PROJECT REPORT

on

CREDIT CARD AND BITCOIN FRAUD MANAGEMENT SYSTEM

submitted by

Name	SAP No.
1) Aditya Prajapati	60002170005
2) Devanshi Shah	60002170027
3) Mansi Parekh	60002170062

under the guidance of

Prof. Vishakha Kelkar

Assistant Professor

DEPARTMENT OF

ELECTRONICS AND TELECOMMUNICATION ENGINEERING

Academic Year: 2020-2021

Shri Vile Parle Kelavani Mandal's

Dwarkadas J. Sanghvi College of Engineering

Plot no. U-15, JVPD Scheme, Bhaktivedanta Swami Marg,

Vile Parle (W), Mumbai – 400 056

TABLE OF CONTENTS

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Department of Electronics and Telecommunication Engineering

This is to certify that the Project Stage – II

“CREDIT CARD AND BITCOIN FRAUD MANAGEMENT SYSTEM”

Submitted by:

1. Aditya Prajapati
2. Devanshi Shah
3. Mansi Parekh

Students of **Electronics and Telecommunication Engineering** have successfully completed their **Project Stage – II** required for the fulfillment of **SEM VIII** as per the norms prescribed by the **University of Mumbai** during the First half of the year 2021. The project report has been assessed and found to be satisfactory.

Internal Guide **External Guide**

Head of Department **Principal**

Internal Examiner **External Examiner**

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature

Name

1. Aditya Prajapati
2. Devanshi Shah
3. Mansi Parekh

SAP ID:

1. 60002170005
2. 60002170027
3. 60002170062

Date

ABSTRACT

With the age of digitization, there is an increased amount of usage of online transactions via credit cards and net banking option. With e-commerce services offered by most of the websites, credit cards have become a popular mode of payment. With increase in credit card transactions, there has been a hike in the number of fraudulent transactions as well. Such an increase hampers the trust of the people on online transactions as well leads to billions of losses for banks all over the world. As per the reports of 2018, credit card frauds increased by 18.4% and led to a loss of 24.26 billion dollars. To curb these tremendous losses, the researchers began using different techniques and methods to prevent and detect unusual transactions. Frauds are known to be dynamic and have no patterns, hence they are not easy to identify. Fraudsters use recent technological advancements to their advantage. They somehow bypass security checks, leading to the loss of millions of dollars. Analyzing and detecting unusual activities using data mining, machine learning and deep learning techniques is one way of tracing fraudulent transactions. Many artificial intelligence techniques have been proposed to deal with the anomaly detection problem; some results appear to be considerably assuring, but there is no explicit superior solution. This project aims to elucidate the modelling of credit card transactions with the statistical data of the transactions that turned out to be fraud. It also presents an overview of the block chain technology and its application in the financial sector in light of anomaly detection. A range of algorithms such as isolation forest, histogram based outlier detection (HBOS), cluster based local outlier factor (CBLOF), principal component analysis (PCA), K-means, deep auto encoder networks and ensemble method are evaluated and compared.This is exhibited by making data sets using machine learning with Credit Card Fraud Detection. The Credit Card Fraud Detection model is then used to determine whether a new transaction is fraudulent or not. Our objective here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications. Credit Card Fraud Detection is a typical sample of classification. In this process, we have focused on analyzing and pre-processing data sets as well as the organization of multiple anomaly detection algorithms.

Keywords- Upsampling, Downsampling, Neural Networks, Isolation Forest, K-Means, K-Nearest Neighbors (KNN), Local Outlier Factor, Support Vector Machine (SVM), block chain, bitcoin, anomaly detection.

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

INTRODUCTION

Credit card plays a very important rule in today's economy. It becomes an unavoidable part of household, business and global activities. Although using credit cards provides enormous benefits when used carefully and responsibly, significant credit and financial damages may be caused by fraudulent activities. Fraud in credit card transactions is unauthorized and unwanted usage of an account by someone other than the owner of that account. 46% of the world’s credit card fraud happens in the United States out of which 25.7% charges were made to an existing credit card, 21.4% were made using an existing checking or savings account, 10.5% of transactions were made on an existing loan or line of credit. Necessary prevention measures can be taken to stop this abuse and the behaviour of such fraudulent practices can be studied to minimize it and protect against similar occurrences in the future. In other words, Credit Card Fraud can be defined as a case where a person uses someone else’s credit card for personal reasons while the owner and the card issuing authorities are unaware of the fact that the card is being used.

There are two types of credit card frauds. One is theft of physical card, and other one is stealing sensitive information from the card, such as card number, CVV code, type of card and other. By stealing credit card information, a fraudster can broach a large amount of money or make a large amount of purchase before cardholder finds out. Because of that, companies use various machine learning methods to recognize which transactions are fraudulent and which are not. Fraud detection involves monitoring the activities of a number of users in order to estimate, perceive or avoid objectionable behaviour, which consist of fraud, intrusion, and defaulting.

Credit Card Fraud is a very relevant problem that demands the attention of communities such as machine learning and data science where the solution to this problem can be automated. This problem is particularly challenging from the perspective of learning, as it is characterized by various factors such as class imbalance. The number of valid transactions far outnumber fraudulent ones. Also, the transaction patterns often change their statistical properties over the course of time. These are not the only challenges in the implementation of a real-world fraud detection system, however. In real world examples, the massive stream of payment requests is quickly scanned by automatic tools that determine which transactions to authorize. Machine learning algorithms are employed to analyse all the authorized transactions and report the suspicious ones. These reports are investigated by professionals who contact the cardholders to confirm if the transaction was genuine or fraudulent.

Suspicious activities in network structures are as old as the invention of the network structure itself. Entities or their activities which tends to behave abnormally within the system is referred to as anomalies. Anomaly detection is widely used in a variety of cybersecurity applications for targeting financial fraud identification, network invasion detection, anti money laundering, virus detection, and more. Usually, a common goal in these networks is to detect those anomalies and prevent such illegal activities from happening in future. With advancements in technology, blockchain emerges to plays a vital role in securing these network structures. Carlozo (2017) elaborates that a blockchain is a decentralised distributed database that maintains on-growing records and ensures that fraudulent records do not become part of this database and previously added records stay immutable. However, participants of a blockchain network can try to conduct illegal activities and, in some cases, succeed in deceiving the system into their advantage.

Current state-of-the-art anomaly detection methodologies are designed and implemented in light of the centralised systems. With the advent of blockchain technology, it brings a need for anomaly detection procedures within these systems as well.

In this thesis, we extract and analyse the data from a publicly available blockchain named bitcoin. Nakamoto (2008) states that bitcoin is a peer-to-peer digital currency blockchain, by using which users can send and receive a form of electronic cash to each other anonymously without the need for any intermediaries.

This thesis aims to detect anomalous or suspicious transactions in the bitcoin network, where all nodes are unlabeled, and there is no evidence that if any given transaction is an illicit activity. The primary focus is to detect anomalies within the bitcoin transaction network.

CHAPTER 2: LITERATURE REVIEW

LITERATURE REVIEW

The fraud detection is a complex task and there is no system that correctly predicts any transaction as fraudulent. The properties for a good fraud detection system are:

- The system should be able to handle noise. This means that the system should be capable to deal with errors present in the dataset for example incorrect dates. If the system is incapable of dealing with noise then it leads to a reduction in the accuracy of generalization that can be achieved even if the dataset is then extensive and exhaustive.
- The system must be able to handle skewed distributions as a small percentage of credit card transactions are actually fraudulent. This could be overcome by dividing the data into sections where distribution is less skewed.
- The system should be able to detect overlapping data. Many transactions may resemble fraudulent transactions. However, in reality they are legitimate. The vice versa could also happen. These overlapping data values need to be detected by the system.
- The system should comprise of a decision layer which could decide the action to be taken when a fraudulent transaction is detected, taking into consideration the factors such as amount of transaction and quality of customer doing the transaction.
- Since after a while successful fraud technique decrease in efficiency as they become common and thus can be stopped, the system must be able to adapt to new kinds of fraud. This is because a good fraud would then try and find new techniques to execute illegitimate transactions.

Outlier detection alludes to the issue of finding patterns in data that are totally different from the rest of the information dependent on fitting measurements. Such patterns contain significant information with respect to abnormal behaviour of the framework portrayed by the data. These bizarre patterns are typically called anomalies. Outlier detection is a critical task as outliers indicate abnormal running conditions from which significant performance degradation may happen. Anomaly location is a finds gigantic use in application areas, for example, credit card fraud detection. Techniques used in fraud detection can be divided into two:

➢ **Unsupervised outlier detection technique**

Unsupervised are those where there are no prior sets in which the state of the transactions is known to be fraud or legitimate. An unsupervised outlier detection technique does not make any assumption about the availability of labelled data. This method simply seeks those accounts, customer etc, whose behaviour is “unusual”. Unsupervised methods are useful in applications where there is no prior knowledge about the particular class of observations in a data set. An advantage of using unsupervised methods over supervised methods is that previously occurred undiscovered types of fraud may be detected. Some of the techniques are: Peer Group Analysis, Break Point Analysis and K-Means Clustering technique.

- **Supervised outlier detection technique**

  Supervised techniques where past known legitimate/fraud cases are used to build a model which will produce a suspicion score for the new transactions. Supervised outlier detection techniques assume the availability of a data set which has been needed for the normal as well as the outlier class. Supervised method detects fraudulent transactions that can be used to differentiate between those accounts or transactions which are known to be fraudulent and those which are known to be legitimate. Classification techniques such as statistical discriminate analysis and neural networks can be used to discriminate between fraudulent and non-fraudulent transactions to give transactions a suspicion score. Supervised methods are only trained to differentiate between legitimate transactions and previously known fraud. Some of the techniques are: Gass Algorithm, Bayesian Networks, Hidden Markov, Decision tree, Neural Network (NN), Logistic Regression (LR).

- **Distributed ledger technology (DLT)**

  In the crudest form distribute ledger is merely a database whose copy is independently held by each participant who wants to update it. These participants usually are nodes on a network and records written on these databases are usually called transactions. Distribution of these records is unique as they are not communicated to other nodes in the network using a central authority node. However, instead of that, each record is independently composed by each node individually. This leads to all the participant nodes of the network to process all incoming transactions and reach a conclusion about its authenticity. Finally, a consensus is achieved based on majority votes of conclusions of the network. Once there is this consensus achieved the distributed ledger is updated, and at this point, all nodes on the network maintain an identical copy of the ledger which holds all transactions. This architecture enables a new dexterity as the system of records can go beyond being a simple database.

  Distributed ledger is a dynamic type of media that possesses the attributes and capability go beyond legacy databases. The innovation of distributed ledgers allows its users not only to accumulate and communicate information in a distributedly secure manner but also enables them to go beyond relationships among data. Distributed ledgers provide a trustworthy, secure, and accountable way of tracking transactions without the need for a central validating authority, and they could provide the foundation to make the web a genuinely decentralised autonomous system.

➢ **Blockchain**

The blockchain is a type of distributed ledger. Apparent from its name it is a chain of logically connected data blocks. A list of growing records refer to as blocks, and each block contains a timestamp, transactional data and a unique cryptographic hash. The hash in the block links it to the previous block in the distributed ledger all the way to form a chain of logical links to the first block called genesis block using these cryptographic hashes. The blockchain is a discrete design which makes it resistant to modification of the data as well as duplicate entries. It is said that it is an open-distributed ledger that can be utilised to document transactions among two unknown parties verifiably and permanently. A blockchain usually uses a peer-to-peer network for seamless inter-node communication and validating new blocks. Once transaction data is appended to the block, it cannot be altered retroactively without modification of all subsequent blocks. In order to alter any record, it would require a consensus of the majority of network nodes, which is highly challenging to achieve. Although records of a block chain are not immutable, still it may be considered the secure design and less prone to vulnerabilities. It also illustrates a distributed computing system with high byzantine fault tolerance. The decentralization and byzantine problem will be explained later in the project.

CHAPTER 3: THEORY

WORKING PRINCIPLE

A. UNDERSAMPLING AND OVERSAMPLING

Imbalanced datasets have a large bias in the class distribution, such as 1:100 or 1:1000 cases in the minority class compared to the majority class. Many machine learning algorithms can be influenced by this bias in the training dataset, leading some to completely neglect the minority class. This is a concern since predictions are usually most valuable for the minority class. Randomly resampling the training dataset is one approach to remedy the issue of class imbalance. Undersampling (removing some instances from the majority class) and Oversampling (duplicating observations from the minority class) are the two primary ways to randomly resample an imbalanced dataset.

B. MACHINE LEARNING AND DEEP LEARNING ALGORITHMS

Anomaly detection is categorized into three main types of learning methods; Supervised learning methods, Semi-Supervised learning methods and Unsupervised learning methods. Majority of real-world anomaly detection problems fall under unsupervised learning domain such as credit card fraud detection, network intrusion detection and our blockchain fraudulent transaction detection.

- **Unsupervised Learning Techniques**
  Unsupervised are those where there are no prior sets in which the state of the transactions is known to be fraud or legitimate. An unsupervised outlier detection technique does not make any assumption about the availability of labelled data. This method simply seeks those accounts, customer etc, whose behaviour is “unusual”. Unsupervised methods are useful in applications where there is no prior knowledge about the particular class of observations in a data set. An advantage of using unsupervised methods over supervised methods is that previously occurred undiscovered types of fraud may be detected. Some of the techniques are: Peer Group Analysis, Break Point Analysis and K-Means Clustering technique.

- **Supervised Learning Techniques**
  Supervised techniques where past known legitimate/fraud cases are used to build a model which will produce a suspicion score for the new transactions. Supervised outlier detection techniques assume the availability of a data set which has been needed for the normal as well as the outlier class. Supervised method detects fraudulent transactions that can be used to differentiate between those accounts or transactions which are known to be fraudulent and those which are known to be legitimate. Classification techniques such as statistical discriminate analysis and neural networks can be used to discriminate between fraudulent and non-fraudulent transactions to give transactions a suspicion score. Supervised methods are only trained to differentiate between legitimate transactions and previously known fraud. Some of the techniques are: Gass Algorithm, Bayesian Networks, Hidden Markov, Decision tree, Neural Network (NN), Logistic Regression (LR).

- In this project we have applied 6 different Supervised and Unsupervised and algorithms to get the desired results, which are as follows:

1. **<u>K-Means Algorithm</u>**

In clustering algorithms, K-means is one of the most popular choices for anomaly detection. It was introduced as an unsupervised learning algorithm by J. MacQueen (1967). The algorithm aims to partition the data into k di_erent clusters where each sample belongs to the cluster with the nearest mean. Each cluster is represented by a centroid c which is the sample mean of observations in that cluster. The similarity measure used when comparing observations to the cluster is squared Euclidean distance,
which can be computed by using equation given below where xi is the observations and ci is the centroids and n represents the number of observations.

$$d^2(x,c) = \sum_{i=1}^{n}(xi - ci)^2$$

The first step of the process in the K-means algorithm is to initialize the k centroids. Then each observation is iteratively compared with each centroid and assigned to it based on the clusters least sum of squares. These assigned observations ultimately form clusters around the speci_c centroids. This process is the same as assigning each observation to its nearest centroid by using Euclidean distance or squared Euclidean distance as the minimum value for both should be same. This step in the algorithm is also known as the expectation step.

After each observation is assigned to their nearest centroids and first clusters are formed, the centroids are updated based on the new sample mean for each cluster. Centroids can be any arbitrary data point in the cluster representing the mean of the cluster; it does not have to be any one of the observations. The arithmetic mean is also minimizing the sum of squares within the clusters. This centroid updating step is also known as the maximization step.

At the time of termination of the K-means algorithm, results with final centroids and cluster assignment of all observations are delivered. The algorithm terminates when centroids have converged, which means the value of centroids stops updating with more iterations. There is no guarantee that k-means reaches the global optimum before termination, so it is a common practice to run the algorithm several times on the data and choose a model with the lowest within-cluster sum of squared distances.
As the mean of each cluster can be any arbitrary data point running the algorithm every time may produce different solutions that have converged differently, but could be somewhat similar. If the centroids are initialized near to the local optimum, the algorithm is likely to converge faster. K-means is a variant of a more generalized clustering approach Expectation-Maximization (EM). EM algorithm has a more probabilistic approach to clustering and creates soft-clusters in contrary to hardclusters created by K-means.

