

Performance Analysis of Stacking Neural Network and Machine Learning Model for Detecting Fraudulent Transaction

by

Ahnaf Shahriyar Chowdhury

15201009

Nayeem Abdullah

15201027

Hasan Al Mamun

15101065

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
April 2020

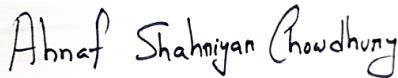
© 2020. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own authentic work while achieving degree at Brac University.
2. The thesis does not contain documents recently issued or composed by a third party, except where this is appropriately cited through exact and precise referencing.
3. The thesis does not contain documents which has been received, or proposed, for any other degree or diploma at a university or other institution.
4. We have recognized every primary sources of help.

Student's Full Name & Signature:



Ahnaf Shahriyar Chowdhury
15201009



Nayeem Abdullah
15201027



Hasan Al Mamun
15101065

Approval

The thesis/project titled “Performance Analysis of Stacking Neural Network and Machine Learning Model for Detecting Fraudulent Transaction” submitted by

1. Ahnaf Shahriyar Chowdhury (15201009)
2. Nayeem Abdullah (15201027)
3. Hasan Al Mamun (15101065)

Of Spring, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on April 19, 2020.

Examining Committee:

Supervisor:
(Member)



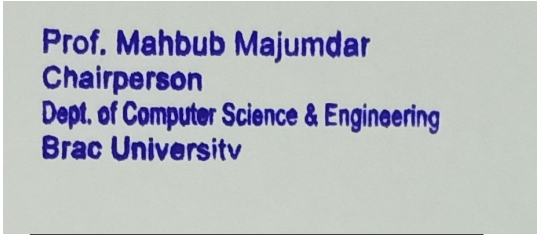
DR. Muhammad Iqbal Hossain
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)



Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)



Prof. Mahbub Majumdar
Chairperson
Dept. of Computer Science & Engineering
Brac University

DR. Mahbubul Alam Majumdar
Professor
Department of Computer Science and Engineering
BRAC University

Abstract

Transaction fraud has become a fast growing issue in the world of modern technology which has become a serious threat to the financial sectors. Although these fraudulent actions have several categories or type but online financial fraud has been a dominant issue so far. In reality, a profoundly precise procedure of identification of fraudulent transaction is required since it is causing a extensive wealth related depletion. Therefore, we have conducted research on financial fraud record using machine learning models and proposed a procedure for precise misrepresentation recognition dependent on the points of interest and restrictions of each exploration. In our initial stage, we implemented machine learning classifiers such as Logistic Regression, K-Nearest Neighbor, Support Vector Classifier, Naïve Bayes, Gaussian Naïve Bayes Classifier, Random Forest Classifier, Extra Tree Classifier, Neural Network and Adaptive Boosting to see how all of them performs separately. We also balanced the dataset that we used in order to overcome the overfit issue. Then again we tested the above mentioned classifiers on the balanced dataset. After that we tried our final step which is the implementation of Stacking technique. The accuracy that stacking method came up with were the best along with very less overfitting issues since K-fold cross validation was applied. To further boost the accuracy, we implemented Grid Search Hyperparameter tuning to get the best possible outcome at a much lower error rate. Therefore, to give a superior outcome for different sorts of online money transaction frauds, we have been keen on working with this issue and build a solid and defensive platform for safe transactions of money.

Keywords: Transaction fraud, Neural Network, machine learning classifiers, Overfit, Stacking technique, K-fold cross validation, Grid Search Hyperparameter tuning

Acknowledgement

Firstly, we would like to give all recognition to the Great Allah for whom our thesis have been finalized with no major hiatus.

Secondly, gratitude towards our thesis supervisor DR. Muhammad Iqbal Hossain sir for his caring help and guidance in our work. He helped us at whatever point we required help.

Lastly to our parents without their constant encouragement it may not be achievable. With their benevolent help and prayer we are currently very close to our graduation.

Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Acknowledgment	iv
Table of Contents	v
List of Figures	vii
List of Tables	x
1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement	2
1.3 Objectives and Contribution	3
1.4 Thesis Structure	4
2 Background	5
2.1 Literature review	5
2.2 Algorithms	8
2.2.1 Logistic Regression	8
2.2.2 Decision Tree and Random Forest Classifier	9
2.2.3 Extra tree Classifier	11
2.2.4 Naïve Bayes Classifier	12
2.2.5 Gaussian Naïve Bayes Classifier	13
2.2.6 K- Nearest Neighbor	14
2.2.7 Support Vector Machine	15
2.2.8 Deep Neural Network (Keras)	17
2.2.9 Adaptive boosting algorithm	18
2.2.10 Stacking Technique	20
3 Dataset Analysis	21
3.1 Dataset Description	21
3.2 Dataset Features	24
3.3 Dataset imbalance and its effect	25
3.4 Solution for Imbalance data	26
3.4.1 Near Miss Algorithm	26

3.4.2	Smote Distribution	27
4	Result Analysis	29
4.1	Evaluation Metrics	29
4.2	Model Performance	34
4.2.1	Logistic regression	34
4.2.2	Decision Tree	39
4.2.3	Random Forest	44
4.2.4	K-Nearest Neighbor	49
4.2.5	Support Vector Classifier	54
4.2.6	Naïve Bayes Classifier	57
4.2.7	Gaussian Naïve Bayes Classifier	62
4.2.8	Adaptive Boosting	67
4.2.9	Deep Neural Network (Keras)	72
4.2.10	Extra Tree Classifier	74
4.2.11	Stacking technique application and comparison	78
5	Conclusion and Future Work	82
	Bibliography	83

List of Figures

2.1	Logistic Regression curve	9
2.2	Random Forest working principle	10
2.3	Gaussian distributing graph	13
2.4	Example of KNN for two class [27]	14
2.5	Performance of Support Vector Machine [31]	16
2.6	Deep Neural Network working principle	17
2.7	Boosting algorithm workflow	18
2.8	Stacking Technique Framework	20
3.1	Sample of the dataset	21
3.2	Heatmap of the input data	22
3.3	Box plot of the dataset	23
3.4	Normal distribution of the dataset	23
3.5	Scatter plot of the dataset after implementing Near Miss algorithm .	27
3.6	Scatter plot of the dataset after implementing SMOTE	28
4.1	02 Scatter plot after using Near Miss	30
4.2	Classification report of Logistic Regression when applied in imbalanced dataset	34
4.3	Plot learning curve of Logistic regression on imbalance dataset	35
4.4	Confusion Matrix of Logistic Regression in imbalance dataset	35
4.5	Classification report of Logistic Regression after near miss application	36
4.6	Confusion Matrix of Logistic Regression after near miss application .	36
4.7	Classification report of Logistic Regression after SMOTE application	37
4.8	Plot learning curve for Logistic Regression on balanced dataset	38
4.9	Confusion Matrix of Logistic Regression after SMOTE application .	38
4.10	Classification report of Decision Tree when applied in imbalanced dataset	39
4.11	Plot learning curve for Decision Tree on imbalance dataset	40
4.12	Confusion Matrix of Decision Tree in the imbalance dataset	40
4.13	Classification report of Decision Tree after near miss application	41
4.14	Confusion Matrix of Decision Tree after near miss application	41
4.15	Classification report of Decision Tree after SMOTE application	42
4.16	Plot learning curve for Decision tree on balanced dataset	43
4.17	Confusion Matrix of Decision Tree after SMOTE application	43
4.18	Classification report of Random Forest when applied in imbalanced dataset	44
4.19	Plot learning curve of Random Forest on the imbalanced dataset	45
4.20	Confusion Matrix of Random Forest in the imbalance dataset	45

4.21	Classification report of Random Forest after near miss application . . .	46
4.22	Confusion Matrix of Random Forest after near miss application . . .	46
4.23	Classification report of Random Forest after SMOTE application . . .	47
4.24	Plot learning curve of Random Forest on the balanced dataset	48
4.25	Confusion Matrix of Random Forest after SMOTE application	48
4.26	Classification report of K-Nearest Neighbor when applied in imbalanced dataset	49
4.27	Plot learning curve of K-Nearest Neighbor on the imbalanced dataset	50
4.28	Confusion Matrix of K-Nearest Neighbor in the imbalance dataset .	50
4.29	Classification report of K-Nearest Neighbor after near miss application	51
4.30	Confusion Matrix of K-Nearest Neighbor after near miss application	51
4.31	Classification report of K-Nearest Neighbor after SMOTE Application	52
4.32	Plot learning curve of K-Nearest Neighbor on the balanced Dataset .	53
4.33	Confusion Matrix of K-Nearest Neighbor after SMOTE Application .	53
4.34	Classification report of Support Vector Classifier when applied in imbalanced dataset	54
4.35	Plot learning curve of Support Vector Classifier on the imbalanced dataset	55
4.36	Confusion Matrix of Support Vector Classifier in the imbalance dataset	55
4.37	Classification report of Support Vector Classifier after SMOTE application	56
4.38	Plot learning curve of Support Vector Classifier on the balanced dataset	56
4.39	Confusion Matrix of Support Vector Classifier after SMOTE application	57
4.40	Classification report of Naïve Bayes Classifier when applied in imbalanced dataset	57
4.41	Plot learning curve of Naïve Bayes on the imbalanced dataset	58
4.42	Confusion Matrix of Naïve Bayes Classifier in the imbalance dataset	58
4.43	Classification report of Naïve Bayes Classifier after near miss application	59
4.44	Confusion Matrix of Naïve Bayes Classifier after Near Miss Application	59
4.45	Classification report of Naïve Bayes Classifier after SMOTE application	60
4.46	Plot learning curve of Naïve Bayes on the balanced dataset	61
4.47	Confusion Matrix of Naïve Bayes Classifier after SMOTE application	61
4.48	Classification report of Gaussian Naïve Bayes Classifier when applied in imbalanced dataset	62
4.49	Plot learning curve of Gaussian Naïve Bayes on the imbalanced dataset	63
4.50	Confusion Matrix of Gaussian Naïve Bayes Classifier in the imbalance dataset	63
4.51	Classification report of Gaussian Naïve Bayes Classifier after near miss application	64

4.52	Confusion Matrix of Gaussian Naïve Bayes Classifier after near miss application	64
4.53	Classification report of Gaussian Naïve Bayes Classifier after SMOTE application	65
4.54	Plot learning curve of Gaussian Naïve Bayes on the balanced dataset	66
4.55	Confusion Matrix of Gaussian Naïve Bayes Classifier after SMOTE application	66
4.56	Classification report of Adaptive boosting when applied in imbalanced dataset	67
4.57	Plot learning curve of Adaptive boosting on the imbalanced dataset .	68
4.58	Confusion Matrix of Adaptive boosting in the imbalance dataset . .	68
4.59	Classification report of Adaptive boosting after near miss application	69
4.60	Confusion Matrix of Adaptive boosting after near miss application .	69
4.61	Classification report of Adaptive boosting after SMOTE application	70
4.62	Plot learning curve of Adaptive boosting on the balanced dataset . .	71
4.63	Confusion Matrix of Adaptive boosting after SMOTE application . .	71
4.64	Classification report of Deep Neural Network on balanced dataset . .	72
4.65	Deep Neural Network model loss	72
4.66	Deep Neural Network model accuracy	73
4.67	Confusion Matrix of Deep Neural Network after Near miss application	73
4.68	Classification report for one tuple of Extra Tree Classifier	74
4.69	Confusion Matrix of a single tuple after Near miss application	74
4.70	Classification report of Extra Tree classifier on the balanced dataset	75
4.71	Confusion Matrix of Extra Tree Classifier after Near miss application	75
4.72	ROC curve analysis for all the classifiers (Near Miss)	77
4.73	Plot learning curve for the 1st experiment	79
4.74	Plot learning curve for the 2nd experiment	79
4.75	Plot learning curve for the 3rd experiment	80
4.76	Grid Search Hyperparameter tuning on the 4th experiment	81
4.77	Plot learning curve for the 4th experiment	81

List of Tables

4.1	Experimental results of ML classifiers on imbalanced dataset	76
4.2	Experimental results of ML classifiers on balanced dataset (Near Miss)	76
4.3	Experimental results of ML classifiers on balanced dataset (SMOTE)	77
4.4	Experimental Results of Stacking Models	81

Chapter 1

Introduction

According to Oxford Learner's Dictionary, the word fraud signifies a person or a group who are involved in crimes of cheating other individuals in order to obtain wealth or goods illegitimately. The fraudulent actions can be categorized into many different types based on the types of the actions taken by a single individual or a group of people. Among those categories of frauds, transaction frauds have become a serious threat to the financial sectors. With the advancements of technology, transaction frauds have increased by several folds. Consistently online frauds and computerized data fraud bring about misfortunes in the millions for the budgetary part, also e-commerce business or telecommunications. This issue of transactions frauds not only occurs in developed countries like USA where online payments have become a daily work for the people but also in well technologically developed European countries as well. According to the information from July 2016 in Great Britain, one out of ten individuals give in to online burglary and crime [1]. Moreover, the expanding applications of mobile phones to operate financial transactions and online buying and payment from different e-commerce sites didn't go unnoticed from the eye of the hackers. During Clab 2016, organized in Peru in September 2016, the most recent data as for portable misrepresentation was represented: it has extended by 170% from the prior year and now addresses 62% of all online extortion. Among this numbers, 95% of the attacks are based on identity thefts. This type of fraudulent activity is very common alongside phishing (fake emails) and hackings [1].

1.1 Motivation

This is a very critical problem which needs the attention of data science and machine learning by which the solution to this problem can be automated. This problem is difficult from a learning context as it is characterized by various factors such as class imbalance. There are considerable amount of fraudulent transactions which are valid in terms of the given datasets. Moreover, in this era of physical and virtual card purchases customers prefer the most accepted payment method via the card which is the most accessible and convenient for the day to day users. This is where the risk of credit card use comes into scene and it is a major problem to avoid the possibility of unauthorized purchase using credit cards. Several data mining tools and machine learning approaches are available to effectively reduce these risks.

Financial fraud in IoT environment is growing fast because of the advancement in IoT field and it has been easier as almost any kind of payment can be done via mobile channel. Mobile payment has emerged enormously with the growth of internet trading and expansion of the IoT environment which conduct to a greater emergence of financial fraud in mobile payment system. Merchants started to endorse mobile web or mobile online service through application or web and more than 87 percent of them use any of these two [2]. Moreover, with the advancement of mobile payment in IoT world, mobile wallets have become more demotic with the increase demand of the society in the IoT field. With the overgrowing demand of mobile payments, financial fraud has reached a fearsome level resulting in a massive fall in the overall economy. Financial fraud can happen in several aspects, but the most common case in a mobile payment is unauthorized use of credit card number and its certification number. Detection of online credit card fraud and its mitigation will result in unhesitating use of its feature and save up all these financial amount to our economy rather than ruined up in the financial fraud loop.

1.2 Problem Statement

If the banking sector is considered, it is a great target for fraudulent activities. At present, banking services have become automated for which this sector is being targeted constantly. In February 2016, hackers managed to steal \$101 million from Bangladesh Bank with the Federal Reserve Bank of New York. Out of the total sum, \$81 million was distributed among four accounts with RCBC in Manila and the remaining \$20 million to a bank in Sri Lanka even though the transfer of money to Sri Lanka's bank failed because of some errors done by the hacker [3].

Recent incidents state that, serious cyberattacks took place in three of the private banks in Bangladesh in May 01,2019. Even though two of the banks named NCC and Prime Bank were the victims but they were able to back up their financial losses as the amount was not big. But in case of Dutch Bangla Bank Limited they faced a terrible loss. This cyberattack affected them so severely that it cost them \$3 million which is almost TK.25 crore. An unorthodox method was used where the hacker set up a malware in the bank's card management system prior three months. As a result, a perfect replica of the bank's switch was made. For this reason, when the hacker went for the transaction and the proxy switch made by the hacker did the function and the bank was unable to detect [4].

Methodically, to commit a credit card fraud offline, the intruder must steal the credit card physically to carry out the fraudulent transactions. Accordingly, in cases of IoT system the customer does not need a physical payment tool, rather it needs the information from the credit card such as expiry date, card verification code and one-time password sent to them via mobile or email with a fixed amount of time of its effectiveness. And correspondingly fraudster gets it easy making fraudulent payment online rather than physically making the fraud by abducting the credit card. As a consequence, the most common form of financial fraud which normally occurs in IoT environment involves collecting or changing information on credit cards.

1.3 Objectives and Contribution

We are living in an express train to a cashless society where alternates of cash are preferred the most. According to the World Payments Survey, there has been a rise of non-money exchanges by 10.1% from 2015 for an aggregate of 482.6 billion exchanges in 2016 [4]. Moreover, the number of non-cash transactions is expected to increase gradually in future years. While the effect of credit card fraud is restricted to approximately 0.1 percent of all card transactions, it has contributed to enormous financial losses as large-scale fraudulent transactions. In 1999, of the 1,200 transactions a year about 10 million were fraudulent or one in 1,200 [5]. Moreover, 0.04% of all monthly active accounts (four out of 10,000) were also fake. Although since that point, the credit card transactions have risen tremendously in volume and size, the proportions remained the same or decreased due to advanced fraud detection and prevention mechanisms. The fraud detection mechanisms currently in place are designed to prevent 1/12th of all transaction processes which still results in losses of trillions of dollars [6].

65 percent of fraud happened through technology-driven systems like ATM, credit cards and Internet banking [7]. Allegations of credit card and ATM have occurred in number of banks in Bangladesh including a foreign bank. Moreover, there was also a record of transfer about 1,50 billion in a third-generation bank during the software process. With the advancement of technology banking sector has started to use technology extensively with the introduction of new ATMs, POS (sales point), internet and mobile banking etc. Despite the fact of the customers becoming more tech savvy, there lies a group of fraudsters who are taking advantage of the technical loopholes. Ability of detect fraud in these systems will result in much more irresistibility and freedom in using these tech amenities. Fraud is an illegal act in which services, goods and funds are obtained. Fraud has encompassed many illegal practices and actions in last ten years. The main reasons for data theft are skimming and card traps [8]. Fraud has caused huge financial losses, data loss, destruction of financial institutions and its reputations. It is important to detect fraud transactions to mitigate the obstruction. In order to detect transaction fraud there has been already several statistical model existing [9], [10].

1.4 Thesis Structure

Chapter 1 – Introduction part where motivation, problem statement and objective is included.

Chapter 2 – Background where the literature review indicates the previous works and researches of our thesis topic and also the theoretical explanation of all the algorithms that are being used.

Chapter 3 – Dataset Analysis which contains data visualization, dataset features, data imbalance issues and its solutions.

Chapter 4 – Result Analysis which contains all the experimental values and graphs including comparison between the algorithms

Chapter 5 – Conclusion and the researches and application that will be done in the future is discussed in the chapter.

Chapter 2

Background

2.1 Literature review

The exponential development of the Internet has offered enormous market potential for the present business including e-banking industry. Although e-banking industry has provided many benefits for the business end but it imposes traditional banking needs and several security issues. Convenience, trust, and social impact were significant influencers to the reception of E-Banking. Very much established trust in paper-based exchanges and change-shirking society still an unclear term in the e-banking area to be utilized on a more extensive territory [11]. However, e-banking consistency inclusion has been eclipsed by the poor notoriety of web based business normality issues. Researchers proposed differing systems for confirmation where multi strategy is prescribed while using biometric strategies in security [12]. All things considered, extortion identification faces more issues when managing new conditions like distributed computing. As per their surveys "Exchange Monitoring" trailed by "Short Message Service (SMS)" and "One-Time Password Tokens" are the best models and respondent's conclusions are "Virtual Keyboards", "Browser Protection", and "Device Identification" are the most noticeably awful model.

In the research paper done by Emad Abu-Shanab and Salam Matalqa, they argued that fraud detection domain, fraud catching rate and bogus alert rate are preferable measurements over the general exactness while assessing the scholarly fraud classifiers. They used meta-learning to join various classifiers to keep up and improve the exhibition of the classifier. In their test, the preparation information was examined from before months, the approval information (for meta-learning) and the testing information were inspected from later months. The instinct behind this plan is that they reenact this present reality condition where models was prepared utilizing information from the earlier months, and ordered information of the present month. Additionally, they utilized just a bit of their unique database for learning instead of loading the entire training data in the main memory and compared the information learning on the adapted information versus the first information and came about no misfortune in exactness. Since their information had a slanted class dispersion they train on data that has higher fraud rate and determined 80% overall accuracy. They kept their rate of fraud misrepresentation catching comparatively lower along with a higher rate of fake caution in the fraud detection domain that countless false exchanges experiences and an enormous number of genuine exchanges gets hindered

by their recognition framework interceding human exertion in approving such exchanges. They likewise considered misrepresentation getting rate as significantly more significant than false caution rate.

Financial fraud activities under IoT domain is the quickly developing issue through the ascent of cell phone and online progress administrations [11]. Right now, the researchers have overviewed fraud detection techniques by utilizing machine learning and deep learning technology, fundamentally during the timeline between 2016 and 2018, and suggested a procedure for precise fraud detection dependent on the favorable circumstances and impediments of each exploration. The exploration additionally actualized both machine learning and deep learning strategy to think about the effectiveness of recognizing the transactions which indicates fraudulent activities. Additionally, the exploration has played out the general procedure of financial fraud recognition in functional point of view dependent on supervised and unsupervised machine learning strategy [12]. Likewise, they are proposing a down to earth technique by applying sampling procedure and selection of dataset feature process for taking care of data imbalance issue and quick identification in reality. The proposed model comprises of data preprocessing, testing, highlight determination, use of arrangement, and bunching calculation dependent on machine learning. The preprocessing performs information connection examination and information cleaning process [12]. Inspecting process assesses dataset with different proportions for check through arbitrary oversampling and undersampling technique. Clustering method with the proposed calculation is performed and this outcome is utilized as a preparation set in the arrangement procedure. The model approval process is performed with exactness and review rate through F-measure. An artificial intelligence network copies the actions done by human mind in handling information and makes designs for use in dynamic territory, through the capacity of solo gaining from information that is unorganized or unregulated. The exactness of every calculation utilizing the element separated through the proposed highlight choice technique was estimated [12]. Notwithstanding real datasets, open information were likewise applied also for progressively exact check of their proposed strategies. The examination was directed utilizing 270,000 bits of information which was extricated into 21 attributes. In the outcome, it was discovered that the notable AI strategy has a higher misrepresentation recognition rate than the fake neural system in spite of the fact that the procedure takes moderately longer than the machine learning procedure.

