A Project report on

CREDIT CARD FRAUD DETECTION USING RANDOM FOREST

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

in

Computer Science and Engineering

Submitted by

CH. NITIN
(20H55A0502)
M. VENKATESH
(20H55A0513)
SK. APSAR
(20H55A0521)

Under the esteemed guidance of

Ms. T. ADARANA
Assistant Professor

Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC & JNTUH, Approved by AICTE, Permanently Affiliated to JNTUH, Accredited by NBA.)
KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2019- 2023

CMR COLLEGE OF ENGINEERING & TECHNOLOGY
KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Major Project report entitled **"CREDIT CARD FRAUD DETECTION USING RANDOM FOREST"** being submitted by CH NITIN (20H55A0502), M.VENKATESH (20H55A0513), SK.APSAR (20H55A0521) in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

Ms. T. Adarana
Assistant Professor
Dept. of CSE

Dr. Siva Skandha Sanagala
Associate Professor and HOD
Dept. of CSE

ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

We are grateful to **Ms. T. Adarana, Assistant Professor**, Department of Computer Science and Engineering for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank **Dr. Siva Skandha Sanagala,** Head of the Department of Computer Science and Engineering, CMR College of Engineering and Technology, who is the major driving forces to complete my project work successfully.

We are very grateful to **Dr. Vijaya Kumar Koppula**, Dean-Academic, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Dr. V A Narayana,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the Teaching & Non- teaching staff of Department of Computer Science and Engineering for their co-operation

We express our sincere thanks to **Mr. Ch. Gopal Reddy**, Secretary, CMR Group of Institutions, for his continuous care.

Finally, we extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work

SIGNATURE

CH. NITIN	20H55A0502
M. VENKATESH	20H55A0513
SK. APSAR	20H55A0521

TABLE OF CONTENTS

List of Figures

List of Tables

ABSTRACT

The project is mainly focused on credit card fraud detection in real world. Over few years, Credit card fraud has become one of the most common types of fraudulent issues. The recent increase in transactions has resulted with such a huge increase in illegal transactions. The purpose is to obtain things without having to pay for them or to remove money from one account without being authorized. All credit card providing institutions must implement robust detecting fraud systems in order to minimize their losses. The fact that neither cards nor card users must be present in the transaction is among the most difficult parts of running a business. As a result, the card distributer has no way of determining whether or not the customer is purchasing the actual cardholder. The accuracy of the proposed scheme is achieved by utilizing the random forest. Our proposed solution, using random forest algorithm gives the accuracy of detecting the fraud

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

Nowadays Credit card usage has been drastically increased across the world, now people believe in going cashless and are completely dependent on online transactions. The credit card has made the digital transaction easier and more accessible. A huge number of dollars of loss are caused every year by the criminal credit card transactions. Fraud is as old as mankind itself and can take an unlimited variety of different forms. The PwC global economic crime survey of 2017 suggests that approximately 48% of organizations experienced economic crime. Therefore, there’s positively a necessity to unravel the matter of credit card fraud detection. Moreover, the growth of new technologies provides supplementary ways in which criminals may commit a scam. The use of credit cards is predominant in modern day society and credit card fraud has been kept on increasing in recent years. Huge Financial losses have been fraudulent effects on not only merchants and banks but also the individual person who are using the credits. Fraud may also affect the reputation and image of a merchant causing non-financial losses that. For example, if a cardholder is a victim of fraud with a certain company, he may no longer trust their business and choose a competitor

1.1 Problem Statement

Billions of dollars of loss are caused every year by the fraudulent credit card transactions. Fraud is old as humanity itself and can take an unlimited variety of different forms. The PwC global economic crime survey of 2017 suggests that approximately 48% of organizations experienced economic crime. Therefore, there is definitely an urge to solve the problem of credit card fraud detection. Moreover, the development of new technologies provides additional ways in which criminals may commit fraud. The use of credit cards is prevalent in modern day society and credit card fraud has been kept on growing in recent years.

1.2 Research Objective

We propose a Machine learning model to detect fraudulent credit card activities in online financial transactions. Analyzing fake transactions manually is impracticable due to vast amounts of data and its complexity. However, adequately given informative features, could make it is possible using Machine Learning. This hypothesis will be explored in the project. To classify fraudulent and legitimate credit card transaction by supervised learning Algorithm such as Random Forest. To help us to get awareness about the fraudulent and without loss of any financially

1.3 Project Scope and Limitations

In this proposed project we designed a protocol or a model to detect the fraud activity in credit card transactions. This system is capable of providing most of the essential features required to detect fraudulent and legitimate transactions.

CHAPTER 2
BACKGROUND WORK

CHAPTER 2

BACKGROUND WORK

2.1 DIGITAL CARD FRAUD DETECTION MACHINE LEARNING

Random forest is a supervised machine learning algorithm based on ensemble learning. Ensemble learning is an algorithm where the predictions are derived by assembling or bagging different models or similar model multiple times. The random forest algorithm works in a similar way and uses multiple algorithm i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks

2.1.1 INTRODUCTION

All steps of the existing system framework are presented in Fig 1 and discussed in the following section.

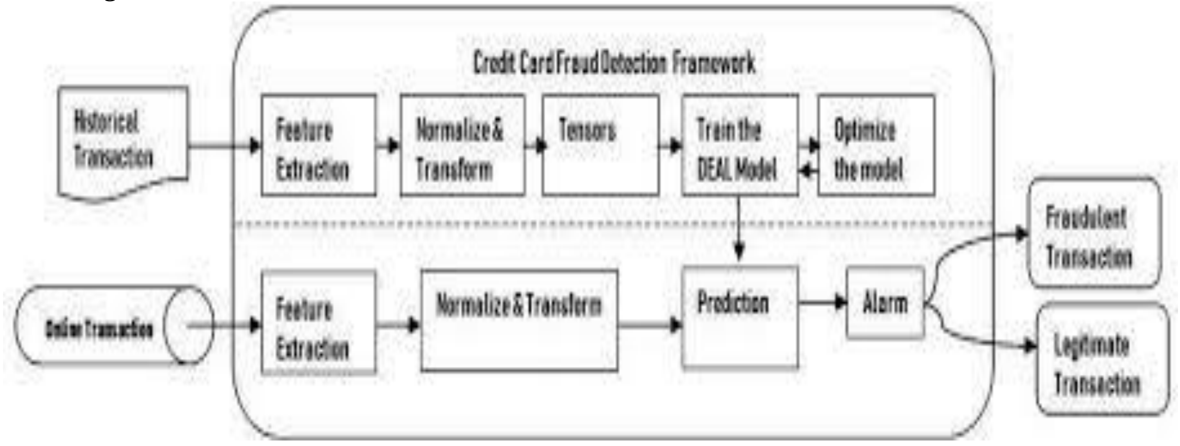


Fig 1. Credit card fraud detection Framework

Methodology: Dataset URL: https://www.kaggle.com/mlg-ulb/creditcardfrau

To provide privacy to user's transaction data Kaggle's peoples have converted transaction data to numerical format using PCA Algorithm. Below are some examples from 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"
0.338320769942518,0.462387777762292,0.239598554061257,0.0986979012610507,0.36378

2.1.2 Merits and Demerits

Merits

- If the eyes are both closed, we increase the score and when eyes are open, we decrease the score. We are drafting the outcome to display the actual time condition of the driver.
- approach enables us to identify driver's face characteristics like eye closure percentage, eye-mouth aspect ratios, blink rate, yawning, head movement.

Demerits

- It is not suitable for real-time processing.
- The existing system uses the orientation of facial characteristics for drowsy detection.
- based on three factors such as physiological, behavioral, and vehicle-based measurements. But these approaches pose some disadvantages in certain real time scenarios.

2.1.3 IMPLEMENTATION OF DIGITAL CARD FRAUD DETECTION

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

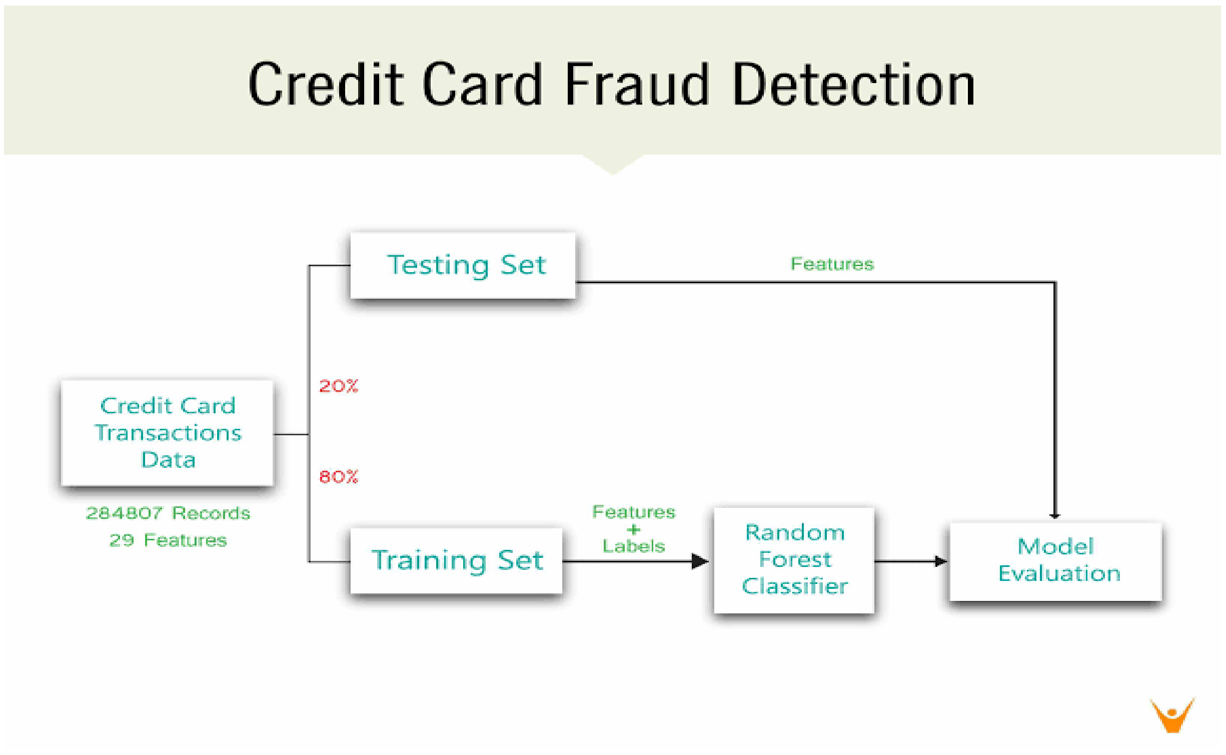
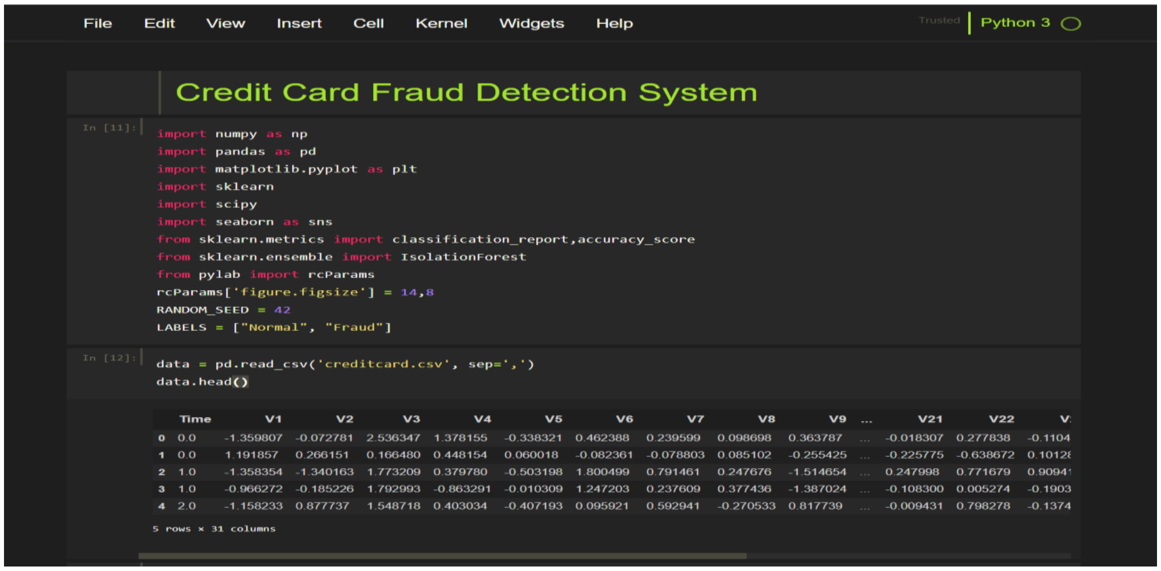


