

---

# **Credit Card Fraud Detection Using Data Science Technique**

Submitted in partial fulfillment of the requirements for the award of Bachelor of  
Engineering Degree in ELECTRONICS AND COMMUNICATION ENGINEERING

by

**Tamilmani.P (38130193)**

**And**

**Mohammed Naufel.M(38130137)**



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION  
ENGINEERING**

**SCHOOL OF ELECTRICAL AND ELECTRONICS**

**SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119**

**March – 2022**



# **SATHYABAMA**

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC | 12B Status by UGC | Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)

## **DEPARTMENT OF ELECTRICAL AND ELECTRONICS**

### **BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of  
Tamilmani.p(38130193) and Mohammed Naufel.M(38130137) who have done the  
Project work as a team who carried out the project entitled “Credit Fraud Detection  
using Data Science Technique” . Under my supervision from December 2022 to January  
2022.

**Internal Guide**

(MARY SAJIN SANJU, M.E)

**Head of the Department**

(Dr. T. RAVI, M.E., Ph.D)

---

**Submitted for Viva voce Examination held on \_\_\_\_\_**

**Internal Examiner**

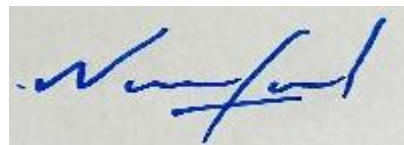
**External Examiner**

---

## DECLARATION

I **TAMILMANI.P(38130219)** and **Mohammed Naufel.M(38130137)** here by declare that the Project Report entitled “**Credit Card Fraud Detection using Data Science Technique**” done by me under the guidance of **MARY SAJIN SANJU** at Sathyabama Institute of Science and Technology, Chennai- 600119 is submitted in partial fulfillment of the requirements for the award of Bachelor Engineering degree in Electronics and Communication Engineering.

Tamilmani-P



DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

---

## **ACKNOWLEDGEMENT**

**I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.**

**I convey my thanks to Dr. N. M. Nandhita, Dean, School of Electrical and Electronics Engineering and Dr. T. RAVI, Head of the Department, Dept. of Electrical and Electronics Engineering for providing me necessary support and details at the right time during the progressive reviews.**

**I would like to express my sincere and deep sense of gratitude to my Project Guide Mary Sajin Sanju for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.**

**I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of ELECTRICAL AND ELECTRONICS who were helpful in many ways for the completion of the project.**

---

# ABSTRACT

- The number one business goal for many banks is to retain high profitable customers. However, Banking fraud gives a threat to achieve this. In case the case of trust, financial losses and credibility this is a serious issue for both customers and banks.
- Using Credit Card fraud detection using machine learning technique is not only a trend in banking industry but also one the best way to prevent and monitor fraud cases. This machine learning is really helpful to the institutions to reduce costly chargebacks, time consuming manual review and denial legal transactions.
- In this project we will be using Machine learning Models to identify fraudulent credit card transactions. The customer level data will be analyzed by us, which is helpful to identify the fraud cases.
- We took the data set from Kaggle. This data set contains credit card transactions that had happened over a couple of days in September 2013 by European card holders. The total numbers transactions are two lakh eighty four thousand eight hundred and seven (2,84,807). Out of these 492 transactions are fraudulent. By looking at this data set we can say that this is a highly imbalanced data set because the fraud transactions percentage is just 0.172%. In order to maintain confidentiality we have modified the data set with principal component analysis (PCA). Other than “amount” and “time” the other features from V1 to V28 are obtained principal components using PCA. The “time” feature contains the time between first and subsequent transaction. The “class” feature represents labelling of class, 1 indicates fraud and 0 non-fraud transactions.

---

## TABLE OF CONTENTS

| TITLE                                     | PAGE NO     |
|-------------------------------------------|-------------|
| <b>ABSTRACT</b>                           | <b>V</b>    |
| <b>LIST OF ABBREVIATIONS</b>              | <b>Xiii</b> |
| <br>                                      |             |
| <b>1. INTRODUCTION</b>                    | <b>1</b>    |
| 1.1 Methodology                           | 2           |
| 1.2 Exploratory Data analysis             | 3           |
| 1.3 Confusion matrix                      | 3           |
| 1.3.1 False Negative                      | 4           |
| 1.3.2 False positive                      | 4           |
| 1.3.1 True Negative                       | 4           |
| 1.4 Classification Report                 | 5           |
| 1.4.1 precision                           | 5           |
| 1.4.2 Recall                              | 5           |
| 1.4.3 F1-score                            | 5           |
| <br>                                      |             |
| 1.5 AUC-ROC                               | 6           |
| <b>2. LITERATURE REVIEW</b>               | <b>7</b>    |
| 2.1 Rimpal R. Popat with Jayesh Chaudhary | 7           |

---

|           |                                             |           |
|-----------|---------------------------------------------|-----------|
| 2.2       | Mohamad Zamini                              | 7         |
| 2.3       | Shiyang Xuan                                | 8         |
| 2.4       | Dejan Varmedja                              | 8         |
| 2.5       | Changjun Jiang                              | 9         |
| 2.6       | Sahil Dhankhad                              | 9         |
| 2.7       | Kuldeep Randhawa                            | 9         |
| 2.8       | Sai Kiran                                   | 10        |
| <b>3.</b> | <b>Model Building</b>                       | <b>11</b> |
| 3.1       | Logistic Regression                         | 11        |
| 3.2       | K – Nearest Neighbor                        | 11        |
| 3.3       | Decision Tree                               | 11        |
| 3.4       | Random Forest Classifier                    | 12        |
| 3.5       | XG Boost                                    | 12        |
| 3.6       | Support Vector Machine                      | 12        |
| <b>4.</b> | <b>Cross Validation in machine learning</b> | <b>13</b> |
| 4.1       | Repeated K fold                             | 13        |
| 4.2       | stratified K fold                           | 13        |
| 4.3       | Model Building with balancing Classes       | 13        |
| 4.3.1     | Random over sampler                         | 13        |
| 4.3.2     | SMOTE                                       | 13        |
| 4.3.3     | ADASYN                                      | 14        |
| 4.3.4     | Hyper Parameter Tuning                      | 14        |

---

|                                                   |           |
|---------------------------------------------------|-----------|
| <b>5. IMPLEMENTATION</b>                          | <b>14</b> |
| 5.1 Perform cross validation with repeated k fold | 18        |
| 5.1.1 Logistic Regression with L1                 | 18        |
| And L2 Regularisation                             |           |
| 5.1.1.1 Logistic Regression with                  | 18        |
| L2 Regularisation                                 |           |
| 5.1.1.2 Logistic Regression with                  | 19        |
| L1 Regularisation                                 |           |
| 5.1.2 KNN Model                                   | 21        |
| 5.1.2 Decision Tree Models with                   | 23        |
| 'gini' & 'entropy' criteria                       |           |
| 5.1.3.1 Gini                                      | 23        |
| 5.1.3.2 Entropy                                   | 25        |
| 5.1.4 Random Forest Model                         | 27        |
| 5.1.5 XG Boost Model                              | 28        |
| 5.1.6 SVM Model with Sigmoid Kernel               | 30        |
| 5.1.7 <i>Results for cross validation</i>         | 32        |
| <i>with Repeated K Fold</i>                       |           |
| 5.2 Perform cross validation with repeated k fold | 33        |
| 5.2.1 Logistic Regression with L1                 | 33        |
| And L2 Regularisation                             |           |
| 5.2.1.1 Logistic Regression with                  | 33        |

---

|                                           |    |
|-------------------------------------------|----|
| L2 Regularisation                         |    |
| 5.2.1.2 Logistic Regression with          | 34 |
| L1 Regularisation                         |    |
| 5.2.2 KNN Model                           | 37 |
| 5.2.2 Decision Tree Models with           | 38 |
| 'gini' & 'entropy' criteria               |    |
| 5.2.3.1 Gini                              | 38 |
| 5.2.3.2 Entropy                           | 40 |
| 5.2.4 Random Forest Model                 | 41 |
| 5.2.5 XG Boost Model                      | 43 |
| 5.2.6 SVM Model with Sigmoid Kernel       | 45 |
| 5.2.7 <i>Results for cross validation</i> | 47 |
| <i>with Repeated K Fold</i>               |    |
| <b>6. MODEL BUILDING WITH</b>             |    |
| <b>BALANCING CLASSES</b>                  |    |
| 6.1 Over Sampling with random over        | 48 |
| Sampler with stratified K fold            |    |
| 6.1.1 Logistic Regression with L1         | 48 |
| And L2 Regularisation                     |    |
| 6.1.1.1 Logistic Regression with          | 48 |
| L2 Regularisation                         |    |
| 6.1.1.2 Logistic Regression with          | 50 |
| L1 Regularisation                         |    |

---

|                                                                    |    |
|--------------------------------------------------------------------|----|
| 6.1.2 KNN Model                                                    | 52 |
| 6.1.2 Decision Tree Models with<br>'gini' & 'entropy' criteria     | 54 |
| 6.1.3.1 Gini                                                       | 54 |
| 6.1.3.2 Entropy                                                    | 56 |
| 6.1.4 Random Forest Model                                          | 58 |
| 6.1.5 XG Boost Model                                               | 60 |
| 6.1.6 <i>Results for cross validation<br/>with Repeated K Fold</i> | 62 |
| 6.2 Over Sampling with SMOTE                                       | 63 |
| Over Sampling                                                      |    |
| 6.2.1 Logistic Regression with L1<br>And L2 Regularisation         | 63 |
| 6.2.1.1 Logistic Regression with<br>L2 Regularisation              | 63 |
| 6.2.1.2 Logistic Regression with<br>L1 Regularisation              | 64 |
| 6.2.2 KNN Model                                                    | 67 |
| 6.2.2 Decision Tree Models with<br>'gini' & 'entropy' criteria     | 68 |
| 6.2.3.1 Gini                                                       | 68 |
| 6.2.3.2 Entropy                                                    | 70 |
| 6.2.4 Random Forest Model                                          | 72 |

---

|                                             |    |
|---------------------------------------------|----|
| 6.2.5 XG Boost Model                        | 74 |
| 6.2.6 <i>Results for cross validation</i>   | 76 |
| <i>with Repeated K Fold</i>                 |    |
| 6.3 Over sampling with ADASYN over sampling | 77 |
| 6.3.1 Logistic Regression with L1           | 77 |
| And L2 Regularisation                       |    |
| 6.3.1.1 Logistic Regression with            | 77 |
| L2 Regularisation                           |    |
| 6.3.1.2 Logistic Regression with            | 78 |
| L1 Regularisation                           |    |
| 6.3.2 KNN Model                             | 81 |
| 6.3.2 Decision Tree Models with             | 82 |
| 'gini' & 'entropy' criteria                 |    |
| 6.2.3.1 Gini                                | 82 |
| 6.2.3.2 Entropy                             | 84 |
| 6.2.4 Random Forest Model                   | 86 |
| 6.2.5 XG Boost Model                        | 88 |
| 6.2.6 <i>Results for cross validation</i>   | 90 |
| <i>with Repeated K Fold</i>                 |    |
| 6.2.7 Overall conclusion after running      | 91 |
| the models on Oversampled data              |    |

---

|                                     |           |
|-------------------------------------|-----------|
| <b>7. HPT – XG BOOST REGRESSION</b> | <b>91</b> |
| <b>8. CONCLUSION</b>                | <b>91</b> |
| <b>9. REFERENCES</b>                | <b>92</b> |

---

### LIST OF ABBREVIATION:

| S.NO | ABBREVIATION | FULL FORM                                          |
|------|--------------|----------------------------------------------------|
| 1    | ROC          | Receiver<br>Operating<br>Characteristic<br>Curve   |
| 2    | AUC          | Area under The<br>ROC curve                        |
| 3    | SMOTE        | Synthetic<br>Minority<br>Oversampling<br>Technique |
| 4    | ADASYN       | Adaptive<br>Synthetic                              |
| 5    | SVM          | Support Vector<br>Machine                          |
| 6    | PCA          | Principal<br>Component<br>Analysis                 |
| 7    | XG Boost     | Extreme<br>Gradient                                |

---

## **1.INTRODUCTION**

Fraud detection involves in monitoring the activities of the populations of users in order to avoid objectionable behavior which consists of fraud and defaulting. This has been a very relevant problem that demands the attention of communities such as machine learning and data science where the solution of this problem can be automated.

As this kind of problem is particularly challenging in a learning perspective, as it is characterized by various factors such as class imbalance. The number of valid transactions far outnumbered fraudulent ones and also the transaction pattern changes in statistical properties over the course of time.

Thus, in real world examples, the massive streams of payment requests are been quickly scanned by the automatic tools that determine which transaction to be authorized in order to prevent the performance of the fraud detection overtime As for many banks which has retained high profitable customers has been the number one business goal, these banking frauds poses a significant threat at different banks but in terms of substantial financial losses, trust and credibility has been a concerning issue for both banks and customers.

In banking industry, credit card fraud detection using machine learning is not only a trend but it has been a necessity to put the proactive monitoring and fraud prevention mechanisms in place.

---

As machine learning has helped these institutions for reducing time consuming manual reviews and as well as the denial of legitimate transactions

So in this project, we have detected the fraudulent credit card transactions with the help of machine learning models and analyze the customer data which has been collected.

Machine learning is a subfield of Artificial Intelligence (AI) which is generally used to understand the structure of data and fit the data into models which can be understood and used by people.

Since, it is a field of learning machines and it differs from traditional computing approach and the algorithms which are instructed by the computer to calculate on solving problem.

The algorithm instead allows the computer to train on data inputs and use statistical analysis for output values which falls in a specific range.

So the machine learning makes easy for computers in building models from sample data in order to operate the decision making process based on the data inputs.

## **1.1 Methodology:**

Most of this project has been done in google collab of language python because it is convenient for executing the program code and it does not take more time to execute with machine learning algorithms in order to detect with different activities.

So, first of all, we obtained the dataset from Kaggle which is a data analysis website used for providing datasets. Inside the dataset, there are 31 columns of which there are 28 features represented as v1 to v28 in order to protect the sensitive data and the other column represents Time, Amount and Class.

---

The time shows the time gap between the first transaction, amount represents the total of money transacted and class represents the target value where 0 represents legitimate transaction and 1 represents fraudulent transactions.

Since python libraries are free and open source, it has been built using Numpy and pandas for analyzing the data, matplotlib and seaborn to visualize the data and also provides simple and efficient tools for machine learning which has various classification and regression algorithms and it is designed to inter-operate the numerical libraries.

## **1.2 Exploratory Data Analysis (EDA):**

It is the process of investigating the dataset for discovering the patterns and forms hypotheses based on the understanding of the data set . It involves in generating the summary statistics for numerical data in the dataset and creates various graphical representations to understand the data.

## **1.3 Confusion matrix:**

It is a table which is used to describe the performance of a classification model or a classifier on a set of test data for which the true values are known.

By visualizing the confusion matrix, an individual could determine the accuracy of the model by observing the diagonal values for measuring the number of accurate classifications.

The structure of the matrix is considered as the size of the matrix which is directly proportional to the number of output classes.

---

This confusion matrix is said to be in the form of a square matrix where the column represents the True class and the row represents the predicted class of a model

|                 |          | True Class |          |
|-----------------|----------|------------|----------|
|                 |          | Positive   | Negative |
| Predicted Class | Positive | TP         | FP       |
|                 | Negative | FN         | TN       |

**Fig 1.3: Confusion Matrix**

And it also consists of four types such as:

Where observation is positive and the classification model is positive

**1.3.1 False negative:**

Where observation is positive and model classification is negative

**1.3.2 False positive:**

Where observation is negative and model classification is positive

**1.3.3 True negative:**

Where observation is negative and model classification is negative

---

## 1.4 Classification Report:

It is a performance evaluation metric in machine learning and it is used to show the precision, recall, F1-score and supports the trained classification model.

It helps to provide a better understanding of the overall performance of the trained model. To understand the classification report of the machine learning model, we need to know the metrics as follows:

### 1.4.1 Precision:

- It is defined as the ratio of true positive to the sum of true and false positive.
- It is given as,  $\text{True positives} / (\text{True positives} + \text{False positives})$
- It tells how many of the correctly predicted cases actually turned out to be positive.

### 1.4.2 Recall:

- It is defined as the ratio of the true positives to the sum of true positives and false negative.
- It is given as,  $\text{True positives} / (\text{True positives} + \text{False Negative})$ .
- It tells us how many of the actual positive cases are able to predict correctly with the model classification.

### 1.4.3 F1-score:

- It is the harmonic mean of precision and recall.
- It gives a combined idea of the two metrics and it will be maximum when the precision is equal to recall. i.e.,  $\text{precision} = \text{recall}$ .

---

## 1.5 AUC – ROC:

It is a curve which performs the measurement for the classification problem at various threshold settings where the term AUC stands for “Area Under the Curve” and ROC stands for “Receiver Operator Characteristics”.

The Receiver Operator Characteristics (ROC) curve is an evaluation metric for binary classification problems and also it is a probability curve that plots the TPR against FPR at various threshold values where it essentially separates the signal from noise.

The Area Under the Curve (AUC) is the measure of the ability of a classifier in order to distinguish between the classes and it is used as a summary for the ROC curve.

It tells how the model is capable to distinguish between the classes. As AUC is higher, the performance of the model will be better due to which it is distinguished between the positive and negative classes at predicting 0 classes as 0 and 1 classes as 1.

And the performance of the model has been speculated by the best model which has AUC which is near to 1 which means that it has a good measure of separation and the poor model has an AUC which is near to 0 which means that it has the worst measure of separation.

It also reciprocates the result when it is predicted in 0 as 1 and 1 as 0 and when the AUC is 0.5, then the model has no class separation capacity.

---

## 2. LITERATURE REVIEW

### 2.1 Rimpal R. Popat with 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 had focused on the techniques which are Regression, classification, Logistic regression, Support vector machine, Neural network, Artificial Immune system, K-nearest Neighbor, Naïve Bayes, Genetic Algorithm, Data mining, Decision Tree, Fuzzy logic-based system, etc. In which, they have explained six data mining approaches as theoretical background that are classification, clustering, prediction, outlier detection, Regression, and visualization. Then have explained about existing techniques based on statistical and computation which is Artificial Immune system (AIS), Bayesian Belief Network, Neural Network, Logistic Regression, Support Vector Machine, Tree, Self organizing map, Hybrid Methods, As a result, they had concluded that all the present machine learning techniques mentioned above can provide high accuracy for the detection rate and industries are looking forward to finding new methods to increase their profit and reduce the cost. Machine learning can be a good choice for it.

### 2.2 Mohamad Zamini:

Mohamad Zamini purposed an unsupervised fraud detection method using autoencoder based clustering. The autoencoder is an auto associator neural network they have used it to lower the dimensionality, extract the useful features, and increase the efficiency of learning in a neural network. They had used European dataset with 284807 transactions in which 0.17% is the fraud and trained there autoencoder based clustering with the following parameters Number of iterations = 300 Number of clusters = 2 Clustering initialization = k-means++ Divergence tolerance = 0.001 Learning rate of the model = 0.1 Number of epochs = 200 Activation function = elu, Relu. As a result, they got their training loss as 0.024 and validation loss as 0.027 and the mean of not fraud data 75% less than the mean of reconstructive error that is 25% the design of there is model is context-free. In

---

concern about the model predictions, the True positive are 56,257, False negative is 607, False positive are 18, True negatives is 80 and the best preferred are (56,257 + 80 = 56,337). The right predictions made are 56,337 out of 284807. [Credit Card Fraud Detection using autoencoder based clustering].

### **2.3 Shiyang Xuan:**

They made a comparison based on two random forests. Random-tree-based random forest CART-based random forest. They use different random forest algorithms to train the behavior features of normal and abnormal transactions and both of the algorithms are different in their base classifications and their 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. [Random Forest for Credit Card Fraud Detection].

### **2.4 Dejan Varmedja:**

He proposed the various machine learning algorithms and analyzed them concerning to credit card fraud detection methods. The various methods of machine learning are Logistic Regression, Naïve Bayes, Random Forest, Multilayer Perceptron. Here for multilayer perceptron (ANN) is used (Artificial neural network) which consist of 4 hidden layers and relu activation functioned is used that is to avoid negative values and optimizer used is Adam for its best performance As a result for Logistic regression the accuracy score is 97.46% with the data set containing 56962 samples in which 98 fraud transactions. For the same dataset Naïve Bayes and Random Forest, accuracy score is 99.23% and 99.96% respectively. At last for ANN it was 99.93% of accuracy as we can observe that random forest gives the best result in case of credit card fraud detection. [ Credit Card Fraud Detection - Machine Learning methods]

---

## **2.5 Changjun Jiang:**

He proposed a novel fraud detection method that has four stages they first utilize the historical transaction data to divide them into groups to form clusters of transactions having the same behavior then thus they came up with a sliding window strategy to aggregate transactions .This algorithm is used to characterize the behavioral pattern of a cardholder then after aggregation, we use the new window formed the feature extraction is done. At last, the classification takes place and classifies behavioral patterns and assignments. As a result, their method of Logistic Regression with raw data Random Forest with aggregation data and Random Forest and feedback technique with aggregation data are the best method with 80% accuracy as compared to other methods.[ Credit Card Fraud Detection: A Novel Approach Using Aggregation Strategy and Feedback Mechanism]

## **2.6 Sahil Dhankhad:**

They applied supervised machine learning algorithms on the real-world data set and then used those algorithms to implement a super classifier using ensemble learning and then they compared the performance of supervised algorithms with their implementation of a super classifier. They used ten machine learning algorithms such as Random Forest, Stacking Classifier, XGB Classifier, Gradient Boosting, Logistic Regression, MLP Classifier, SVM, Decision Tree, KNN, Naïve Bayes. And compared the accuracy, Recall Precision, confusion matrix with the result of their super classifier. As a result, they found that the Logistic Regression is better for predicting fraud transactions. [Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study].

## **2.7 Kuldeep Randhawa:**

They used twelve machine learning algorithms for credit card fraud detection in which their range standard from a neural network to deep learning. They are tracing the performance of benchmark and real world datasets. In addition, the AdaBoost and majority voting methods are applied for forming the hybrid models. As there related study explains about single and hybrid models. For both the parameters (Benchmark and real-world datasets) they had given the results using there twelve selected algorithms that are Naïve Bayes, Random Forest, Decision Tree, Gradient Boosted Tree, Decision Stump, Random Tree, Neural Network,

---

Linear Regression, Deep Learning, Logistic Regression, SVM, Multilayer Perceptron. As a result, when standard algorithms used with AdaBoost and majority voting methods under benchmark data the best accuracy and sensitivity acquired by Random Forest algorithm 95% and 91% respectively. When experimented with real-world data the accuracy rate is still above 90% even with 30% noise in the dataset. MCC (Mathews correlation coefficient) is standard to measure the performance of a model so in case of majority voting the best MCC score is 0.823 whereas 0.942 with 30% of noise added to the dataset. [Credit Card Fraud Detection Using AdaBoost and Majority Voting]

## **2.8 Sai Kiran:**

He proposed an improved algorithm for credit card fraud detection. That is named as Naïve Bayes improved K-nearest Neighbor method (NBKNN). They have used a dataset on which they had applied the algorithms to identify the fraudulent transaction in the taken dataset. Literature Review of Different Machine Learning Algorithms for Credit Card Fraud Detection 104 Published By: Blue Eyes Intelligence Engineering and Sciences The dataset has the record of European Cardholders who made a transaction using their credit cards within 2 days they made 284,807 transitions in which 492 transaction is fraudulent. The techniques used which were used are classification techniques but work differently on the same dataset. They had used both of the techniques (Naïve Bayes and k-nearest neighbor) to enhance the accuracy and flexibility of the algorithm. As a result, they got the accuracy of approximately 95% from Naïve Bayes and 90% from K -nearest Neighbor techniques.

---

### 3 MODEL BUILDING

The machine learning models need to be built because to unbalance the dataset according to its algorithm as follows:

- I. Logistic Regression
- II. K-Nearest Neighbor (KNN)
- III. Decision Tree classifier
- IV. Random Forest classifier
- V. XG Boost classifier
- VI. Support Vector Machines (SVM)

#### 3.1 Logistic Regression:

It is one of the most popular machine learning algorithms which comes under supervised learning technique and also it is used for predicting the dependent variable using a given set of independent variables.

#### 3.2 K-Nearest Neighbor (KNN):

It is a supervised machine learning algorithm where the algorithm is used to solve both classification and regression problem statements.

Since the number of nearest neighbors to a new unknown variable which has predicted or classified denoted as the symbol 'K'.

#### 3.3 Decision tree classifier:

It is a tree – structured classifier in which the internal nodes represents the features of the dataset, the branches represents the decision rules and the leaf node represents the outcome.

It is a supervised learning technique that is used for both classification and regression problems but mostly it is mostly preferred for solving classification problems

---

### **3.4 Random Forest Classifier:**

It is a machine learning technique which is used to solve regression and classification problems and it also utilizes the ensemble learning which is a technique that combines many classifiers to provide solutions to the complex problems

### **3.5 XG Boost Classifier:**

It is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements the machine learning algorithms under the gradient boosting frameworks and it provides a parallel tree boosting to solve many data science problems in an accurate way.

### **3.6 Support Vector Machine (SVM):**

It is a supervised learning machine algorithm which is used for both classification or regression problems and mostly it is used as classification problems. In SVM algorithm, it plots the data item as a point in  $n$  – dimensional space with the value of each features being the value of a particular coordinate. It also performs classification by finding the hyper-plane that differentiates the two classes.

Hence, these algorithms are the part of SK learn package libraries in which it includes ensemble based methods for classification and regression.

---

## 4 Cross validation in machine learning

It is a technique used for validating the model efficiency by training it on the subset of input data and testing on previous subset of the input data.

In this we have used two types of methods for cross validation which are as follows:

### 4.1 Repeated K Fold:

It is the most preferred cross validation technique for both classification and machine learning models.

Shuffling and random sampling of the dataset in multiple times is the core procedure of repeated K-Fold algorithm and it results in making a strong model as it covers the maximum training and testing operations

### 4.2 Stratified K Fold:

It is a variation of K Fold that returns the stratified folds.

These folds are made by preserving the percentage of samples for each class and also it splits the data into K consecutive terms

### 4.3 Model Building with balancing classes:

In order to balance the classes, the following methods have been taken as follows:

#### 4.3.1 Random Over sampler:

It involves in selecting random examples from the minority classes with replacement and add them to the trained dataset.

#### 4.3.2 SMOTE:

It stands for Synthetic Minority Oversampling Technique and it is a statistical technique in order to increase the number of cases in the dataset in a balanced way. This module works by generating the synthetic data based on the feature similarities between existing minority instances.

---

In order to create a synthetic instance, it will find the K - Nearest neighbors of each minority instances.

In order to create a synthetic instance, it finds the K - Nearest Neighbor of each minority instance, randomly selects one of it and then it calculates the linear interpolation to produce a new minority instance.

#### **4.3.3 ADASYN:**

It stands for Adaptive Synthetic, it is an algorithm that generates the synthetic data and one of it's greatest advantage is not to copy the same minority data and generating the model.

#### **4.3.4 Hyperparameter tuning:**

It is a collection of individual records containing the important features in the machine learning problem

It is used during trained data for configuring the model and it accurately make predictions of the similar data.

### **5 Implementation:**

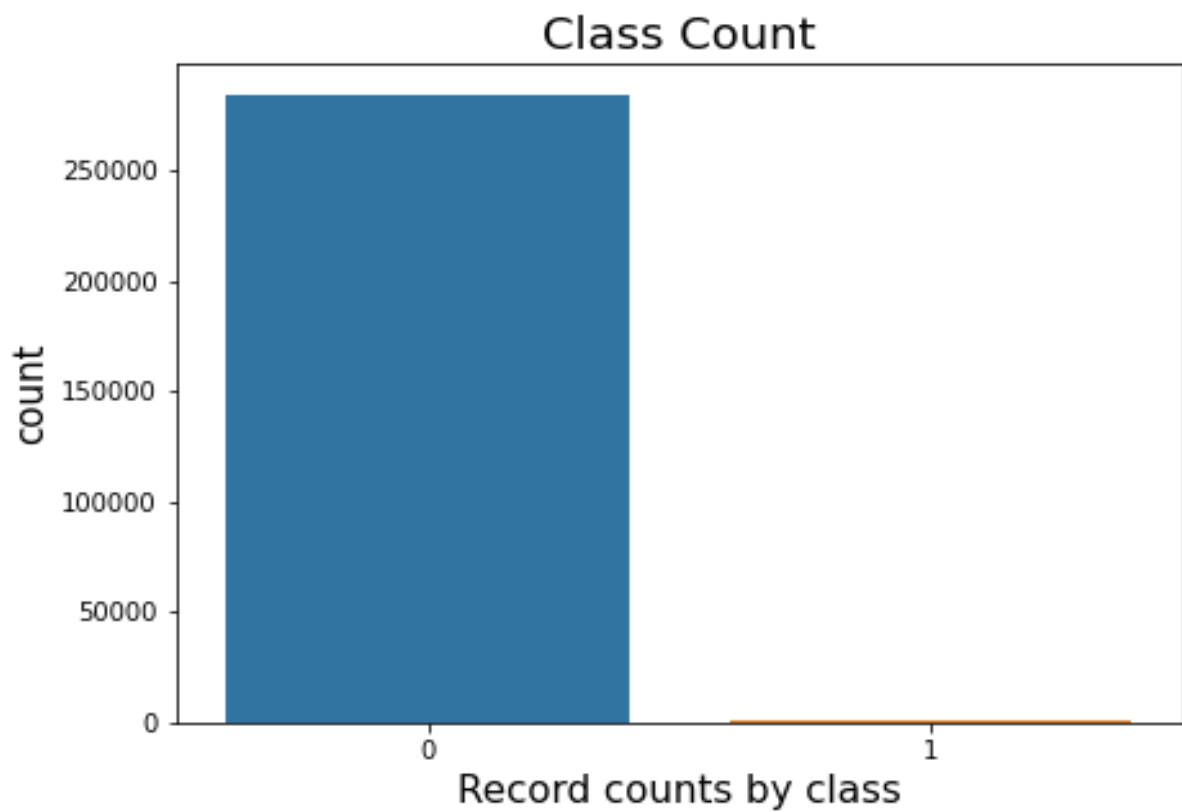
This dataset contains transactions made by European card holders over a period of two days in September 2013 out of which the total transaction is 2,84,807 from which 492 were fraudulent.

