
# Fraud Detection

**A Report for the Final Evaluation of Project 2**

Submitted by

**HARSHIT ARORA**

**(1613101287/16SCSE101181)**

in partial fulfillment for the award of the degree

of

**Bachelor of Technology**

**IN**

**Computer Science and Engineering**

**School of Computing Science and Engineering**

**Under the Supervision of**

**Prof. C. Ramesh**

**APRIL / MAY- 2020**


# SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

# BONAFIDE CERTIFICATE

Certified that this project report "**FRAUD DETECTION**" is the

bonafide work of "**HARSHIT ARORA (1613101287)**" who carried out the project

work under my supervision.

| | |
|---|---|
| **SIGNATURE** | **SIGNATURE** |
| | **Mr. C. RAMESH** |
| **HEAD OF THE DEPARTMENT** | **SUPERVISOR** |
| **Professor & Dean** | **Professor** |
| **School of Computing Science & &** | **School of Computing Science** |
| **Engineering** | **Engineering** |

# Abstract

Financial fraud is an ever-growing menace with far consequences in the financial industry. Machine Learning had played an imperative role in the detection of credit card fraud in online transactions. Credit card fraud detection, which is a problem, becomes challenging due to two major reasons – first, the profiles of normal and fraudulent behaviors change constantly and secondly, credit card fraud data sets are highly skewed. The performance of fraud detection in credit card transactions is greatly affected by the sampling approach on dataset, selection of variables and detection technique(s) used. This project implements the Random forest algorithm, Logistics regression and Decision Tree classifier on highly skewed credit card fraud data. Dataset of credit card transactions is sourced from European cardholders containing 284,807 transactions. A hybrid technique of under-sampling and oversampling is carried out on the skewed data. Fraud is one of the major ethical issues in the credit card industry. The main aims are, firstly, to analyze the imbalance dataset of credit card fraud and secondly, to review alternative techniques that have been used in fraud detection. The sub-aim is to present, compare and analyze recently published findings in credit card fraud detection. The proposals made in this project are likely to have beneficial attributes in terms of cost savings and time efficiency. The significance of the application of the techniques reviewed here is in the minimization of credit card fraud. Yet there are still ethical issues when genuine credit card customers are misclassified as fraudulent.

# TABLE OF CONTENTS

# Introduction

## Overall description

At the current state of the world, financial organizations expand the availability of financial facilities by employing of innovative services such as credit cards, Automated Teller Machines (ATM), internet and mobile banking services. Besides, along with the rapid advances of e-commerce, the use of credit card has become a convenience and necessary part of financial life. Credit card is a payment card supplied to customers as a system of payment. There are lots of advantages in using credit cards such as: Credit cards can make life easier. They allow customers to purchase on credit in arbitrary time, location and amount, without carrying the cash. Provide a convenient payment method for purchases made on the internet, over the telephone, through ATMs, etc. Having a good credit history is often important in detecting loyal customers. This history is valuable not only for credit cards, but also for other financial services like loans, rental applications, or even some jobs. Lenders and issuers of credit mortgage companies, credit card companies, retail stores, and utility companies can review customer credit score and history to see how punctual and responsible customers are in paying back their debts. Credit cards may also offer customers, additional protection if the purchased merchandise becomes lost, damaged, or stolen. Both the buyer's credit card statement and company can confirm that the customer has bought if the original receipt is lost or stolen. In addition, some credit card companies provide insurance for large purchases.

A making example of the digitalization of our society over the last year is the expansion of the credit card use. The evolution did also bring forth the problem of credit card fraud where ever more phisticated methods are used to steal considerable amount of money. This project discusses the problem of identifying or detecting fraudulent behavior in a credit card transition system. Reasoning under uncertainty is a key element in many artificial intelligence applications in general and machine learning in particular. Therefore, many formalisms have been developed that support that kind of reasoning: probabilistic reasoning, fuzzy logic etc. Here we focus on different machine learning techniques to the credit card fraud detections problem: Logistics

Regression, Isolation Factor and the random forest classifier. Fraud' in credit card transactions is unauthorized and unwanted usage of an account by someone other than the owner of that account. Necessary prevention measures can be taken to stop this abuse and the behavior of such fraudulent practices can be studied to minimize it and protect against similar occurrences in the future. In other words, Credit Card Fraud can be defined as a case where a person uses someone else's credit card for personal reasons while the owner and the card issuing authorities are unaware of the fact that the card is being used. Fraud detection involves monitoring the activities of populations of users in order to estimate, perceive or avoid objectionable behavior, which consist of fraud, intrusion, and defaulting. Fraud Detection monitors all the approved transactions and alerts the most suspicious one. Verifying all the alerts every day is a time consuming and costly process the central idea is to provide some computational learner with a set of training data consisting of some features values that are inherent to the system in which we want to do the fraud detection

In spite of all mentioned advantages, the problem of fraud is a serious issue in-banking services that threaten credit card transactions especially. Fraud is an intentional deception with the purpose of obtaining financial gain or causing loss by implicit or explicit trick. Fraud is a public law violation in which the fraudster gains an unlawful advantage or causes unlawful damage. The estimation of amount of damage made by fraud activities indicates that fraud costs a very considerable sum of money. Credit card fraud is increasing significantly with the development of modern technology resulting in the loss of billions of dollars worldwide each year. Statistics from the Internet Crime Complaint Center show that there has been a significant rising in reported fraud in last decade. Financial losses caused due to online fraud only in US, was reported $3.4 billion in 2011. Fraud detection involves identifying scarce fraud activities among numerous legitimate transactions as quickly as possible. Fraud detection methods are developing rapidly in order to adapt with new incoming fraudulent strategies across the world. But, development of new fraud detection techniques becomes more difficult due to the severe limitation of the ideas exchange in fraud detection. On the other hand, fraud detection is essentially a rare event problem, which has been variously called outlier analysis, anomaly detection, exception mining, mining rare classes, mining imbalanced data etc. The number of fraudulent transactions is usually a very low fraction of the total transactions. Hence the task of detecting fraud transactions in an accurate and efficient manner is fairly difficult and challengeable. Therefore, development of efficient methods which can distinguish rare fraud activities from billions of legitimate transactions seems essential. Although, credit card fraud detection has gained attention and extensive study especially in recent years and there are lots of surveys about this kind of fraud neither classify all credit card fraud detection techniques with analysis of datasets and attributes. Therefore, in this paper, we attempt to collect and integrate a complete set of researches of literature and analyze them from various aspects. The main contributions of this work are highlighted as follows: To the best of our knowledge, the absence of complete and detailed credit card fraud detection survey is an important issue, which is addressed by analyzing the state of the art in credit card fraud detection or benchmark to evaluate detection methods. We attempt to gather different datasets investigated by researchers,

categorize them into real and synthetized groups and extract the common attributes affects the quality of detection.

# Purpose

In the credit card business, fraud occurs when a lender is fooled by a borrower offering him/her purchases, believing that the borrower credit card account will provide payment for this purchase. Ideally, no payment will be made. If the payment is made, the credit card issuer will reclaim the amount paid. Today, with the expansion of e-commerce, it is on the internet that half of all credit card fraud is conducted. Fraudsters have usually connections with the affected business. In the credit card business, it can be an internal party but most likely an external party. As an external party, fraud is committed being a prospective/existing customer or a prospective/existing supplier. Three different profiles can be identified for external fraudsters: the average offender, criminal offender, and organized crime offender. For many companies sometimes dealing with millions of external parties, it is cost-prohibitive to manually check the majority of the external parties' identity and activities. Indeed, to investigate each suspicious transaction, they incur a direct overhead cost for each of them. In order to avoid these overheads and depending on the type of fraud committed, diverse solutions can be implemented.

# Motivations and scope

The explosion of credit card fraud is not only due to the constant increase of card usage but also to the ease of perpetuating credit card fraud. The complexity of credit card fraud is that it may be committed in various ways, including theft fraud, application fraud, counterfeit fraud, bankruptcy fraud. By not paying enough attention to fraud prevention or detection, the risk for the bank is that "credit card fraud remains usually undetected until long after the criminal has completed the crime". Therefore, it will generate irrecoverable costs for the bank. Credit is a method of selling goods or services without the buyer having cash in hand. A credit card is only an automatic way of offering credit to a consumer. Today, every credit card carries an identifying number that speeds shopping transactions. However, references to credit cards have been made as far back as 1890 in Europe. Early credit cards involved sales directly between the merchant offering the credit and credit card, and that merchant's customer.