In 2014, D.Olszewski proposed a method of detecting fraudulent activities by using an algorithm known as Self-Organizing Maps [13] which is based on dimensionality of a dataset. It is a kind of machine learning that falls under unsupervised machine learning and also uses neural network technique. Its basic function is to anticipate the datasets from high to low dimensionality. SOM also works in real time situations that means output is shown during the actual time of the work. No used dataset in used in this algorithm. As a real time dataset, information of 200million customer is used to work on this algorithm. The function of this machine learning algorithm is that it can identify new hidden data patterns in the training dataset which becomes very crucial when it tests the dataset designated for testing. As a result, the processing time decreases as well as the processing costs.

P.Ravishankar in his paper which was published in 2011 showed the use of another unsupervised machine learning technique called Group Method of Data Handling [14] to detect financial fraud statement along with feature selection of the dataset. This technique is used to construct complex architecture or system. It examines different complex models and based on the criterion of different parts of data sample, it estimates all the models and gives a suitable output. According to the observation done by using the algorithm, it is seen that the accuracy reaches 95.09% which is undeniable a great result but there is a backlash in using this method. The major issue with this algorithm is that the processing time is very slow since it evaluates through various complex models to give an output. As a result, the processing costs also increases in proportion to the processing time. In 2007, B.Hoogs and T.Kiels also researched on detecting fraudulent activities in financial statements. They proposed genetic algorithm [15] as their solution. The accuracy of this algorithm is 95% and this algorithm can accurately detect biased patterns within the dataset.

E.Duman and H.Ozcelik in their paper [16] which was published in 2011 demonstrated the an algorithm called Scatter Search. It follows the iterative method in which each best solution is being clustered in a set and analyzing those outputs in that set the best result in being produced. This algorithm has some similarities with Genetic Algorithm. They used 100,000 fraudulent transactions dataset in their research. In another research paper written by S.Pangrahi, A.Kundu and S.Sural [17] they used the model called Dempstar Shafer Theory (DST). This model can calculate the overall belief values form each of the transaction found in the datasets. This value is used to generate the final output to detect fraudulent transactions among all the transactions done. The best thing about this method is that any new rules can be adjusted with the existing framework of DST. As a result, more unique models can be generated to give better output. O.Adewumi and A.Akinyelu in their paper [18] used hybrid machine learning models to determine credit card transaction frauds. They also used the application of big data technologies. They did research on different types of machine learning algorithms to come up with a hybrid solution for better performance.

A.Mubalik and E.Adali in their research paper [19] based on detection fraud transactions used Multilayer Perceptron Neural Network (MLP). This model uses layers of perceptron which can feed information forward to the neural network for further processing. From their analysis they were successfully able to enhance the performance of their algorithm and was also able to decrease error rate in detecting fraudulent transactions from 19% to 12.23%. K-nearest Neighbors (KNN and Outlier detection was another method of fraud detection technique used by N.Malini and M.Pushpa. In their research paper [20] which was published in 2017, concluded with the information that outlier detection can give better results if applied on a large transaction dataset. On the other hand, KNN can perform fraud transactions analysis easily on a limited memory. In another paper written by N.S. Halvaiee and M.K.Akbari [21] which was published in 2014, took a unique approach in order to solve the transaction fraud detection issues. They used Artificial Immune Systems (AIS) and Cloud computing in order to detect the fraud transactions more accurately. The dataset that is used in this research contains about 3.74% transactions that are marked as fraud transactions. AIS generates output based on the performance of the number

cell. If this performance can be improved by any ways then the detection rate can also be improved.

2.2 Algorithms

From the related works that we studied, we have seen many different types of approaches along with the observations and shortcomings. In this section, our proposed models to detect whether a transaction is resulted in fraud or not are discussed. Our proposed models include Decision Tree, Extra Tree Classifier, Random Forest Classifier, Naive Bayes Classifier, Support Vector Classifier (SVC), K Nearest Neighbor (KNN), Naive Bayes (Gaussian) Classifier, Logistic Regression Classifier, Adaptive boosting and Stacking technique.

2.2.1 Logistic Regression

Out of the most generally utilized machine learning algorithm that is implemented in order to resolve any classification problem, one of them is Logistic Regression. Logistic Regression was introduced because there was a major issue in Linear Regression. The issue was that if the dataset contains any outlier, linear regression model fails to classify that data sample. As a result, the final output might be wrong. To tackle this issue, Logistic Regression model was being developed [22].

Logistic regression uses a function called Sigmoid function which indicates that it can accept any real outputs in the middle of the value zero and one. This outcome can be interpreted as-

$$\sigma(t) = \frac{e^t}{(1 + e^t)} \quad (2.1)$$

Where, ' σ ' is the sigmoid function and t can be any linear equation. If we use the linear function in a regression based on variable quantity, we get the succeeding logistic calculation.

$$\rho(t) = \frac{\exp(\beta_0 + \beta_1 x)}{(1 + \exp(\beta_0 + \beta_1 x))} \quad (2.2)$$

where β_0 and β_1 are the variable quantities.

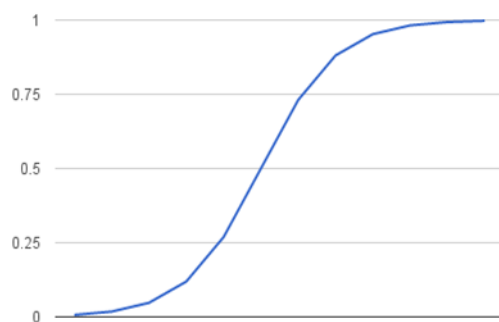


Figure 2.1: Logistic Regression curve

Some of the pros and cons of Logistic regression is given below.

Pros:

- Logistic Regression curve can decrease the bias during the training phase.
- For datasets with minimum dimensionality, logistic regression is a great model since it can generate output much faster.

Cons:

- If the data-set is huge then logistic regression will give overfitting result.
- In the real world, the relation between independent and depend variables are.

2.2.2 Decision Tree and Random Forest Classifier

Random Forest is a type of machine learning algorithm which falls into the classification genre. It is a model that involves a huge number of individual decision trees that works as a group. Each decision tree that is being used by random forest algorithm comes up with a projection and the output with the most support become the overall projection of the model. Even though some decision trees will give wrong outputs but the maximum number of decision tree will give the correct prediction. For this reason, a decision tree's individual error will not affect the overall output of the model. This is one of the advantages of using random forest. Another advantageous point that is to be noticed in this algorithm is that the correlation between the models is low. As a result, it can estimate the overall ensemble prediction more accurately than any other individual model's prediction. Random forest is based on some mathematical equations that includes the equation the determine the outcome of each decision tree. For calculating decision tree, the equation of entropy is needed. Entropy in decision tree is the part where the quality of a data sample is checked using ID3 algorithm [23]. The higher the entropy, the tougher it is for the model to give a correct prediction. The formula of entropy is

$$E = \sum_{i=1}^C -p(i) * \log p(i) \quad (2.3)$$

After that information gain is calculated. Information gain reduces entropy by transforming the dataset. The formula to deduce information gain is

$$Gain(T, X) = Entropy(T) - Entropy(T, X) \quad (2.4)$$

where T is the targeted value and X is the feature that is to be split. By using these equations, the decision tree gives it predictions and the best is elected as the overall output of the classifier.

The workflow for random forest algorithm is given.

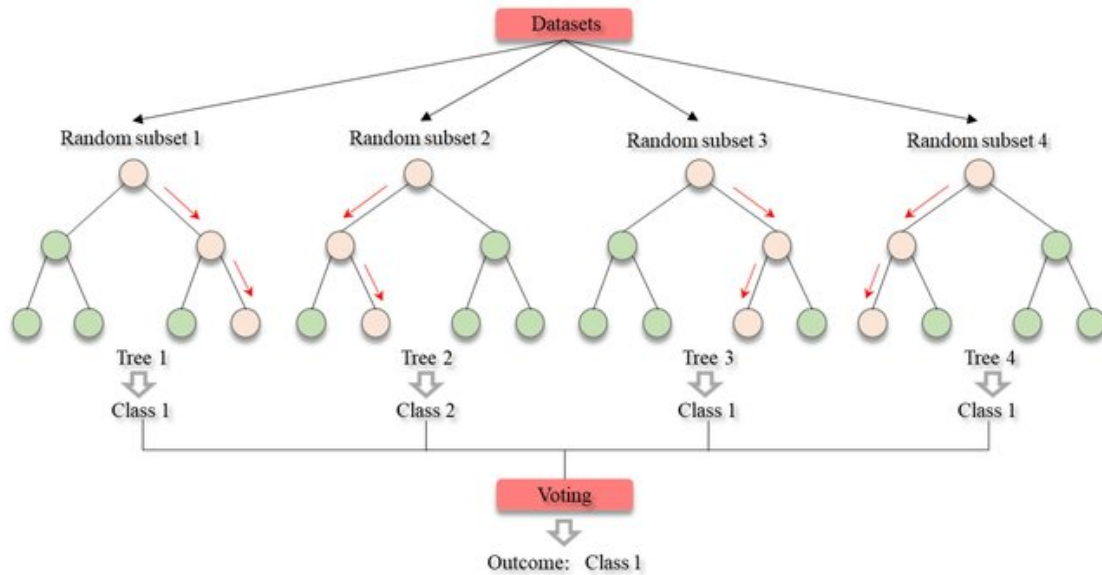


Figure 2.2: Random Forest working principle

Some of the advantages and disadvantages of Random Forest Classifier is given below.

Advantage:

- Handy algorithm for both regression and classification kind of issues.
- Effect of corrupt data sets is fewer than other models.

Disadvantage:

- Time complexity of random forest is more since it takes decision based on many decision tree's output.
- Needs more computational power since random forests deal with many decision trees.

2.2.3 Extra tree Classifier

An ensemble learning method related to decision tree and random forest is Extra Tree Classifier or Extremely Randomized Classifier. Even though there are resemblance between extra tree classifier and random forest, the main contrast is found during the assemble of decision trees in the algorithm while training a dataset. The main two parameters working behind this classifier is the number of features sorted out randomly at every which is denoted by 'K' and another parameter is the sample size on the basis of which a single node is divided. It is denoted by 'nmin' [24]. This is an iterative process where the process of training the dataset goes on according to the number of decision trees. This picking of the trees number is denoted by 'M'. The parameters of extra tree classifiers determine how strongly this algorithm will perform. For instance, the value of 'K' is inversely proportional to the strength of extra tree classifier. If the value of 'K' is minimized, then the random selection of the features for the decision trees strengthens and vice versa.

The mathematical process of extra tree classifier is almost similar to that of random forest. At first the number of decision tree is fixed along with the number of attributes that is the merit of 'K'. Then entropy is calculated and using the value of entropy, Gain is calculated for each decision trees. This is the general process by which extra tree classifier does the calculation and ultimately comes up with the final output.

Some of the pros and cons of Extra Tree Classifier are given below.

Pros:

- Computational speed is much faster compared to random forest classifier
- Chances of less overfit since variance is less.

Cons:

- If the dataset consists of high amount of noisy datasets, then the accuracy may fall for this classifier.

2.2.4 Naïve Bayes Classifier

A Naïve Bayes classifier is a basic probabilistic classifier model that is utilized for classification. It computes a lot of probabilities by tallying the recurrence and blends of qualities in a particular arrangement of information. The calculation utilizes Bayes hypothesis and takes the class variable value for each attribute to be independent. Naïve Bayes classification works better for large datasets and multi-class classification and computationally fast. Although in real cases this conditional independence rarely holds true but with the naïve definition the algorithm yet continues to do well and learn quickly in a number of supervised classification problems. For discrete and multinomial distributed features multinomial naïve Bayes classifier is implemented.

In a multinomial occasion model, p_i being the likelihood that the occasion i happens and x being an element vector where $x = (x_1, \dots, x_n)$ is then a histogram, with x_i tallying the quantities of time an occasion has happened in a specific case can be communicated by

$$\log p(C_k|x) \propto \log p(C_k) \prod_{i=1}^n p_{ki}^{x_i} = \log p(C_k) + \sum_{i=1}^n x_i \cdot \log p_{ki} = b + w_k^T x \quad (2.5)$$

Where $b = \log p(C_k)$ and $w_{ki} = \log p_{ki}$

Some of the pros and cons of Naïve Bayes Classifier are given below.

Pros:

- Application of Naïve Bayes classifier is easy
- It takes less time to compute since the training data set required by the model is very low.

Cons:

- Naïve Bayes is not a good choice when it comes to real life implementations.
- If a data point is missed while training, then Naïve Bayes automatically assumes that data point as 0. As a result, it hampers the overall accuracy of the model.

2.2.5 Gaussian Naïve Bayes Classifier

Gaussian Naïve Bayes is a probabilistic solution algorithm. It is one classifier model. The calculation of the probability classes and the test data given to the classes is required before and after. Both groups are determined using the same formula prior probabilities. Gaussian Naïve Bayes formula is described below-

When features x_i are continuous valued, typically make the assumption they are normally distributed.

$$P(x_i|y) = \frac{1}{\sqrt{(2 * \pi * \sigma_y^2)}} \exp\left(-\frac{(x_i - \mu_y)^2}{2 * \sigma_y^2}\right) \quad (2.6)$$

According to Layman's terms,

When we deal with continuous data we assume the data are distributed according to Gaussian distribution and thus, while applying Bayes theorem we use the property to ascertain the likelihood of an event [25].

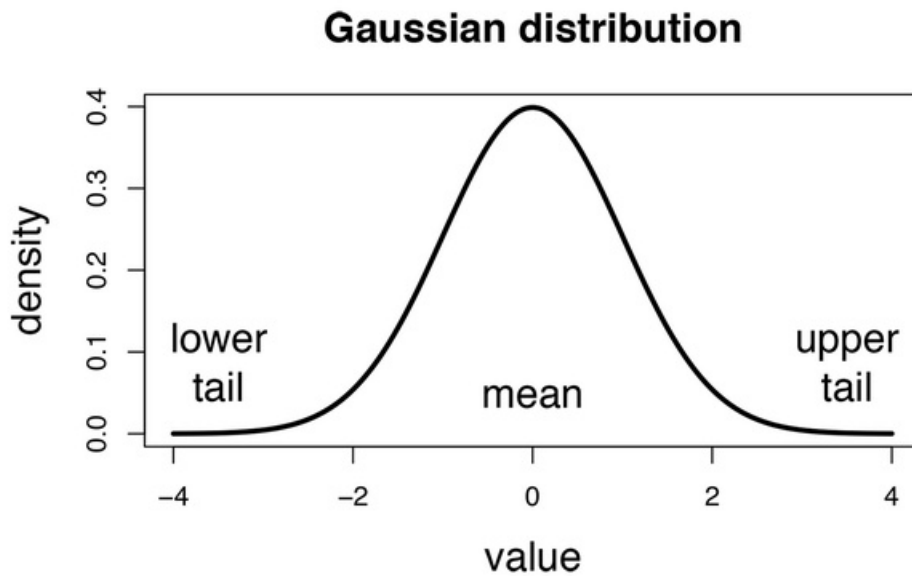


Figure 2.3: Gaussian distributing graph

2.2.6 K- Nearest Neighbor

K- Nearest Neighbors Algorithm is a simple supervised machine learning algorithm that is used for both classification and regressive predictive problems. It is used on labeled data to learn a function where new unbalanced data is given to produce an acceptable output. It stores all available cases and classifies on the basis of previous measure. Lazy learning algorithm and non-parametric algorithm are two properties that would define KNN. It has been already used as statistical estimation and pattern recognition as a non-parametric technique. For its non-parametric meaning and holding no underlying meaning about the data it has been widely used in real life scenarios. It detects K-closest neighbors on a minimum distance from the analysis to the training samples. The majority of K-nearest neighbors are considered to be the assumption of the query instance after gathering the nearest neighbors. K-Nearest Neighbor makes prediction based on the results of the K neighbors. In order to predict the distance from query point and case from the sample cases, we need to define a metric. The Euclidean is one of the popular ways to measure this distance. The Euclidean Square, City-block and Chebychev measures are also used in the process [26].

$$D(x, p) = \begin{cases} \sqrt{(x - p)^2} & \text{Euclidean} \\ (x - p)^2 & \text{EuclidianSquared} \\ |x - p| & \text{Cityblock} \\ \text{Max}(|x - p|) & \text{Chebychev} \end{cases} \quad (2.7)$$

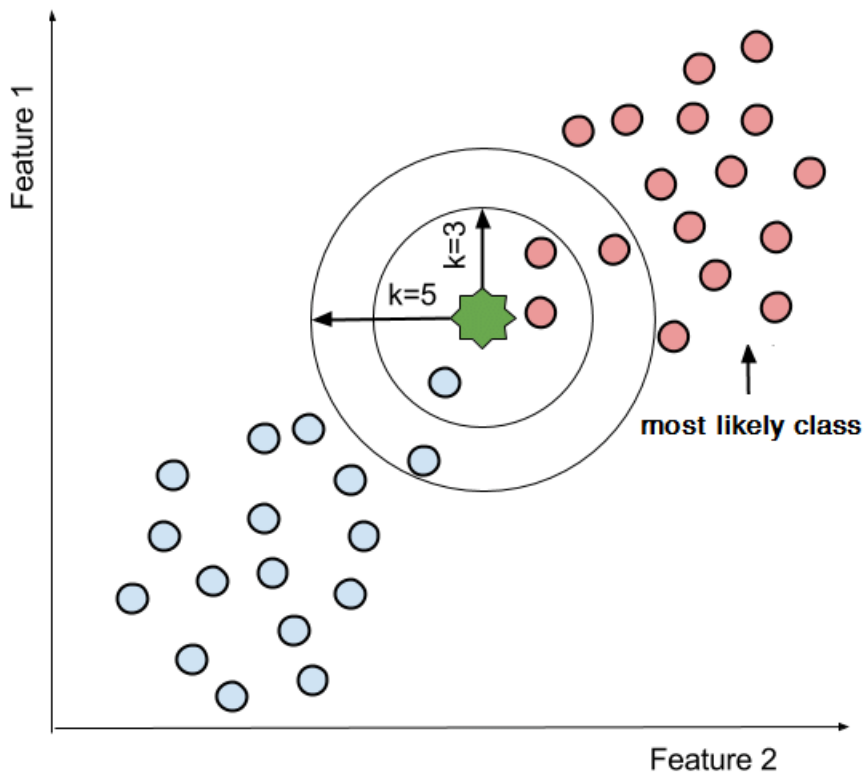


Figure 2.4: Example of KNN for two class [27]

Some of the pros and cons of KNN are given below. [28]

Pros:

- Non parametrical which makes no prediction about the underlying data pattern.
- Used for both classification and regression.
- Training stage for the closest neighbor is quick.

Cons:

- Computationally costly as it searches the nearest neighbor at the prediction level for the new point.
- Prediction stage is very costly.
- Prone to outliers.

2.2.7 Support Vector Machine

Support Vector Classifier (SVC) is a kernel-based supervised learning algorithm that incorporates machine learning theory, operational testing optimization algorithms and mathematical analysis kernel techniques [29]. It has been very famous because of its robust mathematical theory as a large-margin classifier. It is used in numerous practical fields such as bio-informatics, medical sciences for the diagnosis and in various applications in the field of engineering for model prediction. Because of its strong learning capacity in classification it is commonly used in medical science. Using kernel function, it can classify highly non-linear data. It is one of the best learning algorithms among the “off-the-shelf” supervised learning algorithm [30]. It is the binary classification problem kernel based supervised learning algorithm. The training dataset generate a kernel function which distinguish between the two classes. The aim is to create a classifier that works well for unknown cases to give a good generalization in all case.

If there are m training examples (x_i, y_i) , where $y_i = \pm 1$ and $i = 1, 2, 3, \dots, n$

And there exists a hyperplane $w \cdot x + b = 0$, separating the positive and negative training examples using the decision function,

$$f(x) = \text{sign}(w \cdot x + b) > 0, \text{ where } \text{sign}(x) = \begin{cases} -1, & \text{if } x < 0 \\ 0, & \text{if } x = 0 \\ 1, & \text{if } x > 0 \end{cases} \quad (2.8)$$

Where, w is a normal to the weight vector hyper plane, and b is referred to as bias. We can see that,

$$y_i (w \cdot x + b) > 0, \text{ where } i = 1, 2, 3, \dots, n. \quad (2.9)$$

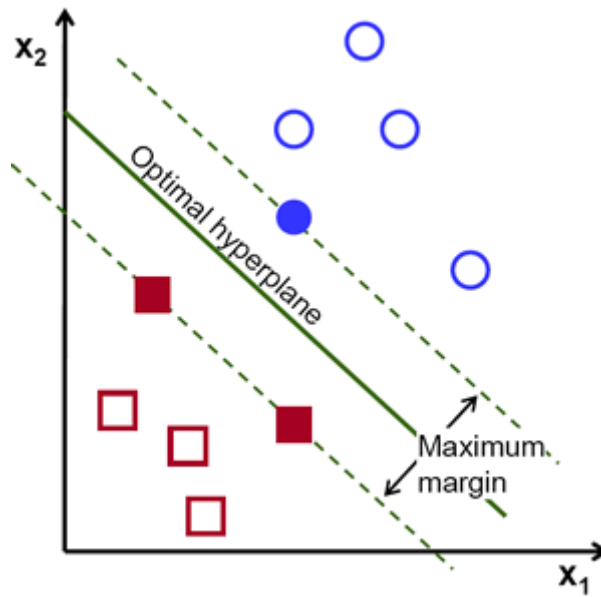


Figure 2.5: Performance of Support Vector Machine [31]

Pros:

- Efficient when the number of functions is more than the training examples.
- Considered the best algorithm for the reparability of its classes.
- Outliers have less effect as only the support vectors affect the hyper planes.

Cons:

- Requires a significant amount of time to process for larger datasets.
- Need to select appropriate hyper parameters to allow sufficient generalization performance.
- Difficult to select the right kernel function.

2.2.8 Deep Neural Network (Keras)

One of deep neural network with high level performance is keras. It's run on top of TensorFlow, CNTK, THEANO. We are using it on top of TensorFlow because it's open source. Classification with keras [32] consist of three main layers. There are input layer, hidden layer, output layer. Noise layers help to abstain from overfitting [33]. Repetitive layers incorporate basic (completely associated repeat), gated, LSTM, and others; these are valuable for language handling, among different applications. Pooling (downscaling) layers run from 1D to 3D and incorporate the most widely recognized variations, for example, max and normal pooling. Privately associated layers' act like convolution layers, then again, actually the loads are unshared.