As this data is highly unbalanced, the positive class or known to be frauds accounted for 0.172% of the total transactions.

This dataset has been modified with Principal Component Analysis (PCA) in order to maintain confidentiality.

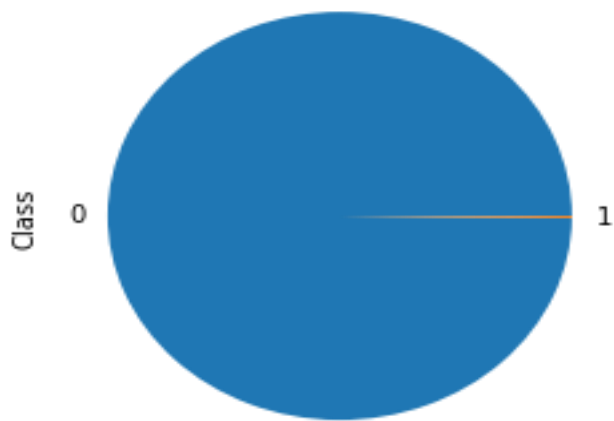
Apart from time and amount, all other features from v1 to v28 are the principal component obtained using PCA.

Since it has done in google collab, it needs to get mounted with the google drive in order to generate the code by importing the csv file to contain the dataset. We have also plotted different graphs to check the transactions in the data.



**Fig 5.1: Class count**

Here this graph shows the number and percentage of fraudulent and non- fraudulent transactions in a bar graph in which the fraudulent transactions are much lower than the non-fraudulent transactions.



**Fig 5.2: Pie Chart of Class Count**

and same as before, this is the graph shows the number and percentage of fraudulent and non-fraudulent in a pie-chart graph.

After the analysis of the data, we have to check whether the data is correlated or not, for that we need to generate a heat map in order to find whether it is strongly correlated or it is weakly correlated before splitting the data into train data and test data.

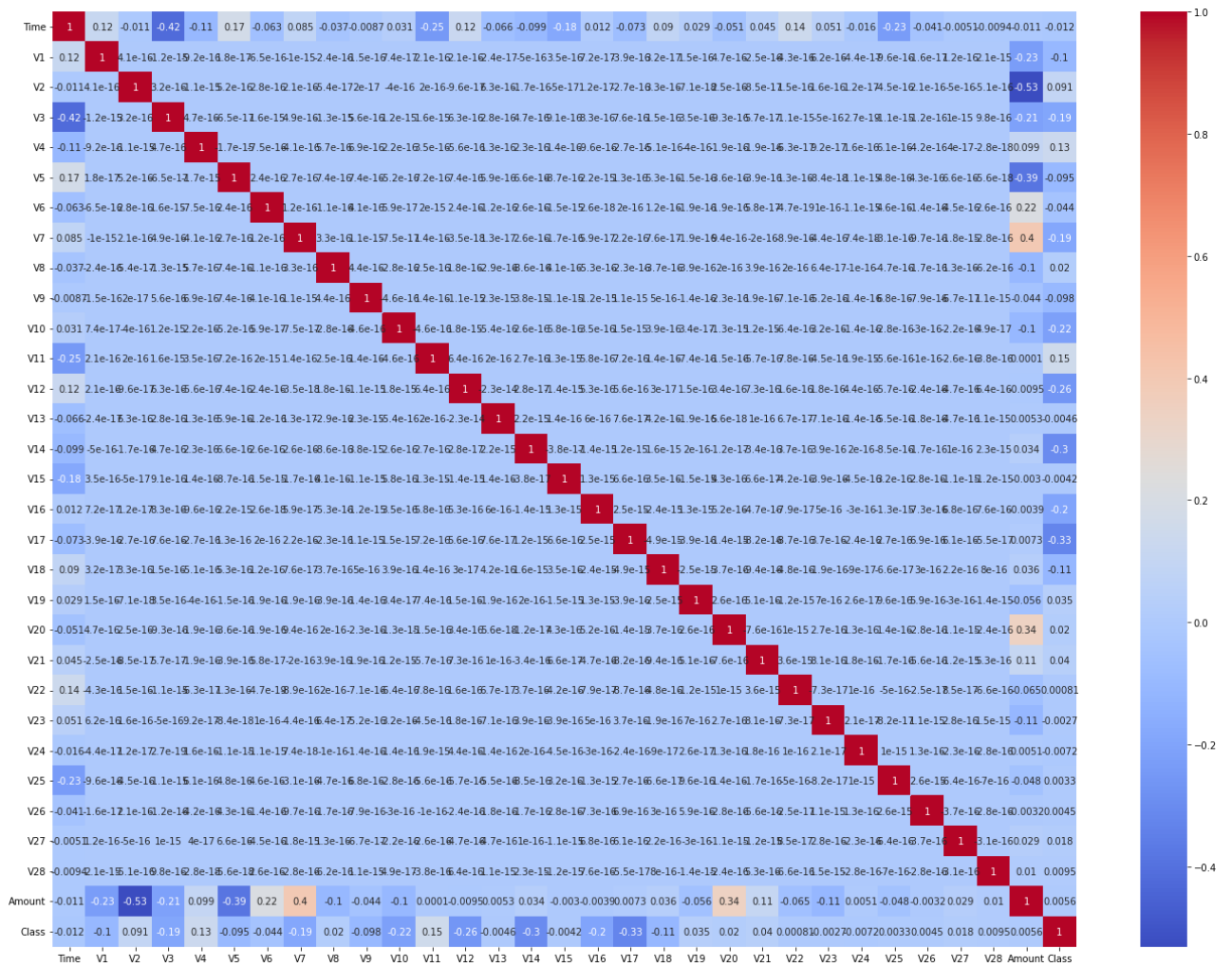


Fig 5.3: correlation in heatmap

So in this graph, the heat map is produced in order to check the correlation where the darkly colored represents that it is strongly correlated and the lightly colored represents it is weakly correlated.

## 5.1 Perform cross validation with repeated k fold:

### 5.1.1 Logistic Regression with L1 And L2 Regularisation:

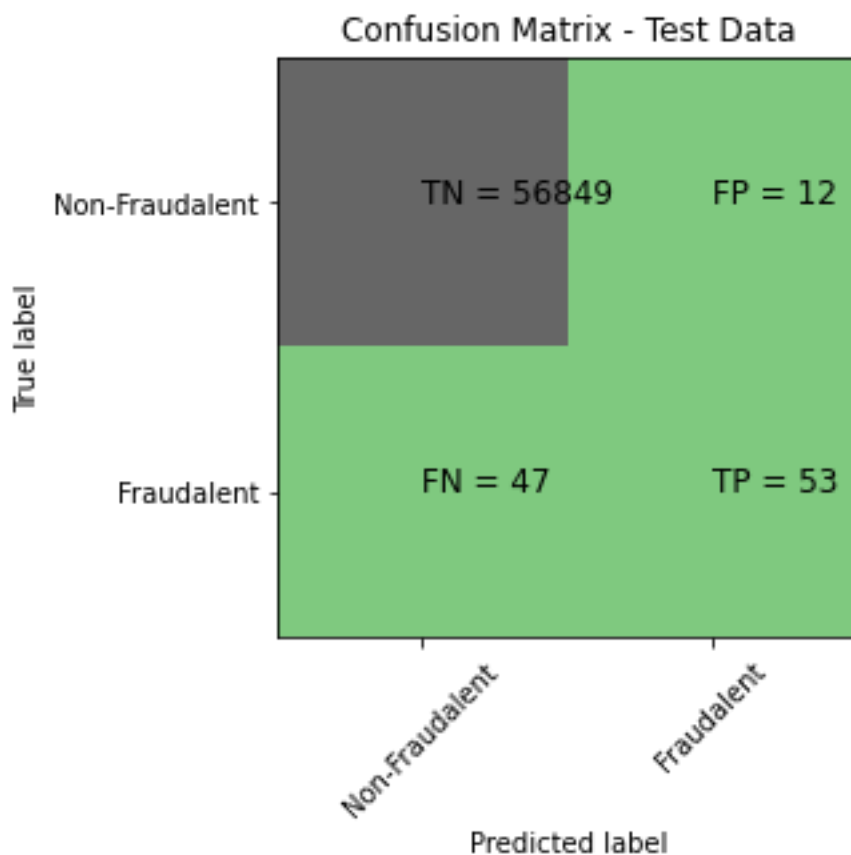
Max auc roc for l1: 0.9757741295537155

Max auc roc for l2: 0.9865741449266722

#### 5.1.1.1 Logistic Regression with L2 Regularisation:

Accuracy of Logistic model with l2 regularisation: 0.9989642035778866

Confusion Matrix



**Fig 5.4: Confusion Matrix of Logistic Regression with L2 Regularisation classification Report:**

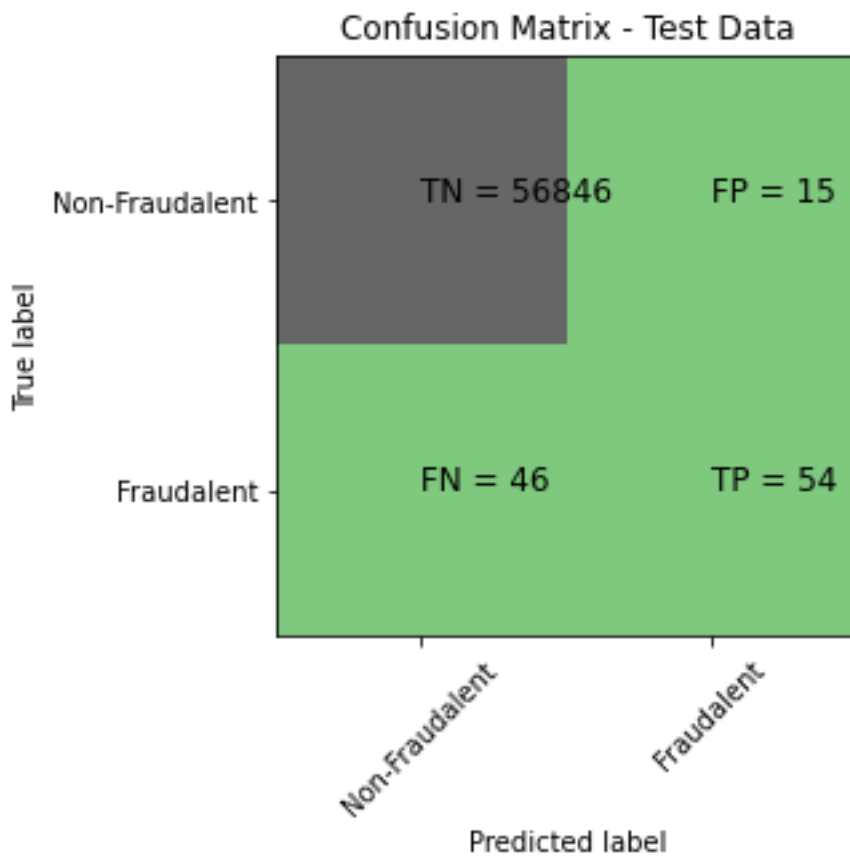
|          | Precision | Recall | F1-score | Support |
|----------|-----------|--------|----------|---------|
| 0        | 0         | 1.00   | 1.00     | 56961   |
| 1        | 1         | 0.82   | 0.64     | 100     |
| accuracy |           |        | 1.00     | 56961   |
| macro    | 0.91      | 0.76   | 0.82     | 56961   |

|                  |      |      |      |       |
|------------------|------|------|------|-------|
| average          |      |      |      |       |
| Weighted average | 1.00 | 1.00 | 1.00 | 56961 |

#### 5.1.1.2 Logistic Regression with L1 Regularisation:

Accuracy of Logistic model with l1 regularisation: 0.9989290918347641

Confusion Matrix



**Fig 5.5: Confusion Matrix of Logistic Regression with L1 Regularisation**

#### classification Report:

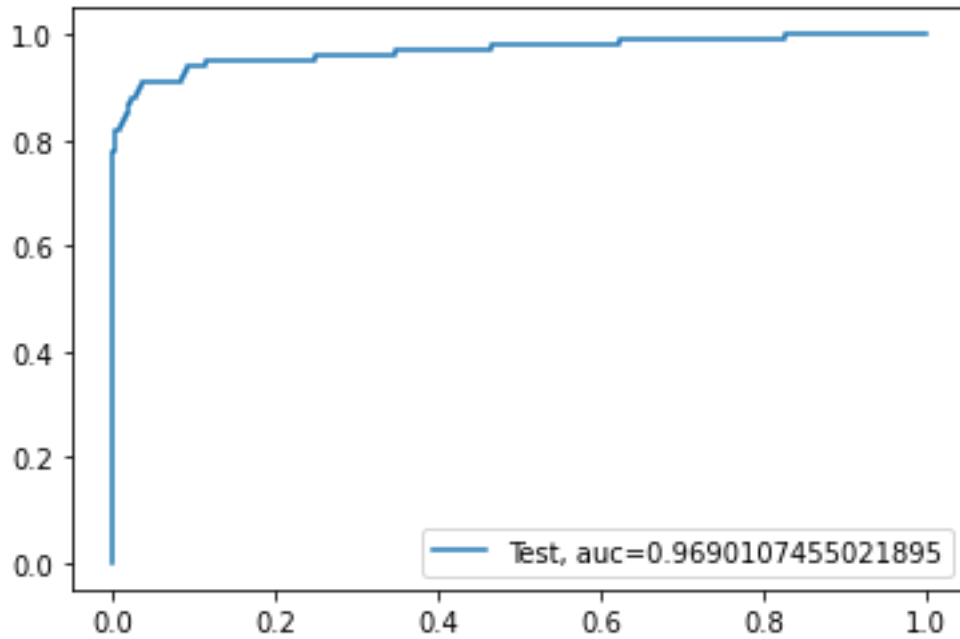
|               | Precision | Recall | F1-score | Support |
|---------------|-----------|--------|----------|---------|
| 0             | 1.00      | 1.00   | 1.00     | 56961   |
| 1             | 0.78      | 0.54   | 0.64     | 100     |
| accuracy      |           |        | 1.00     | 56961   |
| macro average | 0.89      | 0.77   | 0.82     | 56961   |

|                     |      |      |      |       |
|---------------------|------|------|------|-------|
| Weighted<br>average | 1.00 | 1.00 | 1.00 | 56961 |
|---------------------|------|------|------|-------|

l2 roc value: 0.9690107455021895

l2 threshold: 0.0014225337256554363

ROC for the test dataset 96.9%

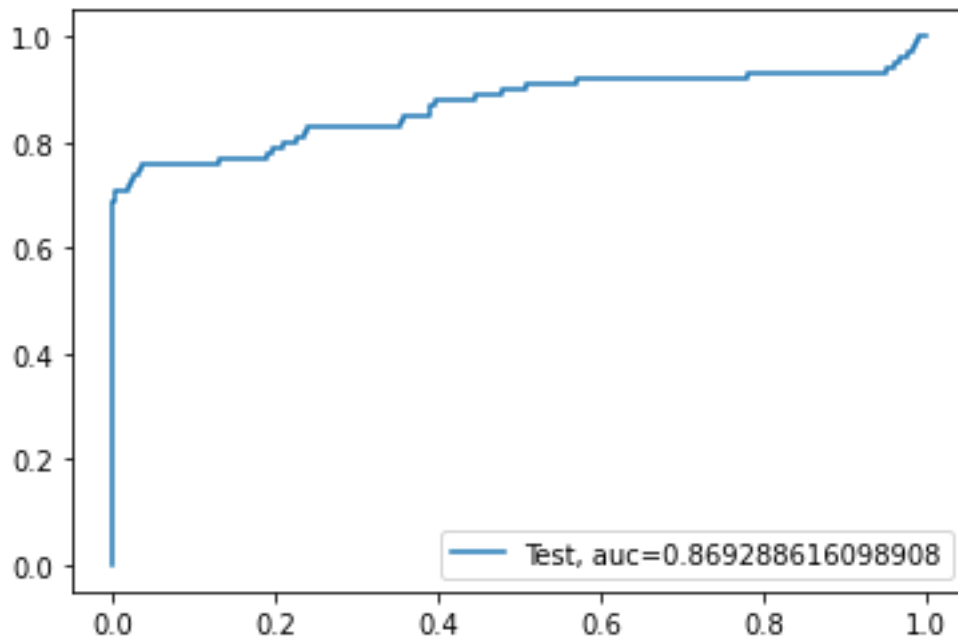


**Fig 5.6: Confusion Matrix of Logistic Regression with L1 Regularisation**

l1 roc value: 0.869288616098908

l1 threshold: 0.0472303427873661

ROC for the test dataset 86.9%



**Fig 5.7: Confusion Matrix of Logistic Regression with L2 Regularisation**

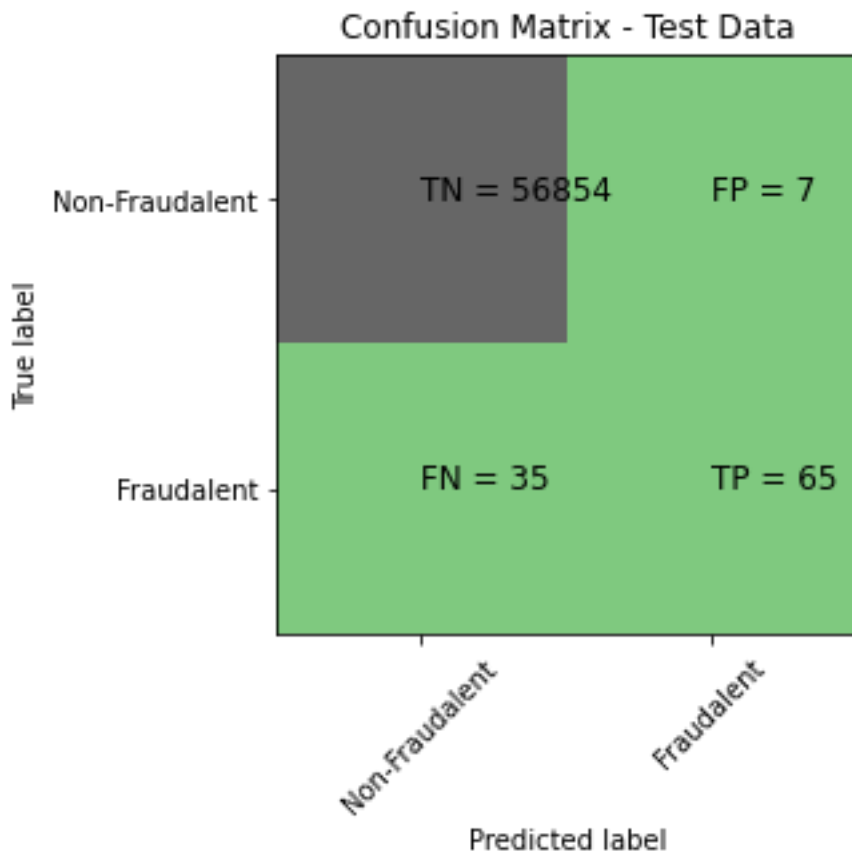
Time Taken by Model: --- 344.7267997264862 seconds ---

-----

### 5.1.2 KNN Model:

model score: 0.9992626533944278

**Confusion Matrix:**



**Fig 5.8: Confusion Matrix of KNN Model**

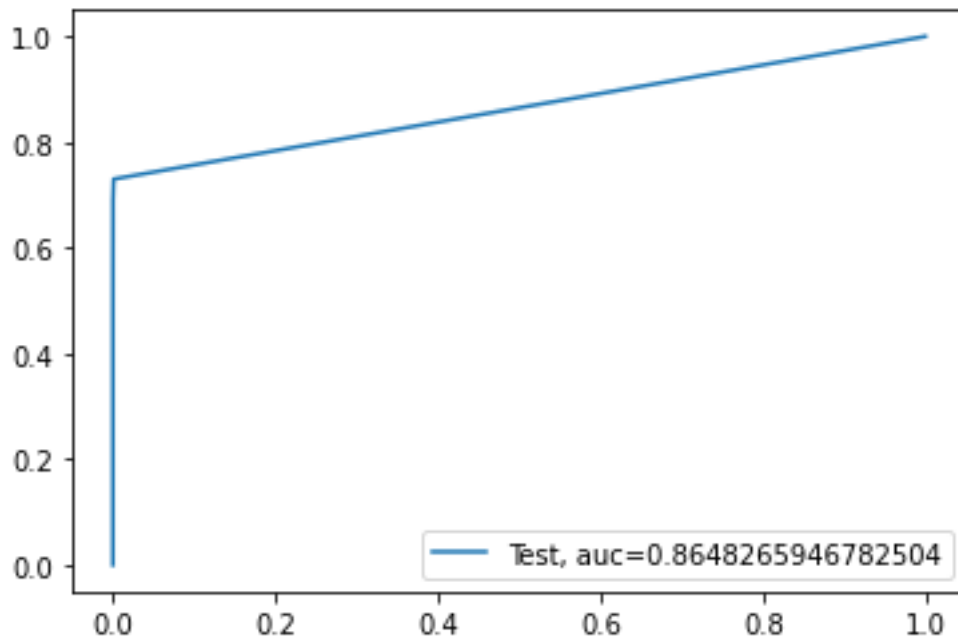
**classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56961   |
| 1                | 0.90      | 0.65   | 0.76     | 100     |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.95      | 0.82   | 0.88     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

KNN roc value: 0.8648265946782504

KNN threshold: 0.2

ROC for the test dataset 86.5%



**Fig 5.9: ROC of KNN Model**

Time Taken by Model: --- 1256.2385637760162 seconds ---

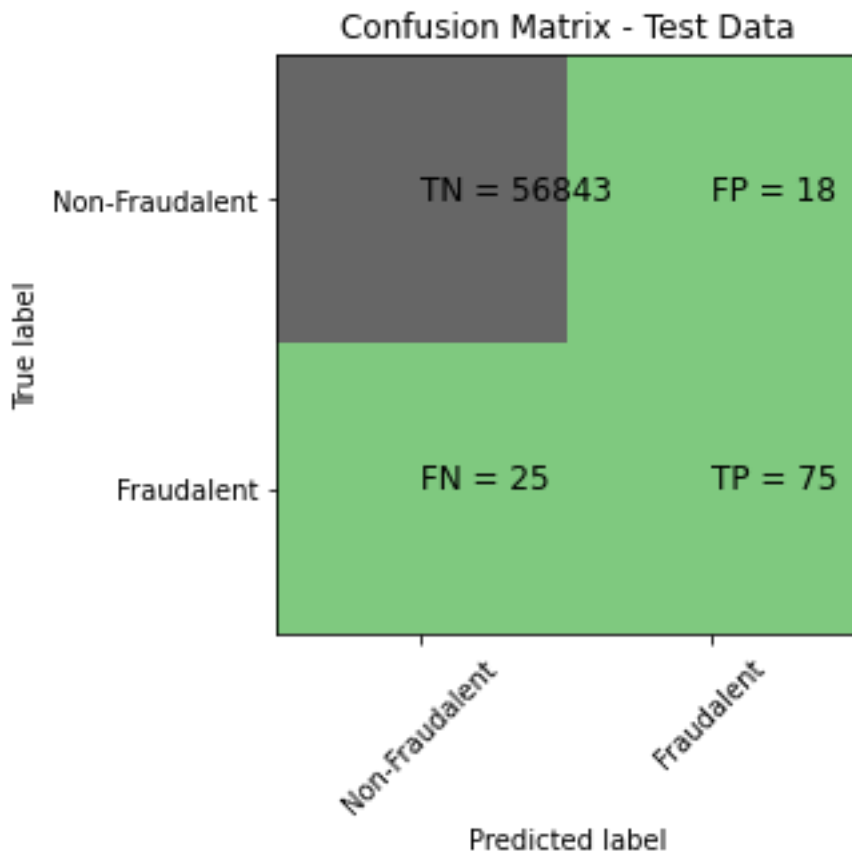
-----

### **5.1.3 Decision Tree Models with 'gini' & 'entropy' criteria:**

#### **5.1.3.1 Gini:**

gini score: 0.9992450975228665

**Confusion Matrix:**



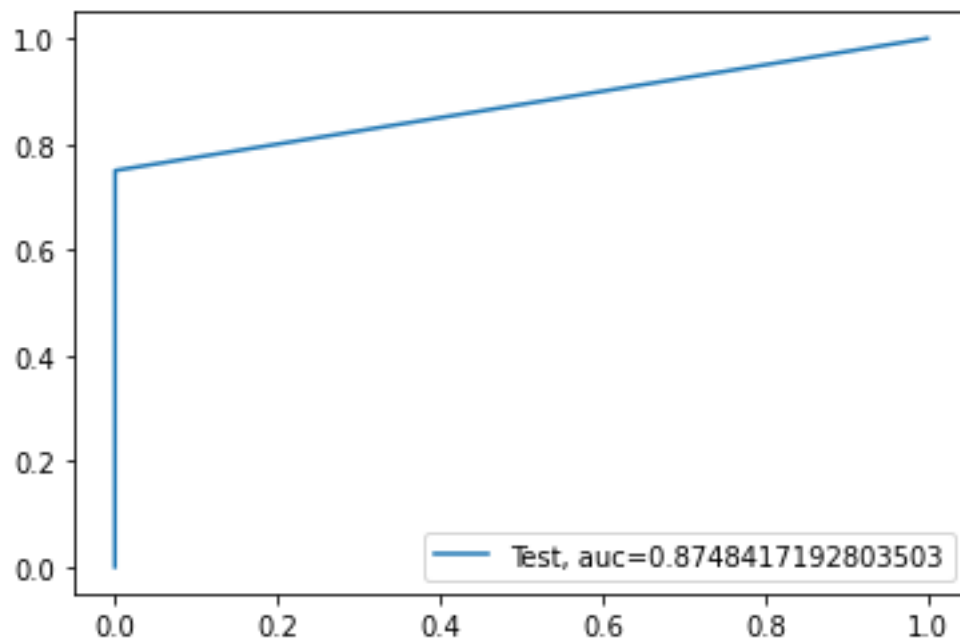
**Fig 5.10: Confusion Matrix of Decision Tree classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56961   |
| 1                | 0.81      | 0.8    | 0.89     | 100     |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.90      | 0.87   | 0.89     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

gini tree roc value: 0.8748417192803503

Tree threshold: 1.0

ROC for the test dataset 87.5%

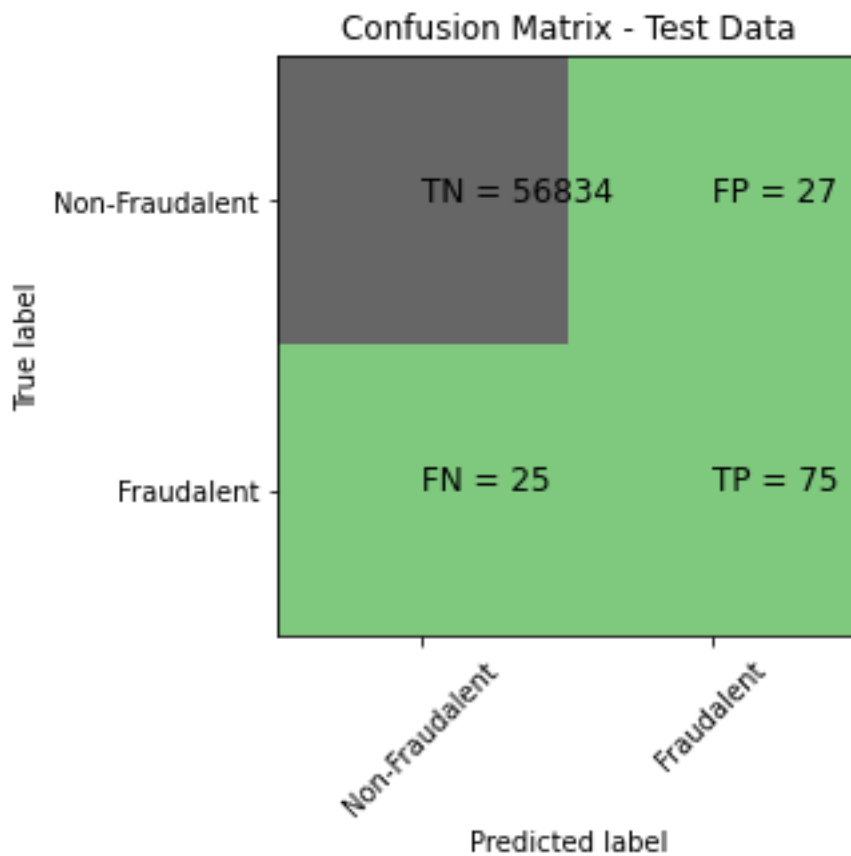


**Fig 5.11: ROC curve of Decision Tree**

### 5.1.3.2 Entropy:

entropy score: 0.9990870946788153

**Confusion Matrix:**



**Fig 5.12: Confusion Matrix of Decision Tree**

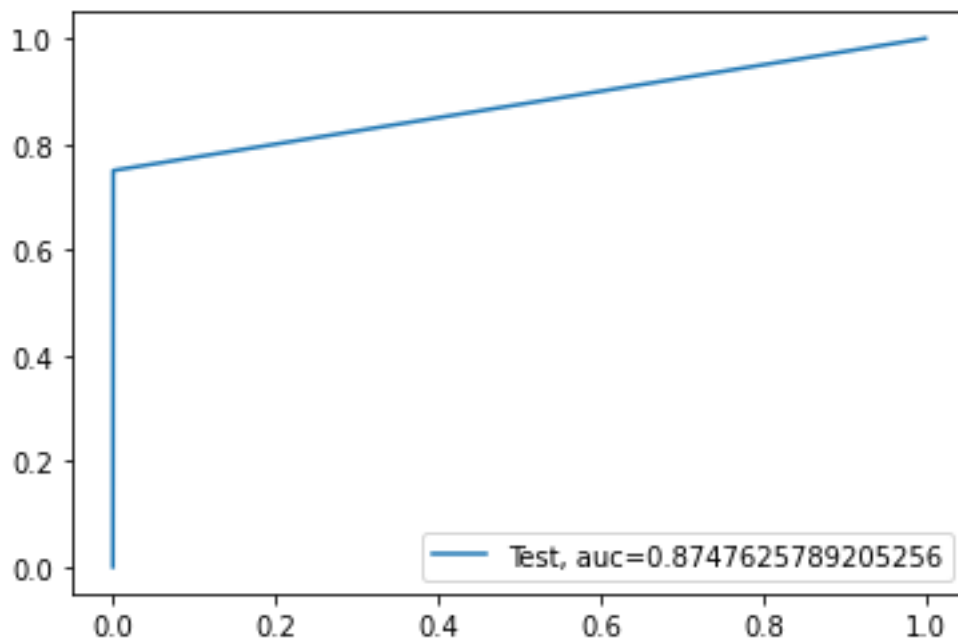
**classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56961   |
| 1                | 0.74      | 0.75   | 0.74     | 100     |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.87      | 0.87   | 0.87     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

entropy tree roc value: 0.8747625789205256

Tree threshold: 1.0

ROC for the test dataset 87.5%



**Fig 5.13: ROC curve of Decision Tree**

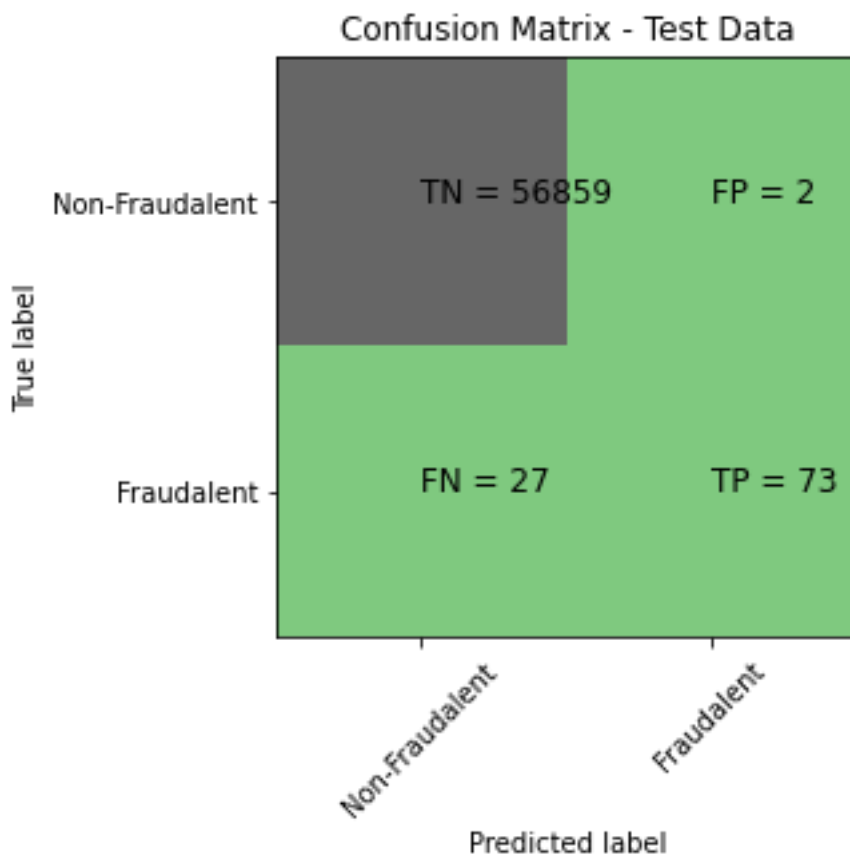
Time Taken by Model: --- 37.0897114276886 seconds ---