# LITERATURE SURVEY

The fraud detection is a complex task and there is no system that correctly predicts any transaction as fraudulent. The properties for a good fraud detection system are: Should identify the frauds accurately. Should detect the frauds quickly. Should not classify a genuine transaction as fraud. Outlier detection is a critical task as outliers indicate abnormal running conditions from which significant performance degradation may happen. Techniques used in fraud detection can be divided into two: Supervised techniques where past known legitimate/fraud cases are used to build a model which will produce a suspicion score for the new transactions. Unsupervised are those where there are no prior sets in which the state of the transactions is known to be fraud or legitimate. An unsupervised outlier detection technique does not make any assumption about the availability of labeled data. This method simply seeks those accounts, customer etc., whose behavior is "unusual". Unsupervised methods are useful in applications where there is no prior knowledge about the particular class of observations in a data set. An advantage of using unsupervised methods over supervised methods is that previously occurred undiscovered types of fraud may be detected. Supervised outlier detection techniques assume the availability of a data set which has been needed for the normal as well as the outlier class. Supervised method detects fraudulent transactions that can be used to differentiate between those accounts or transactions which are known to be fraudulent and those which are known to be legitimate. Classification techniques such as statistical discriminate analysis and neural networks can be used to discriminate between fraudulent and non-fraudulent transactions to give transactions a suspicion score. Supervised methods are only trained to differentiate between legitimate transactions and previously known fraud. The two data mining approaches, are support vector machines and random forests, together with the well-known logistic regression, as part of an attempt to detect the credit card fraud. It is well-understood, easy to use, and it is most

commonly used for data-mining. Thus, it provides a useful baseline for comparing performance of newer methods. Supervised learning methods for fraud detection face two challenges. They are: The unbalanced class sizes of legitimate and fraudulent transactions, with legitimate transactions far outnumbering fraudulent ones. The second is to develop supervised models for fraud that can arise from potentially undetected fraud transactions, leading to mislabeled cases in the data to be used for building the model. For the purpose of the above problems, the fraudulent transactions are those specifically identified by the institutional auditors as those that caused an unlawful transfer of funds from the bank sponsoring the credit cards. These transactions were observed to be fraudulent expose. The study is based on real-life data of transactions from an international credit card operation.

# Existing System

Srivastava has implemented a model to show the sequence of credit card transaction process and presents the experimental results which shows the effectiveness of the system and demonstrate the usefulness of learning the spending profile of cardholders. Comparative studies reveal that the Accuracy of the system is close to 80 percent over a wide variation in the input data. Accuracy represents the fraction of total number of transactions (both genuine and fraudulent) that have been detected correctly. The system is also scalable for handling large volumes of transactions. It has already presented a survey of current techniques used in credit card fraud detection and telecommunication fraud. In this paper, comprehensive review of different techniques to detect fraud is provided. Various types of frauds in this paper include credit card frauds, telecommunication frauds, and computer intrusions, Bankruptcy fraud, Theft fraud/counterfeit fraud, Application fraud, Behavioral fraud. Gass algorithm, Bayesian networks, Hidden markov model, Genetic algorithm, A fusion approach using dempster-shafer theory and Bayesian learning, Decision tree, Neural network and Logistic Regression techniques are explained to detect credit card fraud. One aim of this paper is to identify the user model that best identifies fraud cases. Delamaire has identified the different types of credit card fraud such as bankruptcy fraud, counterfeit fraud, theft fraud, application fraud and behavioral fraud and review alternative techniques that include pair-wise matching, decision trees, clustering techniques, neural networks, and genetic algorithms. Also state the problems that have been faced by the banks and credit card companies. The next step in this research program is to focus on the implement of a „suspicious“ scorecard on a real dataset and its evaluation. The main tasks should be to build scoring models to predict fraudulent behavior, taking into account the fields of behavior that should be related to the different types of credit card fraud identified in this paper, and to evaluate the associated ethical implications. The plan is to take one of the European

countries, probably Germany, and then to extend the research to other EU countries. The design a predictive model with sequence of operations in online transaction by using hidden markov model (HMM) and decides whether the user act as a normal user or fraud user. In the trained system, the new transaction is evaluated with transition and observation probability. Depending upon the observation probability, system finds the acceptance probability and decides whether the transaction should be declined or not. Normally existing fraud detection system for online banking will detect the fraudulent transaction after completion of the transaction. This causes the economic loss and makes the bank name as unsecured. The model predicts the fraudulent during the transaction time and prevents the money transfer. As future work, some effective classification algorithms instead of using clustering which can perform well for the prediction.

# PROBLEM STATEMENT

Fraud detection systems are prune to several difficulties and challenges enumerated bellow. An effective fraud detection technique should have abilities to address these difficulties in order to achieve best performance.

## Main challenges

- Imbalanced data: The credit card fraud detection data has imbalanced nature. It means that very small percentages of all credit card transactions are fraudulent. This cause the detection of fraud transactions very difficult and imprecise.

- Overlapping data: many transactions may be considered fraudulent, while actually they are normal (false positive) and reversely, a fraudulent transaction may also seem to be legitimate (false negative). Hence obtaining low rate of false positive and false negative is a key challenge of fraud detection systems.

- Lack of adaptability: classification algorithms are usually faced with the problem of detecting new types of normal or fraudulent patterns. The supervised and unsupervised fraud detection systems are inefficient in detecting new patterns of normal and fraud behaviors, respectively.

- Lack of standard metrics: there is no standard evaluation criterion for assessing and comparing the results of fraud detection systems.

- Different misclassification importance: in fraud detection task, different misclassification errors have different importance. Misclassification of a normal transaction as fraud is not as harmful as detecting a fraud transaction as normal. Because in the first case the mistake in classification will be identified in further investigations.

- Changing fraud patterns over time: This one is the toughest to address since the fraudsters are always in the lookout to find new and innovative ways to get around the systems to commit the act. Thus, it becomes all-important for the deep learning models to be updated with the evolved patterns to detect.

- Data is highly skewed: Enormous Data is processed every day and the model build must be fast enough to respond to the scam in time. Imbalanced Data i.e. most of the transactions *(99.8%)* are not fraudulent which makes it really hard for detecting the fraudulent ones. Data availability as the data is mostly private. Misclassified Data can be another major issue, as not every fraudulent transaction is caught and reported. Adaptive techniques used against the model by the scammers.

## How to tackle these challenges?

- The model used must be simple and fast enough to detect the anomaly and classify it as a fraudulent transaction as quickly as possible.

- Imbalance can be dealt with by properly using some methods which we will talk about in the next paragraph.

- For protecting the privacy of the user, the dimensionality of the data can be reduced.

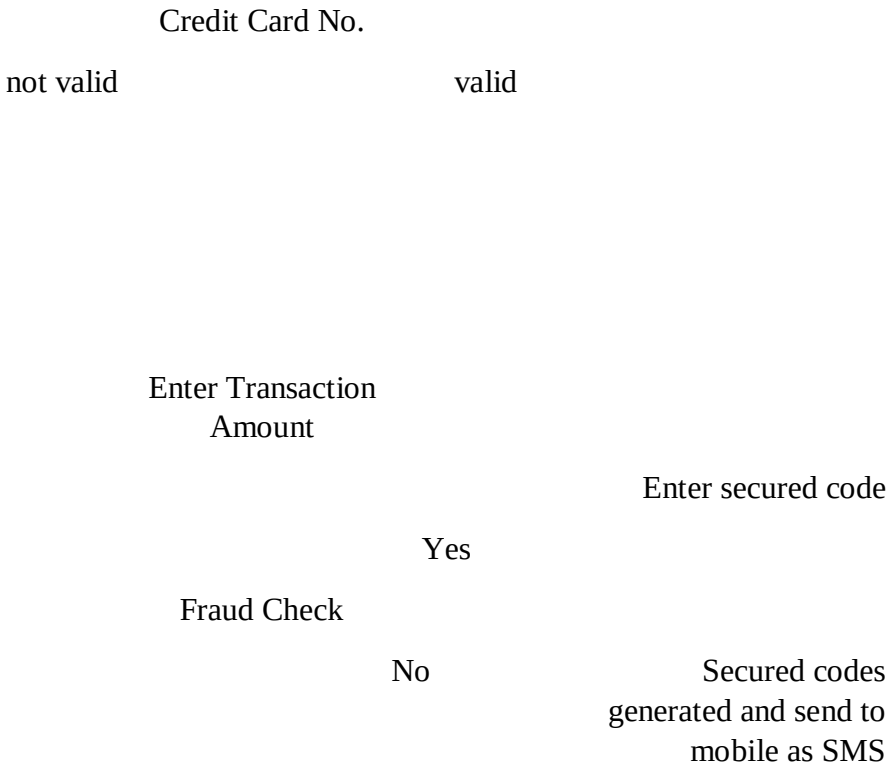
- A more trustworthy source must be taken which double-check the data, at least for training the model.

