# Project Report
# On
# Credit Card Fraud Detection

***Submitted in Partial fulfillment for the award of degree of Bachelor of Engineering in Computer Science and Engineering***

**Submitted to**

## Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)


Submitted By:

**Tanusha Thakur(0126CS191115)**

**Ritika Makhija (0126CS191091)**
**Paridhi Shrivastava(0126CS191070)**
**Neha Kumari( 0126CS191064)**

Under the Guidance of
Dr Amit Dubey
Professor

**Department of Computer Science & Engineering**



**ORIENTAL COLLEGE OF TECHNOLOGY, BHOPAL**
**Approved by AICTE New Delhi & Govt. of M.P Affiliated To**
**RajivGandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)**

**Session:2020-21**

ORIENTAL COLLEGE OF TECHNOLOGY, BHOPAL
Approved by AICTE New Delhi & Govt. of M.P. & Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

# CERTIFICATE

This is to certify that the work embodied in this Project, Dissertation Report entitled as **"Credit Card Fraud Detection"** being Submitted by **Neha Kumari(0126CS191064), Ritika Makhija (0126CS191091), Tanusha Thakur(0126CS191115) and Paridhi Shrivastava(0126CS191070)** in partial fulfillment of the requirement for the award of "Bachelor of Technology " in Computer Science & Engineering discipline to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.) during the academic year 2020-21 is a record of bonafide piece of work, carried out under my supervision and guidance in the Department of computer Science & Engineering, Oriental College of Technology, Bhopal.

## Approved By:-

Dr. Amit Dubey - Head of Department

ORIENTAL COLLEGE OF TECHNOLOGY, BHOPAL

Approved by AICTE New Delhi & Govt. of M.P. & Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

# CERTIFICATE OF APPROVAL

This Project "**Credit Card Fraud Detection**" being submitted by **Neha Kumari(0126CS191064), Tanusha Thakur (0126CS191115), Paridhi Shrivastava (0126CS191070), Ritika Makhija(0126CS191091)** and has been examined by me & hereby approve for the partial fulfillment of the requirement for the award of "Bachelor of Technology in Computer Science & Engineering" for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expresses or conclusion draw there in, but the Project only for the purpose for which it has been submitted.

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

**Date:** **Date:**

# CANDIDATE DECLARATION

We hereby declare that the Project dissertation work presented in the report entitled as “Credit Card Fraud Detection” submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering of Oriental College of Technology is an authentic record of our own work.

We have not submitted the part and partial of this report for the award of any other degree or diploma.

**Paridhi Shrivastava (0126CS191070)**

**Tanusha Thakur (0126CS191115)**

**Ritika Makhija (0126CS191091)**

**Neha Kumari(0126CS191064)**

Date:

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

**Dr Amit Dubey**

**Guide**

# ACKNOWLEDGMENT

We are heartily thankful to the Management of Oriental College of Technology for providing us all the facilities and infrastructure to take our work to the final stage.

It is the constant supervision, moral support and proper guidance of our respected Director Dr. Amita Mahor, who motivated throughout the work.

We express deep sense of gratitude and respect to our learned guide Dr Amit Dubey Professor in the Department of Computer Science & Engineering, during all phases' of our work. Without his enthusiasm and encouragement this dissertation would not have been completed. His valuable knowledge and innovative ideas helped us to take the work to the final stage. He has timely suggested actions and procedures for which we are really grateful and thankful to him.

We express our gratefulness to Dr.Amit Dubey Head of Computer Science & Engineering Department for providing all the facilities available in the department for his continuous support, advice, and encouragement during this work and also help to extend our knowledge and proper guidelines.

Constant help, moral and financial support of our loving parents motivated us to complete the work. We express our heartily thanks to our all family members for their co-operation.

We really admire the fond support of our class-mates for their co-operation and constant help. It gives immense pleasure to acknowledge the encouragement and support extended by them.

Last but not the least we are extremely thankful to all who have directly or indirectly helped us for the completion of the work

**Paridhi Shrivastava (0126CS191070)**

**Tanusha Thakur (0126CS191115)**

**Ritika Makhija (0126CS191091)**

**Neha Kumari(0126CS191064)**

# ABSTRACT

It is vital that credit card companies are able to identify fraudulent credit card transactions so that customers are not charged for items that they did not purchase. Such problems can be tackled with Data Science and its importance, along with Machine Learning, cannot be overstated. This project intends to illustrate the modelling of a data set using machine learning with Credit Card Fraud Detection. The Credit Card Fraud Detection Problem includes modelling past credit card transactions with the data of the ones that turned out to be fraud. This model is then used to recognize whether a new transaction is fraudulent or not. Our objective here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications. Credit Card Fraud Detection is a typical sample of classification. In this process, we have focused on analysing and pre-processing data sets as well as the deployment of multiple anomaly detection algorithms such as Local Outlier Factor and Isolation Forest algorithm on the PCA transformed Credit Card Transaction data.

Keywords Credit card fraud, applications of machine learning, data science, isolation forest algorithm, local outlier factor, automated fraud detection.

# LIST OF FIGURES

# CONTENTS

19. Dataset Analysis and Loading.
20. Class Wise Analysis.
21. Data Modelling.
22. Feature Selection and Data Split.
23. Model Training.
24. Model Evaluation.
25. Webapp of Model and coding.
26. Project Deployment .
27. Screenshots of results.

# INTRODUCTION

## Overview of Project

'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 behaviour of such fraudulent practices can be studied to minimize it and protect against similar occurrences in the future.In other words, Credit Card Fraud can be defined as a case where a person uses someone elses 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 behaviour, which consist of fraud, intrusion, and defaulting.

This is a very relevant problem that demands the attention of communities such as machine learning and data science where the solution to this problem can be automated.

This problem is particularly challenging from the perspective of learning, as it is characterized by various factors such as class imbalance. The number of valid transactions far outnumber fraudulent ones. Also, the transaction patterns often change their statistical properties over the course of time.

These are not the only challenges in the implementation of a real-world fraud detection system, however. In real world examples, the massive stream of payment requests is quickly scanned by automatic tools that determine which transactions to authorize.

Machine learning algorithms are employed to analyse all the authorized transactions and report the suspicious ones. These reports are investigated by professionals who contact the cardholders to confirm if the transaction was genuine or fraudulent.

The investigators provide a feedback to the automated system which is used to train and update the algorithm to eventually improve the fraud-detection performance over time.

Fraud detection methods are continuously developed to defend criminals in adapting to their fraudulent strategies.

## OBJECTIVE:

The key objective of any credit card fraud detection system is to identify suspicious events and report them to an analyst while letting normal transactions be automatically processed.

For years, financial institutions have been entrusting this task to rule-based systems that employ rule sets written by experts. But now they increasingly turn to a machine learning approach, as it can bring significant improvements to the process.

## PROBLEM STATEMENT

- The Credit Card Fraud Detection Problem includes modeling past credit card transactions with the knowledge of the ones that turned out to be a fraud.
- This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

## SOLUTION:

1. The model used must be simple and fast enough to detect the anomaly and classify it as a fraudulent transaction as quickly as possible.
2. Imbalance can be dealt with by properly using some methods which we will talk about in the next paragraph
3. For protecting the privacy of the user the dimensionality of the data can be reduced.
4. A more trustworthy source must be taken which double-check the data, at least for training the model.
5. 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.

# Detailed Project Profile

The credit card fraud detection features uses user behavior and location scanning to check for unusual patterns. These patterns include user characteristics such as user spending patterns as well as usual user geographic locations to verify his identity. If any unusual pattern is detected, the system requires revivification.
The system analyses user credit card data for various characteristics. These characteristics include user country, usual spending procedures. Based upon previous data of that user the system recognizes unusual patterns in the payment procedure. So now the system may require the user to login again or even block the user for more than 3 invalid attempts.

**Core Features:**

- The system stores previous transaction patterns for each user.
- Based upon the user spending ability and even country, it calculates user's characteristics.
- More than 20 -30 %deviation of users transaction(spending history and operating country) is considered as an invalid attempt and system takes action.

## CHALLENGES:

1. Enormous Data is processed every day and the model build must be fast enough to respond to the scam in time.
2. Imbalanced Data i.e most of the transactions *(99.8%)* are not fraudulent which makes it really hard for detecting the fraudulent ones
3. Data availability as the data is mostly private.

4. Misclassified Data can be another major issue, as not every fraudulent transaction is caught and reported.
5. Adaptive techniques used against the model by the scammers.

## Features of our project:-

1. Features capturing properties of transactions, including: cards used, identifying email addresses used, and location
2. Features pertaining to customer behaviour, including: frequency of orders, how the customer navigates a page before placing an order, and time elapsed between orders
3. Typical models only use raw transactional features, such as time, amount, place of the transaction.
4. The computation of the ag-gregated features consists in grouping the transactions made during the last given number of hours, first by card or account number, then by transaction type, merchant group, transaction id followed by calculating the number of transactions or the total amount spent on those transactions.
5. Finally, when calculating the periodic features, it is important to use longer time frames

# Software Requirement Specification

## PURPOSE

- Creating a web app to detect fraud Credit Cards.
- Implementing Support Vector Machine.
- Creating database containing all relevant information of Customer.
- Providing security to the customers at the time of transaction.

## SCOPE

The system prevents fraudulent users from misusing the details of the credit-card of the genuine users for their personal gain. The spending habits of the credit-card owner is detect the fraud. As the fake user might not be aware of the spending habits of the owner, there will be an irregularity in the spending pattern, which the system will detect. The owner is immediately alerted about the attempted fraud and the transaction is blocked. Thus, the system protects legitimate users from financial loss. The system helps in making electronic payment safer and more reliable. The principles in the proposed system can also be adopted and implemented in other electronic payment services such as online banking facility and payment gateways.

# SOFTWARE REQUIREMENT

## PREREQUISITES

- Python 3.7 in jupyter is used for data pre-processing, model training and prediction.
- Operating System: Windows 7 and above or Linux based OS or MAC OS
- Jupyter notebook
- Spyder
- Flask API

## DEPENDENCIES , TOOLS AND LIBRARIES

- Python – 3.x
- Numpy – 1.19.2
- Scikit-learn – 0.24.1
- Matplotlib – 3.3.4
- Imblearn – 0.8.0
- Collections, Itertools

# PROPOSED SYSTEM

The aim of the proposed system is to develop a website which has capability to restrict and block the transaction performing by attacker from genuine user's credit card details. The system here is developed for the transactions higher than the customers current transaction limit. As we seen the existing system detects the fraud after fraud has been occurred [1, 2] i.e. based on customers complained.

The proposed system tries to detect fraudulent transaction before transaction succeed .

1) In proposed system, while registration we take required information which is efficient to detect fraudulent user activity.

2) In proposed system we are using Support Vector Machine which works on transaction behavior of user. By Using SVM, after certain transactions we find one threshold value by using this threshold value we can compare current transaction with threshold value if transaction is quite different from user behavior then check whether it is genuine or fraud.

3)SVM working is quite good after certain transactions i.e. after 10 transactions .So SVM get failed when the transaction is users 1st or less than10 so to overcome this disadvantaged we take limit from user to protect first 10 transactions from fraudulent user.

