-class data set is provided in easily integrates with the machine learning code library; this paper will use this specific data set.

2.5.1 Purpose of the Binary and three class data sets

As previously stated, this paper will be using the multi-class data set. The other two data sets are typically used when less accuracy is required in the specific scenario that needs to be predicted and instead either between two different types in the binary data set three different types in the three-class data set. Many other papers use these three very similar but subtly different data sets to test how increasing the number of scenarios possible for each instance decreases the overall accuracy of the prediction models.

As shown by the paper, ‘Machine learning for power system disturbance and cyber-attack discrimination’ (Raymond C. Borges Hink, 2014), when using the same classifiers but testing across the three different types of data sets, it was found that the accuracy of each classifier decreases when the dataset each classifier is trained on has more possible scenarios. The results for each of these three experiments is shown in the three figures below.

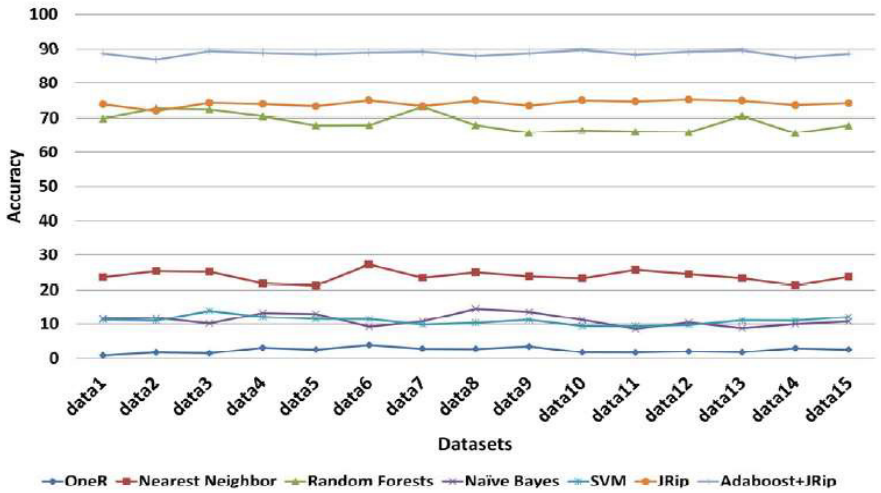


Figure 3. Multiclass Accuracy over Fifteen Datasets (Raymond C. Borges Hink & Mark A. Buckner, 2014)

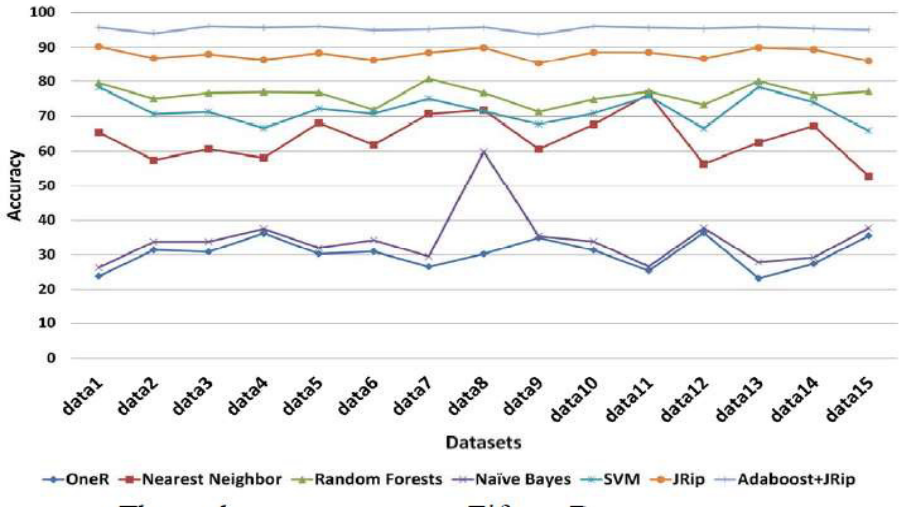


Figure 4. Three-class accuracy over Fifteen Datasets (Raymond C. Borges Hink & Mark A. Buckner, 2014)

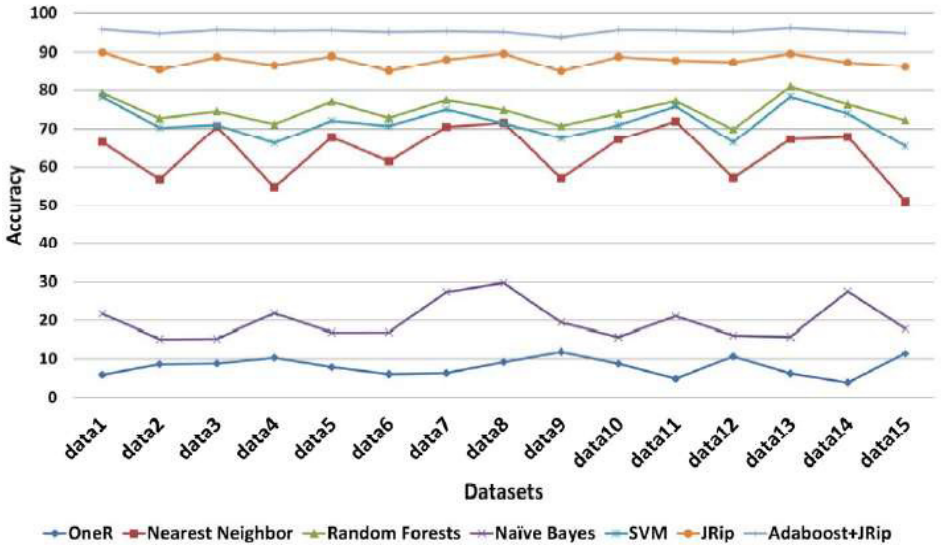


Figure 5. Binary classification accuracy over Fifteen Datasets (Raymond C. Borges Hink & Mark A. Buckner, 2014)

2.5.2 Multi-class data set

The data set is made up of 15 different .ARFF files with 37 power system event scenarios in each. The 37 scenarios are then divided into eight natural events, one no event and 28 attack events. As previously stated, this power system attack data set was produced by Mississippi State University and Oakland Ridge National Laboratory on the 15th of April 2014.

2.5.3 The information gained ranked attributes

Not all attributes are created equal. Some attributes provide more significant amounts of information in terms of predicting the scenario than others. This measurement is known as information gained, and within the WEKA code library, there is functionality to produce a ranked list of attributes based on how much information they provide to a classifier. Using ten-fold cross-validation, each attribute was tested to gain an average metric of information gained. The results are showing in the diagram below.

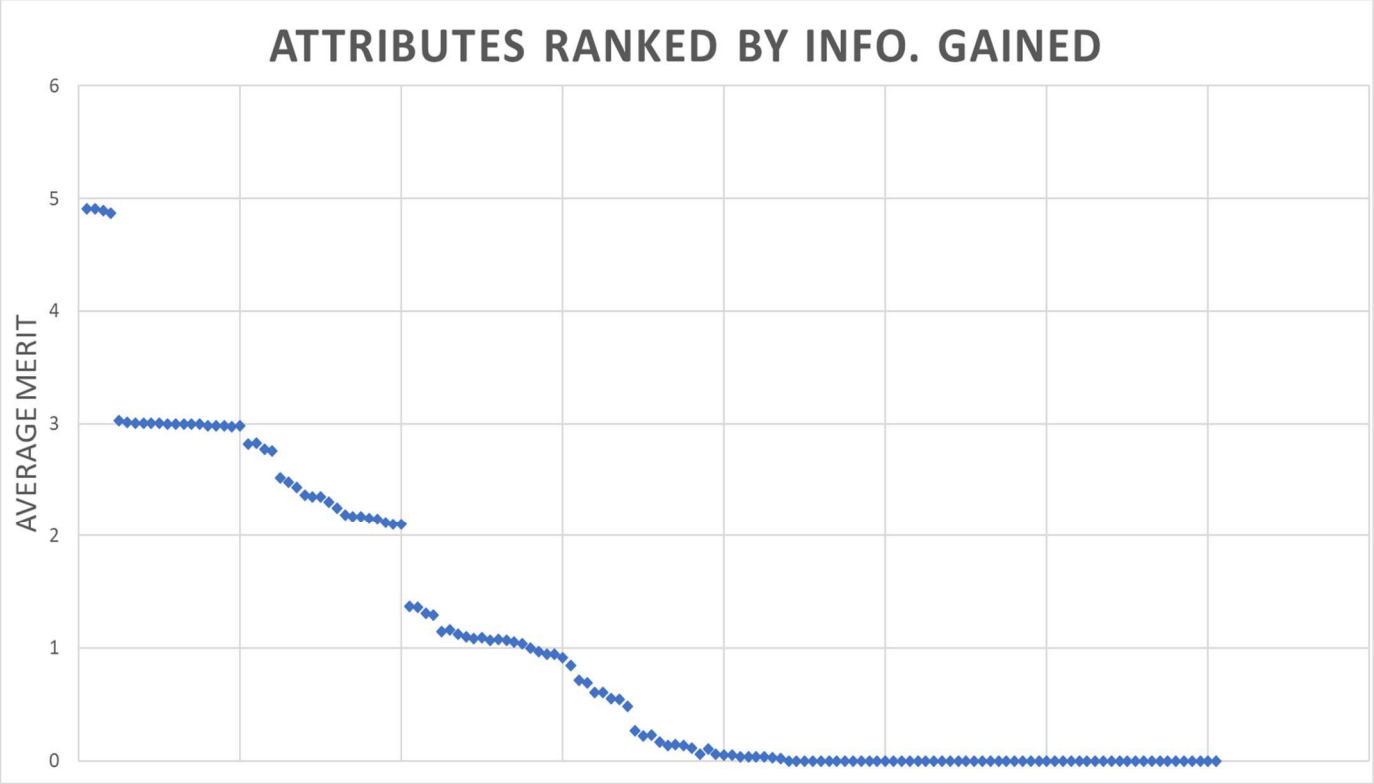


Figure 6. Attributes ranked by info. gained

From these results, it is approximated that about 50% of the 128 attributes provided in the data set produce approximately 96% of the learning value to a classifier. The results can be broken down into approximate clusters. The first cluster is made of the four attributes that produce the highest results. These attributes were the apparent impedance measurement for each relay and ranged between 4.87 and 4.911. The next cluster of results is the voltage phase angles, current phase angles and voltage/current magnitude, ranging between 2 and approximately 3. After this, there is a significant drop in the information gained by each of the attributes. After this drop, there is a precipitous decline which eventually reaches 0 for the last 54 attributes. For these 54 attributes as they give no information to the classifiers, just a hindrance in terms of both processing power required to analyse them and the detrimental effect comprehensively they have on correct predictions.

2.5.3.1 Power system framework configuration

The figure in this section shows the power system framework configuration used in the data set to produce these scenarios. In this diagram, the power generators are labelled G1 and G2. The R1 through to R4 components are called intelligent electronic devices or IED'S; these can switch the Breakers on or off. The corresponding Breakers are labelled BR1 through BR4. There are also have two lines in the grid. The first line spans from breaker one (BR1) to break it to (BR2), and the second line spans from the third breaker (BR3) to breaker four (BR4). Each IED automatically controls an individual breaker; for instance, R1 controls BR1, R2 controls BR2 and so on accordingly. The IED'S use a distance protection scheme that will trip the breaker if a fault is detected, whether that be a valid or fake fault, since they have no internal validation method to detect the differences. Any operators on this smart grid will manually issue commands to the IED'S to manually trip the breakers if needed. The manual override is used when performing maintenance on lines or other system components.

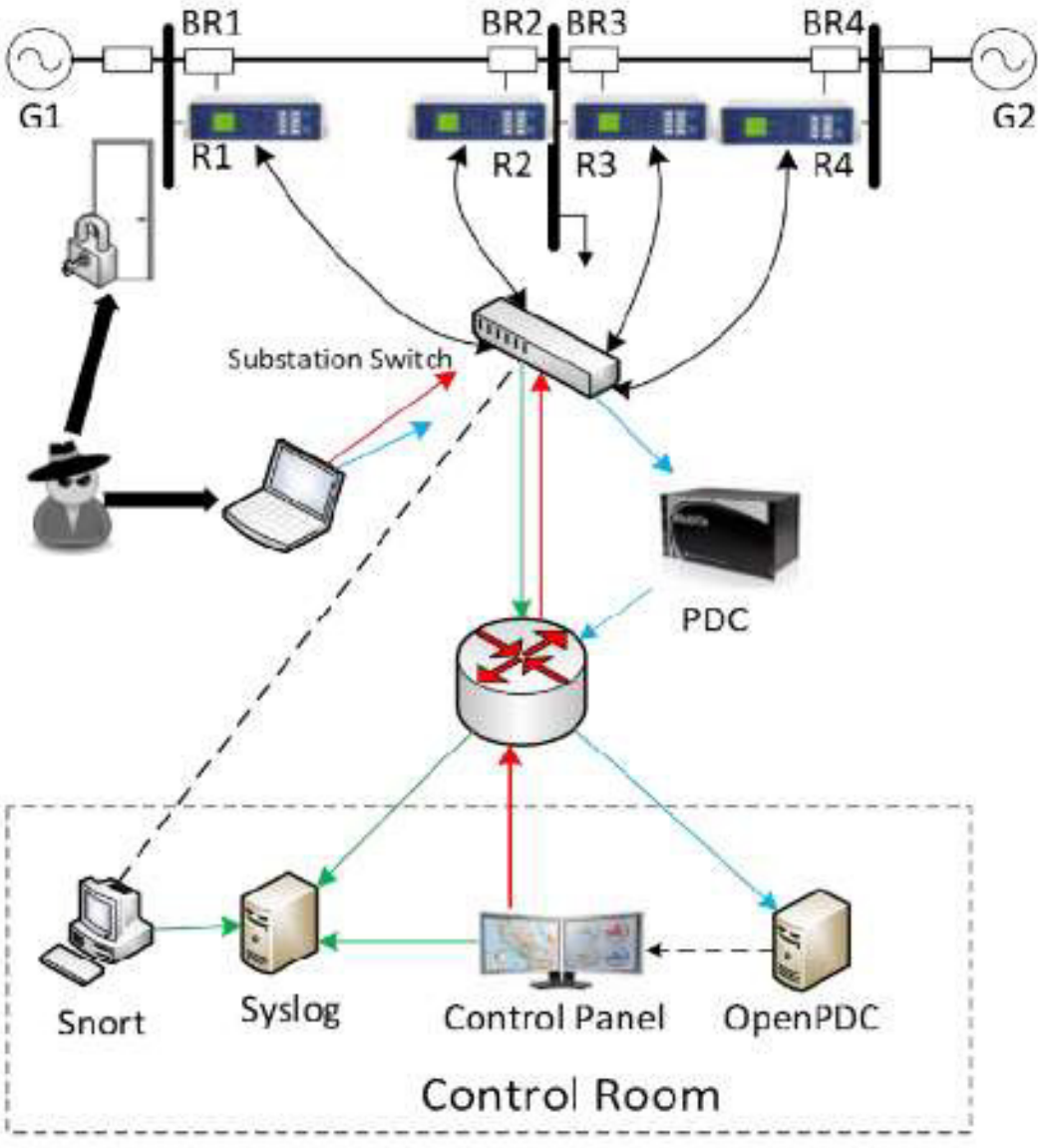


Figure 7. the power system framework configuration used in generating the scenarios (Mississippi State University and Oak Ridge National Laboratory, 2014)

2.5.3.2 Technical Description

As previously stated, the data was drawn from 15 data sets, which included thousands of individual instances of measurements throughout the power system for each type of event listed. They were then randomly sampled at 1% to reduce the size and evaluate a small sample size effectiveness. From this, we can see an average of 294 no event instances, 3711 attacking stairs and finally 1221 natural events instances. The data file in total contains 78,963,019 bytes divided up into 15 separate .ARFF files. The open-source data will be using is classed as scientific data. There are 37 separate scenarios within the data set. Scenarios 1-6, 13 and 14 are natural event scenarios. Scenario 7-12, 15-30 and 35-40 Attack event scenarios and make up the vast majority of the data set scenarios. Finally, scenario 41 is classed as a no event scenario. Scenarios 31 to 34 do not exist within the data set.

Table 1. THREE-CLASS CLASSIFICATION GROUP (Mississippi State University and Oak Ridge National Laboratory, 2014)

	Attack Events	Natural Events	No Events
Scenarios	7,8,9,10,11,12,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,35,36,37,38,39,40	1,2,3,4,5,6,13,14	41

2.5.3.3 Non-Attack Scenarios

Within the data set, three distinct scenario types can be classed as non-attack scenarios.

2.5.3.3.1 Short circuit fault (Natural Event)

a. This is a short in the power line that can occur in various locations along the line; the percentage range indicates the location.
b. Six scenarios fall under this type.
c. Faults in these scenarios can either occur on line one or line two.
d. Three of the scenarios are dedicated to each line.
e. Faults range from 10% to 19%, 20% to 79% and 80% to 90% on both lines.

2.5.3.3.2 Line maintenance (Natural Event)

f. One or more relays are disabled on a specific line to do maintenance for that line.
g. This can occur on either line one or line two and only contains two separate scenarios.

2.5.3.3.3 Normal operation load changes (No Event)

h. This scenario describes the regular and expected load changes within the smart grid and can be considered the standard operating data.

2.5.3.4 Feature Description

There are a total of 128 features for each of the instances of data. There are 29 types of measurements from each phasor measurement unit (PMU), with a total of four of these measurement units in our smart grid. Phasor measurement unit or synchroniser is the primary method used to collect all data, and it is responsible for 116 PMU measurement columns in total, making up the majority of the data's features. The remaining 12 features are used for control panel logs, snort alerts and relay logs of the four PMU/relay. Finally, the last column is used as a marker and defines which scenario each of the instances relates to. This information would not be available in a realistic context and instead would be replaced by a question mark. The application will recognise this question mark and will know the instance is live data. Data used to train the classifiers will contain this information as the classifiers will need this to build an accurate model of the data set.

2.6 DESCRIBE THE ATTACKS

The other three types of scenarios available for us to test are all different attack events. Each attack scenario type may be divided into subtypes; these subtypes will all follow the same method as their over-arcing attack type but typically differ by performing the same attack on multiple components of the smart grid at the same time or Increasing the complexity of the attack compared to the previous attack subtype.

2.6.1 Remote tripping command injection (Attack)

a. This is an attack that sends a command to a relay which causes a breaker to open. It can only be done once an attacker has penetrated outside defences. (Mississippi State University and Oak Ridge National Laboratory, 2014).
b. This scenario is then subdivided into two types of attack. These being command injection against a single relay or command Injection against two relays.
c. There is a command injection scenario for each of the single relays creating four in total for the command injection against a single relay attack subtype.
d. There is also a command injection scenario against relay one and relay two co-occurring and a command injection against relay three and relay four in another scenario for a total of two scenarios for this attack subtype.

2.6.2 Relay setting change (Attack)

e. Relays are configured with a distance protection scheme, and the attacker changes the setting to disable the relay function such that the relay will not trip for a valid fault or a valid command. (Mississippi State University and Oak Ridge National Laboratory, 2014).
f. This attack type is divided into three separate subtypes: disabling relay function against a single relay disabled and fault, disabling relay function against two relays disabled and fault, and finally disabling relay function against two relays disabled as well as line maintenance.
g. This attack type is the largest out of all three attack types.
h. Within the first subtype of this attack, there will be a fault on one of the two lines within the network and one of the relays being disabled and showing a fault; this is repeated against different areas of the line and different relays being disabled.
i. This is again repeated but disabling two relays instead of 1.
j. Finally, the last subtype describes standard line maintenance, as well as two relays, being disabled.

2.6.3 Data Injection (Attack)

k. Here we imitate a valid fault by changing values to parameters such as current, voltage, sequence components etc. This attack aims to blind the operator and causes a blackout. (Mississippi State University and Oak Ridge National Laboratory, 2014).

l. This type of attack scenario attempts to mask itself by mirroring natural fault events on line one and line two and including a trip command to attack the system.

2.7 DATA SET SCENARIOS

From the three main groups, the dataset is broken down into 37 separate scenarios. This is further explained in the table below.

