Credit card fraud detection using machine learning

Carolyne Njoki Macharia(carolyne@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Prof. Phillip Mashele
North West University, South Africa

14 May 2020

Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

Abstract

Technological improvement has led people into using their credit cards in almost all the transactions they make daily. This rise of credit card use has, in turn, led to the rise of fraud activities by fraudsters thus leading financial institutions into increasingly large losses. In this essay, we will use machine learning algorithms to detect fraudulent transactions made. The dataset is highly imbalanced as we have 492 fraudulent transactions out of 284,807 observations and this is dealt with using several resampling approaches, Synthetic Minority Over-Sampling Technique (SMOTE), Tomek Links Removal, Edited Nearest Neighbor (ENN), Condensed Nearest Neighbor (CNN) and Nearmiss approach. We create the model using four classifiers, Random Forest, K-Nearest Neighbor (K-NN), Support Vector Machine (SVM) and Naive Bayes. We will evaluate the classifiers using the metrics, recall, precision and F1 score. Experimental results showed that Random Forest used with the resampling technique SMOTE and Tomek Links Removal gave good results as it had a precision of 86%, a recall of 87% and an F1 score of 86%. When used with ENN and CNN, Random Forest gave the same results of 84% for, precision, recall and F1 score. The code for this project is provided here

Keywords: Credit card fraud detection, Random Forest, K-NN, Naive Bayes, SVM, SMOTE, Nearmiss, ENN, Condensed Nearest Neighbor (CNN), Tomek Links Removal.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

[illegible]

Carolyne Njoki Macharia, 14 May 2020

Contents

1. Introduction

1.1 Background

E-commerce is one of the fastest-growing sectors worldwide. Due to this growth, Kurien and Chikkamannur (2019) shows that usage of credit cards is standing out as the easiest and convenient means of payment for the online transactions. As credit card usage increases over the years, so does fraudulent activities. The Figure 1.1 shows the rise in frauds in USA from the year 2014 to 2018.

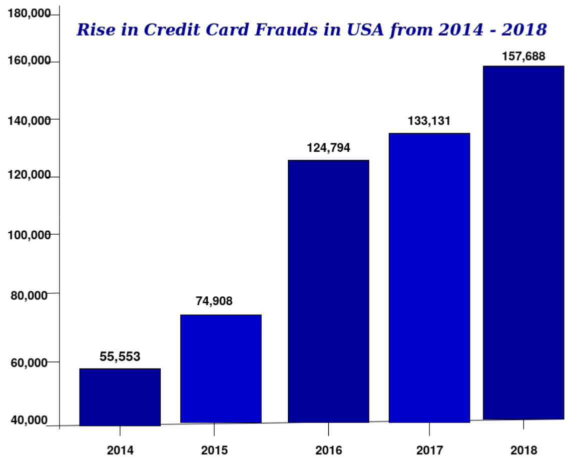


Figure 1.1: Credit Card fraud report for $2014 - 2018$

In an analysis made by Anderson et al. (2013), there are 24.26 billion in 2018 which were lost due to fraudulent transactions made using credit cards worldwide. According to Renjith (2018), frauds recorded in 2019 were approximately 70 trillion dollars. The frauds associated with credit card stands out to be the highest over the years. A comparison of credit card related fraud and other types of fraud is shown in Figure 1.2. We notice that from Figure 1.2 we have the credit card frauds standing out at 41% of the total frauds recorded.

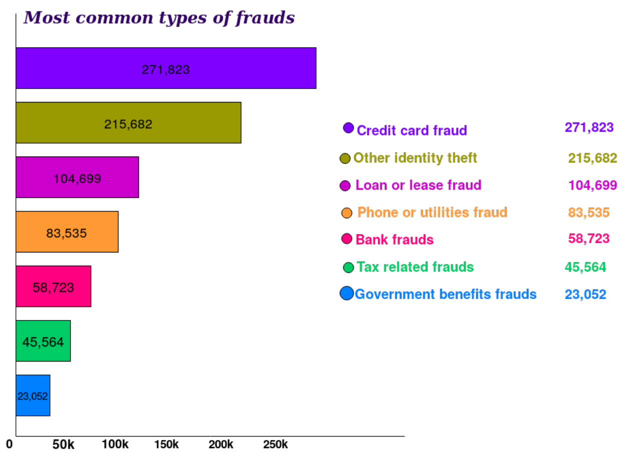


Figure 1.2: Reports on different types of frauds

The fraudsters use different methods to perform this crime which includes stolen cards, duplicating credit cards, and card not present. The most common technique used according to Anderson et al. (2013), is the card not present and holds the highest percentage of (81%). Fraud detection is carried out by many financial institutions to reduce the losses they incur from this crime. Maes et al. (1993) defines fraud detection as the process of identifying from a group of transactions those that are legitimate and those that are fraudulent.

Researchers identify these fraudulent transactions using the fraud detection models. Creating these models is not an easy task as there are many challenges associated with the process. Firstly, the researchers have problems getting the data as shown by Dal Pozzolo et al. (2014) the credit card transaction data is very confidential and institutions are not willing to give it to the public due to privacy policies. The other problem according to Holte et al. (1989) is class overlapping where we find that some of the transactions classified as fraudulent are legitimate transactions. The other main problem shown by Baader and Krcmar (2018) is the highly imbalanced dataset where the number of fraudulent cases is very few compared to the non-fraudulent cases.

This class imbalance can be very problematic especially when dealing with the traditional classifiers during their training as shown by Japkowicz and Stephen (2002). Thus, for any researcher, this becomes the first problem they have to tackle otherwise the results obtained will be inaccurate as they will favor the class with the many observations. Many researchers have carried out experiments to try and identify models that can detect these fraudulent activities.

1.2 Existing Work

Perantalu and BhargavKiran (2017) performed predictive modeling of credit card fraud detection. The authors used two machine learning classifiers, Logistic Regression, and Decision Trees. For the attribute selection they used information gain and used accuracy as the evaluation metric. Perantalu and BhargavKiran (2017) found out that the Decision Trees performed better than the Logistic Regression in detecting any fraudulent transaction.

Pun and Lawryshyn (2012) performed a study of whether a meta classifier performs better than the Falcon based fraud prediction performance. The data used was a neural network filtered dataset from a Canadian bank. The multi-based classifier algorithm used 3 classifiers, Naive Bayesian, K-Nearest Neighbor(K-NN), and Decision Trees. Naive Bayesian algorithm was used as the final classifier to combine the other classifiers. Pun and Lawryshyn (2012) concluded that the meta-classifier improved in saving money and in fraud detection.

In a study performed by Shakya (2018) using Random Forest, Logistic Regression and XGBoost showed that Random Forest performed better than the other two classifiers. He said the reason was because of the ensemble nature of Random Forest. The class imbalance problem was solved using Synthetic Minority Over-Sampling Technique (SMOTE). The combination of Random Forest and SMOTE gave the best results in detecting fraudulent transactions.

Wiese and Omlin (2009) performed a study on fraud detection using three different machine learning techniques, Conventional Feedforward Neural Networks, Support Vector Machine(SVM), and Long Short Term Memory Recurrent Neural Networks(LSTM). In his studies he took account of a sequence of transactions instead of individual transactions and so making this a time series problem. He concluded that LSTM outperformed the other two techniques reason been that it can deal with the noises in the data unlike the other techniques.

1.3 Aims and Objective

In this study, we aim to detect any fraudulent transactions made from all the transactions made using a credit card in the given dataset. This aim will be met by accomplishing the following objectives:

- Performing a predictive analysis using traditional machine learning classifiers such as Random Forest, SVM, Naive Bayes and K-NN.
- Resample, train and test the data.
- Evaluate the efficiency of the classifiers using different evaluation metrics.

1.4 Contribution

The main problem that researchers face when researching in this area is a class imbalance. The contribution in this research is to try different resampling techniques that have not been used in credit card fraud detection research but have worked well in other researches that involve a highly imbalanced dataset. The resampling techniques are NearMiss that have worked well with SVM and the hybrid techniques SMOTE and Tomek Links , ENN and CNN that works very well with Random Forest classifier.

1.5 Structure of the Essay

The essay will be structured in the following way: Chapter 2 will have the literature review on past work, the machine learning techniques and resampling techniques. Chapter 3 will give the methodology used and the different evaluation metrics. Chapter 4 we will have the results from the four classifiers. Chapter 5 will present the conclusion, challenges faced in the essay and future work.

2. Literature Review

2.1 Introduction to Machine Learning

This section will contain the explanation of how machine learning works and the different types of traditional classifiers that can be used in detecting whether a transaction is fraudulent or legitimate. We will also have an explanation of different ways to deal with the biggest challenge of class imbalance in the dataset used.

2.1.1 Machine Learning.

Shabbir and Anwer (2018) defines machine learning as part of artificial intelligence which involves the computer implementing different algorithms to solve certain problems without a human being performing any programming. Machine learning is categorized into 3 categories, supervised learning, unsupervised learning and reinforcement learning.

2.1.2 Supervised Learning.

According to Mohri et al. (2012), the goal of supervised learning is to have a model that can predict a label when it is given a vector of features. In supervised learning as shown by Kotsiantis (2007), it contains labeled features that are used to predict a target value. The features can be categorical or continuous. Under supervised learning we can have two types of problems depending on the nature of the label we are predicting.

- **Regression**-This is when the label is continuous for example if we want to predict the stock prices in a given period.
- **Classification** - When the problem is a classification problem then we want to predict a categorical target. For example spam detection, credit default of a loan, fraud detection among many more.

In this essay we are going to focus more on supervised learning and specifically the classification problem.

2.1.3 Unsupervised Learning.

In the unsupervised learning, unlike the supervised learning here we only have the input variables but no output variable, Usama et al. (2019) refers to the data as unlabeled. In this learning the algorithm learns by itself the pattern and the structure of a given data and then it comes up with a conclusion. There are two types of unsupervised learning:

- **Clustering**- According to Jain et al. (1999), the algorithm learns the patterns of a given data and groups it according to this pattern. For example, a certain group of customers likes buying these types of products.
- **Association**- This is also an important type of unsupervised learning where it checks the relation between one group with another. For example, if this customer buys wine and chocolate then they are also likely to buy some kinds of pasta.

2.1.4 Reinforcement Learning.

This is a less commonly used machine learning technique. In this technique, François-Lavet et al. (2018) shows that we have both the input and output variables and the algorithm tries to learn from its mistakes by maximizing the rewards of a certain situation and minimize the mistakes.