Fig.2 Digital card fraud detection framework

2.2 Fraud Detection using Machine Learning

2.2.1 INTRODUCTION

Frauds are known to be dynamic and have no patterns, hence they are not easy to identify. Fraudsters use recent technological advancements to their advantage. They somehow bypass security checks, leading to the loss of millions of dollars. Analyzing and detecting unusual activities using data mining techniques is one way of tracing fraudulent transactions. transactions. This paper aims to benchmark multiple machine learning methods such as k-nearest neighbor (KNN), random forest and support vector machines (SVM), while the deep learning methods such as autoencoders, convolutional neural networks (CNN), restricted Boltzmann machine (RBM) and deep belief networks (DBN). The datasets which will be used are the European (EU) Australian and German dataset. The Area Under the ROC Curve (AUC), Matthews Correlation Coefficient (MCC) and Cost of failure are the 3-evaluation metrics that would be used



Credit Card Fraud Detection System

```
In [11]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import sklearn
         import scipy
         import seaborn as sns
         from sklearn.metrics import classification_report,accuracy_score
         from sklearn.ensemble import IsolationForest
         from pylab import rcParams
         rcParams['figure.figsize'] = 14,8
         RANDOM_SEED = 42
         LABELS = ["Normal", "Fraud"]
```

```
In [12]: data = pd.read_csv('creditcard.csv', sep=',')
         data.head()
```

```
   Time        V1        V2        V3        V4        V5        V6        V7        V8        V9  ...       V21       V22      V
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 -0.1104
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  0.1012
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  0.9094
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 -0.1903
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 -0.1374

5 rows × 31 columns
```

Fig2 Fraud detection using random forest

2.2.2 Merits and Demerits

Merits

- If the eyes are both closed, we increase the score and when eyes are open, we decrease the score. We are drafting the outcome to display the actual time condition of the driver.
- approach enables us to identify driver's face characteristics like eye closure percentage, eye-mouth aspect ratios, blink rate, yawning, head movement.

Demerits

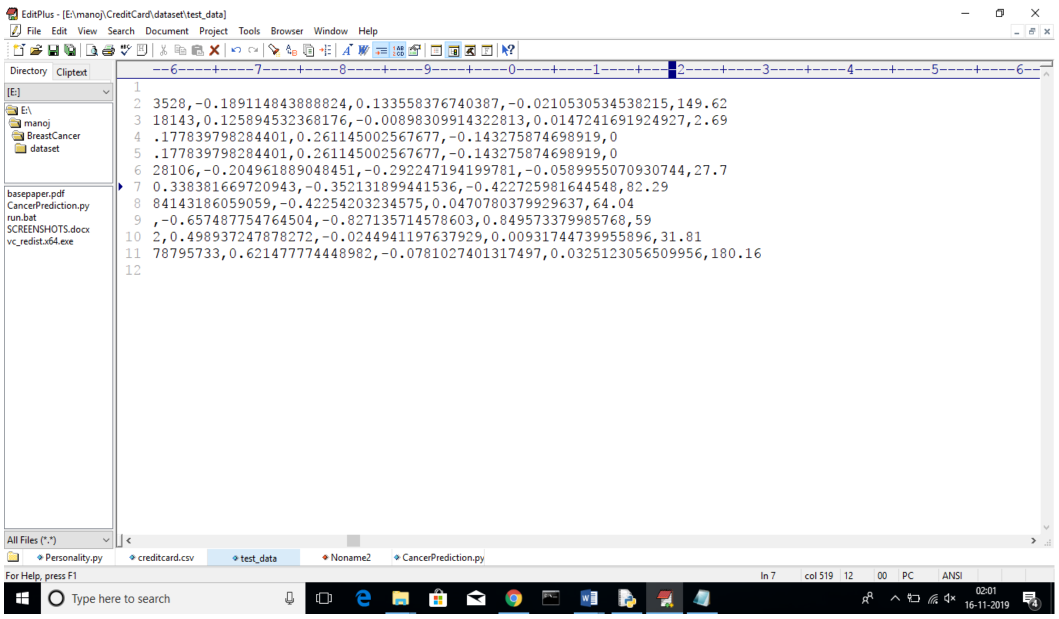
- It is not suitable for real-time processing.
- The existing system uses the orientation of facial characteristics for drowsy detection.
- based on three factors such as physiological, behavioral, and vehicle-based measurements. But these approaches pose some disadvantages in certain real time scenarios.

2.2.3 Implementation of digital card fraud detection

Credit Card Fraud Detection Using Random Forest & Cart Algorithm
In this project we are using python Random Forest inbuilt Cart algorithm to detect fraud transaction from credit card dataset, we downloaded this dataset from 'Kaggle's' web site from below URL
Dataset URL: https://www.kaggle.com/mlg-ulb/creditcardfrau
To provide privacy to user's transaction data Kaggle's peoples have converted transaction data to numerical format using PCA Algorithm. Below are some examples from 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"
0,-1.3598071336738,-0.0727811733098497,2.53634673796914,1.37815522427443,-0.338320769942518,0.462387777762292,0.239598554061257,0.0986979012610507,0.363786969611213,0.0907941719789316,-0.551599533260813,-0.617800855762348,-0.991389847235408,-0.311169353699879,1.46817697209427,-0.470400525259478,0.207971241929242,0.0257905801985591,0.403992960255733,0.251412098239705,-0.018306777944153,0.277837575558899,-0.110473910188767,0.0669280749146731,0.128539358273528,-0.189114843888824,0.133558376740387,-0.0210530534538215,149.62,"0"
0,1.19185711131486,0.26615071205963,0.16648011335321,0.448154078460911,0.0600176492822243,-0.0823608088155687,-0.0788029833323113,0.0851016549148104,-0.255425128109186,-0.166974414004614,1.61272666105479,1.06523531137287,0.48909501589608,-0.143772296441519,0.635558093258208,0.463917041022171,-0.114804663102346,-0.183361270123994,-0.145783041325259,-0.0690831352230203,-0.225775248033138,-0.638671952771851,0.101288021253234,-

0.00898309914322813,0.0147241691924927,2.69,"0"
406-2.3122265423263,1.95199201064158, -1.60985073229769,3.9979055875468, -0.522187864667764, -1.42654531920595, -2.53738730624579,1.39165724829804, -2.77008927719433, -2.77227214465915,3.20203320709635, -2.89990738849473, -0.595221881324605, -4.28925378244217,0.389724120274487, -1.14074717980657, -2.83005567450437, -0.0168224681808257,0.416955705037907,0.126910559061474,0.517232370861764, -0.0350493686052974, -0.465211076182388,0.320198198514526,0.0445191674731724,0.177839798284401,0.261145002567677, -0.143275874698919,0,"1"

Above bold names are the column names of this dataset and others decimal values are the content of dataset and in above 3 rows last column contains class label where 0 means transaction values are normal and 1 means contains fraud values.
Using above ‘CreditCardFraud.csv’ file we will train Random Forest algorithm and then we will upload test data file and this test data will be applied on Random Forest train model to predict whether test data contains normal or fraud transaction signatures. When we upload test data then it will contain only transaction data no class label will be there application will predict and give the result. See below test data file



```

3528,-0.189114843888824,0.133558376740387,-0.0210530534538215,149.62
18143,0.125894532368176,-0.00898309914322813,0.0147241691924927,2.69
.177839798284401,0.261145002567677,-0.143275874698919,0
.177839798284401,0.261145002567677,-0.143275874698919,0
28106,-0.204961889048451,-0.292247194199781,-0.0589955070930744,27.7
0.338381669720943,-0.352131899441536,-0.422725981644548,82.29
84143186059059,-0.42254203234575,0.0470780379929637,64.04
,-0.657487754764504,-0.827135714578603,0.849573379985768,59
2,0.498937247878272,-0.0244941197637929,0.00931744739955896,31.81
78795733,0.621477774448982,-0.0781027401317497,0.0325123056509956,180.16

```

In above screen in test data file there are no 0 or 1 values, application will predict from this test data using random forest and give the result.

Random Forest Algorithm

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests create decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance. Python SKLEARN inbuilt contains support for CART with all decision trees and random forest classifier.

CHAPTER 3
PROPOSED SYSTEM

CHAPTER -3
PROPOSED SYSTEM

3.1 Objective of proposed Model

Random Forest Algorithm

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance. Python SKLEARN inbuilt contains support for CART with all decision trees and random forest classifier.

Random forests have a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

3.2 Algorithms used for proposed Model

Random forests or random decision forests is an ensemble learning method
for classification, regression and other tasks that operates by constructing a multitude
of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit
of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees.[citation needed] However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.). The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho and later independently by Amit and Geman in order to construct a collection of decision trees with controlled variance.

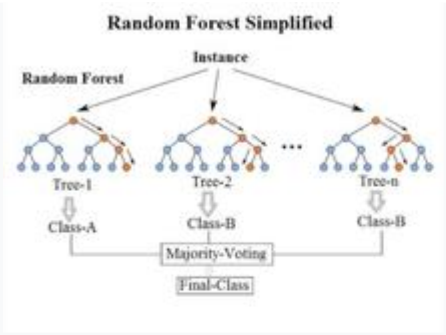


Diagram of a random decision forest

The general method of random decision forests was first proposed by Ho in 1995.[1] Ho established that forests of trees splitting with oblique hyperplanes can gain accuracy as they grow without suffering from overtraining, as long as the forests are randomly restricted to be sensitive to only selected feature dimensions. A subsequent work along the same lines[2] concluded that other splitting methods behave similarly, as long as they are randomly forced to be insensitive to some feature dimensions. Note that this observation of a more complex classifier (a larger)

3.3 Designing

INPUT AND OUTPUT DESIGN

INPUT DESIGN: -

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- ➢ What data should be given as input?
- ➢ How the data should be arranged or coded?
- ➢ The dialog to guide the operating personnel in providing input.
- ➢ Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy

OUTPUT DESIGN: -

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

3.3.1 UML DIAGRAMS:

UML represents Unified Modeling Language. UML is an institutionalized universally useful showing dialect in the subject of article situated programming designing. The fashionable is overseen, and become made by way of, the Object Management Group.

The goal is for UML to become a regular dialect for making fashions of item arranged PC programming. In its gift frame UML is contained two noteworthy components: a Meta-show and documentation. Later on, a few types of method or system can also likewise be brought to; or related with, UML.

The Unified Modeling Language is a popular dialect for indicating, Visualization, Constructing and archiving the curios of programming framework, and for business demonstrating and different non-programming frameworks.

The UML speaks to an accumulation of first-rate building practices which have verified fruitful in the showing of full-size and complicated frameworks.

The UML is a essential piece of creating gadgets located programming and the product development method. The UML makes use of commonly graphical documentations to specific the plan of programming ventures.

