On the Efficacy of Enhanced Feature Selection Methods for Supervised Crime Prediction

UNIVERSITY *of the*
WESTERN CAPE

by

Sphamandla Innocent, MAY (3437444@myuwc.ac.za)

This thesis is submitted in fulfillment of requirements for the degree of

Master of Science in Computer Science

In the Faculty of Natural Science

Department of Computer Science

Supervisor: Dr. Omowunmi Isafiade

October 23, 2023

CERTIFICATION

As the candidate's supervisors, we have approved this thesis for submission.

Supervisor: Dr. Omowunmi Elizabeth Isafiade

Signature: _____

Date: _____

Co-supervisor: Dr. Olasupo Ajayi

Signature: _____

Date: _____

DECLARATION

I declare that this thesis is my own work. Where collaboration with other people has taken place, or material generated by other researchers is included, the parties and/or materials are indicated in the acknowledgements or are explicitly stated with references as appropriate.

This thesis is being submitted for the master's degree in Computer Science at the University of the Western Cape. It has not been submitted to any other university for any other degree or examination.

Sphamandla May Date

ABSTRACT

The challenge of crime across the globe has necessitated several considerations for crime preventive measures. There exist a variety of crime prevention strategies, such as the use of necessary weapons or tools to respond to crime. However, for resource-constrained nations such as South Africa, where the current police to civilian ratio is overwhelming, this may not suffice. Consequently, crime continues to be on the rise, necessitating alternative prevention strategies. Among alternative prevention approaches, the use of historical crime data can be explored through machine learning. Crime prediction using machine learning has been explored and has shown promising results. However, the choice of algorithm and feature selection methods play a critical role in creating an effective predictive model. This study, therefore, explores the efficacy of enhanced feature selection methods in supervised machine learning algorithms for crime prediction. Four (4) baseline algorithms are adopted, which are Random Forest (RF), Extremely Randomized Trees (ERT), Naïve Bayes (NB), and Support Vector Machine (SVM). This research further proposes three algorithms, with the first derived from hybridizing RF and ERT (RF-Plus), while the other two (2) were obtained from enhancing NB and SVM using recursive feature elimination (RFE), obtaining (RFE-NB) and (RFE-SVM) respectively, totaling seven algorithms. Finally, a comparative evaluation of these algorithms with their respective baselines is conducted to report on their efficacy and contrasted against additional two (2) algorithms from the literature, which amounts to a total of nine (9) algorithms. The study conducted performance evaluation on the models using two distinct publicly available datasets, which are the Chicago and Los Angeles crime datasets. Results confirm that feature selection positively impacts prediction accuracy. The enhancement on the pure NB improved its accuracy from 72.5% to 96.6% and 80.45% to 95.78% for Chicago and Los Angeles datasets, respectively. The enhancement improved the accuracy of pure SVM from 74.73% to 89.91% and 75.73% to 88.70% for the Chicago and Los Angeles datasets, respectively, while achieving 97.04% and 95.5% on RF-Plus for both Chicago and Los Angeles datasets, respectively.

ACKNOWLEDGEMENT

First and foremost, I would like to extend my sincerest gratitude to my supervisors, Dr Omowunmi Isafiade and Dr Olasupo Ajayi for whom, in their absence, this study would not have been realized. I would like to thank you for your limitless guidance, support and patience throughout the duration of my studies.

I would also like to thank myself for the courage and fortitude to withstand the difficulties and challenges encountered over the course of the studies. I certainly never thought I had it in me to undertake such an undertaking.

Furthermore, I would like to thank my family and friends for their encouragements and support.

PUBLICATIONS

Conference Proceedings Published from the Research:

- May, S.; Isafiade, O. and Ajayi, O. (2022). An Improved Support Vector Model with Recursive Feature Elimination for Crime Prediction. In Proceedings of the 14th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KDIR, ISBN 978-989-758-614-9; ISSN 2184-3228, pages 196-203.
  DOI: 10.5220/0011524200003335

- May, S., Isafiade, O. and Ajayi, O. (2021). Hybridizing Extremely Randomized Trees with Bootstrap Aggregation for Crime Prediction. In 2021 4th International Conference on Artificial Intelligence and Pattern Recognition (AIPR 2021),ISBN 978-1-4503-8408-7; pages 536–541.
  DOI:10.1145/3488933.3488972

- May, S., Isafiade, O. and Ajayi, O., (2021). An enhanced Naïve Bayes model for crime prediction using recursive feature elimination. In 2021 4th International Conference on Artificial Intelligence and Pattern Recognition (AIPR 2021), ISBN 978-1-4503-8408-7; pages 580-586.
  DOI:10.1145/3488933.3488973

Contents

Glossary

ARIMA : Auto Regressive Integrated Moving Averages. 6

CART : Classification And Regression Trees. 16, 17, 20, 21

CHAID : Chi-square Automatic Interaction Detector. 19, 20

CLEAR : Citizen Law Enforcement Analysis and Reporting. 6, 43, 44

DT : Decision Trees. 2, 6, 12

ERT : Extremely Randomized Trees. 2, 4, 7, 8, 11, 12, 27, 28, 47, 48, 50, 51, 56, 61

KDE : Kernel Density Estimation. 6

LASSO : Least Absolute Shrinkage Operator. 8

MI : Mutual Information. 10

NB : Naive Bayes. 2–4, 6–8, 11, 47, 51, 54, 56–58, 60, 61

PCA : Principal Component Analysis. 5

RF : Random Forest. 7, 26–28, 47, 48, 50, 51, 56, 61

RF-Plus : Proposed Hybrid model. 47, 48, 50, 51, 56, 57, 60

RFE : Recursive Feature Elimination. 2, 7, 11, 38, 60

RFE-NB : Naïve Bayes enhanced by RFE. 47, 57, 60

RFE-SVM : Support Vector Machine enhanced by RFE. 57

SVM : Support Vector Machine. 2, 3, 6, 8, 13, 47, 56, 60

WEKA : Waikato Environment for Knowledge Analysis. 11

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Crime is defined as an act that is deemed socially harmful, prohibited, and punishable by law. It is a serious problem which affects everyone. Globally, crime rates have continued to increase at an alarming rate, with financial, physiological, and psychological implications (such as Post-Traumatic Stress Disorder (PTSD) [3]). This challenge is magnified in resource-constrained settings such as in South Africa, where the current police to civilian ratio is 1:413 [4]. This means that there is one police officer for every 413 residents. Unfortunately, as reprehensible as crime is, it continues to increase. The continuous increase in crime rates serves as an indication that the current crime prevention strategies may have some limitations and there is a need for more intervention [5]. Therefore, it is of utmost importance to devise new strategies and measures to curb this ever increasing menace in the society.

While crime is believed to be random and could happen anywhere, the theory of repeat victimization suggests that criminals work with opportunities that present themselves and in areas where they have previously succeeded [6, 7]. This further suggests that there might be a pattern to crime. By investigating previous crimes, the observed patterns can be used to take preventive measures to deter future occurrences. Typically, crime data are recorded for analysis and investigations to discover interesting insights and possible patterns. However, the monumental increase in recorded crimes in the last few years [8], coupled with human limitations, has made it somewhat difficult for crime intelligence officers to effectively analyse millions of records, particularly in resource-constrained settings. Furthermore, the manual means of processing is laborious, inefficient and prone to human errors. Thus, there is a need for intelligent analysis of crime with minimal human intervention. This has spurred law enforcement agencies to consider advanced Geographic Information Systems(GIS) and machine learning approaches.

Machine Learning (ML) is a promising field of study that has proven reliable in addressing crime issues through knowledge support [7,9,10]. ML lies at the intersection of Statistics and Computer Science [11]. On one hand, it utilises statistical theories to devise robust and powerful algorithms to analyze huge records of data; while on the other hand, it uses computers to carry out the analysis in an efficient manner, immune to human errors [12]. The ubiquity of ML has resulted in its application in numerous fields, including crime prediction as done in [13–17].

Crime prediction as a preventative strategy utilizes ML to fully exploit insights and trends from previously collected crime data. There are various applications of ML in crime prediction, including crime hotspot detection as well as determining the time, location and class of crimes [18]. Numerous ML models and algorithms exist to achieve the goal, such as regression, clustering and classification. A common supervised ML model used for crime related studies is the Decision Tree (DT) [19,20]. In this study, two variants of DTs are investigated, these are, Random Forest and Extremely Randomized Trees(ERT). Then a new algorithm is proposed by hybridizing the best attributes from both DT-based algorithms. The newly proposed hybrid algorithm, called Random Forest-Plus, is then contrasted against both the Random Forest and ERT for performance.

Besides DTs, the probabilistic Naïve Bayes(NB) [21] is a popular statistical model that has wide application in ML domains. Its popularity can be attributed to its ease of application and interpretation. Unfortunately, NB is often ignored in crime prediction problems because it does not have intrinsic feature selection abilities. This reason makes it ill-suited for feature-based crime prediction. In this study, this limitation was addressed by enhancing NB with RFE. This enhanced version of Naive Bayes was then called RFE-NB. Furthermore, the geometrically inspired Support Vector MachineSVM is another ML model which has extensively been used for crime prediction. Also, in this research, the classic SVM was enhanced by Recursive Feature Elimination (RFE) to obtain the hybrid RFE-SVM. These 7 algorithms were then used to predict crime and their performance analysis is documented.

The Random Forest, ERT and RF-Plus algorithms, together with the pure NB and SVM as well as the enhanced variants (RFE-NB and RFE-SVM), were used to predict crime and their performance was contrasted against each other. By enhancing NB and SVM with feature selection capabilities, the comparison of RFE-NB and RFE-SVM with the DT-based algorithms (with intrinsic feature selection capabilities) is justified.

1.1 Research Questions

The research question is in two folds. The first question is in relation to the construction of a hybrid algorithm from the baselines, RF and ERT. The second question is in relation to incorporation of feature selection techniques into supervised algorithms, as well as the performance of all models considered. Hence, the following questions emerged:

1. How can a hybrid algorithm be created from RF and ERT ?
2. How can feature selection models such as RFE be used to augment supervised models such as SVM and NB for improved performance

Furthermore, the following sub-question emerged:

- How do the newly created hybrid algorithms compare to their parent algorithms in terms of performance?

1.2 Research Aims and Objectives

The aim of this work is to explore the efficacy of feature selection methods in supervised learning models for crime prediction. The objectives are thus as follows:

1. Create a variant of NB and SVM that introduces feature selection capabilities, and compare the performance of enhanced NB and SVM with tree-based (RF and ERT) models that have intrinsic feature selection capabilities.
2. Construct hybrid algorithms, which are; RF-Plus, RFE-NB and RFE-SVM from the aforementioned parent algorithms.
3. Conduct a comparative evaluation of all seven models taken into consideration, with two other relevant models from past literature studies as a benchmark.

1.3 Thesis Outline

The remainder of the thesis is organized as follows:

Chapter 2 - *Literature Review*: In this chapter, existing literature in the field of crime prediction using supervised algorithms is reviewed. For algorithms without intrinsic feature selection capabilities, studies which have attempted to enhance their performance through the incorporation of feature selection algorithms are reviewed, presenting various feature selection algorithms in order to elucidate their relevance and use in crime prediction problems. This exploration guides and informs this research on the most suited algorithms for this study, with the most appropriate feature selection methods.

Chapter 3 - *Models and Methodology*: This chapter presents the theoretical underpinning of the four major models explored in this research, which serves as the basis and framework for the study on crime prediction. The models are Random Forest(RF), ERT, Naive Bayes(NB) and Support Vector Machine (SVM). Furthermore, theoretical description of how an additional three hybrid algorithms were constructed from the four baselines is provided. From RF and ERT, a hybrid algorithm which is referred to as RF-Plus was constructed. NB and SVM algorithms required an additional feature selection model, Recursive Feature Elimination (RFE), to construct relevant hybrid algorithms and the theoretical description and all the pseudo-codes are also detailed. The detailed methodological approach, and the machine learning pipeline is provided, including data acquisition, pre-processing and feature selection. Finally, the evaluation metrics that were used to compare the performance of the aforementioned models are discussed.

Chapter 4 - *Results and Discussion*: This chapter provides a detailed explanation of the experiment conducted on the two crime data-set used, as well as the evaluation of the models explored. The configuration and parameters of the models used to conduct experiments are discussed. Furthermore, a detailed description of the experiments with relevant results are presented, both in tabular and graphical format. These results are in line with the research questions and objectives that are stated in this research.

Chapter 5 - *Summary and Conclusions*: This chapter concludes the thesis by providing a summary of the findings from the experiment, highlighting the relevance, effectiveness and limitations of the proposed hybrid algorithms, and provides direction for future work.

Chapter 2

LITERATURE REVIEW

2.1 Trend on Crime Prediction and Analysis

Crime prediction and analysis have been widely investigated by researchers [22–24], with various techniques and theories employed to achieve the common goal of anticipating crime and taking preventive measures [25, 26]. Machine learning has been used for that purpose, which can be broadly classified into three main categories, namely, reinforcement learning, unsupervised learning and supervised learning. In reinforcement learning, algorithms are created to teach an autonomous agent to act and sense its environment to decide the best actions to take in order to achieve its goals. The agent obtains information about the present state in the environment and performs actions to alter it. Each and every time an agent performs an action he receives a negative or a positive reward or punishment, thus reinforcement [27]. Reinforcement learning is used in robotics, personalised recommendations, resource management and gaming to name a few [28–31]. On the other hand, in unsupervised learning, training data is neither labelled, classified nor categorised. The main goal of the algorithm is to learn, discover and uncover previously hidden and potentially interesting patterns in the data. Unsupervised learning finds applications in algorithms such as clustering, anomaly detection, outlier detection, neural network and Principal Component Analysis(PCA) [32]. In contrast,in supervised learning, data is labelled, classified and categorised to guide and supervise the agent, thus the idea of supervised learning. Supervised learning has two main categories, regression and classification. Regression aims to predict a continuous quantity based on input vector while classification predict discrete class labels.

Techniques such as deep learning algorithms, owing to their ability to extract features from unstruc-

tured and raw datasets, have also been used for crime prediction. They have been widely used with unstructured data such as social networks [33], weather and newspapers analysis [17, 34]. Despite the fact that deep learning algorithms have been proven to produce positive results, they are not considered in this work, because the researcher's focus is on exploring the impact(s) of features using supervised learning algorithms.

In addition, approaches like Kernel Density Estimation (KDE), which determines crime hotspots and concentration zones, have been used in the literature on crime prediction [33, 35]. In [36], the authors show how KDE techniques fail to account for the temporal characteristics of crime. They designed a Spatial-Temporal Kernel Density Estimation framework to address these flaws, which they then compared to the KDE and found to be superior.

Other than using the KDE to determine the hostpots and high crime concentration zones, regression techniques have been used to predict crime occurrences with varied degrees of success. Prominent among these regression models is the Auto Regressive Integrated Moving Averages (ARIMA) as used in [37–42]. For instance, in [41], the authors utilized the Chicago Police Department's Citizen Law Enforcement Analysis and Reporting(CLEAR) system [1] dataset to predict monthly, weekly and daily crime occurrences. Their work proved that it is possible to use past crime data coupled with a predictive model to reduce crime count, thus, ensuring security of the citizens.

Similarly, clustering methods have also been utilized in crime prediction, such as in [43], [37, 44, 45]. In [43], researchers used the San Fransisco data to show that k-means clustering improved the accuracy of the Naïve Bayes(NB), Decision Trees and several ensemble learning approaches, including Bagging, Voting and Stacking. However, in as much as clustering enhanced models have proven to increase classification accuracy, researchers mostly still rely on manual means of feature selection which is prone to human errors. This is the case in the works done in [43], [46–50].

There have been various other comparative evaluation studies conducted in literature on crime prediction using supervised classification approaches. Some of the reviewed research indicate that Decision Trees [51] are superior to NB [52], Support Vector Machine (SVM) [49] and Perceptron [50]. A recent study conducted by O Llaha [53] to evaluate the performance of various ML algorithms, also concluded that DT models do perform better. However, it has also been shown that DTs suffer from high variance and bias limitations. Unfortunately, most of the studies that apply DT models did not show how they mitigated this shortcoming of DT. Though there have been methods proposed to resolve this shortcoming, they

have not been fully utilized in crime prediction domain. This indicates a gap between the progression of the decision trees and how they have been utilized in crime prediction.

To overcome the high variance and bias shortcoming of the decision trees, Leo Breiman in 2001 proposed the Random Forest (RF) [54]. Random Forest uses the Bagging method also referred to as Bootstrap aggregation proposed by Leo Breiman in 1996 [55]. Leo derived the Bagging concept from the Bootstrap framework developed by Bradely Efron in 1979 [56]. This overcomes the high variance and bias problem of the decision trees by training a multitude of the classifiers also referred to as weak learners (decision trees) using the bootstrapped dataset, then aggregating the results to make a strong and powerful classifier. The decision trees are susceptible to overfitting partially due to the fact that all features are considered when constructing the tree, however, in RF Leo utilized the idea of random selection of a subset of features. There are various functions utilized to select the random number of features to be considered when constructing each tree in the ensemble, such functions include $\log_2(p)$ and $\sqrt{p}$, where p is all the features in the dataset [57]. The idea of a random selection of a subset of features was proposed by works of Ho [58] and Amit and Geman [59]. He writes, Leo, that these works were highly influential on his thinking when he was working on RF.