- We can make the model simple and interpretable so that when the scammer adapts to it with just some tweaks, we can have a new model up and running to deploy.

# Proposed model

Our goal is to implement 3 different machine learning models in order to classify, to the highest possible degree of accuracy, credit card fraud from a dataset gathered in Europe in 2 days in September 2019. After initial data exploration, we knew would implement a logistic regression model, a Random forest classifier and local outlier factor. Some challenges we observed from the start were the huge imbalance in the dataset: frauds only account for 0.172% of fraud transactions. In this case, it is much worse to have false negatives than false positives in our predictions because false negatives mean that someone gets away with credit card fraud. False positives, on the other hand, merely cause a complication and possible hassle when a cardholder must verify that they did, in fact, complete said transaction (and not a thief).

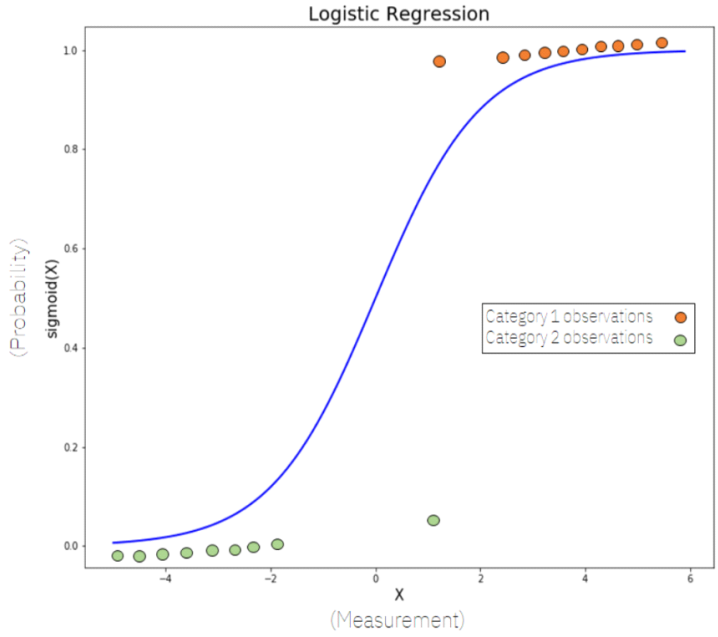
USER




There is a several algorithms that we will use in our project which we need to explain how exactly works the first one that we will start with:

# Logistics Regression

Logistic Regression is one of the basic and popular algorithms to solve a classification problem. It is named as 'Logistic Regression', because it's underlying technique is quite the same as Linear Regression. The term "Logistic" is taken from the Logit function that is used in this method of classification. In the Machine Learning world, Logistic Regression is a kind of parametric classification model, despite having the word 'regression' in its name. This means that logistic regression models are models that have a certain fixed number of parameters that depend on the number of input features, and they output categorical prediction, like for example if a plant belongs to a certain species or not. In reality, the theory behind Logistic Regression is very similar to the one from Linear Regression. In Logistic Regression, we don't directly fit a straight line to our data like in linear regression. Instead, we fit a S shaped curve, called Sigmoid, to our observations.




Let's examine this figure closely.

First of all, like we said before, Logistic Regression models are classification models; specifically, binary classification models (they can only be used to distinguish between 2 different categories — like if a person is obese or not given its weight, or if a house is big or small given its size). This means that our data has two kinds of observations (Category 1 and Category 2 observations) like we can observe in the figure. Secondly, as we can see, the Y-axis goes from 0 to 1. This is because the sigmoid function always takes as maximum and minimum these two values, and this fits very well our goal of classifying samples in two different categories. By computing the *sigmoid* function of X (that is a weighted sum of the input features, just like in Linear Regression), we get a probability (between 0 and 1 obviously) of an observation belonging to one of the two categories.

The formula for the sigmoid function is the following:

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

Implementation of Logistics Regression include the following steps:

- Collecting of the data set and generate the data set using the make classification() function. We need to specify the number of samples, the number of classes, the number of features and other parameters.

- Analyze the dataset in this we create a simple scatter plot between different classes and variables we are using.

- Data wrangling is the process of checking dataset that it contains null values or any duplicate values or remove the duplicity of the dataset.

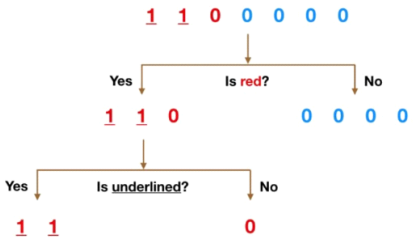
- In the Train and testing we split the dataset into training dataset or testing dataset. The training dataset is used to train the mode while the test dataset is used to test the model's performance on new data.

- Perform Logistics Regression on the dataset and check the accuracy and display the confusion matrix.

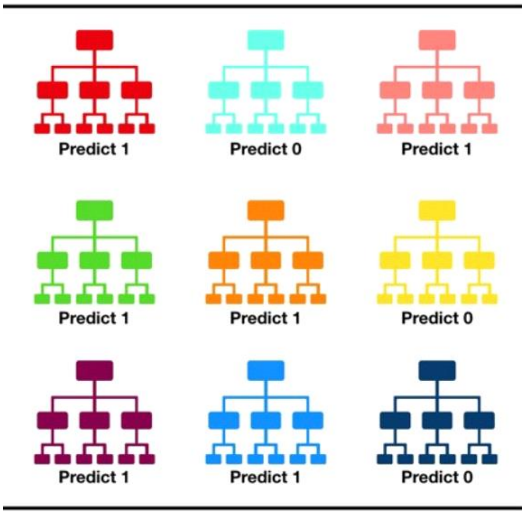
# Random Forest Classifier

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees. The random forest is a model made up of many decision trees. Rather than just simply averaging the prediction of trees (which we could call a "forest"), this model uses two key concepts that gives it the name random*:* random sampling of training data points when building trees and random subsets of features considered when splitting nodes. Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction

The fundamental concept behind random forest is a simple but powerful one the wisdom of crowds. In data science speak, the reason that the random forest model works so well is: A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So, the prerequisites for random forest to perform well are:




Simple Decision tree example




Random forest example

There needs to be some actual signal in our features so thatmodels built using those features do better than random guessing. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other. Bagging (Bootstrap Aggregation) — Decisions trees are very sensitive to the data they are trained on — small changes to the training set can result in significantly different tree structures. Random forest takes advantage of this by allowing each individual tree to randomly sample from the dataset with replacement, resulting in different trees. This process is known as bagging.

In a normal decision tree, when it is time to split a node, we consider every possible feature and pick the one that produces the most separation between the observations in the left node vs. those in the right node. In contrast, each tree in a random forest can pick only from a random subset of features. This forces even more variation amongst the trees in the model and ultimately results in lower correlation across trees and more diversification. A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. The

basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

Approach:

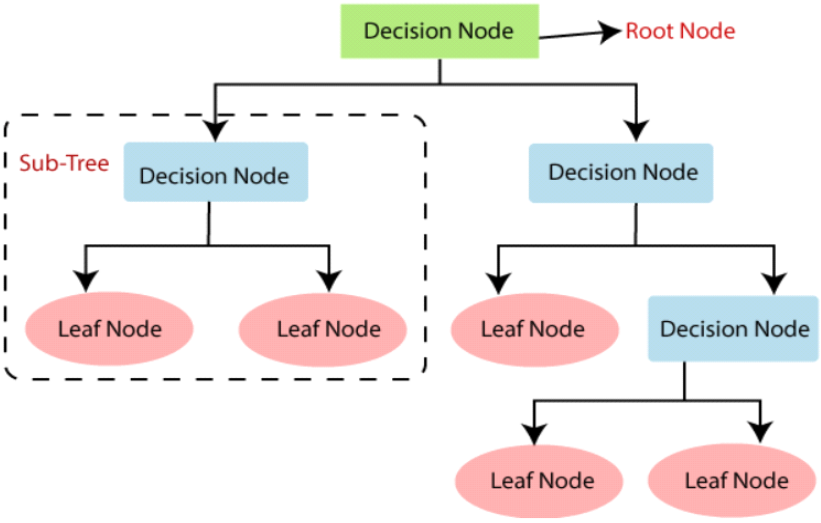
- Pick at random K data points from the training set.
- Build the decision tree associated with those K data points.
- Choose the number N tree of trees you want to build and repeat step 1 & 2.
- For a new data point, make each one of your N tree trees predict the value of Y for the data point, and assign the new data point the average across all of the predicted Y values.

