

**AN EVALUATION OF HYBRID MACHINE LEARNING CLASSIFIER MODELS FOR
IDENTIFICATION OF TERRORIST GROUPS IN THE AFTERMATH OF AN
ATTACK**

BY

PETER OPIYO OKETCH

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF A MASTER OF SCIENCE IN INFORMATION
TECHNOLOGY**

SCHOOL OF COMPUTING AND INFORMATICS

MASENO UNIVERSITY

©2020

DECLARATION

I declare that this thesis is my original work and that it has not been presented in any other university, defense panel or institute of higher learning for academic award or any other award.

Peter Opiyo Oketch:

MSC/CI/00110/2014

Signature: _____

Date: _____

APPROVAL

The undersigned certify that they have read and hereby recommend for acceptance of Maseno University a thesis entitled **“An Evaluation of Hybrid Machine Learning Classifier Models for identification of Terrorist Groups in the aftermath of an Attack”**

Dr. Titus Muhambe Mukisa (PhD)

School of Computing and Informatics, Maseno University

Signature: _____

Date: _____

Dr. Ratemo Makiya Cyprian (PhD)

Directorate of e-Learning, Kisii University

Signature: _____

Date: _____

ACKNOWLEDGEMENT

To the Almighty for this great gift of life to accomplish this far I have come.

To my loved ones; family and friends, for their great support and encouragement throughout my academic years.

To my supervisors Dr. Muhambe & Dr. Ratemo, for their support, guidance, time, and critique during my research process and the panelists for their positive criticisms thus have led to success of this research.

I wish to thank the National Consortium for the study of terrorism and response to terrorism (START) at the University of Maryland for granting access to their database for the purpose of this research.

To all my classmates who shared ideas and helped during this study, I say thank you.

DEDICATION

This work is dedicated to my Parents.

ABSTRACT

Terrorist attacks have globally led to loss of life and property, fear, and general insecurity. Terrorist acts are planned and perpetrated by collections of loosely organized people operating in shadowy networks that are difficult to identify. Machine learning classifier algorithms have been used in accurate identification of terrorist groups and weapon types in India, Egypt, Pakistan, and United Kingdom. However, the urgency of responding to a terrorist attack and the subsequent nature of analysis required to identify the terrorist group involved in an attack demands that the performance of the classifiers yield highly accurate outcomes. The concept of combining classifier algorithms into hybrid is proposed as a new way of improving the accuracy. To date there has not been sufficient research that attempts to find combinations of Naïve Bayes, K-Nearest Neighbor, Decision Trees, Support Vector Machines and Multi-Layer Perceptron as base classifier algorithm models and resample sample size percent for optimum accuracy in the identification of terrorist groups in the aftermath of an attack. The aim of the study is to build and evaluate hybrid classifier algorithm models for identification of terrorist groups. Specifically, it builds and evaluates base classifier algorithm models, builds, and evaluates hybrid classifier algorithm models by combining and evaluating the base classifier algorithm models, and compares the performance of the classifier algorithm models. The study adopts a randomized block experimental research design using Waikato Environment for Knowledge Analysis (WEKA) tool for building and evaluating the classifier algorithm models, and 1999-2017 sub-Sahara terrorist dataset from the Global Terrorist Database (GTD). The features country, region, attack type, target type, group name and weapon type are ranked highest of 23 attributes of the dataset for identification of the terrorist group name the using WEKA filter-based search and ranker routine. Data imbalance in the dataset is addressed by varying resample sample size percent for optimum performance. The classifier algorithm models were evaluated and compared on accuracy and build time as performance metrics using 10-fold cross validation, test split and ANOVA test. The results suggest that hybrid classifier algorithm models yield higher accuracy rates, accuracy rates for 10-fold cross validation are higher than the rates for test split and that resample sample size percent as a technique to solve class imbalance affects accuracy and yields optimum accuracy rates at resample sample size percent of 1000 for the available dataset. The results show a significant improvement in accuracy between the control group and the experimental group. The study concludes that hybrid KD (a combination of K-Nearest Neighbor and Decision trees) outperformed all other classifier algorithm models at resample sample size percent of 1000 with an accuracy rate of 88.18% and build time of 0.03 seconds for 10- fold cross validation and accuracy rate of 87.66% and build time of 1.03 seconds for test split in the identification of terrorist groups in the aftermath of an attack for the sub-Sahara Africa dataset. The study makes contribution by developing a systematic process of building a hybrid classifier algorithm model and establishing a resample sample size percent of 1000 for optimum accuracy rates for the dataset.

TABLE OF CONTENTS

TITLE PAGE.....	i
DECLARATION	ii
ACKNOWLEDGEMENT	iii
DEDICATION	iv
ABSTRACT.....	v
TABLE OF CONTENTS.....	vi
LIST OF ABBREVIATIONS.....	ix
OPERATIONAL DEFINITION OF KEY TERMS	x
LIST OF TABLES	xii
LIST OF FIGURES	xiii
LIST OF APPENDICES.....	xv
CHAPTER ONE: INTRODUCTION	1
1.1 Background to the Study.....	1
1.2 Statement of the Problem.....	2
1.3 Purpose of the Study	3
1.4 Specific Objectives of the Study.....	3
1.5 Research Questions.....	3
1.6 Significance of the Study	3
1.7 Scope and Limitations of the Study	4
1.8 Assumptions of the Study	4
1.9 Contribution	4
CHAPTER TWO: LITERATURE REVIEW	5
2.1 Terrorism and Response to Terrorism	5
2.2 Machine Learning Methods	6
2.2.1 Naïve Bayes	7
2.2.2 K-Nearest Neighbor	8
2.2.3 Decision Trees	10
2.2.4 Support Vector Machine	12
2.2.5 Multi-Layer Perceptron.....	14

2.2.6 The Concept of Ensemble and Hybrid Machine Learning Models	19
2.2.7 Ensemble Combination Techniques	21
2.2.8 Review of Previous works	26
2.3 Classifier Algorithm Performance Evaluation and Comparison Metrics	33
2.4 Data Mining Tools	35
2.4.1 Waikato Environment for Knowledge Analysis	35
2.4.2 Knostanz Information Miner.....	35
2.4.3 Rapid Miner	35
2.4.4 Orange.....	36
2.5 Data Mining Methodologies	36
2.6 Feature Selection.....	37
2.7 Class Imbalance Problem.....	38
2.8 Chapter Summary and Gap.....	39
CHAPTER THREE: RESEARCH METHODOLOGY	40
3.1 Research Design.....	40
3.2 Data Mining Methodology and Tools.....	42
3.3 Terrorism Dataset and Collection Methodology.....	43
3.4 Data Pre-processing	43
3.5 The Flow of Building the Hybrid Classifier Model for Identification of Terrorist groups	45
3.5.1 Build and Evaluate Base Classifier Algorithm Models for Identification of Terrorist Groups	47
3.5.2 Build and Evaluate Hybrid Classifier algorithm models for Identification of Terrorist Groups	54
3.5.3 The KD Hybrid Architecture	58
CHAPTER FOUR: RESULTS & DISCUSSIONS.....	60
4.1 Build and Evaluate Base Classifier algorithm models for the Identification of Terrorist Group	60
4.1.1 Decision Tree	60
4.1.2 K-Nearest Neighbor	64
4.1.3 Support Vector Machine	68

4.1.4 Multi -Layer Perceptron.....	72
4.1.5 Naïve’ Bayes.....	76
4.2 Build and Evaluate Hybrid Classifier Algorithm Models for the Identification of Terrorist Groups	81
4.2.1 Hybrid KNMSD.....	82
4.2.2 Hybrid KNSD	86
4.2.3 Hybrid KND.....	90
4.2.4 Hybrid KD	94
4.3 ComparePerformance of Classifier Algorithm Models in the Identification of Terrorist Groups	99
4.3.1 Compare Error Rate Percentage and Build Time for 10-Fold Cross Validation	100
4.3.2 Compare Error Rate and Build Time for Test Split.....	104
4.3.3 Statistical Test of Significance for Optimum Accuracy Averages	109
CHAPTER FIVE: CONCLUSIONS AND RECOMMENDATIONS.....	112
5.1 Conclusion	112
5.2 Recommendation	113
5.3 Future Work.....	114
REFERENCES.....	115
APPENDICES.....	126

LIST OF ABBREVIATIONS

ANN	-	Artificial Neural Networks
ARFF	-	Attribute-Related File Format
CRISP-DM	-	Cross Industry Standard Process for Data Mining
CSV	-	Comma Separated Values
DM	-	Data mining
DT	-	Decision Trees
GTD	-	Global Terrorism Database
KNN	-	K-Nearest Neighbor
LR	-	Logistic Regression
MLP	-	Multi Layer Perception
NB	-	Naïve Bayes
ROC	-	Receiver Operating Characteristic
SVM	-	Support Vector Machines
WEKA	-	Waikato Environment for Knowledge Analysis

OPERATIONAL DEFINITION OF KEY TERMS

Anti-terrorism:	The defensive efforts to reduce vulnerabilities of targets to terrorist attacks and to lessen the effects of terrorist attacks that do occur.
Building:	Refers to Construction, Training, validating, and testing of a machine learning classifier algorithm.
Classification:	Refers to supervised learning approach which specifies the class to which data elements belong to and is best used when output has finite and discrete values.
Classifier:	Special case of a hypothesis or discrete valued function that is used to assign categorical class labels to a data point.
Counterterrorism:	Offensive and military measures against terrorists to prevent, deter and respond to terrorist acts.
Hypothesis:	A certain function that is believed to be similar to the true function, the target function to be modeled machine learning classification.
Hybrid/Ensemble:	The concept of combining different classifier algorithm families to improve classification accuracy.
Identification:	The act of specifying the class to which a terrorist group belongs based on known attributes.
Prediction:	Refers to the output of an algorithm after it has been trained on historical dataset and applied to new data when forecasting the likelihood of a particular income.
Resample :	A data-based technique that produces a random sub-sample of dataset using either sampling with replacement or without replacement for imbalanced dataset, to achieve oversampling of the minority class, rather than under sampling of the majority class

Supervised learning	A machine learning paradigm where the algorithm learns on a labeled dataset, providing an answer key that the algorithm can use to evaluate its accuracy on training data.
Target function:	The function to be learnt or approximated in predictive modeling
Training sample:	Data point x in an available training set that is used for tackling a predicting modeling task
Terrorism:	Premeditated, ideologically, politically motivated violence perpetrated against non-combatant targets by sub-national groups or clandestine agents usually intended to influence an audience.

LIST OF TABLES

Table 4.1: DT Accuracy.....	61
Table 4.2: DT build time.....	63
Table 4.3: KNN Accuracy	65
Table 4.4: KNN Build Time	67
Table 4.5: SVM Accuracy	69
Table 4.6: SVM build time	71
Table 4.7: MLP accuracy	73
Table 4.8: MLP build time.....	75
Table 4.9: NB accuracy.....	77
Table 4.10: NB Build Time	79
Table 4.11: Hybrid KNMSD accuracy	83
Table 4.12: Hybrid KNMSD build time	85
Table 4.13: Hybrid KNSD accuracy.....	87
Table 4.14: Hybrid KNSD build time.....	89
Table 4.15: Hybrid KND accuracy	91
Table 4.16: Hybrid KND build time	93
Table 4.17: Hybrid KD Accuracy.....	95
Table 4.18: Hybrid KD Build Time.....	97
Table 4.19: 10- fold cross validation percentage error rate comparison.....	101
Table 4.20: Comparison of build time in seconds	103
Table 4.21: Test split percentage error rate comparison.....	105
Table 4.22: Test Split build time Comparison	107
Table 4.23: 10-fold cross validation optimum accuracy comparison between control and optimum	109
Table 4.24: AVOVA test for 10-fold cross validation accuracy	109
Table 4.25: Test split comparison between control and optimum accuracy.....	110
Table 4.26: ANOVA test for test split	110

LIST OF FIGURES

Figure 2.1: KNN model (Bouziane, Messabih, & Chouarfia, 2011)	9
Figure 2.2: Decision tree model (Chen & Liu, 2010)	11
Figure 2.3: An example of a two-class problem in a two-dimensional space. (Cortes & Vapnic, 1995).	13
Figure 2.4: MLP model (Kotsiantis, Zaharakis, & Pintelas, 2004)	15
Figure 2.5: Concept of Ensemble Classifier (Baba, Makhtar, Fadzili, & Awang, 2015)	19
Figure 2.6: CRISP-DM (Rahim, 2014).....	37
Figure 3.1: The hybrid classifier algorithm model process flow	46
Figure 3.2: NB Configuration setup (author).....	48
Figure 3.3: KNN configuration setup (author)	49
Figure 3.4: DT (J48) configuration setup (author).....	50
Figure 3.5: SVM (SMO) configuration setup (author)	52
Figure 3.6: MLP (back propagation) configuration setup	53
Figure 3.7: KNMSD configuration setup (author).....	55
Figure 3.8: KNSD configuration setup (author)	56
Figure 3.9: KND configuration setup (author)	57
Figure 3.10: KD configuration setup (author)	58
Figure 3.11: Architecture of the KD hybrid classifier algorithm model (Author).....	59
Figure 4.1: DT Accuracy	62
Figure 4.2: DT build time	64
Figure 4.3: KNN Accuracy	66
Figure 4.4: KNN build time	68
Figure 4.5: SVM accuracy	70
Figure 4.6: SVM build time	72
Figure 4.7: MLP accuracy.....	74
Figure 4.8: MLP build time	76
Figure 4.9: NB accuracy	78
Figure 4.10: NB build time	80
Figure 4.11: Hybrid KNMSD Accuracy.....	84
Figure 4.12: Hybrid KNMSD build time	86

Figure 4.13: Hybrid KNSD Accuracy	88
Figure 4.14: Hybrid KNSD build time	90
Figure 4.15: Hybrid KND Accuracy.....	92
Figure 4.16: Hybrid KND build time.....	94
Figure 4.17: KD accuracy	96
Figure 4.18: Hybrid KD build time.....	98
Figure 4.19: 10-fold cross validation error rate comparison.....	102
Figure 4.20: 10-fold cross build time comparison	104
Figure 4.21: Test Split Error rate comparison	106
Figure 4.22: Test split build time comparison	108

LIST OF APPENDICES

Appendix A: SGS Approval	126
Appendix B: MUERC Approval.....	127
Appendix C: NACOSTI Approval.....	128
Appendix D: GTD Distribution Letter.....	129
Appendix E: Sample Dataset	130
Appendix F: WEKA URL	131

CHAPTER ONE

INTRODUCTION

This chapter consists of background to the study, statement of the problem, purpose of the study, specific objectives, research questions, significance of the study, scope and limitations, assumptions of the study and ethical considerations.

1.1 Background to the Study

According to the Global Terrorism Index (2018), there has been a continued rise of terrorism which is a serious concern. Terrorist groups with time have continued to expand into other countries for instance, the Islamic state of Iraq and the Levant (ISIL) and its affiliates have expanded into 15 new countries. More so, Boko Haram which was formerly in Nigeria is also currently in Niger, Cameroon, and Chad. Other countries that have also been affected by terrorism over the past years are UK, USA, Turkey, France, Afghanistan, Syria, Iraq, India, Pakistan, Central Africa Republic, Somalia, Sudan, and Kenya to mention a few. Consequently, terrorist attacks and activities have globally led to rise in loss of life and property, fear, and general insecurity. Terrorist acts are planned and perpetrated by collections of loosely organized people operating in shadowy networks that are difficult to define and identify. Terrorism is considered a low-intensity form of warfare; however, terrorist plots and activities will leave an information signature, albeit not one that is easily detected (Popp, Armour, Senator, & Numrych, 2004). There is need to develop more accurate mechanisms of identifying terrorist groups. Data mining and automated data analysis techniques have become used as effective branch of the most important key features for many applications, data mining has a wide number of applications ranging from marketing and advertising of goods, services or products, artificial intelligence research, biological sciences, crime investigations to high-level government intelligence (Kalpana & Bansa, 2014). Recently there has been much concern on using data mining in detecting and investigating unusual patterns, crimes, terrorist activities and preventing the fraudulent behavior (Prasad, Sonali, & Sonali, 2014), some of different techniques used in that regard are entity extraction, clustering techniques, deviation detection, classification techniques, string comparator, and social network (Osemengbe & Uddin, 2014). Data mining, Sentiment analysis, text mining, machine learning techniques and predictive analytics are some of methodologies being used to identify and combat terrorism (Foster, 2017). The serious consequences of acts of terrorism and the difficulty in identifying the culprits has necessitated the need to improve on counterterrorism interventions.

The concept of combining classifier algorithms is proposed as a new direction for the improvement of the performance of individual machine learning algorithms (Kuncheva & Whitaker, 2003). Multiple, ensemble learning models have been theoretically and empirically shown to provide significantly better performance than single weak learners, especially while dealing with high dimensional, complex regression and classification problems. Hybrid classifier systems are ensemble classifiers that combine, and integrate different standard machine learning algorithms, resulting in improved performance, and more adaptivity (Kainulainen, 2010). For that reason, hybrid methods have found application in various real-world problems ranging from pattern recognition, medical diagnosis, financial forecasting, weather prediction as well as terrorism prediction (Pillry & Sikchi, 2014). Thus, active area of research in supervised learning is the study methods for the construction of good ensembles of classifiers (Villada & Drissi, 2002). The performance of hybrid/ensemble depends on accuracy of base classifiers, diversity among the base classifiers, decision making strategy (aggregation technique) and the number of the base classifiers among other factors (Vladislav, 2014).

While there has been previous work on evaluation of hybrid classifiers in identification of terrorist groups, no sufficient research has been conducted to determine the combinations of diverse classifier algorithms K-nearest Neighbor (KNN), Naïve Bayes (NB), Decision Trees (DT), Support Vector Machines (SVM), Multi-Layer Perceptron (MLP) and resample sample size percent that would yield optimum accuracy estimates. The need to build and evaluate hybrid machine learning classifier models becomes imperative with class imbalanced datasets. Class imbalance leads to class imbalance problem.

1.2 Statement of the Problem

The urgency of responding to a terrorist attack and the subsequent nature of analysis required to identify the group involved in the attack demands that the performance of machine learning classifier algorithms used in identification of the terrorist groups yield highly accurate outcomes. A strong case can be made for combining models across algorithm families as a means of improving performance. To date there has not been sufficient research that attempts to find combinations of KNN, DT, MLP, SVM and NB and resample sample size percent for better accuracy in the identification of terrorist groups in the aftermath of an attack. It is equally important to understand the hybrid component classifier algorithms and the correlation between resample sample size percent and performance.

1.3 Purpose of the Study

The aim of the study was to build and evaluate hybrid classifier algorithms for identification of terrorist groups in the aftermath of an attack.

1.4 Specific Objectives of the Study

The specific objectives of the study were:

- i. To build base classifier algorithm models for identification of terrorist groups in the aftermath of an attack.
- ii. To evaluate base classifier algorithm models of terrorist groups in the aftermath of an attack.
- iii. To build hybrid classifier algorithm models of terrorist groups in the aftermath of an attack.
- iv. To evaluate hybrid classifier algorithm models of terrorist groups in the aftermath of an attack
- v. To compare the performance of the classifier algorithm models of terrorist groups in the aftermath of an attack.

1.5 Research Questions

To achieve the objectives, the following research questions were answered:

- i. What approach can be used to build and evaluate base classifier algorithm models of terrorist groups in the aftermath of an attack?
- ii. What approach can be used to build and evaluate hybrid classifier algorithm models of terrorist groups in the aftermath of an attack?
- iii. What are the outcomes of analysis of performance of classifier algorithm models of terrorist groups in the aftermath of an attack?

1.6 Significance of the Study

The study is important to law enforcement community in improving performance of their data mining tools and thus better response to terror attacks. The study is important to the researchers as it lays foundation for further research in hybrid machine learning approaches.

1.7 Scope and Limitations of the Study

The scope of study was to build and evaluate a hybrid machine learning classifier algorithm for optimum accuracy from combinations of KNN, DT, MLP, SVM, & NB using bagging and majority voting combination approach and evaluate the effects of resample sample size percent on the accuracy. The generalizability of the study results is limited to machine learning classifier algorithms KNN, DT, NB, SVM, and MLP; Bagging and majority voting as the combination technique; the available dataset for sub-Saharan Africa; accuracy and build time as the performance measure metrics.

1.8 Assumptions of the Study

The researcher assumed that Global terrorism database (GTD) is a credible data source for terrorism related research activities.

1.9 Contribution

The study makes the following contributions:

- i. Methodological contribution by developing a systematic process of building a hybrid machine learning classifier algorithm model for better accuracy.
- ii. Theoretical contribution by establishing a combination and resample sample size percent of base classifiers for better identification accuracy.
- iii. Policy and practice by developing hybrid architecture for better accuracy in identification of terrorist group provides can be used in improving policy and practice on counter terrorism efforts.

CHAPTER TWO

LITERATURE REVIEW

This chapter consists reviews literature on terrorism and counterterrorism, machine learning methods, ensemble learning concepts, ensemble techniques, related literature, cross validation, classifier algorithm performance evaluation and comparison metrics, data mining tools, data mining methodologies, feature selection, class imbalance problem and summary and gap.

2.1 Terrorism and Response to Terrorism

Terrorism is one of the major threats to human life in the 21st Century. It is instigated by the emergence of extremists who propagate use of violence with the intention of causing mass fatalities and casualties. As destructive activity that destroys property, and causes human casualties and fatalities, terrorism has obvious economic impacts and consequences for countries in general (Barth, Tong, McCarthy, Phumiwasana, & Yago, 2006). According to the Global Terrorism Index there has been a continued rise in terrorism (GTI, 2018) which is a cause for serious concern today. Terrorist groups with time have managed to expand into other countries for instance; the Islamic State of Iraq and the Levant (ISIL) and its affiliates have expanded into 15 new countries. More so, Boko Haram which was formerly in Nigeria is also currently in Niger, Cameroon, and Chad. This has consequently resulted in more terrorist attacks and activities globally leading to rise in insecurity, fear and a higher perceived risk of terrorism (Foster, 2017).

According to Magogo, a terrorist carried an attack in Spain by ploughing a van into crowds and thus killing people. In Germany, July 2016, in separate incidences; a man blew himself up, an Iranian German shot people, while another terrorist hacked passengers in a train. In France on the 14th July 2016, a terrorist in a lorry mowed down revelers on Bastille Day, in another attack on January 7th, 2015 in the Charlie Hebdo attack, two gunmen carried out an attack on the French satirical weekly Newspaper, Charlie Hebdo, in Paris killing and injuring many. These attacks were by such Islamic militant terror groups such as Al-Qaeda, ISIS, and others were by Muslim individuals labelled as terrorists. In the USA apart from the 11 September 2001 attack which brought attention to terrorism in the world, there have been shooting attacks, bombings, stabbings and vehicle attacks on people (Magogo, 2017). Africa as a continent is no exception to terrorism as the terror attacks by radical groups increased

immensely. While Global terror groups such as the Islamic State of Iraq and Syria (ISIS) and Al-Qaeda have made their presence felt in the region, other local groups are Boko Haram and Al-Shabaab. Boko haram has claimed over 20,000 lives, displaced 2.6 million people, created over 75,000 orphans and caused unimaginable damage financially (Barth, Tong, McCarthy, Phumiwasana, & Yago, 2006). Counter terrorism measures have been put in place over time to prevent and respond to terrorist activities. Accurate identification of terrorist groups immediately after an attack is one of the most important steps for counter terrorism interventions. As soon as the group name involved is found, responders are able to develop effective strategies to catch the culprits (Robert & Johnson, 2016). Technology has been employed in countering terrorism with the aim of improving the effectiveness of counterterrorism interventions (Popp, Armour, Senator, & Numrych, 2004).

By utilizing emerging technologies to collect and effectively analyze intelligence, law enforcement are able to better understand how the enemy operates, or plans to operate, as well as identify possible threats before an attack occurs (Hongbo, 2010). The use of predictive analytics can assist in providing law enforcement with information to better prevent, prepare for, and recover from an all-hazards event. While this technology was originally developed for private sector use, a partnership with the law enforcement community can help develop a process and procedure to use predictive analytics to better safeguard against terrorist-related threats (Isson, 2012). Employment of machine learning algorithms is the conspicuous approach in data mining to generate better performing predictive models for improved counter terrorism response.

2.2 Machine Learning Methods

Machine learning is a data mining Technique. The main goal of data mining is to extract useful, hidden predictive knowledge from large data sets in a human understandable structure. It involves database, data management and pre-processing tools, model and interface capabilities, post-processing of discovered structure, visualization, and online updating methods for finding hidden patterns, and predictive information that experts may miss because it lies outside their expectations. Machine learning can be either supervised learning, unsupervised or reinforcement learning. Machine learning models can be classified as parametric or nonparametric; parametric models operate within a fixed number of parameters regardless of data growth and make particular assumptions about data based on parameter characteristics, whereas nonparametric models can adapt and grow with an

increase in data size and do not make any assumptions of raw data (Hongbo, 2010). Supervised learning is one of the sub-disciplines of Machine Learning. In supervised learning we are in search for the optimized function (a.k.a. model) to map input features to an output. And to find that optimized function we have a set of data with all the features and the outputs in our hands. By learning or finding the patterns from this data, we get the mapping function. Supervised learning branches to two sub-parts depending on the output property. If we are searching for a group, category or class it is called classification; or if we are looking for a value (usually continuous) then it is called regression(Rizwan, Masrah, Aida, Payam, & Nasim, 2013).Classification is used to predict the categorical class label of a given data instance, so as to classify it into one of the predetermined classes. It is a two-step process, in first step classification algorithm uses training dataset to build a classifier, and then in second step this classifier is used to predict the class label of a given unlabelled data instance (Verma, 2019) The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, and the value class label is unknown. The input data for the classification is a set of instances. Each instance is a record of data in the form of (x, y) where x is the features set and y is the target variable (class label). A classification equation can be expressed as follows:

$$Y=f(x).....2.1$$

Where Y is the output, f is the prediction function and (x) features

The key question when dealing with classification is not whether a learning algorithm is superior to others, but under which conditions a particular method can significantly outperform others on a given application problem (Kotsiantis, Zaharakis, & Pintea, 2004).There are various classification approaches proposed by the researchers in machine learning, statistics, and pattern recognition (Jantan, Hamdan, & Othman, 2009).These approaches are Naïve Bayes, K-Nearest Neighbor, Decision Trees, Multi-Layer Perceptron, Support Vector Machine and Hybrid Classifiers (Ozekes & Osman, 2003).

2.2.1 Naïve Bayes

Naive Bayes ‘classification algorithm is based on Bayes’ Theorem of Posterior Probability. This algorithm works by predicting probability that a given data instance belongs to a particular class. The probability that a given data vector is in class C, is called posterior

probability and is denoted by $P(C|X)$. Finally, Maximum a posteriori hypothesis is applied (Verma, 2018).

$$P(c/x) = (P(x/c)P(c))/P(x) \dots \dots \dots 2.2$$

Where $P(c/x)$ is the posterior probability, $P(x/c)$ is the likelihood, $P(c)$ is the class prior probability, and $P(x)$, is the predictor prior probability. Bayes theorem provides a way of calculating the posterior probability, $P(c/x)$, from $P(c)$, $P(x)$, and $P(x/c)$. Naive Bayes classifier considers that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. Naïve Bayes assumes that descriptive attributes are conditionally independent of each other given the class label is known; in other words, Bayesian Classifiers have the ability to predict the probability that a given tuple belongs to a particular class (Hongbo, 2010). Practically there are some complexities with Bayesian Classifier for instance, it requires prior information of probabilities and in absence of that they are frequently predicted on the basis of background knowledge and earlier available data about original distributions (Mitchel, 1997). The other complexity is the computational cost that is required to find out the Bayes finest hypothesis in common case, but in certain cases this cost could be minimized. The Advantages of Naïve Bayes Classifier are it proves success in solving different classification tasks effectively, as it is robust to isolated noisy data, and also robust against irrelevant attributes. The Naïve Bayes method can also cope with null values (Batch & Aravindan, 2011).

2.2.2 K-Nearest Neighbor

K-Nearest Neighbor (KNN) Classifier algorithm is one of the top ten algorithms used for the classification and regression. It is, also known as lazy learner or instance-based, in that it stores all of the training samples and do not build a classifier until a new sample needed to be classified that makes predictions based on KNN labels assigned to test sample (Breiman, 2001). KNN is based on the principle that the instances within a dataset will generally exist in close proximity to other instances that have similar properties as shown in fig.2.1. If the instances are tagged with a classification label, then the value of the label of an unclassified instance can be determined by observing the class of its nearest neighbours. The KNN locates the k nearest instances to the query instance and determines its class by identifying the single most frequent class label. For more accurate results, several algorithms

use weighting schemes that alter the distance measurements and voting influence of each instance.

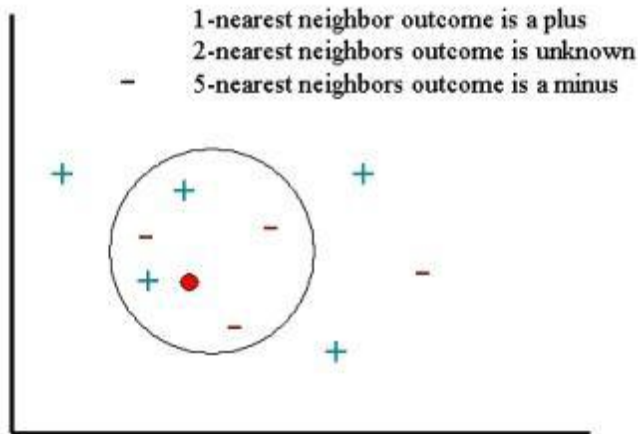


Figure 2.1: KNN model (Bouziane, Messabih, & Chouarfia, 2011)

A survey of weighting schemes is given by (Wettschereck, Aha, & Mohrit, 1997). The power of KNN has been demonstrated in a number of real domains, but there are some reservations about the usefulness of KNN, such as: they have large storage requirements, they are sensitive to the choice of the similarity function that is used to compare instances, and they lack a principled way to choose k , except through cross-validation or similar, computationally-expensive technique (Guo, Wang, Bell, & Greer, 2003). The choice of k affects the performance of the KNN algorithm. Consider the following reasons why a k nearest Neighbour classifier might incorrectly classify a query instance: When noise is present in the locality of the query instance, the noisy instance(s) win the majority vote, resulting in the incorrect class being predicted. A larger k could solve this problem; When the region defining the class, or fragment of the class, is so small that instances belonging to the class that surrounds the fragment win the majority vote. A smaller k could solve this problem. Breiman reported that the stability of nearest neighbor classifiers distinguishes them from decision trees and some kinds of neural networks (Breiman, 1996). A learning method is termed “unstable” if small changes in the training-test set split can result in large changes in the resulting classifier. A major disadvantage of instance-based classifiers is their large computational time for classification. A key issue in many applications is to determine which of the available input features should be used in modeling via feature selection (Yu & Liu, 2002), because it could improve the classification accuracy and scale down the required

classification time. Furthermore, choosing a more suitable distance metric for the specific dataset can improve the accuracy of instance-based classifiers. KNN is implemented in algorithm below.

KNN implementation algorithm (Kotsiantis, 2007)

Function KNN (train_patterns, train_targets, test_patterns)

end

Uc-a set of unique labels of train-targets;

N-size of test patterns

For $i=1 \dots N$,

dist:=EQ-Dist(train-patterns,test-patterns(i))

idxs:=sort(dist)

topk Classes:=train_targets (idxs(1:Knn))

cls:=Dominating Class(topk Classes)

test-targets (i):=cls

2.2.3 Decision Trees

A Decision Tree Classifier consists of a decision tree generated on the basis of instances. A decision tree is a classifier expressed as a recursive partition of the instance space. The decision tree (Chen & Liu, 2010) consists of nodes that form a rooted tree, meaning it is a directed tree with a node called “root” that has no incoming edges. All other nodes have exactly one incoming edge. A node with outgoing edges is called an internal or test node. All other nodes are called leaves (also known as terminal or decision nodes) as shown in fig. 2.2. In a decision tree, each internal node splits the instance space into two or more sub-spaces a certain discrete function of the input attributes values. The root and the internal nodes are associated with attributes, leaf nodes are associated with classes. Basically, each non-leaf node has an outgoing branch for each possible value of the attribute associated with the node. To determine the class for a new instance using a decision tree, beginning with the root, successive internal nodes are visited until a leaf node is reached. At the root node and at each internal node, a test is applied. The outcome of the test determines the branch traversed, and the next node visited. The class for the instance is the class of the final leaf node. The estimation criterion in the decision tree algorithm is the selection of an attribute to test at each decision node in the tree. The goal is to select the attribute that is most useful for classifying examples (Kotsiantis, Zaharakis, & Pintelas, 2004). A good quantitative measure of the worth

of an attribute is a statistical property called information gain that measures how well a given attribute separates the training examples according to their target classification. This measure is used to select among the candidate attributes at each step while growing the tree.

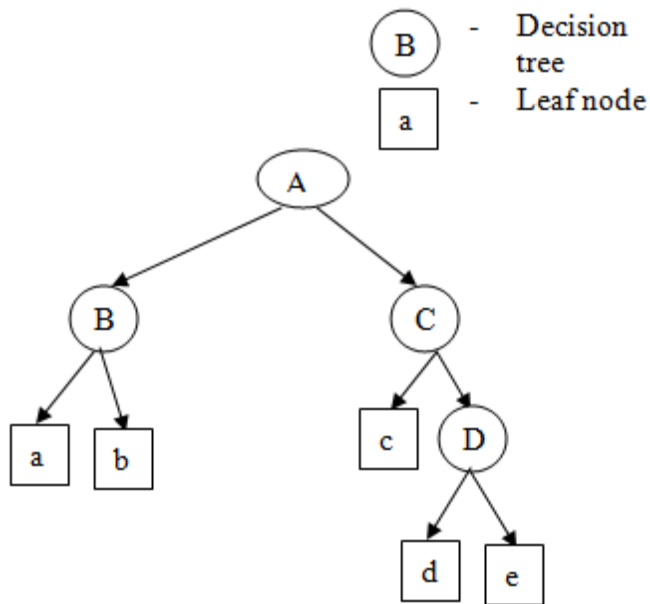


Figure 2.2: Decision tree model (Chen & Liu, 2010)

Decision tree implementation algorithm (Kotsiantis, Zaharakis, & Pintelas, 2004)

Function DTL (examples, attributes, default) **returns a decision tree**

if examples is **empty** then return *default*

else if all examples have the same classification **then return the classification**

else if attributes is empty then return $\text{MODE}(\text{examples})$

else

best ← CHOOSE-ATTRIBUTE (*attributes*, *examples*)

tree ← a new decision tree with root test *best*

for each value v_i **of** *best* **do**

examples_i ← { elements of *examples* with *best* = v_i }

subtree ← DTL (*examples_i*, *attributes*-*best*, $\text{MODE}(\text{examples})$)

add a branch to *tree* **with label** v_i **and subtree** *subtree*

return *tree*

Decision Trees (DT) are predictive decision support tools that create mapping from observations to possible consequences, a statistical data mining technique that express independent and dependent attributes logically and, in a tree, shaped structure(Sohini &

Shaikh, 2014). As a major approach decision tree induction has received a great attention from researchers in the last two decades, as a result there are a number of decision tree induction methods have been developed such as ID3, C4.5, C5, C&RT, and CHAID(Hongbo, 2010). The strengths of DT are it assigns a class label to an unseen record, as well as explains why the decision is made in an easy-to-understand classification rule. DT classifies unseen records efficiently, and it can handle both categorical and continuous attributes, the attribute selection measures used by DT induction method are capable of indicating the most important attribute in relation to class. The researchers mentioned the weaknesses points of DT; it has high error rates when the training set contains a small number of instances of a large variety of different classes, DT algorithm may not work well on data sets when attribute split in any other shape exist. DT are automatically quite expensive to build.ID3 is one of the popular DT algorithms that deal with nominal data sets and does not deal with missing values(Kalpana & Bansa, 2014).ID3 is the classical version of the decision tree induction. It mainly works on the selections of attributes at all the levels of decision tree that is based on information entropy(Farysal, Wasi, & Usman, 2014).This algorithms is a good selection where the research needs accuracy as it improves the accuracy and speed of classification; it is helpful when dealing with a large scale problem. Furthermore, it has some other weaknesses such as; it does not have the quality of backtracking during the search, and it is sensitive to noise (Chen & Liu, 2010).

2.2.4 Support Vector Machine

SVMs revolve around the notion of a “margin”—either side of a hyper-plane that separates two data classes as shown in fig. 2.3. Maximizing the margin and thereby creating the largest possible distance between the separating hyper-plane and the instances on either side of it has been proven to reduce an upper bound on the expected generalization error. In the case of linearly separable data, once the optimum separating hyper-plane is found, data points that lie on its margin are known as support vector points and the solution is represented as a linear combination of only these points. Other data points are ignored.

