Anomaly Detection in Highly Imbalanced Dataset

Sridhar Amirneni

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Data Science)

Supervisor: Dr. Bahman Honari

August 2019

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work. A section of this study has been researched upon as a part of Security and Privacy (CS7NS5) module.

Sridhar Amirneni

August 12, 2019

Acknowledgments

Foremost, I would like to express my sincere gratitude to my advisor, Dr. Bahman Honari, for his valuable and constructive suggestions during the planning and development of this research work. I am also grateful to Computer Science and Statistics Department of Trinity College Dublin to provide me with infrastructure and good environment to work on my thesis.

I would also like to thank my second reader Dr. Georgios Iosifidis for his time and suggestions which helped me immensely.

I must express my profound gratitude to my parents Vijayprasad Amirneni and Anuradha Amirneni for providing me with unfailing support and motivation throughout my year of study. Finally, I would like to thank all my friends and classmates who made my time here at Trinity College Dublin a memorable one.

Sridhar Amirneni

University of Dublin, Trinity College
August 2019

Anomaly Detection in Highly Imbalanced Dataset

Sridhar Amirneni, Master of Science in Computer Science

University of Dublin, Trinity College, 2019

Supervisor: Dr. Bahman Honari

Credit Card Fraud is a major problem in todays financial world. There has been an exponential increase in the losses due to fraud in the recent years. These losses are expected to increase in the future, due to which need for research and investment is required in this field. The major problems in this are the dynamic nature of the fraud and the less financial data available. The data available has a high class imbalance with very less number of fraudulent cases present.

When this data is given as input to the current classification models, it is tending to be partial towards the majority class. Consequently, it is labeling a fraudulent transaction as a non fraudulent one. To overcome this, a Random undersampling and Synthetic Minority Over-sampling Technique (SMOTE) at the data level are implemented. The algorithms implemented were Logistic Regression, k Nearest Neighbour, Support Vector Machine, and Decision Tree Classifier. A neural network with one hidden layer was also constructed. All the models were then compared for performance between undersampling and SMOTE. The evaluation metrics used were precision, recall, f1-score, support, area under ROC curve, and confusion matrices. The results showed that Logistic Regression performed better than all the other classifiers but Neural Network with SMOTE implemented had the least number of misclassified transactions.

Contents

Acknowledgments **iii**

Abstract **iv**

List of Tables **vii**

List of Figures **viii**

Chapter 1 Introduction **1**

- 1.1 Background 1
- 1.2 Challenges in Fraud Detection 2
- 1.3 Objective 3
- 1.4 Motivation 4

Chapter 2 Related Works **5**

Chapter 3 Model **8**

- 3.1 Dimensionality Reduction 8
  - 3.1.1 Principal Component Analysis (PCA) Transformation 8
  - 3.1.2 t-Distributed Stochastic Neighbor Embedding (tSNE) 10
- 3.2 Classification Algorithms 11
  - 3.2.1 Logistic Regression 11
  - 3.2.2 K Nearest Neighbours (kNN) 12
  - 3.2.3 Support Vector Classification 18
  - 3.2.4 Decision Tree Classification 20
- 3.3 Sampling Techniques 23

3.3.1 Near Miss Technique 23
3.3.2 Random Undersampling 23
3.3.3 Synthetic Minority Over-Sampling Technique (SMOTE) 24
3.4 Neural Networks 25
3.5 Evaluation Metrics 29

Chapter 4 Methodology **32**
4.1 Dataset 32
4.2 Data Preparation 32
4.3 Data Exploration 33

Chapter 5 Results **40**

Chapter 6 Security and Privacy **45**
6.1 Challenges 45
6.2 Data Security 45
6.3 Data Privacy 46

Chapter 7 Conclusion **48**

Chapter 8 Future Work **49**

Bibliography **50**

List of Tables

5.1 Training Scores 40
5.2 Cross Validation Scores 41
5.3 Area under ROC Curve 41
5.4 Logistic Regression Metrics 42
5.5 KNears Neighbors Metrics 43
5.6 Support Vector Classifier Metrics 43
5.7 Decision Tree Classifier Metrics 43

List of Figures

1.1 Losses due to Card fraud in the European Union during 2012-2016 . . . 2
1.2 Fraud Detection Process 3

3.1 Normal vs Cauchy Distribution 11
3.2 Sigmoid Function 12
3.3 Cluster Identification 13
3.4 Cluster Identification 13
3.5 Cluster Segregation for K=1 14
3.6 Cluster Segregation for K=5 14
3.7 Cluster Segregation for K=7 15
3.8 Training Error Rate for kNN 15
3.9 Validation Error Curve for kNN 16
3.10 Variance of Clusters with respect to K 17
3.11 Choosing of the hyperplane 18
3.12 Hyperplane in different dimensions 19
3.13 Ambiguities in Hyperplane Position 19
3.14 Decision Tree Example 20
3.15 Generation of Decision Tree 22
3.16 SMOTE Over Sampling 24
3.17 Data Flow in a Neural Network 26
3.18 Architecture of a Neural Network 27
3.19 Mathematical form of Neural Network 27
3.20 Confusion Matrix Structure 30

4.1 Class Imbalance in the dataset 33

4.2 Distribution of Amount & Transaction Time 33
4.3 Correlation Matrices before and after Under-sampling 35
4.4 Negative Correlation Box Plots 36
4.5 Positive Correlation Box Plots 36
4.6 Gaussian Distribution of the variables 37
4.7 Clusters after dimensionality reduction 37
4.8 Learning Curves of the Classifiers 38

5.1 ROC Curves for the different classifiers 41
5.2 Confusion Matrices 42
5.3 Confusion Matrix of Neural Network with Undersampling 44
5.4 Confusion Matrix of Neural Network with SMOTE 44

Chapter 1

Introduction

1.1 Background

Society nowadays is dependent on the internet more than ever. This has led to increased use of credit and debit cards. The use of them is so high that they have become the standard for ecommerce and Point of Sale (POS) transactions. Recent reports from Central Bank of Ireland states that card transactions rose to 17.2 billion indicating an increase of 9%. The total number of card transactions excluding ATM transactions was 244 million. Due to the high volume of electronic transactions mainly due to its accessibility, it has opened up new avenues for perpetrators to commit fraud.

The number of card transactions seems to have a rising trend over the years leading to a similar trend of stolen card numbers. The fraud losses also follow a similar trend with higher losses being reported each quarter. Fraud Detection System (FDS) has become an indispensable tool to maintain the integrity of the electronic payments. A safe and reliable banking system for ecommerce requires real-time analysis of the transactions to allow genuine users to carry out their dealings.

Basel Committee separated the fraud into two main categories: internal and external. Internal fraud arises from the malicious intent of the management or employees, while external fraud refers to the stolen, lost, or forged cards to accept or withdraw cash in camouflaged ways.

Credit card fraud solutions can be split into two categories mainly prevention which is not allowing the fraud to occur in the first place and detection which is the action

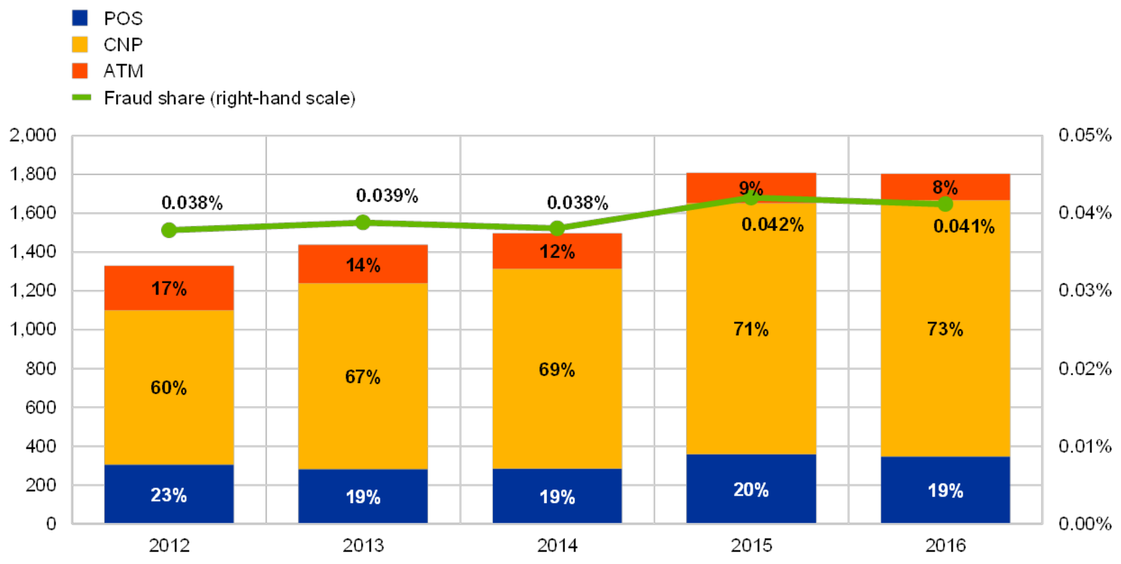

Figure 1.1: Losses due to Card fraud in the European Union during 2012-2016

taken after a fraud has occurred. Data mining methods and rule-based models are the main types of models that are used for the prevention of frauds. Fraudsters in recent times are coming up with innovative methods to commit fraud. Due to the high volume of transactions that take place every day, it is not possible for humans to manually check each transaction if it is valid or not.

1.2 Challenges in Fraud Detection

Designing a fraud detection system is not as easy as it looks. The developer needs to have a complete list of type of fraud cases that can occur. This can use various permutations of the shortcomings at various stages. Another main problem with these systems is that the number of fraud cases are very less. The fraud cases are sometime also labelled as not fraud due to limited information about the transaction. This leads to low performance of the machine learning models and loss of customer value.

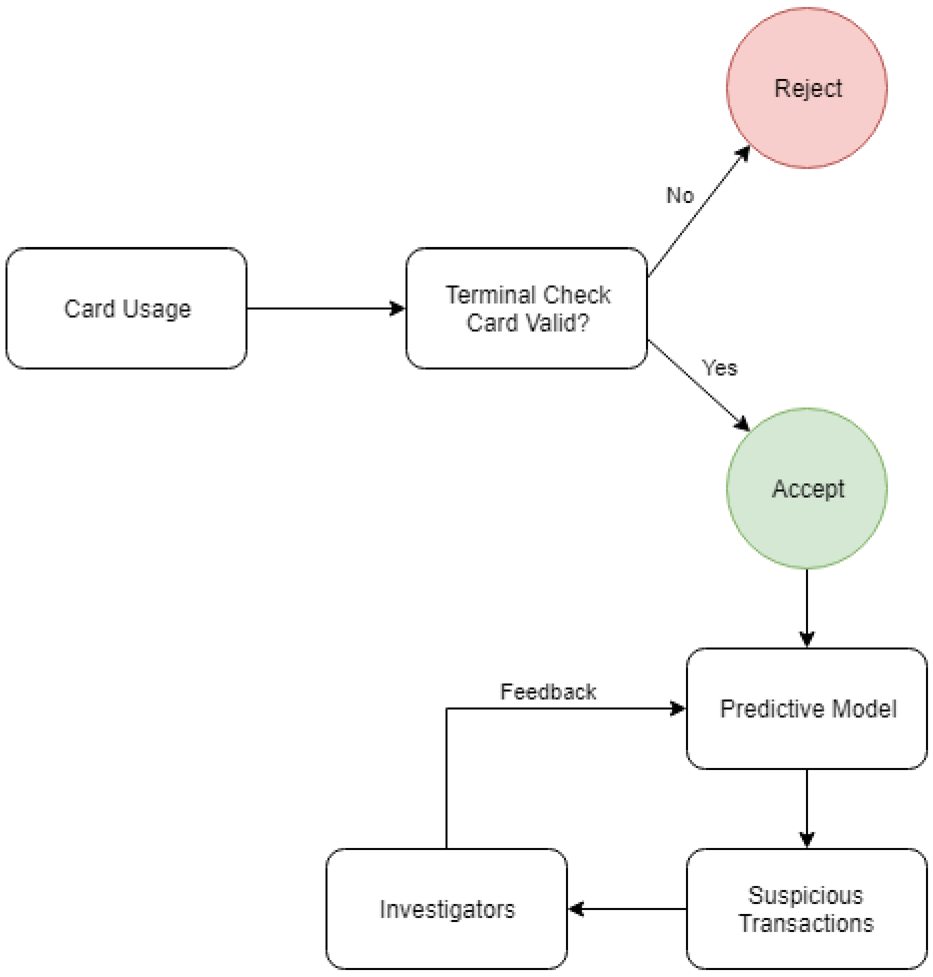

Figure 1.2: Fraud Detection Process

1.3 Objective

The main objective of this thesis is to reduce the impact of class imbalance and make sure that the models perform better even when there are a smaller number of fraud cases that are available. The focus is to reduce the false positives as well as false negatives to detect fraud transactions correctly as well as not to affect the customer experience. Different methods are implemented to deal with the class imbalance problem and a series of models are also implemented to document the effect of these techniques on the prediction models.

1.4 Motivation

The solution if found can be very useful in improving the quality of service of financial institutions and optimizing the customer experience. This can also reduce the amount lost due to fraud transactions with the extra available money being used for training and research purposes. Also, the relationship between machine learning and anomaly detection in this domain is still under-examined. Studies have been done before, about the impact of machine learning in this area, especially with regards to key event periods, like financial recession.)