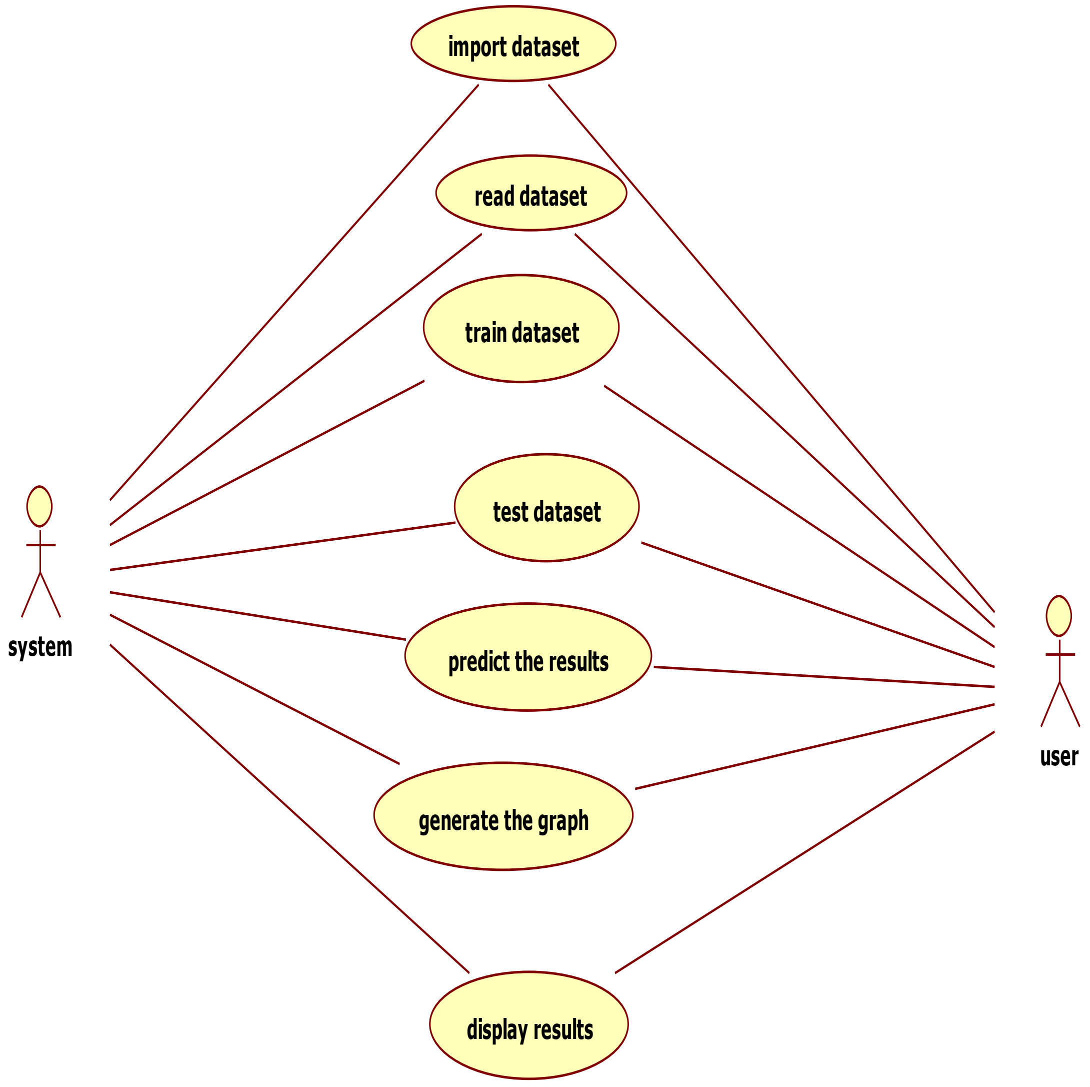
GOALS:

The Primary goals inside the plan of the UML are as in step with the subsequent:

1. Provide clients a prepared to-utilize, expressive visual showing Language on the way to create and change massive models.
2. Provide extendibility and specialization units to make bigger the middle ideas.
3. Be free of specific programming dialects and advancement manner.
4. Provide a proper cause for understanding the displaying dialect.
5. Encourage the improvement of OO gadgets exhibit.
6. Support large amount advancement thoughts, for example, joint efforts, systems.

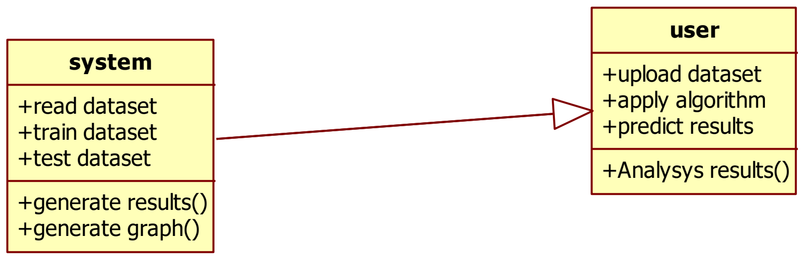
USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



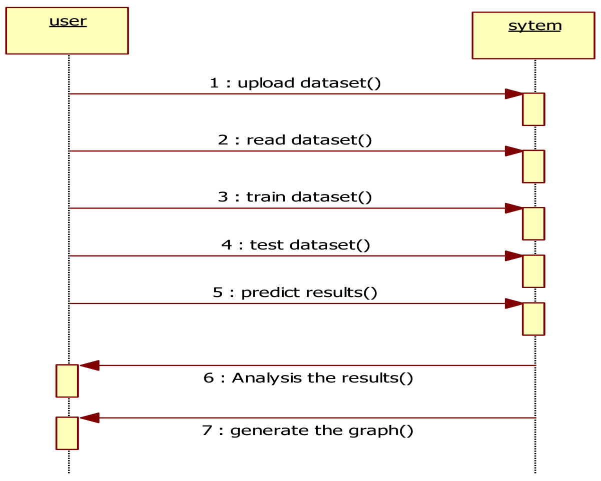
CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



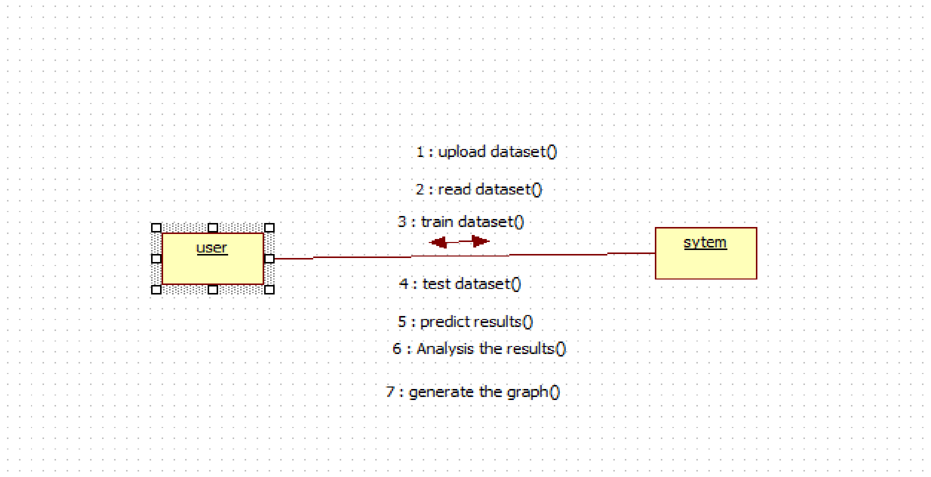
SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



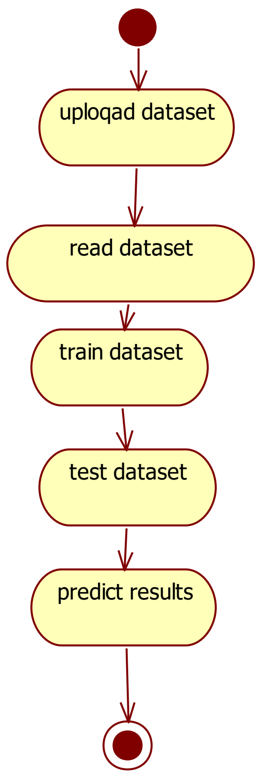
COLLABORATION DIAGRAM:

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization where as the collaboration diagram shows the object organization.



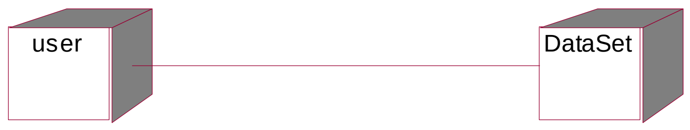
ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



DEPLOYMENT DIAGRAM:

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware's used to deploy the application.



3.4 Stepwise Implementation and code:

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
import numpy as np
import numpy as np
from tkinter.filedialog import askopenfilename
import pandas as pd
from sklearn import *
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
#from sklearn.tree import export_graphviz
#from IPython import display
main = tkinter.Tk()
main.title("Credit Card Fraud Detection") #designing main screen
main.geometry("1300x1200")
```

```
global filename
global cls
global X, Y, X_train, X_test, y_train, y_test
global random_acc # all global variables names define in above lines
global clean
global attack
global total
def traintest(train):     #method to generate test and train data from dataset
X = train.values[:, 0:29]
Y = train.values[:, 30]
print(X)
print(Y)
X_train, X_test, y_train, y_test = train_test_split(
X, Y, test_size = 0.3, random_state = 0)
return X, Y, X_train, X_test, y_train, y_test
def generateModel(): #method to read dataset values which contains all five features data
global X, Y, X_train, X_test, y_train, y_test
train = pd.read_csv(filename)
X, Y, X_train, X_test, y_train, y_test = traintest(train)
text.insert(END,"Train & Test Model Generated\n\n")
text.insert(END,"Total Dataset Size : "+str(len(train))+"\n")
text.insert(END,"Split Training Size : "+str(len(X_train))+"\n")
text.insert(END,"Split Test Size : "+str(len(X_test))+"\n")
```

```
def upload(): #function to upload tweeter profile
global filename
filename = filedialog.askopenfilename(initialdir="dataset")
text.delete('1.0', END)
text.insert(END,filename+" loaded\n");
def prediction(X_test, cls):  #prediction done here
y_pred = cls.predict(X_test)
for i in range(50):
print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))
return y_pred
# Function to calculate accuracy
def cal_accuracy(y_test, y_pred, details):
accuracy = accuracy_score(y_test,y_pred)*100
text.insert(END,details+"\n\n")
text.insert(END,"Accuracy : "+str(accuracy)+"\n\n")
return accuracy
def runRandomForest():
headers = ["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"]
global random_acc
global cls
global X, Y, X_train, X_test, y_train, y_test
cls = RandomForestClassifier(n_estimators=50,max_depth=2,random_state=0,class_weight='balanced')
cls.fit(X_train, y_train)
text.insert(END,"Prediction Results\n\n")
prediction_data = prediction(X_test, cls)
random_acc = cal_accuracy(y_test, prediction_data,'Random Forest Accuracy')
#str_tree = export_graphviz(cls, out_file=None, feature_names=headers,filled=True, special_characters=True, rotate=True, precision=0.6)
#display.display(str_tree)
def predicts():
global clean
global attack
global total
clean = 0;
attack = 0;
text.delete('1.0', END)
filename = filedialog.askopenfilename(initialdir="dataset")
test = pd.read_csv(filename)
test = test.values[:, 0:29]
total = len(test)
text.insert(END,filename+" test file loaded\n");
y_pred = cls.predict(test)
```

```
for i in range(len(test)):
if str(y_pred[i]) == '1.0':
attack = attack + 1
text.insert(END,"X=%s, Predicted = %s" % (test[i], 'Contains Fraud Transaction Signature')+"\n\n")
else:
clean = clean + 1
text.insert(END,"X=%s, Predicted = %s" % (test[i], 'Transaction Contains Cleaned Signatures')+"\n\n")
def graph():
height = [total,clean,attack]
bars = ('Total Transactions','Normal Transaction','Fraud Transaction')
y_pos = np.arange(len(bars))
plt.bar(y_pos, height)
plt.xticks(y_pos, bars)
plt.show()
font = ('times', 16, 'bold')
title = Label(main, text='Credit Card Fraud Detection Using Random Forest Tree Based Classifier')
title.config(bg='greenyellow', fg='dodger blue')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)
font1 = ('times', 12, 'bold')
```

```
text=Text(main,height=20,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=50,y=120)
text.config(font=font1)
font1 = ('times', 14, 'bold')
uploadButton = Button(main, text="Upload Credit Card Dataset", command=upload)
uploadButton.place(x=50,y=550)
uploadButton.config(font=font1)
modelButton = Button(main, text="Generate Train & Test Model", command=generateModel)
modelButton.place(x=350,y=550)
modelButton.config(font=font1)
runrandomButton = Button(main, text="Run Random Forest Algorithm", command=runRandomForest)
runrandomButton.place(x=650,y=550)
runrandomButton.config(font=font1)
predictButton = Button(main, text="Detect Fraud From Test Data", command=predicts)
predictButton.place(x=50,y=600)
predictButton.config(font=font1)
graphButton = Button(main, text="Clean & Fraud Transaction Detection Graph", command=graph)
```