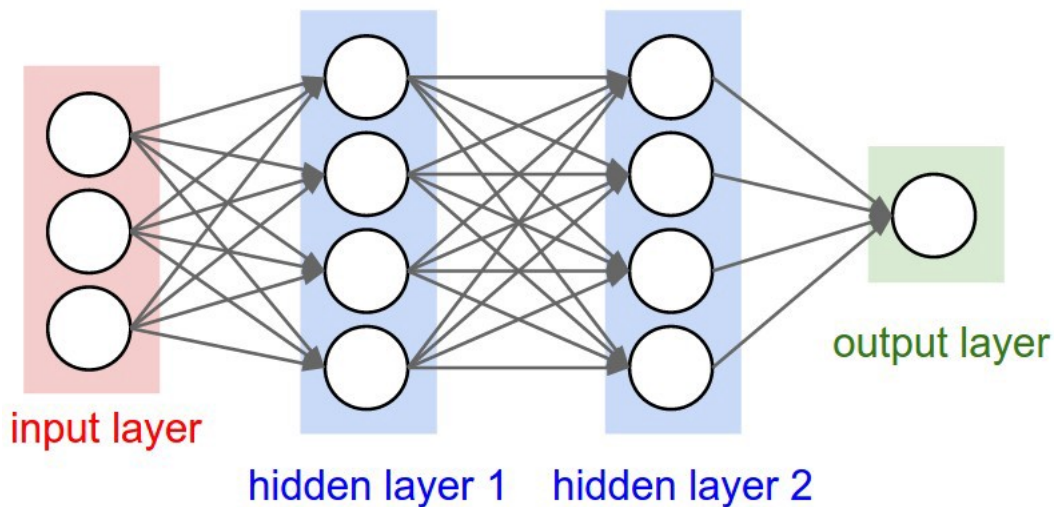


Figure 2.6: Deep Neural Network working principle

Some pros and cons of Deep Neural Network is given below [34].

Pros:

- Keras is high level API of neural network. Most of its packages is maintain by google. So, we have secure and fast performance.
- It uses NVIDIA's CUDA toolkit for fast operation to boost performance.

Cons:

- As it is maintained by google not all of it source is open source. For this reason, it is hard to customize the model.

2.2.9 Adaptive boosting algorithm

Usually when it comes to showing result in machine learning projects, one model is used. A certain accuracy is generated as a result but there are some algorithms which can boost the outcome of any machine learning model. These algorithms are called boosting algorithms. Boosting algorithm follows technique where it combines comparatively weaker models in order to generate a strong model with a better output. For example, there is three separate model used for classification purpose which are “X”, “Y” and “Z”. These three models are considered a weak model in this case. All the three models gave a faulty result and the error rate is close to 1 but if a correlation between these three weak classifier is established by combining them all together then by applying voting technique, the overall classifier gives a better output then the individual weak classifiers. It also follows sequential process. It means that the weakness of one model goes to the next model where the weight of each misclassified points is increased in order to decrease the weakness of the models. After sequential iterations, the overall model becomes a strong one. For this reason, boosting is considered a great ensemble method.

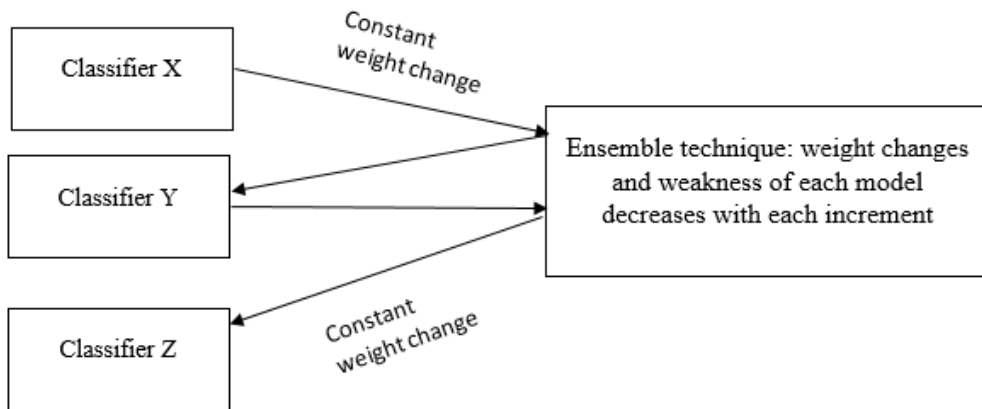


Figure 2.7: Boosting algorithm workflow

Adaptive boosting in short AdaBoost was the first efficient binary application boosting algorithm. It is typically used in short decision trees to boost the performance of decision trees which is based on binary classification problems. AdaBoost, often called discrete AdaBoost as it is used for classification more than regression. It can be used to boost any machine learning algorithm’s efficiency. It is mostly used with weak learners. It focuses on classification problem setting a strong classifier from a weaker one. The classification equation can be represented as [35]-

$$F(x) = \text{sign} \left(\sum_{m=1}^n \theta_m f_m \right) \quad (2.10)$$

Where f_m represents the m_{th} weak classifier and θ_m corresponds to the appropriate weight. It is precisely the weighted combination of weak classifiers for m .

AdaBoost incorporates multiple classifiers to make the classifiers more accurate. It creates a strong classification by combining multiple weak classifier to achieve a high accuracy strong classifier. The main goal of this classifier is to decide in each iteration the weights and train the sample accordingly so no uncommon observations are predicted inaccurately [36]. If a machine learning algorithm accepts weight in the training set, it can be used as a base classifier. AdaBoost should be directly trained on different weighed training examples. Loop wise it aims to match the examples perfectly by reducing training errors.

Some of the pros and cons of Adaptive boosting algorithm are mentioned below [37].

Pros:

- Versatility to be paired with any machine learning algorithms
- No need to modify the parameters except for T .
- It has been applied to problems in the study of text and numeric details beyond binary classification and is flexible.

Cons:

- Sensitive to uniform noise as it is from scientific evidence.
- Weak classifier can cause low margins and excess.

2.2.10 Stacking Technique

Stacking is a sort of ensemble method where various sorts of algorithms are combined with the help of a meta classifier or learner [38]. These algorithms are called base learners or classifiers in this case. At first the dataset is being trained under the base classifiers and then the meta classifier makes the final outcome based on the base classifier's outputs. There are many alternative ways how the stacking technique works. One of them is the output of the base level classifiers are used as inputs for the meta learner. Another way is that the probability of the base level classifiers can be used as features for the meta classifier [39].

For example, let X_1 , X_2 and X_3 be three base level learners and Y be the meta classifier. At first the training dataset will go through the base learners that is X_1 , X_2 and X_3 . Finally, the overall output will be decided by the meta learner Y .

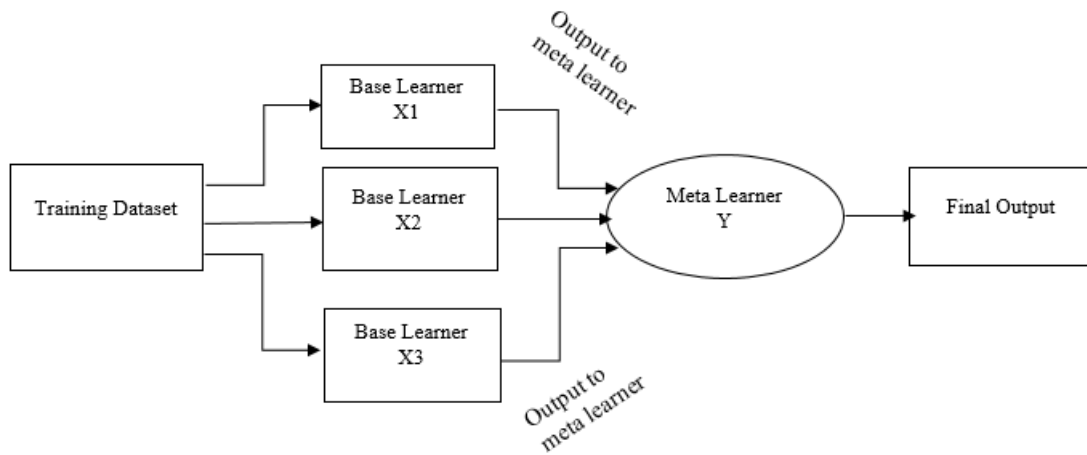


Figure 2.8: Stacking Technique Framework

Stacking technique is a great way to solve overfitting issue because of terms like cross validation and regularization. This method is also useful in real life problems as the method can handle huge amount of data and still gives a less error rate and high accuracy [40].

Chapter 3

Dataset Analysis

3.1 Dataset Description

One of the difficult issues that is faced during researches related to detect fraudulent activities is that the available datasets are very scarce in number and the datasets that are available are not public that means that all the information are kept private. Even after that we chose a dataset which is a fabricated dataset initiated by a simulator called PaySim [41] so as to tackle the previously mentioned issues. The original dataset that is private is given by a multinational company which provides mobile financial service around 14 countries all over the world.

type	amount	nameOrig	oldbalanc	newbalan	nameDest	oldbalanc	newbalan	isFraud
PAYMENT	9839.64	C12310068	170136	160296.4	M1979787	0	0	0
PAYMENT	1864.28	C16665442	21249	19384.72	M2044282	0	0	0
TRANSFER	181	C13054861	181	0	C55326406	0	0	1
CASH_OUT	181	C84008367	181	0	C38997010	21182	0	1
PAYMENT	11668.14	C20485377	41554	29885.86	M1230701	0	0	0
PAYMENT	7817.71	C90045638	53860	46042.29	M5734872	0	0	0
PAYMENT	7107.77	C15498889	183195	176087.2	M4080691	0	0	0
PAYMENT	7861.64	C19128504	176087.2	168225.6	M6333263	0	0	0
PAYMENT	4024.36	C12650129	2671	0	M1176932	0	0	0
DEBIT	5337.77	C71241012	41720	36382.23	C19560086	41898	40348.79	0
DEBIT	9644.94	C19003667	4465	0	C99760839	10845	157982.1	0
PAYMENT	3099.97	C24917757	20771	17671.03	M2096539	0	0	0
PAYMENT	2560.74	C16482329	5070	2509.26	M9728652	0	0	0
PAYMENT	11633.76	C17169328	10127	0	M8015691	0	0	0
PAYMENT	4098.78	C10264838	503264	499165.2	M1635378	0	0	0
CASH_OUT	229133.9	C90508043	15325	0	C47640220	5083	51513.44	0
PAYMENT	1563.82	C76175070	450	0	M1731217	0	0	0
PAYMENT	1157.86	C12377626	21156	19998.14	M1877062	0	0	0
PAYMENT	671.64	C20335249	15123	14451.36	M4730532	0	0	0
TRANSFER	215310.3	C16709931	705	0	C11004390	22425	0	0
PAYMENT	1373.43	C20804602	13854	12480.57	M1344519	0	0	0

Figure 3.1: Sample of the dataset

The datasets consist of 1048578 data samples. Among them 70% of the dataset is considered as training dataset and the lrftovrt 30% is considered as testing dataset. For visualizing the dataset that we are using in our dataset, heatmap is used which shows graphical representations of the datasets in colors. It shows individual value in a matrix and according the values, the data is designated in the respective heatmap. It is mainly used for numerical datasets. Seaborn library is used to examine the heatmap of our used dataset which is shown in Figure 3.2.

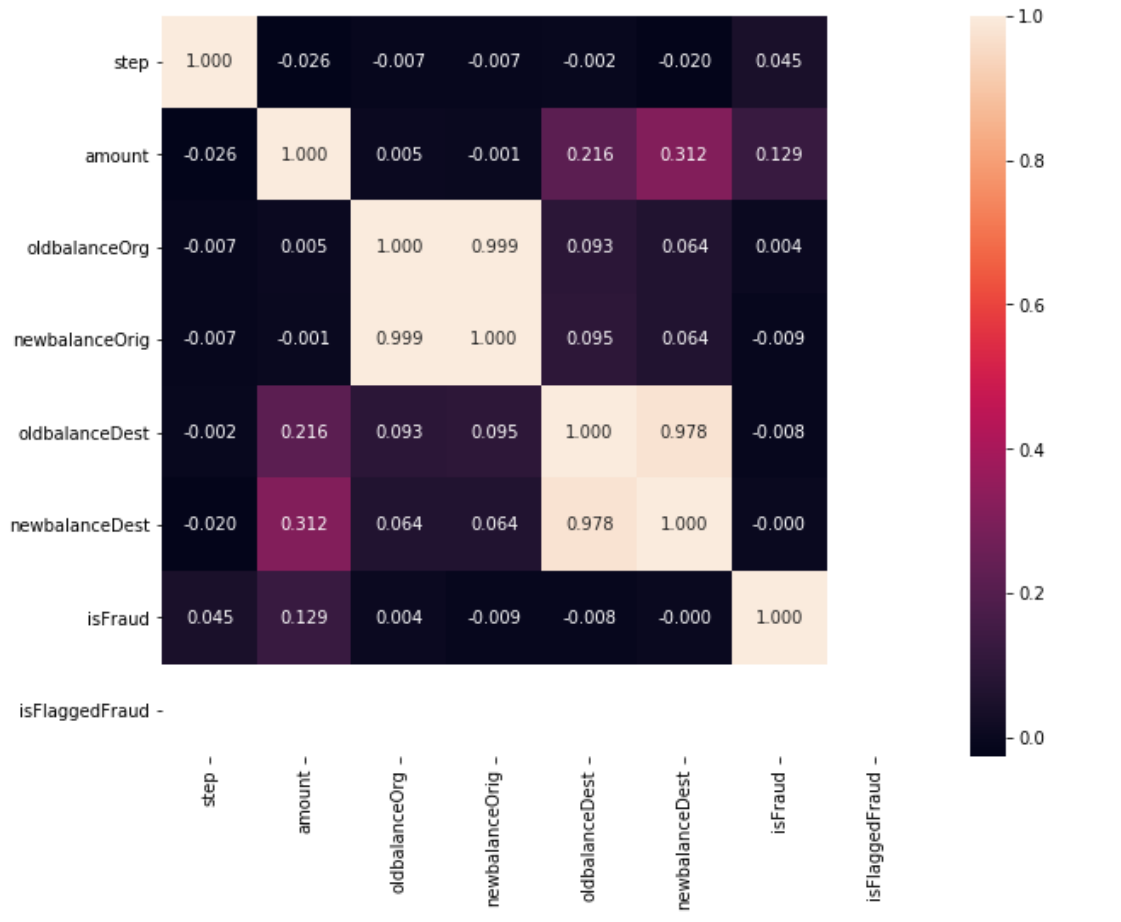


Figure 3.2: Heatmap of the input data

In addition to that, box plot is also being shown in Figure 3.3 which shows how well the data is distributed in a dataset. It is a two dimensional graph which demonstrates the maximum, minimum, midpoint, first quartile and third quartile in the dataset. The x-axis of the graph shows the numeric 'type' that is found in the dataset and the y-axis is the 'amount' which is based on the logarithmic root.

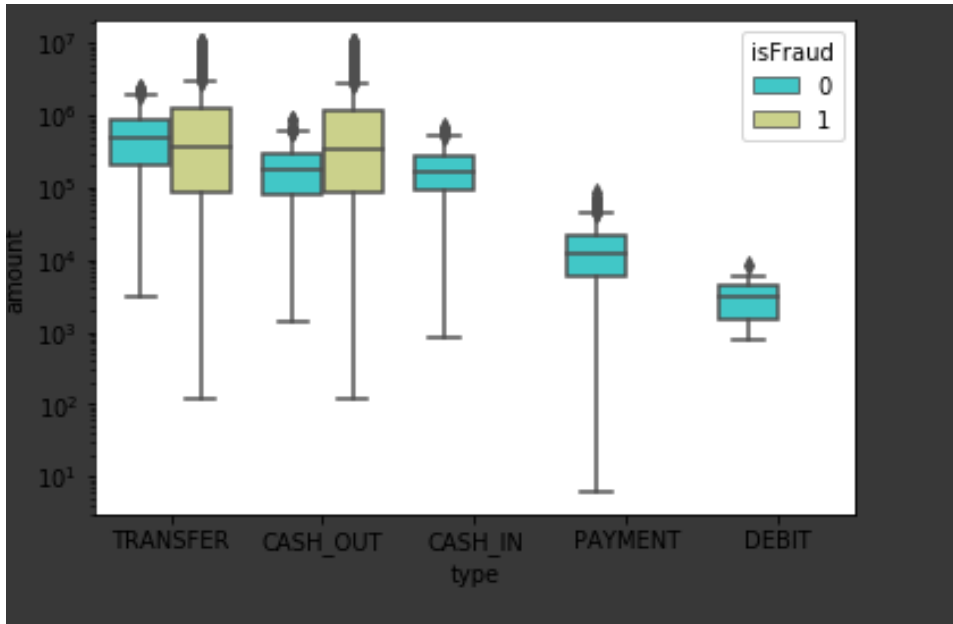


Figure 3.3: Box plot of the dataset

In addition to heat map and box plot data visualization, we are also demonstrating the normal distribution that our dataset is going through. From this distribution, we can easily determine the how the data points of a variable are allocated. There are some certain parameters in normal distribution such as mean and standard deviation. Mean shows the highest point of the curve for every features. Maximum datapoints of that features is found close to that mean value. On the other hand, standard deviation indicates how distanced a data point of a particular feature is from its mean value.

	step	amount	oldbalanceOrig	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
count	10000.000000	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	10000.000000	10000.0
mean	37.818600	2.959185e+05	6.985845e+05	5.785870e+05	1.002366e+06	1.197537e+06	0.11420	0.0
std	9.839409	7.970945e+05	2.458515e+06	2.380834e+06	2.294458e+06	2.440277e+06	0.31807	0.0
min	1.000000	2.190000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.00000	0.0
25%	36.000000	2.127438e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.00000	0.0
50%	36.000000	1.142241e+05	1.491976e+04	0.000000e+00	1.415394e+05	2.936248e+05	0.00000	0.0
75%	37.000000	2.786749e+05	1.503308e+05	5.090342e+04	9.744728e+05	1.300444e+06	0.00000	0.0
max	95.000000	1.000000e+07	2.510000e+07	2.520000e+07	3.570000e+07	3.590000e+07	1.00000	0.0

Figure 3.4: Normal distribution of the dataset

3.2 Dataset Features

The dataset that we are using is composed of several features. These are described below:

type: This feature consists of the actions taken by the customer. There are different types of action or transactions that can be initiated by a customer. First is PAYMENT which implies that the customer is doing a transaction if he or she is paying for a product online or paying any bills that includes current bills, gas bills, tuition fees etc. Next is TRANSFER which implies if the customer is transferring any wealth from one account to another account. After that comes CASHOUT. This action shows that the customer is withdrawing money from his or her account. Next action is CASHIN. This action implies that the customer is storing some money or assets in his or her account. Lastly comes the action called DEBIT. This actions shows a specific amount of money is being taken out from the customer's account.

amount: This features stores the amount of money that will be either payed for an object or bills, transferring from one account to another account, withdrawing from the account or being taken out from the customer's account.

nameOrig: This features stores the ID of the customer who initiated the transaction. Every customer will have their own ID.

oldbalanceOrg: This feature shows what was the old balance of the customer before the start of the transaction.

newbalanceOrig: This feature shows the latest balance of the customer after the transaction is being initiated and fulfilled.

nameDest: This feature stores the ID of the recipient to whom the money was transacted by the customer. Every ID is unique for unique recipient.

oldbalanceDest: This features shows the balance of the recipient before the transaction was initiated by the customer.

newbalanceDest: This feature includes the latest balance of the recipient after the transaction is complete.

isFraud: This is the main features of our research work. This feature gives decision whether the transaction between the customer and the recipient is fraud or not. It is classified by '1' for fraudulent transaction and '0' for non-fraudulent transaction.

3.3 Dataset imbalance and its effect

The situation of an imbalanced dataset can create a false output when the machine learning algorithms are trained on that dataset. If a class of a dataset is considered, then there is a presence of majority class and minority class. When majority class overwhelms the minority class by a huge margin then the imbalance is seen. If machine learning models are used in these type of datasets, then the final output will be heavily influenced by the majority class. This in turn might lead to increase in the overall error rate. The minority class may contain information which are very much essential for the overall generated result [42]. We can easily determine whether a dataset is imbalance or not by analyzing the imbalance ratio of a class. The formula to calculate the imbalance ration is given below.

$$\text{Imbalance ratio of the class} = \frac{\text{total minority datasamples}}{\text{total majority datasamples}} \quad (3.1)$$

After working with the dataset, it is seen that the dataset is highly imbalance for which the machine learning models fail to give an unbiased result. In our thesis research, the class isFraud contains data which is highly imbalance. There are two output which is '1' for fraud transaction and '0' for non-fraudulent transactions. The number of '0' is more than the number of '1' for which '0' is considered as the majority data and '1' is considered as the minority data. The imbalance ratio that is found is very low in our case which states that there is an imbalance in the dataset. This creates a huge problem when the models or the classifiers are being trained. Even though the result will indicate high accuracy but the result will be generated based on the number of dataset which is more that is the majority data '0'. As a result, depending on the accuracy rate, the model cannot be considered a good model.

3.4 Solution for Imbalance data

There are different approaches to tackle the data imbalance issue. In our case, we applied an algorithm known as Near Miss algorithm which is an effective way to handle imbalance dataset and to make the model work without any biased outputs. We also used Synthetic Minority Oversampling Technique (SMOTE) to handle this issue.