Chapter 2

Related Works

Since credit card fraud has grown exponentially over the past years, many researchers have started to develop fraud detection systems (FDS). They are actively working to solve various problems in the FDS such as class imbalance, class overlapping, changes in fraud techniques, etc. In this section, we will discuss some of the recent research works in this sector. In a recent study, [1] have emphasised on the hidden Markov model for credit card fraud detection. They have explained the underlying stochastic process of the model. They have also done some research on the spending patterns of the customers. This has led to an increase in accuracy of the model. They have developed the system in such a way that the model can be used in real time scenarios to check if a transaction is fraud or not. The system is also highly scalable.

[2] in his research study has created a memory enhanced framework for the credit card fraud detection. The model consists of a sequential memory-enhanced neural network. The authors have used logs of transactions which brings this model closer to a real-world implementation. It also has a low false positive rate which helps in keeping the system robust and reducing false complaints. The drawback of this model is that a detailed description of all the customers was a part of the model. This is difficult to maintain in most of the financial institutions.

In [3], the researchers have done some a simple comparison of models for clustering and classification of the credit card data. They have explained the different types of outlier detection that has been mentioned in various research papers. They have not performed any experimental analysis of a credit card dataset and hence have not published any metrics for the performance of the models.

Feature engineering for the purpose of fraud detection is the main emphasis for [4]. They use label propagation combined with network characteristics to extract the features of the fraudsters. They are proving that the fraudster mentality is a network based one. Based on their model, they are assigning a fraud score to each node. This score gives a probability of how much chance there is for that node to commit a fraud. The weight setting and the initialisation method have been changed to improve the model. Once this information is gathered, they then implement techniques like graph calculation and machine learning to improve the model score.

[5] extract users profiles based on their online transaction records. This method also considers the assortment of user behaviours that are present online. This is then used to overcome the limitation of the Markov chain models. Their experiment with the dataset also proves the same. A major improvement can be made if they use clustering algorithms to classify the different features of the transactions. They can also use various other data sources such as user comments and user click history to raise the accuracy.

CoDetect is a new architecture that has been proposed by [6]. Based on graph similarity and feature matrix, it detects the various frauds that are committed. They have also introduced a confidence level which is used to flag suspicious transactions as well. With the help of this, investigators can not only see the fraud transaction but can also go ahead and see the feature based on which the flag was raised. They have done experiments on real world datasets with positive results. Feature patterns are also generated which can help detect new ways of fraud.

A detailed study of different machine learning models is done by [7] to detect frauds. They have used ensemble method to stack various models and compare the results. The models that they have compared are Bayesian, Decision Trees, Neural Networks, Logistic Regression, and Support Vector Machine. The focus of the study is Adaboost. Matthews Correlation Coefficient is the metric used. They achieve a perfect score of 1.0 using Adaboost and majority voting. They also added various amounts of noise to the data so that they can be sure of the results. Up to 30% of noise add yielded acceptable results.

[8] uses the behavioural patterns of the users to build a profile of the user. Since the system has a feedback option, the classifier adjusts its ratings based on the stream of

transactions. The profile also keeps changing dynamically based on the new behaviours by the user. In the experiment, they improved the accuracy of two basic models to more than 80%.

Artificial Neural Networks are used by [9] to detect and prevent fraud transactions. This model helps in detecting frauds in real time. They use a combination of metrics to evaluate the model such as Value Detection Rate and True False Positive Rate.

Another method which can be successful in FDS is the bagging ensemble classifier. [10] explores this method with positive results. They use decision tree algorithms to make the model, because it is independent of various features values. They have also done a little bit of work in handling the class imbalance. They tested the models with different number of frauds cases ranging from 3% - 20%. The model got the same results across different test cases.

The major drawback of all these papers is that they have not considered the real-world scenarios as it is. Only a few have them have talked about class imbalance but not realistically. The realistic value of class imbalance is 0.1%. The least which any paper has considered is close to 3%. [11] has taken the class imbalance to less than 1%. The strategy they have used is oversampling to even the classes. For the detection algorithm, they have used K-Means algorithm. They have not implemented the model; hence they do not have any metrics to show the performance.

[12] has done an evaluation of various methods of oversampling to tackle class imbalance. They have testes different types of SMOTE namely ENN, SAFE, ROS, and EL. They have then used boosting methods to test the performance. They have taken an array of metrics like sensitivity, precision, and Area under ROC to evaluate the models. This paper however has not checked if the behaviour of the users changes or the performance of Neural Networks in this scenario.

Chapter 3

Model

3.1 Dimensionality Reduction

3.1.1 Principal Component Analysis (PCA) Transformation

In machine learning, we often deal with large amount of data to increase the accuracy and determine the right features of the model. Determining the right feature of the model is the crucial part of the machine model to make good judgment of the input data. If we get a method to reduce the data we need to process without having any trade-off in the quality of the data, it gives us the opportunity to process the model with limited amount of resources (Computation/Time). Principal Component Analysis (PCA) gives us the opportunity to achieve this. PCA is a dimensionality reduction technique which can be used to decrease the dimensionality of the data without compromising the information the data holds. The accuracy of the data would be reduced little bit but it might not be of significant value.

Accuracy is reduced a little bit, but it is a trade-off for simplicity. This is used because smaller datasets are easier to work with, faster, and the visualisation is more viable.

The steps for PCA transformation are:

1. Standardization

   This technique makes the array of the continuous variables so that each of them plays an equal role in the analysis of the data. To put this into perspective, a variable with values between 1 and 50 will rule over the variable with ranges between 0 and 1. This can lead to influence the results. In terms of mathematics, this can be done by dividing the difference of the value and mean by the standard deviation. This is done for all the rows across each column.

$$z = \frac{value - mean}{StandardDeviation} \tag{3.1}$$

2. Correlation Matrix

   The goal of this matrix is to understand the relations between the different features. Sometimes due to high correlation, there is redundant information in the dataset. This can be used to remove that extra information. The matrix has the dimensions of N x N where N is the number of features in the dataset. For a 2 x 2 dataset, the matrix can be visualised as

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) \\ Cov(y,x) & Cov(y,y) \end{bmatrix} \tag{3.2}$$

   Where cov() is the covariance between the two variables. Covariance of a variable with itself is the variance of the variable. Hence the diagonal values are the variance of the variable. Covariance is also commutative, because of this the upper and the lower part of the matrix are the same. The covariance of the sign also matters. If it is positive then the two variables are highly positively correlated and highly negatively correlated in case of a negative sign. In other words, positive correlation signals direct proportion and negative correlation signals inverse proportion.

3. Eigenvalues and Eigenvectors to identify principal components

   Principal components are the new features which are obtained as the result of linear model of the initial variables. The principal components are uncorrelated, and the information is compressed in the first features. The principal components are not as interpretable as the initial features. Eigenvectors are the direction of the line which can hold the most information and eigenvalues are the constants given to the vector denoting the volume of information held.

4. Feature Vector

   When the eigenvectors are arranged in ascending order, we can decide whether to maintain all of them or shed some of them. The remaining vectors are used to form the feature vector.

5. Reorganize the principal components

   Since we must create the new axes, we can now reshape the original axes to the ones denoted by the principal axes. This is done by matrix multiplication of the transpose of the feature vector and the original dataset.

$$FinalDataSet = FeatureMatrix^T * StandardizedOriginalDataset^T \tag{3.3}$$

3.1.2 t-Distributed Stochastic Neighbor Embedding (tSNE)

t-SNE is a non-linear technique that is mainly used for data pre-processing. It plots data from many dimensions onto two dimensions that is easy for human readability. This algorithm calculates a measure between two data points in multi-dimensional space and plots them in two-dimensional space. There are three main steps for this algorithm:

1. For each point in space, a gaussian distribution is calculated for that point. The concentration of all those points are measured. Once these points are normalized, it gives us a set of probabilities for all these points.

2. This step is also quite like the first one, except that the distribution being used is known as Cauchy distribution. This step gives us another set of probabilities.

3. The last step calculates the differences between the two probability distributions using Kullback-Liebler divergence. These differences are minimized using gradient descent.

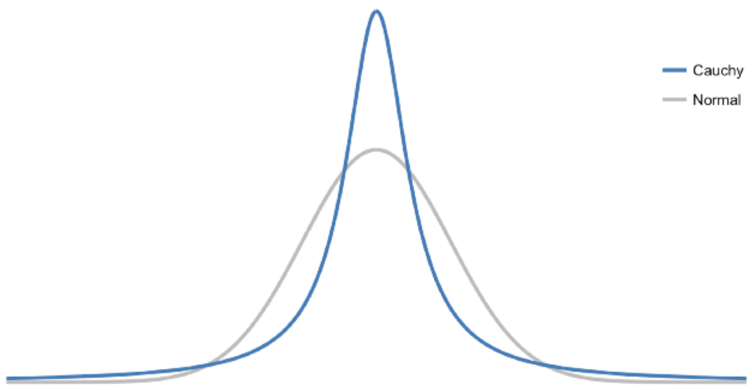

Figure 3.1: Normal vs Cauchy Distribution

3.2 Classification Algorithms

3.2.1 Logistic Regression

At the core of the logistic regression is the logistic function. This function is also called as the sigmoid function. It is shown in Figure 3.2. The x axis is denoted by z and the y-axis is $\phi(z)$. It is expressed mathematically as

$$\phi(z) = \frac{1}{1+e^{-z}} \tag{3.4}$$

$$\phi(z) = \frac{1}{1+e^{-(b_0+b_1x)}} \tag{3.5}$$

Equation 3.5 is the logistic regression equation. The input values are combined in a linear manner using weights. The output is shaped to be 0 or 1 rather than a continuous value. The output depends on the coefficients that are based on the train data. Here b_0 is the intercept and b_1 is the coefficient for the input value. The different types of Logistic Regression are:

1. Binary Logistic Regression: This type of unmitigated response has only two potential outcomes

   (a) Spam

   (b) Not Spam

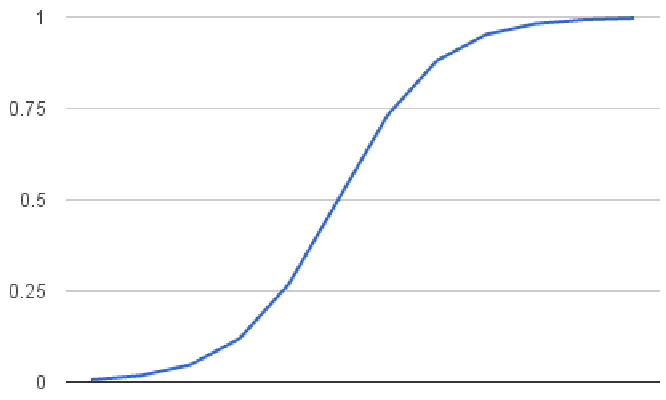

Figure 3.2: Sigmoid Function

2. Multinomial Logistic Regression: This category of logistic regression allows three or more outcomes without ordering. e.g. Food(veg, non-veg, vegan), metal(steel, aluminium, copper, titanium)

3. Ordinal Logistic Regression: This category of logistic regression allows three or more outcomes with ordering. e.g. Hotel, Movie, Customer service ratings from 1 - 5

3.2.2 K Nearest Neighbours (kNN)

K Nearest Neighbours is one classifier technique used in supervised learning. This technique can be applied to analytical problems that require deciding. In the case of the K Nearest Neighbours, K is a positive integer, typically of small value. An object will be classified by a majority vote of its neighbours, with the object being assigned to the class most common among its k nearest neighbours.

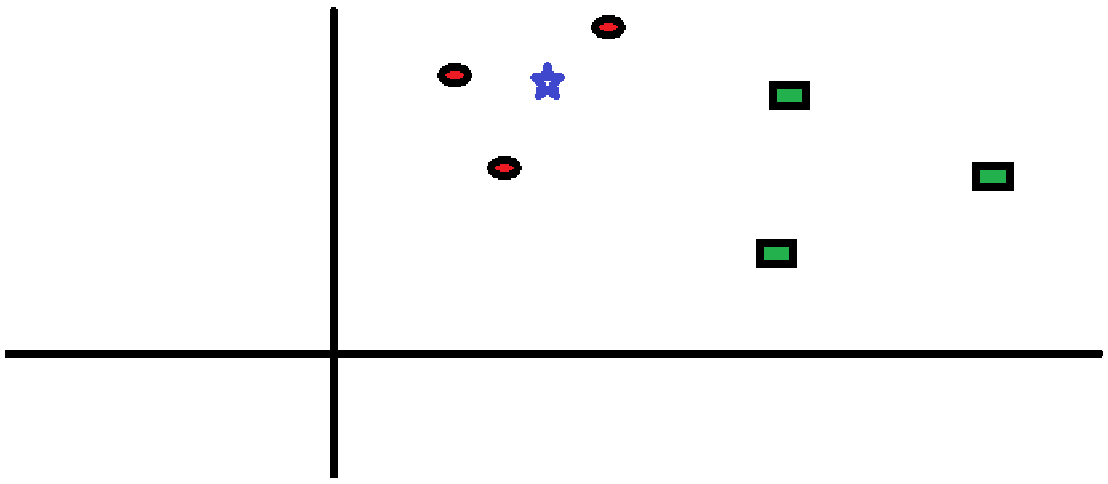
Figure 3.3: Cluster Identification

In the figure 3.3, we want to find the characteristics of the blue star. The blue star can be either red circle or green square. If we take k=3, taking the blue star as the centre, draw a circle that encompasses 3 data points.