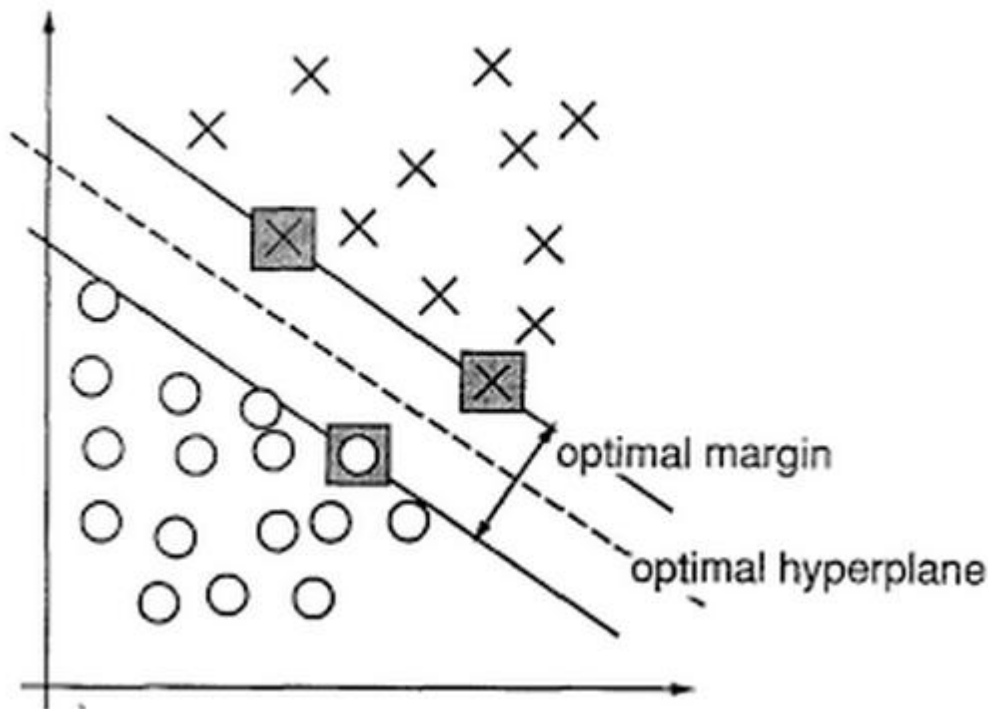


Figure 2.3: An example of a two-class problem in a two-dimensional space. The support vectors are marked with grey squares(Cortes & Vapnic, 1995).

Therefore, the model complexity of an SVM is unaffected by the number of features encountered in the training data (the number of support vectors selected by the SVM learning algorithm is generally small). For this reason, SVMs are well suited to deal with learning tasks where the number of features is large with respect to the number of training instances. Even though the maximum margin allows the SVM to select among multiple candidate hyper-planes, for many datasets, the SVM may not be able to find any separating hyper-plane at all because the data contains misclassified instances. The problem can be addressed by using a soft margin that accepts some misclassifications of the training instances (Veropoulos, Campbell, & Cristianini, 1999). Nevertheless, most real-world problems involve non-separable data for which no hyper-plane exists that successfully separates the positive from negative instances in the training set. One solution to the inseparability problem is to map the data onto a higher-dimensional space and define a separating hyper-plane there. This higher-dimensional space is called the feature space, as opposed to the input space occupied by the training instances. With an appropriately chosen feature space of sufficient dimensionality, any consistent training set can be made separable. A linear separation in feature space corresponds to a non-linear separation in the original input space. Mapping the

data to some other (possibly infinite dimensional) Hilbert space H as $\Phi : R^d \rightarrow H$. Then the training algorithm would only depend on the data through dot products in H , i.e., on functions of the form $\Phi(x_i) \cdot \Phi(x_j)$. If there were a “kernel function” K such that $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$, we would only need to use K in the training algorithm and would never need to explicitly determine Φ . Thus, kernels are a special class of function that allow inner products to be calculated directly in feature space, without performing the mapping described above (Scholkopf, Burges, & Smola, 1999). Once a hyper-plane has been created, the kernel function is used to map new points into the feature space for classification. The selection of an appropriate kernel function is important, since the kernel function defines the feature space in which the training set instances will be classified. Genton described several classes of kernels, however, he did not address the question of which class is best suited to a given problem. It is common practice to estimate a range of potential settings and use cross-validation over the training set to find the best one. For this reason, a limitation of SVMs is the low speed of the training. Training the SVM is done by solving N th dimensional Quadratic Programming (QP) problem, where N is the number of samples in the training dataset. Solving this problem in standard QP methods involves large matrix operations, as well as time-consuming numerical computations, and is mostly slow and impractical for large problems. Sequential Minimal Optimization (SMO) is a simple algorithm that can, relatively quickly, solve the SVM QP problem without any extra matrix storage and without using numerical QP optimization steps at all (Platt, 1999). SMO decomposes the overall QP problem into QP sub-problems. The training optimization problem of the SVM necessarily reaches a global minimum, and avoids ending in a local minimum, which may happen in other search algorithms such as neural networks. However, the SVM methods are binary, thus in the case of multiclass problem one must reduce the problem to a set of multiple binary classification problems. Discrete data presents another problem, although with suitable rescaling good results can be obtained (Keerthi & Gilbert, 2002).

2.2.5 Multi-Layer Perceptron

A multi-layer neural network consists of large number of units (neurons) joined together in a pattern of connections as shown in fig. 2.4. Units in a net are usually segregated into three classes: input units, which receive information to be processed; output units, where the results of the processing are found; and units in between known as hidden units. Feed-forward Artificial Neural Networks (ANNs) (Figure 2.4) allow signals to travel one way only, from input to output.

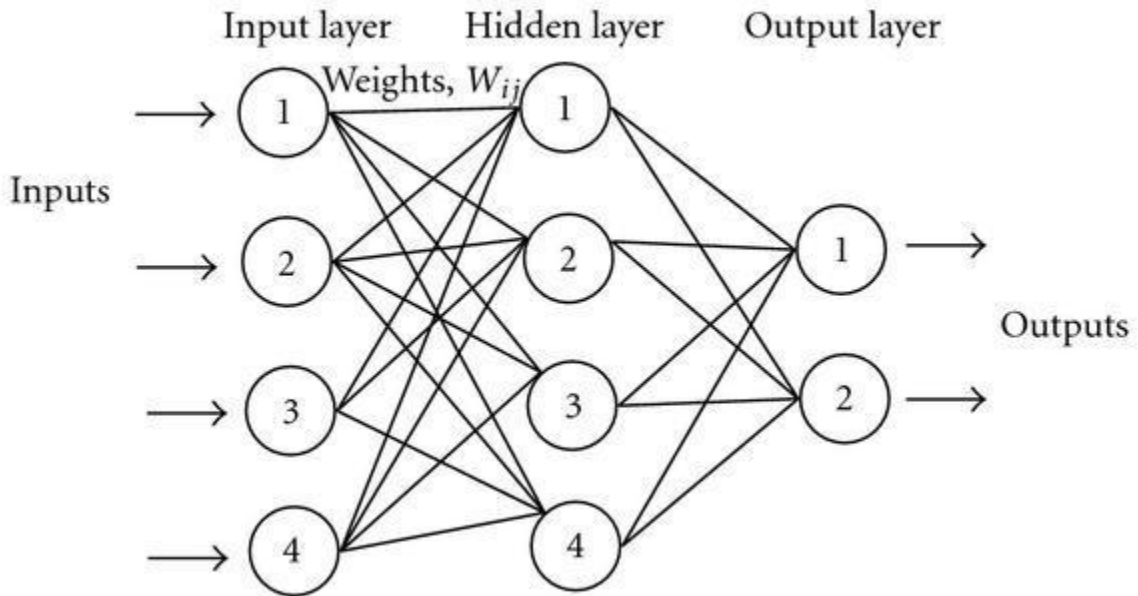


Figure 2.4: MLP model (Kotsiantis, Zaharakis, & Pintelas, 2004)

Generally, properly determining the size of the hidden layer is a problem, because an underestimate of the number of neurons can lead to poor approximation and generalization capabilities, while excessive nodes can result in overfitting and eventually make the search for the global optimum more difficult (Camargo & Yoneyama, 2001). ANN depends upon three fundamental aspects, input and activation functions of the unit, network architecture and the weight of each input connection. Given that the first two aspects are fixed, the behavior of the ANN is defined by the current values of the weights. The weights of the net to be trained are initially set to random values, and then instances of the training set are repeatedly exposed to the net. The values for the input of an instance are placed on the input units and the output of the net is compared with the desired output for this instance. Then, all the weights in the net are adjusted slightly in the direction that would bring the output values of the net closer to the values for the desired output (Kon & Plaskota, 2000). There are several algorithms with which a network can be trained (Neocleous & Schizas, 2002). However, the most well-known and widely used learning algorithm to estimate the values of the weights is the Back Propagation (BP) algorithm. The general rule for updating weights is:

$$\Delta W_{ji} = \eta \delta_j O_i \quad \dots\dots\dots 2.3$$

where:

- η is a positive number (called learning rate), which determines the step size in the gradient descent search. A large value enables back propagation to move faster to the target weight configuration, but it also increases the chance of its never reaching this target.
- O_i is the output computed by neuron I ,
- $\delta_j = O_j (1 - O_j)(T_j - O_j)$ for the output neurons, where T_j is the wanted output for the neuron j and,
- $\delta_j = O_j (1 - O_j) \sum_k \delta_k W_{kj}$ for the internal (hidden) neurons.

During classification, the signal at the input units propagates all the way through the net to determine the activation values at all the output units. Each input unit has an activation value that represents some feature external to the net. Then, every input unit sends its activation value to each of the hidden units to which it is connected. Each of these hidden units calculates its own activation value and this signal are then passed on to output units. The activation value for each receiving unit is calculated according to a simple activation function. The function sums together the contributions of all sending units, where the contribution of a unit is defined as the weight of the connection between the sending and receiving units multiplied by the sending unit's activation value. This sum is usually then further modified, for example, by adjusting the activation sum to a value between 0 and 1 and/or by setting the activation value to zero unless a threshold level for that sum is reached. Feed-forward neural networks are usually trained by the original back propagation algorithm or by some variant. Their greatest problem is that they are too slow for most applications. One of the approaches to speed up the training rate is to estimate optimal initial weights (Yam & Chow, 2001). Genetic algorithms have been used to train the weights of neural networks (Siddique & Tokhi, 2001) and to find the architecture of neural networks (Yen & LU, 2000). There are also Bayesian methods in existence which attempt to train neural networks. Vivarelli and Williams compare two Bayesian methods for training neural networks. A number of other techniques have emerged recently which attempt to improve ANNs training algorithms by changing the architecture of the networks as training proceeds (Vivarelli & Williams, 2001). These techniques include pruning useless nodes or weights (Castellano, Fanelli, & Pelillo, 1997), and constructive algorithms, where extra nodes are added as required (Parekh, Yang, & Honavar, 2000). ANN learning can be achieved, among others, through synaptic weight modification, network structure modifications (creating or deleting neurons or synaptic connections), use of suitable attractors or other suitable stable

state points, and appropriate choice of activation functions. Since back-propagation training is a gradient descending process, it may get stuck in local minima in this weight-space. It is because of this possibility that neural network models are characterized by high variance and unsteadiness. Radial Basis Function (RBF) networks have been also widely applied in many science and engineering fields (Robert & Howlet, 2001). An RBF network is a three-layer feedback network, in which each hidden unit implements a radial activation function and each output unit implements a weighted sum of hidden units' outputs. Its training procedure is usually divided into two stages. First, the centers and widths of the hidden layer are determined by clustering algorithms. Second, the weights connecting the hidden layer with the output layer are determined by Singular Value Decomposition (SVD) or Least Mean Squared (LMS) algorithms. The problem of selecting the appropriate number of basis functions remains acritical issue for RBF networks. The number of basis functions controls the complexity and the generalization ability of RBF networks. RBF networks with too few basis functions cannot fit the training data adequately due to limited flexibility. On the other hand, those with too many basis functions yield poor generalization abilities since they are too flexible and erroneously fit the noise in the training data. To sum up, ANNs have been applied to many real-world problems but still, their most striking disadvantage is their lack of ability to reason about their output in a way that can be effectively communicated. For this reason, many researchers have tried to address the issue of improving the comprehensibility of neural networks, where the most attractive solution is to extract symbolic rules from trained neural networks. Setiono and Loew divided the activation values of relevant hidden units into two sub-intervals and then found the set of relevant connections of those relevant units to construct rules (Setiono & Loew, 2000). However, it is also worth mentioning that Roy identified the conflict between the idea of rule extraction and traditional connectionism (Roy, 2000). In detail, the idea of rule extraction from a neural network involves certain procedures, specifically the reading of parameters from a network, which is not allowed by the traditional connectionist framework that these neural networks are based on. Neural networks are usually more able to easily provide incremental learning than decision trees (Saad, 1998).

MLP back propagation implementation algorithm (Kotsiantis, 2007)

Neural network learning for classification or
prediction using the back propagation:

Input: D, a data set consisting of the training tuples and their associated target values; η , the learning rate;

Network: A multilayer feed-forward network.

Output: A trained neural network.

Method:

Initialize all weights and biases in network;

While terminating condition is not satisfied{

for each training tuple X in D

{

//Propagate the inputs forward:

for each input layer unit j

{

$O_j = I_j$; //output of an input unit is its actual value

for each hidden or output layer unit j

{

$I_j = \sum_i W_{ij} O_i + q_j$ //compute the net input of unit j with respect to the previous layer, i

$O_j = 1 / (1 + e^{-I_j})$;

}

// compute the output of each unit j

//Backpropagate the errors:

for each unit j in output layer

$Err_j = O_j(1 - O_j)(T_j - O_j)$; //compute the error

for each unit j in the hidden layers, from the last to the first hidden layer

$Err_j = O_j(1 - O_j) \sum_k Err_k W_{jk}$; //compute the error with respect to the next higher layer, k

```

for each weight  $w_{ij}$  in network
{
 $\Delta W_{ij} = (1) \text{Err}_j O_i$ ; //weight increment
 $W_{ij} = W_{ij} + \Delta W_{ij}$ ; g//weight update
for each bias  $\Theta_j$  in network
{
 $\Delta \Theta_j = (1) \text{Err}_j$ ; //bias increment
 $\Theta_j = \Theta_j + \Delta \Theta_j$ ; } bias update
}}

```

2.2.6 The Concept of Ensemble and Hybrid Machine Learning Models

Ensemble learning is a machine learning discipline in which many base classifiers are trained on given datasets in order to provide a solution to given problems (Hui, 2013),(Tao, 2003),(Zhou & Tang, 2003),(Dietterich, 2000). An ensemble consists of a group of base classifiers that are trained (such as neural networks or decision trees), whose decisions are integrated for classifying new instances (Dietterich, 2000). It is sometimes referred to as a mixture of experts (Ricardo, 2014)(Polikar, 2012), committees (Santana, Siva, Canuto, Pinto, & Vale, 2010), multi-classifier systems, fusion of experts (Neto & Canuto, 2004), selection or thinning (Banfield, Hall, Bowyer, & Kegelmeyer, 2002). The main aim of ensemble method is to integrate a set of models that are used for solving different tasks so as to come up with enhanced composite global model which produces higher accuracy and reliable estimate than what can be achieved through a single model (Quinlan, 1996),(Optiz & Maclin, 1999),(Kuncheva & Whitaker, 2003)].

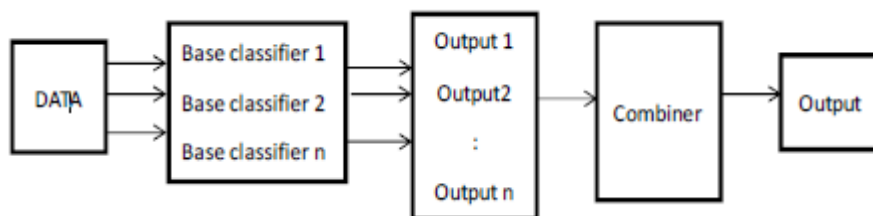


Figure 2.5: Concept of Ensemble Classifier (Baba, Makhtar, Fadzili, & Awang, 2015)

In fig. 2.5, data is fed into various classifiers, different outputs were obtained, and they are then combined into a single output by the combiner. The method falls into two categories, namely homogeneous and heterogeneous ensemble; if the ensemble is made up of the same type of learning algorithm, say neural network, then it is called homogeneous, but if it is made up of more than one different learning algorithm, for example, neural network and decision trees, then it is referred to as heterogeneous (Kuncheva & Whitaker, 2003),(Santana, Siva, Canuto, Pinto, & Vale, 2010). However, when ensemble is built with models that are homogeneous, neither high accuracy nor diversity would make the ensemble obtain higher accuracy than the individual classifiers (Wang, 2010). Ensemble method is chosen because it has been proven that it produces more accurate results than when a single model is used to solve the same problem (Dietterich, 2000). Ensemble technology was introduced to the area of data classification and has since obtained great success. In order to achieve great success with the ensemble method, two criteria are taken into consideration, which are that the ensemble should have enough diversity introduced into it, and secondly, a suitable integrated method must be chosen in order to combine the decision of the base classifiers to a single output. The term diversity refers to the fact that indicates that the base classifiers errors are uncorrelated. Diversity is typically considered as a quantified estimation of the distinction of making the same errors among models in an ensemble (Wang, 2008) or it can simply be put as the difference between base classifiers in the ensemble(Wang & Yao, 2013). It is grouped into two: pairwise and non-pairwise (Kuncheva, 2005), (Kuncheva & Whitaker, 2003). Their examples can be found in literatures (Kuncheva & Whitaker, 2003), (Tang, Suganthan, & Yao, 2006) such as entropy, double default measure and Q-statistic to generalize diversity measures. However, the pairwise has the drawback of not having effectiveness in measuring diversity and shows no or little relation with the accuracy of the ensemble because it only considers the difference of two models and hence, they are not valuable. Others split diversity into destructive and constructive or negative and positive diversity (Wang, 2008). Combination or integration method is used to combine the output of the base classifiers in the ensemble. They are categorized differently in the literature, such as fusion, selection, and hybrid (Santana, Siva, Canuto, Pinto, & Vale, 2010), static and adaptive (Ricardo, 2014), utility-based and evidence-based (Ghosh, NG, & Srinivasan, 2011), evidence, fusion, genetic algorithm, and voting based aggregation techniques (Asmita & Shukla, 2014). Fusion based methods are the ones in which all classifiers are assumed to be of equal experience in the whole feature space, and all classifier's decision are considered for any given input pattern. Examples are sum, majority voting, naïve Bayesian, neural networks, fuzzy neural networks,

and fuzzy connectives, among others. For selection based methods, only one classifier is needed to classify the input pattern correctly, for example, dynamic classifier selection, as suggested in (Kuncheva, 2004), is one of the main methods. Hybrid Methods combine both selection and fusion techniques to provide the most suitable output to classify the input pattern. The main idea is to use selection only, and only if the best classifier is good enough to classify the test pattern, otherwise, a combination method is used. Examples are dynamic classifier selection based on multiple classifier behavior and dynamic classifier selection based on decision templates (Kuncheva, 2004). Static combiners are independent of the feature vector. They are further subdivided into trainable and nontrainable. The trainable combiner undergoes individual training phase to increase the ensemble performance, e.g. weighted averaging, and stacked generalization. Non-trainable performs voting independently of the performance. Examples are: Borda count, averaging and voting. Adaptive means individual experts only need to perform well in their region of expertise and not on all inputs, e.g. a mixture of experts and hierarchical mixture of experts. Utility-based is the type that does not make use of prior knowledge or evidence to make decisions, e.g. simple averaging, voting techniques while decision-based are the ones that use previous evidence to make decision, for example, Dempster-Shafer theory of evidence (Shafer, 1976). Other researchers suggest that the performance of the ensembles depends on two properties, which are the individual success of the base classifiers of the ensemble and the independence of the base classifier's results from each other (Brown, Wyatt, & Tino, 2005). Another researcher suggests that the accuracy of individual models, diversity among the individual models, decision making strategy, and number of base classifiers used for constructing an ensemble (Makhtar, Yang, Neagu, & Ridley, 2012), (Wang, 2008) are among the factors responsible for the success of an ensemble. Ensemble methods had been widely applied in many fields such as web ranking algorithm, classification and clustering, time series and regression problems, and water quality application, among others (Baba, Makhtar, Fadzili, & Awang, 2015).

2.2.7 Ensemble Combination Techniques

Numerous methods have been suggested for the creation of ensemble of classifiers (Dietterich, 2000). Although or perhaps because many methods of ensemble creation have been proposed, there is as yet no clear picture of which method is best (Villada & Drissi, 2002). Thus, an active area of research in supervised learning is the study of methods for the construction of good ensembles of classifiers. Mechanisms that are used to build ensemble of classifiers include: using different subsets of training data with a single

learning method, using different training parameters with a single training method (e.g., using different initial weights for each neural network in an ensemble) and using different learning methods.

Bagging is a method for building ensembles that uses different subsets of training data with a single learning method (Breiman, 1996). Given a training set of size t , bagging draws t random instances from the dataset with replacement (i.e. using a uniform distribution). These t instances are learned, and this process is repeated several times. Since the draw is with replacement, usually the instances drawn will contain some duplicates and some omissions, as compared to the original training set. Each cycle through the process results in one classifier. After the construction of several classifiers, taking a vote of the predictions of each classifier produces the final prediction.

Bagging implementation algorithm (Bauer & Kohavi, 1999)

Input: I (an inducer), T (the number of iterations), S (the training set), N
(the subsample size)

Output: $C_t; t=1, \dots, T$

- 1: $t \leftarrow 1$
- 2: **repeat**
- 3: $S_t \leftarrow$ Samples N instances from S with replacement
- 4: Build classifier C_t using I on S_t
- 5: $t++$
- 6: **until** $t > T$

Inputs: Training data S ; supervised learning algorithm, **Base Classifier**, integer T

Specifying ensemble size; percent R to create bootstrapped training data.

Do $t = 1, \dots, T$

1. Take a bootstrapped replica S_t by randomly drawing $R\%$ of S .
2. Call **Base Classifier** with S_t and receive the hypothesis (classifier) h_t .
3. Add h_t to the ensemble, $\square \leftarrow \square \cup h_t$

End

Ensemble Combination: Simple Majority Voting —Given unlabeled instance x

1. Evaluate the ensemble $\square = \{h_1, \dots, h_T\}$ on x .
2. Let $v_{t,c} = 1$ if h_t chooses class ω_c , and 0, otherwise.

3. Obtain total vote received by each class

$$V_c = \sum_{t=1}^T V_{t,c}, c=1, \dots, C$$

Output: Class with the highest V_c

Breiman made the important observation that instability (responsiveness to changes in the training data) is a prerequisite for bagging to be effective. A committee of classifiers that all agree in all circumstances will give identical performance to any of its members in isolation (Breiman, 1996). A variance reduction process will have no effect if there is no variance. If there is too little data, the gains achieved via a bagged ensemble cannot compensate for the decrease in accuracy of individual models, each of which now considers an even smaller training set. On the other end, if the dataset is extremely large and computation time is not an issue, even a single flexible classifier can be quite adequate. Another method that uses different subsets of training data with a single learning method is the boosting approach (Freund & Schapire, 1997).

Boosting is similar in overall structure to bagging, except that it keeps track of the performance of the learning algorithm and concentrates on instances that have not been correctly learned. Instead of choosing the t training instances randomly using a uniform distribution, it chooses the training instances in such a manner as to favor the instances that have not been accurately learned. After several cycles, the prediction is performed by taking a weighted vote of the predictions of each classifier, with the weights being proportional to each classifier's accuracy on its training set. AdaBoost (short for Adaptive Boosting) (Freund & Schapire, 1997), and its several variations later extended the original boosting algorithm to multiple classes (AdaBoost.M1, AdaBost.M2), as well as to regression problems (AdaBoost.R). Here we describe the AdaBoost.M1, the most popular version of the AdaBoost algorithms.

AdaBoost. M1 implementation algorithm (Bauer & Kohavi, 1999)

Inputs: Training data = $\{x_i, y_i\}$, $I = 1, \dots, N$ $y_i \in \{\omega_1, \dots, \omega_c\}$, supervised

BaseClassifier; ensemble size T .

Initialize $D_1(i) = 1/N$.

Do for $t = 1, 2, \dots, T$:

1. Draw training subset S_t from the distribution D_t .
2. Train **Base Classifier** on S_t , receive hypothesis $h_t: X \rightarrow Y$
3. Calculate the error of h_t :

$$\epsilon_t = \sum_i I[h_t(x_i) \neq y_i] D_t(x_i)$$

If $\epsilon_t > 1/2$ **abort**

4. Set

$$\beta_t = \epsilon_t / (1 - \epsilon_t)$$

5. Update sampling distribution

$$D_{t+1}(i) = D_t(i) / Z_t \cdot \begin{cases} \beta_t, & \text{if } h_t(x_i) = y_i \\ 1, & \text{otherwise} \end{cases}$$

Where $Z_t = \sum_i D_t(i)$ is a normalization constant to ensure that D_{t+1} is a proper distribution function.

End

Weighted Majority Voting: Given unlabeled instance z ,

Obtain total vote received by each class

$$V_c = \sum_{t: h_t(z) = c} \log(1 / \beta_t), \quad c = 1, \dots, C$$

Output: Class with the highest V_c .

AdaBoost is a practical version of the boosting approach (Freund & Schapire, 1997). AdaBoost requires less instability than bagging, because AdaBoost can make much larger changes in the training set. A number of studies that compare AdaBoost and bagging suggest that AdaBoost and bagging have quite different operational profiles (Bauer & Kohavi, 1999); (Quinlan, 1996). In general, it appears that bagging is more consistent, increasing the error of the base learner less frequently than does AdaBoost. However, AdaBoost appears to have greater average effect, leading to substantially larger error reductions than bagging on average. Generally, bagging tends to decrease variance without unduly affecting bias (Breiman, 1996); (Bauer & Kohavi, 1999). On the contrary, in empirical studies AdaBoost appears to reduce both bias and variance ((Breiman, 1996); (Bauer & Kohavi, 1999)). Thus, AdaBoost is more effective at reducing bias than bagging, but bagging is more effective than AdaBoost at reducing variance. The decision on limiting the number of sub-classifiers is important for practical applications. To be competitive, it is important that the algorithms run in reasonable time. For both bagging and boosting, much of the reduction in error appears to have occurred after ten to fifteen classifiers. However, AdaBoost continues to measurably improve test-set

error until around 25 classifiers for decision trees (Optiz & Maclin, 1999). As mentioned in Bauer and Kohavi, the main problem with boosting seems to be robustness to noise. This is expected because noisy instances tend to be misclassified, and the weight will increase for these instances. They presented several cases where the performance of boosted algorithms degraded compared to the original algorithms. On the contrary, they pointed out that bagging improves the accuracy in all datasets used in the experimental evaluation (Bauer & Kohavi, 1999). MultiBoosting is another method of the same category. It can be conceptualized as wagging committees formed by AdaBoost (Webb, 2000). Wagging is a variant of bagging: bagging uses resampling to get the datasets for training and producing a weak hypothesis, whereas wagging uses reweighting for each training instance, pursuing the effect of bagging in a different way. Webb in a number of experiments, showed that MultiBoost achieved greater mean error reductions than any of AdaBoost or bagging decision trees in both committee sizes that were investigated (10 and 100). Another meta-learner, DECORATE (Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples), was presented. This method uses a learner (one that provides high accuracy on the training data) to build a diverse committee (Melville & Mooney, 2003). This is accomplished by adding different randomly constructed examples to the training set when building new committee members. These artificially constructed examples are given category labels that disagree with the current decision of the committee, thereby directly increasing diversity when a new classifier is trained on the augmented data and added to the committee.

Voting denotes the simplest method of combining predictions from multiple classifiers (Roli, Giacinto, & Vernazza, 2001). In its simplest form, called plurality or majority voting, each classification model contributes a single vote (Hall, Bowyer, Kegelmeyer, Moore, & Chao, 2000). The collective prediction is decided by the majority of the votes, i.e., the class with the most votes is the final prediction. In weighted voting, on the other hand, the classifiers have varying degrees of influence on the collective prediction that is relative to their predictive accuracy. Each classifier is associated with a specific weight determined by its performance (e.g., accuracy, cost model) on a validation set. The final prediction is decided by summing up all weighted votes and by choosing the class with the highest aggregate. Kotsiantis and Pintelas combined the advantages of classifier fusion and dynamic selection. The algorithms that are initially used to build the ensemble are tested on a small subset of the training set and, if they have statistically worse accuracy than the most accurate algorithm, do not participate in the final voting (Kotsiantis, Zaharakis, & Pintelas, 2004). Except for voting,

stacking aims to improve efficiency and scalability by executing a number of learning processes and combining the collective results (Ting & Witten, 1999). The main difference between voting and stacking is that the latter combines base classifiers in a non-linear fashion. The combining task, called a meta-learner, integrates the independently computed base classifiers into a higher-level classifier, a meta-classifier, by relearning the meta-level training set. This meta-level training set is created by using the base classifiers' predictions on the validation set as attribute values and the true class as the target. Ting and Witten have shown that successful stacked generalization requires the use of output class distributions rather than class predictions. In their experiments, only the MLR algorithm (a linear discriminant) was suitable for use as a level-1 classifier. Cascade Generalization (Gama & Brazdil, 2000) is another algorithm that belongs to the family of stacking algorithms. Cascade Generalization uses the set of classifiers sequentially, at each step performing an extension of the original data by the insertion of new attributes. The new attributes are derived from the probability class distribution given by a base classifier. This constructive step extends the representational language for the high-level classifiers, reducing their bias.

2.2.8 Review of Previous works

The previous related studies primarily involved research on various applications of machine learning in crime prevention and counterterrorism. Sachan and Roy (2012) in their study developed Terrorist Group Prediction Model (TGPM) to identify the responsible terrorist group by using historical data. TGPM uses the concept of Crime Prediction Model (CPM), Group Detection Model (GDM) and Offender Group Detection Model (OGDM). TGPM uses various parameters like attack type, location, target type, weapon type, hostage/kidnapping and suicide attack, terrorist corpus, parameter's value and parameters weight as input. After pre-processing of database, percentage of attacks of each group is calculated based on input parameters. Each parameter is assigned a weight based on its impact over the incident. The group weight is calculated by using the percentage of attacks of each group and the parameters weight. Different clusters are created. Association between these clusters is performed and highest value from these associations is obtained. Group name corresponding to the highest value may be the most probable responsible terrorist group. The model realized more 80% accuracy in identifying the terrorist group involved in an attack in India from the year 1998 to 2008. The study neither used classification nor hybrid of classifier algorithms. In a study, Rizwan, Masrah, Aida, Payam, & Nasim (2013), presented a comparison between two classification algorithms namely, Decision Tree and Naïve Bayesian for predicting the

'Crime Category' attribute, having labels, namely 'Low', 'Medium', and 'High'. For Decision Tree, the Accuracy, Precision and Recall were 83.95%, 83.5% and 84%. On the other hand, Accuracy, Precision and Recall values for Naïve Bayesian were 70.8124%, 66.4% and 70.8%, respectively. Experimental results for both the algorithms manifest that, Decision Tree performed better than the Naïve Bayesian for the crime dataset, using WEKA and 10-fold cross validation (Rizwan, Masrah, Aida, Payam, & Nasim, 2013). This experiment was performed using 10-fold cross-validation. This experiment was a comparisons of two individual classifier algorithms and not a hybrid combination.

Khorsid, Abou, & Soliman (2015), conducted a study in which A supervised standard, ensemble, and hybrid machine learning classification algorithms and models are compared in identification of the terrorist groups responsible of terrorist attacks in Middle East and North Africa from year 2009 up to 2013, by conducting different experiments, the data used in the experimental study was based on real data represented by Global terrorism Database (GTD) from National Consortium for the study of terrorism and Responses of Terrorism (START). To achieve the goal of this research; two different experiments are conducted on the used data, as well as using Litwise deletion approach to handle the missing data and provide a detailed comparative study of the used classification algorithms between 10 different classifiers which categorize into four main types namely, standard classification algorithms, hybrid classifiers, ensemble classifiers, and ensemble hybrid classifiers. The study used WEKA software and evaluated the obtained results via two different test options which are evaluation on training set, and 10- fold cross-validation during the experiments. The results from the first experiment conducted using the whole training data showed that KNN yielded accuracy of 100% and SVM yielded accuracy of 95% outperformed the other standard classifiers, FT hybrid yielded accuracy of 99% was more accurate and outperformed other hybrid classifier, In Ensemble Method(s); Random Forests (RFs) with accuracy of 99% was more accurate. In hybrid Ensemble classifiers, stacking classifier with accuracy of 20% performed badly and not accurate. The overall results showed that hybrid machine learning classifiers demonstrated good and proved obvious improvement in predictive accuracy over some standard comprehensible and ensemble methods. The results obtained from the second experiment which based on using 10-fold cross validation showed that: C45 67% and NB 63% standard classifier were almost good; FT with accuracy of 65% and DTNB with accuracy of 63% hybrid classifiers were good Machine Learning hybrid classifiers. And so the overall performance of the different types of classifiers used proved that hybrid machine

learning classifiers perform accurate and in some cases it could outperformed the single classifiers with some enhancement, but ensemble methods are more accurate and outperformed the hybrid ensemble methods in their prediction of terrorist groups' attacks results.

In a study by Tolan & Soliman(2015), a data mining classification ensemble approach was introduced in this paper research for the classification and prediction of the terrorist groups in Egypt from 1970 to 2013,the data used in the experimental study was based on real data represented by Global terrorism Database (GTD) from National Consortium for the study of terrorism and Responses of Terrorism (START). To achieve the goal of the research; two different approaches were implemented to handle the missing data namely; Mode-Imputation, and Litwise-Deletionas well as a detailed comparative study of the used classification algorithms by using WEKA software and results evaluated via the two different test options which are; evaluation on test split of the input data set into66% for the training data and 34% for the test set, and 10-fold cross-validation during the experiments. Five main classification algorithms were used in the study, those classification algorithms are: Naïve Bayes, K-Nearest Neighbour, Tree Induction C4.5, Iterative Dichotomiser, and Support Vector Machine. These classification algorithms were evaluated and compared according to four performance measures namely, classification accuracy, precision, recall and F-measure. The experiment conducted during the mode-imputation approach, in case of test split of the input data with splits 66%for training data, and 34% for testing data showed that SVM with accuracy of 71.83% was more accurate than other classifiers especially NB with accuracy of 69.01%, and KNN with accuracy of 72.54%,the overall performance of NB and KNN is almost the same.ID3 with accuracy of 21.13% has the lowest accuracy, but it performs well in other measures. In 10-fold cross validation case; KNN with accuracy of 73.03% classifier is near the SVM 75.42% accuracy, precision, and F-measure. ID3 26.01% accurate, NB classifier performs as KNN in most measures, and C4.5 performs badly than other classifiers in precision, recall, and f-measure. The experiment conducted during Litwisedeletion approach, in case of test split showed that KNN outperformed the other classifiers in its accuracy especially SVM that proved successful in the mode imputation approach. C4.5had lowest precision, recall, and F-measure results. KNN and SVM performed almost the same in precision, recall, and F-measure as they perform effectively in the first approach. In10- fold cross validation case; SVM was more accurate than other classifiers. KNN, and SVM are almost the same in their results, NB precision, recall, and F-measures are near KNN

classifier. C4.5 had the lowest precision, recall, and F-measure in contrast with ID3 which had highest results in precision, recall, and F-measure although it was not accurate.

Gundabathula & Vaidhehi (2018), conducted a study on efficient modeling of behavior of terrorist groups in India, the study presented machine learning models that can be employed on terrorism related data to identify the most accurate terrorist group responsible for an attack based on historic data. The study analyses terrorism challenges faced in India by modelling the behavior of terrorist groups using famous machine learning algorithms like J48, IBK, Naive Bayes and ensemble approach using majority voting combination method. The SMOTE (Synthetic Minority Over-Sampling Technique) algorithm is used to perform the oversampling and the hybrid sampling procedures as a solution to class imbalance problem. The results of the evaluation of the models show the accuracy percentage of various models employed and their relevance to the dataset. It was found that when the classification models are created on a data that has class imbalance problem the percentage of correctly classified instances will be very less with the following results for the classifier models: Decision Trees (J48) 66.13%, NB 64.64%, KNN(IBK) 55.71% and Ensemble 66.17%. The study established that sampling plays a main role in determining the accuracy percentage of classifier models and gives better results J48 98.79%, NB 88.06%, KNN 95.44% and Ensemble 88.70%. The study also shows the accuracy percentage of correctly classified instances for various algorithms and shows the ideal one for the dataset was DT.

