# Credit Card Fraud Detection-Innovation

## Abstract

The purpose of this project is to detect the fraudulent transactions made by credit cards by the use of machine learning techniques, to stop fraudsters from the unauthorized usage of customers' accounts. The increase of credit card fraud is growing rapidly worldwide, which is the reason actions should be taken to stop fraudsters. Putting a limit for those actions would have a positive impact on the customers as their money would be recovered and retrieved back into their accounts and they won't be charged for items or services that were not purchased by them which is the main goal of the project. Detection of the fraudulent transactions will be made by using three machine learning techniques KNN, SVM and Logistic Regression, those models will be used on a credit card transaction dataset.

## Introduction

We are using credit card daily for our expenses. In a physical-card based purchase, the card holder presents his card physically to a merchant for making a payment. If the card holder does not realize the loss of card, it can lead to a substantial financial loss to the credit card company. Credit Card Misrepresentation is one of the greatest dangers to business and business foundations today. Just, Master card Misrepresentation is characterized as, "when an individual uses another individuals‟. 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 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. In order to minimize costs of detection it is important to use expert rules and statistical based models to make a first screen between genuine and potential fraud and ask the investigators to review only the cases with high risk. When a fraud cannot be prevented, it is desirable to identify. The number of domain constraints and characteristics exaggerate the problem of detection and prevention. Customer irritation is to be avoided. Most banks considers huge transactions, among which very few is fraudulent, often less. Also, only a limited number of transactions can be checked by fraud investigators, i.e. we cannot ask a human person to check all transactions one by one if it is fraudulent or not.

# Implementation

## ➢ Python Libraries

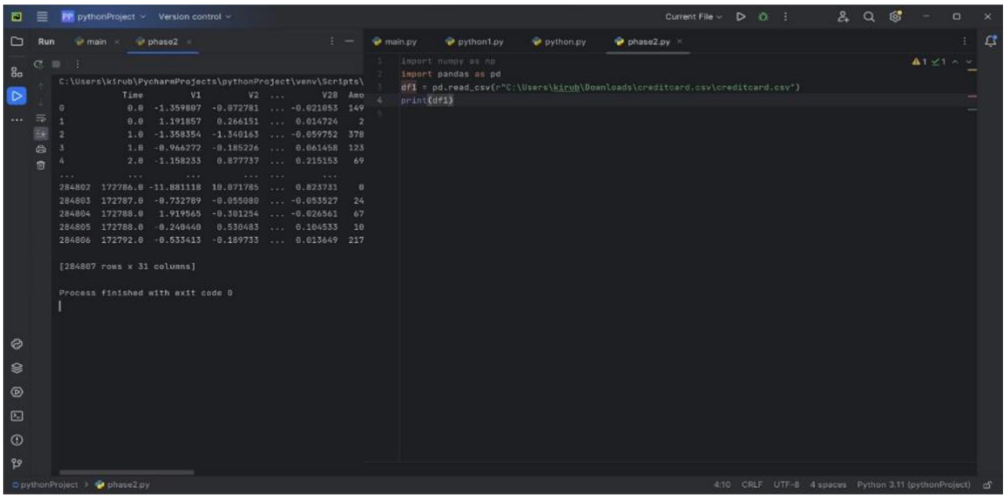
Python libraries are collections of modules that contain useful codes and functions, eliminating the need to write them from scratch. There are tens of thousands of Python libraries that help machine learning developers, as well as professionals working in data science, data visualization, and more.

- ❖ Pandas
- ❖ Numpy
- ❖ Mtplotlib
- ❖ Sci-Kit Learn

## ➢ Data Reading and Activities

The dataset was retrieved from an open-source website, Kaggle.com. it contains data of transactions that were made in 2013 by credit card users in Europe, in two days only. The dataset consists of 31 attributes, 284,808 rows. 28 attributes are numeric variables that due to confidentiality and privacy of the customers have been transformed using PCA transformation, the three remaining attributes are "Time" which contains the elapsed seconds between the first and other transactions of each attribute, "Amount" is the amount of each transaction, and the final attribute "Class" which contains binary variables where "1" is a case of fraudulent transaction, and "0" is not as case of fraudulent transaction.

