A PROJECT REPORT ON

CREDIT CARD FRAUDULENT DETECTION

Submitted in partial fulfilment for the award of degree

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted By

Navya Gangidi	**1602-17-735-030**
K. Madhu Sai Kalyan	**1602-17-735-024**
S. Dhvijesh	**1602-17-735-015**

Under the esteemed guidance of

MRS. CH. NEETU

Assistant Professor

Department Of ECE

Department of Electronics and Communication Engineering

VASAVI COLLEGE OF ENGINEERING (Autonomous)

(Affiliated to Osmania University)

Accredited by NAAC with ‘A++’ Grade

Ibrahimbagh, Hyderabad-500031

2017-2021

Department of Electronics and Communication Engineering

VASAVI COLLEGE OF ENGINEERING

(AUTONOMOUS)

HYDERABAD

CERTIFICATE

This is to certify that this project report entitled "***Credit Card Fraudulent Detection***" by Navya Gangidi (H.T.No:1602-17-735-030), K. Madhu Sai Kalyan (H.T.No:1602-17-735-024) and S. Dhvijesh (H.T.No:1602-17-735-015) submitted in partial fulfillment of the requirements for the degree of BACHELOR OF ENGINEERING in Electronics and Communication Engineering in Vasavi College of Engineering, Hyderabad, during the academic year 2020-2021, is a bonafide record of work carried out under my guidance and supervision.

Head of the Department	**External Examiner**	**Internal guide**
DR.E.SREENIVASA RAO		Mrs.CH.NEETHU
Prof. & HOD, ECE		Associate professor, ECE
VCE (A), Hyderabad		VCE (A), Hyderabad

DECLARATION

*We at this moment declare that the work presented in this project report entitled "**Credit Card Fraudulent Detection**" submitted in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Electronics and Communication Engineering, Vasavi College of Engineering, Hyderabad, is an authentic record of our work carried out during the year 2020-2021 under the guidance and supervision of **Mrs. Ch. Neetu, Asst. Prof., Dept. of E.C.E** and have not submitted for the award of any other degree.*

This report's findings have not been submitted to any other university or institute for the purpose of conferring a degree or certificate. If plagiarism is discovered, we will be held entirely accountable.

Signature:	*Signature:*
Name: ***Navya Gangidi***	*Name:* ***K. Madhu Sai Kalyan***
Roll No: 1602-17-735-030	*Roll No: 1602-17-735-024*

Signature:

Name: ***S. Dhvijesh***

Roll No: 1602-17-735-015

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our guide ***Mrs. Ch. Neetu, Asst. Professor, Department of Electronics and Communication Engineering****, for her invaluable counsel and help. She was a huge inspiration for us to work on this project. Her ability to encourage us was really beneficial to our endeavor.*

We express our earnest gratitude and venerable regards to our project coordinators ***Mrs. G.R. Padmini, Assoc. Prof. , Mrs. A. Sri Lakshmi, Assoc. Prof., Department of ECE and Mr. D.M.K Chaitanya ,Assoc. Prof. Dept. of ECE*** *for their continual monitoring and direction, as well as for supplying required project information and for their assistance in finishing the project.*

We are also grateful to ***Dr. E. Sreenivasa Rao, Head of the department****, for providing great facilities and a pleasant atmosphere to complete this project.*

We sincerely acknowledge all the people mentioned above, without whose co-operation we couldn't have completed the project successfully.

Navya Gangidi

(1602-17-735-030)

K. Madhu Sai Kalyan

(1602-17-735-024)

S. Dhvijesh

(1602-17-735-015)

ABSTRACT

The goal of this research is to use machine learning and learning data analysis to detect fraudulent credit card transactions. This is an attempt to address all of the problems associated with such detection while still safeguarding users' privacy. The basic issue of processing enormous data every single day and responding to a scam in a stipulated time is tackled in this project by establishing a model which is simple and fast in detecting an anomaly and acknowledging it. In order to provide a secure interface, reduction of dimension of data is done and to increase the accuracy and resolve the issue of misclassified data, a trustworthy source is implemented. Scammers' adaptive methods against the model are handled by building a simple and interpretable model, so that when the scammer adapts to it, we may have a new model up and running to deploy with just a few modifications. We are doing a comparative analysis of various models based on their accuracy. Therefore, we would implement the concepts of co-relation and machine learning in python with help of Jupyter note book.[1]

Keywords: Software, Machine learning, Python, Jupyter NoteBook.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

1.1 Aim

The project's main goal is to design and develop different fraud detection methods for streaming transaction data, with the goal of analyzing past customer transaction details and extracting behavioral patterns using concepts from various machine learning models such as Decision Trees, Support Vector Machines, Random Forest, and Naive Bayes, among others. Following the installation of multiple machine learning models, the best model with the greatest MCC score must be found after evaluating the data and examining the models from many angles in order to have a running model that is efficient.[6]

1.2 What is Machine Learning?

Machine Learning is the study of teaching machines or computers to learn and act in the same way that people do, and to enhance their learning over time in an autonomous manner, by feeding them data and knowledge in the form of observations and real-world interactions. However there is a lot of difference and distinction can be made between Machine Learning and Artificial Intelligence. In simple words, the main difference is that machine learning is just one aspect of artificial intelligence. Machine learning study is a subset of artificial intelligence research that aims to equip computers with knowledge through data, observations, and interaction with the outside world. This learned information enables computers to appropriately generalize to new situations. AI is a very vast term which includes various technologies like Natural Language Processing, Speech Recognition etc.[7]

1.3 Why Machine Learning for Fraud Detection?

There are some other conventional methods for detecting credit card fraud. But over all those methods we have chosen Machine Learning for the following reasons: [8]

- Using Machine Learning the model detects the fraud automatically once it is trained and tested whereas in case of conventional methods the rules of detection should be set manually.
- Real time streaming is possible in Machine Learning.
- Conventional methods take an enormous amount of time compared to Machine Learning models.
- Machine Learning models identify hidden correlation of data as well.
- Conventional methods detect only obvious fraud cases.

1.4 Challenges

- Millions of transactions take place every day so the data that needs to be processed is also huge. Thus, the built model must be fast enough as well as accurate.
- Data imbalance is high I.e., more than 99.5% of transactions are genuine and only less than 0.5% are fraud which makes it even harder for the model to detect fraud cases.
- Data is not available directly with regard to privacy.
- However accurate the system might be if a fraud activity is not detected then it could be a major issue.
- Adaptive strategies can be used in opposition to the version by way of the scammers.

1.5 How to overcome challenges

- The model must be basic and rapid in order to detect frauds as quickly as feasible.
- Imbalance may be addressed by splitting the data into smaller subsets and training the model if necessary. It can also be dealt using oversampling and undersampling methods.
- Data privacy can be achieved by changing the actual data into numbers using principal Component Analysis (PCA).

- While training the data an already developed model can be used as a double check for accurate detection.
- When a scammer is detected, we should be able to modify the model with some simple changes. This can be done if the model is simple, fast and accurate.

1.6 Process Flow

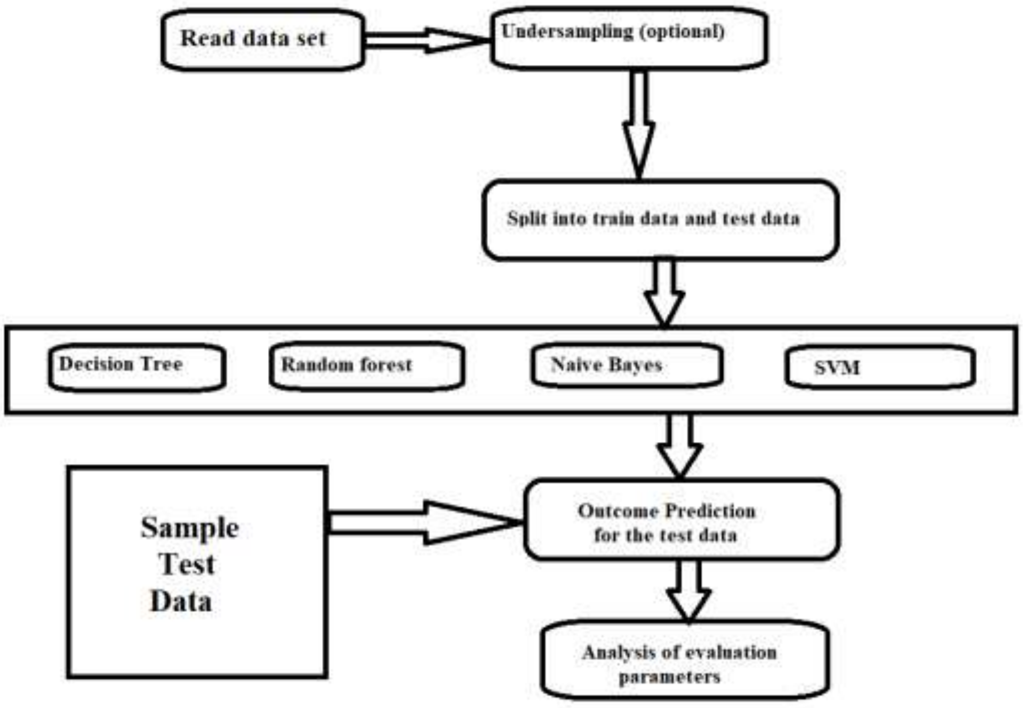

Read data set
Undersampling (optional)
Split into train data and test data
Decision Tree
Random forest
Naive Bayes
SVM
Sample Test Data
Outcome Prediction for the test data
Analysis of evaluation parameters

Figure1.1: Process flow

We implemented four models. The Basic steps involved in implementing each model are shown in the above figure.

Step 1: Firstly, the model is created using sklearn. Some models have different approaches like, results of SVM are based on the kernel we choose for the model. Kernels of SVM consist of Linear, RBF, Polynomial etc…

Step 2: The models are trained using the dataset i.e. “Creditcard.csv”. We used an 80 percent dataset to train the model. To replicate the results, we used a certain random_state for each model.

Step 3: We used the remaining 20 data to test the trained model. Evaluation metrics are noted for each model. The Evaluation metrics we used are accuracy, precision, recall, F1 score, MCC.

Step 4: After finding the Evaluation metrics of all models. We compare the MCC of the models, and find the model with the highest MCC of all. That model is the best model for approaching this project.

II. LITERATURE SURVEY

Fraud is defined as the unlawful or criminal deceit designed to obtain monetary or personal gain. It is an intentional conduct that violates a law, regulation, or policy with the intent of obtaining unlawful financial advantage.[9]

A large number of literature on fraud detection in this sector have been earlier published and are available for everyone. Data-mining applications and Computerized fraud detection are among the approaches used in this sector, according to a detailed survey done by Clifton Phua and his colleagues.[2]

One of the techniques that provided fresh light on this subject was the Artificial Genetic Algorithm, which tackled fraud from another approach. This was successful in detecting fraud transactions and thereby, reducing the number of false warnings. Despite the fact that it was accompanied with a categorization issue with variable misclassification costs.[3]

Na Wang and Wen-Fang YU presented another research paper, in which they used Outlier mining, Outlier-detection mining, Distance sum algorithms to precisely predict fraudulent transactions in a simulation experiment of credit-card transaction data from a specific corporate bank.[10]

There have also been attempts to go forward from an entirely other perspective. In the event of a fraudulent transaction, efforts are made to improve alert-feedback interaction. In case of a fraud transaction, the authorized system will be notified, and a response will be delivered to refuse the ongoing transaction.

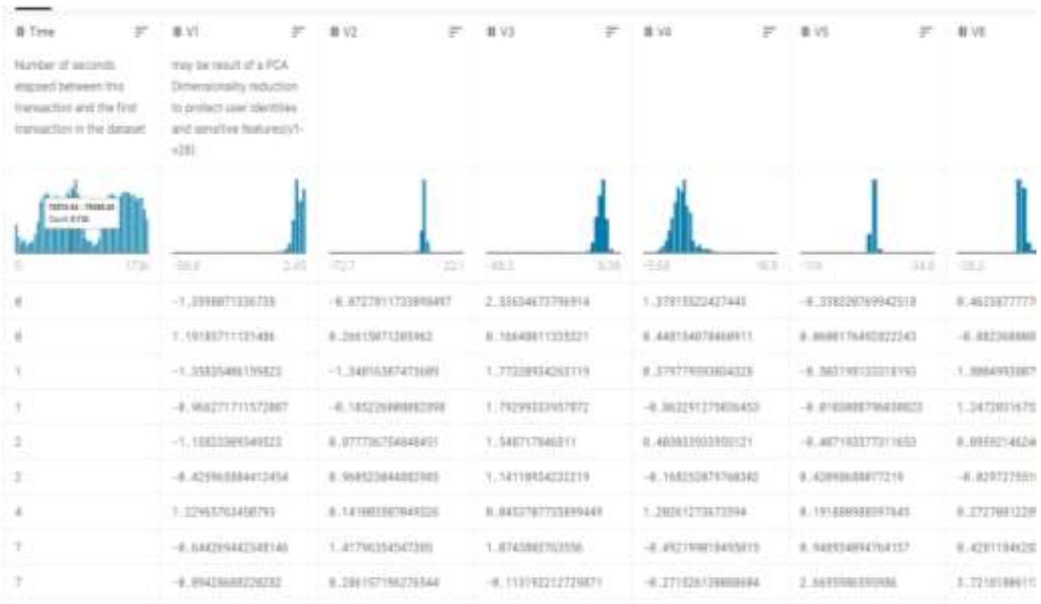
Rimpal R.Popat and Jayesh Chaudhary: They made a survey on credit card fraud detection, considering the major areas of credit card fraud detection that are bank fraud, corporate fraud, Insurance fraud. With these they have focused on the two ways of credit card transactions i) Virtually (card, not present) ii) With Card or physically present. They concentrated on techniques including regression, classification, logistic regression, support vector machines, neural networks, Naive Bayes, genetic algorithms, data mining, decision trees, and fuzzy logic-based models, among others. They describe six data mining techniques as theoretical foundation, including classification, clustering, prediction, outlier identification, regression, and visualization. Then have explained about existing techniques based on statistical and computation which is Artificial Immune system, Bayesian belief Network, Neural Network, Logistic Regression, Support Vector Machine, Tree, as a result, they had concluded that all the present machine learning techniques mentioned above can provide high accuracy for the detection rate, Industries are looking forward to finding new methods to improve accuracy and reduce the computational cost. Machine learning might be an excellent option.[11]

Shiyang Xuan: They compared the results of two random forests. Random-tree-
CART-based random forest is a random forest that is based on CART. They trained the behavior characteristics of regular and aberrant transactions using two separate random forest methods, each with its own set of base classifications and performance. They applied both of the algorithms on the dataset e-commerce company in China in which the fraud transaction in the subsets ratio is 1:1 to 10:1. As a result, accuracy from the random-tree-based random forest is 91.96% whereas in CART-based random forest is 96.7%. Since the data used is from the B2C dataset many problems arrived such as unbalanced data. Hence, the algorithm can be improved. [12]

III. DESIGN METHODOLOGY

3.1 Kaggle Dataset

All of the models datasets are taken from the Kaggle website. By separating the data into two halves, it may be utilized for both training and testing the models. It consists of the details of all the transactions that occurred in two days in September 2013.This dataset have got 2,84,807 transactions in total, out of which only 492 transactions are fraud i.e., about 0.172%. This makes the dataset highly imbalanced.[13]
Due to confidential issues, we are only given the main components of the data, not the original data, in order to protect the privacy of the clients. There are a total of 28 characteristics in the data (V1–V28). All of these characteristics are key elements of the original features/attributes.
However, there are two characteristics that are not changed by PCA. They're called 'Time' and 'Amount.' The elapsed time between each transaction and the first transaction in the dataset is stored in the ‘Time' feature. The Transaction's amount data are stored in the feature 'Amount.' ‘Class’ is the responsive variable based on which the transaction status is known. If it is 1 then the transaction is said to be fraud otherwise it is said to be a valid transaction.

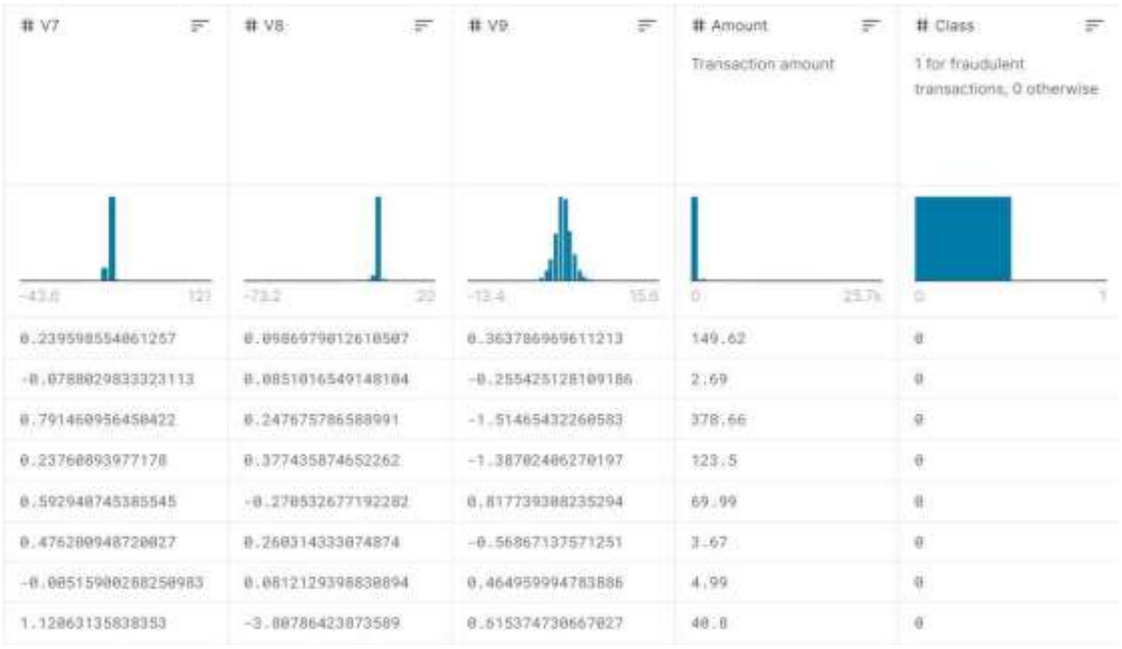


Time: Number of seconds elapsed between this transaction and the first transaction in the dataset

V1: may be result of a PCA Dimensionality reduction to protect user identities and sensitive features(v1-v28)

Time	V1	V2	V3	V4	V5	V6
0	-1.3598071336738	-0.0727811733098497	2.53634673796914	1.37815522427443	-0.338320769942518	0.4623877777
0	1.19185711131486	0.26615071205963	0.16648011335321	0.448154078460911	0.0600176492822243	-0.0823608088
1	-1.35835406159823	-1.34016307473609	1.77320934263119	0.379779593034328	-0.503198133318193	1.8004993807
1	-0.966271711572087	-0.185226008082898	1.79299333957872	-0.863291275036453	-0.0103088796030823	1.247203167
2	-1.15823309349523	0.877736754848451	1.548717846511	0.403033933955121	-0.407193377311653	0.0959214624
2	-0.425965884412454	0.960523044882985	1.14110934232219	-0.168252079760302	0.42098688077219	-0.0297275516
4	1.22965763450793	0.141003507049326	0.0453707735899449	1.20261273673594	0.191880988597645	0.272708122
7	-0.644269442348146	1.41796354547385	1.0743803763556	-0.492199018495015	0.948934094764157	0.428118462
7	-0.89428608220282	0.286157196276544	-0.113192212729871	-0.271526130088604	2.6695986595986	3.721818061

Figure 3.1: Data (1)



# V7	# V8	# V9	# Amount Transaction amount	# Class 1 for fraudulent transactions, 0 otherwise
-43.6 – 121	-73.2 – 22	-13.4 – 15.6	0 – 25.7k	0 – 1
0.239598554061257	0.0986979012610507	0.363786969611213	149.62	0
-0.0788029833323113	0.0851016549148104	-0.255425128109186	2.69	0
0.791460956450422	0.247675786588991	-1.51465432260583	378.66	0
0.23760893977178	0.377435874652262	-1.38702406270197	123.5	0
0.592940745385545	-0.270532677192282	0.817739308235294	69.99	0
0.476200948720027	0.260314333074874	-0.56867137571251	3.67	0
-0.00515900288258983	0.0812129398838894	0.464959994783886	4.99	0
1.12063135838353	-3.00786423873589	0.615374730667027	40.8	0

Figure 3.2: Data(2)

3.2 Software Requirements

3.2.1 Anaconda

To fabricate the Machine Learning models anaconda and jupyter notebook are used. Anaconda Individual Edition is the world's most well-known Python appropriation platform with more than 25 million clients around the world. Anaconda constrictor is a distribution of the Python and R programming dialects for scientific computing, data science, AI applications, massive scale data processing, and predictive analytics that aims to improve on board and arrangement.