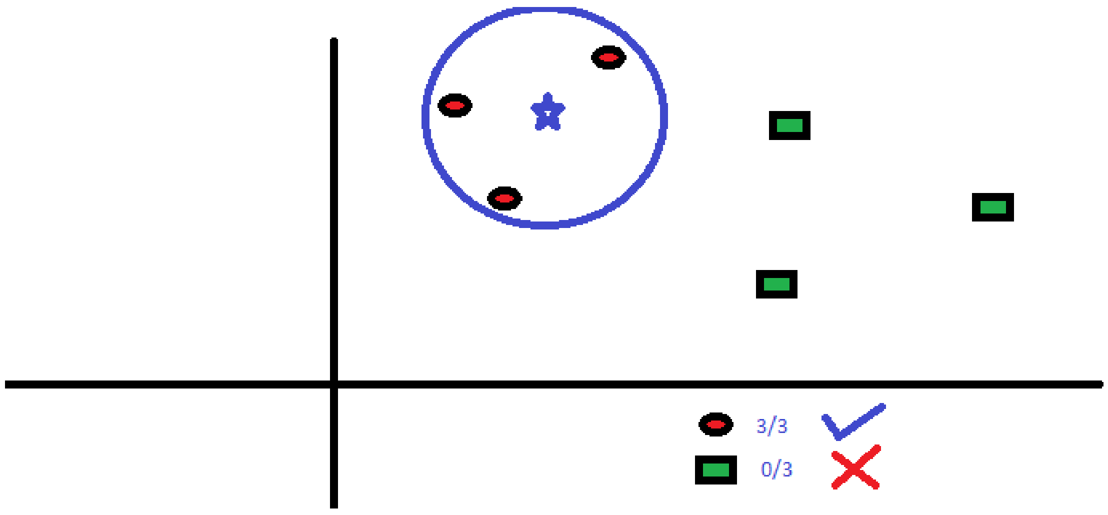

Figure 3.4: Cluster Identification

The closest 3 points to the blue star are all red circles. With this information, we can conclude that the blue star is also a red circle. This is shown in figure 3.4.Choosing the right k is very important in this method. Consider that all the 6 training observations remain constant, changing the k value by specific values will establish boundaries for the classes. The result will segregate red circles from green squares. For example, establishing a value of 1 for K can establish the following boundary:

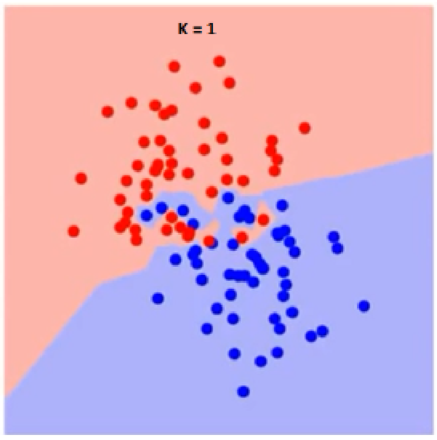

Figure 3.5: Cluster Segregation for K=1

whereas establishing a value of 5 for K can establish the following boundary:

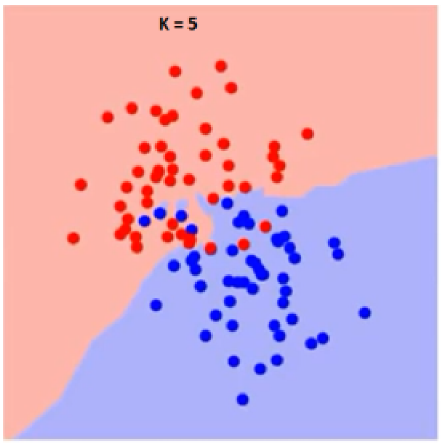

Figure 3.6: Cluster Segregation for K=5

It is interesting to note that with increasing values of K, the boundary becomes smoother. Lets take the example of the value 7 for K:

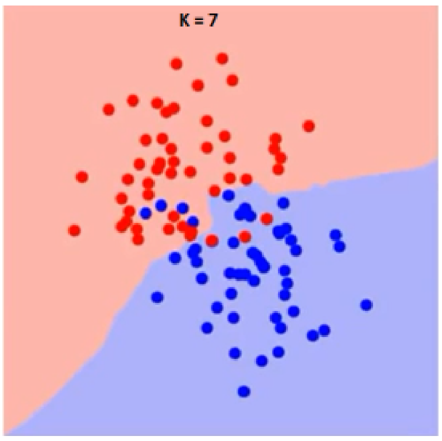

Figure 3.7: Cluster Segregation for K=7

As K increases to infinity, the sample becomes all blue or red depending on the total majority. The two important parameters that we need to weigh in for different K values are:

1. The training error rate
2. The validation error rate

The following is the curve for the training error rate:

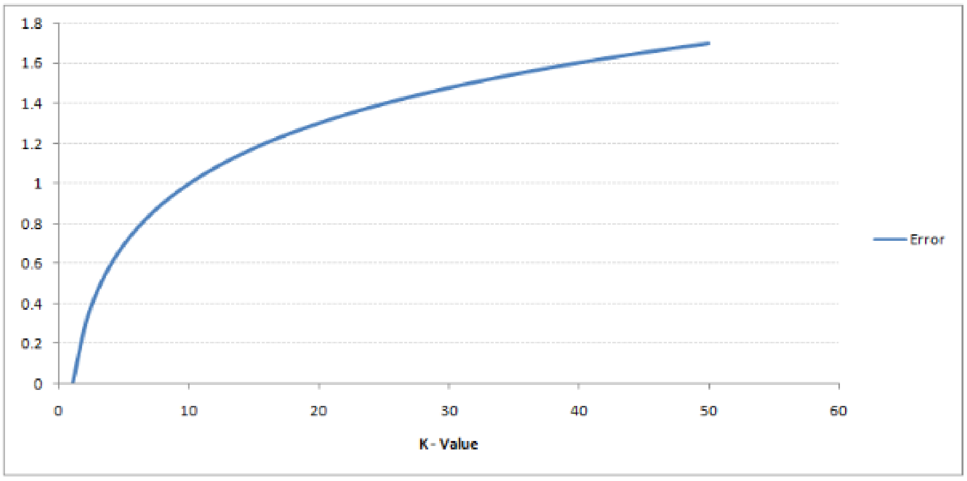

Figure 3.8: Training Error Rate for kNN

Since the closest point to a training data point is itself, at K = 1, error rate is zero. Hence the prediction is always accurate at K = 1.

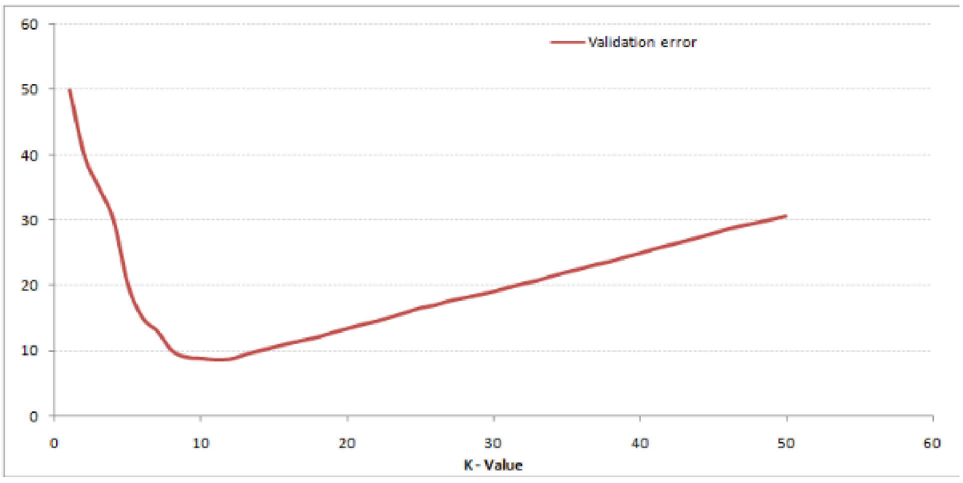

Figure 3.9: Validation Error Curve for kNN

The curve above is the curve for the validation error curve. When we overfit the boundaries at K = 1, the validation error rate is at 47. As the K value increases to 10, the validation error rate reaches a minimum and after reaching the minima, the validation error rate gradually increases along with the value of K.

To reach at an optimal rate for K, the training and validation must be separated from the initial dataset. After separating, it is worthwhile to plot the validation error curve to arrive at the optimal value for K. This value of K must be used for all predictions. We can choose neighbours next according to:
Brute Force: If we consider a basic 2-dimensional plot as below:

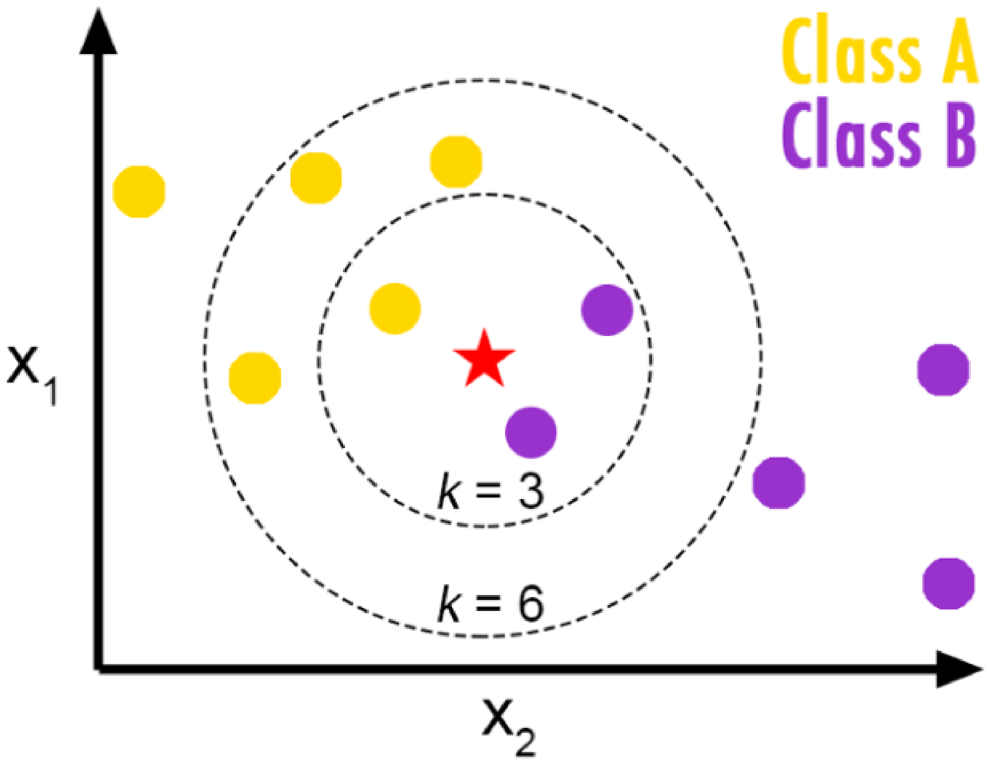

Figure 3.10: Variance of Clusters with respect to K

Method of selection: Calculate the Euclidean distance from point of interest (of whose class we need to determine) to all points in the training data set. Then choose the class with majority points. This method is called brute force method. This method can be applied when there are a smaller number of dimensions and training set. For N samples in D dimensions the running time complexity will be $O[DN^2]$. This method can result in increased running times along with the increase in the size of data sets. Brute Force performs worst when there are larger dimensions and large training sets.

To improve running time, several methods were invented to build a growing tree from a point of interest. These methods were successful in reducing the required number of distance calculations by encoding aggregate distance for the sample. By avoiding the need to explicitly calculate distance information in certain scenarios (example: If point D and point E are very far from each other and point F is very close to point E, then point D and point F are very far from each other). The computational cost of a nearest neighbour search can be reduced to $O[DN * log(N)]$, marking a significant improvement over brute force for cases of large values of N. The K-D Tree method can be very useful in cases where D $<$ 20. The K-D Tree method can result in longer running times for cases where D $>$ 20. This case is called curse of dimensionality.

Ball Tree method creates the nested hyper spheres with a query complexity of $O[Dlog(N)]$ Except for brute force, K Nearest Neighbours implements a tree like data structure based on the distance from point of interest to other points in the sample. Based on sparsity of data, number of requested neighbours and the number of dimensions/numbers of features the right algorithm can be chosen.

3.2.3 Support Vector Classification

The objective is to find the hyperplane in an N(number of features)-dimensional space which classifies the data points into different clusters. This method is known as Support Vector Classification algorithm is used.

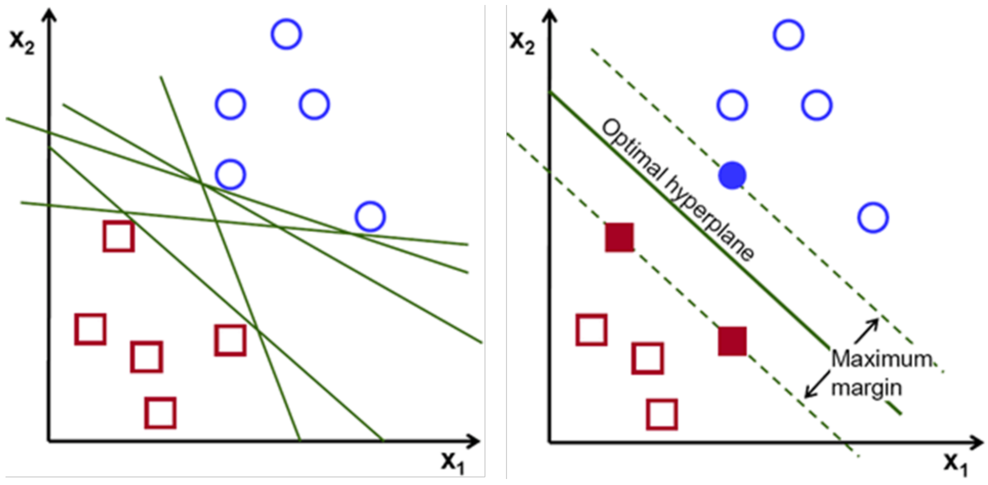

Figure 3.11: Choosing of the hyperplane

There are many possible hyperplanes with the maximum margin that could be chosen to separate the two classes of data points. With this hyperplanes, the main objective is to find the plane with the maximum distance between the two clusters of classes i.e maximum margin.