The solution is to not rely on any one individual, but pool the votes of each analyst. Furthermore, like in a random forest, allow each analyst access to only a section of the reports and hope the effects of the noisy information will be cancelled out by the sampling. In real life, we rely on multiple sources (never trust a solitary Amazon review), and therefore, not only is a decision tree intuitive, but so is the idea of combining them in a random forest.

# Decision Tree Algorithm

Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems. Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree. We can represent any Boolean function on discrete attributes using the decision tree. A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn't split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively. Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can't ignore the simplicity of this algorithm. The feature importance is clear and relations can be viewed easily. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree as the target

is to classify passenger as survived or died. Regression trees are represented in the same manner, just they predict continuous values like price of a house.




Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets. Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node. Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions. A tree formed by splitting the tree. Pruning is the process of removing the unwanted branches from the tree. The root node of the tree is called the parent node, and other nodes are called the child nodes. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees. So, what is actually going on in the background? Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, you will need to trim it down for it to look beautiful. Below are some assumptions that we made while using decision tree: At the beginning, we consider the whole training set as the root. Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model. On the basis of attribute values records are distributed recursively. We use statistical methods for ordering attributes as root or the internal node. CART (classification and regression tree) divides the data in homogenous subsets using binary recursive partitions. The most discriminative variable is first selected as the root node to partition the data set into branch nodes. The partitioning is repeated until the nodes are homogenous enough to be terminal which are called leaves.

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm: Begin the tree with the root node, says S, which contains the complete dataset. Find the best attribute in the dataset using Attribute Selection Measure (ASM**).** Divide the S into subsets that contains possible values for the best attributes. Generate the decision tree node, which contains the best attribute. Recursively make

new decision trees using the subsets of the dataset created in above step. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node. While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. There are popular techniques for ASM, which are: Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute It calculates how much information a feature provides us about a class. According to the value of information gain, we split the node and build the decision tree. A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first.

# Implementation

Imbalance Dataset

Imbalanced dataset is relevant primarily in the context of supervised machine learning involving two or more classes. Imbalance means that the number of data points available for different the classes is different: If there are two classes, then balanced data would mean 50% points for each of the class. For most machine learning techniques, little imbalance is not a problem. So, if there are 60% points for one class and 40% for the other class, it should not cause any significant performance degradation. Only when the class imbalance is high, e.g. 90% points for one class and 10% for the other, standard optimization criteria or performance measures may not be as effective and would need modification.

The dataset we are using in this project is highly skewed, consisting of 492 frauds in a total of 284,807 observations. This resulted in only 0.172% fraud cases. This skewed set is justified by

the low number of fraudulent transactions. The ‘Time’ and ‘Amount’ features are not transformed data. There is no missing value in the dataset.

Principle Component Analysis

The dataset contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example- dependent cost - sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Exploratory Data Analysis

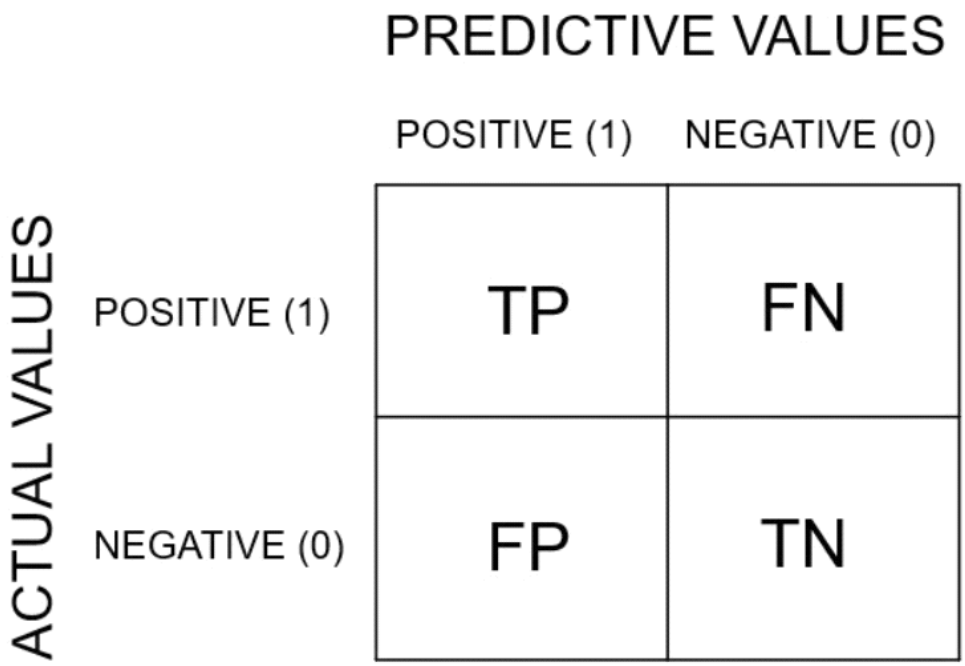
Exploratory Data Analysis (EDA) is an approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) to maximize insight into a data set, uncover underlying structure, extract important variables, detect outliers and anomalies, test underlying assumptions, develop parsimonious models and determine optimal factor settings. The EDA approach is precisely that--an approach--not a set of techniques, but an attitude/philosophy about how a data analysis should be carried out.

## Correlation Matrix

Correlation matrix graphically gives us an idea of how features correlate with each other and can help us predict what are the features that are most relevant for the prediction. In the HeatMap we can clearly see that most of the features do not correlate to other features but there are some features that either has a positive or a negative correlation with each other. For example, *V2* and *V5* are highly negatively correlated with the feature called *Amount*. We also see some correlation with *V20* and *Amount*. This gives us a deeper understanding of the Data available to us.

### Confusion Matrix

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made. A confusion matrix for binary classification shows the four different outcomes: true positive, false positive, true negative, and false negative. The actual values form the columns, and the predicted values (labels) form the rows. The intersection of the rows and columns show one of the four outcomes. For example, if we predict a data point is positive, but it actually is negative, this is a false positive.




Confusion Matrix for the Binary Classification

Accuracy

Accuracy is calculated as the total no. of corrected prediction divided by the total number od dataset. Accuracy works well on the balanced dataset. In case of imbalanced dataset accuracy mislead the performance.

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

Precision

Precision is defined as the ratio of the total number of correctly classified positive classes divided by the total number of predicted positive classes. Or, out of all the predictive positive classes, how much we predicted correctly. Precision should be high.

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

Recall

Recall is defined as the ratio of the total number of correctly classified positive classes divide by the total number of positive classes. Or, out of all the positive classes, how much we have predicted correctly. Recall should be high.

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

F-1 Score

It is difficult to compare two models with different Precision and Recall. So to make them comparable, we use F-Score. It is the Harmonic Mean of Precision and Recall. As compared to Arithmetic Mean, Harmonic Mean punishes the extreme values more. F-score should be high.

$$F\text{-}score = \frac{2 * Recall * Precision}{Recall + Precision}$$

MCC

For binary classification, there is another solution: treat the true class and the predicted class as two (binary) variables, and compute their correlation coefficient (in a similar way to computing correlation coefficient between any two variables). The higher the correlation between true and predicted values, the better the prediction. This is the phi-coefficient (φ), rechristened Matthews Correlation Coefficient **(**MCC) when applied to classifiers.

# Coding

## Importing python libraries

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

```
import warnings
warnings.filterwarnings('ignore')
LABELS = ["Normal", "Fraud"]
```

## Load the data set

```
data = pd.read_csv('creditcard.csv')
print(data.columns)
```

```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

## Read the data set

```
data = pd.read_csv('creditcard.csv')
data.head()
```

Out[4]:

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | ... | V21 | V22 | V23 | V24 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.098698 | 0.363787 | ... | -0.018307 | 0.277838 | -0.110474 | 0.066928 | 0. |
| **1** | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0.255425 | ... | -0.225775 | -0.638672 | 0.101288 | -0.339846 | 0. |
| **2** | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 | -1.514654 | ... | 0.247998 | 0.771679 | 0.909412 | -0.689281 | -0. |
| **3** | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1.387024 | ... | -0.108300 | 0.005274 | -0.190321 | -1.175575 | 0. |
| **4** | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 | 0.817739 | ... | -0.009431 | 0.798278 | -0.137458 | 0.141267 | -0. |

5 rows × 31 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

## Data analysis

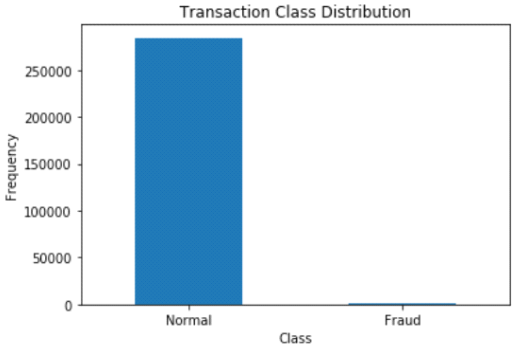
data.isnull().values.any()