3.2.2 Jupyter Notebook

Jupyter Notebook is a free and open-source web tool for creating and sharing live code, conditions, representations, and text archives. Jupyter Notebook is managed by Project Jupyter. Jupyter Notebooks is just a side project of the IPython development, which originally had its own IPython Notebook project. Jupyter gets its name from the three most popular programming languages it supports: Julia, Python, and R.

Jupyter is included with the IPython component, which allows you to write Python projects, but there are currently over 100 distinct pieces that you can use.

3.2.2 Python 3.6

Python is an interpreted programming language that is used for a wide variety of tasks. Python was designed by Guido van Rossum and was originally released in 1991. Its design philosophy emphasizes code readability and makes comprehensive use of whitespace. Its salient feature is that its object-oriented approach is designed to assist programmers in writing clear, logical code for both small and big projects. It is a garbage-collected and dynamically typed language. It supports a variety of programming paradigms, including structured (especially procedural) programming, object-oriented programming, and functional programming. Because of its extensive standard library, Python is frequently referred to as a "batteries included" language.

3.3 Libraries or Modules Used

NumPy: NumPy is a Python library that adds support for huge, multidimensional arrays and matrices, as well as a wide set of high-level mathematical functions for manipulating the arrays.

Pandas: Wes McKinney created Pandas, a high-level data manipulation tool. It is based on the NumPy library, and the Data Frame is its primary data structure. Data Frames are a type of data structure that allows you to store and modify tabular data in the form of rows of observations and columns of variables.

Matplotlib: Matplotlib is a Python package that allows you to create static, animated, and interactive visualizations. Matplotlib makes simple things simple and difficult things possible.

Scikit-learn/sklearn: Scikit-learn (previously scikits.learn, and also known as sklearn) is a free Python machine learning package. It includes support vector machines and random forests, among other classification, regression, and clustering techniques, and is designed to work with the Python numerical and scientific libraries NumPy and SciPy. Linearizing our labels, dividing data for training/testing.

3.4 Decision Tree Classifier

Classes are pre-determined in supervised machine learning algorithms. A set of data is represented by a class. Classification will be used to label a certain set of data. The main goal at hand is to look for a variety of patterns and construct various mathematical models. The classifier's goal is to properly predict which class the data samples belong to.

A decision tree is a flowchart-like tree structure in which each internal node represents an attribute test, each branch indicates the test's result, and each leaf node (terminal node) carries a class label.

3.4.1 History

In 1963, The first regression tree was created at the University of Wisconsin-Department Madison's of Statistics in the year. It divided data into two subgroups in a recursive manner. Years later in 1966, Hunt was the first person to publish a decision tree. In Psychology, these approaches were used to model the human idea of learning. In 1972, The very first classification tree appeared on the screen. It divided data in order to maximize the number of cases in each category. In 1974, Development of CART algorithm started by professors from various universities. By 1977, CART's 1st version was invented. In 1984, The CART algorithm was published and created a new revolution in the world of algorithms. In 1986,
Invented ID3 algorithm by using impurity criterion called "Gain Ratio."[15]

3.4.2 Terminology

- ***Root node:*** The feature to be placed as root node is chosen using attribute selection techniques which are discussed further below in this document. This property is used to divide data into two or more groups.
- ***Splitting:*** If-else conditions are used to divide a node into more sub-nodes.
- ***Parent and Child node:*** A node which is divided into sub-nodes is called a parent of those sub-nodes and these sub-nodes are the child nodes.

- ***Sub-Tree or Branch:*** A branch or sub-tree is a section of the overall decision tree.
- ***Decision nodes:*** The nodes that emerged from subdividing the sub nodes into even more sub nodes.
- ***Terminal or Leaf node:*** The nodes at the decision tree's end that can't be divided any more.
- ***Pruning:*** Sub nodes are removed from a tree. Pruning is the reversal of splitting.

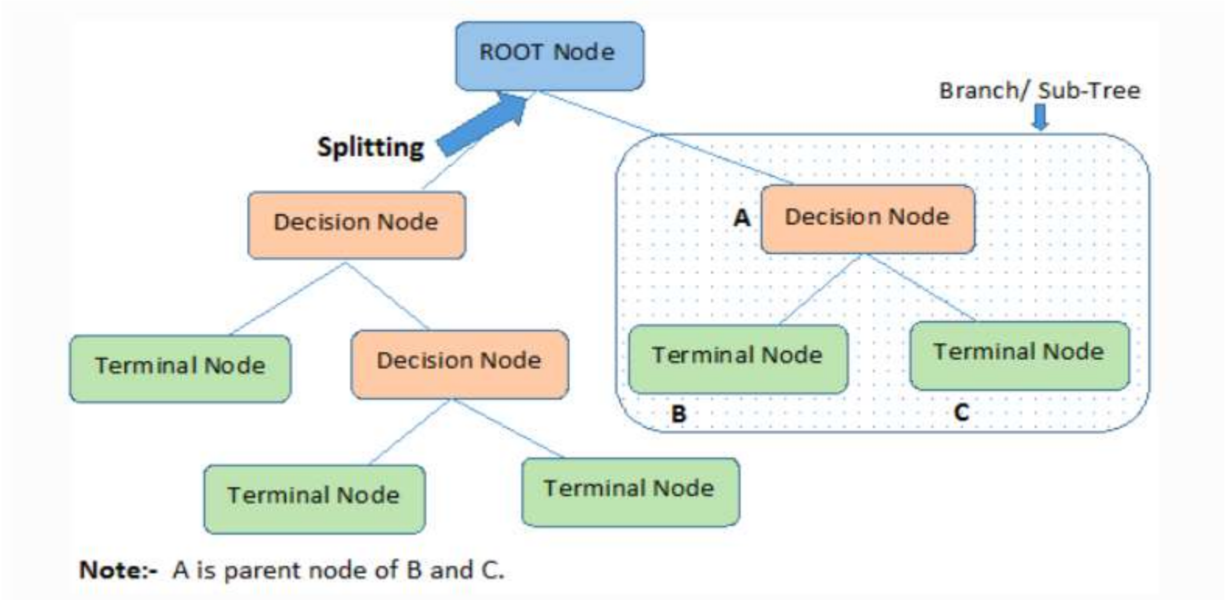


Figure 3.3: Terminology of decision tree classification

3.4.3 Working

Internal decision nodes and terminal leaves make up a decision tree. Each branch is labelled by a check function f(x) that is implemented by each option node 'm'. Given an input, a test is run at each node, and one of the branches is chosen based on the final results. This method begins at the root and continues recursively until it reaches a leaf node, at which point the value stored in the leaf becomes the output.

Steps Involved :

Step 1 : In order to split the records, choose the best attribute using one of the ASM's (Attribute Selection Measures).

Step 2 : That selected attribute breaks the dataset into subsets and it is called the decision node.

Step 3: Repeat the above process recursively until one of the conditions is satisfied below:

a) If we arrive at a pure sub split
b) No more instances
c) No more remaining attributes

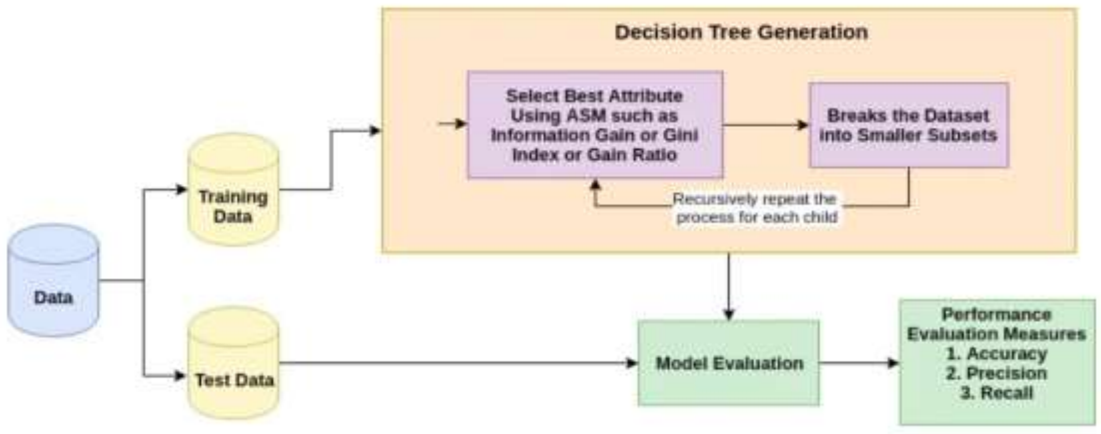

Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3.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

Figure 3.4: Working of Decision Tree classifier

3.4.3.1 Attribute Selection Measures

The attribute choice degree is a heuristic way for selecting a differentiating criteria that divides the data in best practical way. It is considered as differentiating policies since, this allows users to set break-points for tuples on a certain node. As a means of understanding the supplied data-set, Attribute Selection Measures allocates a rating for each feature. As a splitting feature, best rating feature might be selected. Breakup-points for branches should be defined in case of continuous valued property. Gini and the I.G(Information Gain) are two of the most well-known selection criteria.

INFORMATION GAIN:

Shannon's entropy quantifies the impureness of the incoming data set. Entropy is frequently referred to as the unpredictability or impurity of a system in science and mathematics. This refers to the impurity in a certain set of instances in stats. Generally, Change in Entropy is measured by Information-Gain.

Based on supplied characteristic values, information gain computes the difference between entropy before split and average entropy after split of the dataset. Information gain is used in the ID3 (Iterative Dichotomiser) decision tree technique.

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$$

- Where, H(X) = Entropy
    - P is the probability of the chosen variable
    - b is the base if the logarithm
    - x is the chosen variable
    - i is different types of outputs

Information gain may alternatively be defined as the difference between the parent node's entropy and the weighted common entropy of child nodes.

$$IG(S, A) = H(S) - H(S, A)$$

Alternatively,

$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Where, H(S) = entropy of parent node

H(S, A) = weighted entropy of child nodes

IG(S, A) = Information Gain

That node which has highest information gain is considered to be the root node.

GINI INDEX : A calculation that adds the ratios of each class instance to the total number of node instances. It then subtracts one from it. A ‘0' impurity would be found in an ‘Ideal' node.

$$Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

Where, c = 2(for binary classification)

p = probabilities of positive and negative class

Generally, Gini Index is considered as a better measure because it is computationally efficient and takes shorter period of time for execution.

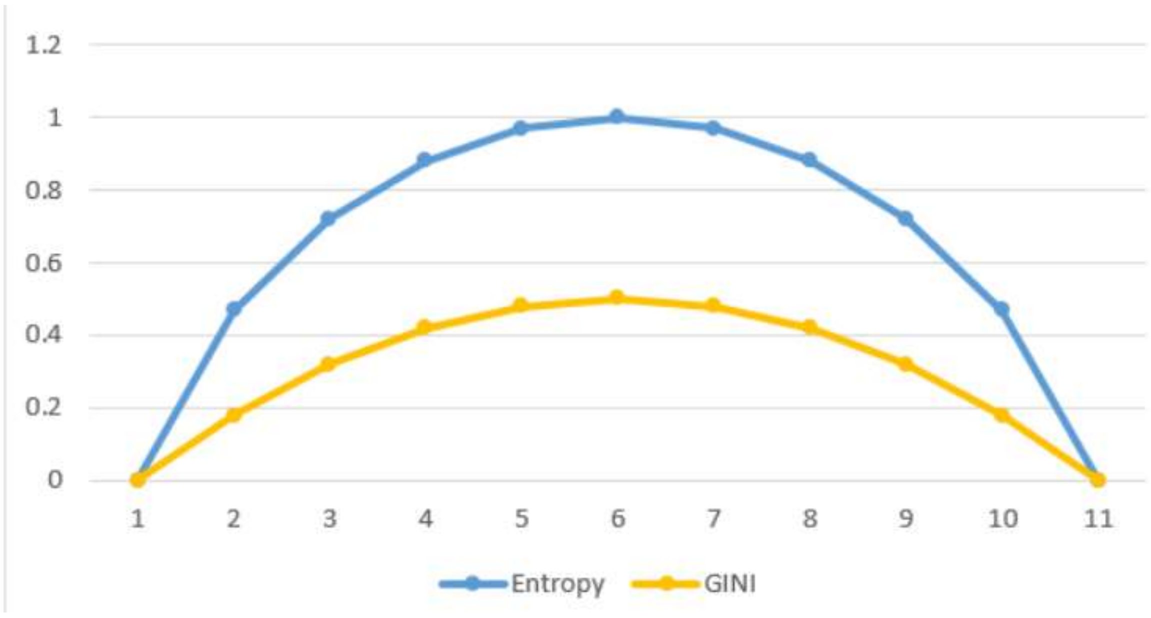


Figure 3.5: Entropy Vs Gini

3.4.3.2 Algorithms and Examples

The following are two of the most widely used decision tree algorithms:

a) ID3 (Iterative Dichotomiser 3) – This uses entropy and information gain as a metric.
b) CART (Classification and Regression Trees) – This uses gini index as a metric.

Example calculations using IG:

We built our model based using Entropy and Information Gain. So here is an example of the calculations. Consider the below dataset.

X	Y	Z	T(TARGET)
A	D	F	H
A	E	F	H
A	E	G	I
B	D	F	H
B	D	G	H
B	E	G	I
B	E	F	I
C	D	F	H
C	E	G	H
C	E	F	H

Table 3.1: Example dataset

To identify the dependent variable, there exist independent variables. X, Y, and Z are the independent variables. T is the dependent variable.

Finding the parent node for our decision tree is the first step. To do so, follow the instructions below:

Determine the class variable's entropy. The letter 'S' stands for 'class.'

E(S) = -[(7/10)log(7/10) + (3/10)log(3/10)] = 0.88

Note: Normally, we'll take the log to base 2. There are ten H/I in total. Seven of them are H and three are I. We estimated probability above based on that.

The following table is derived from the above data for 'X'.

X	H	I	TOTAL
A	2	1	3
B	2	2	4
C	3	0	3
			10

Now we must discover the total of weights of each feature multiplied by probabilities in order to calculate average weighted entropy.

E(S, X) = (3/10)*E(2,1) + (4/10)*E(2,2) + (3/10)*E(3,0) = (3/10)(-(2/3)log(2/3)-(1/3)log(1/3))+ (4/10)(1) + (3/10)(0) = 0.6754

The next step is to find the information gain. It is the difference between parent entropy and average weighted entropy we found above.

IG(S, X) = 0.88 - 0.6754 = 0.2046

Similarly find Information gain for Y and Z.

Y	H	I	TOTAL
D	4	0	4
E	3	3	6
			10

E(S, Y) = (4/10)*E(4,0) + (6/10)*E(3,3) = (4/10)(0)+ (6/10)(1) = 0.60

IG(S, Y) = 0.88 - 0.60 = 0.28

Z	H	I	TOTAL
G	2	2	4
F	5	1	6
			10

E(S, Z) = (4/10)*E(2,2) + (6/10)*E(5,1) = (4/10)(1)+ (6/10)(0.65) = 0.79

IG(S, Z) = 0.88 - 0.79 = 0.09

Select the feature with the highest entropy gain now. It's 'Y' in this case. As a result, it is the initial (root) node in our decision tree.

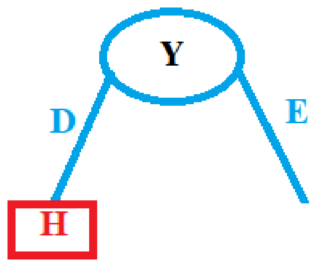
Our data now looks like this:

Y	X	Z	T
D	A	F	H

D	B	F	H
D	B	G	H
D	C	F	H

Y	X	Z	T
E	A	F	H
E	A	G	I
E	B	G	I
E	B	F	I
E	C	G	H
E	C	F	H

We can assign 'D' to H since it only contains examples of class 'H.' That is to say, if Y is D, T will always be 'H'. Our decision tree now looks like this.



The next stage in our decision tree is to locate the next node. We'll now look beneath the letter 'E' for one. We need to figure out which of the following 'X' or 'Z' has the most information gain.

Parent entropy E(E) : E(E) = (-(3/6)log(3/6)-(3/6)log(3/6)) = 1.

Now calculate the information gain of 'X'[IG(E,X)]

X	H	I	TOTAL
A	1	1	2
B	0	2	2
C	2	0	2

E(E, X) = (2/6)*E(1,1) + (2/6)*E(0,2) + (2/6)*E(2,0) = 1/3 = 0.33

IG(E, X) = 1–0.33 =0.67

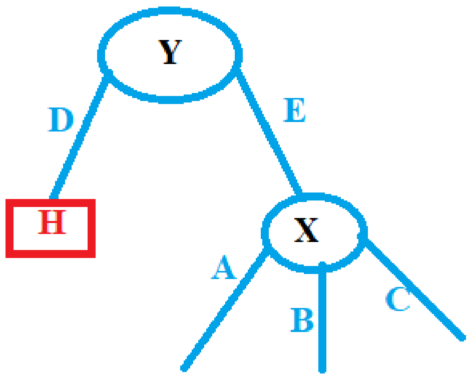
Similarly Calculate the information gain of 'Z'[IG(E,Z)]

Z	H	I	TOTAL
G	1	2	3
F	2	1	3

E(E, Z) = (3/6)*E(1,2) + (3/6)*E(2,1) = 0.918

IG(E, Z) = 1–0.918 =0.082

The biggest value is IG(E,X). So, beneath 'Y,' there is a node called 'X.'