## GENERAL DESCRIPTIONS

The credit card fraud detection system has been developed to alert the customer regarding the fraud of their credit card. After the payment processthe transactions performed is verified whether the performed transaction is real or fraud transaction and minimizes the false alert by implementing genetic algorithm.

### PRODUCT FUNCTION:

The project is guaranteed to provide reliable results and the functionality of the product to detect the fraud transactions effectively and provide flexibility to the user in a secured and accurate manner.

**USER CHARACTERISTICS:**

The user of the system are classified as customers and administrator,

1.Customers are those who make the transaction through any means.

2.Administrator who computes on the transaction and reports about the fraud usage

**GENERAL CONSTRAINTS:**

1.Hardware Limitations: There are no hardware limitations.

2.Interfaces to other Applications: There shall be no interfaces.

3. Parallel Operations: There are parallel operations.

4.Audit Functions: There shall be no audit functions.

5.Control Functions: There shall be no control functions

# INTERFACE REQUIREMENTS

The system performance is adequate. However, Virtual travel agency is working with the user internet connection, 60% of the performance is up to the client side.

1. **HARDWARE REQUIREMENTS**

 Processor type : Pentium III-compatible processor or faster.

 Processor speed : Minimum: 1.0 GHz, Recommended: 2.0 GHz or faster

 RAM : 512 MB or more

 HARD DISK : 20GB or more

 Monitor : VGA or higher resolution 800x600 or higher resolution

 Pointing device : Microsoft Mouse or compatible pointing device

 CD-ROM : Actual requirements will vary based on system configuration and the applications and features chosen to install.

**SOFTWARE REQUIREMENTS**

 Application software Framework : Python , ML

 Back End : APIs

 Operating System : Windows XP Professional or more

## PERFORMANCE REQUIREMENTS :

- The project has the following performance requirements
- The prime requirement is that no error condition causes a project to exit abruptly.
- Any error occurred in any process should return an understandable error message
- The response should be fairly fast, the action participants should not be confused at any point of time about action that is happening.
- The system performance is adequate.

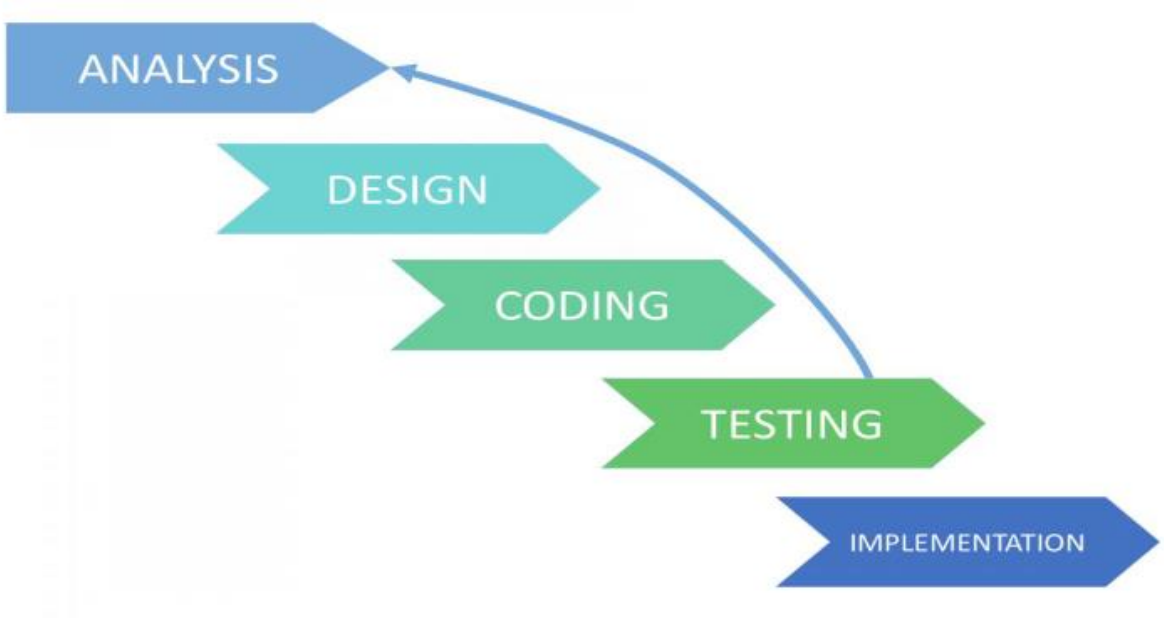
## Modules with Sample Code

- Data Loading
- Class wise Analysis
- Data Modelling
- Model Training
- Model Evaluation
- Web App

# SOFTWARE PROCESS MODEL USED

## Iterative Model

An iterative life cycle model does not attempt to start with a full specification of requirements by first focusing on an initial, simplified set user features, which then progressively gains more complexity and a broader set of features until the targeted system is complete. When adopting the iterative approach, the philosophy of incremental development will also often be used liberally and interchangeably.




Iterative Model

## Advantages of Iterative Model:-

Some working functionality can be developed and early in the software development life cycle
(SDLC).

- It is easily adaptable to the ever changing needs of the project as well as the client.
- It is best suited for agile organizations.
- It is more cost effective to change the scope or requirements in Iterative model.
- Parallel development can be planned.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed.
- In iterative model less time is spent on documenting and more time is given for designing.

## Disadvantages of Iterative Model:

- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- It is not suitable for smaller projects.