Table 2. Data set Scenarios Breakdown Table

Scenario Number	Description	Type
1	Fault from 10-19% on Line 1	Natural
2	Fault from 20-79% on Line 1	Natural
3	Fault from 80-90% on Line 1	Natural
4	Fault from 10-19% on Line 2	Natural
5	Fault from 20-79% on Line 2	Natural
6	Fault from 80-90% on Line 2	Natural
7	Fault from 10-19% on Line 1 w/ tripping command – data injection	Attack
8	Fault from 20-79% on Line 1 w/ tripping command – data injection	Attack
9	Fault from 80-90% on Line 1 w/ tripping command – data injection	Attack
10	Fault from 10-19% on Line 2 w/ tripping command – data injection	Attack
11	Fault from 20-79% on Line 2 w/ tripping command – data injection	Attack
12	Fault from 80-90% on Line 2 w/ tripping command – data injection	Attack
13	Line 1 maintenance	Natural
14	Line 2 maintenance	Natural
15	Remote Tripping Command Injection to R1	Attack
16	Remote Tripping Command Injection to R2	Attack
17	Remote Tripping Command Injection to R3	Attack
18	Remote Tripping Command Injection to R4	Attack
19	Remote Tripping Command Injection to R1 and R2	Attack

20	Remote Tripping Command Injection to R3 and R4	Attack
21	Fault from 10-19% on Line 1 with R1 disabled and fault – relay setting change	Attack
22	Fault from 20-90% on Line 1 with R1 disabled and fault – relay setting change	Attack
23	Fault from 10-49% on Line 1 with R2 disabled and fault – relay setting change	Attack
24	Fault from 50-79% on Line 1 with R2 disabled and fault – relay setting change	Attack
25	Fault from 80-90% on Line 1 with R2 disabled and fault – relay setting change	Attack
26	Fault from 10-19% on Line 2 with R3 disabled and fault – relay setting change	Attack
27	Fault from 20-49% on Line 2 with R3 disabled and fault – relay setting change	Attack
28	Fault from 50-90% on Line 2 with R3 disabled and fault – relay setting change	Attack
29	Fault from 10-79% on Line 2 with R4 disabled and fault – relay setting change	Attack
30	Fault from 80-90% on Line 2 with R4 disabled and fault – relay setting change	Attack
31	Scenario Number Not Used	None
32	Scenario Number Not Used	None
33	Scenario Number Not Used	None
34	Scenario Number Not Used	None
35	Fault from 10-49% on Line 1 with R1 and R2 disabled and fault – relay setting change	Attack
36	Fault from 10-49% on Line 1 with R1 and R2 disabled and fault – relay setting change	Attack
37	Fault from 10-49% on Line 1 with R1 and R2 disabled and fault – relay setting change	Attack
38	Fault from 10-49% on Line 1 with R1 and R2 disabled and fault – relay setting change	Attack
39	Fault from 10-49% on Line 1 with R1 and R2 disabled and fault – relay setting change	Attack
40	L1 maintenance with R1 and R2 disabled – relay setting change	Attack
41	Normal operational load changes	Natural

2.8 TYPES OF CLASSIFIERS

Available through the machine learning library, there are many different classification algorithm groups. It is essential to have a basic understanding of the different types of classifier so when the specific classifiers that will be used later emerge, patterns between the types of classifiers that are successful will be more apparent.

2.8.1 Bayesian

This type of classifier uses the Bayes theorem in some capacity. This theorem is used to predict class values by probabilities.

2.8.2 Functions

These classifiers can be written as an equation and estimate a function.

2.8.3 Lazy

These store training instances and work occurs during classification.

2.8.4 Meta

Meta classifiers allow us to combine multiple algorithms, which may, in turn, convert them into more powerful learners.

2.8.5 Miscellaneous

This group is a catch-all group for any classifiers which do not fit into any other groups.

2.8.6 Rules

These classifiers generate a list of rules used to interpret which classification the new data best fit.

2.8.7 Tree

This group of classifiers makes use of decision trees based on route attributes and leaf nodes.

2.9 SUMMERY

Throughout this section, the different elements involved in this project have been detailed. This topic is extensive, but a limit had to be applied in terms of scope as not to be overwhelmed with the amount of work that needs to be done and still being able to do that work in enough detail to make a significant contribution. This background knowledge enables us to understand better the topic and the pre-existing solutions that surround it. With this understanding going forward, this project can focus its efforts on areas relevant to both the research aspect of this project and the application development element.

3 CHAPTER 3: APPROACH

3.1 OVERVIEW

This section will be used to cover the project planning and methodologies that have been used to develop the tool. The functional and non-functional requirements identified to be critical to the project's success or failure will be listed and explained. In addition to this, there will also be an explanation of the tool's intended function using diagrams.

3.2 PROJECT PLANNING

During this project's planning and research phase, an initial plan to lay out the path and rough timetable for how this project should go was developed. However, the timetable has changed since developing the tool, and the paper writes up—first, a proof of concept tool to verify that it was possible to complete all the requirements was developed. Upon completion, it was found that all the requirements, both functional and non-functional, would be completed in the time frame.

3.3 METHODOLOGIES USED

To structure how a project like this should be implemented, a methodology must establish the implementation cycles used. This section will explore the methodologies that were considered to use to develop the tool.

When considering which methodology to use, there are some aspects to the project that must be considered. The overarching characteristic of the project when it comes to the development process is that the number of developers for this tool is minimal. It is challenging to get regular and informed feedback on the tool without detailing every element of the project and the context involved as no one besides the development team is as heavily involved in the development process. Working on this project also means that no customer or target audience will frequently request changes or enhancements to the requirements or solution. This limitation does limit the amount of feedback from the target audience that can be gained. Clear and defined requirements will be laid out, all of which need to be achieved, but the method to achieving this has not necessarily been strictly defined yet, Meaning that the overall solution to the problem can change as development proceeds. The final key aspect that will be used to select the development methodology is the level of experience of the development team in the various aspects of the project.

After considering many different software development methodologies, three were further considered; *Scrum, Kanban, and Waterfall.*

Scrum was the initial methodology considered as it has previously used in other projects and was found to have worked to excellent effect. This methodology implements the same foundational beliefs and philosophies as an agile approach. This methodology heavily involves the use of teams and collaborations. Due to this reliance on a group, this methodology was discarded. Despite its intuitive approach that breaks down more significant problems into minor, more manageable problems, which are then distributed out, the heavy reliance on collaboration would not be possible for this project.

The second methodology considered was Kanban. It is typically used in projects that maintain and upgrade an already produced product by performing minor tweaks and analysing whether these small adjustments help the projects' overall functionality or hinder it. This methodology was not chosen as it requires a minimum cost of delays to perform and analyse these minor tweaks, which is not possible

within this projects' context. As the tool is built from scratch, there is no original project to build upon, Which would cause methodology to struggle, particularly at the beginning of the project.

Finally, we come to the methodology chosen to use when developing the project, Waterfall. This methodology uses strictly defined phases that must be completed to move on to the next phase. These phases include requirement analysis, design, implementation, testing and deployment. This methodology requires the requirements to be laid out very clearly, with the bare minimum of changes made once the project has started. This methodology does not require a defined process to develop the end product as the requirements have been defined and strictly kept to. This method gives a lot more control within the development process, which would cause significant delays if there was a large team working on this project. Despite the freedom given in the development process, this could lead to an underdeveloped product or a significantly delayed one. The strictly defined phases and deadlines set mean that there will always be constant incremental progress towards the optimal solution and will have a functional and practical tool long before the deadline requires one.

3.4 REQUIREMENTS SPECIFICATION

The requirement specification is the basis for this entire project. It helps to lay out a framework that will be followed throughout the development process. It helps to provide crucial information that is necessary to fulfil the task we have set out to. Functional requirements are requirements that must be completed in order for the tool to complete its most basic functionality. Non-functional requirements help to enhance these abilities and improve the tool, so it is more user friendly.

3.4.1 Functional Requirements

Requirement 1: Read a .ARFF file from a specified folder.

This file is the only input of data that the tool will receive. Therefore, it is essential for this tool. A . ARFF file type will be used as it can be created without any additional classes besides the one already provided by WEKA. It can be quickly analysed using other tools to gain valuable information about the data that can then use to inform choices in the development process. The file is to be fetched from a specific folder so that any external capture tool that the user may use to collect the System data to place in a file will only have to place the completed file in that folder to input it into the tool.

Requirement 2: Error check captured data.

As previously stated, the data captured from the file is essential and must be checked to ensure that the data is not redundant, repeated, and is in the correct format. This will take the form of ensuring the file came from the correct location and is in the correct format to be processed.

Requirement 3: Filter unnecessary attributes.

Some attributes within the data do not provide any information to the classifiers. Therefore they are useless for our purposes. Removing them ensures that we are not processing extra information that is not needed and has no impact on the prediction. The benefits of doing this increase exponentially as we process larger quantities of data. This will ensure a speedy prediction in real-time.

Requirement 4: Analysis and Detection of short circuit faults.

When a file is located and passes error checking, the selected machine learning algorithm that the user would like to use must search for signs of a short circuit fault and flag the relevant information. As this type of fault is not an attack, no extra action must be taken

Requirement 5: Analyse and Detection of line maintenance.

When a file is located and passes error checking, the selected machine learning algorithm that the user wants to use must search for signs of line maintenance and check if the user expects said maintenance. If not, the tool must decide whether the line maintenance is an attack on the system using the cover of line maintenance to camouflage the attack.

Requirement 6: Analyse and Detection of relay setting change attacks.

When a file is located and passes error checking, the selected machine learning algorithm that the user would like to use must search for signs of a relay setting change attack on this system. If found, the relevant information and context must be flagged.

Requirement 7: Analyse and Detection of data injection attacks.

When a file is located and passes error checking, the selected machine learning algorithm that the user wants to use must search for signs of a data injection attack on this system. If found, the relevant information and context must be flagged.

Requirement 8: Analyse and Detection of remote trip command injection attacks.

When a file is located and passes error checking, the selected machine learning algorithm that the user wants to use must search for signs of a remote trip command injection attack on this system. If found, the relevant information and context must be flagged.

Requirement 9: Analyse and Detection of single or multiple attacks.

Typically, an attacker will use multiple methods to cause malicious damage to a system; thus, multiple attacks will likely co-occur on the system, which will be reflected in the data. The tool should simultaneously identify any attacks on the system and then inform the user about which attacks have been detected.

Requirement 10: Continuous monitoring for new data.

The tool must continuously check folders for new data while it is running on the system as it is imperative that as soon as the data of the system is collected and formatted, it is being processed and analysed.

Requirement 11: Creating voting methodologies.

The tool must take predictions made by single classifiers and combine them using different methods to produce a more accurate result. There must be multiple different methods to this in order to give the user a large variety.

Requirement 12: Saving and Loading of classifiers.

Classifiers take up a large amount of memory, and in order to prevent the application from crashing once an application has been built, they need to be saved to an external source. Doing this frees up memory and will help to speed up the overall process of the application. These classifiers will then need to be loaded back into the application so they can be used to make predictions.

Requirement 13: Allow the user to select an overall prediction method.

To make a definitive decision about the scenario currently occurring within the smart grid. The application must pick one prediction to go with. This functionality will allow the user to select which method they feel will produce the best results to inform the application, making the appropriate responses.

Different classifiers may be more beneficial in specific contexts than others, and as such, the user must have the ability to select and use whichever classifier they choose to get the best results. Providing this wide variety will give the user plenty of options to customise the tool for their specific needs.

Requirement 14: Provide essential information and context about the attack.

The user will require all critical information and context about the attack to decide how to proceed. They are likely to be the systems admin so that they will be knowledgeable about the different elements of the smart grid, and as such, we can show them precise data about the attack and in great detail, although this does need to be balanced and measured as the amount of data we could give them would be overwhelming.

Requirement 15: Provide recommended actions on how to proceed.

Depending on which attack or attacks have been detected within the data, the tool should display some recommended countermeasures to prevent this type of attack or attacks from occurring again.

3.4.2 Non-Functional Requirements

Requirement 16: Ease of implementation/deployment.

The tool must easily be deployed and set up as deviating from this will only detur potential users from using the tool. Once the tool has been produced, it must be published as an executable file as most users will be familiar with this file type.

Requirement 17: Information is displayed in an orderly manner.

In the same way that too many options can be overwhelming for the user, the same can also be said about having too much information. When information is presented to the user, whether that be information about the data or the tool's status, the information presented must be clear, concise, and ordered so that it is easily digestible by the reader.

Requirement 18: Start processing new data as soon as it becomes available.

It is critical that when the data becomes available, it is immediately inputted into the tool as this would be the most up to date data and best reflects the current situation happening within the grid so in order to understand what is happening within the smart grid at present this data needs to be processed.

Requirement 19: Each new data set must be processed promptly.

Similarly to the previous requirement, all data must be processed quickly to pass the relevant information on to the user. Any delay between the data being captured and the final report being produced increases the likelihood that an attack will damage the smart grid.

Requirement 20: Completely processed and analysed data must be changed all moved to prevent duplicate analysis. Processing and analysing data multiple times will decrease the accuracy of the classifiers as they will have been exposed to more instances of a particular type than what has occurred. This will be achieved by moving the file from the input folder to a processed folder and changing the name

3.5 SYSTEM DESIGN

This section of the paper provides a better understanding of the inner workings of both the smart grid and the tool itself. There will be a diagram describing the standard operation of the smart grid without the tool.

3.5.1 System model

A system model is a process-orientated representation the emphasise is the flow of information between its different modules and can easily demonstrate how this information influences the smart grid's operations. Using this model will allow us to understand how processes interact and what operations these processes perform but are abstracted to make them more comprehensible. In our system model, there are two main actors: the operator and malicious hackers. The operator actor's actions will change depending on Whether the tool is currently running on their system, and as such, it is necessary to demonstrate the impact that our tool will have on the system by using two separate system model diagrams. This also allows us to go into greater detail and understand the context that our tool operates in.

3.5.1.1 Standard operation of the smart grid without the tool

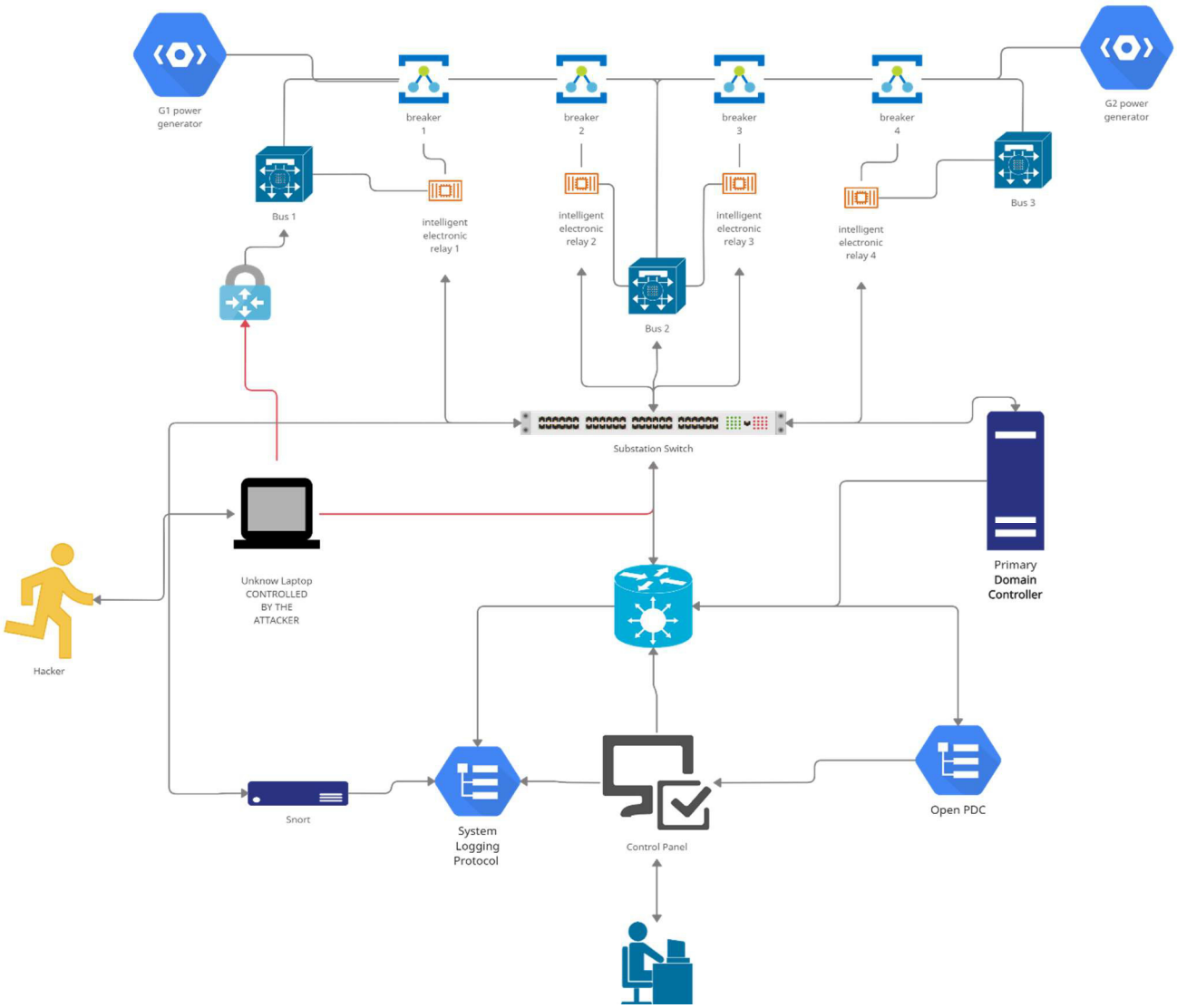


Figure 8. Smart Grid System Breakdown

As shown in the diagram, a hacker will try to access the smart grid system through critical weakness points such as a back door at the intelligent electronic driver/power generators or try to gain access through the substation switch, both known vulnerabilities. The other actor, the operator or systems admin, will operate within the control room and work from the control panel element. The control panel element will give them access and control over the entire smart grid system. Within this current version of the smart grid, the element responsible for the security of the smart grid and subsequently the detection of any attacks is the snort element. This element will make use of packet sniffers that monitor network traffic in real-time and check each packet closely to detect any dangerous or suspicious data within the packet. While this is the main element responsible for the security, it is not the only part of the smart grid that does this. On the control panel PC, there will be Proprietary software responsible for detecting ransomware, Malware, fishing, man in the middle attacks, denial of service attacks, SQL Injection, zero-day exploits and DNS tunnelling, among the other standard security available within the operating system, such as windows, they would use Windows Defender.

Information will first be gathered from the power generators via intelligent electronic devices, which will feed this information to the smart substation grid. There is also a breaker for each intelligent electronic device if they needed to be switched on-off. Once this information has been gathered, it is fed through the primary domain controller responsible for security within this specific local domain. Once it has cleared, this data is fed into the hub and distributed to the systems log to keep an accurate record and passed to the other primary domain controller on the smart grid. Once the information has cleared, the open primary domain controller passes it on to the control panel, where the systems admin operator can view and make decisions based on the data. Any operator can manually issue commands to the intelligent electronic driver to manually trip the breakers if they choose. These commands are fed back into the hub that passes it onto the substation switch and is passed along to the relevant elements within the smart grid's power generation section.

The intelligent electronic drivers will use a distance protection scheme that automatically trips the breaker upon detecting a fault, whether they are valid or fake since they do not have any internal validation method to detect the difference. These manual overrides and commands are typically used when performing maintenance on the lines or any other components that require it.

3.5.1.2 Standard operation of the smart grid with the tool

Within the smart grid, the tool will operate within the control room on a personal computer. This tool will run in the background of this PC to not interrupt the standard operating procedure of the control room. Data for the tool will be gathered by an external source supplied by the user. Typically these two will go unnoticed most days as scheduled maintenance is performed and no events happen. The only time a user should notice the tool is when an unexpected event occurs. The user will view the tool from the personal computer and interact to change settings from there. The produced reports will also be displayed within the tool to access them from the personal computer.

3.6 SUMMARY

The project plan was presented using a Gantt chart to easily visualise the different elements of the project that would have to be completed and at what time they needed to be completed. This can be found in the appendix. The methodology that has been chosen to use allows the freedom to continuously change the plan for the project as new challenges arise. Thus the plan needed to be adapted to reflect these challenges better. This freedom has been invaluable as it means that there are no limitations in this thinking. As the requirements for this project were firmly defined at the start, one can make these changes as long as the requirements are met in the final product.