Many techniques are proposed to initialize the centroids for fast convergence and fast termination. However, Arthur and Vassilvitskii (2007) proposed an approach named k-means++ in which the standard initialization of the centroids is done by uniformly selecting data points from the dataset. Arthur and Vassilvitskii (2007) proved in their original paper that standard k-means++ outperforms standard Kmeans in convergence time and minimizing within-cluster sum of squared distance.

2. Local Outlier Factor Algorithm

He, Xu, and Deng (2003) proposed an algorithm named cluster-based local out- lier factor (CBLOF) that has properties of a cluster-based algorithm as well as the Breunig et al. (2000) local outlier factor (LOF) algorithm. The idea of anomaly de- tection in this algorithm revolves around clustering the data which are later used to compute an anomaly score in a similar way to LOF. Any arbitrary clustering algo- rithm can be plugged in the clustering step. However, the quality of the clustering algorithm's results directly influences the quality of CBLOF's results.

The algorithm proceeds in a manner that it clusters a dataset *D* using an arbitrary clustering algorithm and assigns each observation in *D* to a cluster. Their respective sizes sort the clusters such that $|C_1| \geq |C_2| \geq \cdots \geq |C_k|$ where C_1, C_2, ... , C_k represent all clusters with *k* as the number of clusters. The intersection of any pair of clusters should be an empty set, while the union of all clusters should represent all the observations of dataset *D*. The next step requires searching a boundary index value which separates Small Clusters (SC) from the Large Clusters (LC). There are two ways to calculate this boundary. The first is given by Equation.

$$(|C_1| + |C_2| + \cdots + |C_b|) \geq |D| \cdot \alpha$$

An alternative way touse this equation is:

$$(|C_b| / |C_{b+1}|) \geq \beta$$

α and β are user-defined parameters in the equations mentioned above. α is be- tween the range of [0,1] and represents the data comprised by the Large Clusters (LC). β value should be set to greater than one and defines the lower bound on relative size among a pair of consecutive clusters. equation (4.4) represents the approach which separates bigger clusters from the smaller ones by adding the biggest clusters to LC until some defined part of data is covered. On the other hand, implements the idea that a significant boundary location can be found between LC and SC among clusters which vary relatively in size. Concisely, it chooses a boundary between two consecutive clusters if the next cluster is β times smaller than the previous one. If two equations result in a way that value is different for boundary *b*, the smallest value is selected.

After deciding the value of *b*, the sets LC and SC are created. LC contains all the clusters with the index equal or smaller to the *b* and can be represented with

LC = C_i | i ≤ b. Whereas SC consists of the remaining clusters and is represented by SC = C_i | i > b. Finally, CBLOF scores are calculated for each observation in *D* using equation.

$$CBLOF(t) = \begin{cases} |C_i| \cdot \min(\text{dist}(t, C_j)) & : t \in C_i, C_i \in SC \text{ and } C_j \in LC \\ |C_i| \cdot \text{dist}(t, C_i) & : t \in C_i, C_i \in LC \end{cases}$$

Data points in the large clusters (LC) are estimated as normal observations, and data points in small clusters (SC) are estimated as anomalous observations. A spe- cific central point is calculated to represent each cluster and the distance from this point to each observation in the large cluster defines its anomaly score. A data point further away from the cluster centre would have a higher chance of being an anomaly, hence resulting in a higher anomaly score. For the data points not residing inside large clusters, their anomaly score is calculated based on the nearest data points to a large cluster. Data points which are far from large clusters end up having even larger anomaly score. CBLOF computes anomaly score in a similar way than LOF with the significant difference that distances are calculated with clusters insteadof specific points.

3. Isolation Forest Algorithm

Liu, Ting, and Zhou (2008) published an algorithm called "isolation forest" for unsupervised anomaly detection that achieved much popularity in recent years. The idea behind isolation forest is that it is comparatively more straightforward to isolate anomalies from the data than to build a model that could estimate normal behaviour. In order to isolate an anomalous observation, the dataset is randomly and recursively partitioned until the only data point left in the partition is the observation. A tree structure is used to represent the recursive partition. The base of the isolation forest algorithm is on the ensemble of isolation trees, A forest of random isolation trees is constructed to estimate the measure of normality and abnormality of observations in the dataset. An isolation forest is a combination of isolation trees, and these trees are actual binary trees in which each node has either zero or two child nodes. Let *N* be a node which is either a leaf node with no children or parent node with two child nodes N_l and N_r. To decide which children nodes go to which parent node a test is associated with node *N*. This test involves selecting a random feature *q* from the data and a random split point *p* such that the nodes $p < q$ are under N_l and $p \geq q$ are under N_r. Multiple random trees are formed recursively, and data is partitioned in each one of them unless all observations are isolated for each iteration. The path length from the root node of the tree to the leaf defines the number of partitions required to isolate that observation. It is observed that for randomly partitioned data, the path length of anomalies is shorter than of standard observations. One of the reasons for this phenomenon to occur is that usually the number of anomalies in a dataset is smaller than the number of standard observations, which results in a shorter number of partitions and hence makes isolation of anomalous observation easier. Another reason for anomalous observations to be separated in earlier partitioning is that they have distinct attribute values as compared with standard observations. Figure 4.2 illustrates that anomalies are more sensitive to isolation. The figure below depicts the anomalous observation x_o requires only four partitions to be isolated, whereas isolating standard observation x_i requires twelve random partitions. Given a dataset $X = \{x_1, x_2, \dots, x_n\}$ containing n observations, random recursive isolation trees are built until each tree reaches the height limit of $|X| = 1$ or all data in X have the same value. Presuming majority of data in X is distinct, and the isolation tree is fully grown, each observation is isolated to a leaf node. Then the tree will have n leaves and $n-1$ internal nodes, in total $2n-1$ nodes. The path length of an observation x can be calculated by heuristic $h(x)$ which is a measurement of the count of edges it takes to traverse from the root node to the leaf node. In order to use the isolation forest as an anomaly detection technique, it has to generate comparable anomaly scores as $h(x)$ cannot be directly used as an anomaly score.

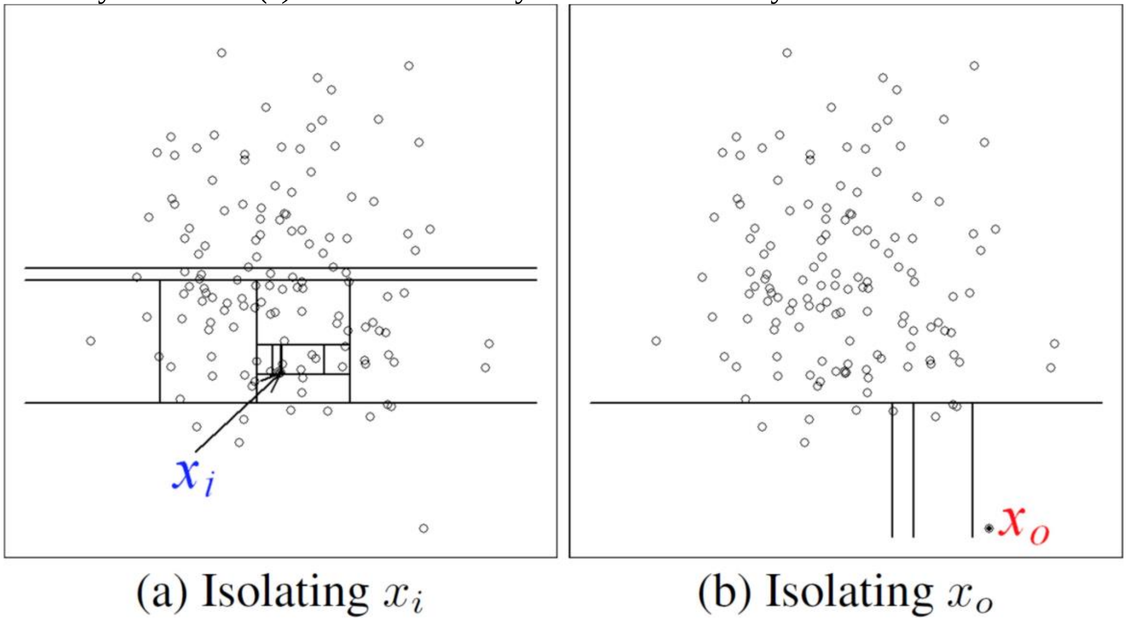


Figure 3.1. The visual difference between the isolation of a standard observation x_i versus an anomalous observation x_o; Liu, Ting, and Zhou (2008)

It is because while the maximum height of a tree grows in the order of n, the average height grows in the order of $\log n$. Isolation trees have a similar structure as binary search trees (BSTs). Therefore the average path length of a failed search in a binary search tree is comparable to the average of $h(x)$. As stated by Preiss (2008), the average path length of a failed BST search in a tree with n nodes can be calculated as

$$c(n) = 2H(n-1) - (2(n-1)/n)$$

where $H(i) \approx \ln(i) + 0.5772156649$ (Eulers constant). The anomaly score s for an observation x is defined as:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Here *E(h(x))* is the mean of all the *h(x)* from all isolation trees. While training an isolation forest algorithm, isolation trees are randomly created as explained above. Training requires two parameters: the number of trees *t* and the sub-sampling size. Tree height limit *l* is calculated based on the sub-sampling size. The sub-sampling process controls the data size for training and is an essential variable as it can directly affect the performance of the classifier. As a different sub-sample is picked to create an isolation tree each time, it can help isolate the anomalies in a better way. As for anomaly prediction from the trained model, Equation (4.2) is used to calculate an anomaly score *s* as explained above. This anomaly score *s* ranges between the values from [0,1] where values being more close to 1 are more likely to be anomalies.

4. K-Nearest Neighbor Algorithm

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

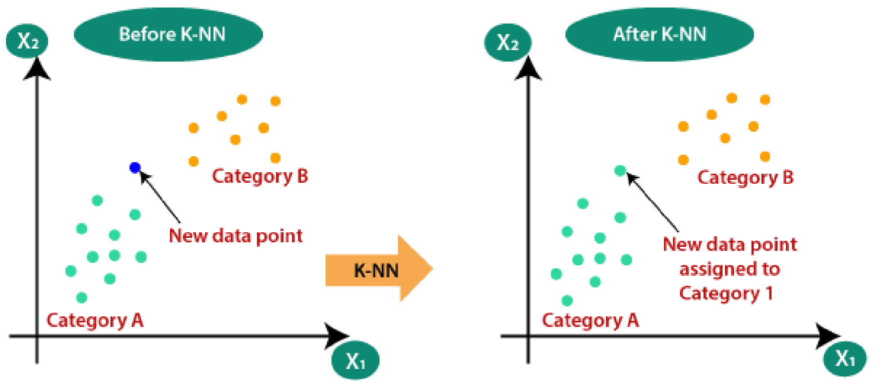


Figure 3.2: K-NN Explaination

<u>How does K-NN work?</u>

The K-NN working can be explained on the basis of the below algorithm:

- Step-1: Select the number K of the neighbors
- Step-2: Calculate the Euclidean distance of K number of neighbors
- Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
- Step-4: Among these k neighbors, count the number of the data points in each category.
- Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.
- Step-6: Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

Firstly, we will choose the number of neighbors, so we will choose the k=5.Next, we will calculate the Euclidean distance between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:

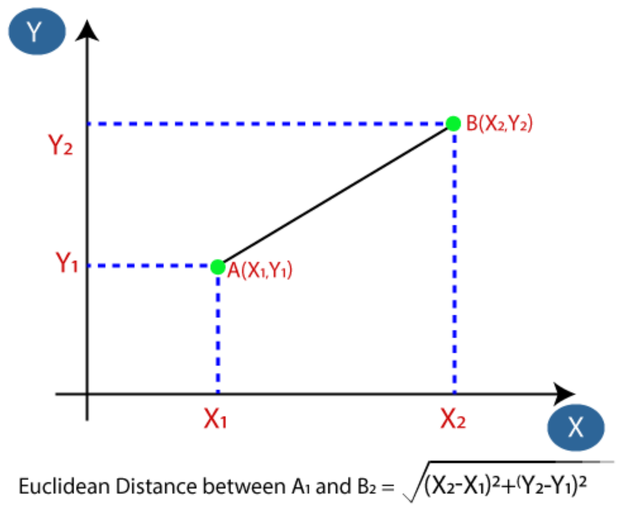

Y
X
Y_2
Y_1
X_1
X_2
$B(X_2,Y_2)$
$A(X_1,Y_1)$
Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

Figure 3.3: Euclidean Distance for K-NN

By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5. A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model. Large values for K are good, but it may find some difficulties.

5. Support Vector Machine Algorithm

Supervised Machine learning algorithm can be used for classification and regression problems. When utilized for classification it separates a given set of binary labelled training data with a hyper plane that is maximally distant from them. They work with the technique of kernels when linear separation is not viable wherein a non-linear mapping to feature space is automatically realized. The hyper-plane found by SVM in feature space corresponds to a non-linear decision boundary in input space.

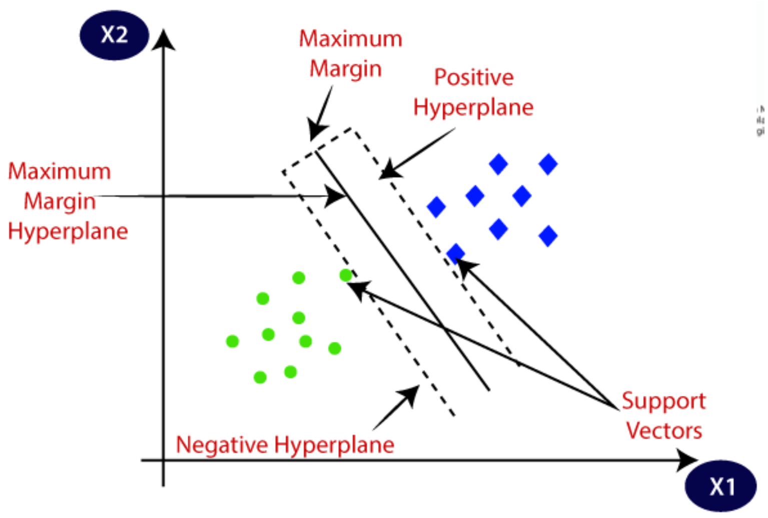


Figure 3.4: SVM Explaination 1

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

SVM can be of two types:

- **Linear SVM:**

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:

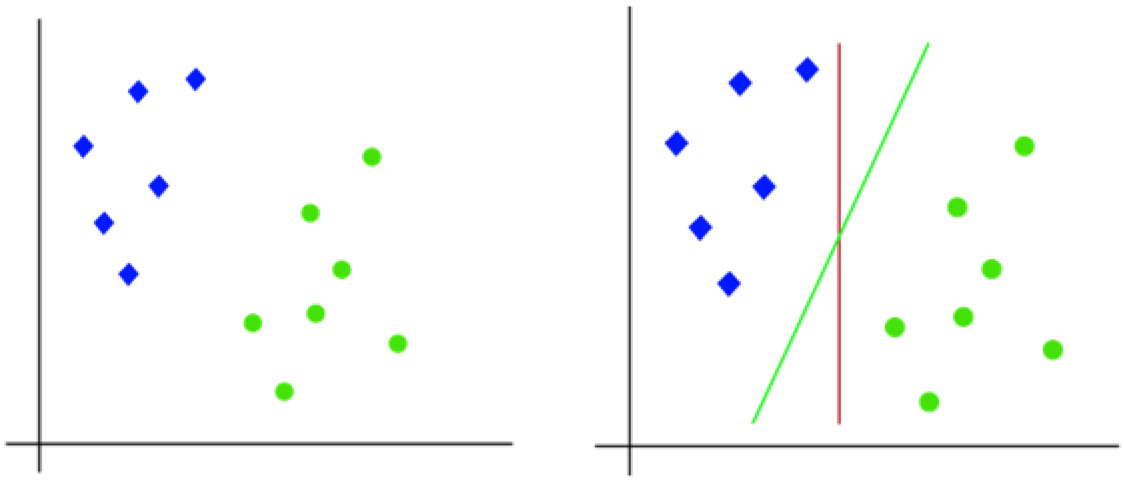


Figure 3.5: SVM Explaination 2

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes.

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.