2.2.9 Approaches to Classifier Algorithm Models Evaluation and Comparison

Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model (Refaeilzadeh, Tang, & Liu, 2007). In typical cross-validation, the training and validation sets must cross-over in successive rounds such that each data point has a chance of being validated against. The basic form of cross-validation is k-fold cross-validation. Other forms of cross-validation are special cases of k-fold cross-validation or involve repeated rounds of k-fold cross-validation (Refaeilzadeh, Tang, & Liu, 2007). In k-fold cross-validation the data is first partitioned into k equally (or nearly equally) sized segments or folds. Subsequently k iterations of training and validation are performed such that within each iteration a different fold of the data is held-out for validation while the remaining k-1 folds are used for learning. In data mining and machine learning 10-fold cross validation (k = 10) is the most common. Cross-validation is used to evaluate or compare

learning algorithms as follows: in each iteration, one or more learning algorithms use $k-1$ folds of data to learn one or more models, and subsequently the learned models are asked to make predictions about the data in the validation fold. The performance of each learning algorithm on each fold can be tracked using some predetermined performance metric like accuracy. Upon completion, k samples of the performance metric will be available for each algorithm. Different methodologies such as averaging can be used to obtain an aggregate measure from these sample, or these samples can be used in a statistical hypothesis test to show that one algorithm is superior to another(Refaeilzadeh, Tang, & Liu, 2007).In statistics or data mining, a typical task is to learn a model from available data. Such a model may be a regression model or a classifier. The problem with evaluating such a model is that it may demonstrate adequate prediction capability on the training data but might fail to predict future unseen data. Cross-validation is a procedure for estimating the generalization performance in this context(Refaeilzadeh, Tang, & Liu, 2007). The idea for cross-validation originated in the 1930s.Larson used one sample for regression and a second for prediction (Larson, 1931). Mosteller and Turkey and various other people further developed the idea (Mosteller & Turkey, 1968). A clear statement of cross-validation, which is similar to current version of k -fold cross-validation, first appeared in 1968 (Mosteller & Turkey, 1968). In 1970s, both Stone and Geisser employed cross-validation as means for choosing proper model parameters, as opposed to using cross-validation purely for estimating model performance(Stone, 1974), (Geisser, 1975).Currently, cross-validation is widely accepted in data mining and machine learning community and serves as a standard procedure for performance estimation and model selection(Refaeilzadeh, Tang, & Liu, 2007).There are two possible goals in cross-validation: To estimate performance of the learned model from available data using one algorithm. In other words, to gauge the generalizability of an algorithm and to compare the performance of two or more different algorithms and find out the best algorithm for the available data, or alternatively to compare the performance of two or more variants of a parameterized model. The above two goals are highly related, since the second goal is automatically achieved if one knows the accurate estimates of performance. Given a sample of N data instances and a learning algorithm A , the average cross-validated accuracy of A on these N instances may be taken as an estimate for the accuracy of A on unseen data when A is trained on all N instances. Alternatively if the end goal is to compare two learning algorithms, the performance samples obtained through cross-validation can be used to perform two-sample statistical hypothesis tests, comparing a pair of learning algorithms(Dietterich, 1998).

In re-substitution validation, the model is learned from all the available data and then tested on the same set of data. This validation process uses all the available data but suffers seriously from over-fitting. That is, the algorithm might perform well on the available data yet poorly on future unseen test data (Refaeilzadeh, Tang, & Liu, 2007). To avoid over-fitting, an independent test set is preferred. A natural approach is to split the available data into two non-overlapped parts: one for training and the other for testing. The test data is held out and not looked at during training. Hold-out validation avoids the overlap between training data and test data, yielding more accurate estimate for the generalization performance of the algorithm. The downside is that this procedure does not use all the available data and the results are highly dependent on the choice for the training/test split (Dietterich, 1998). The instances chosen for inclusion in the test set may be too easy or too difficult to classify and this can skew the results. Furthermore, the data in the test set may be valuable for training and if it is held out prediction performance may suffer, again leading to skewed results. These problems can be partially addressed by repeating hold-out validation multiple times and averaging the results, but unless this repetition is performed in a systematic manner, some data may be included in the test set multiple times while others are not included at all, or conversely some data may always fall in the test set and never get a chance to contribute to the learning phase. To deal with these challenges and utilize the available data to the maximum-fold cross-validation is used. In k-fold cross-validation the data is first partitioned into k equally (or nearly equally) sized segments or folds. Subsequently k iterations of training and validation are performed such that within each iteration a different fold of the data is held-out for validation while the remaining $k - 1$ folds are used for learning. Data is commonly stratified prior to being split into k folds. Stratification is the process of rearranging the data as to ensure each fold is a good representative of the whole (Kohavi, 1995). For example, in a binary classification problem where each class comprises 50% of the data, it is best to arrange the data such that in every fold, each class comprises around half the instances.

Leave-one-out cross-validation (LOOCV) is a special case of k-fold cross-validation where k equals the number of instances in the data. In other words, in each iteration nearly all the data except for a single observation are used for training and the model is tested on that single observation. An accuracy estimate obtained using LOOCV is known to be almost unbiased, but it has high variance, leading to unreliable estimates (Efron, 1983). It is still widely used when the available data are rare, especially in bioinformatics where only dozens of data

samples are available. However, this method has not been widely adopted in data mining field either and 10-fold cross-validation remains the most widely used validation procedure (Refaeilzadeh, Tang, & Liu, 2007). Two factors affect the performance measure: the training set, and the test set. The training set affects the measurement indirectly through the learning algorithm, whereas the composition of the test set has a direct impact on the performance measure (Dietterich, 1998). A reasonable experimental compromise may be to allow for overlapping training sets, while keeping the test sets independent. K-fold cross-validation does just that. Now the issue becomes selecting an appropriate value for k. A large k is seemingly desirable, since with a larger k there are more performance estimates, and the training set size is closer to the full data size, thus increasing the possibility that any conclusion made about the learning algorithm(s) under test will generalize to the case where all the data is used to train the learning model. As the value of k increases, however, the overlap between training sets also increases (Kohavi, 1995). For example, with 5-fold cross-validation, each training set shares only $\frac{3}{4}$ of its instances with each of the other four training sets whereas with 10-fold cross validation, each training set shares $\frac{8}{9}$ of its instances with each of the other nine training sets. Furthermore, increasing k shrinks the size of the test set, leading to less precise, less fine-grained measurements of the performance metric. For example, with a test set size of 10 instances, one can only measure accuracy to the nearest 10%, whereas with 20 instances the accuracy can be measured to the nearest 5%. These competing factors have all been considered and the general consensus in the data mining community seems to be that $k = 10$ is a good compromise (Refaeilzadeh, Tang, & Liu, 2007). This value of k is particularly attractive because it makes predictions using 90% of the data, making it more likely to be generalizable to the full data. Cross-validation can be applied in three contexts: performance estimation, model selection, and tuning learning model parameters (Refaeilzadeh, Tang, & Liu, 2007). Using 10-fold cross-validation one repeatedly uses 90% of the data to build a model and test its accuracy on the remaining 10%. The resulting average accuracy is likely somewhat of an underestimate for the true accuracy when the model is trained on all data and tested on unseen data, but in most cases this estimate is reliable, particularly if the amount of labeled data is sufficiently large and if the unseen data follows the same distribution as the labeled examples. Alternatively, cross-validation may be used to compare a pair of learning algorithms. This may be done in the case of newly developed learning algorithms, in which case the designer may wish to compare the performance of the classifier with some existing baseline classifier on some benchmark dataset, or it may be done in a generalized model-selection setting (Boukaert,

2003). In generalized model selection one has a large library of learning algorithms or classifiers to choose from and wish to select the model that will perform best for a particular dataset. There is no easy way of learning the best value for the soft margin parameter for a particular dataset other than trying it out and seeing how it works. In such cases, cross-validation can be performed on the training data as to measure the performance with each value being tested. Alternatively, a portion of the training set can be reserved for this purpose and not used in the rest of the learning process. But if the amount of labeled data is limited, this can significantly degrade the performance of the learned model and cross-validation may be the best option(Kohavi, 1995).Cross validation is a technique for assessing how the statistical analysis generalizes to an independent data set(Refaeilzadeh, Tang, & Liu, 2007).

2.3 Classifier Algorithm Performance Evaluation and Comparison Metrics

Usually, the problem of evaluating a new classifier is tackled by using the score that try to summarize the specific conditions of interest. Classification error and accuracy are widely used scores in the classification problems. In practice, classification error must be estimated from all the available samples. Thek-fold cross-validation, for example, is one of the most frequently used such estimation methods. Then, questions are whether such a new, proposed classifier (or enhancement of the existing one) yields an improved score over the competitor classifier (or classifiers) or the state of the art. It is almost impossible now to do any research work without an experimental section where the score of anew classifier is tested and compared with the scores of the existing ones. This last step also requires the selection of datasets on which the compared classifiers are learned and evaluated. The purpose of dataset selection step should not be to demonstrate classifiers superiority to another in all cases, but rather to identify its areas of strengths with respect to domain characteristics. The whole evaluation process of a classifier should include the following steps (Santafe, 2015):choosing an evaluation metric (i.e. a score) according to the properties of aclassifier, deciding the score estimation method to be used, checking whether the assumptions are fulfilled, running the evaluation method and interpret the results with respect to the domain, and compare a new classifier with the existing ones selected according to the different criteria, for example problem dependent; this step requires selection of datasets. Typical scores for measuring the performance of a classifier are accuracy and classification error, which for a two-class problem can be easily derived from a 2×2 confusion matrix as shown in the table below

These scores can be computed as:

$$\text{Accuracy} = (TP+TN)/(TP +FN+TN+FP).....2.4$$

$$\text{Error} = (1-\text{Acc}).....2.5$$

When both class labels are relevant and the proportion of data samples for each class is similar, these scores are a good choice. Unfortunately, equally class proportions are quite rare in real problems. This situation is known as the imbalance problem(Japkowicz & Stephen, 2002),(Sun, 2007).Empirical evidence shows that accuracy and error rate are biased with respect to data imbalance: the use of these scores might produce misleading conclusions since they do not take into account misclassification costs, the results are strongly biased to favor the majority class, and are sensitive to class skews. The comparison of the scores obtained by two or more classifiers in a set of problems is a central task in machine learning, so it is almost impossible to do any research work without an experimental section where the score of a new classifier is tested and compared with the scores of the existing ones. When the differences are clear (e.g., when the classifier is the best in all the problems considered), the direct comparison of the scores may be enough. But in most situations, a direct comparison may be misleading and not enough to draw sound conclusions. In such situations, the statistical assessment of the scores such as hypothesis testing is required. Statistical tests arise with the aim of giving answers to the above-mentioned questions, providing more precise assessments of the obtained scores by analyzing them to decide whether the observed differences between the classifiers are real or random. However, although the statistical tests have been established as a basic part of classifier comparison task, they are not a definitive tool, we have to be aware about their limitations and misuses. The statistical tests for comparing classifiers are usually bound to a specific estimation method of classifier score. Better differentiation of algorithms can be obtained by examining computational performance metrics such as build time and classification speed (Williams, Zander, & Armitage, 2006). Although training time varies according to the nature of the application task and dataset, specialists generally agree on a partial ordering of the major classes of learning algorithms. For instance, lazy learning methods require zero training time because the training instance is simply stored. NB methods also train very quickly since they require only a single pass on the data either to count frequencies (for discrete variables) or to compute the normal probability density function (for continuous variables under normality assumptions). Univariate DT are also fast, several orders of magnitude than neural networks and SVM (Kotsiantis, Zaharakis, & Pintea, 2004).

2.4 Data Mining Tools

There are various tools available that have been developed for various usage, examples are: Waikato Environment for Knowledge Analysis (WEKA), Rapid miner, Knostanz Information Miner (KNIME), Clementine, etc. They provide a set of methods and algorithms that help in better utilization of data information available to users; that is data analysis, cluster analysis, genetic algorithms, nearest neighbor, data visualization, regression analysis, decision trees, predictive analytics, text mining among others (Wahbeh, Al-Radaideh, Al-Kabi, & Al-Shawakfa, 2008)

2.4.1 Waikato Environment for Knowledge Analysis

It is an open source software that contains a collection of visualization tools and algorithms for data analysis and predictive modeling together with graphical user interface for easy access to this functionality. It supports several standard data mining tasks like data processing, clustering, classification, regression, visualization, and feature selection. WEKA through its workbench provides a collection of state-of-the-art machine learning algorithms and data pre-processing tools. It includes virtually all algorithms in data mining thus its diverse functionality characteristic, so one can quickly try out existing methods on new datasets in flexible ways. It also provides extensive support for the whole process of experimental data mining, including preparing the input data, evaluating learning schemes statistically and visualizing the input data and the results of learning. WEKA capabilities include API, database system support, visualization capabilities, PMML support and statistical analysis capabilities (Gundabathula & Vaidhehi, 2018).

2.4.2 Knostanz Information Miner

KNIME is an open source data analytics, reporting and integration platform, as it integrates various components for machine learning and data mining through its modular data pipelining concept. Mostly has been used in pharmaceutical research, customer data analysis, business intelligence and financial data (Tiwaria, Abhishek, Sekhar, & Arvind, 2007).Its capabilities includes; API, database system support, visualization, statistical analysis capabilities among others (Kavoc, 2012)

2.4.3 Rapid Miner

The capabilities are same as for WEKA and KNIME, but the variation comes in on rapid miner an advanced user will be able to achieve more functions compared to less advanced user (Gikaru, 2012)

2.4.4 Orange

It is similar with the other data mining tools mentioned above on functions that can be performed. Though for one to achieve full functionality additional add-ons, widgets have to be obtained and added to the program as it is a library of objects and routines written in C++. This may have some effect on the software's functionality and performance. It has no additional functionality that seems relevant for the end user, as it's quite basic in its performance and operations (Kavoc, 2012).

2.5 Data Mining Methodologies

As in data mining there are various methodologies and no standard one for applying. Thus, several vendors have created their own proprietary methodologies where the approaches are strongly correlated with the design of their own software packages and solutions. The popular methodologies include Sample Explore Modify Model and Assess (SEMMA) and Cross Industry Standard Process for Data Mining (CRISP-DM) as shown in fig. 2.6. SEMMA may contain essentials elements of data mining project that is statistical, modeling and data manipulation but it lacks some fundamental parts of any information systems project like analysis, design, and implementation phase. While CRISP-DM comprises of six (6) phases which are not rigid, and they include business understanding, data understanding, data preparation, modeling, evaluation and deployment much emphasis is on data which must be divided into training and Test/validation sets. But it is limiting as techniques are selected according to data available only and not on organization goals and requirements(Chawla, 2005).

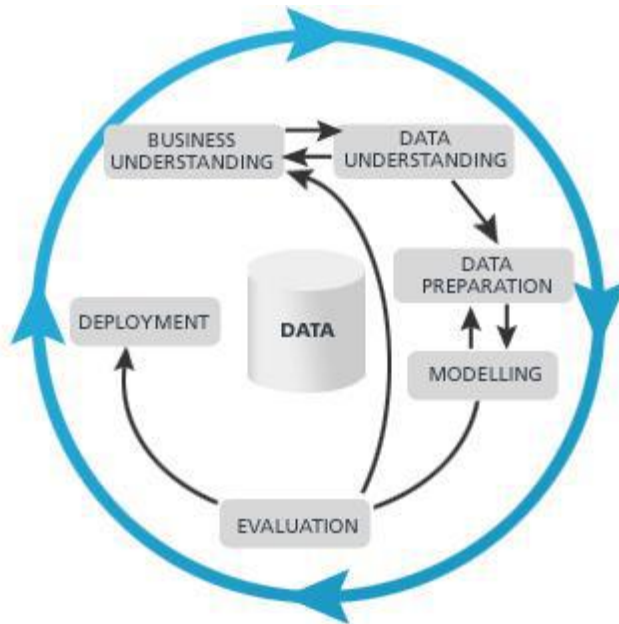


Figure 2.6: CRISP-DM(Rahim, 2014)

2.6 Feature Selection

Feature subset selection is the process of identifying and removing as many irrelevant and redundant features as possible (Yu & Liu, 2004). This reduces the dimensionality of the data and enables data mining algorithms to operate faster and more effectively. Feature selection methods can be classified in a number of ways. The most common one is the classification into filters, wrappers, embedded, and hybrid methods (Hoque, Bhattachryya, & Kalita, 2014). The abovementioned classification assumes feature independency or near independency. Additional methods have been devised for datasets with structured features where dependencies exist and for streaming features (Tang, Aleylani, & Liu, 2014). Filter methods select features based on a performance measure regardless of the employed data modeling algorithm. Only after the best features are found, the modeling algorithms can use them. Filter methods can rank individual features or evaluate entire feature subsets. We can roughly classify the developed measures for feature filtering into information, distance, consistency, similarity, and statistical measures. Wrapper methods consider feature subsets by the quality of the performance on a modelling algorithm, which is taken as a black box evaluator. Thus, for classification tasks, a wrapper will evaluate subsets based on the classifier performance (e.g. Naïve Bayes or SVM) (Bradley & Mangasarian, 1998),(Maldonado, Weber, & Famili, 2014), while for clustering, a wrapper will evaluate subsets based on the performance of a clustering algorithm (e.g. *K*-means)(Kim & Street, 2002). The evaluation is repeated for each subset, and the subset generation is dependent on the search strategy, in the same way as with

filters. Wrappers are much slower than filters in finding sufficiently good subsets because they depend on the resource demands of the modeling algorithm. The feature subsets are also biased towards the modelling algorithm on which they were evaluated (even when using cross-validation). Therefore, for a reliable generalization error estimate, it is necessary that both an independent validation sample and another modeling algorithm are used after the final subset is found. On the other hand, it has been empirically proven that wrappers obtain subsets with better performance than filters because the subsets are evaluated using a real modelling algorithm. Practically any combination of search strategy and modelling algorithm can be used as a wrapper, but wrappers are only feasible for greedy search strategies and fast modelling algorithms such as Naïve Bayes (Cortizo & Giraldez, 2006), linear SVM(Liu, Tiang, & Yang, 2014), and Extreme Learning Machines (Benoit, Van Heeswijk, Miche, Verleysan, & Lendasse, 2013).Embedded and hybrid methods perform feature selection during the modelling algorithm's execution. These methods are thus embedded in the algorithm either as its normal or extended functionality.

2.7Class Imbalance Problem

There can be an imbalance dataset provided for classification. Imbalance dataset means that one of the two classes has very a smaller number of samples compared to number of samples in the other class, $|C_2| \ll |C_1|$.Then C_2 is called the minority class, and C_1 is called the majority class. The minority class is of our interest. The machine learning algorithm always performs well if it is given balanced dataset, but this is not always the case , as an example the dataset for fraud detection ,will have a smaller number of fraud transactions than genuine transaction. Anomaly detection, medical diagnostic and fault monitoring are other examples. The prediction in case of unbalanced dataset is biased towards majority class. The approach to solve this problem is sampling based approach (Verma, 2019). Sampling based approach also known as data level approach works by artificially balancing the instances of class in the dataset. To artificially balance the class we apply resampling technique, such as random under sampling the majority class, random oversampling of minority class, and Synthetic Minority Over-Sampling Technique (SMOTE)(Verma, 2019). Random under sampling of majority class balances the class distribution in the dataset by randomly throwing away some data samples from majority class. Although it balances class distribution, but it leads to losing some important characteristics in dataset, due to removal of some samples, this is a disadvantage of this approach (Verma, 2019). Random oversampling of minority class balances the class distribution by the random replication of minority class instances, to

increase their number. There is no information loss in this case. The problem with this approach is that it leads to overfitting (Verma, 2019). Synthetic minority oversampling technique (SMOTE) reduces the problem of overfitting a method of by creating synthetic instances of minority class. This technique is known as the synthetic minority over-sampling technique (SMOTE). In this the training set is altered by adding synthetically generated minority class instances, causing the class distribution to become more balanced. The instances are said to be synthetic, as they are new minority instances that has being created out of existing minority class instances. In order to create the new synthetic minority class instances, SMOTE first selects an instance of minority class at random say 'x' and proceeds by finding its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors say 'y' at random and connecting 'x' and 'y' to form a line segment in the feature space. The synthetic instance say 'z' is generated as a convex combination of the two chosen instances 'x' and 'y'. $z.attribute = x.attribute + (y.attribute - x.attribute) * rand(0,1)$ (Verma, 2019).

2.8 Chapter Summary and Gap

From the literature review it is established that various data mining techniques such as clustering, asocial rule mining, social network analysis, and classification have been used to model frameworks and implement various models in identification of terrorist groups. However, the studies have not sufficiently evaluated the effect of resampling for class imbalance on various hybrid bagging combinations of KNN, NB, DT, SVM, and MLP using 10-fold cross validation test option and test split in WEKA data mining software environment. This would provide guidance for building optimally performing hybrid classifier algorithm model for identification of terrorist groups.

CHAPTER THREE

RESEARCH METHODOLOGY

This chapter consists of research design, data mining methods and tools, terrorism dataset and collection methodology, data pre-processing, resampled dataset instances, building and evaluating base classifier algorithm models, building, and evaluating hybrid classifier algorithm models and the architecture of the hybrid model.

3.1 Research Design

The study adopted an experimental research design in form of a randomized block design. The term “experimental research design” is centrally concerned with constructing research that is high in causal (or internal) validity. Causal validity concerns the accuracy of statements regarding cause and effect relationships

(Alexandrie, 2017). Experimental research is a study in which participants are randomly assigned to groups that undergo various researcher-imposed treatments or interviews, followed by observations or measurements to assess the effects of the treatments. The noteworthy key to an experiment is the researcher’s complete control over the research that enables him or her to randomize the study participants in order to provide better assessment of the treatments provided (Leedy & Ormrod, 2010). The experiment adopted a randomized block design. A block can contain a single participant who is observed under all p treatment levels or p participants (experimental units) who are similar with respect to a variable that is positively correlated with the dependent variable. If each block contains one participant, the order in which the treatment levels are administered is randomized independently for each block, assuming that the nature of the treatment permit this. If a block contains p matched participants, the participants in each block are randomly assigned to the treatment levels. The use of repeated measures or matched participants does not affect the statistical analysis. To be able to average out the effect of randomness and hence arrive at conclusions deemed statistically significant, the training and validation is done multiple times randomly (randomization), run the algorithms many times (replication) and compare the distributions of results rather than single values (Irsoy, Yildiz, & Alpaydin, 2012). All the algorithms used the same training and validation splits to ensure that any difference is due to the algorithm and not due to the split of the data; that is the idea behind paired tests (blocking). Stratification is also required, that is, the proportion of positive to negative

instances is respected in all parts so that the prior class probabilities do not change between fold. Two null hypotheses were tested.

$H_0 : \mu_{.100} = \mu_{.200} = \mu_{.300} \dots \mu_{.1300}$ (treatment population means are equal)

$H_1 : \mu_{100} \neq \mu_{200} \neq \dots \neq \mu_{1300}$

$H_0 : \mu_1 = \mu_2 = \dots = \mu_9$ (block population means are equal)

$H_1 : \mu_1 \neq \mu_2 \neq \dots \neq \mu_9$

Table 3.1: Randomized block experiment design with experimental units, control experiment and treatment levels

		Experimental treatment levels with varying resample sample size percent												
		Control experiment with none resampled dataset												
		100	200	300	400	500	600	700	800	900	1000	1100	1200	1300
Experimental units of different classifier algorithms	DT													
	KNN													
	NB													
	SVM													
	MLP													
	KNMS													
	D													
	KNSD													
	KND													
KD														
		Y^{-}	Y^{-1}	Y^{-2}	Y^{-3}	Y^{-4}				Y^{-13}
		Treatment means												

Layout for a randomized block design with $p = 14$ treatment levels and $n = 9$ blocks (experimental units). In the experiment, the p participants in each experimental unit/block were randomly assigned to the treatment levels. The means of treatment are denoted by Y^{-1} , Y^{-2} , and $Y^{-3} \dots Y^{-13}$. The means of the experimental units are denoted by x^1, \dots, x^9 .

The experiment was set up to examine whether resample sample size percent has an effect on the classification accuracy of machine learning classifier algorithms (base and combination of the bases). The dependent variable is the accuracy and the independent variable is the sample size percent, which consists of fourteen (14) treatment levels (representing 14 datasets). The experiment was set up for 9 experimental units (9 classifier algorithms). To test whether sample size percent has an effect on accuracy, we built classifier algorithms which were evaluated on 14 datasets. For consistency, the evaluation environment was maintained the same for all the datasets. In a random order, each group (classifier algorithm) was evaluated on datasets(treatments) as follows: none resampled dataset (as the control experiment/baseline), dataset resample sample size percent as follows at 100, dataset resampled at 200, dataset resampled at 300,dataset resampled at 400,dataset resampled at 500, dataset resampled at 600,dataset resampled at 700, dataset resampled at 800, dataset resampled at 900,dataset resampled at 1000, dataset resampled at 1100, dataset resampled at 1200, dataset resampled at 1300. The following classifier algorithms were used Decision Trees(DT), K-Nearest Neighbor(KNN), Naïve Bayes(NB), Support Vector Machine(SVM), Multi-layer perceptron (MLP),hybrid KNMSD (combination of KNN, NB,MLP,SVM,DT), Hybrid KNSD (combination of KNN, NB,SVM,DT), hybrid KND (combination of KNN, NB,DT), and hybrid KD (combination of KNN, DT).The hybrid machine learning classifier algorithms for the control experiments are also used to check on the effect of hybrid classifier algorithm components on performance. The experiments were setup on Lenovo Ideapad 320 machine running Intel core i5, 1 terabyte HDD, 4GB RAM, Windows 10 pro and WEKA 3.8.3 explorer environment.

3.2 Data Mining Methodology and Tools

The data mining framework methodology followed in this study is the Cross-Industry Standard Process for Data Mining (CRISP-DM), a non-proprietary hierarchical process model designed by practitioners from different domains. It has proven to be the most commonly preferred framework (Piatestsky, 2014).The framework breaks down the data mining process into six phases: understanding the business process and determining the ultimate data mining goals; identifying, collecting, and understanding key data sources; preparing data for data mining; selecting modelling techniques to use; evaluating and comparing results of different models against the initial goals and deploying the model. It also emphasizes on data which must be divided into training and test/validation sets(Chawla, 2005). WEKA is used as a data mining tool because of its support for experimental data

mining tasks and virtually all algorithms (Witten, Frank, & Hall, 2011). This enables one to quickly try out existing methods on new datasets in flexible ways. It also provides extensive support for the whole process of experimental data mining, including preparing the input data, evaluating learning schemes statistically and visualizing the input data and the results of learning.

3.3 Terrorism Dataset and Collection Methodology

The GTD data set is an open source, most comprehensive and world's largest dataset available on terrorism incidents used for the experiment, taken from an open source of the National Consortium for the study of terrorism and Responses of Terrorism (START) initiative at University of Maryland USA, which broadcasts the terrorism incidents report about the globe from 1970 to 2017, and includes information about more than 87,000 terrorist events as well as the vast information on 134 variables, and contain information over than 13,000 eliminations, 38,000 bombing and 4,000 kidnappings (START, 2018). For the hybrid classifier modeling, Global Terrorism Database (GTD) was used as the data set source. The data base is obtained from the National Consortium for the Study of Terrorism and Responses to Terrorism (START) initiative at University of Maryland, from their online interface at <http://www.start.umd.edu/gtd/> as an open source database. The researcher was allowed access to the database vide authorization as shown in appendix D. The study also complied with the national regulatory requirements by seeking approval from National Commission for Science, technology, and Innovation (NACOSTI) and Maseno University Ethical Review Committee (MUERC) refer to appendices C and B respectively

3.4 Data Pre-processing

The dataset for the study was 1999-2017 dataset for sub-Saharan Africa region from the GTD. The data set used for our study consists of a total of 15,022 terrorist events (instances), and 23 attributes, the attribute group consists of 25 diverse terrorist groups. Before applying classification algorithm(s) usually some pre-processing is performed on the data set. In order to perform data processing, it is essential to improve the data (Han & Kamber, 2006). There are a few number of techniques used for the purpose of data pre-processing as data aggregation, data sampling, dimension reduction, feature creation, data discretization, variable transformation, and dealing with missing values (Hongbo, 2010).

The attribute Group name was identified as the class label. All the instances with unknown class values were removed. Class label that had less than three frequency of occurrence was

also eliminated. This meant that the instances in which a terrorist group had participated in an incident less than three times for the whole time period of 1999-2017 had been removed. The column Group name which indicates the perpetrator group responsible for an incident consisted of apostrophes which were not accepted by WEKA for constructing the classification models. Therefore, these apostrophes from the group names were replaced by spaces and duplicate instances were removed. In this study the WEKA remove duplicates filter was used to remove duplicate instances. Feature selection/ dimension reduction was addressed through data approach method by use of WEKA supervised filter method. Feature selection refers to retaining only those features that are “meaningful” or relevant in building a good classifier. It is based on domain knowledge and careful exploratory analyses. In the filter approach, features are selected before running a machine learning algorithm. The main reason of feature selection is to remove noise, increase computational efficiency and to avoid over fitting. WEKA provides a supervised attribute filter allows various search and evaluation methods to be combined. The attribute evaluator is the technique by which each attribute in the dataset (also called a column or feature) is evaluated in the context of the output variable (e.g. the class). The search method is the technique by which to try or navigate different combinations of attributes in the dataset in order to arrive on a short list of chosen features (Patil & Sane, 2014). Filter techniques examine the significance of features by investigating the real characteristics of the data. In most cases feature rank is calculated, and low-ranking features are ignored during the learning process. Afterwards, the high-ranking subset of features is used as training set to the classification algorithm (Kiage, 2015). The 23 features in the dataset were subjected to the inbuilt WEKA attribute evaluator and search methods after which six attributes were ranked more relevant in the context of the output variable (class label). Data obtained after reduction was both in numerical as well as textual nominal, only nominal numbers were considered. Attributes selected for the identification of terrorist groups are listed below:

- i. Country (This attribute represents the country or location where the incident has happened)
- ii. Region (It is a categorical variable and it represents the region in which the incident occurred)
- iii. Attack type (It is a categorical variable and shows which kind of attack types executed e.g. assassination, bombing, kidnapping etc. there are total of 9 kinds of attack types recorded)

- iv. Target type (This attribute represents the target category)
- v. Group name (This attribute represents the group that is responsible for attack)
- vi. Weapon type (This attribute shows the type of weapon used in an attack)

The major pre-processing to be done was to solve the class imbalance problem in the data. Class imbalance problem is a situation in which the observations that belong to one class are significantly lower when compared to the observations of other classes. This causes biasness towards the classes with higher observations. In the final dataset WEKA resampling filter was applied to solve the class imbalance problem. This reduces class skew before applying a base classifier. It also ensures that the sub-sample is stratified so that the original class distribution is preserved in the sub-sample. WEKA Resampling filter produces a random sub-sample of dataset using either sampling with replacement or without replacement for imbalanced dataset, to achieve oversampling of the minority class, rather than under sampling of the majority class, so that both classes have the same numbers of instances(Gundabathula & Vaidhehi, 2018).The pre-processed dataset was converted to. ARFF to be used by WEKA.

3.5 The Flow of Building the Hybrid Classifier Model for Identification of Terrorist groups

Building the hybrid classifier model is in two phases. The first phase is the building of base classifiers. The second phase is the integration and evaluation of different combination of selected base classifiers through bagging majority voting ensemble technique with the aim of yielding optimum accuracy rates for the available dataset. Bagging combines them by majority voting and the most voted class is predicted. Majority voting is a bagging technique recommended for use if the output consists of class labels(Breiman, 2001). The flow of the process of building the hybrid classifier algorithm model is shown in fig. 3.1.

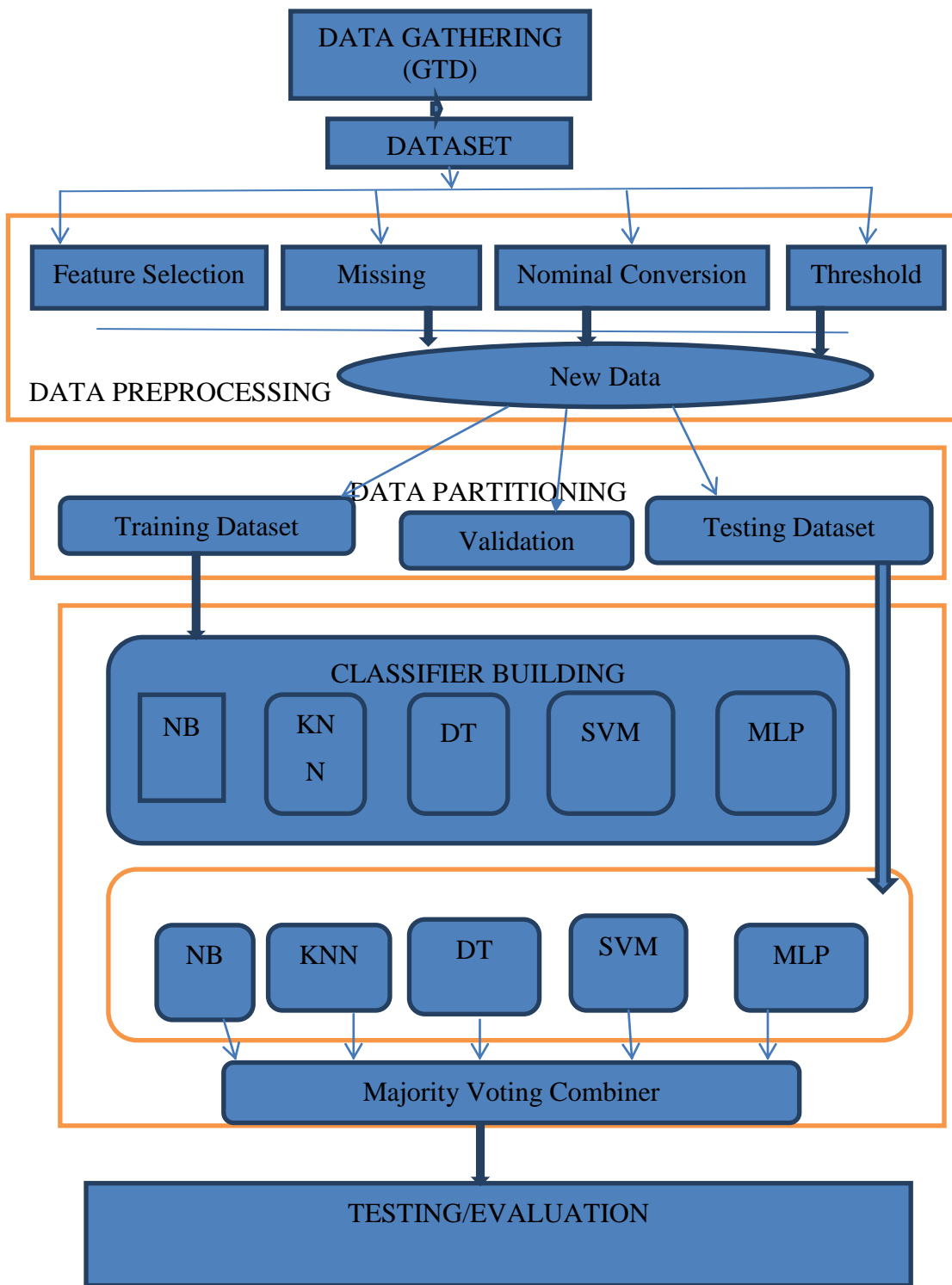


Figure 3.1: The hybrid classifier algorithm model process flow

3.5.1 Build and Evaluate Base Classifier Algorithm Models for Identification of Terrorist Groups

The classification models were built for the control and resampled dataset using these machine learning classification algorithms Decision Trees, Naive Bayes, K-Nearest Neighbour, Support Vector Machines and Multi-Layer Perceptron and evaluated using 10-fold cross validation and Test split test.

In 10-fold cross validation method data set is divided randomly into 10 parts. 9 of those parts are used for training and reserved one tenth for testing. The procedure is repeated 10 times each time reserving a different tenth for testing. The aim is to overcome the problem of overfitting and make prediction more general. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once (Krogh & Vedelshy, 1995). 10-fold cross validation makes predictions using 90% of data, making it more likely to generalize to the full dataset (Refaeilzadeh, Tang, & Liu, 2007). In test split test option data set is split into 66% train set for training the classifier and 33% test for testing the classifier. The experiments were conducted as shown below.

J48 algorithm belongs to the Decision trees family which is used to generate decision trees. Naive Bayes is a probabilistic classifier which belongs to the Bayes family. IBK is a lazy learning algorithm which is the implementation of K-nearest neighbor's algorithm, SMO and MLP are function algorithms for the implementation of SVM and MLP, respectively. 10 experiments were set up, two for each different machine learning classifier model as follows:

- i. Naive Bayes calculates the posterior probability for each class and makes a prediction for the class with the highest probability. As such, it supports both binary classification and multi-class classification problems. A Naïve Bayes classifier model was built with WEKA configurations as shown in fig. 3.2. The resulting classifier model was evaluated using 10-fold cross validation and test split (66%/34%) options. The rate of classification accuracy and build time tabulated. The process was repeated for all resampled datasets.