4 CHAPTER 4: SOLUTION IDEA AND IMPLEMENTATION

4.1 OVERVIEW

In this section, the platform used to develop the tool and the specifications will be mentioned. A general overview will be included about how the tool works, including the different features and functionality included in the tool. While also utilising diagrams to help visualise the workflow of the idea and the process of the tool, including snapshots of the tool to demonstrate these theoretical points. This section will discuss the issues faced while developing the tool, and a satisfactory solution to the problems encountered was found.

4.2 TECHNOLOGY CHOICES

4.2.1 Computer specifications

This project was developed from a personal computer, an HP zed one entry tower G5 using the windows operating system. There are 16 gigabytes of installed physical memory, ten of which was assigned to the Java heap pile when running the application. The processor is an I7-9700 CPU with eight cores. The architecture is 64 bit. There is also have a GPU on the machine, which can be used to help with calculations.

4.2.2 Libraries

In order to have access to data mining tools and machine learning algorithms, one needs to be able to access them through a library. The library chosen for this is the Waikato Environment for Knowledge Analysis (Weka), developed at the University of Waikato, New Zealand. The library itself contains a collection of both visualisation and the actual algorithms themselves used for data analysis and predictive modelling, both of which have been a big part of this project. The library is written in JAVA .the latest release is currently 3. 9.5, released on the 21st of December 2020. Once all the libraries have been downloaded, they can easily be accessed through the Java API available on the WEKA website.

4.2.3 Programming language

4.2.3.1 JAVA VS Python

The WEKA Java API, as the name would suggest, is coded in Java, and as such, this language is needed to access the library. However, upon investigating, Python would work better for this application. This is where the Python WEKA wrapper package steps in. This package makes it easy to run the algorithms and other functions from within Python 3. It offers access to the WEKA API using wrappers around JNI calls using a downloaded Java bridge package. This type of application requires much experimentation, mainly revolving around selecting algorithms and options for the classifiers, and Python is easier to use and easier to read. Both of these languages are object-orientated, but Java uses static types, whereas Python uses dynamic. For these reasons, Python will be used as the programming language for the tool.

4.3 THE OVERALL WORKFLOW OF THE IDEA

When the user starts the tool, it first gathers all the training data available inside the import file. Once this data has been gathered and checked, it is saved to the all data .arff file. This enables classifiers to be trained on the entire data set at once rather than incrementally train then, which would take significantly longer as one would be training the same classifiers on the same data as each file is added to the old data file that is used to build the classifiers. At this stage, an attribute selection is performed on the data using a genetic search for these search algorithm and consistency subset evaluation for the evaluator. Doing this allows the data to reflect the class attribute better and significantly improves the accuracy of some classifiers.

Next, this filtered data is used to train each of the classifiers that the user wishes to use. Once each of the classifiers has been created using the filtered data, it is then serialised to a file outside of the application. This reduces the memory that the application uses significantly. Once the classifiers have been trained, they will then be used for predictions. Data is imported from the new data file and is then split between training data and live data. Training data has an integer in its class attribute as we have predefined which scenario this instant belongs to, whereas live data will have a question mark in its place detailing to us that the scenario is unknown.

If the data is training, data predictions are performed on it and then it is passed to be added to the all data file to be used in updating the classifiers at the end of the test phase. However, if the data is live, predictions are performed using the individual classifiers and all other prediction methodologies. These prediction methodologies use the predictions from the classifier to make a judgement on what they believe is likely to be the correct prediction. The user preferences are then checked to see which prediction methodology they would like to use as the overall prediction. Upon receiving this prediction, the tool will then warn the user appropriately and supply relevant advice.

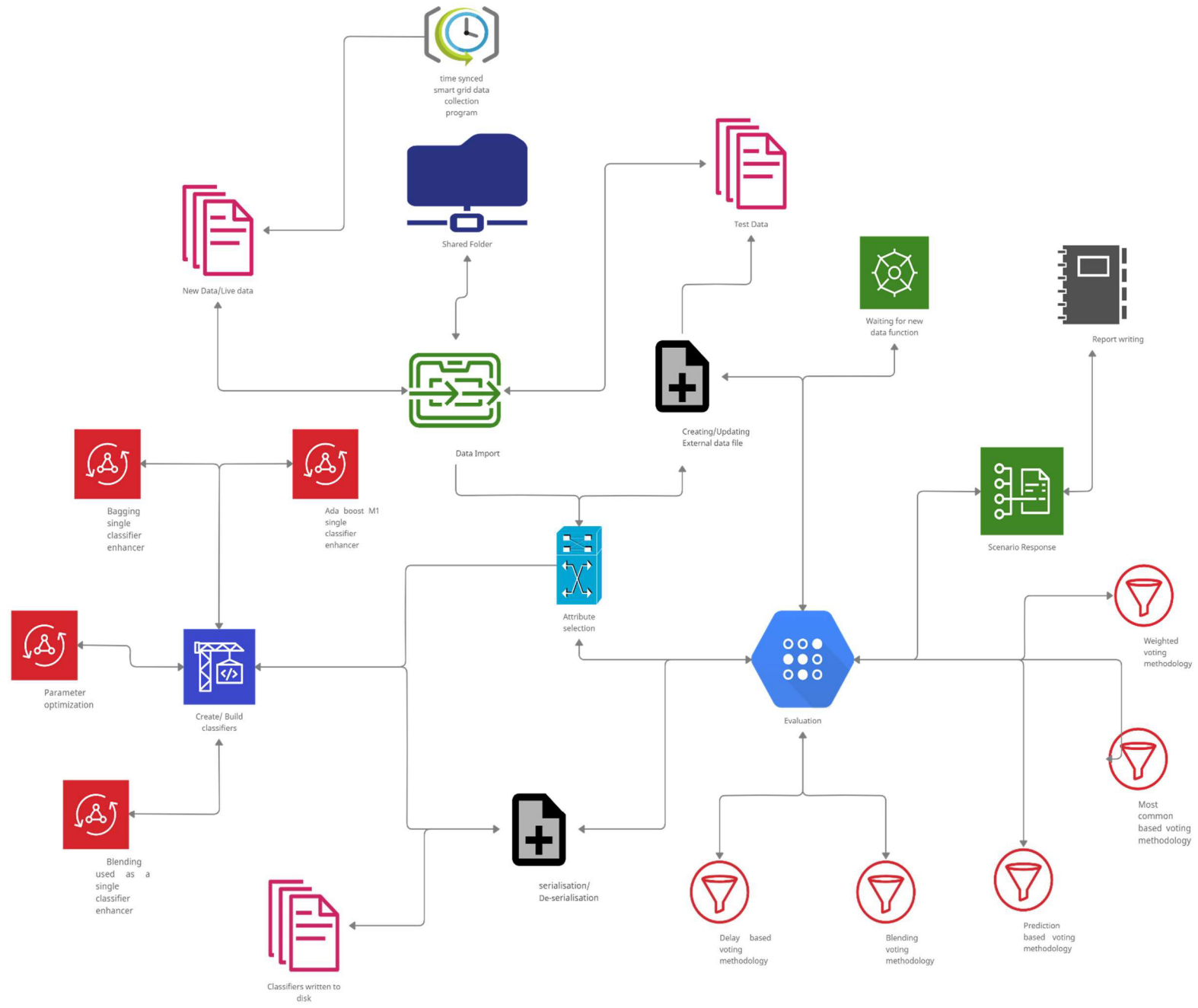


Figure 9. OVERALL WORKFLOW OF THE IDEA

4.4 OVERALL EXECUTION ELEMENTS

4.4.1 Data access

Within the execution, there are two times when the application would need to fetch data. The first of these occurs when the user starts up the programme. The application will first fetch all the file names found within the import file directory. The individual file names are then iterated through, and using a separate function, the data associated with that filename is imported into the application but not before the file is checked for overall size, if it can be successfully imported and if the attributes are matching. Once a file has been successfully imported, it is used to create the old data file or update the all data file if it currently exists; this will be explained in a separate section. The WEKA function used to import the files is called the ARFF loader.

4.4.2 Creating and updating an external data file

If a piece of data needs to be written to a file, this can be done using a few functions. The first is to create a data file. We use the saver function called ARFF saver and save it using a predetermined name. This specific function found within this part of the library is save_file(). If a file needs to be updated, however, then this is a bit more involved. First, the current data file is fetched and is combined with the data to be used in the update. Next, the old file is removed, and a new file is created with the same name but with the combined data sets.

4.4.3 Attribute selection procedure

Attribute selection is the first stage in the application that uses machine learning algorithms. The purpose of attribute selection is to find a particular combination of the 127 available attributes that best reflects the scenario the individual instance belongs to. This process uses two machine learning algorithms; one is used as a search algorithm and the other as an evaluation. When the search algorithm finds a subset, it is passed to the evaluation algorithm, which evaluates how well the subset reflects the class attribute. Focused on how well the subset reflects the class attribute because this is an excellent indicator of how well the classifiers will process the data. The better the subset of attributes reflects the class attribute, the greater the likelihood of prediction is. Reducing the number of attributes that have to be processed every time the classifier needs to evaluate a new piece of data or when an initial classifier is built will result in significantly less computing power being used and the overall performance of the tool improving including the number of correct predictions when testing new data. The easiest way to think about attribute selection is that it reduces the noise of the data set, thus making patterns easier to find.

When the training data is first imported into our application before the classifiers are trained, the attributes selection is performed on the data set. The recommended filter is then applied to the data before a classifier is built on it. This filter is also applied to any test or live data that is fed into the application. During the development stage of the tool, each combination of evaluator was tested and search algorithm to discover which produced the best result. The best combination found was consistency subset evaluation for the evaluator and genetic search for the search algorithm. Then worked on parameter optimisation to ensure that regardless of the amount of data they were fed, they would produce the best possible combination of attributes.

4.4.4 Creation and serialisation of individual classifiers

When a classifier is created in the application first, the class name and data used to build the classifier are passed to create a classification function. This function takes the class name and builds a full classpath to the library that contains the classification algorithm. The classifier is then built using the passed data and then returned to wherever it was called. Once this classifier has been created, it is then serialised using a temporary file in a temporary directory; the class name is used as the temporary file name. This is achieved to save memory as holding multiple fully trained classifiers would require an excessive amount of memory. Using this method means that only one fully trained classifier is in memory at one time.

(a) Ada boost M1 single classifier enhancer

Ada Boost EM1 is short for Adapted Boosting. It is a statistical classification meta-algorithm used by other weak learner algorithms to improve performance. It uses the weak learner's output and combines it into a weighted sum representing the final overall boosted classifier. The boosting algorithm has to be weak to see significant improvement; otherwise, there is a minimal improvement. This meta-algorithm is placed after the creation of individual classifiers in order to boost them and improve performance. The only negative impact of using this is that it significantly increases the time taken to create a classifier. It achieves this improvement by taking the base classifier trained on the training data set and creating a second classifier behind it to focus on any instances in the training data set that the first classifier incorrectly classified. This process continues to add classifiers until a specific limit is reached, whether that be the number of classifiers allowed or accuracy.

4.4.5 Bagging single classifier enhancer

Bagging is also known as bootstrap aggregating, is an ensemble method that creates multiple unique separate samples of the training data set we use and creates a classifier for each sample. The predictions that these classifiers then produce is then combined to form an overall prediction for the bootstrap aggregated classifier. This technique gives better predictions for some classifiers as each sample of the training data set is unique and gives the classifier trained only on it a different perspective on the data set and subsequently the problem. These individual predictions can then be combined to produce a single prediction that is, in some instances, more accurate. Unfortunately, this single classifier enhancer also significantly increases the creation time of a classifier, particularly when it comes to sizeable initial training data sets.

4.4.6 Blending used as a single classifier enhancer

Blending is when multiple different algorithms are prepared on the same training data set, and a meta classifier is prepared that learns how to take the predictions of each of the classifiers and make a more accurate prediction on unseen data. This blending is also known as WEKA's built-in voting methodology. However, it will be used as a voting methodology and as a single classifier enhancer. This process is done as some classifiers see improvements when using bootstraps aggregating and adaptive boosting. Using this voting system will combine two meta classifiers made up of the same base classifier trained on the same data set.

4.4.7 Delay based single classifier enhancer

This single classifier enhancer takes the previous predictions that the classifier has made up to a certain number and returns the most common value found. First, an individual classifier classifies a test instance. This prediction is then added to a previous prediction list. The most common prediction found in this list is then returned as the new prediction. Once a certain number of predictions are within the prediction list, then the oldest prediction still in the list is removed.

This voting methodology uses the context of this data, which is that any data that needs to be tested is produced in real-time from the smart grid and as such will be in order, so scenarios are unlikely to change after every instance instead after a group of 50 to 200.

4.4.8 Blending voting methodology

It has been previously explained how blending, also known as stacking, works. This technique will also be applied to the predictions of all the classifiers into one single meta classifier. Classifiers as used in this all have to be trained on the same data.

4.4.9 Most common based voting methodology

This is the most simple voting methodology been created in this project. This voting methodology takes each of the predictions from the individual classifiers, and whichever scenario appears the most in the list of predictions is returned as this voting methodologies prediction.

4.4.10 Weighted voting methodology

This voting methodology works in very much the same way as the most common voting methodology, except the individual predictions from the classifiers is duplicated based on the user input. If a user trusts a particular classifier more, they can weigh this classifier two times, three times, four times ETC. More than the other classifiers. This can also be used to rank the classifiers ranging from one to the number of classifiers available. If the prediction is weighted twice more than another prediction, this means that the more trusted classifier's prediction is added twice to the prediction list. Once the -weighting has been applied, the most common scenario in the prediction list is returned as the prediction.

4.4.11 Deserialisation of a classifier

Deserialisation is the process of fetching a classification model from the disc, which the application has previously saved the classification model. This moves it into memory, where it can then be used to make predictions. This is done by a simple get function that retrieves the data associated with the given filename.

4.4.12 Individual scenario response

The individual scenario response dictates how the application should respond to a predicted scenario. This means supplying the user with appropriate data and advice about the incident promptly. The individual scenario response will report the type of attack and where it is occurring. This will then supply advice that the user should follow to mitigate any potential attacks. This prediction response will compare any new prediction with the previous prediction, and if they match, then the prediction is confirmed if they do not; however, then the new prediction is stored, and the next instance is processed. Once the new instance has been processed, its prediction is then compared to the previous prediction and the prediction before that; if the new prediction matches either one of the previous predictions, this becomes the new prediction, and the application will respond to whichever scenario has been predicted. If none of the three predictions matched, then the application will stay with its first prediction until any new prediction matches the prediction that came before it. This is to prevent sudden changes within the data from having two drastic impacts on the prediction accuracy.

4.4.13 Changing filename and location

Changing the filename and location occurs twice in the application. When training data has been inputted into the system, and the classifiers have been trained on said data, the data set files then moved to a separate processed data directory, and the checked is appended to the first part of the filename. This process also occurs with any data placed into the new data file. Its location is changed to the process data directory and tested in front of the data filename.

4.4.14 Retraining of classifiers

Unfortunately, most of the classifiers being used cannot be trained on or updated with new training data; instead, what has to happen is an entirely new classifier is built with all the previous training data and the new training data you wish to update the classifiers with appended to the end of that data set. In the application, this is done by appending new training data instances to the all data file, the file used to initially train the classifiers initially, and then once the end of the testing has occurred, all classifiers are updated using all data files. Once the classifiers have been updated, they are then are realised out of memory to save space.

4.5 TOOL SNAPSHOTS

Below is the startup screen for when the user initially starts the tool

```
|##---##| START       :    STARTING TOOL X

Welcome to welcome Tool X created by Kyle Swire Thompson
which mode would you like to run Tool X in?

MODE 1
Mode 1 is the live mode of the tool. If you select this mode, the tool will process the data within the import file and will then wait for new live data to be added into the new data folder.
MODE 2
Mode 2 is the evaluation model of this tool. This evaluation model will train the classifiers and then evaluate all different prediction methodologies available, producing a percentage of the correct correlation result

|##---##| ERROR     :   more than 1 file dataset has been found
|##---##| FILE IMPORT   :   file  data1 Sampled Scenarios.arff has passed checking
|##---##| DATA FILE :       ALL DATA file has been created
|##---##| SYSTEM INFO  : Fetching all data
|##---##| DATA FILE :       ALL DATA file has been created
|##---##| CLASSIFIER    :    Creating IB1 Classifier
|##---##| CLASSIFIER    :    Creating IBk Classifier
|##---##| CLASSIFIER    :    Creating RandomForest Classifier
|##---##| CLASSIFIER    :    Creating lmt.LogisticBase Classifier
```

Figure 10. tool screenshot showing start up

```
def detailed_senario_responce():
    print("|##---##| |##---##| |##---##| |##---##| |##---##| |##---##| |##---##| |##---##| |##---##| |##---##| |##---##| |##---##| |##---##| |##---##| |##---##| |##---##| |##---##| ")
    print("")
    if senario == 36:
        print("|##---##| STATUS UPDATE: NORMAL OPERATIONS")
        print("|##---##| SYSTEM INFO  : Regular load changes detected")
        print()
        print("|##---##| Security STATUS     : There is NO immediate security risk to the smart grid ")
    if senario in (0, 1, 2, 3, 4, 5):
        print("|##---##| STATUS CONFORMATION:   NATURAL EVENT DETECTED")
        print("|##---##| NATURAL EVENT INFO :    SLG fault")
        if senario == 0:
            print("|##---##| NATURAL EVENT INFO :    A single line to ground fault (SLG FAULT) has been detected at 10-19 percent on LINE 1")
        if senario == 1:
            print("|##---##| NATURAL EVENT INFO :    A single line to ground fault (SLG FAULT) has been detected at 20-79 percent on LINE 1")
        if senario == 2:
            print("|##---##| NATURAL EVENT INFO :    A single line to ground fault (SLG FAULT) has been detected at 80-90 percent on LINE 1")
        if senario == 3:
            print("|##---##| NATURAL EVENT INFO :    A single line to ground fault (SLG FAULT) has been detected at 10-19 percent on LINE 2")
        if senario == 4:
            print("|##---##| NATURAL EVENT INFO :    A single line to ground fault (SLG FAULT) has been detected at 20-79 percent on LINE 2")
        if senario == 5:
            print("|##---##| NATURAL EVENT INFO :    A single line to ground fault (SLG FAULT) has been detected at 80-90 percent on LINE 2")
        print("|##---##| NATURAL EVENT INFO : This fault occurs on a transmission line when one conductor drops to the ground or has come into contact with the neutral conductor.")
        print("|##---##| NATURAL EVENT INFO : This fault may be happening for various reasons such as high wind speed, lightning or even debry falling.")
        print()
        print("|##---##| WARNING ADVICE     : The line must be physically checked by qualified personnel for obstructions in order to clear error. ")
        print("|##---##| WARNING ADVICE     : It is advised that standard tests be performed through the affected line before the power supply is resumed. ")
        print()
        print("|##---##| Security STATUS     : There is NO immediate security risk to the smart grid ")
```

Figure 11. Break down off scenario response for scenarios 36 and 1-6.