In 2006 Geurtes developed the Extremely Randomized Trees (ERT), also referred to as Extra Trees [60]. The ERT are similar to Random Forest in a sense that they build a multitude (ensemble) of decision trees and aggregate the results. Also, they adopt the idea of a random selection of a subset of features. There are two major differences between Random Forest and ERT, which are: the absence of bagging (bootstrap aggregation) and instead of calculating the optimal cut-point for every feature under consideration, a random cut-point is selected. The additional layer of randomization (random cut-point) also decreases variance in ERT [60].

In addition to enhancing NB using RFE three properties that were identified to be integral to the robustness of both RF and ERT were extracted. These are dataset randomization through bootstrap aggregation (bagging), random selection of a subset of features, and random selection of a cut-off point when splitting the data. Based on these three randomization properties, the proposed hybrid algorithm was constructed. The proposed hybrid algorithm is contrasted against pure Random Forest and ERT in terms of predictive accuracy and model computational complexity time.

2.2 Feature Selection Methods

The processed mentioned in Section 2.1 have high probability of error, an opportunity for automatic means of feature selection arose. These automatic feature selection methods improve the base model's accuracy and reduces its complexity [61]. There are various feature selection techniques, such as, Intrinsic (or Embedded), Filter and Wrapper.

2.2.1 Embedded Feature Selection Methods

The embedded methods refer to the ability of the algorithm itself to perform feature selection. Tree-based algorithms, such as Random Forest and ERT, are common examples of this [62] [63].

Other than the tree-based algorithms, there are also regularization methods which utilize the intrinsic penalization function to reduce over-fitting. Examples of these regularization techniques are the Least Absolute Shrinkage Operator (LASSO), which performs L1 regularization [64], and the Ridge Regression that performs L2 regularization [65]. The LASSO feature selection technique was used in [66] to select the best subset of features to build a NB and SVM classifier for crime prediction and classification.

2.2.2 Filter Feature Selection Methods

Filter methods assess features relevance outside of predictive algorithm and then model only the features that meet some criterion [67,68]. Ranking methods are used mainly because of their simplicity and good success. The features are scored using an appropriate ranking criterion, and those that score below the threshold are removed. For example, in classification problems, each feature could be evaluated individually to see if it has a plausible relationship with the observed classes. Furthermore, fundamental property of a unique feature is that it contains useful information about the observed classes in the dataset. This property is known as feature relevance, and it measures a feature's usefulness in distinguishing between different classes [69]. A classification algorithm would then include only features with significant relationships with observed classes. Ranking methods are filter methods since they are used prior to the classification process to filter out the less important features.

The relevance of a particular feature is better understood after examination of two ranking strategies. In this section, data and variables are represented using a standard notation. The input data $[X_{i_j}, y_k]$

has N samples $i = 1$ to N with D variables $j = 1$ to D, x_i is the i^{th} sample and y_k is the class label $k = 1$ to Y

2.2.2.1 Correlation Criteria

The Pearson correlation coefficient [70, 71] is one of the most basic criteria as defined in equation 2.1.

$$R(i) = \frac{cov(x_i, Y)}{\sqrt{var(x_i) * var(Y)}}. \tag{2.1}$$

where x_i is the i_{th} variable and Y is the output label, class label. Then cov and var are covariance and variance, respectively. Correlation ranking only detects linear dependencies between variable and target label. Amongst correlation criteria there are other statistical approaches such as correlation feature evaluator and correlation-based feature subset evaluator.

2.2.2.2 Mutual Information (MI)

Information theoretic ranking criteria utilizes the measure of dependency between two variables [69–74]. MI is best described by starting with Shannons definition of entropy in equation 2.2.

$$H(Y) = -\sum_{y} p(y)log(p(y)). \tag{2.2}$$

Equation 2.2 represents what is refered to as uncertainty or rather information content in output Y. Lets us supposed that an observation is made of the variable X then the conditional entropy is given by equation 2.3.

$$H(Y|X) = -\sum_{x}\sum_{y} p(x,y)log(p(y|x)). \tag{2.3}$$

Equation 2.3 implies that the uncertainty(information content) in target label Y is reduced by observing a variable x.

$$I(Y,X) = H(Y) - H(Y|X). \tag{2.4}$$

Equation 2.4 gives MI (MI) between the target label Y and variable X, which implies that if X and Y are independent, then MI will be zero and if MI is greater than zero then X and Y are dependent. The definition provided in equations 2.2, 2.3 and 2.4 holds for discrete variables. To achieve the same with continuous variables then the summations will have to be replaced with integration. MI can also be defined as a distance measure given by equation 2.5

$$K(f, g) = \int f(y) \log\left(\frac{f(y)}{g(y)}\right). \tag{2.5}$$

In equation 2.5 the measure K represents the Kullback-Leibler divergence [75, 76] between two distinct densities which can also be used as a measure of MI. From the equations above, it is required to know the probability density function(PDF) of all the variables to calculate MI. Given that the data obtained is finite samples, the PDF cannot be accurately calculated. There exist a variety of methods developed for estimating MI [71, 75, 76]. Once a method for calculating MI has been selected, one of the simplest methods for feature selection is to find the MI between each feature and the class label and rank them based on this value. Set a threshold to choose $d < D$ features. This is the simplest method which can lead to poor results since inter-feature MI is not taken into consideration [73]. However, MI is an integral concept which is also used in embedded methods of feature selection to be discussed in.

Using Conditional MI the authors in [77] developed a feature ranking criteria for boolean data(binary data). As features are added to the subset using the conditional MI criterion, a score table is updated. Equation 2.6 is used to calculate the score at each iteration.

$$ps[n] = min_{l<k} \hat{I}(Y; X_n | X_{v(l)}). \tag{2.6}$$

Where $ps[n]$ is the partial score to be updated at every iteration, X_n is the score of the current evaluated feature and $X_{v(l)}$ is the set of features that have been selected already. Iteratively, equation 2.6 selects features that maximize the MI with respect to the class and features similar to features already picked are skipped. This provide trade-off between discrimination and independence.

Information gain from information theory [78] is utilized in [79]. Authors in [79] and [78] utilized these three methods to find the best subset of features to construct a crime prediction classification algorithm. Evaluating their methods on a community crime dataset using Waikato Environment for

Knowledge Analysis (WEKA) (an open source data mining tool), they concluded that the combination of correlation feature evaluator and correlation-based feature subset evaluator was the best feature selection method.

2.2.3 Wrapper Feature Selection Methods

There are also wrapper methods of feature selection that search for the best performing subset of features. These wrapper techniques select a subset of features, which are then used to train given models. Based on those inferences returned, wrapper methods adds or remove features from the subset. There are various techniques within the wrapper category, notable among which are the Forward Selection, Backward Selection and Recursive Feature Elimination (RFE). Forward feature selection is an iterative process that starts off with no features, then incrementally adds features until further additions do not improve the model's performance. Contrary to forward feature selection, backward feature selection starts will all features and iteratively removes features until further removal of features does not improve the model's performance. In [52], backward feature selection technique is used to select the optimal subset of features, which are then used to train a Decision Tree and NB classifier to predict crime.

In [80] RFE is used to select the best of features which are then fed to Linear Regression. The authors then compared their model against pure Linear Regression and Random Forest using burglary prediction tested on the Los Angeles crime dataset. They concluded that the modified Linear Regression produced better results than the modified Random Forest and the bench-mark models. Furthermore, in [81] the researchers constructed a crime prediction model to predict crime counts on the New York foursquare dataset using the RFE model.

Contrary to both [80] and [81], this study combines RFE with NB algorithm to construct a crime prediction model. The NB algorithm was utilized because it is a probabilistic algorithm with no intrinsic feature selection capabilities. Then the enhanced NB is compared with two algorithms that have intrinsic feature selection capabilities which are; Random Forest and ERT.

Chapter 3

MODELS AND METHODOLOGY

This work explored supervised machine learning algorithms, namely: Decision Trees, Naive Bayes, Support Vector Machine, Random Forest , Extremely Randomized Trees (ERT). From Random Forest, ERT, Naive Bayes and Support Vector Machine hybrid models were constructed which are explained in this chapter, with their respective pseudocodes presented.

3.1 Decision Tree-Based Algorithms

Statistics, data mining and machine learning comprise of various predictive modeling approaches and decision tree learning is one of them. Decision trees take observations about an item and depict them as branches before reaching a conclusion on the item's target value, which is usually represented by the tree's leaves [82]. Classification trees are trees where the target value (leaves) are discrete values. In classification trees, the leaves are class labels and branches are a set of features that lead to that class label [83]. In contrast, regression trees are trees where the target value(leaves) are continuous values [83].

To make a prediction for a given case, DT takes the feature set of the case and traverse the tree from the root node to the leaf node that contains the decision. Traversal is from root node to the leaf node(s) because the decision trees are constructed as a top-to-down structure utilizing the divide-and-conquer approach. Data is repeatedly split based on predictor variables to maximize homogeneity of the leaf node. From the set of attributes, one will be selected as a root node and construct the rest of the

branches using the remaining features. Recursive partitioning of decision trees creates subsets of the dataset at each split. A rule is a path taken from the root node to the leaf node [82].

The decision tree is deterministic, however, the nature of the model constructed differs depending on the algorithm used. The distinction in algorithms arises from the split criteria used to split the data. The traversals from root node to the leaf node, the decision rules are used to predict labels of a sample, the decision rules are constructed by inference from the feature of the dataset. To make a classification, a sample is passed through numerous tests that determine the class that the samples belong to even though learning is carried out by decision trees. The organization of the steps to find a label of a sample forms a hierarchical tree-like structure referred to as a decision tree.

In theory, the recursive partitioning continues and terminates only when a singleton subsets as leaf nodes have been obtained.However, in practice this is as efficient, because decision trees sometimes fail to generalize from unseen data. This phenomenon in machine learning is called over-fitting and happens when decision tree are trained to be rigid, trained to perfectly fit the training data and unable to make predictions on unseen data. The branches become rigid with strict rules of sparse data. To overcome overfitting in the decision tree, the method employed is called pruning and is undertaken after completing training. In the next section pruning will be discussed.

Decision trees are among the most popular machine learning models, due to various reasons such as simplicity and intelligibility [84]. Furthermore, they have a short learning curve, it's easy for non-experts to understand them after a brief explanation and they can also be graphically represented for easy interpretation. Among many other reasons, they can handle both categorical and numerical data, have intrinsic embedded feature selection and as opposed to black-box models, they are a white-box models with increased interpretability and understandability of the results.

Due to their numerous features and advantages, decision trees have been widely applied in various fields, such as biology [85], legal studies, aviation, cyber security, and many other fields [86]. Particularly in crime prediction systems, decision trees has been superior than algorithms such as Naive Bayes, and SVM.

The Decision Tree Induction Algorithm is shown in Algorithm 3.1. This algorithm is fed a set of training instances E and an attribute set F. The algorithm works by recursively selecting the best feature to split the data and expanding the nodes of the tree until the stopping criterion is met. The details of the algorithms are explained as follows:

1. The **createNode()** function expands the decision tree by creating a new node. A node in the decision tree has either a class label, denoted as node.label , or a test condition, denoted as node.test cond.

2. The **find_best_split()** function computes the attribute test condition for partitioning the training observation associated with a node. The splitting feature chosen depends on the impurity measure used. Predominantly used measures include Gini index and entropy.

3. The **Classify()** function computes the class label to be assigned to a leaf node. For each leaf node t, let $p(i|t)$ denote the fraction of training observations from class i associated with the node t. The label assigned to the leaf node is the typically the one that frequently occurs in the training observations that are associated with that particular node.

$$leaf.label = \underset{i}{argmax}\ p(i|t) \tag{3.1}$$

where the argmax operator outputs the class i that maximizes $p(i|t)$. Other than providing the information needed to com the class label of a leaf node, $p(i|t)$ can also be used as a rough estimate of the probability that an observation assigned to the leaf node t belongs to class i.

4. The **stopping_cond()** function is used to halt the tree-growing process by evaluating whether all the observations have the same class label or feature values. Since decision tree classifiers uses a top-down, recursive partitioning approach for constructing a model, the number of training observations associated with a node decreases as the depth of the tree increases. Consequently, a leaf node may contain too few training observations to make a statistically significant decision about its class label. This phenomenon is known as the data fragmentation problem. One way to avoid this problem is to disallow splitting of a node when the number of observations associated with the node fall below a certain threshold.

3.1.1 Splitting Criterion

Recursive partitioning is the core concept underpinning decision trees [87]. At first, there would be all of the observations, which are represented by the node at the top of the tree. This parent node is divided into two or more child nodes by the algorithm in a way that ensures that the response variable values (or levels) in each child region are as comparable as feasible [88]. Each of the child nodes goes through the

Algorithm 1 Decision Tree induction algorithm

```
1: function TreeGrowth(E, F)
2:     if stopping_cond(E,F) = true then
3:         leaf=createNode()
4:         leaf.label=Classify(E)
5:         return leaf
6:     else
7:         root =CreateNode()
8:         root.test_cond= find_best_split(E, F)
9:         let V= v|v is a possible outcome of root.test_cond
10:        for each v ∈ V do
11:            E_v= {e | root.test_cond(e) = v and e ∈ E}
12:            child = TreeGrowth(E_v, F)
13:            add child as descendent of root and label the end (root → child) as v)
14:        end for
15:    end if
16:    return root
```

splitting procedure again, and the recursion keeps going until a halting requirement is met. The tree is deemed complete once that happens (grown). In essence, recursive partitioning subdivide the predictor space in such a way that the values of the target variable for the observations within a leaf node are similar as possible; i.e homogeneous. This section provides an overview of the splitting criterion used when building decision trees for both classification and regression trees [89].

3.1.1.1 Gini Impurity

Primarily used in CART(Classification and Regression Trees), Gini Impurity measures the rate by which a randomly selected sample from the subset would be misclassified if classified randomly according to class distribution in the subset. The Gini Impurity is the probability of incorrect classification of the randomly selected sample [90].

$$\text{Gini}_X(t) = \sum_{i=1}^{c} [p(i)]^2 \tag{3.2}$$

$$\text{Gini\_Impurity}(t) = 1 - \sum_{i=1}^{c} [p(i)]^2 \tag{3.3}$$

$$Classification\ \ error(t) = 1 - \max_i[p(i|t)]. \tag{3.4}$$

For both equations 3.2 and 3.2, c and $p(i)$, represent the total number of classes and probability of picking an observation class i, respectively. Essentially, Gini Impurity is 1 minus Gini as seen on equation 3.2.

Then, a feature(X) with the lowest Gini Impurity is preferred in decision Tree Construction. Gini Indices only performs binary splits, thus, mainly used in CART.

3.1.1.2 Entropy

Entropy is predicated upon information theory initially introduced by Calude Elwood Shannon in 1948 [91]. By definition, entropy is a measure of uncertainty of a random variable. Entropy is utilized to compute the homogeneity of a sample in a dataset in order to divide the dataset into several subsets. If the subset is homogeneous then entropy is zero and if data points can be equally divided then entropy is one. Entropy measures impurity of a dataset, lower entropy values results in lesser information content. Entropy is indicative of how informative a node is [92]. Entropy is calculated as indicated in equation 3.5.

$$Entropy(D) = -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t). \tag{3.5}$$

where D is the dataset for which the entropy is calculated and $p(i|t)$ is the proportion of a number of data points in class x_i to the total number of elements in D and c is the number of classes.

Entropy utilizes probability distribution of a sample drawn from the dataset to characterize data. Hence, entropy is referred to as the negative logarithm of the probability distribution of sample's occurrence in the sample space. $Entropy(D) = 0$ when correctly classified, meaning all data points in the dataset D belong to 1 class.

3.1.1.3 Information Gain

Contrary to Gini Impurity being used primarily by CART, information gain is primarily used by Iterative Dichotomiser 3(ID3), C4.5 and C5.0 .The previous section has indicated that entropy changes when the root node changes. Thus, decision tree construction is about finding the best feature to be the root node, indicated by high information gain. By definition, information gain is a measure of homogeneity in a dataset. Information gain is based on entropies, the following indicate entropies that need to be calculated to obtain information gain.

1. Dataset entropy as indicated in 3.5
2. Split dataset based on difference feature and subsequently calculate the entropy of each branch generated thereby,
3. Calculate the total entropy of the resultant decision tree.

So, the information gain is the difference between entropy calculated before carrying out the split and the resultant entropy.

$$\mathrm{IG}(A, B) = \mathrm{Entropy}(D) - \sum_{i=1}^{c} P(T_i)\mathrm{Entropy}(Ti). \tag{3.6}$$

Where D is the dataset for which the entropy is calculated, A is the feature chosen for the split,B is the the dataset subset, T_i is a subset created after splitting with feature A and $P(T_i)$ is the proportion of the number of elements in T_i to the number of elements in D.