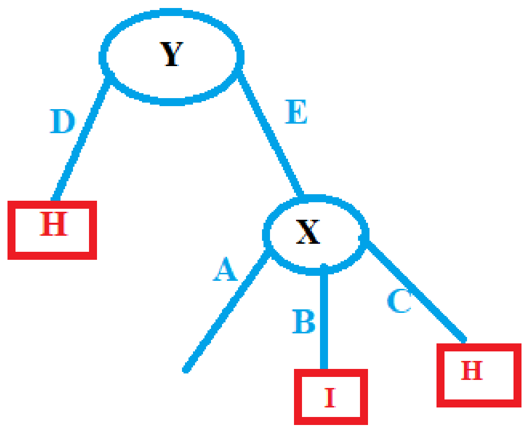
Now our data looks as follows:

X	Z	T
A	F	H
A	G	I

X	Z	T
B	G	I
B	F	I

X	Z	T
C	G	H
C	F	H

We may assign 'B' to I because it only includes examples of class 'I,' and 'C' to H because it only contains examples of class 'H.' Our decision tree now looks like this.



The next stage in our decision tree is to locate the next node. We'll utilise it under 'X' now that there's just one node remaining, namely 'Z.'

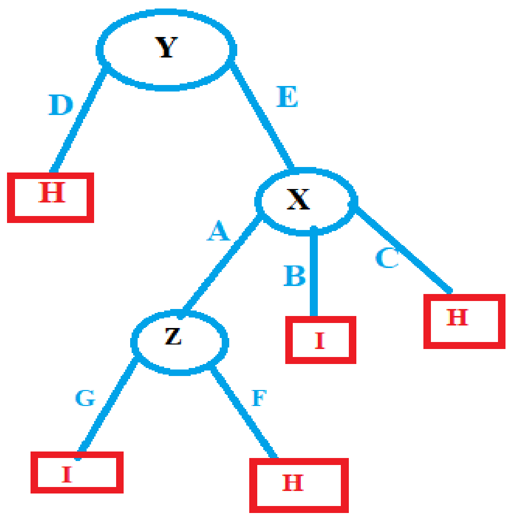
Calculate parent entropy E(A) : E(A) = (-(1/2)log(1/2)-(1/2)log(1/2)) = 1.

Now Calculate the information gain of 'Z'[IG(EA,Z)]

E(EA, Z) = (1/2)*E(1,0) + (1/2)*E(0,1) = 0

IG(EA, Z) = 1

Our final decision tree looks as below:



Example Calculations Using Gini Index:

As we know Gini Impurity is also one of the attribute selection measure, we will use it in CART algorithm. It is similar to the previous one. We utilise Gini Impurity instead of Entropy. We were able to discover the process by using the same dataset.

Finding the parent node for our decision tree is the first step. To do so, follow the instructions below:

Calculate the Gini index of the class variable

From Table 3.1: Gini(S) = $1 - [(7/10)^2 + (3/10)^2] = 0.42$

Next, we should calculate the Gini gain.

The average weighted Gini impurity of X,Y, and Z will be found first.

Consider the situation of 'X.'

gini(S, X) = (3/10)*gini (2,1) + (4/10)*gini(2,2) + (3/10)*gini(3,0) = 0.332

Gini gain(S, X) = 0.42 – 0.331 = 0.088

Consider the situation of 'Y.'

gini(S, Y) = (4/10)*gini (4,0) + (6/10)*gini(3,3) = (4/10)(0)+ (6/10)(0.5) = 0.30

Gini gain(S, Y) = 0.42 - 0.30 = 0.12

Consider the situation of 'Z.'

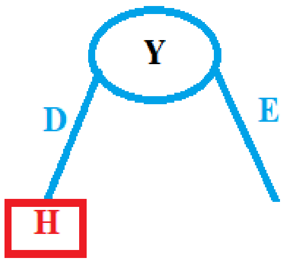
gini(S, Z) = (4/10)*gini (2,2) + (6/10)*gini(5,1) = (4/10)(0.5)+ (6/10)(0.277) = 0.3662

Gini gain(S, Z) = 0.42 - 0.3662 = 0.0538

Select the feature with the highest gini gain now. It's 'Y' in this case. As a result, it is the root node in our decision tree.

Now the data gets further divided.

We can assign 'D' to H since it only contains examples of class 'H.' That is to say, if Y is D, T will always be H. Our decision tree now looks like this.



The next step is to locate the decision tree's next node. Now we'll look under the letter 'E' for one. Which of the following 'X' or 'Z' has the higher gini gain?

Calculate parent gini- Gini(E) : Gini(E) = 1 - [$(3/6)^2$ + $(3/6)^2$] = 0.5

Now Calculate the Gini gain of 'X'[Gini gain(E,X)]

From Table xxx :

gini(E, X) = (2/6)*gini(1,1) + (2/6)*gini(0,2) + (2/6)*gini(2,0) = 0.166

Gini gain(E, X) = 0.5–0.166 = 0.334

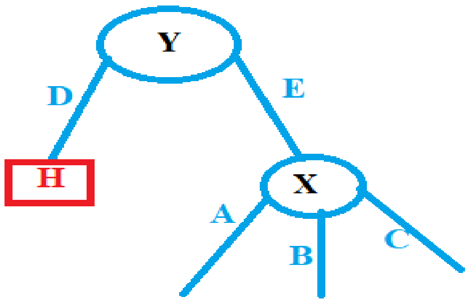
Similarly Calculate the Gini gain of 'Z'[Gini gain(E,Z)]

From Table xxx :

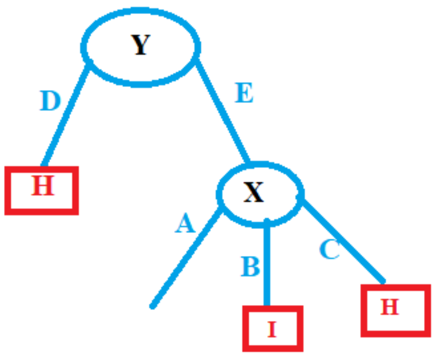
gini(E, Z) = (3/6)*gini(1,2) + (3/6)*gini(2,1) = 0.44

Gini gain(E, Z) = 0.5–0.44 =0.06

The greatest value is Gini gain(E,X). So, under 'Y,' there is a node called 'X.'



We can assign ‘B' to I because it only contains examples of class ‘I,' and ‘C' to H because it only contains examples of class ‘H.' Our decision tree now looks like this.



The next stage in our decision tree is to locate the next node. Now since there is only one node left i.e. ‘Z’, we will use it under ‘X’.

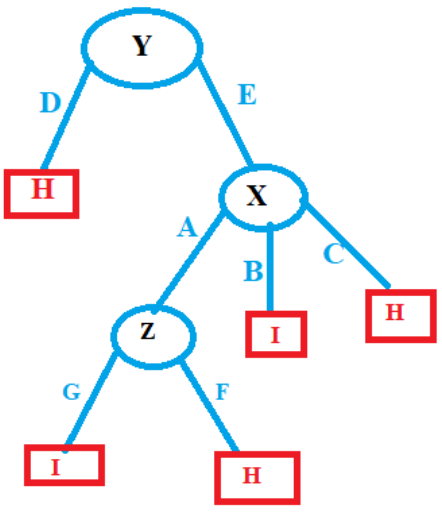
From Table xxx: ***Calculate parent gini- Gini(A)*** : Gini(A) = 1 - [$(1/2)^2 + (1/2)^2$] = 0.5

Now Calculate the Gini gain of ‘Z’[Gini gain(EA,Z)]

gini(EA, Z) = (1/2)*gini(1,0) + (1/2)*gini(0,1) = 0

Gini Gain(EA, Z) = 1

So, our final decision tree looks the same.



3.4.4 Advantages of Decision Trees

1. Easy to calculate and understand.

2. Computation involved in performing classification is much less compared to that of any other classification techniques.

3. They can deal with both continuous values as well as categorical values i.e., used for both classification and regression techniques.

4. Can be used as one of the feature selection techniques since the attributes with low IG values can be neglected.

5. Very efficient with small datasets.

3.4.5 Disadvantages of Decision Trees

1. It can lead to overfitting problems.

2. Not much effective with large datasets.

3. These are sensitive to errors in category problems with a large number of classes and a limited number of training samples.

4. To train a decision tree it is computationally expensive.

3.4.6 Decision Tree classifier with undersampling

When the data is uneven, the performance estimation will be overstated. As a result, incorrect assumptions about the algorithm's relevance in surpassing chance may be made. Therefore it is very important that we work on a balanced data set. Because most algorithms are intended to increase accuracy while minimising error if we train the model on a balanced dataset. Solutions for this problem are as follows:

- We can restore balance on the training set by undersampling the large class or oversampling the small class to avoid bias in the first place.

- Use SMOTE technique (Synthetic Minority Oversampling Technique).
- In order to avoid bias, we can change the costs of misclassification.

Here we used the first method in our project and did undersampling of data which improves all the performance metrics of the model.

3.4.6.1 Terminology

- ***Oversampling:*** The process of raising the number of copies of a minority class is known as oversampling. Oversampling is a good alternative when you don't have a lot of data to work with.
- ***Undersampling:*** The term "undersampling" refers to the practice of excluding certain observations from the majority class. When you have a lot of data, such as millions of rows, undersampling might be a viable option.
- ***SMOTE:*** It stands for synthetic minority oversampling technique. It is a technique which creates synthetic samples. In order to generate new and synthetic data it uses nearest neighbors algorithm. New samples are generated only in the training data set.

Before using sampling techniques, you should always segregate your data into test and training sets. Sampling the data before dividing it might result in the same observations appearing in both the test and train sets. This can lead to overfitting and poor generalization to test data if our model just memorizes individual data points.

3.4.6.2 Disadvantage of Undersampling

- We're getting rid of information that may be useful. This might result in under fitting and poor test set generalization.

3.5 Random Forest Classifier

A supervised classification and ensemble classification algorithm, the random forest approach is used. This algorithm generates a forest with a large number of trees in it. The bigger the number of trees in the forest, the higher the accuracy of the random forest classifier. The random forest algorithm is an ensemble classification algorithm. A group of classifiers is referred to as an ensemble classifier. Rather than relying on a single classifier to forecast the target, we employ many classifiers. These ensemble classifiers are randomly built decision trees in random forest. The target prediction is based on the majority voting approach, and each decision tree is a single classifier.

The notion of majority voting is similar to that of political voting. Each voter casts a vote for one of the political parties that are competing in the elections. Similarly, each classifier will vote for one target class from among all target classes. To announce the results of the election, the votes will be counted, and the party with the most votes will be declared the election winner. Similarly, the target class that received the most votes was chosen as the final anticipated target class.

3.5.1 History

In 1995, Tim Kam Ho used the random subspace approach to build the first Random Forest algorithm. Later in 2006, Leo Breiman and Adele Cutler came up with the idea for the algorithm's expansion.[16]

3.5.2 Terminology

Same terminology is used as mentioned in Decision Tree Classifier. (3.4.2)

3.5.3 Working

How a tree in random forest looks like:

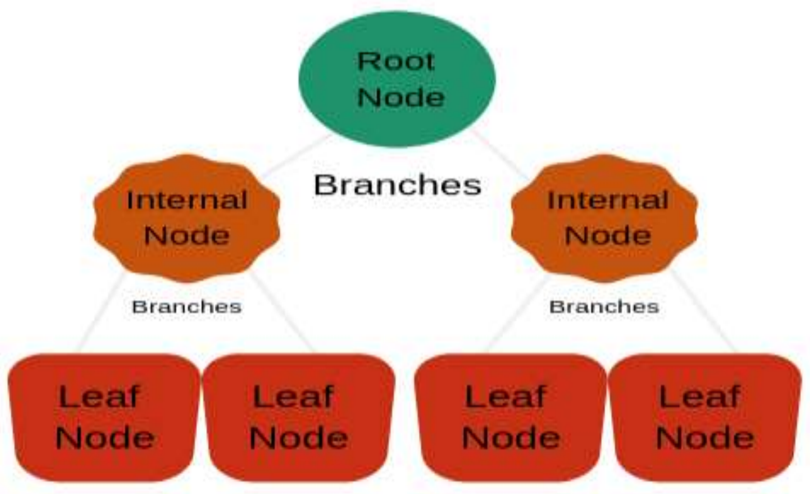


Figure 3.6: Example of a Random Forest Tree

It consists of a root node, branch nodes which consists of internal nodes, and

Branches are further divided into leaf nodes.

Let us take a look at an example of a random forest of differentiating fruits.

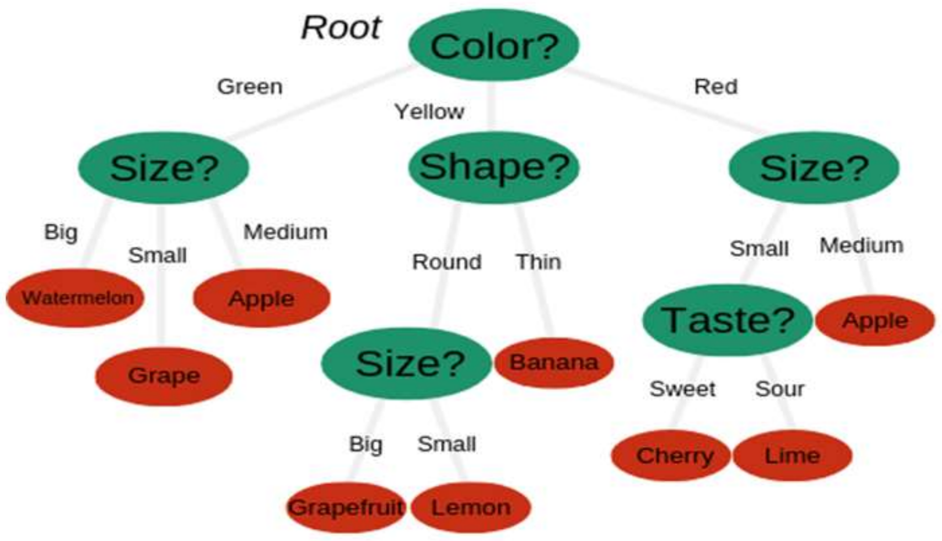


Figure 3.7: Example of a Random Forest Tree differentiating fruits

In this example, we can see how decision trees are used in random forest.

For root nodes we used color as the main criteria to differentiate fruits, and size, shape as Branch Nodes.

Every algorithm has a set of rules and instructions to be performed for its proper functionality to get desired output.

So, let us take a look at pseudo code of Random Forest Algorithm:

Random forest algorithms pseudocode may be divided into two parts.

- Random forest creation pseudocode
- Pseudocode to perform prediction from the created random forest classifier.

3.5.3.1 Random Forest pseudocode

- Choose "x" features at random from a total of "y" features.
- Using the optimal split point, compute the node "d" among the "x" characteristics.

- Using the best split, split the node into daughter nodes.
- Repeat steps 1–3 until the “l” number of nodes is attained.
- Create a forest by repeating steps 1–4 for a “n” number of times, resulting in a “n” number of trees.

The random forest method begins by picking “x” features at random from a total of “y” features. As you can see, we're collecting features and observations at random.

In the following stage, we'll use the best split technique to locate the root node using the randomly picked "x" characteristics.

The daughter nodes will then be calculated using the best split technique. We'll keep going through the first three phases until we have a tree with a root node and the goal as a leaf node.

Finally, we repeat steps 1–4 to generate “n” trees at random. All of these trees were produced at random, resulting in a random forest.

3.5.3.2 Random forest prediction pseudocode

The pseudocode following is used to forecast using the trained random forest algorithm:

- Takes the test characteristics and predicts the outcome using the rules of each randomly generated decision tree, then saves the anticipated result (target).
- Determines how many votes each anticipated target class will receive.
- The random forest algorithm's final forecast should be the goal with the most votes.

To use the trained random forest method to make a prediction, we must pass the test features through the rules of each randomly generated tree. Let's pretend we created 100 random decision trees from a random forest. For the same test feature, each random forest will predict a different goal (outcomes). After that, votes will be computed by taking into account each anticipated objective.

If 100 random decision trees predict three distinct targets x, y, and z, then the votes for x are nothing more than the number of trees that predict x out of 100 random decision trees. The same goes for the other two objectives (y, z). If x receives a large number of votes. Let's assume that 60 of a hundred random decision trees indicate that the target will be x. The final random forest then provides the anticipated target as x.

Majority voting is the term for this method of voting.

3.5.4 Tuning Hyper Parameters

Hyper parameters are employed in Random Forest to increase the model's predictive power or speed.

1. ***Increasing predictive power:***

- **n_estimators:** The number of trees that the model constructs before averaging predictions or voting by majority.
- **max_features:** The maximum number of characteristics that the model examines when deciding whether or not to divide into a node.
- **min_sample_leaf:** The bare minimum of leafs required to divide an internal node.

2. ***Improving the model's speed:***

- **random_state:** As a result, the output of our model is repeatable. When a random state is set to a specific value, the model will generate the same results.

3.5.5 Advantages of Random Forest:

- By integrating the outcomes of all decision trees, it overcomes the problem of overfitting.
- Can handle vast amounts of data and has less variation than a decision tree.
- It automates missing values in the data.
- Easy implementation and flexible to classification and regression problems.
- Normalizing of data is not required as it is a rule-based approach.

3.5.6 Disadvantages of Random Forest:

- It requires more computational power and resources because it creates more trees to combine all the outputs.

- Training takes a lot of time as it has to combine all trees to determine the target class.

3.6 Naive Bayes Classifier

On bigger datasets, the Naive Bayes classifier approach is a basic supervised algorithm technique that is quick and accurate. The likelihood of all potential outcomes of our problem is calculated using this procedure, and the output with the highest probability is proclaimed the real result.

Naive Bayes is a collection of algorithms, but not a single algorithm, based on Bayes Theorem.

3.6.1 Bayes Theorem

Conditional Probability is calculated using Bayes' theory. The theorem is named after Thomas Bayes, an English statistician who developed the formula in 1763. It is the cornerstone of the Bayesian inference approach, which is a type of statistical reasoning.

The chance of an event occurring with some link to the occurrence of one or more other occurrences is known as conditional probability.

P(A|B) stands for conditional probability.

Given that event ‘B' has already occurred, ‘A' is the event whose probability of occurrence needs to be estimated.