```
Out[7]: False
```

count_classes = pd.value_counts(data['Class'], sort = True)

```
count_classes.plot(kind = 'bar', rot=0)
plt.title("Transaction Class Distribution")
plt.xticks(range(2), LABELS)
plt.xlabel("Class")
plt.ylabel("Frequency")
```

```
Out[8]: Text(0, 0.5, 'Frequency')
```




## Get the fraud and normal data set

```
fraud = data[data['Class']==1]
normal = data[data['Class']==0]
print(fraud.shape,normal.shape)
```

```
(492, 31) (284315, 31)
```

## Print the amount details for Fraudulent Transaction

```
fraud.Amount.describe()
```

```
Out[11]: count     492.000000
         mean      122.211321
         std       256.683288
         min         0.000000
         25%         1.000000
         50%         9.250000
         75%       105.890000
         max      2125.870000
         Name: Amount, dtype: float64
```

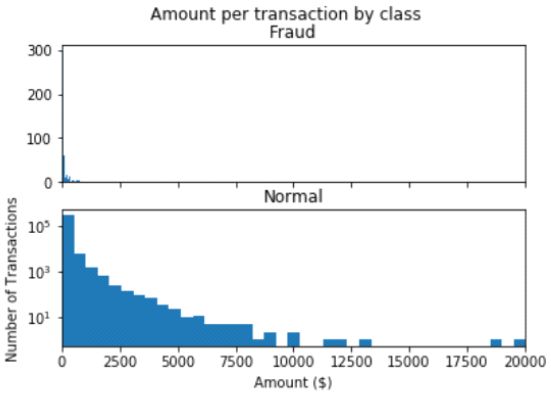
Print the amount details for Normal Transaction

```
normal.Amount.describe()
```

```
Out[12]: count    284315.000000
         mean         88.291022
         std         250.105092
         min           0.000000
         25%           5.650000
         50%          22.000000
         75%          77.050000
         max       25691.160000
         Name: Amount, dtype: float64
```

# Need to analyze more amount of information from the transaction data

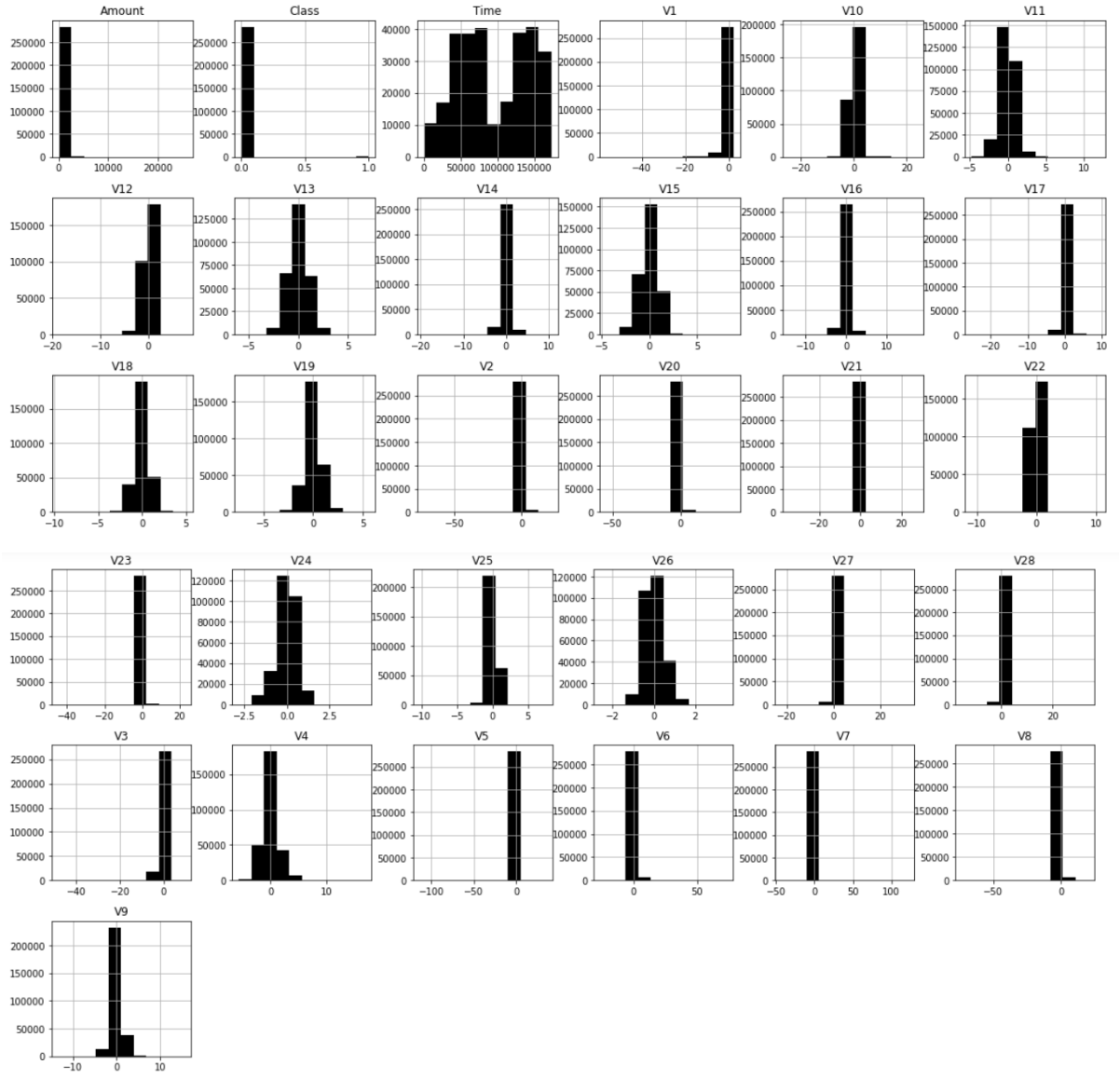
```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Amount per transaction by class')
bins = 50
ax1.hist(fraud.Amount, bins = bins)
ax1.set_title('Fraud')
ax2.hist(normal.Amount, bins = bins)
ax2.set_title('Normal')
plt.xlabel('Amount ($)')
plt.ylabel('Number of Transactions')
plt.xlim((0, 20000))
plt.yscale('log')
plt.show();
```



Exploratory data analysis (Histrogram)

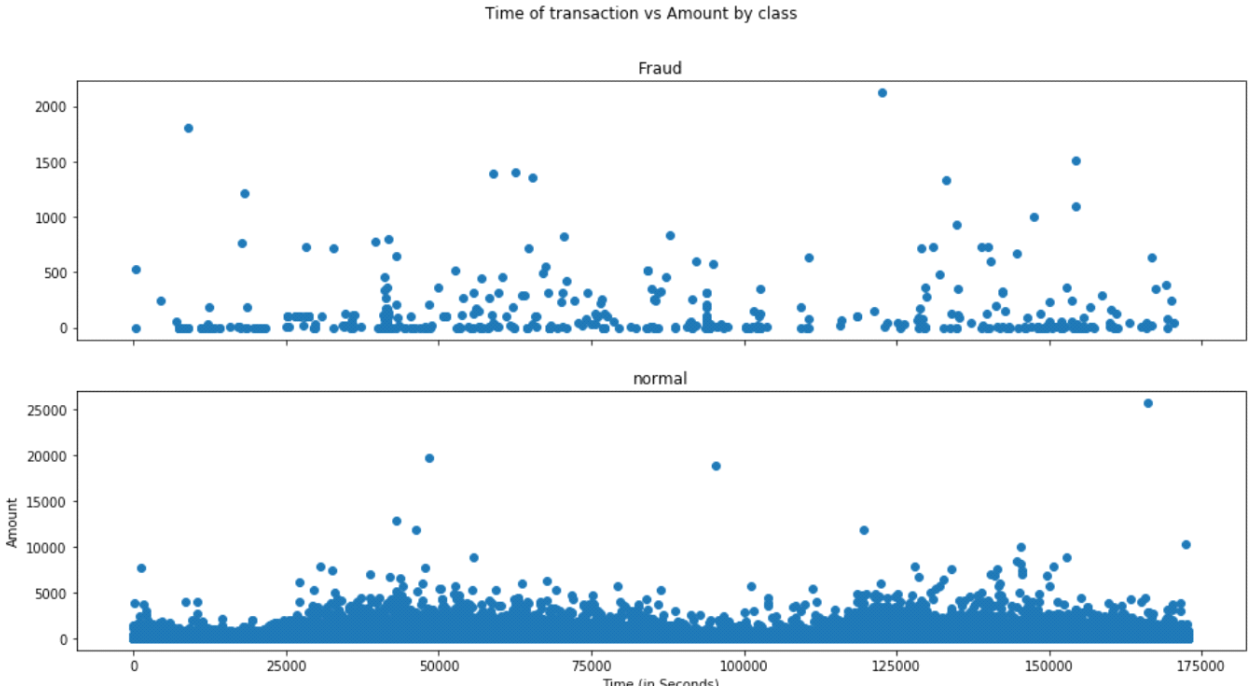
data.hist(figsize=(20,20),color='black')

plt.show()



Find out transaction in certain time frame