3.4.1 Near Miss Algorithm

When it comes to tackling imbalance datasets, one of the ways by which the dataset can be tackled in Near Miss algorithm. Since our thesis topic is based on classifying whether the transaction is fraud or not. In this case the algorithms that we are usually using are classification based algorithms such as Logistic regression, Naïve Bayes classifier, K-nearest neighbor, SVM, Decision tree classifier, Random forest classifier, boosting algorithm etc. These algorithms have a tendency to avoid the minority class and leans its final decision based on the majority class that is a biased output is being generated from these algorithms. As a result, even though the accuracy rate is resulted to be very high but the recall rate and precision rate is low compared to the accuracy rate. For this reason, the constructed models cannot be considered a good model for classifying. Near miss algorithm follows under-sampling technique where the equilibrium between the majority and the minority class is being maintained by decreasing the amount of majority class data. By doing so the ratio between the minority and the majority class decreases and becomes almost equal. Near miss algorithm follows three different versions known as NearMiss-1, NearMiss-2 and NearMiss-3. Each version has a common action to perform which is to figure out the distance between the examples of both majority and minority class. That is the initial stage that the Near miss algorithm goes through. The distance between the examples are usually calculated using the Euclidian distance formula. In NearMiss-1, some of the datasamples from the majority class are selected where the mean distance of the three nearest minority class is regarded as the smallest. Again, in NearMiss-2 the examples of the majority class are selected where the mean distance of the three furthest minority class is regarded as the smallest and finally NearMiss-3 where an example of majority class is selected for each closest minority class. While dealing with the imbalance dataset that we are using for our research we applied Near Miss algorithm version-

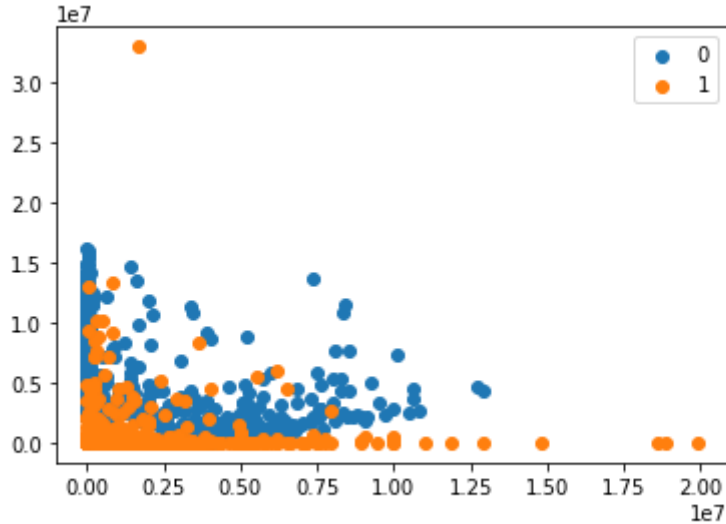


Figure 3.5: Scatter plot of the dataset after implementing Near Miss algorithm

3.4.2 Smote Distribution

SMOTE (Synthetic Minority Oversampling Method) is one of the most commonly used oversampling method to solve the imbalance in the dataset. It produces synthetic method from the minority class. It generates a synthetically balanced or almost class-balanced training set to train the classifier. Smote creates synthetized instances of minority class by operating in the “feature space” rather than the “data space”. Synthetically producing more minority class cases helps the learners to expand their minority class judgment regions. To produce new synthetic minority samples subsequent steps are taken accordingly [43]. Firstly, a variation (minority class sample) between a vector function and one of its nearest neighbours is made. Afterwards, this difference is established to the original feature vector to create a new feature vector.

If the minority class sample is x and the class sample among the k neighbors is y , then the synthetic sample f can be determined by interpolating between x and y as follows [44].

$$f = x + rand(0, 1) * (x - y) \quad (3.2)$$

where $rand(0,1)$ refers to a random number between 0 and 1.

There are many issues that are to be faced while working with SMOTE. While creating fake data points, SMOTE doesn’t think about neighboring data samples that can be from different classes. As a result, classes might imbricate with each other. Due to this many noisy data points might be inserted in the original dataset. Because of this issue, the models that goes though SMOTE process faces overfitting problems.

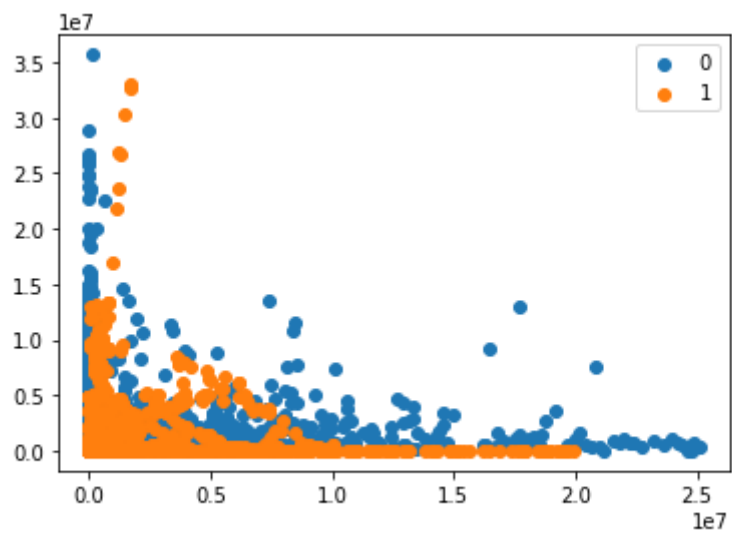


Figure 3.6: Scatter plot of the dataset after implementing SMOTE

Chapter 4

Result Analysis

4.1 Evaluation Metrics

In this section, the study of the overall result shown by the machine learning algorithms implemented by us will be discussed accordingly. In the data preprocessing part, we discussed about the imbalanced dataset and how we tackled that issue by applying near miss algorithm and SMOTE to balance the data. As a result, we will also compare the outputs given by the classifier before and after balancing the dataset. By doing that the analysis will give a better picture of how the classifiers are working in the dataset.

For analyzing the outcomes of the classifiers, it is imperative to know about the evaluation metrics that will give the judgement of whether the applied model is good enough or not. There is a misconception that only classification accuracy can indicate which model works better and faster. This is a vague concept because there are other evaluation metrics other than accuracy which also plays vital role in deciding which classifier gives better output. Before analyzing the outputs of each classifier, we will first describe briefly each of the evaluation metrics that is taken into account while determining the best classifier.

Even though we balanced our dataset, there are some backlash for balancing an imbalanced data. When the dataset is balanced, we are making minority class data points equal to the majority class data points. For this reason, many identical data samples will be created for which it will hamper the evaluation metrics. A balanced accuracy will be shown since we divided the majority and minority class equally. For this, we are considering 0.5 as the threshold for accuracy when we apply our classifiers on a balanced dataset.

Confusion Matrix:

One of the best and easiest way to determine the efficiency of a classification model is by observing the confusion matrix. The confusion matrix can give the output of two or more classes of a dataset. It demonstrates the outcomes in the form of a table consisting of rows and columns. In our case the matrix is 2x2 matrix which contains the true positive (TP), false positive (FP), true negative (TN), and false negative (FN). The confusion matrix can be curved in python by simply by importing confusion matrix from scikit-learn.

	Actual Value Positive	Actual Value Negative
Predicted Value Positive	TP	FP
Predicted Value Negative	FN	TN

Figure 4.1: 02 Scatter plot after using Near Miss

Now the terms that are found in a confusion matrix will be explained serially.

Actual Value: This value indicates the exact data which is stored in the dataset. This values will go through different machine learning algorithms during the training phase. It is also divided into two parts which are positive and negative.

Predicted Value: This value shows the amount of data that has been predicted correctly by the classifiers when the testing phase is done. It is also divided into two portions which are positive and negative.

True Positive (TP): When the machine learning models predicts an outcome in a positive manner and it matches with the actual dataset then it falls under true positive category. For instance, if the model predicts that a certain transaction is fraud and it matches with the actual dataset then that is considered as true positive.

True Negative (TN): When the machine learning models predicts an outcome in a negative manner and it matches with the actual dataset then it falls under true negative category. For instance, if the model predicts that a certain transaction is not fraud and it matches with the actual dataset then that is considered as true negative.

False Positive (FP): When the machine learning models predicts an outcome in a positive manner but the actual dataset shows that the outcome is negative then it falls under false positive category. For instance, if the model predicts that a certain transaction is fraud but in the actual dataset, it shows that the transaction is fraud proof then that is considered as false positive.

False Negative (FN): When the machine learning models predicts an outcome in a positive manner but the actual dataset shows that the outcome is negative then it falls under false positive category. For instance, if the model predicts that a certain transaction is fraud but in the actual dataset, it shows that the transaction is fraud proof then that is considered as false positive.

Classification Accuracy:

Classification accuracy shows the performance done by the machine learning algorithms in predicting the outcome. The value accuracy is done between 0 and 1. The model which gives accuracy rate close to one is considered a good model but has issues with overfitting the dataset and the model which gives accuracy rate close to zero is considered a weak model and also has issues with underfitting the dataset. The classification accuracy also depends on whether a dataset is balanced or unbalanced. A balanced dataset can come up with a better accuracy results when it is trained and tested using a model. Classification accuracy can be calculated from the values that are attained via the confusion matrix. The formula is shown below.

$$\textit{Classification Accuracy} = (TP + TN) / (TP + FP + TN + FN) \quad (4.1)$$

Precision:

Precision is a type of evaluation metric which shows that how correctly the machine learning models guessed the outcome. In our case the precision rate will depend based on the model's performance on correctly identifying the fraud transactions. The high precision rate will say that the model has classified the outcome correctly compared to the wrong outcomes. In other words, the ratio of the true positive and the combination of both the true positive and false positive is considered as precision. The formula is shown below.

$$\textit{Precision} = TP / (TP + FP) \quad (4.2)$$

Recall:

It is another kind of evaluation metric which determines how the machine learning models can detect the positive outcomes after training and testing phase. The higher recall rate indicates that the models are giving higher rate of positive outcomes. In other words, the ratio of the true positive and the combination of the false negative and true positive is called recall. There is always a settlement among recall and precision. The formula to calculate recall is shown below.

$$\textit{Recall} = TP / (TP + FN) \quad (4.3)$$

F1 Score:

The weighted mean between recall and precision is called F1 score. This value ranges from 0 to 1. F1 score also expresses the stability between the precision and recall. This stability relation is directly proportional to the execution of a classifier. The better the balance, the better performance and durable the model will be.

$$\textit{F1 Score} = 2 * (\textit{Recall} * \textit{Precision}) / (\textit{Recall} + \textit{Precision}) \quad (4.4)$$

Macro Average:

This is a type of evaluation metric where the measurements of each classes are calculated and the unweighted average that is found is the macro average.

Weighted Average:

This is another type of metric where the measurements of each classes are calculated and the weighted average which consists of support that is found is considered as the macro average.

Support:

Support is an evaluation metric indicates the number of data of the original dataset that falls in that specific class that means how often that class occurs.

ROC curve:

Receiver Operating Characteristic curve or ROC curve is an essential evaluation metrics in a classification problem. It is a graphical demonstration of a model's performance where the x-axis and y-axis contains the false positive (FP) and true positive (TP) respectively. By screening ROC curve, it can be determined that how a model can easily differentiate different classes of a dataset. The graph can be generated by using python library called scikit-learn.

Plot Learning Curve:

Plot Learning curve is a type of evaluation metric by which someone can understand whether a model is giving overfit, underfit or good fit. The performance of the models can be evaluated in a particular amount of time to see how durable the model is [45]. There are two curves which are training curve and testing curve. The x-axis holds the training set size and the y-axis holds the performance of the model. When a model gives underfitting result, the training curve and the testing curve both separates from each other with time as loss of training dataset takes place. Secondly, when the model gives overfitting result, the training curve at first remains with the testing curve but with time the curve separates which indicates the loss of training dataset. Finally, a model giving a good fit shows that both the curve goes on together that is there is no performance loss.

Hyperparameter Tuning:

While implementing and testing all the machine learning algorithms, we have also implemented hyperparameter tuning in every algorithm. Hyperparameter tuning is a process by which a model's performance curve can be enhanced by selecting different parameters which will instead optimize the model's overall performance. Hyper parameters are the parameters which is randomly selected. Based on those values the classifier will show its outcome. In order to come up with a better accuracy result and other evaluation metrics, hyperparameter tuning is done.

There are important things that is needed to be understood when tuning hyperparameter. There are many possible ways by which a classifier can be tuned [46]. For this, we have to notice which parameter influences the classifier most. Among the possible parameters, it is required to come up with the most influencing as we have less time to work with every available parameter. Moreover, it is imperative to understand how changing one parameter effects the training phase when applying a model on a dataset. This tuning process can be done in two ways. These are:

a) Manual process: In this process, the user has the freedom to give any value to the parameter and check the overall output. He or she can do this process over and over again till an optimal output is generated by the classifier.

b) Automatic process: In this process, the user can use different types of hyperparameter optimization tools such as Hyperas, DeepRepaly, Talos, HyperOpt etc [47].

While performing hyperparameter tuning, we tried to observe if the model is showing either overfitting or underfitting result. According to that, we tried to alter the parameter to get a better fit of the data points [48]. It is to be noted that same parameters will never work for every model that we implemented. A parameter which gives better result for random forest classifier will not guarantee the fact that it will give the same better result to logistic regression if the same values are used in both the algorithms.

For example, the performance of neural network algorithm can be enhanced by using hyperparameter tuning. A major parameter of neural network neurons number. If we increase the number of neurons per layer, then the network will be able to attain more complex decision making information. For this, there is a chance that overfitting might occur rapidly. Same case happens if we increase the layers of the neural network model to attain more knowledge [49]. In addition to that, increasing or decreasing the batch size also results to overfitting and underfitting. Another better way is to regularize each layers and then tuning the parameters will result to better outputs.

Another way by which hyperparameter tuning is done is Grid Search hyperparameter tuning. In this tuning process, it takes all the values of different parameters of a classifier and combines all of them [50]. As a result, a set of values of each parameters is produced. The advantage of using this technique is that one classifier will be able to train the dataset in different parameters at the same time whereas in randomly choosing the parameters will only be able to train for one specific parameter.

4.2 Model Performance

In this part we will witness how all the models that we implemented performed to detect fraud in transaction dataset. This will contain both the results of imbalance dataset and balanced dataset.

4.2.1 Logistic regression

Imbalance dataset:

After implementing the logistic regression model in the imbalanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	0.96	0.99	0.97	2638
1	0.88	0.68	0.77	362
accuracy			0.95	3000
macro avg	0.92	0.83	0.87	3000
weighted avg	0.95	0.95	0.95	3000

Figure 4.2: Classification report of Logistic Regression when applied in imbalanced dataset

From the classification report we see that the accuracy is 0.95. For the transactions that are not fraud, the precision, recall and f1-score is 0.96, 0.99 and 0.97 respectively. On the other hand, the transactions that are fraud gives the value 0.88, 0.68 and 0.77 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-eight whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 0.92, 0.83 and 0.87 respectively and for the weighted average, precision, recall and f1-score is 0.95 for all the three metrics.

Secondly, we will examine the plot learning curve for Logistic Regression for imbalance dataset.

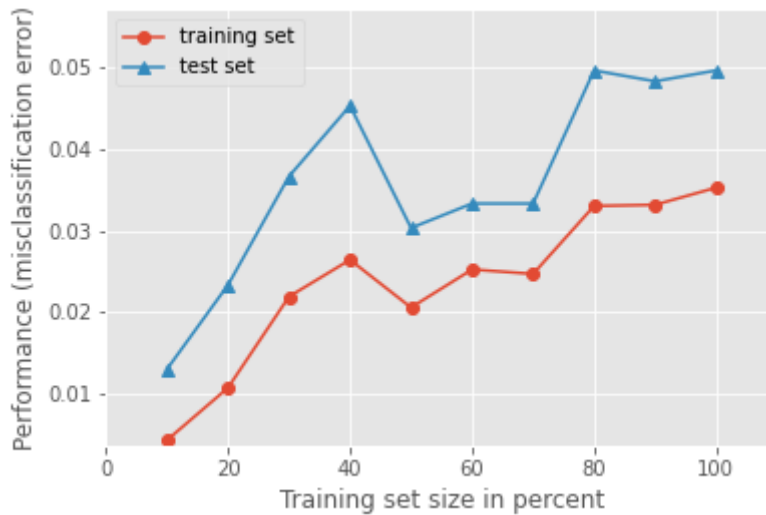


Figure 4.3: Plot learning curve of Logistic regression on imbalance dataset

Finally, the confusion matrix that is established after applying logistic regression in the imbalanced dataset is shown below.

$$\begin{bmatrix} 2604 & 34 \\ 115 & 247 \end{bmatrix}$$

Figure 4.4: Confusion Matrix of Logistic Regression in imbalance dataset

From the confusion matrix, we see that the value of true positive is two thousand six hundred and four, true negative is two hundred and forty-seven, false positive is thirty-four and false negative is one hundred and fifteen.

Balance dataset (Near Miss Application):

After implementing the logistic regression model in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	0.87	0.41	0.56	2214
1	0.10	0.51	0.17	286
accuracy			0.42	2500
macro avg	0.48	0.46	0.36	2500
weighted avg	0.78	0.42	0.51	2500

Figure 4.5: Classification report of Logistic Regression after near miss application

From the classification report we see that the accuracy is 0.42. For the transactions that are not fraud, the precision, recall and f1-score is 0.87, 0.41 and 0.56 respectively. On the other hand, the transactions that are fraud gives the value 0.10, 0.51 and 0.17 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand two hundred and fourteen whereas for fraudulent transactions, the number of data detected is only two hundred and eighty-six.

In addition to that, the macro average for precision, recall and f1-score is 0.48, 0.46 and 0.36 respectively and for the weighted average, precision, recall and f1-score is 0.78, 0.42 and 0.51 respectively.

Now, the confusion matrix that is established after applying logistic regression in the balanced dataset is shown below.

$$\begin{bmatrix} 907 & 1307 \\ 141 & 145 \end{bmatrix}$$

Figure 4.6: Confusion Matrix of Logistic Regression after near miss application

From the confusion matrix, we see that the value of true positive is nine hundred and seven, true negative is one hundred and forty-five, false positive is one thousand three hundred and seven and false negative is one hundred and one.

Balance dataset (SMOTE Application):

After implementing the logistic regression model in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	0.96	0.99	0.97	2638
1	0.88	0.68	0.77	362
accuracy			0.95	3000
macro avg	0.92	0.83	0.87	3000
weighted avg	0.95	0.95	0.95	3000

Figure 4.7: Classification report of Logistic Regression after SMOTE application

From the classification report we see that the accuracy is 0.95. For the transactions that are not fraud, the precision, recall and f1-score is 0.96, 0.99 and 0.97 respectively. On the other hand, the transactions that are fraud gives the value 0.88, 0.68 and 0.77 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand two hundred and fourteen whereas for fraudulent transactions, the number of data detected is only two hundred and eighty-six.

In addition to that, the macro average for precision, recall and f1-score is 0.92, 0.83 and 0.87 respectively and for the weighted average, precision, recall and f1-score is 0.95 for all the evaluation metrics.

Secondly, we will examine the plot learning curve for Logistic Regression for balanced dataset.

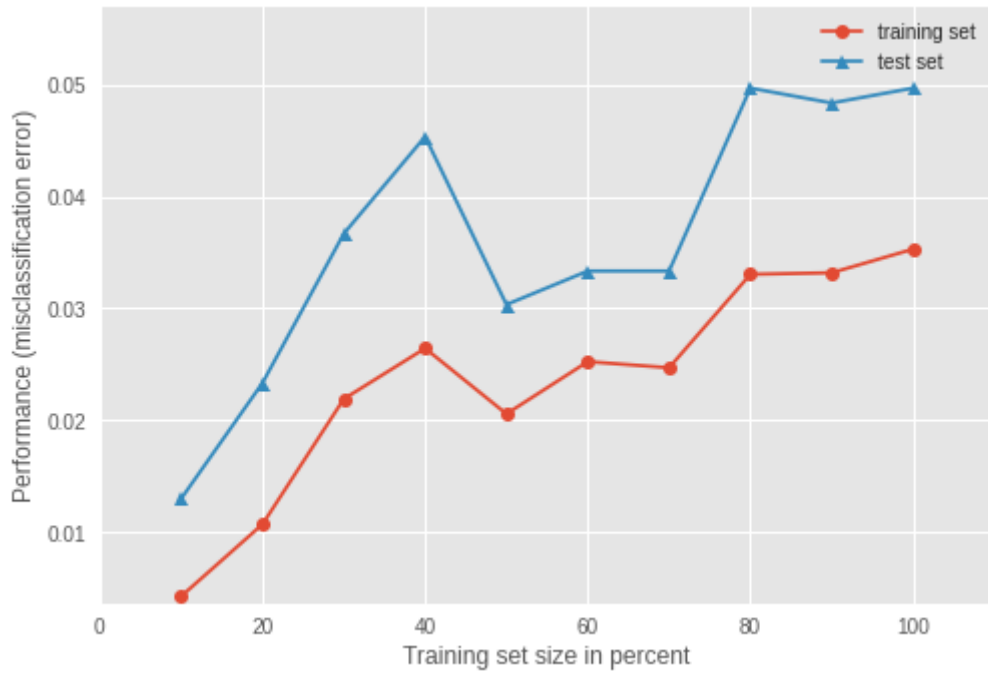


Figure 4.8: Plot learning curve for Logistic Regression on balanced dataset

Finally, the confusion matrix that is established after applying logistic regression in the balanced dataset is shown below.

$$\begin{bmatrix} 2604 & 34 \\ 115 & 247 \end{bmatrix}$$

Figure 4.9: Confusion Matrix of Logistic Regression after SMOTE application

From the confusion matrix, we see that the value of true positive is two thousand six hundred and four, true negative is two hundred and forty-seven, false positive is thirty-four and false negative is one hundred and fifteen.

Comparing the three classification reports and confusion matrix, it is seen that if logistic regression is implemented on the imbalance dataset then overfitting is seen. SMOTE was unable to solve the issue as well but if after applying Near Miss Algorithm, it is seen that the classifier now gives a balanced accuracy score. Even though the accuracy is less, it is close to the threshold.

4.2.2 Decision Tree

Imbalance dataset:

After implementing the Decision Tree model in the imbalanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2650
1	1.00	0.98	0.99	350
accuracy			1.00	3000
macro avg	1.00	0.99	0.99	3000
weighted avg	1.00	1.00	1.00	3000

Figure 4.10: Classification report of Decision Tree when applied in imbalanced dataset

From the classification report we see that the accuracy is 1.00. For the transactions that are not fraud, the precision, recall and f1-score is 1.00 for all the metrics. On the other hand, the transactions that are fraud gives the value 1.00, 0.98 and 0.99 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and fifty whereas for fraudulent transactions, the number of data detected is only three hundred and fifty.

In addition to that, the macro average for precision, recall and f1-score is 1.00, 0.99 and 0.99 respectively and for the weighted average, precision, recall and f1-score is 1.00 for all the three metrics.

Secondly, we will examine the plot learning curve for Decision Tree on imbalanced dataset.