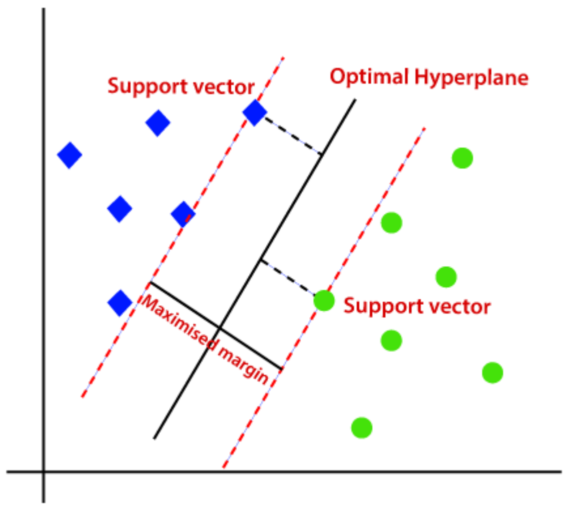


Figure 3.6: SVM Explaination 3

Non-Linear SVM:

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:

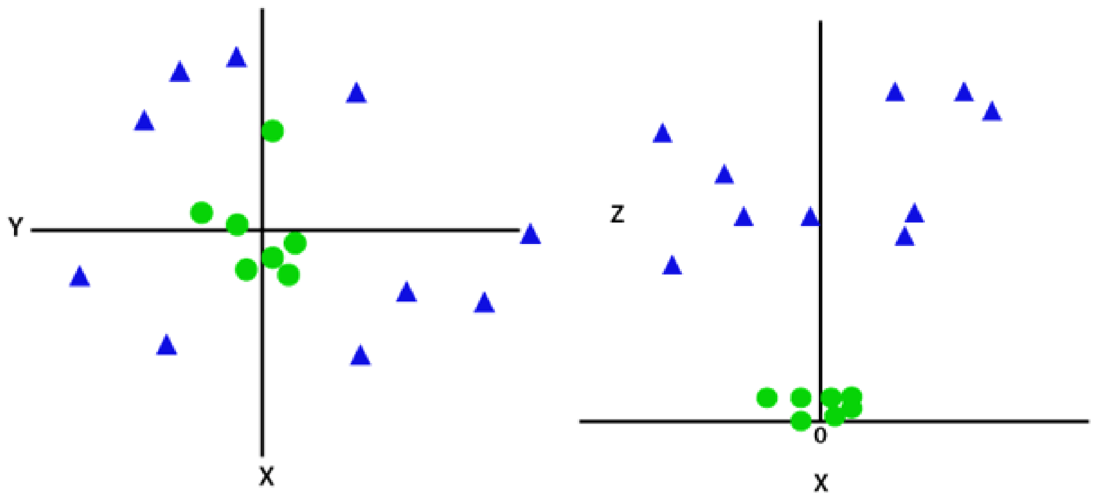


Figure 3.7: SVM Explaination 4

So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$z=x^2 +y^2$

So now, SVM will divide the datasets into classes in the following way. Consider the below image:

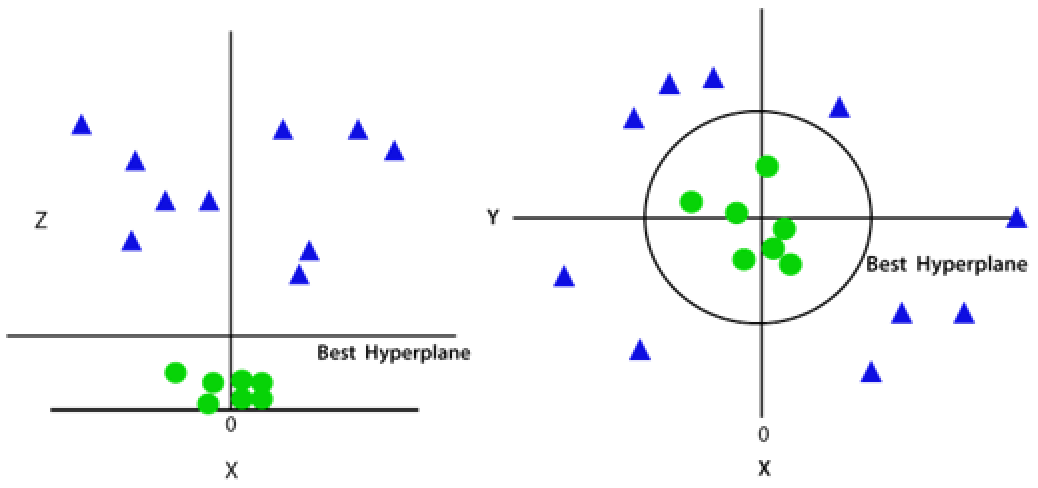


Figure 3.8: SVM Explaination 5

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1.Hence we get a circumference of radius 1 in case of non-linear data.

6. <u>Neural Networks Clustering</u>

Neural network methods are famously used for fraud detection. The standard of neural network is persuaded by the functions of the mind particularly pattern recognition and associative memory. The neural network distinguish comparable examples, predicts future dependent on the affiliated memory of the learned patterns. It is applied in classification and clustering. gains from an earlier time and along these lines, improve results over the long haul. They can likewise extricate rules and foresee future action dependent on the current circumstance. The two parts of neural network are training and recognition. NNs are best for enormous transaction dataset. It is conceivable to utilize the neural network with unsupervised learning for cluster analysis that is based on depends on assessment of the distinction of the weighted vector w of the neural network from the vector of input pattern x and search of neuron, whose weighted coefficient have the minimum distance of w from x. This neuron, which won among the neurons of the network, has the option to adjust its weights and the weights of neurons in its surroundings and accordingly the reaction on submitted learning example to more readily esteem. Subsequent to submitting of a further learning pattern, it can win another neuron of the net that can adjust its weights and the weights of neurons in its surroundings and thus to increase better answer and so on. The clusters are hence made in the net that in specific spots of the network optimally respond to specific indications of submitted patterns, as well as obscured patterns.

Since anomalies are highly rare and heterogeneous, it is difficult to identify all of the anomalies. Many normal instances are wrongly reported as anomalies while true yet sophisticated anomalies are missed. Although a plethora of anomaly detection methods have been introduced over the years, the current state-ofthe-art methods, especially unsupervised methods (e.g., [17, 84]), still often incur high false positives on real-world datasets [20, 115]. How to reduce false positives and enhance detection recall rates is one of the most important and yet difficult challenges, particularly for the significant expense of failing to spotting anomalies.

Anomaly detection in high-dimensional and/or not-independent data. Anomalies often exhibit evident abnormal characteristics in a low-dimensional space yet become hidden and unnoticeable in a high-dimensional space. High-dimensional anomaly detection has been a long-standing problem [178]. Performing anomaly detection in a reduced lower dimensional space spanned by a small subset of original features or newly constructed features is a straightforward solution, e.g., in subspace-based [70, 77, 84, 123] and feature selectionbased methods [12, 109, 111]. However, identifying intricate (e.g., high-order, nonlinear and heterogeneous) feature interactions and couplings [22] may be essential in high-dimensional data, but it remains a major challenge for anomaly detection. Further, how to guarantee the new feature space preserved proper information for specific detection methods is critical to downstream accurate anomaly detection, but it is challenging due to the aforementioned unknowns and heterogeneities of anomalies. Also, it is challenging to detect anomalies from instances that may be dependent on each other such as by temporal, spatial, graph-based and other interdependency relationships [2, 4, 22, 53].

FLOWCHART

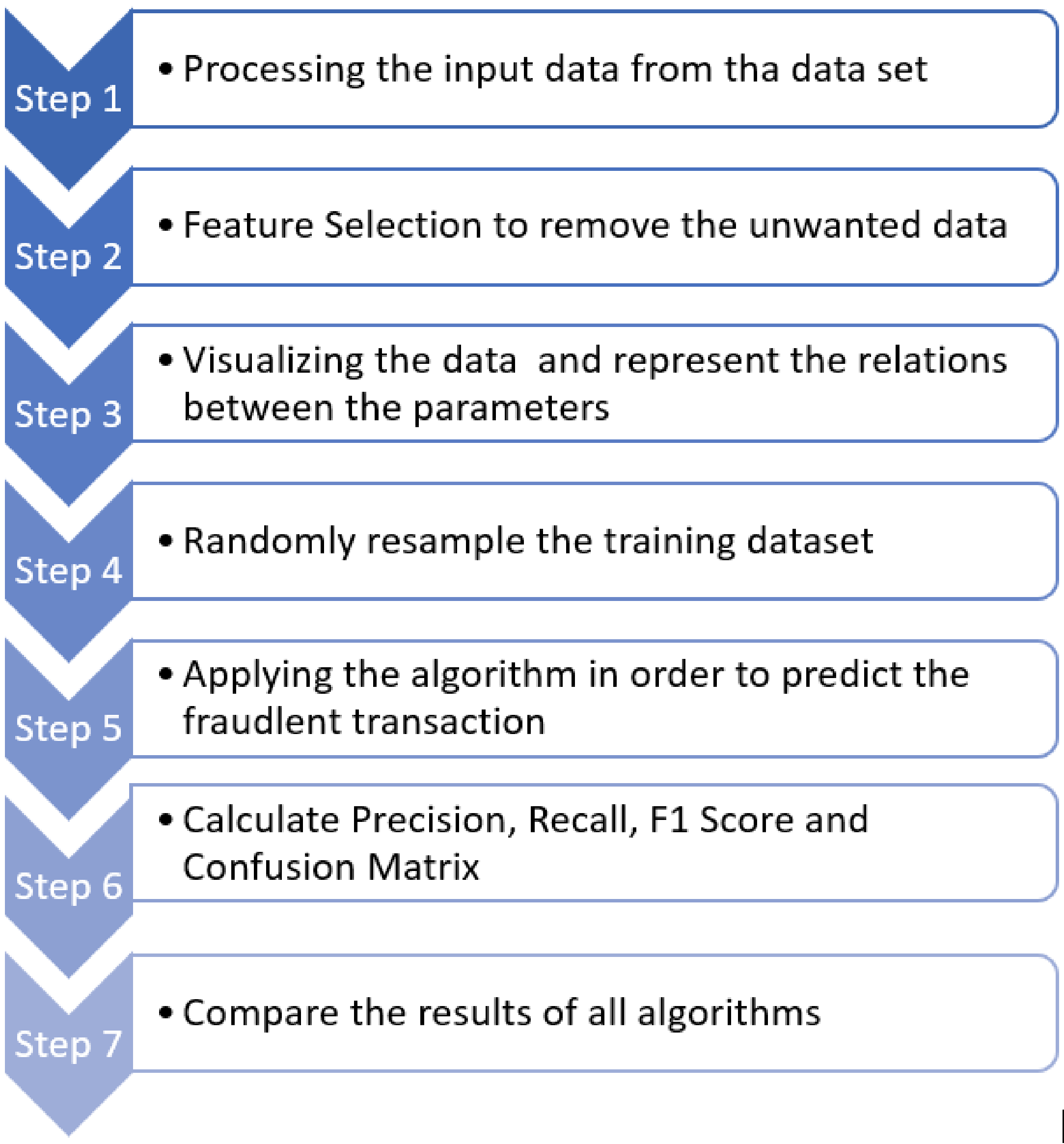

Step 1
• Processing the input data from tha data set
Step 2
• Feature Selection to remove the unwanted data
Step 3
• Visualizing the data and represent the relations between the parameters
Step 4
• Randomly resample the training dataset
Step 5
• Applying the algorithm in order to predict the fraudlent transaction
Step 6
• Calculate Precision, Recall, F1 Score and Confusion Matrix
Step 7
• Compare the results of all algorithms

Figure 3.9: Flowchart of the anomaly detection process

DATASET DESCRIPTION

1. <u>Credit Card Dataset:</u>

The datasets contains transactions made by credit cards in September 2013 by european cardholders.
This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.
It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

2. <u>Bitcoin Dataset</u>

Raw bitcoin blockchain data is used to create the dataset for this thesis. All bitcoin data was synced from the internet using the bitcoin client software. All the data in bitcoin blockchain is stored into a public ledger and is represented by the currency unit called the Bitcoin (BTC). The ledger data contains all the bitcoin transactions from the beginning of the network creation to now. As recorded by Blockchain.com (2019) there are approximately 450,000,000 transactions within the bitcoin ledger. Our dataset consists of 3,02,48,134 transactions with 108 fraudulent transactions. It is highly unbalanced, the positive class (frauds) account for 0.00036% of all transactions. For each transaction, there can be multiple numbers of sender and receiver addresses. Moreover, a single user can own multiple addresses, and each user is anonymous as there is no personal information associated with any of the addresses. Following are the different columns in the dataset.

- tx_hash: Hash of the bitcoin transaction.
- indegree: Number of transactions that are inputs of tx_hash
- outdegree: Number of transactions that are outputs of tx_hash.
- in*btc: Number of bitcoins on each incoming edge to tx*hash.
- out*btc: Number of bitcoins on each outgoing edge from tx*hash.
- total*btc: Net number of bitcoins flowing in and out from tx*hash.
- mean*in*btc: Average number of bitcoins flowing in for tx_hash.
- mean*out*btc: Average number of bitcoins flowing out for tx_hash.
- in-malicious: Will be 1 if the tx_hash is an input of a malicious transaction.
- out-malicious: Will be 1 if the tx_hash is an output of a malicious transaction.
- is-malicious: Will be 1 if the tx_hash is a malicious transaction
- out*and*tx*malicious: Will be 1 if the tx*hash is a malicious transaction or an output of a malicious transaction.

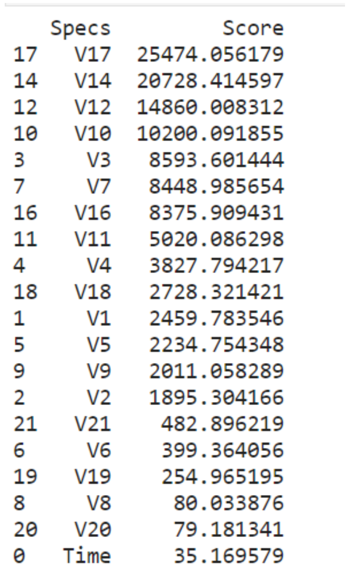
- all*malicious: Will be 1 if the tx*hash is a malicious transaction or an output of a malicious transaction or input of a malicious transaction.

Exploratory data analysis revealed some insights into the data. As also mentioned in previous section, the class distribution of the dataset is highly imbalanced with 30,248,026 non-malicious data points and only 108 malicious transactions, make malicious transactions to be only approximately 0.00035% of whole data.

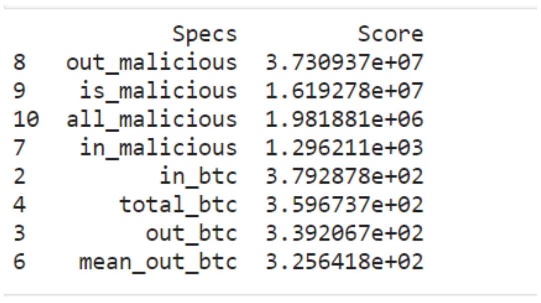
CHAPTER 4: IMPLEMENTATION

FEATURE SELECTION

When creating a predictive model, feature selection is the method of reducing the number of input variables. The number of input variables should be reduced to reduce the computational cost of modelling and, in some cases, to increase the model's accuracy. We have selected 20 best features and 8 best features for credit card and bitcoin datasets respectively using univariate feature selection method. In Univariate statistical-based feature selection the relationship between each input variable and the target variable is evaluated using statistics, and the input variables with the strongest relationship with the target variable are selected. This is called as analysis of variance (ANOVA).



```
     Specs         Score
17     V17  25474.056179
14     V14  20728.414597
12     V12  14860.008312
10     V10  10200.091855
3       V3   8593.601444
7       V7   8448.985654
16     V16   8375.909431
11     V11   5020.086298
4       V4   3827.794217
18     V18   2728.321421
1       V1   2459.783546
5       V5   2234.754348
9       V9   2011.058289
2       V2   1895.304166
21     V21    482.896219
6       V6    399.364056
19     V19    254.965195
8       V8     80.033876
20     V20     79.181341
0     Time     35.169579
```



```
             Specs         Score
8    out_malicious  3.730937e+07
9     is_malicious  1.619278e+07
10   all_malicious  1.981881e+06
7     in_malicious  1.296211e+03
2           in_btc  3.792878e+02
4        total_btc  3.596737e+02
3          out_btc  3.392067e+02
6     mean_out_btc  3.256418e+02
```

a b

Figure 4.1: Feature Selection for a. Credit Card dataset and b. Bitcoin dataset

RANDOM RESAMPLING

We have implemented data pre-processing, data manipulation and data analysis on the datasets for both Credit Card and Bitcoin Fraud Management. We have obtained the following analytical result. The results in both cases indicate that the non-fraudulent transactions far outweigh the fraudulent transactions. To overcome this issue, we have applied resampling techniques to the training datasets of both sub-sections of the projects.