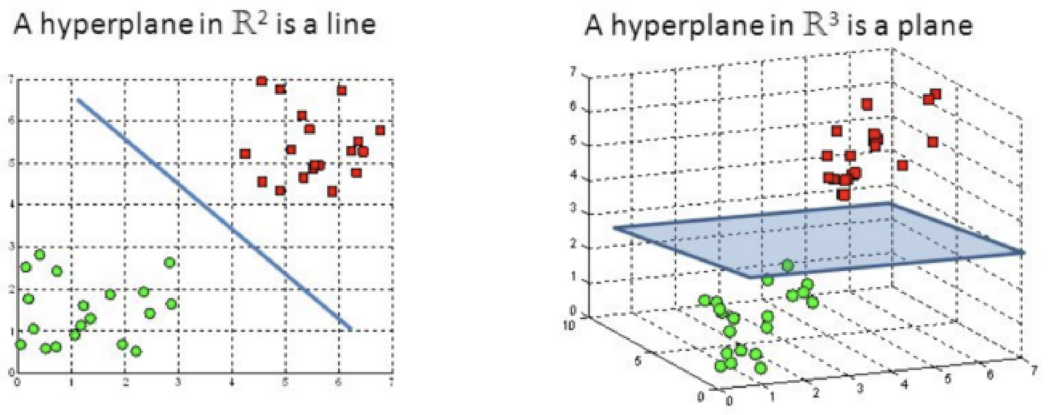

Figure 3.12: Hyperplane in different dimensions

The hyperplanes are called decision boundaries which helps in classifying the data point. The data points and the dimension of the hyperplane depend on the number of features. For example, if the number of features is 2, then the hyperplane is just a straight line and if the number of features is 3, then it is a two-dimensional plane.

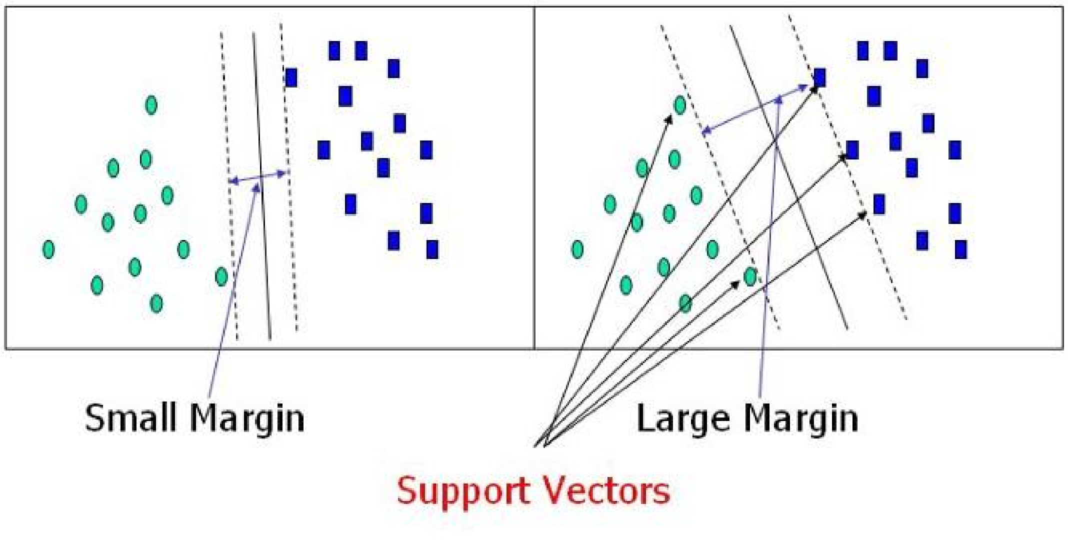

Figure 3.13: Ambiguities in Hyperplane Position

The data points which lies closer to the hyperplane and influence the position and alignment of the hyperplane are support vectors and which helps in maximising the margin of the classifier. By deleting the support vectors, it will change the position and alignment of the hyperplane.

3.2.4 Decision Tree Classification

Decision Tree Classification is a supervised learning model that is used in predicting a dependent variable based on a series of training variables. The flowchart-like tree structure is comprised of internal nodes - features or attributes, a branch that denotes a decision rule and each leaf node will represent the outcome of the features and decision rules.

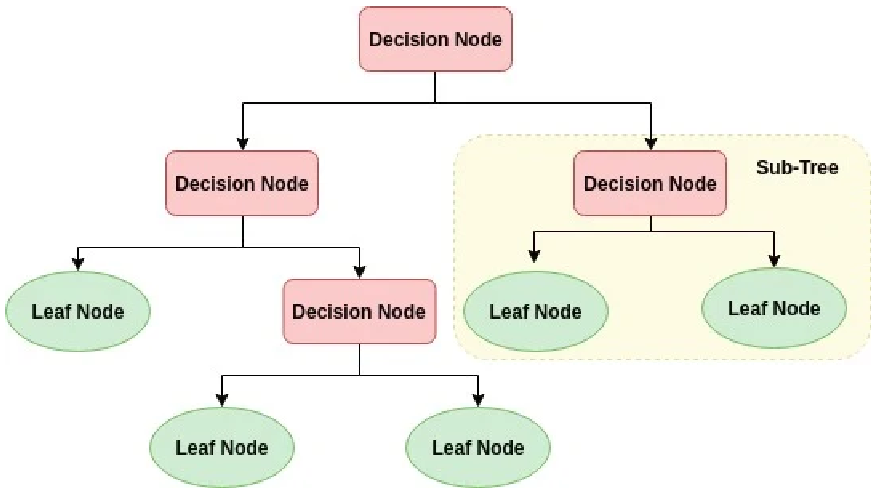

Figure 3.14: Decision Tree Example

The decision tree is known as a white box type of ML algorithm since it shares the internal decision-making logic which is not the case in the black box type of algorithms such as Neural Networks. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. It is worth noting that decision tree classifications are distribution-free and a non-parametric method since it does not depend upon probability distribution assumptions.

Decision Trees can be used when accuracy is needed and when predictions need to be made on high dimensional data. The best attribute can be selected using the

Attribute Selection Measures (ASM) - a heuristic to select the splitting criterion that will help partition data. ASM can provide a rank to each feature by exploring and explaining the sample set. The splitting attribute will then be accordingly applied. In a more complex scenario, the continuous-valued attribute will be used to split points for branches. Information Gain, Gain Ratio and Gini Index are often touted as the most popular selection measures. Entropy refers to impurity in a group of examples in Information Theory. Simplistically, information gain is the decrease in entropy. The difference between entropy before and after split is calculated based on given attribute values.

$$Info(D) = -\sum_{i=1}^{m} P_i * log_2 * P_i \tag{3.6}$$

In this equation, P_i is the probability that an arbitrary tuple in D belongs to class Ci

$$Info_A(D) = -\sum_{j=1}^{V} \frac{|D_j|}{D} * Info(D_j)$$

$$Gain(A) = Info(D) - Info_A(D)$$

where
Info(D) - average amount of information needed to identify the class info of tuple D
InfoA(D) - Expected information to classify tuple from D based on partitioning by A
Best Attribute- Attribute A with highest information gain
Gain(A) - chosen as splitting attribute for Node N()
Gain Ratio is used to improve upon useless partitioning created for an information gain with an attribute with a smaller number of distinct values. Gain ratio handles the issue of bias by normalizing the information gain using Split Info.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2 \frac{|D_j|}{|D|} \tag{3.7}$$

where
$\frac{|D_j|}{|D|}$ - weight of the j_{th} parition
v - number of discrete values in attribute A

Attribute with the highest gain ratio will be chosen as the splitting attribute.The Gain Index of attribute A partitions data D into D1 and D2 and then the subset that gives the minimum Gini index will be chosen as a splitting attribute in the case of a discrete-valued attribute. In case of a continuous-valued attribute, each pair of adjacent values is selected as a possible split-point and point with smaller Gini index will be chosen as best attribute. Upon selecting the best attribute using one of the above-mentioned methods, the attribute is then the decided node that breaks down the dataset into smaller subsets.

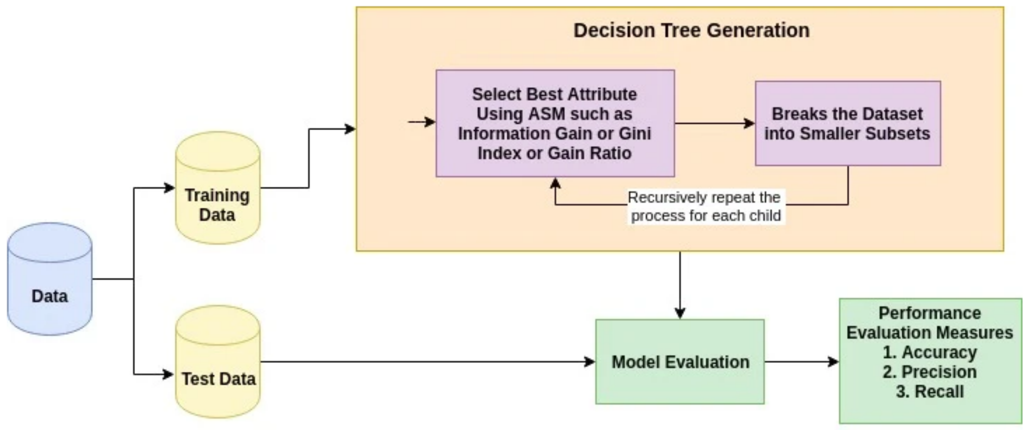
Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Figure 3.15: Generation of Decision Tree

At this point, the tree building can start by repeating the above-mentioned process recursively for each child until one of the below listed conditions match:

- All the tuples belong to the same feature value
- No more attributes are pending
- No more instances pending

Decision Tree Classifications can be used in cases where attribute selection needs to be easy to interpret and visualize and in cases where user data pre-processing is minimal. It can also be used where missing values in datasets need to be predicted. It is advantageous to use decision trees where no assumptions can be made about the distribution. Decision Tree Classifications fall short when noisy data need to be carefully maneuvered. The tricky nature of small variation (or variance) in data can create an altogether different decision tree. This drawback can be mitigated by bagging and boosting algorithms. Decision Tree Classifications can also be biased with imbalance dataset, this drawback can be mitigated by balancing out the dataset before starting the tree.

3.3 Sampling Techniques

3.3.1 Near Miss Technique

Near Miss is an under-sampling technique that can be used for imbalanced datasets. An imbalanced dataset can be defined as one where at least one of the classes constitute only a very small minority. When a model is tried and applied on an imbalanced dataset, the predictions can drastically vary. When we apply the NearMiss to an imbalanced dataset, the majority class will be reduced to the total number of minority class, hence the two classes will have an equal number of records.

To understand the basic algorithm behind this method, first, the method calculates the distances between all instances of the majority class and the instances of the minority class. Once distances are known, then the k instances of majority class having the smallest distances to the k instances in the minority class are selected.

If there are n instances in minority class, the nearest method will be k*n instances of majority class. There are three different interpretations of the NearMiss method. The NearMiss1 will select the samples from the majority class whose distances to the three closest instances of the minority class are the smallest. The NearMiss2 uses three farthest samples of the minority class. The NearMiss3 selects a definite number of closest samples of majority class for each sample from the minority class.

3.3.2 Random Undersampling

Undersampling is one of the strategies used to handle imbalanced data. In Random undersampling the majority class will be under sampled to match the minority class. The training set documents are eliminated randomly based on the desired rate of majority and minority class. In theory, the problem by using random undersampling is about the information loss. There is no control over the information thrown in majority class and the vital information about there is an elimination of the entire resolution boundary between the majority and minority. Regardless of its conjure, random undersampling has been one of the effectual sampling techniques among sampling methods experimentally. In many experimental studies, there are few undersampling methods which result in surpassed random undersampling.

3.3.3 Synthetic Minority Over-Sampling Technique (SMOTE)

Synthetic Minority Over-sampling Technique (SMOTE) can be used to solve the problem when one class in the training set dominates the other class in the training set. An imbalanced dataset can be defined as one where at least one of the classes constitute only a very small minority. SMOTE is an over-sampling method that can be used for imbalanced datasets. When a model is tried and applied on an imbalanced dataset, the predictions can drastically vary. When we apply SMOTE to an imbalanced dataset, the minority class will be increased to equal the total number of majority class.