-----

#### 5.1.4 Random Forest Model:

Model Accuracy: 0.999490879724724

Confusion Matrix



**Fig 5.14: Confusion Matrix of Random Forest Model**

**classification Report:**

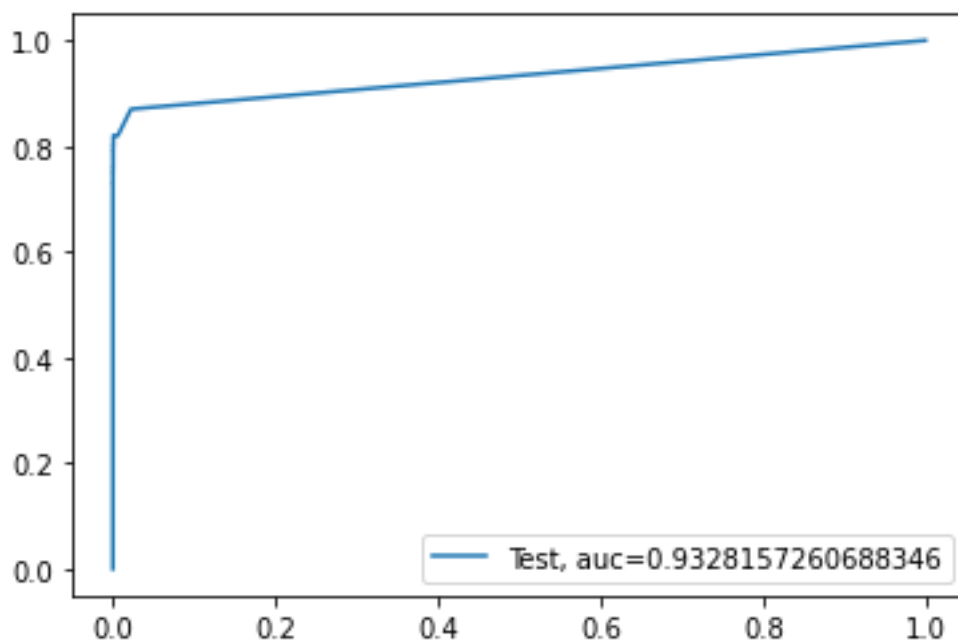
|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56961   |
| 1                | 0.97      | 0.73   | 0.83     | 100     |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.99      | 0.86   | 0.92     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

Random Forest roc value: 0.9328157260688346

Random Forest threshold: 0.01

---

ROC for the test dataset 93.3%



**Fig 5.15: ROC curve of Random Forest model**

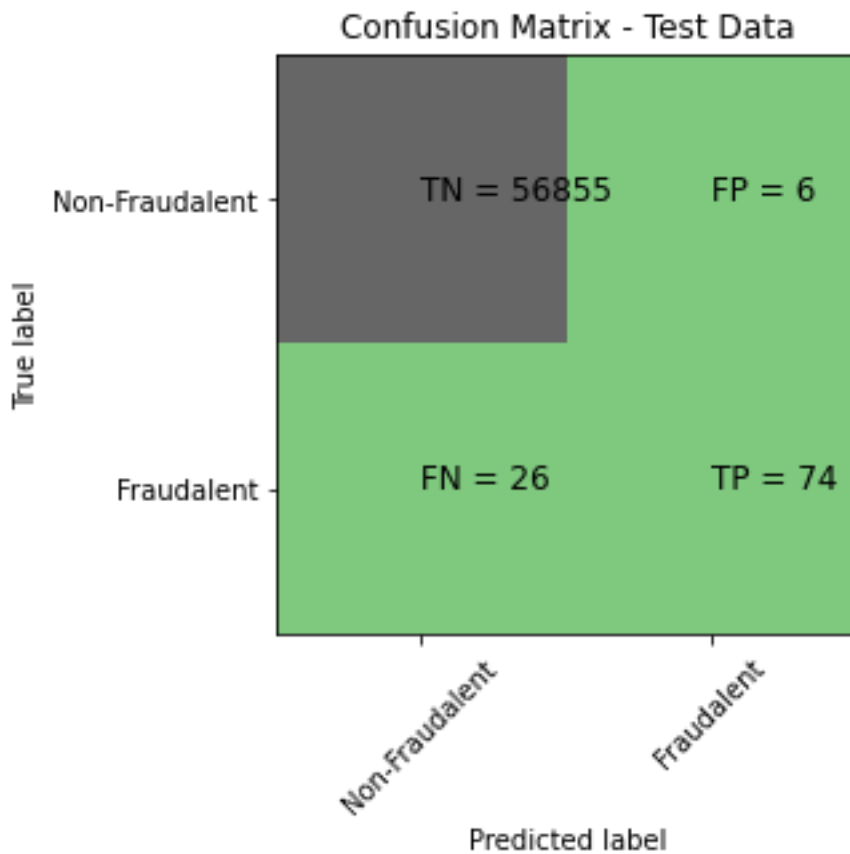
Time Taken by Model: --- 214.1870892047882 seconds ---

-----

#### **5.1.5. XG Boost Model:**

Model Accuracy: 0.9994382121100402

**Confusion Matrix:**



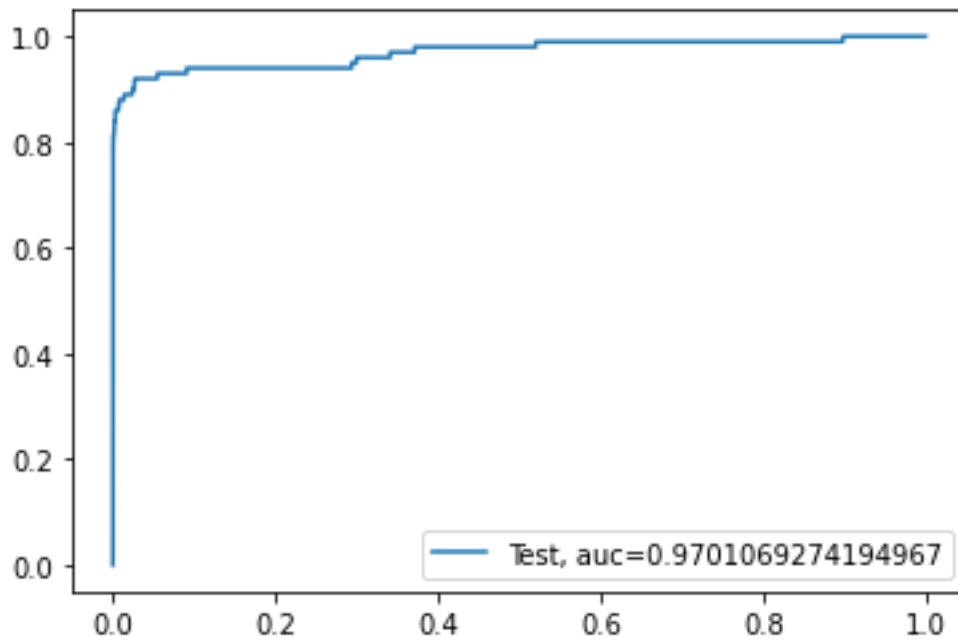
**Fig 5.16: Confusion Matrix of XG Boost classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56961   |
| 1                | 0.93      | 0.74   | 0.82     | 100     |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.96      | 0.87   | 0.91     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

XG boost roc value: 0.9701069274194967

XG Boost threshold: 0.0014467865694314241

ROC for the test dataset 97.0%



**Fig 5.17: ROC curve of XG Boost**

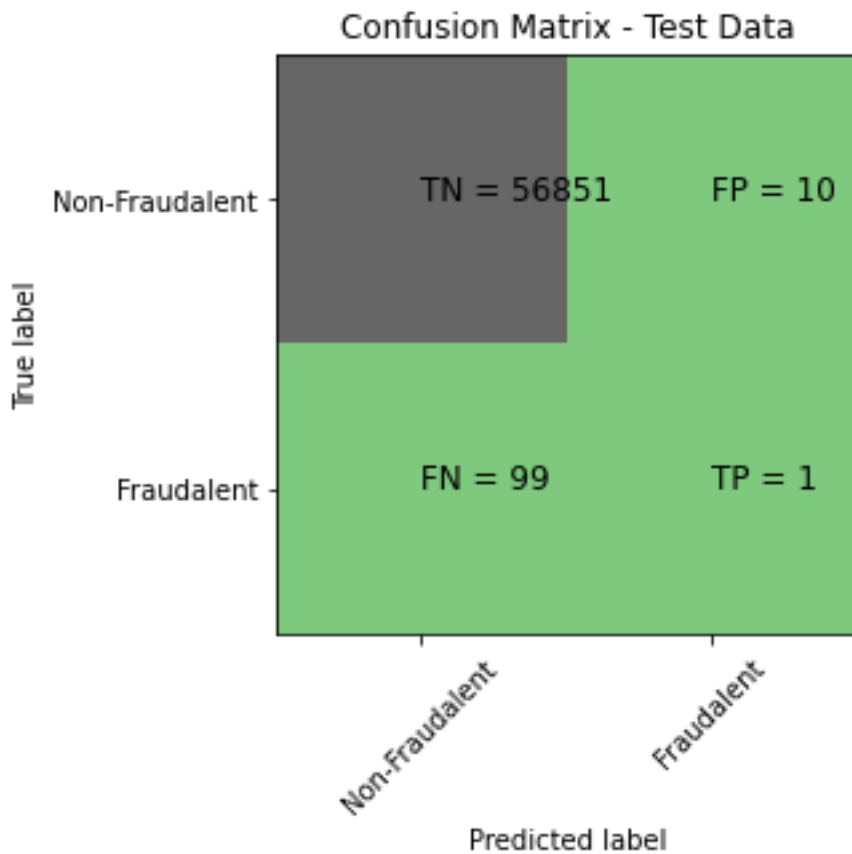
Time Taken by Model: --- 53.274091482162476 seconds ---

-----

#### **5.1.6 SVM Model with Sigmoid Kernel:**

Accuracy score: 0.9980864099998245

**Confusion Matrix:**



**Fig 5.18: Confusion Matrix of SVM model**

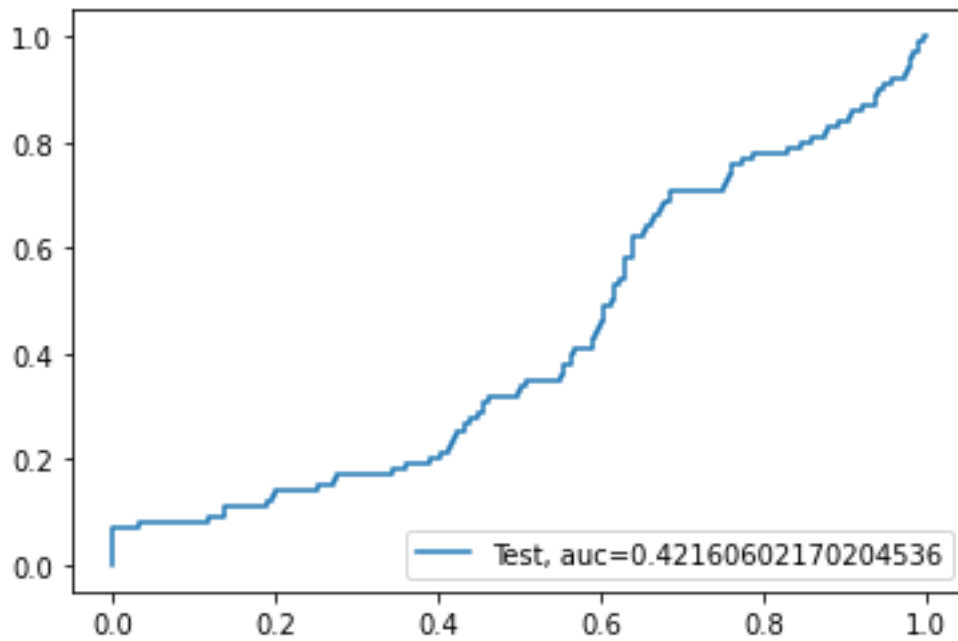
**classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56961   |
| 1                | 0.09      | 0.01   | 0.02     | 100     |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.54      | 0.50   | 0.51     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

SVM roc value: 0.42160602170204536

SVM threshold: 0.002741365025150363

ROC for the test dataset 42.2%



**Fig 5.19: ROC curve of SVM Model**

Time Taken by Model: --- 633.070867061615 seconds ---

### 5.1.7 Results for cross validation with Repeated K Fold:

|   | Methodology                    | Model                                      | Accuracy | roc_value | threshold |
|---|--------------------------------|--------------------------------------------|----------|-----------|-----------|
| 0 | RepeatedKFold Cross Validation | Logistic Regression with L2 Regularisation | 0.998964 | 0.969011  | 0.001423  |
| 1 | RepeatedKFold Cross Validation | Logistic Regression with L1 Regularisation | 0.998929 | 0.869289  | 0.047230  |
| 2 | RepeatedKFold Cross Validation | KNN                                        | 0.999263 | 0.864827  | 0.200000  |
| 3 | RepeatedKFold Cross Validation | Tree Model with gini criteria              | 0.999245 | 0.874842  | 1.000000  |
| 4 | RepeatedKFold Cross Validation | Tree Model with entropy criteria           | 0.999087 | 0.874763  | 1.000000  |
| 5 | RepeatedKFold Cross Validation | Random Forest                              | 0.999491 | 0.932816  | 0.010000  |
| 6 | RepeatedKFold Cross Validation | XGBoost                                    | 0.999438 | 0.970107  | 0.001447  |
| 7 | RepeatedKFold Cross Validation | SVM                                        | 0.998086 | 0.421606  | 0.002741  |

**Fig 5.20: Results for cross validation with Repeated K Fold**

Looking at Accuracy and ROC value we have "Logistic Regression with L2 Regularization" which has provided best results for cross validation with Repeated K Fold technique.

## 5.2 Perform cross validation with StratifiedKFold:

### 5.2.1 Logistic Regression with L1 And L2 Regularisation:

Max auc roc for l1: 0.9652106893977332

Max auc roc for l2: 0.9825816914072499

#### 5.2.1.1 Logistic Regression with L1 And L2 Regularisation:

Accuracy of Logistic model with l2 regularisation: 0.998771088990713

#### Confusion Matrix:

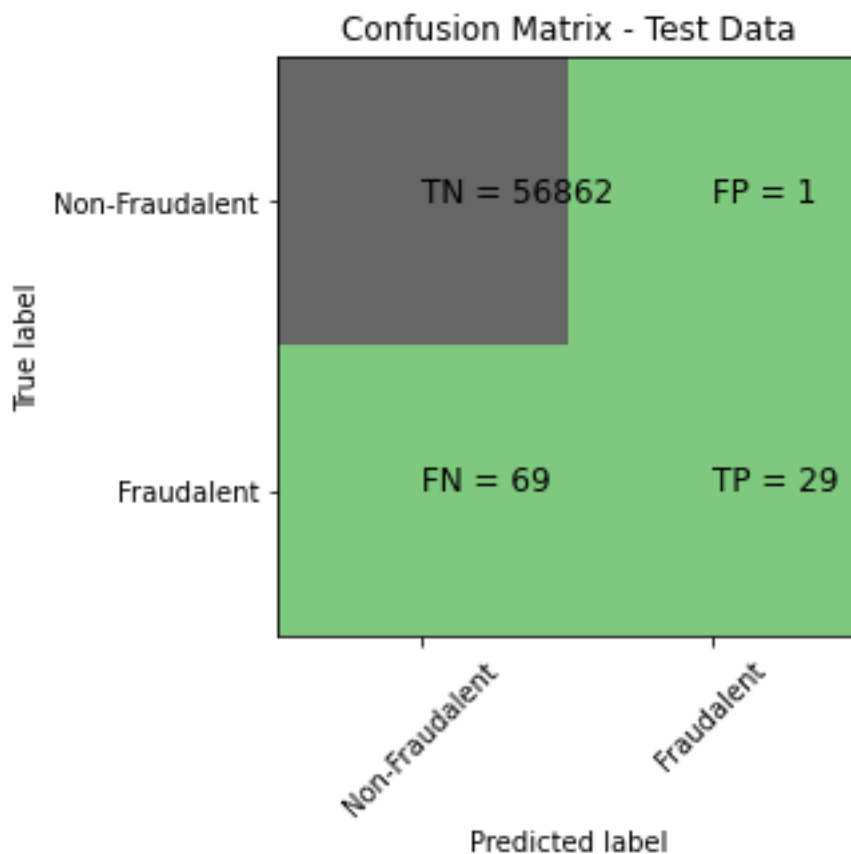


Fig 5.21: Confusion Matrix of Logistic Regression with L2 Regularisation

---

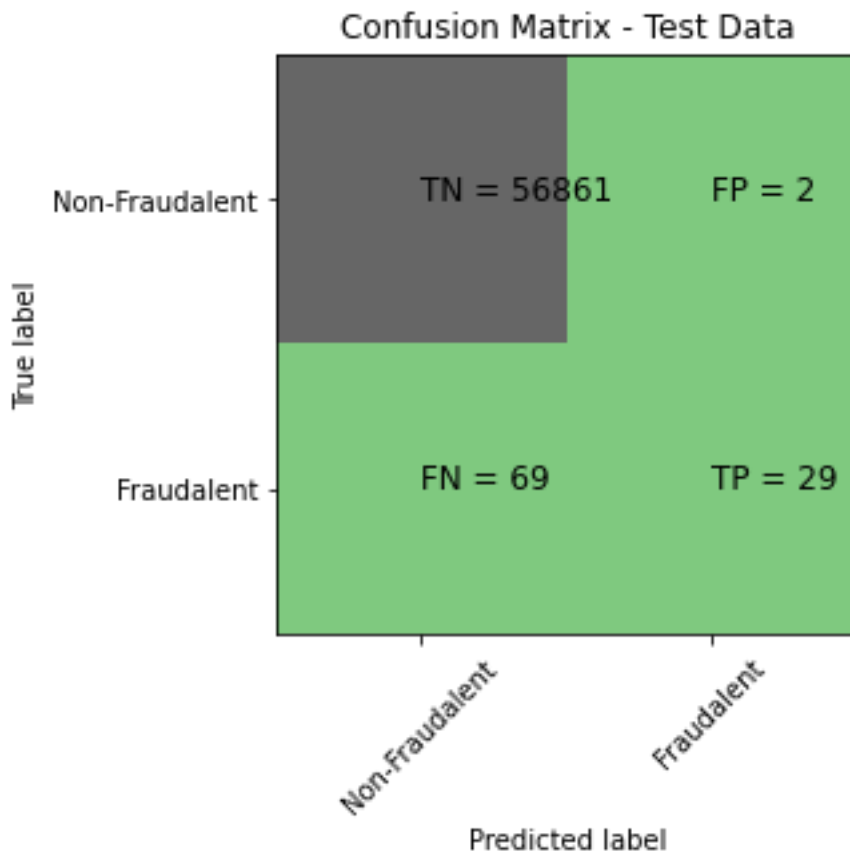
**classification Report:**

|                     | Precision | Recall | F1-score | Support |
|---------------------|-----------|--------|----------|---------|
| 0                   | 1.00      | 1.00   | 1.00     | 56961   |
| 1                   | 0.97      | 0.30   | 0.45     | 98      |
| accuracy            |           |        | 1.00     | 56961   |
| macro<br>average    | 0.98      | 0.65   | 0.73     | 56961   |
| Weighted<br>average | 1.00      | 1.00   | 1.00     | 56961   |

**5.2.1.2 Logistic Regression with L1 And L2 Regularisation:**

Accuracy of Logistic model with l1 regularisation: 0.9987535331191517

**Confusion Matrix:**



**Fig 5.22: Confusion Matrix of Logistic Regression with L1 Regularisation**

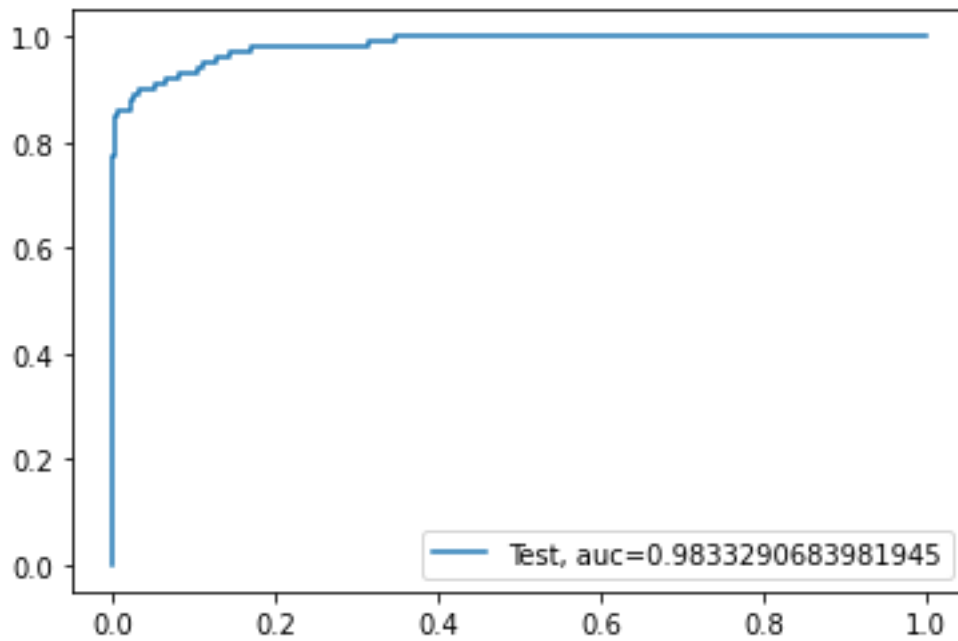
**classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56961   |
| 1                | 0.94      | 0.30   | 0.45     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.977     | 0.65   | 0.72     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

l2 roc value: 0.9833290683981945

l2 threshold: 0.0015722938315703933

ROC for the test dataset 98.3%

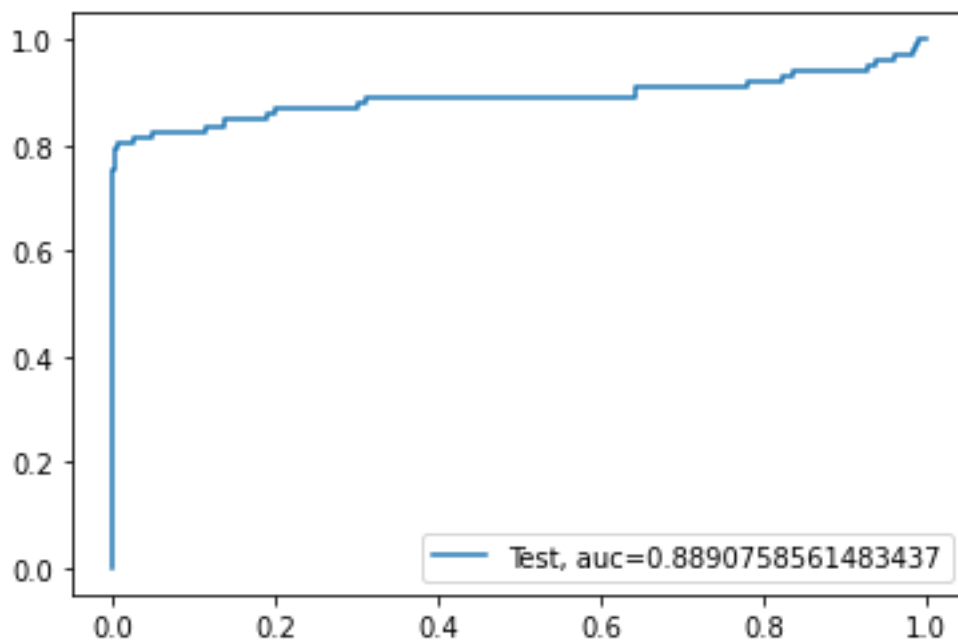


**Fig 5.23: Confusion Matrix of Logistic Regression with L1 Regularisation**

l1 roc value: 0.8890758561483437

l1 threshold: 0.021087492766199985

ROC for the test dataset 88.9%



**Fig 5.24: Confusion Matrix of Logistic Regression with L2 Regularisation**

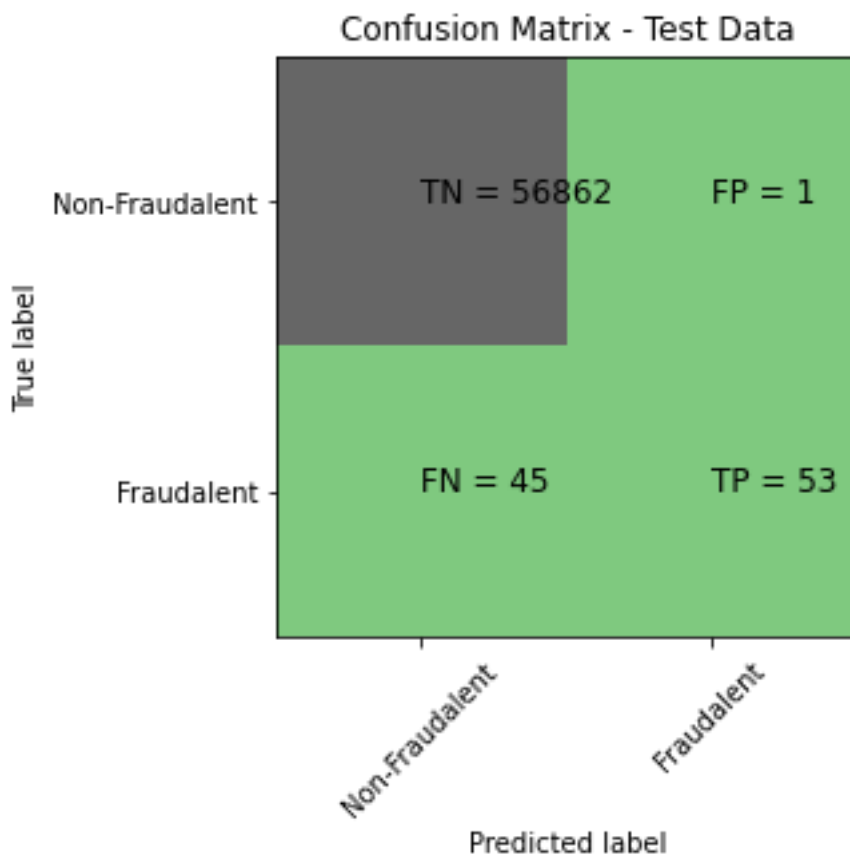
Time Taken by Model: --- 348.1904556751251 seconds ---