2.2 Techniques to Solve Class Imbalance

According to Ling and Sheng (2010) they define class imbalance as, given a classification problem, the number of observations under one class is far less than those in the other class. In this essay, this is the main problem as the number of fraudulent transactions are very few, 0.0173% of the whole dataset. We have different approaches used to solve the class imbalance problem.

2.2.1 Ensemble Approaches.

According to Rokach (2005), these approaches combines different weak classifiers and makes a better classifier that can deal with this class imbalance problem. This approach uses two different methods:

- **Bagging**- In a paper by Breiman (1994) he explains that this method uses weak classifiers separately and then combines their results by either taking the average or by taking a vote. The most common classifier that works this way is the Random Forest.
- **Boosting**- On the hand, Freund and Schapire (1999) shows that this is different from the bagging since we work with the weak classifiers together in a way that the performance of one classifier depends on the other classifier.

2.2.2 Resampling Approaches.

This is another approach that is used when trying to deal with the class imbalance. We will now discuss the different approaches under the resampling approaches as this is the approach that this essay focusses on.

- **Under-sampling** According to He and Garcia (2009), this is when we take the majority class and reduce it to have the same number of observations as the minority class as shown in Figure 2.1. The disadvantage of this method is that when we reduce the majority class we are likely to lose some information and thus not making a good analysis. We can have many undersampling approaches that we will discuss below.

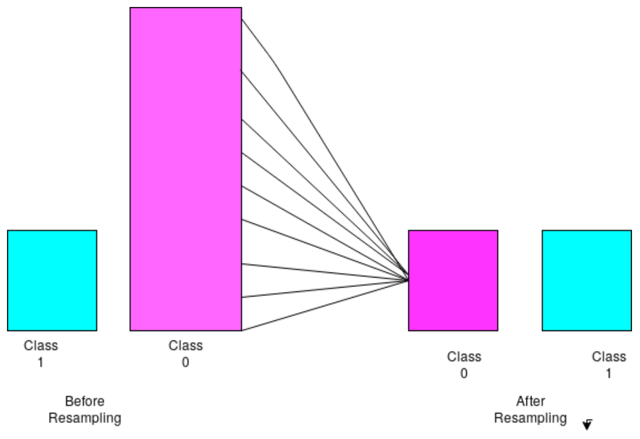


Figure 2.1: Undersampling approach

1. **NearMiss** This is an undersampling method where we take the distance between each majority class observations and the minority class observations and depending on the approach of nearmiss one uses we delete the majority class observations not needed appropriately. There are 3 types of the NearMiss that we can use.
    - **NearMiss-1** Yen and Lee (2006) explains this as where we take the observations from the majority class that have a minimum average distance with the 3 nearest observations from the minority class.
    - **NearMiss-2** this takes observations from the majority class that have the minimum average distance with the 3 furthest observations from the minority class and removes the other observations.

- **NearMiss-3** Zhang and Mani (2003) shows that in this approach we take the majority class observations that have a minimum distance with each of the minority class instance.

We use the Euclidean distance metrics to calculate this distance between the majority class instances and the minority class instances. In this essay we will use Nearmiss-3 since it works well with the highly imbalanced data as shown in an analysis by Zheng (2004). Also since the dataset is highly dimensional, NearMiss-3 will be convenient compared to the other two Figure 2.2 illustrates how NearMiss-3 works.

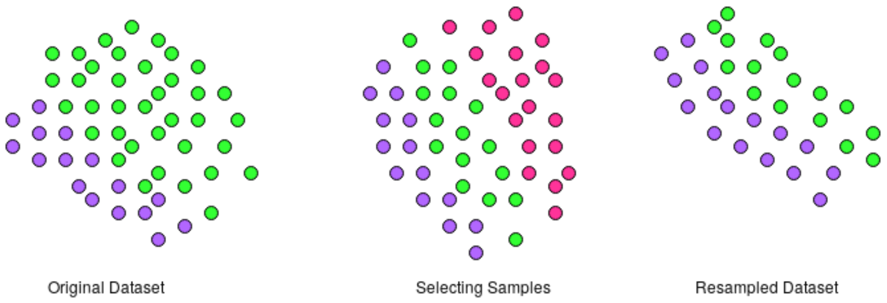


Figure 2.2: Nearmiss-3 Approach

2. **Edited Nearest Neighbor (ENN)** -This is also an undersampling method that we take an instance and check its 3 nearest neighbor as shown by Alejo et al. (2010). If the instance is from the majority class and the other 3 neighboring instances are from the minority class then we remove this one instance from the majority class. On the other hand, if the instance is from the minority class and the 3 neighbors are from the majority class, then we remove the three instances from the majority class as explained in a paper by Batista et al. (2004).

3. **Tomek Links Removal** This is another undersampling method where in a given dataset according to Tomek (1976), we look for the observations from the majority and minority class that is close to each other and we remove those instances from the majority class. The following Figure 2.3 shows how Tomek Links Removal works.

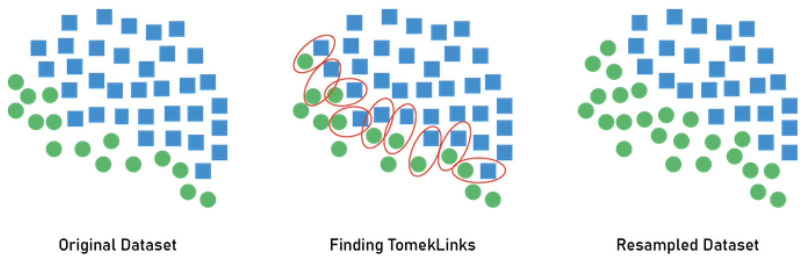


Figure 2.3: Tomek Links Removal

4. **Condensed Nearest Neighbor(CNN)** According to Hart (1968), this is referred to as the Hart algorithm. It is mostly used with the K-Nearest Neighbor(K-NN) classifier and works as follows:

   - Given a training set X we take all those instances x close to each other but they are from the different labels and put them in the set S.

   - Repeat the selection until no more instances can be added in S.

- Use the set S for classification instead of the training set X.

CNN helps in saving space and time used for training.

- **Over-sampling** This is the opposite of the undersampling where we take the minority instances and duplicate them until they are the same numbers as the majority class as explained by Rahman and Davis (2013). We have different oversampling techniques but we will focus on one of them in this essay, SMOTE. The following Figure 2.4 shows what is oversampling.

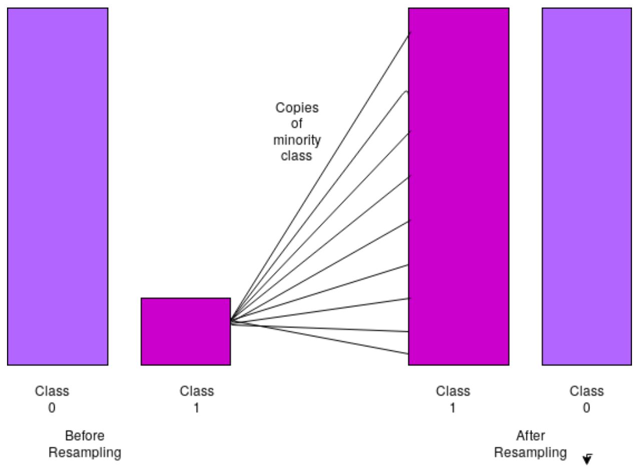


Figure 2.4: Oversampling

1. **Synthetic Minority Oversampling Techniques(SMOTE)** According to Bunkhumpornpat et al. (2009), this is an oversampling technique where new instances are synthetically created from the existing minority class observations. It works as follows:

   - Take the difference between the feature vector of an instance and the nearest neighbor say 1 nearest neighbor's feature vector.
   - Then multiply this difference by a value between 0 and 1 and not by the two numbers.
   - Then add the result to the instance's value and this becomes the new instance.

   The Figure 2.5 shows how this technique works where the red dots are the synthetic instances created.

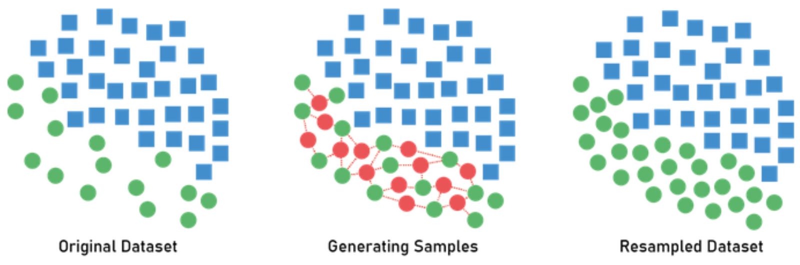


Figure 2.5: SMOTE

In this essay we are going to use the above discussed resampling techniques together with different classifiers that we will discuss in the next section.

2.3 Machine Learning Classifiers

We have many traditional machine learning classifiers that can be used to solve a classification problem but we will focus on the four main ones.
2.3.1 Random Forest.
Given a dataset $D = \{(\mathrm{x}_1, \mathrm{y}_1), (\mathrm{x}_2, \mathrm{y}_2), \ldots, (\mathrm{x}_n, \mathrm{y}_n)\}$ where $i \in \mathbb{R}^+$ let x_i be the i^{th} observation and y_i the output given by the i^{th} element in our dataset D. Then the Random Forest which is an ensemble classifier of a number of decision trees is defined by Biau (2010) as:

$$H_{rf}(x) = \{h_1(x), h_2(x), \ldots, h_k(x)\} \tag{2.3.1}$$

$$\text{where} \quad 1, 2, \ldots, k \quad \text{are the number of trees}$$

To have an understanding of the Random Forest we look into the decision trees. $H_1(x)$ is the function representing the first decision tree $(i = 1)$ which is made by making different decisions just as the name suggests.