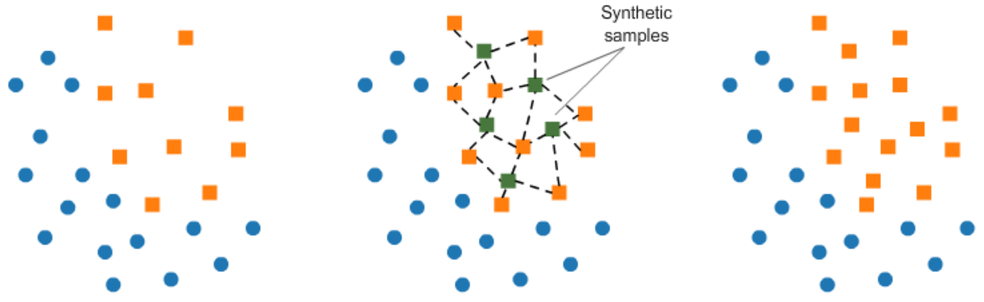

Figure 3.16: SMOTE Over Sampling

SMOTE synthesises new minority instances between existing minority instances. After establishing a set of lines along the minority instances, SMOTE then imagines new, synthetic minority instances somewhere on those lines. After synthesising new minority instances, the imbalance between the minority and majority classes shrinks and the classes eventually become comparable for predictions. The method for determining the new synthetic minority instances is based on nearest neighbours judged by the Euclidean distance between data points. For each minority instance, k number of nearest neighbours are created such that they also belong to the same class. Following is the equation applied while creating the new nearest neighbours.

$$k = \frac{SMOTe\%}{100} \tag{3.8}$$

k number of difference vectors are created to match the difference between the feature vector of the considered instance and the feature vectors of k nearest neighbours. After the k number of difference vectors are created, each number is multiplied by a random number (between 0 and 1). After being multiplied, the resulting difference vectors are added to feature vector of considered instance (original minority instance) at every iteration. All the k number of nearest neighbours will belong to the same class. Depending on how much oversampling is desired, more than one of the k nearest neighbours can be used to create new observations.

The main advantage of using the SMOTE sampling method over other traditional methods is the creation of synthetic observations instead of reusing existing observations, thus resulting in a classifier that is less likely to overfit. The only point of caution while using the SMOTE method is to ensure that the observations created as a result are realistic. The SMOTE over-sampling method is effective only when the synthetic observations created as a result are realistic and can potentially be observed. When up sampling using SMOTE, the problem of duplicate observations gets mitigated.

There are different ideologies regarding the right way to oversample data. By oversampling before splitting into training and validation datasets, there is a possibility of information bleeding from the validation set into the training of the model. However, when oversampling is performed only on the training data, none of the information in the validation data will be used to create synthetic observations. Hence, the results are generalizable, which makes this ideology accurate for deploying a model in production. When the model predicts on unseen data, it is important to remember that incorrect oversampling can lead to thinking that a model can generalize better than it does.

3.4 Neural Networks

Human beings have the incredible ability to learn new tasks just by considering examples, without being specifically programmed for that task. In the human brain, this kind of learning is brought about by communication between millions of neurons that are brain cells capable of instant communication with each other through chemical messengers called neurotransmitters. These neurotransmitters can either stimulate electrical activity of the brain cell or calm this activity. The activity of a brain is largely determined by the balance of these stimulations or calming mechanisms.

An artificial neural network (ANN) is a model inspired by the functioning of neurons in the human brain where each neuron is a mathematical function that collects and classifies information according to a specific architecture. In artificial neural networks, neurons are arranged in layers and in each layer, instead of an electrical impulse a numeric value is passed from one neuron to another. There are typically three kinds of layers in a neural network: an input layer, with each neuron representing the predictor variables plus a constant; one or more hidden layers; and an output layer, with a neuron or neurons representing the dependent variable(s). The units are linked with varying weights that define the strength of the connection. The Input data are given as input to the first layer, and values are carried forward from each neuron to every neuron in the next layer until the result is obtained in the final layer. Figure 3.18 represents the structure of an ANN highlighting the different layers in the network.

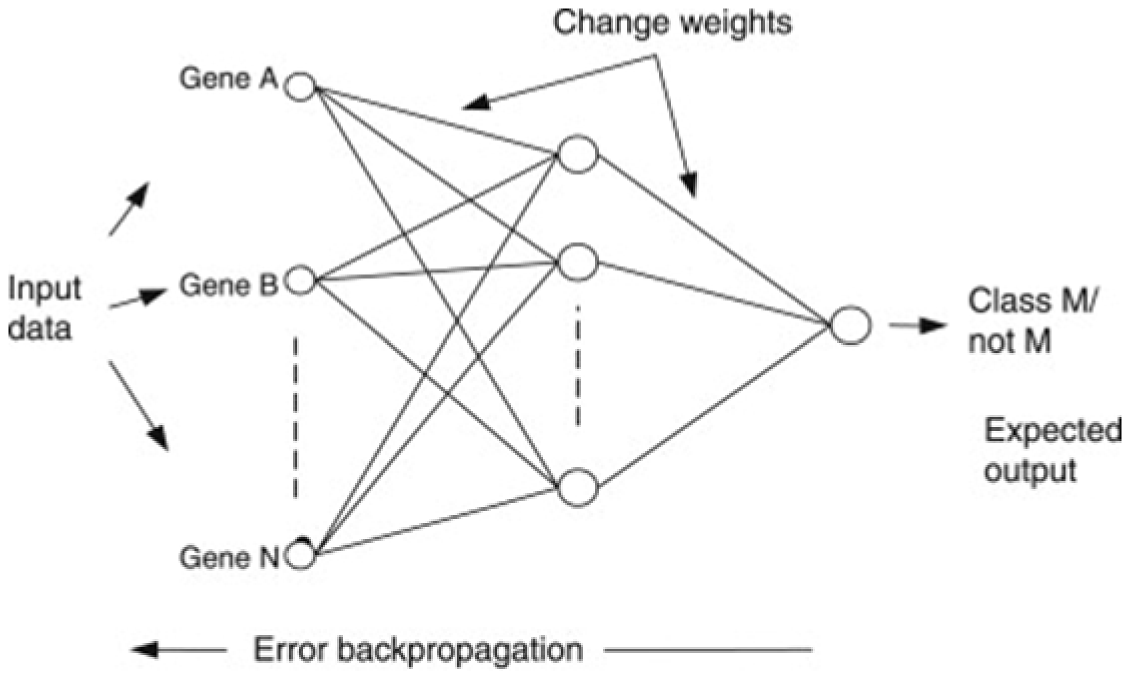

Figure 3.17: Data Flow in a Neural Network

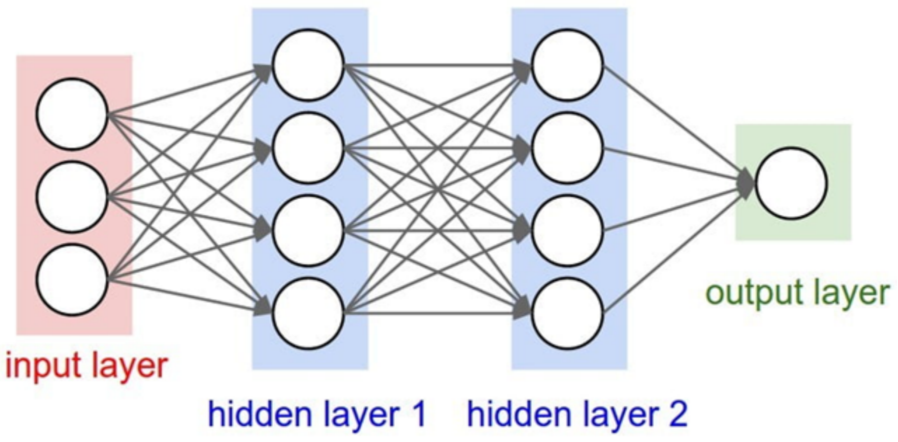

Figure 3.18: Architecture of a Neural Network

The neural network learns by generating a prediction for each set of inputs and modifying the weights in each node until the correct prediction is made. This process is repeated on an iterative basis until an acceptable level of model performance is met. Each node in the network is called a perceptron and learns by taking inputs and applying weights to each of those inputs followed by an activation function and passes the output to the next node. Figure 3.19 represents the process of learning for each perceptron.

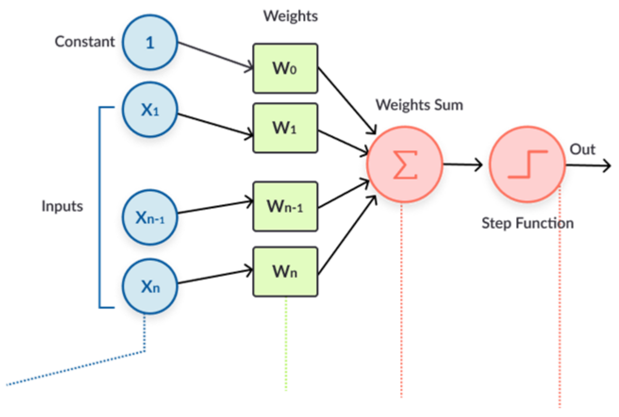

Figure 3.19: Mathematical form of Neural Network

The learning process is as follows:

1. The model takes the input values corresponding to each independent variable from the input layer and feeds into the perceptron in the next layer, multiplies them by their corresponding weights, and calculates the sum.
2. Adds the constant as one and multiplies by a bias weight. This step gives way to move the activation function in different directions on the number graph.
3. Sends the sum through the activation function
4. The result of the activation function is the output.

Initially, all weights are chosen randomly, and thus the output predictions are not accurate. The model learns through training where examples for which the output is known are repeatedly presented to the network, and the predicted output is compared to the known output. This error calculated is then passed back to the network and modifying the weights to try and reduce the error to as low as possible. This technique is known as back propagation and plays a huge part in increasing the model performance. As training continues, the network becomes increasingly accurate in replicating the known outputs. Once trained, the network will be able to predict future cases with unknown outputs.

Some of the parameters that need to be considered when building a neural network are learning rate, type of activation function, and optimizer. Learning rate is a parameter that helps control the rate of change of weights for our network. During back propagation the weights are changed rapidly based on the error calculated. When the change in weight is high, it might lead to higher jumps and miss the global minima. Thus, it is essential to maintain the right learning rate. Activation functions are mostly used to introduce non-linearity in the neural network algorithm. There are a variety of activation functions used such as Sigmoid, Tanh, ReLU, Leaky ReLU etc. ReLU is the most used activation function and is used when the negative values need to be interpreted as zeros. The internal parameters of a model such as weight and bias are key in building a good model. Optimization techniques play a major role in training the model by finding the optimal solution by minimizing loss. The Adaptive Moment Estimation (Adam) optimizer is a method that calculates adaptive learning rates for each parameter. In practice, it has been noted that Adam works well to minimize the loss function and outperforms other techniques.

3.5 Evaluation Metrics

In machine learning, the data is split into testing and training dataset. The test data is used for training the model and this consists of most of the dataset. The machine learning model learns from this dataset provided. The test dataset is used for evaluating the performance of the model built. These are measured using evaluation metrics. Metrics differ based on the machine learning algorithms implied on the dataset. The evaluation metrics for classification models are explained below.

Confusion Matrix

A confusion matrix is a table that displays combinations of predicted and actual values. It is a simple, effective model that is used to evaluate the efficiency of the classification model. The metric also helps in computing other evaluation metrics such as recall, accuracy, prediction., etc. The table contains a N x N matrix where N denotes the class labels used in the classification model. For example, binary classification holds two class labels, therefore the confusion matrix here is 2 x 2 matrix. The terms used in the matrix are true positives, true negatives, false positives, and false negatives.

True positive: The case holds true when the predicted value is positive, and the actual value was positive.

True negative: The case holds true when the predicted value is negative, and the actual value was negative.

False positive: The case holds true when the predicted value is positive, and the actual value was negative.

False negative: The case holds true when the predicted value is negative, and the actual value was positive.

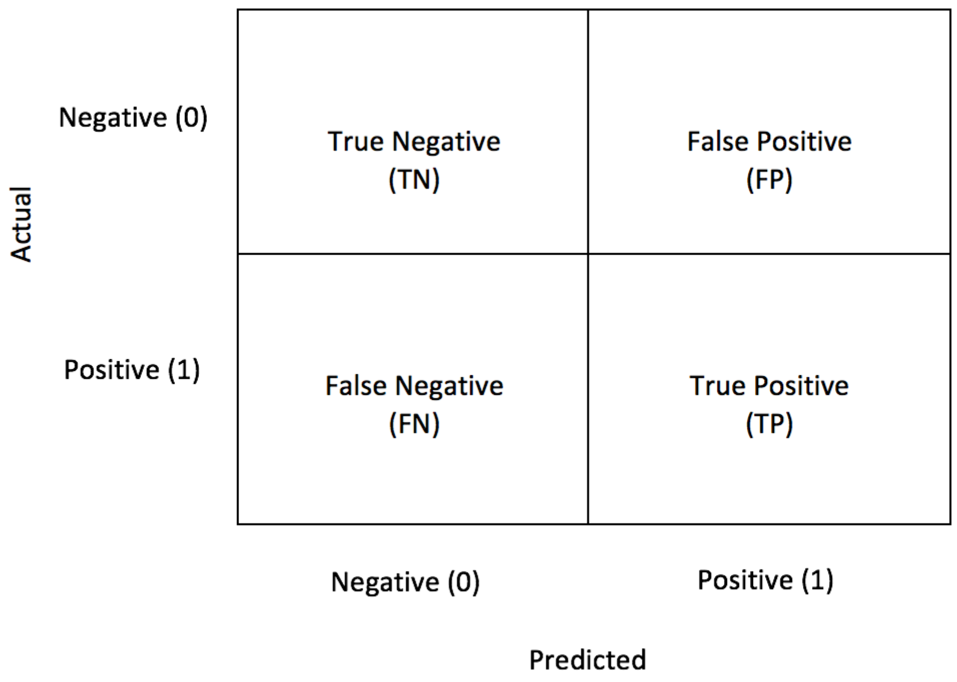

Figure 3.20: Confusion Matrix Structure

Recall

Recall in other words is called Sensitivity, which is the ratio of correct positive predictions made from the total number of positive predictions. The fraction is shown in the equation 3.9

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{3.9}$$

Precision

Precision is the ratio of correct positive predictions made from the total number of predictions. The fraction is shown in the equation 3.10

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{3.10}$$

F1 Score

The equation 3.11 explains the F1- score, it is the mean of two metrics, recall, and precision. The output value is between the range of 0 to 1, where 0 denotes worst-case scenario and 1 denotes the best-case scenario.