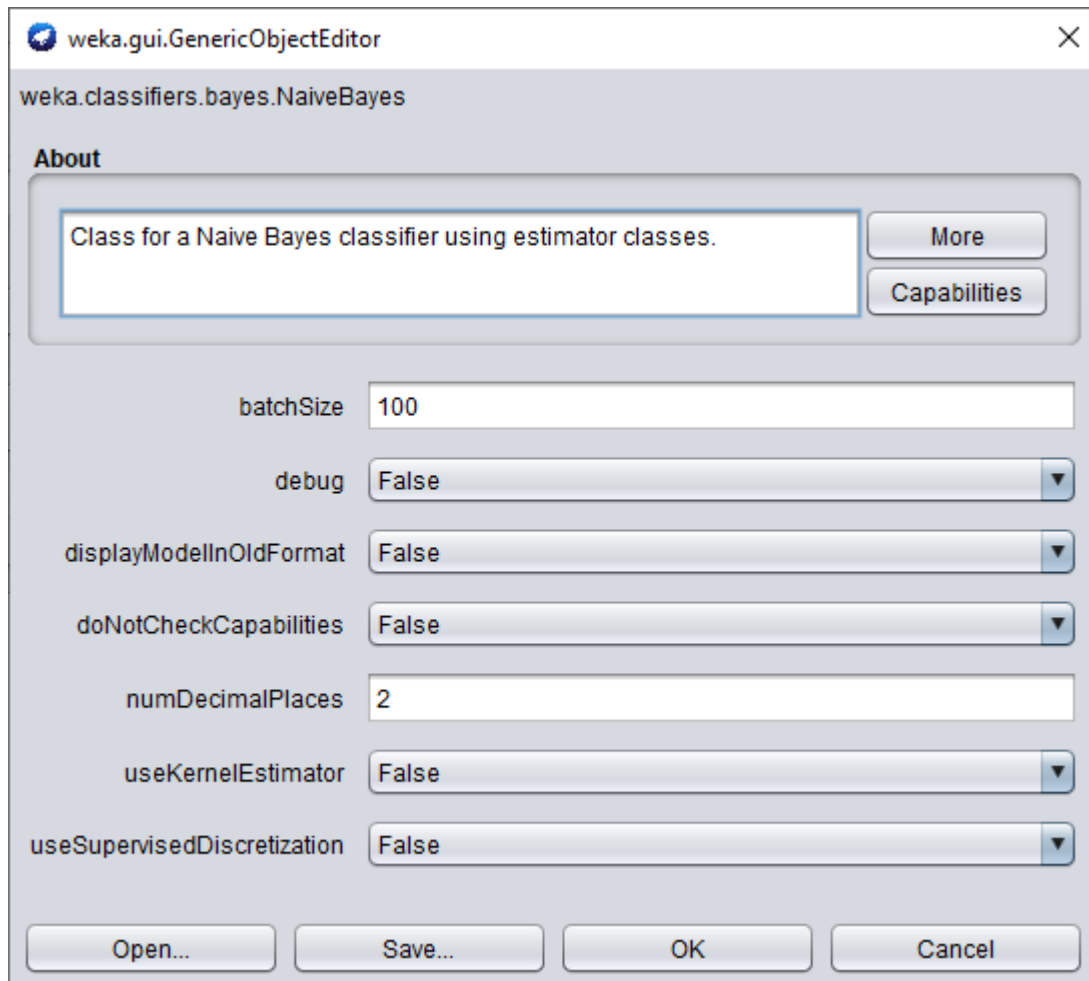


Figure 3.2: NB Configuration setup(author)

- ii. K-Nearest Neighbor classifier model was developed. For this purpose, pre-processed dataset was loaded and split for testing and training purpose and a model was created using training dataset on IBK model. The model was then evaluated using 10- fold cross validation and test split options. Accuracy build time and classification error tabulated. K-Nearest Neighbor classifier is a simple algorithm, but one that does not assume very much about the problem other than that the distance between data instances is meaningful in making predictions. As such, it often achieves particularly good performance. When making predictions on classification problems, KNN will take the mode (most common class) of the k most similar instances in the training dataset. IBK (instanced based learner k) is the WEKA implementation of K-nearest neighbor's algorithm. An IBK classifier model was built with WEKA configurations as shown in fig. 3.3. The resulting classifier model was evaluated using 10- fold cross

validation and test split (66%/34%) options. The rate of classification accuracy and build time tabulated. The process was repeated for all resampled datasets.

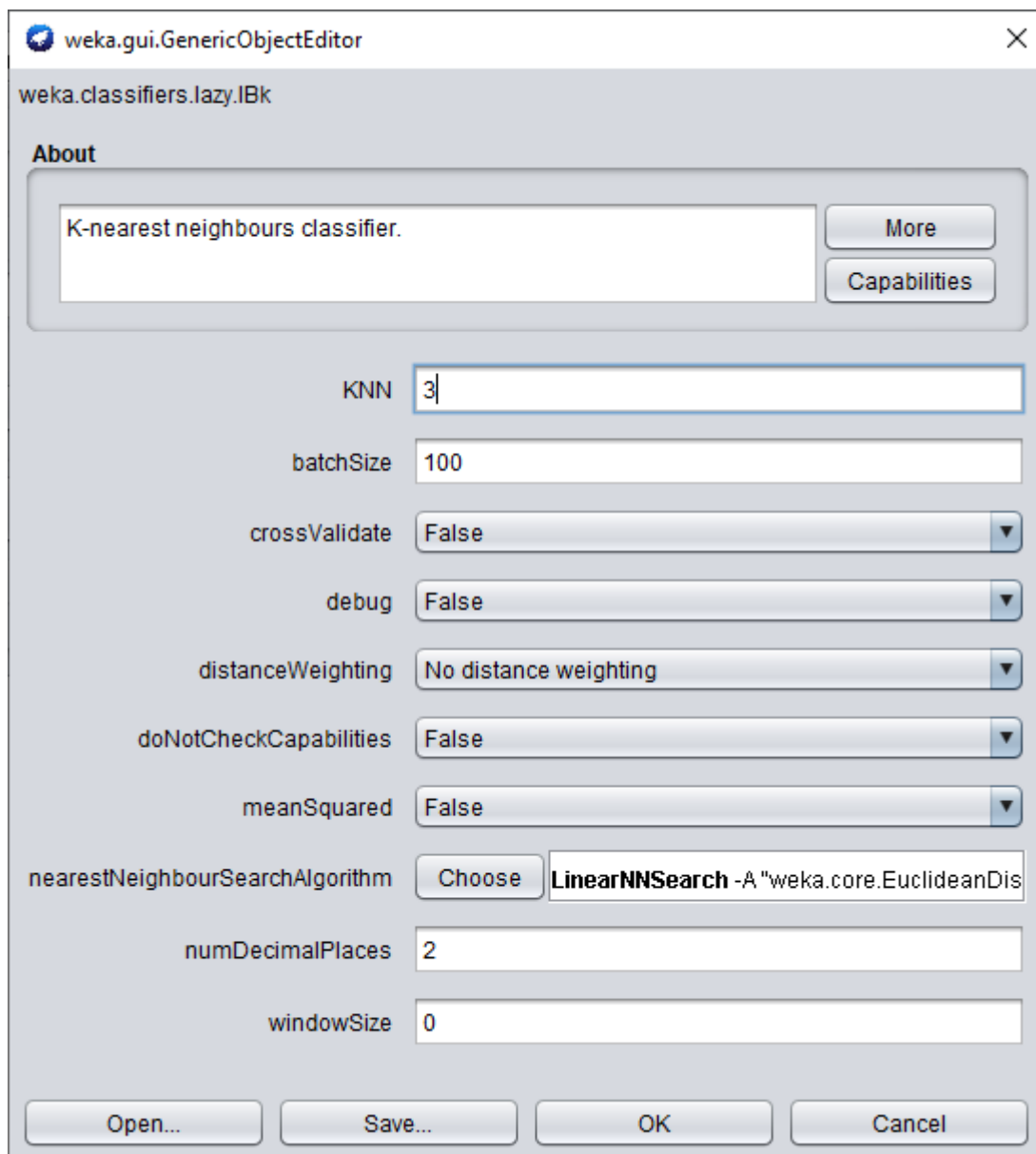


Figure 3.3: KNN configuration setup (author)

- iii. Decision Trees work by creating a tree to evaluate an instance of data, start at the root of the tree and moving town to the leaves (roots) until a prediction can be made. The process of creating a decision tree works by greedily selecting the best split point in order to make predictions and repeating the process until the tree is a fixed depth. After the tree is constructed, it is pruned in order to improve the model's ability to generalize to new data. A C4.5 (J48) is an algorithm used to generate a decision tree. It is the WEKA implementation of decision trees. J48 classifier model was built with

WEKA configurations as shown in fig. 3.4. The resulting classifier model was evaluated using 10- fold cross validation and test split(66%/34%) options. The rate of classification accuracy and build time tabulated. The process was repeated for all resampled datasets.

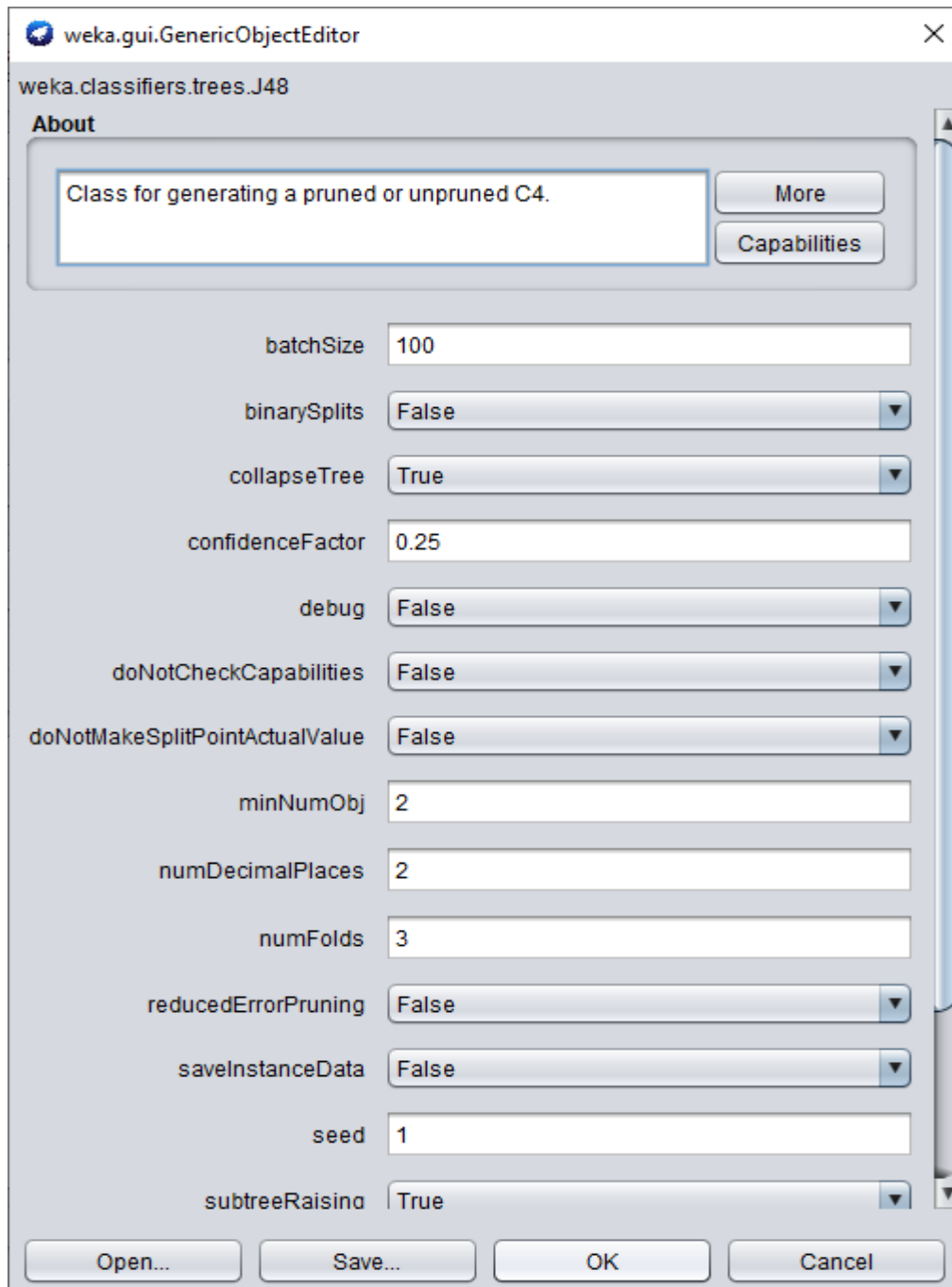


Figure 3.4: DT (J48) configuration setup (author)

iv. SVM works by finding a line that best separates the data into the two groups. This is done using an optimization process that only considers those data instances in the training dataset that are closest to the line that best separates the classes. The instances are called support vectors, hence the name of the technique. In almost all problems of interest, a line cannot be drawn to neatly separate the classes, therefore a margin is added around the line to relax the constraint, allowing some instances to be misclassified but allowing a better result overall. Finally, few datasets can be separated with just a straight line. Sometimes a line with curves or even polygonal regions need to be marked out. This is achieved with SVM by projecting the data into a higher dimensional space in order to draw the lines and make predictions. Different kernels can be used to control the projection and the amount of flexibility in separating the classes. SMO refers to the specific efficient optimization algorithm used inside the SVM implementation, which stands for Sequential Minimal Optimization. It is the WEKA implementation of support vector machines). SMO classifier model was built with WEKA configurations as shown in fig.3.5. The resulting classifier model was evaluated using 10- fold cross validation and test split (66%/34%) options. The rate of classification accuracy and build time tabulated. The process was repeated for all resampled datasets.

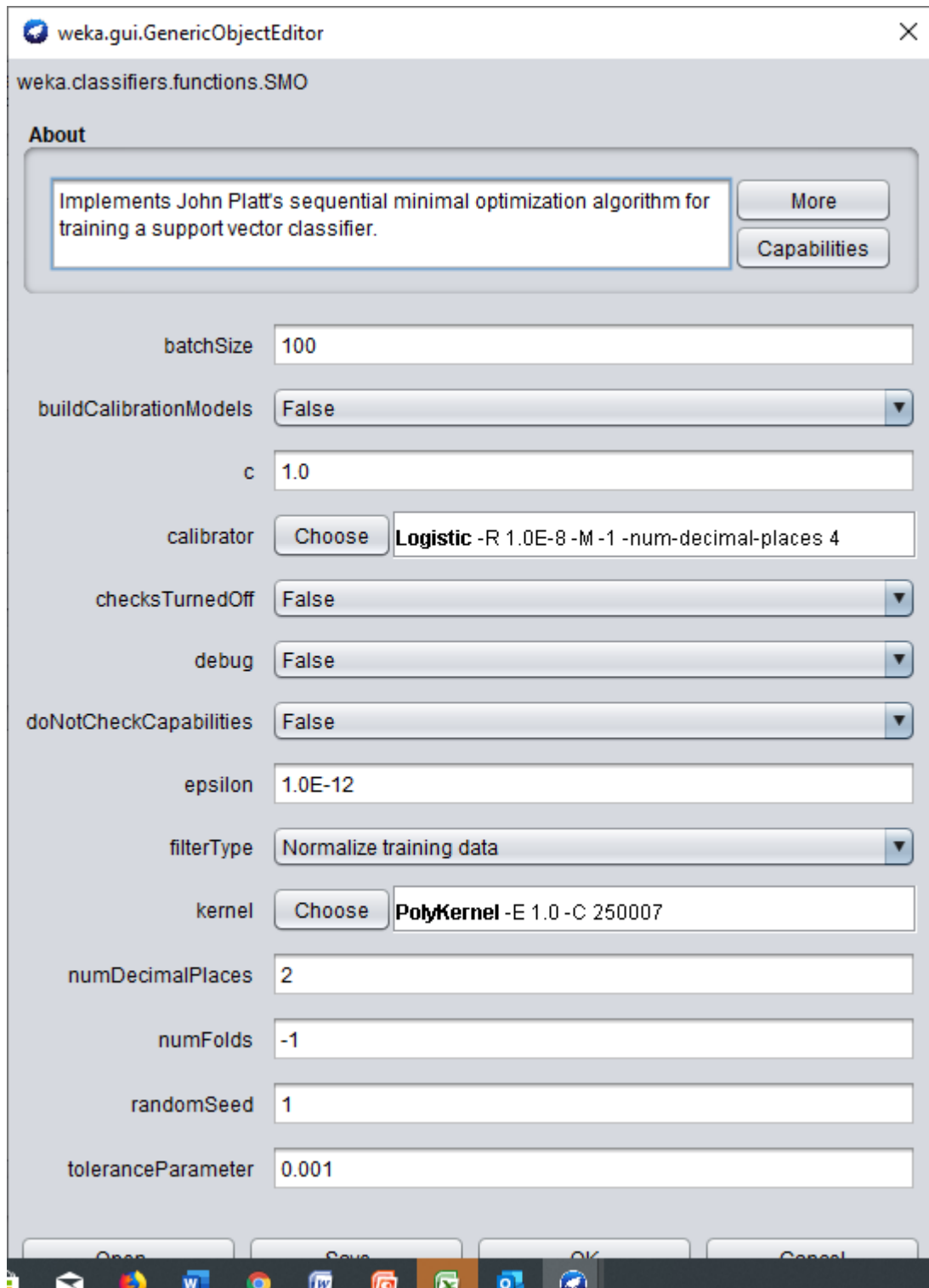


Figure 3.5: SVM(SMO) configuration setup (author)

- v. A multilayer perceptron (MLP) is a class of feed forward artificial neural network. MLP utilizes a supervised learning called back propagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable. SMO classifier model was built with

WEKA configurations as shown in fig.3.6. The resulting classifier model was evaluated using 10- fold cross validation and test split (66%/34%) options. The rate of classification accuracy and build time tabulated. The process was repeated for all resampled datasets

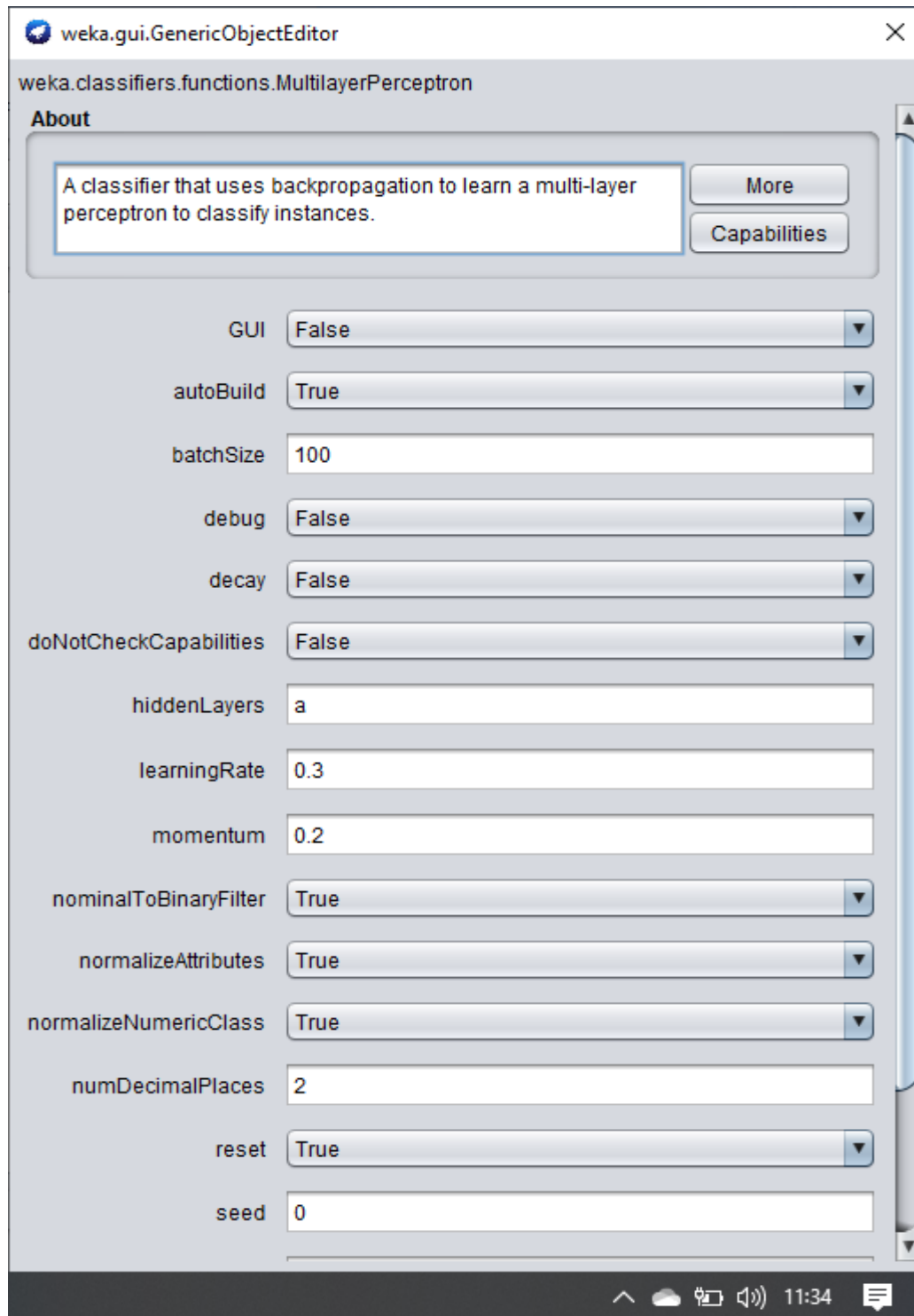


Figure 3.6: MLP(back propagation) configuration setup

3.5.2 Build and Evaluate Hybrid Classifier algorithm models for Identification of Terrorist Groups

The hybrid classifier algorithm models were built for the control and resampled dataset by combining different machine learning classification algorithms Decision Trees, Naive Bayes, K-Nearest Neighbour, Support Vector Machines and Multi-Layer Perceptron and evaluated using 10-fold cross validation and Test split test.

In 10-fold cross validation method data set is divided randomly into 10 parts. 9 of those parts are used for training and reserved one tenth for testing. The procedure is repeated 10 times each time reserving a different tenth for testing. The aim is to overcome the problem of overfitting and make prediction more general. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once (Krogh & Vedelshy, 1995). 10-fold cross validation makes predictions using 90% of data, making it more likely to generalize to the full dataset (Refaeilzadeh, Tang, & Liu, 2007). In test split test option data set is split into 66% train set for training the classifier and 33% test for testing the classifier. The experiments were conducted as shown below.

Experiments were set up for various optimum combinations of base classifiers based on the classifier model accuracy and build time as follows:

- i. KNMSD hybrid classifier was built by combining concurrently through bagging all the developed base classifiers: KNN, NB, MLP, SVM and DT. In the concurrent ensemble methodology, the original dataset was partitioned into several subsets from which multiple classifiers were induced concurrently. A majority voting combining procedure was then applied in order to produce a single classification for a given instance. The configuration for combination of the base classifiers is shown in fig.3.7. This model was then evaluated using 10-fold cross validation test option and test split test option. Classification accuracy and build tabulated. The process was repeated for all the datasets.

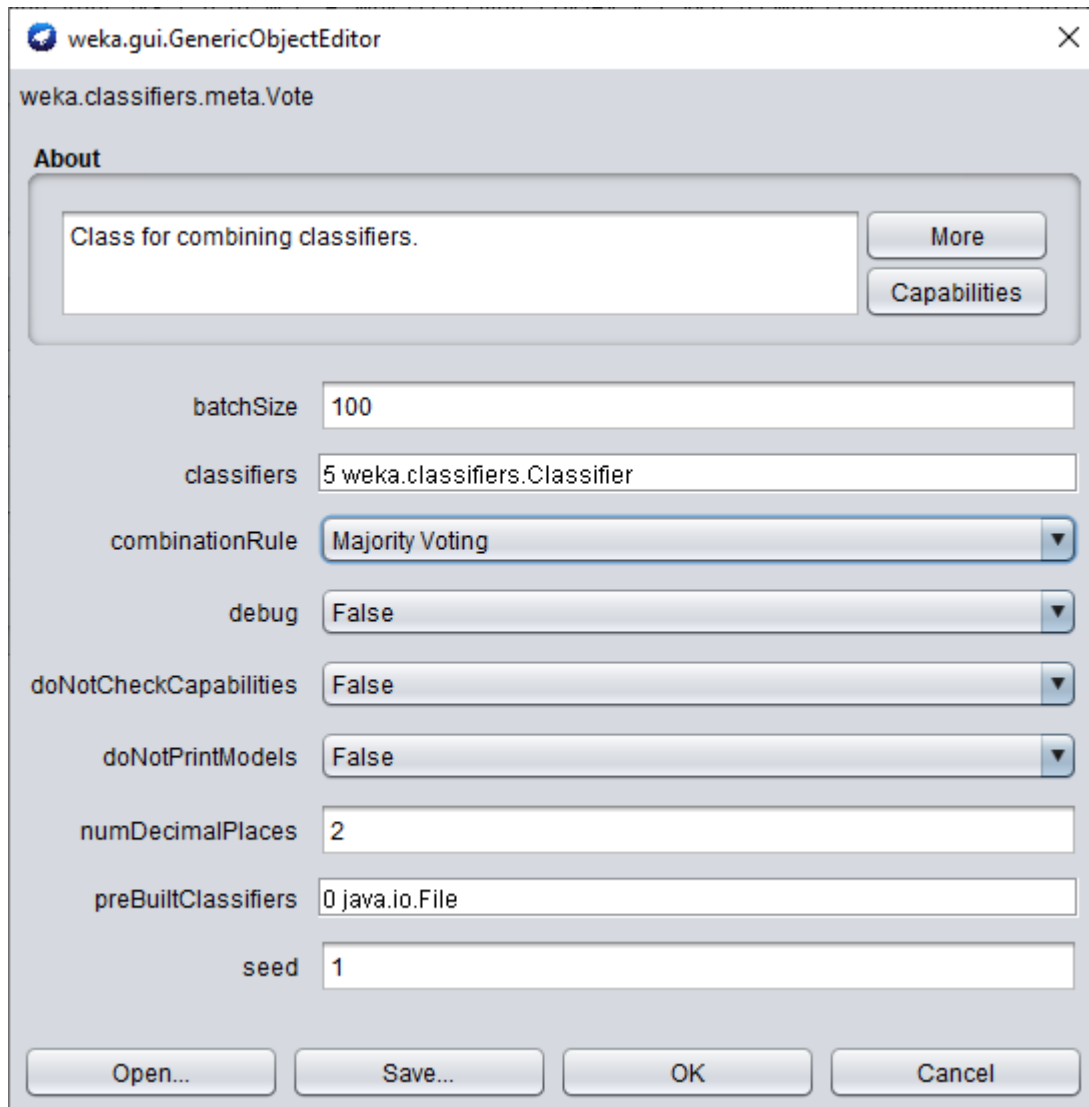


Figure 3.7: KNMSD configuration setup(author)

- ii. KNSD hybrid classifier was built by combining concurrently through bagging four base classifiers KNN, NB, SVM and DT which outperformed MLP at the identified resample rate for optimum performance. In the concurrent ensemble methodology, the original dataset was partitioned into several subsets from which multiple classifiers were induced concurrently. A majority voting combining procedure was then applied in order to produce a single classification for a given instance. The configuration for combination of the base classifiers is shown in fig. 3.8. This model was then evaluated using 10-fold cross validation test option and test split test option. Classification accuracy and build tabulated. The process was repeated for all the datasets

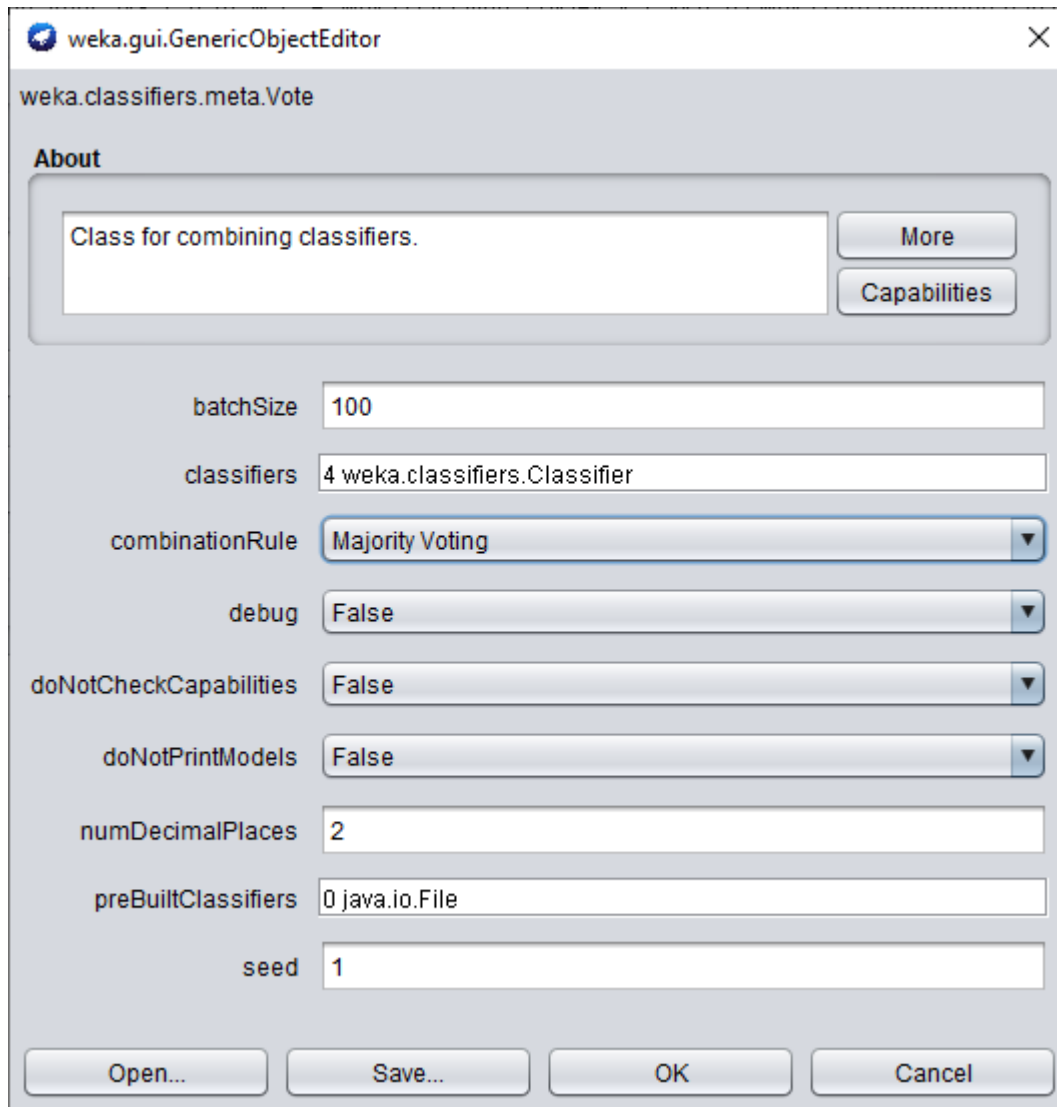


Figure 3.8: KNSD configuration setup (author)

iii. KND hybrid classifier was built by combining concurrently through bagging three base classifiers KNN, NB, and DT which outperformed MLP and SVM at the identified resample rate for optimum performance. In the concurrent ensemble methodology, the original dataset was partitioned into several subsets from which multiple classifiers were induced concurrently. A majority voting combining procedure was then applied in order to produce a single classification for a given instance. The configuration for combination of the base classifiers is shown in fig.3.9. This model was then evaluated using 10-fold cross validation test option and test split test option. Classification accuracy and build tabulated. The process was repeated for all the datasets

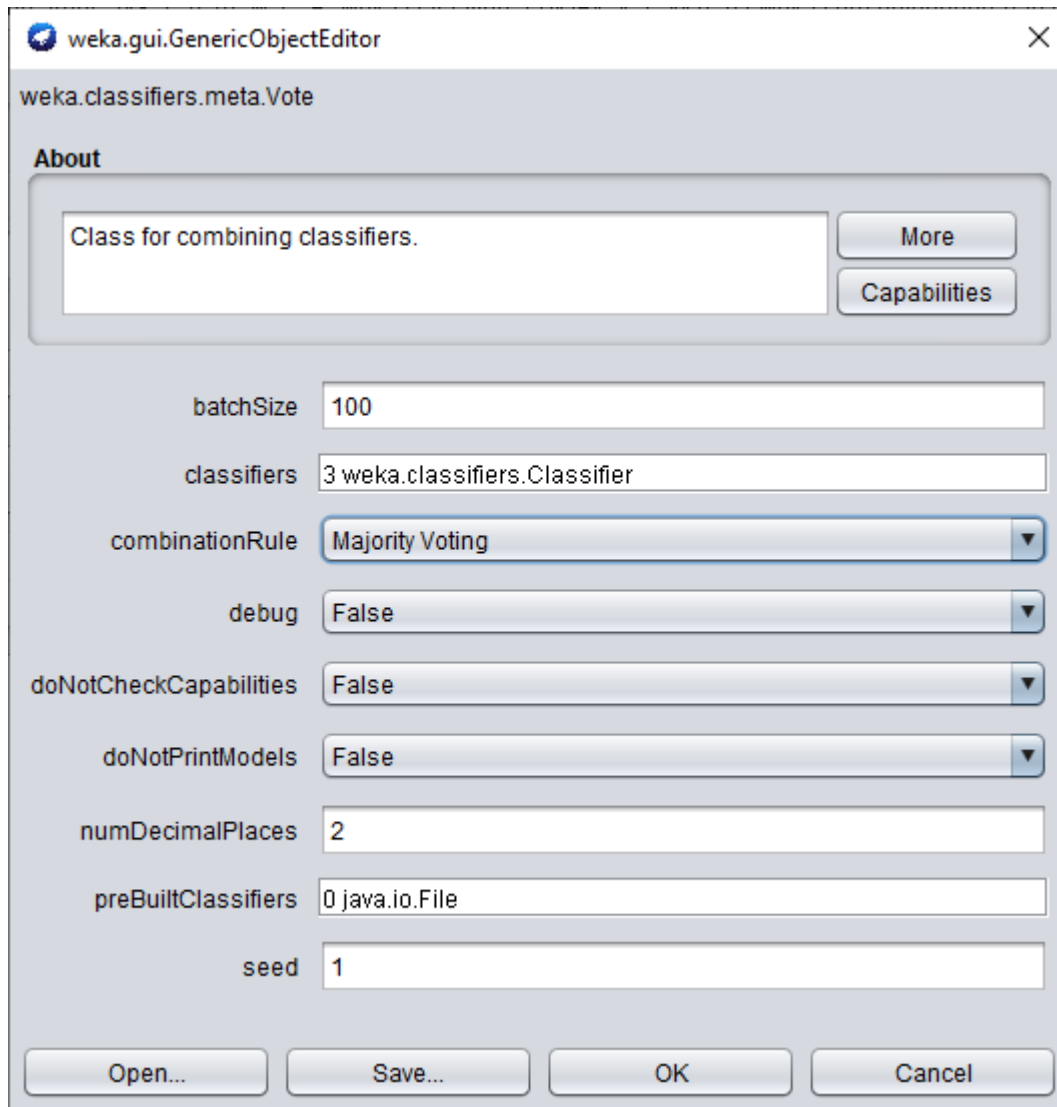


Figure 3.9: KND configuration setup(author)

iv. KD hybrid classifier was built by combining concurrently two base classifiers KNN, and DT which outperformed MLP, DT and SVM at the identified resample rate for optimum performance. In the concurrent ensemble methodology, the original dataset was partitioned into several subsets from which multiple classifiers were induced concurrently. A majority voting combining procedure was then applied in order to produce a single classification for a given instance. The configuration for combination of the base classifiers is shown in fig.3.10. The model was then evaluated using 10-fold cross validation test option and test split test option. Classification accuracy and build tabulated. The process was repeated for all the datasets

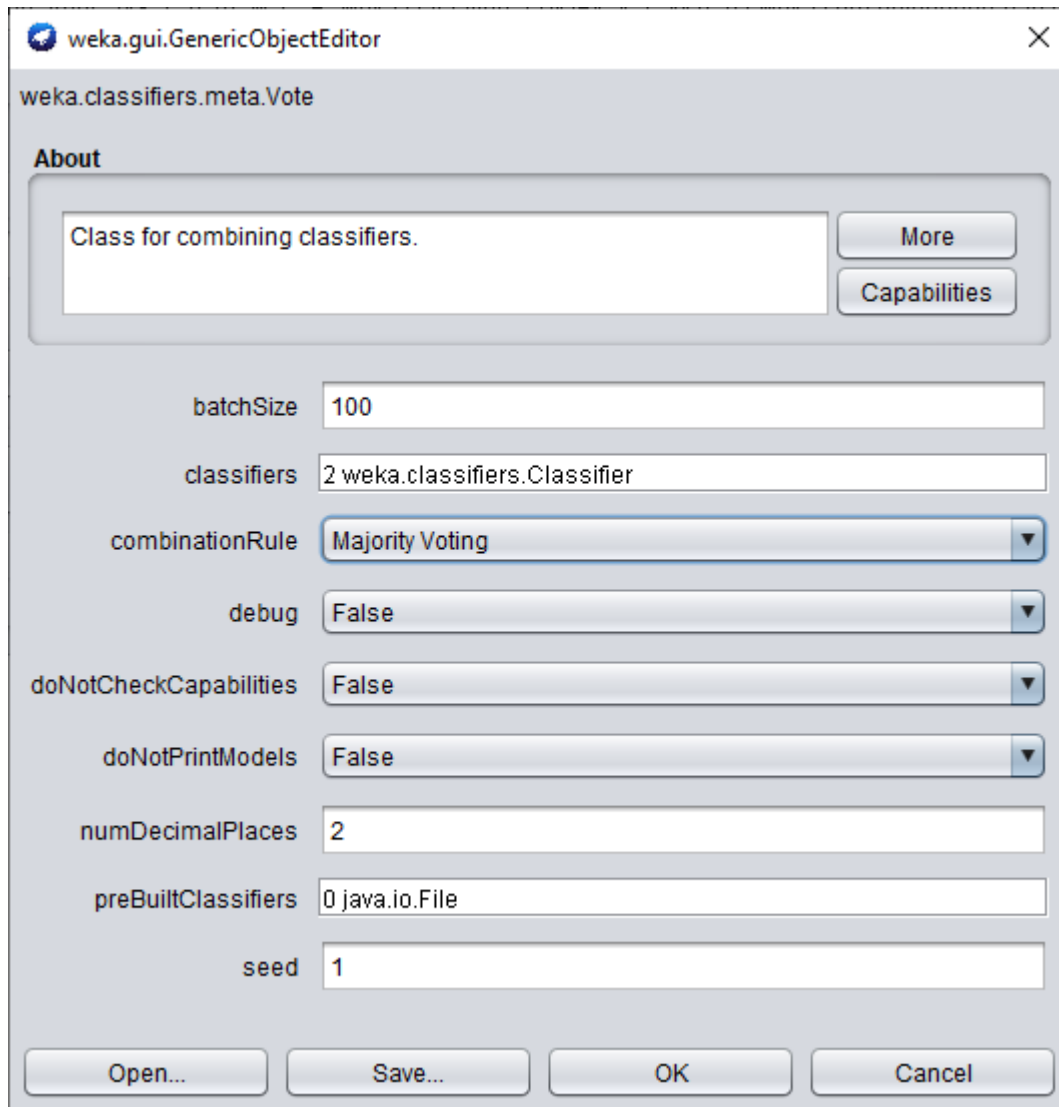


Figure 3.10: KD configuration setup(author)

3.5.3 The KD Hybrid Architecture

The hybrid KD which is a combination of the KNN, and DT has outperformed other hybrid in classification accuracy. The general architecture of the hybrid is shown in fig.3.11.