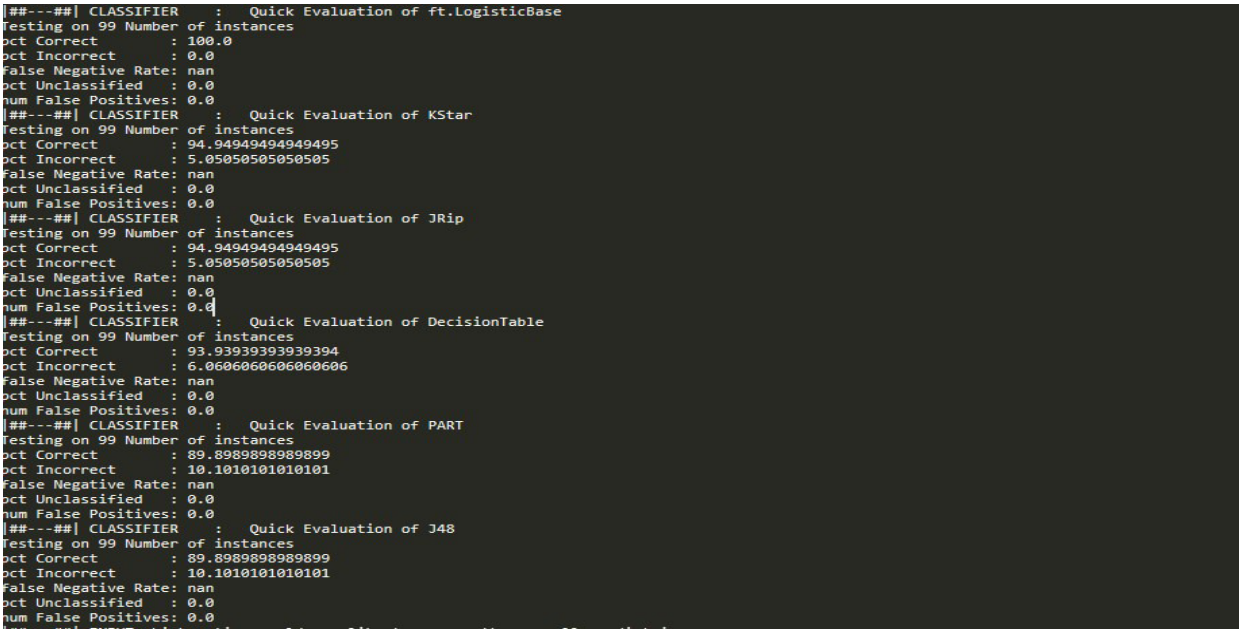


Figure 12. Results from the low-level classifier evaluation

4.6 POSSIBLE CLASSIFIERS

Through WEKA, numerous machine learning algorithms could be used. These classifiers come from many different types. The list of classifiers tested include:

IB1 ,IBk, J48, PART, RandomForest,KStar, lmt.LogisticBase, ft.LogisticBase,JRip, DecisionTable,VFI, RandomTree, LADTree, NNge, ClassificationViaClustering, DecisionStump, ConjunctiveRule, REPTree, ZeroR, HyperPipes, LibLINEAR , OneR

As there is such an extensive list, it will need to be cut down to an appropriate and manageable amount. The classifiers that will be chosen will produce the best results based on initial testing.

4.7 BASIC IMPLEMENTATION

The basic implementation of using a classifier to produce predictions based on previous data is as follows. First, the data needs to be imported from an external source; this can be done using built-in features from either Python or WEKA. Once this data has been gathered, It must then be filtered using attribute selection as previously described. A classifier is then created from the extensive list stored in the library. The filtered data is then used to build the classifier. A trained classifier is obtained to make predictions on either individual instances or multiple instances within a data set. These predictions can then be passed to a separate script, which generates a report and advice based on the predicted scenario. This report is then passed on to the user to inform their decisions about the smart grid.

4.8 DISCUSSION

4.8.1 Sparse data

Sparse data is when the vast majority of entries for an individual attribute ah 0 (zero), and the attribute is therefore sparsely populated with non zero data. This data is useless to machine learning algorithms and can hinder the result. When investigating our data, only the last few attributes would be classed as sparsely populated and therefore should be removed. The implementation of the attribute selection algorithms means that these attributes will always be removed upon evaluation by the attribute selection function. This means we do not need to influence any extra functionality to overcome this.

4.8.2 Discrediting attributes

Discretetising attributes transform numeric attributes to nominal, meaning that all possible values that the attribute could be are predetermined. One would want to do this as some classification algorithms cannot handle numeric classes or produce a better result. Doing this will produce a more comprehensible model, such as a more straightforward decision tree. This, unfortunately, cannot be applied to data as there is far too much of it to do this process. The amount of computing power required to do this would be impractical, and the classifiers would have to be retrained every time a new possible value is discovered. This would also be impossible to do with our testing functionality as we do not know what variables will be produced when gathering the information from the smart grid, and therefore we cannot predetermine those possible values.

4.8.3 Classification selection

Initially, all possible classifiers were considered. The first step in this selection process was to remove all classifiers that could not process the data due to their inability to handle numeric attributes or multi-value nominal classes, or multi-valued nominal attributes. Some classifiers were removed due to their inability to process the data set within a reasonable amount of time. Due to the nature of this tool, any classifier used needs to be able to produce an accurate result within a reasonable time frame as not to create a undo delay between the data being recorded and the results being published. Any classifiers that can process the data would be passed on to be tested.

4.8.4 Sensitivity reduction

What was found while testing the application was that it would very quickly change between scenarios based on its prediction, which is not necessarily realistic to how the data would be coming in. natural variation within data would sometimes cause the predictions to change for a couple of instances but then change back to normal. For instant, the predictions from a classifier could look like this while experiencing one of these episodes :

1	1	1	21	24	23	1	1	1	1

Once the sensitivity reduction functionality has processed these results. the predictions that will be produced look like the following:

1	1	1	1	1	1	1	1	1	1

This type of sensitivity reduction does need to be correctly balanced with the probability that an attack may be occurring and may only happen for a few short instances. If the prediction changes and then held at a constant level in the application, then the prediction produced by the application will change to reflect the results correctly. This is done speedily to not negatively impact the timely response that the application is meant to have.

4.8.5 Removal of the graphic user interface

When developing this tool, it was discovered that an individual classifier used up a significant amount of memory when trained on the majority of data and employing all improvement techniques. Simultaneously, using multiple classifiers, even when serialised and deserialized as appropriate, would require excessive memory, especially when running all improvements and all tool functionality, most notably the graphic user interface. With this considered, it was decided to remove the graphic user interface from the project to minimise the amount of memory the application uses. The resources available to this application must be prioritised appropriately to produce the most accurate result within a reasonable time frame. The application must also run in the background of the control panel personal computer held in the smart grid's control room. The application should not hinder the performance of this personal computer in any way, as any diminishment in the performance of this computer could cascade out and affect the rest of the smart grid. With this considered, there is now a need to minimise the number of inputs required as they will be done through the command input window.

4.8.6 Clustering

Associators and clusters are two other methods to use machine learning algorithms to create predictions on data. Clustering is an unsupervised machine learning task. This means it does not take the class attribute used to define what scenario the instance belongs to into account when processing the data. It involves automatic discovering natural grouping within the data. Clustering algorithms only interpret the data and find where it naturally groups together according to its attribute values. These different clusters would be used to represent different types of scenarios. It was chosen not to use clustering in the project as it would not produce an accurate enough prediction from the given data. It is better at finding large patterns in large amounts of data as opposed to specific scenarios within data.

4.9 SUMMARY

In this section, the different elements of the tool have been explained alongside how it will be integrated into the tool. From the information in this section, one should have a good idea of the different aspects of this project and how this paper will achieve these goals.

5 RESULTS AND EVALUATION

5.1 OVERVIEW

5.2 DATASET AND PRE-PROCESSING

Having a thorough understanding of the data set that we will use to test and evaluate all the different elements of this project is very important. This understanding will enable us to understand better and judge the classifiers' performance and understand any downfalls that may appear. Here will be highlighted the relevant elements of the dataset to the results and evaluation in this section.

5.2.1 Understanding the data set

The data set is made up of 37 different unique scenarios. These 37 unique scenarios are then subdivided into three separate sections: natural event scenarios, no event scenarios and attack event scenarios. These separate different scenario types (excluding no event scenarios) are then subdivided into subtypes.

Natural events are split into two different subtypes of scenarios. The first type of natural event is a single line to ground fault. This occurs on a transmission line when one conductor drops to the ground or accidentally comes in contact with a neutral conductor. These types of faults may occur on power systems due to not actual events such as high-speed wind, a falling tree damaging the line or even lightning. The next type of natural event is line maintenance. Line maintenance occurs when scheduled, and planned maintenance must occur to keep it in working order. This event will produce some faults but is to be expected by the user.

Attack event scenarios are split into three main subtypes. The 1st is a relay setting change; this occurs when the attacker changes the setting distance of a protection scheme on the relay to no longer trip for any valid fault or command. The second of these attacks is a remote trip command injection. This occurs when an attacker sends commands to a relay which causes the Breakers to open. The final type of attack is a data injection. Data injection occurs when an attacker deliberately changes the value such as current oh voltage to imitate correct faults, with the goal being too blind an operator and cause a blackout. This is the type of attack that we have to be most aware of, as it directly interferes with the data we rely on.

5.2.2 Pre-processing and attribute selection

As previously stated, the data set will go through some pre-processing and attribute selection to determine the best possible combination of attributes to yield the best results. Attribute selection occurs on the initial training data set and will change based on the data given to the algorithms for training. This is so the best possible result is always reached. If the attributes were to stay the same awesome fixed philtre were to be applied, which produces the best attribute combination when using all the data sets available, it would perform poorly when classifiers are only trained on a small number of datasets due to the attributes selected not correctly reflecting the patterns found within the data.

5.3 Individual classifier Performance

Some of the classifiers were unable to build and process the dates or were rejected for other reasons. The reasons for rejection are shown in the pie chart below.

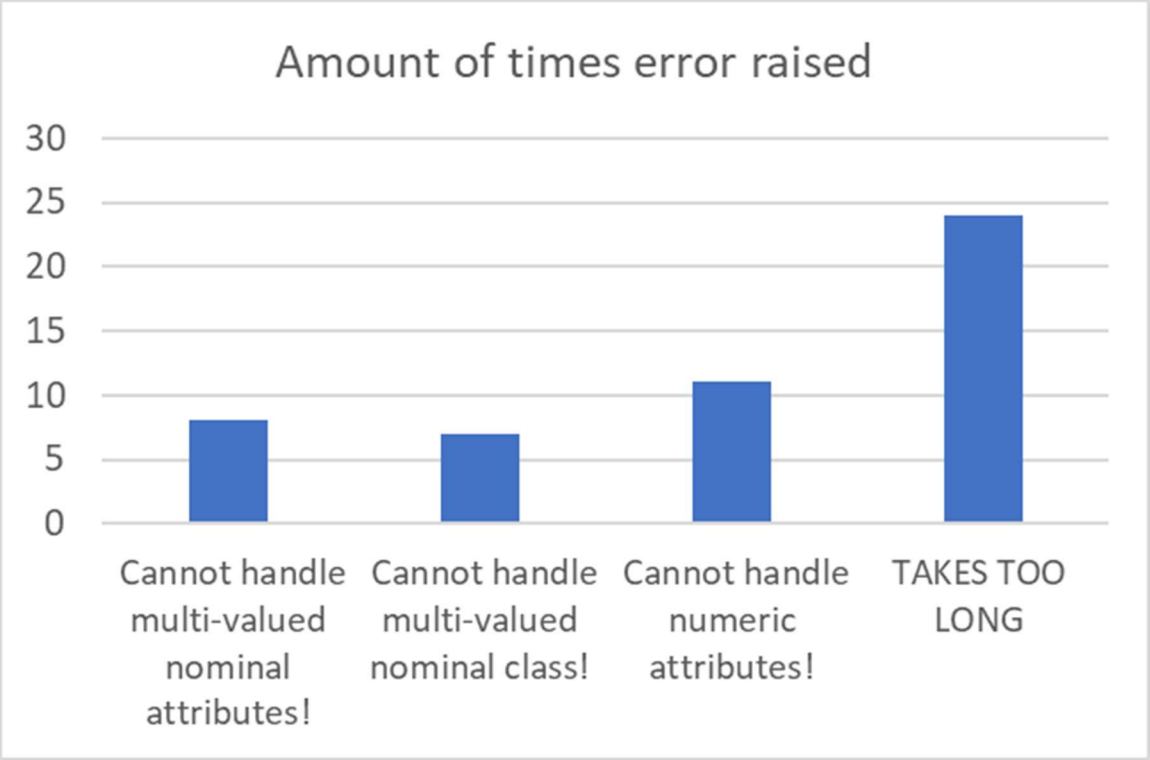


Figure 13. Breakdown of Errors raised from classifiers

The vast majority of classifiers well remove due to their lack of ability to process a relatively small amount of data in a reasonable time. This failure means that they would not be practical for our application. The cut off time was 10 minutes to process 5000 instances of data with an 80 % to 20% training test split

5.3.1 Control data group

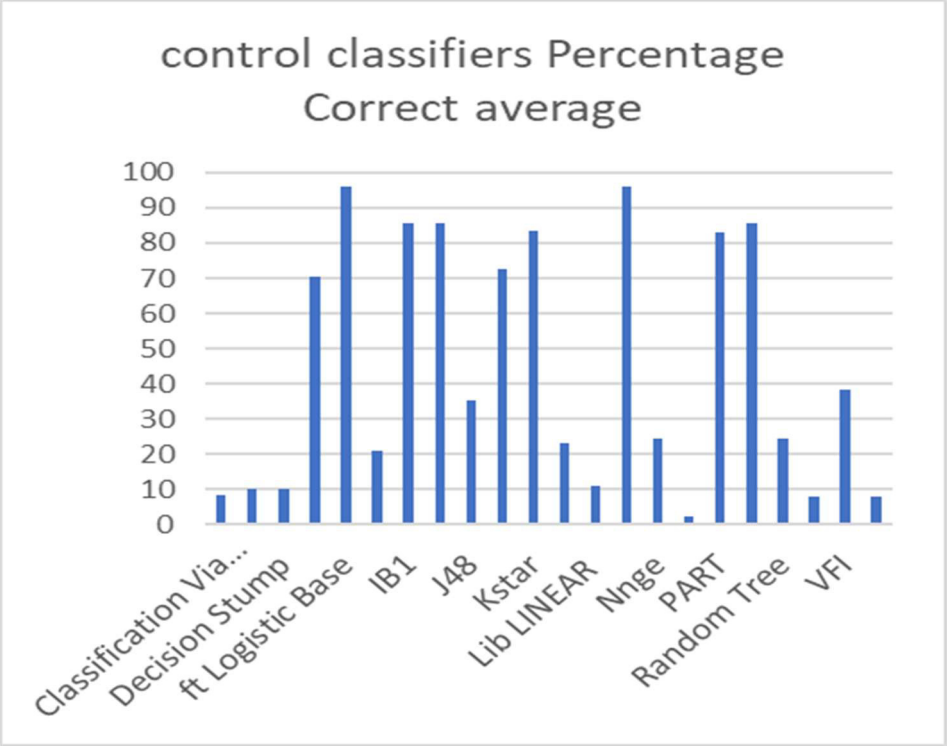


Figure 14. control classifiers Percentage Correct average

This experiment involved using the base classifiers without any modifications to make predictions on a data set that contains no attribute selection. This is the baseline of data, so anything done to the classifiers or data sets needs to improve upon these results. Individual classifier success is based on how many correct predictions they make, which is the core element of this paper. First, the individual classifiers without any attribute selection, parameter optimisation or subsequent improvements from applying meta classifications will be tested. This group of data will be known as the control group. Then this will be compared to the data produced after applying each improvement method. As shown in the data, two classification algorithms performed exceptionally well: the LMT logic base and the FT logic base. The LMT logic base scored 95.948%, and the FT logic base being the highest scoring classifier at 96.226%, following five classifiers achieved above 80% classification. These five classifiers are KStar, PART, random forest, IBk and IB1. Apart from these seven classifiers, only two others managed to achieve above 40%. The decision table achieved just over 70% at 70.479%, and Jrip achieved 72.481%. The 13 other classifiers all failed to achieve classification correctly.

5.4 Base classifier improver Performance

One of the first steps of this project was to select which based classifiers to using the final application. In order to investigate this, all of the possible classifiers were built and using 10-fold cross-validation. Each classifier improvement methodology was evaluated against each data set, and The results were averaged and shown below. Each subsection displays each of the base classifier improvements' results.

5.4.1 Attribute selection group

This experiment involved applying only the attribute selection to the data set and nothing to the classifier. This attribute selection should help focus the classifiers efforts and create a clearer picture for the base classifier to create a model based on. Some classifiers have internal functionality that performs a similar task, so little to no effect may be seen on some classifiers. After extensive experimentation evaluating different combinations of evaluation and search algorithms, it was concluded that the combination of Consistency Subset Eval for the evaluation algorithm and the genetic search for the search algorithm produced the best results across all the classifiers. As such, this will be the combination of algorithms that are used in this experiment. Each classifier will be compared against each other based on the average percentage correct when evaluating each using ten-fold cross-validation.

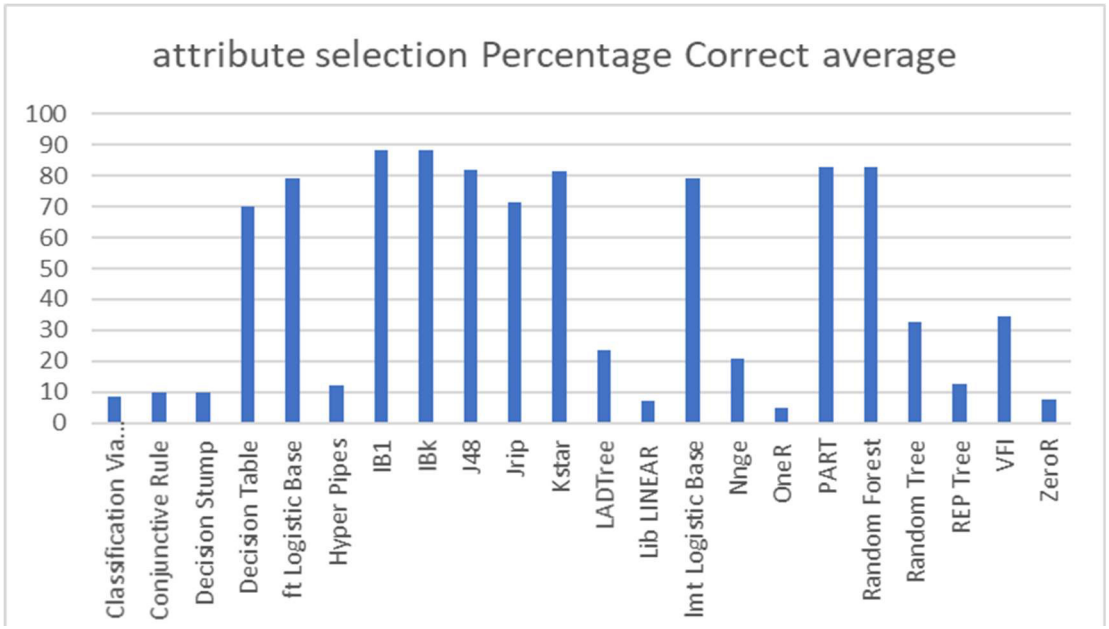


Figure 15. classification Percentage correct avg. w/ att. Selection using ConsistencySubsetEval & GeneticSearch

In the results shown above, there is a clear distinction between two groups of classifiers. The first group includes very popular classifiers such as decision stump, lad tree, and RPE tree, performing very poorly when attempting to classify a data set that has had attribute selection performed on it. The two most successful classifiers are IBK and IB1, which classify instances at a success rate of 88.362%. Four other classifiers managed to achieve above 80% classification. These classifiers include J48, which achieved 81.929% classification, KStar, which achieved 81.54897% classification; the PART algorithm achieved 82.671% classification. Finally, the random forest classifier managed to achieve 83.0112% correct classification. Beyond these previously mentioned classifiers, there were four that managed to achieve high levels of classification. The logistic base classifiers managed to achieve 79%, followed by JRIP, which achieved just over 70% at 71.299%. The classifier that performed the worst but still managed to classify with some certainty is the decision table that did not quite manage to achieve 70%, but an edge case could be added to this group. The decision table managed to achieve 69.886% correct classification.

5.4.2 Bootstrap aggregating results

This experiment involved applying only the bootstrap aggregation to the base classifier and recording the average percentage correct produced by each base classifier.