According to Bayes Theorem,

$$P(A|B) = P(B|A) * P(A) / P(B)$$

Sometimes often we do not have the access to or the value of P(B) directly available, in that case an alternate calculation can be done as shown:

$$P(B) = P(B|A) * P(A) + P(B|\text{not } A) * P(\text{not } A)$$

Here, P(not A) = 1 – P(A)

In the Bayes Theorem, substituting the value of this P(B) yields:

$$P(A|B) = P(B|A) * P(A)/[P(B|A) * P(A) + P(B|\text{not } A)* P(\text{not } A)]$$

If we're given P(not B|Not A), we may compute P(B|not A) as the complement as follows:

$$P(B|\text{not } A) = 1 – P(\text{not } B|\text{not } A)$$

3.6.2 Terminology:

The terms in the equation are given different names so as to easily understand the concept from different perspectives and think clearly.

- ***Posterior Probability:*** P(A|B), the probability of occurrence of event A given that event B has already occurred.
- ***Prior Probability:*** P(A), the probability of occurrence of the event A
- ***Likelihood:*** P(B|A), the probability of occurrence of event B given that event A has already occurred.
- ***Evidence:*** P(B), the probability of occurrence of the event B
- ***P(not A):*** The probability of non-occurrence of the event A
- ***P(not B):*** The probability of non-occurrence of the event B

We must remember that both events, A and B, are independent events, meaning that the probability of each event is unrelated to the likelihood of the other.

3.6.3 Assumptions

There are two assumptions that are to be made for each feature in Naïve Bayes algorithm is:

- No two features are dependent on each other i.e., one feature’s occurrence should not depend on whether or not another feature takes place.

If two features are independent then the probability is spilt as:

P(A,B) = P(A) * P(B)

- Equal weightage or importance for all the outcomes. Suppose there are more than two outcomes present then the final output should not be dependent on only one or two features. All the features should have equal importance in making the final decision.

Note that the assumptions made in the Naive Bayes may not be ideal in real time scenarios especially the independence assumption but often works well in practice.

If the dataset has more than two, let's say n number of, features. Then the Bayes theorem can be expressed as:

$$P(y|x_1,...,x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

This can be expressed mathematically as:

$$P(y|x_1,...,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$$

For every possible outcome the denominator remains the same, hence we can remove that term:

$$P(y|x_1,...,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

The equation becomes directly proportional to the numerator as the denominator is constant.

Here, the class variable is y, and the class probability is P(y).

We must find the likelihood of the set of input variables for all potential outputs using the aforementioned computations, and present the output with the highest probability as the real output to the provided issue. This may be expressed numerically as:

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

Argmax function is used to pick the variable with maximum probability of all the variables.

3.6.4 Example

Let us solve a problem.

Given below are two tables, Outlook and Temperature, that helps us in deciding whether today is suitable to play cricket or not. Given the conditions are today is Sunny and Hot. We have to find the solution using Naïve Bayes algorithm.

OUTLOOK TABLE:

	YES	NO
SUNNY	2	3
OVERCAST	4	0
RAINY	3	2
TOTAL	9	5

Table 3.2: Outlook Table

TEMPERATURE TABLE:

	YES	NO
HOT	3	2
MILD	1	2
COOL	5	1
TOTAL	9	5

Table 3.3: Temperature Table

Solution:

Given that today is sunny and hot we have two possible solutions. One is YES and the other is NO. We have to calculate the probability for all the possible solutions and decide the answer.

Calculating the probability for the solution YES:

$$P(Yes|Today) = P(Sunny|Yes) * P(Hot|Yes) * P(Yes) / P(Today)$$

Since Sunny and Hot are the two given input variables. We are finding the probability based on those two variables.

P(Sunny|Yes) = (value of yes for sunny in outlook)/(total value of yes)

$$= 2/(2+4+3) = 2/9$$

P(Hot|Yes) = (value of yes for hot in Temperature)/(total value of yes)

= 3/(3+1+5)

= 3/9

P(Yes) = (value of yes)/(sum of values of Yes and No)

= (2+4+3+3+1+5)/((2+4+3+3+1+5+3+0+2+2+2+1)

= 18/28

= 9/14

By substituting all the pre computed values in the Bayes Theorem Formula:

We get,

P(Yes|Today) = 2/9 * 3/9 * 9/14 /P(Today)

= 0.047/P(Today)

P(Today) will be same for all the possible outcomes. Hence can be ignored, and the equation becomes proportional, and can be written as:

P(Yes|Today) α 0.047

Similarly find the probability of No:

P(No|Today) = P(Sunny|No) * P(Hot|No) * P(No) / P(Today)

P(Sunny|No) = (value of No for sunny in outlook)/(total value of No)

= 3/(3+0+2)

= 3/5

P(Hot|No) = (value of No for hot in Temperature)/(total value of No)

= 2/(2+2+1)

= 2/5

P(No) = (value of No)/(sum of values of Yes and No)

= (3+0+2+2+2+1)/((2+4+3+3+1+5+3+0+2+2+2+1)

= 10/28 = 5/14

By substituting all the pre computed values in the Bayes Theorem Formula:

We get,

P(No|Today) = 3/5 * 2/5 * 5/14 /P(Today)

= 0.085/P(Today)

By ignoring P(Today), we get:

P(No|Today) α 0.085

By normalizing the calculated results with respect to each other we will get the actual probability of the two solutions.

P(Yes|Today) = P(Yes|Today) / [P(Yes|Today) + P(No|Today)]

= 0.047/(0.047+0.085)

= 0.047/0.132

= 0.356

P(No|Today) = P(No|Today)/ [P(Yes|Today) + P(No|Today)]

= 0.085/(0.047+0.085)

= 0.085/0.132

= 0.644

P(No|Today) can also be calculated as:

P(No|Today) = 1- P(Yes|Today)

= 1-0.356

= 0.644

Here the probability of No is greater than that of the probability of Yes. Hence we can say that No is the output of the solution.

Thus, today I.e., Sunny and Hot is **not** suitable for playing Cricket.

SOLUTION is **NO.**

3.6.5 Gaussian Naive Bayes Algorithm

The Gaussian NB algorithm is a variation of the Naive Bayes algorithm. It's especially useful when the dataset comprises continuous values and all of the characteristics are assumed to follow the Gaussian distribution, commonly known as the Normal Distribution. Gaussian NB only works with the data's Mean and Standard Deviation, making it the most user-friendly.

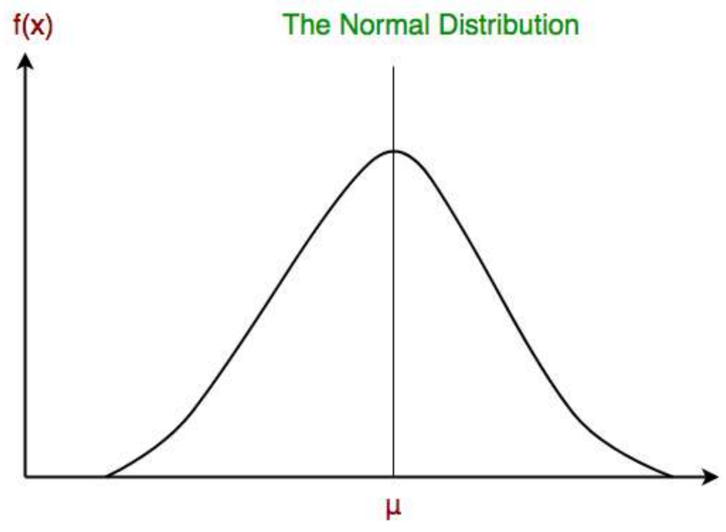


Figure 3.8: Normal Distribution

The following is an estimate of the likelihood of the features:

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

Here,

- μ_y is the mean of the data
- σ is the Standard Deviation of the data

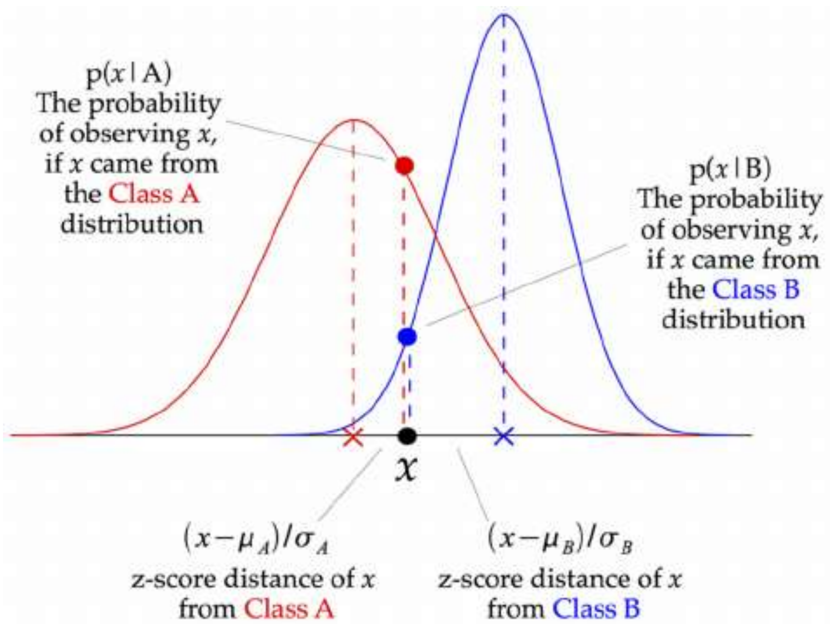

p(x | A)
The probability of observing x, if x came from the Class A distribution
p(x | B)
The probability of observing x, if x came from the Class B distribution
x
(x−μ_A)/σ_A
z-score distance of x from Class A
(x−μ_B)/σ_B
z-score distance of x from Class B

Figure 3.9: Gaussian distribution example

3.6.5.1 Example

Person(sex)	Height(feet)	Weight(lbs)	Foot(inches)
Male	5.58	170	10
Male	5.92	165	12
Male	5.92	180	11
Male	6	190	12
Female	5.75	150	7
Female	5	130	9
Female	5.42	150	6
Female	5.5	100	8

Table 3.4: Example

Using Gaussian Naïve Bayes classifier find out the sex of the person with height 6feet, weight 130lbs and foot size of 8 inches.

Solution:

First calculate the probability using Bayes Theorem and then mean and standard deviation for all the variables in the data separately for Male and Female.

Mean = (sum of the data) / (total no of records)

Mean(Height, Male) = (6+5.92+5.58+5.92)/4

= 5.855

Mean(Height, Female) = (5+5.5+5.42+5.75)/4

= 5.4175

Mean(Weight, Male) = (180+190+170+165)/4

= 176.25

Mean(Weight, Female) = (100+150+130+150)/4

= 132.5

Mean(Foot, Male) = (12+11+12+10)/4

= 11.25

Mean(Foot, Female) = (6+8+7+9)/4

= 7.5

Variance, denoted as S^2 or σ^2, is calculated as:

$$S^2 = \frac{\sum(x_i - \bar{x})^2}{n-1}$$

Where,

- x is the data's mean
- n is the number of records
- X_i is the value of ith record

Variance(Height, Male) = 0.30533 ; Variance(Height, Female) = 0.097225

Variance(Weight, Male) = 122.92 ; Variance(Weight, Female) = 558.33

Variance(Foot, Male) = 0.91667 ; Variance(Foot, Female) = 1.6667

According to Gaussian Naïve Bayes algorithm, likelihood is calculated as:

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

In order to find the solution substitute x_i values as 6, 130, 8 for the values of likelihood for height, weight and foot respectively as follows:

P(Height|Male) = 1.5789 (approx.) ; P(Height|Female) = 0.22346

P(Weight|Male) = 5.9811 x 10^{-6} ; P(Weight|Female) = 1.6789 x 10^{-2}

P(Foot|Male) = 1.3112 x 10^{-3}; P(Foot|Female) = 0.28669

Posterior Calculations:

Male:

P(Male|X) = P(Male)[P(Weight|Male)*P(Foot|Male)*P(Height|Male)] / [P(Male)[P(Weight|Male)*P(Foot|Male)*P(Height|Male)] + P(Female)[P(Weight|Female)*P(Foot|Female)*P(Height|Female)]

By computing further using the above obtained results, we get:

Numerator of P(Male|X) = 6.194 x 10^{-9}

Female:

P(Female|X) = P(Female)[P(Weight|Female)*P(Foot|Female)*P(Height|Female)] / [P(Male)[P(Weight|Male)*P(Foot|Male)*P(Height|Male)] + P(Female)[P(Weight|Female)*P(Foot|Female)*P(Height|Female)]

Numerator of P(Female|X) = 5.3778 x 10^{-4}

Since the value of posterior value of female is greater than that of the posterior value of male. The given data belongs to Female class.

SOLUTION is **FEMALE.**

3.6.6 Advantages of Naive Bayes

- Implementation of the model is easy
- Computation time is very less since a smaller number of computations are involves
- Estimation of parameters require very small amount of training data. Hence, training will be completed soon.

- Handles real as well as discrete data.
- Not sensitive to irrelevant features.
- In most of the cases, good results will be shown.

3.6.7 Disadvantages of Naive Bayes

- Accuracy is less because of the class independence assumption.
- Assumptions made in building the model are not often ideal in the real-time scenarios.
- Partial dependencies among the variables/attributes may exist. E.g., Symptoms: headache, cough, fever, etc. , Profile: age, Family history.
- These dependencies are not dealt in the Naïve Bayes classifier
- The precision will decrease if the dataset is small.

3.7 Support Vector Machine Classifier

Among all the supervised machine learning algorithms, SVM was critically acclaimed the best because of its numerous successful applications. These are based on a statistical approach unlike other classification techniques. The main difference lies in its way of selection of decision boundary. They are mainly known for classification rather than regression. Some of its applications include classification of news, articles, emails etc.., character recognition, bioinformatics and digital image analysis.

If we have ‘n' features in an SVM classifier, each data item is displayed as a point in n-dimensional space, with each value of the feature representing the value of a certain coordinate. Then they construct hyperplane or a set of hyperplanes which is/are used to differentiate between classes. Intrinsically, SVM is a binary classifier but after several contributions by scientists, it is being adapted to multi-class as well.

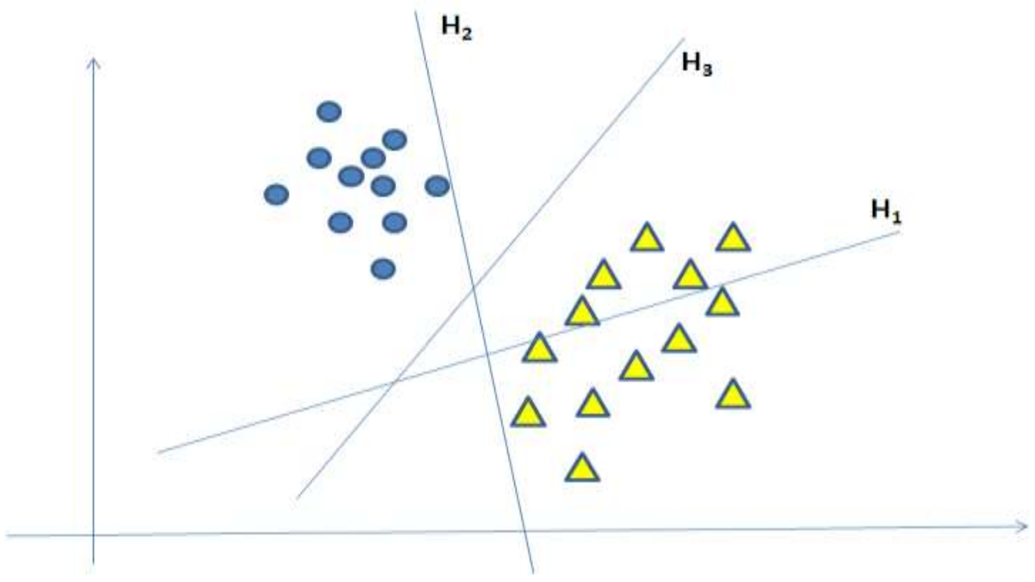


Figure 3.10: Different hyperplanes dividing data into two separate classes

SVM chooses a hyperplane from the above diagram that offers the greatest margin distance between the spots closest to the class. This separation is called as linear classification. So here 'H3' is the maximum margin hyperplane and such a classifier which uses this technique is known as maximum margin classifier.

3.7.1 History

In 1936 R. A. Fisher recommended the first algorithm for pattern recognition. Later in 1957, '*Perceptron*', a notable linear classifier was invented by Frank Rosemblat which adopted feed forward neural network. In 1963, A generalized portrait algorithm was introduced by Lerner and Vapnik. Current SVM algorithm is a non-linear generalization of this one. In 1965 First discussion about large margin hyperplanes came into picture. Later in 1974, Statistical learning theory was first developed and proposed by Vapnik and Chervonenkis. In 1979, The concepts of SVM were first introduced in a naïve fashion. In 1992, SVMs near their current shape were first brought with a paper presented at the COLT conference.[17]

3.7.2 Terminology

- ***Decision Boundary:*** The boundary which differentiates between the two classes. Technically it is known as Hyperplane. It has (n-1) dimensions if the data has 'n' features.

- ***Support Vectors:*** Those data points from either class that are near to the decision border. They have an impact on the hyperplane's location and orientation.
- ***Margin Width:*** The maximum width of the band that can be drawn to separate the two classes' support vectors.
- ***Hard Margin:*** Implies that model is very rigid in classification, worked extremely well in training and therefore causes overfitting.
- ***Soft Margin:*** This allows the model to make certain mistakes intentionally so that overfitting could be avoided. It is achieved by slightly modifying the objective of SVM.

3.7.3 Working of Linear SVM

Let's start with a basic understanding of what a hyperplane is. It may be thought of as a function that is used to distinguish between characteristics. For instance, in 2D, the function represents a line; in 3D, it represents a plane; and in higher dimensions, it represents a hyperplane.

Consider the fact that the data has ‘m' dimensions.