The decision tree is made by making splits where the top node in the tree is called the root node and when we split it we make decision nodes and the lowest node is called the terminal/leaf node as defined in an anlysis by Breiman (2004). We represent the features in the decision trees as the set

$$\Omega = \{\theta_1, \theta_2, \ldots, \theta_m\} \tag{2.3.2}$$

When we do the split we have to consider which feature is more important and this can be done using 2 methods suggested by Hjerpe (2016):

- Information entropy
- Gini Index

The information entropy of a feature(attribute) is defined by Shannon (1948) as the information that will be needed to make any classification, say, is a transaction made fraudulent or legitimate. It is mostly used when the attributes are categorical. It is calculated as follows:

$$\text{entropy} = -\sum_{i=1}^{n} p_i \text{log} p_i$$

The higher the entropy the better to use that attribute as the best split. On the other hand, the Gini index is defined by Han et al. (2012) as a representation of the cost incurred in making the wrong classification so we take the attribute with the lower index as suggested by Liu and Haig (2018). It is mostly used when the attributes are continuous and when the data is highly imbalanced. It is calculated as:

$$\text{Gini} = 1 - \sum_{i=1}^{n} (p_i)^2 = \sum_{i=1}^{n} p_i(1 - p_i)$$

$$\text{where} \quad p_i = \text{probability of event i}$$

In this case, since our attributes are continuous and we are working on a highly imbalanced dataset, we will use the Gini index. So at each split in the decision tree the attribute with the lowest Gini index is selected.

Now with the understanding of how the decision trees work we now focus on the Random Forest which is just a collection of many decision trees. And this is how the Random Forest works.

- From our set D we create with replacement resamples $\{c_1, c_2, \ldots, c_m\}$ this is done by randomly choosing a given number of features mostly if we have n features we choose $k < n$ features where, $k = \sqrt{n}$.
- For each bootstrap resample, grow a tree using the Gini index for the best split.
- When we have a grown tree we assign either 0 or 1 at each terminal node to represent a legitimate or fraudulent transaction.
- Having a group of m trees $T_M, M = 1, 2, \ldots, m$ each with a predicted value, we perform the following, for regression problem Biau and Scornet (2016) suggests that we average all the predicted value from the trees. For classification, we do the voting and choose the highest voted class.
- For regression, let $h_i(x), i = 1, 2, \ldots, m$ be the result of the i^{th} tree then the final result is given as:
$$H_{rf}(x) = \frac{1}{M}\sum_{i=1}^{m} h_i(x)$$
- For classification, let $c_i(x), i = 1, 2, \ldots, m$ be the class prediction of the i^{th} tree, then
$$C^i_{rf}(x) := \text{argmax}\{c_i(x)\}$$
- If we have, say, 100 trees we have 100 results for a given observation, say x_{20}, so we perform a vote on the one with the highest votes if for example, in the case where 80 came out as a fraudulent transaction and 20 as a legitimate transaction then we vote that the observation is a fraudulent transaction.
- The more the trees the better but the slower the process as it has to do the voting from all the trees.

2.3.2 Support Vector Machine(SVM).

SVM is used in both classification and regression problems. When dealing with a regression problem we use the support vector regression (SVR) and when dealing with a classification problem we use the support vector classifier (SVC). In most cases, according to Dheepa and Dhanapal (2012), it is used in classification problems.

When we have an n-dimensional space, the hyperplane is the line that separates the data points of one class from the other if we have 2 classes and the hyperplane can be represented by the equation:

$$wx + b = 0 \tag{2.3.3}$$

where w is the weight vector and is normal to the hyperplane $\quad w = \{w_1, w_2, \ldots, w_n\}$

In a simple way, SVM creates a hyperplane such that given two classes, all the points from one class satisfies the equation $wx + b > 0$ and the points from the other class satisfies the equation $wx + b < 0$. More specifically, suppose we have a dataset $D = \{(\mathrm{x}_1, \mathrm{y}_1), (\mathrm{x}_2, \mathrm{y}_2), \ldots, (\mathrm{x}_n, \mathrm{y}_n)\}$ where $x_i \in \mathbb{R}^n$ represent the i^{th} observation which is a feature vector and $y_i \in \{-1, 1\}$ is the label associated to that instance.

The main focus of SVM according to Maimon and Rokach (2005), is to maximize the distance between the hyperplane and the closest points to the hyperplane. These closest points are called the support vectors. Maimon and Rokach (2005) gives the reason that a hyperplane whose margin is wide is likely to be resistant to any noise such as an observation been misclassified.

If we make the assumption that the observations from the two classes are separable, then we end up with a hyperplane that satisfiies the equation:

$$y_i(wx_i + b) \geq 1, \quad i \in \{1, \dots, n\} \tag{2.3.4}$$

Since we will have many hyperplanes that can separate the two classes, we need to find the one that maximizes the margin. The margin is given as $\frac{1}{\|w\|}$ and so maximizing the margin is the same as minimizing the $\| w \|$. Kroon and Omlin (2004) converts this from a maximization problem to a constrained optimization problem by solving the equation:

$$\text{minimize} \quad \frac{1}{2} \| w \|^2, \quad \text{subject to} \quad y_i(wx_i + b) \geq 1 \quad i \in \{1, \dots, n\}, \tag{2.3.5}$$

To deal with the constraint 2.3.4, Wiese and Omlin (2009) introduces the lagrangian multipliers α_i, $i \in \{1, \dots, n\}$ and we get the primal form equation by multiplying the constraints by the lagrangian multipliers and subtracting it from the objective function:

$$\mathcal{L}_p(w, b, \alpha) = \frac{1}{2} \| w \|^2 - \sum_{i=1}^{n} \Big(\alpha_i (y_i(wx_i + b) - 1) \Big) \tag{2.3.6}$$

When we are dealing with an optimization problem, Kroon and Omlin (2004) shows that the solution lies on the saddle points of 2.3.6 which we have to minimize with respect to (w, b) and maximize with respect to α_i. We maximize by taking its partial derivative with respect to the variables w and b.

$$\frac{\partial \mathcal{L}(w, b, \alpha)}{\partial w} = w - \sum_{i=1}^{n} a_i x_i y_i = 0 \implies w = \sum_{i=1}^{n} a_i x_i y_i \tag{2.3.7}$$

$$\frac{\partial \mathcal{L}(w, b, \alpha)}{\partial b} = - \sum_{i=1}^{n} a_i y_i = 0 \tag{2.3.8}$$

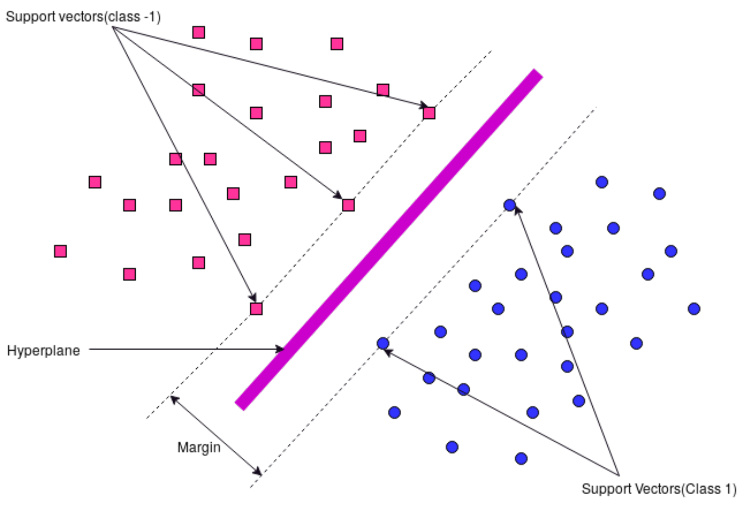


Figure 2.6: Support Vector Machine classifier

The Figure 2.6 above shows the hyperplane, the support vectors and the margin.

Using Karush-Kuhn-Tucker(KKT) conditions, we check any relationship between the lagrangian multipliers and the constraints and in this case we have:

$$\alpha_i(y_i(wx_i + b) - 1) = 0, \quad i \in \{1, \ldots, n\} \tag{2.3.9}$$

Using 2.3.9 we can either have $\alpha_i = 0$ or $y_i(wx_i + b) - 1 = 0$ meaning that for the support vectors $\alpha_i > 0$ and for the other instances that are not the support vectors, $\alpha_i = 0$.

Substituting 2.3.7 and 2.3.8 in the primal problem 2.3.6 we convert it into a dual maximization problem:

$$\mathcal{L}_D(\alpha) = \frac{1}{2}\Bigg[\Bigg(\sum_{i=1}^{n} \alpha_i x_i y_i\Bigg).\Bigg(\sum_{j=1}^{n} \alpha_j x_j y_j\Bigg)\Bigg] - \Bigg[\Bigg(\sum_{i=1}^{n} \alpha_i x_i y_i\Bigg).\Bigg(\sum_{j=1}^{n} \alpha_j x_j y_j\Bigg)\Bigg] + \sum_{i=1}^{n} \alpha_i y_i b + \sum_{i=1}^{n} \alpha_i$$

$$\mathcal{L}_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j x_i x_j y_i y_j, \quad \text{subject to} \quad \alpha_i \geq 0 \quad \text{and} \quad 2.3.8 \tag{2.3.10}$$

To solve the dual maximization problem 2.3.10 we let α^* to be the maximizing solution and w^* and b^* be the parameters of the hyperplane. In case we have a new instance x that need to be classified then the decision function used is:

$$\text{sign}(w^* x + b^*) = \text{sign}(\sum_{i=1}^{n} \alpha_i^* x_i y_i.x + b^*) \tag{2.3.11}$$

In most cases, the data points are not linearly separable so we use the kernel trick where we transform the low dimension input space to a higher dimensional feature space which can make the data points to be linearly separable.