3.1.1.4 Information Gain Ratio

Previously discussed information gain has a major drawback, and that is it prefers to use features with larger numbers of distinct values as the best feature to split by, to use as decision nodes. Information gain is biased towards multi-valued features and that bias needs to be corrected. To counter that bias, intrinsic information of a feature is calculated, intrinsic information is defined in equation 3.7. Where S is the dataset, S_i is a subset of the dataset after splitting and $|S|$ and $|S_i|$ are the number of observation belonging to the original and subset dataset, respectively.

$$\mathrm{InstrinsicInfo}(A, S) = -\sum_{i=1}^{n} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}. \tag{3.7}$$

If the intrinsic information value of a feature is high, then that feature contains sparse values thus, less informative. If intrinsic information value of the feature is less, then the feature contains fewer distinct values thus, more informative. Intrinsic information value is used to normalize information gain, by penalizing features with larger numbers of distinct values. Equation 3.7 provides information gain equation, equation 3.8 is information gain ration equation, which is a ratio of information gain and intrinsic information value.

$$\mathrm{Gain}(A, S) = \frac{\mathrm{IG}(A, S)}{\mathrm{InstrinsicInfo}(A, S)}. \tag{3.8}$$

The dataset, S, is split on the basis of information gain ratio, the feature with highest information gain ratio is used to split the dataset, thus used as a split node.The idea of information gain ratio was initially introduced by Ross Quinlan [90].

3.1.1.5 Chi-square

Chi-square (χ^2) measures the statistical significance of the difference between the child node and the parent node. χ^2 is a measure of the standardized difference between observed and expected frequencies of the target variables for each node [93].

$$\chi^2 = \sum \frac{\mathrm{observed}_i - \mathrm{expected}_i}{\mathrm{expected}_i}. \tag{3.9}$$

Larger chi-square value indicated greater statistical significance of the difference between child node and a parent node. Chi-square is computed by evaluating the deviation of success and failure of each observed node in the tree. The decision tree construction carries out multiple splits, for each one of those splits chi-square is the sum of all chi-square of the success and failure for every node. A tree generated by the chi-square is called Chi-Squared Automatic Interaction Detector(CHAID). Primarily, chi-square is used to determine whether further splitting of a node will improve performance on the dataset or of a particular sample of the training dataset. Feature selection is machine learning independent, feature

scores are evaluated using different statistical methods to determine their correlation with the target variable. Feature to be used for splitting is determined by the algorithm by calculating scores and probability values associated with the scores. Then when constructing the tree, the feature to be considered for splitting is the one with maximum scores.

3.1.2 Chi-Square Automatic Interaction Detector (CHAID)

Chi-square Automatic Interaction detection detector is one of decision tree techniques predicated upon Bonferroni testing [94]. The technique was developed and published by Gordon.V Kass in 1980 [95]. Similarly to regression analysis, CHAID is used for classification problems to determine relationships between variables. CHAID is a decision tree technique that uses previously discussed chi-square split criterion instead of every other split criterion, i.e, information gain, information gain ratio, entropy and gini index.

Kass(1980) states that the CHAID algorithm operates using a series of merging, splitting and stopping steps based on a criteria specified by the users.

Pseudocode below presents a merging step which operates by taking each feature and CHAID merges non-significant categories;

1. For every feature, cross-tabulate the features with binary target values.
2. If the feature has only 2 categories, skip to step 6.
3. Perform chi-square for independence of each pair of categories of the features relating to the binary target variable using chi-square distribution with significance test(α_{merge})set to 0.05. If outcomes are nonsignificant, those pairs are not merged.
4. Nonsignificant test are identified by $\alpha_{\mathrm{merge}} > 0.05$, paired categories are paired into a single category. Tests with significance identified by $\alpha_{\mathrm{merge}} \leqslant 0.05$ are not merged together.
5. Categories with less than the minimum segment size specified by the user are merged with similar categories.
6. Type 1 error rate is controlled by adjusted P value for the merged categories using the Bonferroni adjustment.

After all the possible merges for each feature have been determined, the splitting step follows. This step serves as a selection criterion to select the best feature to be used as a split node using the following algorithm.

1. Chi-square test for independence utilizing the P value for every feature.
2. The most statistically significant feature, feature with smallest P value, is split if the P value is less than the significance level (α_{split}) specified by the user at 0.05 else, the node is not split thus considered a leaf node.

The stopping step uses rules specified by the user to determine if growing the tree should be halted, the rules are as followed:

1. If maximum tree depth level is reached, current tree growing is halted.
2. If node size reaches user-specified node size, the node will not be split any further.
3. If further splitting of a node results in a child node with a size value less than that specified by the user, further split of a node will be halted. The parent node is a node level that further splitting it will result in child nodes that can become parents nodes themselves or leaf nodes
4. The CHAID algorithm continues until all the stopping rules are satisfied.

3.1.3 Classification And Regression Trees(CART)

CART is a recursive partitioning decision tree that has both classification and regression, classification produces categorical target variables and predicts their labels. On the other hand, regression processes continuous variables and predicts their values. CART constructs binary trees making use of the following stopping conditions [83]:

1. A minimum number of records have to be traversed
2. Assigning a majority of the records in the dataset to nodes, and
3. Splitting for maxim number of levels

It is not possible to split if the dataset contains duplicate entries, has only a single entry and is homogeneous.

3.1.3.1 CART Pseudocode

The three major steps of the CART algorithm are as follows:

1. Find the best split for each feature. If there exists k different values for each feature in the training dataset, then the possible number of splits can be k-1
2. Once the best split is found for each feature, find that split which maximizes the splitting criterion.
3. Split the node using the best splitting criterion found in Step-2 and repeat from Step-1 until the stopping criterion is satisfied.

3.1.3.2 Pruning in CART

CART algorithm produces decision trees that are grown larger than what is required. Hence, it is pruned back to find the optimal tree. CART uses test data or X-fold cross-validation to determine the best tree that efficiently classifies unknown incoming samples. To validate the tree using a test dataset, the tree is scored by making use of samples that were not used by the tree during its training phase. Cross-validation evaluates the tree by training the decision tree model on a certain number of samples from the distribution. Pruning can be done as follows [96]:

1. Split the training data randomly into X folds.
2. Train X trees on different combinations of X-1.
3. X-1 folds, and find the estimated error for the Xth fold each time.
4. Find the accuracy of the decision tree by making use of the mean of the X error measurements.
5. Make use of parameters such as complexity penalty to minimize the error in Step-3. Complexity parameter controls the size of the tree and selects the optimal tree. The tree stops building if another variable is added at the current node position to the decision tree and the cost is more than the value of the complexity parameter.

6. Now refit the tree using the data and parameters obtained from Step-4.

3.1.4 Pruning

Using tightly stopping criteria generally constructs small and under–fitted decision trees. On the contrary, employing loosely stopping criteria generally construct large decision trees that are over–fitted to the training set. In 1984 Breiman originally suggested a pruning method developed for this dilemma [97]. The size of the trees is reduced by pruning whereby redundant or non-critical sections of the tree are trimmed(removed). It has been shown that pruning improved generalization performance thus, improves classification accuracy while reducing complexity of the final model. Data is divided into training data and cross validation data, decision tree learning is carried out on a training set and trimming is carried out on the validation set while optimizing predictive accuracy.

There are various pruning algorithms in existence, some perform top-down traversal some bottom-up traversal. Pruning occurs when a certain criteria is improved. In the following sections various pruning techniques will be discussed.

3.1.4.1 Reduced Error Pruning

This is one of the simplest pruning methods introduced by Quinlan in 1987. Utilizing the bottom-up traversal approach, the procedure traverses the internal nodes and check if replacing the node with the most frequent class will decrease the tree's accuracy. If so, the node is pruned. The procedure continues until continuous pruning negatively affect the tree's accuracy. In [98] Quinlan suggested that to estimate accuracy, a separate pruning set is used for pruning.

Reduced error pruning method generates series of pruned trees straight from the test data, instead of only using it for the selection of the optimal tree [98]. The method is as follows: Test data is ran through the entire data at first, the numbers that appear in each class in each node at the method progresses. At every non-leaf node, count the number of errors if the sub-tree is kept and the number if it becomes a leaf as a consequence of pruning. The pruned tree will usually makes minimal errors on the test data than the sub-tree makes. The difference between the number of errors is a measure of the gain from pruning the tree. The gain from the pruning of the tree is measured by the difference between the numbers of errors(if positive). From all the nodes the one with the largest difference as the

sub-tree is chosen to prune. The process is continued including the node where the reduction is zero until further pruning would results in an increase in miss-classification rate. This result in a smallest version possible of the most accurate tree respect to the test data.

The number of sub trees with the same difference can be obtained, which is the largest difference. According to [98], it is not specified which sub-tree to choose from in that situation. For instance, the larges or the smallest difference. Experiments show that the choice makes little to no difference in term of classification accuracy; thus, the largest is often chosen on the basis of it reduces the number of iterations required to completely prune the tree.

This method creates a number of trees, ending with the smallest possible accurate minimum-error tree on the with respect to test data set.

3.1.4.2 Minimum Description Length Pruning

There exist a number of different methods to prune Decision Trees based on Minimum Description Length (MDL) principle that was proposed by (Quinlan and Rivest [99]; Wallace and Patrick [100]; Mehta et al [101]; Oliveira et al [102]; Kononenko [103]). All of the above mention authors share the same sentiments that, the best classification tree constructed for a particular dataset is the one that offers the minimum length of the description of class labels for the training data objects. Such approaches address the trade-off between tree size and the number of exceptions from tree decision. Naturally, a smaller tree has a shorter encoding, but a larger portion of the training data is misclassified by the tree, necessitating additional code for the exceptions. In other words, a tree leaf can be described relatively short by the code of the class if it contains just objects from that class. However, if the leaf contains objects from many classes, the class assignment for each object must be encoded, making the description substantially longer.

3.1.4.3 Cost-Complexity Optimization Pruning

This method of pruning was introduced in 1984 by Breiman [97] for pruning CART after learning because stop criteria are not flexible or accurate for post learning pruning methods. Despite the fact that CART has a stop condition in the form of a minimum node size, the cost-complexity optimization is the fundamental method for adjusting tree size to a given problem(minimum number of training data

dropping into the node). The optimization, as the name implies, is concerned with a risk measure including both misclassification cost and model size (complexity), defined as the number of leaves of the tree, rather than only training error minimization, which usually leads to overfitting the training data. Breiman proposed α to control the trade-off with α parameter in a measure of decision tree miss-classification cost involving tree size defined as the number $|T|$ of leaves of the tree T:

$$R_\alpha(T) = R(T) + \alpha|T| \tag{3.10}$$

Where $R(T)$ represent the general misclassification cost. In order to evaluate the optimal value of α, in 1984 Breiman [97] defined validation procedures which estimate performance of the possible α values and select the best one. There are significant properties that have been demonstrated in [97] that show how effective the pursuit of α's optimal values may be. The first property to have been noticed is as follows.

Property 1. There is a unique smallest subtree $T_\alpha \prec T$ minimizing R_α for each value of α.

Property 2. There is a distinct increasing sequence $\alpha_i, ..., \alpha_n$ as well as a decreasing tree sequence $T_i \succ, ... \succ T_n$, such that $\alpha = 0$, $|T_n| = 1$ and for all $i = 1, ..., n, T_i$ minimizes R_α for each $\alpha \in [\alpha_i, \alpha_{i+1})$ (for precision,additionally, $\alpha_{n+1} = \infty$ is defined).

In Property 1., it means that if any other subtree $T' \prec T$ also minimizes R_α, then $T_\alpha \prec T'$. Further reasoning proved a theorem rephrased as property Property 2., laying the groundwork for cost-complexity optimization and cost-complexity validation methodology.

More casually: all feasible trees that optimize R_α are explored by a single decreasing sequence of αs, and the trees are also exactly ordered. As a logical outcome, one can operate sequentially to identify all the αs by starting with the entire tree and selecting the nodes that need to be pruned one at a time (where pruning two or more nodes is dependent on the same value of, they must be pruned simultaneously).

3.1.5 Decision Trees Ensemble Methods

Ensemble methods train a multitude of learners to solve the problem at hand. Contrary to ordinary learning techniques which attempts to build one learner from the training data, ensemble methods attempts to build a set of weak learners and put them together. Ensemble learning methods are also refereed to as multiple classifier system or committee-based learning [104].

An ensemble has a multitude of learners called weak learners or base learners. Weak learners are generated from training data using a weak learning algorithm which can either be a neural network, Naive Bayes, decision tree or any other learning algorithm. Predominantly, ensemble methods if constructed by a single weak learning algorithm to create homogeneous weak learners, i.e learners of the same kind which leads to homogeneous ensembles, however, there are also some other methods that uses a multitude of weak learners to create what is referred to as heterogeneous learners which are weak learners of different types which leads to heterogeneous ensemble methods [105].

The generalization capability of ensemble methods is usually superior than that of weak learners. In fact, ensemble methods are attractive mainly because they are able to boost weak learners which are better than random guess to strong learners which usually make accurate predictions [106].

There are three distinct threads of early contributions to current field of ensemble methods; i.e., mixture of experts, combining classifiers and ensemble of weak learners. Mixture of experts was predominantly studied in the community of neural networks. In this thread, researchers mostly consider a divide-and-conquer approach, attempts to a mixture of parametric models jointly and utilize combining rules to attain an overall solution. Combining classifiers is predominantly studied in community of pattern recognition. In this thread, researchers usually work on strong classifiers, and attempt to design powerful combing rules to achieve stronger combined classifiers. Consequently, this thread work has accumulated elaborate understanding on the design and use of a number of combining rules. Ensemble of weak learners is mostly studies in the community of machine learning. Here, researchers usually work on weak learners and attempts to design and construct powerful to enhance the performance from weak to strong. Consequently, this thread of work resulted in the creation of well-known ensemble methods such as Bagging, AdaBoost, and many more, etc., and a theoretical comprehension of how and why weak learners can be enhanced to strong ones [107].

3.1.5.1 Bagging

It has been discussed that bagging falls under the ensemble of weak learners thread. Generally, there are two paradigms of ensemble methods that determines how the weak learners are generated. One, the sequential ensemble methods to which weak learners are constructed in a sequential manner, AdaBoost is a representative in this paradigm. Two, parallel ensemble methods where the weak learners are constructed in a parallel manner with Bagging [55] as a representative. Contrary to sequential ensemble methods, parallel ensemble methods exploits the independence between weak learners, thus, the error can be drastically reduced by a combination of weak learners. Given that weak learners are generated from the same training dataset it is practically unattainable to get absolutely independent base learners, however, independence can be obtained by the introduction of randomness in the learning process, thus, a good generalization capability can be expected by the ensemble. Furthermore, another advantage of parallel ensemble method favourable to parallel computing, thus, training speed can easily be accelerated by the use of parallel computers or multi-core computing processors. This is rather appealing as multi-core processor commonly available these days.

This research focused on the Decision Trees as week learners, so, referenced to weak learners is referring to the decision trees.

3.1.5.2 Random Forest (RF)

RF is a clear representation of the state of the art ensemble methods. RF is the amalgamation process of a multitude of decision trees at training. For classification problems, RF outputs a majority vote and for regression problems, it outputs an average of the predictions from all the individual trees [54].RF is extended from Bagging where it differs with Bagging since it incorporate the randomized feature selection approach. When constructing the decision tree as the base learner, at every step of the split selection, RF select the subset of features at random, and then proceed to use the normal split selection technique within the selected feature subset.

Algorithm 2 presents an adaptation of the Random Forest algorithm from [108].

Algorithm 2 Random Forest Pseudo-code

1: **for** b=1 to B **do**

a) Draw a bootstrap observation Z^* of size N from the training data

b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} reached.

i) Select m variables at random from the p variables.

ii) Pick the best variable(split-point) among the m.

iii) Split the node into two daughter nodes.

2: **end for**

3: Output the ensemble of trees $\{T_b\}_1^B$ =0

3.1.5.3 Extremely Randomized Trees (ERT)

Similarly to RF, ERT is also an amalgamation of a multitude of weak decision trees learners. ERT also outputs the majority vote for classification problem and mean for regression problems. However, contrary to RF, ERT used the entire dataset to train weak learners instead of a bootstrap replica as done in RF. Algorithm 3 presents an adaptation of the ERT pseudo-code from [60].

Algorithm 3 Extremely Randomized Tree Pseudo-code

1: **for** b=1 to B **do**

a) Select the the entire training dataset.

b) Grow a tree T_b to the selected data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} reached.

i) Select m variables at random from the p variables.

ii) Generate m possible splits.

1) Select cut-point at random.

2) Compute the best split variable.

iii) Split the node into two daughter nodes.