```
graphButton.place(x=350,y=600)
graphButton.config(font=font1)
exitButton = Button(main, text="Exit", command=exit)
exitButton.place(x=770,y=600)
exitButton.config(font=font1)
main.config(bg='LightSkyBlue')
main.mainloop()
```

FEASIBILITY STUDY:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ♦ ECONOMICAL FEASIBILITY
- ♦ TECHNICAL FEASIBILITY
- ♦ SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY:

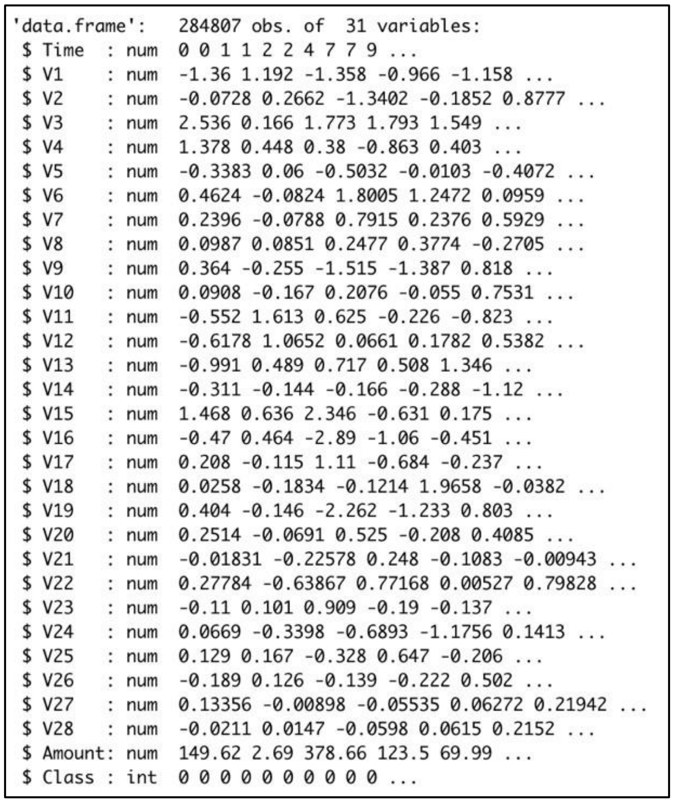
The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system

CHAPTER 4
RESULTS AND DISCUSSION

CHAPTER - 4
RESULTS AND DISCUSSION

4.1 Comparison of Existing Solutions

The first figure bellow shows the structure of the dataset where all attributes are shown, with their type, in addition to glimpse of the variables within each attribute, as shown at the end of the figure the Class type is integer which I needed to change to factor and identify the 0 as Not Fraud and the 1 as Fraud to ease the process of creating the model and obtain visualizations



```
'data.frame':   284807 obs. of  31 variables:
 $ Time  : num  0 0 1 1 2 2 4 7 7 9 ...
 $ V1    : num  -1.36 1.192 -1.358 -0.966 -1.158 ...
 $ V2    : num  -0.0728 0.2662 -1.3402 -0.1852 0.8777 ...
 $ V3    : num  2.536 0.166 1.773 1.793 1.549 ...
 $ V4    : num  1.378 0.448 0.38 -0.863 0.403 ...
 $ V5    : num  -0.3383 0.06 -0.5032 -0.0103 -0.4072 ...
 $ V6    : num  0.4624 -0.0824 1.8005 1.2472 0.0959 ...
 $ V7    : num  0.2396 -0.0788 0.7915 0.2376 0.5929 ...
 $ V8    : num  0.0987 0.0851 0.2477 0.3774 -0.2705 ...
 $ V9    : num  0.364 -0.255 -1.515 -1.387 0.818 ...
 $ V10   : num  0.0908 -0.167 0.2076 -0.055 0.7531 ...
 $ V11   : num  -0.552 1.613 0.625 -0.226 -0.823 ...
 $ V12   : num  -0.6178 1.0652 0.0661 0.1782 0.5382 ...
 $ V13   : num  -0.991 0.489 0.717 0.508 1.346 ...
 $ V14   : num  -0.311 -0.144 -0.166 -0.288 -1.12 ...
 $ V15   : num  1.468 0.636 2.346 -0.631 0.175 ...
 $ V16   : num  -0.47 0.464 -2.89 -1.06 -0.451 ...
 $ V17   : num  0.208 -0.115 1.11 -0.684 -0.237 ...
 $ V18   : num  0.0258 -0.1834 -0.1214 1.9658 -0.0382 ...
 $ V19   : num  0.404 -0.146 -2.262 -1.233 0.803 ...
 $ V20   : num  0.2514 -0.0691 0.525 -0.208 0.4085 ...
 $ V21   : num  -0.01831 -0.22578 0.248 -0.1083 -0.00943 ...
 $ V22   : num  0.27784 -0.63867 0.77168 0.00527 0.79828 ...
 $ V23   : num  -0.11 0.101 0.909 -0.19 -0.137 ...
 $ V24   : num  0.0669 -0.3398 -0.6893 -1.1756 0.1413 ...
 $ V25   : num  0.129 0.167 -0.328 0.647 -0.206 ...
 $ V26   : num  -0.189 0.126 -0.139 -0.222 0.502 ...
 $ V27   : num  0.13356 -0.00898 -0.05535 0.06272 0.21942 ...
 $ V28   : num  -0.0211 0.0147 -0.0598 0.0615 0.2152 ...
 $ Amount: num  149.62 2.69 378.66 123.5 69.99 ...
 $ Class : int  0 0 0 0 0 0 0 0 0 0 ...
```

Figure 1 - Dataset Structure

The second figure shows the distribution of the class, the red bar which contains 284,315 variables represents the non-fraudulent transactions, and the blue bar with 492 variables represents the fraudulent transactions.

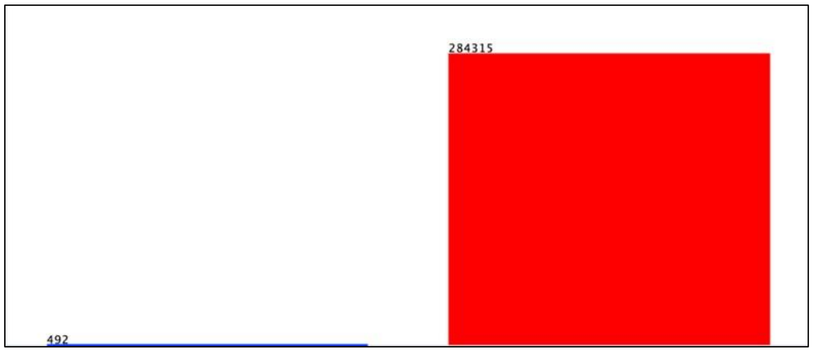


Figure 2 - Class Distribution

Correlation between attributes "Image from R"

The correlations between all the of the attributes within the dataset are presented in the figure below.

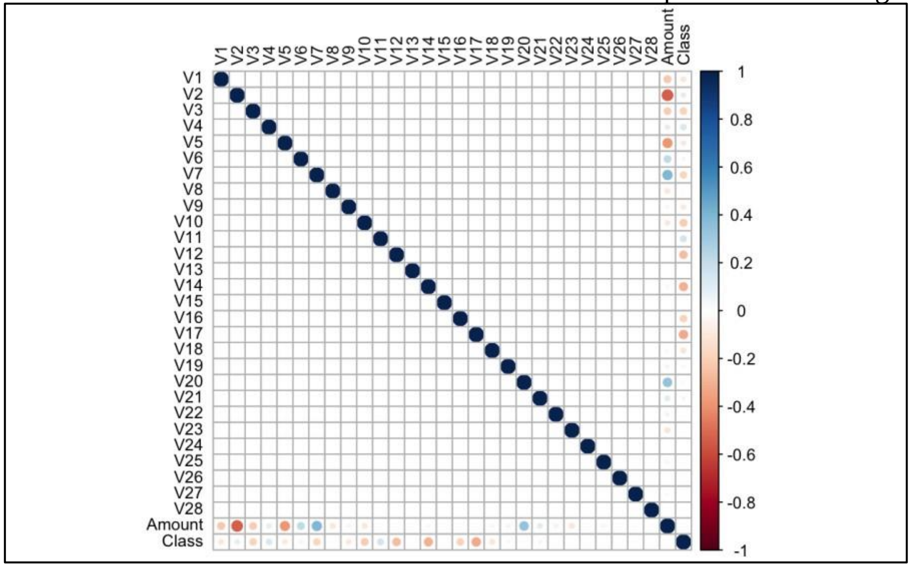


Figure 3 - Correlations

Attribute with the most fraud

Figure 4 below shows attribute 18 the attribute with the most credit card fraudulent transactions, the blue line represents the variable 1 which is the fraudulent transactions.

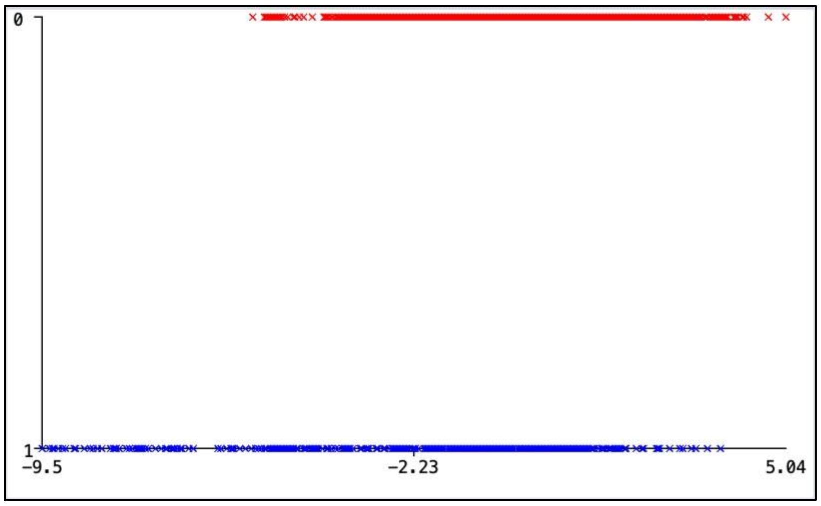


Figure 4 _ Variable 18

Attribute with the less fraud

The figure below shows the variable that have the lowest number of fraudulent transactions, as mentioned earlier the blue line represents the fraudulent instances within the dataset.

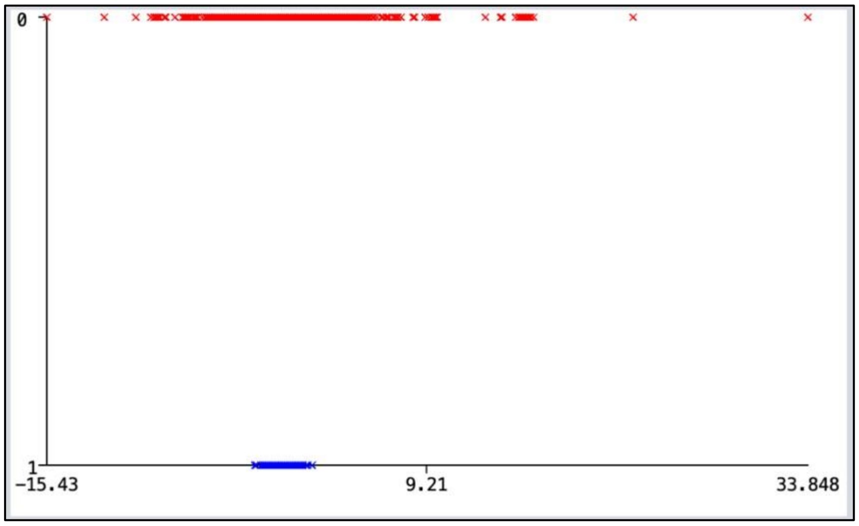


Figure 5 - Variable 28

Data Preprocessing

As there are no NAs nor duplicated variables, the preparation of the dataset was simple the first alteration that was made to be able to open the dataset on Weka program is changing the type of the class attribute from Numeric to Class and identify the class as {1,0} using the

program Sublime Text. Another alteration was made on the type as well on the R program to be able to create the model and the visualization.

Data Modeling

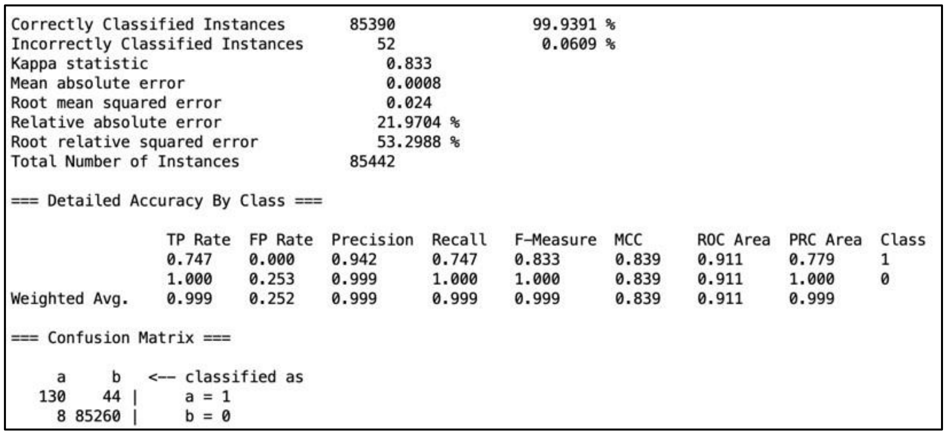
After making sure that the data is ready to get modeled the four models were created using both Weka and R. the model SVM was created using Weka only, as for KNN, Logistic Regression and Naïve-Bayes they were created using R and Weka.

KNN

The K-Nearest Neighbor algorithm (KNN) is a supervised ML technique that can be applied in both scenario instances, classification instances along with regression instances (Mahesh, 2020).To figure the best KNN model two Ks where used K=3 and K=7, both are presented with figures from both Weka and R.

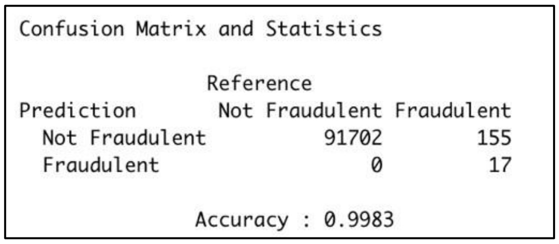
- K = 3

During the making of the KNN model, I decided to create two models where K=3 and K=7. Figure 5 shows the model created in R, the model scored an accuracy of 99.83% and managed to correctly identify 91,719 transactions and missed 155. As for the Weka program the model scored 99.94% for the accuracy and miss-classified 52 transactions.
As there are different accuracies the average of the accuracies is 99.89%.