# SYSTEM DOCUMENTATION

## DATA FLOW DIAGRAMS

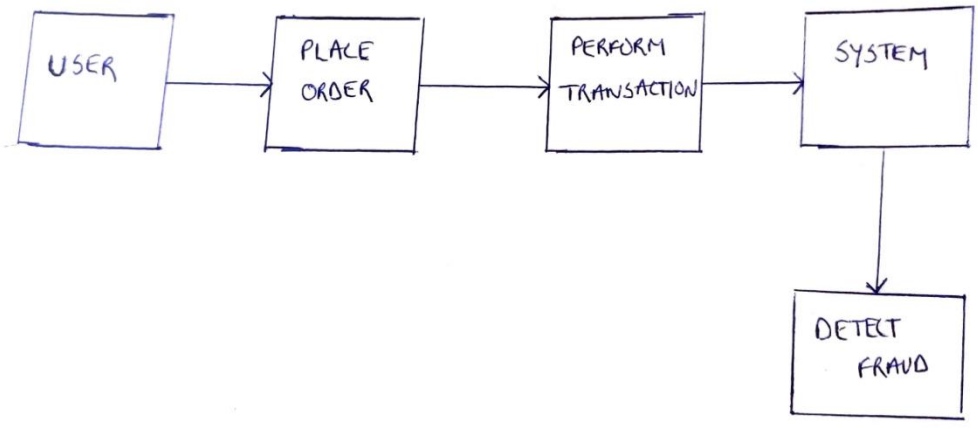



**FIG.NO 1 ZERO LEVEL DFD**

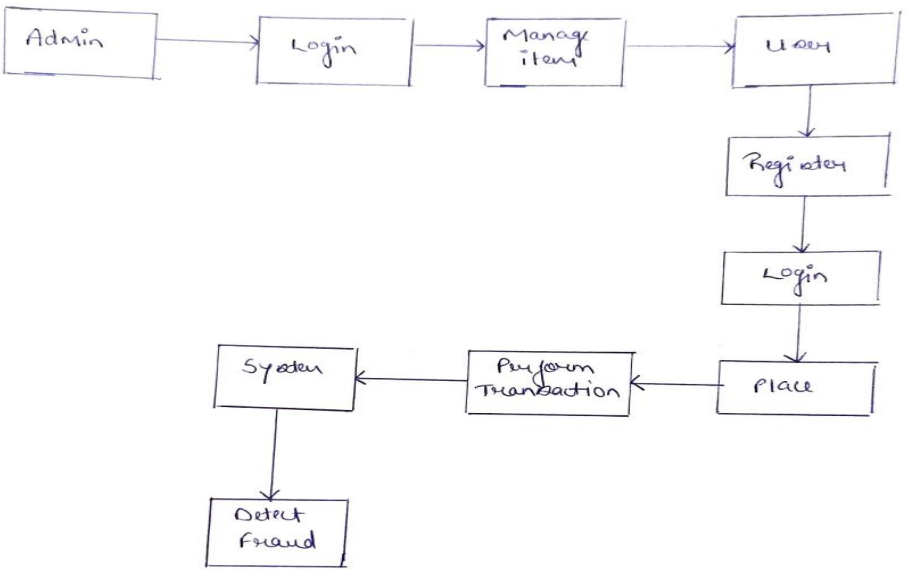



**FIG NO.2 FIRST LEVEL DFD**

## SYSTEM FLOW CHART

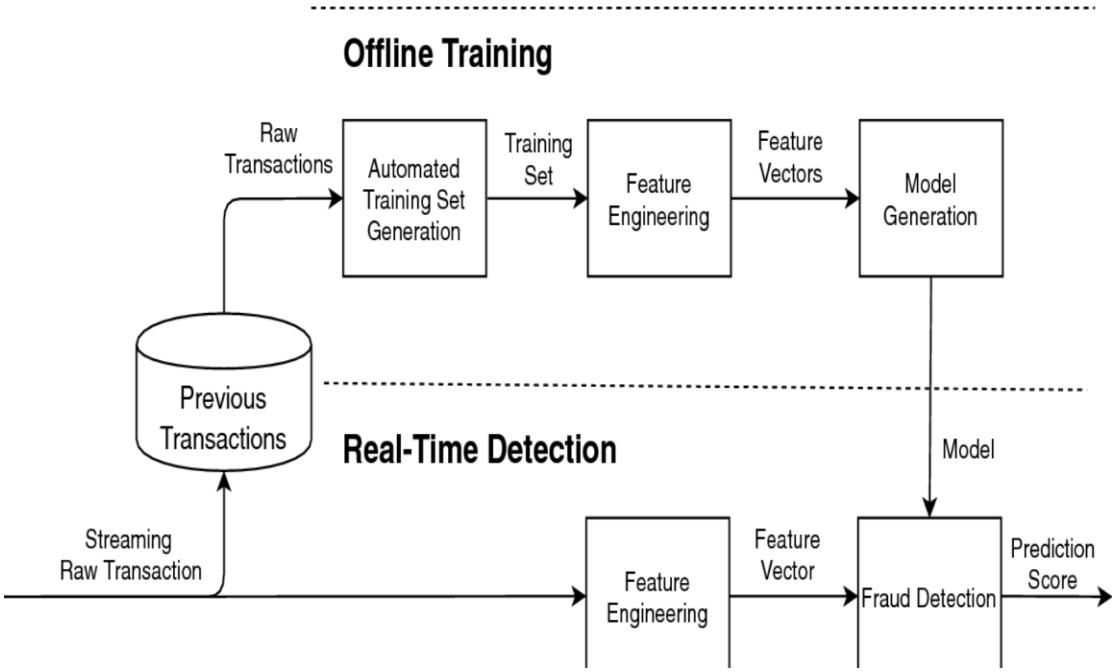


**FIG.NO.4 SYSTEM FLOW CHART**

# ER DIAGRAM

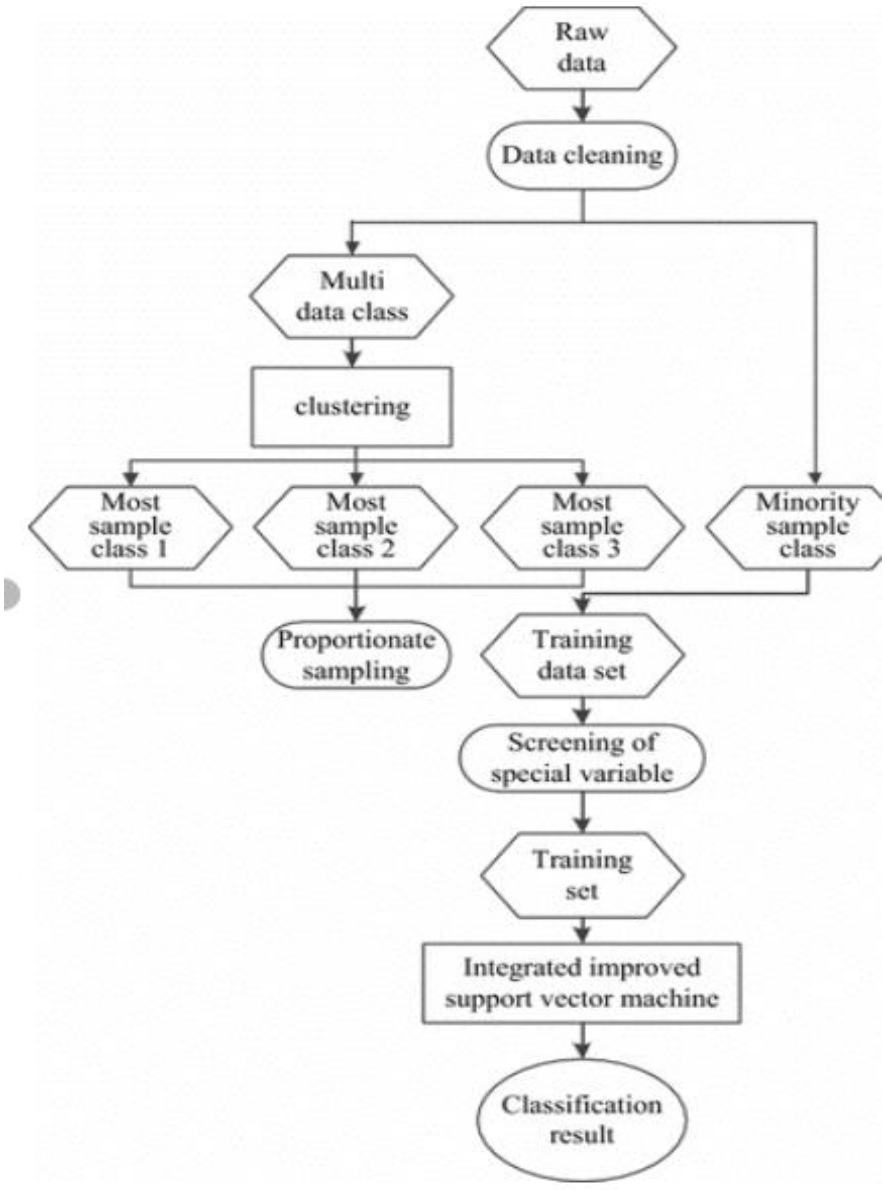



**FIG.NO-5 ER DIAGRAM**

## DATA DICTIONARY

Summary of typical raw credit card fraud detection feature

| Attribute name | Description |
|---|---|
| Transaction ID | Transaction identification number |
| Time | Date and time of the transaction |
| Account number | Identification number of the customer |
| Card numbe r | Identification of the credit card |
| Transaction type | ie. Internet, ATM, POS, ... |
| Amount | Amount of the transaction in Euros |
| Type of card | ie . Visa debit, Mastercard, American Express... |
| Location | location of the card holder |
| Bank | Issuer bank of the card |

# SCREEN LAYOUTS AND DESCRIPTIONS

# DATASET

## DataSet Analysis:

The dataset contains an European cardholder transactions that occurred in two days with 284,807 transactions from September 2013. The dataset was obtained from Kaggle. It contains 28 attributes,which have been scaled and modified.However theirdescription has not been given.
The only attributes known to us are
1.Amount
2.Time
3.Output class[ 0 for a normal transaction and 1 for a fraudulent transaction]

The total dataset has 284807 rows and 31 columns. Out of these, the normal transactionswere 284315 and fraudulent transactions were 492.

**Sample Data:** These are the 2 sample transactions from the dataset.
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