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.11}$$

Area Under Receiver Operating Characteristic Curve (AUC-ROC Curve)

The AUC-ROC Curve is used to test the performance of the model at various probability thresholds. The default threshold value used for the classification type is 0.5. Sensitivity and Specificity are two terms used in the AUC-ROC Curve.

Specificity

The specificity denotes the true negative rate (FPR). The formula 3.12 explains the specificity equation.

$$Specificity = \frac{TrueNegatives}{TrueNegatives + FalsePositives} \tag{3.12}$$

Chapter 4

Methodology

4.1 Dataset

The dataset has purchases made by people in the continent of Europe. The transactions are made over a period of two days. There are 492 fraud cases out of total 284,807 transactions. The frauds stand at only 0.172% of the total transactions by number. The dataset has only 28 numerical features. All these features have undergone PCA transformation. This is done to protect the data privacy part of the customers. Two columns namely Time and Amount have not undergone the transformation. The time field has the seconds that have passed between each transaction and the first one. The column Class is the response variable that has 1 in case the transaction is fraud else it is 0.

4.2 Data Preparation

The data set had a small average for the transactions amount. The average of all the amounts is 88 euros. This is comparatively smaller when compared to the high value of the transactions across the continent. There are no missing values in the dataset. Hence there is no need to delve into the techniques for filling the missing values.

4.3 Data Exploration

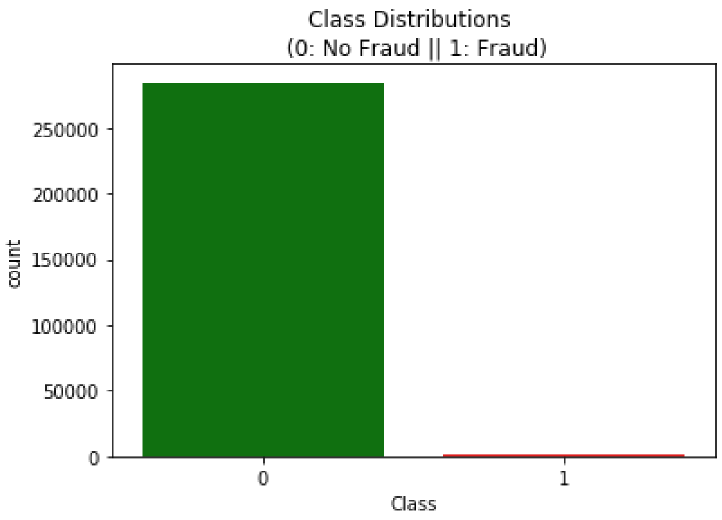

Figure 4.1: Class Imbalance in the dataset

Figure 4.1 shows the class imbalance. If the data frame is used as it is, the classification algorithms will overfit the data as it will work under the assumption that most of the incoming transactions are not fraud.

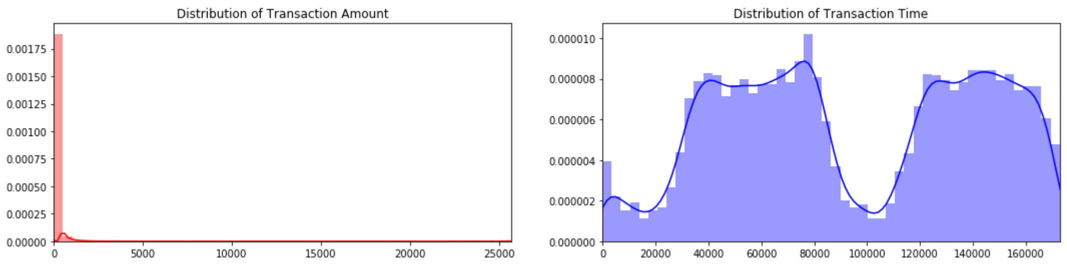

Figure 4.2: Distribution of Amount & Transaction Time

The distribution of the data shown in Figure 4.2 is for the features time and amount. Since these are the only two features that are not scaled, scaling them is very important.

The first step is to create a subsample of the data, since the model needs to understand the patterns of whether the transaction is fraud or not. The ratio of the fraud and the non-fraud cases are made equal so that the algorithm can better understand the fraud cases. The problem with using the data without the transformation is the model will overfit the data and the correlations calculated are also not correct. It becomes important to see the effect of the variables on each other. The subsample of the data is made with choosing the 492 cases of fraud. Another 492 cases of non-fraud are then chosen at random and then concatenated at the end. This dataset of combined cases is the new dataset. The dataset is then split into test and train as the testing of the model must be done on the original test rather than the dataset created by either of these techniques.

The first sampling which is undertaken is the random under sampling, which involves removing of data in order to have a more balanced dataset and avoid overfitting. This is done by removing the non-fraud cases and making them equal to the fraud cases. This dataset is then shuffled to check the performance of the models every time the model is run. The main problem is that the amount of data that is lost since a great deal of data is deleted to make the class balanced.

Correlation Matrices

Correlation Matrices are used to understand the data that is present. This is used to check the amount of impact each variable has on each other. The difference between the two datasets is shown below. The stark difference in the two matrices is the proof that sampling the data has positive effects on the correlations.

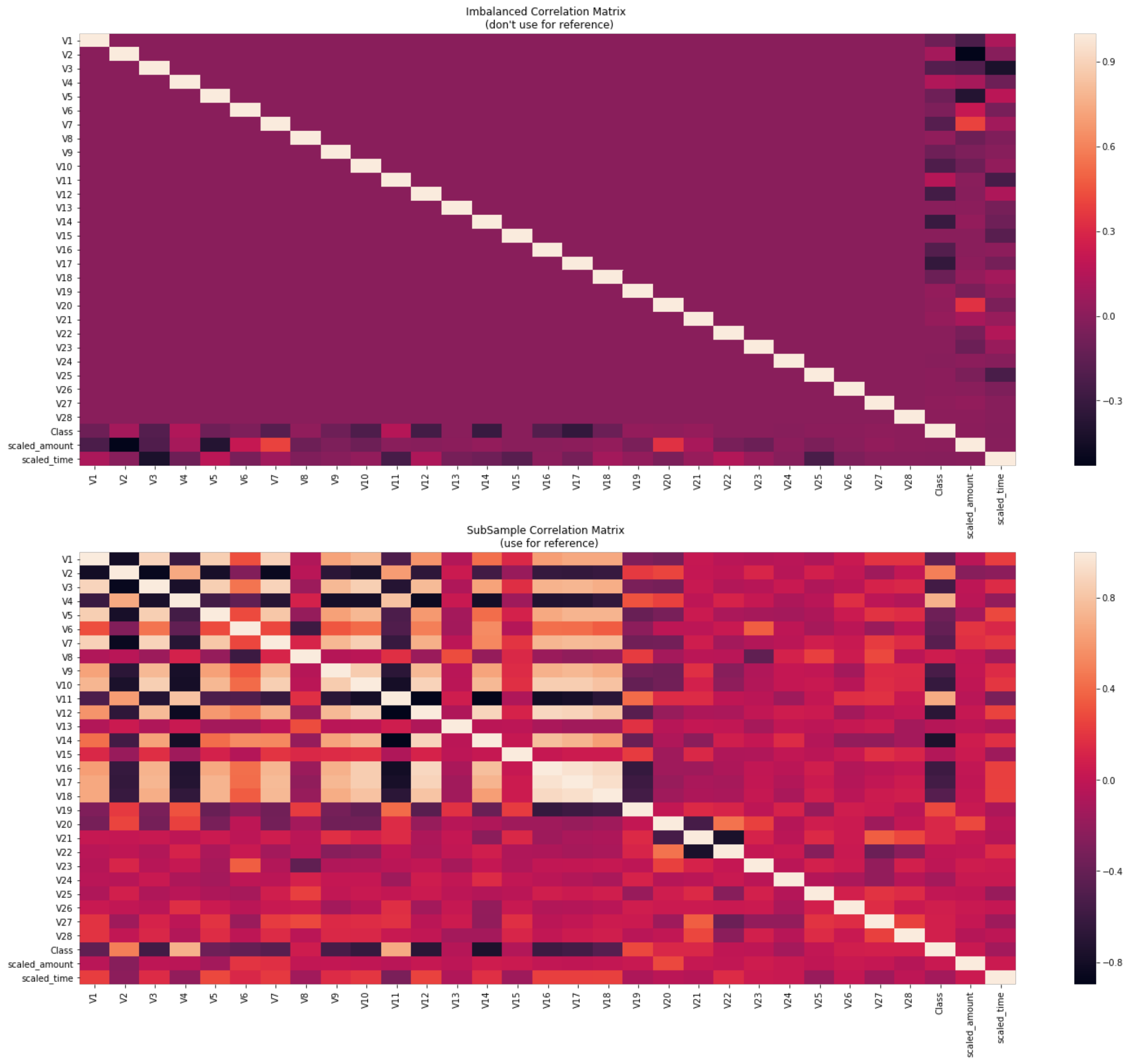

Figure 4.3: Correlation Matrices before and after Under-sampling

There are two main types of correlations: positive and negative. V17, V14, V12 and V10 are on the negative side, which shows that the variables are inversely correlated to the response variable. The lower these values are, the higher are the chances that the transaction is fraud. The variables with positive correlations are V2, V4, V11, and V19. The higher these values are, there is more likelihood that the transaction is fraud.

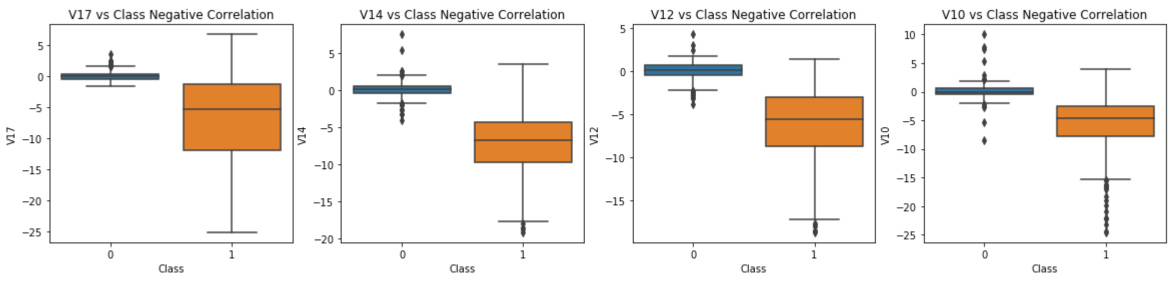

Figure 4.4: Negative Correlation Box Plots

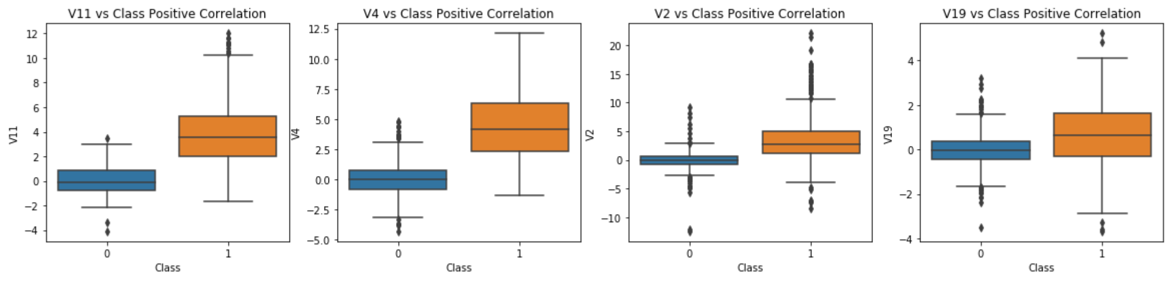

Figure 4.5: Positive Correlation Box Plots

The main agenda of this step is that the extreme outliers are removed so that the accuracy of the model can be improved. This is done with the help of box plots and Interquartile Ranges (IQR). The interquartile range is calculated by taking the difference between the 75 percentile and the 25 percentiles. The objective is to create a threshold that can be used to delete the values which pass this border. This threshold needs to be decided in such a manner that the information loss is not at the cost of the performance of the model.

The distribution of the various variables is plotted to check the match with gaussian distribution

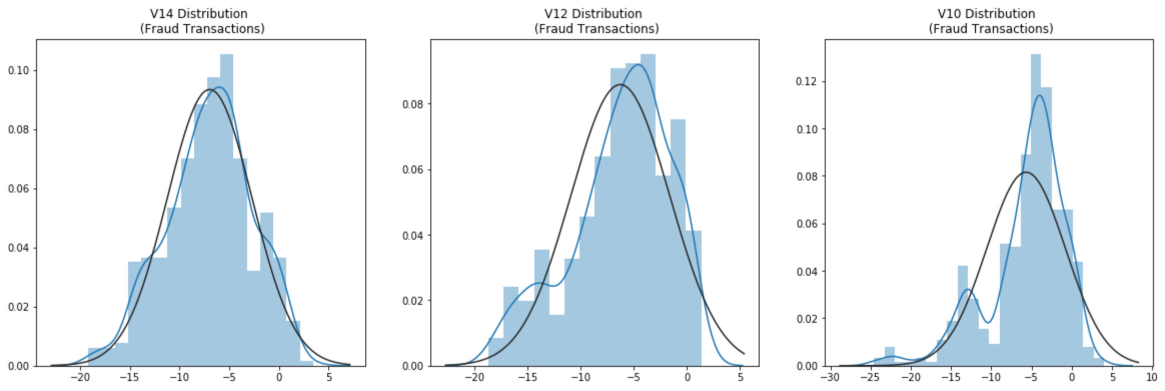

Figure 4.6: Gaussian Distribution of the variables

As evident from the figure 4.6, only V14 has Gaussian distribution. The upper and the lower threshold are then calculated. Once these borders are calculated for each of the features, then the extreme outliers are removed.