2: **end for**

3: Output the ensemble of trees $\{T_b\}_1^B$

3.1.5.4 Proposed Hybrid Algorithm (RF-Plus)

As discussed in the literature review section, the proposed hybrid algorithms comprises of the best attributes from the RF and ERT. Algorithm 4 shows the pseudo-code of the proposed RF-Plus model [109].

Algorithm 4 Pseudocode of the Proposed Hybrid RF-Plus model

1: **for** b=1 to B **do**

a) Draw a bootstrap observation Z^* of size N from the training data

b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} reached.

i) Select m variables at random from the p variables.

ii) Generate m possible splits

1) Select cut-point at random

2) Compute the best split variable

iii) Split the node into two daughter nodes.

2: **end for**

3: Output the ensemble of trees $\{T_b\}_1^B$

Table 3.1: Description of Parameters for Algorithms 2, 3 & 4

Parameter	Descriptions
N	Training set size
Z^*	Bootstrap observation
B	Number of Trees
b	Given Tree number between 1 and B
n_{min}	The minimum observation size for splitting a node
T_b	A single tree of the ensemble of trees
T	An ensemble of trees

3.2 Naïve Bayes

The Naïve Bayes classification algorithm can be used for both binary and multi classification problems [110]. It is also called the Idiot's Bayes due to its simplicity in calculating probabilities. It is based

on class conditional independence, which assumes that the degree to which a predictor (attribute) has effect on a given class is entirely independent of every other predictor (attribute) [111]. However, the conditional independence of predictors does not always holds depending on the nature of the predictors and the dataset [112]. Equation 3.11 presents a representation of conditional probability using Bayes theorem.

$$P(class|data) = \frac{P(data|class) \times P(class)}{P(data)} \tag{3.11}$$

Where, $P(class|data)$ is the posterior probability of a particular class (target variable) on given data (predictor, attribute or feature); $P(data|class)$ is the likelihood of the probability of a data (predictor, attribute, feature)) given a class (target variable); $P(data)$ is the prior probability of a predictor and $P(class)$ is the prior probability of a class.

Algorithm 5 presents Naïve Bayes pseudo-code with the various parameters and descriptions defined on Table 3.3.

Algorithm 5 Naïve Bayes Pseudo-code

1: Get the training dataset T
2: Compute the mean and standard deviation of the features for every class.
3: **For** i=1 **to** n **do**:
    a) Compute the probability of f_i utilizing the guass density equation of every class
4: Get the probabilities of all features (f_1, f_2, ... , f_n)
5: Compute the likelihood for every class
6: Obtain the highest likelihood

3.3 Support Vector Machines

There exist a wide variety of machine learning algorithms and the Artificial Neural Network(ANN) is one of them having developed in the early 1940's predicated upon the biological neurons structure of the human brain. Later in the 1980's it was applied to most engineering based application mainly due

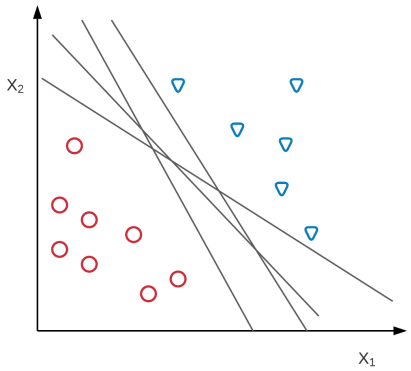


Figure 3.1: Two Class Linearly Separable data

to it's capability in extracting complex and non-linear relationships between features of a variety of systems [113], [114],and [115].

However, it was later discovered that the ANN required huge data for training to yield reliable results. As a consequence of the aforementioned ANN shortcomings, Vapnik developed a new machine learning technique called Support Vector Machine as a non-liner solution for classification and regression problems [116]. The success of SVMs can be attributed to at least three main reasons; Computational efficiency in comparison to other machine learning techniques, requires relatively small number of features to train ,and robustness against model's error [117], [118].

3.3.1 Linearly Separable Data: Hard Margin

Let the i-th observation be presented by (X_i, y_i) where X_i is a feature vector and y_i is the target variable with associated class labels with two possible values $+1$ and -1. In Fig.3.1 the blue triangles have class label $+1$ and the red circle has class label -1. A straight line can be drawn to separate the data into 2 respective classes. The represented above in a two-dimensional space is clearly linearly separable.

However, there exist an infinite number of lines that can be drawn to separate the $+1$ class from the -1 class. With that being said, the question that can be posed is; Which among the many lines is optimal, meaning, it can have low classification error on a new observation ? SVM attempts to create a decision boundary known as a hyperplane that separates the two classes. SVM proceed to choose a hyper plane which will have the largest margin between the two classes, thus SVM are also referred to as largest margin classifier. The largest margin hyperplane is also know as "optimal hyperplane" as indicated in

Fig.3.2.

Formally, the hyperplane in a n-dimensional space is an n-1 dimensional subspace. For instance, in a two dimensional space, the hyperplane is a one dimensional straight line. For a set of n training samples, $x_i, (i = 1, 2, \ldots, n)$, the optimal hyperplane is defines (see 3.12) as:

$$w^T x + b = 0 \tag{3.12}$$

Where w is the n-dimensional normal vector and b a bias term. Data that lies closest to the optimal hyperplane either from the right or from the left is referred to as the support vectors. For the left side of the optimal hyperplane, support vectors lies on equation 3.13:

$$w^T x + b = 1 \tag{3.13}$$

And for the right of the hyperplane, support vectors lies on equation 3.14:

$$w^T x + b = -1 \tag{3.14}$$

Combining equation 3.13 and 3.14 equation 3.15 was obtained.

$$y_i(w^T x + b) - 1 = 0 \tag{3.15}$$

There are two specific properties that categorise the optimal hyper-plane: must maximise the distance from support vectors of each class, and it must have the smallest possible data separation error [119].

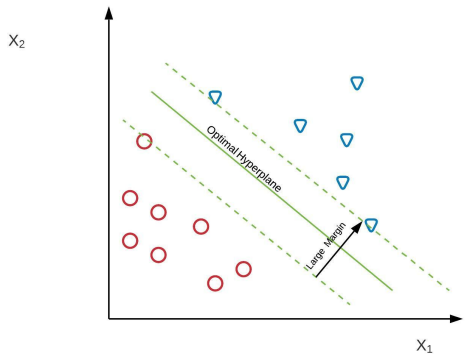


Figure 3.2: Two Class Separated by the Maximum Margin Optimal Hyperplane

Obeying these properties, data falls on either of two sides of the optimal hyper-plane, the left ($y = 1$) or right ($y = -1$) as it can be seen in equation 3.16.

$$w^T x + b \begin{cases} \leqslant 1 \text{ for } y_i = 1 \\ \leqslant -1 \text{ for } y_i = -1 \end{cases} \tag{3.16}$$

As stated above that the generalization region for the optimal hyperplane could be anywhere between 1 and -1, thus, there could be infinite margins considered for each respective class. So, the margin distance d between the left and right margins must be measured and maximised. Recall the equation between two parallel planes presented in equation 3.16.

$$d = \frac{|(w^T x + b - 1) - (w^T x + b + 1)|}{\|w\|} = \frac{2}{\|w\|} \tag{3.17}$$

To maximise equation 3.17 which implies minimization of the inverse of the inverse given by $\frac{\|w\|}{2}$. For the sake of mathematical efficiency, the equation to be minimized is $\frac{1}{2}(\|w\|^2)$. Then an optimization equation to obtaining the optimal hyperplane can be expressed as:

$$Min_{w,b} = Min\ \frac{1}{2}(\|w\|^2) \quad s.t \quad y_i(w^T x + b) \leqslant 1 \tag{3.18}$$

Making use of Lagrange multipliers as indicated in equation 3.19, a solution to equation 3.18 can

be obtained with subject to the constraint of the margins of both classes. With α_i as the Lagrange multiplier, the optimization problem is expressed as follows:

$$L(w, b, \alpha) = \frac{1}{2}(\|w\|^2) - \sum_{i=1}^{N} \alpha_i [y_i(w^T x + b) - 1] \tag{3.19}$$

To find the saddle point in equation 3.19, Karush -Kuhn -Tucker (KKT) conditions expressed equation 3.20 and 3.21 had to be satisfied.

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{N} \alpha_i x_i y_y \tag{3.20}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow 0 = \sum_{i=1}^{N} \alpha_i x_i y_i \tag{3.21}$$

It is worth noting that α_i will not be equal to zero if and only if the corresponding input x_i is a support vector [120]. The general equation for a linearly separable data is obtained by substituting back equation 3.20 and 3.21 to equation 3.19 subjected to two constraints as indicated in equation 3.22.

$$\begin{aligned} \text{Max} \quad L_d(\alpha) \quad &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ s.t \quad &\begin{cases} \alpha_i \leqslant 0 \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \end{aligned} \tag{3.22}$$

The Support Vector Machine algorithm then uses Equation 3.22 and its constraints to find support

vectors and their corresponding input data. The parameter w of the hyperplane can then be computed and written as shown in equation 3.20, and the parameter b can then be calculated and written in the mean form as shown in equation 3.23.

$$b_o = \frac{1}{N}\sum_{i=1}^{N}(y_s - w^T x_s) \tag{3.23}$$

3.3.2 Linearly Non-Separable Data: Soft Margin

In the preceding subsection, SVM classifier where data can be classified into two classes is discussed, thus, linearly separable. However, in practice data is not always going to be linearly separable. This is where the concept of a soft margin comes from. Greater margin for separating hyperplane is often achieved by violating the constraints for a few data points rather than trying to fit a strictly separating hyperplane. The resultant separating hyperplane has a relatively tiny margin if the constraints are strictly obeyed for all data points. Consequently, greater margins are obtained by allowing a few mistakes while splitting the training data.

Only if a penalty function is introduced in such a way that the distance (ξ_i) between the wrongly separated data points of each class from the margin of each respective class can be measured and minimized can a linear SVM achieve good performance for this problem.

In these cases, the penalty function can be defined as follows [120]:

$$F(\xi) = \sum_{N}^{i=1} \xi_i \tag{3.24}$$

Consequently, the convex optimization function presented in in equation 3.18 changes for the linearly non-separable with the introduction of the slack variable (ξ_i) in the penalty function and becomes:

$$Min_{w,b} = Min\ \frac{1}{2}(\|w\|^2) + C\sum_{N}^{i=1}\xi_i \quad s.t\ \ y_i(w^T x + b) \geqslant 1 - \xi_i \tag{3.25}$$

To maximise the margin and minimize the classification error a parameter C know as "Trade-Off" is introduced as it can been seen in equation 3.25 [121].

To address the optimization issue in equation 3.25, which is subject to margin constraints, Langrage Multiplies ($\alpha(\alpha_i), \beta(\beta_i)$) are used. Equation 3.26 can thus be written in its unconstrained form as:

$$L_p(w, b, \xi, \alpha, \beta) = Min\ \frac{1}{2}(\|w\|^2) + C\sum_{N}^{i=1}\xi_i - \sum_{1=1}^{N}\alpha_i\{y_i[w^T x_i + b] - 1 + \xi_i\} \quad - \sum_{i=1}^{N}\beta_i\xi_i \tag{3.26}$$

By satisfying the Karush -Kuhn -Tucker (KKT) condition, the optimal solution for equation 3.26 can be obtained.

$$\frac{\partial L}{\partial w} = 0 \Rightarrow \sum_{N}^{i=1}\alpha_i y_i x_i \tag{3.27}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{N}^{i=1}\alpha_1 y_i \tag{3.28}$$

$$\frac{\partial L}{\partial b} = \alpha_i + \beta_i = C \tag{3.29}$$

Equations 3.27 through 3.29 can be substituted into equation 3.26 to obtain equation 3.30, which is the soft margin for the dual problem.

$$\begin{aligned} Max L_d(\alpha) &= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{N}^{i=1} y_i y_j \alpha_{jj} x_i^T x_j \\ s.t &\begin{cases} 0 \leqslant \ \alpha \ \leqslant C \\ \sum_{N}^{i=1}\alpha_i y_i = 0 \end{cases} \end{aligned} \tag{3.30}$$

Between equations 3.22 3.33 the difference is the constraint posed on the Lagrange multiplier compelling it to be between 0 and the value of the parameter C.

3.3.3 Non-linear Data: Kernel Function

3.3.3.1 Feature Space Transformation

There are three types of input data: linear, linearly non-separable, and non-linear. When a linear separating hyperplane cannot be used to separate the data, it is said to be non-linear. The linear separating hyperplane suffices for linearly separable data, and it also suffices for linearly non-separable data when slack variables are used, as previously described. The input features are now transferred onto a higher dimensional space for non-linearly separable data using a method known as Kernel Method [122]. Although data remains non-linear, it can now be divided into two or more sub-spaces using appropriate SVM classifiers in a higher dimensional space.

In accordance with Mercer's Theorem [122], the input data x is represented by $\Phi(x)$ in the feature space despite the fact that prior knowledge of the formal definition of Φ is not required. Furthermore, in the feature space, the only thing required is the calculation of the inner product, not the show of the input data [119]. By selecting the advanced kernel function in equation 3.31, the inner product can be computed. After that, it's possible to use SVM to solve nonlinear problems.

$$\Phi(x_i, x_j) = K(x_i, x_j) \tag{3.31}$$

It should be noticed, however, that equation 3.31 is only valid if and only if the integration equation 3.32 can be satisfied for a nonlinear vector function like $h(x)$ [119].

$$\int K(x_i, x_j) h(x_i) h(x_j) dx_i dx_j \geqslant 0 \tag{3.32}$$

There are several Kernel Functions that may be used to separate non-linearly separable data by changing it into a higher dimensional space where it is linearly separable. Table 3.2

Table 3.2: Various kernel functions used for nonlinear data classifications

Type of Classifier	Kernel Function
Sigmoid	$K(x_i, x_j) = (\alpha(x_i \cdot x_i) + \vartheta)$
Multilayer perceptron	$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + \mu)$
Linear	$K(x_i, x_j) = (x_i^T x_j)^\rho$
Guassian RBF	$K(x_i, x_j) = \exp(\frac{-[\|x_i - x_j\|^2]}{2\upsilon^2})$
Dirichlet	$\frac{\sin(\frac{n+1}{2})(x_i - x_j)}{2\sin((x_i - x_j)/2)}$

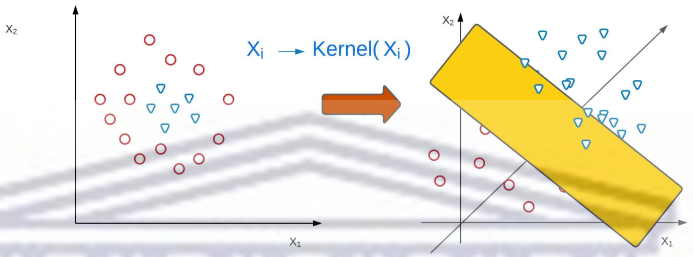


Figure 3.3: Input Data Mapping into a Higher Dimensional Space for better Separation

3.3.4 Kernel Trick

While the research has focused on using the inner product to determine input data in the feature space, the general dual equation for the classification of linearly non-separable data, as shown in equation 3.30, may be reformulated for nonlinear classification as represented in equation 3.33.

$$\begin{aligned} \mathrm{Max} L_d(\alpha) &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{N}^{i=1} y_i y_j \alpha_{jj} K(x_i, x_j) \\ \text{s.t} &\begin{cases} 0 \leqslant \alpha_i \leqslant C \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \end{aligned} \tag{3.33}$$

Finding the best hyperplane under these conditions, however, is not an easy or straightforward operation due to the unknown value of the vector w as indicated in equation 3.34 in the feature space.

$$w = \sum_{i=1}^{N} y_i \alpha_i \varphi(x_i) \tag{3.34}$$

On these occasions, the kernel trick can be used to avoid having to determine the parameter w directly.

If equation 3.34 is substituted into equation 3.23, the bias parameter b can be calculated using the kernel function as in equation 3.35:

$$b = y_i - \sum_{i,j=1}^{N_{SV}} y_i \alpha_i \varphi(x_i) K(x_i, x_j) \tag{3.35}$$

With this knowledge, the hyperplane is defined as equation 3.36:

$$d(x) = w^T \varphi(x) + b \tag{3.36}$$

By putting equation 3.34 into equation 3.36 and considering a suitable kernel function, now the optimal decision function can be obtained. As a result, equation 3.37 is the general equation for the optimal hyperplane.

$$d(x) = \sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b \tag{3.37}$$

As a result, determining the weighting parameter w in the feature space is no longer necessary, and SVMs can be utilized to solve nonlinear problems effectively by selecting an appropriate kernel function.

3.4 Recursive Feature Elimination

Recursive Feature Elimination (RFE) is a variant of the backward method of selecting features [123], which initially utilizes the entire set of features, then scores each feature based on its contribution to the overall performance of the model. There are various feature importance ranking techniques used as ranking criteria. One possible use of feature ranking is the construction of a class predictor (or classifier) based on a pre-selected set of features. Each feature that is correlated (or anti-correlated) with the separation of interests is, in and of itself, a poor class predictor. This results in a simple weighted voting classification strategy in which features vote proportionally to their correlation coefficient. Thus, in this study, this was used as a ranking criterion. Poorly performing features are then eliminated and the model is re-built on the remaining set of features. The elimination process continues at each loop, thus a "recursive" elimination.