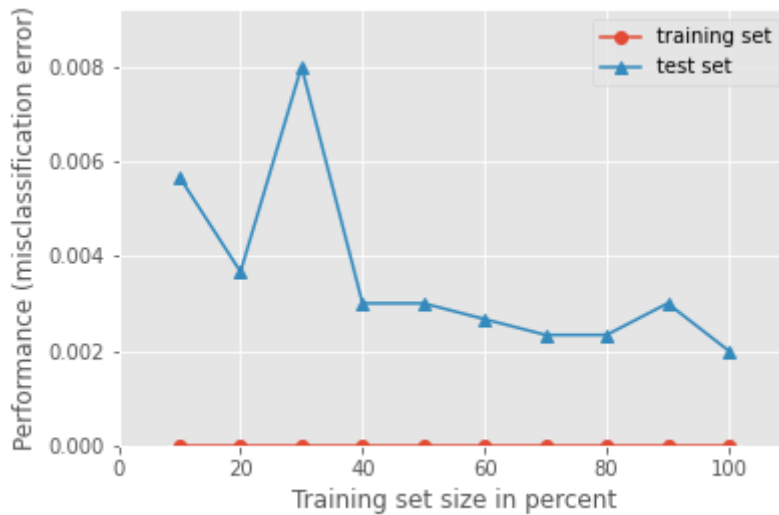


Figure 4.11: Plot learning curve for Decision Tree on imbalance dataset

Finally, the confusion matrix that is established after applying Decision Tree in the imbalanced dataset is shown below.

$$\begin{bmatrix} [2649 & 1] \\ [8 & 342] \end{bmatrix}$$

Figure 4.12: Confusion Matrix of Decision Tree in the imbalance dataset

From the confusion matrix, we see that the value of true positive is two thousand six hundred and forty-nine, true negative is three hundred and forty-two, false positive is one and false negative is eight.

Balance Dataset (Near Miss Application):

After implementing the Decision Tree model in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	1.00	0.42	0.59	2214
1	0.18	1.00	0.31	286
accuracy			0.49	2500
macro avg	0.59	0.71	0.45	2500
weighted avg	0.91	0.49	0.56	2500

Figure 4.13: Classification report of Decision Tree after near miss application

From the classification report we see that the accuracy is 0.49. For the transactions that are not fraud, the precision, recall and f1-score is 1.00, 0.42 and 0.59 respectively. On the other hand, the transactions that are fraud gives the value 0.18, 1.00 and 0.31 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand two hundred and fourteen whereas for fraudulent transactions, the number of data detected is only two hundred and eighty-six.

In addition to that, the macro average for precision, recall and f1-score is 0.59, 0.71 and 0.45 respectively and for the weighted average, precision, recall and f1-score is 0.91, 0.49 and 0.56 respectively.

Now, the confusion matrix that is established after applying decision tree model in the balanced dataset is shown below.

$$\begin{bmatrix} 935 & 1279 \\ 0 & 286 \end{bmatrix}$$

Figure 4.14: Confusion Matrix of Decision Tree after near miss application

From the confusion matrix, we see that the value of true positive is nine hundred and thirty-five, true negative is two hundred and eighty-six, false positive is one thousand two hundred and seventy-nine and false negative is zero.

Balance dataset (SMOTE Application):

After implementing the decision tree model in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2638
1	0.99	0.99	0.99	362
accuracy			1.00	3000
macro avg	0.99	1.00	1.00	3000
weighted avg	1.00	1.00	1.00	3000

Figure 4.15: Classification report of Decision Tree after SMOTE application

From the classification report we see that the accuracy is 1.00. For the transactions that are not fraud, the precision, recall and f1-score is all 1.00. On the other hand, the transactions that are fraud gives the value 1.00 for the precision, recall and f1-score. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-eight whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 0.99, 1.00 and 1.00 respectively and for the weighted average, precision, recall and f1-score is 1.00 for all the evaluation metrics.

Secondly, we will examine the plot learning curve for Decision tree on balanced dataset.

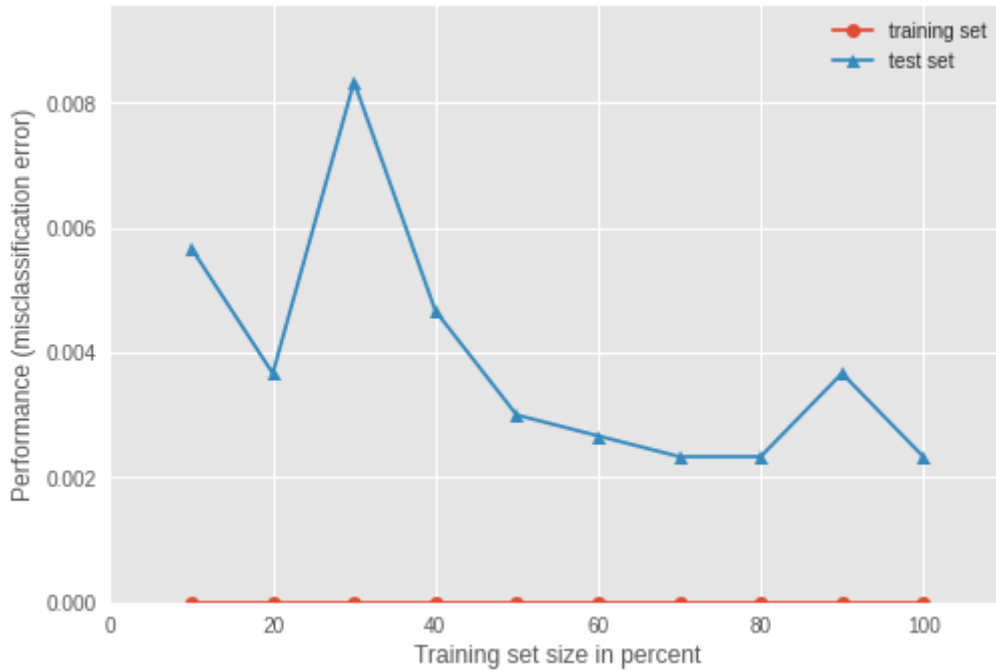


Figure 4.16: Plot learning curve for Decision tree on balanced dataset

Finally, the confusion matrix that is established after applying decision tree in the balanced dataset is shown below.

$$\begin{bmatrix} [2634 & 4] \\ [2 & 360] \end{bmatrix}$$

Figure 4.17: Confusion Matrix of Decision Tree after SMOTE application

From the confusion matrix, we see that the value of true positive is two thousand six hundred and thirty-four, true negative is three hundred and sixty, false positive is four and false negative is two.

Comparing the three classification reports and confusion matrix, it is seen that if decision tree is implemented on the imbalance dataset then overfitting is seen. SMOTE was unable to solve the issue as well but if after applying Near Miss Algorithm, it is seen that the classifier now gives a balanced accuracy score. Even though the accuracy is less, it is almost equal to the threshold.

4.2.3 Random Forest

Imbalance dataset:

After implementing the Random Forest model in the imbalanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2638
1	1.00	0.98	0.99	362
accuracy			1.00	3000
macro avg	1.00	0.99	0.99	3000
weighted avg	1.00	1.00	1.00	3000

Figure 4.18: Classification report of Random Forest when applied in imbalanced dataset

From the classification report we see that the accuracy is 1.00 just like Decision Tree. For the transactions that are not fraud, the precision, recall and f1-score is 1.00 for all the metrics. On the other hand, the transactions that are fraud gives the value 1.00, 0.98 and 0.99 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-right whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 1.00, 0.99 and 0.99 respectively and for the weighted average, precision, recall and f1-score is 1.00 for all the three metrics.

Secondly, we will examine the plot learning curve of Random Forest on the imbalanced dataset.

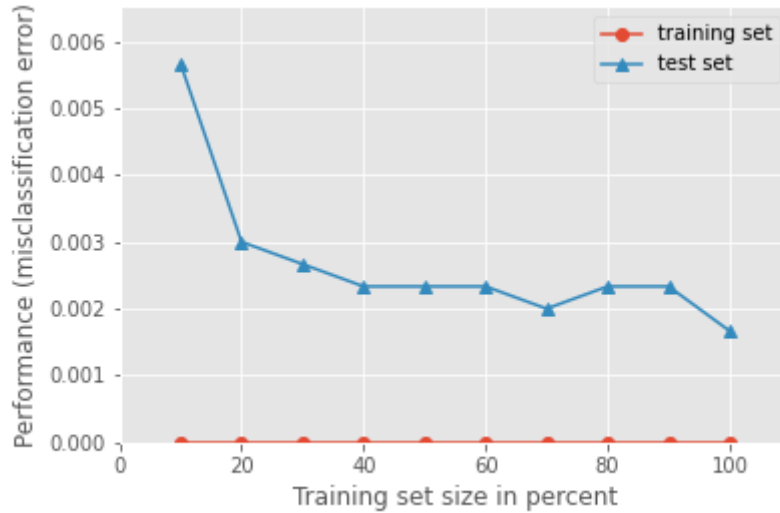


Figure 4.19: Plot learning curve of Random Forest on the imbalanced dataset

Finally, the confusion matrix that is established after applying Random Forest in the imbalanced dataset is shown below.

$$\begin{bmatrix} 2637 & 1 \\ 7 & 355 \end{bmatrix}$$

Figure 4.20: Confusion Matrix of Random Forest in the imbalance dataset

From the confusion matrix, we see that the value of true positive is two thousand six hundred and thirty-seven, true negative is three hundred and fifty-five, false positive is one and false negative is seven.

Balance Dataset (Near Miss Application):

After implementing the Random Forest model in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	1.00	0.48	0.65	2214
1	0.20	0.99	0.33	286
accuracy			0.54	2500
macro avg	0.60	0.74	0.49	2500
weighted avg	0.91	0.54	0.61	2500

Figure 4.21: Classification report of Random Forest after near miss application

From the classification report we see that the accuracy is 0.54. For the transactions that are not fraud, the precision, recall and f1-score is 1.00, 0.48 and 0.65 respectively. On the other hand, the transactions that are fraud gives the value 0.20, 0.99 and 0.33 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand two hundred and fourteen whereas for fraudulent transactions, the number of data detected is only two hundred and eighty-six.

In addition to that, the macro average for precision, recall and f1-score is 0.60, 0.74 and 0.49 respectively and for the weighted average, precision, recall and f1-score is 0.91, 0.54 and 0.61 respectively.

Now, the confusion matrix that is established after applying Random Forest in the balanced dataset is shown below.

$$\begin{bmatrix} 1066 & 1148 \\ 2 & 284 \end{bmatrix}$$

Figure 4.22: Confusion Matrix of Random Forest after near miss application

From the confusion matrix, we see that the value of true positive is one thousand and sixty-six, true negative is two hundred and eighty-four, false positive is one thousand one hundred and forty-eight and false negative is two.

Balance dataset (SMOTE Application):

After implementing the Random Forest model in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2638
1	1.00	0.98	0.99	362
accuracy			1.00	3000
macro avg	1.00	0.99	0.99	3000
weighted avg	1.00	1.00	1.00	3000

Figure 4.23: Classification report of Random Forest after SMOTE application

From the classification report we see that the accuracy is 1.00. For the transactions that are not fraud, the precision, recall and f1-score is all 1.00. On the other hand, the transactions that are fraud gives the value 1.00, 0.98 and 0.99 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-eight whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 1.00, 0.99 and 0.99 respectively and for the weighted average, precision, recall and f1-score is 1.00 for all the evaluation metrics.

Secondly, we will examine the plot learning curve of Random Forest on the balanced dataset.

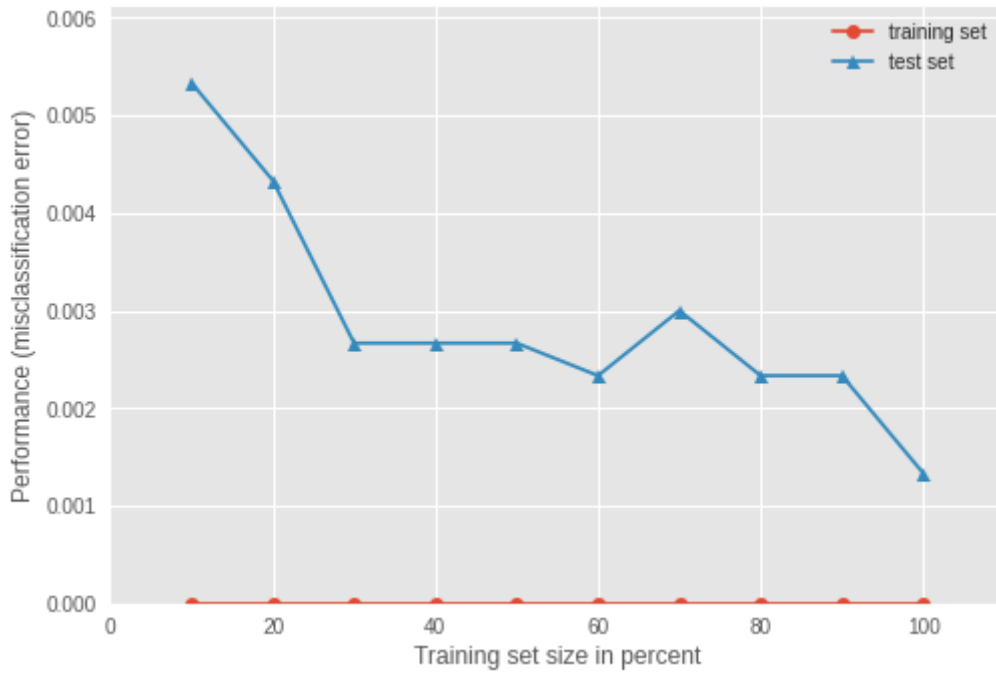


Figure 4.24: Plot learning curve of Random Forest on the balanced dataset

Finally, the confusion matrix that is established after applying Random Forest in the balanced dataset is shown below.

$$\begin{bmatrix} 2637 & 1 \\ 6 & 356 \end{bmatrix}$$

Figure 4.25: Confusion Matrix of Random Forest after SMOTE application

From the confusion matrix, we see that the value of true positive is two thousand six hundred and thirty-seven, true negative is three hundred and fifty-six, false positive is one and false negative is six.

Comparing the three classification reports and confusion matrix, it is seen that if Random Forest is implemented on the imbalance dataset then overfitting is seen. SMOTE was unable to solve the issue as well but if after applying Near Miss Algorithm, it is seen that the classifier now gives a balanced accuracy score. Even though the accuracy is less, it is almost equal to the threshold.

4.2.4 K-Nearest Neighbor

Imbalance dataset:

After implementing the K-Nearest Neighbor model in the imbalanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	0.97	0.98	0.98	2638
1	0.85	0.79	0.82	362
accuracy			0.96	3000
macro avg	0.91	0.88	0.90	3000
weighted avg	0.96	0.96	0.96	3000

Figure 4.26: Classification report of K-Nearest Neighbor when applied in imbalanced dataset

From the classification report we see that the accuracy is 0.96. For the transactions that are not fraud, the precision, recall and f1-score is 0.97, 0.98 and 0.98 respectively. On the other hand, the transactions that are fraud gives the value 0.85, 0.79 and 0.82 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-eight whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 0.91, 0.88 and 0.90 respectively and for the weighted average, precision, recall and f1-score is 0.96 for all the three metrics.

Secondly, we will examine the plot learning curve of K-Nearest Neighbor on the imbalanced dataset.

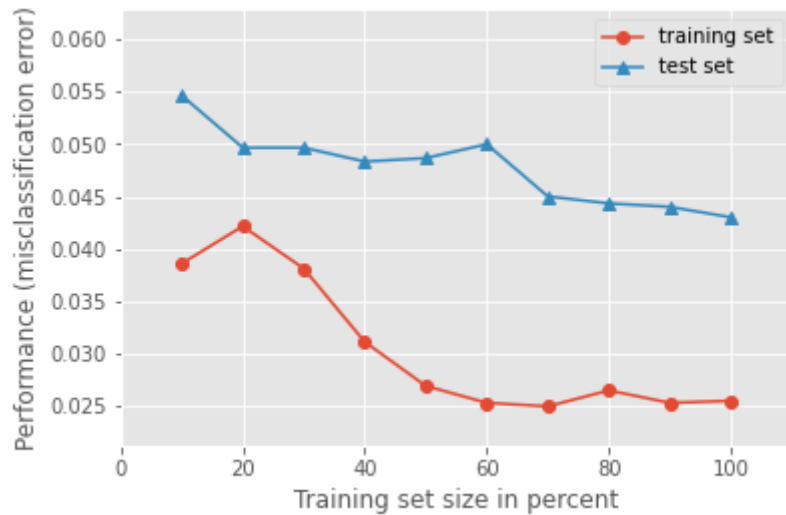


Figure 4.27: Plot learning curve of K-Nearest Neighbor on the imbalanced dataset

Finally, the confusion matrix that is established after applying K-Nearest Neighbor in the imbalanced dataset is shown below.

$$\begin{bmatrix} [2586 & 52] \\ [77 & 285] \end{bmatrix}$$

Figure 4.28: Confusion Matrix of K-Nearest Neighbor in the imbalance dataset

From the confusion matrix, we see that the value of true positive is two thousand five hundred and eighty-six, true negative is two hundred and eighty-five, false positive is fifty-two and false negative is seventy-seven.

Balance Dataset (Near Miss Application):

After implementing the K-Nearest Neighbor model in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	0.99	0.25	0.40	2214
1	0.14	0.98	0.25	286
accuracy			0.33	2500
macro avg	0.57	0.62	0.33	2500
weighted avg	0.89	0.33	0.38	2500

Figure 4.29: Classification report of K-Nearest Neighbor after near miss application

From the classification report we see that the accuracy is 0.33. For the transactions that are not fraud, the precision, recall and f1-score is 0.99, 0.25 and 0.40 respectively. On the other hand, the transactions that are fraud gives the value 0.14, 0.98 and 0.25 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand two hundred and fourteen whereas for fraudulent transactions, the number of data detected is only two hundred and eighty-six.

In addition to that, the macro average for precision, recall and f1-score is 0.57, 0.62 and 0.33 respectively and for the weighted average, precision, recall and f1-score is 0.89, 0.33 and 0.38 respectively.

Now, the confusion matrix that is established after applying K-Nearest Neighbor in the balanced dataset is shown below.

$$\begin{bmatrix} 553 & 1661 \\ 5 & 281 \end{bmatrix}$$

Figure 4.30: Confusion Matrix of K-Nearest Neighbor after near miss application

From the confusion matrix, we see that the value of true positive is five hundred and fifty-three, true negative is two hundred and eighty-one, false positive is one thousand six hundred and sixty-one and false negative is five.

Balance dataset (SMOTE Application):

After implementing the K-Nearest Neighbor model in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	0.97	0.98	0.98	2638
1	0.85	0.79	0.82	362
accuracy			0.96	3000
macro avg	0.91	0.88	0.90	3000
weighted avg	0.96	0.96	0.96	3000

Figure 4.31: Classification report of K-Nearest Neighbor after SMOTE Application

From the classification report we see that the accuracy is 0.96. For the transactions that are not fraud, the precision, recall and f1-score is all 0.97, 0.98 and 0.98. On the other hand, the transactions that are fraud gives the value 0.85, 0.79 and 0.82 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-eight whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 0.91, 0.88 and 0.90 respectively and for the weighted average, precision, recall and f1-score is 0.96 for all the evaluation metrics.

Secondly, we will examine the plot learning curve of Random Forest on the balanced dataset.

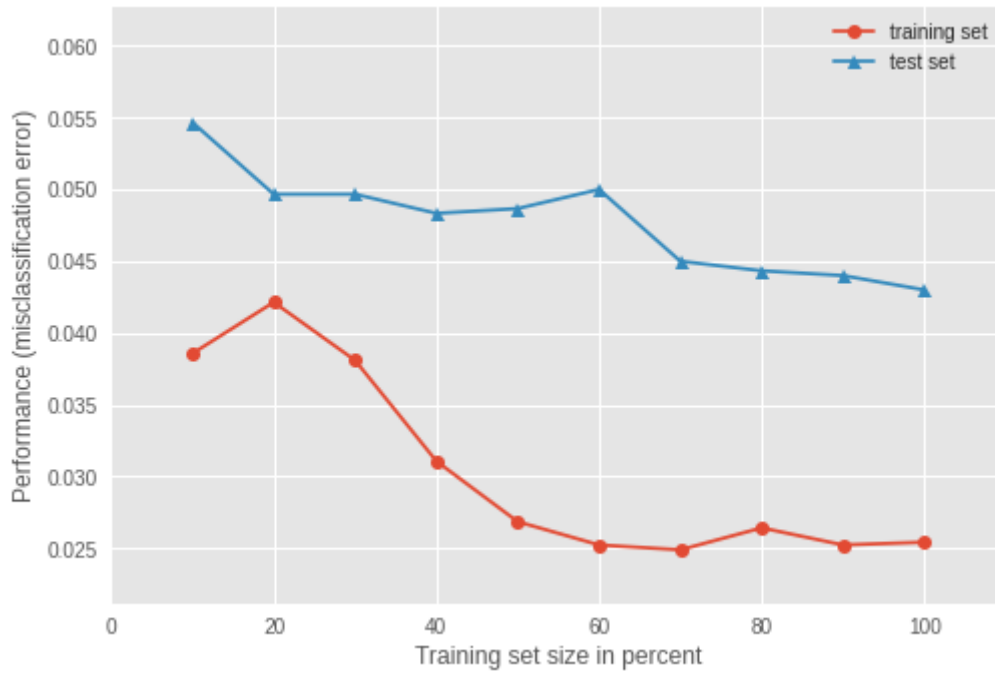


Figure 4.32: Plot learning curve of K-Nearest Neighbor on the balanced Dataset

Finally, the confusion matrix that is established after applying K-Nearest Neighbor in the balanced dataset is shown.

$$\begin{bmatrix} [2586 & 52] \\ [77 & 285] \end{bmatrix}$$

Figure 4.33: Confusion Matrix of K-Nearest Neighbor after SMOTE Application

From the confusion matrix, we see that the value of true positive is two thousand five hundred and eighty-six, true negative is two hundred and eighty-five, false positive is fifty-two and false negative is seventy-seven.

Comparing the three classification reports and confusion matrix, it is seen that if K-Nearest Neighbor is implemented on the imbalance dataset then overfitting is seen. SMOTE was unable to solve the issue as well but if after applying Near Miss Algorithm, it is seen that the classifier now gives the accuracy of 0.33 which can be called a balanced accuracy score since it is within the range of the threshold.