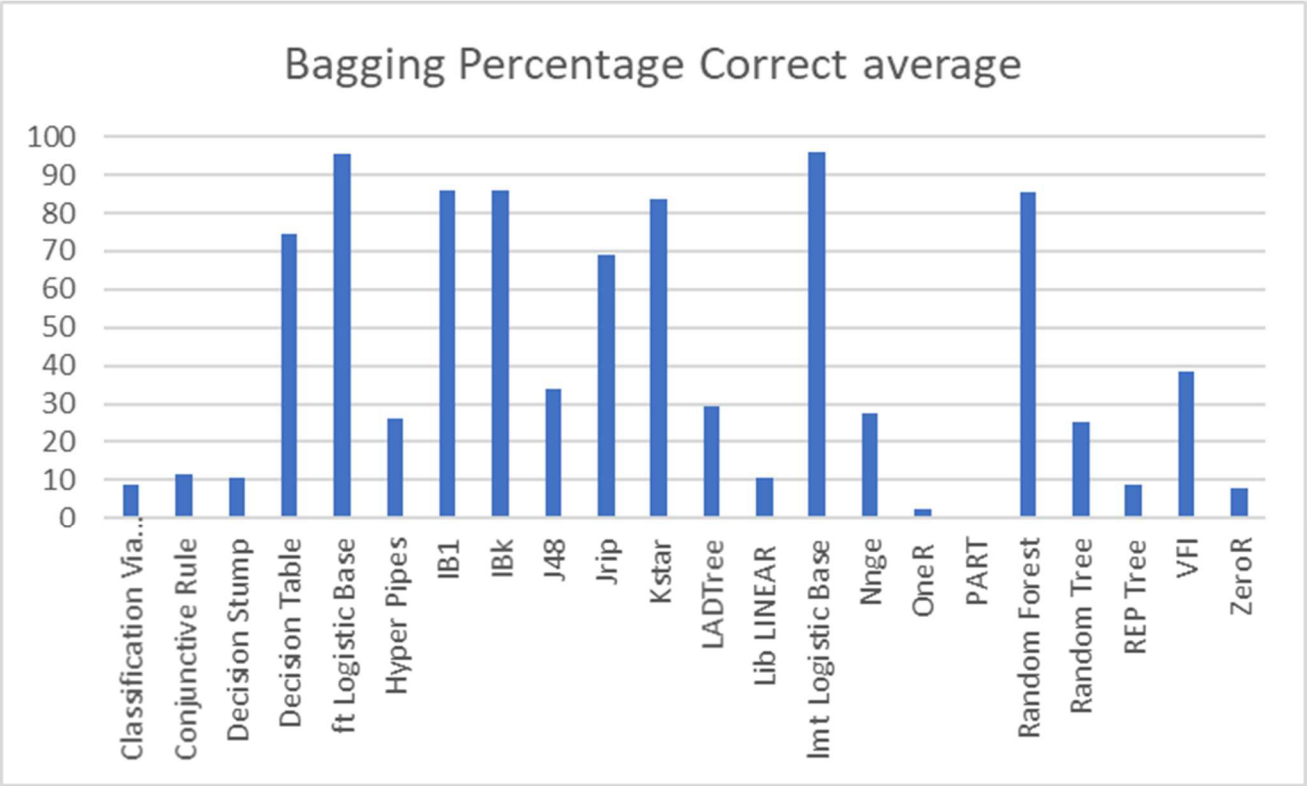


Figure 16. Bootstrap aggregating average results

As shown in the figure above, eight classifiers perform well while using bootstrap aggregation. The best performing classifier was the LMT logistic base, which correctly classified 95.993% of the time. The next classifier to perform well belongs to the same family and is the FT logistics base which also scored above 95% and recorded correct results of 95.436%. These two classifiers are clustered together as the two best performing classifiers. After a small gap in performance at approximately 85% comes the next cluster of classifiers. There are four classifiers within this cluster of performance, and they are the IBK and IB1 classifiers, both managing to score 85.704%. Also in this group is the random forest classifier which also managed to classify at 85%, precisely 85.474%. The final classifier in this group is the lower bound of this group and has a 1% drop in performance than the other classifiers. This is the KStar classifier that manages to achieve 83.792% correct classification. The only other classifier to perform moderately well in this group is the JRIP classifier which, while failing to achieve 70% classification, does get close with 68.813%. There are two other clusters of classifier results, but these do not perform well with bootstrap aggregation, and all failed to achieve above 40% classification. The only classifier to get closed is the VFI classifier, which manages to achieve 38.667% classification. Notably missing from the results is the PART classifier. When testing this classifier, it required over 25 gigabytes of memory to perform classification on one data set and, as such, is impractical for our purposes, so it has been left out of the results.

5.4.3 adaptive boosting group

This experiment involved only applying the adaptive boost to the base classifier and recording the average percentage correct produced by each base classifier.

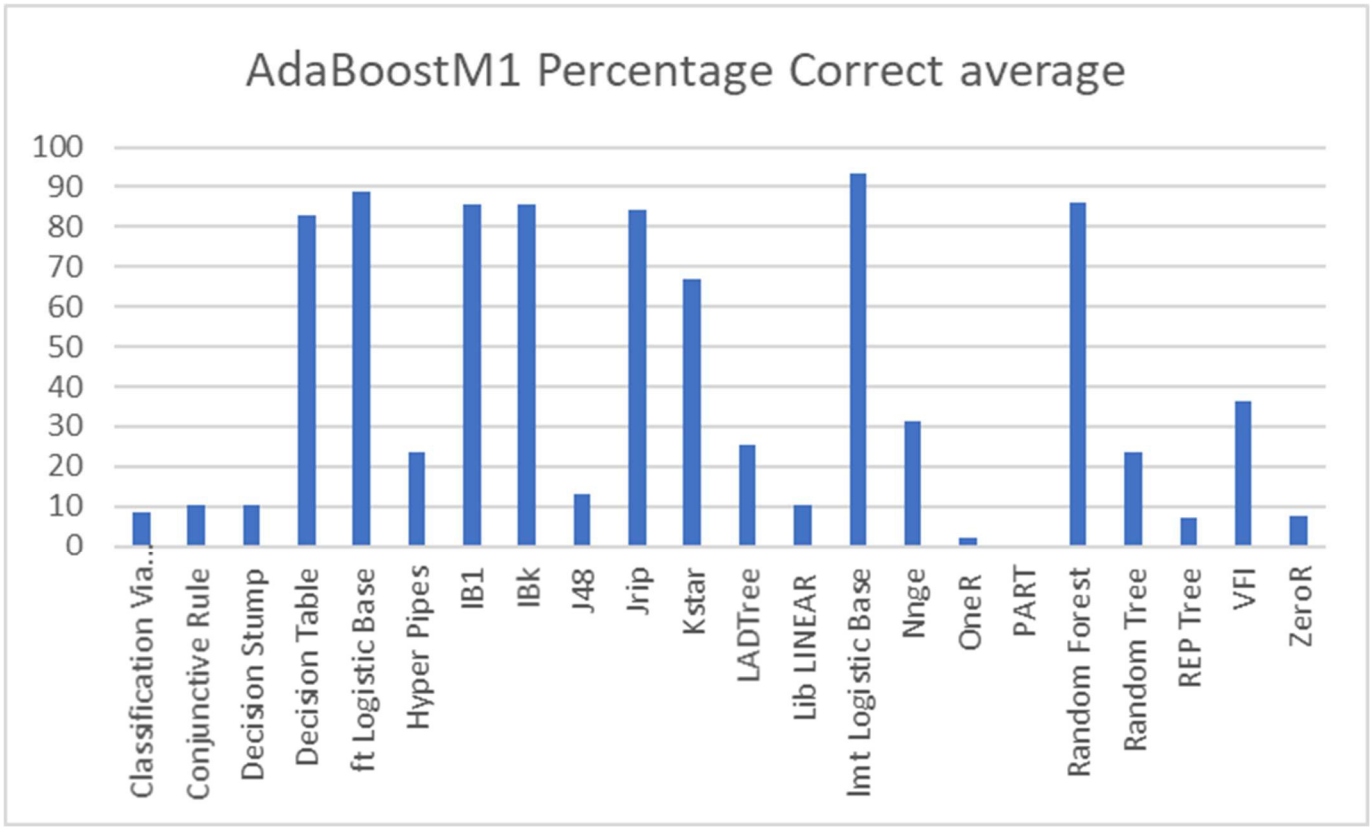


Figure 17. adaptive boosting correct classification average

Within the data, seven classifiers managed to achieve above 80% correct classification. The best classifier among these was the LMT logistic base, which correctly classified 93.584% of the time. This is the only classifier that managed to score above 90%. The classifier that came second is the FT logistic base, which achieved 89.065% correct classification. The subsequent five classifiers all managed to score between 82.97% and 85.807. These five classifiers include decision table, IB1, IBK, JRIP and random forest. There is then a sharp decline in performance before Kstar is reached as it managed to achieve 67.081% correct classification. This is the final classifier to manage to score above 40%. Notably, the PART classifier is missing from this test, too, as it was to memory intensive to complete the experiment.

5.4.4 Delay based single classifier enhancer

The Delay based single classifier enhancer is challenging to test. The nature of the methodology requires test data to be in the order of collection so that a realistic stream of data can be processed. If the data received by this methodology is randomized, there is no pattern to the data, and therefore, no benefit can be gained from making predictions based on previous results. The problem occurs when trying to get this test data. In order to get this test data, we would need to split the data set used at some point to create the test and training data sets to use. If we split the training and test data sets before randomization, then the patterns and correlations that the machine learning algorithms required to make accurate predictions are lost. The randomisation process helps to preserve these patterns and correlations, so removing some instances to test against the classifiers does not affect the classification results as heavily. With a fully randomised data set, the delay prediction voting methodology is next to useless. When performing the most common voting methodology experiment, the delay prediction voting methodology was tested alongside it with a randomized data set from the start. The most common voting methodology scored 93.944% correct classification, whereas the delay prediction voting methodology only managed to score 9.306% correct classification even though it was receiving its predictions from the most common voting methodology. The problem of missing correlations and patterns does not occur if the data set is randomized, as randomization preserves all the patterns that the classifiers rely on, so any splitting done to the data set does not significantly affect the performance of the classifiers. There is likely to be a significant difference in the results between classifiers that perform well and classifiers that perform poorly. As such, both types need to be tested. In order to test the delay prediction voting methodology on classifiers that performed poorly, The last 20% of the data was taken from the data set to use as a test data set. The remaining 80% would then be used to train the classifier and was randomized, whereas the test data set is not. This does mean that the base classifiers will perform worse than if they were tested on a data set that was randomized from the start, but for this experiment, we are testing the effect that the delay voting method has on poor performing base classifiers predictions and how well it either improves or hinders the results. The percentage of improvement the delay prediction voting methodology produced is shown below.

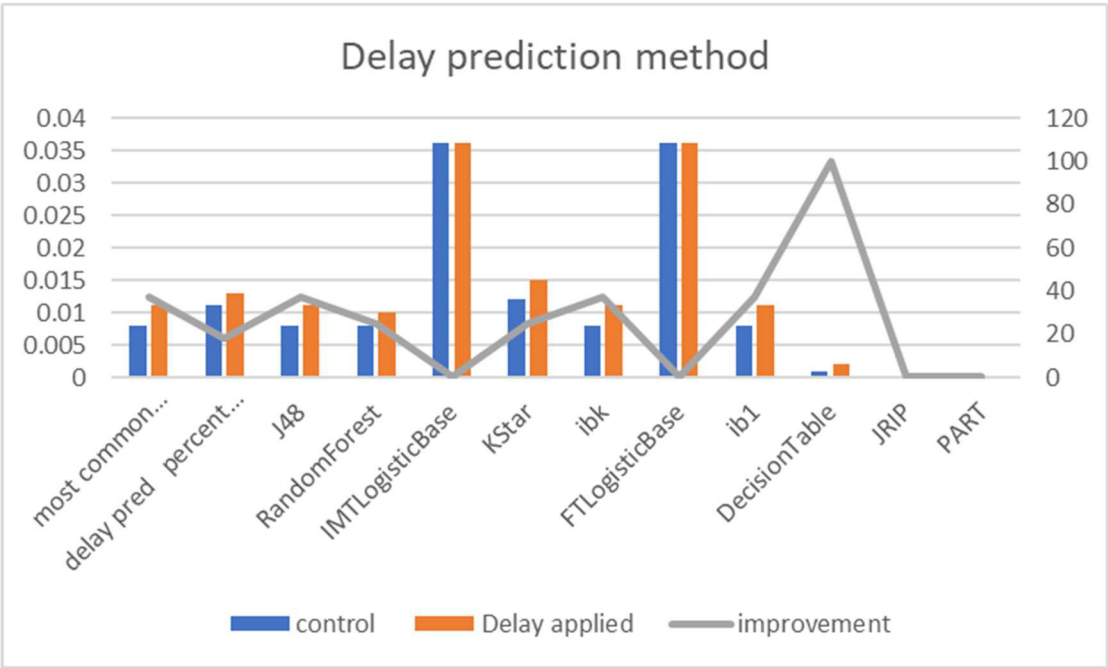


Figure 18. Delay based single classifier enhancer

When looking at the results from this experiment, every single classifier that this method was applied to improved or stayed the same compared to its control result. The most improved classifier was the decision table classifier which showed 100% improvement. Three managed to show 37.5% improvement compared to their control result, those classifiers being J48, IBK, and IB1. The only two other classifiers to show improvements were the random forest classifier and the Kstar classifier, which managed to show 25% improvement. Also included in the experiment was the most common and most common results, plus a double delay. From these results, we can see that with the delay applied, the most common prediction methodology managed to improve by 37.5% and when a double delay was applied improved that result by a further 18.182%.

5.4.5 Combination classifier single classifier enhancer

This experiment involves combining two different META-classifiers of the same type of base classifier, but one uses bootstrap aggregation, and the other uses adaptive boosting.

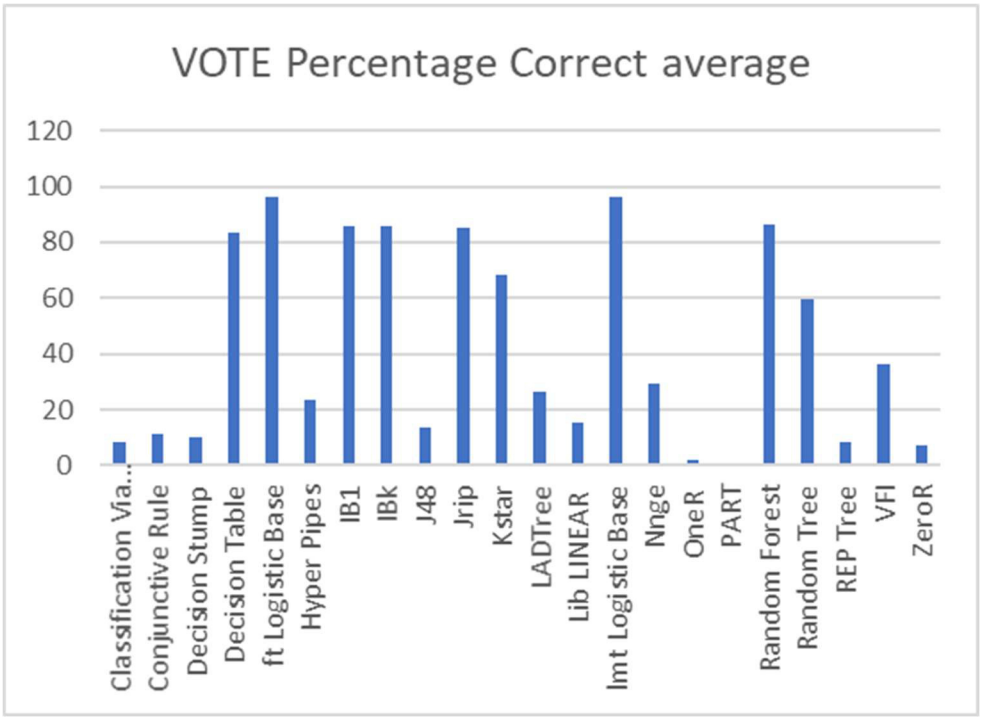


Figure 19. Combination classifier single classifier enhancer results

As shown in the figure above, seven classifiers performed above 80% correct classification while using the vote. The best performing classification algorithm was the LMT logistic base that scored 96.179% correct classification, and the FT logic base managed to score 96.123% correct classification; these were by far the two frontrunners in this experiment. All other classifiers that manage to predict above 70% scored 83% and 86%. Only two other classifiers managed to perform moderately well, with KStar scoring 68.202% correct classification, pushing it just under the 70% threshold and random tree scoring under 60% at 59.529% correct classification. Every other classifier failed to achieve a 40% correct classification.

5.5 VOTING METHODOLOGIES RESULTS

This section shows how well each of the voting methodologies improves on the predictions made by these single classifiers. Like in the previous section, these voting methodologies will be judged on how many correct predictions they make. They will be tested on the same data sets, and the classifiers will be trained on the same data. In order to pass on predictions to the voting methodologies, which will enable them to produce the best results, the classifiers we pick to use must achieve a high level of classification as such is essential to decide at what percentage correct classification should the threshold for inclusion into this group be. An argument could be made that any classifier that achieves above 51% should be included in the group as it will be correct most of the time, and as such, its vote for what the scenario is may have some weight. Well, this argument may have some merit on the surface when diving deeper into which instances are correctly classified. This argument falls apart. If you were to take the instances that a classifier that achieves around 50% correct classification got correct and compare them to the instances that other better-performing classifiers got correct, then you would see that the high-level classifier would have correctly classified most or all of the instances correctly classified by the medium level classifier. With this being considered, any classifier that achieves above 70% correct classification was included in the voting methodologies test.

5.5.1 Most common prediction voting methodology

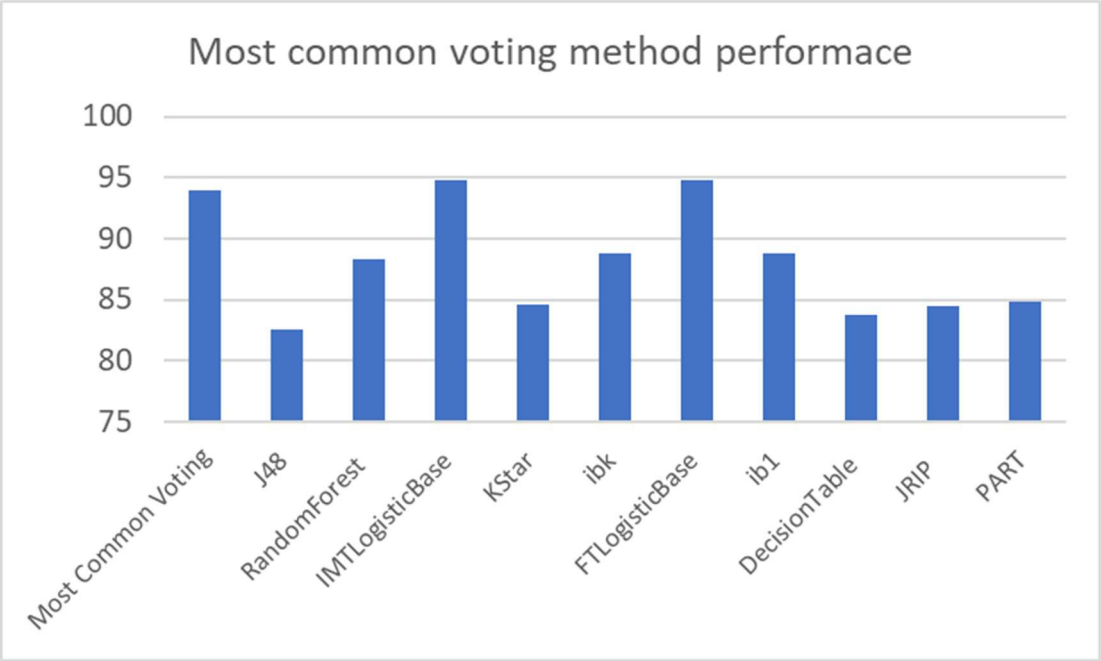


Figure 20. Most common prediction voting methodology results

The most common voting methodology managed to perform better than a large majority of the base classifiers. The only two classifiers to perform better were both types of logistic base classifiers. Both logistic base classifiers managed to classify correctly 94.830% of the time, whereas the most common voting methodology performed just under 1% worse at 93.944% correct classification. The other classifiers all performed under 89% correct classification but over 82% correct classification.

5.5.2 Weighted voting methodology

the weighted voting methodology requires external input in order to be effective. For this voting methodology to produce the correct results, the weights of the individual classifier predictions must be defined beforehand. In order to test this, the weights well assigned based on the control results for each classifier. In order to produce a weight for each of the classifiers, they were placed in order based on the results from the control experiment. To carry out this experiment, weight was applied to each prediction from a base classifier and was used to produce a weighted prediction. The results and weights of this experiment are shown below.