In Algorithm 6 the pseudo-code for the RFE method as adapted from [124] is presented with parameters and descriptions presented in Table 3.3.

Algorithm 6 Recursive Feature Elimination Pseudo-code

1: Get the training dataset T

2: **For** $i=1$ **to** n **do**:

a) Compute ranking criteria M

b) Rank set F utilizing $M(T, F)$

c) $f^* \leftarrow$ least ranking feature in set F

d) $R(n-i+1) \leftarrow f^*$ eliminate least ranking feature

e) $F \leftarrow F$ - f^*

3.5 Support Vector Machine and Naïve Bayes Enhanced by Recursive Feature Elimination

An algorithmic representation of the Naïve Bayes enhancement (RFE-NB) is presented in Algorithm 7 with the parameters used and their descriptions summarized on Table 3.3.

Algorithm 7 Naïve Bayes Enhanced by Recursive Feature Elimination Pseudo-code

1: Get the training dataset T

2: **For** $i=1$ **to** n **do**:

   a) Compute ranking criteria M

   b) Rank set F utilizing $M(T, F)$

   c) $f^* \leftarrow$ least ranking feature in set F

   d) $R(n-i+1) \leftarrow f^*$ eliminate least ranking feature

   e) $F \leftarrow F - f^*$

3: Compute the mean and standard deviation of optimal features of every class.

4: **For** $i=1$ **to** n **do**:

   a) Compute the probability of f_i utilizing the Gaussian density equation of every class

5: Obtain probabilities of optimal features $(f_1, f_2, \ldots, f_n)$

6: Compute the likelihood for every class

7: Obtain the highest likelihood

Data: Dataset with p^* variables

Output: Ranked list of variables in accordence to their relevance.

Algorithm 8 Support Vector Machine Enhanced by Recursive Feature Elimination Pseudo-code

1: $p \leftarrow p^*$

2: **while** $p > 2$ **do**

3: $SVM_p \leftarrow SVM$ with the ideal tuning parameters for the samples and p variables in **Data**;

4: $w_p \leftarrow$ compute weight vector of the SVM_p $(w_{p1}, \ldots, W_{pp})$;

5: $min.rank.criteria \leftarrow$ variable with lowest value in rank.criteria vector

6: $rank.criteria \leftarrow (w_{p1}^2, .., w_{pp}^2)$

7: Remove $min.rank.criteria$ from **Data**

8: $p \leftarrow p-1$

9: **end while**

10: $Rank_1 \leftarrow$ variable in **Data** $\notin (Rank_2, \ldots, Rank_p*)$

11: **return** $(Rank_1, \ldots, Rank_p*)$

Table 3.3: Description of Parameters for Algorithms 5, 6 and 7

Parameters	Descriptions
T	Training dataset
$F = f_1, f_2, \ldots, f_n$	Set of n features
$M(T, F)$	Ranking method
R	Final ranked features
$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$	Gaussian Density equation
σ	Standard Division
μ	Mean
exp or e	2.71828
π	3.14159265359

3.6 Evaluation Metrics

Since different models were explored, it was required then to have a way fairly comparing them. The metrics considered were Accuracy, Precision, Recall, F1_Score, Error Rate and Execution Time. Accuracy is a measure of how often the model correctly classifies instances. Precision is the fraction of relevant instances among the successfully retrieved instances, while Recall is the fraction of relevant instances that were successfully retrieved [125]. The Error Rate which is the proportion of incorrectly classified instances incurred by the classifier and execution time was measured as a measure of model's complexity.

There exist a number of evaluation metrics that can be used to discriminate against a variety of classification algorithms. Researcher is then required to use appropriate evaluation metrics for analytical studies with respect to the problem domain. A confusing matrix, also refereed to as contingency matrix, can be utilized to analyse the performance of a classification algorithms with respect to the test dataset for which all the positive(i.e., true) values are known beforehand. Fig. 3.4 shows a visual depiction of how a confusion matrix is used for the classification of negative and positive examples. True Positive, TP, represent instanced where the predicted and actual values corresponds (i.e., positive), thus, true positive. True negatives, TN, represents instances where the prediction and actual values corresponds (i.e.,negative). False negatives, FN, represent instances, where the predicted value is negative and the actual value is positive. Lastly, false positives, FP, represents instances, where the predicted value is positive while the actual value is negative. Respectively, false negatives and false positives are referred to as *Type 1 errors* and *Type II errors*.

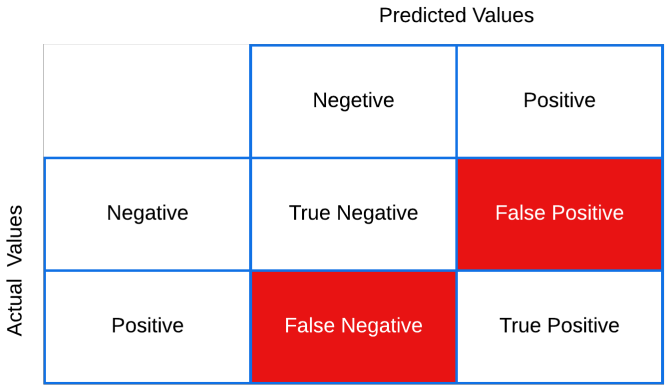


Figure 3.4: Confusion Matrix

Table 3.4: Metrics Derived from the Confusion Matrix

Metric	Formula
Accuracy	(TP+TN)/(TN+TP+FN+FP)
Precision	(TP)/(FP+TP
Recall	(TP)/(FN+TP)
F1 Score	(2 x*Precision* x*Recall*)/(*Precision*+*Recall*)
Specificity	T (TN)/(TN+FP) or 1 minus False-Positive Rate
Prevalence	(FN+TP)/(TN+TP+FN+FP)
False-positive rate	(FP)/(TN+FP

Optimal performance of any classification algorithm is obtained when false positives and false negatives are reduced as much as possible. Evaluation metrics represented in Table 3.4 for a classification algorithm can be obtained from statistical measures as presented in Fig 3.4. To evaluate evaluation metrics efficiently, dataset need to be partitioned. A labelled dataset, which has known class labels was taken and placed in a seperate dataset, that labelled dataset was referred to as *training dataset*. Then, a *testing dataset* was used to identify how a classification algorithm performs on unseen data. This will indicate generalization capabilities of the classification algorithm. In this case, both datasets contain labelled data.

3.7 METHODOLOGY

This chapter details the experimental process, it was then sub-divided into three main subsection; Experimental set up which details the particulars of the machine used to carry out the experiments,

followed by Data Description which details the two datasets used in this study including the sources where they were obtained from. Lastly, data pre-processing which details how data was pre-processed to ensure that it was ready to be fed through into the machine learning models considered.

3.7.1 Experimental Setup

Experiments were conducted on a Dell Desktop PC with Intel Core i5 10^{th} Gen Processor (Intel(R) Core(TM) i5-1035G1 CPU @ 1.19 Ghz) and 8GB of RAM. The interactive environment provided by Jupyter notebook was used to carry the experiments. Data processing and exploration, coding and evaluation of the predictive models were all done using Python and Jupyter Notebook. Furthermore, 10 fold cross-validation to evaluate the models was used.

3.7.2 Dataset Description and Pre-processing

This work used two crime datasets for experiment, which are (i) the Chicago dataset, and (ii) Los Angeles dataset. Due to bureaucratic reasons, it was difficult to obtain datasets from the local South African police stations. The Chicago dataset was obtained from the Chicago Data Portal which was extracted from the Chicago Police Department's (CLEAR) system [1]. The dataset contains 7.26 million records of crime data with 21 features collected between year 2001-2020 [1]. Furthermore, the study was expanded by utilizing the Los Angeles Dataset that was obtained from the City of Los Angeles data portal This dataset contains 2.12 million records of crime data with 24 features collected between year 2010 and 2019 [2].

The data was transformed into a suitable format for the ML problem. The CLEAR dataset initially had 32 unique crimes; however, not all were rampant. To this end, 5 frequently occurring crimes were selected, mainly because they are 68 % of the entire dataset, they are indicative of the crimes police officers should pay the most attention to. Thus, other rows that contained crimes that were not in the 5 most occurring crimes were removed. The Los Angeles dataset on the other hand had 110 unique crimes to which, also, the top 5 most occurring ones were selected.

Furthermore, the data contained categorical variables that were encoded to numerical variables using label encoding. Also, the target variable contained 5 crime categories as previously discussed, namely: Theft, Battery, Criminal Damage, Narcotics and Assault. For the Los Angeles dataset, crime categories

were: Robbery, Battery -Simple Assault, Assault With Deadly Weapon, Aggravated Assault, Intimate Partner - Simple Assault. These were encoded to numerals, 0 to 4 to represent the crimes respectively.

Moreover, one other form of data pre-processing is feature engineering. Using feature engineering, certain features that were identified to not having high predictive power were removed, these features were ID, X Coordinate, and Y Coordinate. To this end, the features left were in the CLEAR dataset: Description, IUCR, FBI Code, Arrest, Longitude, Community Area, Block, Beat, District, Location Description, Ward, Year, Case Number, Domestic, Updated On and Latitude. Additionally, from the Date feature, features; Day, Month, Hour, DayOfWeek and WeekOfYear were extracted. Since additional features were removed from the Date features, it was itself then removed. In total, there were had 21 features with the target variable "Primary Type". Table 3.5 provides descriptions for all 21 features used. Similar processes were carried out for the Los Angeles dataset, with the final feature set summarized on Table 3.6.

Table 3.5: Features considered in the Chicago dataset [1]

Feature	Description
Description	The secondary description of the IUCR code, a subcategory of the primary description
IUCR	The Illinois Unifrom Crime Reporting code
FBI Code	Indicates the crime classification as outlined in the FBI's FBI National Incident-Based Reporting System (NIBRS).
Arrest	Indicates if an arrest was made
Longitude	The longitude of the location where the incident occurred
Latitude	The latitude of the location where the incident occurred
Community Area	Indicates the community area where the incident occurred
Block	Partially redacted address where the incident occurred but within the same block as the actual address
Beat	Indicates the beat where the incident occurred. A beat is the smallest police geographic area
District	Indicates the police district where the incident occurred
Location Description	Description of the location where the incident occurred
Ward	The ward (City Council district) where the incident occurred
Year	Year the incident occurred
Case Number	The Chicago Police Department RD Number (Records Division Number), which is unique to the incident
Domestic	Indicates whether the incident was domestic-related as defined by the Illinois Domestic Violence Act
Updated On	Date and time the record was last updated
Month	The month the incident occurred
Day	The day the incident occurred
DayOfWeek	The day of the week the incident occurred
WeekOfYear	The week of year the incident occurred
Hour	The hour of the day the incident occurred

Table 3.6: Features considered in the Los Angeles dataset [2]

Feature	Description
Weapon Used	The type of weapon used in the crime
Weapon Desc	Defines the Weapon Used Code provided.
Vict Sex	Victim Sex, F - Female, M - Male, X - Unknown
Vict Age	Two character numeric.
Mocodes	Modus Operandi: Activities associated with the suspect in commission of the crime.
LON	The longitude of the location where the incident occurred
LAT	The latitude of the location where the incident occurred
Vict Descent	Descent Code: A - Other Asian, B - Black, C - Chinese D- Cambodian, F - Filipino, G - Guamanian ,H - Hispanic/Latin/Mexican I - American Indian/Alaskan Native, J - Japanese, K - Korean, L - Laotian O - Other P - Pacific Islander S - Samoan U - Hawaiian V - Vietnamese W - White X - Unknown Z - Asian Indian
LOCATION	Street address of crime incident rounded to the nearest hundred block to maintain anonymity.
Date Rptd	Date Reported, MM/DD/YYYY
AREA NAME	The 21 Geographic Areas or Patrol Divisions are also given a name designation that references a landmark or the surrounding community that it is responsible for. For example 77th Street Division is located at the intersection of South Broadway and 77th Street, serving neighborhoods in South Los Angeles.
Premis Cd	The type of structure, vehicle, or location where the crime took place.
Premis Desc	Defines the Premise Code provided
Status Desc	Defines the Status Code provided
Status	Status of the case. (IC is the default)
Cross Street	Cross Street of rounded Address
Month	The month the incident occurred
Day	The day the incident occurred
DayOfWeek	The day of the week the incident occurred
WeekOfYear	The week of year the incident occurred
Hour	The hour of the day the incident occurred
TIME OCC	The hour of the day the incident occurred
Rpt Dist No	A four-digit code that represents a sub-area within a Geographic Area. All crime records reference the "RD" that it occurred in for statistical comparisons

Chapter 4

RESULTS AND DISCUSSION

This chapter evaluates the proposed crime prediction models in an attempt to answer the research questions posed in the introductory chapter of the work. This chapter is comprised of 4 sections; Section 4.1 presents the results obtained from the RF, ERT and the proposed hybrid algorithm (RF-Plus). Section 4.2 presents the results of the feature selection method, which is recursive feature selection (RFE). Section 4.3 discusses both SVM and Naive Bayes results and how the incorporation of RFE has enhanced their respective performance. These were followed by Section 4.4, a comparative evaluation of all the nine (9) models used in this work.

4.1 Random Forest and Extremely Randomized Trees Results

As discussed in Chapter 3, the models considered are RF, ERT, the proposed hybrid algorithm (RF-Plus), Naïve Bayes(NB) and the NB enhanced by RFE (RFE-NB). Tree-based models require that certain parameters are tuned for an effective analysis. To obtain a fair comparison, the parameters were kept the same across all classifiers. Rather than randomly selecting parameters, several iterations to obtain the optimal values to use were performed.

This section is subdivided into two subsection: Subsection 4.1.1 which discusses how optimal hyper-parameters were fine-tuned and obtained to be used in the tree-based models, namely; RF, ERT (RF-Plus). Then Section 4.1.2 conducts a comparative evaluation of the three models listed in Section 4.1.1.

Table 4.1: Description of the parameters considered for Random Forest and Extremely Randomized Trees

Parameters	Description
n_trees	Number of trees in a forest
max_dept	Maximum depth of a tree
n_folds	Number of folds in cross validation
min_size	The minimum number of samples in a terminal node
$n\_feature$	The number of features to be considered for each tree
$\sqrt{p}$	where p is the number of features in a training set

Table 4.2: Parameter values set to 5

Parameter Values	Dataset	Evaluation Metrics	RF	ERT	RF-Plus
$n\_tree = 5$	Chicago	Accuracy	75	69	77
		Precision	72	66	73
$max\_depth = 5$		Recall	71	65	74
		f1_Score	71	65.49	72
$n\_folds = 5$		Error Rate	0.10	0.17	0.15
		Execution Time (Seconds)	14.59	3.31	6.40
	Los Angeles	Accuracy	73	68	75
$min\_size = 5$		Precision	71	63	72
		Recall	70	64	71
		f1_Score	71.41	65	71.98
$features = \sqrt{features}$		Error Rate	0.12	0.19	0.16
		Execution Time (Seconds)	15.31	4.01	7.00

4.1.1 Obtaining the Optimal Parameter values for Tree-Based Algorithms

In this study, three (3) tree-predicated algorithms are used which are RF, ERT and RF-Plus. These require well tuned parameters in order to obtain optimal results and table 4.1 summarizes the parameters considered and fine-tuned. For each of these parameters values were adjusted in incremental steps of 5 and considered five different values: 5, 10, 15, 20, and 25. For instance, a parameter value of 5 indicates that n_tree, max_depth,min_size and n_folds were all to 5. At each step, the performance was recorded particularly the training and evaluation times and accuracy. As expected, as the parameter values increased, training and evaluation times increased. However, peak accuracy were recorded when using parameter value of 15, after which the accuracy begins to drop.

Tables 4.2 to 4.6 show recorded results for all parameters for both datasets. The parameters are discussed in Table 4.1. It can be seen that the highest accuracy was recorded on Table 4.4. Then this value (15) was chosen and plotted the results as shown on Figure 4.1.