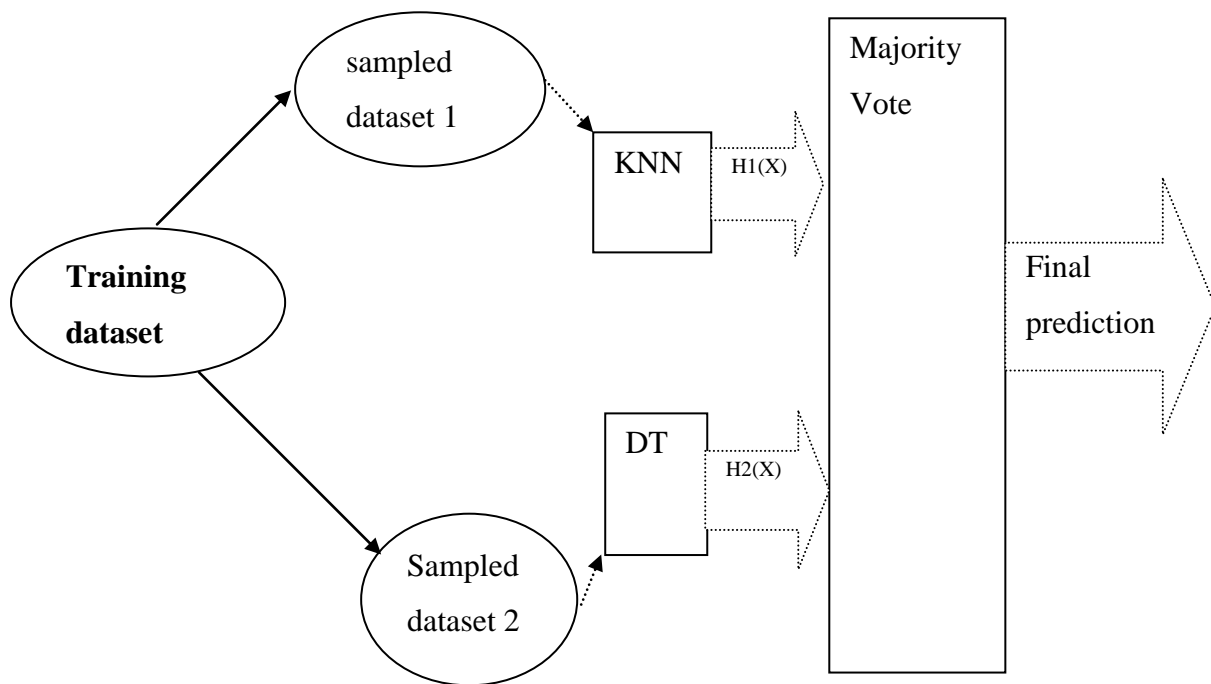


Figure 3.11: Architecture of the KD hybrid classifier algorithm model (Author)

The hybrid classifier algorithm has the following components

Sampled datasets 1 and 2 is made of bootstrapped samples, sampled randomly from the original dataset with replacement.

A set of diverse classifiers KNN and DT are selected after outperforming NB, MLP and SVM, trained on each of these different subsets of the original datasets balanced by resampling at the sample size percent of 1000 for optimum performance of the classifier algorithm models.

Aggregation stage where all the predictions of all the base classifier algorithm models are combined. This is a classification problem and therefore majority voting is used. The evaluation is done in 10-fold cross validation and test split test options.

The outputs of the KNN and DT are combined together to predict the output of the final model. The output with the majority vote is predicted.

CHAPTER FOUR

RESULTS & DISCUSSIONS

In this chapter data is presented and analysed. The aim of the study was to build and evaluate hybrid classifier algorithm models for identification of terrorist groups in the aftermath of an attack. Several experiments were conducted, and the purpose was to determine the approaches to be used to build and evaluate base and hybrid classifier algorithms and compare performance of the classifier algorithms. Data mining and prediction tool WEKA used in this study and the model building and evaluation has proved to be very efficient in prediction from the big data available in GTD. The results of the study are organised in sub-topics focusing on answering research questions.

4.1 Build and Evaluate Base Classifier algorithm models for the Identification of Terrorist Group

The study sought to establish what approach to be used to build and evaluate base classifier algorithm models in the identification of terrorist group in the aftermath of an attack.

4.1.1 Decision Tree

Table 4.1 shows tabulated accuracy results for 10-fold cross validation and test split test option. The accuracy rate in percentage for the control experiment is for dataset with no resample at 77.42 for 10-fold cross validation and 77.54 for test split, respectively. As the resample sample size percent increases so does the accuracy rate to a Resample sample size percent of 1000 and accuracy rate in percentage of 87.53 for 10-fold cross validation and 86.68 for test split respectively, and accuracy rate starts to fall with further increase in the resample sample size percentage. At resample sample size percent of 1300, the accuracy rate in percentage is 86.93 for 10-fold cross validation and 85.40 for test split, respectively. This can be attributed to the fact that the dataset is initially imbalanced without resampling, as the dataset is resampled at different rates, it becomes balanced and optimizes performance at resample sample size percent rate of 1000 and starts to suffer again from overfitting with increasing resample sample size percent eventually affecting the rate of accuracy of the classifier model. The accuracy rates for 10-fold cross validation is generally higher than those for test split even though with a small margin as shown in fig. 4.1. Both 10-fold cross validation and test split yield optimum accuracy results at the resample sample size percent of 1000.

Table 4.1: DT Accuracy

Classifier model rate of accuracy (%)		
Resample (%)	10-foldCross Validation	Test Split
None	77.42	77.53
100	79.08	77.85
200	82.46	82.74
300	83.72	83.55
400	85.59	84.62
500	86.31	84.62
600	86.40	85.07
700	86.42	86.01
800	86.89	86.02
900	87.34	86.50
1000	87.53	86.68
1100	87.45	86.55
1200	87.13	85.90
1300	86.93	85.40

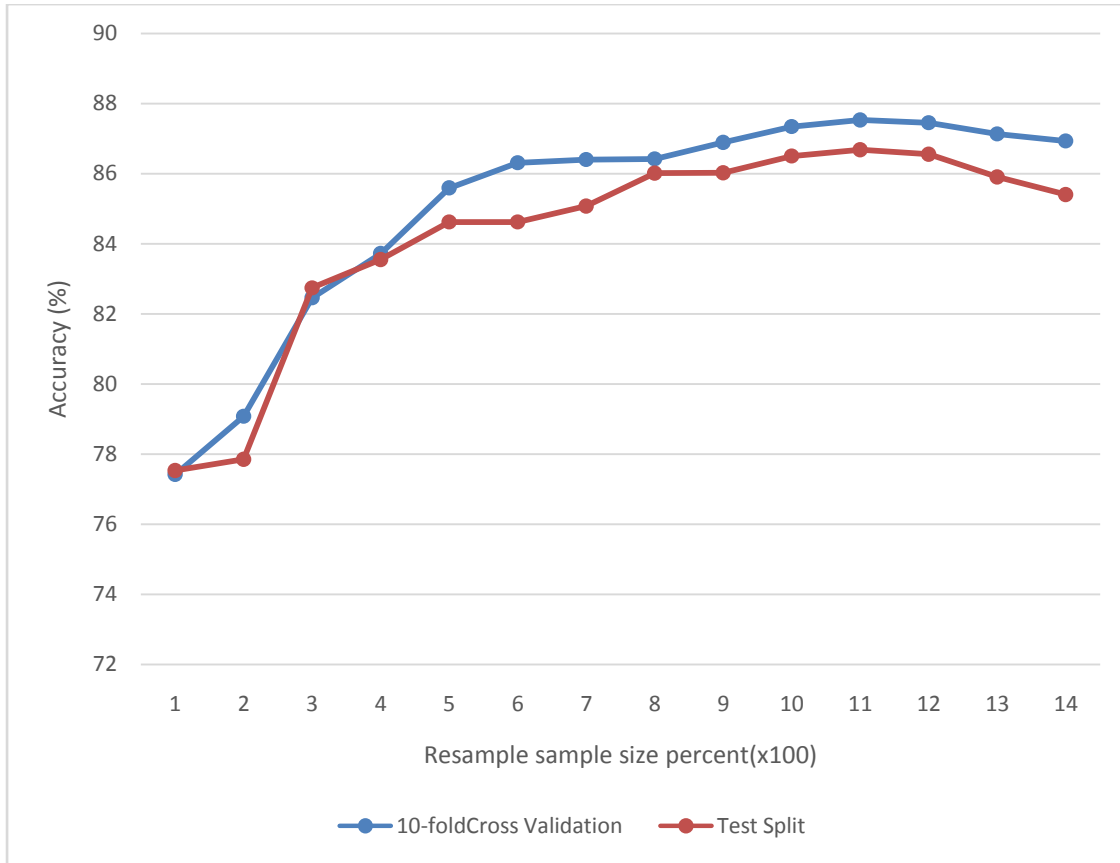


Figure 4.1: DT Accuracy

Table 4.2 shows tabulated build time results for 10-foldcross validation and test split test option. Build time for test split are generally longer than build time for 10-fold cross validation and increases with the resample sample size percent as shown in fig.4.2.This is attributed to more instances in resampled datasets that increase the computational cost of the classifier algorithm models. The build time in seconds for the control experiments is 0.01 for 10-fold cross validation, and 0.01 for test split. The build time in seconds at the resample sample size percentage of 1000 is 0.02 for 10-fold cross validation and 0.05 for test split, respectively. The build time in seconds at the resample sample size percentage of 1300 is 0.04 for 10-fold cross validation and 0.02 for test split, respectively.

Table 4.2: DT build time

Classifier model build time (sec)		
Resample (%)	10-fold Cross Validation	Test Split(66/34)
None	0.01	0.01
100	0.01	0.01
200	0.02	0.02
300	0.02	0.03
400	0.01	0.03
500	0.03	0.03
600	0.03	0.02
700	0.02	0.02
800	0.04	0.04
900	0.04	0.06
1000	0.02	0.05
1100	0.02	0.06
1200	0.03	0.04
1300	0.04	0.02



Figure 4.2: DT build time

4.1.2 K-Nearest Neighbor

Table 4.3 shows tabulated accuracy results for 10-fold cross validation and test split test option. The accuracy rate in percentage for the control experiment is for dataset with no resample at 72.70 for 10-fold cross validation and 70.41 for test split, respectively. As the resample sample size percentage increases so does the accuracy rate to a Resample sample size percentage of 1000 and accuracy rate in percentage of 87.73 for 10-fold cross validation and 87.06 for test split respectively, and accuracy rate starts to fall with further increase in the resample sample size percent. At resample sample size percent of 1300, the accuracy rate in percentage is 87.52 for 10-fold cross validation and 86.70 for test split, respectively. This can be attributed to the fact that the dataset is initially imbalanced without resampling, as the dataset is resampled at different rates, it becomes balanced and optimizes performance at a resample sample size percent of 1000 and starts to suffer again from overfitting with increasing resample sample size percent eventually affecting the rate of accuracy of the classifier model. The accuracy rates for 10-fold cross validation is generally higher than those for test split even though with a small margin as shown in fig. 4.3. Both 10-fold cross validation and test split yield optimum accuracy results at the resample sample size percent of 1000.

Table 4.3: KNN Accuracy

Classifier model rate of accuracy (%)		
Resample (%)	10-fold Cross Validation	Test Split
None	72.70	70.41
100	73.34	71.16
200	75.89	75.42
300	80.10	79.38
400	82.62	80.49
500	86.24	82.37
600	86.37	83.38
700	86.44	85.96
800	86.66	85.98
900	87.56	87.01
1000	87.73	87.06
1100	87.63	87.02
1200	87.54	86.81
1300	87.52	86.70

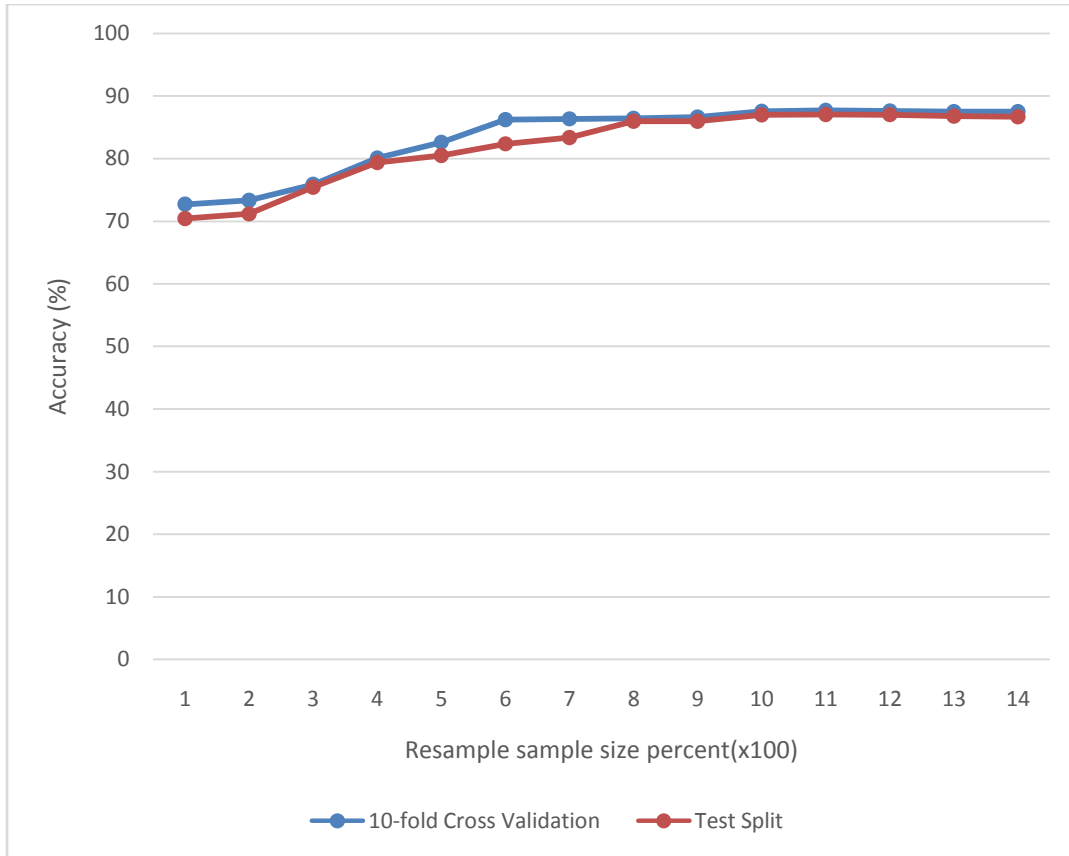


Figure 4.3: KNN Accuracy

Table 4.4 shows tabulated build time results for 10-fold cross validation and test split test option. Build time for test split is generally longer than build time for 10-fold cross validation and increases with the resample sample size percent as shown in fig.4.4. This is attributed to more instances in resampled datasets that increase the computational cost of the classifier algorithm models. The build time in seconds for the control experiments is 0.01 for 10-fold cross validation, and 0.02 for test split. The build time in seconds at the resample sample size percent of 1000 is 0.01 for 10-fold cross validation and 0.89 for test split, respectively. The build time in seconds at the resample sample size percentage of 1300 is 0.01 for 10-fold cross validation and 1.45 for test split, respectively.

Table 4.4: KNN Build Time

Classifier model Build time (sec)		
Resample (%)	10-fold Cross Validation	Test Split(66%/34%)
None	0.01	0.02
100	0.01	0.02
200	0.01	0.08
300	0.01	0.13
400	0.01	0.19
500	0.01	0.28
600	0.01	0.39
700	0.01	0.51
800	0.01	0.65
900	0.01	0.77
1000	0.01	0.89
1100	0.01	1.01
1200	0.01	1.26
1300	0.01	1.45

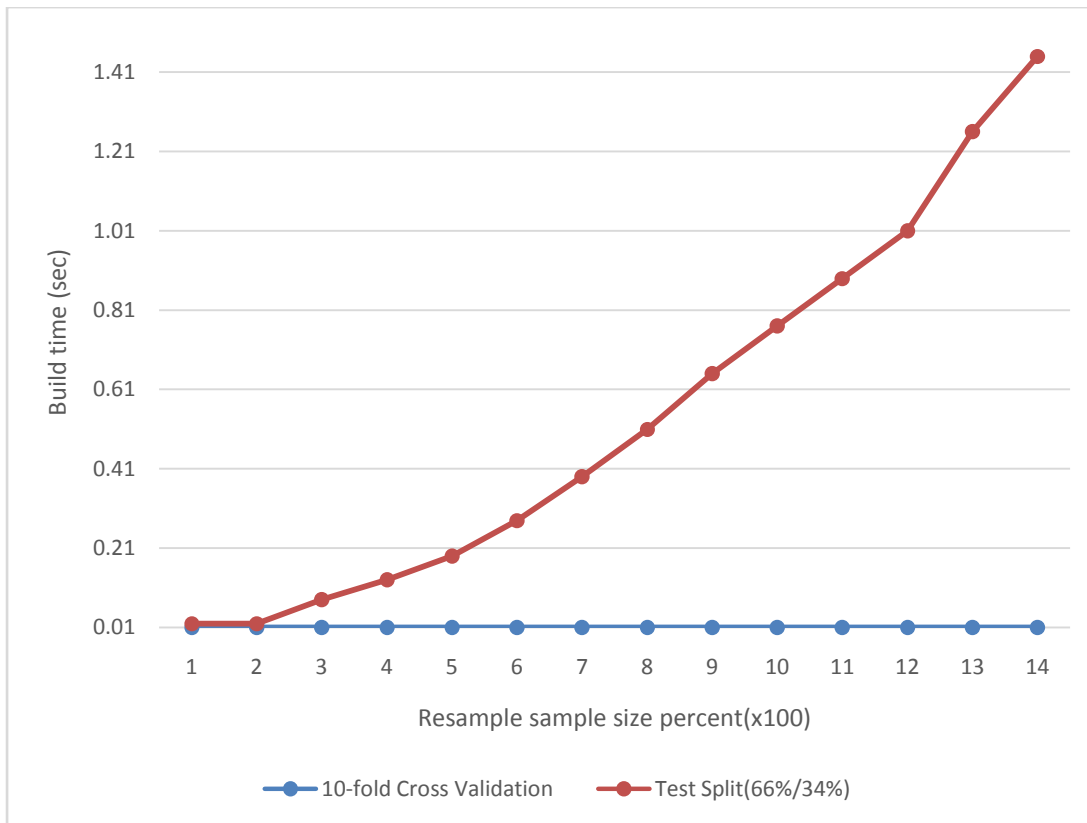


Figure 4.4: KNN build time

4.1.3 Support Vector Machine

Table 4.5 shows tabulated accuracy results for 10-fold cross validation and test split test option. The accuracy rate in percentage for the control experiment is for dataset with no resample at 76.15 for 10-fold cross validation and 77.90 for test split, respectively. As the resample sample size percent increases so does the accuracy rate to a Resample sample size percent of 1000 and accuracy rate in percentage of 86.17 for 10-fold cross validation and 86.57 for test split respectively, and accuracy rate starts to fall with further increase in the resample sample size percent. At resample sample size percent of 1300, the accuracy rate in percentage is 85.6 for 10-fold cross validation and 85.00 for test split, respectively. This can be attributed to the fact that the dataset is initially imbalanced without resampling, as the dataset is resampled at different rates, it becomes balanced and optimizes at a resample sample size percent rate of 1000 and starts to suffer again from overfitting with increasing resample sample size percent eventually affecting the rate of accuracy of the classifier algorithm model. The accuracy rates for 10-fold cross validation are slightly better than test split as resample sample size percent increases to 1000 and starts to be slightly lower than the accuracy rates for test split as the resample sample size percent exceeds 1000 to 1200 and

risers slightly above the accuracy rate for test split as shown in fig.4.5. The performance of test split test option is at its peak and better than that of 10-fold cross validation at the resample sample size percent of 1000. Both 10-fold cross validation and test split yield optimum accuracy results at the resample sample size percent of 1000.

Table 4.5: SVM Accuracy

Classifier model rate of accuracy (%)		
Resample (%)	10-fold Cross Validation	Test Split
None	76.15	77.90
100	80.10	77.15
200	83.61	81.80
300	83.93	82.38
400	84.02	82.55
500	84.55	83.96
600	84.89	84.62
700	85.42	84.64
800	85.48	84.99
900	85.63	85.54
1000	86.17	86.57
1100	86.01	86.39
1200	85.96	85.89
1300	85.60	85.00

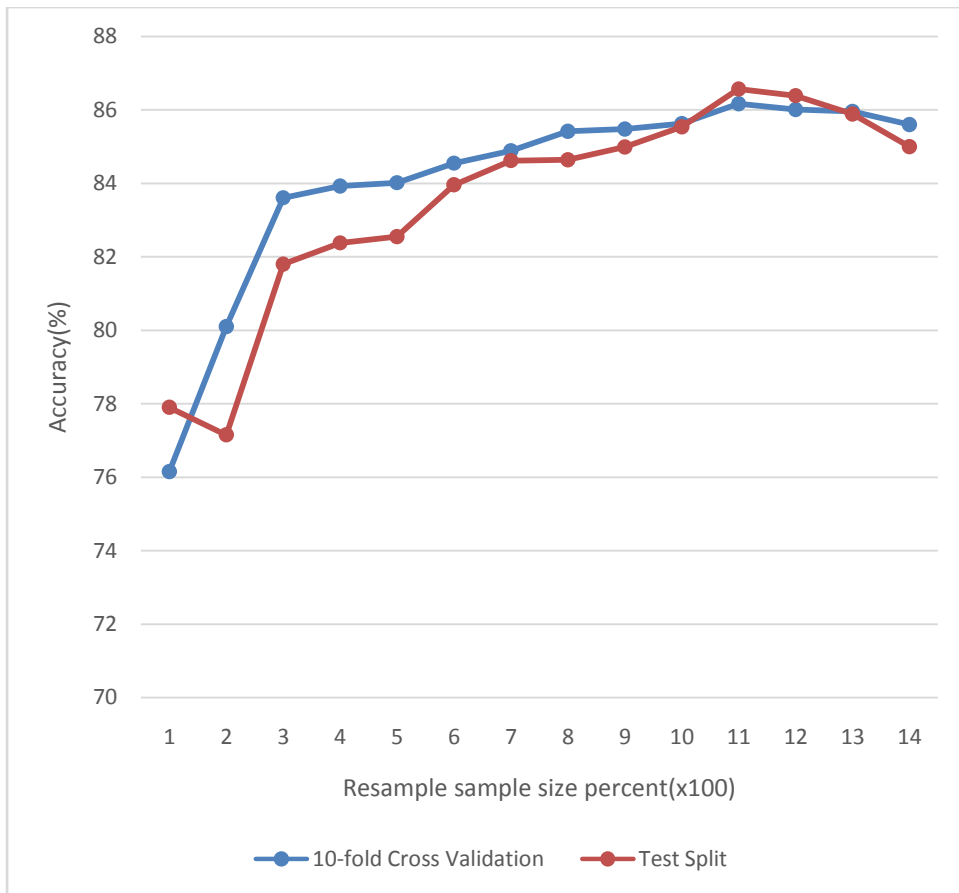


Figure 4.5: SVM accuracy

Table 4.6 shows tabulated build time results for 10-foldcross validation and test split test option. Build time for test split and generally longer than build time for 10-fold cross validation and increase with the resample sample size percent as shown in fig.4.6.This is attributed to more instances in resampled datasets that increase the computational cost of the classifier algorithm models. The build time in seconds for the control experiments is 4.27 for 10-fold cross validation, and 4.30 for test split. The build time in seconds at the resample sample size percent of 1000 is 17.08 for 10-fold cross validation and 18.37 for test split, respectively. The build time in seconds at the resample sample size percent of 1300 is 36.58 for 10-fold cross validation and 28.53 for test split, respectively.

Table 4.6: SVM build time

Classifier model Build time (sec)		
Resample (%)	10-fold Cross Validation	Test Split(66/34)
None	4.27	4.30
100	4.05	4.22
200	3.76	4.19
300	4.77	5.29
400	7.33	6.55
500	8.74	7.89
600	18.82	11.99
700	21.22	13.29
800	13.32	14.04
900	14.16	14.44
1000	17.08	18.37
1100	30.97	23.72
1200	25.66	24.59
1300	36.58	28.53

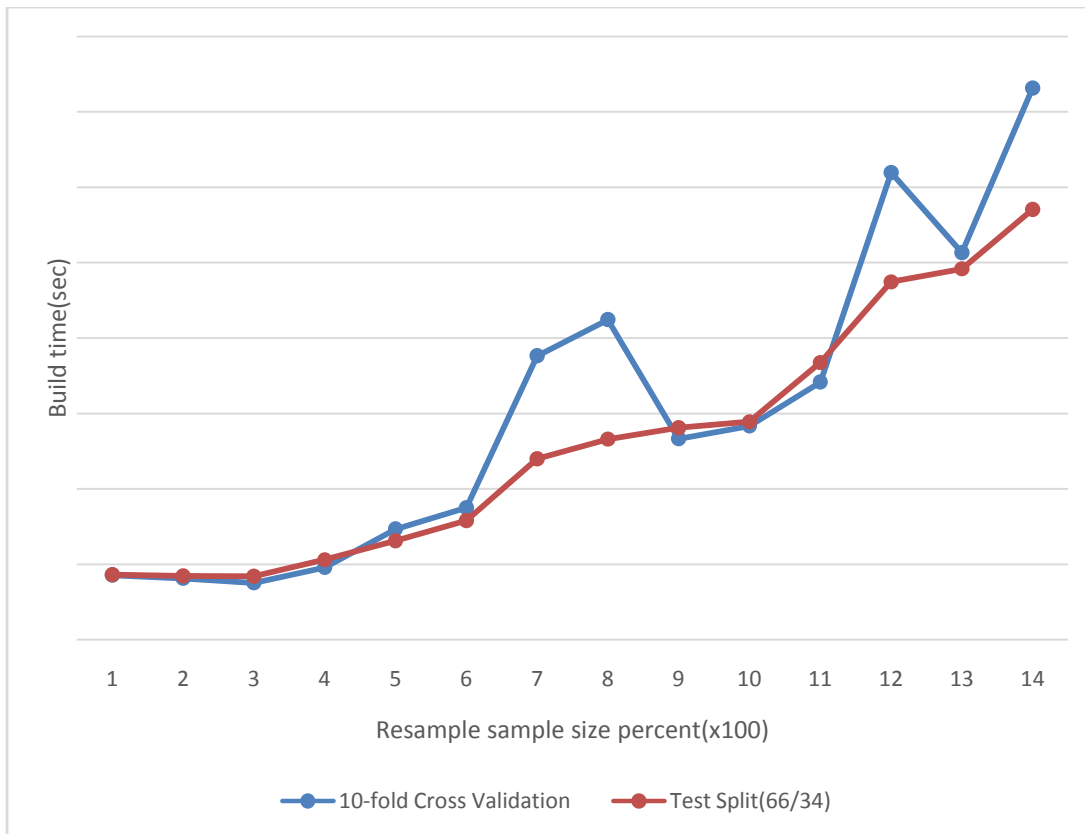


Figure 4.6: SVM build time

4.1.4 Multi -Layer Perceptron

Table 4.7 shows tabulated accuracy results for 10-fold cross validation and test split test option. The accuracy rate in percentage for the control experiment is for dataset with no resample at 68.62 for 10-fold cross validation and 69.66 for test split, respectively. As the resample sample size percent increases so does the accuracy rate to a Resample sample size percent of 1000 and accuracy rate in percentage of 87.06 for 10-fold cross validation and 87.58 for test split respectively, and accuracy rate starts to fall with further increase in the resample sample size percent. At resample sample size percentage of 1300, the accuracy rate in percentage is 86.41 for 10-fold cross validation and 85.67 for test split respectively. This can be attributed to the fact that the dataset is initially imbalanced without resampling, as the dataset is resampled at different rates, it becomes balanced and optimizes performance at a resample sample size percent rate of 1000 and starts to suffer again from overfitting with increasing resample sample size percent eventually affecting the rate of accuracy of the classifier model. The accuracy rates for 10-fold cross validation are slightly better than test split as resample sample size percent increases to 500 and starts to be slightly lower than the accuracy rates for test split as the resample sample size percent exceeds 1200 and rises

slightly above the accuracy rate for test split as shown in fig.4.7. The performance of test split test option is at its peak and better than that of 10-fold cross validation at the resample sample size percent of 1000. Both 10-fold cross validation and test split yield optimum accuracy results at the resample sample size percent of 1000.

Table 4.7: MLP accuracy

Classifier models rate of accuracy (%)		
Resample (%)	10-fold Cross Validation	Test Split
None	68.62	69.66
100	80.23	76.40
200	82.27	81.61
300	84.53	84.00
400	85.20	85.27
500	85.90	85.90
600	86.12	86.18
700	86.34	87.26
800	86.73	87.32
900	86.77	87.46
1000	87.09	87.58
1100	86.55	86.70
1200	86.47	85.68
1300	86.41	85.50

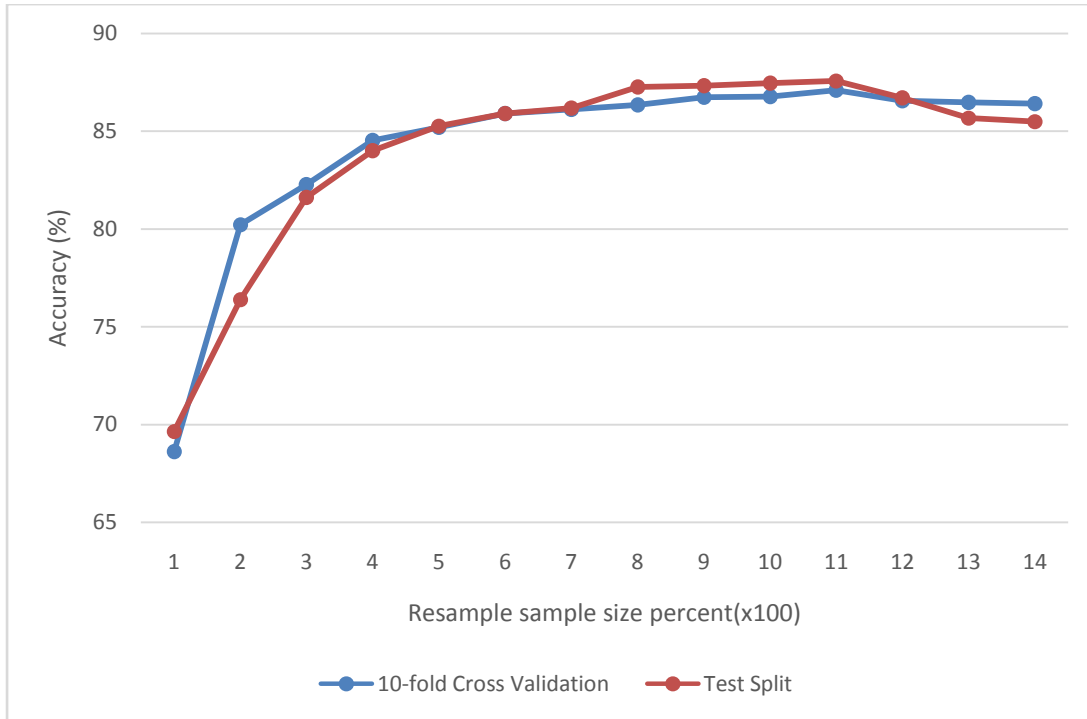


Figure 4.7: MLP accuracy

Table 4.8 shows tabulated build time results for 10-fold cross validation and test split test option. Build time for test split is generally longer than build time for 10-fold cross validation and increases with the resample sample size percent as shown in fig.4.8. This is attributed to more instances in resampled datasets that increase the computational cost of the classifier algorithm models. The build time in seconds for the control experiments is 54.34 for 10-fold cross validation, and 54.06 for test split. The build time in seconds at the resample sample size percent of 1000 is 530.40 for 10-fold cross validation and 543.30 for test split, respectively. The build time in seconds at the resample sample size percent of 1300 is 674.80 for 10-fold cross validation and 698.40 for test split, respectively.

Table 4.8: MLP build time

Classifier model build time (sec)		
Resample (%)	10-fold Cross Validation	Test Split(66/34)
None	54.34	54.06
100	53.42	52.67
200	107.40	106.90
300	157.80	155.90
400	212.90	213.90
500	270.60	269.00
600	318.40	319.60
700	365.00	371.30
800	413.10	421.80
900	473.80	466.90
1000	530.40	543.30
1100	569.10	567.30
1200	630.40	622.90
1300	674.80	698.40

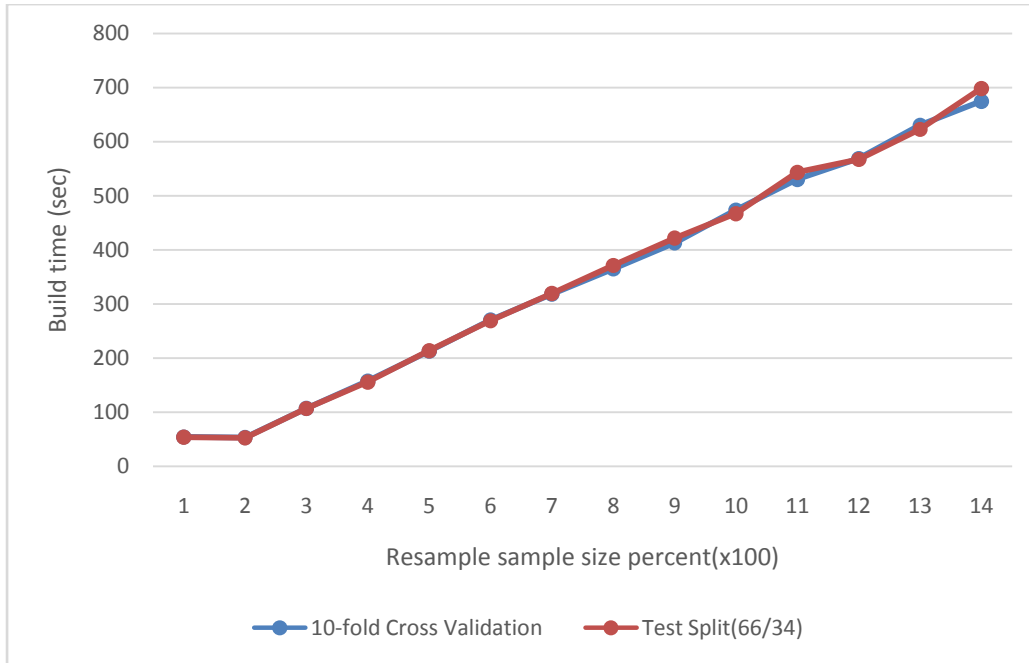


Figure 4.8: MLP build time

4.1.5 Naïve' Bayes

Table 4.9 shows tabulated accuracy results for 10-fold cross validation and test split test option. The accuracy rate in percentage for the control experiment is for dataset with no resample at 76.66 for 10-fold cross validation and 77.15 for test split, respectively. As the resample sample size percent increases so does the accuracy rate to a Resample sample size percent of 1000 and accuracy rate in percentage of 83.61 for 10-fold cross validation and 82.86 for test split respectively, and accuracy rate starts to fall with further increase in the resample sample size percentage. At resample sample size percent of 1300, the accuracy rate in percentage is 82.61 for 10-fold cross validation and 82.02 for test split, respectively. This can be attributed to the fact that the dataset is initially imbalanced without resampling, as the dataset is resampled at different rates, it becomes balanced and optimizes performance at a resample sample size percent rate of 1000 and starts to suffer again from overfitting with increasing resample sample size percent eventually affecting the rate of accuracy of the classifier model. The accuracy rates for 10-fold cross validation is generally higher than those for test split even though with a small margin as shown in fig. 4.9. Both 10-fold cross validation and test split yield optimum accuracy results at the resample sample size percent of 1000.