Dimensionality Reduction

Two algorithms namely t-SNE, PCA are used to reduce the dimensions of the dataset. Even though the sub sample of the dataset does not have many records, the algorithm is able to distinguish the clusters of fraud and non-fraud cases.

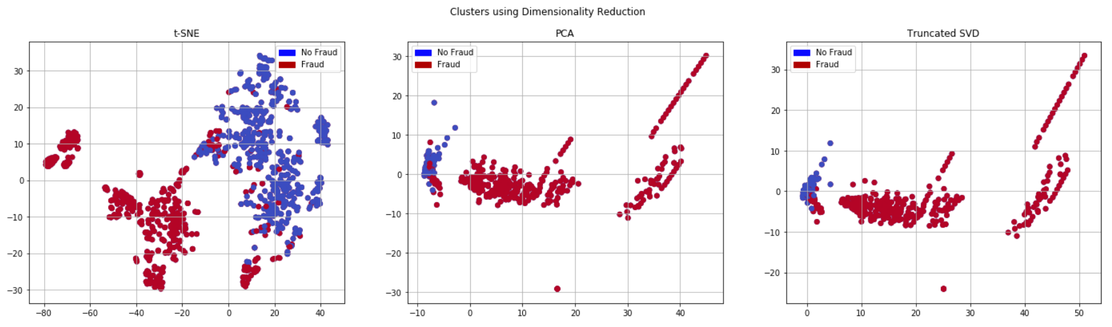

Figure 4.7: Clusters after dimensionality reduction

Data Splitting

For each classifier, we split the dataset into test and train set. The data is split in such a way that the ratio of the data split is 70:30. We then use this dataset for various stages of the experiment like sampling, classification, hyperparameter tuning. A random number is not mentioned so that the data is split in random every time. This helps us to keep in check the performance of the models each time the data is loaded.

Classification Algorithms

The four algorithms that are implemented for the data are Logistic Regression, KNeighborsClassifier, SVC and Decision Tree Classifier. Grid Search is used to determine the best parameters for each of the classifier. Logistic Regression has the highest ROC score, which suggests that it is easily able to distinguish between fraud and non-fraud cases.

The learning curves of the classifiers are also plotted. if there is more gap between the training curve and the cross-validation score, it is more probable that the model is overfitting. if the gap is less between the training and the cross-validation curve, it is more probable that the model is underfitting.

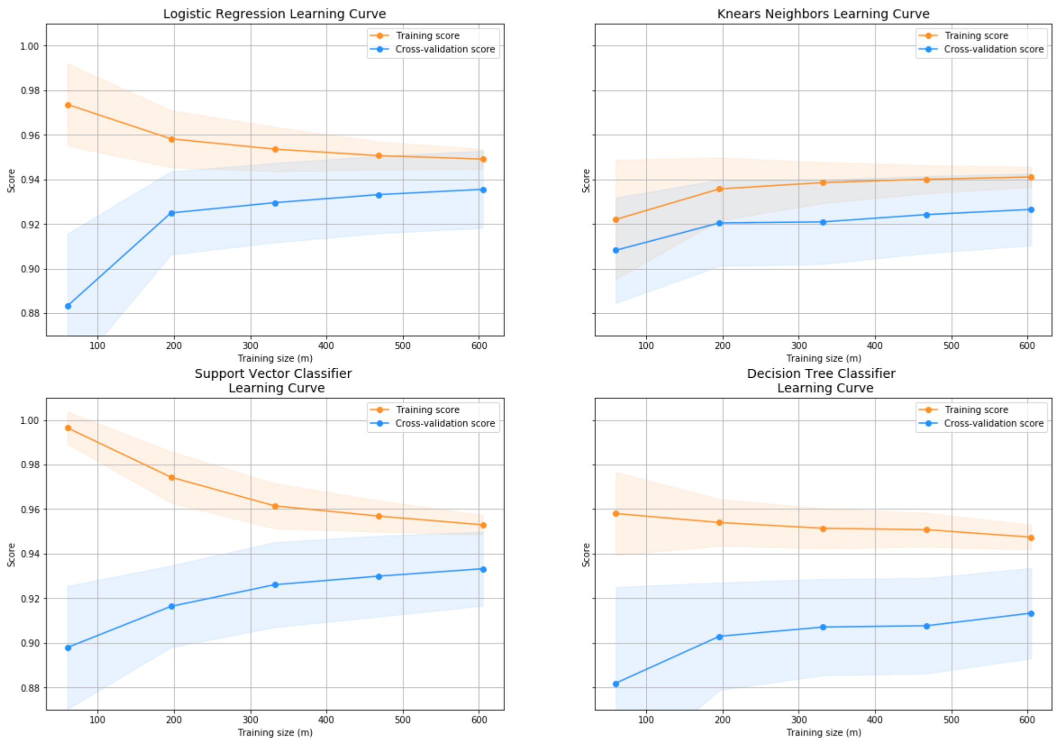

Figure 4.8: Learning Curves of the Classifiers

Neural Networks

The neural network constructed is a simple one. It has one input layer, one hidden layer, and one output layer. The number of nodes in the first layer is equal to the number of features in the dataset. The activation function used in this layer is 'ReLU'. The hidden second layer has 32 nodes and the activation function is also 'ReLU'. The output layer has two nodes and the activation function is 'softmax'. The total number of trainable parameters obtained are 1,988.

The optimizer used is Adam Optimizer as it learns the fastest and there are no sudden drops in the learning rate. The loss function is set to *sparse_categorical_crossentropy* and the metrics used is accuracy. For fitting of the model, the *batch_size* is changed from default 32 to 25. The logging is set to each line by assigning '2' to the parameter verbose. The number of epochs is set to 20 for both undersampling and SMOTE. The training data is also shuffled after each epoch. The training data is split into 80:20 for the validation set. For the undersampling, train dataset had 605 samples and validation dataset had 152 samples. For SMOTE, train dataset had 363923 samples and validation dataset had 90981 samples. The batch size is also increased in SMOTE neural network to 300 as there are more samples that are processed in this method.

Chapter 5

Results

The extreme outlier removal removed several records from the dataset. This led to an increase of accuracy by 3%.
Feature V14 Outliers for Fraud Cases: 4
Feature V12 Outliers for Fraud Cases: 4
Feature V10 Outliers for Fraud Cases: 27
After removing the outliers, the total number of instances stood at 947. This outlier detection was done in the subsample of the data, hence the low number of instances.

t-SNE took around 7 seconds to run, while PCA took 0.032 seconds to run. The results of tSNE are much better as the clusters are less overlapping than the PCA. As we can see that Logistic Regression has the highest accuracy score. To further strengthen the stand of Logistic Regression as the better performing model of the ones chosen, cross validation testing is done. The cross-validation score of Logistic Regression is also the highest amongst the models chosen. The main metric chosen to judge the performance of the models is the ROC curve. The ROC score of the models is shown in Table 5.3. As expected the area under ROC is the highest for Logistic Regression. Based on the three scores, it can be said that Logistic Regression is performing the best for this data.

LogisticRegression	KNeighborsClassifier	SVC	DecisionTreeClassifier
93%	93%	92%	89%

Table 5.1: Training Scores

LogisticRegression	KNeighborsClassifier	SVC	DecisionTreeClassifier
94.05%	92.73%	93.79%	91.41%

Table 5.2: Cross Validation Scores

LogisticRegression	KNeighborsClassifier	SVC	DecisionTreeClassifier
0.9682	0.9223	0.9623	0.9180

Table 5.3: Area under ROC Curve

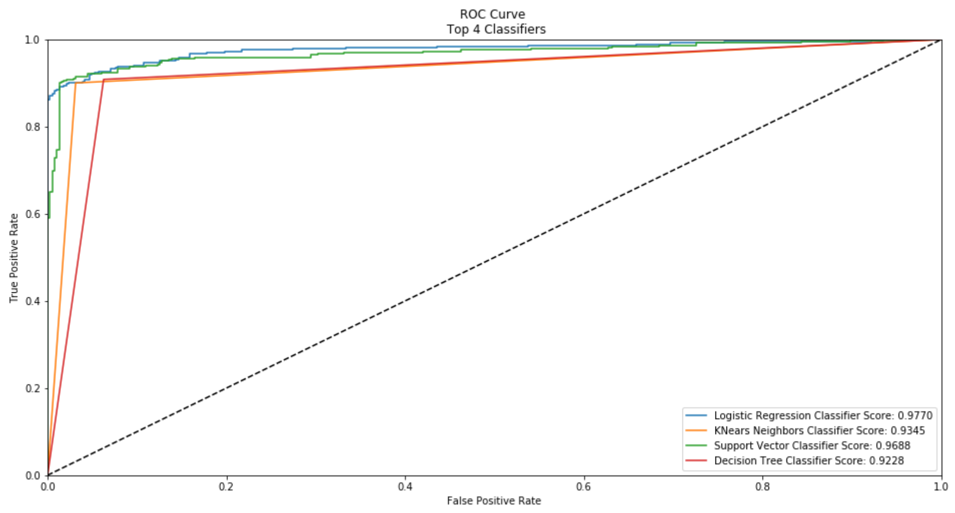

Figure 5.1: ROC Curves for the different classifiers

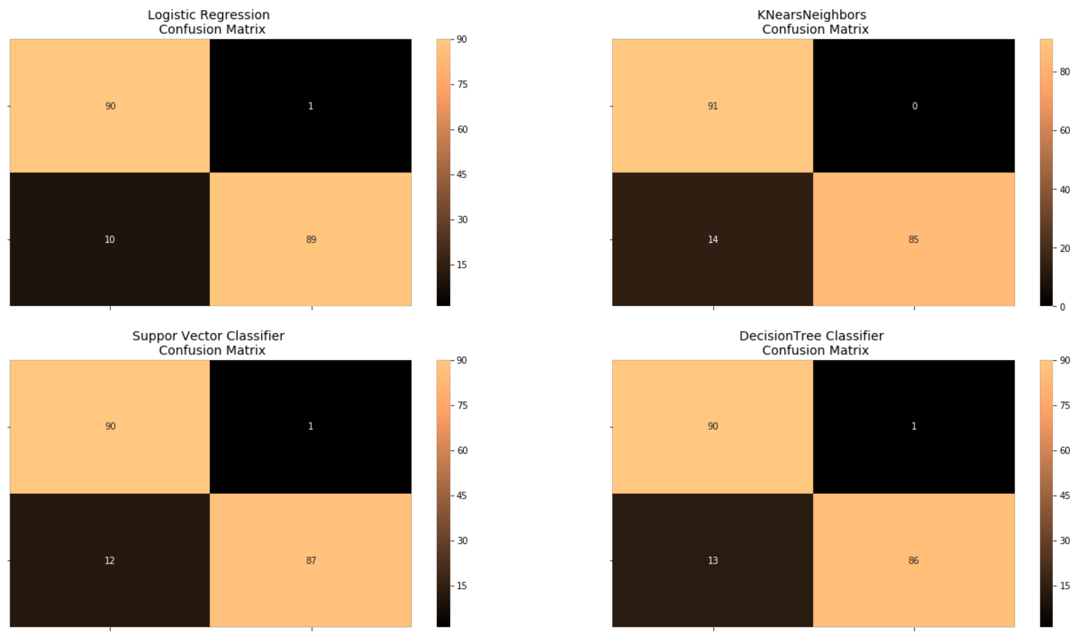

Figure 5.2: Confusion Matrices

The confusion matrices are used for checking the false positive and false negatives. The number of misclassifications in Logistic Regression are the least. The model which performs the worst is the Decision Tree Classifier.

Logistic Regression				
	precision	recall	f1-score	support
Non Fraud	0.9	0.99	0.94	91
Fraud	0.99	0.9	0.94	99
accuracy			0.94	190
macro avg	0.94	0.94	0.94	190
weighted avg	0.95	0.94	0.94	190

Table 5.4: Logistic Regression Metrics

The Table 5.4 shows that Logistic Regression did better with resampling. Knears algorithm does not perform as well in detecting either of the cases. The precision-recall

KNears Neighbors				
	precision	recall	f1-score	support
Non Fraud	0.87	1	0.93	91
Fraud	1	0.86	0.92	99
accuracy			0.93	190
macro avg	0.93	0.93	0.93	190
weighted avg	0.94	0.93	0.93	190

Table 5.5: KNears Neighbors Metrics

Support Vector Classifier				
	precision	recall	f1-score	support
Non Fraud	0.88	0.99	0.93	91
Fraud	0.99	0.88	0.93	99
accuracy			0.93	190
macro avg	0.94	0.93	0.93	190
weighted avg	0.94	0.93	0.93	190

Table 5.6: Support Vector Classifier Metrics

Decision Tree Classifier				
	precision	recall	f1-score	support
Non Fraud	0.87	0.99	0.93	91
Fraud	0.99	0.87	0.92	99
accuracy			0.93	190
macro avg	0.93	0.93	0.93	190
weighted avg	0.93	0.93	0.93	190

Table 5.7: Decision Tree Classifier Metrics

tradeoff was a major factor in the bad performance of the model. This is the same case with the support vector classifier. The score for the support vector is 0.88. The f1 score of all the models lies between the range of 0.92 - 0.94 under various conditions.