<u>CREDIT CARD FRAUD MANAGEMENT DATASET</u>

The dataset is highly imbalanced with only 0.173% of the total transactions being fraudulent. Therefore, during the pre-processing, we oversample the minority class (fraudulent transactions) in the training dataset to have 0.25 times the number of examples as the majority class (normal transactions). As a result, the fraudulent transactions account for 15.819% of the entire dataset.

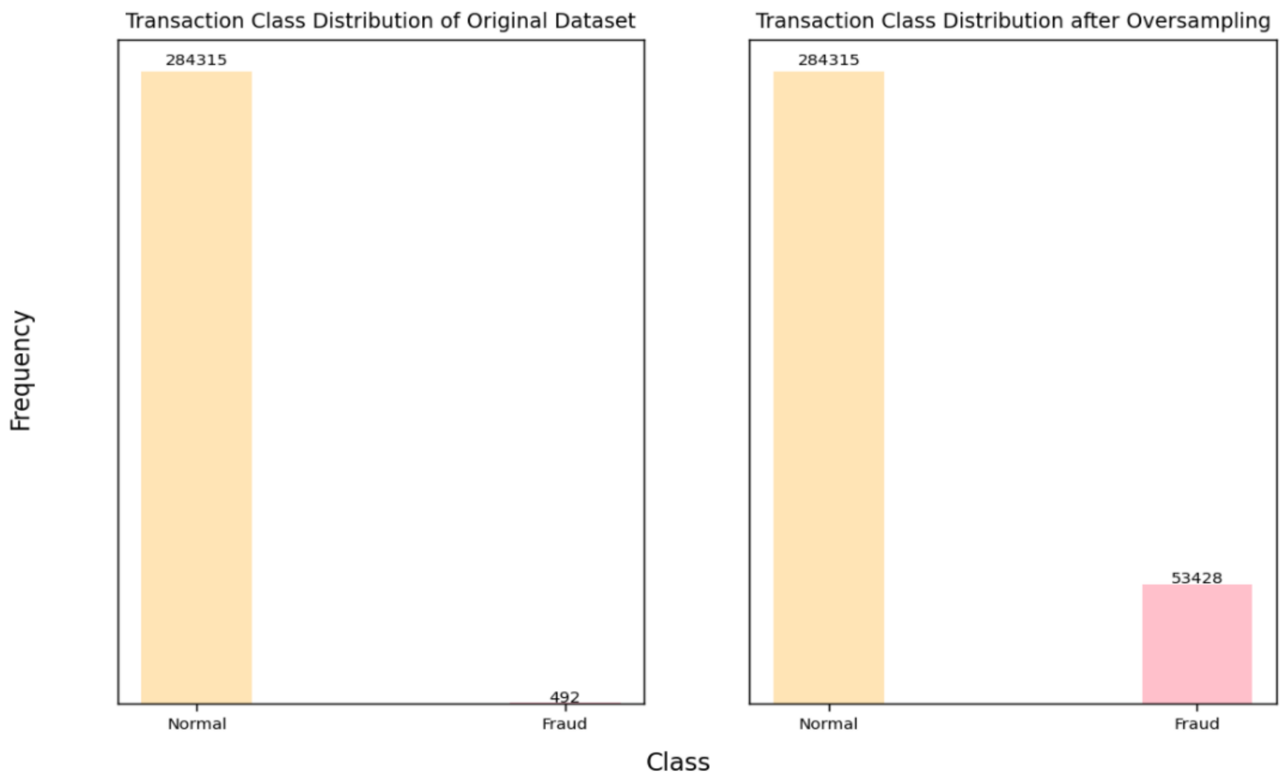


Figure 4.2: Comparison of Transaction Class Distribution before and after Oversampling for Credit Card dataset

Additionally, the dataset is analysed by segregating the values based on the amount values in the transaction for both normal as well as fraud classes. Furthermore, we have analysed the data based on the time taken for the transaction versus the sum of money which was transferred.

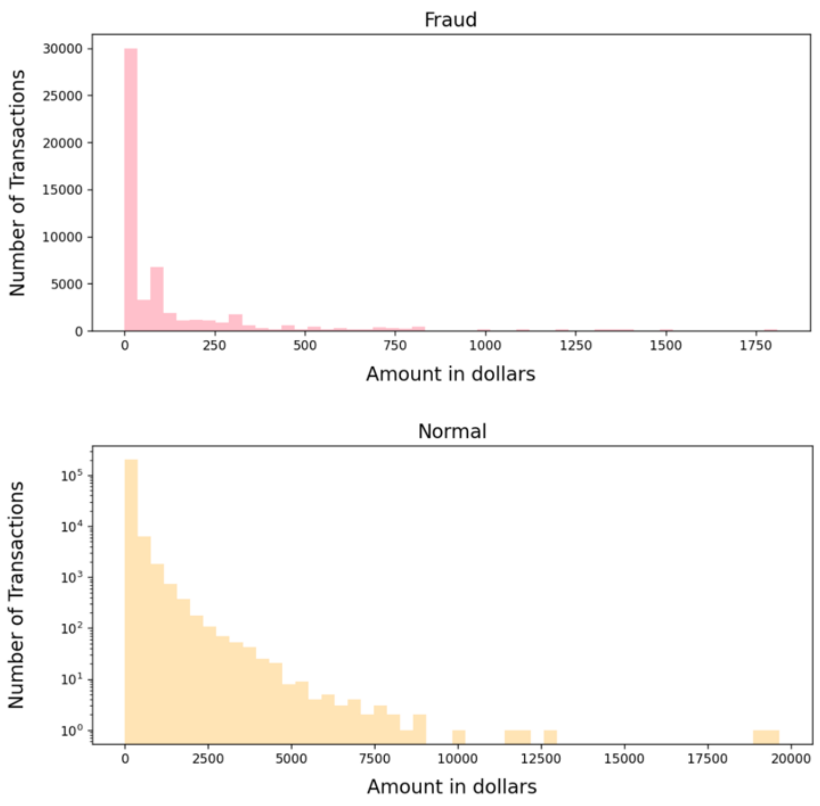


Figure 4.3: Graphical Representation of Amount transferred per transaction by class

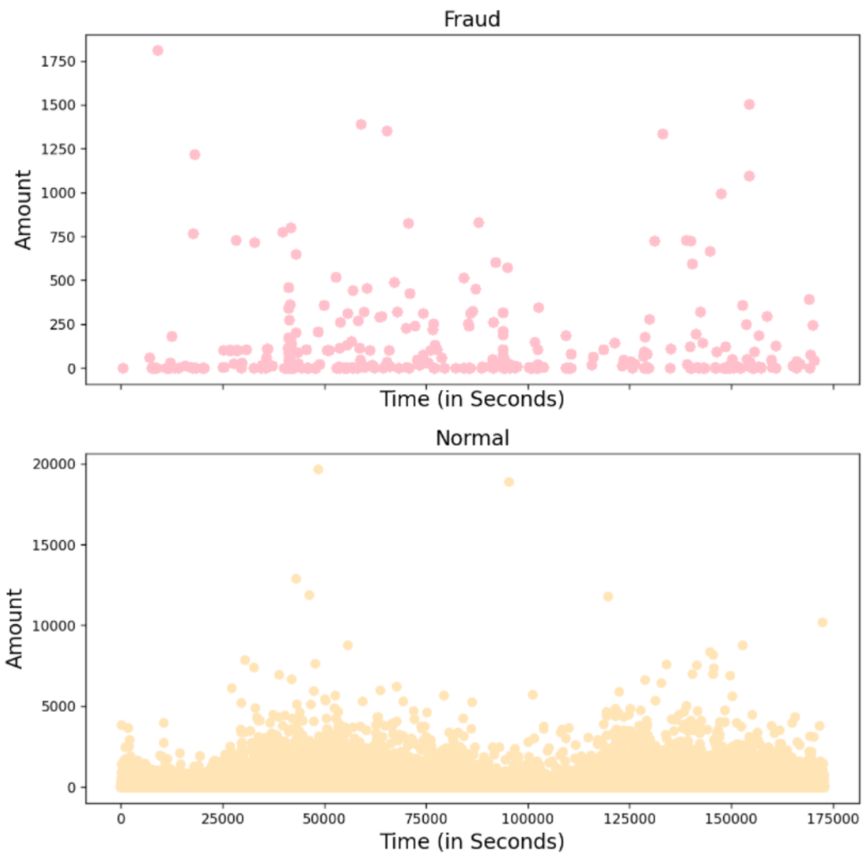


Figure 4.4: Graphical Representation of Time of transaction vs Amount by class

BITCOIN FRAUD MANAGEMENT DATASET

Only 0.00036% of the overall transactions are fraudulent, indicating that the dataset is highly skewed. As a response, during pre-processing, we first undersample the majority class (normal transactions) in the training dataset so that the minority class (fraudulent transactions) is 0.0002 times as the majority class (normal transactions). Secondly, in the training dataset, we oversample the minority class to have 0.05 times the number of examples as the majority class. As a result, fraudulent transactions make up 0.228% of the total data.

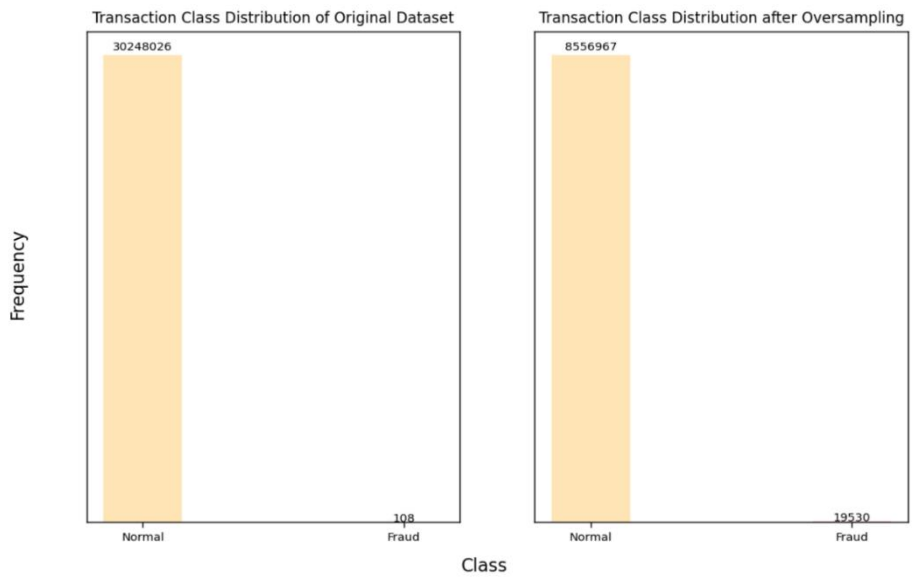


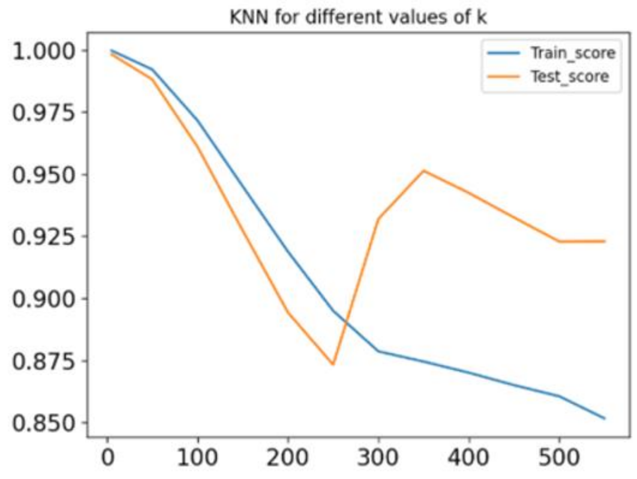
Figure 4.5: Comparison of Transaction Class Distribution before and after Oversampling for Bitcoin dataset

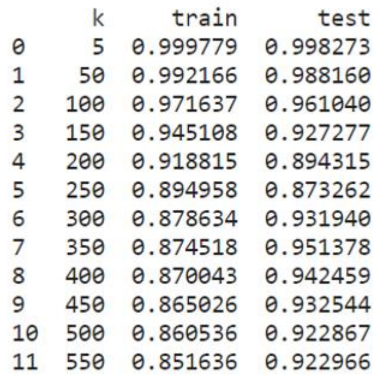
ALGORITHM IMPLEMENTATION

Our project implements three supervised and three unsupervised machine learning models on both the datasets namely, K Nearest Neighbor, Support Vector Machine, Neural Networks, K-means, Isolation Forest and Local Outlier Factor Algorithm.

The general idea for KNN is that as we decrease the value of ‘k’, we increase the risk of overfitting as it increases the variance of the decision boundaries. This is because at lower k values, the decision boundaries are sensitive to noise, outliers etc. leading it to capture even local changes resulting in a complex model. On the other hand, higher k values suppress the effect of noisy individual labels and lead to a better generalization model. However, extremely large values of k could lead to underfitting.

In case of credit card dataset, after experimenting with various values of k, following are the obtained results. As observed, k=5 leads to a decision boundary with high variation and as we increase the k value, we obtain better generalization results. Hence, we opted for our n_neighbors value to be 350.





```
      k     train      test
0     5  0.999779  0.998273
1    50  0.992166  0.988160
2   100  0.971637  0.961040
3   150  0.945108  0.927277
4   200  0.918815  0.894315
5   250  0.894958  0.873262
6   300  0.878634  0.931940
7   350  0.874518  0.951378
8   400  0.870043  0.942459
9   450  0.865026  0.932544
10  500  0.860536  0.922867
11  550  0.851636  0.922966
```

Figure 4.6: Train and Test scores for different values of k in KNN for Credit Card dataset

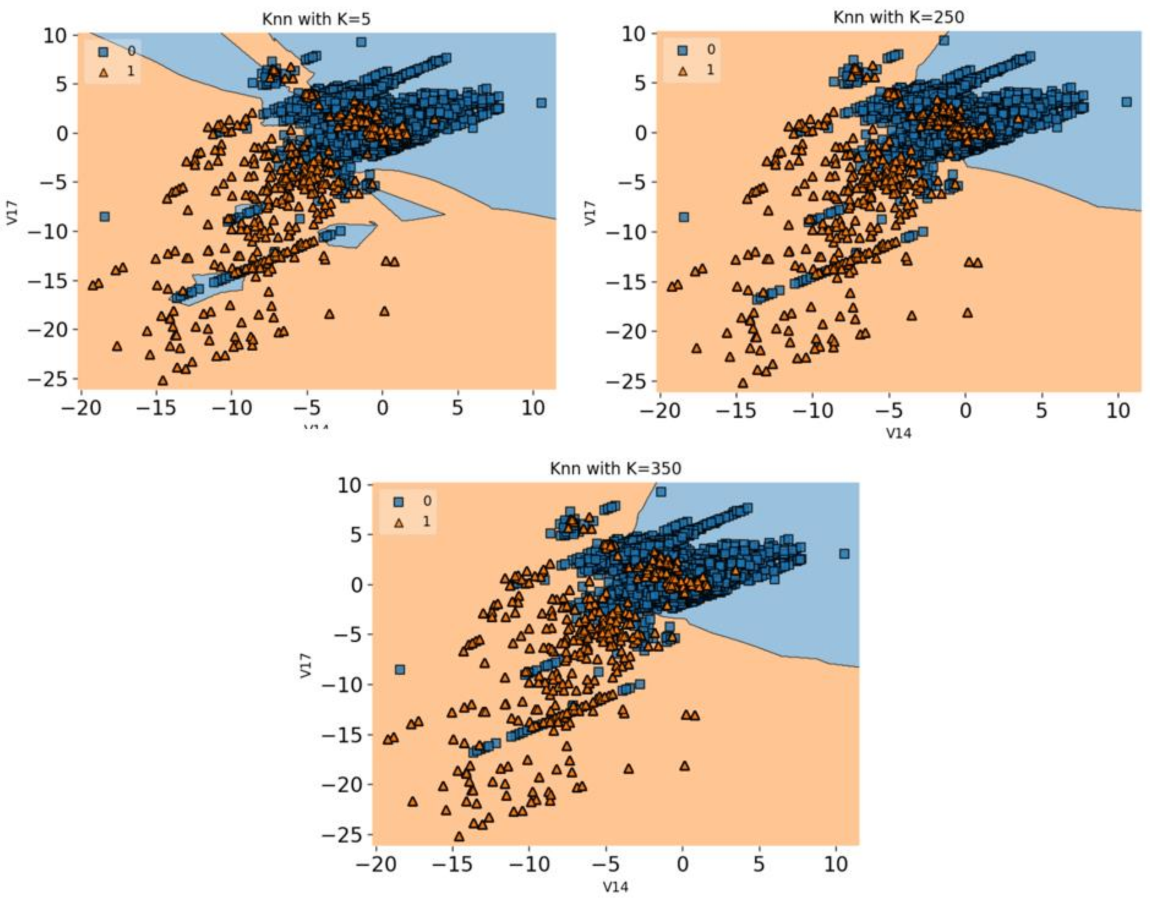
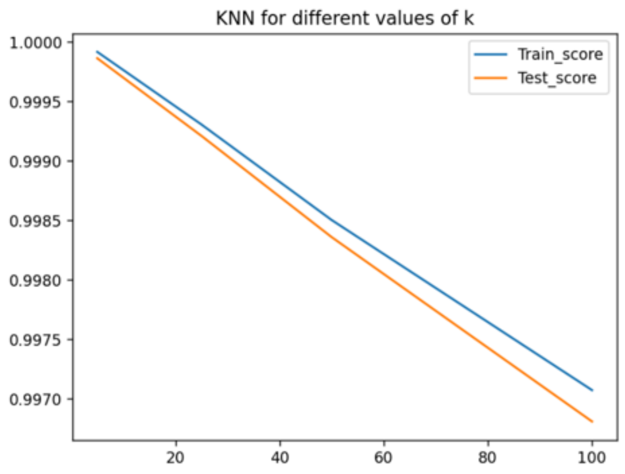
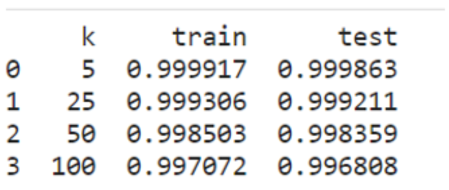