Table 4.9: NB accuracy

Classifier model rate of accuracy (%)		
Resample (%)	10-fold Cross Validation	Test Split
None	76.66	77.15
100	79.08	77.15
200	78.95	78.99
300	80.57	81.13
400	80.68	81.14
500	81.30	81.50
600	81.91	81.95
700	82.45	82.05
800	82.60	82.40
900	82.96	82.62
1000	83.61	82.86
1100	82.75	82.77
1200	82.66	82.68
1300	82.61	82.02

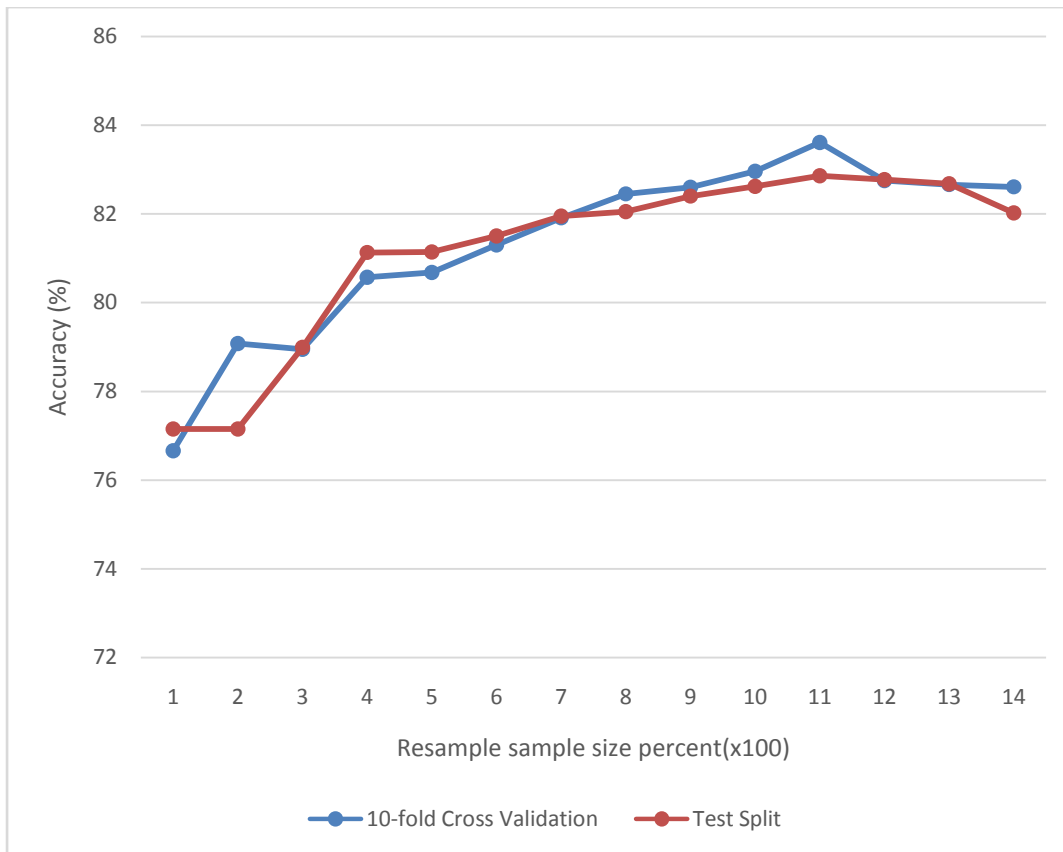


Figure 4.9: NB accuracy

Table 4.10 shows tabulated build time results for 10-fold cross validation and test split test option. Build time for test split is generally longer than build time for 10-fold cross validation and increases with the resample sample size percentage as shown in fig.4.10. This is attributed to more instances in resampled datasets that increase the computational cost of the classifier algorithm models. The build time in seconds for the control experiments is 0.01 for 10-fold cross validation, and 0.01 for test split. The build time in seconds at the resample sample size percentage of 1000 is 0.01 for 10-fold cross validation and 0.06 for test split, respectively. The build time in seconds at the resample sample size percentage of 1300 is 0.01 for 10-fold cross validation and 0.08 for test split, respectively.

Table 4.10: NB Build Time

Classifier model Build time (sec)		
Resample (%)	10-fold Cross Validation	Test Split(66/34)
None	0.01	0.01
100	0.01	0.01
200	0.02	0.01
300	0.01	0.02
400	0.01	0.03
500	0.01	0.03
600	0.01	0.05
700	0.01	0.05
800	0.01	0.05
900	0.01	0.05
1000	0.01	0.06
1100	0.01	0.08
1200	0.01	0.08
1300	0.01	0.08

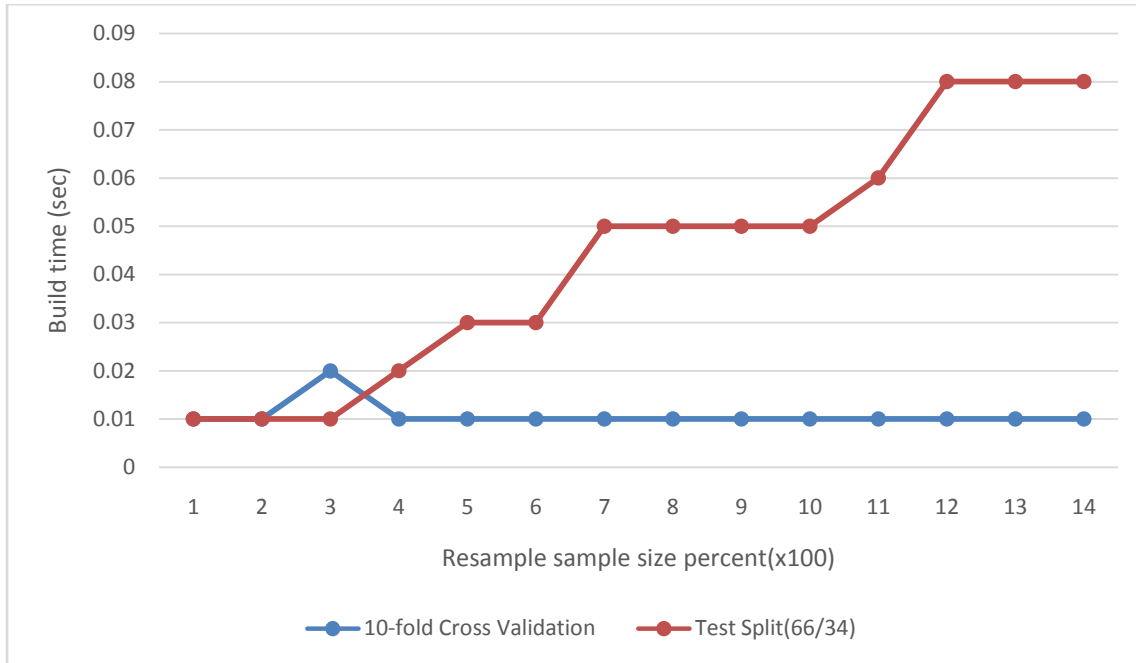


Figure 4.10: NB build time

From the experimental results, 10-fold cross validation test option yield higher accuracy rates for KNN, DT and NB, as compared to test split test options for a resample sample size percent of 1000. Test split on the other hand yields better accuracy for MLP and SVM as compared to 10-fold cross validation for a resample sample size percent of 1000. The study demonstrates that the resample sample size percent for optimum percentage accuracy is at 1000 for the dataset for all base classifier algorithm models. An insight we can draw from the results is that class imbalance affects accuracy of classifiers. The results are consistent with studies which show that imbalanced data set reduces performance and demonstrating the gains of using resampling in imbalanced data set (Garcia, Marques, & Sanchez, 2012). For 10-fold cross validation NB performed worst with the lowest accuracy percentage rate of 83.61 and build time of 0.01 seconds, followed by MLP with accuracy percentage rate of 87.09 and build time of 530.4 seconds, SVM with accuracy percentage rate of 86.17 and build time of 17.08 seconds, DT with accuracy percentage rate of 87.53 and build time of 0.02 seconds and KNN outperformed all the base classifiers with accuracy percentage rate of 87.78, build time of 0.01 seconds. For test split NB performed worst with the lowest accuracy percentage rate of 82.86 and build time of 0.06 seconds, followed by SVM with accuracy percentage rate of 86.57 and build time of 18.37 seconds, DT with accuracy percentage rate of 86.68 and build time of 0.05 seconds, KNN with accuracy percentage rate of 87.06 and build time of 0.89 seconds and MLP outperformed all the base classifiers with accuracy

percentage rate of 87.58, build time of 543.30 seconds. Test split avoids the overlap between training data and test data, yielding more accurate estimate for the generalization performance of the algorithm. The downside is that this procedure does not use all the available data and the results are highly dependent on the choice of the training/test split (Dietterich, 1998). The instances chosen for inclusion in the test set may be too easy or too difficult to classify and this can skew the results. Furthermore, the data in the test set may be valuable for training and if it is held out prediction performance may suffer, again leading to skewed results. In typical cross validation, the training and validation sets must cross-over in successive rounds such that each data points has a chance of being validated against. 10-fold cross validation is a good compromise in data mining. It is particularly attractive because it makes predictions using 90% of data, making it more likely to be generalizable to the full data(Refaeilzadeh, Tang, & Liu, 2007). The study answers the research question by determining the optimal design parameters for respective base classifiers and fit with the theory that different machine learning algorithms make different assumptions about the shape and structure of the function and how best to optimize a representation to approximate it, and why it is important to try a suite of different algorithms on a classification problem. The key question when dealing with classification is not whether a learning algorithm is superior to others, but under which conditions a particular method can significantly outperform others on a given application problem, and that KNN, NB, & DT take shortest Build time, SVM and MLP take longer Build time (Kotsiantis, Zaharakis, & Pintea, 2004).Generalizability of the results of the experiment is limited by resample technique as a data approach to solving class imbalance.

4.2 Build and Evaluate Hybrid Classifier Algorithm Models for the Identification of Terrorist Groups

The study sought to establish what approaches to be used to build and evaluate hybrid classifier models for identification of terrorist groups in the aftermath of an attack.

4.2.1 Hybrid KNMSD

Table 4.11 shows tabulated accuracy results for 10-fold cross validation and test split test option. The accuracy rate in percentage for the control experiment is for dataset with no resampling at 77.17 for 10-fold cross validation and 77.53 for test split, respectively. As the resample sample size percent increases so does the accuracy rate to a Resample sample size percentage of 1000 and accuracy rate in percentage of 87.83 for 10-fold cross validation and 87.36 for test split respectively, and accuracy rate starts to fall with further increase in the resample sample size percent. At resample sample size percent of 1300, the accuracy rate in percentage is 87.25 for 10-fold cross validation and 86.20 for test split, respectively. This can be attributed to the fact that the dataset is initially imbalanced without resampling, as the dataset is resampled at different rates, it becomes balanced and optimizes performance at a resample sample size percent rate of 1000 and starts to suffer again from overfitting with increasing resample sample size percentage eventually affecting the rate of accuracy of the classifier model. The accuracy rates for 10-fold cross validation is generally higher than those for test split even though with a small margin as shown in fig. 4.11. Both 10-fold cross validation and test split yield optimum accuracy results at the resample sample size percent of 1000.

Table 4.11: Hybrid KNMSD accuracy

Classifier model rate of accuracy (%)		
Resample (%)	10-fold Cross Validation	Test Split
None	77.17	77.53
100	79.60	75.28
200	83.48	82.93
300	84.23	84.00
400	85.78	84.90
500	86.40	85.37
600	86.82	85.83
700	86.93	86.66
800	87.04	86.87
900	87.54	86.99
1000	87.83	87.36
1100	87.47	87.34
1200	87.44	86.34
1300	87.25	86.20

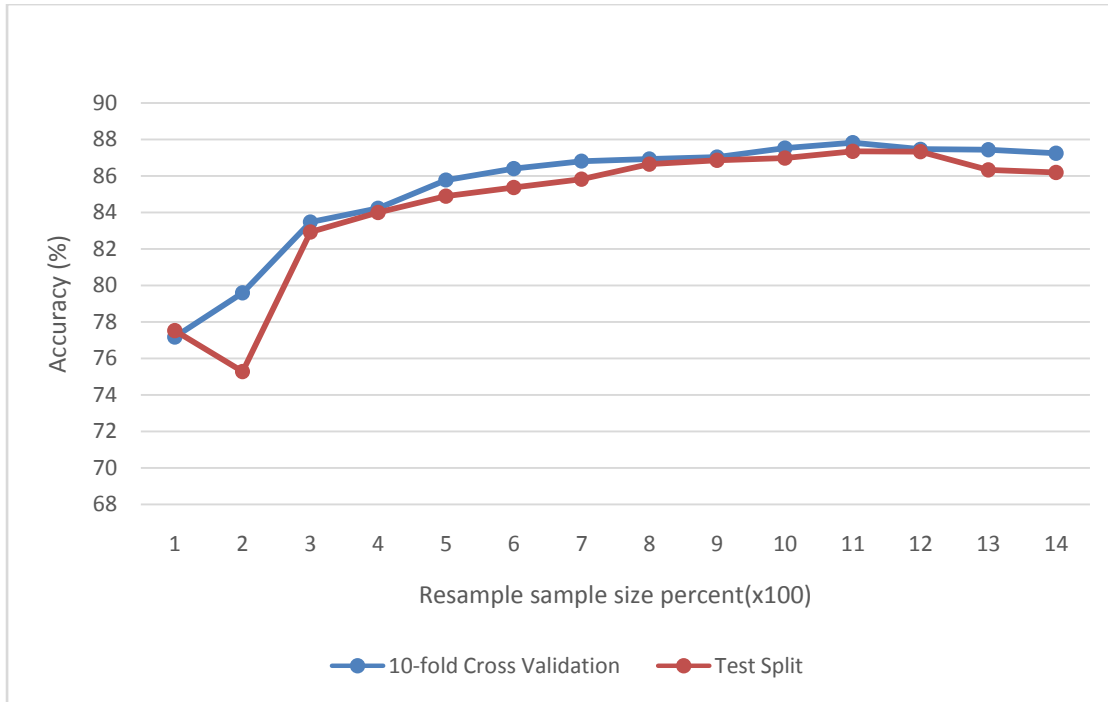


Figure 4.11: Hybrid KNMSD Accuracy

Table 4.12 shows tabulated build time results for 10-fold cross validation and test split test option. Build time for test split is generally longer than build time for 10-fold cross validation and increases with the resample sample size percent as shown in fig.4.12. This is attributed to more instances in resampled datasets that increase the computational cost of the classifier algorithm models. The build time in seconds for the control experiments is 66.91 for 10-fold cross validation, and 57.70 for test split. The build time in seconds at the resample sample size percent of 1000 is 577.90 for 10-fold cross validation and 575.6 for test split, respectively. The build time in seconds at the resample sample size percent of 1300 is 741.80 for 10-fold cross validation and 730.40 for test split, respectively.

Table 4.12: Hybrid KNMSD build time

Classifier model Build time (sec)		
Resample (%)	10-fold Cross Validation	Test Split(66/34)
None	66.91	57.70
100	56.67	58.50
200	113.20	114.30
300	168.20	168.40
400	222.60	233.80
500	276.80	279.80
600	335.10	341.20
700	393.80	389.90
800	437.50	465.10
900	516.70	513.40
1000	577.90	575.60
1100	622.20	648.10
1200	672.10	673.00
1300	741.80	730.40

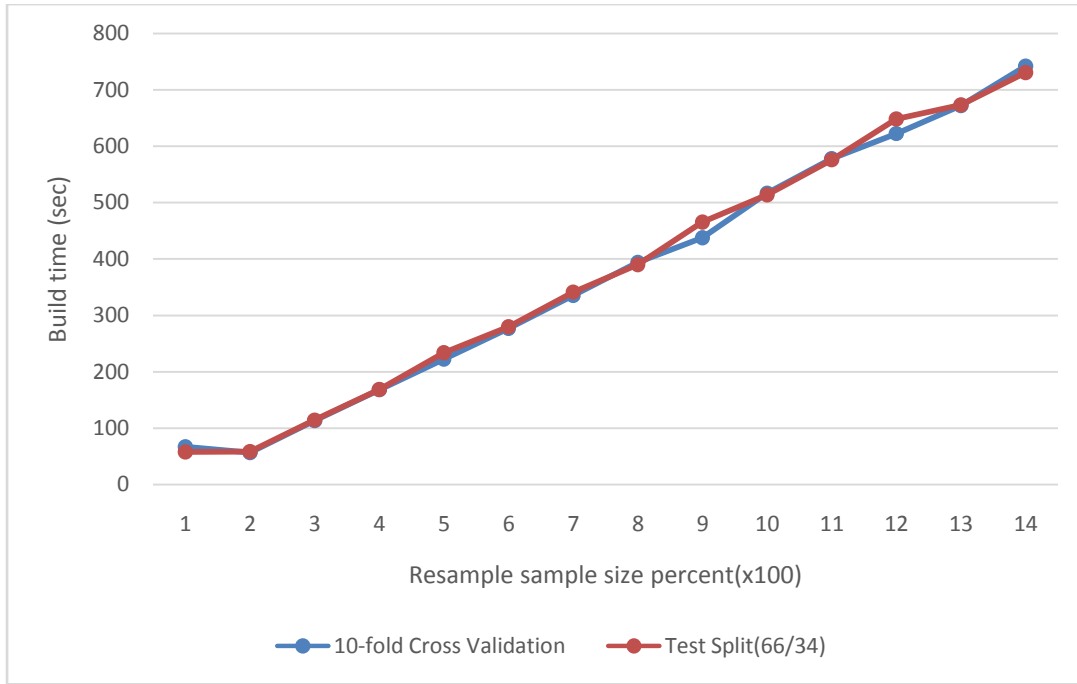


Figure 4.12: Hybrid KNMSD build time

4.2.2 Hybrid KNSD

Table 4.13 shows tabulated accuracy results for 10-fold cross validation and test split test option. The accuracy rate in percentage for the control experiment is for dataset with no resample at 77.93 for 10-fold cross validation and 77.53 for test split, respectively. As the resample sample size percent increases so does the accuracy rate to a Resample sample size percent of 1000 and accuracy rate in percentage of 87.98 for 10-fold cross validation and 87.21 for test split respectively, and accuracy rate starts to fall with further increase in the resample sample size percent. At resample sample size percent of 1300, the accuracy rate in percentage is 87.33 for 10-fold cross validation and 86.24 for test split, respectively. This can be attributed to the fact that the dataset is initially imbalanced without resampling, as the dataset is resampled at different rates, it becomes balanced and optimizes performance at a resample sample size percent rate of 1000 and starts to suffer again from overfitting with increasing resample sample size percent eventually affecting the rate of accuracy of the classifier model. The accuracy rates for 10-fold cross validation is generally higher than those for test split even though with a small margin as shown in fig. 4.13. Both 10-fold cross validation and test split yield optimum accuracy results at the resample sample size percent of 1000.

Table 4.13: Hybrid KNSD accuracy

Classifier model rate of accuracy (%)		
Resample (%)	10-fold Cross Validation	Test Split
None	77.93	77.53
100	77.93	77.90
200	83.16	82.74
300	83.80	83.75
400	85.71	84.30
500	86.59	85.00
600	86.86	85.18
700	86.93	85.91
800	87.27	86.44
900	87.60	87.04
1000	87.98	87.21
1100	87.62	86.92
1200	87.50	86.59
1300	87.33	86.24

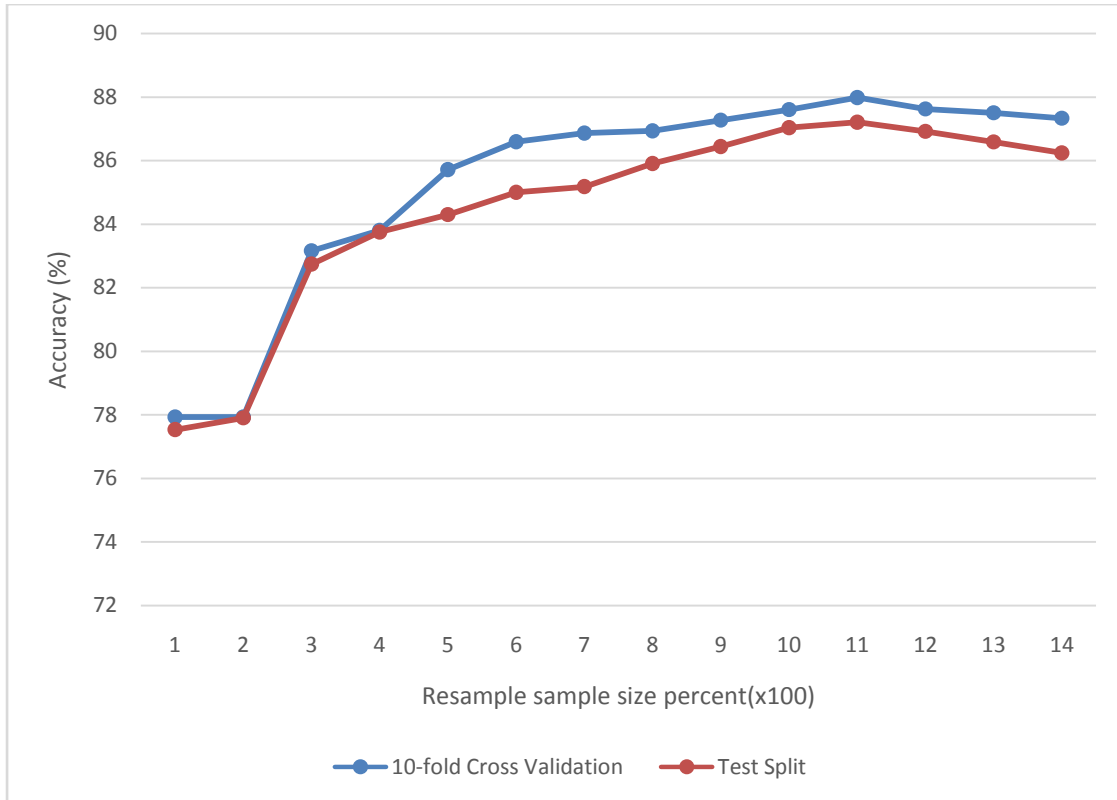


Figure 4.13: Hybrid KNSD Accuracy

Table 4.14 shows tabulated build time results for 10-foldcross validation and test split test option. Build time for test split and generally longer than build time for 10-fold cross validation and increases with the resample sample size percent as shown in Fig.4.14. This is attributed to more instances in resampled datasets that increase the computational cost of the classifier algorithm models. The build time in seconds for the control experiments is 1.97 for 10-fold cross validation, and 4.02 for test split. The build time in seconds at the resample sample size percent of 1000 is 17.83 for 10-fold cross validation and 21.61 for test split, respectively. The build time in seconds at the resample sample size percent of 1300 is 26.27 for 10-fold cross validation and 27.17 for test split, respectively.

Table 4.14: Hybrid KNSD build time

Classifier model build time (sec)		
Resample (%)	10-fold Cross Validation	Test Split(66/34)
None	1.97	4.02
100	1.94	3.61
200	3.37	4.26
300	4.85	5.37
400	7.58	6.58
500	9.63	8.46
600	10.42	10.2
700	11.37	11.05
800	13.34	14.72
900	13.34	15.58
1000	17.83	21.61
1100	22.71	22.52
1200	24.80	26.25
1300	26.27	27.17

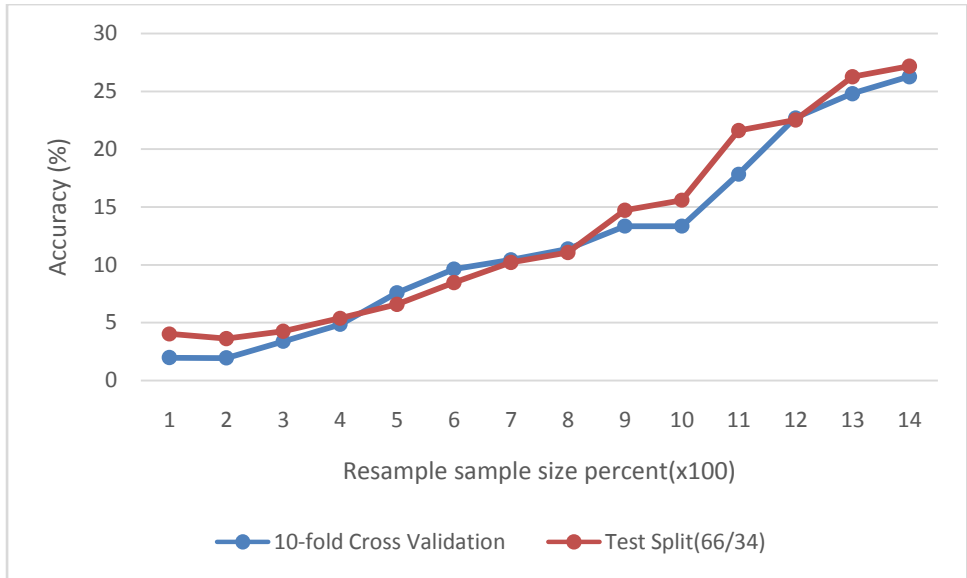


Figure 4.14: Hybrid KNSD build time

4.2.3 Hybrid KND

Table 4.15 shows tabulated accuracy results for 10-fold cross validation and test split test option. The accuracy rate in percentage for the control experiment is for dataset with no resampling at 78.57 for 10-fold cross validation and 77.90 for test split, respectively. As the resample sample size percent increases so does the accuracy rate to a Resample sample size percent of 1000 and accuracy rate in percentage of 87.93 for 10-fold cross validation and 86.95 for test split respectively, and accuracy rate starts to fall with further increase in the resample sample size percent .At resample sample size percent of 1300, the accuracy rate in percentage is 87.37 for 10-fold cross validation and 86.26 for test split respectively. This can be attributed to the fact that the dataset is initially imbalanced without resampling, as the dataset is resampled at different rates, it becomes balanced and optimizes performance at a resample sample size percentage rate of 1000 and starts to suffer again from overfitting with increasing resample sample size percent eventually affecting the rate of accuracy of the classifier model. The accuracy rates for 10-fold cross validation is generally higher than those for test split even though with a small margin as shown in fig. 4.15.Both 10-fold cross validation and test split yield optimum accuracy results at the resample sample size percent of 1000.

Table 4.15: Hybrid KND accuracy

Classifier model rate of accuracy (%)		
Resample (%)	10-fold Cross Validation	Test Split
None	78.57	77.90
100	79.21	78.21
200	81.95	81.99
300	83.21	83.00
400	85.33	83.93
500	85.80	84.77
600	86.50	85.50
700	86.66	86.01
800	86.80	86.54
900	87.60	86.80
1000	87.93	86.95
1100	87.50	86.56
1200	87.39	86.37
1300	87.37	86.26

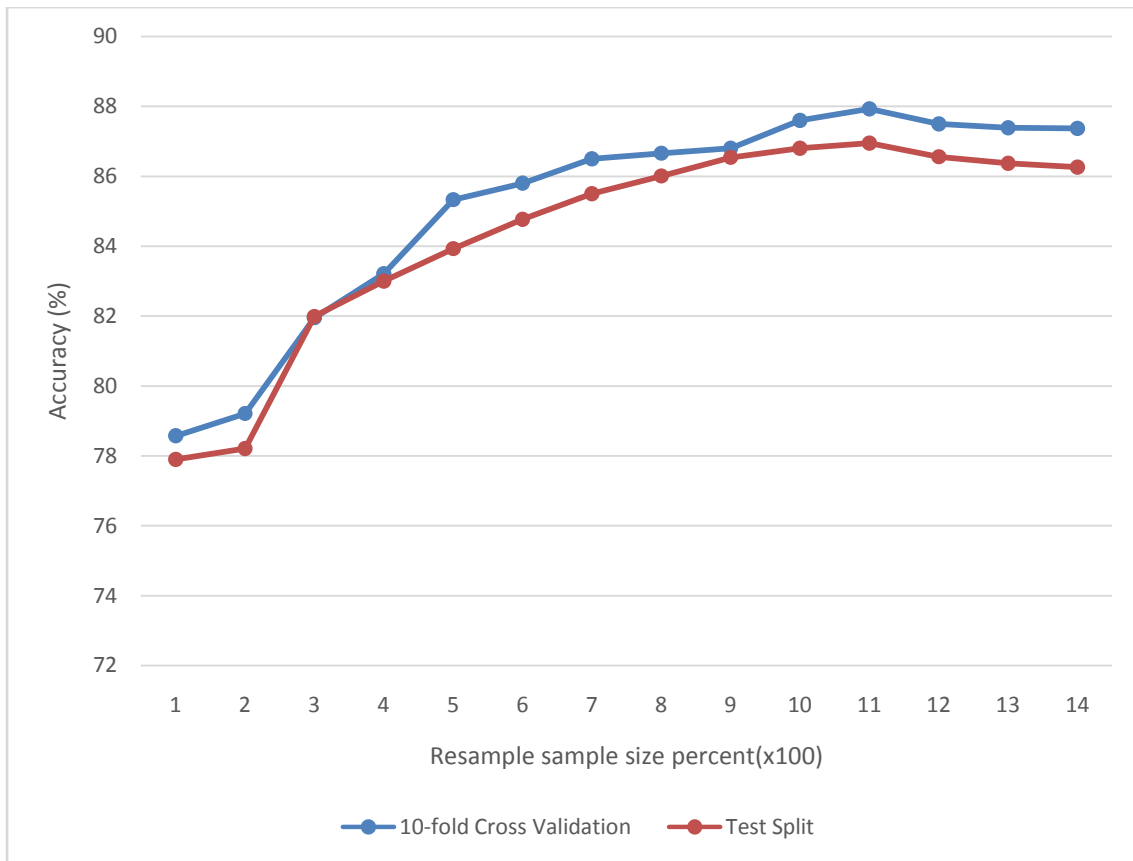


Figure 4.15: Hybrid KND Accuracy

Table 4.16 shows tabulated build time results for 10-fold cross validation and test split test option. Build time for test split is generally longer than build time for 10-fold cross validation and increases with the resample sample size percent as shown in Fig4.16. This is attributed to more instances in resampled datasets that increase the computational cost of the classifier algorithm models. The build time in seconds for the control experiments is 0.01 for 10-fold cross validation, and 0.03 for test split. The build time in seconds at the resample sample size percent of 1000 is 0.03 for 10-fold cross validation and 1.03 for test split respectively. The build time in seconds at the resample sample size percent of 1300 is 0.05 for 10-fold cross validation and 1.58 for test split, respectively.

Table 4.16: Hybrid KND build time

Classifier model Build time (sec)		
Resample (%)	10-fold Cross Validation	Test Split(66/34)
None	0.01	0.03
100	0.01	0.05
200	0.01	0.07
300	0.02	0.14
400	0.02	0.27
500	0.03	0.32
600	0.03	0.42
700	0.01	0.55
800	0.01	0.71
900	0.02	0.85
1000	0.03	1.03
1100	0.03	1.18
1200	0.05	1.35
1300	0.05	1.58

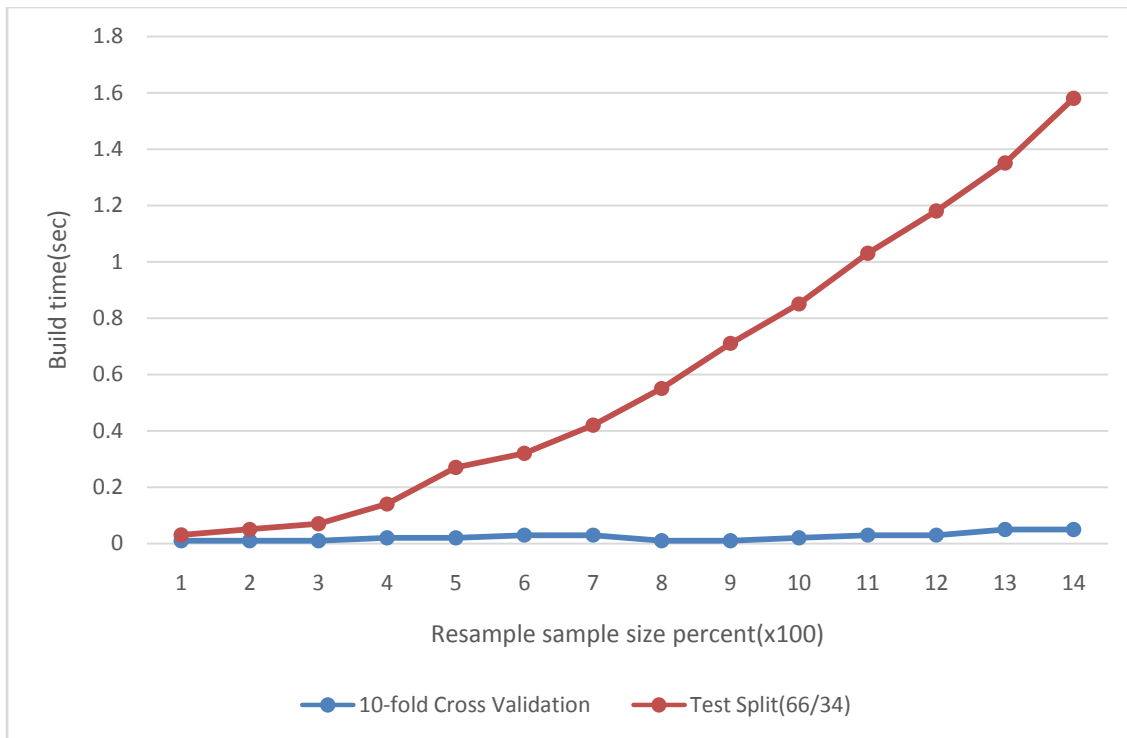


Figure 4.16: Hybrid KND build time

4.2.4 Hybrid KD

Table 4.17 shows tabulated accuracy results for 10-fold cross validation and test split test option. The accuracy rate in percentage for the control experiment is for dataset with no resampling at 77.81 for 10-fold cross validation and 77.40 for test split, respectively. As the resample sample size percent increases so does the accuracy rate to a resample sample size percent of 1000 and accuracy rate in percentage of 88.18 for 10-fold cross validation and 87.66 for test split respectively, and accuracy rate starts to fall with further increase in the resample sample size percentage. At resample sample size percent of 1300, the accuracy rate in percentage is 87.35 for 10-fold cross validation and 86.35 for test split, respectively. This can be attributed to the fact that the dataset is initially imbalanced without resampling, as the dataset is resampled at different rates, it becomes balanced and optimizes performance at a resample sample size percent rate of 1000 and starts to suffer again from overfitting with increasing resample sample size percent eventually affecting the rate of accuracy of the classifier model. The accuracy rates for 10-fold cross validation is generally higher than those for test split even though with a small margin as shown in fig. 4.17. Both 10-fold cross validation and test split yield optimum accuracy results at the resample sample size percent of 1000.