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -1.35981 | -0.07278 | 2.53635 | 1.37816 | -0.33832 | 0.46239 | 0.2396 | 0.0987 | 0.36379 | 0.09079 | -0.5516 | -0.6178 | -0.99139 | -0.31117 | 1.46818 | -0.4704 | 0.20797 | 0.02579 | 0.40399 | 0.25141 | -0.01831 | 0.27784 | -0.11047 | 0.06693 | 0.12854 | -0.18911 | 0.13356 | -0.02105 | 149.62 | 0 |
| 0 | 1.19186 | 0.26615 | 0.16648 | 0.44815 | 0.06002 | -0.08236 | -0.0788 | 0.0851 | -0.25543 | -0.16697 | 1.61273 | 1.06524 | 0.4891 | -0.14377 | 0.63556 | 0.46392 | -0.1148 | -0.18336 | -0.14578 | -0.06908 | -0.22578 | -0.63867 | 0.10129 | -0.33985 | 0.16717 | 0.12589 | -0.00898 | 0.01472 | 2.69 | 0 |
| 1 | -1.35835 | -1.34016 | 1.77321 | 0.37978 | -0.5032 | 1.8005 | 0.79146 | 0.24768 | -1.51465 | 0.20764 | 0.6245 | 0.06608 | 0.71729 | -0.16595 | 2.34586 | -2.89008 | 1.10997 | -0.12136 | -2.26186 | 0.52498 | 0.248 | 0.77168 | 0.90941 | -0.68928 | -0.32764 | -0.1391 | -0.05535 | -0.05975 | 378.66 | 0 |
| 1 | -0.96627 | -0.18523 | 1.79299 | -0.86329 | -0.01031 | 1.2472 | 0.23761 | 0.37744 | -1.38702 | -0.05495 | -0.22649 | 0.17823 | 0.50776 | -0.28792 | -0.63142 | -1.05965 | -0.68409 | 1.96578 | -1.23262 | -0.20804 | -0.1083 | 0.00527 | -0.19032 | -1.17558 | 0.64738 | -0.22193 | 0.06272 | 0.06146 | 123.5 | 0 |
| 2 | -1.15823 | 0.87774 | 1.54872 | 0.40303 | -0.40719 | 0.09592 | 0.59294 | -0.27053 | 0.81774 | 0.75307 | -0.82284 | 0.5382 | 1.34585 | -1.11967 | 0.17512 | -0.45145 | -0.23703 | -0.03819 | 0.80349 | 0.40854 | -0.00943 | 0.79828 | -0.13746 | 0.14127 | -0.20601 | 0.50229 | 0.21942 | 0.21515 | 69.99 | 0 |
| 2 | -0.42597 | 0.96052 | 1.14111 | -0.16825 | 0.42099 | -0.02973 | 0.4762 | 0.26031 | -0.56867 | -0.37141 | 1.34126 | 0.35989 | -0.35809 | -0.13713 | 0.51762 | 0.40173 | -0.05813 | 0.06865 | -0.03319 | 0.08497 | -0.20825 | -0.55982 | -0.0264 | -0.37143 | -0.23279 | 0.10591 | 0.25384 | 0.08108 | 3.67 | 0 |
| 4 | 1.22966 | 0.141 | 0.04537 | 1.20261 | 0.19188 | 0.27271 | -0.00516 | 0.08121 | 0.46496 | -0.09925 | -1.41691 | -0.15383 | -0.75106 | 0.16737 | 0.05014 | -0.44359 | 0.00282 | -0.61199 | -0.04558 | -0.21963 | -0.16772 | -0.27071 | -0.1541 | -0.78006 | 0.75014 | -0.25724 | 0.03451 | 0.00517 | 4.99 | 0 |
| 7 | -0.64427 | 1.41796 | 1.07438 | -0.4922 | 0.94893 | 0.42812 | 1.12063 | -3.80786 | 0.61537 | 1.24938 | -0.61947 | 0.29147 | 1.75796 | -1.32387 | 0.68613 | -0.07613 | -1.22213 | -0.35822 | 0.3245 | -0.15674 | 1.94347 | -1.01545 | 0.0575 | -0.64971 | -0.41527 | -0.05163 | -1.20692 | -1.08534 | 40.8 | 0 |
| 7 | -0.89429 | 0.28616 | -0.11319 | -0.27153 | 2.6696 | 3.72182 | 0.37015 | 0.85108 | -0.39205 | -0.41043 | -0.70512 | -0.11045 | -0.28625 | 0.07436 | -0.32878 | -0.21008 | -0.49977 | 0.11876 | 0.57033 | 0.05274 | -0.07343 | -0.26809 | -0.20423 | 1.01159 | 0.3732 | -0.38416 | 0.01175 | 0.1424 | 93.2 | 0 |
| 9 | -0.33826 | 1.11959 | 1.04437 | -0.22219 | 0.49936 | -0.24676 | 0.65158 | 0.06954 | -0.73673 | -0.36685 | 1.01761 | 0.83639 | 1.00684 | -0.44352 | 0.15022 | 0.73945 | -0.54098 | 0.47668 | 0.45177 | 0.20371 | -0.24691 | -0.63375 | -0.12079 | -0.38505 | -0.06973 | 0.0942 | 0.24622 | 0.08308 | 3.68 | 0 |
| 10 | 1.44904 | -1.17634 | 0.91386 | -1.37567 | -1.97138 | -0.62915 | -1.42324 | 0.04846 | -1.72041 | 1.62666 | 1.19964 | -0.67144 | -0.51395 | -0.09505 | 0.23093 | 0.03197 | 0.25341 | 0.85434 | -0.22137 | -0.38723 | -0.0093 | 0.31389 | 0.02774 | 0.50051 | 0.25137 | -0.12948 | 0.04285 | 0.01625 | 7.8 | 0 |
| 10 | 0.38498 | 0.61611 | -0.8743 | -0.09402 | 2.92458 | 3.31703 | 0.47045 | 0.53825 | -0.55889 | 0.30976 | -0.25912 | -0.32614 | -0.09005 | 0.36283 | 0.9289 | -0.12949 | -0.80998 | 0.35999 | 0.70766 | 0.12599 | 0.04992 | 0.23842 | 0.00913 | 0.99671 | -0.76731 | -0.49221 | 0.04247 | -0.05434 | 9.99 | 0 |
| 10 | 1.25 | -1.22164 | 0.38393 | -1.2349 | -1.48542 | -0.75323 | -0.6894 | -0.22749 | -2.09401 | 1.32373 | 0.22767 | -0.24268 | 1.20542 | -0.31763 | 0.72567 | -0.81561 | 0.87394 | -0.84779 | -0.68319 | -0.10276 | -0.23181 | -0.48329 | 0.08467 | 0.39283 | 0.16113 | -0.35499 | 0.02642 | 0.04242 | 121.5 | 0 |
| 11 | 1.06937 | 0.28772 | 0.82861 | 2.71252 | -0.1784 | 0.33754 | -0.09672 | 0.11598 | -0.22108 | 0.46023 | -0.77366 | 0.32339 | -0.01108 | -0.17849 | -0.65556 | -0.19993 | 0.12401 | -0.9805 | -0.98292 | -0.1532 | -0.03688 | 0.07441 | -0.07141 | 0.10474 | 0.54826 | 0.10409 | 0.02149 | 0.02129 | 27.5 | 0 |
| 12 | -2.79185 | -0.32777 | 1.64175 | 1.76747 | -0.13659 | 0.8076 | -0.42291 | -1.90711 | 0.75571 | 1.15109 | 0.84456 | 0.79294 | 0.37045 | -0.73498 | 0.4068 | -0.30306 | -0.15587 | 0.77827 | 2.22187 | -1.58212 | 1.15166 | 0.22218 | 1.02059 | 0.02832 | -0.23275 | -0.23556 | -0.16478 | -0.03015 | 58.8 | 0 |
| 12 | -0.75242 | 0.34549 | 2.05732 | -1.46864 | -1.15839 | -0.07785 | -0.60858 | 0.0036 | -0.43617 | 0.74773 | -0.79398 | -0.77041 | 1.04763 | -1.0666 | 1.10695 | 1.66011 | -0.27927 | -0.41999 | 0.43254 | 0.26345 | 0.49962 | 1.35365 | -0.25657 | -0.06508 | -0.03912 | -0.08709 | -0.181 | 0.12939 | 15.99 | 0 |
| 12 | 1.10322 | -0.0403 | 1.26733 | 1.28909 | -0.736 | 0.28807 | -0.58606 | 0.18938 | 0.78233 | -0.26798 | -0.45031 | 0.93671 | 0.70838 | -0.46865 | 0.35457 | -0.24663 | -0.00921 | -0.59591 | -0.57568 | -0.11391 | -0.02461 | 0.196 | 0.0138 | 0.10376 | 0.3643 | -0.38226 | 0.09281 | 0.03705 | 12.99 | 0 |
| 13 | -0.43691 | 0.91897 | 0.92459 | -0.72722 | 0.91568 | -0.12787 | 0.70764 | 0.08796 | -0.66527 | -0.73798 | 0.3241 | 0.27719 | 0.25262 | -0.2919 | -0.18452 | 1.14317 | -0.92871 | 0.68047 | 0.02544 | -0.04702 | -0.1948 | -0.67264 | -0.15686 | -0.88839 | -0.34241 | -0.04903 | 0.07969 | 0.13102 | 0.89 | 0 |
| 14 | -5.40126 | -5.45015 | 1.1863 | 1.73624 | 3.04911 | -1.76341 | -1.55974 | 0.16084 | 1.23309 | 0.34517 | 0.91723 | 0.97012 | -0.26657 | -0.47913 | -0.52661 | 0.472 | -0.72548 | 0.07508 | -0.40687 | -2.19685 | -0.5036 | 0.98446 | 2.45859 | 0.04212 | -0.48163 | -0.62127 | 0.39205 | 0.94959 | 46.8 | 0 |
| 15 | 1.49294 | -1.02935 | 0.45479 | -1.43803 | -1.55543 | -0.72096 | -1.08066 | -0.05313 | -1.97868 | 1.63808 | 1.07754 | -0.63205 | -0.41696 | 0.05201 | -0.04298 | -0.16643 | 0.30424 | 0.55443 | 0.05423 | -0.38791 | -0.17765 | -0.17507 | 0.04 | 0.29581 | 0.33293 | -0.22038 | 0.0223 | 0.0076 | 5 | 0 |
| 16 | 0.69488 | -1.36182 | 1.02922 | 0.83416 | -1.19121 | 1.30911 | -0.87859 | 0.44529 | -0.4462 | 0.56852 | 1.01915 | 1.29833 | 0.42048 | -0.37265 | -0.80798 | -2.04456 | 0.51566 | 0.62585 | -1.30041 | -0.13833 | -0.29558 | -0.57196 | -0.05088 | -0.30421 | 0.072 | -0.42223 | 0.08655 | 0.0635 | 231.71 | 0 |
| 17 | 0.9625 | 0.32846 | -0.17148 | 2.1092 | 1.12957 | 1.69604 | 0.10771 | 0.5215 | -1.19131 | 0.7244 | 1.69033 | 0.40677 | -0.93642 | 0.98374 | 0.71091 | -0.60223 | 0.40248 | -1.73716 | -2.02761 | -0.26932 | 0.144 | 0.40249 | -0.04851 | -1.37187 | 0.39081 | 0.19996 | 0.01637 | -0.01461 | 34.09 | 0 |
| 18 | 1.16662 | 0.50212 | -0.0673 | 2.26157 | 0.4288 | 0.08947 | 0.24115 | 0.13808 | -0.98916 | 0.92217 | 0.74479 | -0.53138 | -2.10535 | 1.12687 | 0.00308 | 0.42442 | -0.45448 | -0.09887 | -0.8166 | -0.30717 | 0.0187 | -0.06197 | -0.10385 | -0.37042 | 0.6032 | 0.10856 | -0.04052 | -0.01142 | 2.28 | 0 |
| 18 | 0.24749 | 0.27767 | 1.18547 | -0.0926 | -1.31439 | -0.15012 | -0.94636 | -1.61794 | 1.54407 | -0.82988 | -0.5832 | 0.52493 | -0.45338 | 0.08139 | 1.5552 | -1.39689 | 0.78313 | 0.43662 | 2.17781 | -0.23098 | 1.65018 | 0.20045 | -0.18535 | 0.42307 | 0.82059 | -0.22763 | 0.33663 | 0.25048 | 22.75 | 0 |
| 22 | -1.94653 | -0.0449 | -0.40557 | -1.01306 | 2.94197 | 2.95505 | -0.06306 | 0.85555 | 0.04997 | 0.57374 | -0.08126 | -0.21575 | 0.04416 | 0.0339 | 1.19072 | 0.57884 | -0.97567 | 0.04406 | 0.4886 | -0.21672 | -0.57953 | -0.79923 | 0.8703 | 0.98342 | 0.3212 | 0.14965 | 0.70752 | 0.0146 | 0.89 | 0 |
| 22 | -2.07439 | -0.11148 | 1.32202 | 0.41001 | 0.2953 | -0.95954 | 0.54899 | -0.10463 | 0.47566 | 0.14045 | -0.95657 | -0.19052 | -0.65523 | -0.2798 | -0.21167 | -0.33332 | 0.01075 | -0.48847 | 0.50575 | -0.38669 | -0.40364 | -0.2274 | 0.74243 | 0.29853 | 0.24931 | 0.2744 | 0.35997 | 0.24333 | 16.43 | 0 |

## Modules with Sample Code

- Data Loading
- Class wise Analysis
- Data Modelling
- Model Training
- Model Evaluation
- Web App

# Data Loading:

The dataset was obtained from Kaggle. It contains 28 attributes,which have been scaled and modified.However their description has not been given.

The only attributes known to us are

1.Amount

2.Time

3.Output class[ 0 for a normal transaction and 1 for a fraudulent transaction].

The dataset contains float data values for every class except the Output class which is of int.

The data from the dataset in csv format was loaded into the dataframe of Pandas Python

package.Pandas is an open-source, BSD-licensed Python library providing high-performance,

easy-to-use data structures and data analysis tools for the Python programming language.

**#Data Loading**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**# %matplotlib inline**

**# Commented out IPython magic to ensure Python compatibility.import sklearn**

**import random**

**from sklearn.utils import shuffle**

**# %matplotlib inline**

**data=pd.read_csv('creditcard.csv')**

**data.head(n=7)**

Out[3]:

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.098698 | 0 |
| **1** | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0 |
| **2** | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 | -1 |
| **3** | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1 |
| **4** | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 | 0 |
| **5** | 2.0 | -0.425966 | 0.960523 | 1.141109 | -0.168252 | 0.420987 | -0.029728 | 0.476201 | 0.260314 | -0 |
| **6** | 4.0 | 1.229658 | 0.141004 | 0.045371 | 1.202613 | 0.191881 | 0.272708 | -0.005159 | 0.081213 | 0 |

7 rows × 31 columns

# CLASS WISE ANALYSIS:

**The only non-transformed variables to work with are:**

- `Time`
- `Amount`
- `Class` (1: fraud, 0: not_fraud)

**LABELS = ["Normal", "Fraud"]**

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

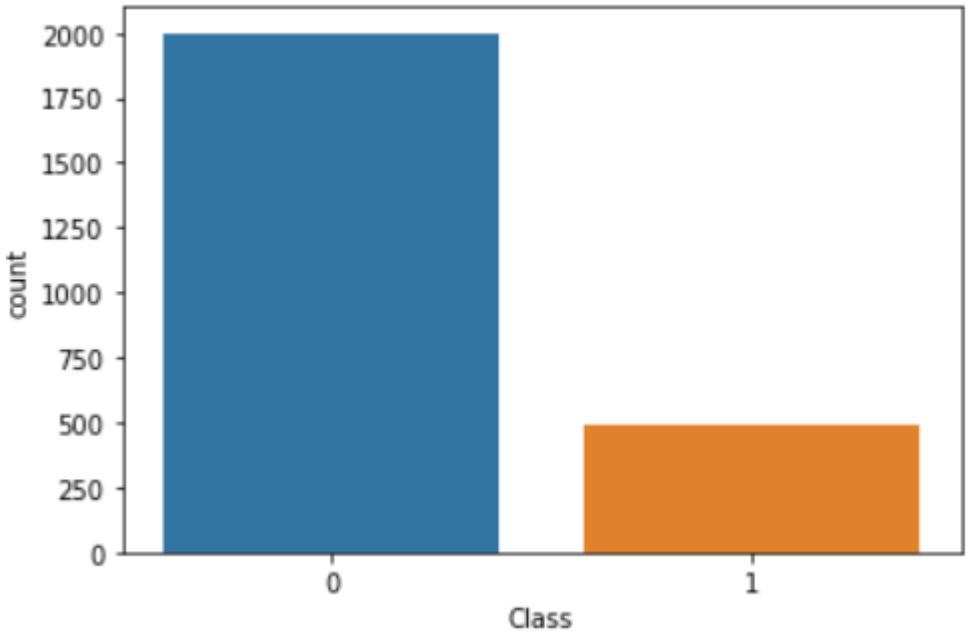
**count_classes.plot(kind = 'bar', rot=0)**

**plt.title("Transaction Class Distribution")**

**plt.xticks(range(2), LABELS)**

**plt.xlabel("Class")**

**plt.ylabel("Frequency");**



```
df.describe()
normal['Amount'].describe()
fraud['Amount'].describe()
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize = (10,10) )
f.suptitle('Amount per transaction by class')
bins = 10
ax1.hist(fraud.Amount, bins = bins)
ax1.set_title('Fraud')
ax2.hist(normal.Amount, bins = bins)
ax2.set_title('Normal')
ax1.grid()
ax2.grid()
plt.xlabel('Amount ($)')
plt.ylabel('Number of Transactions')
plt.xlim((0, 20000))
plt.yscale('log')
plt.show();
```