Figure 4.7: Graphical Representation of KNN for different k values for Credit Card dataset

In case of bitcoin dataset, after experimenting with various values of k, following are the obtained results. As observed, with increase in k value, the model doesn’t do well on the training dataset as it captures more non-fraudulent values withing the decision boundary of fraudulent transactions which could lead to an increase in the false positives. Therefore, we opt for k=25 which is able to generalize well and avoid underfitting.





```
     k     train      test
0    5  0.999917  0.999863
1   25  0.999306  0.999211
2   50  0.998503  0.998359
3  100  0.997072  0.996808
```

Figure 4.8: Train and Test scores for different values of k in KNN for bitcoin dataset

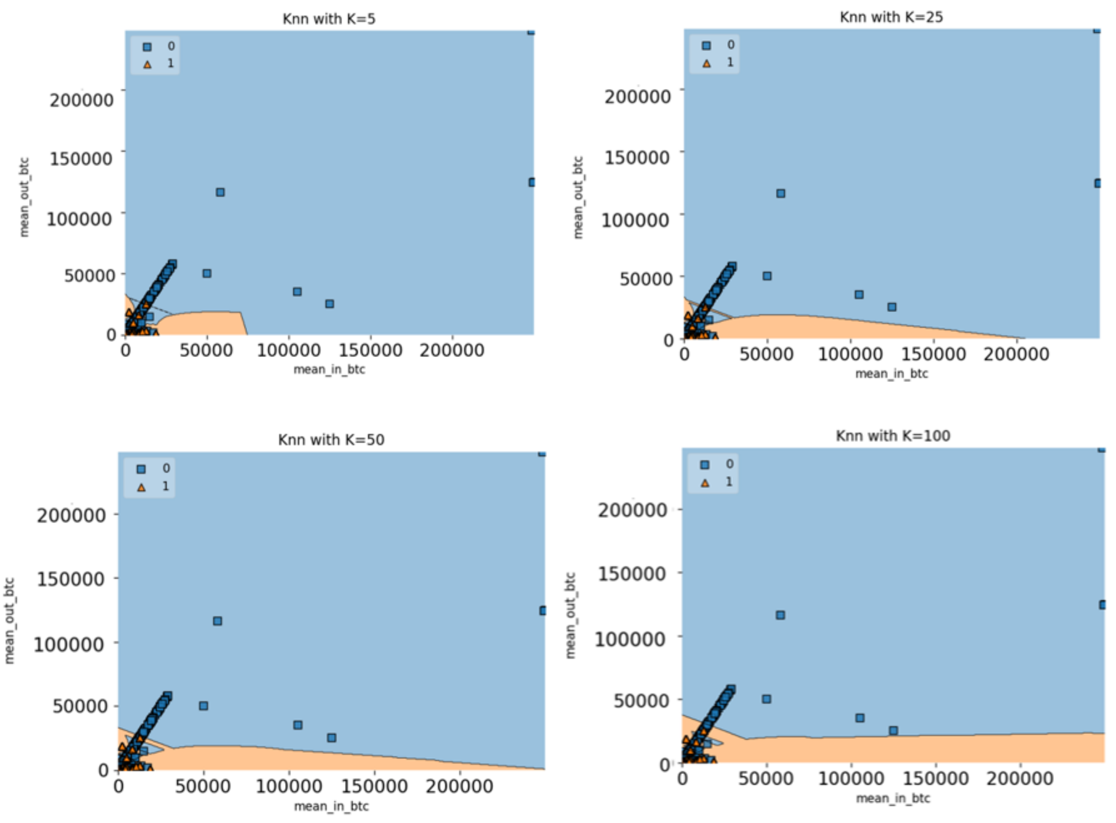


Figure 4.9: Graphical Representation of KNN for different k values for Bitcoin dataset

CHAPTER 5: RESULT

RESULT

The results obtained for both sub-sections of the project comprise following details about each algorithm:

Accuracy: Accuracy is defined as a fraction of all instances for which the classifier's predictions are correct (for either positive or negative class).

$$\text{Accuracy} = \frac{TN+TP}{TN+TP+FN+FP}$$

Recall: Recall, or True Positive Rate (TPR) is defined as a fraction of all positive instances for which the classifier correctly identifies as positive.

$$\text{Recall} = \frac{TP}{TP+FN}$$

Precision: Precision is defined as fraction of positive predictions that are correct.

$$\text{Precision} = \frac{TP}{TP+FP}$$

F1: F1 score combines precision and recall into a single number. Mathematically, it is based on the harmonic mean of precision and recall.

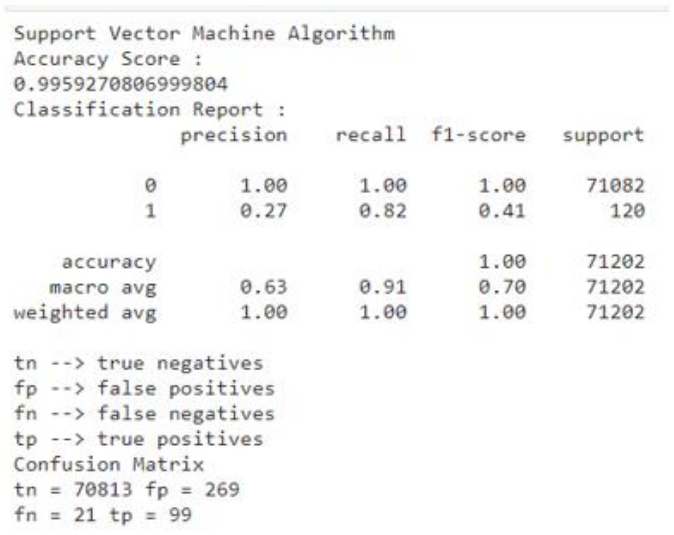
$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Macro average: Macro average computes the average of precision, recall and F1 without considering the proportion for each label in the dataset.

Weighted average: Weighted average computes the average of precision, recall and F1says the function considering the proportion for each label in the dataset.

Confusion Matrix: A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class.

1. CREDIT CARD FRAUD MANAGEMENT



```
Support Vector Machine Algorithm
Accuracy Score :
0.9959270806999804
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71082
           1       0.27      0.82      0.41       120

    accuracy                           1.00     71202
   macro avg       0.63      0.91      0.70     71202
weighted avg       1.00      1.00      1.00     71202

tn --> true negatives
fp --> false positives
fn --> false negatives
tp --> true positives
Confusion Matrix
tn = 70813 fp = 269
fn = 21 tp = 99
```

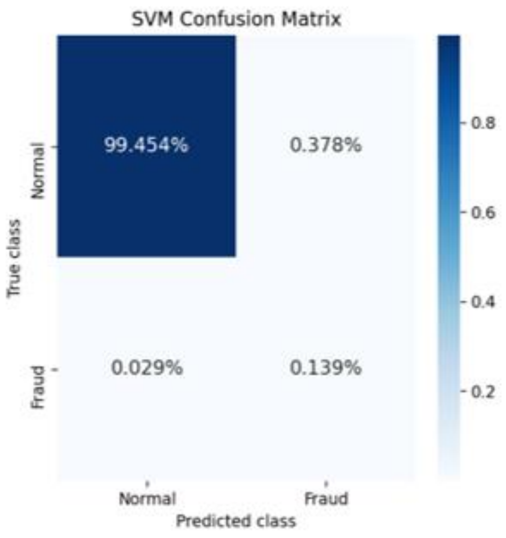
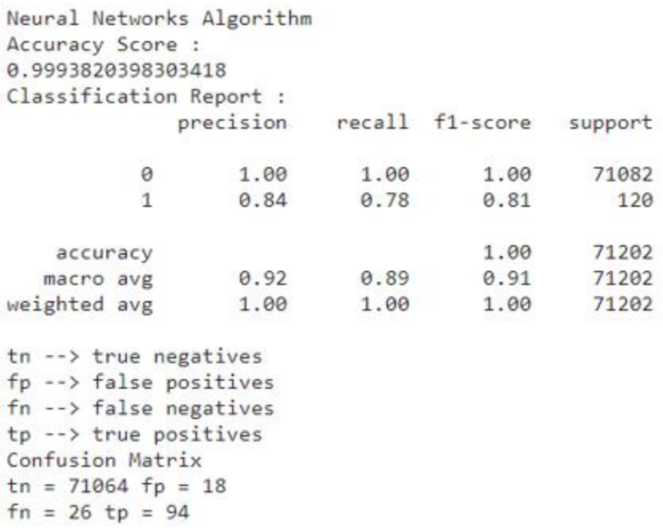


Figure 5.1.1: Classification Report and Confusion Matrix for SVM Algorithm



```
Neural Networks Algorithm
Accuracy Score :
0.9993820398303418
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71082
           1       0.84      0.78      0.81       120

    accuracy                           1.00     71202
   macro avg       0.92      0.89      0.91     71202
weighted avg       1.00      1.00      1.00     71202

tn --> true negatives
fp --> false positives
fn --> false negatives
tp --> true positives
Confusion Matrix
tn = 71064 fp = 18
fn = 26 tp = 94
```

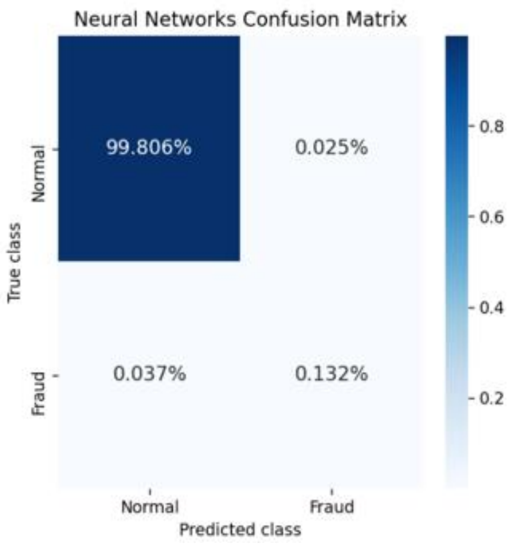
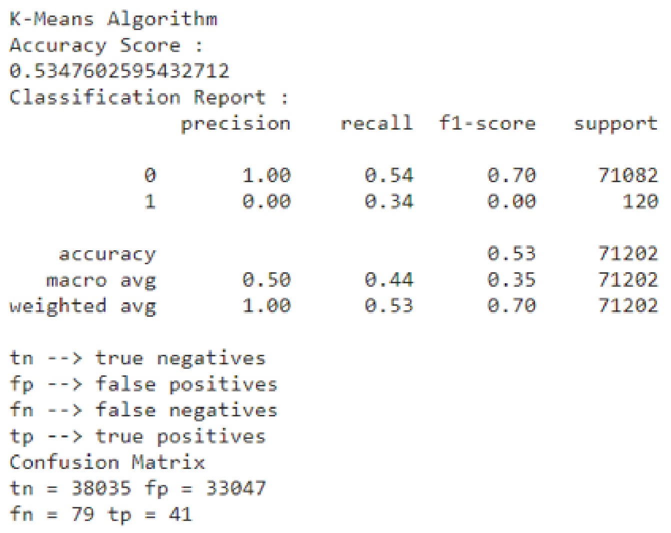


Figure 5.1.2: Classification Report and Confusion Matrix for Neural Networks Algorithm



```
K-Means Algorithm
Accuracy Score :
0.5347602595432712
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.54      0.70     71082
           1       0.00      0.34      0.00       120

    accuracy                           0.53     71202
   macro avg       0.50      0.44      0.35     71202
weighted avg       1.00      0.53      0.70     71202

tn --> true negatives
fp --> false positives
fn --> false negatives
tp --> true positives
Confusion Matrix
tn = 38035 fp = 33047
fn = 79 tp = 41
```