The hyperplane equation in the ‘m' dimension is as follows:

$$y = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \ldots\ldots\ldots\ldots + w_mx_m$$

$$y = w_0 + \sum^{m}_{i=1} w_ix_i$$

$$y = w_0 + w^T * X$$

$$y = b + w^T * X$$

where, X = features

w^T = transpose of matrix of vectors that represent slope

b = constant or intercept

Now let’s derive the condition for optimisation:

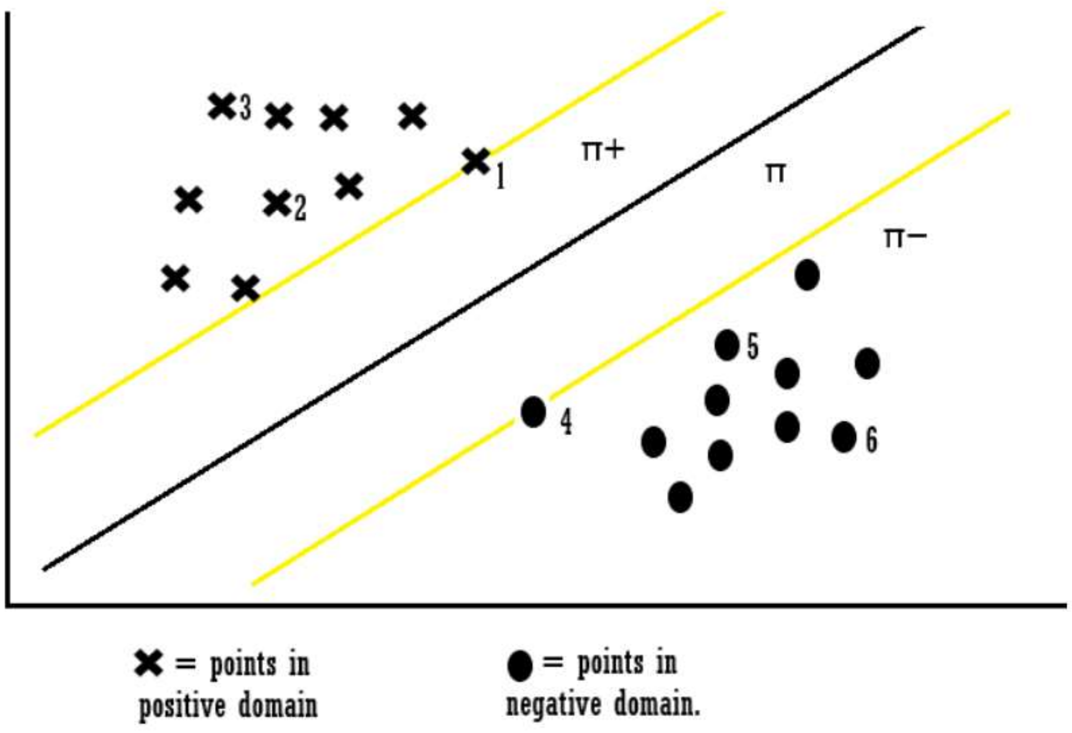


Figure 3.11: Sample data for SVM (1)

Assume there are three hyperplanes, namely,, + and, where '+' is parallel to '' passing through the support vectors on the positive side and '' is parallel to '' passing through the support vectors on the negative side, as shown in figure 3.11.

As a result, each hyperplane's equations will be:

$$\pi = b + w^T * X = 0$$

$$\pi + = b + w^T * X = 1$$

$$\pi - = b + w^T * X = -1$$

From the equations we get to know that all the points in the area above $\pi+$ belong to positive class and all the points in the area below $\pi-$ belong to negative class.

Consider two data points x1 in the positive class and x2 in the negative class respectively. So the equations would be as follows:

$$b + w^T * x1 = 1 \quad \text{(Equation 1)}$$

$$b + w^T * x2 = -1 \quad \text{(Equation 2)}$$

Subtracting both the equations we get:

$$w^T(x2 - x1) = 2 \quad \text{(Equation 3)}$$

Dividing equation3 on both sides with the norm of the vectors which is often referred to as magnitude and is denoted by "|| ||".

$$(w^T/||w||)\,(x2 - x1) = 2 / ||w|| \quad \text{(Equation 4)}$$

We know that $(w^T/||w||)$ equals 1. And also let us remind ourselves that according to SVM classifier the margin should be maximum. So, the RHS of equation 4 should be maximum.

We may deduce that we need to find (w, b) such that 2 / ||w|| is the maximum and yi = 1 when b + wT * x >= 1.

$$y_i = -1 \text{ when } \quad b + w^T * x <= -1$$

The above both equations can be combined and can be represented together as:

$$y_i * (b + w^T * x) >= 1 \quad \text{(Equation 5)}$$

If the above condition is not satisfied then it comes under a misclassification.

For easy mathematical calculations, we can also find (w, b) such that ||w|| / 2 is minimum given it follows the condition from equation 5.

Let's look at an example:

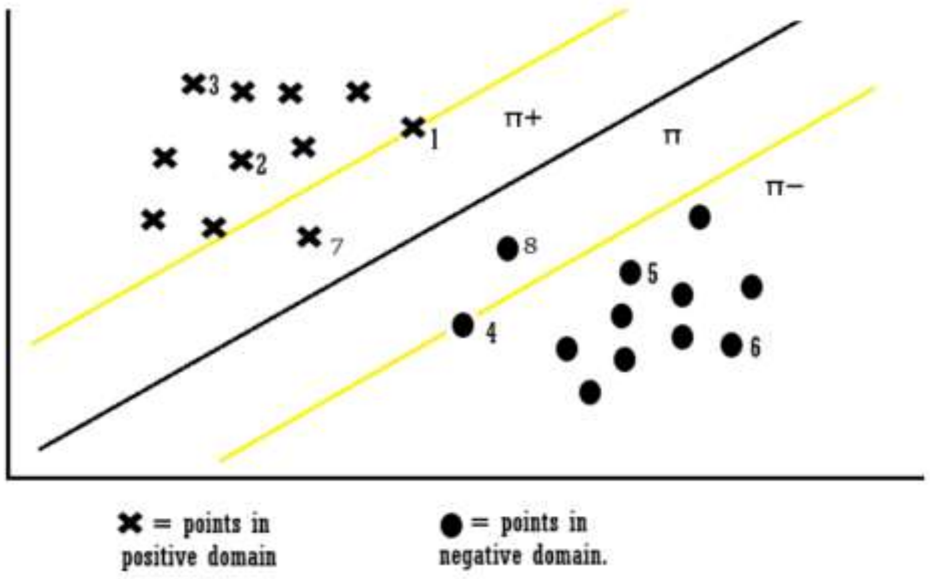


Figure 3.12: Sample data for SVM (2)

From the above figure,

- x1 and x4 are support vectors
- π+ is positive hyperplane and π- is negative hyperplane
- x7 and x8 are misclassified
- Rest of the points are classified correctly

From the above analysis we can conclude that our hyperplane will be able to distinguish only linearly seperable points. We should follow very strict constraints to classify all data points. This type of classifier is known as ***"Hard Margin SVM."***

In real life scenario, we may not encounter cases which are perfectly separable in a linear fashion. We need to update our function so that it will be able to classify almost linearly separable data and it might skip some outliers too. This comes under ***"Soft Margin SVM."***

Therefore, there is a new varaible introduced which is called as Slack variable and is denoted by ξ. Equation 5 can be re-written as:

$$y_i * (b + w^T * x) >= 1 - \xi_i \quad \text{(Equation 6)}$$

We can decide how many errors we can manage to introduce previously itself and modify the equation accordingly.

If $\xi_i = 0$, it indicates that all the points are classified correctly.

If $\xi_i > 0$, it indicates that there might be some points which are incorrectly classified.

When $\xi_i > 0$, it implies that the related variables Xi are in the wrong dimension, and this may be seen as the mistake connected with variable Xi. The average error is computed as follows:

$$\frac{1}{n}\sum_{i=1}^{n} \xi_i$$

Therefore, this average error should be added to our function.Now we need to find the value of (w, b) such that

$$\underset{w,b}{minimize}\ \frac{1}{2}\,||w||^2 + \sum_{i=1}^{n} \zeta_i$$

$$such\ that\ y_i\left(w^T.X_i + b\right) \geq 1 - \zeta_i \quad For\ all\ i = 1, 2, \ldots.., n$$

This concept is known as regularization. To summarise, the soft margin approach tries to identify the vector w and scalar b values that maximise the margin distance and minimise the loss term in the hyperplane represented by w and b.

3.7.4 Loss Function Interpretation:

A loss function called hinge loss is used to train classifiers. There is a tiny hinge loss if a calculated value yields an accurate classification but is too near to zero (where too close is determined by a margin). There will always be a hinge loss if a calculated value results in an erroneous categorization. For a margin-based classifier, this is the concept of hinge loss.

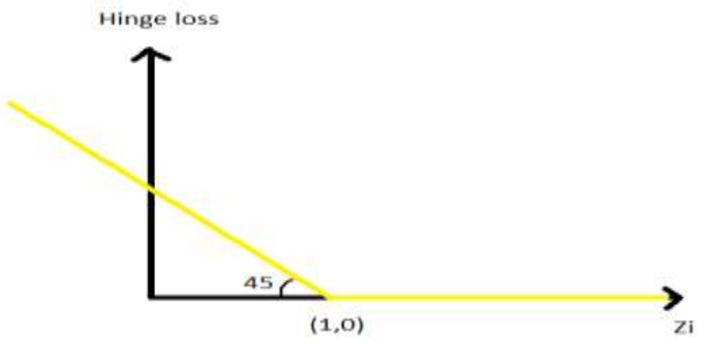


Figure 3.13: Loss function plot

$Z_i = y_i * (b + w^T * x_i)$

When Zi >= 1, the loss equals zero.

When Zi < 1, loss increases.

As a result, we may deduce that Hinge loss equals the maximum of (0, 1-Zi).

3.7.5 For Non-linear Data

When our data is not linearly separable at any cost, then the above two functions are not of any use. For Example, consider the figure xxx. If the data is like that we have to explore new methods unlike the previous ones.

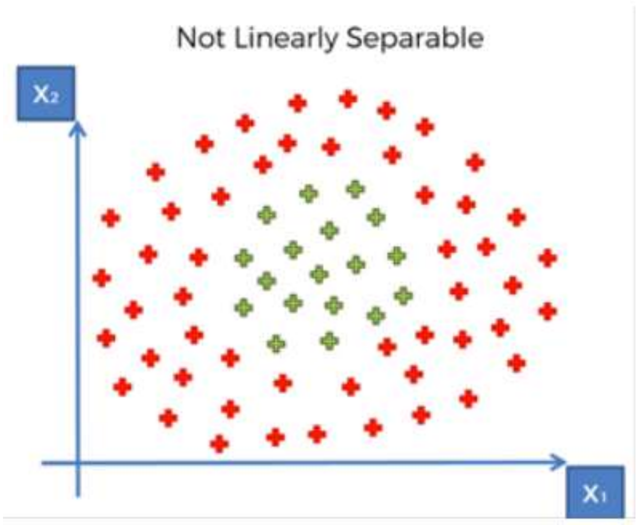


Figure 3.14: Non-Linear data points

There are two methods using which we can classify these points.

3.7.5.1 Method A - Mapping to a higher dimension

Steps:

- Take a non-linear dataset
- Map it to a higher dimension using a mapping function.
- Then we have a dataset that is linearly separable.
- Activate the SVM classifier
- Create a decision criterion.
- When we project that onto the original dataset, we can see that the classifier is obviously non-linear.

Let us look at further examples for proper understanding of the steps.

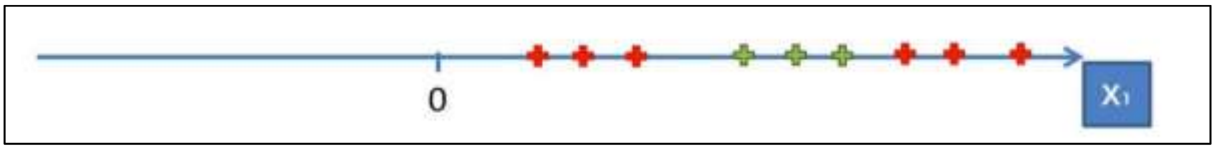


Figure 3.15: Data on a 1D-plane

From the above figure 3.15, it is clear that the data points are non-linear because here the hyperplane is a dot. Therefore, if we map these points on a 2D-plane using a function, say $(x-5)^2$ we can separate them using a hyperplane in one dimension i.e. line. Figure 3.16 illustrates this.

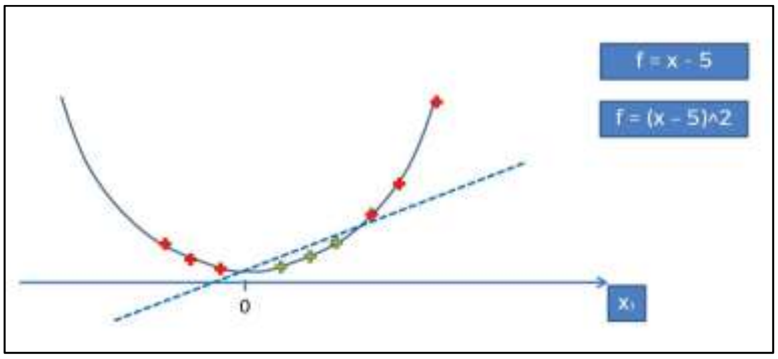


Figure3.16: Linearly separable data when mapped to a higher dimension

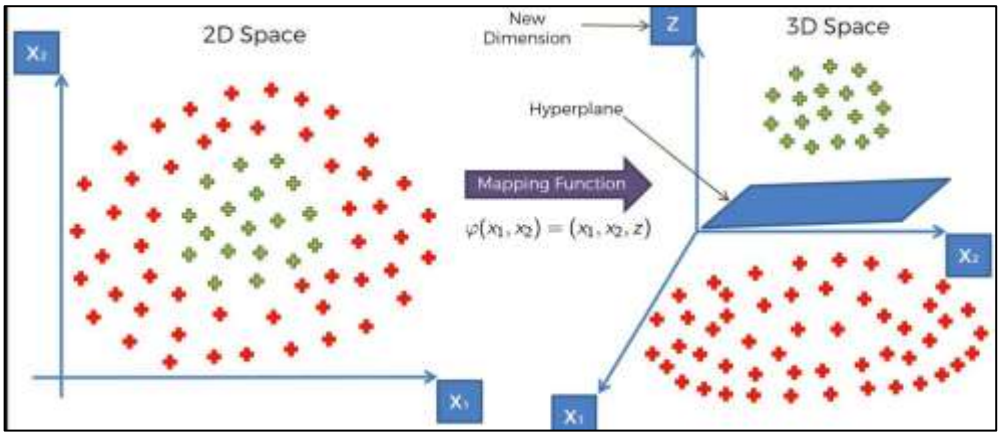


Figure 3.17: Mapping of 2D data to 3D

Figure 3.17 illustrates how non-linear data on 2D-plane is converted into separable data in 3D space using a hyperplane.

When we project back onto the original plane, it looks as shown in figure 3.18.

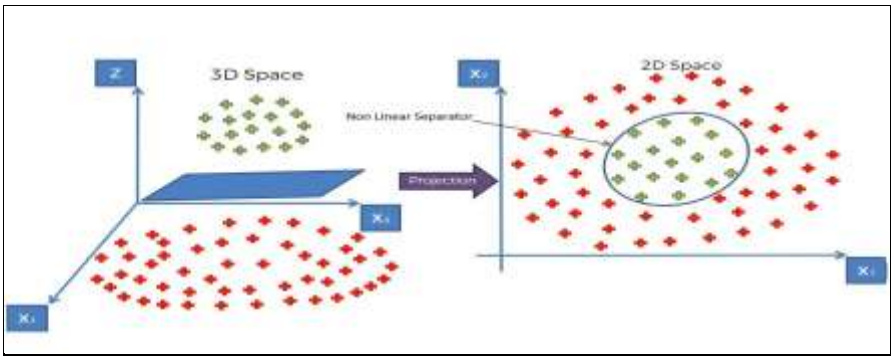


Figure 3.18: Projection of 3D to 2D

Therefore, we get a non-linear decision boundary.

However, mapping to a higher-dimensional space might be time-consuming. It has the potential to cause significant delays and processing backlogs.

SVM does this easily by using Kernel Trick.

3.7.5.2 Method B - Kernel Trick

Generalized dot product is another name for kernel functions. It's a method of computing the dot product of two vectors, x and y, in a (very high-dimensional) feature space. Using the kernel technique simply means using the kernel function to replace the dot product of two vectors. The kernel technique aids in the development of a more accurate classifier.

Types of Kernel Functions:

There are numerous types of Kernel functions. But the most widely used ones are:

- Polynomial Kernel
- Gaussian Kernel / Radial basis function kernel

Polynomial Kernel:

Dot product is obtained in Polynomial Kernel by raising the kernel's power. This is how it's defined:

$$K(X_1, X_2) = (a + X_1^T X_2)^b$$

Where, a = constant and b = degree of kernel

Let's suppose X space was initially two-dimensional, with Xa = (a1, a2) and Xb = (a1, a2) (b1, b2)

Now, if we wish to translate our data into a higher dimension, such as Z space, we may use the method above to do it directly.

$$Z_a^T Z_b = k(X_a, X_b) = (1 + X_a^T X_b)^2$$
$$Z_a^T Z_b = 1 + a_1 b_1 + a_2 b_2 + a_1^2 b_1^2 + a_2^2 b_2^2 + a_1 b_1 a_2 b_2$$

Gaussian Kernel or RBF Kernel:

The RBF kernel is a function whose value is proportional to the distance between the origin and a given location. This is how it's defined:

$$K(X_1, X_2) = exponent(-\gamma\|X_1 - X_2\|^2)$$

||X1 — X2 || = Euclidean distance between X1 & X2.

The dot product is calculated using the distance in the original space.

Here, ‘γ' is a constant number that specifies the circumferential width. It is only present in RBF Kernel. As the value of ‘γ' grows larger, the circumference grows larger, and the model may become over fit. As the value of ‘γ' falls, the circumference shrinks, and the model may become under fit.

Therefore, if the value calculated is greater than zero it comes under positive class and if the value calculated is equal to zero (nearly) it comes under negative class. Figure xxx illustrates Gaussian Kernel pictorial representation.

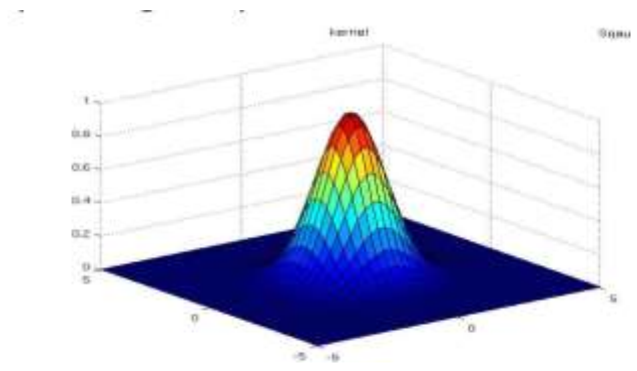


Figure 3.19: Gaussian Kernel representation

3.7.6 Tuning Hyper Parameters

Some of the parameters we can add while building the model are:

- ***Kernel:*** We can choose from a variety of kernel types. For non-linear hyperplane, polynomial and RBF are useful.
- ***Regularization:*** It is represented by ‘C’. It refers to a misclassification or an erroneous word. We can control the trade-off between the decision boundary and the misclassification term using this method. As the value of ‘C' rises, the

margin hyperplane expands potentially leading to overfitting. As the value of ‘C’ falls, the margin hyperplane shrinks, potentially resulting in underfitting.

- ***Gamma(γ):*** Used only in RBF kernel and is discussed in that section.