---

### 5.2.2 KNN Model:

model score: 0.9991924299081828

**Confusion Matrix:**



**Fig 5.25: Confusion Matrix of KNN Model**

**classification Report:**

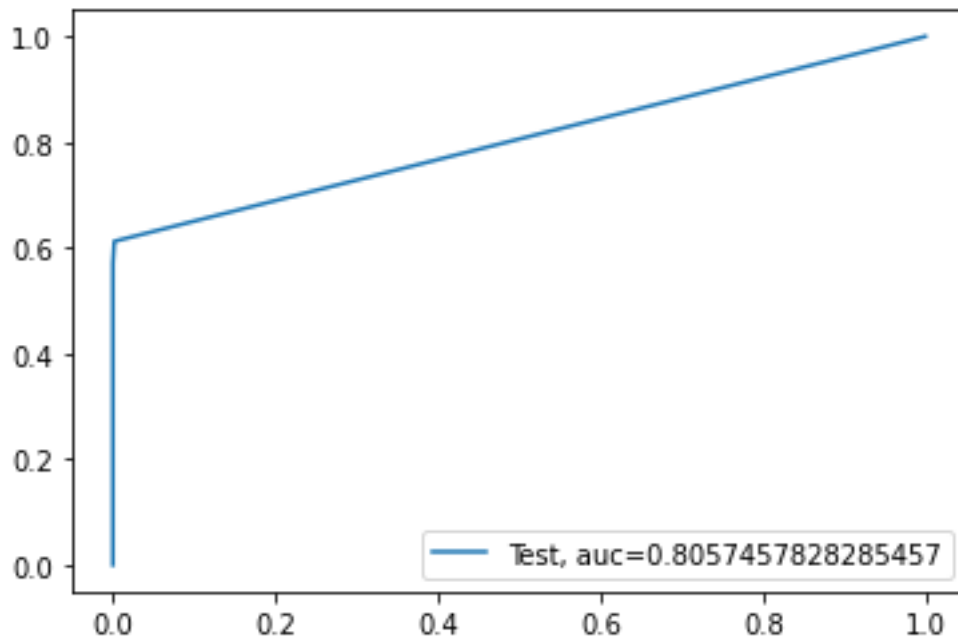
|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56961   |
| 1                | 0.98      | 0.54   | 0.70     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.99      | 0.77   | 0.85     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

KNN roc value: 0.8057457828285457

---

KNN threshold: 0.2

ROC for the test dataset 80.6%



**Fig 5.26: ROC of KNN Model**

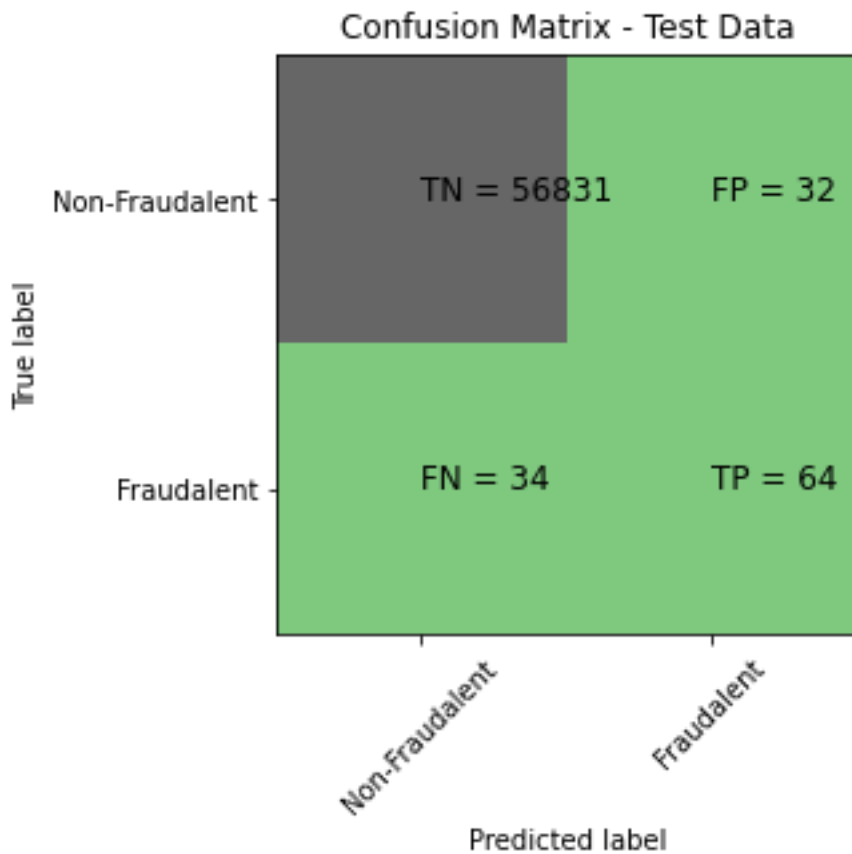
Time Taken by Model: --- 1204.463027715683 seconds ---

### **5.2.3 Decision Tree Models with 'gini' & 'entropy' criteria:**

#### **5.2.3.1 Gini:**

gini score: 0.998841312476958

**Confusion Matrix:**



**Fig 5.27: Confusion Matrix of Decision Tree**

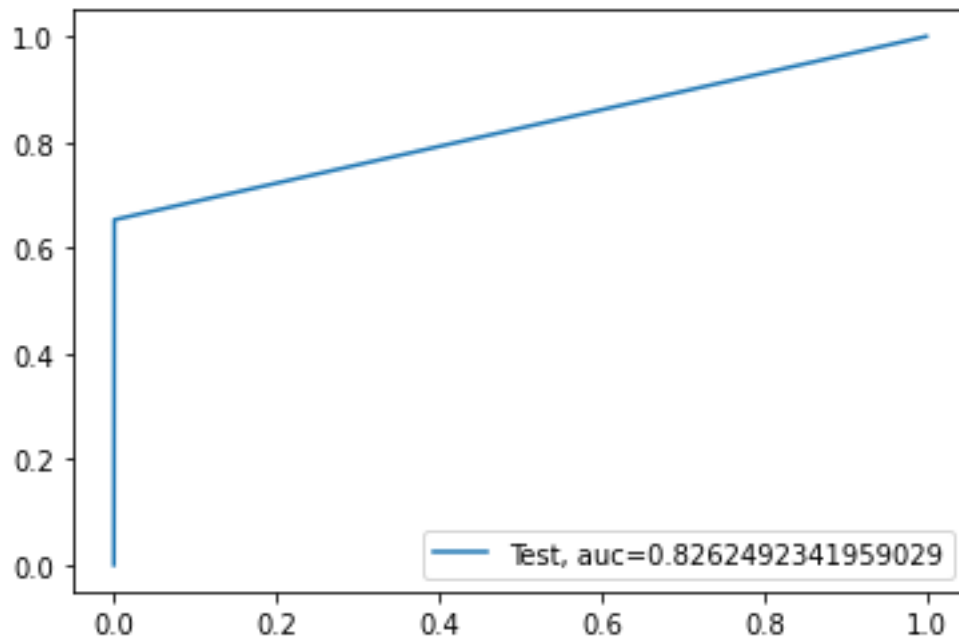
**classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56961   |
| 1                | 0.67      | 0.65   | 0.66     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.83      | 0.83   | 0.83     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

gini tree roc value: 0.8262492341959029

Tree threshold: 1.0

ROC for the test dataset 82.6%

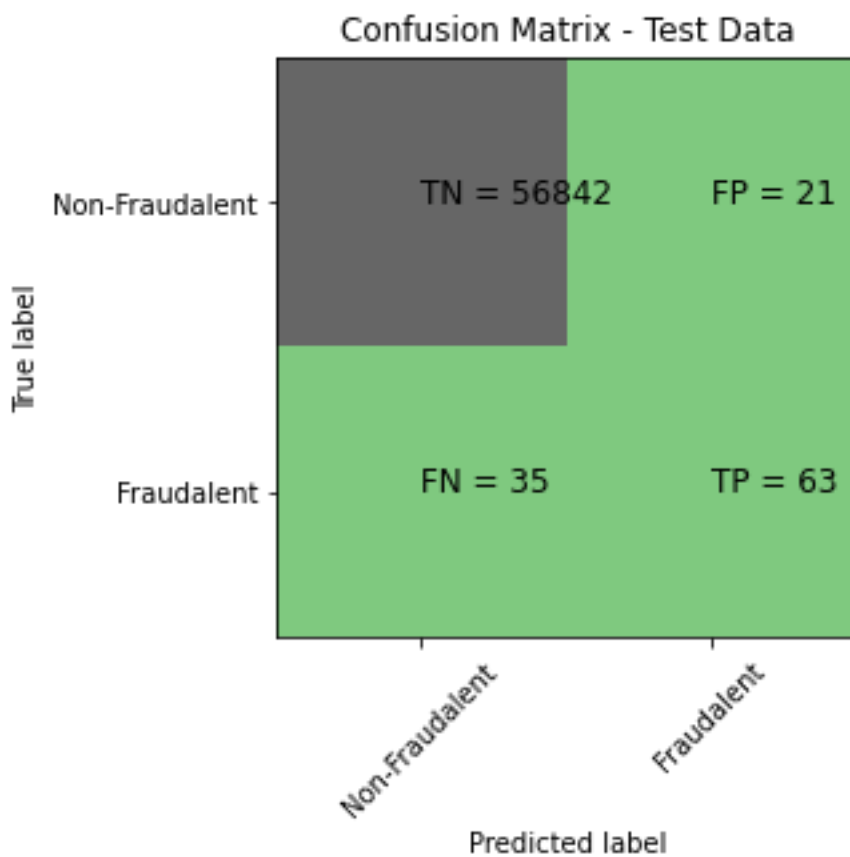


**Fig 5.28: ROC curve of Decision Tree**

### 5.2.3.2 Entropy:

entropy score: 0.9990168711925703

**Confusion Matrix:**



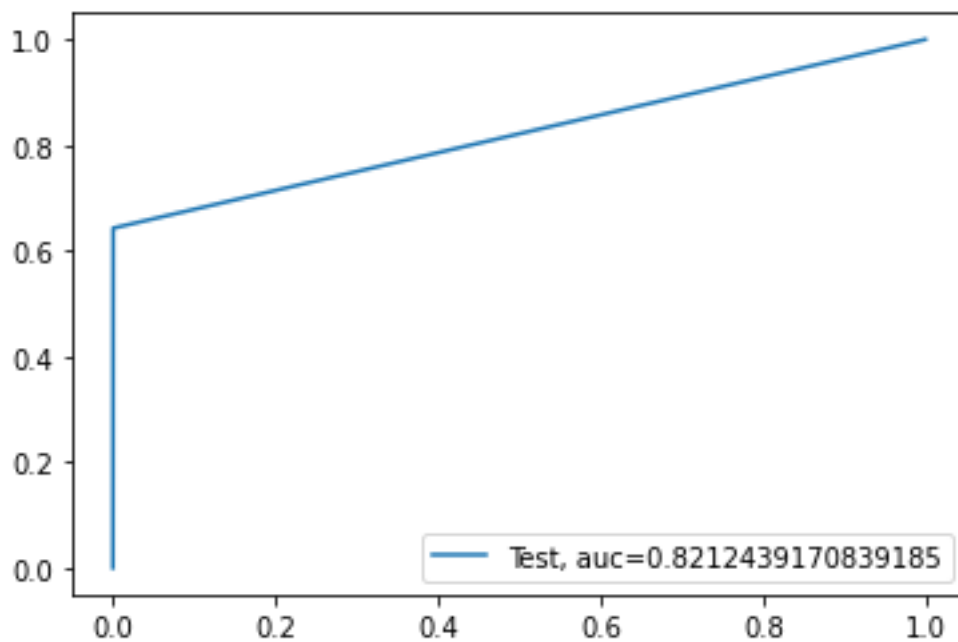
**Fig 5.29: Confusion Matrix of Decision Tree**

|                     | Precision | Recall | F1-score | Support |
|---------------------|-----------|--------|----------|---------|
| 0                   | 1.00      | 1.00   | 1.00     | 56961   |
| 1                   | 0.75      | 0.64   | 0.69     | 98      |
| accuracy            |           |        | 1.00     | 56961   |
| macro<br>average    | 0.87      | 0.82   | 0.85     | 56961   |
| Weighted<br>average | 1.00      | 1.00   | 1.00     | 56961   |

entropy tree roc value: 0.8212439170839185

Tree threshold: 1.0

ROC for the test dataset 82.1%



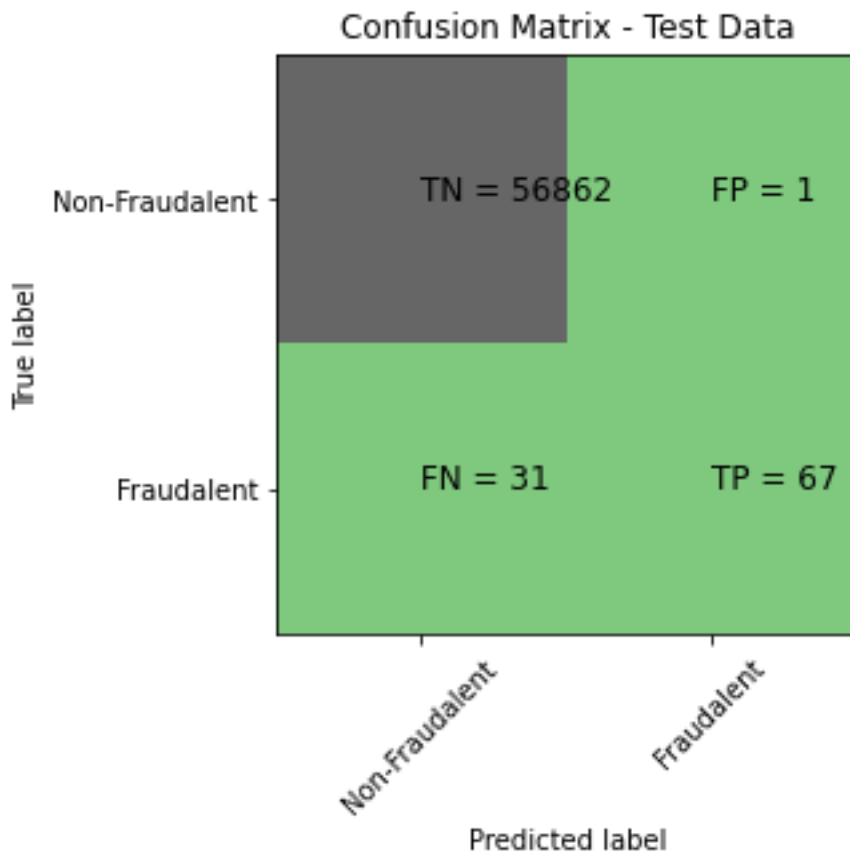
**Fig 5.30: ROC curve of Decision Tree**

Time Taken by Model: --- 35.2624135017395 seconds ---

#### 5.2.4 Random Forest Model:

Model Accuracy: 0.9994382121100402

**Confusion Matrix:**



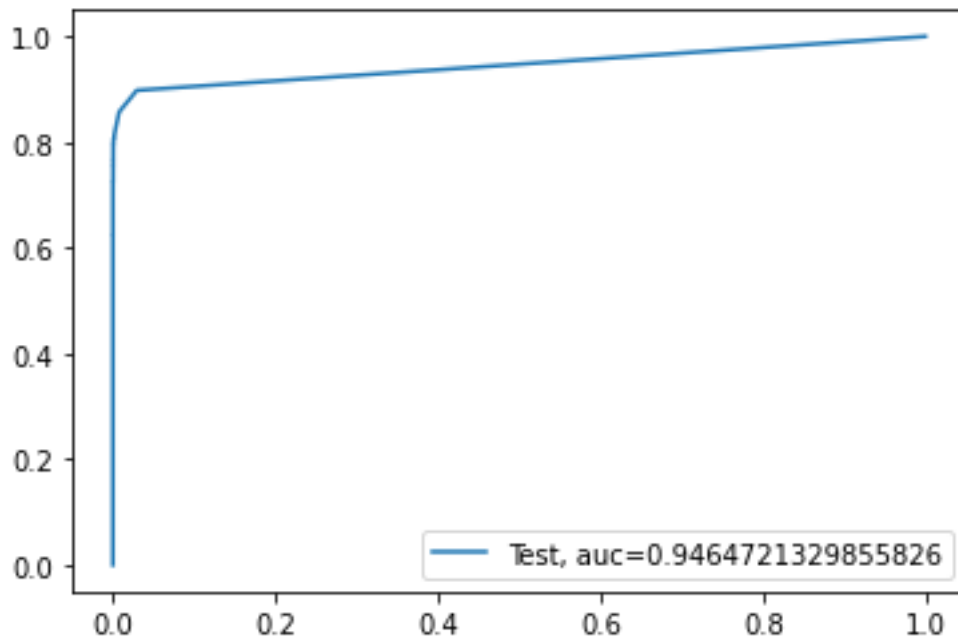
**Fig 5.31: Confusion Matrix of Random Forest Model classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56961   |
| 1                | 0.99      | 0.68   | 0.81     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.99      | 0.84   | 0.90     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

Random Forest roc value: 0.9464721329855826

Random Forest threshold: 0.01

ROC for the test dataset 94.6%



**Fig 5.32: ROC curve of Random Forest model**

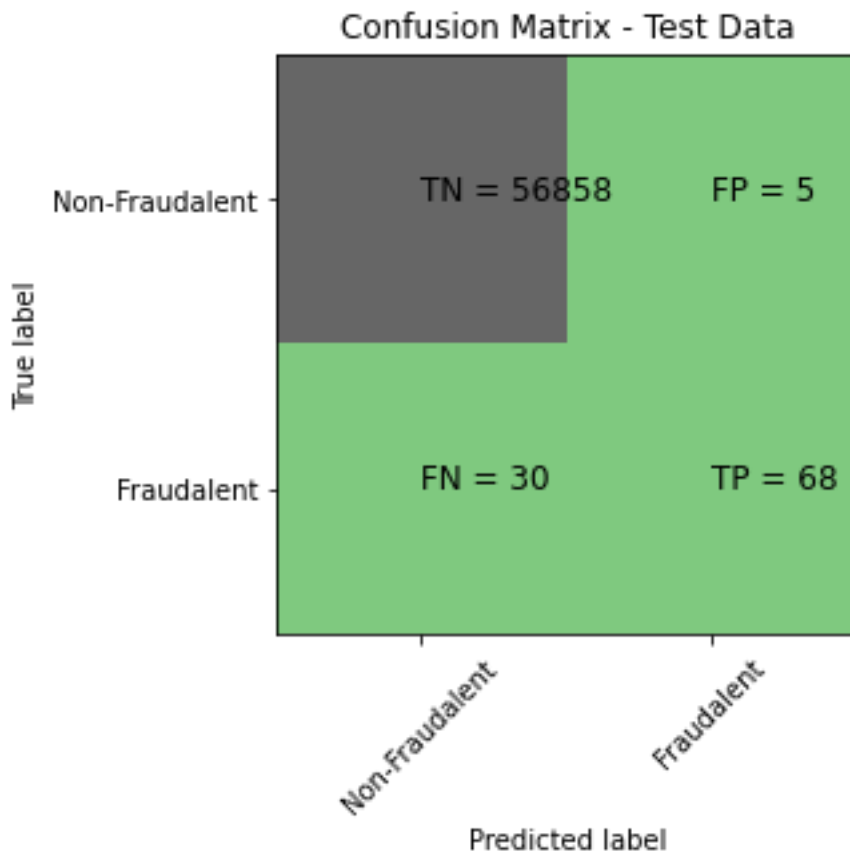
Time Taken by Model: --- 212.89058995246887 seconds ---

-----

#### **5.2.5 XG Boost Model:**

Model Accuracy: 0.9993855444953564

**Confusion Matrix:**



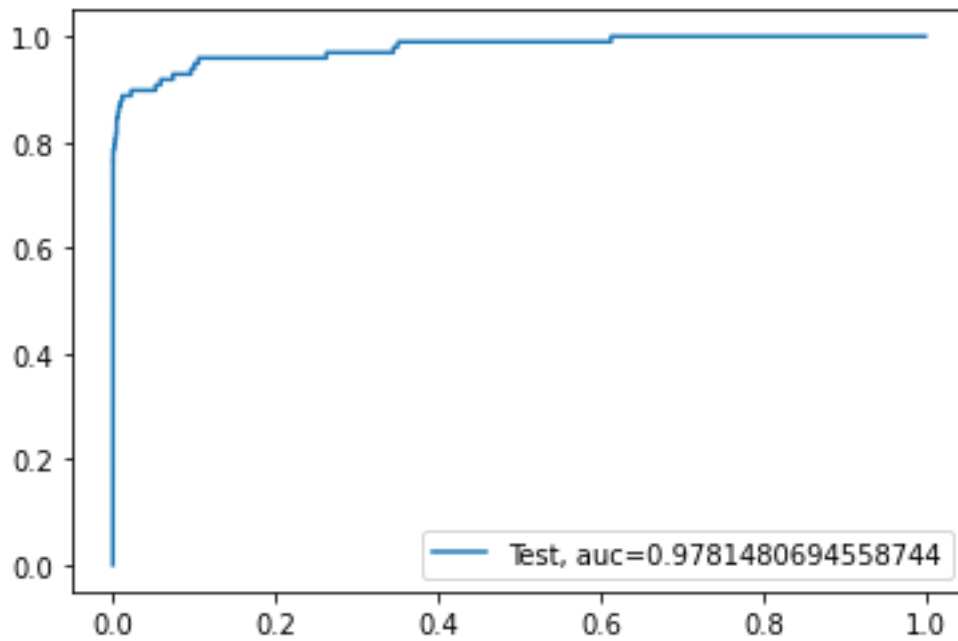
**Fig 5.33: Confusion Matrix of XG Boost classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.93      | 0.69   | 0.80     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.97      | 0.85   | 0.90     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

XG boost roc value: 0.9781480694558744

XG Boost threshold: 0.002443221863359213

ROC for the test dataset 97.8%



**Fig 5.34: ROC curve of XG Boost**

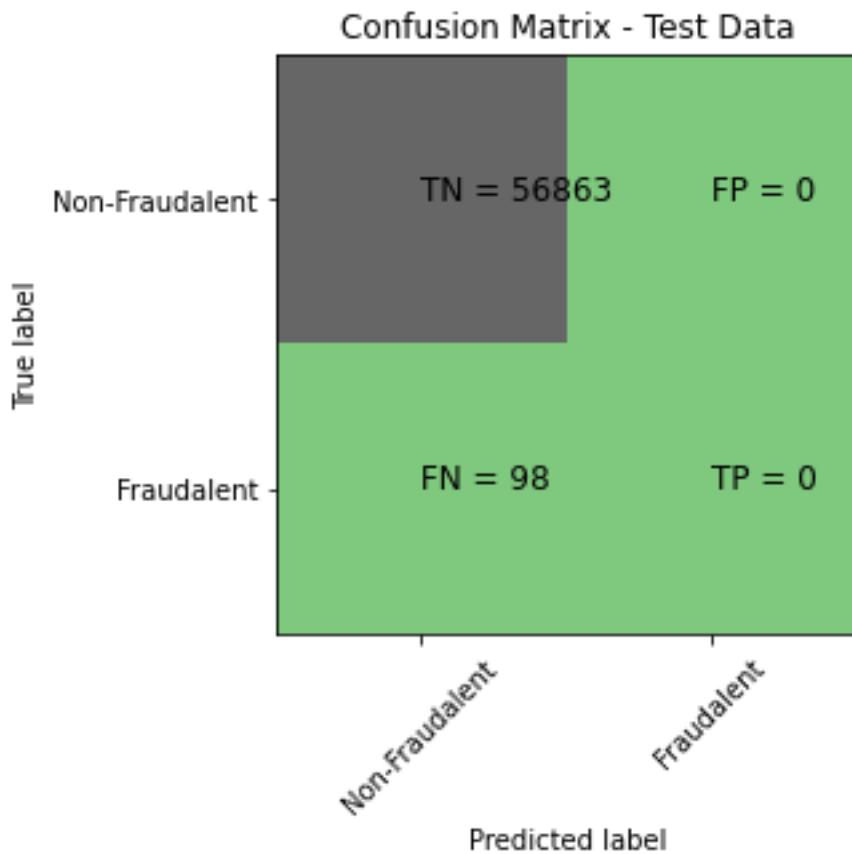
Time Taken by Model: --- 53.351813554763794 seconds ---

-----

#### **5.2.6 SVM Model with Sigmoid Kernel:**

Accuracy score: 0.9982795245869981

**Confusion Matrix:**



**Fig 5.35: Confusion Matrix of SVM model**

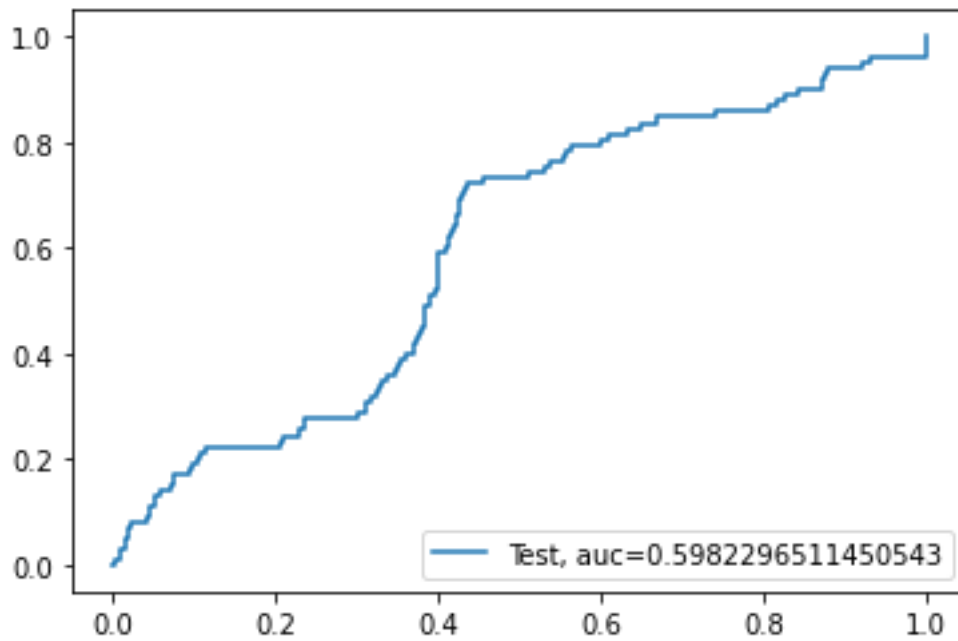
**classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.00      | 0.00   | 0.00     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.50      | 0.50   | 0.50     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

SVM roc value: 0.5982296511450543

SVM threshold: 0.0016944278710473513

ROC for the test dataset 59.8%



**Fig 5.36: ROC curve of SVM Model**

Time Taken by Model: --- 1369.9990856647491 seconds ---

### 5.2.7 Results for cross validation with StratifiedKFold:

|    | Methodology                      | Model                                      | Accuracy | roc_value | threshold |
|----|----------------------------------|--------------------------------------------|----------|-----------|-----------|
| 0  | RepeatedKFold Cross Validation   | Logistic Regression with L2 Regularisation | 0.998964 | 0.969011  | 0.001423  |
| 1  | RepeatedKFold Cross Validation   | Logistic Regression with L1 Regularisation | 0.998929 | 0.869289  | 0.047230  |
| 2  | RepeatedKFold Cross Validation   | KNN                                        | 0.999263 | 0.864827  | 0.200000  |
| 3  | RepeatedKFold Cross Validation   | Tree Model with gini criteria              | 0.999245 | 0.874842  | 1.000000  |
| 4  | RepeatedKFold Cross Validation   | Tree Model with entropy criteria           | 0.999087 | 0.874763  | 1.000000  |
| 5  | RepeatedKFold Cross Validation   | Random Forest                              | 0.999491 | 0.932816  | 0.010000  |
| 6  | RepeatedKFold Cross Validation   | XGBoost                                    | 0.999438 | 0.970107  | 0.001447  |
| 7  | RepeatedKFold Cross Validation   | SVM                                        | 0.998086 | 0.421606  | 0.002741  |
| 8  | StratifiedKFold Cross Validation | Logistic Regression with L2 Regularisation | 0.998771 | 0.983329  | 0.001572  |
| 9  | StratifiedKFold Cross Validation | Logistic Regression with L1 Regularisation | 0.998754 | 0.889076  | 0.021087  |
| 10 | StratifiedKFold Cross Validation | KNN                                        | 0.999192 | 0.805746  | 0.200000  |
| 11 | StratifiedKFold Cross Validation | Tree Model with gini criteria              | 0.998841 | 0.826249  | 1.000000  |
| 12 | StratifiedKFold Cross Validation | Tree Model with entropy criteria           | 0.999017 | 0.821244  | 1.000000  |
| 13 | StratifiedKFold Cross Validation | Random Forest                              | 0.999438 | 0.946472  | 0.010000  |
| 14 | StratifiedKFold Cross Validation | XGBoost                                    | 0.999386 | 0.978148  | 0.002443  |
| 15 | StratifiedKFold Cross Validation | SVM                                        | 0.998280 | 0.598230  | 0.001694  |