Table 4.17: Hybrid KD Accuracy

Classifier model rate of accuracy (%)		
Resample (%)	10-fold Cross Validation	Test Split
None	77.81	77.40
100	79.85	77.15
200	82.78	82.93
300	84.23	83.63
400	86.72	85.00
500	87.45	86.50
600	87.50	86.87
700	87.54	87.35
800	87.68	87.55
900	87.98	87.62
1000	88.18	87.66
1100	88.08	87.56
1200	87.35	86.44
1300	87.35	86.35

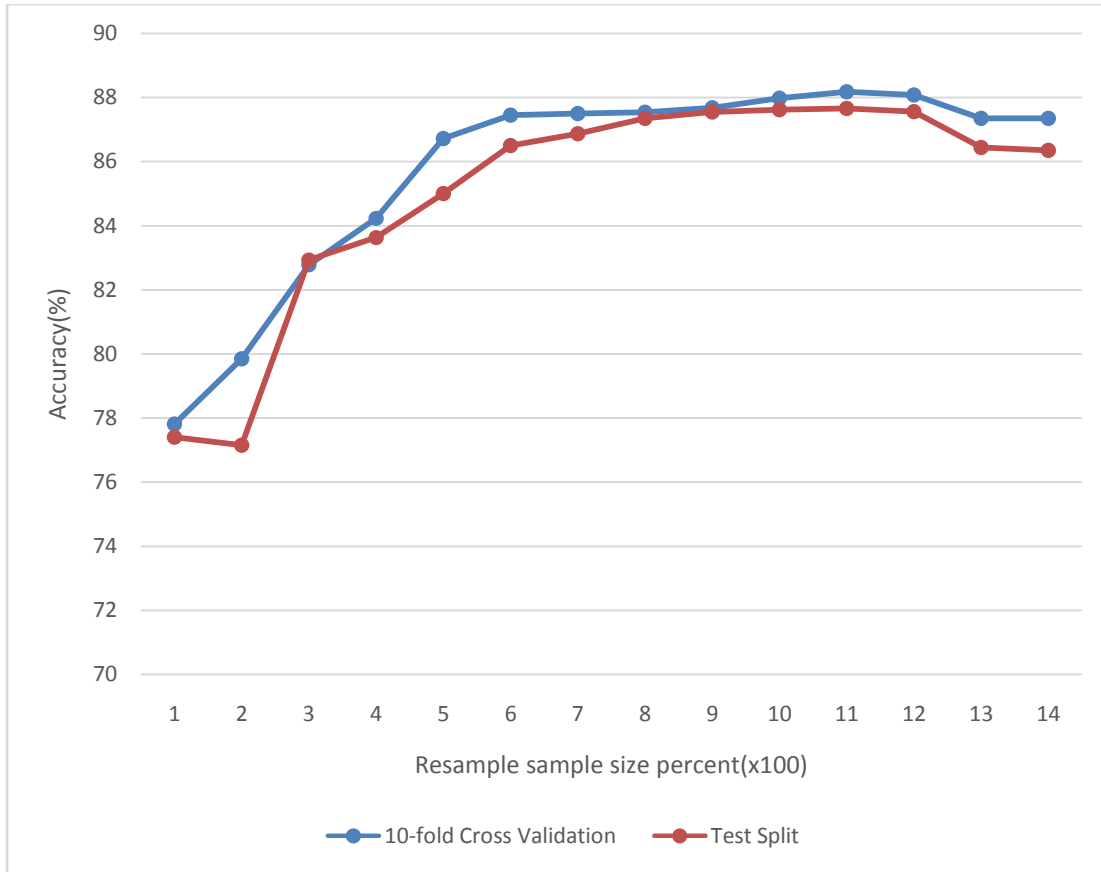


Figure 4.17: KD accuracy

Table 4.18 shows tabulated build time results for 10-fold cross validation and test split test option. Build time for test split is generally longer than build time for 10-fold cross validation and increases with the resample sample size percent as shown in Fig4.18. This is attributed to more instances in resampled datasets that increase the computational cost of the classifier algorithm models. The build time in seconds for the control experiments is 0.01 for 10-fold cross validation, and 0.01 for test split. The build time in seconds at the resample sample size percent of 1000 is 0.03 for 10-fold cross validation and 1.03 for test split respectively. The build time in seconds at the resample sample size percent of 1300 is 0.03 for 10-fold cross validation and 1.65 for test split, respectively.

Table 4.18: Hybrid KD Build Time

Classifier model Build time (sec)		
Resample (%)	10-fold Cross Validation	Test Split(66/34)
None	0.01	0.01
100	0.01	0.03
200	0.01	0.12
300	0.02	0.14
400	0.03	0.23
500	0.03	0.34
600	0.03	0.46
700	0.03	0.60
800	0.03	0.75
900	0.02	0.86
1000	0.03	1.03
1100	0.03	1.20
1200	0.03	1.38
1300	0.03	1.65

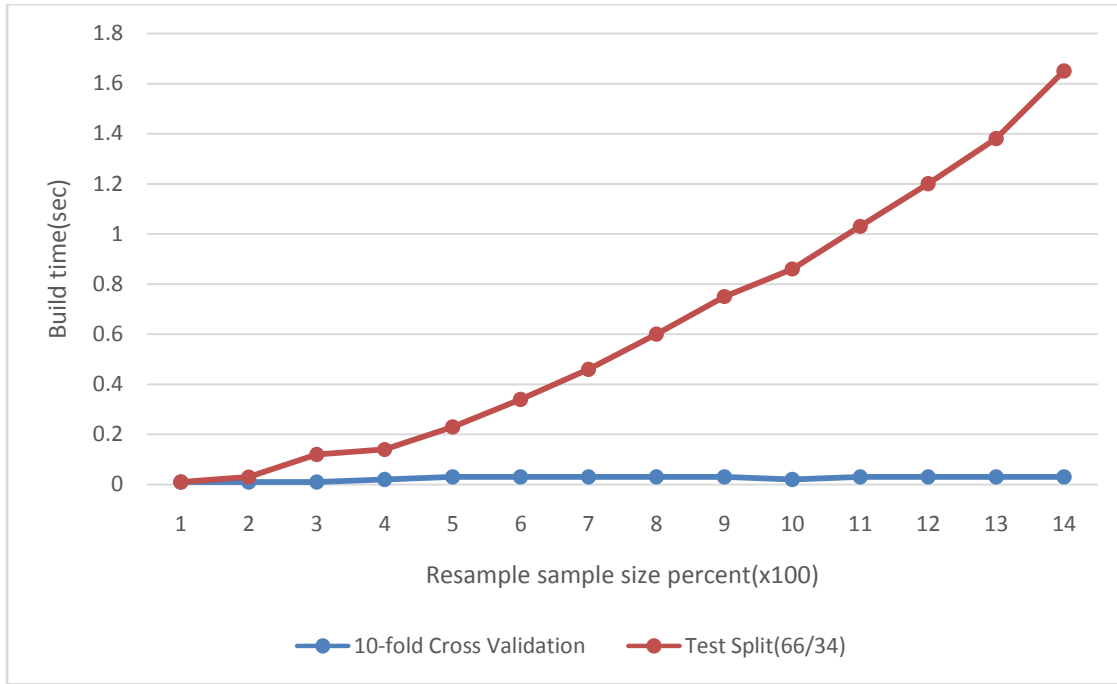


Figure 4.18: Hybrid KD build time

From the experimental results, 10-fold cross validation test option yield higher accuracy rates for all the hybrid classifier algorithm models, as compared to test split test options for a resample sample size percent of 1000. The study establishes that the resample sample size percent for optimum percentage accuracy is at 1000 for the dataset for all the hybrid classifier algorithm models. At resample sample size percent of 1000, Hybrid KNMSD performed worst with the lowest accuracy rate of 87.83%, and longest build time of 577.9 seconds, followed by hybrid KND with accuracy rate of 87.93%, and build time of 0.03 seconds, hybrid KNSD with accuracy rate of 87.98%, and build time of 17.83 seconds, and the best was a hybrid KD with 88.18% accuracy rate, and 0.03 seconds build time. The results fit with the theory that for good ensembles, base learners should be as more accurate as possible (Krogh & Vedelshy, 1995) and reinforces the belief “many could be better than all” theorem may not be the fact (Zhou, Wu, & Tang, 2002). The study demonstrates that hybrid methods combine both selection and fusion techniques. The main idea is to use selection only and only if the best classifier is good enough to classify, otherwise a combination method is used (Vladislav, 2014). The results show that errors made by classifiers are independent and, and therefore the reason for majority vote hybrid outperform the best single classifier (Kim, Kim, Moon, & Ahn, 2011). The study suggests that resample sample size percent affects accuracy and that a resample sample size percent of 1000 for the dataset and a combination of

KNN and DT through bagging majority voting yields optimum hybrid rate of accuracy. The study results are constrained by bagging as an ensemble combination technique.

4.3 Compare Performance of Classifier Algorithm Models in the Identification of Terrorist Groups

The study sought to compare optimum performance levels of the classifier algorithm models and identify the best classifier algorithm model for the identification of terrorist groups in the aftermath of an attack. This was done using test split/hold out validation and 10-fold cross validation. Hold out validation is also called test split and avoids the overlap between training data and test data, yielding more accurate estimate for the generalization performance of the algorithm. The downward is that the procedure does not use all the available data and the results are highly dependent on the choice of the training/test split(Dietterich, 1998). The instances chosen for inclusion in the test set may be too easy or too difficult to classify and this can skew the results. Furthermore, the data in the test set may be valuable for training and if it is held out prediction performance may suffer, again leading to skewed results.

Cross validation is a statistical method of evaluating and comparing learning algorithms. It is used to evaluate learning algorithms as follow: in each iteration, one or more learning algorithms use k-1 fold of data to learn one or more models. And subsequently the learned models are asked to make predictions about the data in the validation fold. The performance of each learning algorithm on each fold can be tracked using some predetermined metric like accuracy. Upon completion k samples of the performance metric will be available for each algorithm. These samples can be used in a statistical hypothesis to show that an algorithm is superior to another (Refaeilzadeh, Tang, & Liu, 2007). Cross validation is used to gauge the generalizability of an algorithm and to compare the performance of two or more different algorithms and to find the best algorithm for the available data, or alternatively compare more variants of a parameterized model (Refaeilzadeh, Tang, & Liu, 2007). The above two goals are highly related, since the second goal is automatically achieved if one knows the accurate estimates of performance. Given a sample N data instances and a learning algorithm A, the average cross validated accuracy of A on these N instances may be taken as estimates of accuracy of A on unseen data when A is trained on all N instances. Alternatively if the end goal is to compare two learning algorithms, the performance samples obtained through cross validation can be used to perform two sample statistical hypothesis test, comparing a pair of learning algorithms(Dietterich, 1998).Currently, cross-validation is widely accepted in data

mining and machine learning community and serves as a standard procedure for performance estimation and model selection (Refaeilzadeh, Tang, & Liu, 2007). 10-fold cross validation is a good compromise in data mining. It is particularly attractive because it makes predictions using 90% of data, making it more likely to be generalizable to the full data (Refaeilzadeh, Tang, & Liu, 2007).

4.3.1 Compare Error Rate Percentage and Build Time for 10-Fold Cross Validation

Table 4.19 shows results of 10-fold cross validation percentage error rate comparison. The percentage error rate generally decreases as the resample sample size percent increases to 1000 and starts to rise, hybrid KD generally outperforms all other classifiers and NB performs worst for the dataset as shown in fig.4.19. Without resample, i.e. the control experiment yields the following percentage error rates with MLP yielding the highest 31.38, KNN 27.3, SVM 23.85, NB 23.34, hybrid KNMSD 22.83, DT 22.58, hybrid KD 22.19, hybrid KNSD 22.07 and hybrid KND with the lowest 21.43. The optimum performance rate for every classifier algorithm for the dataset was achieved at resample sample size percent of 1000 with respective classifier algorithm model yielding the lowest error rate percentages with NB yielding the highest error rate of 16.39, SVM 13.83, MLP 12.91, DT 12.47, KNN 12.27, hybrid KNMSD 12.27, hybrid KND 12.07, hybrid KNSD 12.02 and hybrid KD with the lowest error rate of 11.82. Error rate percentage at resample sample size percent of 1300 are NB 17.39, SVM 14.4, MLP 13.59, DT 13.37, hybrid KNMSD 12.75, hybrid KNSD 12.67, hybrid KND 12.63, hybrid KD 12.65, KNN with the lowest 12.48.

Table 4.19: 10-foldcross validation percentage error rate comparison

Resample (%)	DT	KNN	NB	SVM	MLP	KNMSD	KNSD	KND	KD
None	22.58	27.3	23.34	23.85	31.38	22.83	22.07	21.43	22.19
100	20.92	26.66	20.92	19.9	19.77	20.4	22.07	20.79	20.15
200	17.54	24.11	21.05	16.39	17.73	16.52	16.84	18.05	17.22
300	16.28	19.9	19.43	16.07	15.47	15.77	16.2	16.79	15.77
400	14.41	17.38	19.32	15.98	14.8	14.22	14.29	14.67	13.28
500	13.69	13.76	18.7	15.45	14.1	13.6	13.41	14.2	12.55
600	13.6	13.63	18.09	15.11	13.88	13.18	13.14	13.5	12.5
700	13.58	13.56	17.55	14.58	13.66	13.07	13.07	13.34	12.46
800	13.11	13.34	17.4	14.52	13.27	12.96	12.73	13.2	12.32
900	12.66	12.44	17.04	14.37	13.23	12.46	12.4	12.4	12.02
1000	12.47	12.27	16.39	13.83	12.91	12.17	12.02	12.07	11.82
1100	12.55	12.37	17.25	13.99	13.45	12.53	12.38	12.5	11.92
1200	12.87	12.46	17.34	14.04	13.53	12.56	12.5	12.61	12.65
1300	13.37	12.48	17.39	14.4	13.59	12.75	12.67	12.63	12.65

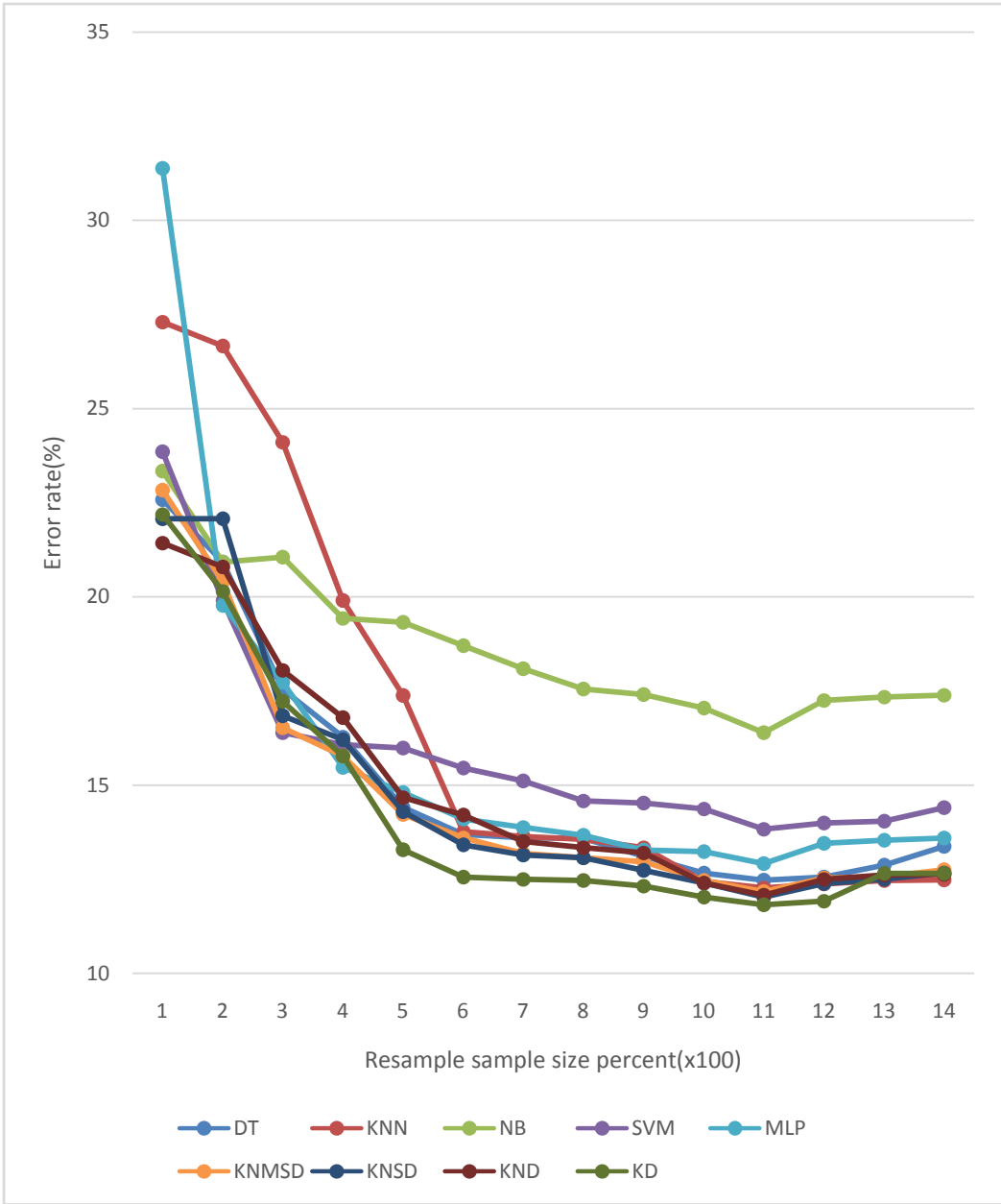


Figure 4.19: 10-fold cross validation error rate comparison

Table 4.20 shows 10- fold cross validation time comparison in seconds as follows without resampling, KNN, NB, DT, hybrid KD and hybrid KND all have shortest build time of 0.01 each. Hybrid of KNSD takes 1.97, SVM takes 4.27, MLP takes 54.34 and hybrid KNMSD the longest 66.91 seconds. Generally, at resample sample size percent of 1000, KNN& NB 0.01 seconds, DT 0.02 seconds, Hybrid KD and hybrid KND 0.03 seconds, hybrid of KNSD 17.83 seconds, SVM 21.03 seconds , MLP 530.4 seconds and hybrid KNMSD the longest 577.9 seconds. Fig. 4.20 shows how build time for the classifiers with change in resample sample size percent.

Table 4.20: Comparison of build time in seconds

Resample (%)	DT	KNN	NB	SVM	MLP	KNMSD	KNSD	KND	KD
None	0.01	0.01	0.01	4.27	54.34	66.91	1.97	0.01	0.01
100	0.01	0.01	0.01	4.05	53.42	56.67	1.94	0.01	0.01
200	0.02	0.01	0.02	3.76	107.4	113.2	3.37	0.01	0.01
300	0.02	0.01	0.01	4.77	157.8	168.2	4.85	0.02	0.02
400	0.01	0.01	0.01	7.33	212.9	222.6	7.58	0.02	0.03
500	0.03	0.01	0.01	8.74	270.6	276.8	9.63	0.03	0.03
600	0.03	0.01	0.01	18.82	318.4	335.1	10.42	0.03	0.03
700	0.02	0.01	0.01	21.22	365	393.8	11.37	0.01	0.03
800	0.04	0.01	0.01	13.32	413.1	437.5	13.34	0.01	0.03
900	0.04	0.01	0.01	14.16	473.8	516.7	13.34	0.02	0.02
1000	0.02	0.01	0.01	21.03	530.4	577.9	17.83	0.03	0.03
1100	0.02	0.01	0.01	30.97	569.1	622.2	22.71	0.03	0.03
1200	0.03	0.01	0.01	25.66	630.4	672.1	24.8	0.05	0.03
1300	0.04	0.01	0.01	36.58	674.8	741.8	26.27	0.05	0.03

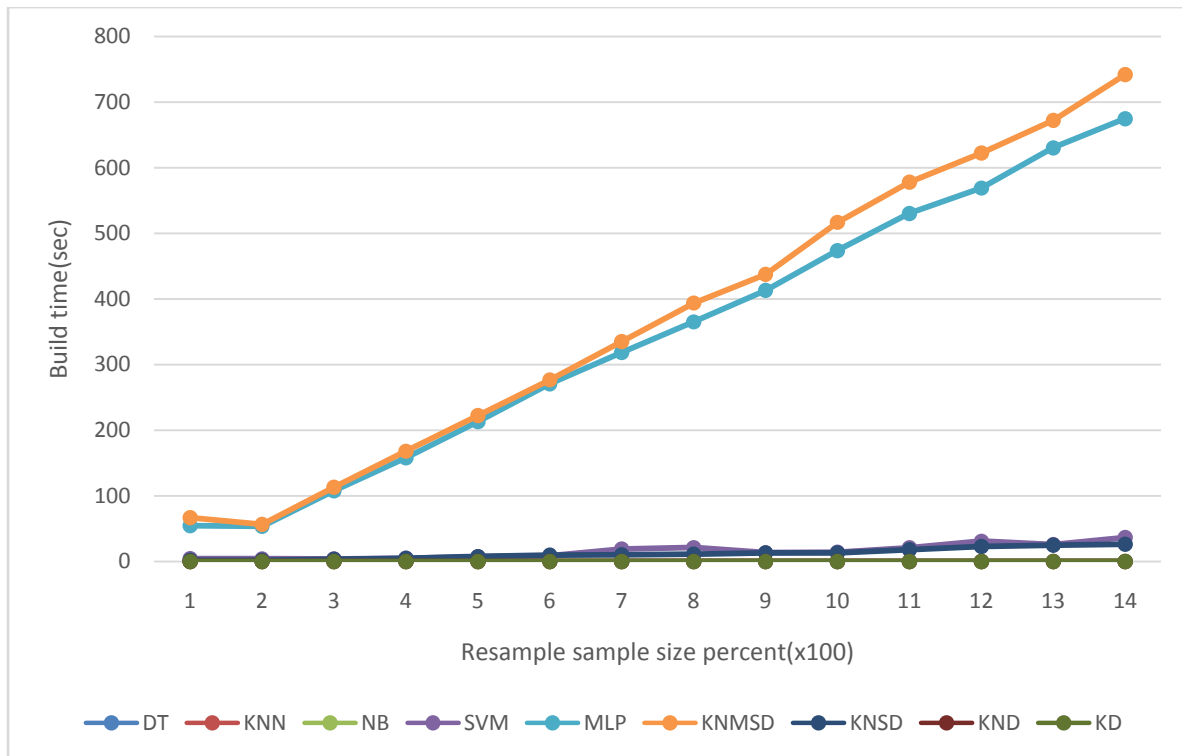


Figure 4.20: 10-fold cross build time comparison

4.3.2 Compare Error Rate and Build Time for Test Split

Table 4.21 shows results of test split percentage error rate comparison. The percentage error rate generally decreases as the resample sample size percent increases to 1000 and starts to rise, hybrid KD generally outperforms all other classifiers and NB performs worst for the dataset as shown in fig.4.20 .Without resample, i.e. the control experiments yield the following percentage error rates with MLP yielding the highest 30.34, KNN 29.59, hybrid KNMSD 22.47, DT 22.47, NB 22.85, hybrid KD 22.60, hybrid KNSD 22.47, SVM 22.10 and hybrid KND 22.10. The optimum accuracy rates for every classifier algorithm for the dataset was achieved at resample sample size percent of 1000 with respective classifier algorithms yielding the following error rate percentages NB yielding the highest error rate of 17.14, SVM 13.43, DT 13.32, hybrid KND 13.05, KNN 12.94, hybrid KNSD 12.79, hybrid KNMSD 12.64, MLP 12.42, hybrid KD with the lowest 12.34. Error percentage rates at resample sample size percent of 1300 are NB 17.98 , SVM 15.00, MLP 14.50, DT 14.60, hybrid KNMSD 13.80, hybrid KNSD 13.76, hybrid KND 13.74, hybrid KD 13.65 and KNN with the lowest 13.30.

Table 4.21: Test split percentage error rate comparison

Resample (%)	DT	KNN	NB	SVM	MLP	KNMSD	KNSD	KND	KD
None	22.47	29.59	22.85	22.1	30.34	22.47	22.47	22.1	22.6
100	22.15	28.84	22.85	22.85	23.6	24.72	22.1	21.79	22.85
200	17.26	24.58	21.01	18.2	18.39	17.07	17.26	18.01	17.07
300	16.45	20.62	18.87	17.62	16	16	16.25	17	16.37
400	15.38	19.51	18.86	17.45	14.73	15.1	15.7	16.07	15
500	15.38	17.63	18.5	16.04	14.1	14.63	15	15.23	13.5
600	14.93	16.62	18.05	15.38	13.82	14.17	14.82	14.5	13.13
700	13.99	14.04	17.95	15.36	12.74	13.34	14.09	13.99	12.65
800	13.98	14.02	17.6	15.01	12.68	13.13	13.56	13.46	12.45
900	13.5	12.99	17.38	14.46	12.54	13.01	12.96	13.2	12.38
1000	13.32	12.94	17.14	13.43	12.42	12.64	12.79	13.05	12.34
1100	13.45	12.98	17.23	13.61	13.3	12.66	13.08	13.44	12.44
1200	14.1	13.19	17.32	14.11	14.32	13.66	13.41	13.63	13.56
1300	14.6	13.3	17.98	15	14.5	13.8	13.76	13.74	13.65

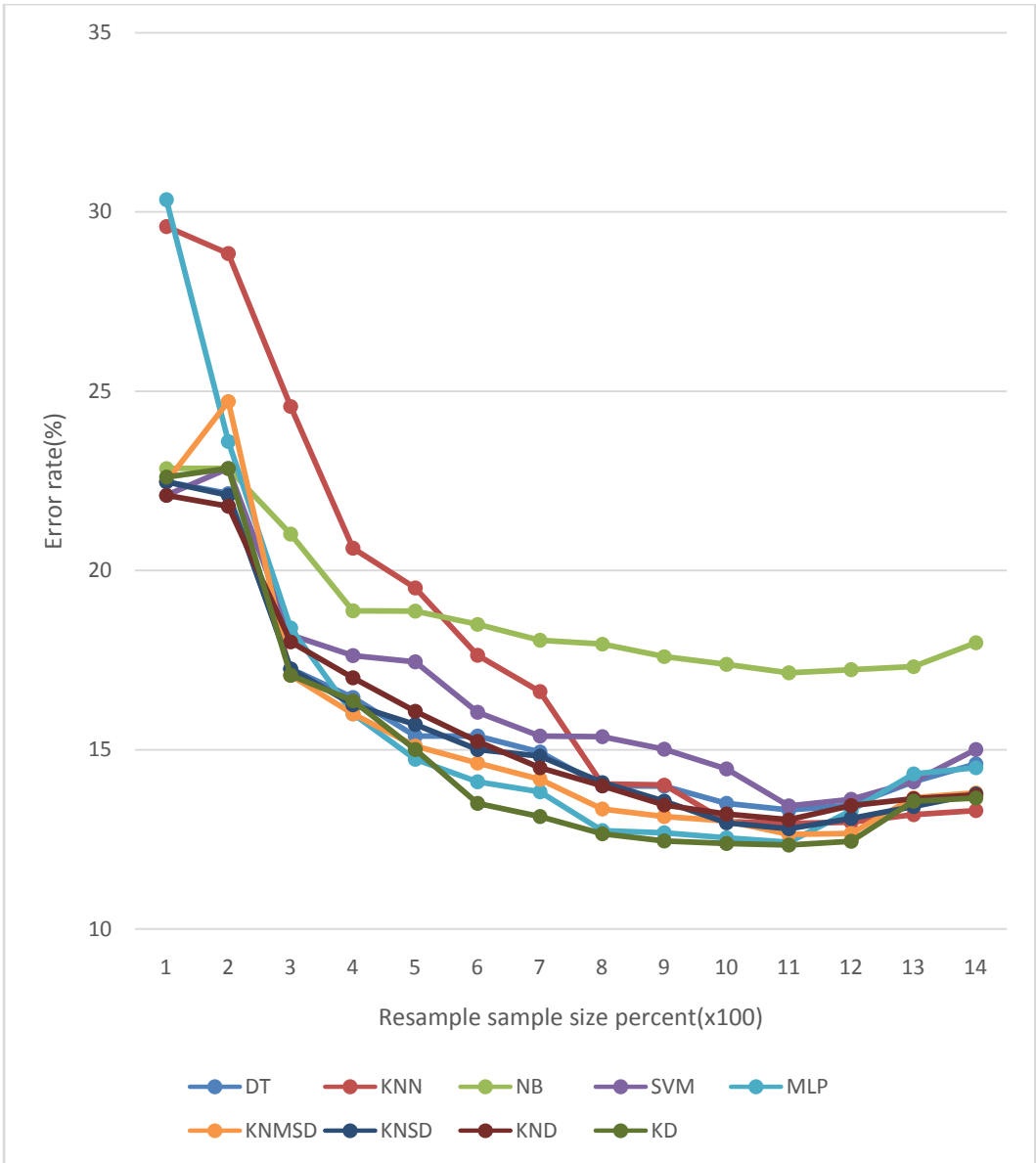


Figure 4.21: Test Split Error rate comparison

Table 4.22 shows test split build time comparison in seconds as follows without resample, DT, NB, and hybrid KD all have shortest build time of 0.01 seconds each. KNN 0.02 seconds, hybrid KND 0.03 seconds, hybrid KNSD 4.02 seconds, SVM 4.30 seconds, MLP 54.06 seconds, and hybrid KNMSD 57.70seconds. Generally, at resample sample size percent of 1000, DT 0.05 seconds, ND 0.06 seconds, KNN 0.89 seconds, hybrid KD & hybrid KND 1.03 seconds each, SVM 18.37 seconds, hybrid KNSD 21.61 seconds, MLP 543.30 seconds, and hybrid KNSD 575.60 seconds. Fig. 4.22 shows build time for the classifiers with change in resample sample size percent.

Table 4.22: Test Split build time Comparison

Resample (%)	DT	KNN	NB	SVM	MLP	KNMSD	KNSD	KND	KD
None	0.01	0.02	0.01	4.30	54.06	57.70	4.02	0.03	0.01
100	0.01	0.02	0.01	4.22	52.67	58.50	3.61	0.05	0.03
200	0.02	0.08	0.01	4.19	106.90	114.30	4.26	0.07	0.12
300	0.03	0.13	0.02	5.29	155.90	168.40	5.37	0.14	0.14
400	0.03	0.19	0.03	6.55	213.90	233.80	6.58	0.27	0.23
500	0.03	0.28	0.03	7.89	269.00	279.80	8.46	0.32	0.34
600	0.02	0.39	0.05	11.99	319.60	341.20	10.02	0.42	0.46
700	0.02	0.51	0.05	13.29	371.30	389.90	11.05	0.55	0.60
800	0.04	0.65	0.05	14.04	421.80	465.10	14.72	0.71	0.75
900	0.06	0.77	0.05	14.44	466.90	513.40	15.58	0.85	0.86
1000	0.05	0.89	0.06	18.37	543.30	575.60	21.61	1.03	1.03
1100	0.06	1.01	0.08	23.72	567.30	648.10	22.52	1.18	1.20
1200	0.04	1.26	0.08	24.59	622.9	673.00	26.25	1.35	1.38
1300	0.02	1.45	0.08	28.53	698.40	730.40	27.17	1.58	1.65

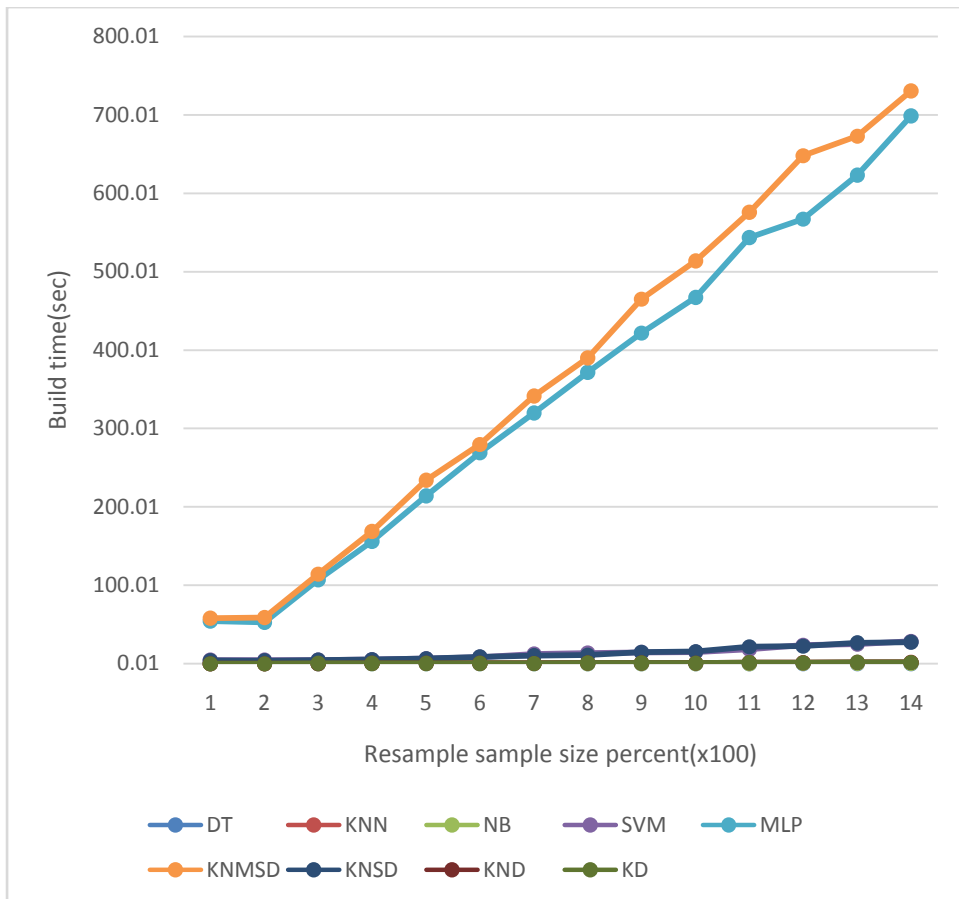


Figure 4.22: Test split build time comparison

4.3.3 Statistical Test of Significance for Optimum Accuracy Averages

The comparison of accuracy for statistical significance was calculated between the control experiment (non-resampled dataset) accuracy averages and the resampled sample size percent of 1000 (optimum) accuracy averages for both base classifiers and hybrid classifier algorithms as shown in table 4.23 for 10-fold cross validation and table 4.25 for test split respectively. Two null hypotheses as set out in the experimental design were tested.

$H_0 : \mu_{.1} = \mu_{.2}$ (treatment population means are equal)

$H_1 : \mu_{.1} \neq \mu_{.2}$.

$H_0 : \mu_{.1} = \mu_{.2} = \dots = \mu_{.9}$ (block population means are equal)

$H_1 : \mu_{.1} \neq \mu_{.2} \neq \dots \neq \mu_{.9}$

When comparing more than two algorithms, on each dataset there is no win/loss/tie; instead, each algorithm assumes a rank between 1 and L in terms of its performance (averaged over different folds)(Irsoy, Yildiz, & Alpaydin, 2012). We then use non-parametric tests to check for significant difference in average ranks over M data sets. Friedman’s test is a non-parametric version of ANOVA and uses ranks instead of absolute performance(Demsar, 2006).

Table 4.23: 10-fold cross validation optimum accuracy comparison between control and optimum

Classifier	Control	1000
DT	77.42	87.53
KNN	72.70	87.73
NB	77.66	83.61
SVM	76.15	86.17
MLP	68.62	87.09
KNMSD	77.17	87.83
KNSD	77.93	87.98
KND	78.57	87.93
KD	77.81	88.18

Table 4.24: AVOVA test for 10-fold cross validation accuracy

Anova: Single Factor

SUMMARY

<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>
Column 1	9	683.03	75.89222	10.33739
Column 2	9	784.05	87.11667	2.098375

ANOVA

<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	566.9466889	1	566.9467	91.17999	5.2E-08	4.493998
Within Groups	99.48615556	16	6.217885			
Total	666.4328444	17				

The F-value is greater than the F-critical value for the alpha level selected (0.05) as shown in table 4.24. Therefore, we have evidence to reject the null hypothesis and say that at least one of the samples have significantly different means. Another measure is the p-value. The p-value 5.2E-08 is less than the alpha level 0.05, we therefore reject the null hypotheses.

Table 4.25: Test split comparison between control and optimum accuracy

Classifier	Control	1000
DT	77.53	86.68
KNN	70.41	87.06
NB	77.15	82.86
SVM	77.90	86.57
MLP	69.66	87.58
KNMSD	77.53	87.36
KNSD	77.53	87.21
KND	77.90	86.95
KD	77.40	87.66

Table 4.26: ANOVA test for test split

Anova: Single Factor**SUMMARY**

<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>
Column 1	9	683.01	75.89	11.1075
Column 2	9	780.2	86.68889	2.221861

ANOVA

<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	524.772	1	524.772	78.73926	1.41E-07	4.493998
Within Groups	106.6349	16	6.664681			
Total	631.4069	17				

The F-value is greater than the F-critical value for the alpha level selected(0.05) as shown in table 4.26. Therefore, we have evidence to reject the null hypothesis and say that at least one of the samples have significantly different means. Another measure is the p-value. The p-value 1.41E-07 is less than the alpha level 0.05, we therefore reject the null hypotheses.