Table 3. classifier weight table

Classifier	**Weight**
J48	1
RandomForest	6
IMTLogisticBase	8
KStar	4
ibk	7
FTLogisticBase	8
ib1	7
DecisionTable	2
JRIP	3
PART	5

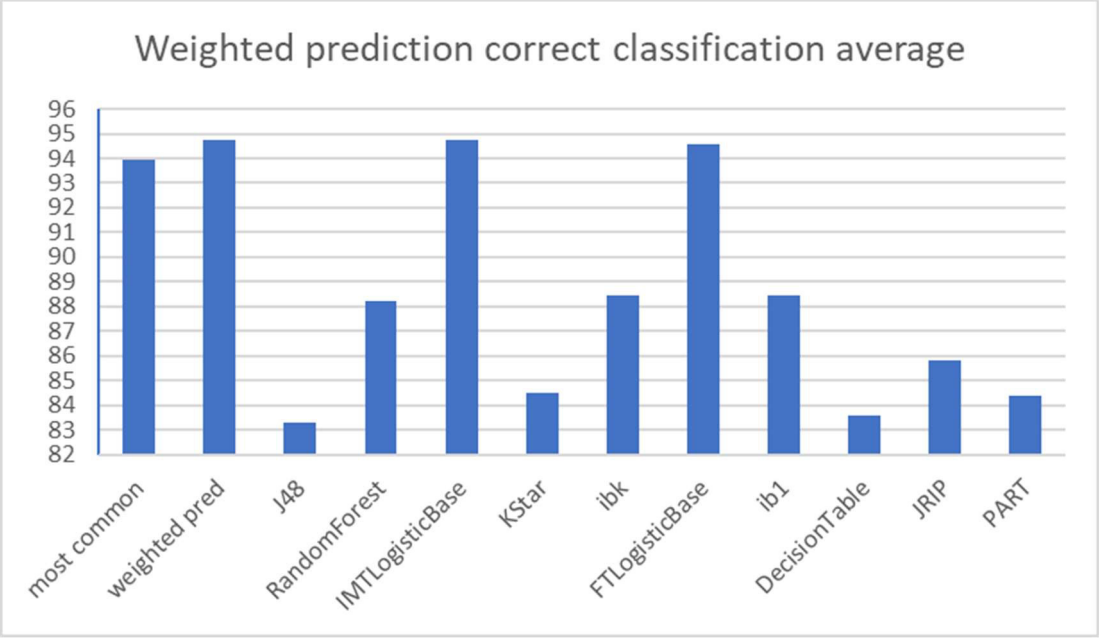


Figure 21. weighted voting methodology average results

When looking at how well the weighted prediction voting methodology performed, we can see that it managed to achieve the joint highest correct classification rate at 94.763%. This weighted prediction correct classification level is the same as both of the logistic base classification algorithms.

5.5.3 Blending voting methodology

In order to test the effectiveness of the blending voting methodology, firstly based classifier is that it will use need to be defined. For this, each of the classifiers needs to be trained on the same data. This means that the three classifiers that would make use of attribute selection cannot if we wish to include them in this voting methodology. So in order to carry out this experiment, all of the ten classifiers chosen to test all the voting methodologies were trained on data that did not have attribute selection applied to and were then added to the blending meta classifier. This blended meta classifier was then tested to produce a correct classification rate. In the figure below, the results are displayed. The single based classifiers that could use attribute selection did make use of attribute selection to produce their results. The most common voting methodology is also included to provide context for the blending voting methodology result.

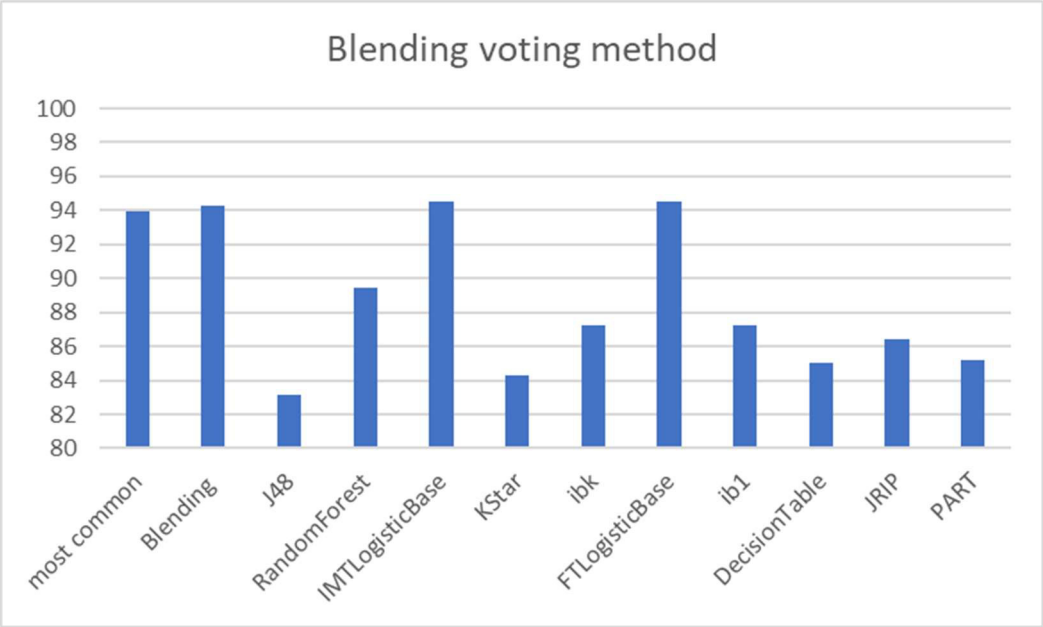


Figure 22. blending voting methodology average result

The blending voting methodology managed to produce a correct classification rate of 94.291%. This result is a 0.38% increase from the most common prediction methodology result. The blending voting methodology did not produce the best classification rate. Both of the logistic base classifiers performed better, with an average of 0.19% difference between the blending voting methodology and the logistic base classifiers.

5.6 SCENARIO DETECTION RESULTS

In this evaluation section, each of the individual scenarios and types of attack they belong to will be evaluated against how often the individual prediction algorithms detect them. For this experiment, if a classifier failed to classify an instance correctly, then the correct scenario value was recorded. This means that any instances that failed to be classified would be recorded. As it is always recorded if a classifier fails to classify the instance. the results will also reflect how well each of the scenarios performed across all classifications. When conducting this experiment, a critical element that must be noted is the frequencies of the scenarios in the original data set. To ensure that the results of this experiment can be trusted, the results were adjusted to accurately reflect the frequency at which each scenario appeared in the original data set. Scenarios 31 to 34 should be ignored in the data as these scenarios are nonexistent in the data set.

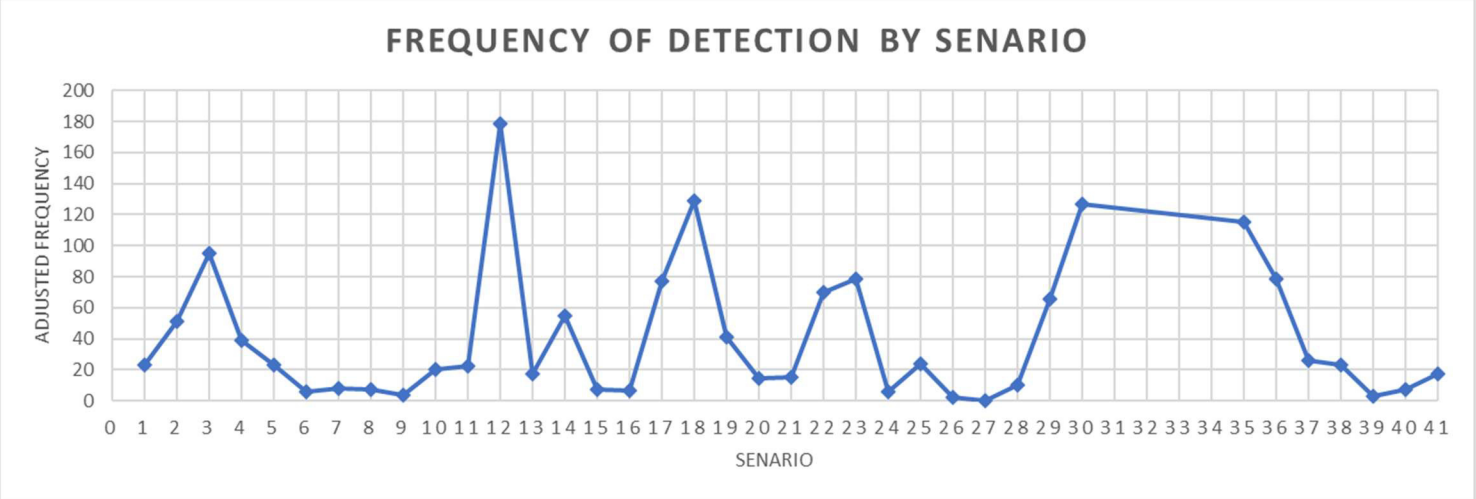


Figure 23. frequency of detection

As shown in the data above, the classifiers failed to classify accurately five types of scenarios. The scenario that was predicted the least was scenario 12. This scenario is an attack-type scenario that simultaneously uses a trip command data injection with an SLG fault replay at 80%-90% on line two. However, this result seems to be a bit of an anomaly as it is the only result where the rest of the scenarios in that type of scenario are detected very often. The scenario that was predicted least often next is scenario 18. Like scenario 12, scenario 18 two is an attack-type, precisely a remote trip command injection against relay 4. Notably, the following two scenarios detected the least both are disabling relay functions against either a single or double relay, and both include a fault. The following least detected scenario is scenario 3. Snow yeah three is the first non-attack type to be undetected at a significant rate. Scenario three is an SLG fault from 80 to 90% online one.

5.7 KEY HIGHLIGHTS AND TAKEAWAYS

This section will go over crucial highlights when creating these evaluations and detail any takeaways from these results.

5.7.1 Single classifier improvements

A key element to mention when thinking about the single classifier improvement methodologies is that two methodologies can be applied regardless of whatever other methodology is already acting on the classifier. The attribute selection improvement methodology and the delay-based prediction

improvement methodology can be added to a single base classifier while other single classifier improvement methodologies act on the classifier. This means that any classifier that sees any improvement with these two methods should always have these methods to impact the single classifier positively.

The delay-based prediction method consistently returned a positive impact on any classifier applied to bar two classifiers of the same family. This improvement methodology showed significantly better results with base classifiers that performed poorly in the control test. From this, we can conclude that classifiers that show less correct classification have far more variance in their predictions than consistently getting a prediction wrong. This means that as opposed to consistently getting scenario 7 mixed up with scenario 13, they get scenario seven mixed up with a few different scenarios.

Across all of the improvement methodology, the family of logistic base classifiers was consistently the highest performing. The next to best-performing classifiers were the IB classifiers. These high-performing classifiers often get the same result and show very little difference in the instances they classified correctly.

5.7.2 voting methodologies

The best performing voting methodology was the weighted voting methodology, with weights being applied by order in the control experiment—the correct classification percentage produced by this voting methodology matched up precisely with two of the base classifiers. The voting methodology which came second is the blended voting methodology with no attribute selection applied and blending all ten of the classification algorithms used to test the voting methodology. These two verses methodologies managed to fluctuate above and below the results for the base classifiers, so they are not intrinsically linked to them. The worst performing voting methodology was the most common voting methodology, which still performed better than the single classifiers bar the front running two logistic base classifiers.

5.7.3 Scenario detection

as previously mentioned, there are four clusters of scenarios whose detection rate is significantly lower than the other scenarios and one outlying scenario whose detection rate is significantly lower than every other scenario. When looking at the patterns for the clusters of scenarios with low detection rates, typically, a family of scenarios will perform poorly. Then a couple of unique characteristics found within those families will create a scenario that is the worst performing out of all of the scenarios in the family. Upon further inspection, common patterns can be found amongst these characteristics. These characteristics are listed below in the table. In order to produce these results, the six worst-performing scenarios will be recorded as well as their characteristics.

Table 4. attack characteristics

characteristics	Count
Fault from 80-90%	3
relay disabled & fault	3
Fault from 10-49%	2
Command Injection to R4	1
tripping command	1

5.8 EVALUATION

5.8.1 Single classifiers evaluation

Table 5. CONTROL Percentage Correct average

classifier	CONTROL Percentage Correct average
Classification Via Clustering	8.53565171
Conjunctive Rule	10.2716174
Decision Stump	10.22848282
Decision Table	70.47949084
ft Logistic Base	96.22587015
Hyper Pipes	21.0498755
IB1	85.66941195
IBk	85.66941195
J48	35.42578777
Jrip	72.48050764
Kstar	83.58494328
LADTree	22.96650718
Lib LINEAR	10.85170485
lmt Logistic Base	95.94802631
Nnge	24.58453575
OneR	2.432015324
PART	83.09390818
Random Forest	85.53405739
Random Tree	24.42142602
REP Tree	7.892782659
VFI	38.29078731
ZeroR	7.770055399

Looking at the table above, any of the highlighted green results indicate that this classifier achieved correct classification at a high enough rate that it can be used in other prediction methodologies as a reliable source of data. This group included nine classifiers. After these nine classifiers, there is then a wide gap in performance before two classifiers are reached. These classifiers have been highlighted as orange as they show the potential to achieve high levels of classification if the correct single classifier enhancers are added but do not show a high enough correct classification percentage to be included in the first group. The final and most prominent group is the classifiers that failed to achieve confidence in their classifications. This included eleven classifiers, and unless they show surprise improvement from the individual classifier, they will be excluded from further experimentation.

5.8.1.1 Classifier type evaluation

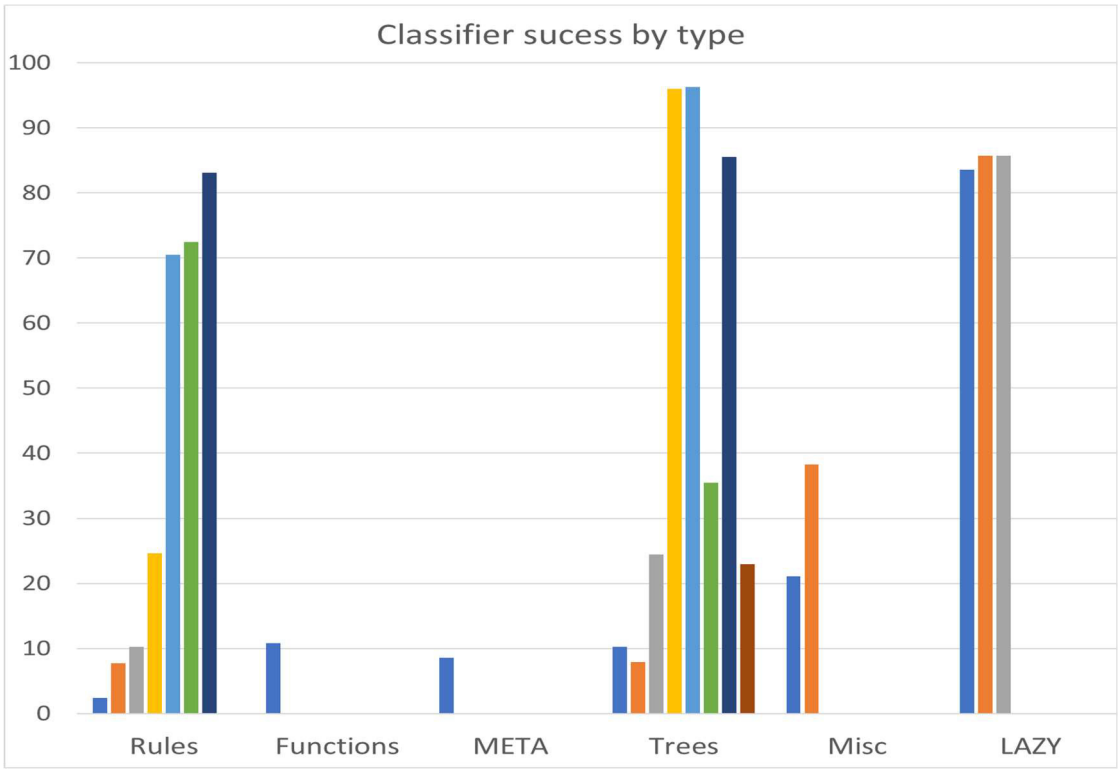


Figure 24. Classifier type evaluation

A clear trend emerges when looking at the types of classifiers that perform well in the control experiment. Three types of classifiers can perform well within this context: rules, trees, and lazy classifiers. This success does not guarantee that each classifier performs well, just that it belongs to a family that can perform well. All three of the tested lazy classifiers performed exceptionally well, all achieving over 80% correct classification. The best performing classifiers belong to the tree family of classifiers achieving a high of 85.7%. However, 5 of the tree type classifiers failed to achieve 40% correct classification proving that success in this context is not based on classifier type. When looking at the rules type classifiers, three perform well, with only one achieving above 80% but all three achieving above 70%. The classification types that performed poorly word the function types, meta types and Misc types. These failed to reach 40%, with the functions and meta classifier failing to classify above 11% correctly. These results are shown in the bar chart below.

5.8.2 Individual classifier improver

5.8.2.1 attribute selection

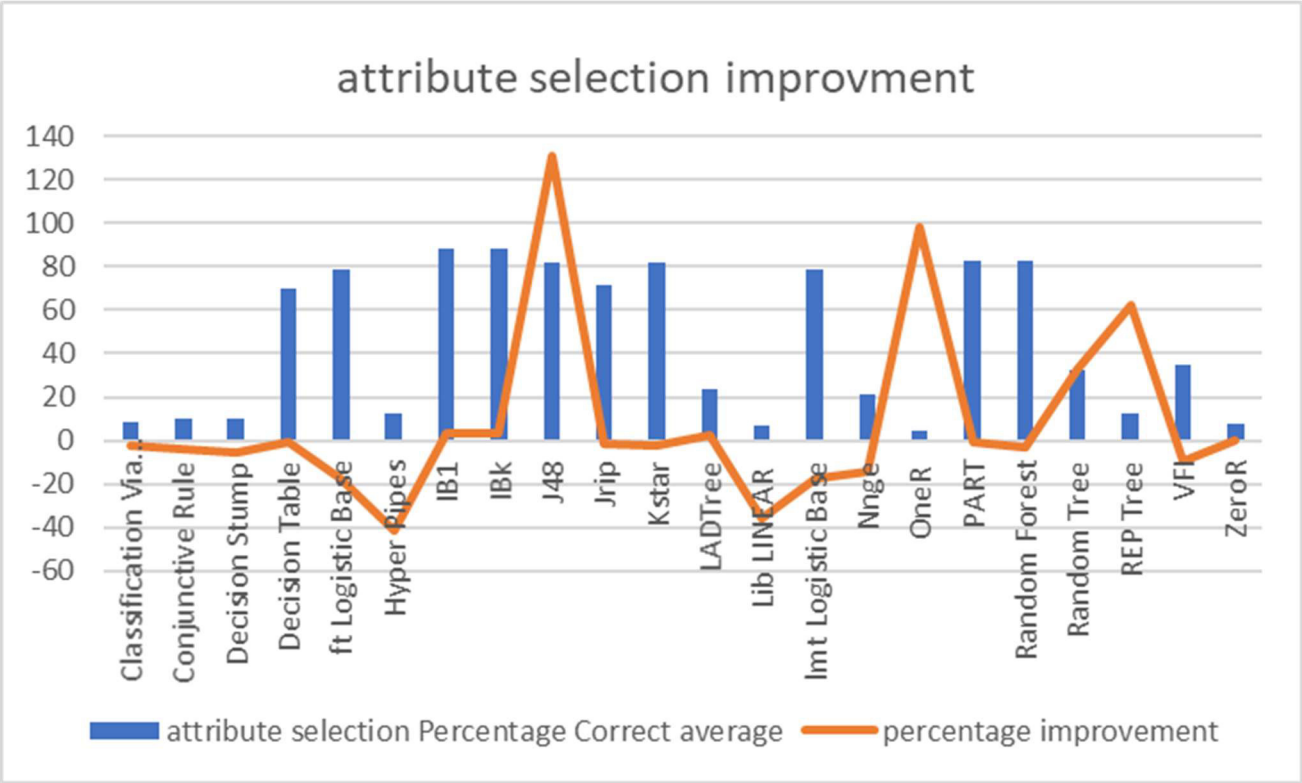


Figure 25. attribute selection improment

In the figure above, the percentage of improvement attribute selection for each classifier's improvement is plotted against the classifier's performance. In the results, we can see that attribute selection for classifiers helped improve their classification rate for some classifiers. The most successful of these is the J48 classifier. Attribute selection applied to this classifier managed to improve from the control results 131.629%. This statistical improvement alone is why J48 is now an extremely reliable classifier Managing to achieve 81.929% correct classification. Attribute selection did not help all classifiers it was applied to. Both of the logistic base classifiers showed a 17.651% reduction in correct classification when using attribute selection. The most successful classifiers did show some improvement with attribute selection, but this was only small at 3.143% but did mean that they could achieve 88.36 2% classification but as this short and the time is taken to process the data set, it is worth applying to every classifier that shows any improvement with it.