3.7.7 Advantages of SVM

- Highly accurate compared to several other models
- SVM can handle many features
- It is robust with respect to high input dimension
- Irrelevant features are very less
- For data that isn't linearly separable, we can utilise kernels to map it into a high-dimensional space and make it linearly separable using hyperplanes.
- Memory efficient due to the utilisation of a subset of training data (support vector) as significant classification variables.

3.7.8 Disadvantages of SVM

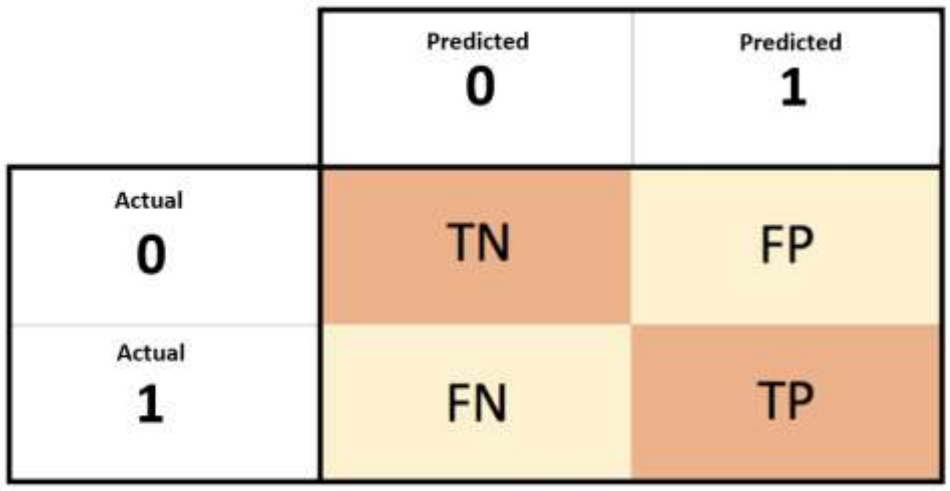
- Processing speed is very low
- Not much efficient with large datasets
- If the features are more in number compared to samples, then it leads to poor performance of the model
- Probability estimates are to be indirectly calculated as SVM does not provide them.
- Training and Testing of the model is very time consuming
- Very difficult to understand
- Choice of kernel may be the biggest disadvantage

IV. IMPLEMENTATION AND EVALUATION PARAMETERS

4.1 Confusion Matrix

The Confusion Matrix is a matrix-based approach for describing the performance of a model or classification system. The matrix is NxN, with N being the number of classes. This field contains the number of correct and erroneous categorization model predictions. A confusion matrix has each row representing an actual class and each column representing a projected class.

In 1971, Townsend was the first to publish a study on Confusion Matrix. The article describes an experiment with 26 English alphabets, each of which must respond with the same letter, creating a 26x26 confusion matrix. However, in the year 1998, Kohavi and Provost popularized the phrase in Machine Learning.



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

Figure 4.1: Confusion matrix representation

4.1.1 Terminology

- ***True Positive (TP):*** We anticipate that the output belongs to a particular class, and it does.

- ***True Negative (TN):*** We expect the result to be unclassifiable, and it is.
- ***False Positive (FP):*** The output is predicted to belong to a class, but it does not really belong to that class. TYPE I ERROR is another name for this.
- ***False Negative (FN):*** Although we forecast that the result does not belong to a class, it really does. TYPE II ERROR is another name for this.

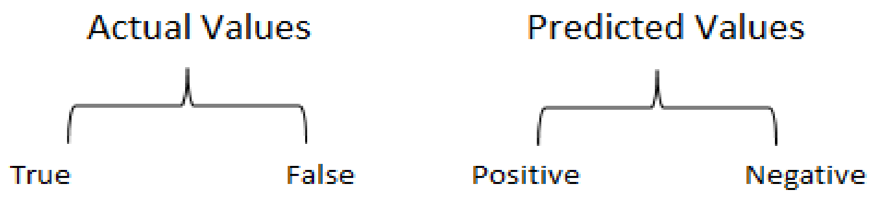


Figure 4.2: Actual values Vs Predicted values

- ***Accuracy***: Tells us how often the classifier is right in its predictions.

  Accuracy = (TP+TN) / Total

  = (TP+TN) / (TP+TN+FP+FN)
- ***Misclassification Rate:*** Indicates how frequently the classifier guesses wrong.

  Misclassification Rate = (FP+FN)/total

  = 1 - Accuracy

  It is also known as "Error Rate"
- ***True Positive Rate:*** When an output is really yes, it tells us how often it predicts yes.

  TP/actual yes = TP/(TP+FN)

  It is also known as "Sensitivity" or "Recall"
- ***False Positive Rate:*** When an output is really no, it tells us how often it predicts yes.

  FP/actual no =FP/(TN+FP)
- ***True Negative Rate:*** When an output is really no, it tells us how frequently it predicts no.

  TN/actual no = TN/(TN+FP)

  = 1 - False Positive Rate

  It is also known as "Specificity"
- ***Precision:*** When the model predicts yes, it tells us how frequently it is right.

TP/predicted yes = TP/(TP+FP)

- ***Prevalence:*** Indicates how frequently the yes condition occurs in our sample.

  actual yes/total = (TP+FN) / (TP+TN+FP+FN)

4.2 Evaluation Parameters

- ***Accuracy:*** The percentage of correct predictions throughout the model's testing phase is known as accuracy. It's calculated as the proportion of correct forecasts to total predictions.

  Accuracy = (Correct predictions)/ (total predictions) * 100

- ***Precision:*** Precision is the ratio or proportion of relevant instances, i.e., true positives, to the total number of examples. I.e., the total number of true positives and false positives that were projected to belong to a specific class.

  Precision = true positives/ (true positives + false positives)

- ***Recall:*** The ratio of predicted instances that belong to a specific class to the total predictions that actually belong to that class is known as recall.

  Recall = true positives/ (true positives + false negatives)

- ***F1 Score:*** The F1 score, commonly known as the F measure, combines the harmonic mean of the accuracy and recall components into a single parameter. Low false positives and false negatives are indicators of a strong F1 score. The F1 score ranges from 0 to 1. As a result, the F1 score for a properly functioning system/model is 1; if it is 0, the model is a complete failure.

  F1 Score = 2*((precision*recall)/ (precision + recall))

- ***Matthews Correlation Coefficient:*** The MCC tells us how well the observed and anticipated binary categories match up. MCC is a scale that ranges from –1 to +1. -1 means that the model is very bad, 0 indicates that the model isn't much better than random guesses, and +1 indicates that the model is great. The MCC value also reflects the fairness of the findings in all four confusion matrix areas.

$$\mathrm{MCC} = \frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

4.3 Implementation

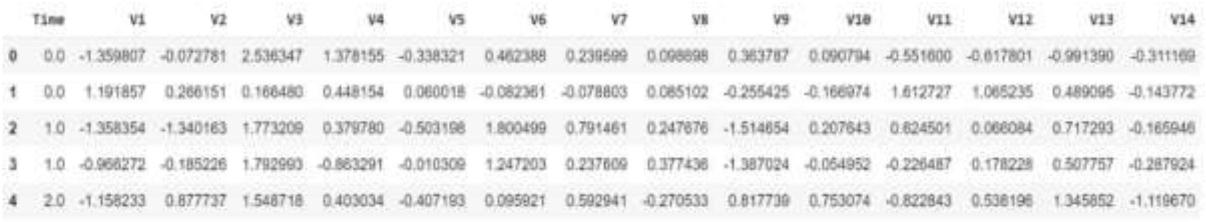
Initial steps involve data manipulation

a. *Importing the required libraries*

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import gridspec
```

b. ***Downloading, Loading and Understanding the data***

```
data = pd.read_csv(“creditcard.csv”)
data.head()
```

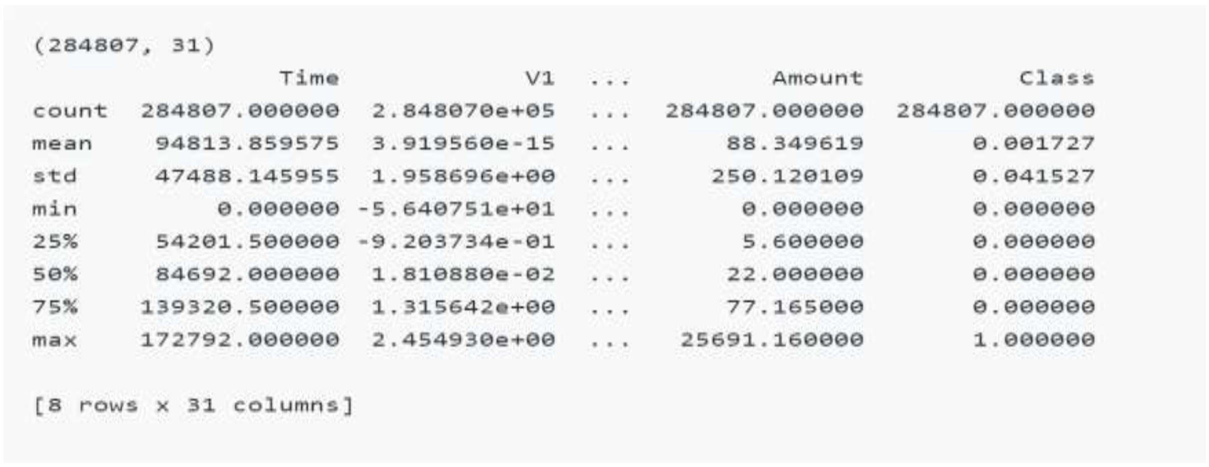


	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670

Figure 4.3: Structure of the dataset is printed

c. ***Describing the data***

```
print(data.shape)
print(data.describe())
```



```
(284807, 31)
                Time            V1  ...         Amount          Class
count  284807.000000  2.848070e+05  ...  284807.000000  284807.000000
mean    94813.859575  3.919560e-15  ...      88.349619       0.001727
std     47488.145955  1.958696e+00  ...     250.120109       0.041527
min         0.000000 -5.640751e+01  ...       0.000000       0.000000
25%     54201.500000 -9.203734e-01  ...       5.600000       0.000000
50%     84692.000000  1.810880e-02  ...      22.000000       0.000000
75%    139320.500000  1.315642e+00  ...      77.165000       0.000000
max    172792.000000  2.454930e+00  ...   25691.160000       1.000000

[8 rows x 31 columns]
```

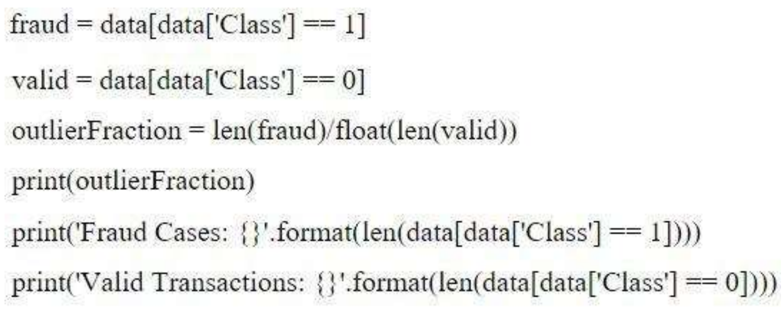
Figure 4.4: Shape of the dataset is printed

Here,   std = standard deviation of the data

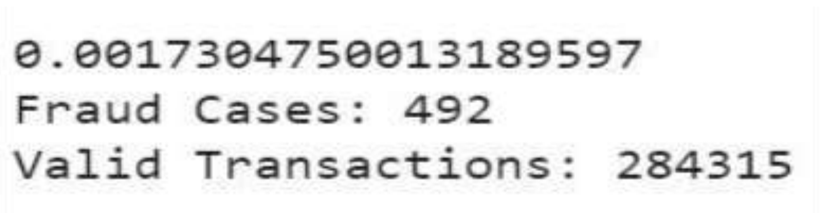
min = minimum transaction details

max = maximum transaction details

d. *Imbalance in the data*



```
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]
outlierFraction = len(fraud)/float(len(valid))
print(outlierFraction)
print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))
```



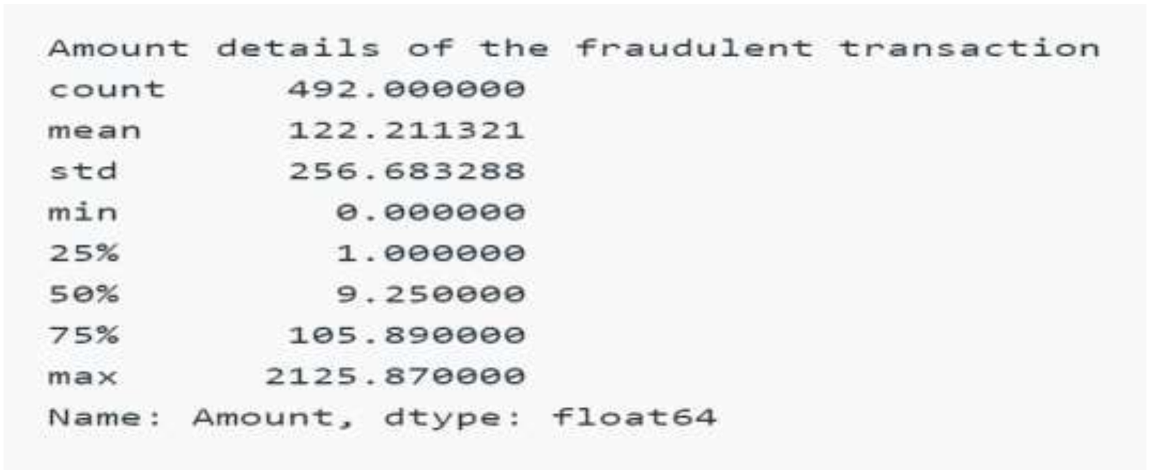
```
0.0017304750013189597
Fraud Cases: 492
Valid Transactions: 284315
```

Figure 4.5: Imbalance

Since the data is highly imbalanced, first we apply this data directly to the model without balancing it. If the accuracy is not good then we will balance the data.

e. *Fraudulent transaction's amount details*

```
print("Amount details of the fraudulent transaction")
fraud.Amount.describe()
```

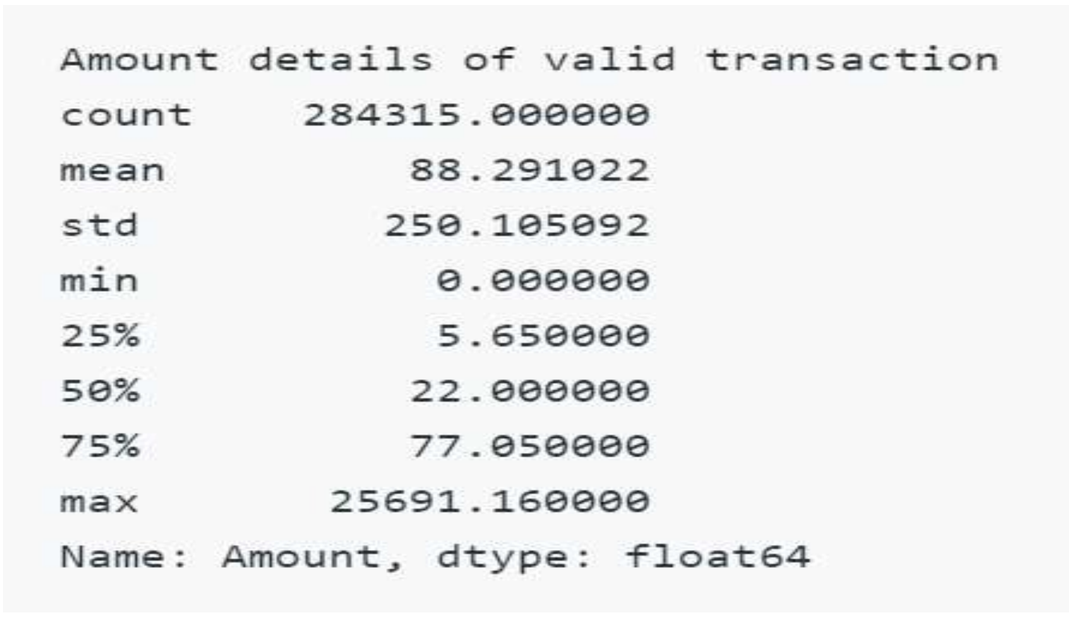


```
Amount details of the fraudulent transaction
count      492.000000
mean       122.211321
std        256.683288
min          0.000000
25%          1.000000
50%          9.250000
75%        105.890000
max       2125.870000
Name: Amount, dtype: float64
```

Figure 4.6: Amount details of fraudulent transaction

f. *Normal transaction's amount details*

```
print("details of valid transaction")
valid.Amount.describe()
```



```
Amount details of valid transaction
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

Figure 4.7: Amount details of valid transaction

g. *Dividing the data into training and testing data subsets*

```
                    X = data.drop(['Class'], axis = 1)
                    Y = data["Class"]
                    print(X.shape)
                    print(Y.shape)
                    xData = X.values
                   yData = Y.values
 from sklearn.model_selection import train_test_split
xTrain, xTest, yTrain, yTest = train_test_split(xData,yData, test_size =0.2,
                                                  random_state =42)
```

- The data is separated into two sections, X and Y. The X data is used to train the model, while the Y data is used to test it.