```
Correctly Classified Instances       85390               99.9391 %
Incorrectly Classified Instances        52                0.0609 %
Kappa statistic                          0.833
Mean absolute error                      0.0008
Root mean squared error                  0.024
Relative absolute error                 21.9704 %
Root relative squared error             53.2988 %
Total Number of Instances            85442

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.747    0.000    0.942      0.747    0.833      0.839    0.911     0.779     1
                 1.000    0.253    0.999      1.000    1.000      0.839    0.911     1.000     0
Weighted Avg.    0.999    0.252    0.999      0.999    0.999      0.839    0.911     0.999

=== Confusion Matrix ===

     a     b   <-- classified as
   130    44 |     a = 1
     8 85260 |     b = 0
```

Figure 6 - Weka K=3

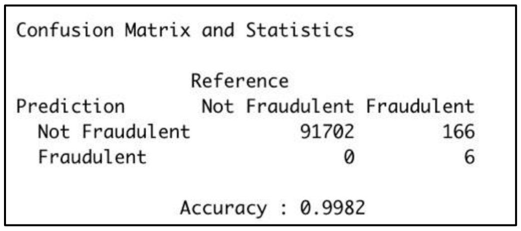


```
Confusion Matrix and Statistics

                Reference
Prediction       Not Fraudulent Fraudulent
  Not Fraudulent          91702        155
  Fraudulent                  0         17

               Accuracy : 0.9983
```

Figure 7 - RStudio K=3

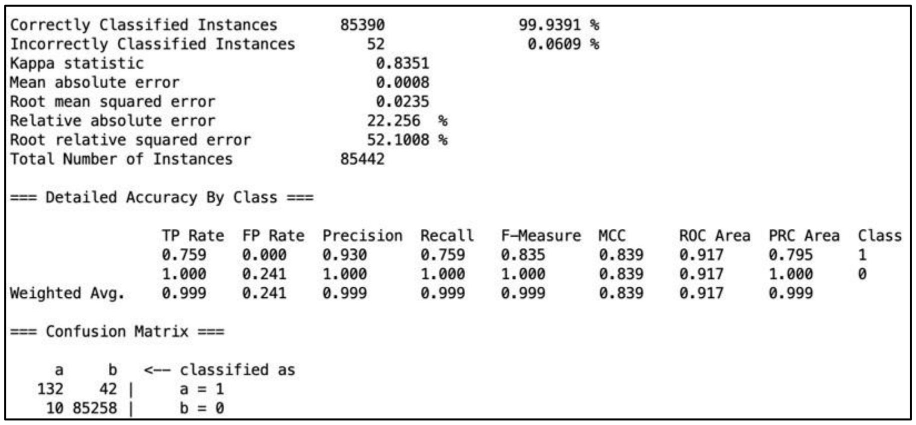
- K = 7

There was a slight decrease in the accuracy in the model created in R (Figure 6) as it scored 99.82% when K is 7, and the model miss classified 166 fraudulent transactions as nonfraudulent. As for Weka (Figure 7) the accuracy is the same as K=3 99.94% with 52 misclassified transactions, the only difference is within the classifications. The average of the accuracies is 99.88%



```
Confusion Matrix and Statistics

                Reference
Prediction       Not Fraudulent Fraudulent
  Not Fraudulent          91702        166
  Fraudulent                  0          6

               Accuracy : 0.9982
```

Figure 8 - RStudio K=7

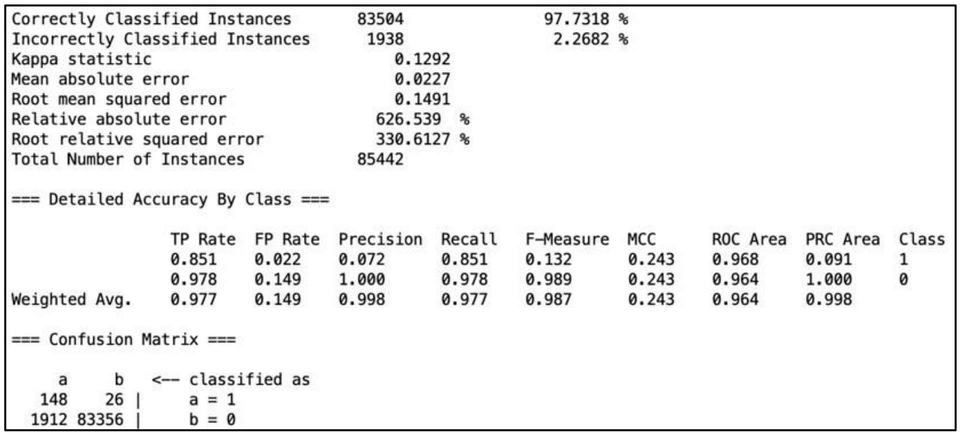


```
Correctly Classified Instances       85390               99.9391 %
Incorrectly Classified Instances        52                0.0609 %
Kappa statistic                          0.8351
Mean absolute error                      0.0008
Root mean squared error                  0.0235
Relative absolute error                 22.256  %
Root relative squared error             52.1008 %
Total Number of Instances            85442

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.759    0.000    0.930      0.759    0.835      0.839    0.917     0.795     1
                 1.000    0.241    1.000      1.000    1.000      0.839    0.917     1.000     0
Weighted Avg.    0.999    0.241    0.999      0.999    0.999      0.839    0.917     0.999

=== Confusion Matrix ===

     a     b   <-- classified as
   132    42 |     a = 1
    10 85258 |     b = 0
```

Figure 9 - Weka K=7

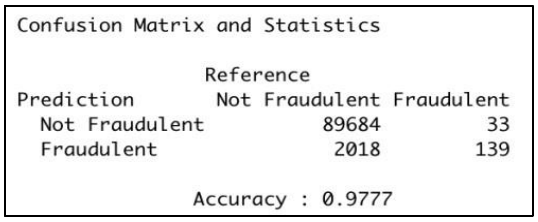
Naïve Bayes

Naïve Bayes is a classification algorithm that consider the being of a certain trait within a class is unrelated to the being of any different feature, the main use of it is for clustering and classifications, depending on the conditional probability of happening (Mahesh, 2020).
The second model created by R is Naïve Bayes, figure 9 shows the performance of the model, it scored an accuracy of 97.77% and misclassified a total of 2,051 transactions, 33 fraudulent as nonfraudulent and 2018 nonfraudulent as fraudulent. There is a slight difference in the accuracy of the Naïve bayes model created within Weka as its 97.73% and the misclassification instances are 1,938.



```
Correctly Classified Instances       83504               97.7318 %
Incorrectly Classified Instances      1938                2.2682 %
Kappa statistic                          0.1292
Mean absolute error                      0.0227
Root mean squared error                  0.1491
Relative absolute error                626.539  %
Root relative squared error            330.6127 %
Total Number of Instances            85442

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.851    0.022    0.072      0.851    0.132      0.243    0.968     0.091     1
                 0.978    0.149    1.000      0.978    0.989      0.243    0.964     1.000     0
Weighted Avg.    0.977    0.149    0.998      0.977    0.987      0.243    0.964     0.998

=== Confusion Matrix ===

     a     b   <-- classified as
   148    26 |     a = 1
  1912 83356 |     b = 0
```

Figure 10 - Weka Naïve Bayes



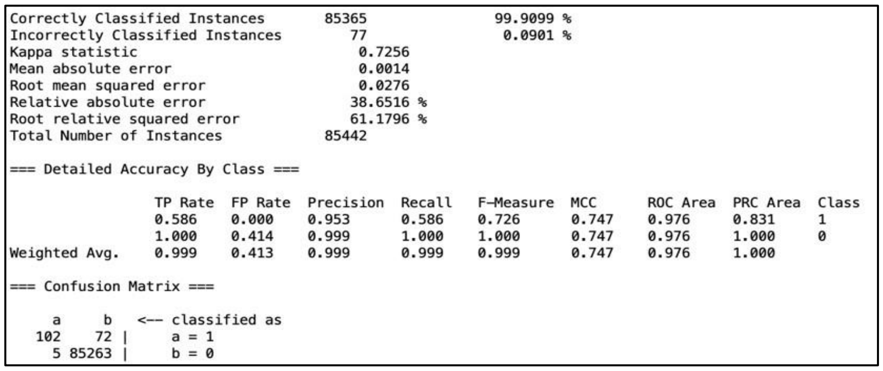
```
Confusion Matrix and Statistics

                Reference
Prediction       Not Fraudulent Fraudulent
  Not Fraudulent          89684         33
  Fraudulent               2018        139

               Accuracy : 0.9777
```

Figure 11 - RStudio Naïve Bayes

4.2 Data Collection and Performance Metrics

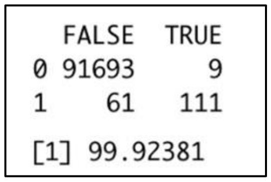
Logistic Regression

Logistic Regression model is statical model where evaluations are formed of the connection among dependent qualitative variable (binary or binomial logistic regression) or variable with three values or higher (multinomial logistic regression) and one independent explanatory variable or higher whether qualitative or quantitative (Domínguez-Almendros et al., 2011).
The last model created using both R and Weka is Logistic Regression, the model managed to score and accuracy of 99.92% in R (figure 11) with 70 misclassified instances, while it scored 99.91% in Weka with 77 misclassified instances as presented in figure 10.



```
Correctly Classified Instances       85365               99.9099 %
Incorrectly Classified Instances        77                0.0901 %
Kappa statistic                          0.7256
Mean absolute error                      0.0014
Root mean squared error                  0.0276
Relative absolute error                 38.6516 %
Root relative squared error             61.1796 %
Total Number of Instances            85442

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.586    0.000    0.953      0.586    0.726      0.747    0.976     0.831     1
                 1.000    0.414    0.999      1.000    1.000      0.747    0.976     1.000     0
Weighted Avg.    0.999    0.413    0.999      0.999    0.999      0.747    0.976     1.000

=== Confusion Matrix ===

     a     b   <-- classified as
   102    72 |     a = 1
     5 85263 |     b = 0
```

Figure12 - WekaLogistic Regression



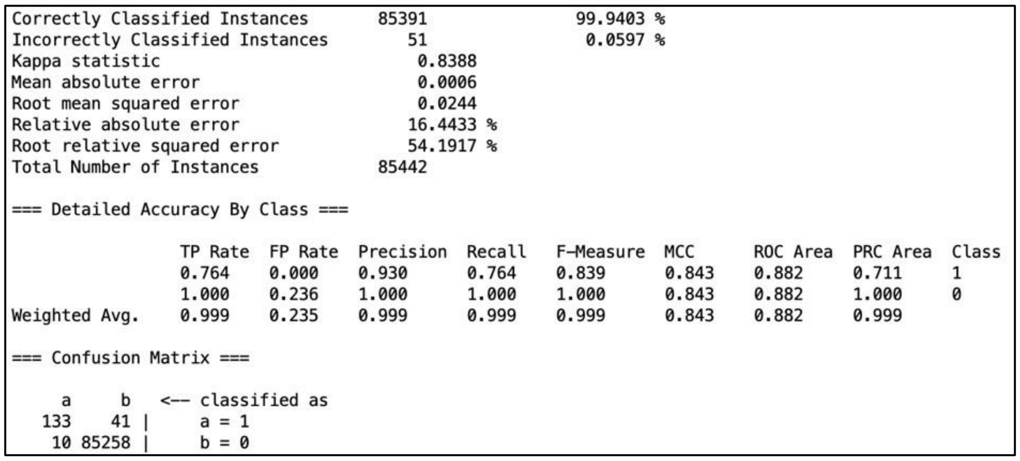
```
   FALSE  TRUE
0 91693      9
1    61    111

[1] 99.92381
```

Figure 13 - RStudio Logistic Regression

Support Vector Machine

Support Vector machine is a supervised ML technique with connected learning algorithms which inspect data used for both classification and regression analyses, it also performs linear classification, additionally to non-linear classification by creating margins between the classes, which are created in such a fashion that the space between the margin and the classes is maximum which minimizes the error of the classification (Mahesh, 2020).
Finally, the model Support Vector Machine as show in figure 12 managed to score

99.94% for the accuracy and misclassified 51 instances.