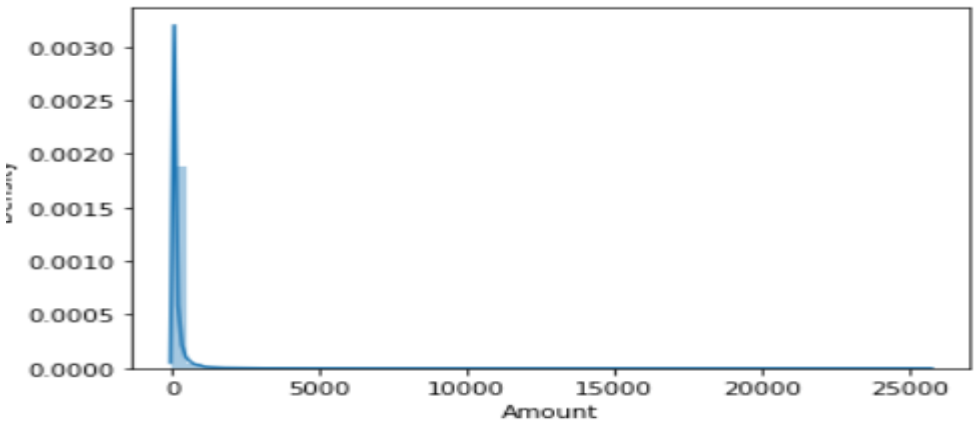



**FIG NO.6 AMOUNT OF TRANSACTION**

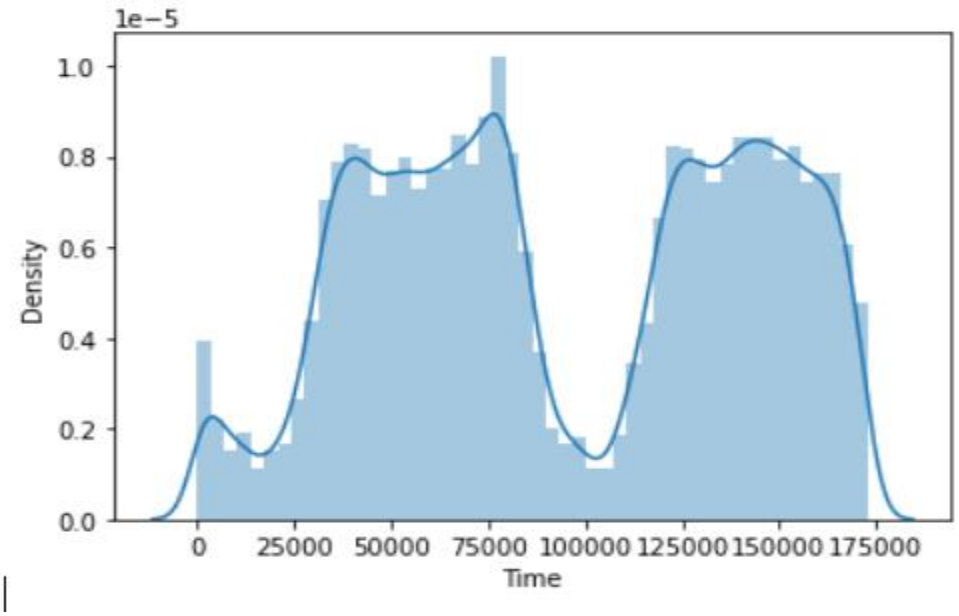



**FIG.NO.7 TIME OF TRANSACTIONS(IN sec)**

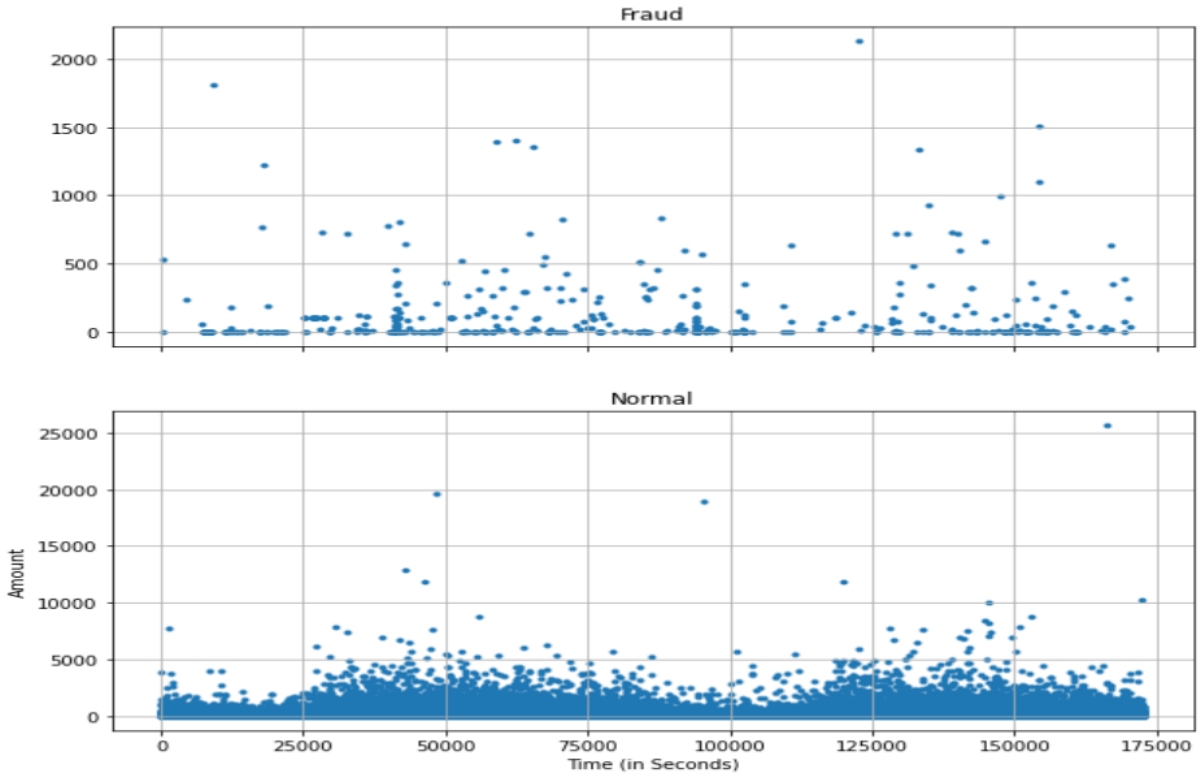



**Fig 8.The relation between time of transaction versus amount by fraud and normal class.**

```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(10,10))
f.suptitle('Time of transaction vs Amount by class')
45
ax1.scatter(fraud.Time, fraud.Amount, marker='.')
ax1.set_title('Fraud')
ax1.grid()
ax2.scatter(normal.Time, normal.Amount, marker='.')
ax2.set_title('Normal')
ax2.grid()
plt.xlabel('Time (in Seconds)')
plt.ylabel('Amount')
plt.show()
```

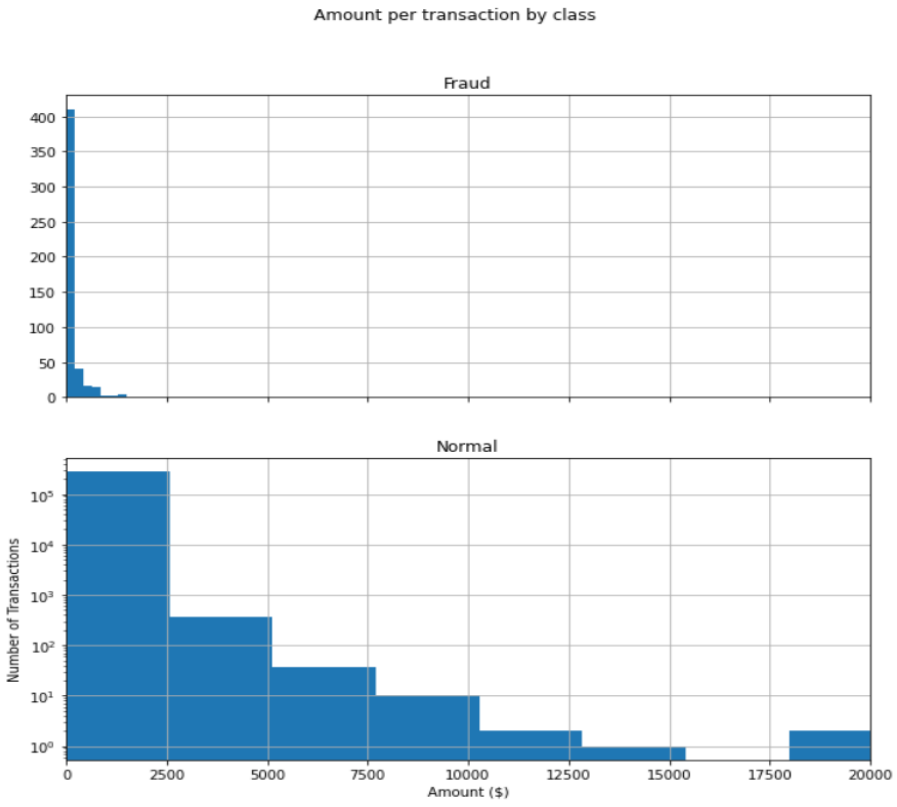



**Fig 14. The amount per transaction by fraud and normal class.**

**data.hist(figsize=(20,20))**

**plt.show()**

By seeing the distributions we can have an idea how skewed are these features, we can also see further distributions of the other features. There are techniques that can help the distributions be less skewed which will be implemented in this notebook in the future.