Table 4.3: Parameter values set to 10

Parameter Values	Dataset	Evaluation Metrics	RF	ERT	RF-Plus
$n\_tree = 10$	Chicago	Accuracy	90	81	89
		Precision	89	78	88
$max\_depth = 10$		Recall	88	77	91
		f1_Score	88.50	79.47	89.47
		Error Rate	0.09	0.13	0.11
$n\_folds = 10$		Execution Time (Seconds)	88.8	14.8	17.50
	Los Angeles	Accuracy	92	83	90
$min\_size = 10$		Precision	89	80	88
		Recall	87	79	87
		f1_Score	87.98	79.50	92.8
		Error Rate	0.08	0.14	0.12
$features = \sqrt{features}$		Execution Time (Seconds)	90.0	15.0	19.71

Table 4.4: Parameter values set to 15

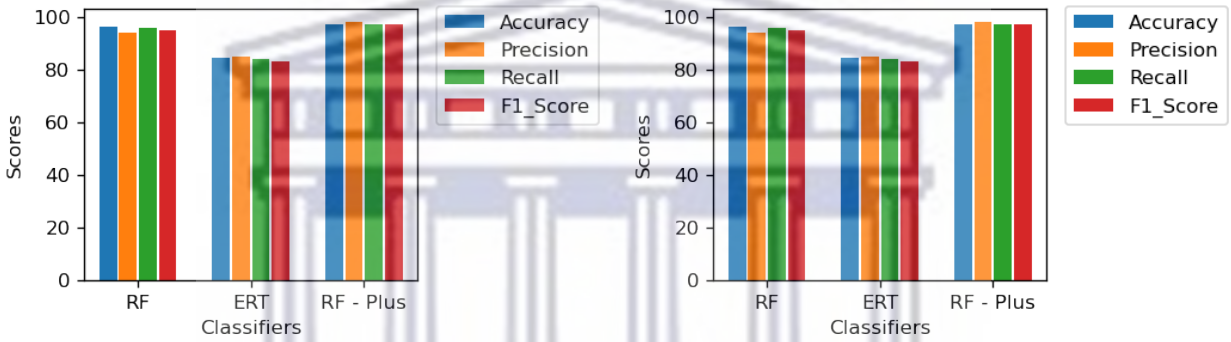
Parameter Values	Dataset	Evaluation Metrics	RF	ERT	RF-Plus
$n\_tree = 15$	Chicago	Accuracy	95.4	85	97
		Precision	94.3	83	96
$max\_depth = 15$		Recall	95.57	84.7	95.87
		f1_Score	94.93	83.84	95.93
		Error Rate	0.02	0.08	0.06
$n\_folds = 15$		Execution Time (Seconds)	210.6	35.1	40.0
	Los Angeles	Accuracy	94.5	84	96
$min\_size = 15$		Precision	93	83	95
		Recall	92	82	93
		f1_Score	92.50	82.50	93.99
		Error Rate	0.03	0.10	0.07
$features = \sqrt{features}$		Execution Time (Seconds)	215.0	36.51	43.0

Table 4.5: Parameter values set to 20

Parameter Values	Dataset	Evaluation Metrics	RF	ERT	RF-Plus
$n\_tree = 20$	Chicago	Accuracy	93.5	82	94
		Precision	92.8	81.7	97.04
$max\_depth = 20$		Recall	90	79	92
		f1_Score	91.38	80.33	
		Error Rate	0.06	0.11	0.05
$n\_folds = 20$		Execution Time (Seconds)	498.6	83.1	90.0
	Los Angeles	Accuracy	92.7	81	95.5
		Precision	91	82	93
$features = \sqrt{features}$		Recall	88	80	91
		f1_Score	89.47	80.99	91.99
		Error Rate	0.07	0.09	0.04
		Execution Time (Seconds)	500.0	80.0	91.0

Table 4.6: Parameter values set to 25

Parameter Values	Dataset	Evaluation Metrics	RF	ERT	RF-Plus
$n\_tree = 25$	Chicago	Accuracy	90	79	92
		Precision	89	77	90
$max\_depth = 25$		Recall	87	75	89
		f1_Score	87.99	75.99	89.50
		Error Rate	0.08	0.12	0.10
$n\_folds = 25$		Execution Time (Seconds)	918.6	153.1	160.53
	Los Angeles	Accuracy	89	82	91
		Precision	87	80	89
$features = \sqrt{features}$		Recall	86	78	88.50
		f1_Score	86.50	78.98	88.75
		Error Rate	0.9	0.13	0.1
		Execution Time (Seconds)	920.00	150.00	063.7



(a) Chicago Dataset (b) Los Angeles Dataset

Figure 4.1: Comparison of RF, ERT and RF-Plus with Parameters set at 15 for both datasets

4.1.2 Comparison of Random Forest and Extremely Randomized Trees against RF-Plus

As described in the models and methodology chapter, a hybrid algorithm (RF-Plus) was constructed from RF and ERT. In this subsection, a comparative evaluation of the proposed hybrid algorithm was conducted and its parent algorithms. Figure 4.1(a) and 4.1(b) show the comparison for both Chicago and Los Angeles datasets respectively.

Figures 4.1(a) and 4.1(b) depicts a graphical representation of the results obtained by the three algorithms in consideration, the RF, ERT and the proposed hybrid algorithm(RF-Plus). The graph is a comparison of four evaluation metrics - accuracy, precision, recall and f1 score. It can be observed that all methods had impressive results. However, the proposed algorithm (RF-Plus) had a slightly higher accuracy than the base RF and significantly higher than ERT.

Table 4.7: Recursive Feature Elimination Results with Three Models

	Chicago Dataset	Los Angeles Dataset
Models	**Accuracy (%)**	**Accuracy (%)**
Linear Regression	38.19	48.38
Extremely Randomized Trees	79.33	95.53
Random Forest	92.83	84.31

Though not shown on the graph, error rate to evaluate the rate that the models incorrectly classify instances was measured. The classifier with the highest mean error rate was the ERT, followed by the RF and then RF-Plus. This result is shown on Table 4.4. To evaluate the complexity of the classifiers the time required to construct and evaluate the classifiers was measured. Again on Table 4.4, it can be seen that the RF is significantly computational expensive, taking close to an average of 6 times longer than the other classifiers, ERT and RF-Plus. ERT was the fastest and this is due to its selection of random cut-point as depicted in Algorithm 3. Our RF-Plus model was significantly faster than the RF, as it borrows steps from the ERT model.

4.2 Recursive Feature Elimination Results

The second phase of this work involves enhancing the NB and SVM algorithm by adding feature selection abilities. This section provide details of the RFE construction process and presents results of infusing it with the NB and SVM. This process was done for both the Chicago and Los Angeles crime datasets.

RFE, as earlier described, begins with all features then recursively eliminates the least influential features using user/researcher selected models. Three models were used in this study for this purpose. These are: (i) (RF); (ii) (ERT); and (iii) Linear Regression (for comparative purposes). It can be observed on Table 4.7 that for the Chicago dataset, among the 3 models considered, the RF model had a substantially higher accuracy (92.83%) compared to the others. Thus, RF was utilized to obtain the optimal number of features from the Chicago dataset, which was 15 (out of the original 21 features). This is as shown on Figure 4.4(a) with the peak at 15. The 15 selected features were: Description, IUCR, FBI Code, Arrest, Day, Longitude, Latitude, Community Area, Block, Beat, District, Ward, WeekOfYear, DayOfWeek, and Year; while the eliminated features were: Case Number, Domestic, Updated On, Location Description, Month, and Hour.

	Feature_Significance	Ranking
IUCR	2.809165e-03	1.0
Description	2.512726e-03	2.0
Location Description	7.726601e-04	3.0
Community Area	3.900114e-04	4.0
Year	3.845280e-04	5.0
FBI Code	2.441376e-04	6.0
Ward	1.913825e-04	7.0
Hour	9.776843e-05	8.0
Beat	9.570508e-05	9.0
Day	4.929147e-05	10.0
Updated On	4.747279e-05	11.0
WeekOfYear	4.198704e-05	12.0
Block	4.117547e-05	13.0
Domestic	3.635535e-05	14.0
Arrest	2.765897e-05	15.0
DayOfWeek	2.242749e-05	16.0
Longitude	1.589027e-05	17.0
Month	9.980777e-06	18.0
Latitude	8.662990e-06	19.0
Case Number	6.593437e-06	20.0
District	5.747992e-07	21.0

(a) Chicago Dataset

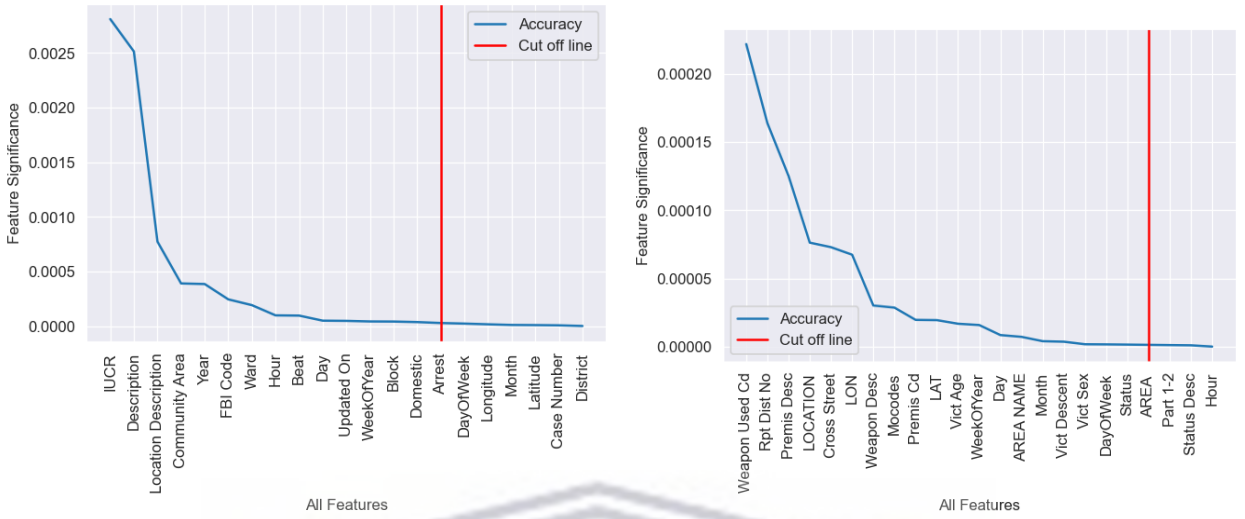
	Feature_Significance	Ranking
Weapon Used Cd	2.220127e-04	1.0
Rpt Dist No	1.639503e-04	2.0
Premis Desc	1.251377e-04	3.0
LOCATION	7.627412e-05	4.0
Cross Street	7.295818e-05	5.0
LON	6.743473e-05	6.0
Weapon Desc	3.016061e-05	7.0
Mocodes	2.858172e-05	8.0
Premis Cd	1.959274e-05	9.0
LAT	1.941254e-05	10.0
Vict Age	1.676251e-05	11.0
WeekOfYear	1.581765e-05	12.0
Day	8.454832e-06	13.0
AREA NAME	7.143385e-06	14.0
Month	3.962796e-06	15.0
Vict Descent	3.597203e-06	16.0
Vict Sex	1.719905e-06	17.0
DayOfWeek	1.623106e-06	18.0
Status	1.470043e-06	19.0
AREA	1.290805e-06	20.0
Part 1-2	1.106825e-06	21.0
Status Desc	9.391981e-07	22.0
Hour	0.000000e+00	23.0

(b) Los Angeles Dataset

Figure 4.2: All Features Ranked by Feature Significance

Presented in Fig.4.2 is a list of all the feature used in our experiments for both the Chicago dataset Fig.4.2(a) and the Los Angeles dataset Fig.4.2(b). The features are ranked using Feature Significance in descending order.

Fig.4.3(a) presents a line graph of all feature versus feature significance. As it has been demonstrated in Fig.4.4(a), that, the optimal number of features to have been selected by RFE for the Chicago dataset is 15, thus, the least important feature to have been eliminated is anything from the 15th feature, Arrest, presented graphically by the red vertical line in Fig.4.3(a). Those features from the Chicago Dataset can be seen to be DayOfWeek, Longitude, Month, Latitude, Case Number, and District. Similarly, for



(a) Chicago Dataset (b) Los Angeles Dataset

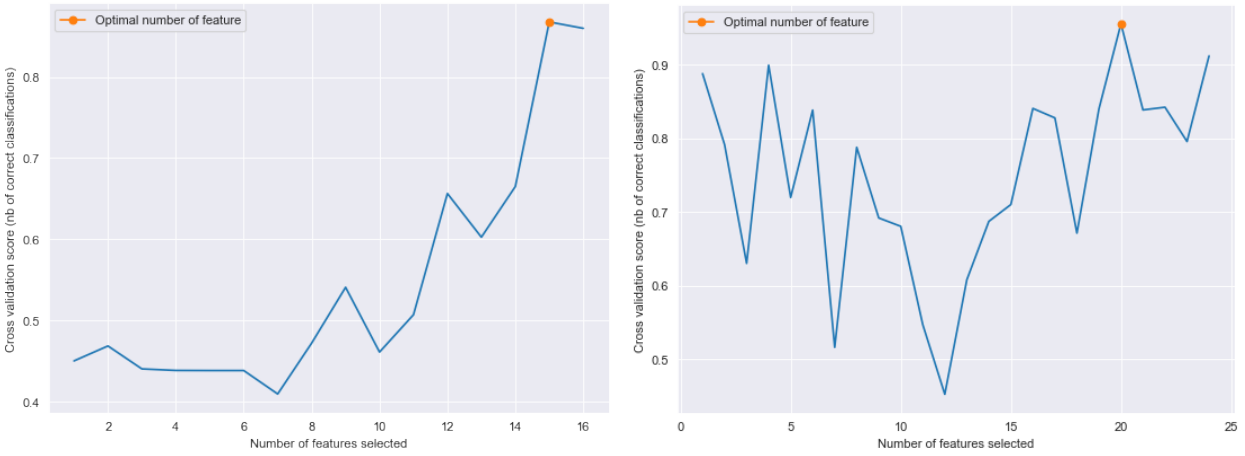
Figure 4.3: Feature Significance Ranking with a Cut-Off line

the Los Angeles dataset, the optimal number of features to have been selected is 20 as graphically represented by the vertical red line Fig.4.3(b), AREA. The subsequent features were then eliminated. The eliminated features are, Part 1-2, Status Desc and Hour.

The process was repeated for the Los Angeles dataset. For this dataset, Table 4.7 shows that the Extremely Randomized Tree (ERT) model gave the highest accuracy at 95.53%. ERT was then used to search and select the optimal number of features. 20 of the 24 features were selected as the optimum as shown on Figure 4.4(b). The 20 selected feature were: Weapon Used Cd, Weapon Desc, Vict Sex, Vict Age, Mocodes, LAT, LON, Vict Descent, LOCATION, AREA NAME, Premis Cd, Premis Desc, Status Desc, Status, Cross Street, Day, Month, Hour, DayOfWeek, and WeekOfYear; while the 4 eliminated features were: TIME OCC, Rpt Dist No, DR_NO and Date Rptd.

4.3 Naïve Bayes and SVM Enhanced with Recursive Feature Elimination

For SVM model, hyper-parameter tuning is important and Grid search was used for this [126]. Different values of C and Gamma using three kernels were considered. The Gamma value dictates the radius of influence for each and every observation, while C is the trade-off between maximization of the decision function's margin and correct classification of observations. For brevity, only the results for values of



(a) Chicago Dataset (b) Los Angeles Dataset

Figure 4.4: Optimal Number of Features for both Chicago and Los Angeles Datasets

0.01, 1 and 10 were shown. Due to the datasets being non-linearly separable, Kernel functions for feature space transformation were used. The kernel functions considered are Linear, Polynomial, and Gaussian RBF kernel, in order to determine which of these will yield a better result [118].

Tables 4.8 and 4.9 provides a brief summary of the results obtained from both the Chicago and Los Angeles datasets with respect to all the parameters considered.

For both datasets a direct proportional relationship trend was observed, with the Gaussian RBF kernel being the superior, followed by Polynomial and Linear Kernel. Observing the different values from 0.01 to 10 for C and Gamma, a gradual performance increase was noted and the best performance was obtained at 10 as seen on Tables 4.8 and 4.9.

SVM results presented on Tables 4.10 and 4.12 were extracted from Tables 4.9 and 4.8 based on the best performing kernel, Gaussian RBF Kernel.