```
Correctly Classified Instances       85391               99.9403 %
Incorrectly Classified Instances        51                0.0597 %
Kappa statistic                          0.8388
Mean absolute error                      0.0006
Root mean squared error                  0.0244
Relative absolute error                 16.4433 %
Root relative squared error             54.1917 %
Total Number of Instances            85442

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.764    0.000    0.930      0.764    0.839      0.843    0.882     0.711     1
                 1.000    0.236    1.000      1.000    1.000      0.843    0.882     1.000     0
Weighted Avg.    0.999    0.235    0.999      0.999    0.999      0.843    0.882     0.999

=== Confusion Matrix ===

     a     b   <-- classified as
   133    41 |     a = 1
    10 85258 |     b = 0
```

Figure 14 - Support Vector Machine

Evaluation and Deployment

The last stage of the CRISP-DM model is the evaluation and deployment stage, as presented in table 2 below all models are being compared to each other to figure the best model in identifying fraudulent credit card transactions.
Accuracy is the overall number of instances that are predicted correctly, accuracies are represented by confusion matrix where it showed the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). True Positive represents the transactions that are fraudulent and was correctly classified by the model as fraudulent. True Negative represents the not fraudulent transactions that were correctly predicted by the model as Not fraudulent. The third rating is False positive which represents the transaction that are fraudulent but was misclassified as not fraudulent. And finally False Negative which are the not fraudulent transactions that were identified as fraudulent, table 1 below shows the confusion matrix.

Actual/Predicted	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Table 1 - Confusion Matrix

The table above shows all the components to calculate an accuracy of a model which is displayed in the below equation.

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

Model		Accuracy
KNN	K = 3	99.89%
	K = 3	
	K = 7	99.88%
	K = 7	
Naïve Bayes	Naïve Bayes	97.76%
	Naïve Bayes	
Logistic Regression	Logistic Regression	99.92%
	Logistic Regression	
Support Vector Machine	SVM	99.94%

Table 2 - Table of Accuracies

Table 2 shows all of the accuracies of all the models that were created in the project, all models performed well in detecting fraudulent transactions and managed to score high accuracies. Out of all the models the model that scored the best is Support Vector Machine as its accuracy is 99.94%, the second best is Logistic Regression, then in third

SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at
exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be Exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Data collection and Performance Metrics

Non-functional requirement

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. Non-functional requirements add tremendous value to business analysis. It is commonly misunderstood by a lot of people. It is important for business stakeholders, and Clients to clearly explain the requirements and their expectations in measurable terms. If the non-functional requirements are not measurable then they should be revised or rewritten to gain better clarity. For example, User stories help in mitigating the gap between developers and the user community in Agile Methodology.

Usability:

Prioritize the important functions of the system based on usage patterns. Frequently used functions should be tested for usability, as should complex and critical functions. Be sure to create a requirement for this.

Reliability:

Reliability defines the trust in the system that is developed after using it for a period of time. It defines the likeability of the software to work without failure for a given time period.

The number of bugs in the code, hardware failures, and problems can reduce the reliability of the software.

Your goal should be a long MTBF (mean time between failures). It is defined as the average period of time the system runs before failing.

Create a requirement that data created in the system will be retained for a number of years without the data being changed by the system.

It’s a good idea to also include requirements that make it easier to monitor system performance.

Performance:

What should system response times be, as measured from any point, under what circumstances?

Are there specific peak times when the load on the system will be unusually high?

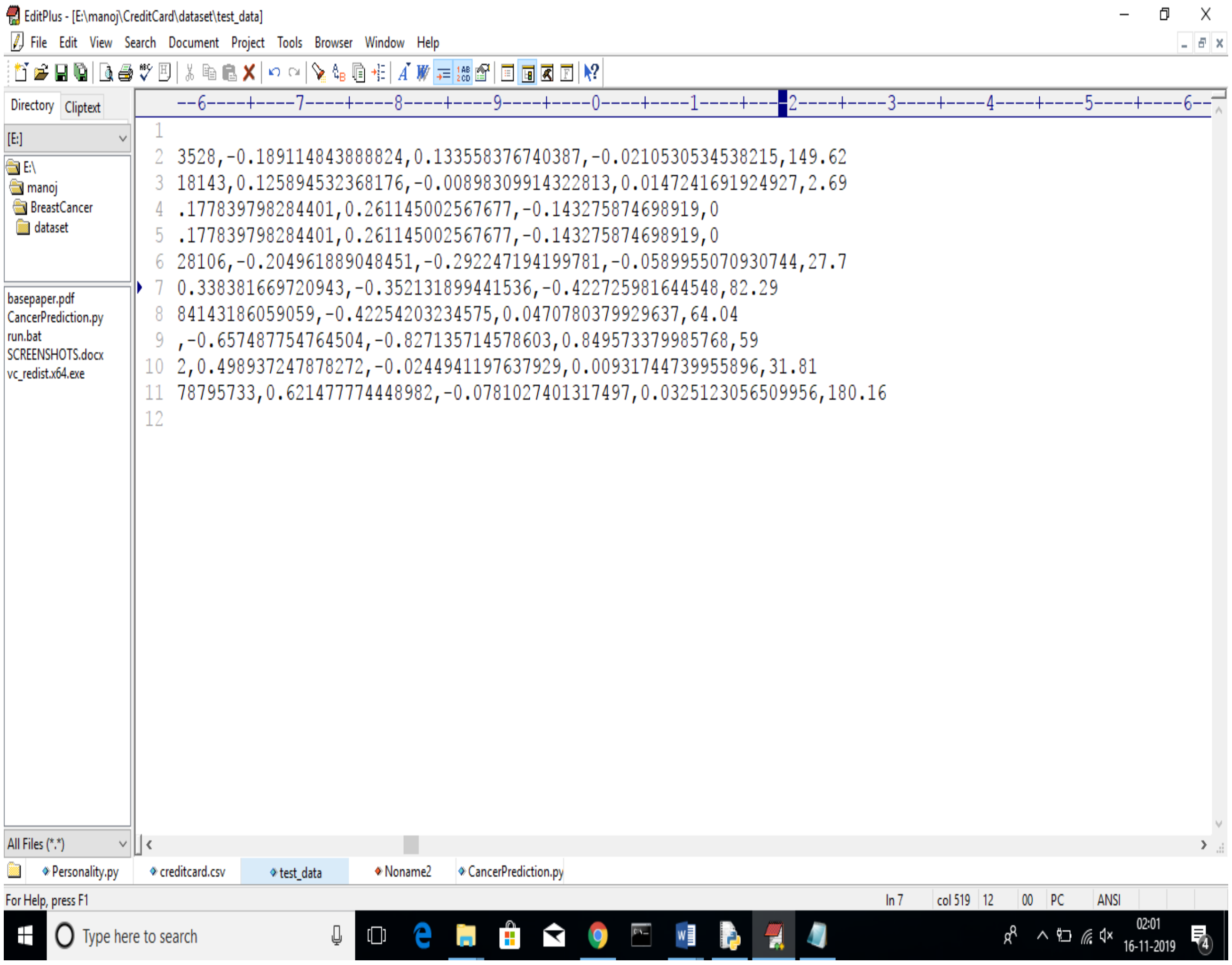
Think of stress periods, for example, at the end of the month or in conjunction with payroll disbursement.

Supportability:

The system needs to be **cost-effective to maintain**.

Maintainability requirements may cover diverse levels of documentation, such as system documentation, as well as test documentation, e.g. which test cases and test plans will accompany the system.

Using above 'CreditCardFraud.csv' file we will train Random Forest algorithm and then we will upload test data file and this test data will be applied on Random Forest train model to predict whether test data contains normal or fraud transaction signatures. When we upload test data then it will contains only transaction data no class label will be there application will predict and give the result. See below test data file

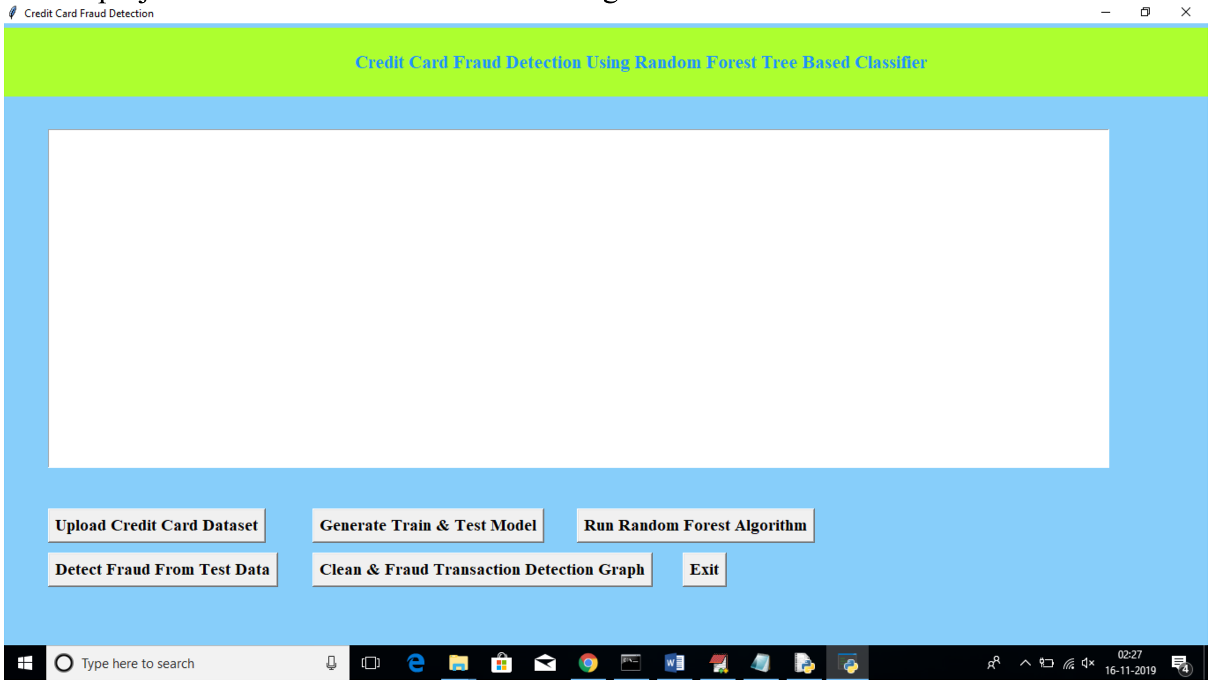