Doesn't seem like the time of transaction really matters here as per above observation. Now let us take a sample of the dataset for out modelling and prediction

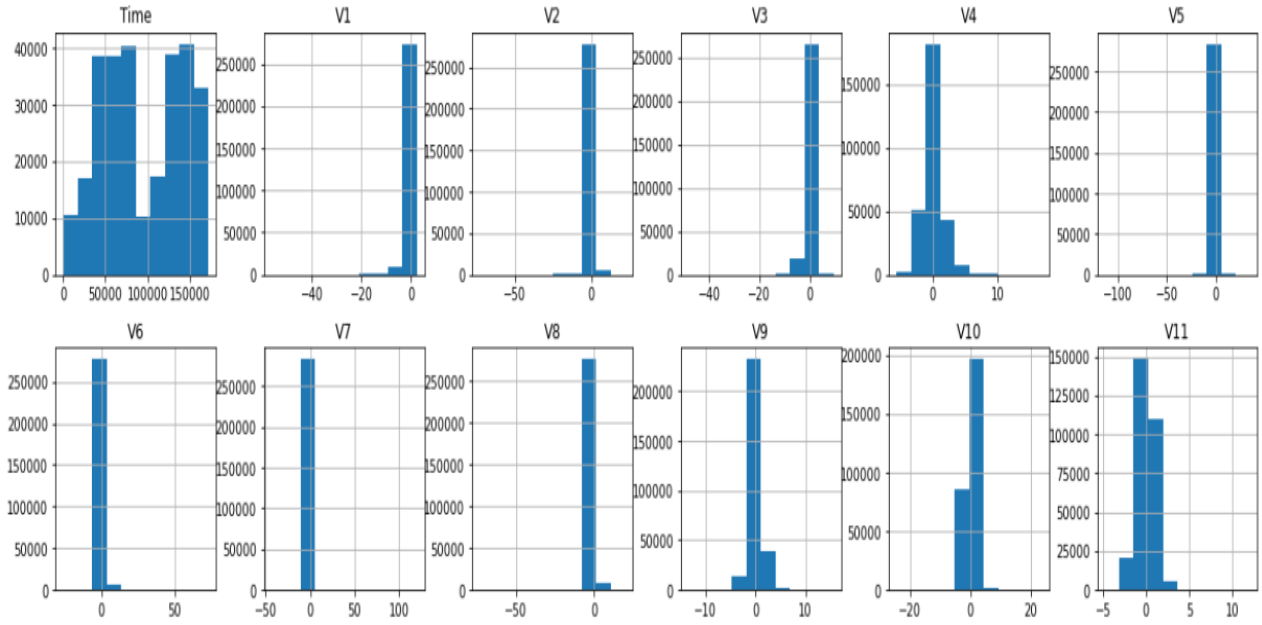



**FIG.NO-10**

**# heatmap to find any high correlations**

**plt.figure(figsize=(10,10))**
**sns.heatmap(data=data.corr(), cmap="seismic")**
**plt.show();**

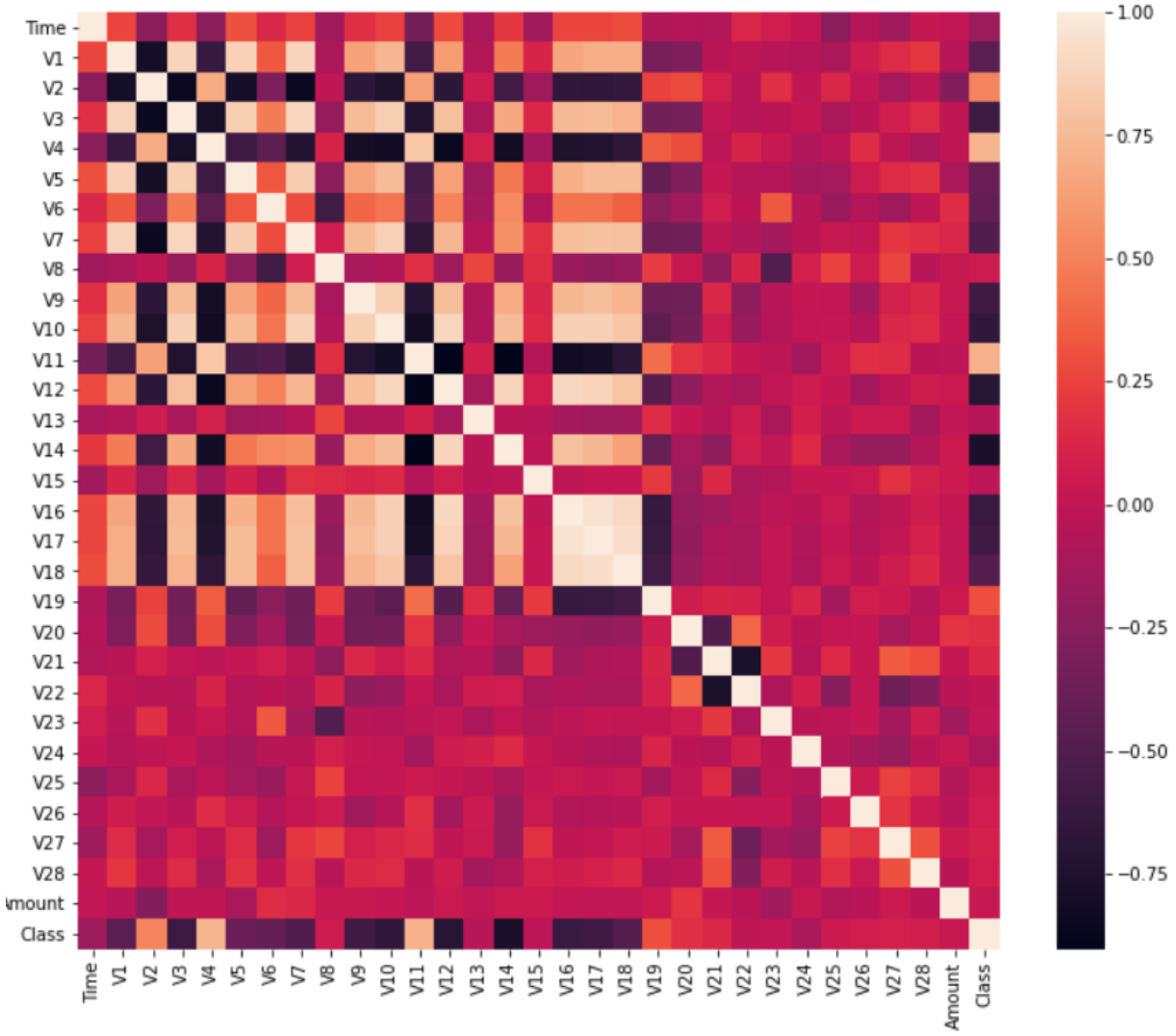



FIG.NO 11

## Data Modelling

1. Removal of the “Time” attribute since it has no contribution towards the prediction of The class.

2. Division of train and test data in the existing dataset , with 80% training data and 20%testing data .

```
import math
import sklearn.preprocessing
```

```
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score , classification_report,
confusion_matrix, precision_recall_curve, f1_score, auc
```

## Feature Selection & Data Split

In this process, we are going to define the independent (X) and the dependent variables (Y). Using the defined variables, we will split the data into a training set and testing set which is further used for modeling and evaluating. We can split the data easily using the 'train_test_split' algorithm in python.

```
X_train, X_test, y_train, y_test = train_test_split(data.drop('Class', axis=1),
data['Class'], test_size=0.3, random_state=42)

"""# Feature Scaling"""

cols= ['V22', 'V24', 'V25', 'V26', 'V27', 'V28']

scaler = StandardScaler()

frames= ['Time', 'Amount']

x= data[frames]

d_temp = data.drop(frames, axis=1)

temp_col=scaler.fit_transform(x)

scaled_col = pd.DataFrame(temp_col, columns=frames)

scaled_col.head()

d_scaled = pd.concat([scaled_col, d_temp], axis =1)

d_scaled.head()
```

y = data['Class']

d_scaled.head(n=9)

Out[50]:

| | Time | Amount | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.596428 | -0.165015 | 0.949241 | 1.333519 | -4.855402 | 1.835006 | -1.053245 | -2.562826 | -2.286986 | 0.260906 | ... |
| **1** | -0.780074 | -0.360042 | 1.087114 | -0.411833 | 1.151477 | 1.316336 | -0.318704 | 2.026130 | -1.064364 | 0.610577 | ... |
| **2** | 1.049370 | -0.436357 | 0.674654 | 0.676751 | -0.712336 | 0.027890 | -0.170944 | -0.738997 | -0.706108 | -2.896737 | ... |
| **3** | -0.571625 | -0.298142 | -1.642073 | 1.301656 | 1.680965 | 1.602124 | -0.896999 | 0.838941 | -0.359057 | 0.955273 | ... |
| **4** | 1.137317 | -0.435128 | 2.014347 | -0.043837 | -0.989070 | 0.314874 | -0.065749 | -0.925282 | 0.132583 | -0.235218 | ... |
| **5** | 1.573689 | -0.249767 | 1.934031 | -0.982692 | -0.702077 | -1.544121 | -0.806132 | -0.261569 | -0.800494 | 0.102993 | ... |
| **6** | 0.872932 | -0.423977 | 2.125515 | 0.384835 | -2.555122 | 0.441350 | 1.003939 | -1.281697 | 0.782023 | -0.514027 | ... |
| **7** | -1.652212 | -0.436357 | 0.314597 | 2.660670 | -5.920037 | 4.522500 | -2.315027 | -2.278352 | -4.684054 | 1.202270 | ... |
| **8** | 1.239849 | 0.287956 | 1.595819 | -0.718612 | -2.292155 | 0.068588 | 1.019654 | 1.309452 | -0.117545 | 0.416642 | ... |

"

"""# Dimensionality Reduction"""

from sklearn.decomposition import PCA

pca = PCA(n_components=7)

X_temp_reduced = pca.fit_transform(d_scaled)

pca.explained_variance_ratio_

pca.explained_variance_

names=['Time','Amount','Transaction Method','Transaction Id','Location','Type of Card','Bank']

X_reduced= pd.DataFrame(X_temp_reduced,columns=names)

X_reduced.head()

Out[56]:

| | Time | Amount | Transaction Method | Transaction Id | Location | Type of Card | Bank |
|---|---|---|---|---|---|---|---|
| 0 | 1.626296 | -3.197111 | -3.426494 | 0.473863 | 0.488745 | -1.095003 | 1.586177 |
| 1 | -9.884008 | 0.530641 | 1.158528 | -1.471263 | 0.232515 | -0.887786 | 0.033045 |
| 2 | -9.151924 | 3.386312 | 0.661400 | -1.170874 | 1.893406 | -0.991919 | 0.470225 |
| 3 | -8.943659 | 1.110908 | 2.651971 | -1.229971 | -0.372953 | -0.549827 | -1.454921 |
| 4 | -9.782567 | 0.464092 | 1.172931 | -0.778629 | 0.303076 | -1.350770 | 1.043664 |
| 5 | -9.911596 | 0.567576 | 1.546208 | -1.686693 | -0.078754 | -1.060243 | 1.946681 |
| 6 | -9.479254 | 0.307918 | 0.654966 | 1.193742 | 0.574164 | -1.319651 | 1.391500 |
| 7 | 7.048448 | -4.479214 | -4.859355 | 1.538096 | 0.939223 | -1.760670 | 0.933230 |
| 8 | -9.457061 | 0.319898 | 0.856216 | 1.213722 | 0.393723 | -0.177186 | 2.068315 |

**Model Training:** In this step, we will be building Support Vector Machine .Even though there are many more models which we can use, these are the most popular models used for solving classification problems. All these models can be built feasibly using the algorithms provided by the scikit-learn package. Let's implement these models in python and keep it in mind that the algorithms used might take time to get implemented.

```
Y=d_scaled['Class']

new_data=pd.concat([X_reduced,Y],axis=1)

new_data.head()

new_data.shape

new_data.to_csv('finaldata.csv')

X_train, X_test, y_train, y_test= train_test_split(X_reduced, d_scaled['Class'],
test_size = 0.30, random_state = 42)

X_train.shape, X_test.shape

from sklearn.metrics import classification_report,confusion_matrix

"""# Support Vector Machine"""
```

```
from sklearn.svm import SVC
svc=SVC(kernel='rbf',probability=True)
svc.fit(X_train,y_train)
y_pred_svc=svc.predict(X_test)
y_pred_svc

type(X_test)
X_test.to_csv('testing.csv')
from sklearn.model_selection import GridSearchCV
parameters = [ {'C': [1, 10, 100, 1000], 'kernel': ['rbf'], 'gamma': [0.1, 1, 0.01,
0.0001 ,0.001]}]
grid_search = GridSearchCV(estimator = svc,
param_grid = parameters,
scoring = 'accuracy',
n_jobs = -1)
grid_search = grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
print("Best Accuracy: {:.2f} %".format(best_accuracy*100))
print("Best Parameters:", best_parameters)

svc_param=SVC(kernel='rbf',gamma=0.01,C=100,probability=True)
svc_param.fit(X_train,y_train)
```

# Model Evaluation:

There are a variety of measures for various algorithms and these measures have been developed to evaluate very different things. So there should be criteria for evaluation of various proposed methods. False Positive (FP), False Negative (FN), True Positive (TP), and

True Negative (TN) and the relation between them are quantities which are usually adopted by credit card fraud detection researchers to compare the accuracy of different approaches.