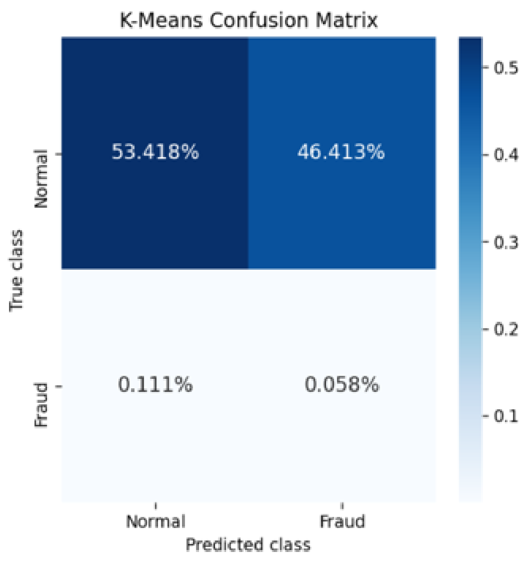


Figure 5.1.3a: Classification Report and Confusion Matrix for K-Means Algorithm

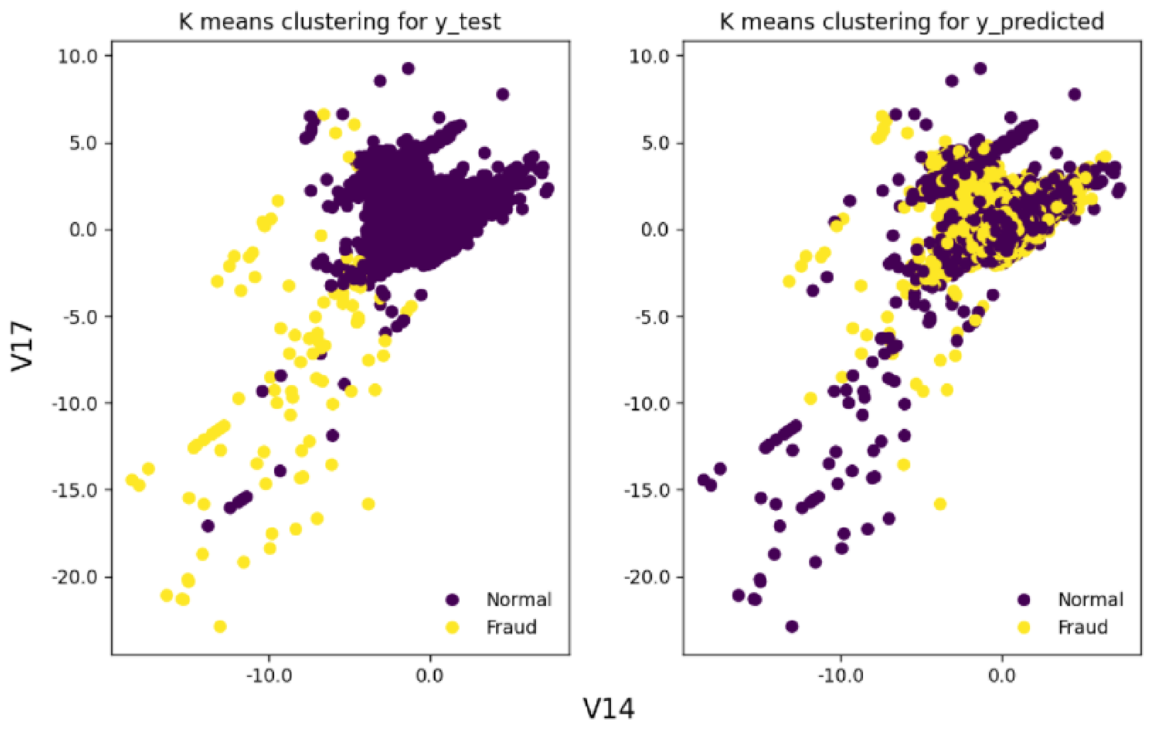
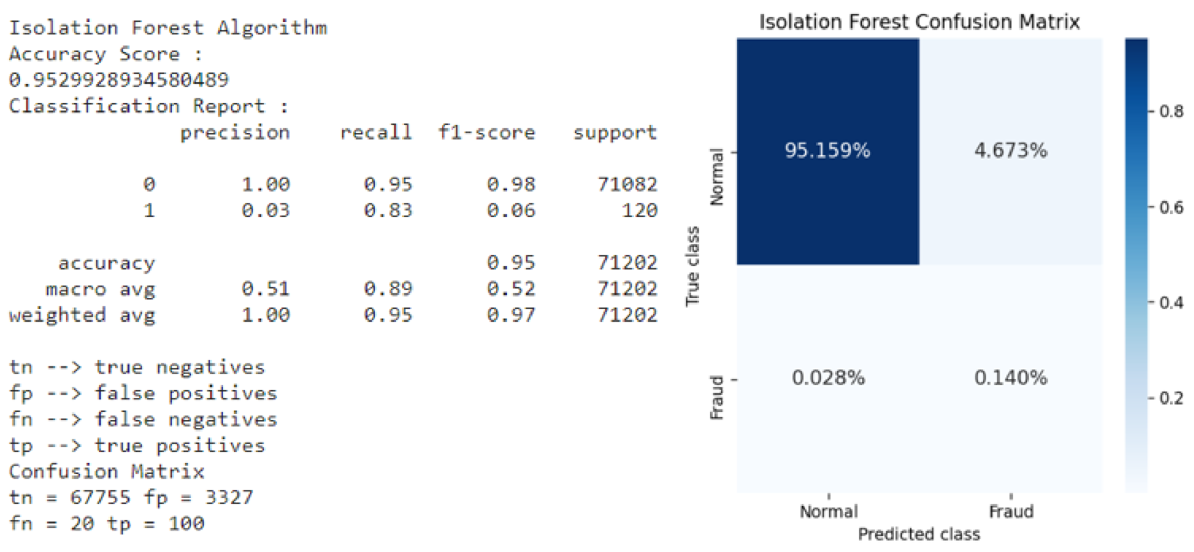


Figure 5.1.3b: K-Means Clustering Representation



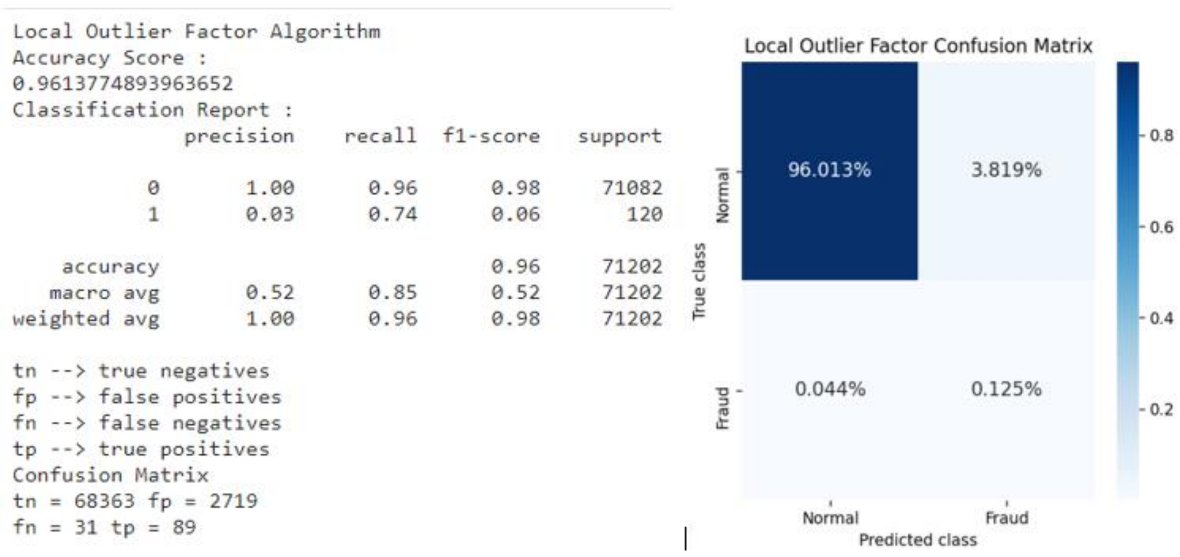
```
Isolation Forest Algorithm
Accuracy Score :
0.9529928934580489
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.95      0.98     71082
           1       0.03      0.83      0.06       120

    accuracy                           0.95     71202
   macro avg       0.51      0.89      0.52     71202
weighted avg       1.00      0.95      0.97     71202

tn --> true negatives
fp --> false positives
fn --> false negatives
tp --> true positives
Confusion Matrix
tn = 67755 fp = 3327
fn = 20 tp = 100
```

Figure 5.1.4: Classification Report and Confusion Matrix for Isolation Forest Algorithm



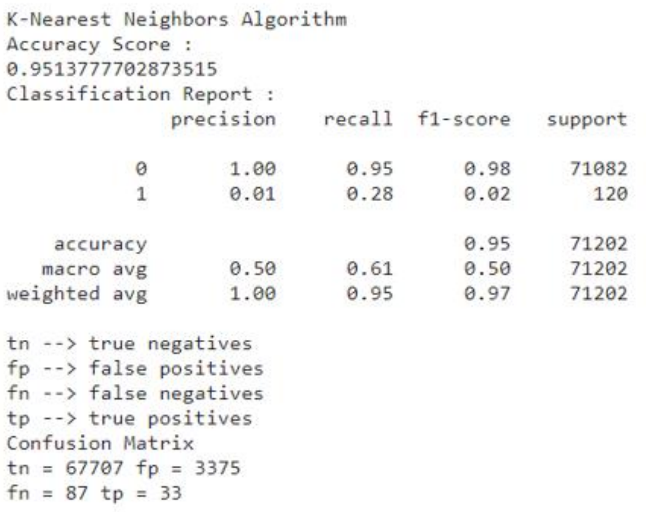
```
Local Outlier Factor Algorithm
Accuracy Score :
0.9613774893963652
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.96      0.98     71082
           1       0.03      0.74      0.06       120

    accuracy                           0.96     71202
   macro avg       0.52      0.85      0.52     71202
weighted avg       1.00      0.96      0.98     71202

tn --> true negatives
fp --> false positives
fn --> false negatives
tp --> true positives
Confusion Matrix
tn = 68363 fp = 2719
fn = 31 tp = 89
```

Figure 5.1.5: Classification Report and Confusion Matrix for Local Outlier Factor Algorithm



```
K-Nearest Neighbors Algorithm
Accuracy Score :
0.9513777702873515
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.95      0.98     71082
           1       0.01      0.28      0.02       120

    accuracy                           0.95     71202
   macro avg       0.50      0.61      0.50     71202
weighted avg       1.00      0.95      0.97     71202

tn --> true negatives
fp --> false positives
fn --> false negatives
tp --> true positives
Confusion Matrix
tn = 67707 fp = 3375
fn = 87 tp = 33
```

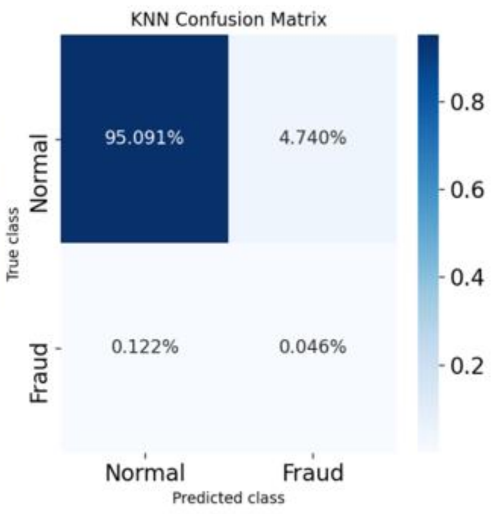
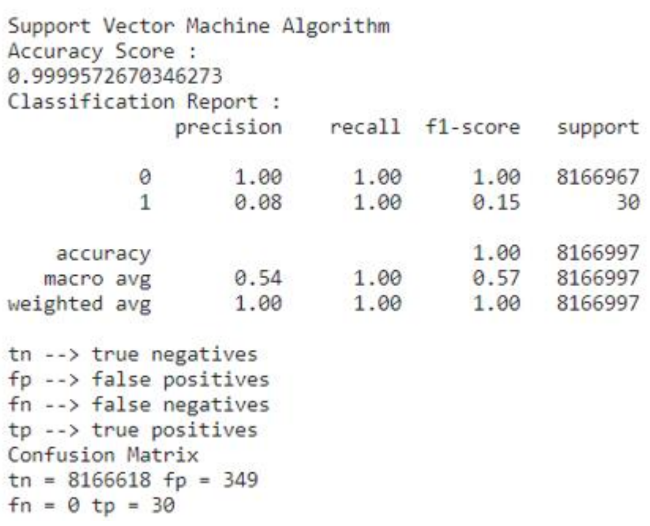


Figure 5.1.6: Classification Report and Confusion Matrix for K-Nearest Neighbors Algorithm

2. BITCOIN FRAUD MANAGEMENT



```
Support Vector Machine Algorithm
Accuracy Score :
0.9999572670346273
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   8166967
           1       0.08      1.00      0.15        30

    accuracy                           1.00   8166997
   macro avg       0.54      1.00      0.57   8166997
weighted avg       1.00      1.00      1.00   8166997

tn --> true negatives
fp --> false positives
fn --> false negatives
tp --> true positives
Confusion Matrix
tn = 8166618 fp = 349
fn = 0 tp = 30
```

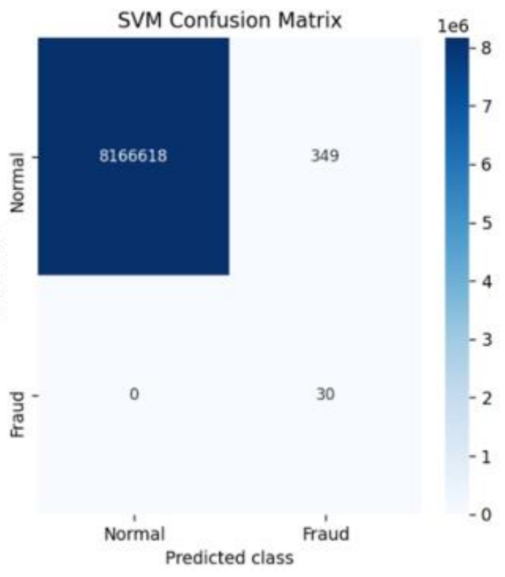
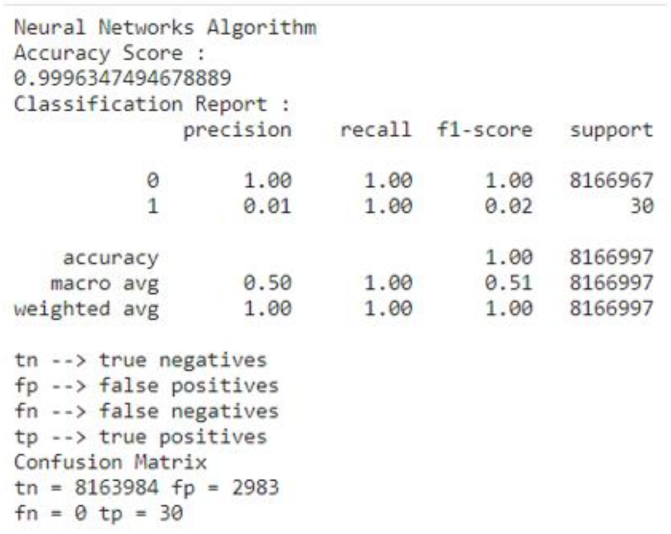


Figure 5.2.1: Classification Report and Confusion Matrix for SVM Algorithm



```
Neural Networks Algorithm
Accuracy Score :
0.9996347494678889
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   8166967
           1       0.01      1.00      0.02        30

    accuracy                           1.00   8166997
   macro avg       0.50      1.00      0.51   8166997
weighted avg       1.00      1.00      1.00   8166997

tn --> true negatives
fp --> false positives
fn --> false negatives
tp --> true positives
Confusion Matrix
tn = 8163984 fp = 2983
fn = 0 tp = 30
```

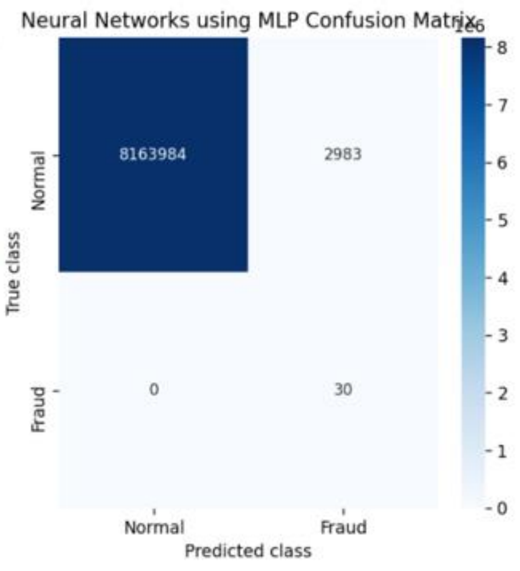
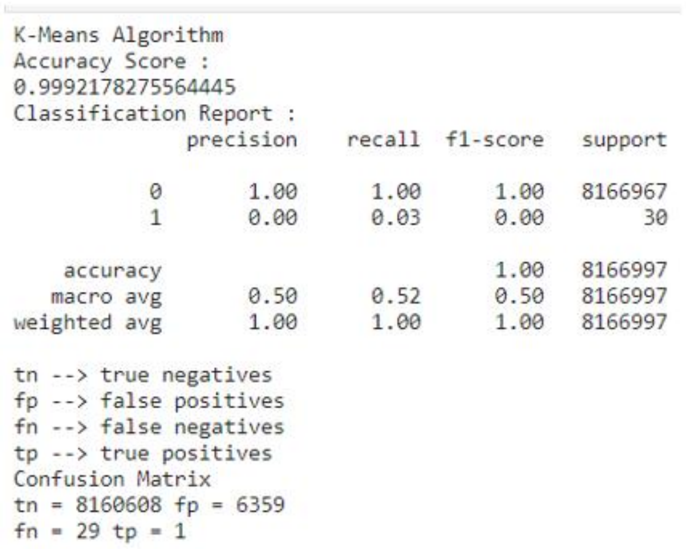


Figure 5.2.2: Classification Report and Confusion Matrix for Neural Networks Algorithm



```
K-Means Algorithm
Accuracy Score :
0.9992178275564445
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   8166967
           1       0.00      0.03      0.00        30

    accuracy                           1.00   8166997
   macro avg       0.50      0.52      0.50   8166997
weighted avg       1.00      1.00      1.00   8166997

tn --> true negatives
fp --> false positives
fn --> false negatives
tp --> true positives
Confusion Matrix
tn = 8160608 fp = 6359
fn = 29 tp = 1
```