4.2.5 Support Vector Classifier

Imbalance dataset:

After implementing the Support Vector Classifier in the imbalanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	0.93	1.00	0.96	2638
1	0.99	0.43	0.60	362
accuracy			0.93	3000
macro avg	0.96	0.72	0.78	3000
weighted avg	0.94	0.93	0.92	3000

Figure 4.34: Classification report of Support Vector Classifier when applied in imbalanced dataset

From the classification report we see that the accuracy is 0.93. For the transactions that are not fraud, the precision, recall and f1-score is 0.93, 1.00 and 0.96 respectively. On the other hand, the transactions that are fraud gives the value 0.99, 0.43 and 0.60 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-eight whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 0.96, 0.72 and 0.78 respectively and for the weighted average, precision, recall and f1-score is 0.94, 0.93 and 0.92 respectively.

Secondly, we will examine the plot learning curve of Support Vector Classifier on the imbalanced dataset.

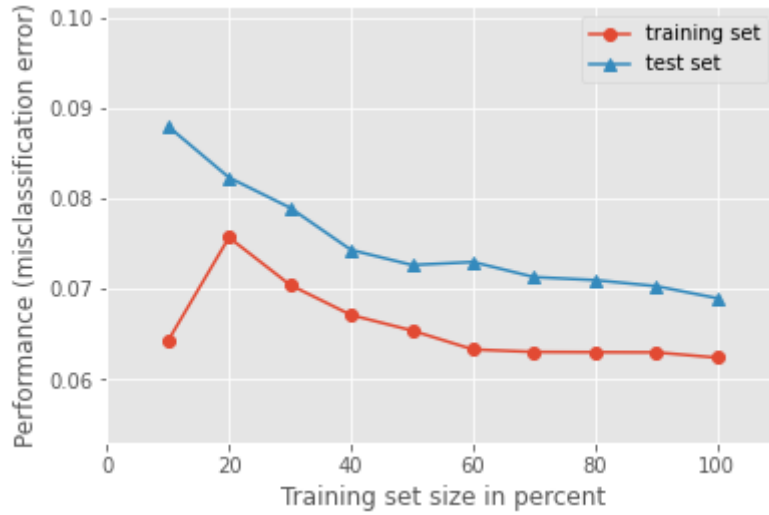


Figure 4.35: Plot learning curve of Support Vector Classifier on the imbalanced dataset

Finally, the confusion matrix that is established after applying Decision Tree in the imbalanced dataset is shown below.

$$\begin{bmatrix} 2637 & 1 \\ 206 & 156 \end{bmatrix}$$

Figure 4.36: Confusion Matrix of Support Vector Classifier in the imbalance dataset

From the confusion matrix, we see that the value of true positive is two thousand six hundred and thirty-seven, true negative is one hundred and fifty-six, false positive is one and false negative is two hundred and six.

Balance Dataset (SMOTE Application):

After implementing the Support Vector Classifier model in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	0.93	1.00	0.96	2638
1	0.99	0.43	0.60	362
accuracy			0.93	3000
macro avg	0.96	0.72	0.78	3000
weighted avg	0.94	0.93	0.92	3000

Figure 4.37: Classification report of Support Vector Classifier after SMOTE application

From the classification report we see all the evaluation metrics that we are using remains unchanged.

Secondly, we will examine the plot learning curve of Random Forest on the balanced dataset.

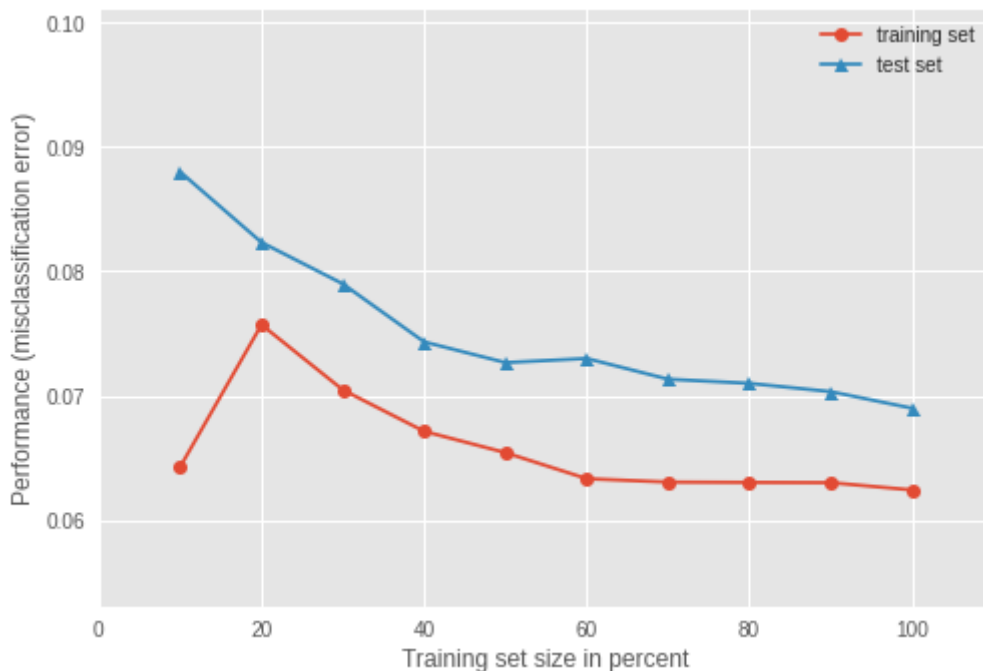


Figure 4.38: Plot learning curve of Support Vector Classifier on the balanced dataset

Finally, the confusion matrix that is established after applying logistic regression in the balanced dataset is shown below.

From the confusion matrix, we see that the value of true positive, true negative, false positive and false negative remains unchanged even though the dataset was balanced.

```

[[2637   1]
 [ 206 156]]

```

Figure 4.39: Confusion Matrix of Support Vector Classifier after SMOTE application

Comparing the three classification reports and confusion matrix, it is seen that if Support Vector Classifier is implemented on the imbalance dataset then overfitting is seen. SMOTE was unable to solve the issue as well. Therefore, it can be said that Support Vector Classifier is not a suitable model if it is used separately but it can perform well if it is used in ensemble learning which will be discussed in the stacking method.

4.2.6 Naïve Bayes Classifier

Imbalance dataset:

After implementing the Naïve Bayes Classifier in the imbalanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	0.97	0.64	0.77	2638
1	0.24	0.85	0.38	362
accuracy			0.67	3000
macro avg	0.61	0.74	0.58	3000
weighted avg	0.88	0.67	0.72	3000

Figure 4.40: Classification report of Naïve Bayes Classifier when applied in imbalanced dataset

From the classification report we see that the accuracy is 0.67. For the transactions that are not fraud, the precision, recall and f1-score is 0.97, 0.64 and 0.77 respectively. On the other hand, the transactions that are fraud gives the value 0.24, 0.85 and 0.38 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-eight whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 0.61, 0.74 and 0.58 respectively and for the weighted average, precision, recall and f1-score is 0.88, 0.67, 0.72 respectively.

Secondly, we will examine the plot learning curve of Naïve Bayes on the imbalanced dataset.

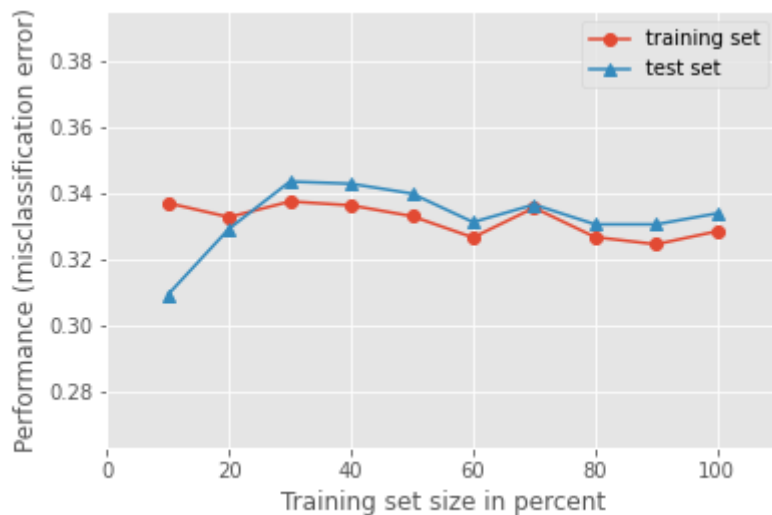


Figure 4.41: Plot learning curve of Naïve Bayes on the imbalanced dataset

Finally, the confusion matrix that is established after applying Naïve Bayes Classifier in the imbalanced dataset is shown below.

$$\begin{bmatrix} [1692 & 946] \\ [56 & 306] \end{bmatrix}$$

Figure 4.42: Confusion Matrix of Naïve Bayes Classifier in the imbalance dataset

From the confusion matrix, we see that the value of true positive is one thousand six hundred and ninety-two, true negative is three hundred and six, false positive is nine hundred and forty-six and false negative is fifty-six.

Balance Dataset (Near Miss Application):

After implementing the Naïve Bayes Classifier in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	0.87	0.41	0.56	2214
1	0.10	0.51	0.17	286
accuracy			0.42	2500
macro avg	0.48	0.46	0.36	2500
weighted avg	0.78	0.42	0.51	2500

Figure 4.43: Classification report of Naïve Bayes Classifier after near miss application

From the classification report we see that the accuracy is 0.42. For the transactions that are not fraud, the precision, recall and f1-score is 0.87, 0.41 and 0.56 respectively. On the other hand, the transactions that are fraud gives the value 0.10, 0.51 and 0.17 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand two hundred and fourteen whereas for fraudulent transactions, the number of data detected is only two hundred and eighty-six.

In addition to that, the macro average for precision, recall and f1-score is 0.48, 0.46 and 0.36 respectively and for the weighted average, precision, recall and f1-score is 0.78, 0.42 and 0.51 respectively.

Now, the confusion matrix that is established after applying Naïve Bayes Classifier in the balanced dataset is shown below.

$$\begin{bmatrix} 907 & 1307 \\ 141 & 145 \end{bmatrix}$$

Figure 4.44: Confusion Matrix of Naïve Bayes Classifier after Near Miss Application

From the confusion matrix, we see that the value of true positive is nine hundred and seven, true negative is one hundred and forty-five, false positive is one thousand three hundred and seven and false negative is one hundred and forty-one.

Balance dataset (SMOTE Application):

After implementing the Naïve Bayes Classifier in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	0.97	0.64	0.77	2638
1	0.24	0.85	0.38	362
accuracy			0.67	3000
macro avg	0.61	0.74	0.58	3000
weighted avg	0.88	0.67	0.72	3000

Figure 4.45: Classification report of Naïve Bayes Classifier after SMOTE application

From the classification report we see that the accuracy is 0.67. For the transactions that are not fraud, the precision, recall and f1-score is 0.97, 0.64 and 0.77 respectively. On the other hand, the transactions that are fraud gives the value 0.24, 0.85 and 0.38 for precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-eight whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 0.61, 0.74 and 0.58 respectively and for the weighted average, precision, recall and f1-score is 0.88, 0.67 and 0.72 respectively.

Secondly, we will examine the plot learning curve of Naïve Bayes on the balanced dataset.

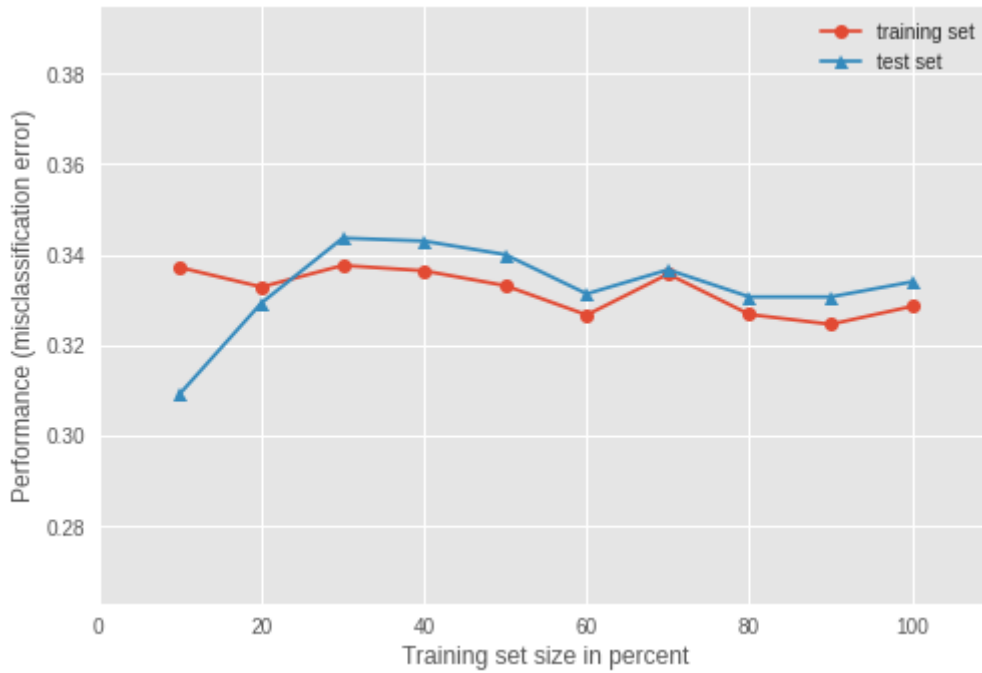


Figure 4.46: Plot learning curve of Naïve Bayes on the balanced dataset

Finally, the confusion matrix that is established after applying Naïve Bayes Classifier in the balanced dataset is shown below.

$$\begin{bmatrix} 1692 & 946 \\ 56 & 306 \end{bmatrix}$$

Figure 4.47: Confusion Matrix of Naïve Bayes Classifier after SMOTE application

From the confusion matrix, we see that the value of true positive is one thousand six hundred and ninety-two, true negative is three hundred and six, false positive is nine hundred and forty-six and false negative is fifty-six.

Comparing the three classification reports and confusion matrix, it is seen that if Naïve Bayes Classifier is implemented on the imbalance dataset then underfitting is seen. Application of SMOTE gives an accuracy result which is acceptable because the accuracy 0.67 is close to the threshold. In addition to that, applying Near Miss Algorithm, it is witnessed that the classifier now gives a balanced accuracy score of 0.42. Even though the accuracy is less, it is almost equal to the threshold.

4.2.7 Gaussian Naïve Bayes Classifier

Imbalance dataset:

After implementing the Gaussian Naïve Bayes Classifier in the imbalanced dataset, the classification report is stated.

	precision	recall	f1-score	support
0	0.92	0.96	0.94	2638
1	0.54	0.36	0.44	362
accuracy			0.89	3000
macro avg	0.73	0.66	0.69	3000
weighted avg	0.87	0.89	0.88	3000

Figure 4.48: Classification report of Gaussian Naïve Bayes Classifier when applied in imbalanced dataset

From the classification report we see that the accuracy is 0.89. For the transactions that are not fraud, the precision, recall and f1-score is 0.92, 0.96 and 0.94 respectively. On the other hand, the transactions that are fraud gives the value 0.54, 0.36 and 0.44 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-eight whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 0.73, 0.66 and 0.69 respectively and for the weighted average, precision, recall and f1-score is 0.87, 0.89, 0.88 respectively.

Secondly, we will examine the plot learning curve of Gaussian Naïve Bayes on the imbalanced dataset.

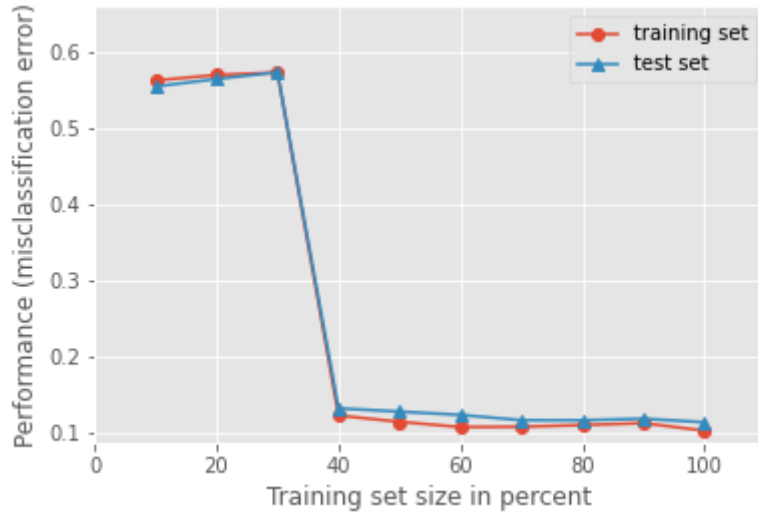


Figure 4.49: Plot learning curve of Gaussian Naïve Bayes on the imbalanced dataset

Finally, the confusion matrix that is established after applying Gaussian Naïve Bayes Classifier in the imbalanced dataset is shown below.

$$\begin{bmatrix} 2527 & 111 \\ 230 & 132 \end{bmatrix}$$

Figure 4.50: Confusion Matrix of Gaussian Naïve Bayes Classifier in the imbalance dataset

From the confusion matrix, we see that the value of true positive is two thousand five hundred and twenty-seven, true negative is one hundred and thirty-two, false positive is one hundred and eleven and false negative is two hundred and thirty.

Balance Dataset (Near Miss Application):

After implementing the Gaussian Naïve Bayes Classifier in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	0.92	0.19	0.32	2214
1	0.12	0.87	0.22	286
accuracy			0.27	2500
macro avg	0.52	0.53	0.27	2500
weighted avg	0.83	0.27	0.31	2500

Figure 4.51: Classification report of Gaussian Naïve Bayes Classifier after near miss application

From the classification report we see that the accuracy is 0.27. For the transactions that are not fraud, the precision, recall and f1-score is 0.92, 0.19 and 0.32 respectively. On the other hand, the transactions that are fraud gives the value 0.12, 0.87 and 0.22 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand two hundred and fourteen whereas for fraudulent transactions, the number of data detected is only two hundred and eighty-six.

In addition to that, the macro average for precision, recall and f1-score is 0.52, 0.53 and 0.27 respectively and for the weighted average, precision, recall and f1-score is 0.83, 0.27 and 0.31 respectively.

Now, the confusion matrix that is established after applying logistic regression in the balanced dataset is shown below.

$$\begin{bmatrix} 425 & 1789 \\ 36 & 250 \end{bmatrix}$$

Figure 4.52: Confusion Matrix of Gaussian Naïve Bayes Classifier after near miss application

From the confusion matrix, we see that the value of true positive is four hundred and twenty-five, true negative is two hundred and fifty, false positive is one thousand seven hundred and eighty-nine and false negative is thirty-six.

Balance dataset (SMOTE Application):

After implementing the logistic regression model in the Gaussian Naïve Bayes Classifier, the classification report is stated below.

	precision	recall	f1-score	support
0	0.92	0.96	0.94	2638
1	0.54	0.36	0.44	362
accuracy			0.89	3000
macro avg	0.73	0.66	0.69	3000
weighted avg	0.87	0.89	0.88	3000

Figure 4.53: Classification report of Gaussian Naïve Bayes Classifier after SMOTE application

From the classification report we see that the accuracy is 0.89. For the transactions that are not fraud, the precision, recall and f1-score is 0.92, 0.96 and 0.94 respectively. On the other hand, the transactions that are fraud gives the value 0.54, 0.36 and 0.44 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-eight whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 0.73, 0.66 and 0.69 respectively and for the weighted average, precision, recall and f1-score is 0.87, 0.89 and 0.88 respectively.

Secondly, we will examine the plot learning curve of Gaussian Naïve Bayes on the balanced dataset.

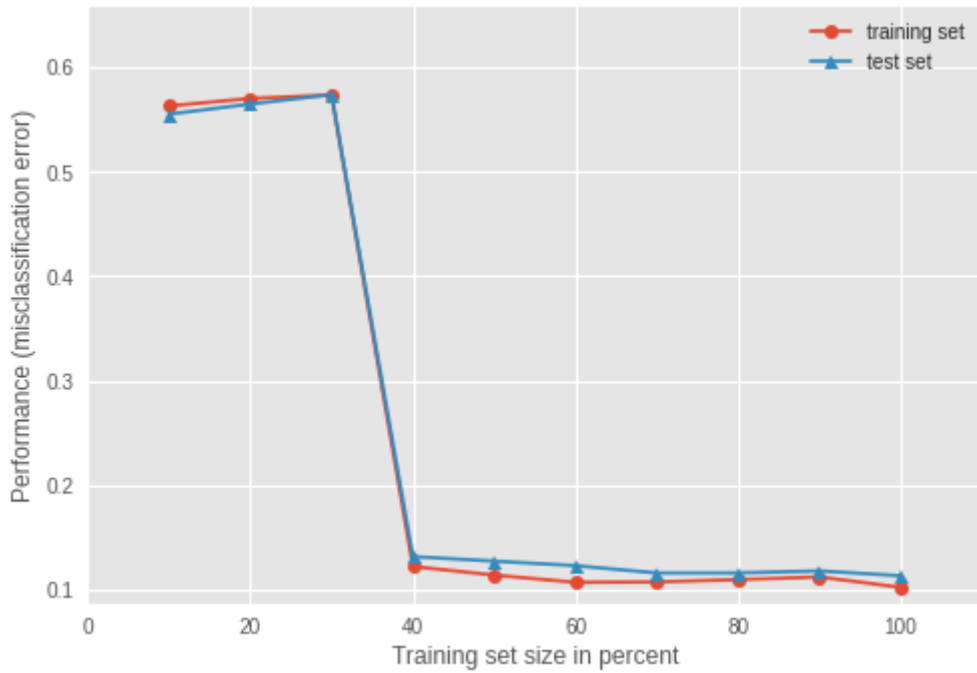


Figure 4.54: Plot learning curve of Gaussian Naïve Bayes on the balanced dataset

Finally, the confusion matrix that is established after applying Gaussian Naïve Bayes Classifier in the balanced dataset is shown below.

$$\begin{bmatrix} [2527 & 111] \\ [230 & 132] \end{bmatrix}$$

Figure 4.55: Confusion Matrix of Gaussian Naïve Bayes Classifier after SMOTE application

From the confusion matrix, we see that the value of true positive is two thousand five hundred and twenty-seven, true negative is one hundred and thirty-two, false positive is one hundred and eleven and false negative is two hundred and thirty.