**Fig 5.37: Results for cross validation with Stratified K Fold**

- Looking at the ROC value we have Logistic Regression with L2 Regularisation has provided best results for cross validation with Stratified K Fold technique
- As the results show Logistic Regression with L2 Regularization for Stratified K Fold cross validation provided best results
- Proceed with the model which shows the best result
- Apply the best hyperparameter on the model
- Predict on the test dataset

## I. **6. MODEL BUILDING WITH BALANCING CLASSES**

*Perform class balancing with:*

- Random Oversampling
- SMOTE
- ADASYN

### **6.1. Over Sampling with random over Sampler with stratified K fold:**

We will use Random Oversampling method to handle the class imbalance

#### **6.1.1. Logistic Regression with L1 And L2 Regularisation:**

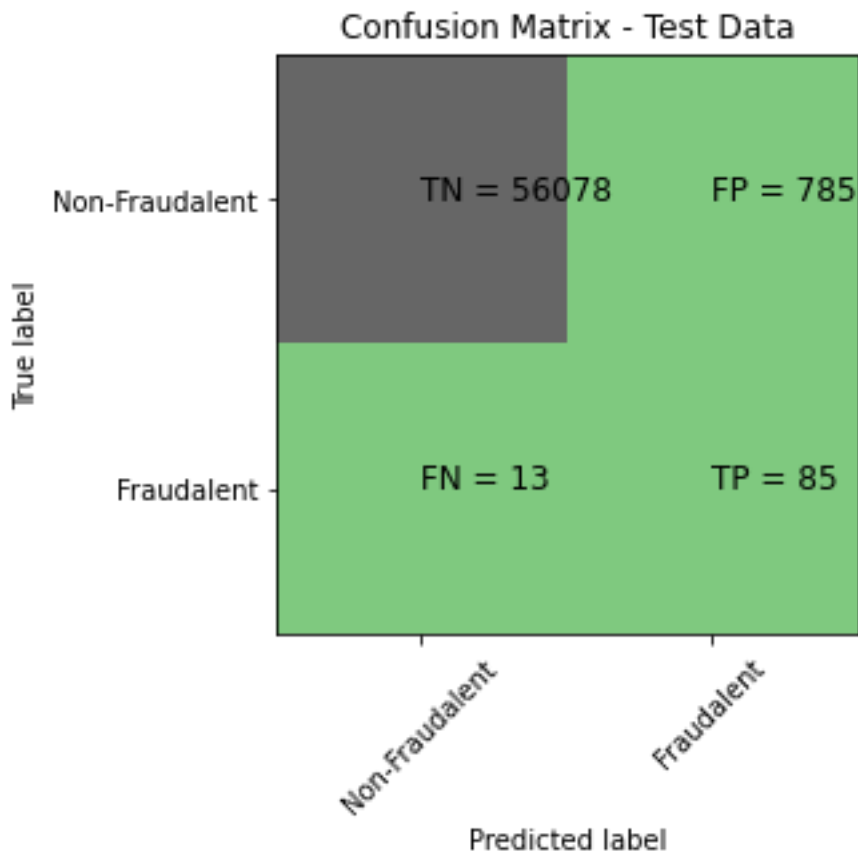
Max auc roc for l1: 0.5

Max auc roc for l2: 0.9865694580981696

##### **6.1.1.1. Logistic Regression with L2 Regularisation:**

Accuracy of Logistic model with l2 regularisation: 0.9859904144941276

**Confusion Matrix:**



**Fig 6.1: Confusion Matrix of Logistic Regression with L2 Regularisation**

**classification Report:**

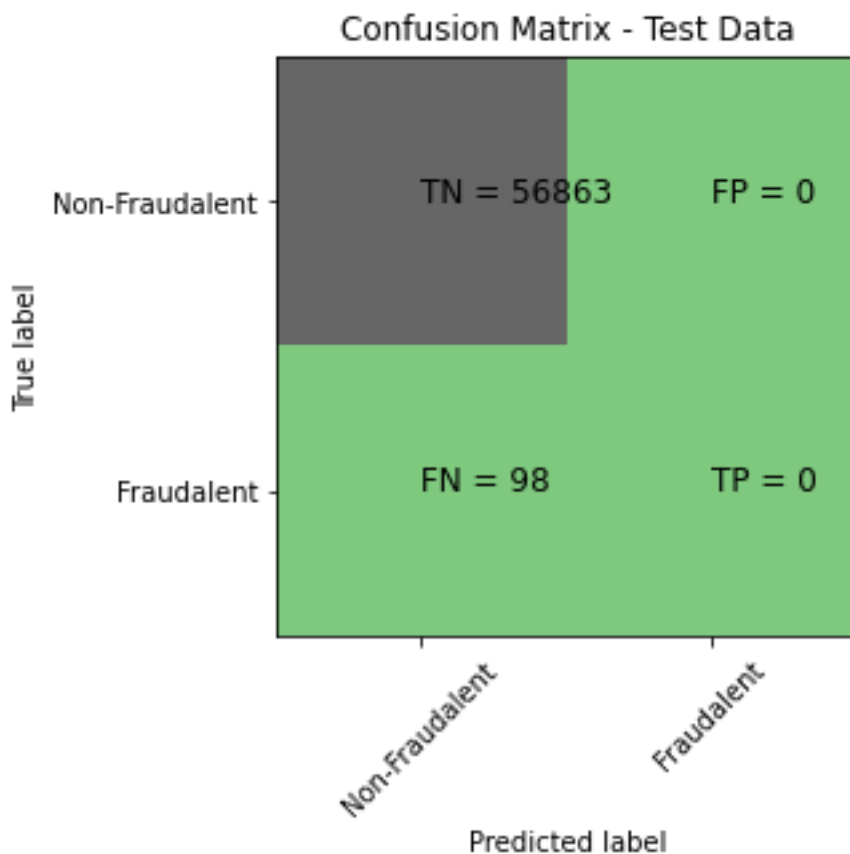
|               | Precision | Recall | F1-score | Support |
|---------------|-----------|--------|----------|---------|
| 0             | 1.00      | 0.99   | 0.99     | 56863   |
| 1             | 0.10      | 0.87   | 0.18     | 98      |
| accuracy      |           |        | 0.99     | 56961   |
| macro average | 0.55      | 0.93   | 0.58     | 56961   |
| Weighted      | 1.00      | 0.99   | 0.99     | 56961   |

|         |  |  |  |  |
|---------|--|--|--|--|
| average |  |  |  |  |
|---------|--|--|--|--|

#### 6.1.1.2. Logistic Regression with L1 Regularisation:

Accuracy of Logistic model with l1 regularisation: 0.9982795245869981

#### Confusion Matrix:



**Fig 6.2: Confusion Matrix of Logistic Regression with L1 Regularisation**

#### classification Report:

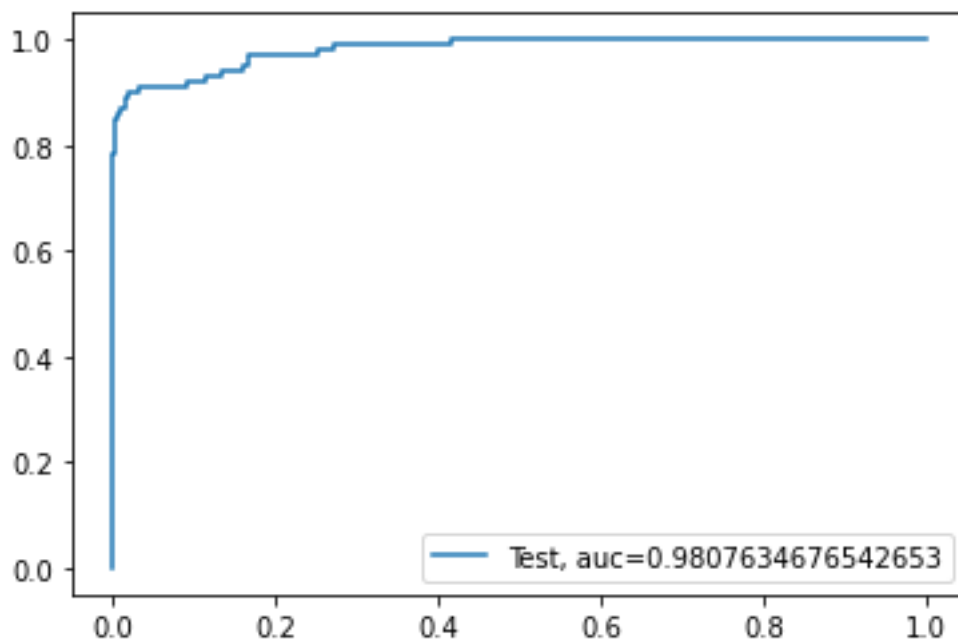
|          | Precision | Recall | F1-score | Support |
|----------|-----------|--------|----------|---------|
| 0        | 1.00      | 1.00   | 1.00     | 56863   |
| 1        | 0.0       | 0.00   | 0.00     | 98      |
| accuracy |           |        | 1.00     | 56961   |
| macro    | 0.50      | 0.50   | 0.50     | 56961   |

|                     |      |      |      |       |
|---------------------|------|------|------|-------|
| average             |      |      |      |       |
| Weighted<br>average | 1.00 | 1.00 | 1.00 | 56961 |

l2 roc value: 0.9807634676542653

l2 threshold: 0.43450954007638304

ROC for the test dataset 98.1%

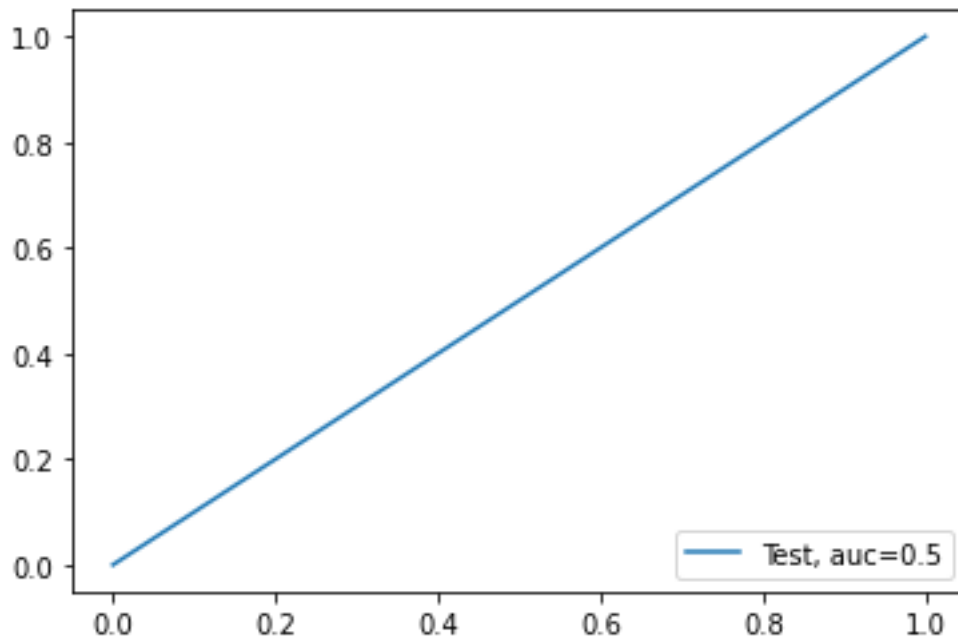


**Fig 6.3: Confusion Matrix of Logistic Regression with L1 Regularisation**

l1 roc value: 0.5

l1 threshold: 1.5

ROC for the test dataset 50.0%



**Fig 6.4: Confusion Matrix of Logistic Regression with L2 Regularisation**

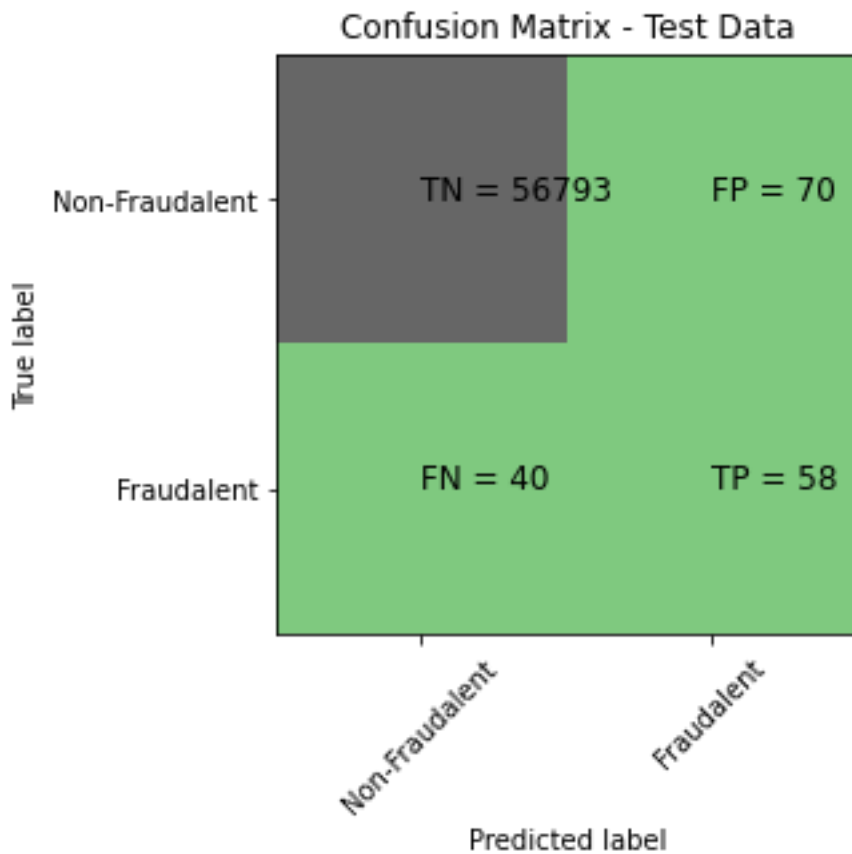
Time Taken by Model: --- 438.3937211036682 seconds ---

-----

#### **6.1.2. KNN Model:**

model score: 0.9980688541282632

**Confusion Matrix:**



**Fig 6.5: ROC of KNN Model**

**classification Report:**

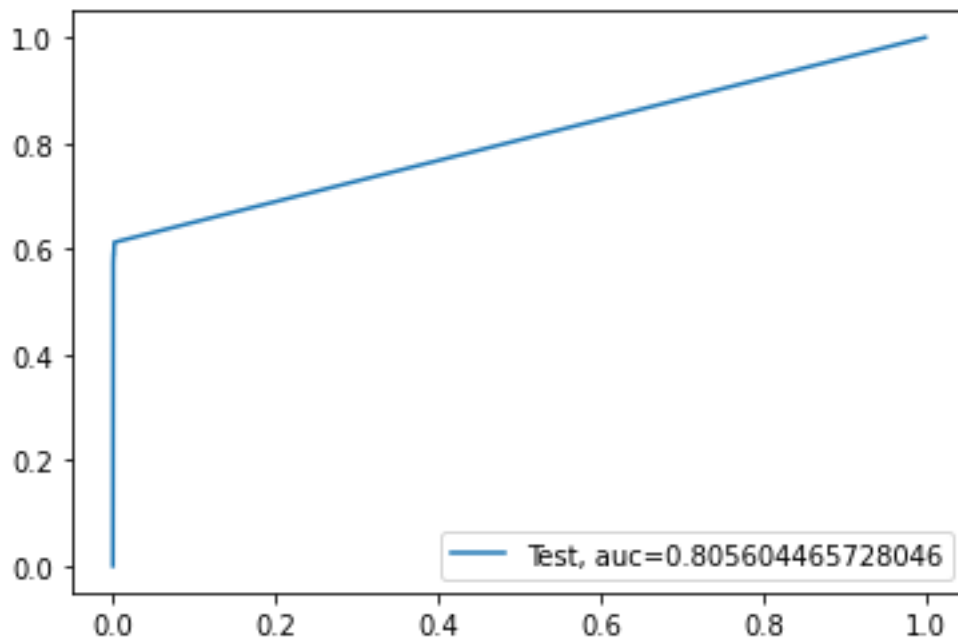
|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.45      | 0.59   | 0.51     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.73      | 0.80   | 0.76     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

KNN roc value: 0.805604465728046

KNN threshold: 0.2

---

ROC for the test dataset 80.6%



**Fig 6.6: ROC of KNN Model**

Time Taken by Model: --- 2119.8244297504425 seconds ---

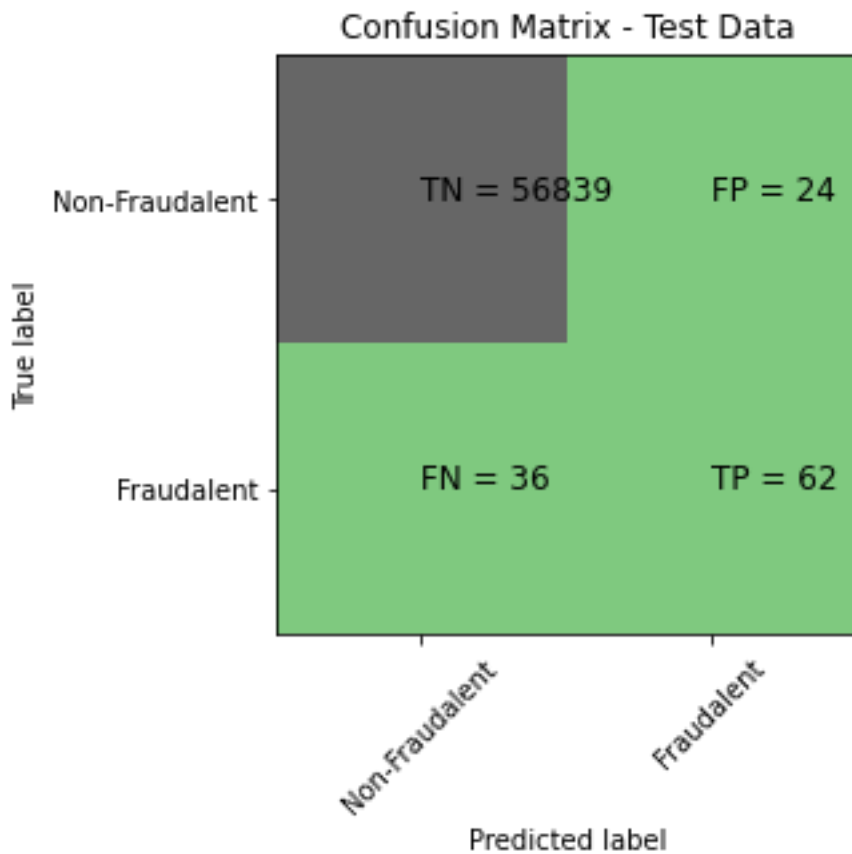
-----

### 6.1.3. Decision Tree Models with 'gini' & 'entropy' criteria:

#### 6.1.3.1 Gini:

gini score: 0.9989466477063254

**Confusion Matrix:**



**Fig 6.7: Confusion Matrix of Decision Tree**

#### **classification Report:**

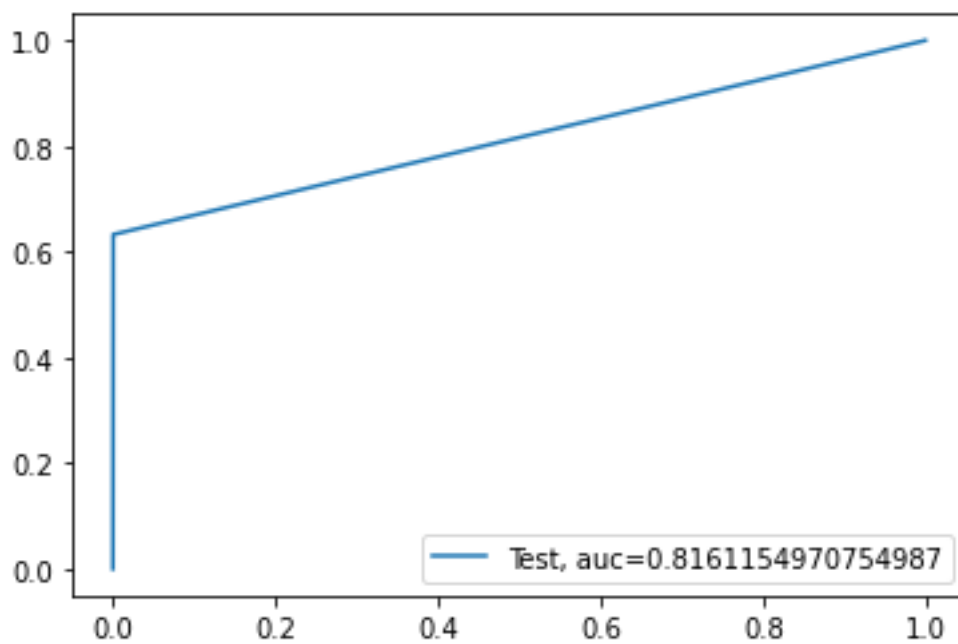
|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.72      | 0.63   | 0.67     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.86      | 0.82   | 0.84     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

#### **6.1.3.1 Gini:**

gini tree roc value: 0.8161154970754987

Tree threshold: 1.0

ROC for the test dataset 81.6%

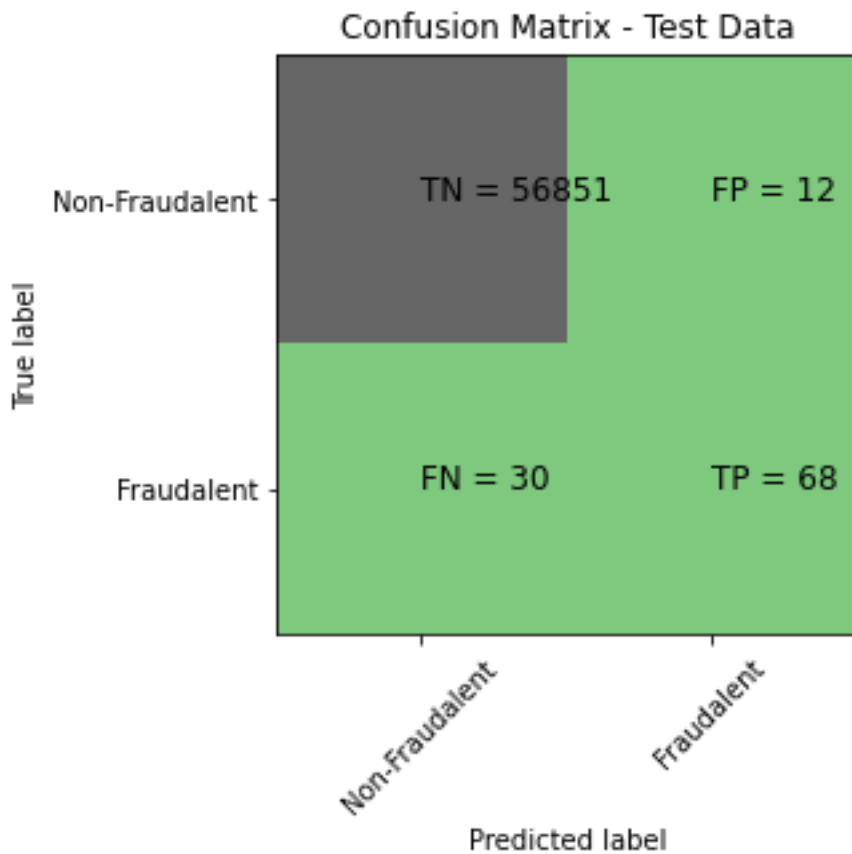


**Fig 6.8: ROC curve of Decision Tree**

#### **6.1.3.2. Entropy:**

entropy score: 0.9992626533944278

**Confusion Matrix:**



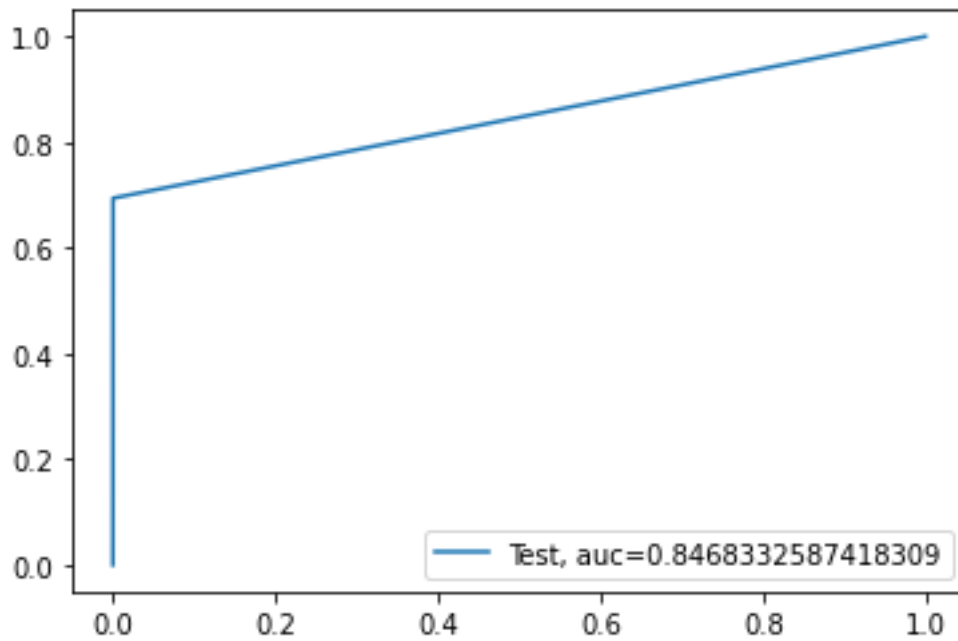
**Fig 6.9: Confusion Matrix of Decision Tree classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.85      | 0.69   | 0.76     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.92      | 0.85   | 0.88     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

entropy tree roc value: 0.8468332587418309

Tree threshold: 1.0

ROC for the test dataset 84.7%



**Fig 6.10: ROC curve of Decision Tree**

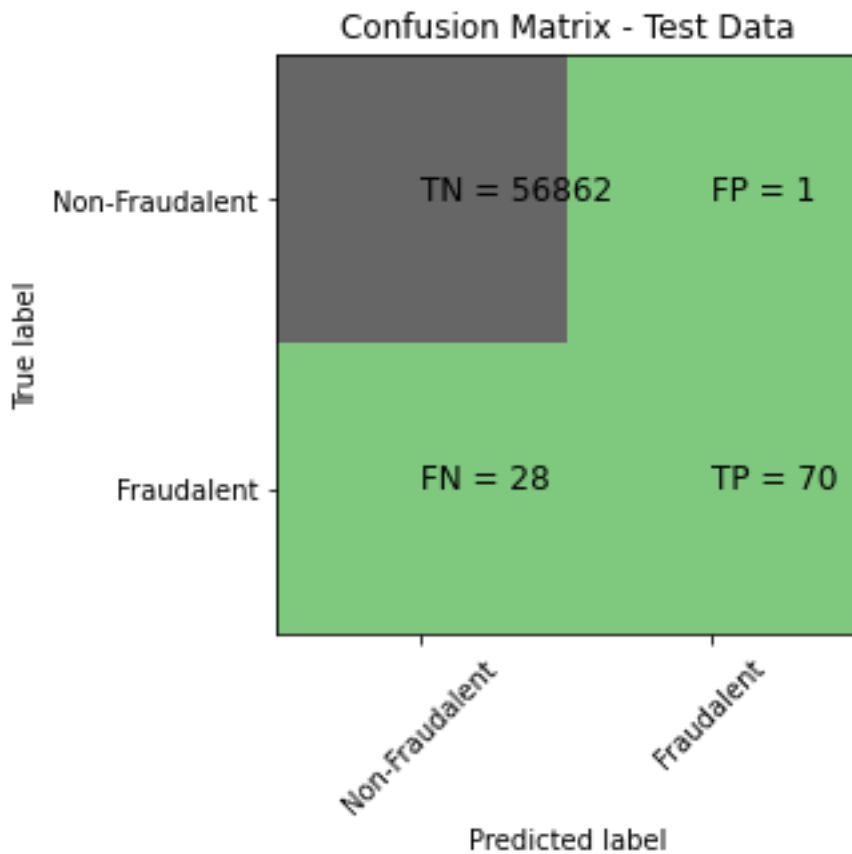
Time Taken by Model: --- 24.41987657546997 seconds ---