```

3528,-0.189114843888824,0.133558376740387,-0.0210530534538215,149.62
18143,0.125894532368176,-0.00898309914322813,0.0147241691924927,2.69
.177839798284401,0.261145002567677,-0.143275874698919,0
.177839798284401,0.261145002567677,-0.143275874698919,0
28106,-0.204961889048451,-0.292247194199781,-0.0589955070930744,27.7
0.338381669720943,-0.352131899441536,-0.422725981644548,82.29
84143186059059,-0.42254203234575,0.0470780379929637,64.04
,-0.657487754764504,-0.827135714578603,0.849573379985768,59
2,0.498937247878272,-0.0244941197637929,0.00931744739955896,31.81
78795733,0.621477774448982,-0.0781027401317497,0.0325123056509956,180.16

```

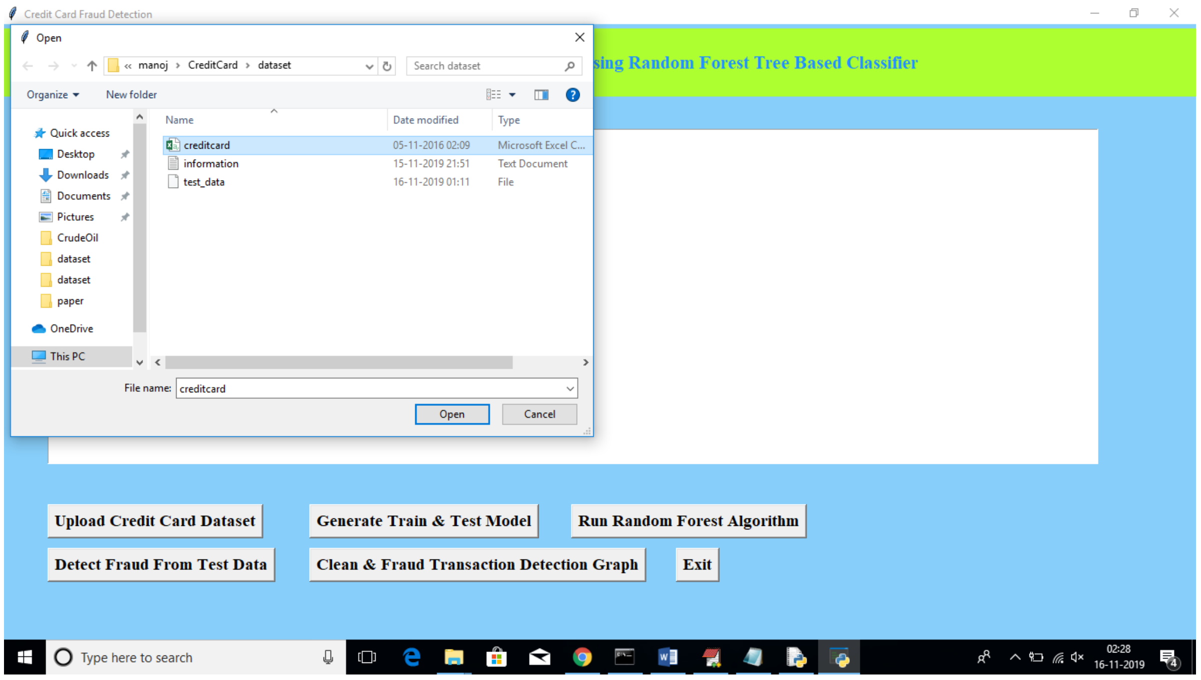
4.3 Results screenshot's

Screen shots

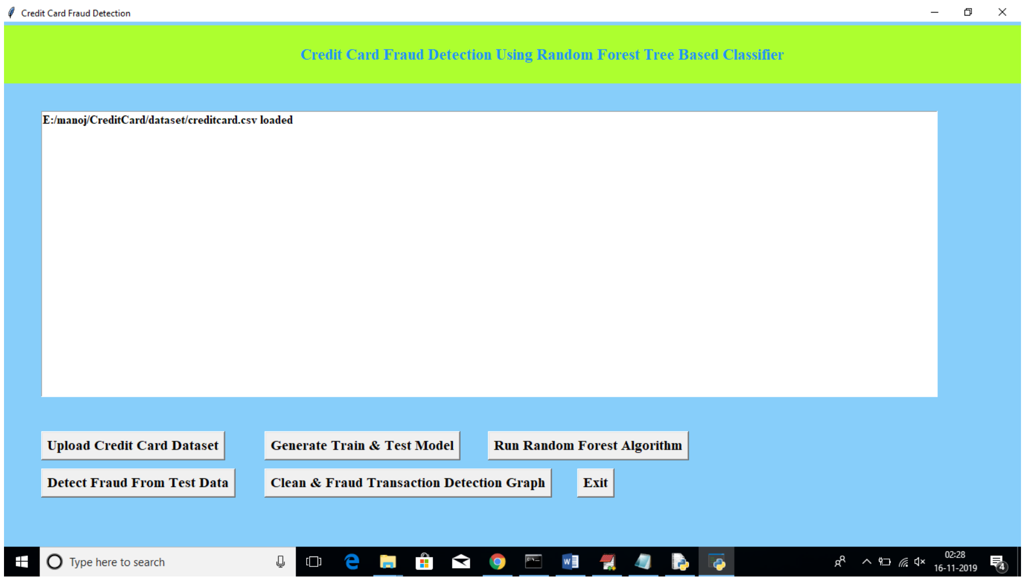
To run project double click on 'run.bat' file to get below screen


Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit
Type here to search
02:27
16-11-2019

In above screen click on 'Upload Credit Card Dataset' button to upload dataset

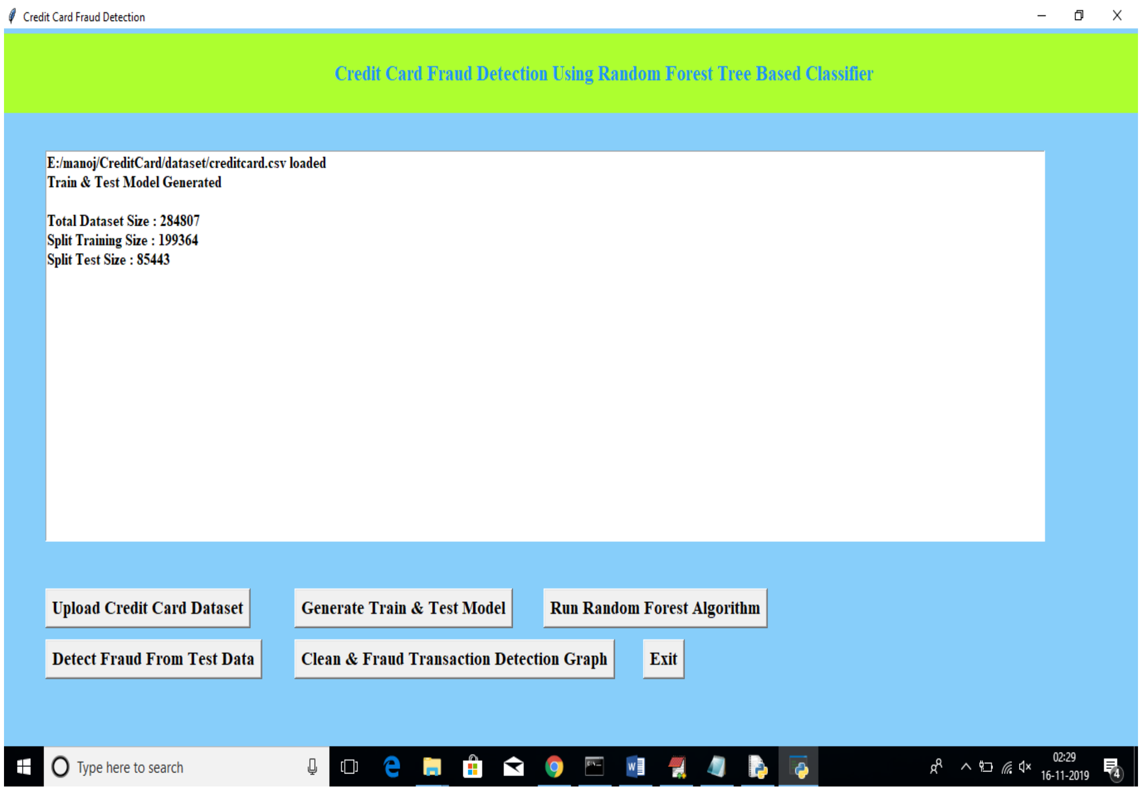


After uploading dataset will get below screen


Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit
Type here to search
02:28
16-11-2019

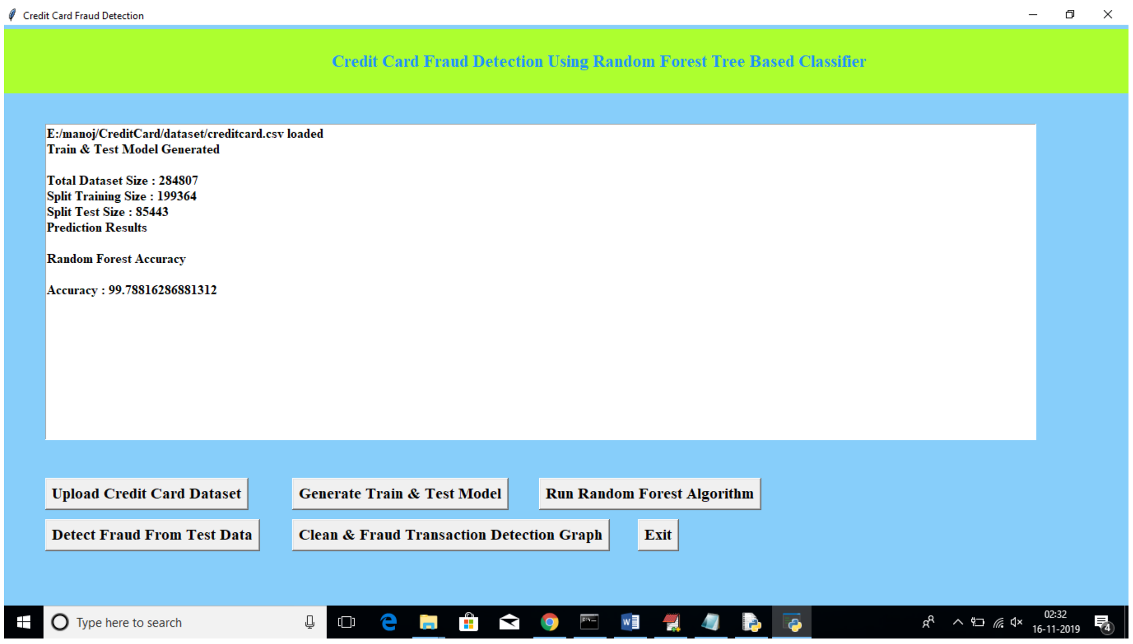
```
E:/manoj/CreditCard/dataset/creditcard.csv loaded
```

Now click on ‘Generate Train & Test Model’ to generate training model for Random Forest Classifier


Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit
Type here to search
02:29
16-11-2019

```
E:/manoj/CreditCard/dataset/creditcard.csv loaded
Train & Test Model Generated

Total Dataset Size : 284807
Split Training Size : 199364
Split Test Size : 85443
```

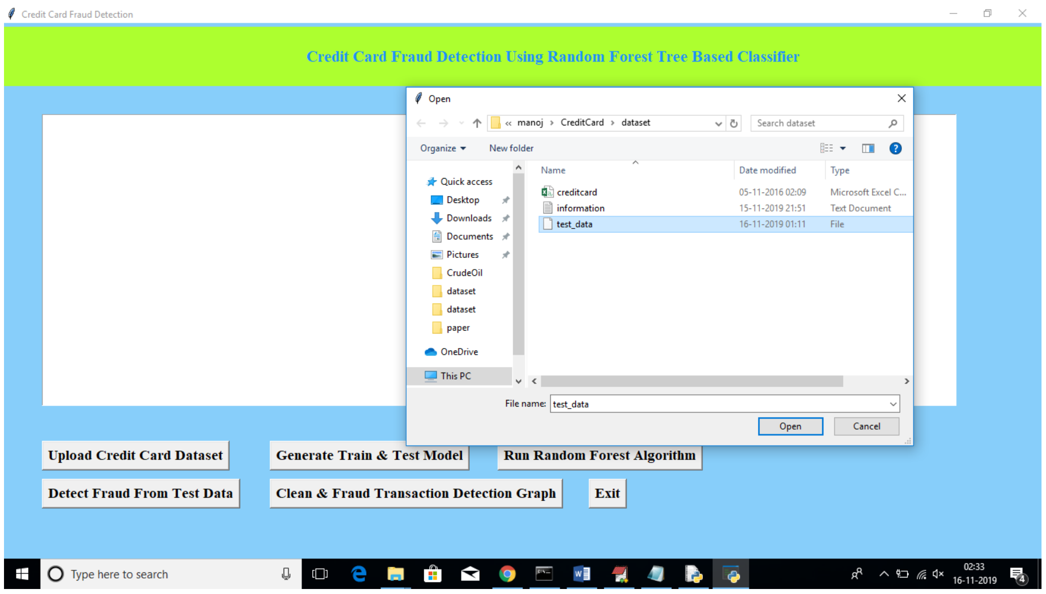
In above screen after generating model we can see total records available in dataset and then application using how many records for training and how many for testing. Now click on "Run Random Forest Algorithm' button to generate Random Forest model on train and test data


Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier

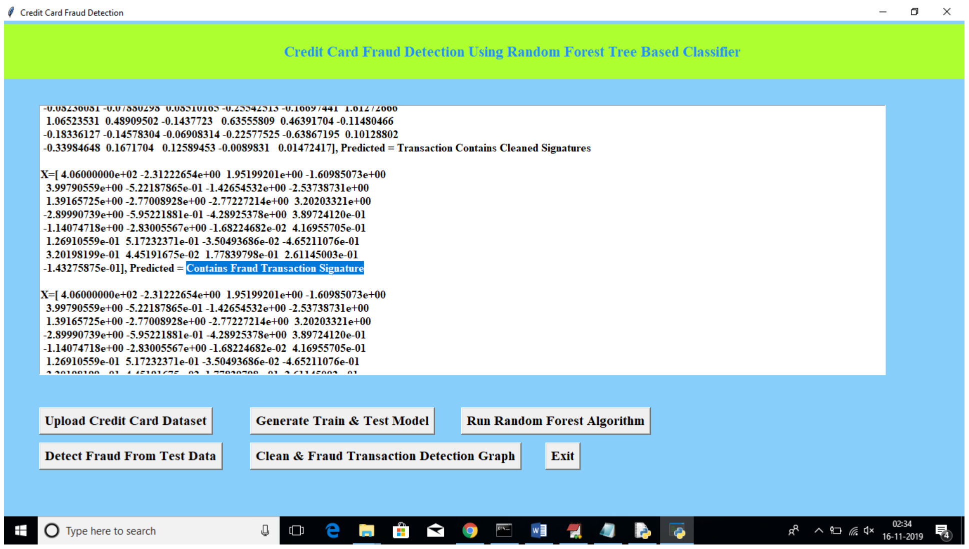
```
E:/manoj/CreditCard/dataset/creditcard.csv loaded
Train & Test Model Generated