When using the kernel trick we map the low dimension input space G into a higher dimension feature space F, i.e, $\psi : G \longmapsto F$ where we transform the data points into $\psi(x_1), \ldots, \psi(x_n)$ which are now linearly separable. According to the kernel trick, we have $\psi(x)\psi(y) = K(x, y)$ where K is the kernel function. When we replace the inner products in 2.3.10 with the kernel function we obtain 2.3.12 which we maximize to get the solution to our problem

$$\mathcal{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i x_j) \tag{2.3.12}$$

When we maximize 2.3.12 the decision function 2.3.11 becomes:

$$\begin{aligned} &\text{sign}(w^* \psi(x) + b^*) \\ &= \text{sign}\Bigg(\Bigg(\sum_{i=1}^{n} \alpha_i^* y_i \psi(x_i)\Bigg).\psi(x) + b^*\Bigg) \\ &= \text{sign}\Bigg(\sum_{i=1}^{n} \alpha_i^* y_i \psi(x_i).\psi(x) + b^*\Bigg) \\ &= \text{sign}\Bigg(\sum_{i=1}^{n} \alpha_i^* y_i K(x_i, x) + b^*\Bigg) \end{aligned} \tag{2.3.13}$$

Sometimes even with the kernel trick it is still not possible to separate the data points into two groups and so we use the soft margin. To deal with this non-separability, Jakkula (2006) introduces slack variables $\xi = (\xi_1, \dots, \xi_n)$ this helps in case our data has noises such as outliers and thus preventing an overfitting of the model. We end up with the constraints as:

$$y_i(wx_i + b) \geq 1 - \xi_i, \quad i \in \{1, \dots, n\}, \quad \xi \geq 0 \tag{2.3.14}$$

Including the slack variables the objective function becomes:

$$\text{min}\frac{1}{2} \parallel w \parallel^2 + C\sum_{i=1}^{n} \xi_i \tag{2.3.15}$$

from 2.3.15 Maimon and Rokach (2005) defines C$\in \mathbb{R}$ as a hyperparameter that represents the cost incurred when the constraint 2.3.4 is violated. We introduce the lagrange multipliers $\alpha > 0$ to deal with the constraints and we use another lagrange multpliers $\beta = \beta_1, \dots, \beta_n$ to take care of the introduced slack variable constraint in 2.3.15. This gives us a primal optimization problem 2.3.16:

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2} \parallel w \parallel^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \Big(\alpha_i(y_i(wx_i + b) - (1 - \xi_i)) \Big) - \sum_{i=1}^{n} \beta_i \xi_i \tag{2.3.16}$$

As we did with the primal problem 2.3.6 to minimize Jakkula (2006) takes the partial derivative of 2.3.16 with respect to w,b and ξ_i

$$\frac{\partial \mathcal{L}(w, b, \xi_i)}{\partial w} = w - \sum_{i=1}^{n} a_i x_i y_i = 0 \implies w = \sum_{i=1}^{n} a_i x_i y_i \tag{2.3.17}$$

$$\frac{\partial \mathcal{L}(w, b, \xi_i)}{\partial b} = \sum_{i=1}^{n} a_i y_i = 0 \tag{2.3.18}$$

$$\frac{\partial \mathcal{L}(w, b, \xi_i)}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \implies C = \beta_i + \alpha_i \tag{2.3.19}$$

Replacing 2.3.17 and 2.3.19 in 2.3.16 we have:

$$\begin{aligned}\mathcal{L}(w, b, \xi_i) = \frac{1}{2}\Big((\sum_{i=1}^{n} a_i x_i y_i)(\sum_{i=1}^{n} a_i x_i y_i) \Big) + C\sum_{i=1}^{n} \xi_i - \sum_{i,j} \alpha_i x_i y_i \alpha_j x_j y_j \\ - b\sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i (1 - \xi_i) - \sum_{i=1}^{n} (C - \alpha_i)\xi_i\end{aligned} \tag{2.3.20}$$

solving 2.3.20 and replacing 2.3.18 we have:

$$\mathcal{L}(w, b, \xi_i) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j x_i x_j y_i y_j \tag{2.3.21}$$

Equation 2.3.21 is similar to 2.3.12 even with the modification of the objective function the dual problem is still the same and the only changes is on the constraints. In the separable case, we have the two constraints, $\alpha \geq 0$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$ and in the non-separable case we have, $\alpha \geq C$ from 2.3.19 and $\beta \geq 0$. After maximizing 2.3.21 and obtaining α^* we can get the other parameters w^*, b^* in both the cases,separable case and the non-separable case.

2.3.3 K-Nearest Neighbor (K-NN).

Fix and Hodges (1952) defines K-NN as a non-parametric, simplest and commonly used classifier. Non-parametric means that we do not need to know the probability distribution of the data we are working on. The classifier assumes that for a given data point, its nearest neighbors are similar to it. According to Dasarathy (1991), the classifiers takes the distance between the test data point and each training data point.

Given a dataset $D = \{(\mathrm{x}_1, \mathrm{y}_1), (\mathrm{x}_2, \mathrm{y}_2), \dots, (\mathrm{x}_n, \mathrm{y}_n)\}$ and we have a new data point X that we want to classify to which class it belongs, we denote the set of the k nearest neighbors of X as S_x where

$$S_x \subseteq D \quad \text{s.t}|S_x| = k \quad \text{and} \quad \forall (x_i, y_i) \in D \backslash S_x :$$

$$\text{dist}(x, x_i) \geq \max \text{dist}(x, x') \tag{2.3.22}$$

From 2.3.22 it means that the distance from the test data point and other training data points apart from those in the set S_x is greater than that with those in the set S_x.

We need to know how to calculate this distance between these points by identifying the right distance metric to use.

There are different metrics that can be used:

- **Minkowski Distance** According to Boriah et al. (2008), this metric is calculated as follows:

$$\text{distance} = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

  And from this we can have other distance metrics depending on the value of p

$$p = 1, \text{Manhattan distance}$$

$$p = 2, \text{Euclidean distance}$$

$$p = \infty, \text{Chebychev distance}$$

- **Manhattan Distance** Wilson and Martinez (2000) defines this as the distance between two points when the value of $p = 1$ and is given as:

$$\text{d} = \sum_{i=1}^{n} |x_i - y_i|, i = 1, 2, \dots, n$$

$$\text{d} = (x_1 - y_1) + (x_2 - y_2) + \dots + (x_n - y_n)$$

- **Euclidean Distance** According to Wilson and Martinez (2000), to measure the distance between 2 points x_1 and x_2 we let them to be represented by the feature vectors:

$$x_1 = (x_{11}, x_{12}, \dots, x_{1m})$$

$$x_2 = (x_{21}, x_{22}, \dots, x_{2m})$$

  where m is the dimension of the feature vector

  To calculate the distance between the two points using the normalized Euclidean metric we use the following:

$$\text{d}(x_1, x_2) = \sqrt{\frac{\sum_{i=1}^{m} (x_{1i} - x_{2i})^2}{m}}$$

The value of k is not a constant value and it depends on the dataset one is working on but in most cases an odd number is considered and according to Chomboon et al. (2015), $k = 5$ or $k = 7$ are the common values used. Some researchers use $k = odd\sqrt{m}$ where m is the number of features. The value k is taken to be an odd value to make it possible for the voting.

Once the distance between the points is measured, according to Chomboon et al. (2015), the most common label in S_x is voted for as the label of the new instance as seen in 2.3.23

$$h(x) = \text{mode}(\{y' : (x', y') \in S_x\}) \tag{2.3.23}$$

The Figure 2.7 below shows that after voting, the new green instance will be a blue triangle since we have two blue triangles versus one purple diamond.

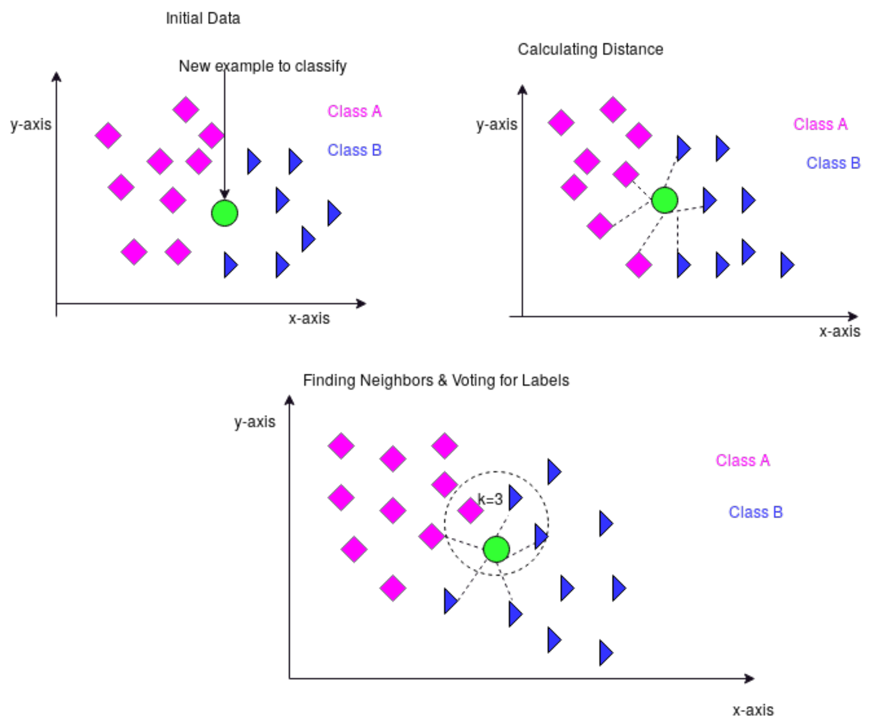


Figure 2.7: How KNN classifies the new instance

2.3.4 Naive Bayes.

The Naive Bayes classifier is based on the Bayes theorem as explained by Berrar (2018). According to Taheri and Mammadov (2015), the classifier makes a naive assumption that the features are independent. Given two random variables A and B, Bayes theorem is given as:

$$\text{p}(A|B) = \frac{\text{p}(B|A)\text{p}(A)}{\text{p}(B)} \tag{2.3.24}$$

where

p(A|B) - is the probability of event A occurring given that event B has already occurred. It is called the posterior probability.

p(A) and p(B) - are the prior probabilities of events A and B occurring.

p(B|A) - is the probability of event B occurring given that event A has already occurred. It is called the conditional probability.