-----

#### **6.1.4. Random Forest Model:**

Model Accuracy: 0.999490879724724

**Confusion Matrix:**



**Fig 6.11: Confusion Matrix of Random Forest Model**

**classification Report:**

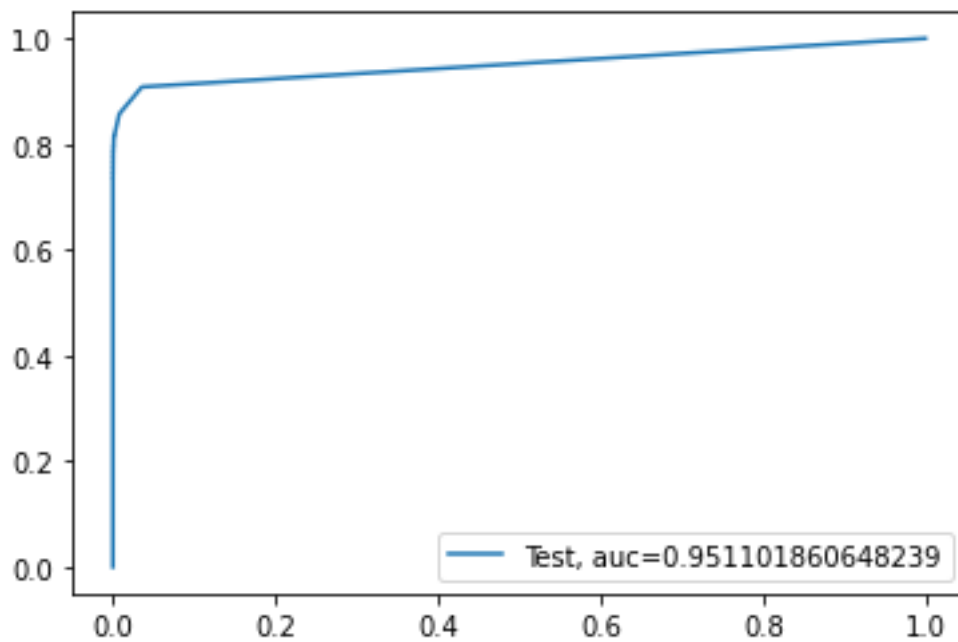
|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.99      | 0.71   | 0.83     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.99      | 0.86   | 0.91     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

---

Random Forest roc value: 0.951101860648239

Random Forest threshold: 0.01

ROC for the test dataset 95.1%



**Fig 6.12: ROC curve of Random Forest model**

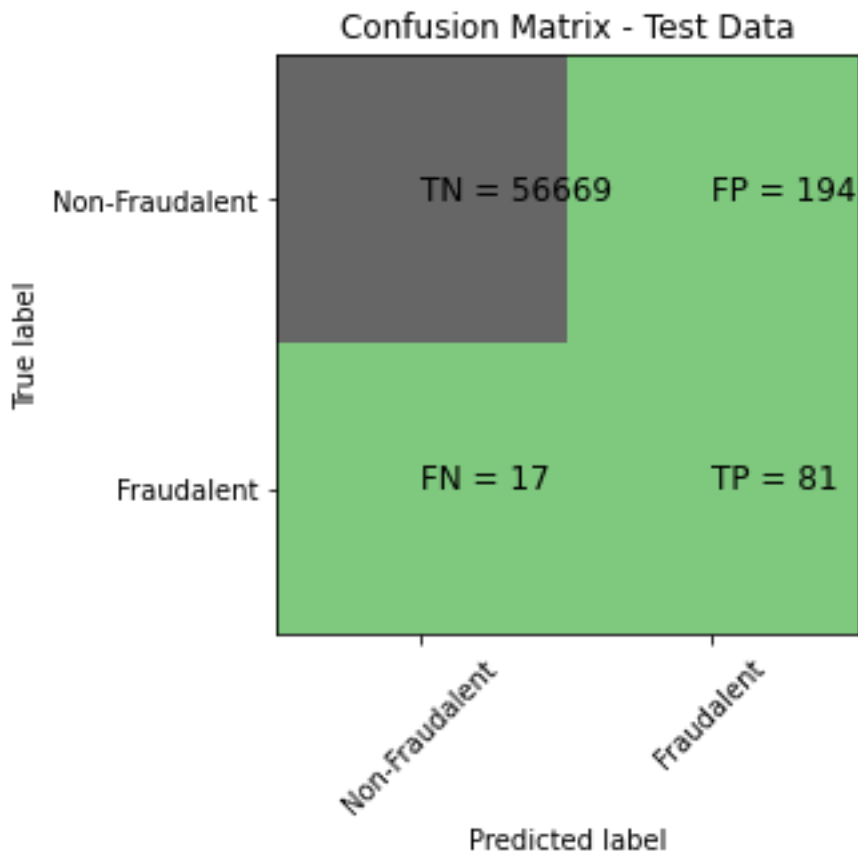
Time Taken by Model: --- 172.4803009033203 seconds ---

---

#### **6.1.6. XG Boost Model:**

Model Accuracy: 0.9962957111005776

**Confusion Matrix:**



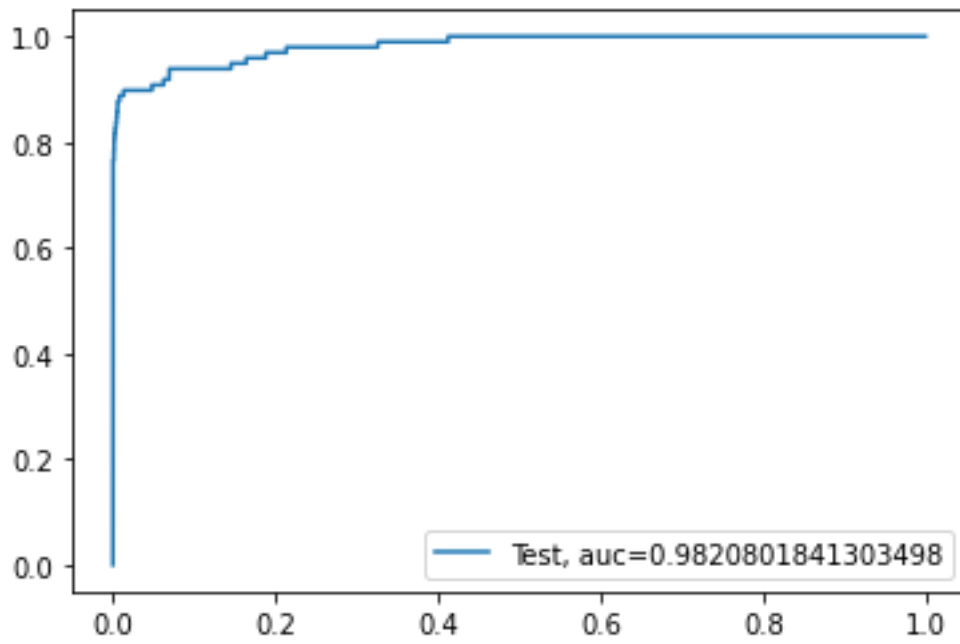
**Fig 6.13: Confusion Matrix of XG Boost classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.29      | 0.83   | 0.43     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.65      | 0.91   | 0.72     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

XG boost roc value: 0.9820801841303498

XG Boost threshold: 0.21093320846557617

ROC for the test dataset 98.2%



**Fig 6.14: ROC curve of XG Boost**

Time Taken by Model: --- 70.41744613647461 seconds ---

-----

#### **6.1.6. Results for Random Oversampling with StratifiedKFold technique:**

|    | Methodology                                 | Model                                      | Accuracy | roc_value | threshold |
|----|---------------------------------------------|--------------------------------------------|----------|-----------|-----------|
| 0  | RepeatedKFold Cross Validation              | Logistic Regression with L2 Regularisation | 0.998964 | 0.969011  | 0.001423  |
| 1  | RepeatedKFold Cross Validation              | Logistic Regression with L1 Regularisation | 0.998929 | 0.869289  | 0.047230  |
| 2  | RepeatedKFold Cross Validation              | KNN                                        | 0.999263 | 0.864827  | 0.200000  |
| 3  | RepeatedKFold Cross Validation              | Tree Model with gini criteria              | 0.999245 | 0.874842  | 1.000000  |
| 4  | RepeatedKFold Cross Validation              | Tree Model with entropy criteria           | 0.999087 | 0.874763  | 1.000000  |
| 5  | RepeatedKFold Cross Validation              | Random Forest                              | 0.999491 | 0.932816  | 0.010000  |
| 6  | RepeatedKFold Cross Validation              | XGBoost                                    | 0.999438 | 0.970107  | 0.001447  |
| 7  | RepeatedKFold Cross Validation              | SVM                                        | 0.998086 | 0.421606  | 0.002741  |
| 8  | StratifiedKFold Cross Validation            | Logistic Regression with L2 Regularisation | 0.998771 | 0.983329  | 0.001572  |
| 9  | StratifiedKFold Cross Validation            | Logistic Regression with L1 Regularisation | 0.998754 | 0.889076  | 0.021087  |
| 10 | StratifiedKFold Cross Validation            | KNN                                        | 0.999192 | 0.805746  | 0.200000  |
| 11 | StratifiedKFold Cross Validation            | Tree Model with gini criteria              | 0.998841 | 0.826249  | 1.000000  |
| 12 | StratifiedKFold Cross Validation            | Tree Model with entropy criteria           | 0.999017 | 0.821244  | 1.000000  |
| 13 | StratifiedKFold Cross Validation            | Random Forest                              | 0.999438 | 0.946472  | 0.010000  |
| 14 | StratifiedKFold Cross Validation            | XGBoost                                    | 0.999386 | 0.978148  | 0.002443  |
| 15 | StratifiedKFold Cross Validation            | SVM                                        | 0.998280 | 0.598230  | 0.001694  |
| 16 | Random Oversampling with StratifiedKFold CV | Logistic Regression with L2 Regularisation | 0.985990 | 0.980763  | 0.434510  |
| 17 | Random Oversampling with StratifiedKFold CV | Logistic Regression with L1 Regularisation | 0.998280 | 0.500000  | 1.500000  |
| 18 | Random Oversampling with StratifiedKFold CV | KNN                                        | 0.998069 | 0.805604  | 0.200000  |
| 19 | Random Oversampling with StratifiedKFold CV | Tree Model with gini criteria              | 0.998947 | 0.816115  | 1.000000  |
| 20 | Random Oversampling with StratifiedKFold CV | Tree Model with entropy criteria           | 0.999263 | 0.846833  | 1.000000  |
| 21 | Random Oversampling with StratifiedKFold CV | Random Forest                              | 0.999491 | 0.951102  | 0.010000  |
| 22 | Random Oversampling with StratifiedKFold CV | XGBoost                                    | 0.996296 | 0.982080  | 0.210933  |

---

**Fig 6.15: Results for cross validation with Stratified K Fold**

Looking at the Accuracy and ROC value we have XG Boost which has provided best results for Random Oversampling and Stratified K Fold technique.

**6.2. Oversampling with SMOTE Oversampling:**

We will use SMOTE Oversampling method to handle the class imbalance

**6.2.1. Logistic Regression with L1 And L2 Regularisation:**

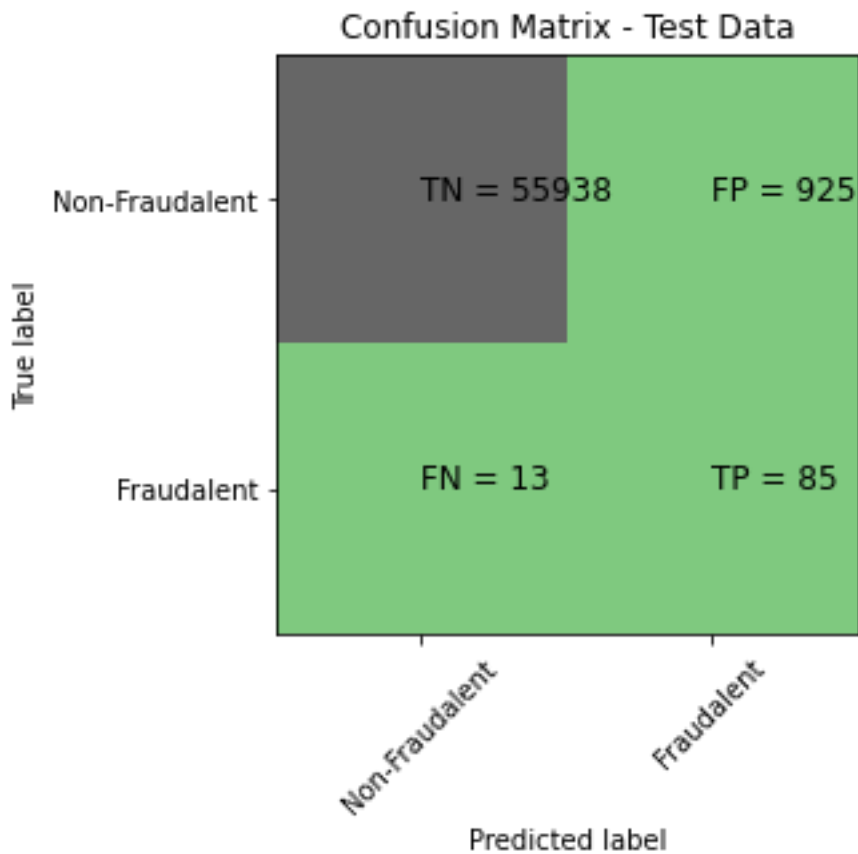
Max auc roc for l1: 0.5

Max auc roc for l2: 0.9930150238550091

**6.2.1.1. Logistic Regression with L2 Regularisation:**

Accuracy of Logistic model with l2 regularisation: 0.9835325924755535

**Confusion Matrix:**



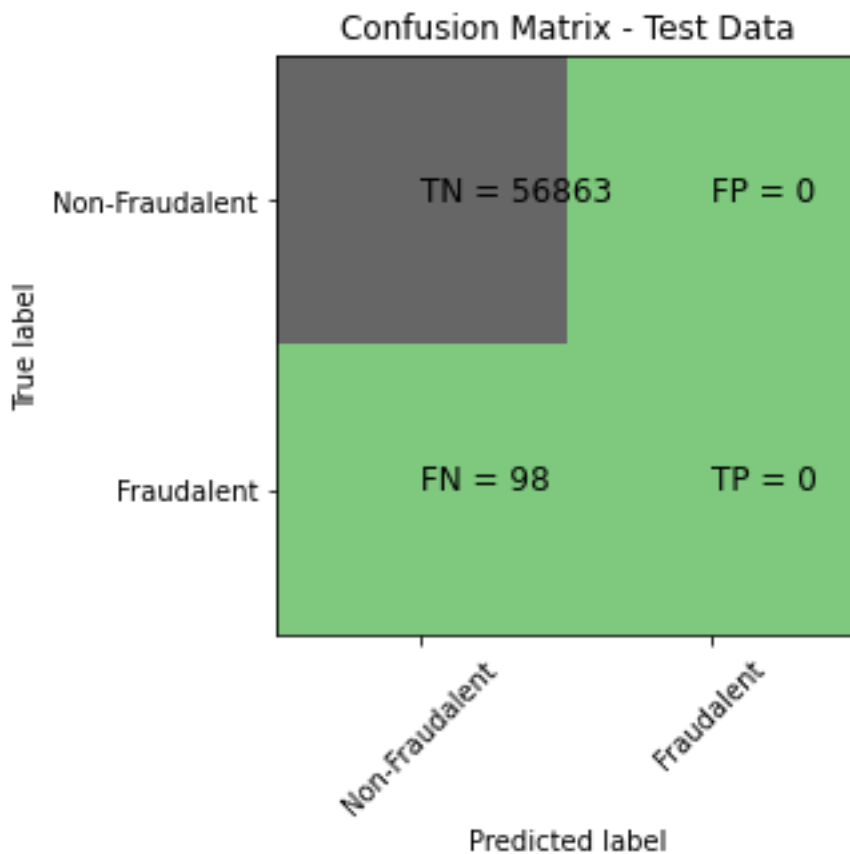
**Fig 6.16: Confusion Matrix of Logistic Regression with L2 Regularisation classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 0.98   | 0.99     | 56863   |
| 1                | 0.08      | 0.87   | 0.15     | 98      |
| accuracy         |           |        | 0.98     | 56961   |
| macro average    | 0.54      | 0.93   | 0.57     | 56961   |
| Weighted average | 1.00      | 0.98   | 0.99     | 56961   |

#### 6.2.1.2. Logistic Regression with L1 Regularisation:

Accuracy of Logistic model with l1 regularisation: 0.9982795245869981

### Confusion Matrix:



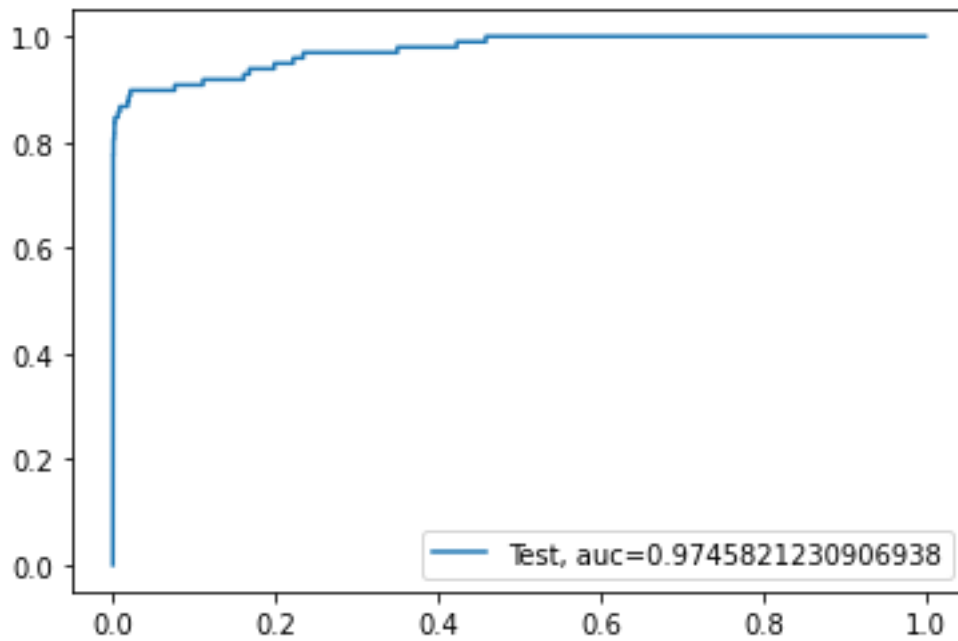
**Fig 6.17: Confusion Matrix of Logistic Regression with L2 Regularisation**  
**classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.00      | 0.00   | 0.00     | 98      |
| accuracy         |           |        | 0.98     | 56961   |
| macro average    | 0.50      | 0.50   | 0.50     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

l2 roc value: 0.9745821230906938

l2 threshold: 0.3940771805735311

ROC for the test dataset 97.5%

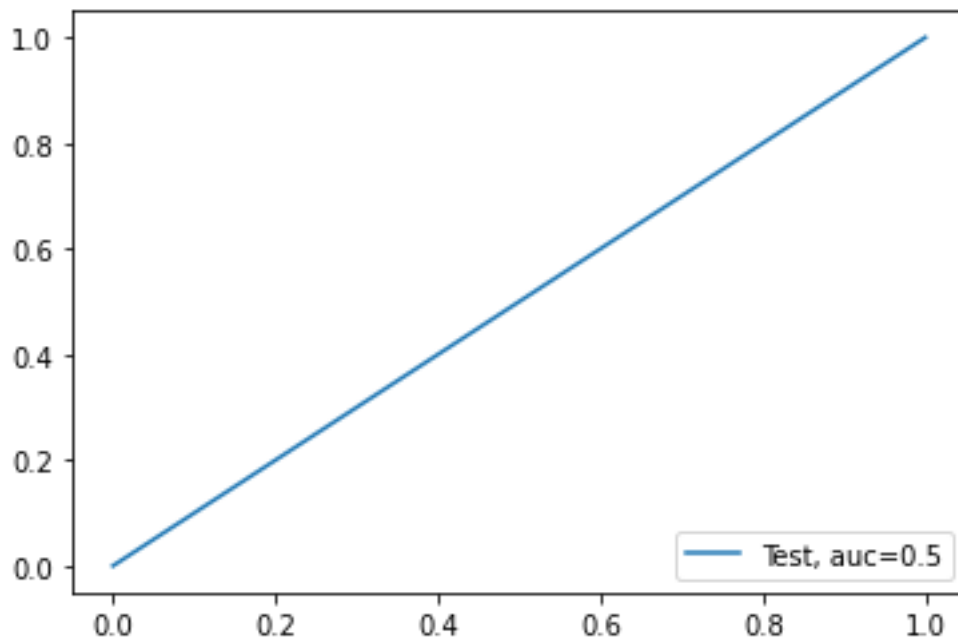


**Fig 6.18: Confusion Matrix of Logistic Regression with L1 Regularisation**

l1 roc value: 0.5

l1 threshold: 1.5

ROC for the test dataset 50.0%



**Fig 6.19: Confusion Matrix of Logistic Regression with L2 Regularisation**

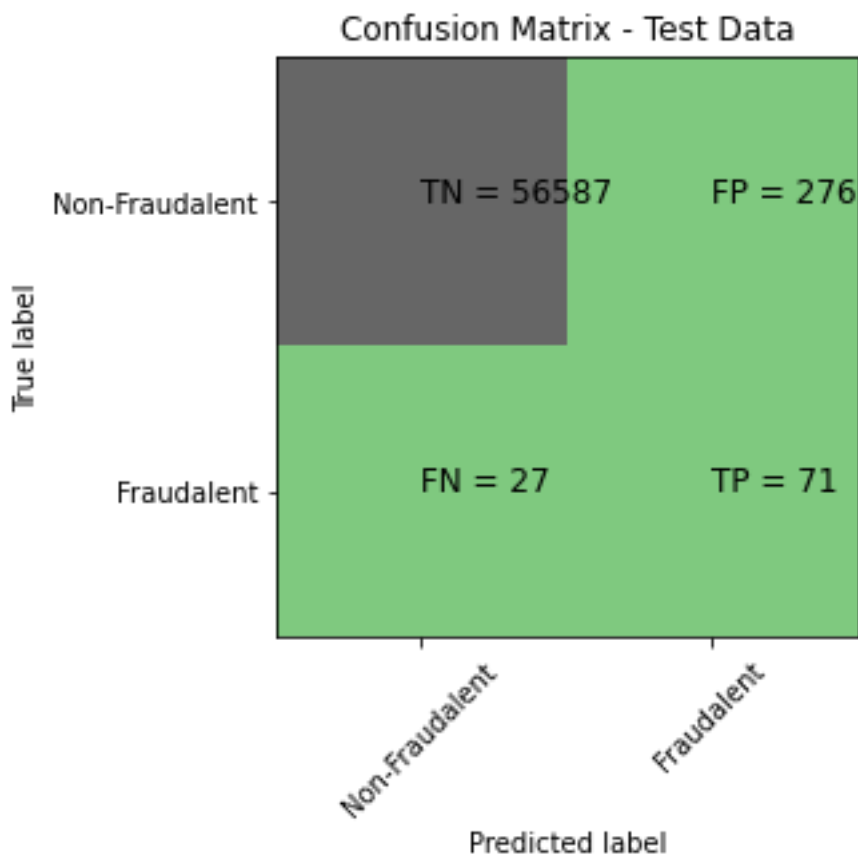
Time Taken by Model: --- 567.4116318225861 seconds ---

-----

### 6.2.2. KNN Model:

model score : 0.9946805709169432

**Confusion Matrix:**



**Fig 6.20: Confusion Matrix of KNN Model**

**classification Report:**

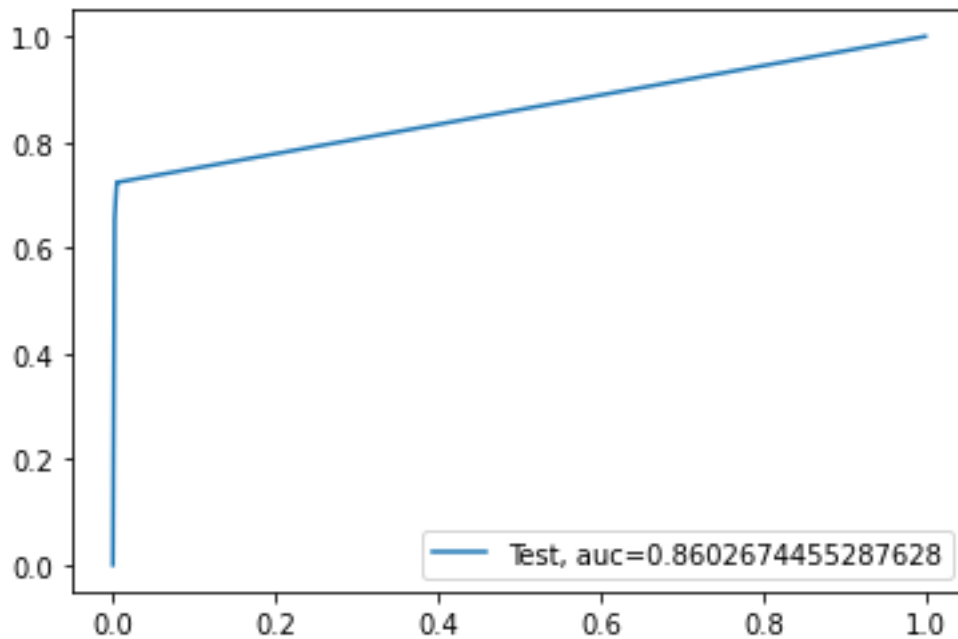
|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.20      | 0.72   | 0.32     | 98      |
| accuracy         |           |        | 0.99     | 56961   |
| macro average    | 0.60      | 0.86   | 0.66     | 56961   |
| Weighted average | 1.00      | 0.99   | 1.00     | 56961   |

KNN roc value: 0.8602674455287628

---

KNN threshold: 0.6

ROC for the test dataset 86.0%



**Fig 6.21: ROC of KNN Model**

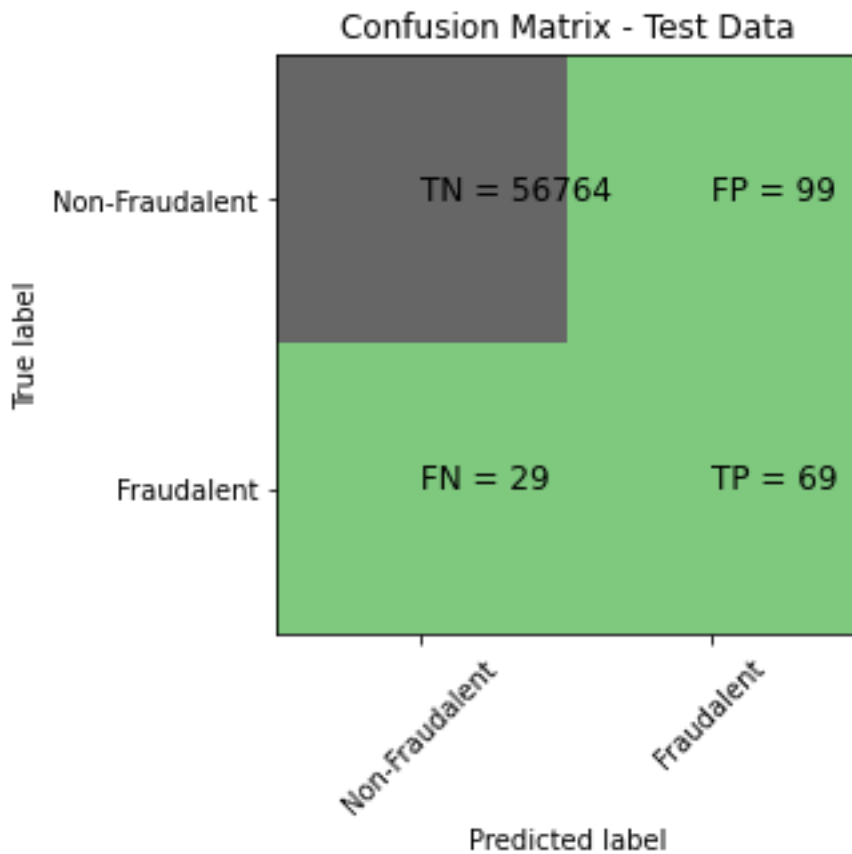
Time Taken by Model: --- 3060.9006798267365 seconds ---

### 6.2.3. Decision Tree Models with 'Gini' & 'Entropy' criteria:

#### 6.2.3.1 Gini:

gini score: 0.9977528484401608

**Confusion Matrix:**



**Fig 6.22: Confusion Matrix of Decision Tree classification Report:**

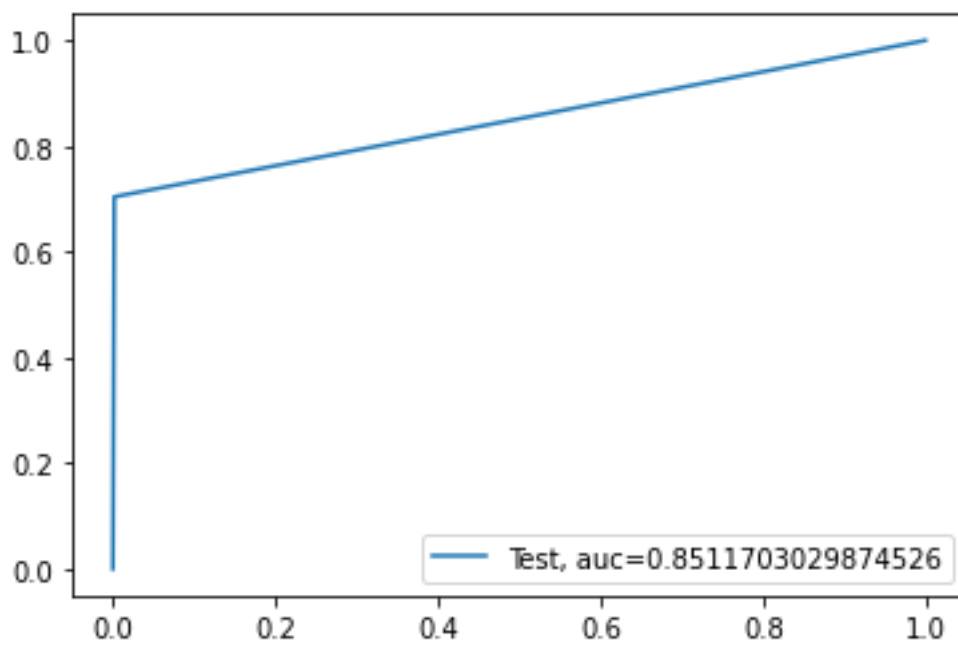
|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.41      | 0.70   | 0.52     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.71      | 0.85   | 0.76     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