Dataset Link: [https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud](https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)



```
import numpy as np
import pandas as pd
df1 = pd.read_csv(r"C:\Users\kirub\Downloads\creditcard.csv\creditcard.csv")
print(df1)
```

```
C:\Users\kirub\PycharmProjects\pythonProject\venv\Scripts\
            Time         V1         V2  ...       V28  Amo
0            0.0  -1.359807  -0.072781  ... -0.021053  149
1            0.0   1.191857   0.266151  ...  0.014724    2
2            1.0  -1.358354  -1.340163  ... -0.059752  378
3            1.0  -0.966272  -0.185226  ...  0.061458  123
4            2.0  -1.158233   0.877737  ...  0.215153   69
...          ...        ...        ...  ...       ...
284802  172786.0 -11.881118  10.071785  ...  0.823731    0
284803  172787.0  -0.732789  -0.055080  ... -0.053527   24
284804  172788.0   1.919565  -0.301254  ... -0.026561   67
284805  172788.0  -0.240440   0.530483  ...  0.104533   10
284806  172792.0  -0.533413  -0.189733  ...  0.013649  217

[284807 rows x 31 columns]

Process finished with exit code 0
```

## ➢ Activity Steps:

1) Data Collection

2)Data Preparation.

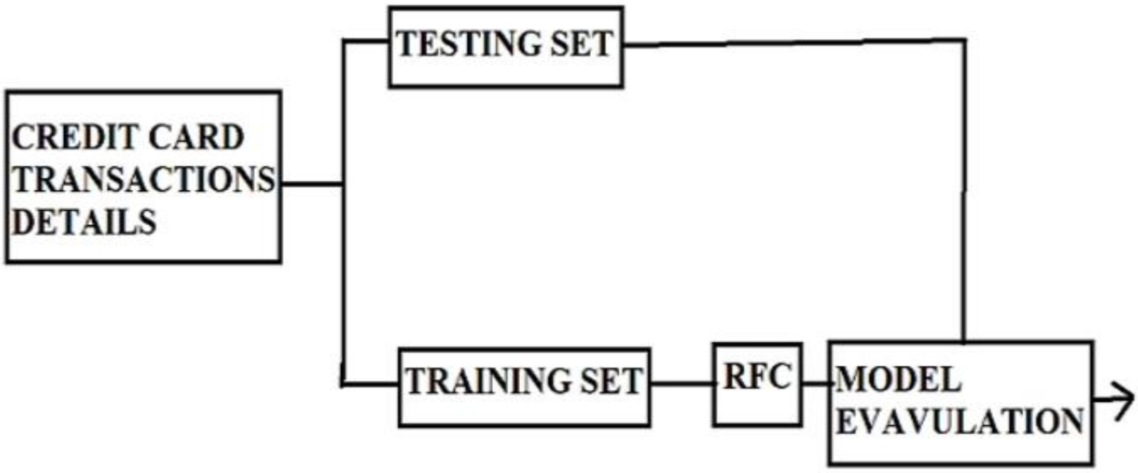
3)Choose a Model.

4)Train the Model.

5)Evaluate the Model.

6)Parameter Tuning.

7)Make Predictions.



## ➢ Data Cleaning:

The preparation of the dataset includes selecting the wanted attributes or variables, cleaning it by excluding Null rows, deleting duplicated variables, treating outlier if necessary, in addition to transforming data types to the wanted type, data merging can be performed as well where two or more attributes get merged. All those alterations lead to the wanted result which is to make the data ready to be modeled.

## EDA(Exploratory Data Analysis)

Exploratory Data Analysis (EDA) refers to the method of studying and exploring record sets to apprehend their predominant traits, discover patterns, locate outliers, and identify relationships between variables.