The definitions of mentioned parameters are presented below:

- FP: the false positive rate indicates the portion of the non-fraudulent transactions wrongly being classified as fraudulent transactions.
- FN: the false negative rate indicates the portion of the fraudulent transactions wrongly being classified as normal transactions.
- TP: the true positive rate represents the portion of the fraudulent transactions correctly being classified as fraudulent transactions.
- TN: the true negative rate represents the portion of the normal transactions correctly being classified as normal transactions.

Table 1shows the details of the most common formulas which are used by researchers for evaluation of their proposed methods. As can be seen in this table some researchers had used multiple formulas in order to evaluate their proposed model.

| Measure | Formula | Description |
|---|---|---|
| Accuracy (ACC)/Detection rate | TN + TP/TP + FP + FN + TN | Accuracy is the percentage of correctly classified instances. It is one the most widely used classification performance metrics |
| Precision/Hit rate | TP/TP + FP | Precision is the number of classified positive or fraudulent instances that actually are positive instances. |
| True positive rate/Sensitivity | TP/TP + FN | TP (true positive) is the number of correctly classified positive or abnormal instances. TP rate measures how well a classifier can recognize abnormal records. It is also called sensitivity measure. In the case of |
| True negative rate /Specificity | TN/TN + FP | TN (true negative) is the number of correctly classified negative or normal instances. TN rate measures how well a classifier can recognize normal records. It is also called specificity measure. |
| False positive rate (FPR) | FP/FP+TN | Ratio of credit card fraud detected incorrectly |
| ROC | True positive rate plotted against false positive rate | Relative Operating Characteristic curve, a comparison of TPR and FPR as the criterion changes |
| Cost | Cost = 100 * FN + 10 * (FP +TP) | |
| F1-measure | 2 × (Precision ×Recall)/(Precision +Recall) | Weighted average of the precision and recall |

Table 1. Evaluation criteria for credit card fraud detection

The aim of all algorithms and techniques is to minimize FP and FN rate and maximize TP
and TN rate and with a good detection rate at the same time.

**ROC CURVE**

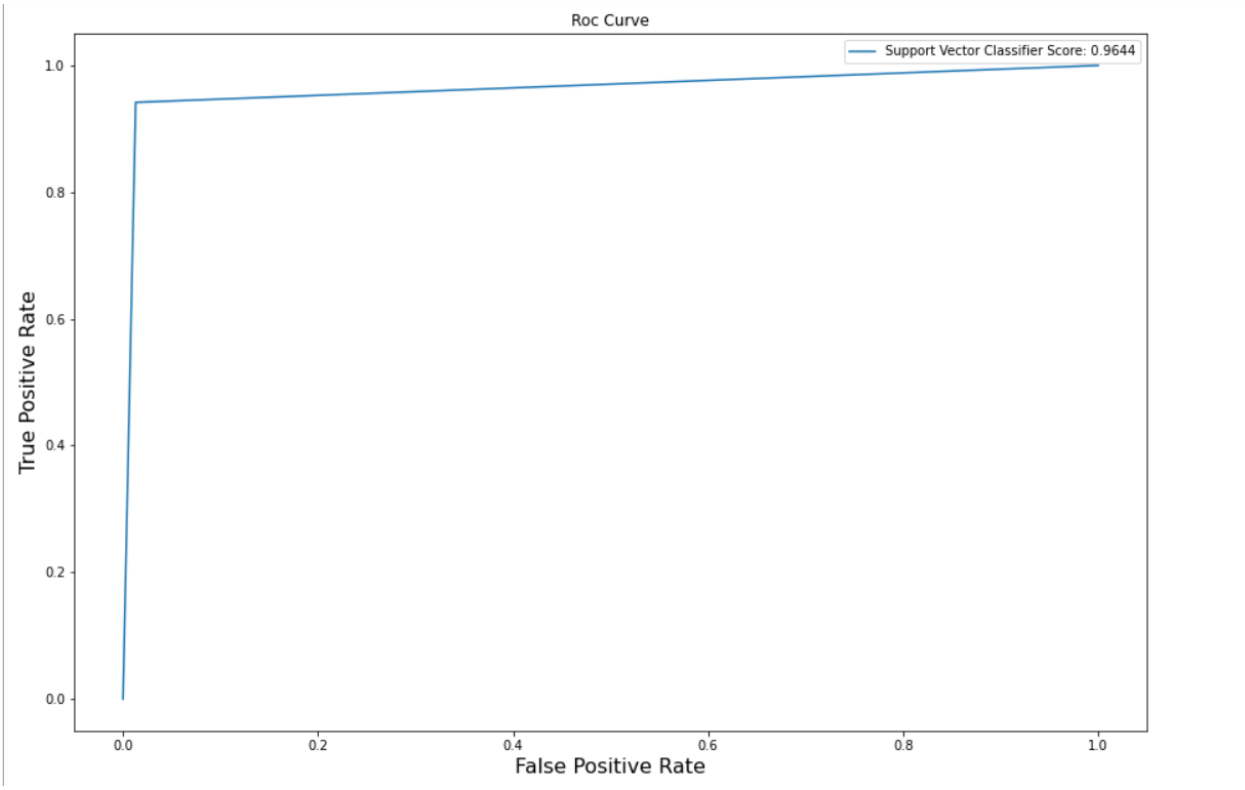



**FIG.NO-12**

## CONFUSION MATRIX:

```
[[609   8]
 [ 34 549]]
```

**FIG.NO-13**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.95 | 0.99 | 0.97 | 617 |
| 1.0 | 0.99 | 0.94 | 0.96 | 583 |
| accuracy | | | 0.96 | 1200 |
| macro avg | 0.97 | 0.96 | 0.96 | 1200 |
| weighted avg | 0.97 | 0.96 | 0.96 | 1200 |

Table 1. Classification Report for Threshold=3.1

## Web App:

We have developed a Flask Web App using the model by saving it with pickle.It contains a form to take the input of 30 attributes and predict if the given value is fraudulent or not by predicting the output with the trained model and showing the results.

requirements.txt:
Pillow==8.2.0
pyparsing==2.4.7
python-dateutil==2.8.1
pytz==2021.1
scikit-learn==0.24.2
scipy==1.6.3
seaborn==0.11.1
six==1.16.0
sklearn==0.0
threadpoolctl==2.1.0
Werkzeug==2.0.1

**app.py:**

**import numpy as np**

**from flask import Flask, request, jsonify, render_template**

```
import pickle

app = Flask(__name__)

# prediction function
def ValuePredictor(to_predict_list):
    to_predict = np.array(to_predict_list).reshape(1, 7)
    loaded_model = pickle.load(open("model.pkl", "rb"))
    result = loaded_model.predict(to_predict)
    return result[0]


@app.route('/')
def home():
    return render_template("index.html")

@app.route('/predict',methods=['POST','GET'])
def predict():
    if request.method == 'POST':
        to_predict_list = request.form.to_dict()
        to_predict_list = list(to_predict_list.values())
        to_predict_list = list(map(float, to_predict_list))
        result = ValuePredictor(to_predict_list)
    if int(result)== 1:
```

```
        prediction ='Given transaction is fradulent'
    else:
        prediction ='Given transaction is NOT fradulent'
    return render_template("result.html", prediction = prediction)
    if __name__ == "__main__":
    app.run(debug=True
```

**templates/index.html.**

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="{{ url_for('static', filename='css/indexstyle.css') }}">
<title>ML API</title>
</head>
<body>
<form action="{{ url_for('predict')}}" method="POST">
  <input id="input-1" type="text" placeholder="Enter time" name ="time" required autofocus />
  <label for="input-1">
    <span class="label-text">TIME</span>
    <span class="nav-dot"></span>
    <div class="signup-button-trigger">Credit Card Fraud Prediction</div>
  </label>
  <input id="input-2" type="text" placeholder="Enter Amount" name="amount" required />
```

```
  <label for="input-2">
    <span class="label-text">AMOUNT</span>
    <span class="nav-dot"></span>
  </label>
  <input id="input-3" type="text" placeholder="Enter Transaction Method" name="tm" required />
  <label for="input-3">
    <span class="label-text">Transaction Method</span>
    <span class="nav-dot"></span>
  </label>
  <input id="input-4" type="text" placeholder="Transaction id" name="ti" required />
  <label for="input-4">
    <span class="label-text">Transaction id</span>
    <span class="nav-dot"></span>
  </label>
  <input id="input-5" type="text" placeholder="Enter Type Of card" name="ct" required />
  <label for="input-5">
    <span class="label-text">Card Type</span>
    <span class="nav-dot"></span>
  </label>
  <input id="input-6" type="text" placeholder="Enter Location" name="location" required />
  <label for="input-6">
    <span class="label-text">Enter Location</span>
    <span class="nav-dot"></span>
```

```
  </label>
  <input id="input-7" type="text" placeholder="Enter Bank" name="em" required />
  <label for="input-7">
    <span class="label-text">Enter Bank</span>
    <span class="nav-dot"></span>
  </label>
  <button type="submit">Predict</button>
  <p class="tip">Press Tab</p>
  <div class="signup-button">Credit card Fraud Detection</div>
</form>
</body>
</html>
```

**templates/result.html**

```
<!--Hey! This is the original version
of Simple CSS Waves-->
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/resultstyle.css') }}">

</head>
```

```
<!--Hey! This is the original version
of Simple CSS Waves-->
<body>
<div class="header">

<!--Content before waves-->
<div class="inner-header flex">
<!--Just the logo.. Don't mind this-->
<h1>{{ prediction }}</h1>
</div>