Generally, the hybrid classifier algorithm models outperform all base classifiers at resample sample size percent rate of 1000 for both 10-fold cross validation and test split test option. The study confirms that hybrid classifiers improve performance accuracy perform better than base classifiers(Gundabathula & Vaidhehi, 2018).The accuracy rates improve with the resample sample size percent up to a resample sample size percent of 1000 and starts to decline. Class imbalance in a dataset affects accuracy rates and optimum accuracy is attained wit balanced datasets. The study is consistent with the opinion that data imbalance affects classification accuracy (Garcia, Marques, & Sanchez, 2012). KNN, DT, NB, hybrid KD, hybrid of KND have generally taken relatively shorter build time of between 0.01 and 0.03 seconds. SVM has a build time of 21.03 seconds MLP hybrid KNMD have the longest build time. The results confirm that better differentiation of classifiers can be done by examining the build time (Williams, Zander, & Armitage, 2006). The inclusion of either an either MLP or SVM increases the overall build time of a hybrid classifier. Increased computation is a weakness of ensembles, because in order to classify an input query, all component classifiers must be processed (Dietterich, 2000).the study is limited by rate of accuracy, and build time as performance measure indicators. Performance of various combinations of the hybrid perform differently in varying situations, confirming that generally, there is no ensemble method which outperforms other ensembles consistently(Ting & Witten, 1999). DT performs better than NB confirming the results of a classification study between NB and DT (Rizwan, Masrah, Aida, Payam, & Nasim, 2013). The research establishes that the hybrid KD outperforms all other classifier algorithm models under the study in the identification of terrorist groups in the aftermath of an attack for the available dataset.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

This chapter consists conclusion to the study, recommendation, and future work

5.1 Conclusion

In this study the problem of terrorism is addressed by using machine learning techniques. The GTD for sub Saharan Africa for the period 1999-2017 dataset is used to build and evaluate machine learning classifier algorithms KNN, NB, DT, SVM and MLP. The dataset was processed by using various pre-processing techniques and the class imbalance problem solved and feature selection were done by using WEKA Resample filter and attribute selector. The hybrid classifier algorithm models were built by combining base classifier algorithm models using bagging and majority voting technique. The models were evaluated on classification accuracy and build time using 10-fold cross validation and Test split option. The key question when dealing with classification is not whether a learning algorithm is superior to others, but under which conditions a particular method can significantly outperform others on a given application problem (Kotsiantis, Zaharakis, & Pintelas, 2004). The performance of hybrid/ensemble depends on accuracy of base classifiers, diversity among the base classifiers, decision making strategy (aggregation technique) and the number of the base classifiers among other factors (Vladislav, 2014).

The study sought to establish what approach to be used to build and evaluate base classifier algorithm models of KNN, NB, DT, SVM and MLP. The study concluded that for WEKA default configuration set up of the base classifier algorithm models the resample size percent for optimum performance is 1000 which is a balanced dataset on both 10-fold cross validation and test split test options.

The study sought to establish what approach to be used to build and evaluate hybrid classifier algorithm models built from the base classifier algorithm models and concluded that hybrid KD (combination of KNN and DT using bagging and majority voting) at a resample sample size percent of 1000 achieved optimum performance on 10- fold cross validation and test split test options.

The study sought to find the outcome of analysis of performance of classifier algorithm models. The study concluded that hybrid KD outperformed all other classifier algorithm models in the identification of terrorist groups in the aftermath of an attack with an accuracy

percentage of 88.18 and build time of 0.03 seconds on 10-fold cross validation, accuracy percentage of 87.66 and build time of 1.03 seconds for test split.

5.2 Recommendation

The performance of a hybrid machine learning classifier model depends largely on the accuracy and diversity of base classifiers and availability of balanced terrorist dataset from GTD. The researcher recommends that the performance of the ensemble members in a hybrid and data imbalance issues be fully addressed for better accuracy rates. Imbalance dataset means that one of the two classes has very a smaller number of samples compared to number of samples in the other class, $|C2| \ll |C1|$. Then C2 is called the minority class, and C1 is called the majority class. The minority class is of our interest. The classification in case of unbalanced dataset is biased towards majority class. The approach to solve this problem is sampling based approach (Verma, 2019). Sampling based approach also known as data level approach works by artificially balancing the instances of class in the dataset. To artificially balance the class we apply resampling technique, such as random under sampling the majority class, random oversampling of minority class, and Synthetic Minority Over-Sampling Technique (SMOTE)(Verma, 2019). RANDOM UNDERSAMPLING OF MAJORITY CLASS balances the class distribution in the dataset by randomly throwing away some data samples from majority class. Although it balances class distribution, but it leads to losing some important characteristics in dataset, due to removal of some samples, this is a disadvantage of this approach (Verma, 2019). RANDOM OVERSAMPLING OF MINORITY CLASS balances the class distribution by the random replication of minority class instances, to increase their number. There is no information loss in this case. The problem with this approach is that it leads to overfitting (Verma, 2019). SYNTHETIC MINORITY OVERSAMPLING TECHNIQUE (SMOTE) reduces the problem of overfitting a method of by creating synthetic instances of minority class. This technique is known as the synthetic minority over-sampling technique (SMOTE). In this the training set is altered by adding synthetically generated minority class instances, causing the class distribution to become more balanced. (Verma, 2019).

5.3 Future Work

This work can further be extended by applying other approaches of solving class imbalance problem in the terrorist dataset, using other forms of ensemble combination techniques such as stacking and boosting or in cooperating other genetic algorithm machine learning classifiers.

REFERENCES

- Alexandrie, G. (2017). Surveillance cameras and crime: a review of randomized and natural experiments. *Journal of Scandinavian Studies in Criminology and Crime Prevention*, 210.
- Asmita , S., & Shukla, K. K. (2014). Review on the Architecture, Algorithm and Fusion Strategies in Ensemble Learning. *International Journal of computer applications*, 21-28.
- Baba, N. M., Makhtar, M., Fadzili, S. A., & Awang, M. K. (2015). Current issues in ensemble methods and its application. *Journal of Theoretical and Applied Information Technology(JATIT)*, 266-272.
- Banfiel, R. E., Hall, L. O., Bowyer, K., & Kegelmeyer, w. P. (2002). Ensemble Diversity Measures and Their Application to thinning. *Journal of Information fusion*, 49-62.
- Barth, J. R., Tong, L., McCarthy, D., Phumiwasana, T., & Yago, G. (2006). *Economic Impact of global Terrorism: From Munich to Bali*. 203-212: Milken Institute.
- Batch, V., & Aravindan, D. J. (2011). A classification based dependent approach for suppressing data. *IJCA proceedings on Wireless information Networks & Business Information Systems*. Lahore: WINBIS 2012.
- Bauer, E., & Kohavi, R. (1999). An emperical comparison of voting classification algorithms: Bagging, Boosting ans variants. *Machine Learning*, 105-139.
- Benoit, F., Van Heeswijk, M., Miche, Y., Verleysan, M., & Lendasse, A. (2013). Feature selection for non-linear models with extreem learning machines. *Journal of Neurocomputing*, 111-124.
- Boukaert, B. R. (2003). Choosing between two learning algorithms based on calibrated tests. *Journal of Machine Learning*, 51-58.
- Bouziane, H., Messabih, B., & Chouarfia, A. (2011). profiles and majority voting based ensemble method for protein secondary structure prediction. *Evolutionary Bioinformatics*, 171-189.

- Bradley, P. S., & Mangasarian, O. L. (1998). Feature Selection by concave minimization and support vector machines in. *International Conference on Machine Learning(ICML-1998)* (pp. 82-90). Madison, Winsconsin, USA: Morgan Kaufmann.
- Breiman. (1996). Bagging predictors. *Machine Learning*, 123-140.
- Breiman. (2001). Random forests. *Machine Learning*, 5-32.
- Brown, G., Wyatt, J. L., & Tino, P. (2005). Managing Diversity in regression Ensembles. *The Journal of machine learning Research*, 1621-1650.
- Camargo, L. S., & Yoneyama, T. (2001). Specification of training sets and the number of hidden neurons for Multi-layer perceptrons. *Journal of Neural Networks*, 2673-2680.
- Castellano, G., Fanelli, A., & Pelillo, M. (1997). An iterative pruning algorithm for feedforward neural networks. *IEEE Trans neural Networks journal*, 519-531.
- Chawla, N. V. (2005). *Data mining for imbalanced datasets: An overview* . In *Data mining and knowledge discovery handbook*. Springer US.
- Chen, D., & Liu, Z. (2010). An optimized algorithm of decision tree based on rough sets model. *international conference on Electrical and Control Engineering*. NY: Springer.
- Cortes, C., & Vapnic, V. (1995). Support vector networks. *Journal of machine learning*, 273-279.
- Cortizo, J. C., & Giraldez, I. (2006). Multi criteria wrapper improvement to Naive Bayes learning. *LNCS*, 419-426.
- Demsar, J. (2006). Statistical comparison of classifiers over multiple data sets. *Journal of Machine Learning Research*, 1-30.
- Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computing*, 1895-1924.
- Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of Decision trees: bagging, boosting, and randomization. *Mach Learn*, 139-159.

- Efron, B. (1983). Estimating the error rate of a prediction rule: improvement on cross-validation. *Journal of American Statisticians Association* , 316-331.
- Farysal, G., Wasi, B. H., & Usman, Q. (2014). Terrorist group prediction using data classification. *International Conferences of Artificial Intelligence and Pattern Recognition* (pp. 17-19). Malaysia: Researchgate.
- Foster, D. (2017). *Global Terrorist Operations*. 25-30: Springer.
- Freitas, C. O., deCarvalho, J. M., & Oliveira, J. J. (2007). Confusion matrix disagreement for multiple classifiers. Progress in Pattern Recognition. *Image Analysis and Application*.
- Freund, Y., & Schapire, R. E. (1997). A decision theoretic generalization of online learning and an application to Boosting. *Journal of Computer and System Sciences*, 119-139.
- Gama, J., & Brazdil, P. (2000). Cascade generalization. *Machine Learning*, 315-343.
- Garcia, V., Marques, A., & Sanchez, J. (2012). *Improving Risk Prediction by Pre processing imbalanced credit data*.
- Geisser, S. (1975). The predictive sample reuse method with applications. *Journal of American association of Statisticians*, 320-328.
- Genton, M. (2001). Classes of kernels for machine learning: a statistics perspective . *journal of machine learning research*, 299-322.
- Gikaru, J. W. (2012). *Predicting Recidivism among inmates population using artificial intelligent(AI) techniques: A case study of Kenya Prisons Department*. Nairobi: UON.
- GTI. (2018). *Global Terrorism Index*. 8-12: Institute for Economics and Peace.
- Gundabathula, V. T., & Vaidhehi, V. (2018). An Efficient Modelling of Terrorist Groups in India using Machine Learning Algorithms. *India Journal of Science and Technology*, 1-3.
- Guo, G., Wang, H., Bell, D., & Greer, K. (2003). KNN model-based approach in classification. *Lecture notes computer science*(2888), 986-996.

- Hall, L., Bowyer, K., Kegelmeyer, W., Moore, T., & Chao, C. (2000). Distributed Learning on very Large datasets. *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 79-84.
- Han, J., & Kamber, M. (2006). *Data Mining: Concepts and techniques*. San Francisco: Morgan Kaufmann.
- Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE trans pattern analysis machine intelligence*, 832-844.
- Hongbo, D. (2010). *Data mining techniques and applications: An introduction*. LA: Cengage learning EMEA.
- Hoque, N., Bhattachryya, D. K., & Kalita, J. K. (2014). MIFS-ND: A mutual information based feature selection method. *Journal of Expert Systems with Applications*, 6371-6383.
- Hui, Z. (2013). Intrusion Detection ensemble Algorithm based on Bagging and Neighborhood Rough set. *International Journal of Security and its Applications IJSIA*, 193-204.
- Irsoy, D., Yildiz, D. T., & Alpaydin, E. (2012). Design and Analysis of Classifier learning Experiments in Bioinformatics: Survey and case studies. *Journal of IEEE*, 1-12.
- Isson, J. P. (2012, Decemeber 25 November 2018). *What is predictive analystics*. Retrieved from The Knowledge Exchange: <http://www.sas.com/knowledge-exchange/businessanalytics/uncategorized/what-is-predictive-analytics/index.html>
- Jantan, H., Hamdan, A. R., & Othman, Z. A. (2009). *Classification for talent management using decision tree induction techniques*.
- Japkowicz, N., & Stephen, N. (2002). The class imbalance problem: a systematic study. *Jouran of Intelligence and Data Analysis*, 40-49.
- Kainulainen, L. (2010). Ensemble of locally linear models: Application to bankruptcy prediction. *Data Mining*, 280-286.

- Kalpana, R., & Bansa, K. L. (2014). A comparative study of data mining tools. *International Journal of Advanced Research in Computer Science and Software Engineering*, 50-60.
- Kavoc, S. (2012). *Suitability analysis of data mining tools and methods*. Retrieved from Degree thesis: Available http://is.muni.cz/th/255695/fi_b/suitability_analysis_of_data_mining_tools.pdf [Accessed on 10 May 2017]
- Keerthi, S., & Gilbert, E. (2002). Convergence of a generalized SMO algorithm for SVM classifier design. *Journal of machine learning*, 351-360.
- Khorsid, M. M., Abou, T. H., & Soliman, G. M. (2015). Hybrid classification Algorithms for Terrorism prediction in Middle East and North Africa., *International Journal of Emerging Trends & technology in Computer Science*, 23-29.
- Kiage, B. N. (2015). *A datamining approach for forecasting cancer threats*. Nairobi: JKUAT.
- Kim, H., Kim, H., Moon, H., & Ahn, H. (2011). A weight adjusted voting algorithm for ensemble of classifiers. *Korean Statistical Society*, 437-449.
- Kim, Y. S., & Street, W. N. (2002). Evolutionary model selection in unsupervised learning. *Journal of Intelligent Data Analysis*, 531-556.
- Kohavi, R. (1995). A study of cross validation and bootstrap for accuracy estimation and model selection. *Journal of Artificial intelligence*, 1137-1145.
- Kon, M., & Plaskota, L. (2000). Information complexity of neural networks. *Journal of neural networks*, 365-375.
- Kotsiantis, S., Zaharakis, I. D., & Pintelas, P. E. (2004). Machine learning: A review of classification and combining techniques. *springer*, 23-32.
- Krogh, A., & Vedelsky, J. (1995). *Neural network ensembles, cross validation and active learning*. (G. Tesauro, D. S. Touretzky, & T. K. Leen, Eds.) Cambridge: MIT Press.
- Kuncheva. (2004). *Combining Pattern Classifier: Methods and Algorithms*. New York: John Wiley.

- Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with ensemble accuracy. *Machine learning*, 181-207.
- Larson, S. (1931). The shrinkage of the coefficient of multiple correlation. *Journal of educational psychology*, 45-55.
- Leedy, P. D., & Ormrod, J. E. (2010). *Practical research: planning and design* (9th ed.). Upper Saddle River NJ: Prentice Hall.
- Liu, C., Tiang, D., & Yang, W. (2014). Global geometric similarity scheme for feature selection in fault diagnosis. *Journal of expert systems with applications*, 3585-3595.
- Magogo, S. (2017). *The Effectiveness of Counter Terrorism Strategies in Kenya: A case Study of Eastleigh Location, Nairobi County*. Nairobi: University of Nairobi.
- Makhtar, M., Yang, L., Neagu, D., & Ridley, M. (2012). Optimization of Classifier Ensemble for Predictive Toxicology Applications. *Journal of Computer Modelling and Applications*, 236-241.
- Maldonado, S., Weber, R., & Famili, F. (2014). Feature selection fo high dimensional class-imbalanced data sets using support vector machines. *Journal of Information services*, 228-246.
- Melville, P., & Mooney, R. (2003). Constructing Diverse Classifier Ensembles Using Artificial training examples. *IJCA*, 505-510.
- Mitchel, T. M. (1997). *Machine learning*. MA: McGraw-Hill Science/Engineering /Math.
- Mosteller, F., & Turkey, J. W. (1968). *Including statistics. In Handbook of socila psychology* . Reading, MA: Addison-Wesley.
- Neocleous, C., & Schizas, C. (2002). Artificial neural network learning: a comparative review, LNAI 2308. *Springer-Verlag*, 300-313.
- Neto, A. A., & Canuto, A. M. (2004). Meta -Learning and Multi-objective optimization to design ensemble of classifiers. *2004 Brazillian Conference on Intelligent Systems IEEE*, 91-96.

- Optiz, D., & Maclin, R. (1999). popular ensemble methods: An emperical study. *Journal of Artificial Intelligence Research*, 169-198.
- Osemengbe, O., & Uddin, P. S. (2014). Data Mining: An activ solution for crime investigation. *IJCST*, 53-61.
- Ozekes, S., & Osman, O. (2003). Classification and prediction in data mining with neural networks. *Journal of Electrical and Electronic Engineering*, 707-712.
- Piatestsky, G. (2014, 12 14). *CRISP-DM, still the top methodology for analytics, data mining, or data science projects*. Retrieved from Retrieved from KDnuggets: <https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html>): Retrieved from KDnuggets: <https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html>)
- Pillry, P. H., & Sikchi, S. S. (2014). Review of group prediction model for counterterrorism usingCLOPE algorithm. *International Journal of Advance Research in Computer Science and Management Studies*, 8-12.
- Platt, J. (1999). *Using sparseness and analytic QP for speed training of support vector machines*. In Kearns.M, Solla.S & Cohn. D(eds) *Advances in neural information processing systems*. MA: MIT Press.
- Polikar, R. (2012). Ensemble Learning in Zhang, C & Ma.D(eds) *Ensemble machine learning: Methods and Applications*. *Springer Science + Business*, 1-34.
- Popp, R., Armour , T., Senator, T., & Numrych. (2004). "Countering Terrorism through Information Technology". *Communications of the ACM*, 36-43.
- Prasad, S. S., Sonali, M., & Sonali, S. (2014). Border security up gradation using data mining. *International journal of soft Computing and Engineering*, 42-46.
- Quinlan, J. R. (1996). Bagging, Boosting and C4.5. *Proceedings of the thirteenth national Conference on Artificial Research* , 725-730.

- Rahim, A. (2014, 11 21). "Best practices for Business intelligence and predictive analytics". Retrieved from Available at: <http://www.informationbuilders.com/new/newsletter/13-04/3ali>: Available at: <http://www.informationbuilders.com/new/newsletter/13-04/3ali>
- Refaeilzadeh, P., Tang, L., & Liu, H. (2007). On comparison of feature selection algorithms . *Journal of Machine learning*, 5-23.
- Ricardo, G. O. (2014). Ensemble learning CSCE 66 Pattern Analysis. *Lecture notes*, 1-15.
- Rizwan, I., Masrah, A., Aida, A. M., Payam, H., & Nasim, K. (2013). An experimental study of classification algorithms for crime prediction. *Indian Journal of Science and Technology*, 250-263.
- Robert, J., & Howlet, L. C. (2001). Radial basis function networks: new advances in design. *Physica-Verlag Heidelberg, ISBN 3790813680*, 42-45.
- Robert, L., & Johnson, A. (2016). *Terrorist Attacks in the West*.
- Roli, F., Giacinto, G., & Vernazza, G. (2001). Methods of designing multiple classifier systems. *Lecture Notes Computer Science*, 78-87.
- Roy, A. (2000). On connectionism, rule extraction, and brain-like learning. *IEEE Trans Fuzzy System*, 222-227.
- Rushita, F. (2017). *Terrorist Groups in Africa*. Cairo: EMZ.
- Saad, D. (1998). *Online learning in neural networks*. london: Cambridge University Press.
- Sachan, A., & Roy, D. (2012). TGPM: Terrorist group prediction model for counterterrorism. *International journal of Computer Applications*, 44(10), 49-52.
- Santana, L. E., Siva, L., Canuto, A. M., Pintro, F., & Vale, K. O. (2010). A comparative Analysis of Genetic Algorithms and Ant colony Optimisation to select attribute for an heterogeneous Ensembles of classifiers. *Journal of Evolutionary Computation*, 1-8.
- Scholkopf, C., Burges, J. C., & Smola, A. J. (1999). *Advances in Kernel Methods*. MA: MIT Press.

- Setiono, R., & Loew, W. K. (2000). FERNN: an algorithm for fast extraction of rules from neural networks. *Applied Artificial Intelligence journal*, 15-25.
- Shafer, G. (1976). *A mathematical Theory of Evidence*. Princeton: Princeton University Press.
- Sohini, C. B., & Shaikh, M. Z. (2014). A comprehensive and relative study of detecting deformed identity crime with different classifier algorithms and multilayer mining algorithm,”. *International Journal of Advanced Research in Computer and Communication Engineering*, 453-460.
- START. (2018, 11 02). *National Consortium for the Study of Terrorism and Responses to Terrorism*. Retrieved from Global Terrorism Database: <https://www.start.umd.edu/gtd>
- Tang, J., Aleylani, S., & Liu, H. (2014). Feature selection for classification: A review in. *Jouranl of Algorithm Application*, 4-9.
- Tang, K., Suganthan, P. N., & Yao, X. (2006). An Analysis of Diversity Measures. *Journal of Machine learning*, 247-271.
- Tao, C. (2003). selective SVM Ensemble Based on Accelarating Genetic Algorithm. *Journal of Application Research of Computers*, 139-141.
- Ting, K. M., & Witten, I. H. (1999). Issues in stacked generalization. *Journal of Artificial Intelligence*, 271-289.
- Tiwaria, Abhishek, Sekhar, & Arvind, K. T. (2007). Workflow based framework for life science informatics. *Computational Biology and Chemistry*, 305-319.
- Tolan, G., & Soliman, O. (2015). An Experimental Study of Classification Algorithms for Terrorism Prediction,. *International Journal of Knowledge engineering*, 107-112.
- Verma, A. (2018). Study and Evaluation of Classification Algorithms in Data Mining”. *International Research Journal of Engieneering and Technology*.
- Verma, A. (2019). Evaluation of Classification Algorithms with Solutions to Class Imbalance Problem on Bank Marketing Dataset using WEKA. *International Research Journal of Engineering and Technology*, 54-60.

- Veropoulos, K., Campbell, C., & Cristianini, N. (1999). Controlling the sensitivity of support vector machines. *International joint conference on Artificial Intelligence IJCAI99*, 89-96.
- Villada, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial intelligence*, 77-95.
- Vivarelli, F., & Williams, C. (2001). Comparing Bayesian neural network algorithms for classifying segmented outdoor images. *journal of neural networks*, 427-437.
- Vladislav, M. (2014). *Machine learning of hybrid classification models for decision support, the use of the internet and development perspectives*. NY: Addison Wesley.
- Wahbeh, A. H., Al-Radaideh, Q. A., Al-Kabi, M. N., & Al-Shawakfa, E. M. (2008). A comparison Study between Data mining Tools over some Classification Methods. Retrieved from <http://www.thesai.org/downloads/SpecialIssueNo3/Paper%204-A%20Comparison%20Study%20between%20Data%20Mining%20Tools%20over%20some%20Classification%20Methods.pdf> [accessed on 10 December 2017]
- Wang. (2008). Some fundamental issues in Ensemble Methods. *Neural Networks, IJCNN. IEEE World Congress on Computational Intelligence*, 2243-2250.
- Wang. (2010). Heterogeneous Bayesian Ensembles for Classifying spam E-mails. *The 2010 International Joint Conference on Neural Networks IJCNN*, 1-8.
- Webb, I. G. (2000). Multiboosting: a technique for combining boosting and wagging. *Machine learning*, 159-196.
- Wettschereck, D., Aha, D. W., & Mohrit, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence*, 10, 1-37.
- Williams, N., Zander, S., & Armitage, G. (2006). A preliminary performance comparison of five Machine learning algorithms for practical IP traffic flow classification. *ACM SIGCOMM Communication Review*, 5-16.
- Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data mining practical machine learning tools and techniques*. Burlington: Morgan Kaufmann.

- Yam, J., & Chow, W. (2001). Feedforward networks training speed enhancement by optimal initialization of the synaptic coefficients. *IEEE Trans Neural networks*, 430-434.
- Yen, G. G., & LU, H. (2000). Hierarchical genetic algorithm based neural network design. *IEEE symposium on combinations of evolutionary computation and neural networks*, 168-175.
- Yu, L., & Liu, H. (2002). Efficient feature selection via analysis of relevance and redundancy. *JMLR*, 1205-1224.
- Yu, L., & Liu, H. (2004). Efficient feature selection via analysis of relevance and redundancy. *JMLR*, 1205-1224.
- Zhou, Z. H., & Tang, W. (2003). Selective Ensemble of Decision Trees. *Rough sets, Fuzzy sets, Data mining, and Granular computing, lecture notes in computer Science* , 476-483.
- Zhou, Z. H., Wu, J., & Tang, W. (2002). Ensembling neural networks: many could be better than all. *Artificial Intelligence*, 239-263.

APPENDICES

Appendix A: SGS Approval



MASENO UNIVERSITY
SCHOOL OF GRADUATE STUDIES

Office of the Dean

Our Ref: MSC/CI/00110/2014


Private Bag, MASENO, KENYA
Tel:(057)351 22/351008/351011
FAX: 254-057-351153/351221
Email: sgs@maseno.ac.ke

Date: 27th February 2019

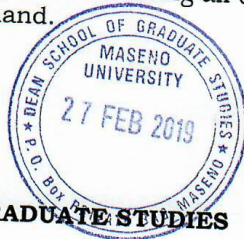
TO WHOM IT MAY CONCERN

**RE: PROPOSAL APPROVAL FOR PETER OPIYO OKECH—
MSC/CI/00110/2014**

The above named is registered in the Master of Science in the School of Computing and Informatics, Maseno University. This is to confirm that his research proposal titled **“An Evaluation of a Hybrid Machine Learning Classifier Model for Improved Identification of Terrorist Groups in Aftermath of an Attack in Subsaharan Africa”** has been approved for conduct of research subject to obtaining all other permissions/clearances that may be required beforehand.


Prof. J.O. Agure

DEAN, SCHOOL OF GRADUATE STUDIES



Appendix B: MUERC Approval



MASENO UNIVERSITY ETHICS REVIEW COMMITTEE

Tel: +254 057 351 622 Ext: 3050
Fax: +254 057 351 221

Private Bag – 40105, Maseno, Kenya
Email: muerc-secretariate@maseno.ac.ke

FROM: Secretary - MUERC

DATE: 26th June, 2019

TO: Peter Opiyo Oketch
PG/MSc/CI/00110/2014
Department of Information Technology
School of Computing and Informatics
Maseno University
P. O. Box, Private Bag, Maseno, Kenya

REF: MSU/DRPI/MUERC/00699/19

RE: An Evaluation of a Hybrid Machine Learning Classifier Model for Improved Identification of Terrorist Groups in the Aftermath of an Attack in Sub-Saharan Africa. Proposal Reference Number MSU/DRPI/MUERC/00699/19

This is to inform you that the Maseno University Ethics Review Committee (MUERC) determined that the ethics issues raised at the initial review were adequately addressed in the revised proposal. Consequently, the study is granted approval for implementation effective this 26th day of June, 2019 for a period of one (1) year. This is subject to getting approvals from NACOSTI and other relevant authorities.

Please note that authorization to conduct this study will automatically expire on 25th June, 2020. If you plan to continue with the study beyond this date, please submit an application for continuation approval to the MUERC Secretariat by 15th May, 2020.

Approval for continuation of the study will be subject to successful submission of an annual progress report that is to reach the MUERC Secretariat by 15th May, 2020.

Please note that any unanticipated problems resulting from the conduct of this study must be reported to MUERC. You are required to submit any proposed changes to this study to MUERC for review and approval prior to initiation. Please advise MUERC when the study is completed or discontinued.

Thank you.


Dr. Bernard Guyah
Ag. Secretary,
Maseno University Ethics Review Committee.



Cc: Chairman,
Maseno University Ethics Review Committee.

MASENO UNIVERSITY IS ISO 9001:2008 CERTIFIED



Appendix C: NACOSTI Approval



NATIONAL COMMISSION FOR SCIENCE, TECHNOLOGY AND INNOVATION

Telephone: +254-20-2213471,
2241349, 3310571, 2219420
Fax: +254-20-318245, 318249
Email: dg@nacosti.go.ke
Website: www.nacosti.go.ke
When replying please quote

NACOSTI, Upper Kabete
Off Waiyaki Way
P.O. Box 30623-00100
NAIROBI-KENYA

Ref. No. **NACOSTI/P/19/32897/29615**

Date: **30th May, 2019.**

Peter Opiyo Oketch
Maseno University
Private Bag
MASENO.

RE: RESEARCH AUTHORIZATION

Following your application for authority to carry out research on *“An evaluation of a hybrid machine learning classifier model for improved identification of terrorist groups in the aftermath of an attack in Sub-Saharan Africa.”* I am pleased to inform you that you have been authorized to undertake research in **all Counties** for the period ending **27th May, 2020.**

You are advised to report to **the County Commissioners and the County Directors of Education, all Counties** before embarking on the research project.

Kindly note that, as an applicant who has been licensed under the Science, Technology and Innovation Act, 2013 to conduct research in Kenya, you shall deposit a **copy** of the final research report to the Commission within **one year** of completion. The soft copy of the same should be submitted through the Online Research Information System.


BONFACE WANYAMA
FOR: DIRECTOR-GENERAL/CEO

Copy to:
The County Commissioners
All Counties.

The County Directors of Education
All Counties.

Appendix D: GTD Distribution Letter



June 28, 2017

Per the request you submitted via the GTD website please note that the data download/disc includes the following files:

- GTD 1970-2016 data file, including data on terrorist attacks between 1970 and 2016
- GTD 1970-1994 data file, including data on terrorist attacks between 1970 and 1994
- GTD 1995-2012 data file, including data on terrorist attacks between 1995 and 2012
- GTD 2013-2016 data file, including data on terrorist attacks between 2013 and 2016
- GTD 1993 data file, including cases collected in an effort to reconstruct the missing 1993 data (see below for additional information)
- GTD codebook, explaining the variables and coding schema for the GTD
- Global Terrorism Database *Terms of Use*

Please note that the data in the first file is identical to the data in the second, third, and fourth files. The smaller files are provided for users who require a limited number of rows per file.

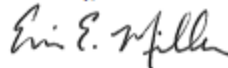
Regarding the fifth file, users should be aware that prior to the transfer of the original GTD data from Pinkerton Global Intelligence Services (PGIS) to START, all records of terrorist attacks during 1993 were lost. Several efforts were made to recover these incidents from original news sources. Unfortunately, due to the challenges of retrospective data collection for events that happened more than fifteen years ago, the number of cases collected for 1993 is only 15% of the number reported by PGIS. As a consequence we exclude all 1993 attacks from the GTD data to prevent users from misinterpreting the low frequency in 1993 as an actual count. However, we provide the reconstructed 1993 cases for those researchers who would find value in exploring these incidents. Together, with the PGIS recorded marginal counts for each country provided in the GTD codebook, analysts can use these data to interpolate values for the missing 1993 cases.

It is the policy of the Department of Homeland Security to protect the privacy of individuals. The information you have requested from the Global Terrorism Database may contain information related to specific individuals. DHS requires that you take every possible precaution to protect this information and that you use it for the purpose of advancing the understanding of terrorism. If you encounter any information about yourself that you believe to be inaccurate and wish to have removed from the database, please contact me at gtd@start.umd.edu.

A web-based interface for the GTD, including additional information on the data and its usage, is available at www.start.umd.edu/gtd.

Thank you for your interest in the Global Terrorism Database. We hope that you find it to be a useful tool. If you encounter any problems with the enclosed CD or its contents, please contact the GTD team via email at gtd@start.umd.edu. We welcome your feedback on the data and its application to your work.

Sincerely,



Erin Miller
GTD Program Manager, START

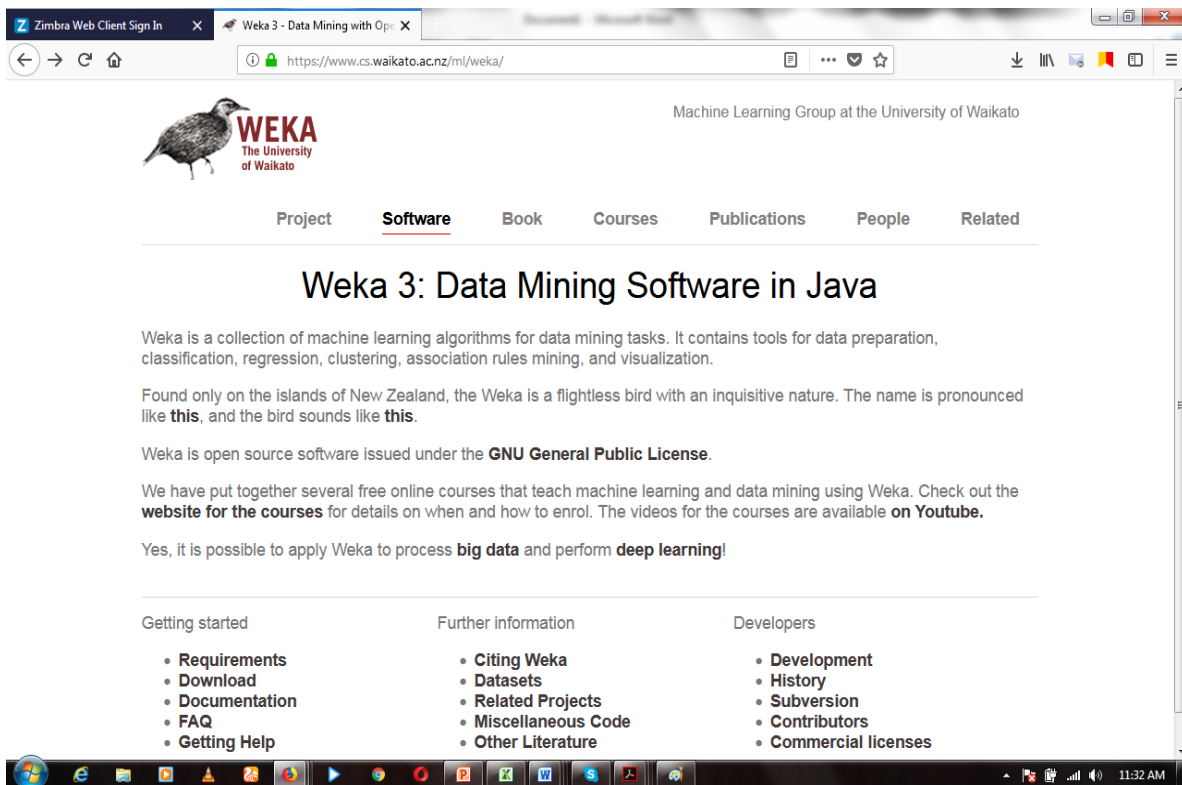


8400 Baltimore Ave, Suite 250 • College Park, MD 20740 • 301.405.6600 • www.start.umd.edu

Appendix E: Sample Dataset

GTD ID	DATE	COUNTRY	CITY	PERPE	GUNCEI	PERPE	GUNCEI	FATALI	INJUREI	TARGE	TARGE	REGION	ATTACI	ATTACI	ATTACI	VEAPO	VEAPO	VEAPON	TYPE 4
1	2017E-II	Sudan	Nertih	Sudan Lt	1	0	0	Private (Sub-Saharan Africa	0	Unknown Private Citizens & Property	Armed F	Sub-Sah	Hostage Taking	Kidnappin	Unknown				
2	2018E-II	Niger	Garoua	Boko H.	0	0	0	0	7	Military	Sub-Sah	Unknown							
3	2018E-II	Nigeria	Maiduguri	Boko H.	0	0	2	1	Private Citizens & Property	Sub-Sah	Bombing/Explosion	Explosives/Bombs/Dynamite							
4	2018E-II	Nigeria	Dogong	Boko H.	1	0	0	0	0	Military	Sub-Sah	Armed Assault	Firearms	Firearms	Melee				
5	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	3	Government (General)	Sub-Sah	Assassination	Explosives/Bombs/Dynamite						
6	2018E-II	Nigeria	Flam	Boko H.	0	0	0	0	0	Military	Sub-Sah	Armed Assault	Firearms						
7	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
8	2018E-II	Nigeria	Dogong	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
9	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
10	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
11	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
12	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
13	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
14	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
15	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
16	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
17	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
18	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
19	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
20	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
21	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
22	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
23	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
24	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
25	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
26	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
27	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
28	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
29	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
30	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
31	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
32	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
33	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
34	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
35	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
36	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
37	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
38	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
39	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
40	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
41	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
42	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
43	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
44	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
45	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						
46	2018E-II	Nigeria	Dan Mai	Boko H.	1	0	0	0	0	Government (General)	Sub-Sah	Assassination	Firearms						

Appendix F: WEKA URL



The screenshot shows a web browser window with the URL <https://www.cs.waikato.ac.nz/ml/weka/>. The page features the WEKA logo (a bird) and the text "WEKA The University of Waikato". The navigation menu includes "Project", "Software" (which is underlined), "Book", "Courses", "Publications", "People", and "Related". The main heading is "Weka 3: Data Mining Software in Java".

Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization.

Found only on the islands of New Zealand, the Weka is a flightless bird with an inquisitive nature. The name is pronounced like **this**, and the bird sounds like **this**.

Weka is open source software issued under the **GNU General Public License**.

We have put together several free online courses that teach machine learning and data mining using Weka. Check out the **website for the courses** for details on when and how to enrol. The videos for the courses are available **on Youtube**.

Yes, it is possible to apply Weka to process **big data** and perform **deep learning!**

Getting started	Further information	Developers
<ul style="list-style-type: none">• Requirements• Download• Documentation• FAQ• Getting Help	<ul style="list-style-type: none">• Citing Weka• Datasets• Related Projects• Miscellaneous Code• Other Literature	<ul style="list-style-type: none">• Development• History• Subversion• Contributors• Commercial licenses