5.8.2.2 Bootstrap aggregating

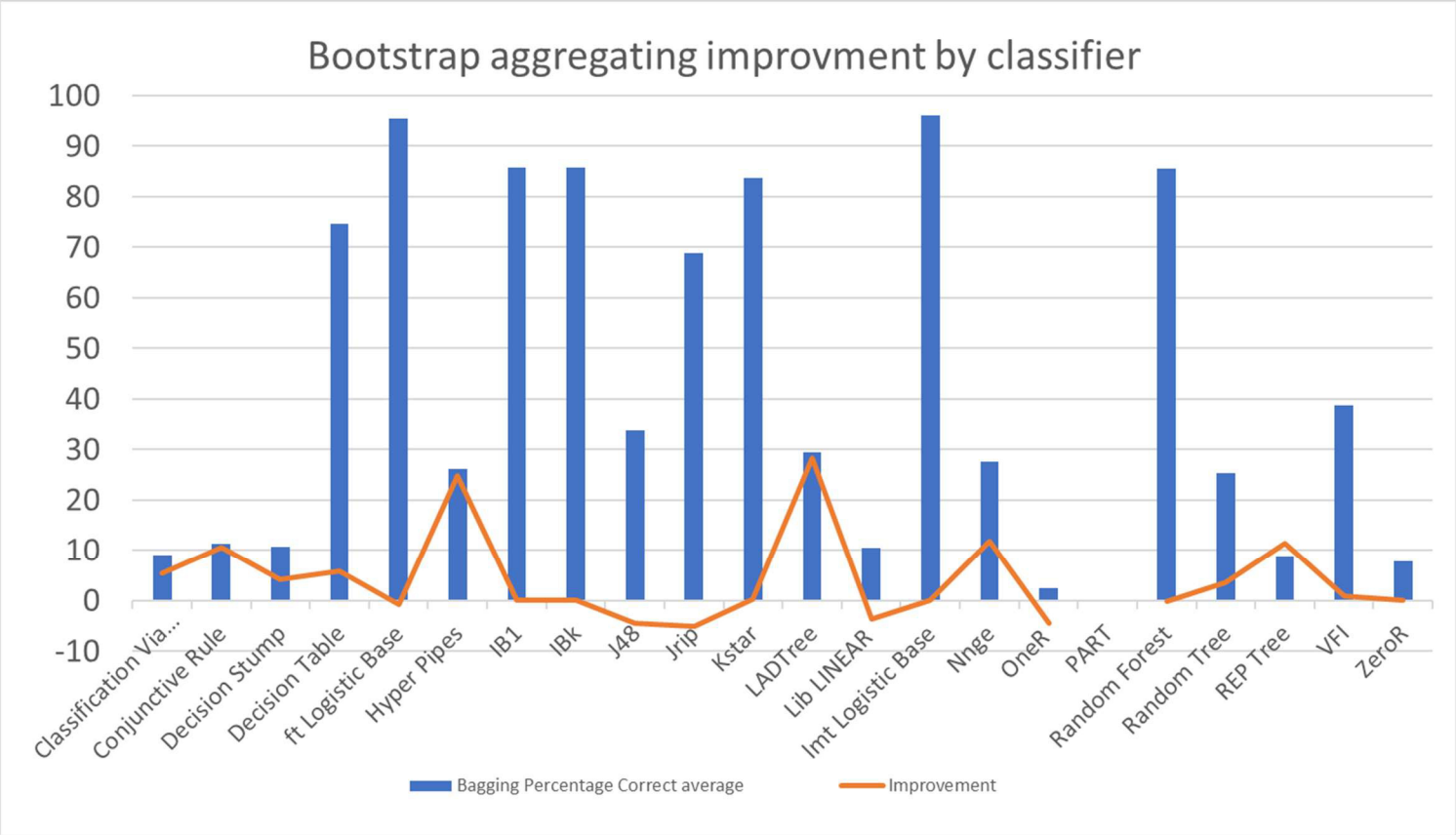


Figure 26. Bootstrap aggregating improvment

When comparing the results from the control experiment against the bootstrap aggregation experiment, only three base classifiers see any significant improvement when using bootstrap aggregation. The best of these is the LADTree classifier which sees a 28% increase in performance over its control results, leaving the classifier with a total of 29.453% correct classification. The next classifier that shows the most improvement is the hyper pipes classifier, which shows an improvement of 24.7% from its control results. The only other classifier to see significant improvement is the RPE tree, which only saw 11% improvement, meaning that it was left with 8.784% correct classification. It is noticeable as the only classifier where the amount of improvement exceeds the percentage of correct classification. For the vast majority of the classified as tested, this does not improve and, for some, hinders the results, most notably the JRIP classifier. The decision table classifier is the best performing classifier that manages is to make any use of bootstrap aggregation. It is already a relatively good classifier, but with the bootstrap aggregation increasing its previous results by 5.9%, it manages to increase the correct classification for the base classifier to 74.64%, which manages to push it past the 70% threshold. The highest performing classifiers, such as both logistic bases and both types of IB classifiers, see no improvement with bootstrap aggregation, and there is only some very minor variance in the results due to shuffling the data before each test. Bootstrap aggregation should, therefore, only be applied to the decision table as it was the only classifier that showed significant improvement while using bootstrap aggregation and managed to achieve above 70% classification.

5.8.2.3 Adaptive boosting

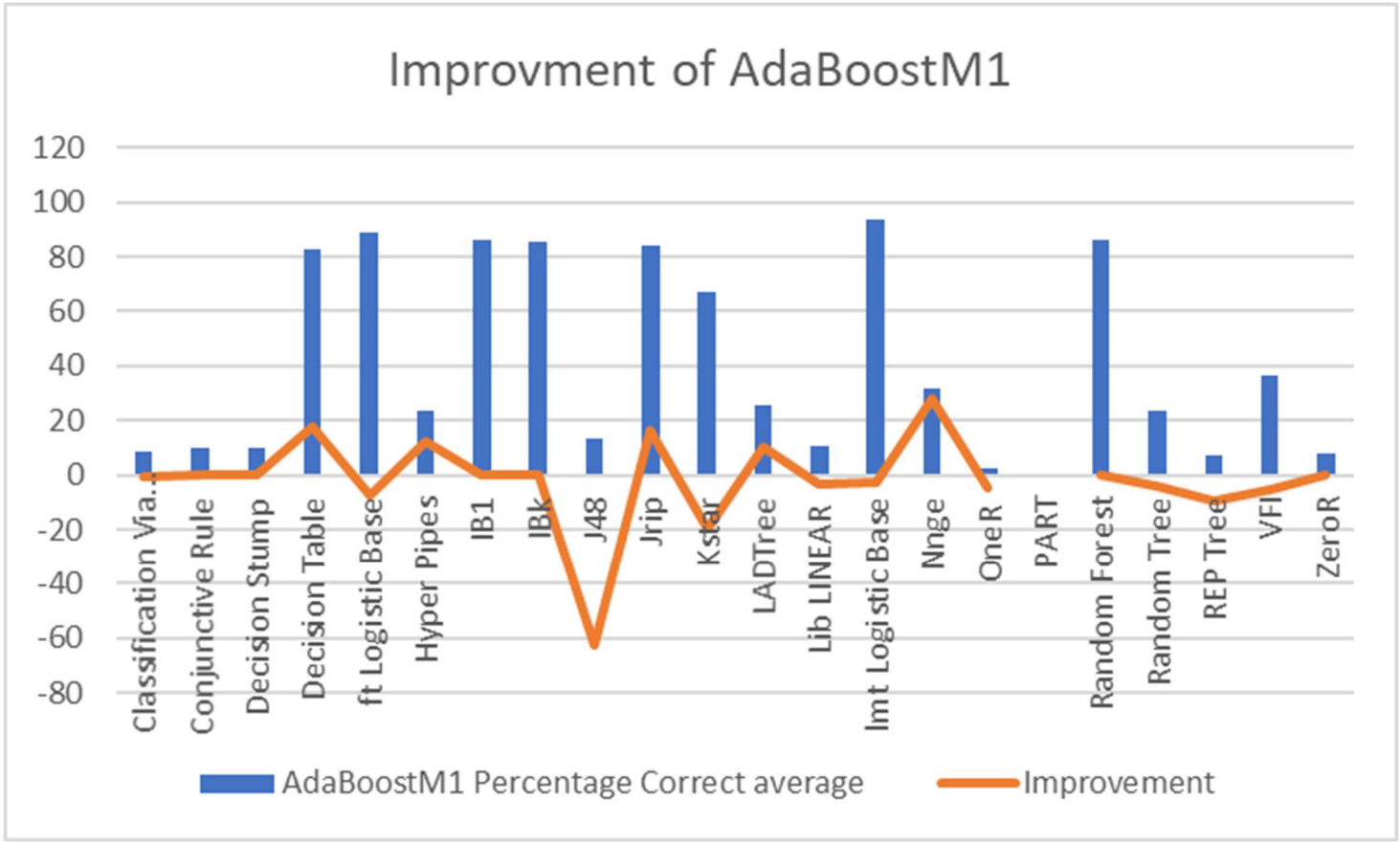


Figure 27. Adaptive boosting improvment

As shown in the figure above, only two classifiers manage to perform well while using adaptive boosting. The most successful of these is the JRIP classifier. This classifier managed to improve its control result by 16.144%, meaning that it could classify successfully 84.182% of the time. The other successful classifier that makes use of adaptive boosting is the decision table algorithm. The decision table classifier managed to improve more than JRIP and improved its control result by 17.724 %, meaning that it could classify at 82.972%. Only two other classifiers managed to improve adaptive boosting, but this improvement was not enough to make them useful in this context. Those classifiers being hyper pipes which managed to improve on its control result by 12.357%, bringing it to a total of 23.651% correct clarification, and the NNGE managed to show the most improvement out of any classifier with 27.833% improvement meaning that its final correct classification percentage was 31.427% correct classification. Notably, some classifiers that usually score well when using adaptive boosting scored significantly lower than expected due to natural variance in randomization. The worst affected of all of these was J48, which experienced a 62.771% reduction in performance from its control result. The KStar classifier performed poorly with this single classifier enhancer and found a performance drop of 19.745%, leaving it on 67.081% correct classification. With all this being considered, adaptive boosting should be applied to the decision table, and JRIP classifiers only as both showed significant improvement when using adaptive boosting and achieved high classification levels.

5.8.2.4 Delay based single classifier enhancer

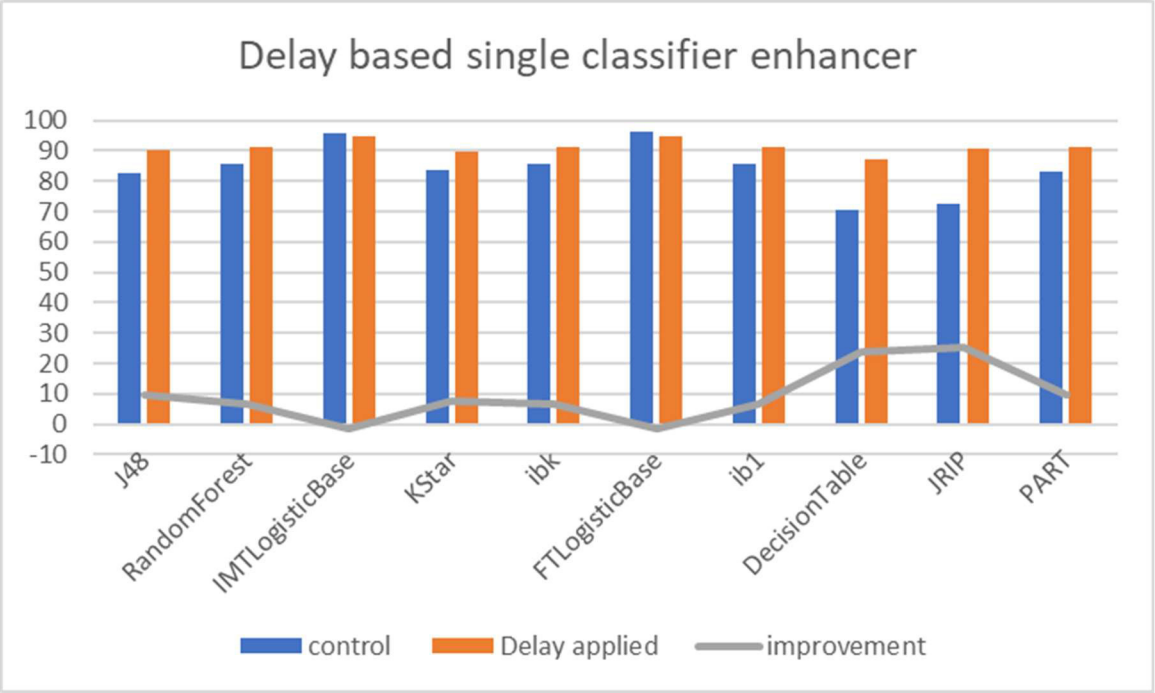


Figure 28. Delay based improvement

In the figure above, the effect of the delay-based single classifier on each classifier is displayed. We can see that for most classifiers, the delay based single classifier enhancer improves the results by 6% or more. The classifier to show the best improvement when using this enhancer is JRIP, which experienced a 25.186% increase in performance. Another classifier that performed exceptionally well is the decision table classifier, which improved its control score by 24.026%. Notably, both of these classifiers were the two worst-performing classifiers out of the group in the control experiment. When looking at the classifiers that did not perform well, the two prominent examples are both logistic base classifiers. Both of these were negatively affected by the delay based single classifier enhancer and experienced over 1% drop in performance, with the LMT logistic base experiencing a decline in the performance of 1.235% and ft logistic base experiencing a 1.311% drop in performance. Every other classifier experience between 6.5% and 10% improvement compared to their control scores. The conclusion we can draw from this evidence is that the delay-based single classifier enhancer is very likely to help a classifier improve its performance. The worst performing classifier originally is the better performing delay-based single classifier enhancer. However, there does reach a threshold where this becomes false.

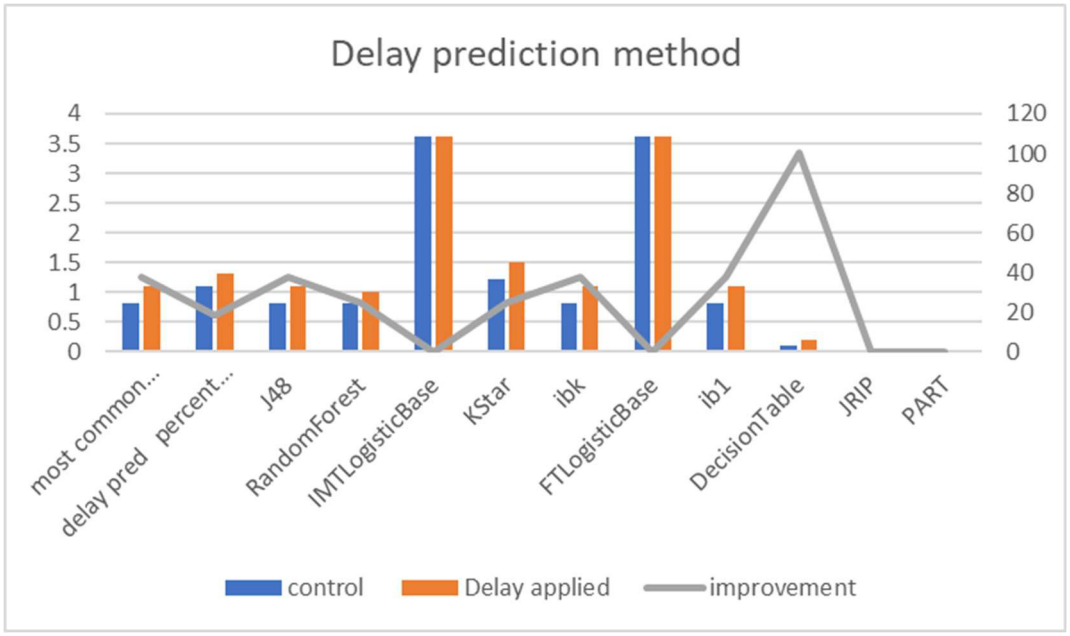


Figure 29. delay prediction methodology with poor performing classifiers

When Looking at the improvement results from the delay prediction methodology experiment on the poor performing classifiers, a conclusion can be drawn that if classifiers are not performing well in their base classifications, then a delay will significantly improve their results. The most improving classifier did show 100% improvement. However, this result is only so significant because the original classifier performed poorly in its classification that any addition to its correct classifications, no matter how small, significantly impacted the improvement measurement. It is sensible to say that this delay methodology will improve the results between 25 and 37.5% for moderately poor-performing classification algorithms. The most exciting results from this experiment come from the most common prediction methodology and the double delay. This double delay managed to improve on a previous delay by a significant margin. This shows the potential that increasing the number of delays or folds may significantly improve the results.

5.8.3 VOTE combination individual classifier improver

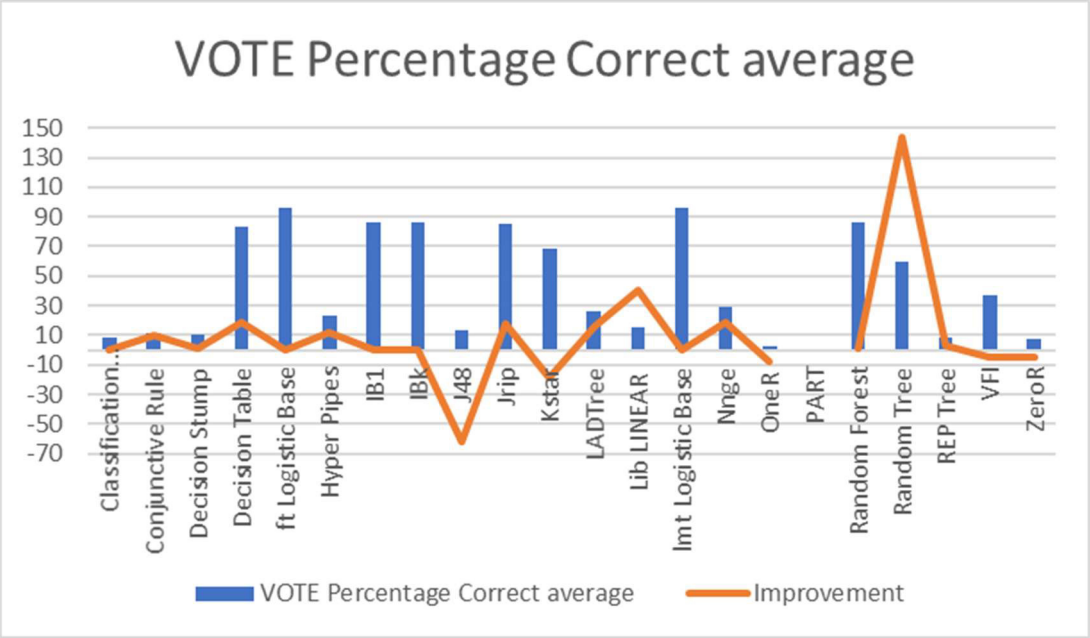


Figure 30. VOTE combination individual classifier improver results

The figure above shows the classifier rate when the vote combination individual classifier improver was applied. The vast majority of classifiers found some improvement with this method; the most successful was the random tree algorithm, which improved by 143.759%. This leaves the classifier 10% short of the 70% threshold but could reach the threshold with some other improvements. Other classifiers that saw great success when using this is the decision table classifier. This classifier managed to improve by 18.649% compared to its control result, which means that it could classify correctly 83.623% of the time, significantly improving from a relatively high result that just made it past the threshold to a very high-level classifier. The best performing classifiers did not make any use of this methodology, however. Both of the logistic base classifiers and both IB classifiers failed to show any significant improvement with this method applied. The creation time for these classifiers, especially the logistic base classifiers, was detrimentally affected by this improved method. Some classifiers were significantly negatively affected by this, most notably the J48 classifier, which found a 62.426% reduction in classification ability, meaning that it was left on 13.311% correct classification. Another classifier that was significantly affected was the case star classifier which found a reduction in prediction ability by 18.404%, meaning that it just missed out on the 70% threshold at a correct classification rate of 68.202%. There seems to be a correlation between the classifieds that performed well in this experiment and the classifiers that performed well when using adaptive boosting and bootstrap aggregation. This correlation can be seen in the decision table and JRIP. Although JRIP is negatively affected by bootstrap aggregation, these strong results from adaptive boosting make up for this.