Comparing the three classification reports and confusion matrix, it is seen that if Gaussian Naïve Bayes Classifier is implemented on the imbalance dataset then overfitting is seen. Both SMOTE and Near Miss algorithm failed to solve the overfitting issue.

4.2.8 Adaptive Boosting

Imbalance dataset:

After implementing the Adaptive boosting in the imbalanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2638
1	1.00	0.98	0.99	362
accuracy			1.00	3000
macro avg	1.00	0.99	0.99	3000
weighted avg	1.00	1.00	1.00	3000

Figure 4.56: Classification report of Adaptive boosting when applied in imbalanced dataset

From the classification report we see that the accuracy is 1.00. For the transactions that are not fraud, the precision, recall and f1-score is 1.00 for all the metrics. On the other hand, the transactions that are fraud gives the value 1.00, 0.98 and 0.99 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-eight whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 1.00, 0.99 and 0.99 respectively and for the weighted average, precision, recall and f1-score is 1.00 for all the three metrics.

Secondly, we will examine the plot learning curve of Adaptive boosting on the imbalanced dataset.

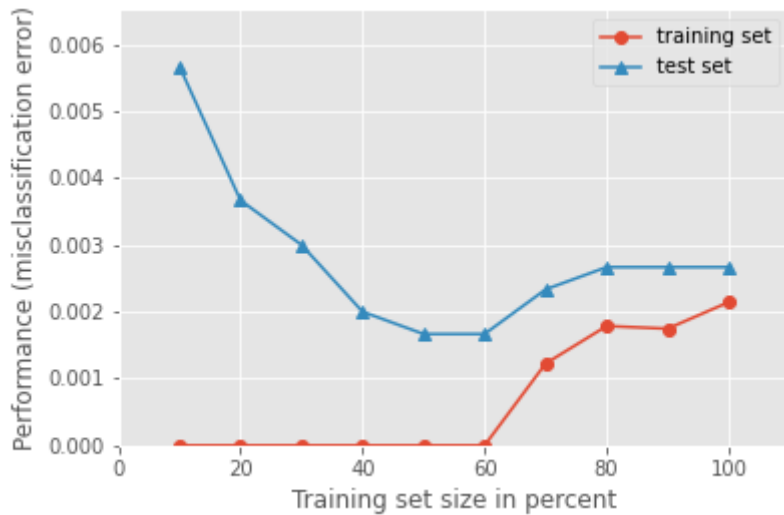


Figure 4.57: Plot learning curve of Adaptive boosting on the imbalanced dataset

Finally, the confusion matrix that is established after applying Adaptive boosting in the imbalanced dataset is shown below.

$$\begin{bmatrix} 2637 & 1 \\ 7 & 355 \end{bmatrix}$$

Figure 4.58: Confusion Matrix of Adaptive boosting in the imbalance dataset

From the confusion matrix, we see that the value of true positive is two thousand six hundred and thirty-seven, true negative is three hundred and fifty-five, false positive is one and false negative is seven.

Balance Dataset (Near Miss Application):

After implementing the Adaptive boosting model in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	1.00	0.41	0.58	2214
1	0.18	1.00	0.30	286
accuracy			0.48	2500
macro avg	0.59	0.71	0.44	2500
weighted avg	0.91	0.48	0.55	2500

Figure 4.59: Classification report of Adaptive boosting after near miss application

From the classification report we see that the accuracy is 0.48. For the transactions that are not fraud, the precision, recall and f1-score is 1.00, 0.41 and 0.58 respectively. On the other hand, the transactions that are fraud gives the value 0.18, 1.00 and 0.30 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand two hundred and fourteen whereas for fraudulent transactions, the number of data detected is only two hundred and eighty-six.

In addition to that, the macro average for precision, recall and f1-score is 0.59, 0.71 and 0.44 respectively and for the weighted average, precision, recall and f1-score is 0.91, 0.48 and 0.55 respectively.

Now, the confusion matrix that is established after applying Adaptive boosting in the balanced dataset is shown below.

$$\begin{bmatrix} 910 & 1304 \\ 0 & 286 \end{bmatrix}$$

Figure 4.60: Confusion Matrix of Adaptive boosting after near miss application

From the confusion matrix, we see that the value of true positive is nine hundred and ten, true negative is two hundred and eighty-six, false positive is one thousand three hundred and four and false negative is zero.

Balance dataset (SMOTE Application):

After implementing the Adaptive boosting in the balanced dataset, the classification report is stated below.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2638
1	1.00	0.98	0.99	362
accuracy			1.00	3000
macro avg	1.00	0.99	0.99	3000
weighted avg	1.00	1.00	1.00	3000

Figure 4.61: Classification report of Adaptive boosting after SMOTE application

From the classification report we see that the accuracy is 1.00. For the transactions that are not fraud, the precision, recall and f1-score is all 1.00. On the other hand, the transactions that are fraud gives the value 1.00, 0.98 and 0.99 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand six hundred and thirty-eight whereas for fraudulent transactions, the number of data detected is only three hundred and sixty-two.

In addition to that, the macro average for precision, recall and f1-score is 1.00, 0.99 and 0.99 respectively and for the weighted average, precision, recall and f1-score is 1.00 for all the evaluation metrics.

Secondly, we will examine the plot learning curve of Adaptive boosting on the balanced dataset.

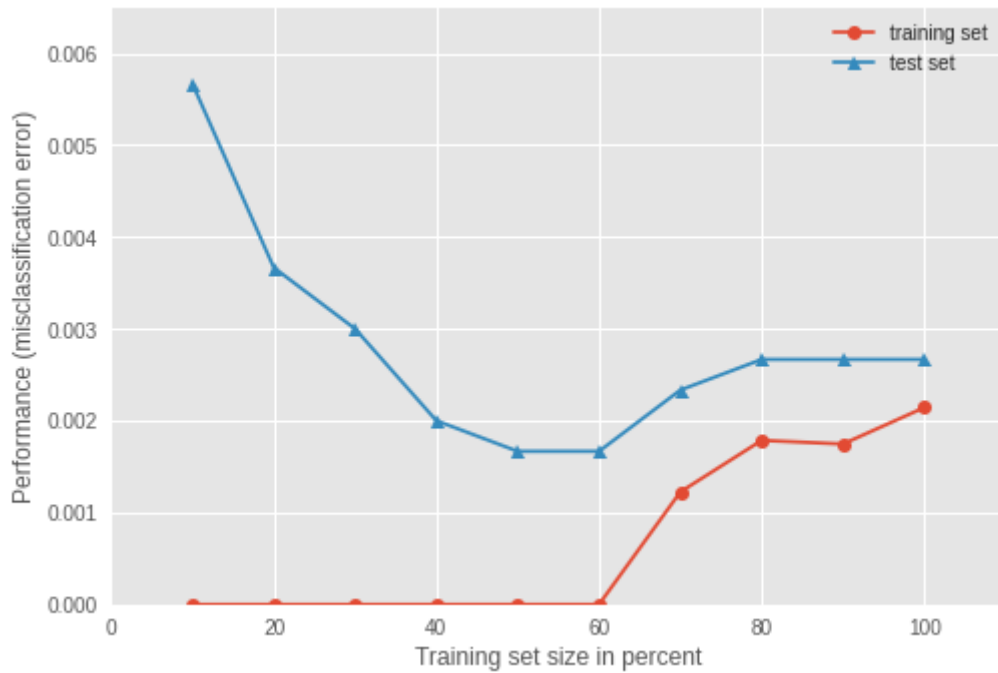


Figure 4.62: Plot learning curve of Adaptive boosting on the balanced dataset

Finally, the confusion matrix that is established after applying Adaptive boosting in the balanced dataset is shown below.

$$\begin{bmatrix} 2637 & 1 \\ 7 & 355 \end{bmatrix}$$

Figure 4.63: Confusion Matrix of Adaptive boosting after SMOTE application

From the confusion matrix, we see that the value of true positive is two thousand six hundred and thirty-seven, true negative is three hundred and fifty-five, false positive is one and false negative is seven.

Comparing the three classification reports and confusion matrix, it is seen that if Adaptive boosting is implemented on the imbalance dataset then high overfitting is seen. SMOTE was unable to solve the issue as well but if after applying Near Miss Algorithm, it is seen that the classifier now gives a balanced accuracy score. Even though the accuracy is less, it is almost equal to the threshold.

4.2.9 Deep Neural Network (Keras)

Deep Neural Network is only applied on the balanced dataset. The following classification report shows the performance of the model.

	precision	recall	f1-score	support
0	0.99	0.85	0.92	856
1	0.87	0.99	0.93	856
accuracy			0.92	1712
macro avg	0.93	0.92	0.92	1712
weighted avg	0.93	0.92	0.92	1712

Figure 4.64: Classification report of Deep Neural Network on balanced dataset

From the classification report we see that the accuracy is 0.92. For the transactions that are not fraud, the precision, recall and f1-score is all 0.99, 0.85 and 0.92 respectively. On the other hand, the transactions that are fraud gives the value 0.87, 0.99 and 0.93 for the precision, recall and f1-score respectively. If we see the support result, for both safe transactions and fraudulent transactions, the number of data detected is eight hundred and fifty-six.

In addition to that, the macro average and weighted average for precision, recall and f1-score is 0.93, 0.92 and 0.92 respectively.

Secondly, we will examine the plot learning curve of Deep Neural Network on the balanced dataset.

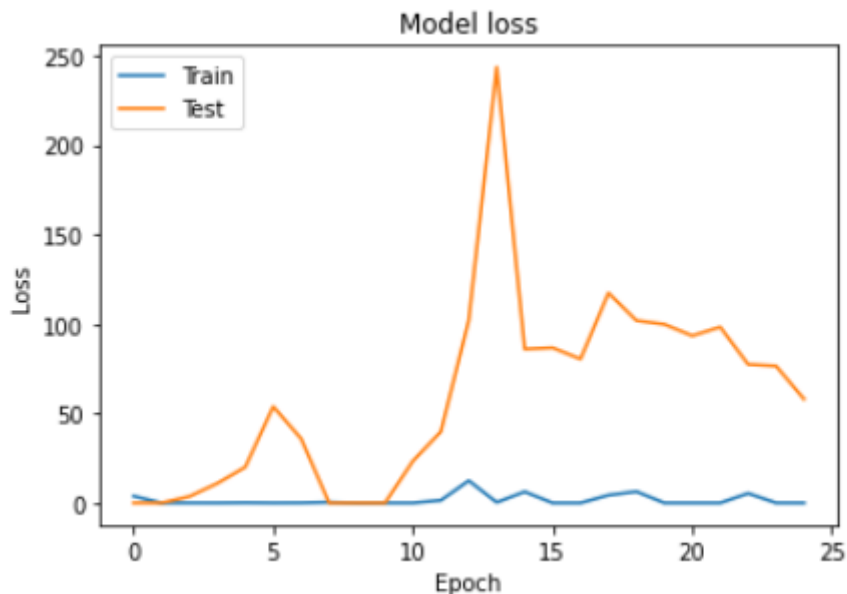


Figure 4.65: Deep Neural Network model loss

Loss is difference between predicted value and our model and true value. We are using cross entropy.

$$Cross - entropy = - \sum_{i=1}^n \sum_{j=1}^m y(i, j) \log(p_{i,j}) \quad (4.5)$$

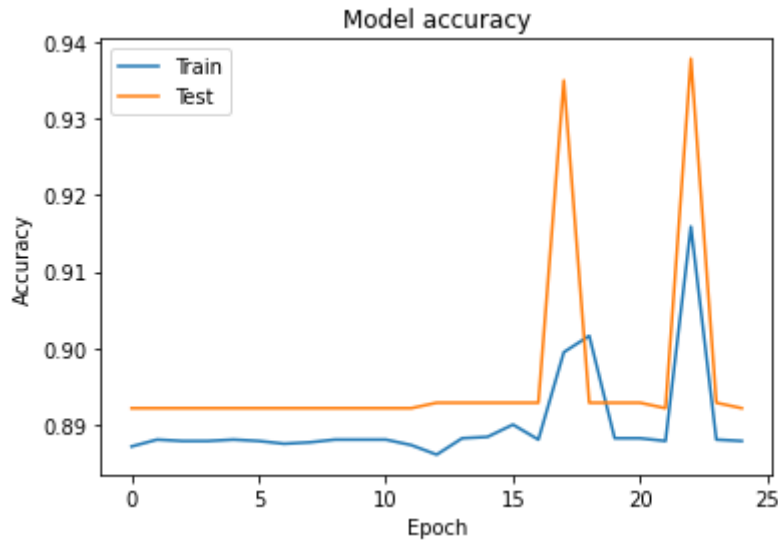


Figure 4.66: Deep Neural Network model accuracy

Finally, the confusion matrix that is established after applying Deep Neural Network in the balanced dataset is shown below.

$$\begin{bmatrix} 729 & 127 \\ 8 & 848 \end{bmatrix}$$

Figure 4.67: Confusion Matrix of Deep Neural Network after Near miss application

From the confusion matrix, we see that the value of true positive is seven hundred and twenty-nine, true negative is eight hundred and forty-eight, false positive is one hundred and twenty-seven and false negative is eight.

From the result analysis, it is seen that applying Deep Neural Network results to high accuracy and also training data loss after some time. As a result, overfitting takes place. Even though, the issue still remains, this model can be used for further experimentation.

4.2.10 Extra Tree Classifier

Extra Tree Classifier is only applied on the balanced dataset. The following classification report shows the performance of a single tuple of this model.

	precision	recall	f1-score	support
0	0.86	0.49	0.63	2214
1	0.08	0.36	0.14	286
accuracy			0.48	2500
macro avg	0.47	0.42	0.38	2500
weighted avg	0.77	0.48	0.57	2500

Figure 4.68: Classification report for one tuple of Extra Tree Classifier

From the classification report we see that the accuracy is 0.48. For the transactions that are not fraud, the precision, recall and f1-score is all 0.86, 0.49 and 0.63 respectively. On the other hand, the transactions that are fraud gives the value 0.08, 0.36 and 0.14 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand two hundred and fourteen whereas for fraudulent transactions, the number of data detected is only two hundred and eighty-six.

In addition to that, the macro average for precision, recall and f1-score is 0.47, 0.42 and 0.38 respectively and for the weighted average, precision, recall and f1-score is 0.77, 0.48 and 0.57 respectively.

Now, the confusion matrix of a single tuple of the Extra Tree Classifier is shown below.

$$\begin{bmatrix} 1091 & 1123 \\ 184 & 102 \end{bmatrix}$$

Figure 4.69: Confusion Matrix of a single tuple after Near miss application

From the confusion matrix, we see that the value of true positive is one thousand and ninety-one, true negative is one hundred and two, false positive is one thousand one hundred and twenty-three and false negative is one hundred and eight-four.

After witnessing the single tuple, we will see the classification report of overall Extra Tree classifier.

	precision	recall	f1-score	support
0	0.96	0.74	0.84	2214
1	0.28	0.79	0.42	286
accuracy			0.75	2500
macro avg	0.62	0.77	0.63	2500
weighted avg	0.89	0.75	0.79	2500

Figure 4.70: Classification report of Extra Tree classifier on the balanced dataset

From the classification report we see that the accuracy is 0.75. For the transactions that are not fraud, the precision, recall and f1-score is all 0.96, 0.74 and 0.84 respectively. On the other hand, the transactions that are fraud gives the value 0.28, 0.79 and 0.42 for the precision, recall and f1-score respectively. If we see the support result, for safe transactions, the number of data detected is two thousand two hundred and fourteen whereas for fraudulent transactions, the number of data detected is only two hundred and eighty-six.

In addition to that, the macro average for precision, recall and f1-score is 0.62, 0.77 and 0.63 respectively and for the weighted average, precision, recall and f1-score is 0.89, 0.75 and 0.79 respectively.

Now, the confusion matrix of a single tuple of the Extra Tree Classifier is shown below.

$$\begin{bmatrix} 1644 & 570 \\ 60 & 226 \end{bmatrix}$$

Figure 4.71: Confusion Matrix of Extra Tree Classifier after Near miss application

From the confusion matrix, we see that the value of true positive is one thousand six hundred and forty-four, true negative is two hundred and twenty-six, false positive is five hundred and seventy and false negative is one hundred and sixty.

Algorithm	Accuracy	0			1		
		Precision	Recall	f1-score	Precision	Recall	f1-score
Logistic Regression	0.950	0.96	0.99	0.97	0.88	0.68	0.77
KNN	0.96	0.97	0.98	0.85	0.85	0.79	0.82
Random Forest	1	1	1	1	1	0.98	0.99
AdaBoost	1	1	1	1	1	0.98	0.99
SVC (kernel)	0.931	0.93	1	0.96	0.99	0.43	0.60
Decision Tree	0.997	1	1	1	0.98	0.98	0.99
Naïve bayes	0.670	0.97	0.64	0.77	0.24	0.85	0.38
Naïve bayes (Gaussian)	0.89	0.92	0.96	0.94	0.54	0.36	0.44

Table 4.1: Experimental results of ML classifiers on imbalanced dataset

Algorithm	Accuracy	0			1		
		Precision	Recall	f1-score	Precision	Recall	f1-score
Logistic Regression	0.420	0.87	0.41	0.56	0.10	0.51	0.17
KNN	0.33	0.99	0.25	0.40	0.14	0.98	0.25
Random Forest	0.540	1	0.48	0.65	0.20	0.99	0.33
AdaBoost	0.478	1	0.41	0.58	0.18	1	0.30
Decision Tree	0.497	1	0.43	0.63	0.19	1	0.31
Naïve bayes	0.420	0.87	0.41	0.56	0.10	0.51	0.17
Naïve bayes (Gaussian)	0.27	0.92	0.19	0.32	0.12	0.87	0.22
Extra tree	0.748	0.96	0.74	0.84	0.28	0.79	0.42
NN	0.920	0.99	0.85	0.92	0.87	0.99	0.93

Table 4.2: Experimental results of ML classifiers on balanced dataset (Near Miss))

Algorithm	Accuracy	0			1		
		Precision	Recall	f1-score	Precision	Recall	f1-score
Logistic Regression	0.950	0.96	0.99	0.97	0.88	0.68	0.77
KNN	0.957	0.97	0.98	0.85	0.85	0.79	0.82
Random Forest	0.997	1	1	1	1	0.98	0.99
AdaBoost	0.997	1	1	1	1	0.98	0.99
SVC (kernel)	0.931	0.93	1	0.96	0.99	0.43	0.60
Decision Tree	0.998	1	1	1	0.99	0.99	0.99
Naïve bayes	0.66	0.97	0.64	0.77	0.24	0.85	0.38
Naïve bayes (Gaussian)	0.886	0.92	0.96	0.96	0.54	0.36	0.44

Table 4.3: Experimental results of ML classifiers on balanced dataset (SMOTE)

Now we will demonstrate the ROC curve for each of the algorithms that we implemented along with their AUC score. This curve is generated based on the results that was found after balancing the dataset using Near Miss algorithm.

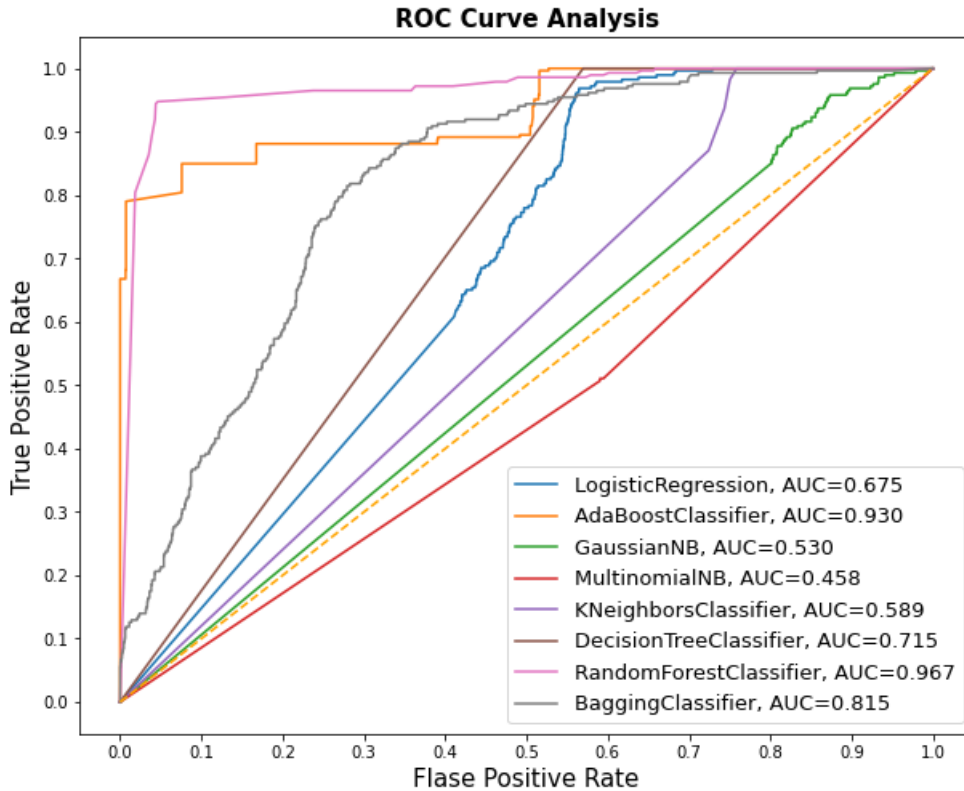


Figure 4.72: ROC curve analysis for all the classifiers (Near Miss)

From the overall result analysis of every algorithm, it can be stated that Extra Tree Classifier works better in detecting fraudulent transactions. The reason behind this claim is when we are forcing noisy datasets in the original dataset in order to balance the biasing decisions, we see that all the algorithm gives low accuracy. For instance, Random forest gives an accuracy of 0.54, Logistic Regression gives an accuracy of 0.42, KNN gives an accuracy of 0.33 etc. In all the cases the accuracy

is very low since they have to tackle the impact of noisy datasets on the original datasets. On the other hand, if we look at Extra Tree Classifier, we find that it gives the best accuracy that is 0.75 at a lower overfitting rate. The reason working behind this is that Extra Tree Classifier works much better even with the noisy datasets. Even though Neural Network is giving an accuracy of 0.92, there is a huge issue of overfitting in this model.

4.2.11 Stacking technique application and comparison

After testing the classifiers separately, we did experimentation based on the stacking method where we used different classifiers as the base level classifier and one classifier as a meta learner. Five different combinations of base level classifiers were made along with two different classifiers. Every models are tested with k-fold cross validation.