Total Dataset Size : 284807
Split Training Size : 199364
Split Test Size : 85443
Prediction Results

Random Forest Accuracy

Accuracy : 99.78816286881312
```

Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit
Type here to search
02:32
16-11-2019

In above screen we can see Random Forest generate 99.78% percent accuracy while building model on train and test data. Now click on ‘Detect Fraud From Test Data’ button to upload test data and to predict whether test data contains normal or fraud transaction

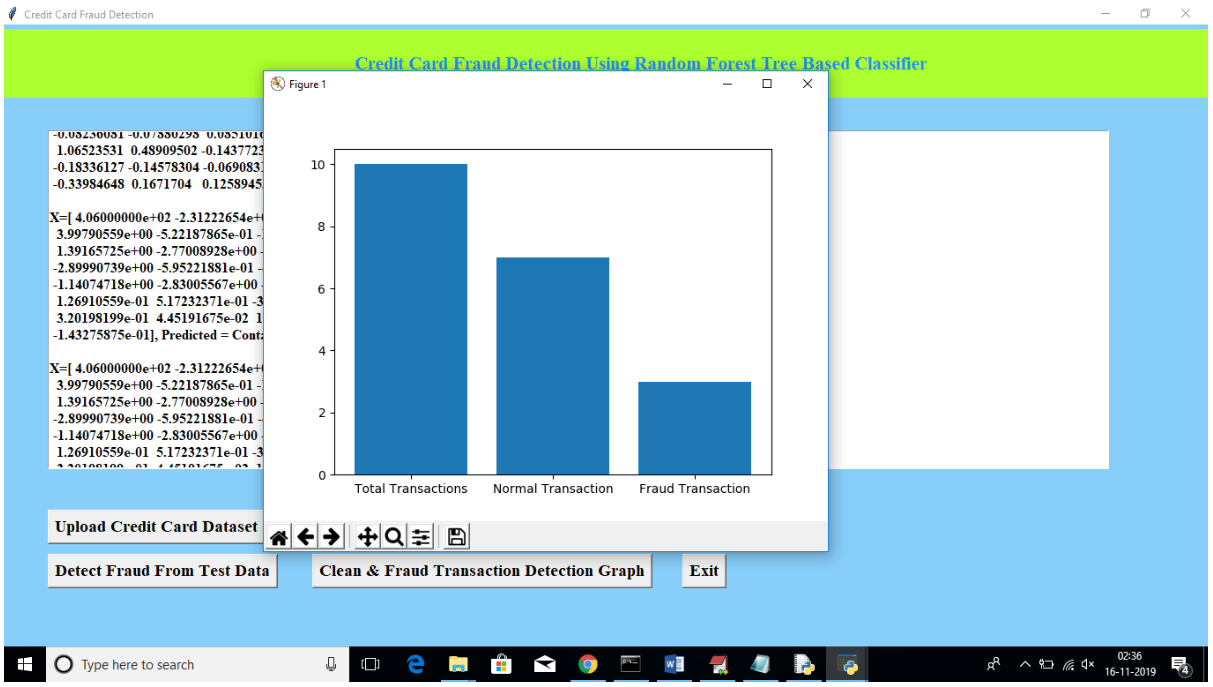


In above screen I am uploading test dataset and after uploading test data will get below prediction details


Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit
Type here to search
02:34
16-11-2019

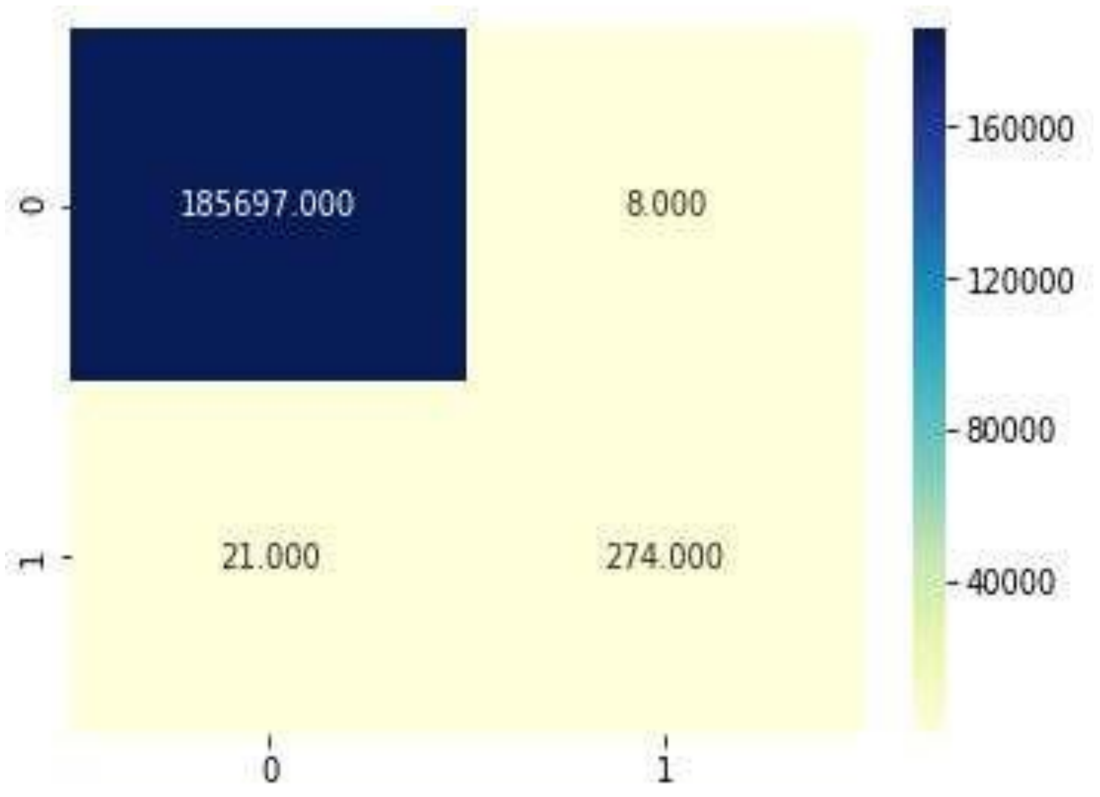
```
-0.05236081 -0.07880298  0.08510165 -0.25542513 -0.16697441  1.61272666
 1.06523531  0.48909502 -0.1437723   0.63555809  0.46391704 -0.11480466
-0.18336127 -0.14578304 -0.06908314 -0.22577525 -0.63867195  0.10128802
-0.33984648  0.1671704   0.12589453 -0.0089831   0.01472417], Predicted = Transaction Contains Cleaned Signatures

X=[ 4.06000000e+02 -2.31222654e+00  1.95199201e+00 -1.60985073e+00
 3.99790559e+00 -5.22187865e-01 -1.42654532e+00 -2.53738731e+00
 1.39165725e+00 -2.77008928e+00 -2.77227214e+00  3.20203321e+00
-2.89990739e+00 -5.95221881e-01 -4.28925378e+00  3.89724120e-01
-1.14074718e+00 -2.83005567e+00 -1.68224682e-02  4.16955705e-01
 1.26910559e-01  5.17232371e-01 -3.50493686e-02 -4.65211076e-01
 3.20198199e-01  4.45191675e-02  1.77839798e-01  2.61145003e-01
-1.43275875e-01], Predicted = Contains Fraud Transaction Signature

X=[ 4.06000000e+02 -2.31222654e+00  1.95199201e+00 -1.60985073e+00
 3.99790559e+00 -5.22187865e-01 -1.42654532e+00 -2.53738731e+00
 1.39165725e+00 -2.77008928e+00 -2.77227214e+00  3.20203321e+00
-2.89990739e+00 -5.95221881e-01 -4.28925378e+00  3.89724120e-01
-1.14074718e+00 -2.83005567e+00 -1.68224682e-02  4.16955705e-01
 1.26910559e-01  5.17232371e-01 -3.50493686e-02 -4.65211076e-01
 3.20198199e-01  4.45191675e-02  1.77839798e-01  2.61145003e-01
```

In above screen beside each test data application will display output as whether transaction contains cleaned or fraud signatures. Now click on ‘Clean & Fraud Transaction Detection Graph’ button to see total test transaction with clean and fraud signature in graphical format. See below screen


Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier
Figure 1
0
2
4
6
8
10
Total Transactions
Normal Transaction
Fraud Transaction
Upload Credit Card Dataset
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit
Type here to search
02:36
16-11-2019

```
-0.08236081 -0.07880298  0.0851016
 1.06523531  0.48909502 -0.1437723
-0.18336127 -0.14578304 -0.0690831
-0.33984648  0.1671704   0.1258945

X=[ 4.06000000e+02 -2.31222654e+
 3.99790559e+00 -5.22187865e-01 -
 1.39165725e+00 -2.77008928e+00 -
-2.89990739e+00 -5.95221881e-01 -
-1.14074718e+00 -2.83005567e+00 -
 1.26910559e-01  5.17232371e-01 -3
 3.20198199e-01  4.45191675e-02  1
-1.43275875e-01], Predicted = Cont

X=[ 4.06000000e+02 -2.31222654e+
 3.99790559e+00 -5.22187865e-01 -
 1.39165725e+00 -2.77008928e+00 -
-2.89990739e+00 -5.95221881e-01 -
-1.14074718e+00 -2.83005567e+00 -
 1.26910559e-01  5.17232371e-01 -3
```

In above graph we can see total test data and number of normal and fraud transaction detected. In above graph x-axis represents type and y-axis represents count of clean and fraud transaction



Exact figures of fake and original credit card

CHAPTER 5
CONCLUSION

In conclusion, the main objective of this project was to find the most suited model in credit card fraud detection in terms of the machine learning techniques chosen for the project, and it was met by building the four models and finding the accuracies of them all, the best model in terms of accuracies is Support Vector Machine which scored 99.94% with only 51 misclassified instances. I believe that using the model will help in decreasing the amount of credit card fraud and increase the customers satisfaction as it will provide them with better experience in addition to feeling secure.

There are many ways to improve the model, such as using it on different datasets with various sizes, different data types or by changing the data splitting ratio, in addition to viewing it from different algorithm perspective. An example can be merging telecom data to calculate the location of people to have better knowledge of the location of the card owner while his/her credit card is being used, this will ease the detection because if the card owner is in Dubai and a transaction of his card was made in Abu Dhabi it will easily be detected as fraud.

REFERENCE

[1] Sudhamathy G: Credit Risk Analysis and Prediction Modelling of Bank Loans Using R, vol. 8, no-5, pp. 1954-1966.

[2] LI Changjian, HU Peng: Credit Risk Assessment for ural Credit Cooperatives based on Improved Neural Network, International Conference on Smart Grid and Electrical Automation vol. 60, no. - 3, pp 227-230, 2017.

[3] Wei Sun, Chen-Guang Yang, Jian-Xun Qi: Credit Risk Assessment in Commercial Banks Based On Support Vector Machines, vol.6, pp 2430-2433, 2006.

[4]https://www.kaggle.com/code/hassanamin/credit-card-fraud-detection-using-random-forest/notebook

[5].https://www.tutorialspoint.com/what-is-a-neural-network-in-machine-learning

[6].https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorithm

GITHUB LINK