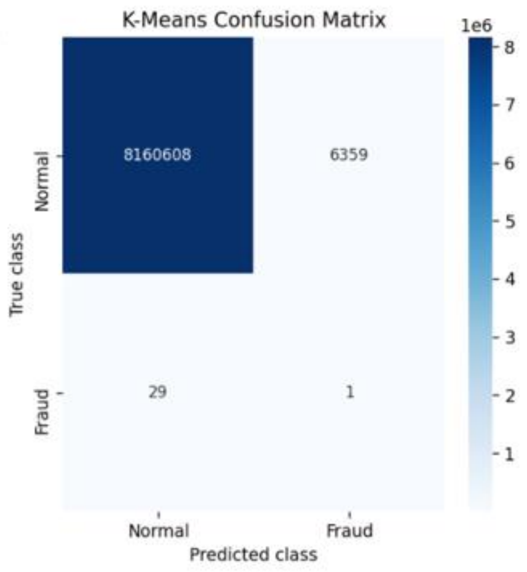


Figure 5.2.3a: Classification Report and Confusion Matrix for K-Means Algorithm

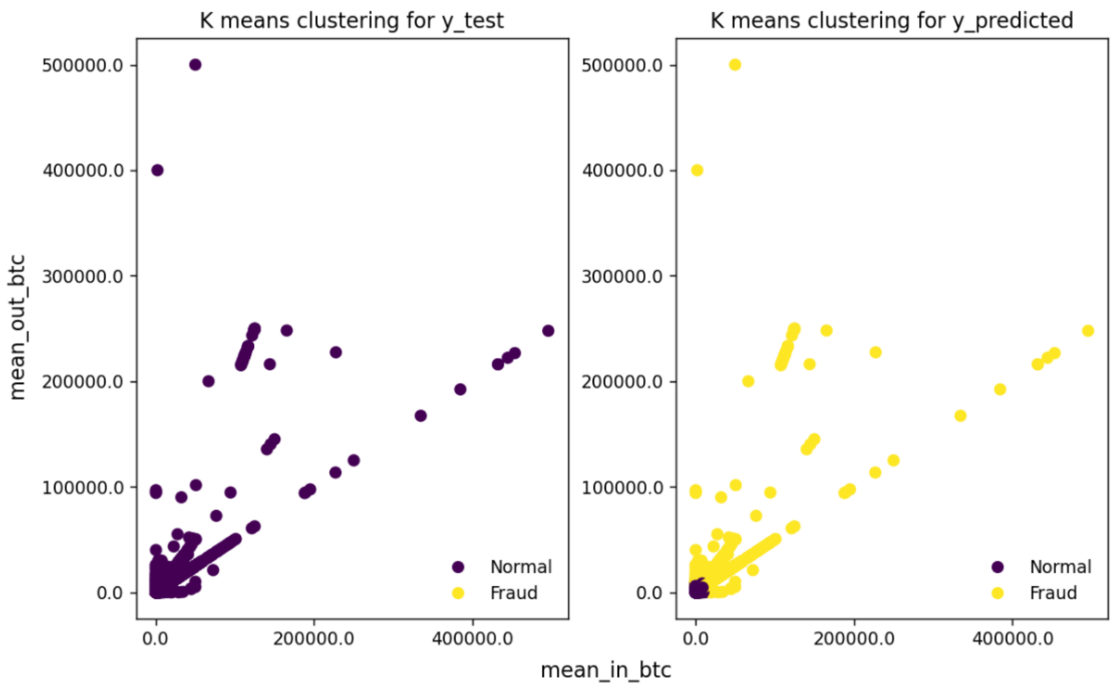
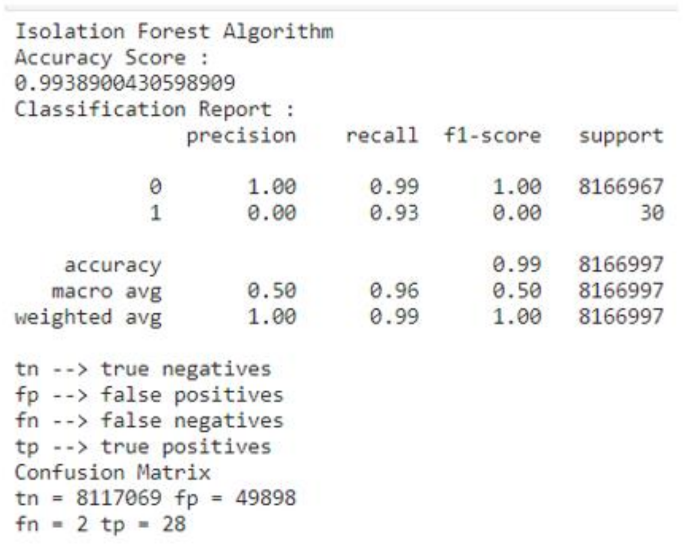


Figure 5.2.3b: K-Means Clustering Representation



```
Isolation Forest Algorithm
Accuracy Score :
0.9938900430598909
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.99      1.00   8166967
           1       0.00      0.93      0.00        30

    accuracy                           0.99   8166997
   macro avg       0.50      0.96      0.50   8166997
weighted avg       1.00      0.99      1.00   8166997

tn --> true negatives
fp --> false positives
fn --> false negatives
tp --> true positives
Confusion Matrix
tn = 8117069 fp = 49898
fn = 2 tp = 28
```

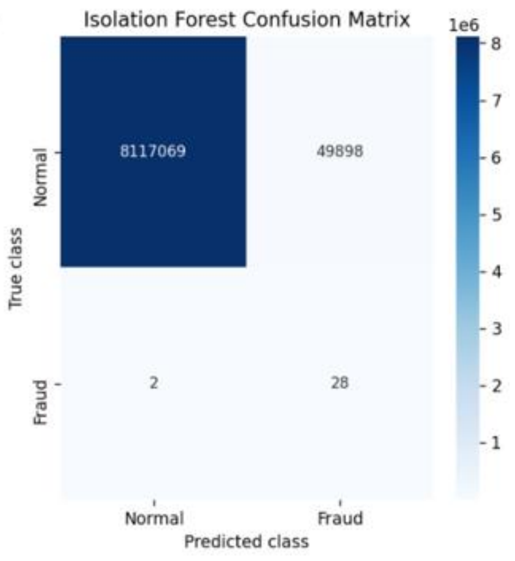
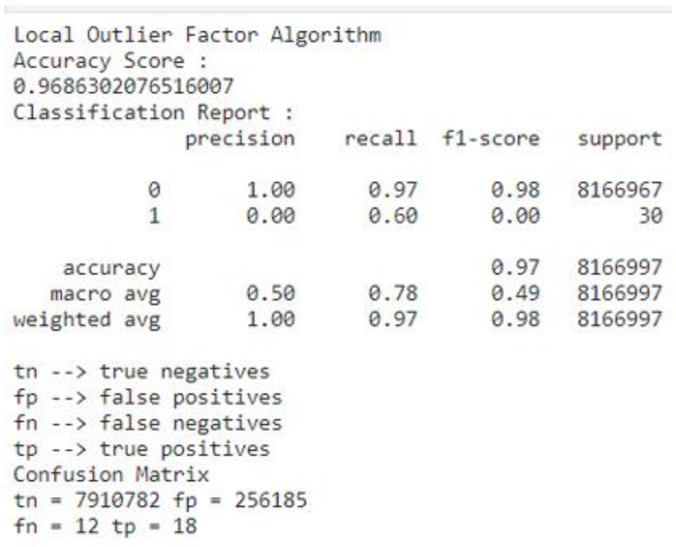


Figure 5.2.4: Classification Report and Confusion Matrix for Isolation Forest Algorithm



```
Local Outlier Factor Algorithm
Accuracy Score :
0.9686302076516007
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.97      0.98   8166967
           1       0.00      0.60      0.00        30

    accuracy                           0.97   8166997
   macro avg       0.50      0.78      0.49   8166997
weighted avg       1.00      0.97      0.98   8166997

tn --> true negatives
fp --> false positives
fn --> false negatives
tp --> true positives
Confusion Matrix
tn = 7910782 fp = 256185
fn = 12 tp = 18
```

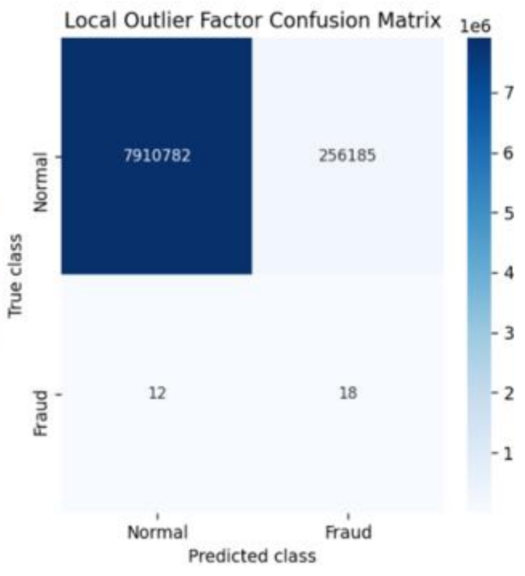
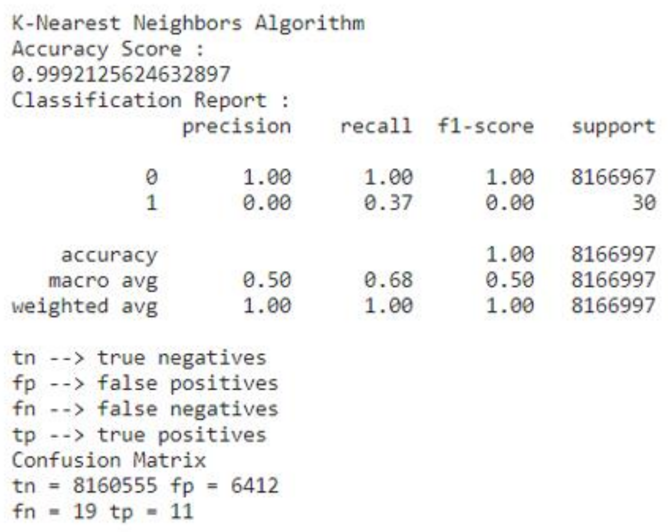


Figure 5.2.5: Classification Report and Confusion Matrix for Local Outlier Factor Algorithm



```
K-Nearest Neighbors Algorithm
Accuracy Score :
0.9992125624632897
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   8166967
           1       0.00      0.37      0.00        30

    accuracy                           1.00   8166997
   macro avg       0.50      0.68      0.50   8166997
weighted avg       1.00      1.00      1.00   8166997

tn --> true negatives
fp --> false positives
fn --> false negatives
tp --> true positives
Confusion Matrix
tn = 8160555 fp = 6412
fn = 19 tp = 11
```

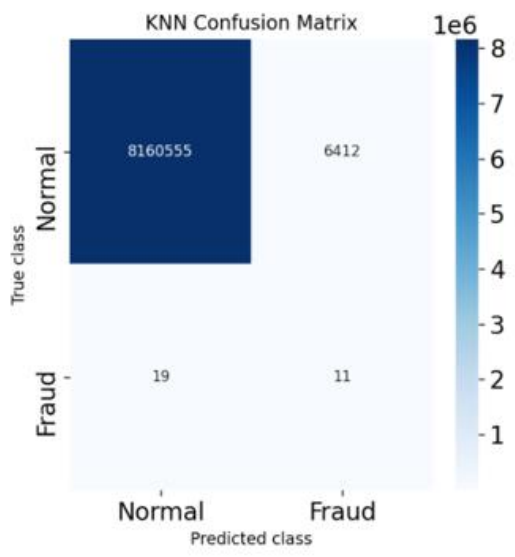


Figure 5.2.6: Classification Report and Confusion Matrix for K-Nearest Neighbor Algorithm

When a dataset is highly imbalanced, macro average is preferred for assessing the results of the performed machine learning models. This is because it formulates the average without using weights for aggregation which results in greater penalization when the model doesn’t do well with the minority class (fraudulent transactions), which is desired in case of an imbalanced dataset. On the other hand, weighted average would lead in the favouring of the majority class (normal transactions).

The computed results are:

1. CREDIT CARD FRAUD MANAGEMENT

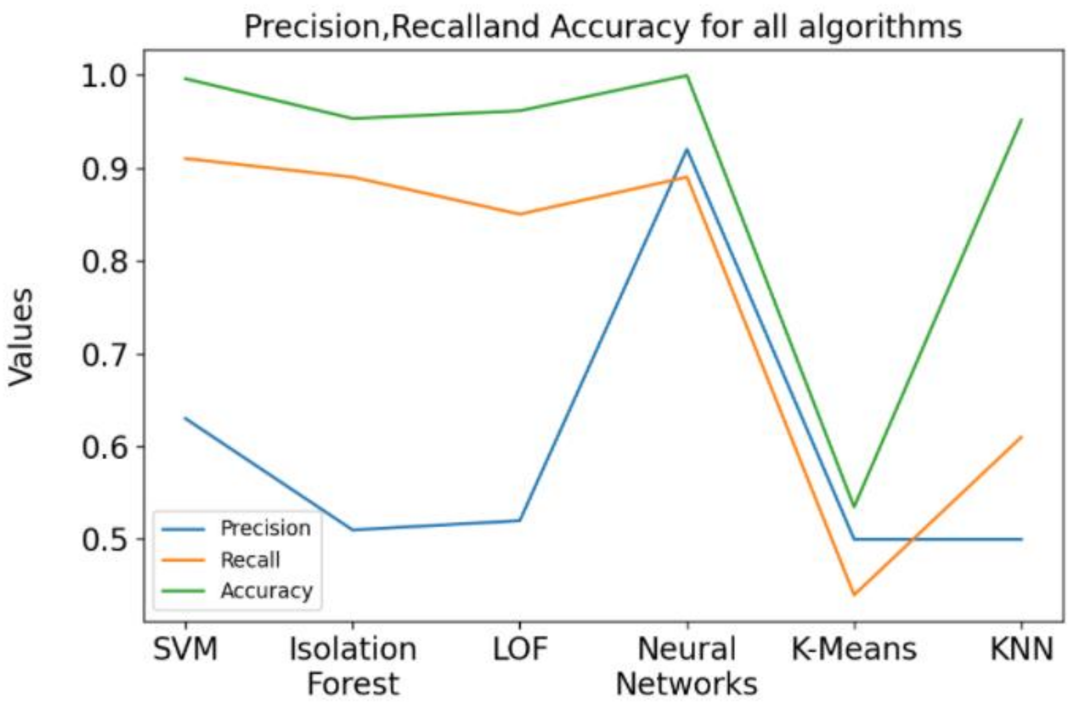


Figure 5.1.7: Precision, Recall and Accuracy for all Algorithms

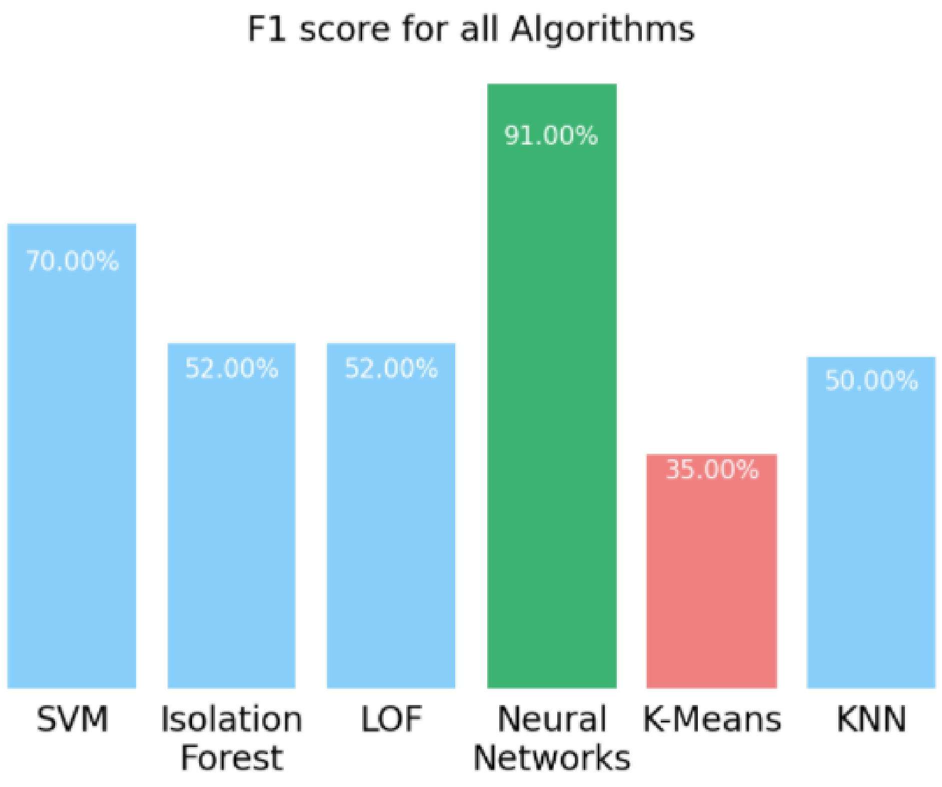


Figure 5.1.8: F1 score for all Algoritms

2. BITCOIN FRAUD MANAGEMENT

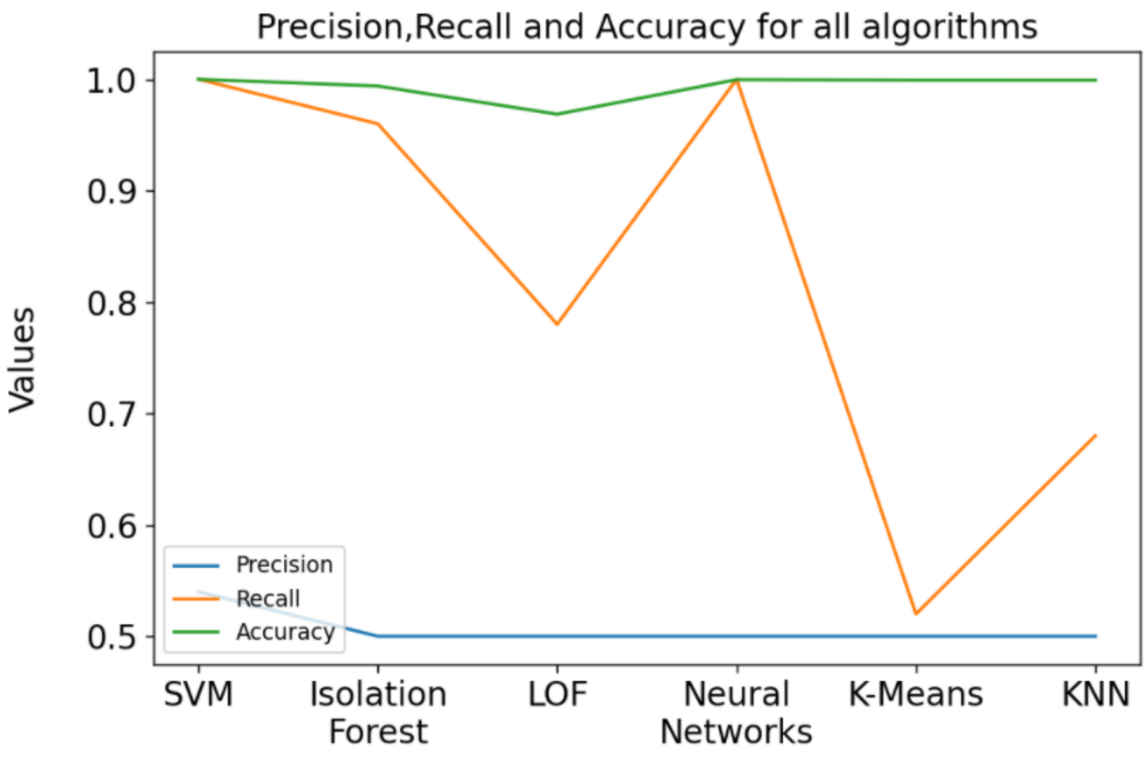


Figure 5.2.7: Precision, Recall and Accuracy for all Algorithms

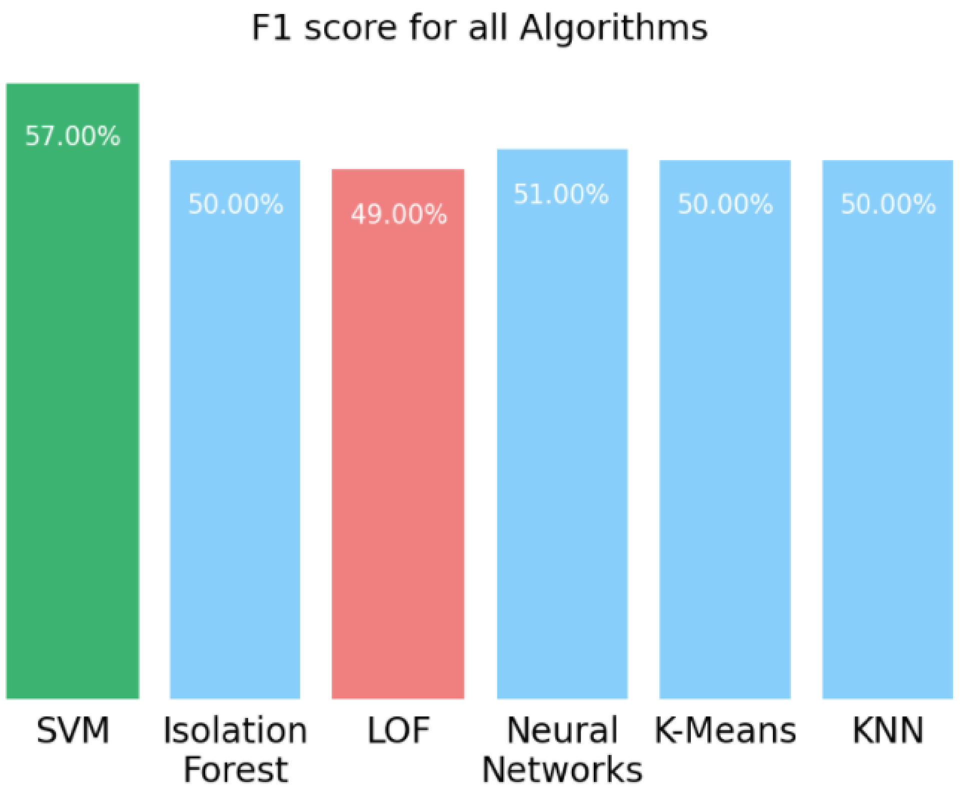


Figure 5.2.8: F1 score for all Algorithms

CHAPTER 6: CONCLUSION

CONCLUSION

In this project we have studied and implemented various supervised and unsupervised machine learning algorithms to predict and classify transactions as fraudulent and non-fraudulent for credit card and bitcoin datasets. We have reduced the imbalance in the datasets using random resampling techniques. Our aim, to obtain a model for both sub-sections which provided us with optimized results in terms of precision, recall and F1-score has been accomplished. As seen from the results above we found that for Credit Card transactions, the Neural Network model achieved optimum results whereas K-Means model gave unsatisfactory results. Similarly, for Bitcoin transactions, the Support Vector Machine model provided finest results and Local Outlier Factor model produced poor outcomes.

CHAPTER 7: APPLICATION AND FUTURE SCOPE

APPLICATION

The most common applications of fraud detection are as follows:

1. **Insurance claims analysis for fraud detection**
   Insurance companies spend several days to weeks assessing a claim, but the insurance business is still affected by scams. The most common issues are property damage, car insurance scams, and fake unemployment claims. The ticket to successful detection is a good dataset and carefully selected models.
   Fake claims. Semantic analysis is a machine learning task that allows for analyzing both structured, table-type data, and unstructured texts. The feature helps detect fake and falsified claims in the insurance industry. For example, it improves car insurance claims processing. Machine learning algorithms analyze files written by insurance agents, police, and clients, searching for inconsistencies in provided evidence. There are many hidden clues in these textual datasets. The rule-based engines don't catch the suspicious correlations in textual data, and fraud analysts can easily miss important evidence in boring investigation files. That's why analyzing claims is one of the most promising spheres for machine learning applications.

2. **Anti-fraud solutions for medical claims and healthcare**
   Healthcare and medical insurance is a rich area for fraud schemes due to a complex and bureaucratic process, which requires many approvals, verifications, and other paperwork. The most common scams are fake claims that use false or invalid social security numbers, claims duplication, billing for medically unnecessary tests, fake diagnosis, etc. Both hospitals and insurance companies are suffering from these issues. Insurance carriers lose money and hospitals take risks being involved in serious crimes, like drug turnover. Multiple data analytics approaches can mitigate such fraud risks.
   Upcoding and abuse scams. Upcoding is a specific type of fraud where a healthcare provider or a healthcare worker tries to charge more to a patient or an insurance company. The first step in this case is the implementation of digital analysis based on Benford's Law. It helps reveal unexpected digits in a dataset. The technique detects upcoding of procedures and other abuse attempts.
   Machine learning isn't typically required in this case, but it may augment rule-based fraud detection. For example, image recognition techniques can be applied to digitalize paper documentation for further analysis.
   Medical receipts and bills. Adding up numerical values helps check sequences of receipts for legal restrictions and find suspicious links between a physician and a patient (or the group of patients) overstating the total limitation for some drugs. Insurance companies can also benefit from this type of analytics by regularly performing reconciliations of bills and thus preventing fake totals.

3. **Fraud prevention solutions in eCommerce**
   The eCommerce scam is closely linked to payments. So, let's discuss the aspects related purely to eCommerce. Typical scams here are identity theft and merchant scams.

Identity theft. This entails a situation in which a scammer breaches a user account, alters personal data, and tries to get money or goods from a retailer using this semi-fake personal information. Nearly all fraud detection tasks here are solved by behavior analytics. Smart algorithms uncover suspicious activities, analyze them, and find inconsistencies in the historical sets of personal data.

Merchant scams. These are related to fraudulent companies or merchants operating through marketplaces. Customers' choices on marketplaces are often driven by reviews. Some fraudsters create fake reviews for their accounts to attract customers. Machine learning algorithms can eliminate the influence of such fraudsters through conducting sentimental and behaviour analytics and detecting suspicious activities linked to merchants or their products.

4. **Fraud detection in banking and credit card payments**

Payments are the most digitalized part of the financial industry, which makes them particularly vulnerable to digital fraudulent activities. The rise of mobile payments and the competition for the best customer experience nudge banks to reduce the number of verification stages. This leads to lower efficiency of the rule-based approach. So, banks and payment companies switch to data analytics, machine learning, and AI-driven methods.

Modern fraud detection system solve a wide range of analytical problems to uncover all scams in the payment streams.

Data credibility assessment. Gap analytics help identify missing values in sequences of transactions. Machine learning algorithms can reconcile paper documents and system data eliminating the human factor. This ensures data credibility by finding gaps in it and verifying personal details via public sources and transactions history.

Duplicate transactions. A common scam method is creating transactions close to original or making a copy of a transaction. For instance, a company tries to charge a counterpart twice with the same invoice by sending it to different branches.

5. **Preventing loan application fraud**

Lending is sensitive to scams that abuse personal information. Just a decade ago, it was difficult for fraudsters to get access to IDs, photos, addresses, and mobile phone numbers. Today, nearly all data can be found in social networks or elsewhere on the Internet. This makes the life of financial institutions more difficult. Fraudsters become smarter, and loan applications require more rigorous assessment, while clients want to get money as soon as possible.

Personal details counterfeiting. A common type of fraud is often based on providing false personal information. Scammers provide personal details with a number of misspelling issues or misrepresentations of income or credit qualifications. It makes debt collection more difficult.

6. **Advanced fraud detection systems**
   Advanced systems aren't limited to finding anomalies but, in many cases, can recognize existing patterns that signal specific fraud scenarios. There are two types of machine learning approaches that are commonly used in anti-fraud systems: *unsupervised* and *supervised machine learning*. They can be used independently or be combined to build more sophisticated anomaly detection algorithms.

7. **Machine learning for anti-money laundering**
   Regulators, banks, and investment firms are often involved in monitoring possible money laundering activity: They must detect and inform each other about suspicious activities.
   For instance, Dr. Miguel Agustín Villalobos and Dr. Eliud Silva describe a case in which a machine learning model was trained on the dataset with transactions conducted by criminals. Such a model combined with the rule-based approach helps discover hidden relations between money movement and criminal activities.
   Systems of this type can minimize the workload of small and medium banks involved into monitoring. The described solution revealed 99.6 percent of money laundering transactions and minimized the number of reported transactions from 30 to 1 percent.

FUTURE SCOPE

- Since fraud patterns change over time, they could be hard to catch. Therefore, new and updated datasets would be required and machine learning models would be needed to be retrained.
- More room for improvement can be found in the F1 of the algorithms if we could obtain a more balanced dataset with more fraudulent transactions.
- The precision of the algorithms is directly proportional to the size of dataset. Hence, more labelled datasets from reliable resources will surely make the model more accurate in detecting frauds and reduce the number of false positives.
- Anomaly detection within blockchains can improve in the future if certain elements progress, such as the availability of relevant and rich data.
- Feature engineering and approaching the problem from another prospect such as time-series or network analysis can unlock new potential solutions.
- Additionally, distributed computing can be utilized to enhance algorithm's computing capabilities.

REFERENCES

1. P. Raghavan and N. E. Gayar, "Fraud Detection using Machine Learning and Deep Learning," 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, 2019, pp. 334-339, doi: 10.1109/ICCIKE47802.2019.9004231.
2. Dejan Varmedja, Mirjana Karanovic, Srdjan Sladojevic, Marko Arsenovic, Andras Anderla. "Credit Card Fraud Detection - Machine Learning methods", 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH), 2019
3. Maes, Sam, et al. "Credit card fraud detection using Bayesian and neural networks." Proceedings of the 1st international naiso congress on neuro fuzzy technologies. 2002.
4. Machine Learning Group, "Credit Card Fraud Detection, " Kaggle,23-Mar-2018.
5. Malini, N. & Pushpa, M.. (2017). Analysis on credit card fraud identification techniques based on KNN and outlier detection. 255-258. 10.1109/AEEICB.2017.7972424.
6. John, Hyder & Naaz, Sameena. (2019). Credit Card Fraud Detection using Local Outlier Factor and Isolation Forest. International Journal of Computer Sciences and Engineering. 7. 1060-1064. 10.26438/ijcse/v7i4.10601064.
7. V. Ceronmani Sharmila, K. K. R., S. R., S. D. and H. R.,"Credit Card Fraud Detection Using Anomaly Techniques," 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), Chennai, India, 2019, pp. 1-6, doi: 10.1109/ICIICT1.2019.8741421.
8. Omer Shafiq, "Bitcoin Network Transactional Metadata," Kaggle,2020.
9. Bolton, Richard J, David J Hand, et al. (2001). \Unsupervised pro_ling methods for fraud detection". In: Credit Scoring and Credit Control VII, pp. 235-255.
10. Breiman, Leo (2001). \Random forests". In: Machine learning 45.1, pp. 5-32.
11. Breunig, Markus M et al. (2000). \LOF: identifying density-based local outliers". In: ACM sigmod record. Vol. 29. 2. ACM, pp. 93-104.
12. Brito, Jerry and Andrea Castillo (2013). \Bitcoin: A primer for policymakers". In: Policy: A Journal of Public Policy and Ideas 29.4, pp. 3-12.
13. Vlachos, Michail, George Kollios, and Dimitrios Gunopulos (2002). \Discovering similar multidimensional trajectories". In: Data Engineering, 2002. Proceedings. 18th International Conference on. IEEE, pp. 673-684.
14. Reid, Fergal and Martin Harrigan (2011). \An analysis of anonymity in the bitcoin system". In: Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on. IEEE, pp. 1318-1326.| (2013). \An analysis of anonymity in the bitcoin system". In: Security and privacyin social networks. Springer, pp. 197-223.
15. Abe, N., Zadrozny, B., and Langford, J. 2006. Outlier detection by active learning. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM Press, New York, 504--509.
16. Aggarwal, C. C. and Yu, P. S. 2008. Outlier detection with uncertain data. In Proceedings of the International Conference on Data Mining (SDM). 483--493.

APPENDIX

TECHNICAL PAPER

Publication Name:

©IETE-SF Department of Electronics and Telecommunication Engineering, DJSCE

DJ SPARK 2021

ISBN: 978-93-5437-739-6

TITLE:

“Credit Card Fraud Management Using Machine Learning and Deep Learning Techniques”