Steps of EDA

1)variable Identification

2)Univariant Analysis

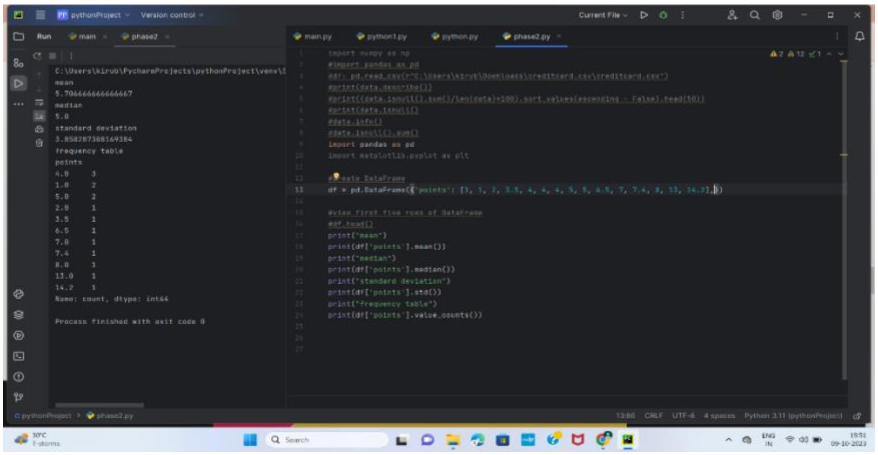
3)Bivariant Analysis

4)Missing value

5)Outlier Handling

## ➢ Univriant Analysis

Univriant Analysis refers to the analysis of one variable.

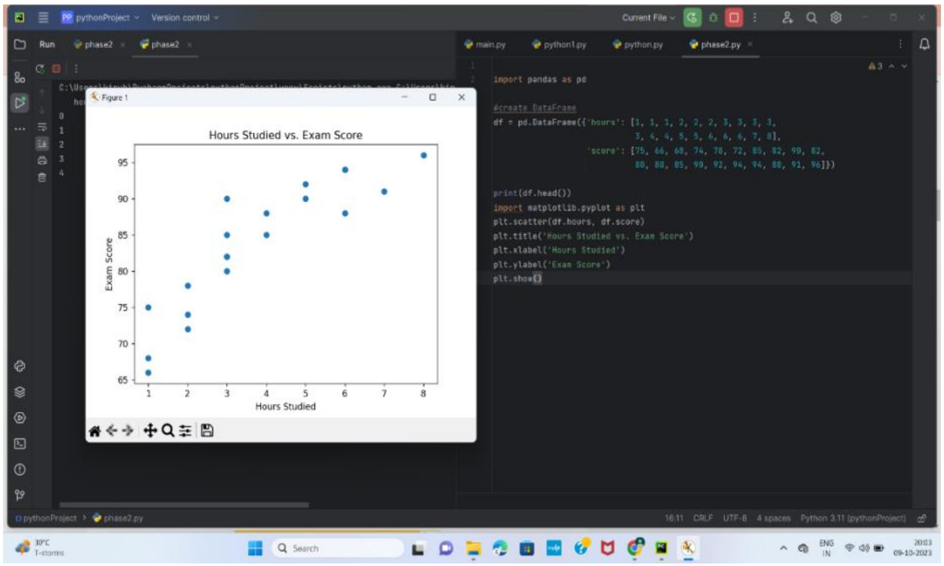


```
C:\Users\kirub\PycharmProjects\pythonProject\venv\S
mean
5.706666666666667
median
5.0
standard deviation
3.858787388169384
frequency table
points
4.0     3
1.0     2
5.0     2
2.0     1
3.5     1
6.5     1
7.0     1
7.4     1
8.0     1
13.0    1
14.2    1
Name: count, dtype: int64

Process finished with exit code 0
```

```python
import numpy as np
#import pandas as pd
#df= pd.read_csv(r"C:\Users\kirub\Downloads\creditcard.csv\creditcard.csv")
#print(data.describe())
#print((data.isnull().sum()/len(data)*100).sort_values(ascending = False).head(50))
#print(data.isnull()
#data.info()
#data.isnull().sum()
import pandas as pd
import matplotlib.pyplot as plt

#create DataFrame
df = pd.DataFrame({'points': [1, 1, 2, 3.5, 4, 4, 4, 5, 5, 6.5, 7, 7.4, 8, 13, 14.2],})

#view first five rows of DataFrame
#df.head()
print("mean")
print(df['points'].mean())
print("median")
print(df['points'].median())
print("standard deviation")
print(df['points'].std())
print("frequency table")
print(df['points'].value_counts())
```