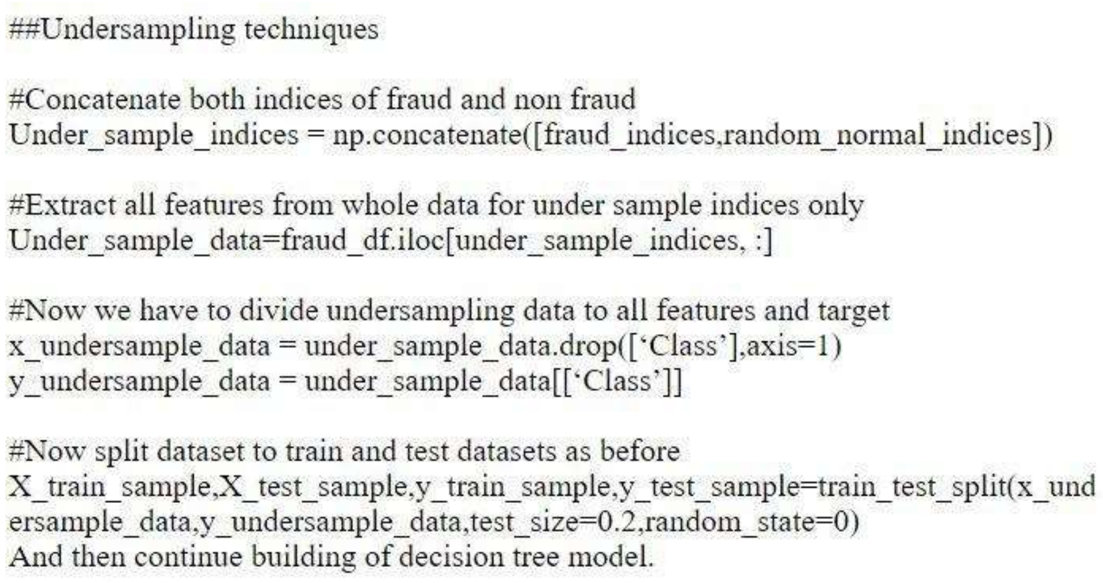
h. *Building the models:*

1. **DECISION TREE**

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

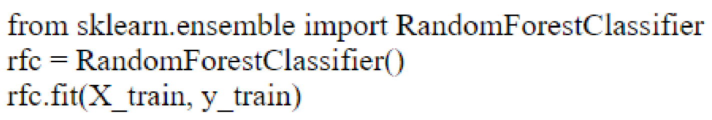
2. **DECISION TREE WITH UNDERSAMPLING**

```
## Equal both the target samples to the same level
# Take indexes of non fraudulent
    nonfraud_indexies = fraud_df[fraud_df.class == 0].index
    fraud_indices = np.array(fraud_df[fraud_df.‘Class’] == 1].index)

# Take random samples from non-fraudulent that are equal to fraudulent samples
 random_normal_indexies=np.random.choice(nonfraud_indexies,fraud,replace=False)
 random_normal_indexies=np.array(random_normal_indexies)
```



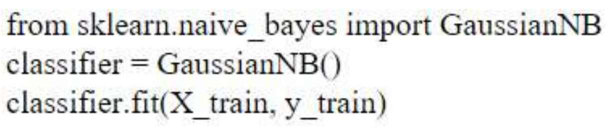
```
##Undersampling techniques

#Concatenate both indices of fraud and non fraud
Under_sample_indices = np.concatenate([fraud_indices,random_normal_indices])

#Extract all features from whole data for under sample indices only
Under_sample_data=fraud_df.iloc[under_sample_indices, :]

#Now we have to divide undersampling data to all features and target
x_undersample_data = under_sample_data.drop(['Class'],axis=1)
y_undersample_data = under_sample_data[['Class']]

#Now split dataset to train and test datasets as before
X_train_sample,X_test_sample,y_train_sample,y_test_sample=train_test_split(x_und
ersample_data,y_undersample_data,test_size=0.2,random_state=0)
And then continue building of decision tree model.
```

3. **RANDOM FOREST**



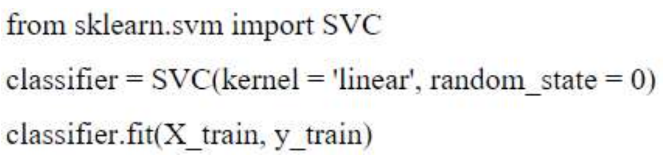
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
```

4. **NAIVE BAYES**



```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

5. **SUPPORT VECTOR MACHINE**



```
from sklearn.svm import SVC

classifier = SVC(kernel = 'linear', random_state = 0)

classifier.fit(X_train, y_train)
```

i. *Make Predictions on Test Data*

```
y_pred = classifier.predict(X_test)
```

j. *Building the confusion matrix*

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

k. *Displaying the confusion matrix*

```
LABELS = ['Normal', 'Fraud']
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize =(12, 12))
sns.heatmap(conf_matrix, xticklabels = LABELS, yticklabels = LABELS, annot = True, fmt
="d");
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
```

l. *Values of Evaluation Parameters*

```
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix

n_errors = (y_pred != y_test).sum()
print("The model used is Naive Bayes classifier")
acc = accuracy_score(y_test, y_pred)
print("The accuracy is {}".format(acc))
prec = precision_score(y_test, y_pred)
print("The precision is {}".format(prec))
rec = recall_score(y_test, y_pred)
print("The recall is {}".format(rec))
f1 = f1_score(y_test, y_pred)
print("The F1-Score is {}".format(f1))
MCC = matthews_corrcoef(y_test, y_pred)
print("The Matthews correlation coefficient is {}".format(MCC))
```

V. RESULTS AND COMPARISON

5.1 Parameters of all Implemented models

a. Decision Tree

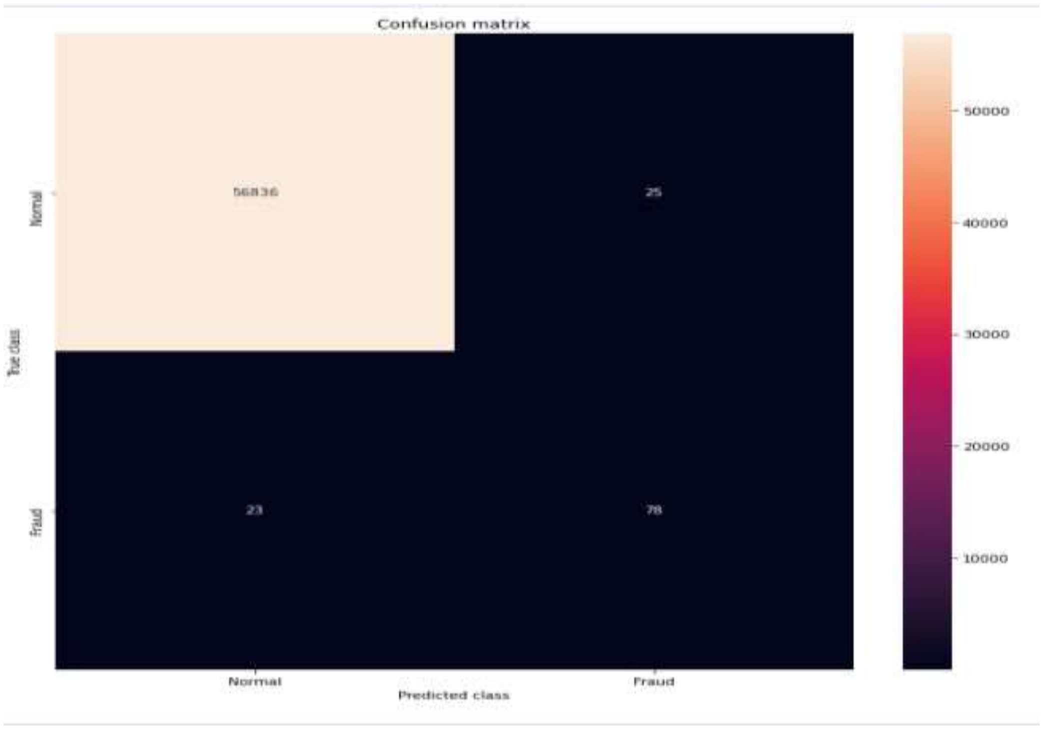
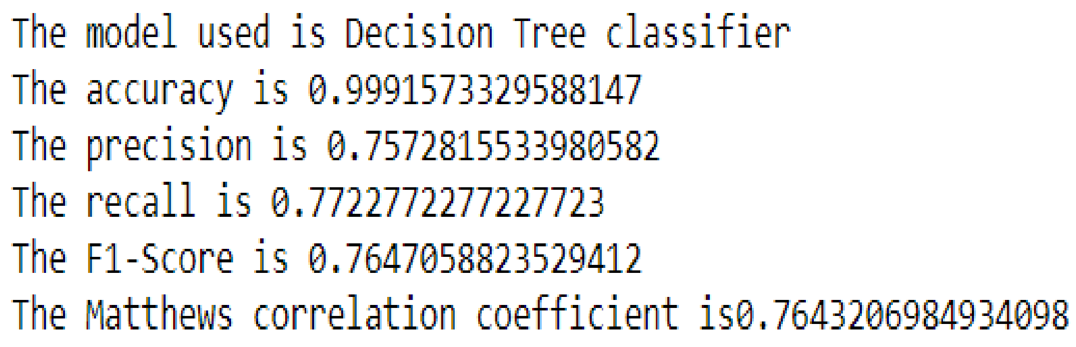


Figure 5.1: Confusion Matrix of decision tree classifier



```
The model used is Decision Tree classifier
The accuracy is 0.9991573329588147
The precision is 0.7572815533980582
The recall is 0.7722772277227723
The F1-Score is 0.7647058823529412
The Matthews correlation coefficient is0.7643206984934098
```

Figure 5.2: Different evaluation parameters of decision tree classifier

b. Decision Tree with Undersampling

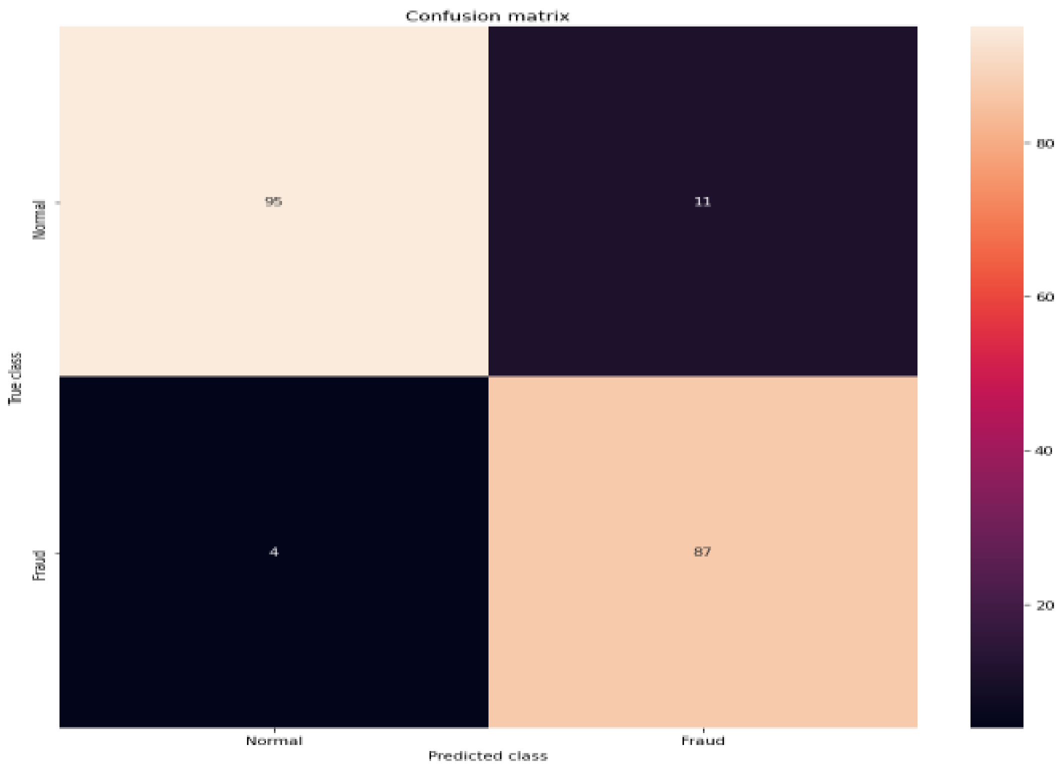
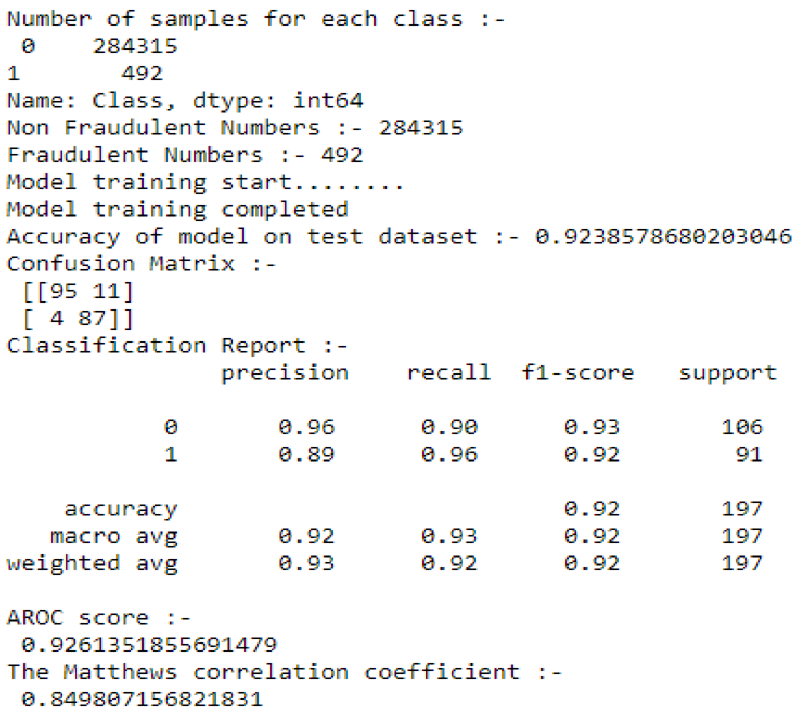


Figure 5.3: Confusion Matrix of decision tree classifier after undersampling



```
Number of samples for each class :-
 0    284315
1       492
Name: Class, dtype: int64
Non Fraudulent Numbers :- 284315
Fraudulent Numbers :- 492
Model training start........
Model training completed
Accuracy of model on test dataset :- 0.9238578680203046
Confusion Matrix :-
 [[95 11]
 [ 4 87]]
Classification Report :-
               precision    recall  f1-score   support

           0       0.96      0.90      0.93       106
           1       0.89      0.96      0.92        91

    accuracy                           0.92       197
   macro avg       0.92      0.93      0.92       197
weighted avg       0.93      0.92      0.92       197

AROC score :-
 0.9261351855691479
The Matthews correlation coefficient :-
 0.849807156821831
```

Figure 5.4: result of decision tree classifier with undersampling

c. Random Forest

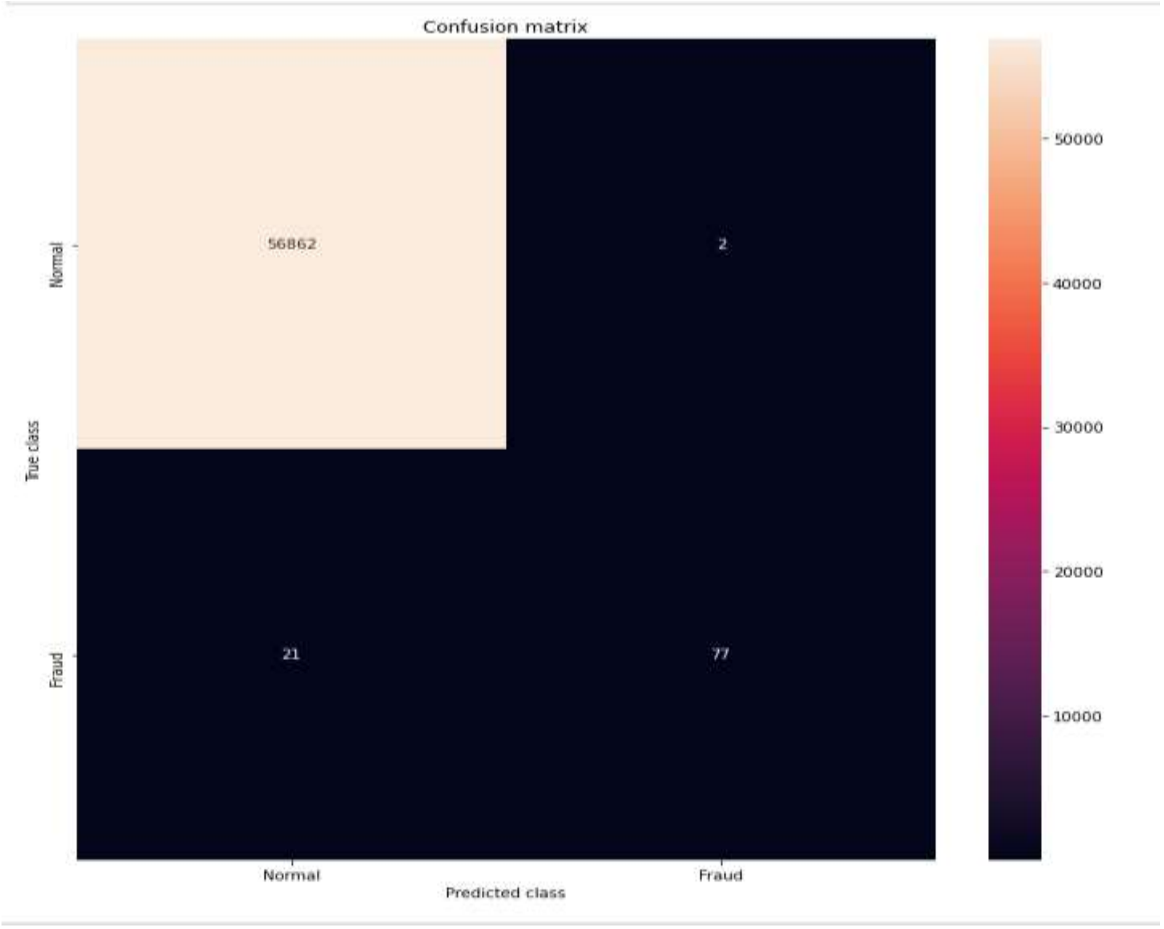
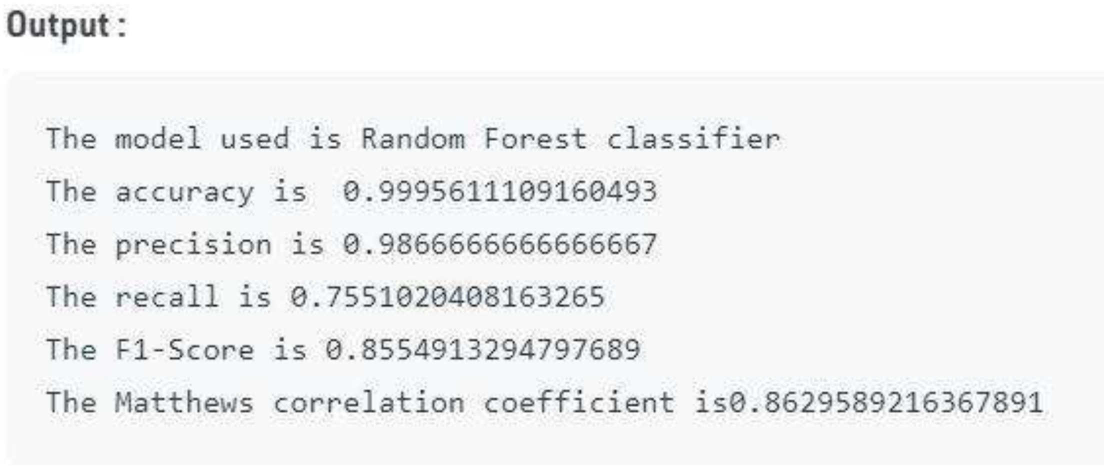


Figure 5.5: Confusion Matrix of Random forest classifier



Output :

```
The model used is Random Forest classifier
The accuracy is  0.9995611109160493
The precision is 0.9866666666666667
The recall is 0.7551020408163265
The F1-Score is 0.8554913294797689
The Matthews correlation coefficient is0.8629589216367891
```