Table 4.10 summarizes the result of the pure NB and SVM using all features from both datasets considered. These models serve as baselines for bench-marking the enhanced versions. By initially running RFE, the most influential (optimal) features were then determined. These are the 15 and 20 depicted by the peak of the graph in Figures 4.4(b) and 4.4(a), for the Chicago and Los Angeles datasets respectively. Then the process was repeated for NB and SVM using the optimal features only. Tables 4.11 and 4.12 show a comparison of the obtained results. From the tables, significant improvements can be seen across all four evaluation metrics in the enhanced versions, RFE-NB and RFE-SVM, compared

Table 4.8: Comparison of Kernel Functions on the Chicago Dataset

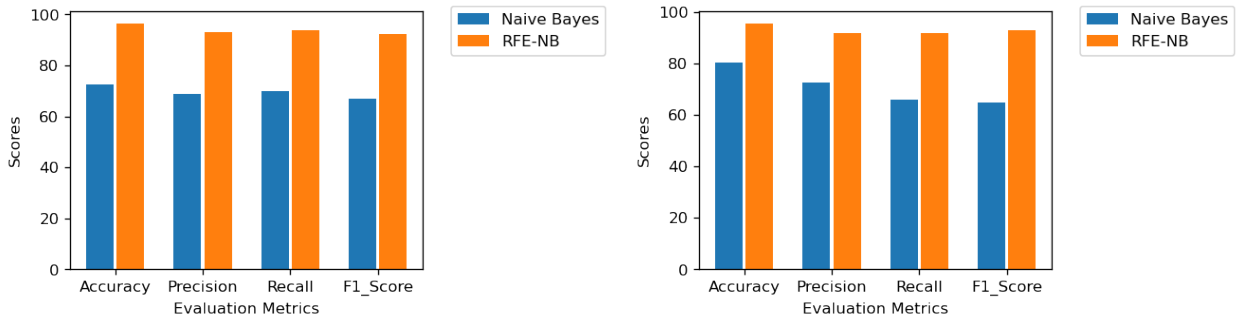
Linear Kernel			
Evaluation Metric (%)	Gamma=0.01, C=0.01	Gamma=1, C=1	Gamma=10, C=10
Accuracy	35.80	40.34	55.78
Precision	28.82	47.9	53.79
Recall	35.80	48.01	55.57
F1-Score	27.88	46.8	54.51
Polynomial Kernel			
Accuracy	49.34	56.1	65.73
Precision	47.57	54.54	62.56
Recall	46.86	55.36	63.2
F1-Score	45.61	54.54	64.78
Gaussian RBF Kernel			
Accuracy	59.27	65.9	74.73
Precision	57.45	64.9	71.56
Recall	55.60	66.6	74.20
F1-Score	56.71	63.57	73.78

Table 4.9: Comparison of Kernel Functions on the Los Angeles Dataset

Linear Kernel			
Evaluation Metric (%)	Gamma=0.01, C=0.01	Gamma=1, C=1	Gamma=10, C=10
Accuracy	36.61	42.19	56.91
Precision	29.19	48.17	55.78
Recall	36.89	41.19	57.20
F1-Score	28.11	48.1	56.15
Polynomial Kernel			
Accuracy	41.52	56.17	6.71
Precision	48.59	56.71	63.19
Recall	44.17	57.61	65.18
F1-Score	45.71	54.81	64.51
Gaussian RBF kernel			
Accuracy	59.34	66.1	75.73
Precision	56.51	65.16	73.16
Recall	57.81	64.16	74.58
F1-Score	56.51	65.34	75.51

Table 4.10: Pure Naïve Bayes and SVM results for both datasets

	Chicago Dataset		Los Angeles Dataset	
Evaluation Metrics	Pure NB	Pure SVM	Pure NB	Pure SVM
Accuracy(%)	72.55	74.73	80.45	75.73
Precision	69.00	71.56	74.00	73.16
Recall	70.00	74.20	66.00	74.58
F1-Score	67.00	73.78	65.00	75.51



(a) Chicago Dataset

(b) Los Angeles Dataset

Figure 4.5: Pure vs RFE-Enhanced Naïve Bayes

to their pure variants. Figures 4.5 and 4.6 present these graphically.

Table 4.11: Comparison of RFE-NB with Pure Naive Bayes

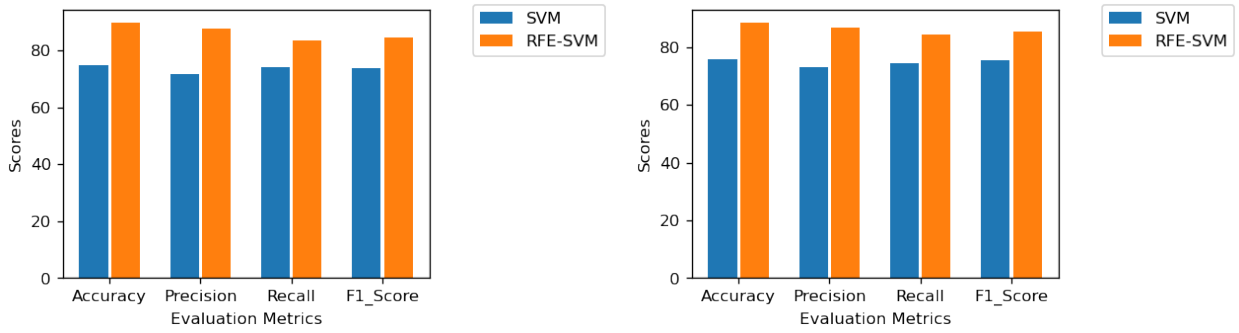
	Chicago Dataset		Los Angeles Dataset	
Evaluation Metrics (%)	Pure NB	RFE-NB	Pure NB	RFE-NB
Accuracy	72.55	96.60	80.45	95.78
Precision	69.00	93.00	74.00	92.07
Recall	70.00	93.70	66.00	92.00
F1-Score	67.00	92.33	65.00	93.00

Table 4.12: Comparison of RFE-SVM with Pure SVM

	Chicago Dataset		Los Angeles Dataset	
Evaluation Metrics(%)	Pure SVM	RFE-SVM	Pure SVM	RFE-SVM
Accuracy	74.73	89.91	75.73	88.70
Precision	71.56	87.57	73.16	86.96
Recall	74.20	83.41	74.58	84.37
F1-Score	73.78	84.59	75.51	85.33

4.4 Comparative Evaluation of all Models Considered

By utilizing the feature selection capabilities of RFE, the performance of both NB and SVM was significantly improved as shown on Tables 4.11 and 4.12. For completeness, a comparative evaluation of the enhanced version of NB (RFE-NB) and SVM (RFE-SVM) with the other tree-based algorithms is presented, the tree-based algorithms are the (RF) and (ERT) and the (RF-Plus). This comparison is justified as all models now have feature selection abilities (Tree-based algorithms have intrinsic feature selection capabilities).



(a) Chicago Dataset (b) Los Angeles Dataset

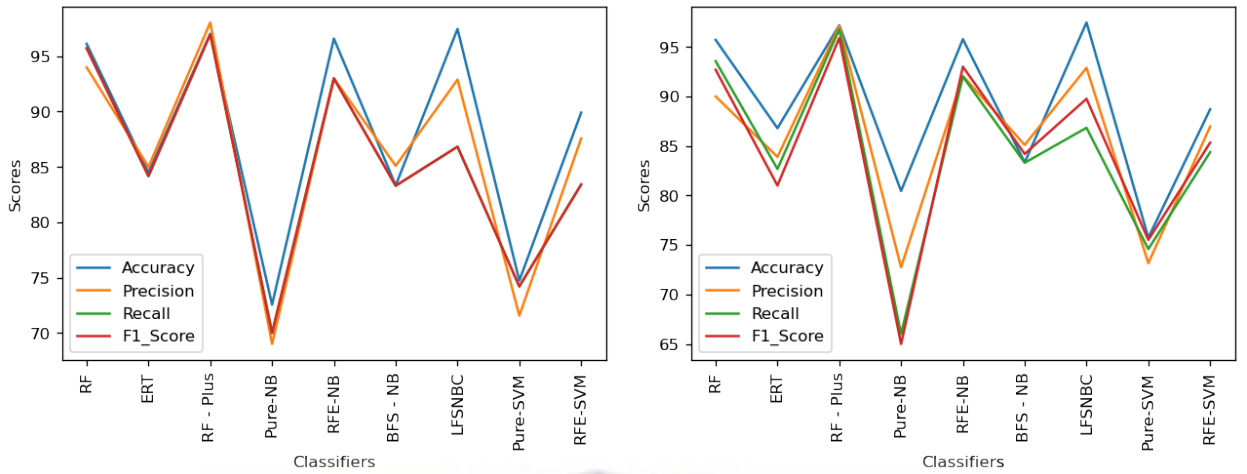
Figure 4.6: Pure vs RFE-Enhanced SVM

Table 4.13: Comparison of all nine(9) Models Considered

Models	Accuracy	Precision	Recall	F1-Score	Feature Selection
RF	96.13	94.30	95.57	94.93	Yes
ERT	85.00	83.00	84.70	83.84	Yes
RF-Plus	97.04	96.00	95.87	95.97	Yes
Pure-NB	72.55	69.00	70.00	67.00	No
RFE-NB	96.60	93.00	93.70	92.33	Yes
Pure SVM	74.73	71.56	74.20	73.78	No
RFE-SVM	89.91	87.57	83.41	84.59	Yes
BFS-NB [52]	83.33	96.90	85.10	84.19	Yes
LFSNBC [127]	97.47	92.89	86.83	89.76	Yes

Figure 4.7(a) shows the obtained results when using the Chicago dataset. RF-Plus, RFE-NB, and RFE-SVM performed well across board, outperforming ERT in all metrics. RF-Plus and RFE-NB gave the highest accuracies at 97.04% and 96.60%, followed by RF (96.13%). However, for F1-score, RF-Plus performed the best at 95.97%, followed by RF (94.93%) then RFE-NB (92.33%). Similar trends were observed with the Los Angeles dataset, with both RF-Plus and RFE-NB leading the pack.

Furthermore, comparative evaluation with two state-of-the-art models proposed in literature was conducted. These models have similar use case (crime prediction) and similar datasets. They are the BFS-NB (Naive Bayes enhanced by Backward Feature Selection) proposed in [52] and LASSO Feature Selection Naive Bayes Classifier (LFSNBC) in [127]. Though comparable to ours (in that the authors sought to improve NB by adding feature selection), these models applied different approaches. In [127], the authors employed Least Absolute Shrinkage Selection Operator (LASSO) as a feature selection



(a) Chicago Dataset Results (b) Los Angeles dataset results

Figure 4.7: Comparative evaluation of all models considered and with other studies

strategy for NB; while in [52] backward feature selection was used.

The results of our study, as well as those from [52] and [127], are also shown in Figure 4.7 and on Table 4.13. The results show that RFE is a superior feature selection technique than both LASSO and Backward feature selection.

Chapter 5

SUMMARY AND CONCLUSIONS

5.1 Summary

Crime continues to increase at an exponential rate and, by implication, there is a continuous increase in crime records that are archived by the police as well as efforts to combat crime. Intelligent analysis of crime records can help predict potential crime and reveal trends, thus serving as a preventative strategy. For example, the US has embraced predictive policing, reporting a 47% gun incidents reduction [128], and the UK Manchester Police Department reported 28% total crime reduction within the first 5 weeks of deployment [129]. This proves that crime prediction models can greatly assist in crime reduction. However, the effectiveness of the analysis is greatly dependent on the choice of model and the parameters or features selected for the analysis.

The aim of this work, therefore, was to explore the efficacy of feature selection in supervised learning models for crime prediction. This research utilized publicly available data-sets from the cities of Chicago and Los Angeles for crime prediction. Nine (9) machine learning models were explored and compared. Out of these nine (9) models, four (4) were state of the art models, three (3) were constructed (hybridized) by means of enhancing the state of the art models, and two (2) were from existing literature serving as a benchmark. The three (3) hybrid models proposed in our study proved to be superior than their parent algorithms, based on the feature selection capabilities introduced and evaluations conducted.

5.2 Concluding Remarks

Among various models and techniques utilized to realize the end goal of crime prediction, decision trees have been prominent and often reportedly superior to other models it has been compared with. In this study, the superiority of the decision trees ensemble was exploited to construct a hybrid algorithm called RF-Plus, which incorporates the best properties of Random Forest (RF) and Extremely Randomized Trees (ERT) methods that ensembles the decision trees, in order to address the variance and bias limitation of the decision tree. The properties that have been incorporated in the hybrid algorithm are, bootstrap aggregation, random selection of a subset of features and random cut-point choice when splitting data in a decision node. Naïve Bayes (NB) and Support Vector Machine (SVM) have also been well used in literature due to their relevance. The NB model has, however, not been well explored in crime prediction problems because of its limited feature selection capability. It was then imperative to address this by enhancing NB using Recursive Feature Elimination (RFE). RFE enabled the selection of an optimal number of influential features, which was then passed onto NB. The proposed models were then trained, which were derived from both NB and SVM, and enhanced with RFE, obtaining RFE-NB and RFE-SVM, respectively. Also, there is RF-Plus, which was created by incorporating certain attributes from RF and ERT. Three hybrid algorithms based on four parent algorithms are presented, making a total of seven algorithms, alongside two from literature, BFS-NB and LFSNBC, were trained and evaluated on publicly available datasets on crimes from the cities of Chicago and Los Angeles. The enhancement on the pure NB, improved its accuracy from 72.5% to 96.6% and 80.45% to 95.78% for Chicago and Los Angeles datasets, respectively. The enhancement improved the accuracy of pure SVM from 74.73% to 89.91% and 75.73% to 88.70% for the Chicago and Los Angeles datasets, respectively. Thus showing the importance of incorporating feature selection when constructing such algorithms, to mitigate the cost of including irrelevant features. When all 9 models were compared, both RF-Plus and RFE-NB gave the best results in terms of accuracy and were at par with RF in terms of F1-score, for both datasets. With equally good performance observed on two distinct datasets, it was then shown that the proposed models are robust.

Our investigations have shown that when constructing crime prediction models and knowledge support systems, it is of utmost importance to consider relevant feature selection models. Moreover, manual feature selection techniques might not always be feasible, especially in large dataset, hence, automated feature selection methods such as RFE should be explored. Finally, though in this research, the proposed

models were compared against RF, ERT and NB, further incorporation and/or benchmarking with deep-learning algorithms could be an avenue for future research, as these have shown promise in recent studies.

Bibliography

[1] "Chicago Crimes Data Portal: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present-Dashboard/5cd6-ry5g," Jan 2001.

[2] "Los Angeles Crimes Data Portal: https://data.lacity.org/Public-Safety/Crime-Data-from-2010-to-2019/63jg-8b9z," Jan 2001.

[3] M. G. Kushner, D. S. Riggs, E. B. Foa, and S. M. Miller, "Perceived controllability and the development of posttraumatic stress disorder (PTSD) in crime victims," *Behaviour research and therapy*, vol. 31, no. 1, pp. 105–110, 1993.

[4] R.-L. Francke, "SAPS short of 90 000 officers, resource allocation behind by 20 years," Jan 2023.

[5] J. van Dijk, P. Nieuwbeerta, and J. J. Larsen, "Global crime patterns: An analysis of survey data from 166 countries around the world, 2006–2019," *Journal of Quantitative Criminology*, pp. 1–36, 2021.

[6] G. Farrell and K. Pease, *Repeat victimization*, vol. 12. Criminal Justice Press, 2001.

[7] O. E. Isafiade and A. B. Bagula, "Series mining for public safety advancement in emerging smart cities," *Future Generation Computer Systems*, vol. 108, pp. 777–802, 2020.

[8] L. Fraser, "Violent crime in south africa is getting worse – here are all the latest stats," Feb 2023.

[9] O. Isafiade, B. Ndingindwayo, and A. Bagula, "Predictive policing using deep learning: A community policing practical case study," in *International Conference on e-Infrastructure and e-Services for Developing Countries*, pp. 269–286, Springer, 2020.

[10] S. Sugania, D. J. D. Raj, S. J. Ratchagan, and C. D. Pandi, "Online crime management system," *International Journal of Scientific Research Engineering Trends*, 2021.

[11] I. El Naqa and M. J. Murphy, "What is machine learning?," in *machine learning in radiation oncology*, pp. 3–11, Springer, 2015.

[12] *Ubiquitous intelligence for smart cities: a public safety approach*, University of Cape Town, 2017.

[13] S. Kim, P. Joshi, P. S. Kalsi, and P. Taheri, "Crime analysis through machine learning," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 415–420, IEEE, 2018.

[14] Y.-L. Lin, M.-F. Yen, and L.-C. Yu, "Grid-based crime prediction using geographical features," *ISPRS International Journal of Geo-Information*, vol. 7, no. 8, p. 298, 2018.

[15] L. G. Alves, H. V. Ribeiro, and F. A. Rodrigues, "Crime prediction through urban metrics and statistical learning," *Physica A: Statistical Mechanics and its Applications*, vol. 505, pp. 435–443, 2018.

[16] A. Bogomolov, B. Lepri, J. Staiano, N. Oliver, F. Pianesi, and A. Pentland, "Once upon a crime: towards crime prediction from demographics and mobile data," in *Proceedings of the 16th international conference on multimodal interaction*, pp. 427–434, 2014.

[17] S. A. Chun, V. Avinash Paturu, S. Yuan, R. Pathak, V. Atluri, and N. R. Adam, "Crime prediction model using deep neural networks," in *Proceedings of the 20th Annual International Conference on Digital Government Research*, pp. 512–514, 2019.

[18] O. E. Isafiade and A. B. Bagula, "Data mining trends and applications in criminal science and investigations," *Data Mining and Database Management Series*, vol. 108, pp. 1–386, 2016.

[19] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[20] L. Rokach and O. Maimon, "Decision trees," in *Data mining and knowledge discovery handbook*, pp. 165–192, Springer, 2005.

[21] G. I. Webb, E. Keogh, and R. Miikkulainen, "Naïve bayes.," *Encyclopedia of machine learning*, vol. 15, pp. 713–714, 2010.

[22] A. Sangani, C. Sampat, and V. Pinjarkar, "Crime prediction and analysis," in *2nd International Conference on Advances in Science & Technology (ICAST)*, 2019.