## ➢ Bivariate analysis

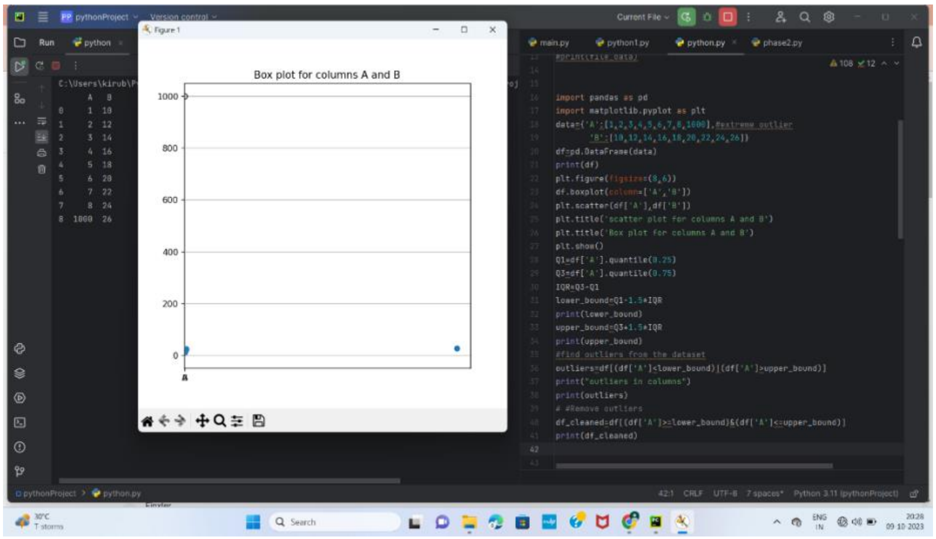
Bivariate analysis refers to the analysis of two variables to perform bivariate analysis Scatterplots:



```python
import pandas as pd

#create DataFrame
df = pd.DataFrame({'hours': [1, 1, 1, 2, 2, 2, 3, 3, 3, 3,
                             3, 4, 4, 5, 5, 6, 6, 6, 7, 8],
                   'score': [75, 66, 68, 74, 78, 72, 85, 82, 90, 82,
                             80, 88, 85, 90, 92, 94, 94, 88, 91, 96]})

print(df.head())
import matplotlib.pyplot as plt
plt.scatter(df.hours, df.score)
plt.title('Hours Studied vs. Exam Score')
plt.xlabel('Hours Studied')
plt.ylabel('Exam Score')
plt.show()
```

## ➢ Outliers

Outliers are those data points that are significantly different from the rest of the dataset.



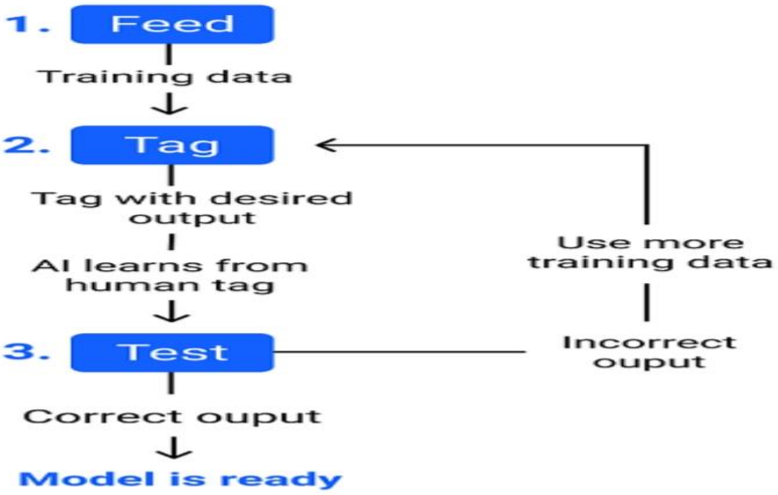
```
C:\Users\kirub\P
      A   B
0     1  10
1     2  12
2     3  14
3     4  16
4     5  18
5     6  20
6     7  22
7     8  24
8  1000  26
```

```
#print(file_data)


import pandas as pd
import matplotlib.pyplot as plt
data={'A':[1,2,3,4,5,6,7,8,1000],#extreme outlier
      'B':[10,12,14,16,18,20,22,24,26]}
df=pd.DataFrame(data)
print(df)
plt.figure(figsize=(8,6))
df.boxplot(column=['A','B'])
plt.scatter(df['A'],df['B'])
plt.title('scatter plot for columns A and B')
plt.title('Box plot for columns A and B')
plt.show()
Q1=df['A'].quantile(0.25)
Q3=df['A'].quantile(0.75)
IQR=Q3-Q1
lower_bound=Q1-1.5*IQR
print(lower_bound)
upper_bound=Q3+1.5*IQR
print(upper_bound)
#find outliers from the dataset
outliers=df[(df['A']<lower_bound)|(df['A']>upper_bound)]
print("outliers in columns")
print(outliers)
# #Remove outliers
df_cleaned=df[(df['A']>=lower_bound)&(df['A']<=upper_bound)]
print(df_cleaned)
```