Figure 5.6: Different evaluation parameters of Random forest classifier

d. Naive Bayes

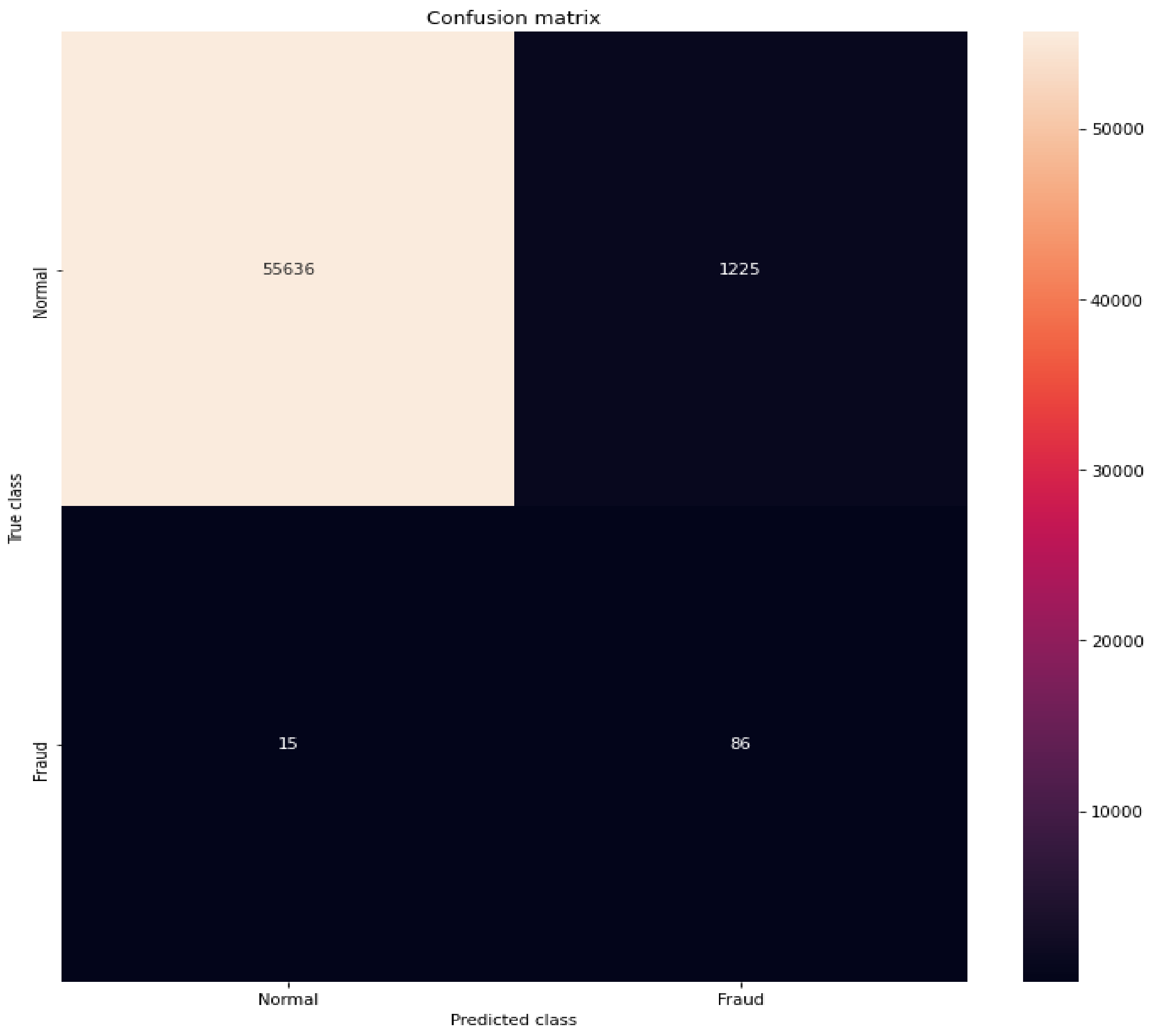
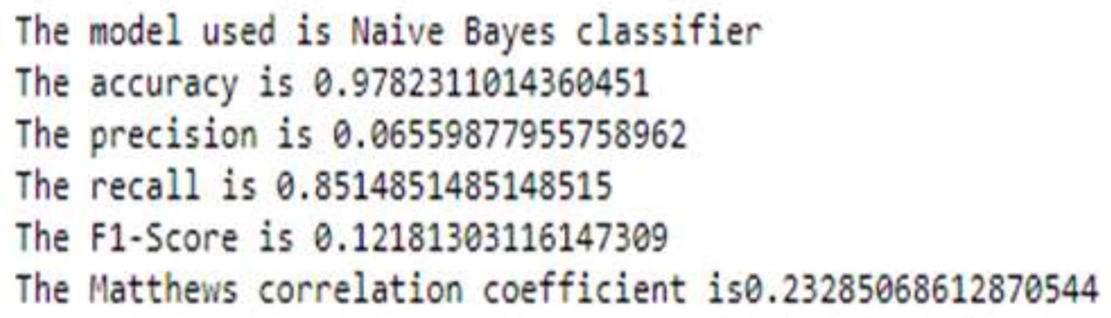


Figure 5.7: Confusion matrix of Naive Bayes classifier



```
The model used is Naive Bayes classifier
The accuracy is 0.9782311014360451
The precision is 0.06559877955758962
The recall is 0.8514851485148515
The F1-Score is 0.12181303116147309
The Matthews correlation coefficient is0.23285068612870544
```

Figure 5.8: Different evaluation parameters of Naive Bayes classifier

e. Support Vector Machine

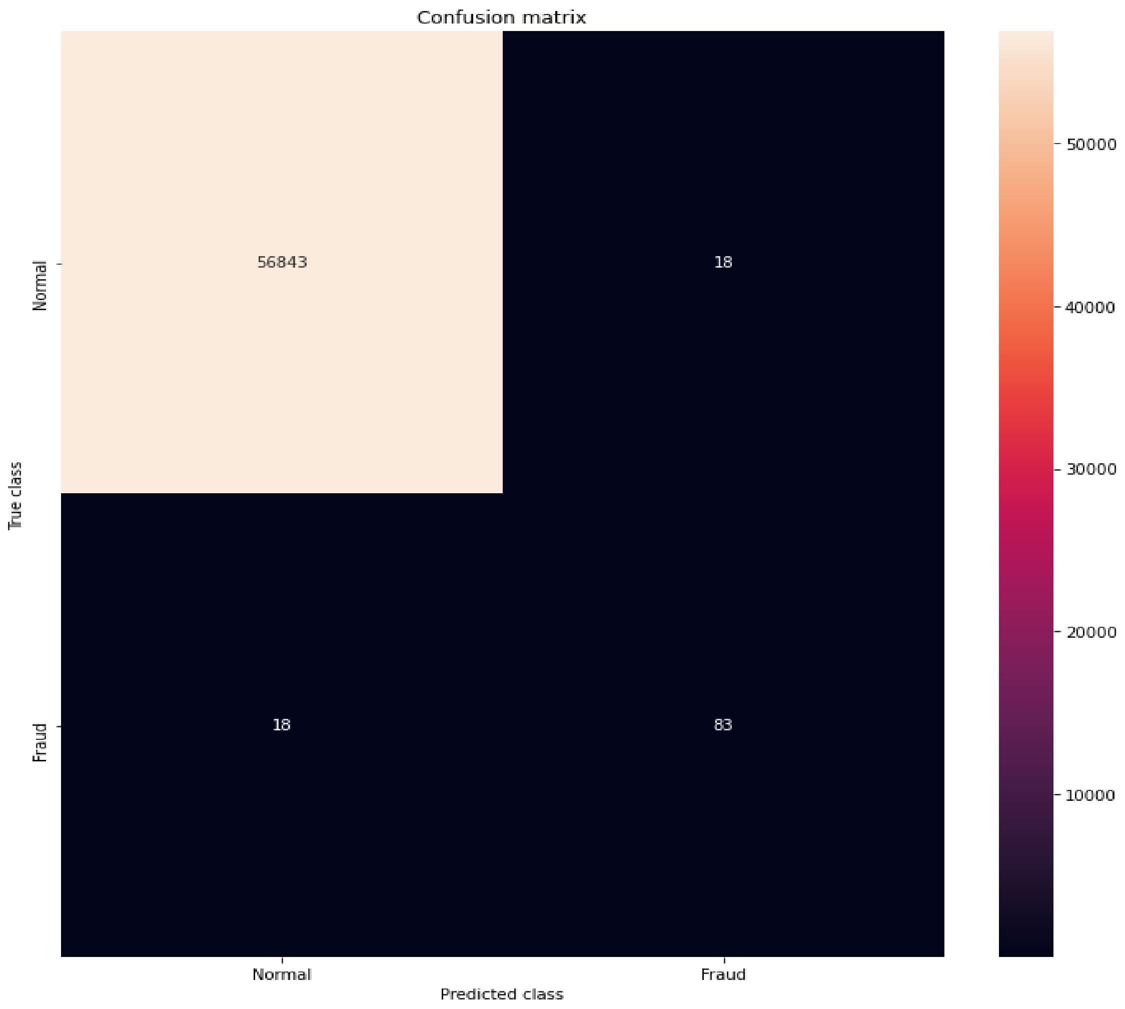
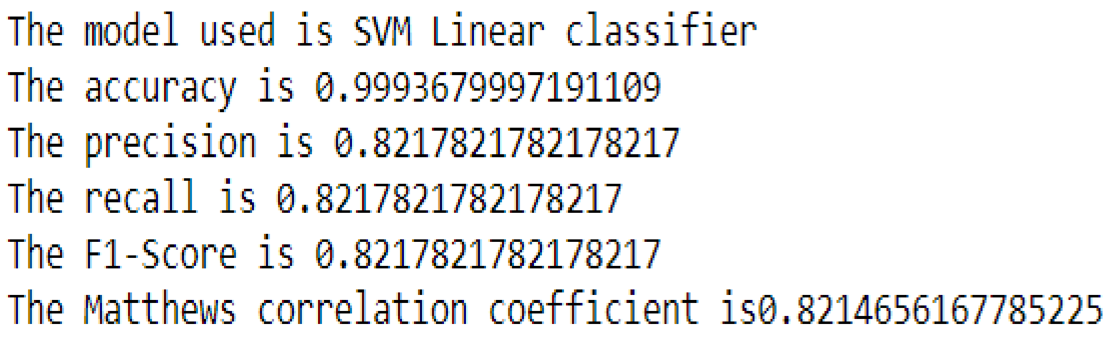


Figure 5.9: Confusion Matrix of SVM classifier



```
The model used is SVM Linear classifier
The accuracy is 0.9993679997191109
The precision is 0.8217821782178217
The recall is 0.8217821782178217
The F1-Score is 0.8217821782178217
The Matthews correlation coefficient is0.8214656167785225
```

Figure 5.10: Different evaluation parameters of SVM classifier

5.2 Comparison

Model	Accuracy	Precision	Recall	F1-Score	MCC
Decision Tree	0.9991	0.7572	0.7722	0.7647	0.7643
Decision Tree 2.0	0.9238	0.9600	0.9000	0.9300	0.8498
Random Forest	0.9995	0.9866	0.7551	0.8554	0.8629
Naive Bayes	0.9782	0.0655	0.8514	0.1218	0.2328
Support Vector Machine	0.9993	0.8217	0.8218	0.8217	0.8214

Table 5.1: Comparison of various parameters

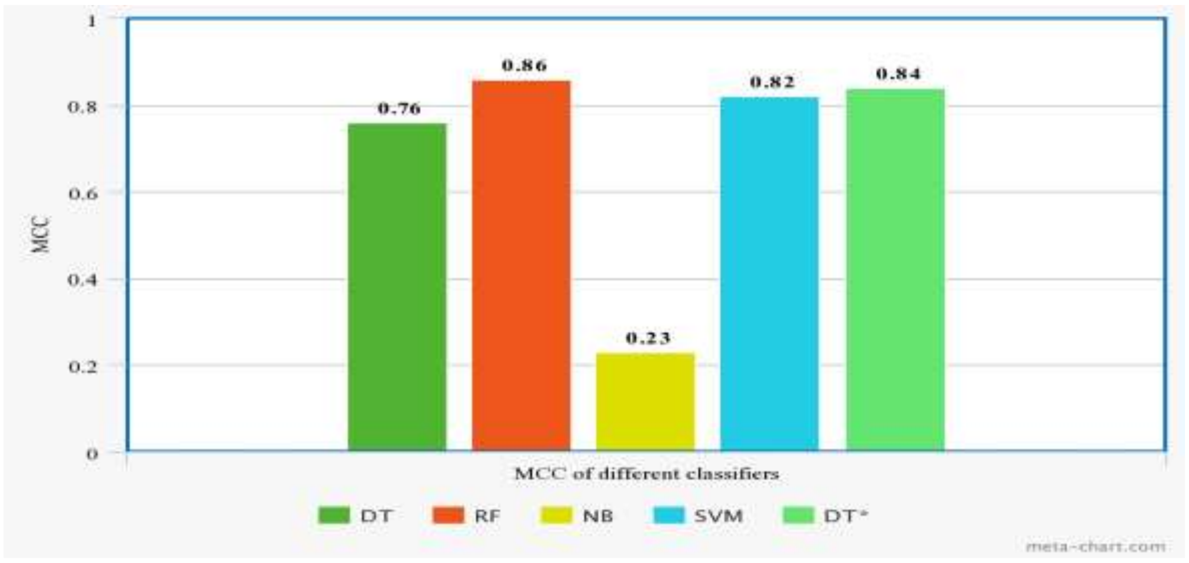


Figure 5.11: Comparison of models

VI. CONCLUSION AND FUTURE SCOPE

In the last chapter, we have seen results of different models with dataset. The evaluation metrics we considered are accuracy, precision, Recall, F1 score, Mathews Correlation coefficient (MCC).

Firstly, let’s compare the results. Here, we cannot compare accuracy of the models because of the highly imbalanced dataset. The 99.9 of data set is non-fraud transactions, so accuracy is biased with at least 99 percent accuracy. Let's compare the models with MCC as the reference. Naive Bayes has the least MCC of all the models i.e. 23, because of the assumptions made. The second least model w.r.t to MCC is Decision tree with a MCC value of 76. We can clearly see that Naïve Bayes is not at all suited for this problem. The model with highest MCC is Random Forest Classifier i.e. 86, and second-best model is SVM with different kernels, with MCC of 82. We tried to deal with imbalance in dataset with under sampling technique, we applied this on Decision tree, the MCC score improved from 76 to 84. Now, with under sampling Decision tree has second best MCC higher than SVM. We conclude that, to approach this credit card fraudulent detection, we can go with Random forest as the first choice and SVM as the second choice, the reason we excluded Decision tree with under sampling because it not always feasible to under sample the data in rea time. So, we go with the above choices to bring this model into real time data application.

Above, we decided which is model better for further development of the project to detect fraudulent transactions in real time.

There are few challenges in bringing this project to real time working, they are:

- There will be New fraudulent techniques always, a static system or model cannot always predict the target class perfectly
- To connect to a server for real-time transactions and continuously computing the model with continuous data takes a lot of computational power and time, if an efficient model is not used.

Since we are working on python to approach this project, connecting to a server in real-time is easy because python has many Web Frame works. Example: Django, Flask etc. Incorporating new fraudulent detection techniques as the new fraudulent ways appear is the best way to tackle the first issue.

For a Real time system, the proposed system looks the fig below, using predictive analytics and an API module the end user is notified the second fraud transaction place.

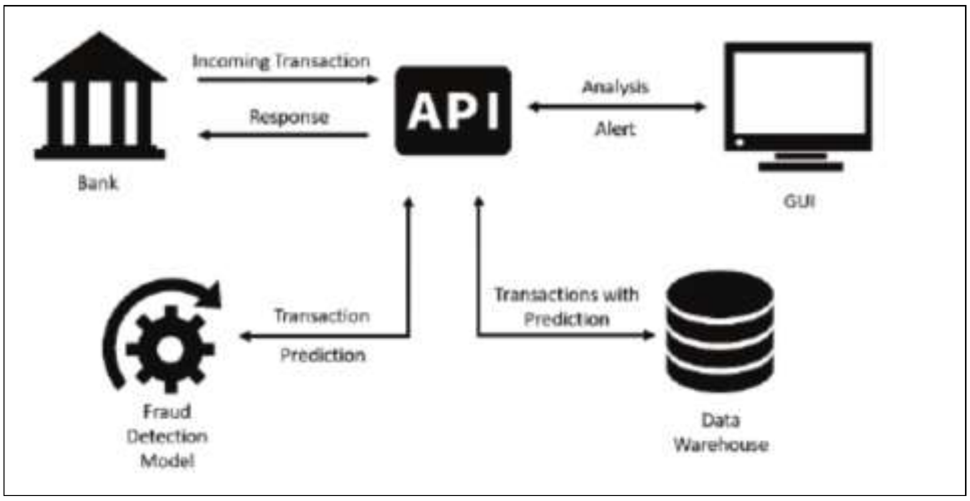


Figure 6.1: Proposed System in Future

VII. REFERENCES

[1]
https://www.sciencedirect.com/science/article/pii/S187705092030065X/pdf?md5=5d8441 a61d34c23f8f24813729334508&pid=1-s2.0-S187705092030065X-main.pdf

[2] Anwer et al. (2009-2010). Online Credit Card Fraud Prevention System for Developing Countries', International Journal of Reviews in Computing, ISSN: 2076-3328, Vol. 2.

[3] E. D. Y. Sahin, "Detecting credit card fraud by decision trees," in Proceedings of the international multiconference of engineers and computer science, Hong Kong, 2011

[4] Credit Card Fraud Detection Based on Transaction Behavior-by John Richard D. Kho, Larry A. Vea" published by Proc. of the 2017 IEEE Region 10 Conference (TENCON), Malaysia, November 5-8, 20107

[5] Awoyemi, John O., et al. "Credit Card Fraud Detection Using Machine Learning Techniques: A Comparative Analysis." 2017 International Conference on Computing Networking and Informatics (ICCNI), 2017, doi:10.1109/iccni.2017.8123782.

[6] https://www.udemy.com/course/machinelearning

[7] https://www.javatpoint.com/machine-learning

[9] https://www.investopedia.com/terms/f/fraud.asp

[10]https://www.researchgate.net/publication/224560136_Research_on_Credit_Card_Fraud_Detection_Model_Based_on_Distance_Sum

[11] https://ieeexplore.ieee.org/document/8553963

[12] https://www.researchgate.net/

[13] https://www.kaggle.com/mlg-ulb/creditcardfraud

[14] http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

[15] Decision Trees and Random Forests: A visual Introduction for beginners by Chris Smith

[16] Decision Trees and Random Forests: A visual Introduction for beginners by Chris Smith

[17] Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, by cholkopf and Smola.