In Naive Bayes we want to classify a new observation X which is a feature vector $X = \{x_1, x_2, \ldots, x_n\}$, into one class $c_i \in C$. We use 2.3.24 to calculate the conditional probability $\mathsf{p}(c_i|x_1, x_2, \ldots, x_n)$ as follows:

$$\mathsf{p}(c_i|x_1, x_2, \ldots, x_n) = \frac{\mathsf{p}(x_1, x_2, \ldots, x_n|c_i)\mathsf{p}(c_i)}{\mathsf{p}(x_1, x_2, \ldots, x_n)} \tag{2.3.25}$$

We can rewrite the numerator of 2.3.25 using the relation $\mathsf{p}(A|B)\mathsf{p}(B) = \mathsf{p}(A, B)$ as:

$$\mathsf{p}(x_1, x_2, \ldots, x_n|c_i)\mathsf{p}(c_i) = \mathsf{p}(x_1, x_2, \ldots, x_n, c_i) \tag{2.3.26}$$

Taheri and Mammadov (2013) simplifies 2.3.26 using the chain rule of probability

$$\begin{aligned}\mathsf{p}(x_1, x_2, \ldots, x_n, c_i) &= \mathsf{p}(x_1|x_2, \ldots, x_n, c_i)\mathsf{p}(x_2, \ldots, x_n, c_i)\\ &= \mathsf{p}(x_1|x_2, \ldots, x_n, c_i)\mathsf{p}(x_2|x_3, \ldots, x_n, c_i)\mathsf{p}(x_3, \ldots, x_n, c_i)\\ &\vdots\\ &= \mathsf{p}(x_1|x_2, \ldots, x_n, c_i)\mathsf{p}(x_2|x_3, \ldots, x_n, c_i)\ldots\mathsf{p}(x_{n-1}|x_n, c_i)\mathsf{p}(x_n|c_i)p(c_i)\end{aligned}$$

$$\mathsf{p}(x_1, x_2, \ldots, x_n, c_i) = \mathsf{p}(x_1|x_2, \ldots, x_n, c_i)\mathsf{p}(x_2|x_3, \ldots, x_n, c_i)\ldots\mathsf{p}(x_{n-1}|x_n, c_i)\mathsf{p}(x_n|c_i)p(c_i) \tag{2.3.27}$$

But since Naive Bayes assumes that the features are independent of each other then it means that what happens on one feature does not affect the other feature and so 2.3.27 becomes:

$$\mathsf{p}(x_1, x_2, \ldots, x_n, c_i) = \mathsf{p}(x_1|c_i)\mathsf{p}(x_2|c_i)\ldots\mathsf{p}(x_{n-1}|c_i)\mathsf{p}(x_n|c_i)p(c_i) \tag{2.3.28}$$

but we can write the joint probability 2.3.28 as a product of the conditional probabilities of the features given the classes as

$$\mathsf{p}(x_1, x_2, \ldots, x_n, c_i) = \prod_{j=1}^{n} \mathsf{p}(x_j|c_i)p(c_i) \tag{2.3.29}$$

Replacing 2.3.29 in 2.3.25

$$\mathsf{p}(c_i|x_1, x_2, \ldots, x_n) = \prod_{j=1}^{n} \mathsf{p}(x_j|c_i)\frac{p(c_i)}{\mathsf{p}(x_1, x_2, \ldots, x_n)} \tag{2.3.30}$$

using maximum a posteriori (MAP) we have 2.3.30 as

$$\begin{aligned}C_{\mathsf{MAP}} &= \text{argmax} \quad \mathsf{p}(c_i|x_1, x_2, \ldots, x_n)\\ &= \text{argmax} \quad \frac{\mathsf{p}(x_1, x_2, \ldots, x_n|c_i)\mathsf{p}(c_i)}{\mathsf{p}(x_1, x_2, \ldots, x_n)}\\ &= \text{argmax} \quad \mathsf{p}(x_1, x_2, \ldots, x_n|c_i)\mathsf{p}(c_i)\end{aligned}$$

$$C_{\mathsf{MAP}} = \text{argmax} \quad p(c_i)\prod_{j=1}^{n} \mathsf{p}(x_j|c_i) \tag{2.3.31}$$

Given the training dataset, we classify the observations according to the class they belong. For example, in this essay, we classify the observations that are fraudulent transactions and those that are legitimate and then get the probability of each class $p(c_i)$. We then multiply $p(c_i)$ by the product of the conditional probability of the new instance given that it belongs to class c_i as shown in 2.3.29. According to Ng and Jordan (2002), the class with the maximum of 2.3.31 is taken to be the class of the new instance that we want to classify.

3. Methodology

In this chapter, we have the steps followed when carrying out the predictive analysis. This will include an explanation on data description, preprocessing the dataset, how the data is split, the methods used for hyperparameter fine tuning and evaluation metrics used.

3.1 Data Description

In this essay, we will use data collected in 2013 from transactions made by European cardholders for two days. The dataset is available in Kaggle website. The dataset contains $284,807$ observations of which only 492 were recorded as fraudulent transactions and the rest were legitimate transactions this means the dataset is highly imbalanced as we have 0.173% of the data as the fraudulent transactions.

[1] We have 30 features and due to privacy policy from the bank, 28 of them are transformed using Principle Component Analysis(PCA) and they are labeled $v1 - v28$ and the features "Amount" and "Time" are not transformed. The feature "Time" represents the time between the first transaction and the other transactions.

Figure 3.1 shows a representation of the two classes in the dataset.

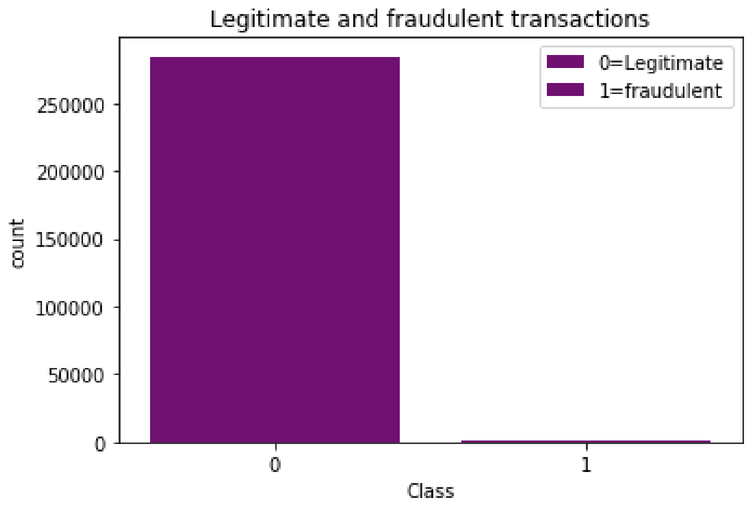


Figure 3.1: Legitimate and fraudulent transactions

We now check whether the features are correlated to each other as this will affect the analysis. We will do this by using a correlation matrix shown below in Figure 3.2.

We see that from the correlation matrix Figure 3.2, there is no correlation between the features V1-V28. The variable "Amount" is negatively correlated with features V2 and V5 and positively correlated with the variables V7 and V20. The variable "Time" has a negative correlation with the variable V3. This is good since there is no high correlation with the features. We can proceed to preprocess our data to make it ready for creating a model.

[1] Credit Card Fraud Detection Dataset

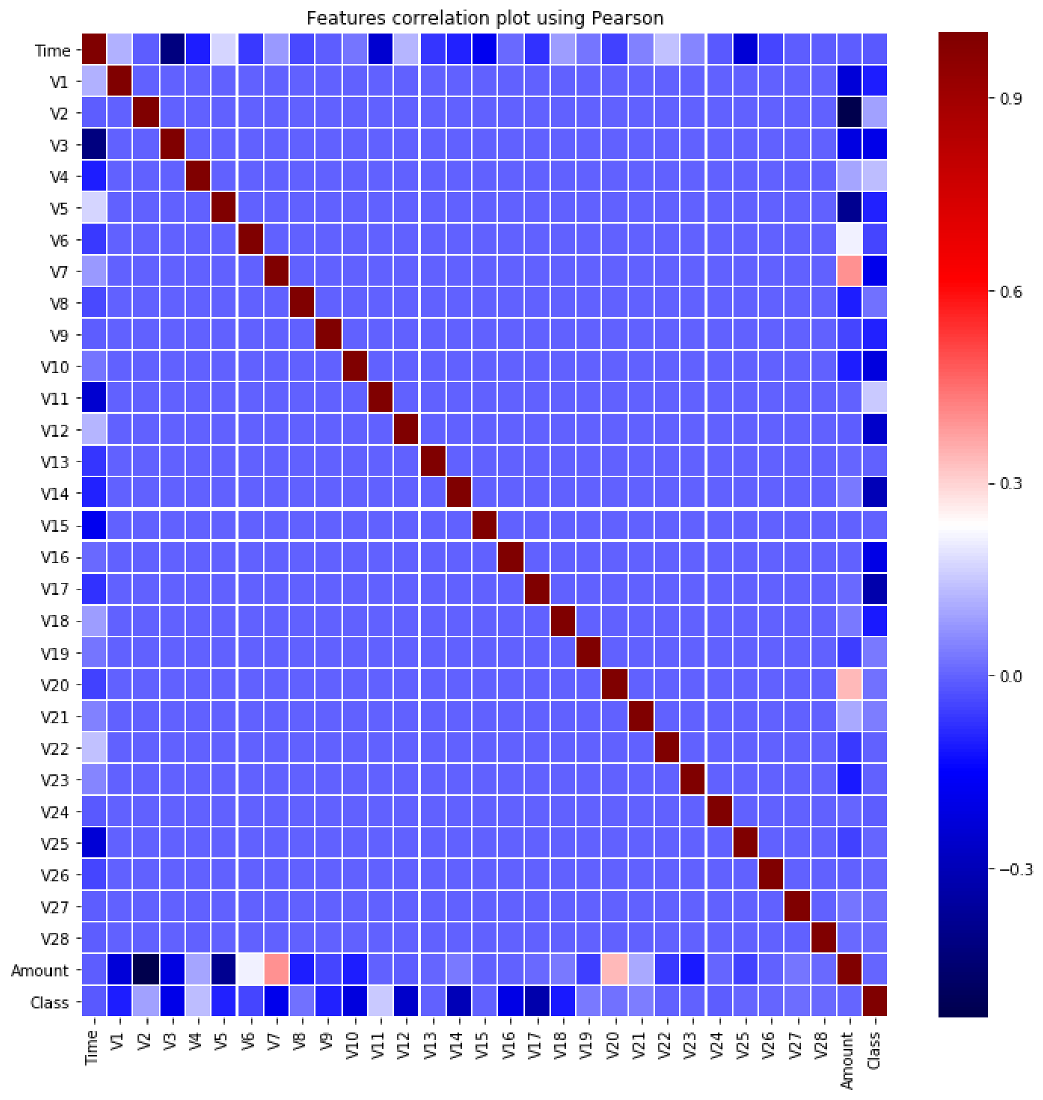


Figure 3.2: Correlation Matrix

3.2 Preprocessing the Dataset

We perform standardization on the feature ”Amount” which is just taking the observation and subtracting the mean then dividing by the standard deviation as follows:

$$\text{z} = \frac{x - \mu}{\sigma} \tag{3.2.1}$$

Standardization helps to keep the observations to be within a given range. We remove the feature ”Time” since it is not important in detecting whether a transaction is legitimate or fraudulent.

3.3 Splitting Data

We split the data into training and testing data. The data we have is shuffled such that when we split we are not likely to face the challenge of having one class in the test data. The training data is used to train the model and the test data is used to check whether the model can be able to work well on data that it has not met before. In this case, we will split the data into 70% for training and 30% for testing.

3.4 Resampling Techniques

We should make sure that we split the dataset before applying any resampling technique otherwise some examples appearing in the training set will also appear in the test data and so the model won't be efficient. In this essay we are going to use the following techniques and compare how they work with different classifiers.