```
rcParams['figure.figsize'] = 16, 8
f,(ax1, ax2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Time of transaction vs Amount by class')
ax1.scatter(fraud.Time, fraud.Amount)
ax1.set_title('Fraud')
rcParams['figure.figsize'] = 16, 8
ax2.scatter(normal.Time, normal.Amount)
ax2.set_title('normal')
plt.xlabel('Time (in Seconds)')
plt.ylabel('Amount')
plt.show()
```



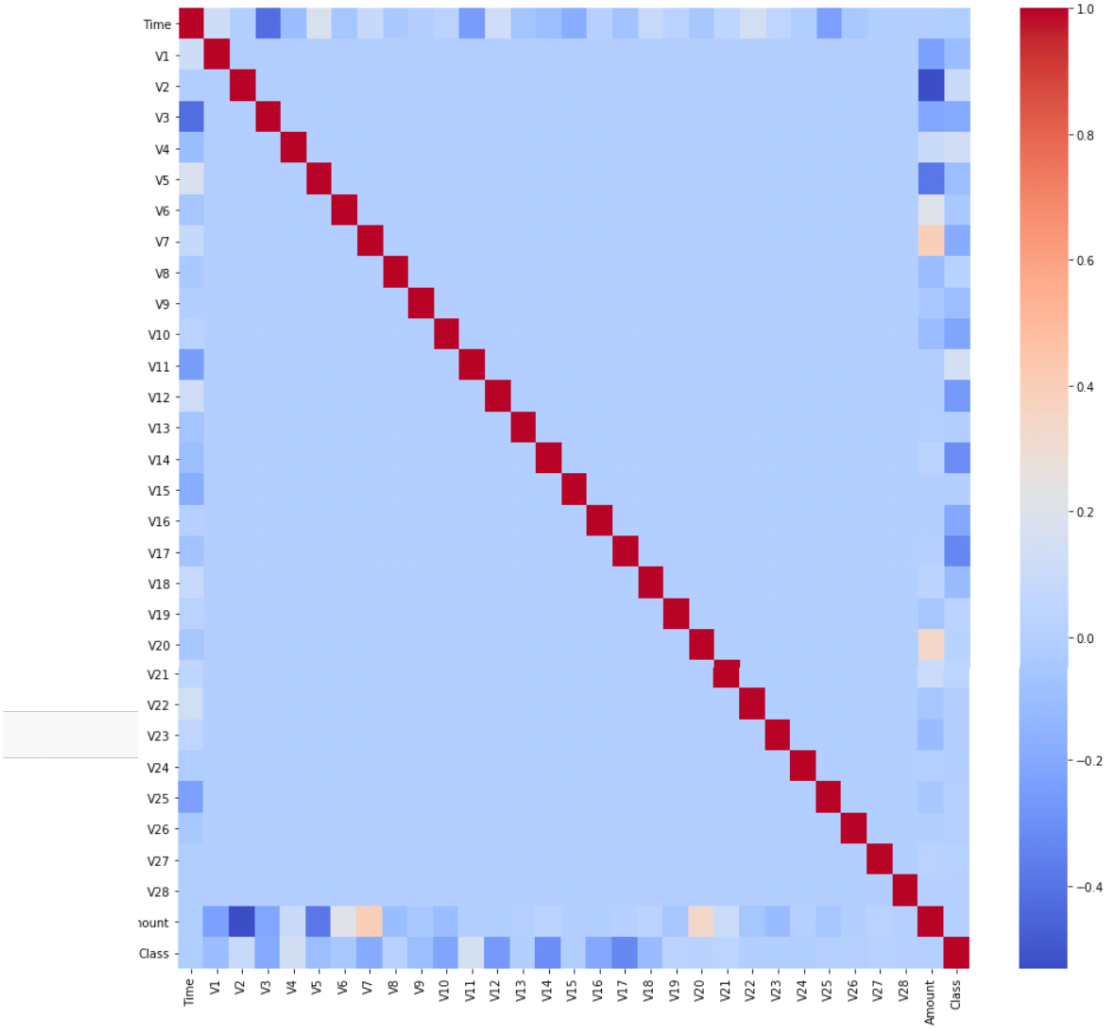
## Correlation Matrix

plt.figure(figsize=(15,15))

corr = data.corr()

sns.heatmap(corr , cmap='coolwarm')

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x29ddf422988>
```



## Dividing X and Y values in the data set

```
X = data.drop(['Class'], axis = 1)

Y = data["Class"]

print(X.shape)

print(Y.shape)

xData = X.values

yData = Y.values
```

```
(284807, 30)
(284807,)
```

## Training and testing of data

```
from sklearn.model_selection import train_test_split

xTrain, xTest, yTrain, yTest = train_test_split( xData, yData, test_size = 0.30, random_state = 1)

xTest
```

```
Out[20]: array([[ 1.19907000e+05, -6.11711999e-01, -7.69705324e-01, ...,
                  6.65013699e-02,  2.21179560e-01,  1.79000000e+00],
                [ 7.83400000e+04, -8.14681711e-01,  1.31921886e+00, ...,
                  1.62427330e-01,  5.94562455e-02,  1.98000000e+00],
                [ 8.23820000e+04, -3.18193485e-01,  1.11861770e+00, ...,
                  2.49049701e-01,  9.25156059e-02,  8.90000000e-01],
                ...,
                [ 3.39870000e+04, -1.76040323e-01,  1.09323870e+00, ...,
                  2.81090331e-02, -4.55463382e-02,  1.52100000e+01],
                [ 1.32357000e+05,  2.00094255e+00, -4.80504194e-01, ...,
                 -1.88874231e-02, -4.26610031e-02,  2.99900000e+01],
                [ 3.38130000e+04,  1.06166420e+00, -3.07508304e-01, ...,
                 -2.49620020e-02,  2.45613875e-02,  7.39700000e+01]])
```

## Model 1 (random Forest Classifier)

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()

rfc.fit(xTrain, yTrain)
```

```
yPred = rfc.predict(xTest)


n_outliers = len(fraud)
n_errors = (yPred != yTest).sum()
print(" Random Forest classifier model is used ")
acc = accuracy_score(yTest, yPred)
print("The accuracy is {}".format(acc))
prec = precision_score(yTest, yPred)
print("The precision is {}".format(prec))
rec = recall_score(yTest, yPred)
print("The recall is {}".format(rec))
f1 = f1_score(yTest, yPred)
print("The F1-Score is {}".format(f1))
MCC = matthews_corrcoef(yTest, yPred)
print("The Matthews correlation coefficient is{}".format(MCC))
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix

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

```
plt.xlabel('Predicted class')
plt.show()
from sklearn.metrics import roc_auc_score
roc_auc_score(yTest, yPred)
```

## Model 2 (Logistics Regression)

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(xTrain, yTrain)
yPred = lr.predict(xTest)
model.score(xtest,ytest)
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix
classification_report(yTest,yPred)
n_outliers = len(fraud)
n_errors = (yPred != yTest).sum()
print(" Logistic Regressiomn model is used ")
  acc = accuracy_score(yTest, yPred)
print("The accuracy is {}".format(acc))
prec = precision_score(yTest, yPred)
print("The precision is {}".format(prec))
rec = recall_score(yTest, yPred)
print("The recall is {}".format(rec))
 f1 = f1_score(yTest, yPred)
```

```
print("The F1-Score is {}".format(f1))
 MCC = matthews_corrcoef(yTest, yPred)
print("The Matthews correlation coefficient is{}".format(MCC))

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

## Model 3(Decission Tree algorithm)

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf.fit(xTrain, yTrain)
yPred = clf.predict(xTest)
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
```