[23] A. Gahalot, S. Dhiman, L. Chouhan, *et al.*, "Crime prediction and analysis," in *2nd International Conference on Data, Engineering and Applications (IDEA)*, pp. 1–6, IEEE, 2020.

[24] N. Jabeen and P. Agarwal, "Data mining in crime analysis," in *Proceedings of Second International Conference on Smart Energy and Communication*, pp. 97–103, Springer, 2021.

[25] N. Shah, N. Bhagat, and M. Shah, "Crime forecasting: a machine learning and computer vision approach to crime prediction and prevention," *Visual Computing for Industry, Biomedicine, and Art*, vol. 4, no. 1, pp. 1–14, 2021.

[26] N. F. M. Zamria, N. M. Tahira, M. S. A. M. Ali1c, N. D. K. Ashard, and A. Abd Al-misrebe, "Mini-review of street crime prediction and classification methods," *Jurnal Kejuruteraan*, vol. 33, no. 3, pp. 391–401, 2021.

[27] I. Kononenko and M. Kukar, *Machine learning and data mining*. Horwood Publishing, 2007.

[28] P. Kormushev, S. Calinon, and D. G. Caldwell, "Reinforcement learning in robotics: Applications and real-world challenges," *Robotics*, vol. 2, no. 3, pp. 122–148, 2013.

[29] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[30] M. Riedmiller and T. Gabel, "On experiences in a complex and competitive gaming domain: Reinforcement learning meets robocup," in *2007 IEEE Symposium on Computational Intelligence and Games*, pp. 17–23, IEEE, 2007.

[31] Y. Björnsson, V. Hafsteinsson, Á. Jóhannsson, and E. Jónsson, "Efficient use of reinforcement learning in a computer game," in *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pp. 379–383, 2004.

[32] T. W. Rondeau and C. W. Bostian, "Cognitive techniques: Physical and link layers," in *Cognitive radio technology*, pp. 219–268, Elsevier, 2006.

[33] M. S. Gerber, "Predicting crime using twitter and kernel density estimation," *Decision Support Systems*, vol. 61, pp. 115–125, 2014.

[34] B. Wang, D. Zhang, D. Zhang, P. J. Brantingham, and A. L. Bertozzi, "Deep learning for real time crime forecasting," *arXiv preprint arXiv:1707.03340*, 2017.

[35] M. Al Boni and M. S. Gerber, "Automatic optimization of localized kernel density estimation for hotspot policing," in *2016 15th IEEE international conference on Machine Learning and Applications (ICMLA)*, pp. 32–38, IEEE, 2016.

[36] Y. Hu, F. Wang, C. Guin, and H. Zhu, "A spatio-temporal kernel density estimation framework for predictive crime hotspot mapping and evaluation," *Applied geography*, vol. 99, pp. 89–97, 2018.

[37] C. D. R. Rodriguez, D. M. Gomez, and M. A. M. Rey, "Forecasting time series from clustering by a memetic differential fuzzy approach: An application to crime prediction," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, IEEE, 2017.

[38] P. Chen, H. Yuan, and X. Shu, "Forecasting crime using the arima model," in *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 5, pp. 627–630, IEEE, 2008.

[39] K. Islam and A. Raza, "Forecasting crime using arima model," *arXiv preprint arXiv:2003.08006*, 2020.

[40] R. Yadav and S. K. Sheoran, "Modified arima model for improving certainty in spatio-temporal crime event prediction," in *2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*, pp. 1–4, IEEE, 2018.

[41] S. Vijayarani, E. Suganya, and C. Navya, "Crime analysis and prediction using enhanced arima model," *Journal homepage: www. ijrpr. com ISSN*, vol. 2582, p. 7421, 2021.

[42] Y. S. Triana and A. Retnowardhani, "Enhance interval width of crime forecasting with arima model-fuzzy alpha cut," *Telkomnika*, vol. 17, no. 3, pp. 1193–1201, 2019.

[43] G. Hajela, M. Chawla, and A. Rasool, "A clustering based hotspot identification approach for crime prediction," *Procedia Computer Science*, vol. 167, pp. 1462–1470, 2020.

[44] S. Sivaranjani, S. Sivakumari, and M. Aasha, "Crime prediction and forecasting in tamilnadu using clustering approaches," in *2016 International Conference on Emerging Technological Trends (ICETT)*, pp. 1–6, IEEE, 2016.

[45] J. Kiran and K. Kaishveen, "Prediction analysis of crime in india using a hybrid clustering approach," in *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and*

Cloud)(I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2018 2nd International Conference on, pp. 520–523, IEEE, 2018.

[46] I. Hussain, "Clustering in machine learning," *International Journal of Computer and Information Technology*, 2011.

[47] N. Ivan, E. Ahishakiye, E. O. Omulo, and D. Taremwa, "Crime prediction using decision tree (j48) classification algorithm.," *International Journal of Computer and Information Technology*, 2017.

[48] N. A. S. Zaidi, A. Mustapha, S. A. Mostafa, and M. N. Razali, "A classification approach for crime prediction," in *International Conference on Applied Computing to Support Industry: Innovation and Technology*, pp. 68–78, Springer, 2019.

[49] R. Iqbal, M. A. A. Murad, A. Mustapha, P. H. S. Panahy, and N. Khanahmadliravi, "An experimental study of classification algorithms for crime prediction," *Indian Journal of Science and Technology*, vol. 6, no. 3, pp. 4219–4225, 2013.

[50] N. Ivan, E. Ahishakiye, E. O. Omulo, and R. Wario, "A performance analysis of business intelligence techniques on crime prediction," *International Journal of Computer and Information Technology*, 2017.

[51] N. Ivan, E. Ahishakiye, E. O. Omulo, and D. Taremwa, "Crime prediction using decision tree (j48) classification algorithm.," *International Journal of Computer and Information Technology*, 2017.

[52] B. S. Aldossari, F. M. Alqahtani, N. S. Alshahrani, M. M. Alhammam, R. M. Alzamanan, and N. Aslam, "A comparative study of decision tree and naive bayes machine learning model for crime category prediction in chicago," in *Proceedings of 2020 the 6th International Conference on Computing and Data Engineering*, pp. 34–38, 2020.

[53] O. Llaha, "Crime analysis and prediction using machine learning," in *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pp. 496–501, IEEE, 2020.

[54] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[55] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[56] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.

[57] A.-L. Boulesteix, S. Janitza, J. Kruppa, and I. R. König, "Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 6, pp. 493–507, 2012.

[58] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1, pp. 278–282, IEEE, 1995.

[59] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural computation*, vol. 9, no. 7, pp. 1545–1588, 1997.

[60] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[61] C. Chu, A.-L. Hsu, K.-H. Chou, P. Bandettini, C. Lin, A. D. N. Initiative, *et al.*, "Does feature selection improve classification accuracy? impact of sample size and feature selection on classification using anatomical magnetic resonance images," *Neuroimage*, vol. 60, no. 1, pp. 59–70, 2012.

[62] E. V. A. Sylvester, P. Bentzen, I. R. Bradbury, M. Clément, J. Pearce, J. Horne, and R. G. Beiko, "Applications of random forest feature selection for fine-scale genetic population assignment," *Evolutionary Applications*, vol. 11, pp. 153–165, feb 2018.

[63] "ML — Extra Tree Classifier for Feature Selection - GeeksforGeeks," jul 2020.

[64] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE signal processing magazine*, vol. 24, no. 4, pp. 118–121, 2007.

[65] D. E. Hilt and D. W. Seegrist, *Ridge, a computer program for calculating ridge regression estimates*, vol. 236. Department of Agriculture, Forest Service, Northeastern Forest Experiment . . . , 1977.

[66] G. R. Nitta, B. Y. Rao, T. Sravani, N. Ramakrishiah, and M. Balaanand, "Lasso-based feature selection and naïve bayes classifier for crime prediction and its type," *Service Oriented Computing and Applications*, vol. 13, no. 3, pp. 187–197, 2019.

[67] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data," *Computational Statistics & Data Analysis*, vol. 143, p. 106839, 2020.

[68] F. Mohd, N. M. M. Noor, *et al.*, "A comparative study to evaluate filtering methods for crime data feature selection," *Procedia computer science*, vol. 116, pp. 113–120, 2017.

[69] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

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

[71] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on neural networks*, vol. 5, no. 4, pp. 537–550, 1994.

[72] C. Lazar, J. Taminau, S. Meganck, D. Steenhoff, A. Coletta, C. Molter, V. de Schaetzen, R. Duque, H. Bersini, and A. Nowe, "A survey on filter techniques for feature selection in gene expression microarray analysis," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 9, no. 4, pp. 1106–1119, 2012.

[73] G. Forman *et al.*, "An extensive empirical study of feature selection metrics for text classification.," *J. Mach. Learn. Res.*, vol. 3, no. Mar, pp. 1289–1305, 2003.

[74] N. Kwak and C.-H. Choi, "Input feature selection for classification problems," *IEEE transactions on neural networks*, vol. 13, no. 1, pp. 143–159, 2002.

[75] P. Comon, "Independent component analysis, a new concept?," *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[76] K. Torkkola, "Feature extraction by non-parametric mutual information maximization," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1415–1438, 2003.

[77] F. Fleuret, "Fast binary feature selection with conditional mutual information.," *Journal of Machine learning research*, vol. 5, no. 9, 2004.

[78] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.

[79] F. Mohd, N. M. M. Noor, *et al.*, "A comparative study to evaluate filtering methods for crime data feature selection," *Procedia computer science*, vol. 116, pp. 113–120, 2017.

[80] Y. Zhu *et al.*, "Comparison of model performance for basic and advanced modeling approaches to crime prediction," *Intelligent Information Management*, vol. 10, no. 06, p. 123, 2018.

[81] C. Kadar, J. Iria, and I. P. Cvijikj, "Exploring foursquare-derived features for crime prediction in new york city," in *The 5th international workshop on urban computing (UrbComp 2016)*, ACM, 2016.

[82] P.-N. Tan, M. Steinbach, and V. Kumar, "Data mining introduction," *People's Posts and Telecommunications Publishing House, Beijing*, 2006.

[83] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Routledge, 2017.

[84] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, *et al.*, "Top 10 algorithms in data mining," *Knowledge and information systems*, vol. 14, no. 1, pp. 1–37, 2008.

[85] C. Kingsford and S. L. Salzberg, "What are decision trees?," *Nature biotechnology*, vol. 26, no. 9, pp. 1011–1013, 2008.

[86] T. Thomas, A. P Vijayaraghavan, and S. Emmanuel, "Applications of decision trees," in *Machine learning approaches in cyber security analytics*, pp. 157–184, Springer, 2020.

[87] E. F. Cook and L. Goldman, "Empiric comparison of multivariate analytic techniques: advantages and disadvantages of recursive partitioning analysis," *Journal of chronic diseases*, vol. 37, no. 9-10, pp. 721–731, 1984.

[88] T. Hancock and C. Smyth, *Tree-Based Clustering and Extensions*. Elsevier, 2009.

[89] L. Rokach and O. Maimon, "Decision trees," in *Data mining and knowledge discovery handbook*, pp. 165–192, Springer, 2005.

[90] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[91] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[92] K. S. Hartati, S. A. Yogyakarta, J. Ringroad, U. Condong, C. Sleman, and Y. Indonesia, "Implementation of c4.5 algorithm to evaluate the cancellation possibility of new student applicants at stmik amikom yogyakarta," in *Proceedings of the International Conference on Electrical Engineering and Informatics Institute Technology*, pp. 17–19, 2009.

[93] M. L. McHugh, "The chi-square test of independence," *Biochemia medica*, vol. 23, no. 2, pp. 143–149, 2013.

[94] C. Bonferroni, "Teoria statistica delle classi e calcolo delle probabilita," *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, vol. 8, pp. 3–62, 1936.

[95] G. V. Kass, "An exploratory technique for investigating large quantities of categorical data," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 29, no. 2, pp. 119–127, 1980.

[96] Y. Mansour, "Pessimistic decision tree pruning based on tree size," in *Machine Learning International Workshop Then Conference*, pp. 195–201, Citeseer, 1997.

[97] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Wadsworth, 1984.

[98] J. R. Quinlan, "Simplifying decision trees," *International journal of man-machine studies*, vol. 27, no. 3, pp. 221–234, 1987.

[99] J. R. Quinlan and R. L. Rivest, "Inferring decision trees using the minimum description lenght principle," *Information and computation*, vol. 80, no. 3, pp. 227–248, 1989.

[100] C. S. Wallace and J. D. Patrick, "Coding decision trees," *Machine Learning*, vol. 11, no. 1, pp. 7–22, 1993.

[101] M. Mehta, J. Rissanen, R. Agrawal, *et al.*, "Mdl-based decision tree pruning.," in *KDD*, vol. 21, pp. 216–221, 1995.

[102] A. L. Oliveira and A. Sangiovanni-Vincentelli, "Using the minimum description length principle to infer reduced ordered decision graphs," *Machine Learning*, vol. 25, no. 1, pp. 23–50, 1996.

[103] I. Kononenko, "The minimum description length based decision tree pruning," in *Pacific Rim International Conference on Artificial Intelligence*, pp. 228–237, Springer, 1998.

[104] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999.

[105] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and systems magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[106] L. Rokach, "Ensemble-based classifiers," *Artificial intelligence review*, vol. 33, no. 1, pp. 1–39, 2010.

[107] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[108] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[109] S. I. May, O. E. Isafiade, and O. O. Ajayi, "Hybridizing extremely randomized trees with bootstrap aggregation for crime prediction," in *2021 4th International Conference on Artificial Intelligence and Pattern Recognition*, pp. 536–541, 2021.

[110] D. M. Rice, *Calculus of thought: neuromorphic logistic regression in cognitive machines*. Academic Press, 2013.

[111] A. P. Dawid, "Conditional independence in statistical theory," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 41, no. 1, pp. 1–15, 1979.

[112] D. M. Rice, "Causal Reasoning," in *Calculus of Thought*, ch. 4, pp. 95–123, Elsevier, jan 2014.

[113] E. Artun, S. D. Mohaghegh, J. Toro, T. Wilson, and A. Sanchez, "Reservoir characterization using intelligent seismic inversion," in *SPE Eastern Regional Meeting*, OnePetro, 2005.

[114] R. Gholami, A. Moradzadeh, S. Maleki, S. Amiri, and J. Hanachi, "Applications of artificial intelligence methods in prediction of permeability in hydrocarbon reservoirs," *J. Petroleum Science and Engineering*, vol. 122, pp. 643–656, 2014.

[115] M. Abbaszadeh, A. Hezarkhani, and S. Soltani-Mohammadi, "Proposing drilling locations based on the 3d modeling results of fluid inclusion data using the support vector regression method," *J. Geochemical Exploration*, vol. 165, pp. 23–34, 2016.

[116] V. N. Vapnik, *An overview of statistical learning theory*, vol. 10. IEEE, 1999.

[117] M. Martínez-Ramón and C. Christodoulou, "Support vector machines for antenna array processing and electromagnetics," *Synthesis Lectures on Computational Electromagnetics*, vol. 1, no. 1, pp. 1–120, 2005.

[118] L. Wang, *Support vector machines: theory and applications*, vol. 177. Springer Science & Business Media, 2005.

[119] I. Steinwart and A. Christmann, *Support vector machines*. Springer Science & Business Media, 2008.

[120] S. Abe, *Support vector machines for pattern classification*, vol. 2. Springer, 2005.

[121] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods*. USA: Cambridge University Press, 1999.

[122] J. Mercer, "Functions ofpositive and negativetypeand theircommection with the theory ofintegral equations," *Philos. Trinsdictions Rogyal Soc*, vol. 209, pp. 4–415, 1909.

[123] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1, pp. 389–422, 2002.

[124] P. M. Granitto, C. Furlanello, F. Biasioli, and F. Gasperi, "Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products," *Chemometrics and intelligent laboratory systems*, vol. 83, no. 2, pp. 83–90, 2006.

[125] G. S. Handelman, H. K. Kok, R. V. Chandra, A. H. Razavi, S. Huang, M. Brooks, M. J. Lee, and H. Asadi, "Peering into the black box of artificial intelligence: evaluation metrics of machine learning methods," *American Journal of Roentgenology*, vol. 212, no. 1, pp. 38–43, 2019.

[126] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *J. machine learning research*, vol. 13, no. 2, 2012.

[127] G. R. Nitta, B. Y. Rao, T. Sravani, N. Ramakrishiah, and M. Balaanand, "Lasso-based feature selection and naïve bayes classifier for crime prediction and its type," *Service Oriented Computing and Applications*, vol. 13, no. 3, pp. 187–197, 2019.

[128] A. Meijer and M. Wessels, "Predictive policing: Review of benefits and drawbacks," *International Journal of Public Administration*, vol. 42, no. 12, pp. 1031–1039, 2019.

[129] N. Willard and T. Gagne, "Predictive policing success for Manchester pd," Apr 2021.