### 6.2.3.1 Gini:

gini tree roc value: 0.8511703029874526

Tree threshold: 1.0

ROC for the test dataset 85.1%

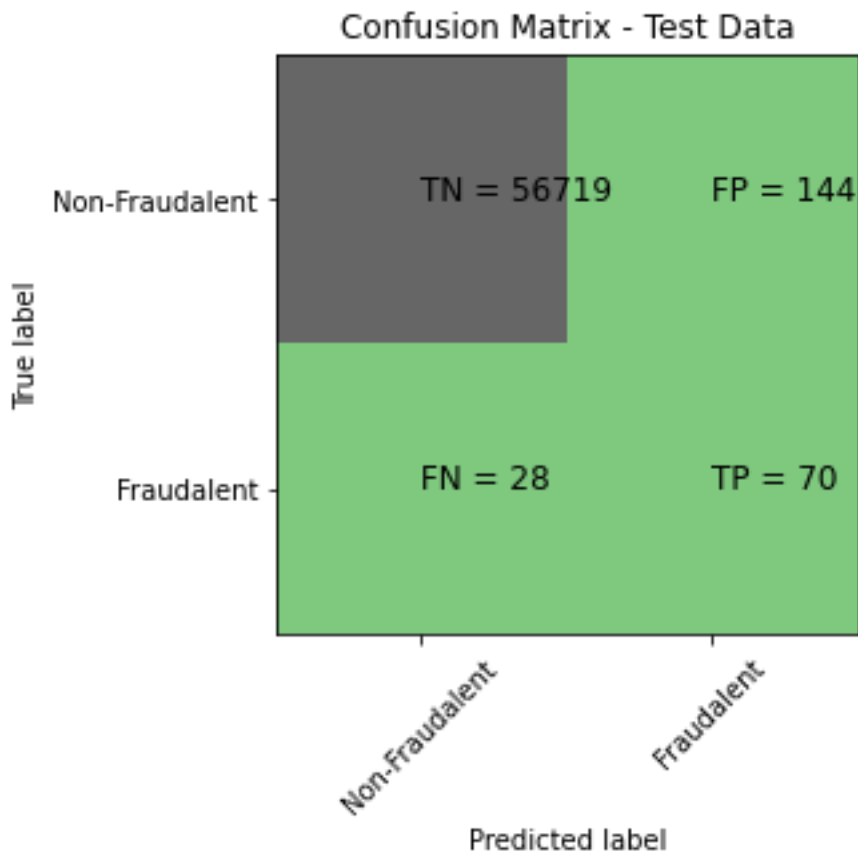


**Fig 6.23: ROC curve of Decision Tree**

#### **6.2.3.2 Entropy:**

entropy score: 0.9969803900914661

**Confusion Matrix:**



**Fig 6.24: Confusion Matrix of Decision Tree**

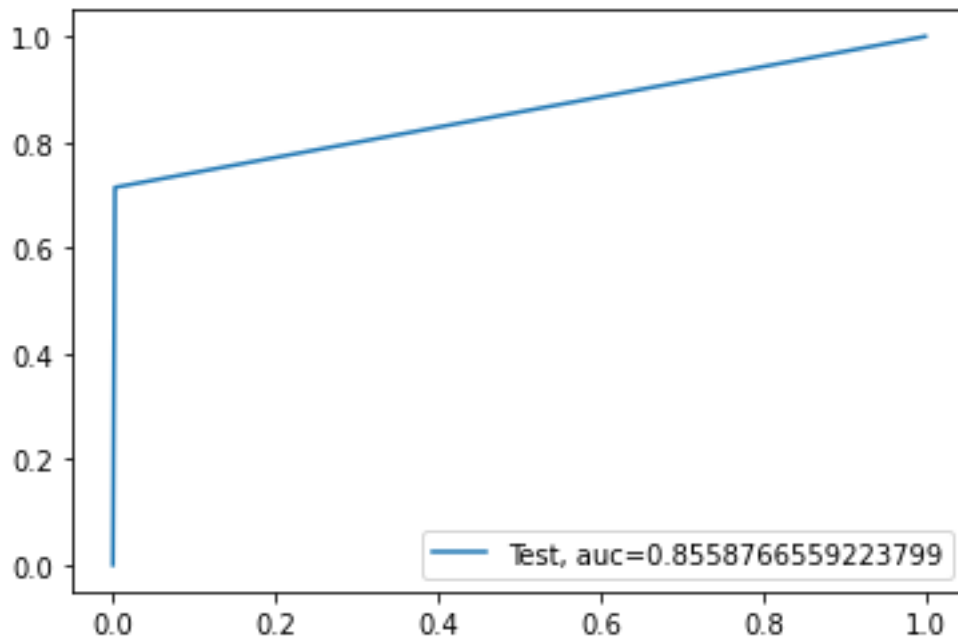
**classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.33      | 0.71   | 0.45     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.66      | 0.86   | 0.72     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

entropy tree roc value: 0.8558766559223799

Tree threshold: 1.0

ROC for the test dataset 85.6%



**Fig 6.25: ROC curve of Decision Tree**

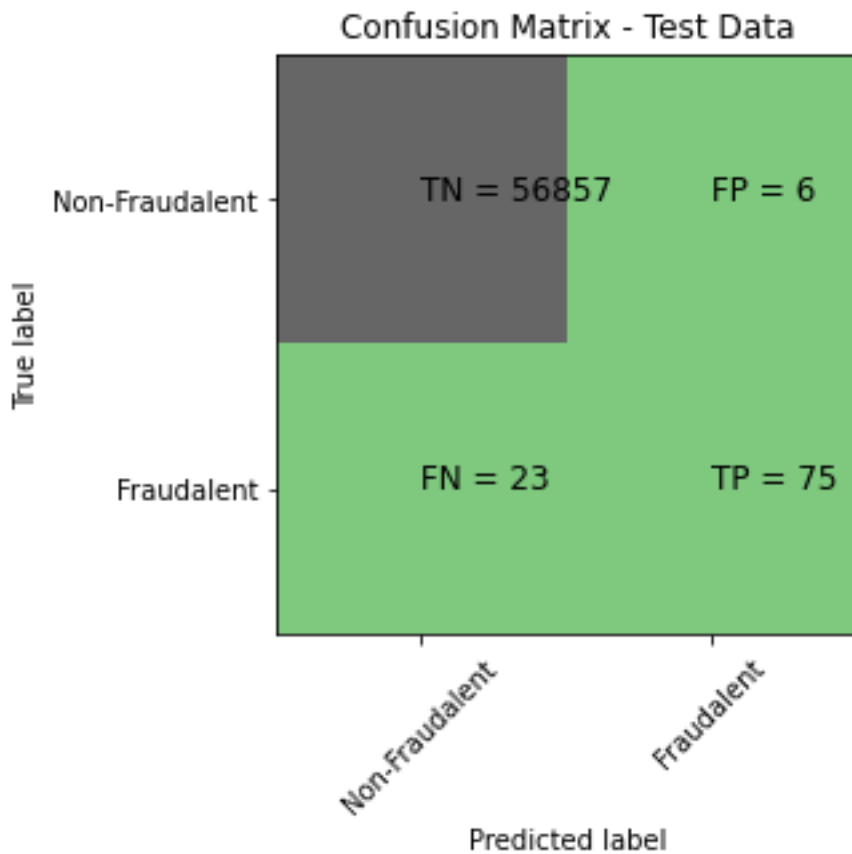
Time Taken by Model: --- 82.8795850276947 seconds ---

-----

#### **6.2.4. Random Forest Model:**

Model Accuracy: 0.999490879724724

**Confusion Matrix:**



**Fig 6.26: Confusion Matrix of Random Forest Model**

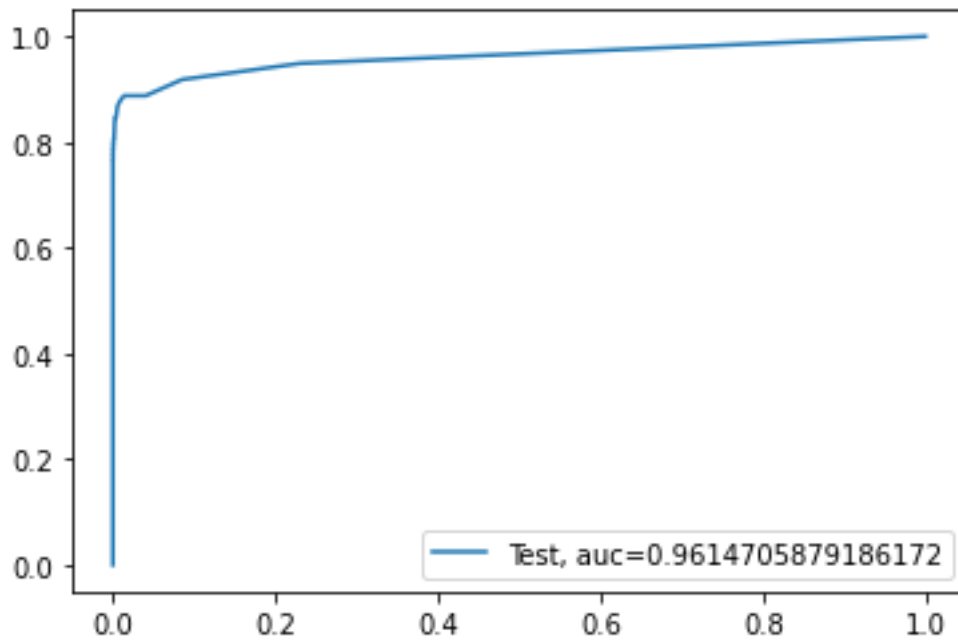
**classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.93      | 0.77   | 0.84     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.96      | 0.88   | 0.92     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

Random Forest roc value: 0.9614705879186172

Random Forest threshold: 0.05

ROC for the test dataset 96.1%



**Fig 6.27: ROC curve of Random Forest model**

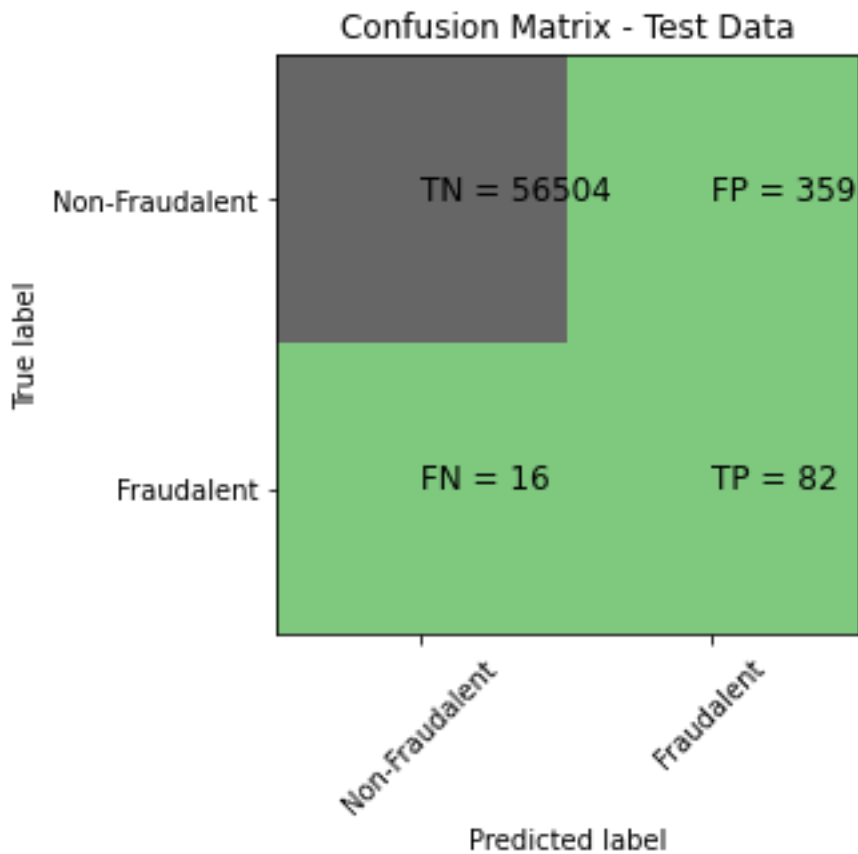
Time Taken by Model: --- 383.57139778137207 seconds ---

-----

#### **6.2.5. XG Boost Model:**

Model Accuracy: 0.9934165481645336

**Confusion Matrix:**



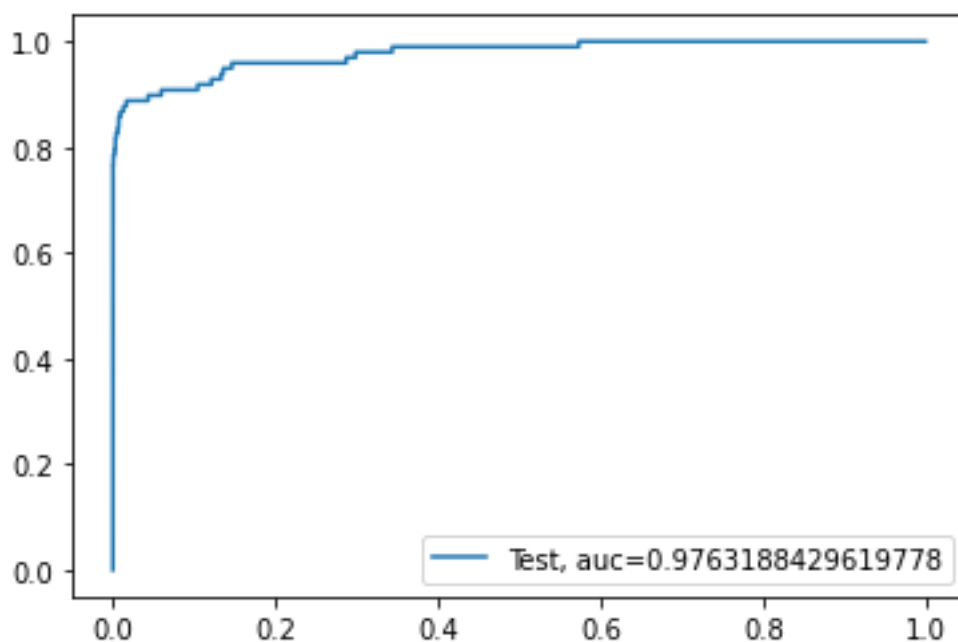
**Fig 6.28: Confusion Matrix of XG Boost classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 0.99   | 1.00     | 56863   |
| 1                | 0.19      | 0.84   | 0.30     | 98      |
| accuracy         |           |        | 0.99     | 56961   |
| macro average    | 0.59      | 0.92   | 0.65     | 56961   |
| Weighted average | 1.00      | 0.99   | 1.00     | 56961   |

XG boost roc value: 0.9763188429619778

XG Boost threshold: 0.27664539217948914

ROC for the test dataset 97.6%



**Fig 6.29: ROC curve of XG Boost**

Time Taken by Model: --- 117.78114032745361 seconds ---

### 6.2.6. Results for SMOTE Oversampling with StratifiedKFold:

|    |                                             |                                            |          |          |          |
|----|---------------------------------------------|--------------------------------------------|----------|----------|----------|
| 5  | RepeatedKFold Cross Validation              | Random Forest                              | 0.999491 | 0.932816 | 0.010000 |
| 6  | RepeatedKFold Cross Validation              | XGBoost                                    | 0.999438 | 0.970107 | 0.001447 |
| 7  | RepeatedKFold Cross Validation              | SVM                                        | 0.998086 | 0.421606 | 0.002741 |
| 8  | StratifiedKFold Cross Validation            | Logistic Regression with L2 Regularisation | 0.998771 | 0.983329 | 0.001572 |
| 9  | StratifiedKFold Cross Validation            | Logistic Regression with L1 Regularisation | 0.998754 | 0.889076 | 0.021087 |
| 10 | StratifiedKFold Cross Validation            | KNN                                        | 0.999192 | 0.805746 | 0.200000 |
| 11 | StratifiedKFold Cross Validation            | Tree Model with gini criteria              | 0.998841 | 0.826249 | 1.000000 |
| 12 | StratifiedKFold Cross Validation            | Tree Model with entropy criteria           | 0.999017 | 0.821244 | 1.000000 |
| 13 | StratifiedKFold Cross Validation            | Random Forest                              | 0.999438 | 0.946472 | 0.010000 |
| 14 | StratifiedKFold Cross Validation            | XGBoost                                    | 0.999386 | 0.978148 | 0.002443 |
| 15 | StratifiedKFold Cross Validation            | SVM                                        | 0.998280 | 0.598230 | 0.001694 |
| 16 | Random Oversampling with StratifiedKFold CV | Logistic Regression with L2 Regularisation | 0.985990 | 0.980763 | 0.434510 |
| 17 | Random Oversampling with StratifiedKFold CV | Logistic Regression with L1 Regularisation | 0.998280 | 0.500000 | 1.500000 |
| 18 | Random Oversampling with StratifiedKFold CV | KNN                                        | 0.998069 | 0.805604 | 0.200000 |
| 19 | Random Oversampling with StratifiedKFold CV | Tree Model with gini criteria              | 0.998947 | 0.816115 | 1.000000 |
| 20 | Random Oversampling with StratifiedKFold CV | Tree Model with entropy criteria           | 0.999263 | 0.846833 | 1.000000 |
| 21 | Random Oversampling with StratifiedKFold CV | Random Forest                              | 0.999491 | 0.951102 | 0.010000 |
| 22 | Random Oversampling with StratifiedKFold CV | XGBoost                                    | 0.996296 | 0.982080 | 0.210933 |
| 23 | SMOTE Oversampling with StratifiedKFold CV  | Logistic Regression with L2 Regularisation | 0.983533 | 0.974582 | 0.394077 |
| 24 | SMOTE Oversampling with StratifiedKFold CV  | Logistic Regression with L1 Regularisation | 0.998280 | 0.500000 | 1.500000 |
| 25 | SMOTE Oversampling with StratifiedKFold CV  | KNN                                        | 0.994681 | 0.860267 | 0.600000 |
| 26 | SMOTE Oversampling with StratifiedKFold CV  | Tree Model with gini criteria              | 0.997753 | 0.851170 | 1.000000 |
| 27 | SMOTE Oversampling with StratifiedKFold CV  | Tree Model with entropy criteria           | 0.996980 | 0.855877 | 1.000000 |
| 28 | SMOTE Oversampling with StratifiedKFold CV  | Random Forest                              | 0.999491 | 0.961471 | 0.050000 |
| 29 | SMOTE Oversampling with StratifiedKFold CV  | XGBoost                                    | 0.993417 | 0.976319 | 0.276645 |

---

### **Fig 6.29: Results for SMOTE Oversampling with Stratified K Fold**

Looking at Accuracy and ROC value we have XG Boost which has provided best results for SMOTE Oversampling with Stratified K Fold technique.

#### **6.3 Over sampling with ADASYN over sampling:**

We will use ADASYN over sampling to handle the class imbalance.

##### **6.3.1. Logistic Regression with L1 And L2 Regularisation:**

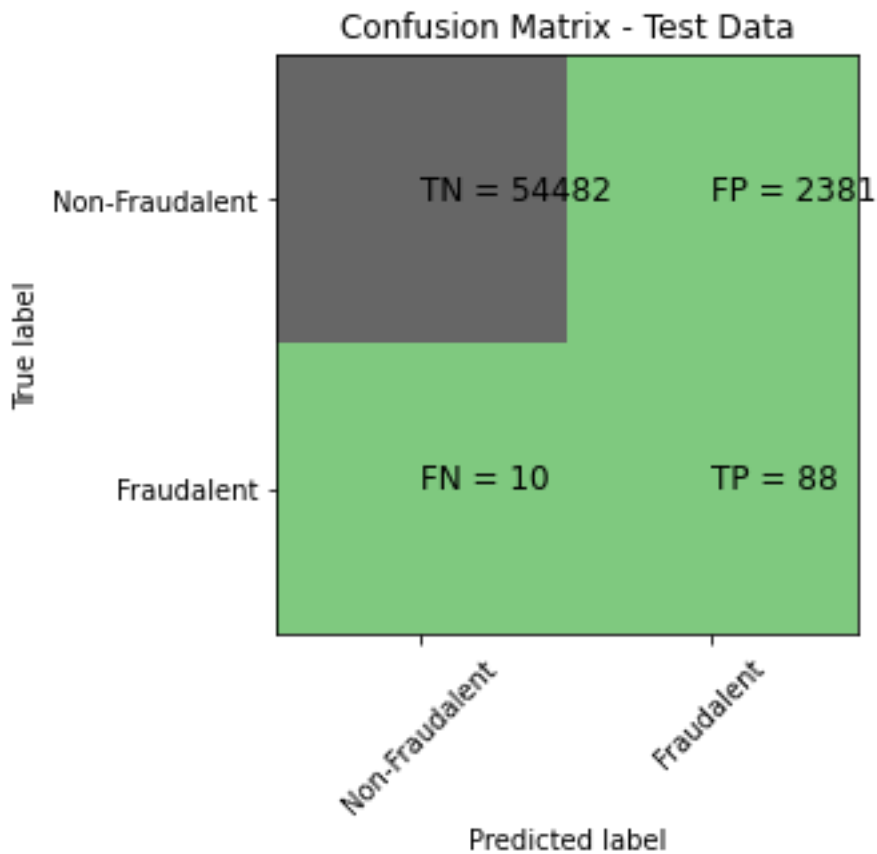
Max auc roc for l1: 0.5

Max auc roc for l2: 0.9835151915234407

##### **6.3.1.1 Logistic Regression with L2 Regularisation:**

Accuracy of Logistic model with l2 regularisation: 0.9580239110970664

#### **Confusion Matrix:**



**Fig 6.30: Confusion Matrix of Logistic Regression with L2 Regularisation classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 0.96   | 0.98     | 56863   |
| 1                | 0.04      | 0.90   | 0.07     | 98      |
| accuracy         |           |        | 0.99     | 56961   |
| macro average    | 0.52      | 0.93   | 0.52     | 56961   |
| Weighted average | 1.00      | 0.96   | 0.98     | 56961   |

### 6.3.1.2 Logistic Regression with L2 Regularisation:

Accuracy of Logistic model with l1 regularisation: 0.9982795245869981

### Confusion Matrix:

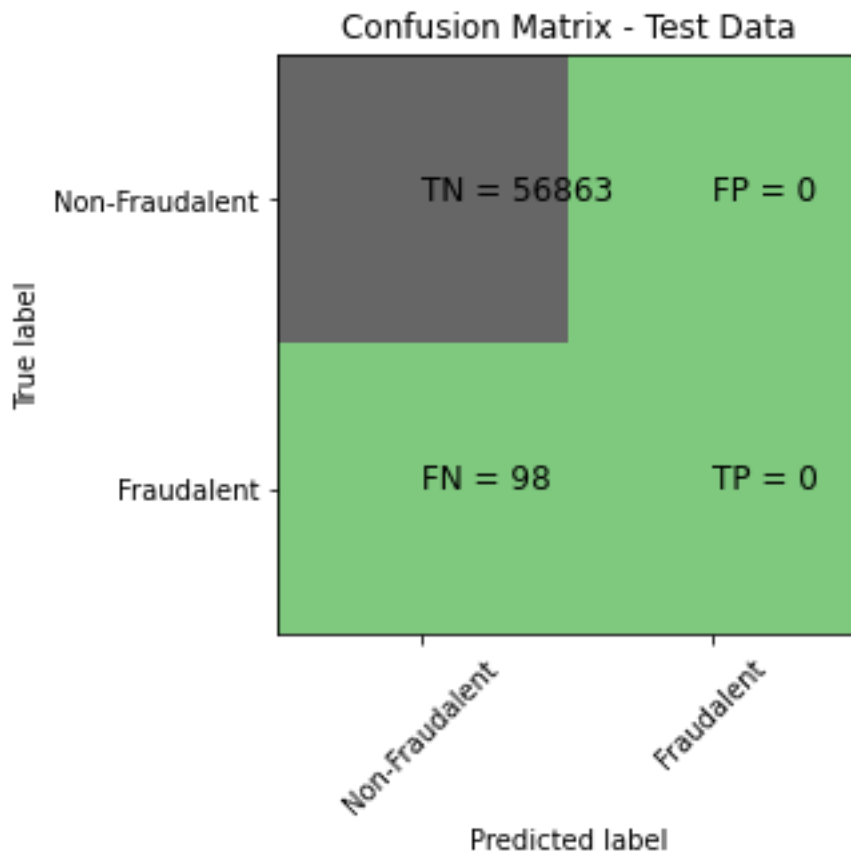


Fig 6.31: Confusion Matrix of Logistic Regression with L1 Regularisation

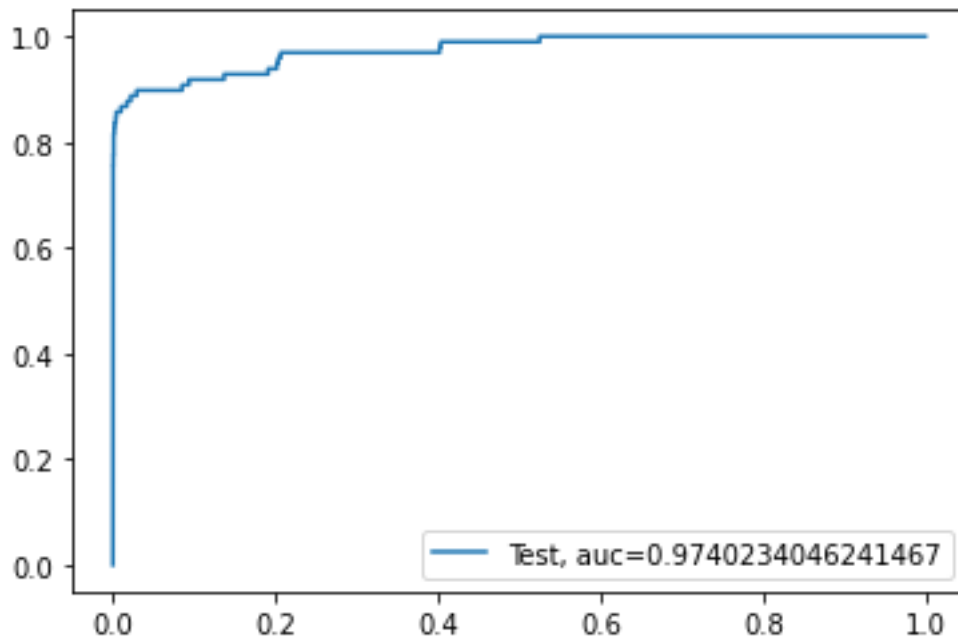
### classification Report:

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.00      | 0.00   | 0.00     | 98      |
| accuracy         |           |        | 0.99     | 56961   |
| macro average    | 0.50      | 0.50   | 0.50     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

l2 roc value : 0.9740234046241467

l2 threshold: 0.6115977022466358

ROC for the test dataset 97.4%

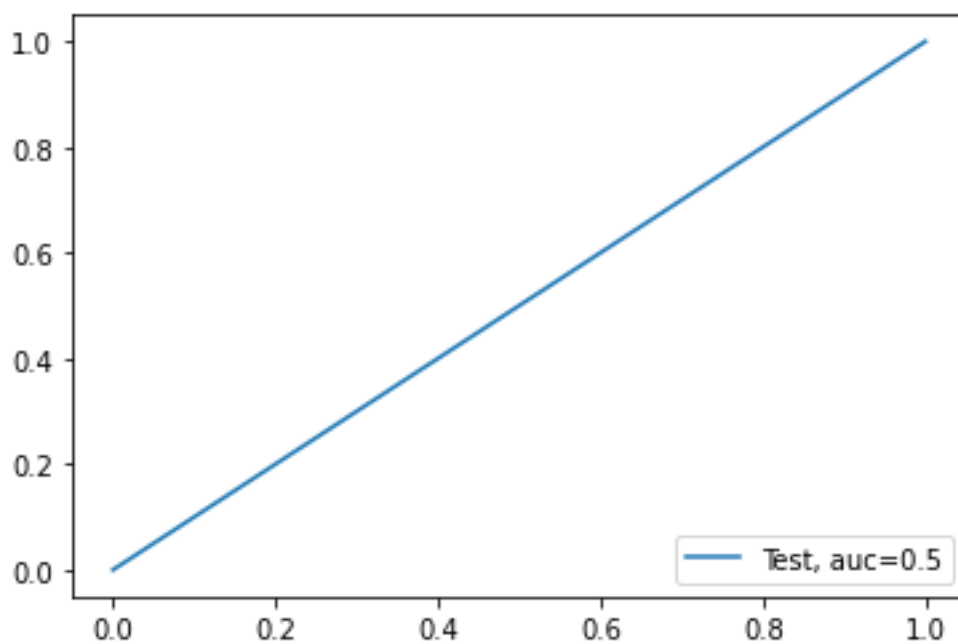


**Fig 6.32: Confusion Matrix of Logistic Regression with L1 Regularisation**

l1 roc value: 0.5

l1 threshold: 1.5

ROC for the test dataset 50.0%



**Fig 6.33: Confusion Matrix of Logistic Regression with L2 Regularisation**

Time Taken by Model: --- 648.8162662982941 seconds ---

### 6.3.2. KNN Model:

model score: 0.9945050122013307

#### Confusion Matrix:

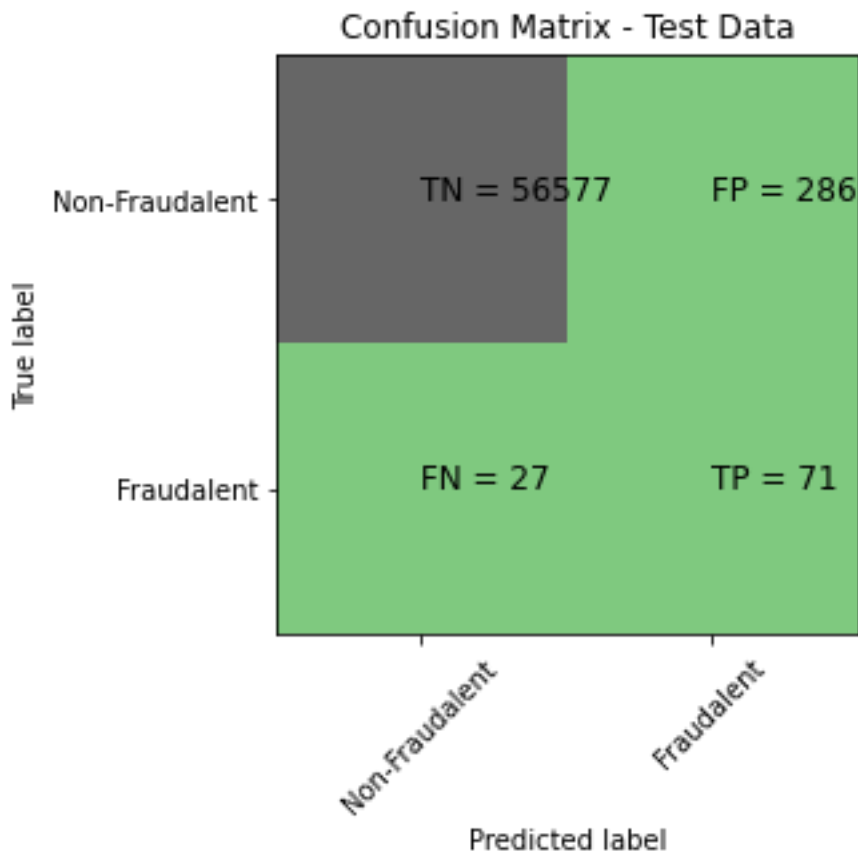


Fig 6.34: ROC of KNN Model

#### classification Report:

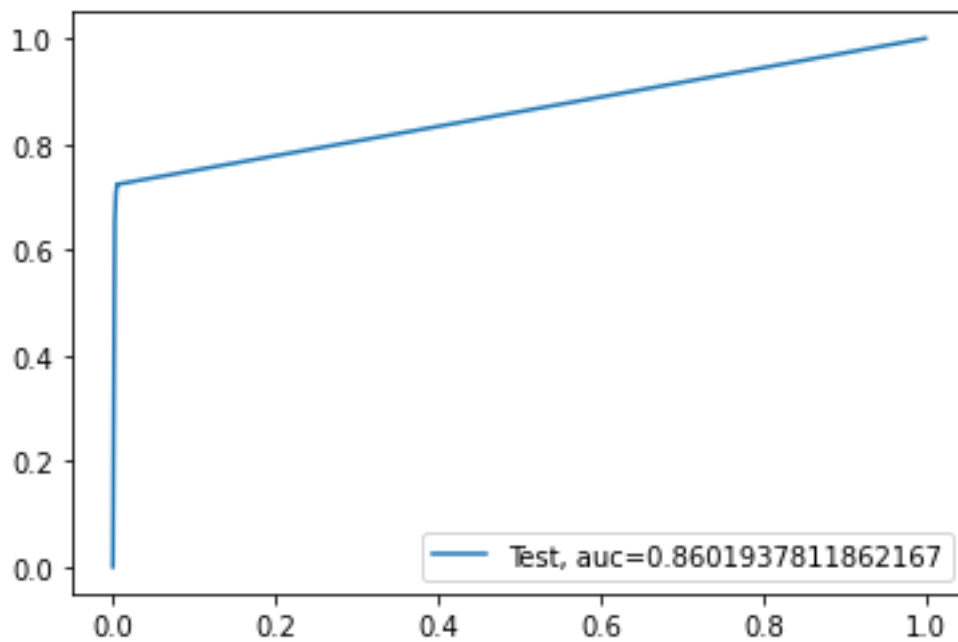
|               | Precision | Recall | F1-score | Support |
|---------------|-----------|--------|----------|---------|
| 0             | 1.00      | 0.99   | 1.00     | 56863   |
| 1             | 0.20      | 0.72   | 0.31     | 98      |
| accuracy      |           |        | 0.99     | 56961   |
| macro average | 0.60      | 0.86   | 0.65     | 56961   |
| Weighted      | 1.00      | 0.99   | 1.00     | 56961   |

|         |  |  |  |  |
|---------|--|--|--|--|
| average |  |  |  |  |
|---------|--|--|--|--|

KNN roc value: 0.8601937811862167

KNN threshold: 0.6

ROC for the test dataset 86.0%



**Fig 6.35: ROC of KNN Model**

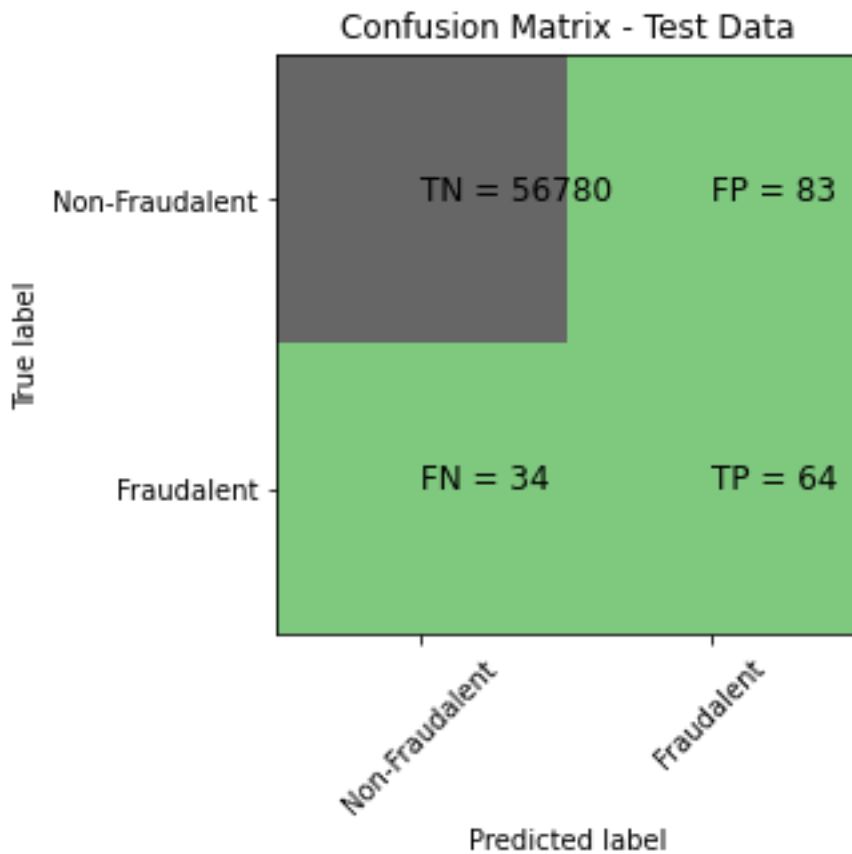
Time Taken by Model: --- 2780.0399057865143 seconds ---

### 6.3.3 Decision Tree Models with 'gini' & 'entropy' criteria:

#### 6.3.3.1 Gini:

gini score: 0.9979459630273345

**Confusion Matrix:**



**Fig 6.36: Confusion Matrix of Decision Tree**

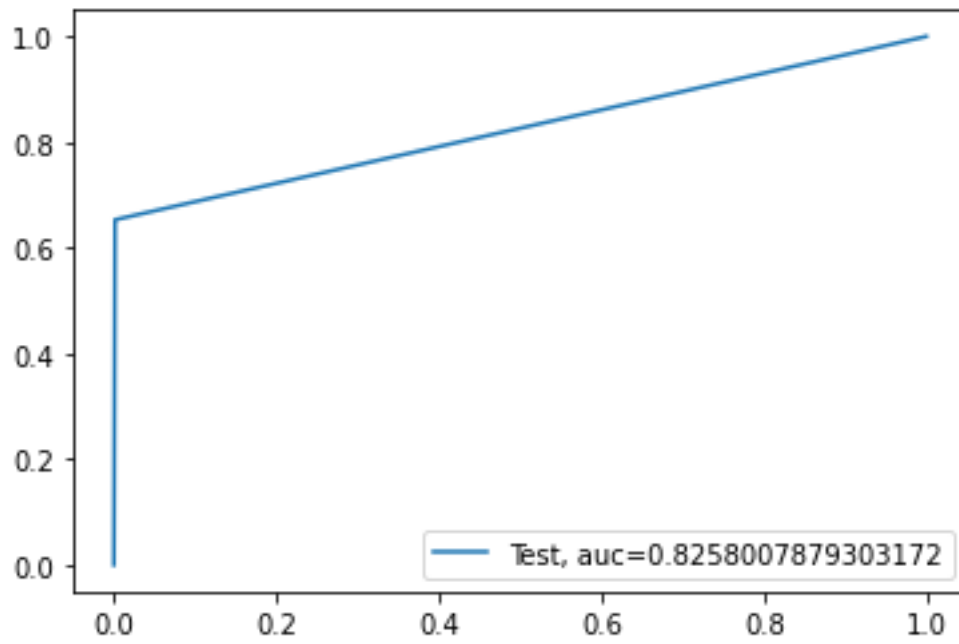
**classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.44      | 0.65   | 0.52     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.7       | 0.83   | 0.76     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

gini tree roc value: 0.8258007879303172

Tree threshold: 1.0

ROC for the test dataset 82.6%

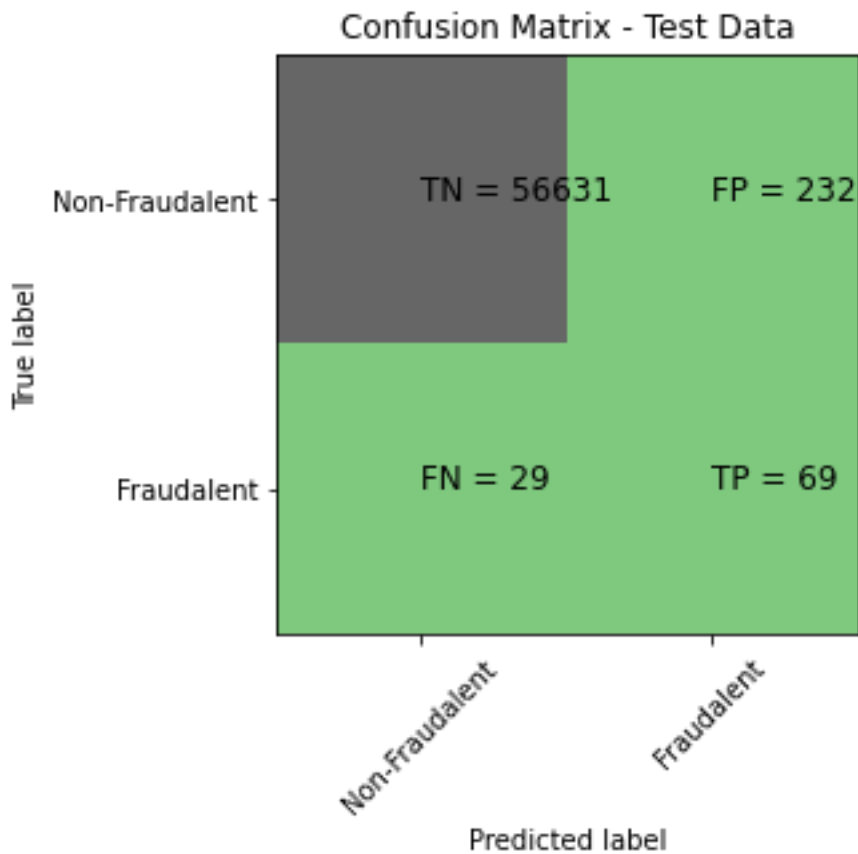


**Fig 6.37: ROC curve of Decision Tree**

#### **6.3.3.2 Entropy:**

entropy score: 0.9954179175225154

**Confusion Matrix:**



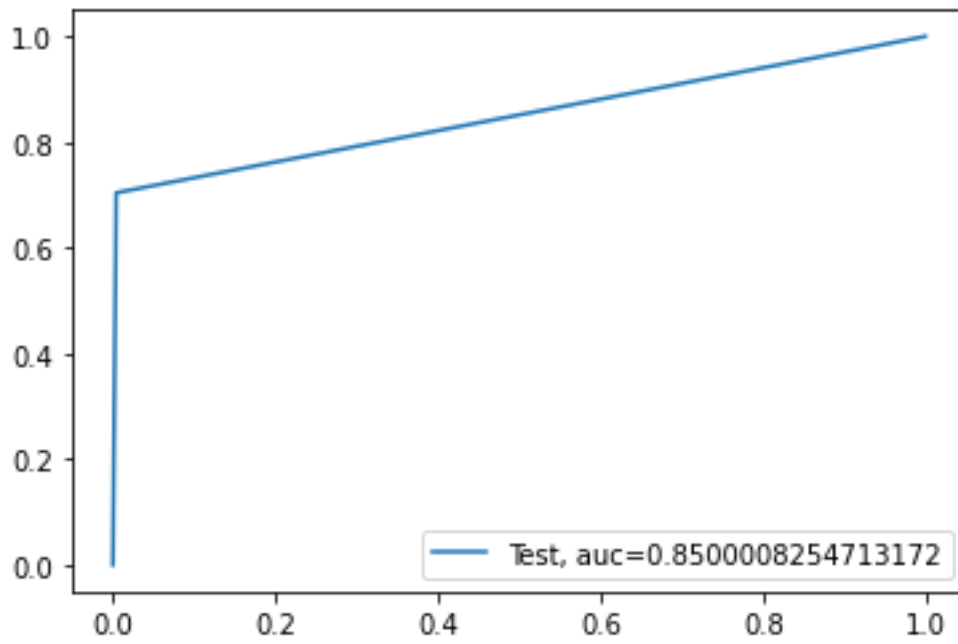
**Fig 6.38: Confusion Matrix of Decision Tree classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.23      | 0.70   | 0.35     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.61      | 0.85   | 0.67     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

entropy tree roc value: 0.8500008254713172

Tree threshold: 1.0

ROC for the test dataset 85.0%



**Fig 6.39: ROC curve of Decision Tree**

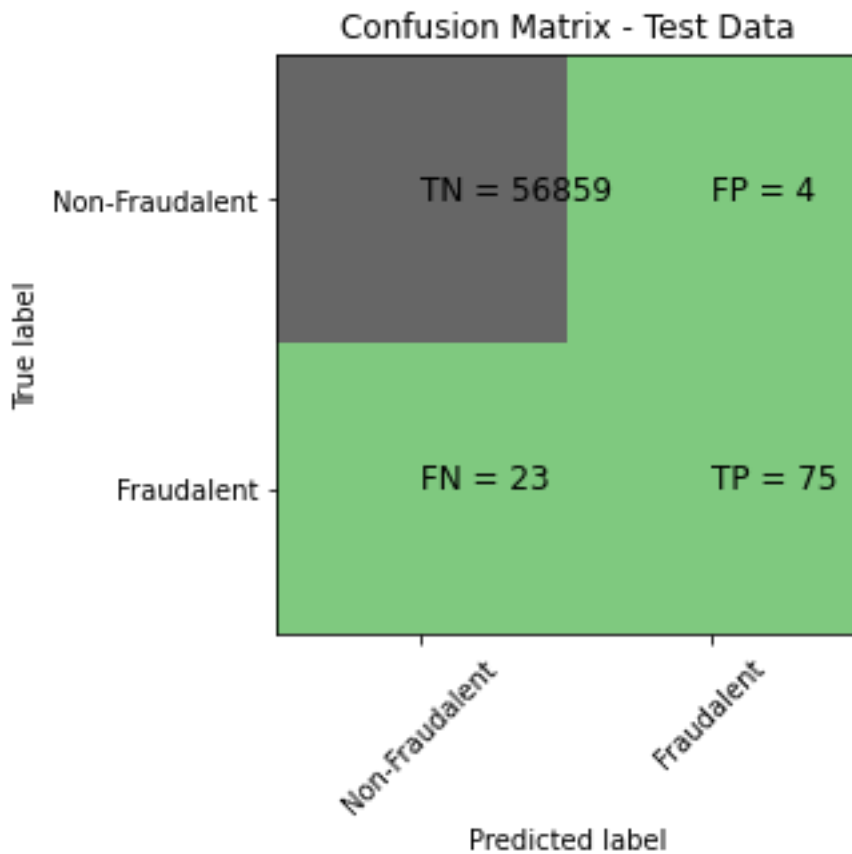
Time Taken by Model: --- 83.75130772590637 seconds ---

---

#### **6.3.4 Random Forest Model:**

Model Accuracy: 0.9995259914678464

**Confusion Matrix:**



**Fig 6.40: Confusion Matrix of Random Forest Model**

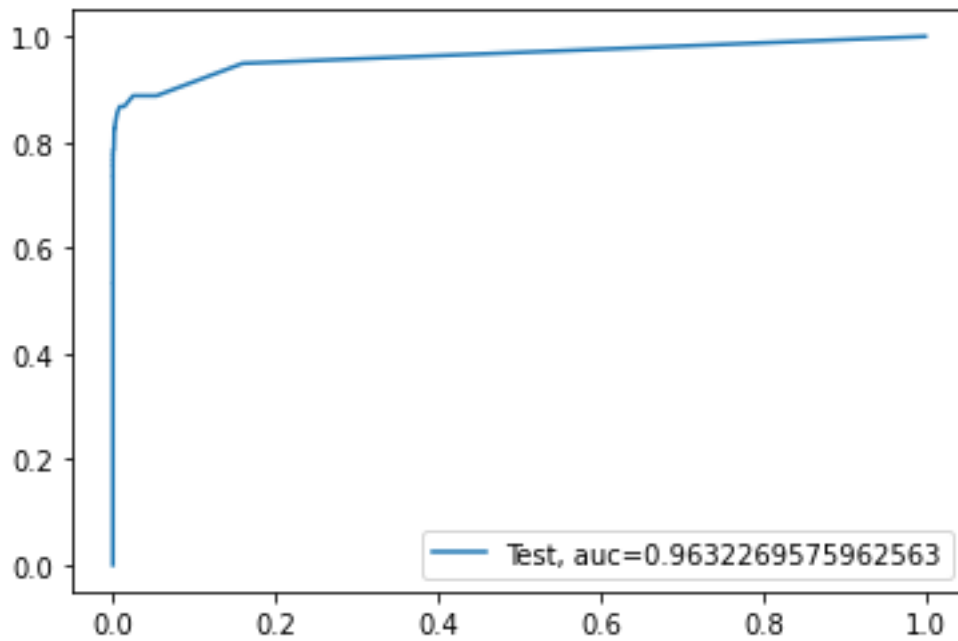
**classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 1.00   | 1.00     | 56863   |
| 1                | 0.95      | 0.7    | 0.85     | 98      |
| accuracy         |           |        | 1.00     | 56961   |
| macro average    | 0.97      | 0.88   | 0.92     | 56961   |
| Weighted average | 1.00      | 1.00   | 1.00     | 56961   |

Random Forest roc value: 0.9632269575962563

Random Forest threshold: 0.03

ROC for the test dataset 96.3%



**Fig 6.41: ROC curve of Random Forest model**

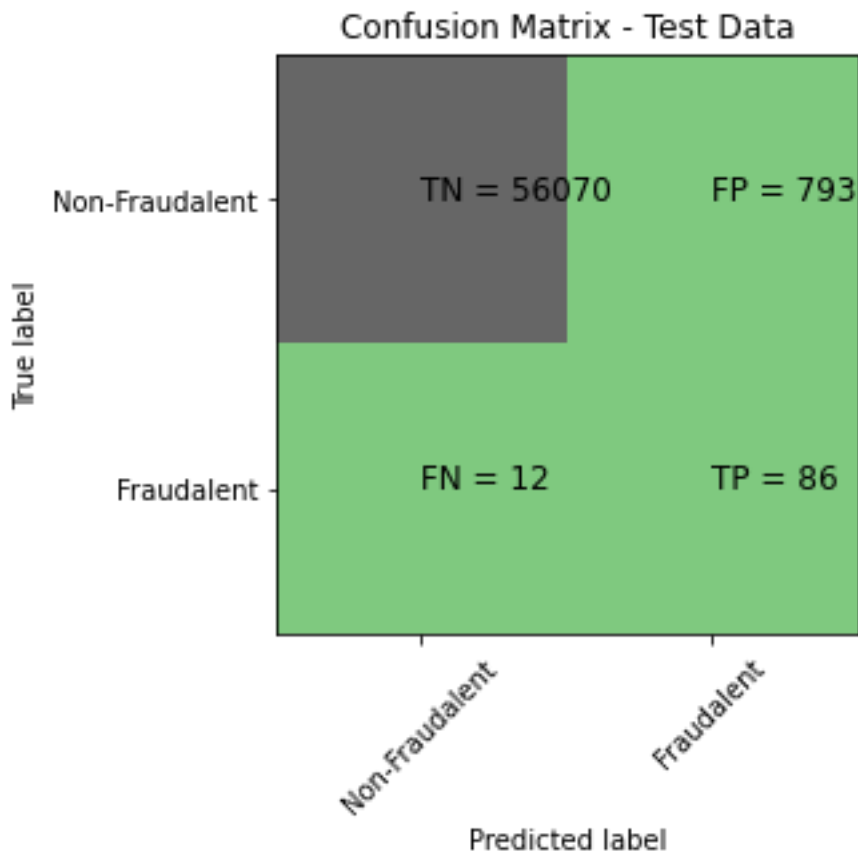
Time Taken by Model: --- 398.62654280662537 seconds ---

-----

### 6.3.5 XG Boost Model:

Model Accuracy: 0.9858675233931988

**Confusion Matrix:**



**Fig 6.42: Confusion Matrix of Random Forest model**

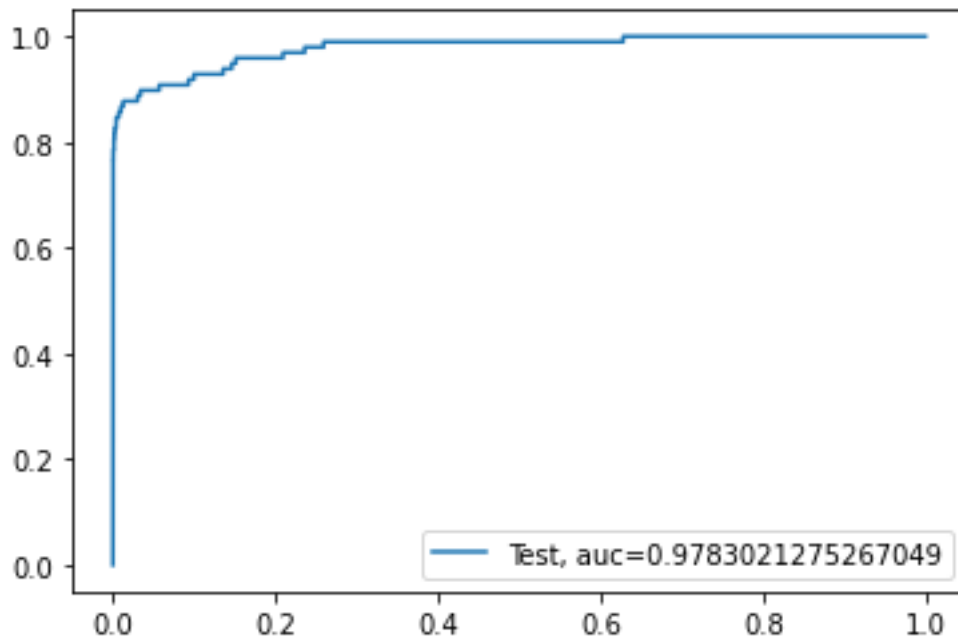
**classification Report:**

|                  | Precision | Recall | F1-score | Support |
|------------------|-----------|--------|----------|---------|
| 0                | 1.00      | 0.99   | 0.99     | 56863   |
| 1                | 0.10      | 0.88   | 0.18     | 98      |
| accuracy         |           |        | 0.99     | 56961   |
| macro average    | 0.55      | 0.93   | 0.5      | 56961   |
| Weighted average | 1.00      | 0.99   | 0.99     | 56961   |

XG boost roc value: 0.9783021275267049

XG Boost threshold: 0.5225471258163452

ROC for the test dataset 97.8%



**Fig 6.43: ROC curve of XG Boost**

Time Taken by Model: --- 119.51173400878906 seconds ---

### 6.3.6 Results for ADASYN Oversampling with StratifiedKFold:

|    |                                             |                                            |          |          |          |
|----|---------------------------------------------|--------------------------------------------|----------|----------|----------|
| 25 | SMOTE Oversampling with StratifiedKFold CV  | KNN                                        | 0.994681 | 0.860267 | 0.600000 |
| 26 | SMOTE Oversampling with StratifiedKFold CV  | Tree Model with gini criteria              | 0.997753 | 0.851170 | 1.000000 |
| 27 | SMOTE Oversampling with StratifiedKFold CV  | Tree Model with entropy criteria           | 0.996980 | 0.855877 | 1.000000 |
| 28 | SMOTE Oversampling with StratifiedKFold CV  | Random Forest                              | 0.999491 | 0.961471 | 0.050000 |
| 29 | SMOTE Oversampling with StratifiedKFold CV  | XGBoost                                    | 0.993417 | 0.976319 | 0.276645 |
| 30 | ADASYN Oversampling with StratifiedKFold CV | Logistic Regression with L2 Regularisation | 0.958024 | 0.974023 | 0.611598 |
| 31 | ADASYN Oversampling with StratifiedKFold CV | Logistic Regression with L1 Regularisation | 0.998280 | 0.500000 | 1.500000 |
| 32 | ADASYN Oversampling with StratifiedKFold CV | KNN                                        | 0.994505 | 0.860194 | 0.600000 |
| 33 | ADASYN Oversampling with StratifiedKFold CV | Tree Model with gini criteria              | 0.997946 | 0.825801 | 1.000000 |
| 34 | ADASYN Oversampling with StratifiedKFold CV | Tree Model with entropy criteria           | 0.995418 | 0.850001 | 1.000000 |
| 35 | ADASYN Oversampling with StratifiedKFold CV | Random Forest                              | 0.999526 | 0.963227 | 0.030000 |
| 36 | ADASYN Oversampling with StratifiedKFold CV | XGBoost                                    | 0.985868 | 0.978302 | 0.522547 |

**Fig 6.44: Results for ADASYN Oversampling with Stratified K Fold**

---

Looking at Accuracy and ROC value we have XG Boost which has provided best results for ADASYN Oversampling with Stratified K Fold technique

### **6.3.7 Overall conclusion after running the models on Oversampled data :**

Looking at above results it seems XGBOOST model with Random Oversampling with Stratified K Fold CV has provided the best results under the category of all oversampling techniques. So we will try to tune the hyperparameters of this model to get best results.

## **7. HPT – XG Boost Regression**

Please note that the hyperparameters found above using Randomized Search CV and the hyperparameters used below in creating the final model might be different, the reason being, I have executed the Randomized Search CV multiple times to find which set of hyperparameters gives the optimum result and finally used the one below which gave me the best performance.

XG boost roc value: 0.9815403079438694

XG Boost threshold: 0.01721232570707798

## **8. Conclusion**

- In the oversample cases, of all the models we build found that the XGBOOST model with Random Oversampling with Stratified K Fold CV gave us the best accuracy and ROC on oversampled data. Post that we performed hyperparameter tuning and got the below metrics:
- XG boost roc value: 0.9815403079438694 XG Boost threshold: 0.01721232570707798
- However, of all the models we created we found Logistic Regression with L2 Regularisation for Stratified K Fold cross validation (without any oversampling or under sampling) gave us the best result.

---

## 9. References

- [Bolton and Hand, 2001] Bolton, R. J. and Hand, D. J. (2001). Unsupervised profiling methods for fraud detection. *Credit Scoring and Credit Control*.
- [Bolton and Hand, 2002] Bolton, R. J. and Hand, D. J. (2002). Statistical fraud detection: a review. *Statistical science*.
- [Bontemps et al., 2016] Bontemps, L., Cao, V. L., McDermott, J., and Le-Khac, N.-A. (2016). Collective anomaly detection based on long short-term memory recurrent neural networks. *International Conference on Future Data and Security Engineering*.
- [Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine learning* 24(2).
- [Breiman, 2001] Breiman, L. (2001). Random forests. *Machine learning* 45(1).
- [Carcillo et al., 2018] Carcillo, F., Pozzolo, A. D., Borgne, Y.-A. L., Caelen, O., Mazzer, Y., and Bontempi, G. (2018). Scarff; a scalable framework for streaming credit card fraud detection with spark. *Information Fusion*.
- [Chandola et al., 2012] Chandola, V., Banerjee, A., and Kumar, V. (2012). Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*.
- [Chawla, 2005] Chawla, N. V. (2005). Data mining for imbalance datasets: an overview. *Data mining and knowledge discovery handbook*.
- [Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* 16.
- [Chawla et al., 2004] Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Editorial: special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*.
- [Chawla et al., 2003] Chawla, N. V., Lazarevic, A., Bowyer, K. W., and Hall, L. O. (2003). Smoteboost: Improving prediction of the minority class in boosting. *Knowledge Discovery in Databases: PKDD*.
- [de Fortuny et al., 2014] de Fortuny, E. J., Stankova, M., Moeyersoms, J., Minnaert, B., Provost, F., and Martens, D. (2014). Corporate residence fraud detection. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*.
- [Delamaire et al., 2009] Delamaire, L., Abdou, H., and Pointon, J. (2009). Credit card fraud and detection techniques: a review. *Banks and Bank systems*.