K-fold cross validation is a simple process by which models can be compared and selected for a specific machine learning problem. It is applied on our balanced dataset to resample the data points and to assess the machine learning models that we are implementing to detect fraudulent transactions. The parameter K indicates the number of groups that our training dataset will be divided into. In our case, the value of K is four. So, we can say that we applied 4-fold cross validation. Since our dataset is divided into four groups, one group can be considered as testing dataset and the other three will be training dataset. With that, the models are trained to get the performance score and same process is done again considering another group as a test dataset and the other three groups as training datasets.

1st experimentation:

Base level classifiers: K-Nearest Neighbor, Random Forest, Gaussian Naïve Bayes.

Meta learner: Logistic Regression.

So, after training the balanced dataset on the base level classifiers, the output works as the input for logistic regression and the final accuracy in detection the fraud transactions is 0.95.

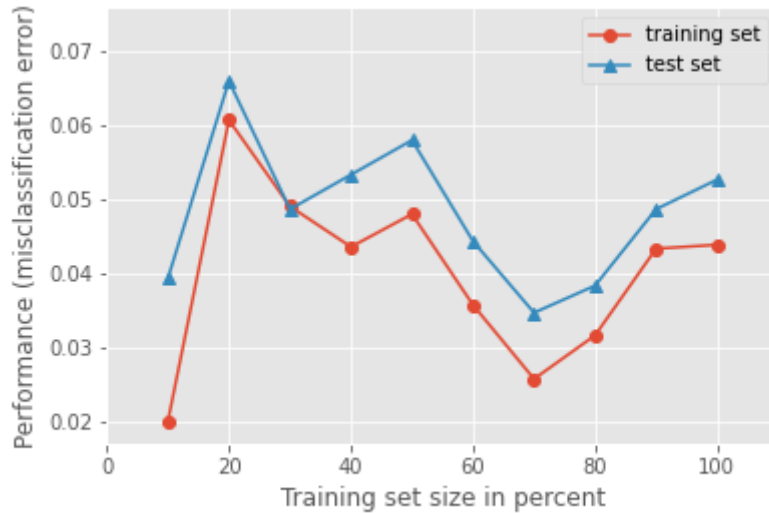


Figure 4.73: Plot learning curve for the 1st experiment

2nd experimentation:

Base level classifiers: K-Nearest Neighbor, Random Forest, Gaussian Naïve Bayes and Neural Network.

Meta learner: Logistic Regression.

So, after training the balanced dataset on the base level classifiers, the output works as the input for logistic regression and the final accuracy in detection the fraud transactions is also 0.95

Therefore, adding Neural network as the base level classifier did not have any effect on the accuracy.

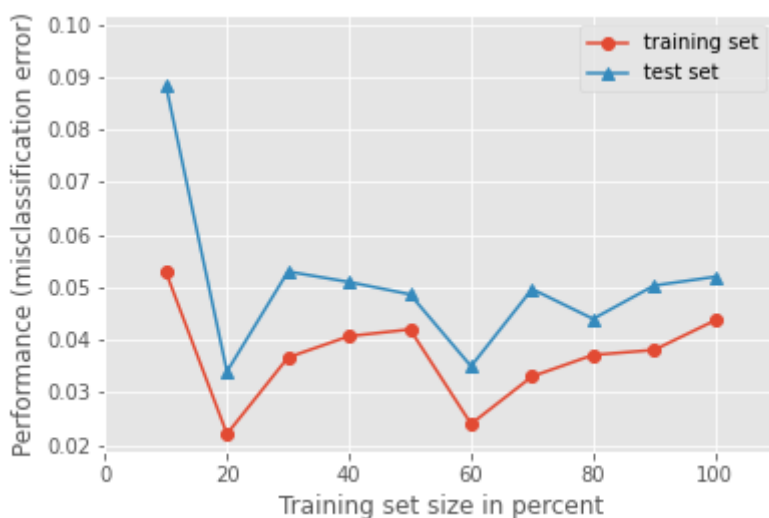


Figure 4.74: Plot learning curve for the 2nd experiment

3rd experimentation:

Base level classifiers: K-Nearest Neighbor, Random Forest, Gaussian Naïve Bayes and Neural Network.

Meta learner: Extreme Gradient Boosting

So, after training the balanced dataset on the base level classifiers, the output works as the input for logistic regression and the final accuracy in detection the fraud transactions is also 0.94.

Therefore, changing the meta learning into extreme gradient boosting did not have any effect on the accuracy. The accuracy did not increase.

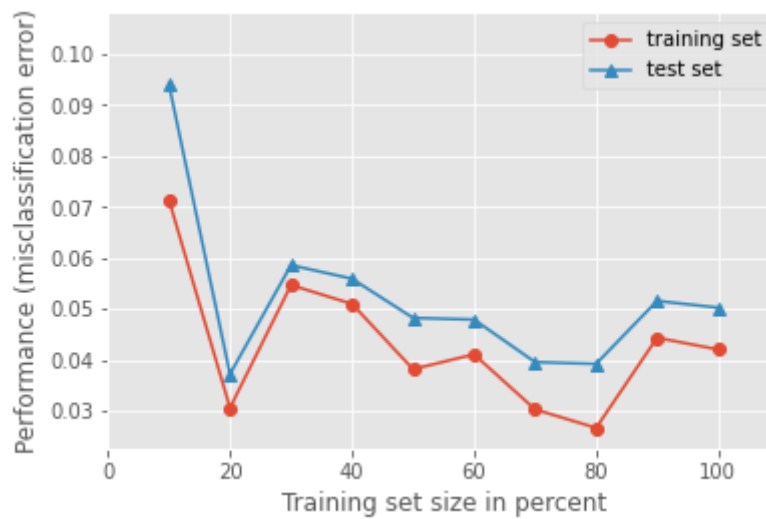


Figure 4.75: Plot learning curve for the 3rd experiment

4th experimentation:

Base level classifiers: K-Nearest Neighbor, Random Forest, Support Vector Classifier and Neural Network.

Meta learner: Logistic Regression

Here, the base level learning is changed from the previous experimentations. In addition to K-Nearest Neighbor and Random Forest, new two classifiers are added which are support vector machine and Neural network. Meta learner remained unchanged. The accuracy in detecting the fraud transactions is 0.95. Even in this case the accuracy rate has not increased.

Now, previously we discussed about grid search hyperparameter tuning. So, in order to increase the accuracy, hyperparameter tuning is done on the 4th experiment. Different sets of parameters were used and the final accuracy was optimized from 0.95 to 0.98.

```

0.967 +/- 0.00 {'kneighborsclassifier__n_neighbors': 1, 'meta_classifier__C': 0.1, 'randomforestclassifier__n_estimators': 10}
0.964 +/- 0.00 {'kneighborsclassifier__n_neighbors': 1, 'meta_classifier__C': 0.1, 'randomforestclassifier__n_estimators': 50}
0.964 +/- 0.00 {'kneighborsclassifier__n_neighbors': 1, 'meta_classifier__C': 10.0, 'randomforestclassifier__n_estimators': 10}
0.964 +/- 0.00 {'kneighborsclassifier__n_neighbors': 1, 'meta_classifier__C': 10.0, 'randomforestclassifier__n_estimators': 50}
0.973 +/- 0.00 {'kneighborsclassifier__n_neighbors': 5, 'meta_classifier__C': 0.1, 'randomforestclassifier__n_estimators': 10}
0.973 +/- 0.00 {'kneighborsclassifier__n_neighbors': 5, 'meta_classifier__C': 0.1, 'randomforestclassifier__n_estimators': 50}
0.981 +/- 0.00 {'kneighborsclassifier__n_neighbors': 5, 'meta_classifier__C': 10.0, 'randomforestclassifier__n_estimators': 10}
0.976 +/- 0.00 {'kneighborsclassifier__n_neighbors': 5, 'meta_classifier__C': 10.0, 'randomforestclassifier__n_estimators': 50}
Best parameters: {'kneighborsclassifier__n_neighbors': 5, 'meta_classifier__C': 10.0, 'randomforestclassifier__n_estimators': 10}
Accuracy: 0.98

```

Figure 4.76: Grid Search Hyperparameter tuning on the 4th experiment

From Fig 4.74, we see that if the parameters that was selected for KNN, random forest classifiers and the meta level classifier that is logistic regression has the value of 5, 10 and 10 respectively then the 4th experiment gives the best accuracy among all the combinations.

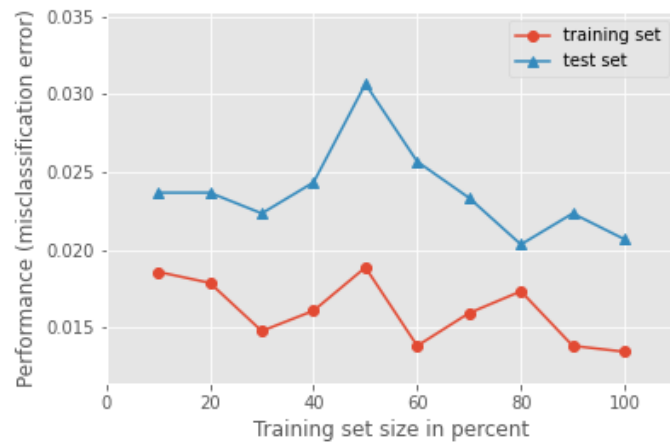


Figure 4.77: Plot learning curve for the 4th experiment

From the above four experiments, we see that every combination constructed using the stacking method gives better accuracy than the classifiers alone which were discussed previously. The classifiers had issues with overfit and underfit but using Stack technique, that issue has been minimized on a great scale. If every plot learning curve is seen that it is clear that the misclassification error has decreased a lot and the 4th experiment gives the lest overfit and the best performance.

Base learner models	Meta learner model	Accuracy
KNN, Random forest, Navie bayes gaussian	Logistic Regression	0.95
KNN, Random forest, Navie bayes gaussian, NN	Logistic Regression	0.95
KNN, Random forest, Navie bayes gaussian, NN	Xgboost	0.94
KNN, Random forest, SVC, NN	Logistic Regression	0.95
KNN, Random forest, SVC, NN (Hyper parameter tuning with grid search)	Logistic Regression	0.98

Table 4.4: Experimental Results of Stacking Models

Chapter 5

Conclusion and Future Work

In our overall research and workings, we tried to differentiate the fraudulent transactions from the safe ones with the help of certain machine learning algorithms. Logistic regression Classifier, Support vector Classifier, K-Nearest Neighbor, Decision Tree Classifier, Random forest Classifier, Extra Tree Classifier, Naïve Bayes, Adaptive Boosting and Gaussian Naïve Bayes were used to determine the accuracy of classifying the fraudulent transactions. We applied the above mentioned algorithms on both imbalanced and balanced datasets. For balancing the dataset, we applied SMOTE and Near Miss algorithm and discussed their result as well as the results that was harnessed from the imbalanced dataset. From the result analysis, we found out that Extra Tree classifier works best when it performs on the balanced dataset. We also applied Neural Network which gives overfitting result. Finally, we applied the Stacking technique where we used many classifiers as the base learners and one model as the meta learner. We did four different experiments taking different algorithms. K-fold cross validation was implemented for tackling the overfitting and underfitting problem. After the experimentation, we found the lowest accuracy was 0.94 that we got from the 3rd experiment and the rest of the experiments gave 0.95 accuracy which is the highest. To further optimize the accuracy, we used grid search hyperparameter tuning and increased the accuracy from 0.95 to 0.98 with very less overfitting. Therefore, it is clear that applying Stack method can increase the performance and also decrease overfit and underfit related problems. In the future, we would like to further do research on this field to come up with a process of identifying fraudulent activities while the transaction is taking place. We have a plan to shift our focus from stacking method to creating a Hybrid machine learning process. Even with the models made by stacking we are planning to coexists those models with a server and test whether it can clearly classify the fraudulent transactions. If we can successfully implement that then this model can be us in banking sectors as well as in the business sectors and can contribute to the society by minimizing the loss of public wealth.

Bibliography

- [1] *What is keras? the deep neural network api explained — infoworld*, <https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html>, (Accessed on 04/02/2020).
- [2] O. Fraud and V. CARE, *Fraud. Babel*, 2007.
- [3] T. Bukth and S. S. M. S. Huda, “The soft threat: The story of the bangladesh bank reserve heist”, 2017. DOI: 10.4135/9781526411228.
- [4] S. Rahman, *Three banks hit by cyberattacks*, Jun. 2019. [Online]. Available: <https://www.thedailystar.net/frontpage/news/three-banks-hit-cyberattacks-1760629>.
- [5] K. Hassibi *et al.*, “Detecting payment card fraud with neural networks”, *World Scientific Book Chapters*, pp. 141–157, 2000.
- [6] E. Abu-Shanab and S. Matalqa, “Security and fraud issues of e-banking”, *International Journal of Computer Networks and Applications*, vol. 2, no. 4, pp. 179–188, 2015.
- [7] *Leader of fraud gang which targeted mobile banking users, held in dhaka — the daily star*, <https://www.thedailystar.net/city/news/leader-fraud-gang-which-targeted-mobile-banking-users-held-dhaka-1880311>, (Accessed on 04/02/2020).
- [8] I. Sutedja *et al.*, “Detection of frauds for debit card transactions at automated teller machine in indonesia using neural network”, in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1196, 2019, p. 012076.
- [9] M. Anderka, T. Klerx, S. Priesterjahn, and H. K. Büning, “Automatic atm fraud detection as a sequence-based anomaly detection problem.”, in *ICPRAM*, 2014, pp. 759–764.
- [10] M. R. Lepoivre, C. O. Avanzini, G. Bignon, L. Legendre, and A. K. Piwele, “Credit card fraud detection with unsupervised algorithms”, *J. Adv. Inf. Technol*, vol. 7, no. 1, 2016.
- [11] E. AbuShanab, J. M. Pearson, and A. J. Setterstrom, “Internet banking and customers’ acceptance in jordan: The unified model’s perspective”, *Communications of the Association for information systems*, vol. 26, no. 1, p. 23, 2010.
- [12] L. Peotta, M. D. Holtz, B. M. David, F. G. Deus, and R. De Sousa, “A formal classification of internet banking attacks and vulnerabilities”, *International Journal of Computer Science & Information Technology*, vol. 3, no. 1, pp. 186–197, 2011.

- [13] D. Olszewski, “Fraud detection using self-organizing map visualizing the user profiles”, *Knowledge-Based Systems*, vol. 70, pp. 324–334, 2014.
- [14] P. Ravisankar, V. Ravi, G. R. Rao, and I. Bose, “Detection of financial statement fraud and feature selection using data mining techniques”, *Decision Support Systems*, vol. 50, no. 2, pp. 491–500, 2011.
- [15] B. Hoogs, T. Kiehl, C. Lacombe, and D. Senturk, “A genetic algorithm approach to detecting temporal patterns indicative of financial statement fraud”, *Intelligent Systems in Accounting, Finance & Management: International Journal*, vol. 15, no. 1-2, pp. 41–56, 2007.
- [16] E. Duman and M. H. Ozcelik, “Detecting credit card fraud by genetic algorithm and scatter search”, *Expert Systems with Applications*, vol. 38, no. 10, pp. 13 057–13 063, 2011.
- [17] S. Panigrahi, A. Kundu, S. Sural, and A. K. Majumdar, “Credit card fraud detection: A fusion approach using dempster–shafer theory and bayesian learning”, *Information Fusion*, vol. 10, no. 4, pp. 354–363, 2009.
- [18] A. O. Adewumi and A. A. Akinyelu, “A survey of machine-learning and nature-inspired based credit card fraud detection techniques”, *International Journal of System Assurance Engineering and Management*, vol. 8, no. 2, pp. 937–953, 2017.
- [19] A. Mubalik, “And e. adali,” multilayer perception neural network technique for fraud detection”, in *IEEE, Computer Science and Engineering (UBMK), International Conference*, 2017, pp. 383–387.
- [20] N. Malini and M. Pushpa, “Analysis on credit card fraud identification techniques based on knn and outlier detection”, in *2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, IEEE, 2017, pp. 255–258.
- [21] N. S. Halvaiee and M. K. Akbari, “A novel model for credit card fraud detection using artificial immune systems”, *Applied soft computing*, vol. 24, pp. 40–49, 2014.
- [22] C.-Y. J. Peng *et al.*, “An introduction to logistic regression analysis and reporting, 96 j”, *Educ. Res*, vol. 3, no. 10, 2002.
- [23] J. Yang, J. Gong, W. Tang, Y. Shen, C. Liu, and J. Gao, “Delineation of urban growth boundaries using a patch-based cellular automata model under multiple spatial and socio-economic scenarios”, *Sustainability*, vol. 11, no. 21, p. 6159, 2019.
- [24] T. D. Gedeon, “Data mining of inputs: Analysing magnitude and functional measures”, *International Journal of Neural Systems*, vol. 8, no. 02, pp. 209–218, 1997.
- [25] (1) *what is the best way to learn machine learning and artificial intelligence (python) for getting good job? - quora*, <https://www.quora.com/What-is-the-best-way-to-learn-machine-learning-and-artificial-intelligence-python-for-getting-good-job>, (Accessed on 04/02/2020).
- [26] *Di35605610.pdf*, https://www.ijera.com/papers/Vol3_issue5/DI35605610.pdf, (Accessed on 04/02/2020).

- [27] *A-typical-example-of-a-knn-classification-for-a-two-class-problem-ie-the-pink-and.png (551510)*, https://www.researchgate.net/profile/Mohammad_Reza_Haji_Samadi/publication/322358139/figure/fig2/AS:581222791127043@1515585720164/A-typical-example-of-a-KNN-classification-for-a-two-class-problem-ie-the-pink-and.png, (Accessed on 04/02/2020).
- [28] *K-nearest neighbors(knn) - data driven investor - medium*, <https://medium.com/datadriveninvestor/k-nearest-neighbors-knn-7b4bd0128da7>, (Accessed on 04/02/2020).
- [29] K. S. Parikh and T. P. Shah, “Support vector machine—a large margin classifier to diagnose skin illnesses”, *Procedia Technology*, vol. 23, pp. 369–375, 2016.
- [30] R. K. Sevakula and N. K. Verma, “Support vector machine for large databases as classifier”, in *International Conference on Swarm, Evolutionary, and Memetic Computing*, Springer, 2012, pp. 303–313.
- [31] *A-typical-example-of-a-knn-classification-for-a-two-class-problem-ie-the-pink-and.png (551510)*, https://www.researchgate.net/profile/Mohammad_Reza_Haji_Samadi/publication/322358139/figure/fig2/AS:581222791127043@1515585720164/A-typical-example-of-a-KNN-classification-for-a-two-class-problem-ie-the-pink-and.png, (Accessed on 04/02/2020).
- [32] *Classification with keras — pluralsight*, <https://www.pluralsight.com/guides/classification-keras>, (Accessed on 04/02/2020).
- [33] *Loss vs accuracy*, <https://kharshit.github.io/blog/2018/12/07/loss-vs-accuracy>, (Accessed on 04/02/2020).
- [34] *What is keras? the deep neural network api explained — infoworld*, <https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html>, (Accessed on 04/02/2020).
- [35] *Boosting algorithm: Adaboost - towards data science*, <https://towardsdatascience.com/boosting-algorithm-adaboost-b6737a9ee60c>, (Accessed on 04/02/2020).
- [36] *Adaboost classifier in python - datacamp*, <https://www.datacamp.com/community/tutorials/adaboost-classifier-python>, (Accessed on 04/02/2020).
- [37] *Adaboost algorithm — how adaboost algorithm works with example?*, <https://www.educba.com/adaboost-algorithm/>, (Accessed on 04/02/2020).
- [38] S. Džeroski and B. Ženko, “Is combining classifiers with stacking better than selecting the best one?”, *Machine learning*, vol. 54, no. 3, pp. 255–273, 2004.
- [39] N. Nahar, F. Ara, M. Nelay, A. Istiek, V. Barua, M. S. Hossain, and K. Andersson, “A comparative analysis of the ensemble method for liver disease prediction”, in *International Conference on Innovation in Engineering and Technology (ICIET)*, 2019.
- [40] K. Saiti, M. Macaš, K. Štechová, P. Pit’hová, and L. Lhotská, “A combined-predictor approach to glycaemia prediction for type 1 diabetes”, in *World Congress on Medical Physics and Biomedical Engineering 2018*, Springer, 2019, pp. 753–756.
- [41] E. Lopez-Rojas, A. Elmir, and S. Axelsson, “Paysim: A financial mobile money simulator for fraud detection”, in *28th European Modeling and Simulation Symposium, EMSS, Larnaca*, Dime University of Genoa, 2016, pp. 249–255.

- [42] H. Ali, M. N. M. Salleh, R. Saedudin, K. Hussain, and M. F. Mushtaq, “Imbalance class problems in data mining: A review”, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 14, no. 3, pp. 1560–1571, 2019.
- [43] S. Hu, Y. Liang, L. Ma, and Y. He, “Msmote: Improving classification performance when training data is imbalanced”, in *2009 second international workshop on computer science and engineering*, IEEE, vol. 2, 2009, pp. 13–17.
- [44] F. Hu and H. Li, “A novel boundary oversampling algorithm based on neighborhood rough set model: Nrsboundary-smote”, *Mathematical Problems in Engineering*, vol. 2013, 2013.
- [45] *How to use learning curves to diagnose machine learning model performance*, <https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/>, (Accessed on 04/02/2020).
- [46] *Proceedings of machine learning research*, <http://proceedings.mlr.press/v48/>, (Accessed on 04/02/2020).
- [47] G. I. Diaz, A. Fokoue-Nkoutche, G. Nannicini, and H. Samulowitz, “An effective algorithm for hyperparameter optimization of neural networks”, *IBM Journal of Research and Development*, vol. 61, no. 4/5, pp. 9–1, 2017.
- [48] M. Feurer and F. Hutter, “Hyperparameter optimization”, in *Automated Machine Learning*, Springer, 2019, pp. 3–33.
- [49] X. Zhang, X. Chen, L. Yao, C. Ge, and M. Dong, “Deep neural network hyperparameter optimization with orthogonal array tuning”, in *International Conference on Neural Information Processing*, Springer, 2019, pp. 287–295.
- [50] (1) (pdf) *the impact of automated parameter optimization on defect prediction models*, https://www.researchgate.net/publication/322586891_The_Impact_of_Automated_Parameter_Optimization_on_Defect_Prediction_Models/figures?lo=1&utm_source=google&utm_medium=organic, (Accessed on 04/02/2020).