- NearMiss
- Synthetic Minority Oversampling Technique(SMOTE) and Tomek links
- Edited Nearest Neighbor and Condensed Nearest Neighbor

3.5 Fine-Tuning Hyperparameters

When creating any model we need to identify which are the parameters and which are the hyperparameters. The parameters are those variables that are internal with the model and as a practitioner, we don't change them.

According to Shakya (2018), a hyperparameter is a configuration that is external to the model and whose value cannot be estimated from the training data. The hyperparameters can be changed by the practitioner to fit the model and make them work well and this is what is called hyperparameter tuning or hyperparameter optimization.

There are three main methods to use when performing the hyperparameter tuning namely:

- **GridSearchCV**- According to Bergstra and Bengio (2012), in this method we give the values of the hyperparameters as a list and the model goes through all the values in the list and chooses the best values.
- **RandomizedSearchCV**- when tuning hyperparameters using a randomized search, the model randomly picks the values provided in the list and from the most important ones the best hyperparameters are chosen as explained by Bergstra and Bengio (2012).
- **Manual Search**- This is where the practitioner chooses the hyperparameters from past experience and tries a range of them and picks the ones that work well with the given dataset and the classifiers used.

We are going to perform hyperparameter tuning for the three classifiers, K-NN, Random Forest and SVM since the Naive Bayes does not have any hyperparameter to tune. We will use the Manual Search because the dataset is very high dimensional and so it needs a lot of time and resources to tune using the first two tuning methods.

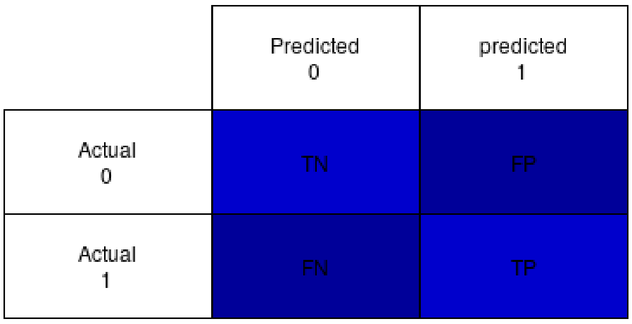
3.5.1 Hyperparameter optimization in Random Forest.

In Random Forest we are going to focus on tuning two main hyperparameters. The n_estimator which is the number of trees making the Random Forest and the max_feature which is the number of features used when doing the splitting on the decision trees.

3.5.2 Hyperparameter tuning in K-Nearest Neighbor (K-NN).
The main hyperparameter in the K-NN classifier is n_neighbor which is the number of neighboring points considered when classifying a new instance.
3.5.3 Hyperparameter tuning in Support Vector Machine.
We are going to tune two main hyperparameters, the kernel and C which is the cost penalty of misclassification.

3.6 Evaluation Metrics

Since our data is highly imbalanced we will not use the accuracy as our evaluation metrics since it will always favor the legitimate class and so give high accuracy which is a misleading measure. We will consider the following evaluation metrics:
3.6.1 Confusion Matrix.
This is the most important metric as from it we can get other metrics. It shows how well the model is working. It is represented as follows:



	Predicted 0	predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

Figure 3.3: A Confusion Matrix

- **True Positive(TP)**- This represents all those observations that we predicted to be positive and that was true. In our case, those predicted to be fraudulent turned out to be fraudulent.
- **False Negative(FN)** - These are those observations taken to be negative but they are positive. For example transactions misclassified as legitimate when they are fraudulent.
- **True Negative(TN)**-This shows all the observations that were taken to be negatives and they were negative. in our case, all the transactions classified as legitimate and they were legitimate.
- **False Positive(FP)**- These are observations that were negative but they were misclassified as positive. For example a legitimate transaction classified as a fraudulent classification.

3.6.2 Recall.
This is an evaluation metric that checks how well can the model detect the true positives in our case it will show how well can a model detect when a transaction made is fraudulent. This will be our main focus since the essay is about fraud detection. A high recall is preferred. It is calculated as:

$$\text{Recall} = \frac{\text{True positive}}{\text{True Positive} + \text{False Negative}}$$

3.6.3 Precision.
This tells us whether what the model classifies in a certain class is the correct classification. In this essay, it will tell us how true is it when a model classifies a transaction to be fraudulent. We will also

focus on the precision since we don't want to misclassify the transactions as this comes at a cost to the financial institutions. We calculate it as follows:

$$\text{Precision} = \frac{\text{True positive}}{\text{True Positive} + \text{False Positive}}$$

3.6.4 F1-Score.

This is just a weighted average of both the precision an the recall. We calculate it as follows

$$\text{F1-score} = 2\left(\frac{\text{Precision * Recall}}{\text{Precision} + \text{Recall}}\right)$$

4. Results

In this section, we will have the results obtained from the four classifiers using the 3 resampling approaches. Due to page limitations, the numbers in the confusion matrices may need some zooming out to be visible.

4.1 Random Forest

4.1.1 NearMiss.

- **The best hyperparameters were n_estimators=400 and max_features=sqrt.**

From Table 4.1 we see the evaluation of Random Forest and NearMiss where the recall is at 86% and the precision is 34% meaning that when we resample the dataset using NearMiss, the model can detect 86% of the fraudulent transactions but we can only be 34% sure that these transactions are actually classified correctly as fraudulent. This is not a good thing because the percentage of misclassification is very high. This is seen in the confusion matrix Figure 4.1 where 229 legitimate transactions are classified as fraudulent transactions.

Table 4.1: Evaluation on Random Forest and Nearmiss

Evaluation Metrics	
Precision	0.34
Recall	0.86
F1-score	0.49

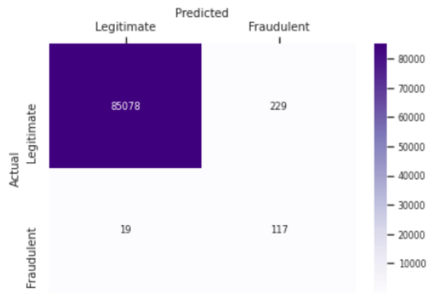


Figure 4.1: Confusion matrix-RF and Nearmiss

4.1.2 Edited Nearest Neighbor(ENN) and Condensed Nearest Neighbor(CNN).

- **The best hyperparameters were n_estimators=400 and max_features=sqrt.**

From Table 4.2 we see that the recall decreased from 86% to 84% which means that now our model can detect 84% of the fraudulent transactions. The precision increased from 34% to 84% so this means that the transactions classified as fraudulent are 84% true. This is a good model as seen in Figure 4.2 but the research is more interested in detecting the fraudulent transactions so we want a higher recall.

Table 4.2: Evaluation Metrics on RF-ENN+CNN

Evaluation Metrics	
Precision	0.84
Recall	0.84
F1-score	0.84

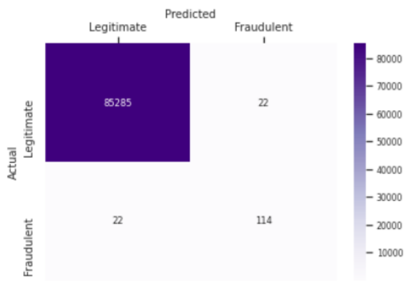


Figure 4.2: Confusion Matrix-Rf and ENN+CNN

4.1.3 Sythetic Minority Oversampling Technique(SMOTE) and Tomek Links Removal.

- **The best hyperparameters were n_estimators=400 and max_features=sqrt.**

Table 4.3 shows that when using SMOTE and Tomek links removal the recall and precision increases to 87% and 86% respectively meaning now the model can detect 87% of the fraudulent transactions with 86% certainity. This is a good model since from the confusion matrix Figure 4.3 we see only 20 legitimate transactions are misclassified as fraudulent.

Table 4.3: Evaluation Metrics on RF-SMOTE and Tomek Links Removal

Evaluation Metrics	
Precision	0.86
Recall	0.87
F1-score	0.86

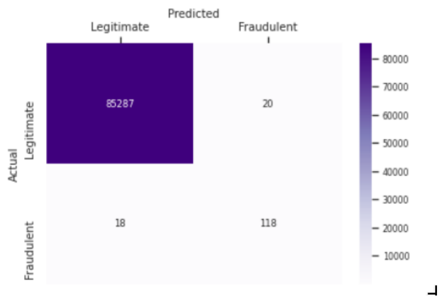


Figure 4.3: Confusion Matrix-RF and Smote and Tomek Links Removal

4.2 K-Nearest Neighbor(K-NN)

4.2.1 NearMiss.

- **The best hyperparameter was n_neighbors=7.**

The Table 4.4 shows that when we use K-NN and NearMiss, 81% of the fraudulent transactions are detected and we are 45% sure that these transactions are truely fraudulent. This is not a good model as we can see from Figure 4.4 that 136 of the legitimate transactions are wrongly classified as fraudulent transactions.

Table 4.4: Evaluation Metrics on KNN-Nearmiss

Evaluation Metrics	
Precision	0.45
Recall	0.81
F1-score	0.58

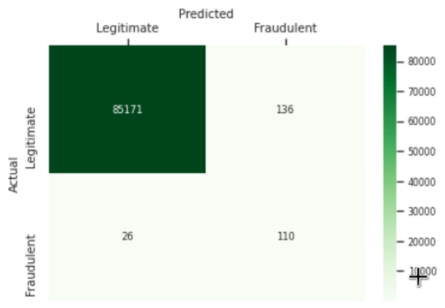


Figure 4.4: Confusion Matrix-KNN and Nearmiss

4.2.2 SMOTE and Tomek Links Removal.

- **The best hyperparameter was n_neighbors=5.**

From the evaluation metrics seen in Table 4.5 we have that the model can detect 90% of the fraudulent transactions but with only 45% certainity. This is not a good model since from Figure 4.5 152 of the legitimate transactions are misclassified as fraudulent transactions.

Table 4.5: Evaluation Metrics on K-NN-SMOTE+Tomek links

Evaluation Metrics	
Precision	0.45
Recall	0.90
F1-score	0.60

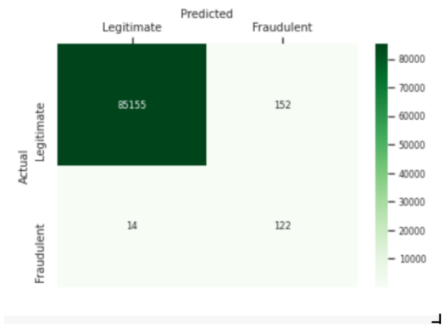


Figure 4.5: Confusion Matrix-KNN and Nearmiss

4.2.3 ENN and CNN.

- **The best hyperparameter was n_neighbors=5.**

The evaluation metrics in Table 4.6 shows that when KNN is used together with ENN and CNN the model is able to detect 83% of the fraudulent transactions with 82% certainity. This is a good model since from Figure 4.6 we have only 22 of the non-fraudulent transactions are misclassified as fraudulent transactions.