```
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix
classification_report(yTest,yPred)
n_outliers = len(fraud)
n_errors = (yPred != yTest).sum()
print(" Decission tree algorithm  is used ")
acc = accuracy_score(yTest, yPred)
print("The accuracy is {}".format(acc))
prec = precision_score(yTest, yPred)
print("The precision is {}".format(prec))
rec = recall_score(yTest, yPred)
print("The recall is {}".format(rec))
f1 = f1_score(yTest, yPred)
print("The F1-Score is {}".format(f1))
MCC = matthews_corrcoef(yTest, yPred)
print("The Matthews correlation coefficient is{}".format(MCC))
conf_matrix = confusion_matrix(yTest, yPred)
LABELS = ['Normal', 'Fraud']
conf_matrix = confusion_matrix(yTest, yPred)
plt.figure(figsize =(12, 12))
sns.heatmap(conf_matrix, xticklabels = LABELS,    yticklabels = LABELS,
annot = True, fmt ="d");
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
from sklearn.metrics import roc_auc_score
roc_auc_score(yTest, yPred)
```

# Output

## Model 1 (Random Forest Classifier)

Implementation of Random Forest

```
Out[22]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                criterion='gini', max_depth=None, max_features='auto',
                                max_leaf_nodes=None, max_samples=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_jobs=None, oob_score=False, random_state=None,
                                verbose=0, warm_start=False)
```

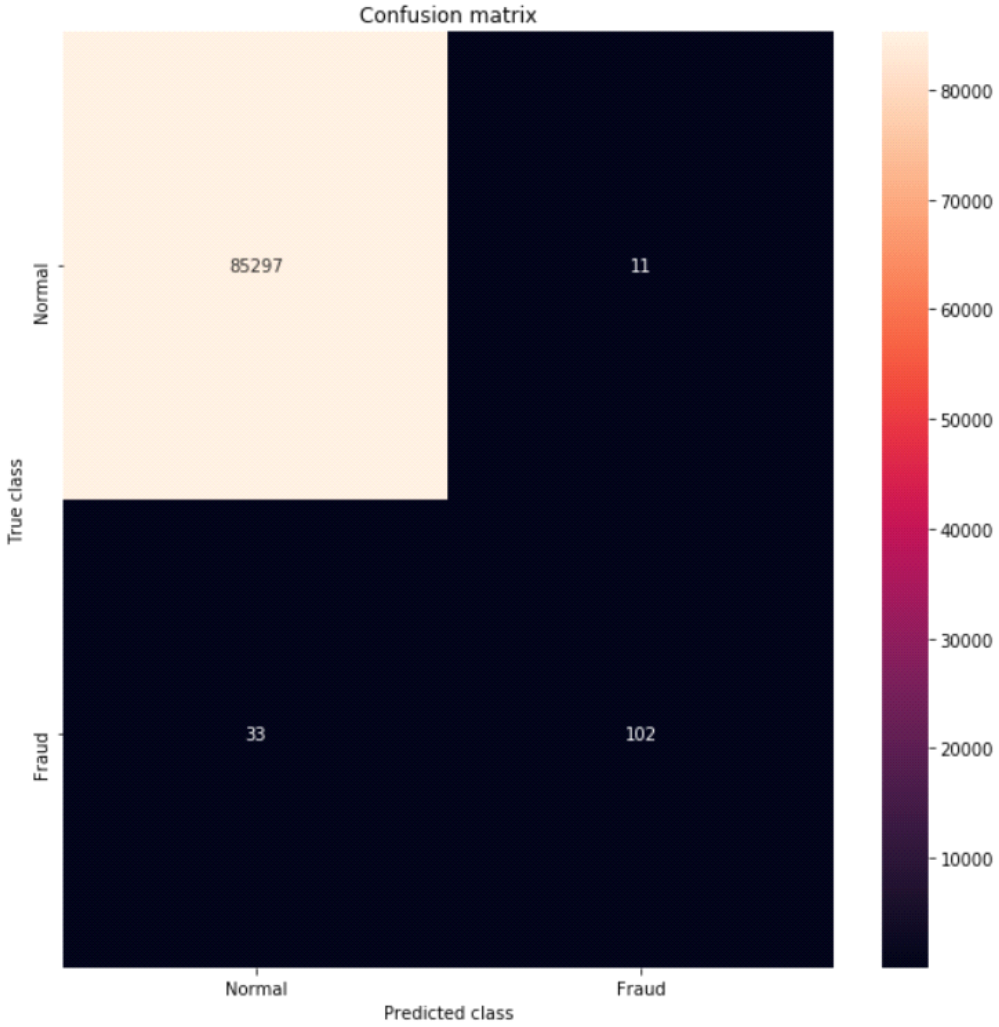
Evaluating accuracy, precision, recall, F1-score and mcc

```
 Random Forest classifier model is used
The accuracy is 0.9994850368081645
The precision is 0.9026548672566371
The recall is 0.7555555555555555
The F1-Score is 0.8225806451612903
The Matthews correlation coefficient is0.8255890977055305
```

Classification Report

```
Out[27]: '              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00     85308\n
         1       0.90      0.76      0.82       135\n\n    accuracy                           1.00     85443\n   macro avg       0.95
         0.88      0.91     85443\nweighted avg       1.00      1.00      1.00     85443\n'
```

Confusion Matrix




ROC_AUC_score

```
Out[29]: 0.8777133055125741
```

Model 2(Logistics Regression)

Implementation of Logistics Regression

```
Out[33]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                            intercept_scaling=1, l1_ratio=None, max_iter=100,
                            multi_class='auto', n_jobs=None, penalty='l2',
                            random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                            warm_start=False)
```

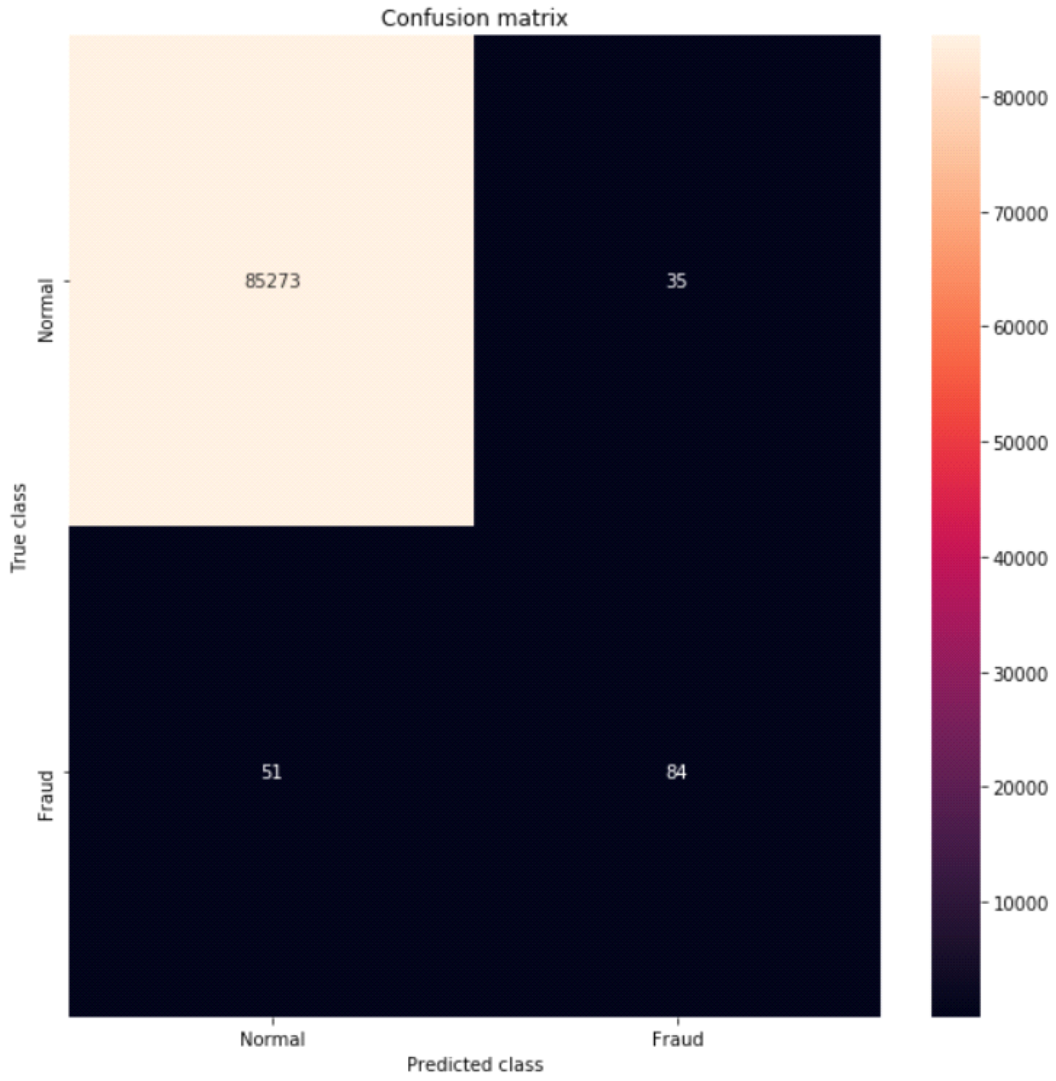
Evaluating accuracy, precision, recall, F1-score and mcc