5.8.4 The average performance of individual classifier improvers

In order to gain a good idea of how each of the improvement methodologies has affected the classifiers. Suppose we sum up the percentage classified correctly for all classifiers per experiment and find the average. Comparing the averages of each of these experiments will determine if the individual classifier improves positively or negatively affects the classifiers.

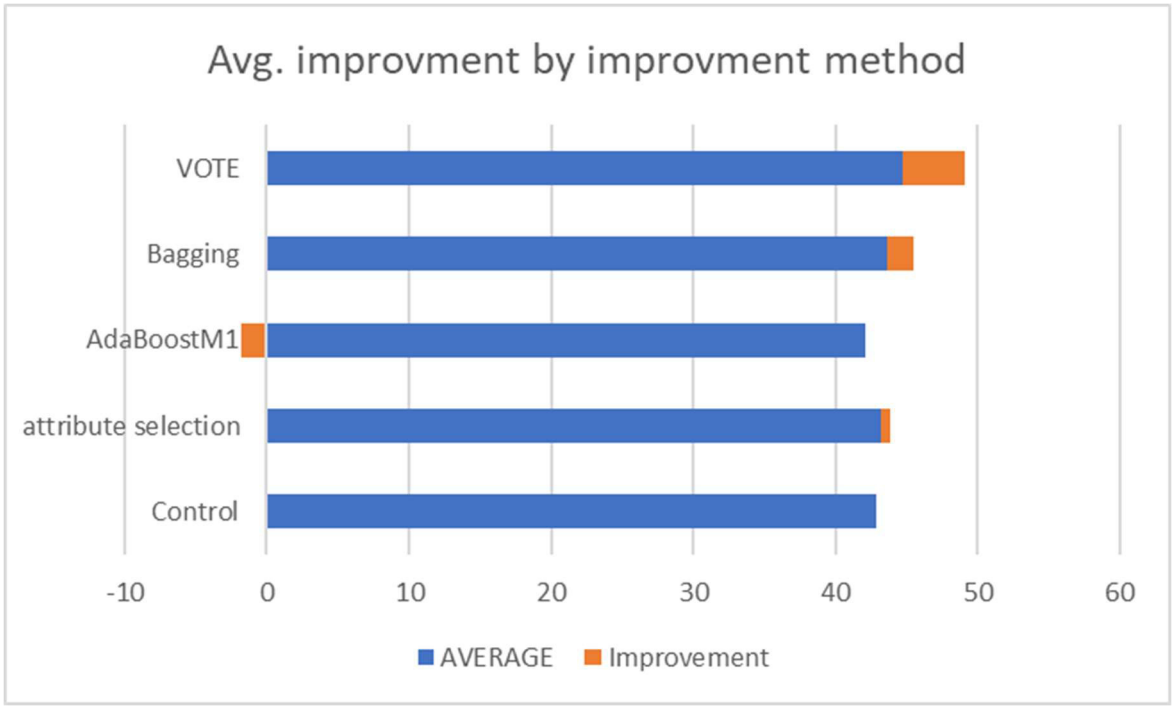


Figure 31. Average improvment by method

Looking at the figure above, we can see that the most successful individual classifier improvement methodology was using VOTE. Overall the vote improvement methodology managed to improve each classifiers classification rate by 4.347%. Only one other improvement methodology managed to show any significant signs of improvement, and that is the bagging single classify improver. This classification method managed to nearly reach 2% improvement but stopped short at 1.836% improvement. Notably, the aggregated boosting improvement methodology had an overall negative effect on the classification ability of the algorithms. Overall this improvement method negatively impacted the classification rate by 1.769%. Attribute selection did improve the classification rate but was the worst performing out of all the improvement methods that did not negatively impact the classification rate. Attribute selection managed to improve the classification rate by 0.682%.

5.8.5 voting methodology evaluation

5.8.5.1 Most common voting methodology

In the most common voting methodology experiment, this voting methodology did not perform better than all of the base classify. One family of classifiers managed to outperform this voting methodology, which means that it is not more accurate than using a base classifier for the tool in its current state. However, upon further inspection of the scenarios and individual instances correctly classified by the most common voting, the scenarios classified correctly were far more spread out. This means that using the most common voting methodology will help ensure an even spread of the possibility of prediction amongst all scenarios. The slight drop in performance, which is less than 1% that comes with using the voting methodology, is worth the trade-off to ensure that specific scenarios that the best performing classifiers may not predict are still classified often.

5.8.5.2 Weighted voting methodology

In the weighted voting methodology experiment, the weighted methodology correct classification average was the same as the average produced by two base classifiers. The conclusion drawn from this is that these classifiers are too heavily weighted in the voting methodology. The classification levels were recorded across the entire experiment to investigate this further, and the weighted prediction correct classification level rises above. It falls below both of these Class A fires. From this, we can prove that these classifiers are not too highly weighted. The classification level for the weighted prediction would have been pinned to whatever value is produced by the logistic base classifiers. Much like the most common voting methodology, the correctly classified scenarios were far more spread out in the data than the two logistic base classifiers.

5.8.5.3 Blending voting methodology

The blending voting methodology does rely on the single base classifiers. Unfortunately, these single base classifiers cannot have the optimal modifications applied to them to produce the best result on their own. This is due to the requirements of the multi classifier combine available through the machine learning library. When viewing the stream of results produced by the experiment, the blending voting methodology classification rate was the highest out of all the results. This means that the blending voting methodology can be the best performing voting methodology if the correct parameters and classifiers are found. This classifier considers all the other classifiers predictions before making a decision. It can adapt and “think” more than the other voting methodology that follows fixed rules, which gives it an edge at both low-level training and high-level training compared to the other methodologies.

5.8.6 attack type evaluation

When evaluating the frequency of detection by scenario and which characteristics make an event harder to detect, it is vital to understand the outliers in the dates are. Scenario 3 is not an attack scenario and should be relatively easy to detect this however is not reflected in the data as it is the 5th list detectable scenario full stop looking at the characteristics of scenario three, we can see that it is an SLG fault occurring on line one at 80 to 90%. Taking this characteristic and comparing it against the other scenarios that performed well in being undetected, the SLG fault typically appears in all of them. We can conclude that if an SLG fault is included in the scenario, this severely hinders a classification algorithms' ability to classify that instance correctly. The least detectable scenario is scenario 12. This scenario does include the fault characteristic previously described. Upon closer inspection of the specific fault found in this scenario, it ranges from 80 to 90% online. Taking this 80 to 90% fault and comparing it to all the other attack subtypes, we can see that when an 80-90% fault occurs, a decrease in the detectability occurs. The severity of this effect depends on which line the

fault occurs. Looking at line one, there is only a minor decrease in detectability from the scenario, but when this same fault is applied to the line to a significant decrease in detectability occurs. These characteristics of an SLG fault occurring at 80-90% on line 2 typically mean that a scenario is less detectable than other scenarios in its family. Other characteristics do help to decrease the scenarios detectability. These are the characteristics typically are applied to the relays found in the smart grid. These functions include acts such as disabling relays or performing command injection against relays.

5.9 SUMMARY

From the single base classifiers, we improved upon their control classifications to produce better results for every classifier. This means that using the methods described. Any based classifier can be improved a pond. The varying effectiveness of each improvement method based on either how much data the classifiers have been exposed to or how well they perform shows that each of the different methods has different effectiveness at specific ranges of instances.

6 CONCLUSION AND FUTURE WORK

6.1 BUILT-IN LIVE DATA CAPTURE

The current tool that has been built only takes pre-recorded data captured from this smart grid imported by the user. This data capture method is not practical as it requires constant user input to keep the tool up today with data from the smart grid. Improving upon this would genuinely make the application that can be left to run in the background.

This improvement will be made by creating a separate script with access to the data recording elements of the smart grid and can take that data and transform it into a.ARFF file type and ensure that it is of the correct format. This data can then be used by the current tool to make predictions about the current state of the smart grid.

6.2 INTEGRATING MORE DETECTION METHODOLOGIES

Denial of service attacks is a severe type of attack that the grid may experience, which we have not prepared. Denial of service attacks is becoming increasingly prevalent due to the relatively low level of skill it takes to perform one and the mass devastation it can cause. These types of attacks can be especially devastating to a smart grid network. One of the significant speed bumps in denial of service detection is that it is challenging to differentiate between legitimate network traffic and malicious network traffic. For instance, if a smart grid word 2 regulates the energy distribution within a city based on the energy distribution of these different areas. However, if legitimate data that the smart grid requires to make these decisions is discarded in detecting a denial of service attack, significant amounts of energy and money will be wasted as the smart grid cannot effectively communicate with the different areas of the city.

This new detection methodology would need to perform log analysis, response size analysis, mismatch in port analysis, large packet dumping alongside other compromise indicators to create a processable data set. This data set can then be analyzed using our machine learning algorithms to detect any possible behaviours associated with denial of service attacks.

6.3 EDGE CASE NEW SCENARIO CLASSIFICATION

Within the data set, there are a set amount of 35 unique scenarios. This limitation, unfortunately, limits the number of scenarios the application can process to 35. A significant improvement on the tool could be made by creating functionality that when an unknown string of instances is inputted, that does not match any specific scenario but is classed as valid pieces of data, then this functionality would create a new scenario based on this data. The previous data can then be analyzed to see if any other instances match this new scenario, which can then be associated with it.

This functionality would use clustering to find the outliers and any potentially new scenarios within the data. This edge case classification would allow the tool to adapt to any new attacks that may be launched against the smart grid that it has not previously been trained on.

7 FURTHER CONCLUSIONS

7.1 MOTIVATION AND PROBLEM

In this project, it was concluded that smart grids particularly vulnerable to cyber-attacks which can lead to significant disruption in the operations of the smart grid. These attacks can affect the communication between the different devices on the smart grid leading to interruptions in data transfer and corruption of data. These attacks' knock-on effects can have disastrous implications, such as localised blackouts in a targeted area. The problem with the research done in this field is that the classifications and responses from these tools are not precise enough to allow the user to make informed decisions about the smart grid. Different types of attacks require different responses, and as such, this needs to be reflected in the classifications.

As such, the tool we created demonstrates the collection and analysis of smart grid data to build multiple machine learning classifiers to make informed predictions about the current state of the smart grid. The accurate and appropriate advice given to the user by the application will be invaluable in the prevention and Minimisation of attacks. The application achieves this by taking training data on which each instance belongs to a scenario and training itself on it. Each different type of classifier does this in a slightly different way as previously described but using whatever methodology it uses, and it will take the new data and evaluate which classification at best fits into.

7.2 THE IDEA AND TOOL

The idea behind creating this tool was that it was able to provide correct scenario-type predictions on live data to help improve the user's situational awareness of their smart grid. Other tools did offer this, but the accuracy they provided is Paul and does not provide enough information to the user. A Python-based tool using a machine learning library was developed in order to be able to read the data from the smart grid and analyse the data in order to predict whether an attack is occurring. There are six attacks subtypes available to test our system those are SG fault replay, command injection against a single relay, command injection against two relays, disabling relay function against a single relay and fault, disabling relay function against two relays and a fault, and finally disabling relay function against two relays as well as line maintenance. These types of attacks provide a large variety to test against our tool and as they are widespread in real-life scenarios so provided good real-world comparison.

7.3 THE RESULTS AND CONCLUSIONS

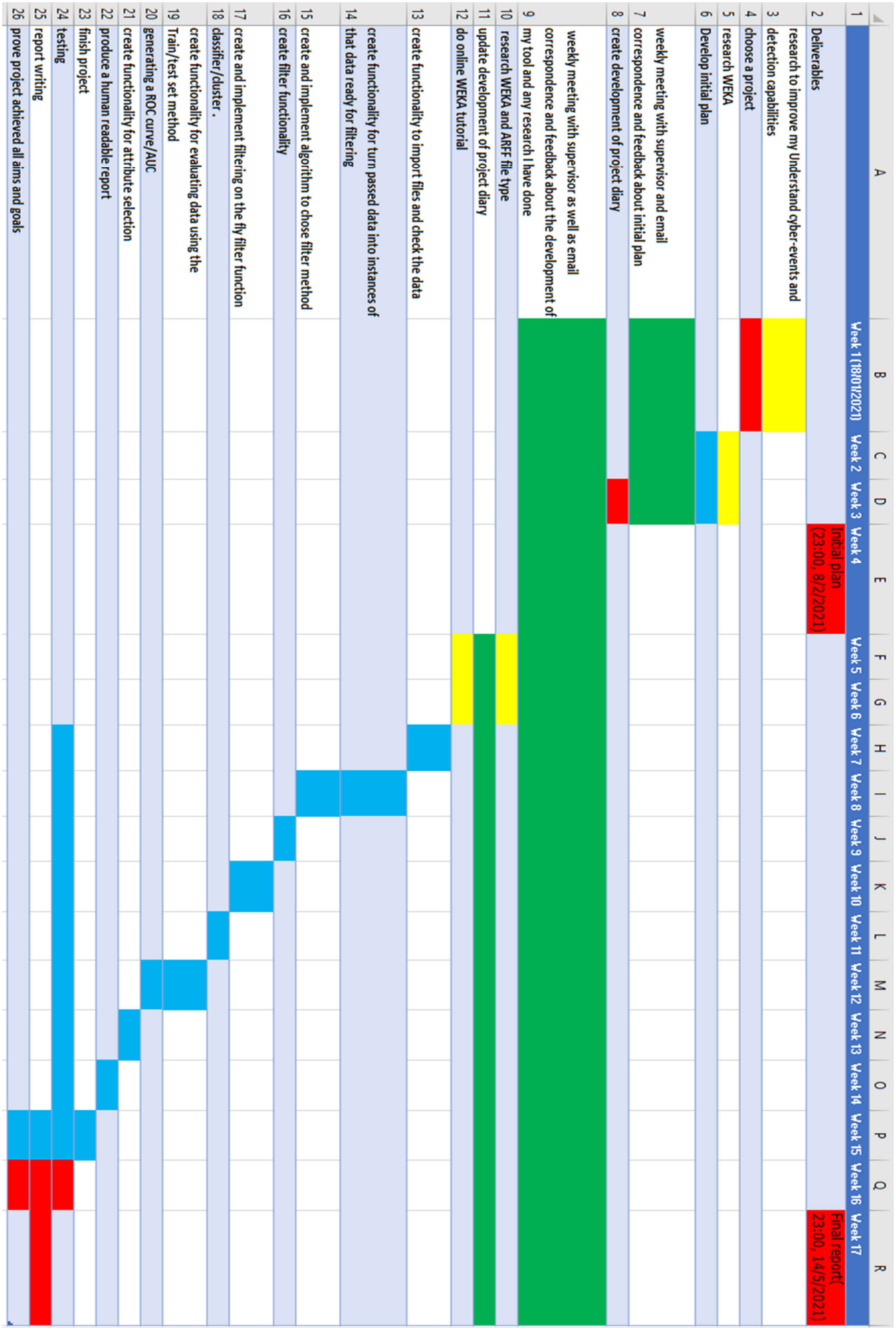
Looking at the results from this paper against the results from the background material provided, the background material claims to produce a more accurate result more often. The reason behind this is twofold. The previous paper exclusively relies on single base classifiers, which means that specific scenarios are classified more often than other scenarios, creating a significant drop in performance when specific scenarios occurring within the data, meaning that for the purposes described in this paper, a more spread out approach is better. The data set used in the background material has three different scenarios, whereas the data set used in testing this tool has 35 different possible unique scenarios. The margins for error and the impact that those errors cause felt far more significantly in the results of this paper. If a classifier word to not know which classification to place an unknown instance into, in the previous literature, if the classifier were to guess, there would be a 33 point 3% chance of producing a correct classification, whereas in this paper, since the amount of scenarios has significantly increased there is only a 2.8% chance of randomly selecting the correct classification.

8 REFLECTIONS MADE ON LEARNING

During this project's development process, I expanded my technical knowledge and skill into many different aspects of computer science. When I was considering a project to do, I knew that I would want it related to three possible areas forensics, cybersecurity and artificial intelligence, as these were areas that I have had an extensive interest in for a few years now. The first time reading about this project, I did not know anything about smart grids and only had a basic understanding of the principles of machine learning algorithms. Investigating this project and the different aspects has helped broaden my horizons, and through meetings with my supervisor and reading various research papers, I have been able to focus this learning into a helpful paper that you have hopefully read through.

9 APPENDICES

9.1 GANTT CHART



BIBLIOGRAPHY

Babak Akhgar, A. S. a. F. B., 2014. *Cyber Crime and Cyber Terrorism Investigator's Handbook.* s.l., Elsevier Inc..

Babak Akhgar, S. Y., 2013. *Strategic Intelligence Management; National Security Imperatives and Information and Communications Technologies.* s.l., Elsevier Inc..

Beaver, J. M., Borges-Hink, R. C. & Buckner, M. A., 2013. *An Evaluation of Machine Learning Methods to Detect Malicious SCADA Communications.* Miami, IEEE.

cyber security firm McAfee, the Center for Strategic and International Studies, 2020. *Economic Impact of Cybercrime - No Slowing Down,* s.l.: McAfee .

D. Aha, D. K., 1991. Instance-based learning algorithms. *Machine Learning,* Volume 6, pp. 37-66.

D. Aha, D. K., 1991. Instance-based learning algorithms. *Machine Learning,* Volume 6, pp. 37-66.

Dong, Z. et al., 2017. *Review and application of situation awareness key technologies for smart grid.* Beijing, China, IEEE.

Langner, R., 2011. *Stuxnet: Dissecting a Cyberwarfare Weapon,* s.l.: IEEE.

Machine Learning Group at the University of Waikato., n.d. *WEKA: The workbench for machine learning.* [Online]
Available at: https://www.cs.waikato.ac.nz/ml/weka/index.html
[Accessed 06 05 2021].

microsoft, 2021. *Microsoft Defender for Endpoint.* [Online]
Available at: https://www.microsoft.com/en-gb/microsoft-365/security/endpoint-defender?ocid=cx-blog-mmpc
[Accessed 19 05 2021].

Mississippi State University and Oak Ridge National Laboratory, 2014. *Power System Attack Datasets,* s.l.: s.n.

Mlitz, K., 2018. *Cybersecurity market revenues worldwide 2017-2023,* s.l.: statista.

Raymond C. Borges Hink, J. M. B. M. A. B. T. M. U. A. S. P., 2014. Machine learning for power system disturbance and cyber-attack discrimination. *7th International Symposium on Resilient Control Systems (ISRCS).*

Raymond C. Borges Hink, J. M. B. & Mark A. Buckner, T. M. U. A. S. P., 2014. *Machine learning for power system disturbance and cyber-attack discrimination.* Denver, CO, USA, IEEE.

Raymond C. Borges Hink, J. M. B. M. U. A. S. P., 2014. *Machine Learning for Power System Disturbance and,* s.l.: IEEE.

Shengyi Pan, T. M. ,. U. A., 2015 . *Classification of Disturbances and Cyber-Attacks in Power Systems Using Heterogeneous Time-Synchronized Data,* s.l.: IEEE.

Timm, C., 2018. *Analysis of Remote Tripping Command Injection Attacks in Industrial Control Systems Through Statistical and Machine Learning Methods,* s.l.: STARS..

Vacca, J. R., 2017. *Computer and Information Security Handbook.* s.l., Elsevier Inc..

Zeller, M., 2011. *Common Questions and Answers.* San Diego, Schweitzer Engineering Laboratories, Inc..