Table 4.6: Evaluation Metrics on KNN-ENN+CNN

Evaluation Metrics	
Precision	0.82
Recall	0.83
F1-score	0.83

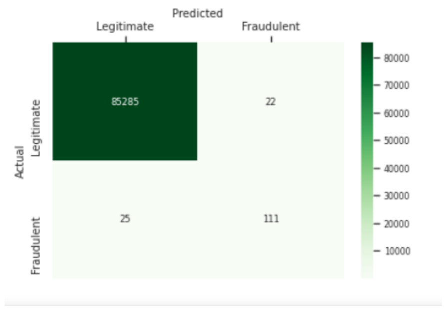


Figure 4.6: Confusion Matrix-KNN and ENN+CNN

4.3 Naive Bayes

4.3.1 NearMiss.

The results when using Naive Bayes and NearMiss are shown in Table 4.7. The model is not good as the recall is at 57% meaning it can only detect 57% of the fraudulent transactions and from the precision, we are only 12% sure that these fraudulent transactions are truly in the right classification. Figure 4.7 shows that we have 554 legitimate transactions classified wrongly as fraudulent. The F1 score is very poor 20%.

Table 4.7: Evaluation Metrics on Naive Bayes-Nearmiss

Evaluation Metrics	
Precision	0.12
Recall	0.57
F1-score	0.20

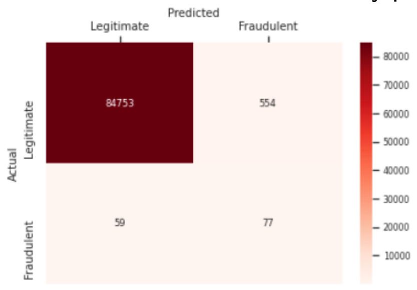


Figure 4.7: Confusion Matrix-Naive Bayes and Nearmiss

4.3.2 ENN and CNN.

From Table 4.8 we see that the recall has improved from 57% to 85% meaning that now our model can detect 85% of the fraudulent transaction but with only 6% of this been true which is half certainty compared to the NearMiss. This is a poor model since from Figure 4.8 1865 of the non-fraudulent transactions are misclassified.

Table 4.8: Evaluation Metrics on Naive Bayes-ENN + CNN

Evaluation Metrics	
Precision	0.06
Recall	0.85
F1-score	0.11

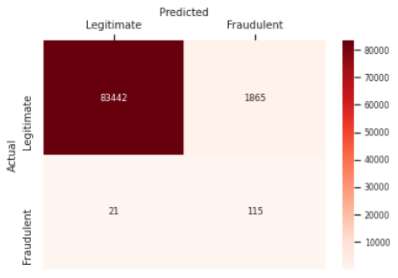


Figure 4.8: Confusion Matrix-Naive Bayes and ENN+CNN

4.3.3 SMOTE and Tomek Link Removal.

The model works well in terms of recall as we can see in Table 4.9, the model can detect 88% of the fraudulent transactions but the possibility of this being true is only 6% and this is not a good model since even the F1 score is very low 11%. The model misclassify 2012 legitimate transactions as fraudulent transactions as seen in Figure 4.9.

Table 4.9: Evaluation Metrics on Naive Bayes-SMOTE+Tomek links

Evaluation Metrics	
Precision	0.06
Recall	0.88
F1-score	0.11

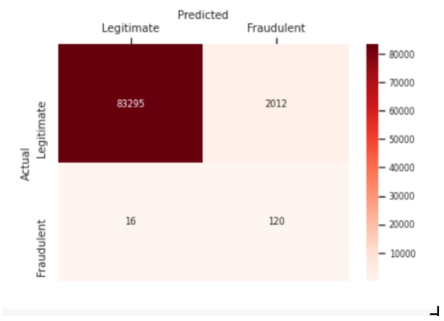


Figure 4.9: Confusion Matrix-Naive Bayes and SMOTE + tomek links removal

4.4 Support Vector Machine (SVM)

4.4.1 SMOTE and Tomek Links.

- **The best hyperparameters were kernel=linear, C=0.01.**

Table 4.10 shows the evaluation metrics when using the svm and SMOTE and Tomek links. The model is able to predict a very high number of the fraudulent transactions as seen in the recall 93% but the certainity that these transactions are actually fraudulent is very low 6%. The moodel is not good as seen in Figure 4.10 2138 of the legitimate transactions are misclassified as fraudulent transactions.

Table 4.10: Evaluation Metrics on SVM-SMOTE+Tomeklinks

Evaluation Metrics	
Precision	0.06
Recall	0.93
F1-score	0.11

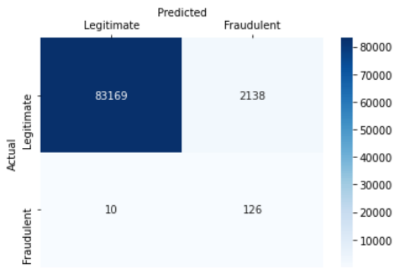


Figure 4.10: Confusion Matrix-SVM and SMOTE+Tomeklinks

4.4.2 Nearmiss.

- **The best hyperparameters were kernel=linear, C=0.001.**

Table 4.11 we see that both the recall and the precision are high which means this is a good model. The recall is at 82% implying that the model can detect 82% of the fraudulent transactions and from the precision, we have that we are 80% certain that the detected fraudulent transactions are true. From Figure 4.11 we see that only 28 of the legitimate transactions are misclassified.

Table 4.11: Evaluation Metrics on SVM-Nearmiss

Evaluation Metrics	
Precision	0.80
Recall	0.82
F1-score	0.81

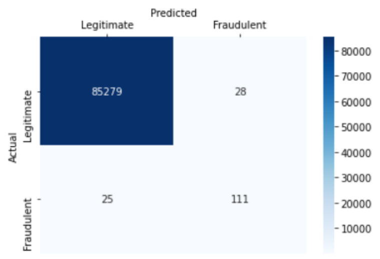


Figure 4.11: Confusion Matrix-SVM and Nearmiss

4.4.3 ENN and CNN.

- **The best hyperparameters were kernel=linear, C=0.01.**

Table 4.12 we see that the recall has incresed from 82% to 83% so now the model can detect 83% of the fraudulent transactions but the precison has decreased to 78% and so we are less certain compared to the nearmiss approach and this is seen in Figure 4.12 the number of misclassified legitimate transactions have increased from 28 to 31 but it is still a good model.

Table 4.12: Evaluation Metrics on SVM-ENN+CNN

Evaluation Metrics	
Precision	0.78
Recall	0.83
F1-score	0.81

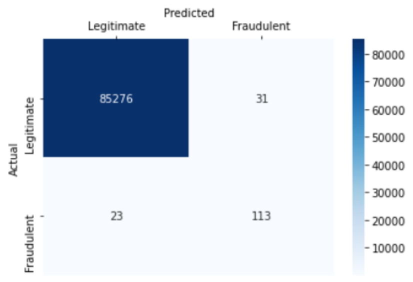


Figure 4.12: Confusion Matrix-SVM and ENN+CNN

4.5 General Discussion

Table 4.13: Summary on the evaluation of the four classifiers

Classifier	Resampling Approach	Precision	Recall	F1 score
Random Forest	Nearmiss	0.34	0.86	0.49
	SMOTE and Tomek links	0.86	0.87	0.86
	ENN and CNN	0.84	0.84	0.84
KNN	Nearmiss	0.45	0.81	0.58
	SMOTE and Tomek links	0.45	0.90	0.60
	ENN and CNN	0.82	0.83	0.83
Naive Bayes	Nearmiss	0.12	0.57	0.20
	SMOTE and Tomek links	0.06	0.88	0.11
	ENN+CNN	0.06	0.85	0.11
SVM	Nearmiss	0.80	0.82	0.81
	SMOTE and Tomek links	0.06	0.93	0.11
	ENN+CNN	0.78	0.83	0.81

Table 4.13 above shows a summary on the evaluation of the four classifiers. We see that all of them have high recall meaning that they can detect any fraudulent transaction easily. On the other hand, the precision is low for some classifiers meaning that even though they can detect the fraudulent transactions most of them are a misclassification of the legitimate transactions.

Random Forest works well with SMOTE and Tomek Links as well as ENN and CNN because of its ensemble nature it can deal with imbalanced data so when some resampling is done the classifier performs very well. NearMiss does not work well with Random Forest because maybe some of the information is lost in the process of reducing the majority instances.

The K-NN is a nearest neighbor approach and that is why it works well when it comes to ENN and CNN since they are also nearest neighbor resampling approaches. Though in the other two approaches the recall is high, the misclassification rate is also likely to happen as seen in the low precision. Naive Bayes does not work well with any of the resampling approaches. This is because of the naive assumption it makes of feature independence which is not likely to happen in real data.

SVM works well with a small dataset and that is why Nearmiss and ENN and CNN works well because they are undersampling techniques. Another reason is that the classifier looks for the hyperplane with the maximum margin that separates the two classes and using the two resampling approaches, they make this easy and thus giving very good results.

5. Conclusion and Future Work

This research was carried out to detect fraudulent transactions when payments are made using credit cards by carrying out a predictive analysis using machine learning algorithms. The dataset used is from Kaggle dataset website which is taken from a European bank in 2013. The main objective was to deal with the highly imbalanced data and evaluate the best models from the four classifiers, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbor (K-NN) and Naive Bayes that best detects these fraudulent transactions. The resampling techniques used were Nearmiss, Synthetic Minority Oversampling Technique (SMOTE) and Tomek links as well as ENN and CNN.

When using NearMiss, SVM performed better than all the other classifiers. It had a recall of 82%, a precision of 80% and an F1 score of 81%. Random Forest worked well with the hybrids of SMOTE and Tomek Links Removal and also ENN and CNN but SMOTE and Tomek Links Removal was better as it had a recall of 87%, a precision of 86% and an F1 score of 86% compared to the latter which had 84% for all the three evaluation metrics.