```
 Logistic Regressiomn model is used
The accuracy is 0.9989934810341398
The precision is 0.7058823529411765
The recall is 0.6222222222222222
The F1-Score is 0.6614173228346457
The Matthews correlation coefficient is0.6622344054729314
```

Classification Report

```
Out[37]: '              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00     85308\n
         1       0.71      0.62      0.66       135\n\n    accuracy                           1.00     85443\n   macro avg       0.85
         0.81      0.83     85443\nweighted avg       1.00      1.00      1.00     85443\n'
```

Confusion Matrix




ROC_AUC_score

```
Out[43]: 0.8109059720854629
```

Model 3(Decision Tree Algorithm)

Implementation of Decision Tree Algorithm

```
Out[46]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                                max_depth=None, max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=None, splitter='best')
```

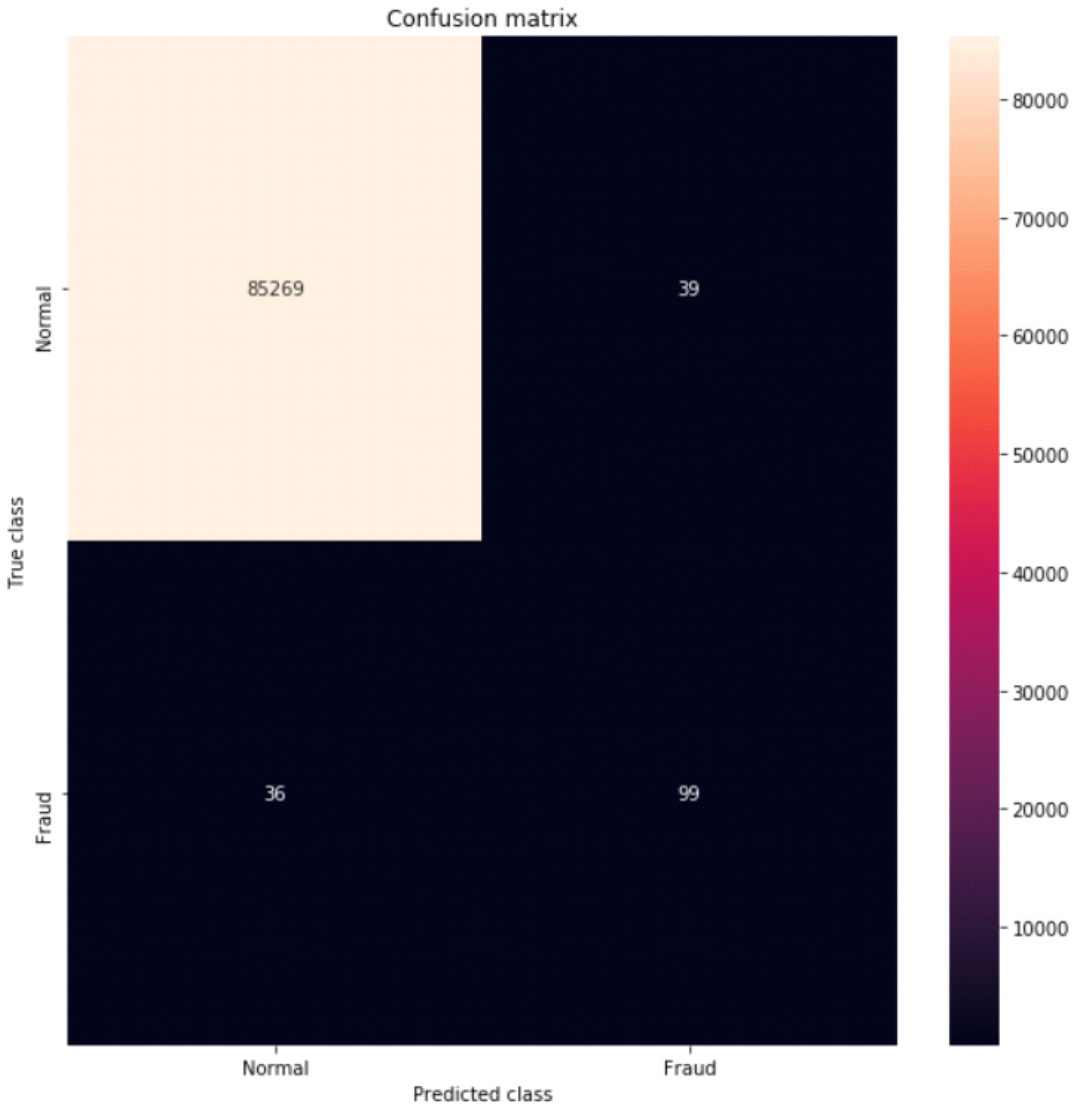
Evaluating accuracy, precision, recall, F1-score and mcc

```
 Decission tree algorithm  is used
The accuracy is 0.9991222218320986
The precision is 0.717391304347826
The recall is 0.7333333333333333
The F1-Score is 0.7252747252747253
The Matthews correlation coefficient is0.7248790961856182
```

Classification Report

```
Out[49]: '              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00     85308\n
         1       0.72      0.73      0.73       135\n\n    accuracy                           1.00     85443\n   macro avg       0.86
         0.87      0.86     85443\nweighted avg       1.00      1.00      1.00     85443\n'
```

Confusion Matrix




ROC_AUC_score

```
Out[54]: 0.8664380831809444
```

# Conclusion

We investigated the data, checking for data unbalancing, visualizing the features and understanding the relationship between different features. We then investigated predictive models. The data was split in 3 parts, a train set, a validation set and a test set. The credit card fraud detection is an imbalanced classification problem: we have two classes we need to identify —normal and Fraud — with one category representing the overwhelming majority of the data points. In these cases the positive class — normal — is greatly outnumbered by the negative class fraud. The all three algorithm Logistics regression, random forest and decision tree are having 0.999 accuracy. These types of problems are examples of the fairly common case in data science when accuracy is not a good measure for assessing model performance. In this case we use other metrics such as precision, recall and f1 score and auc roc curve and mcc. Accuracy as Metrics is not sufficient to judge the 'Goodness' of a Model when data is hugely Imbalanced that's where ROC curve comes into picture. Logistics Regression detected 84 frauds versus Decision Tree detecting 99 frauds vs. Random Forest detecting 102 frauds in the confusion matric. When we compare model on the basis of the precision we get the result that the Random Forest has the highest precision as compared to the logistics Regression and the Decision Tree. In the case of recall also we get same result as we ret in the precision. The value of F1 score is a also high in the random forest algorithm as compared to the Logistics Regression and the Decision Tree. For binary classification, there is another solution: treat the true class and the predicted class as two (binary) variables, and compute their correlation coefficient (in a similar way to computing correlation coefficient between any two variables). The higher the correlation between true and predicted values, the better the prediction. The random forest has the Higher correlation coefficient as compared to the Logistics Regression and the Decision Tree.

Finally, we can quantify a model's by calculating the total ROC AUC, a metric which falls between 0 and 1 with a higher number indicating better classification performance. The Logistics Regression have the 0.81 value of ROC AUC, for the Decision Tree the value is 0.86 and for the Random Forest the value is 0.87. So we conclude that the ROC AUC of the Random Forest is high as compared to other models. So overall Random Forest Method performed much better in determining the fraud cases as comparted to the Logistics Regression and the Decision Tree. We can also improve on this result by increasing the sample size or use deep learning algorithms however at the cost of computational expense. We can also use complex anomaly detection models to get better result in determining more fraudulent cases

# Future Work

Credit card fraud has become more and more rampant in recent years. Fraud detection methods are continuously developed to defend criminals in adapting to their strategies. In Fraud detection, identifying Fraud as quickly as possible once it has been done through fraud detection techniques, is now becoming easier and faster. The techniques which were studied here, through which credit card fraud can be detected quickly and fast and the crime can be stopped. The Future work is to design an improved technique which will be much better than the available techniques and get the better result.

# References

- J. Hand, G. Blunt, M.G. Kelly, and N.M. Adams,
- “Data Mining for Fun and Profit,” Statistical Science.
- Statistics for General and On - Line Card Fraud
- http://www.epaynews.com/stat
- Srivastava, “Credit card fraud detection’
- ROC-AUC characteristic, https://en.wikipedia.org/wiki/Receiver_operating_characteristic
- RandomForrestClassifier, http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- Credit Card Fraud Detection Database, https://www.kaggle.com/mlg-ulb/creditcardfraud