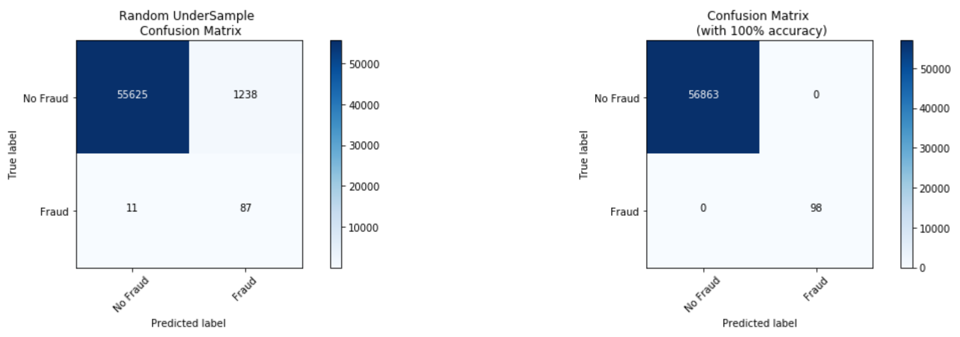

Figure 5.3: Confusion Matrix of Neural Network with Undersampling

This confusion matrix shows the number of false positives in the case of Random undersampling is quite high. This is particularly not helpful in terms of tagging valid transactions as fraud. This leads to the blocking of the cards of the users. The aim is also to reduce the false positives and false negatives. False negatives are the fraud cases that have not been detected. SMOTE oversampling has better results than the undersampling. The number of false negatives has increased by a small number but the number of false positives decreases by a considerable margin. This is useful as the total number of misclassifications are also brought down.

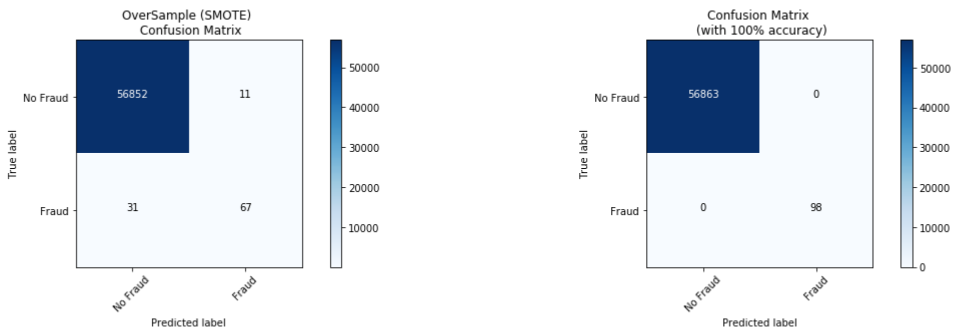

Figure 5.4: Confusion Matrix of Neural Network with SMOTE

Chapter 6

Security and Privacy

6.1 Challenges

With various components such as cloud and the end user devices, various avenues are open for security attacks and privacy breaches due to the reason maintaining diverse protocols across the platforms is a big task. Since we need to handle the data transfers between many devices, allowing free access to data is a major roadblock that must be overcome. Since all the data endpoints are managed by a single user, maintaining the security layers is not that difficult. The lack of a clear boundary is also a vulnerability for attacks from internal and external sources.

The system must be made immune to common attacks such as Denial of Service (DOS), Man in the middle, etc. A summary of the different aspects of security and privacy is made below. Since the architecture of this research is based on network and end user device, the probability of the algorithm being under attack is very less. The main emphasis is placed on the setup and the network.

6.2 Data Security

When the data is being transferred between the cloud and the device, data security is a major concern. Securing our own system is not enough, due to various other applications being hosted on the same server, they also act as a doorway to security attacks. This is the case with most of the major cloud service providers. One way to mitigate this issue would be to encrypt the data stored on the cloud. At the same time, the process can be run on encrypted data which is also known as Homographic Encryption. If this data encryption is implemented on the server side, it would remove the problem of key sharing. This makes the system more flexible and safer. This can be used to run the model on new devices easily with minimal setup. The drawback of this approach is that it requires more computational power to run the algorithm. This would not be suitable for this research area as the end devices are mobile and are running on battery power.

Regarding the authentication, instead of having the credentials for each person, the device acts as the device key. This reduces the number of endpoints from which data can be compromised. The data is routed only on the secure network rather than the internet. Because of this man in the middle attack, DOS, phishing and many more attacks can be controlled to a large extent. On the device level, using private keys and SHA protocols can be particularly useful in securing the data. SHA is particularly useful as the current infrastructure takes a long time to break the keys. By the time, the keys are broken the data would have lost its validity. The keys are also changed at regular intervals so that the data is not lost if the keys have been stolen.

6.3 Data Privacy

A major attack which could be at the cloud layer is that an attacker can take control of the data stream. Anyone with access to this layer can introduce a trojan into the system. Since there is a possibility that sensitive data is also stored on this server which includes medical records of the citizens of Ireland, privacy is of the topmost concern of this system. Due to recent developments regarding GDPR, it is also necessary that the right consent is got from the users so that this data can be stored on the servers. Utmost care must be taken that the data is shown only to the relevant people at the relevant time to reduce the chance of data leakage.

Another main aspect that needs to be investigated is the intrinsic attackers who can misuse the freedom of access to data. Care must be taken against SQL injections which can be used to infiltrate the database. An assessment of the data must be done, and it should be categorised as high or low sensitivity. For data with higher risk, extra measures can be taken to keep it more secure and out of reach. Since the data is owned by the government, there are periodical statistics which are published to check the performance of the department. There have been instances where data has been back engineered to reveal sensitive information about individuals.

Chapter 7

Conclusion

In this thesis, several classification algorithms and a neural network was implemented to classify the transactions. For this, a public data set was used, which has the transactions made by European residents in September 2013. The dataset has a total of 284,807 transactions out of which only 492 fraud transactions were present. The class imbalance stood at 0.172%.

When highly class imbalanced datasets are given as input, the classifiers tend to be swing towards the major class. Consequently, this leads the classifier to label a fraudulent transaction as a non-fraudulent transaction. To overcome this roadblock, Random Undersampling and Synthetic Minority Over-sampling Technique (SMOTE) were implemented. The algorithms implemented were Logistic Regression, kNN, SVM, and Decision Tree Classifier. A neural network with one hidden layer was also constructed. All the models were then compared for performance between undersampling and SMOTE. The results showed that Logistic Regression performed better than all the other classifiers but Neural Network with SMOTE implemented had the least number of misclassified transactions.

Chapter 8

Future Work

The models implemented above have to be tested for real time detection. A cost function can also be added which penalises a misclassification based on the transaction amount. Higher the amount of transaction, higher is the cost of misclassification. This can help in keeping the losses due to fraud transactions as low as possible.

The fraud industry is a dynamic one and the fraudsters are finding innovative ways to attempt to scam common man. Hence, it is important to study how good are the predictive models for catching new frauds. However, this would require huge amounts of data, something which the banks and the financial institutions would have to make open source.

Bibliography

[1] A. Srivastava, A. Kundu, S. Sural, and A. Majumdar, “Credit Card Fraud Detection Using Hidden Markov Model,” *IEEE Transactions on Dependable and Secure Computing*, vol. 5, pp. 37–48, Jan. 2008.

[2] Y. Kunlin, “A Memory-Enhanced Framework for Financial Fraud Detection,” in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, (Orlando, FL), pp. 871–874, IEEE, Dec. 2018.

[3] 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)*, (Chennai, India), pp. 255–258, IEEE, Feb. 2017.

[4] P. Zhao, X. Fu, W. Wu, D. Li, and J. Li, “Network-Based Feature Extraction Method for Fraud Detection via Label Propagation,” in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, (Kyoto, Japan), pp. 1–6, IEEE, Feb. 2019.

[5] L. Zheng, G. Liu, C. Yan, and C. Jiang, “Transaction Fraud Detection Based on Total Order Relation and Behavior Diversity,” *IEEE Transactions on Computational Social Systems*, vol. 5, pp. 796–806, Sept. 2018.

[6] D. Huang, D. Mu, L. Yang, and X. Cai, “CoDetect: Financial Fraud Detection With Anomaly Feature Detection,” *IEEE Access*, vol. 6, pp. 19161–19174, 2018.

[7] K. Randhawa, C. K. Loo, M. Seera, C. P. Lim, and A. K. Nandi, “Credit Card Fraud Detection Using AdaBoost and Majority Voting,” *IEEE Access*, vol. 6, pp. 14277–14284, 2018.

[8] C. Jiang, J. Song, G. Liu, L. Zheng, and W. Luan, “Credit Card Fraud Detection: A Novel Approach Using Aggregation Strategy and Feedback Mechanism,” *IEEE Internet of Things Journal*, vol. 5, pp. 3637–3647, Oct. 2018.

[9] J. A. Gmez, J. Arvalo, R. Paredes, and J. Nin, “End-to-end neural network architecture for fraud scoring in card payments,” *Pattern Recognition Letters*, vol. 105, pp. 175–181, Apr. 2018.

[10] M. Zareapoor and P. Shamsolmoali, “Application of Credit Card Fraud Detection: Based on Bagging Ensemble Classifier,” *Procedia Computer Science*, vol. 48, pp. 679–685, 2015.

[11] I. Benchaji and S. Douzi, “Using Genetic Algorithm to Improve Classification of Imbalanced Datasets for Credit Card Fraud Detection,” p. 5.

[12] D. S. Sisodia, N. K. Reddy, and S. Bhandari, “Performance evaluation of class balancing techniques for credit card fraud detection,” in *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, (Chennai), pp. 2747–2752, IEEE, Sept. 2017.

[13] E. C. Bank, “Fifth report on card fraud, September 2018.”

[14] Y. Gao, C. Sun, R. Li, Q. Li, L. Cui, and B. Gong, “An Efficient Fraud Identification Method Combining Manifold Learning and Outliers Detection in Mobile Healthcare Services,” *IEEE Access*, vol. 6, pp. 60059–60068, 2018.

[15] S. Wang, G. Liu, Z. Li, S. Xuan, C. Yan, and C. Jiang, “Credit Card Fraud Detection Using Capsule Network,” in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, (Miyazaki, Japan), pp. 3679–3684, IEEE, Oct. 2018.

[16] M. Zamini and G. Montazer, “Credit Card Fraud Detection using autoencoder based clustering,” in *2018 9th International Symposium on Telecommunications (IST)*, (Tehran, Iran), pp. 486–491, IEEE, Dec. 2018.

[17] M. Arafat, A. Qusef, and G. Sammour, “Detection of Wangiri Telecommunication Fraud Using Ensemble Learning,” in *2019 IEEE Jordan International Joint Con-*

ference on Electrical Engineering and Information Technology (JEEIT), (Amman, Jordan), pp. 330–335, IEEE, Apr. 2019.

[18] A. Kundu, S. Panigrahi, S. Sural, and A. Majumdar, "BLAST-SSAHA Hybridization for Credit Card Fraud Detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, pp. 309–315, Oct. 2009.

[19] M. Behdad, L. Barone, M. Bennamoun, and T. French, "Nature-Inspired Techniques in the Context of Fraud Detection," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, pp. 1273–1290, Nov. 2012.

[20] "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 3784–3797, Aug. 2018.

[21] Y.-J. Chen and C.-H. Wu, "On Big Data-Based Fraud Detection Method for Financial Statements of Business Groups," in *2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, (Hamamatsu), pp. 986–987, IEEE, July 2017.

[22] A. Zakaryazad and E. Duman, "A profit-driven Artificial Neural Network (ANN) with applications to fraud detection and direct marketing," *Neurocomputing*, vol. 175, pp. 121–131, Jan. 2016.

[23] D. Wang, B. Chen, and J. Chen, "Credit card fraud detection strategies with consumer incentives," *Omega*, vol. 88, pp. 179–195, Oct. 2019.

[24] "Credit Card Fraud Detection | Kaggle."

[25] "Credit Card Fraud Detection - Machine Learning methods," *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH), INFOTEH-JAHORINA (INFOTEH), 2019 18th International Symposium*, p. 1, 2019.

[26] "Real-time Credit Card Fraud Detection Using Machine Learning," *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Cloud Computing, Data Science & Engineering (Confluence), 2019 9th International Conference on*, p. 488, 2019.

[27] "Performance Evaluation of Machine Learning Algorithms for Credit Card Fraud Detection," *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Cloud Computing, Data Science & Engineering (Confluence), 2019 9th International Conference on*, p. 320, 2019.

[28] "A Feature Extraction Method for Credit Card Fraud Detection," *2019 2nd International Conference on Intelligent Autonomous Systems (ICoIAS), Intelligent Autonomous Systems (ICoIAS), 2019 2nd International Conference on*, p. 70, 2019.

[29] "Application of Computational Intelligence to Identify Credit Card Fraud," *2018 International Conference on Innovation in Engineering and Technology (ICIET), Innovation in Engineering and Technology (ICIET), 2018 International Conference on*, p. 1, 2018.

[30] "Credit Card Fraud Detection Using Anomaly Techniques," *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), Innovations in Information and Communication Technology (ICIICT), 2019 1st International Conference on*, p. 1, 2019.