K-NN works well with ENN and CNN with a recall of 83%, a precision of 82% and an F1 score of 83%. Naive Bayes performed poorly in all three resampling approaches due to the naive assumption of features been independent.

In conclusion, Random Forest recorded the best results compared to the other four classifiers and this is because it is an ensemble classifier and so it can deal with the imbalance nature of the data.

In the future, when we have computational resources such as high-performance computers (HPC), we can perform hyperparameter tuning using advanced methods that are likely to improve the performance of these classifiers. They will also save on the time spent when training the models. Also, we can include the cost-sensitive approaches to solve the class imbalance that will take care of misclassification costs as this is important to financial institutions.

Acknowledgements

My sincere gratitude goes to God for giving me good health and enabling me to finish my studies. I want to thank AIMS funders for the funding and support throughout the studies. A big thank you to my supervisor, Prof. Phillip Mashele from North West University for his support.

I sincerely appreciate my tutors Reem and Rock for guidance throughout the essay. I thank all the AIMS staffs for their contribution.

My special thanks go to my mum and dad for their unending support and prayers. I am grateful to Dave, Winnie and my whole family for the support and encouragement. I thank Serge, Gerald and Kiprono for their help throughout my studies.

Finally, I appreciate my AIMS family who made the stay at AIMS meaningful.

References

Alejo, R., Sotoca, J. M., Valdovinos, R. M., and Toribio, P. Edited nearest neighbor rule for improving neural networks classifications. In Zhang, L., Lu, B.-L., and Kwok, J., editors, *Advances in Neural Networks - ISNN 2010*, pages 303–310, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

Anderson, R., Barton, C., Böhme, R., Clayton, R., Van Eeten, M. J., Levi, M., Moore, T., and Savage, S. Measuring the cost of cybercrime. In *The economics of information security and privacy*, pages 265–300. Springer, 2013.

Baader, G. and Krcmar, H. Reducing false positives in fraud detection: Combining the red flag approach with process mining. *International Journal of Accounting Information Systems*, 31:1–16, 2018.

Batista, G., Prati, R., and Monard, M.-C. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, 6:20–29, 06 2004. doi: 10.1145/1007730.1007735.

Beckmann, M., Ebecken, N., and Lima, B. A knn undersampling approach for data balancing. *Journal of Intelligent Learning Systems and Applications*, 7:104–116, 11 2015. doi: 10.4236/jilsa.2015.74010.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012. URL http://dblp.uni-trier.de/db/journals/jmlr/jmlr13.html#BergstraB12.

Berrar, D. Bayes' theorem and naive bayes classifier. *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, page 403, 2018.

Biau, G. Analysis of a random forests model. *J. Mach. Learn. Res.*, 13:1063–1095, 2010.

Biau, G. and Scornet, E. A random forest guided tour. *Springer*, 132:385–395, 01 2016. doi: 10.1016/j.procs.2018.05.199.

Boriah, S., Chandola, V., and Kumar, V. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the 2008 SIAM international conference on data mining*, pages 243–254. SIAM, 2008.

Breiman, L. "bagging predictors"technical report. *UC Berkeley*, 1994.

Breiman, L. Consistency for a simple model of random forests. *Citeseer*, 132:385–395, 01 2004. doi: 10.1016/j.procs.2018.05.199.

Bunkhumpornpat, C., Sinapiromsaran, K., and Lursinsap, C. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In Theeramunkong, T., Kijsirikul, B., Cercone, N., and Ho, T.-B., editors, *Advances in Knowledge Discovery and Data Mining*, pages 475–482, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

Chomboon, K., Chujai, P., Teerarassamee, P., Kerdprasop, K., and Kerdprasop, N. An empirical study of distance metrics for k-nearest neighbor algorithm. In *Proceedings of the 3rd international conference on industrial application engineering*, pages 1–6, 2015.

Dal Pozzolo, A., Caelen, O., Le Borgne, Y.-A., Waterschoot, S., and Bontempi, G. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert systems with applications*, 41(10): 4915–4928, 2014.

Dal Pozzolo, A., Caelen, O., and Bontempi, G. When is undersampling effective in unbalanced classification tasks? In Appice, A., Rodrigues, P. P., Santos Costa, V., Soares, C., Gama, J., and Jorge, A., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 200–215, Cham, 2015. Springer International Publishing.

Dasarathy, B. *Nearest neighbor (NN) norms: nn pattern classification techniques*. IEEE Computer Society Press tutorial. IEEE Computer Society Press, 1991. ISBN 9780818659300. URL https://books.google.co.za/books?id=k2dQAAAAMAAJ.

Dheepa, V. and Dhanapal, R. Behavior based credit card fraud detection using support vector machines. In *SOCO 2012*, 2012.

Fix, E. and Hodges, J. *Discriminatory Analysis: Nonparametric Discrimination, Small Sample Performance*. Report. Air University, USAF School of Aviation Medecine, 1952. URL https://books.google.co.za/books?id=bEsYnQEACAAJ.

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., and Pineau, J. *An Introduction to Deep Reinforcement Learning*. 2018.

Freund, Y. and Schapire, R. E. A short introduction to boosting. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406. Morgan Kaufmann, 1999.

Han, J., Kamber, M., and Pei, J. Data mining concepts and techniques, third edition, 2012. URL http://www.amazon.de/Data-Mining-Concepts-Techniques-Management/dp/0123814790/ref=tmm_hrd_title_0?ie=UTF8&qid=1366039033&sr=1-1.

Hart, P. The condensed nearest neighbor rule (corresp.). *IEEE Transactions on Information Theory*, 14 (3):515–516, 1968.

He, H. and Garcia, E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

Hjerpe, A. Computing random forests variable importance measures (VIM) on mixed continuous and categorical data. *Springer*, 132:385–395, 01 2016. doi: 10.1016/j.procs.2018.05.199.

Holte, R. C., Acker, L., Porter, B. W., et al. Concept learning and the problem of small disjuncts. In *IJCAI*, volume 89, pages 813–818. Citeseer, 1989.

Jain, A. K., Murty, M. N., and Flynn, P. J. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

Jakkula, V. Tutorial on support vector machine (SVM). *School of EECS, Washington State University*, 37, 2006.

Japkowicz, N. and Stephen, S. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

Kotsiantis, S. B. Supervised machine learning: A review of classification techniques. *Informatica (Slovenia)*, 31:249–268, 2007.

Kroon, S. and Omlin, C. Getting to grips with support vector machines: Theory. *South African Statistical Journal*, 38, 01 2004.

Kurien, K. L. and Chikkamannur, A. A. Benford's law and deep learning autoencoders: An approach for fraud detection of credit card transactions in social media. In *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, pages 1030–1035. IEEE, 2019.

Ling, C. X. and Sheng, V. S. *Class Imbalance Problem*, pages 171–171. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_110. URL https://doi.org/10.1007/978-0-387-30164-8_110.

Liu, H. and Haig, E. Induction of classification rules by gini-index based rule generation. *Information Sciences*, 436-437:227–246, 04 2018. doi: 10.1016/j.ins.2018.01.025.

Maes, S., Tuyls, K., Vanschoenwinkel, B., and Manderick, B. Credit card fraud detection using bayesian and neural networks. In *In: Maciunas RJ, editor. Interactive image-guided neurosurgery. American Association Neurological Surgeons*, pages 261–270, 1993.

Maimon, O. and Rokach, L. Data mining and knowledge discovery handbook. *Springer*, 132:385–395, 01 2005. doi: 10.1016/j.procs.2018.05.199.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. Foundations of machine learning. In *Adaptive computation and machine learning*, 2012.

Ng, A. Y. and Jordan, M. I. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

Patil, S., Nemade, V., and Soni, P. Predictive modelling for credit card fraud detection using data analytics. *Procedia Computer Science*, 132:385–395, 01 2018. doi: 10.1016/j.procs.2018.05.199.

Perantalu, V. and BhargavKiran, K. Credit card fraud detection using predictive modeling: a review. *International Journal of Innovative Research in Technology*, 2017.

Pun, J. and Lawryshyn, Y. Improving credit card fraud detection using a meta-classification strategy. *International Journal of Computer Applications*, 56(10), 2012.

Rahman, M. and Davis, D. N. Addressing the class imbalance problem in medical datasets. *International Journal of Machine Learning and Computing*, 3:224, 04 2013. doi: 10.7763/IJMLC.2013.V3.307.

Renjith, S. Detection of fraudulent sellers in online marketplaces using support vector machine approach. *arXiv preprint arXiv:1805.00464*, 2018.

Rokach, L. Ensemble methods for classifiers. In *Data mining and knowledge discovery handbook*, pages 957–980. Springer, 2005.

Shabbir, J. and Anwer, T. Artificial intelligence and its role in near future. *arXiv preprint arXiv:1804.01396*, 2018.

Shakya, R. Application of machine learning techniques in credit card fraud detection. *UNLV Theses, Dissertations, Professional Papers, and Capstones. 3454.*, 2018. URL https://digitalscholarship.unlv.edu/thesesdissertations/3454.

Shannon, C. E. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948. URL http://dblp.uni-trier.de/db/journals/bstj/bstj27.html#Shannon48.

Taheri, S. and Mammadov, M. Learning the naive bayes classifier with optimization models. *International Journal of Applied Mathematics and Computer Science*, 23, 12 2013. doi: 10.2478/amcs-2013-0059.

Taheri, S. and Mammadov, M. A. Structure learning of bayesian networks using global optimization with applications in data classification. *Optimization Letters*, 9:931–948, 2015.

Tomek, I. Two Modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(2): 679–772, 1976.

Usama, M., Qadir, J., Raza, A., Arif, H., Yau, K. A., Elkhatib, Y., Hussain, A., and Al-Fuqaha, A. Unsupervised machine learning for networking: Techniques, applications and research challenges. *IEEE Access*, 7:65579–65615, 2019.

Wiese, B. and Omlin, C. *Credit Card Transactions, Fraud Detection, and Machine Learning: Modelling Time with LSTM Recurrent Neural Networks*, pages 231–268. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

Wilson, D. and Martinez, T. Improved heterogeneous distance functions. *J. of Artif. Intell. Res.*, 6, 06 2000.

Yen, S.-J. and Lee, Y.-S. *Under-Sampling Approaches for Improving Prediction of the Minority Class in an Imbalanced Dataset*, pages 731–740. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

Zhang, J. and Mani, I. KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. In *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets*, 2003.

Zheng, Z. Feature selection for text categorization on imbalanced data. *ACM SIGKDD Explorations Newsletter*, 6:2004, 2004.