<!--Waves Container-->
<div>
<svg class="waves" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
viewBox="0 24 150 28" preserveAspectRatio="none" shape-
rendering="auto">
<defs>
<path id="gentle-wave" d="M-160 44c30 0 58-18 88-18s 58 18 88 18 58-18 88-
18 58 18 88 18 v44h-352z" />
</defs>
<g class="parallax">
<use xlink:href="#gentle-wave" x="48" y="0" fill="#bb1515" />
<use xlink:href="#gentle-wave" x="48" y="3" fill="#bb1515" />
<use xlink:href="#gentle-wave" x="48" y="5" fill="rgba(255,255,255,0.3)"
/>
```

**<use xlink:href="#gentle-wave" x="48" y="7" fill="#bb1515" />**

**</g>**

**</svg>**

** **

**<!--Waves end-->**

** **

**<!--Header ends-->**

**<!--Content starts-->**

** **

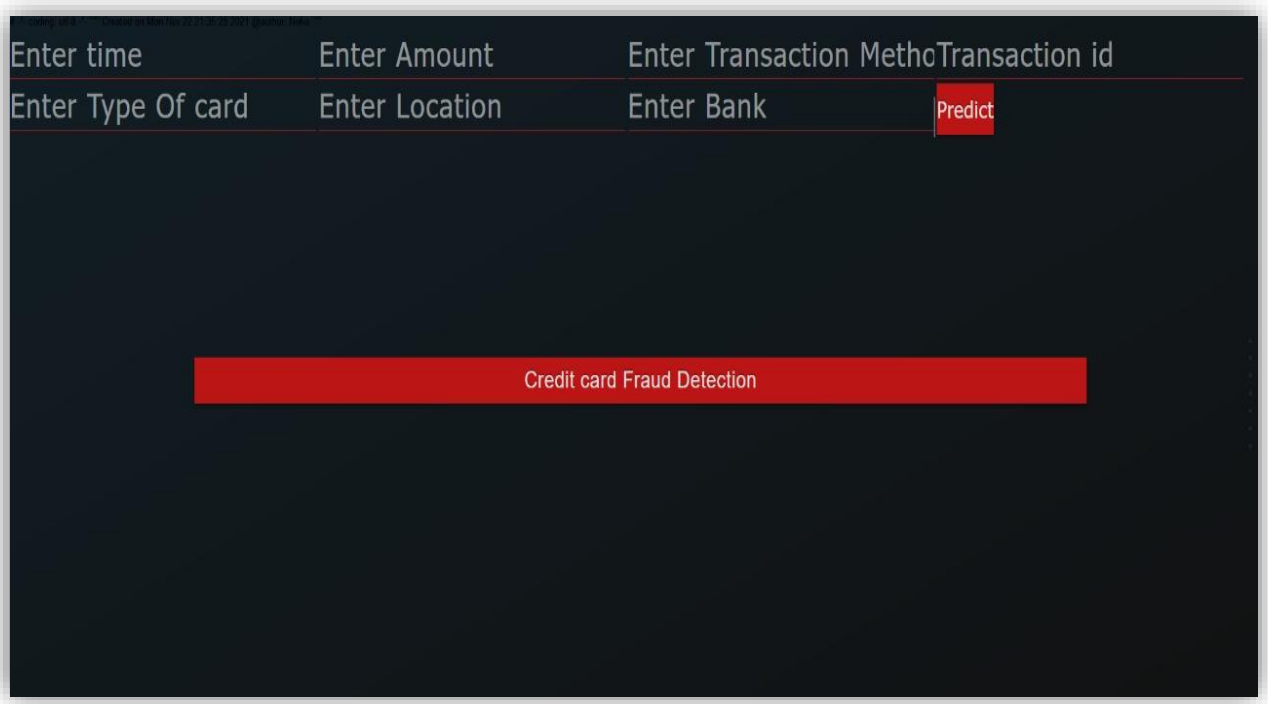
** **

**<!--Content ends-->**

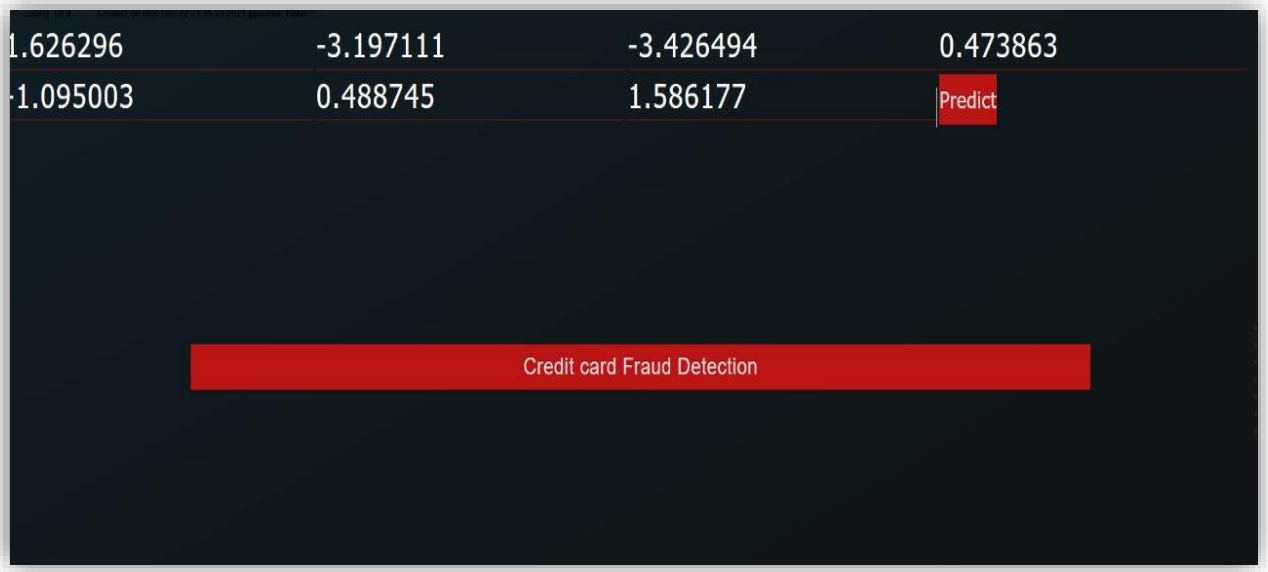
**</body>**

# DEPLOYMENT

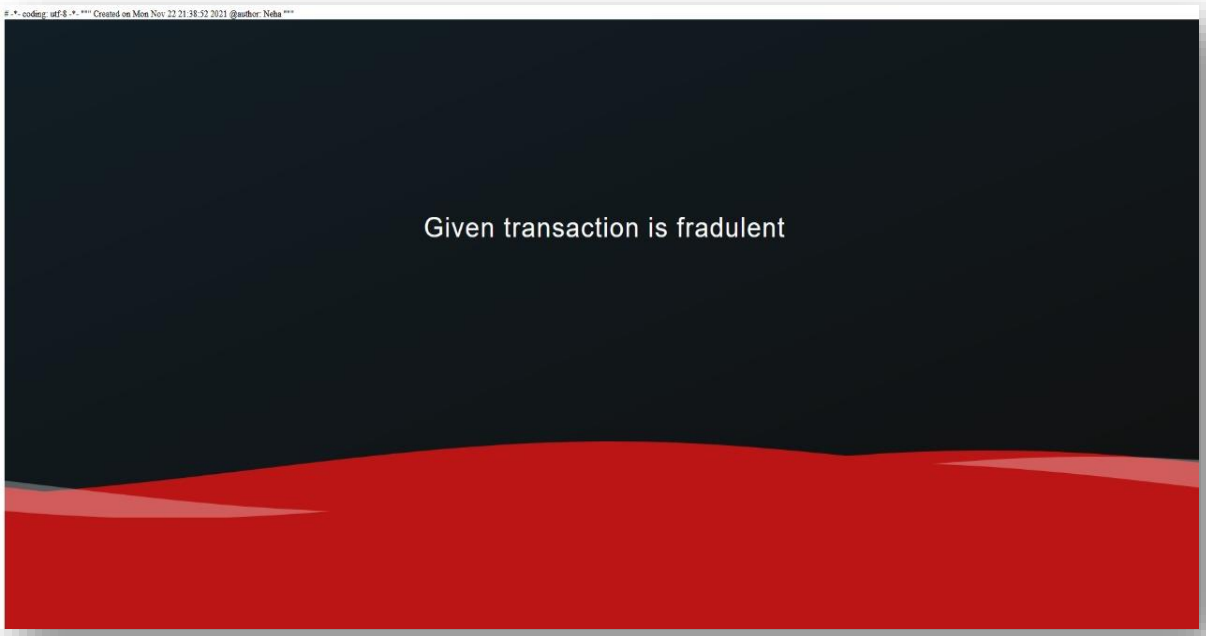
## Screenshot -1



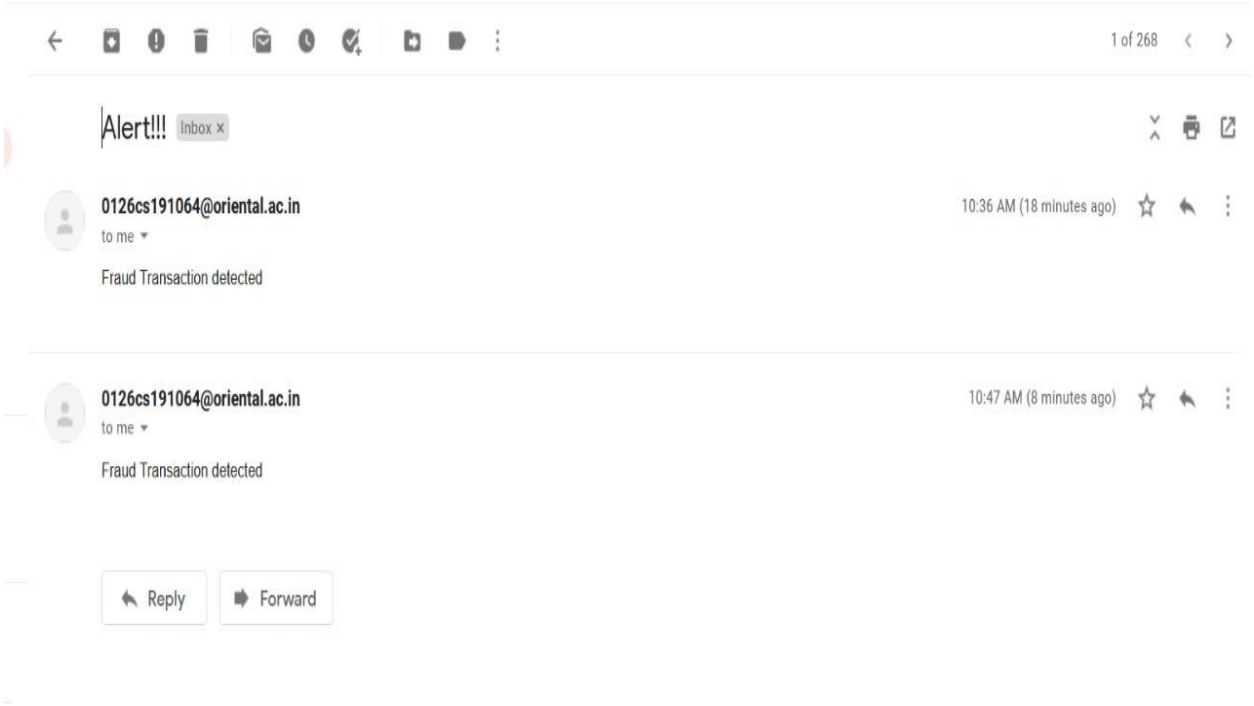

## Screenshot -2



## Screenshot -3




## Screenshot -4 [Email sent from the bank]



## Screenshot -4 [Email received by the customer]