## ➢ Data testing and training

train and test datasets are the two key concepts of machine learning, where the training dataset is used to fit the model, and the test dataset is used to evaluate the model.

- x_train: It is used to represent features for the training data
- x_test: It is used to represent features for testing data
- y_train: It is used to represent dependent variables for training data
- y_test: It is used to represent independent variable for testing data

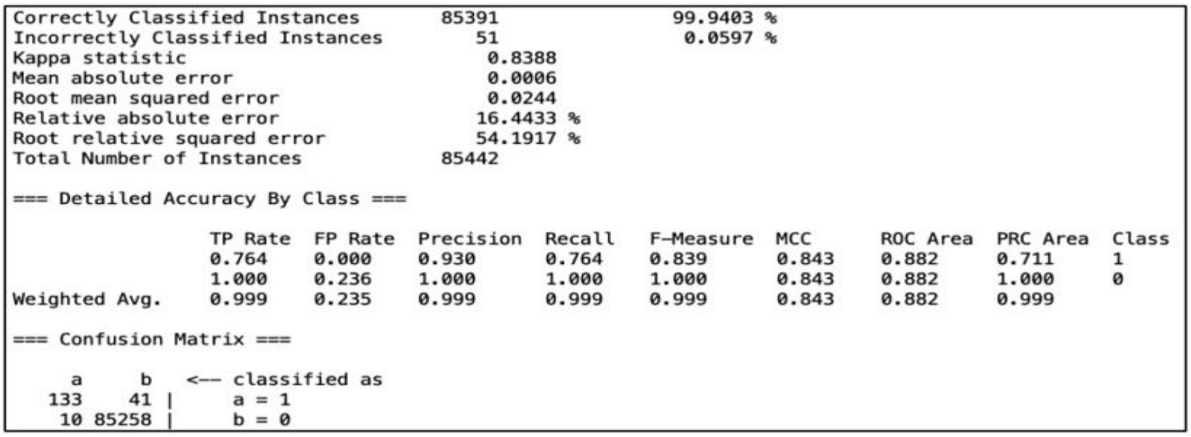


# METHODOLOGY

We are using supervised learning techniques to extract the information as a part of the fraud analysis. The dataset used is a binary classification. Fraud detection is a binary classification task in which any transaction will be predicted and labelled as a fraud or legit.classification techniques were tried for this task and their performances were compared. The following subsections briefly explain these classification techniques, data set and metrics used for performance measure.

## ➢ Support vector machines (SVM)

SVM is introduced in 1992 to solve binary classification problems and then they are extended to nonlinear regression problems. SVMs are based on structural risk minimization unlike ANNs which is based on empirical risk minimization. SVM map the data to a predetermined very high- dimensional space via a kernel function and finds the hyper plane that maximizes the margin between NOVATEUR PUBLICATIONS INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY. The solution is based only on those data points, which are at the margin. These points are called support vectors.



```
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
```

## ➢ Naive Bayes Algorithm

Naive Bayes is based on two assumptions. Firstly, all features in an entry that needs to be classified are contributing evenly in the decision Secondly, all attributes are statistically independent, meaning that, knowing an attribute's value does not indicate anything about other attributes' values which is not always true in practice. The process of classifying an instance is done by applying the Bayes rule for each class given the instance. In the fraud detection task, the following formula is calculated for each of the two classes and the class associated with the higher probability is the predicted class for the instance.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

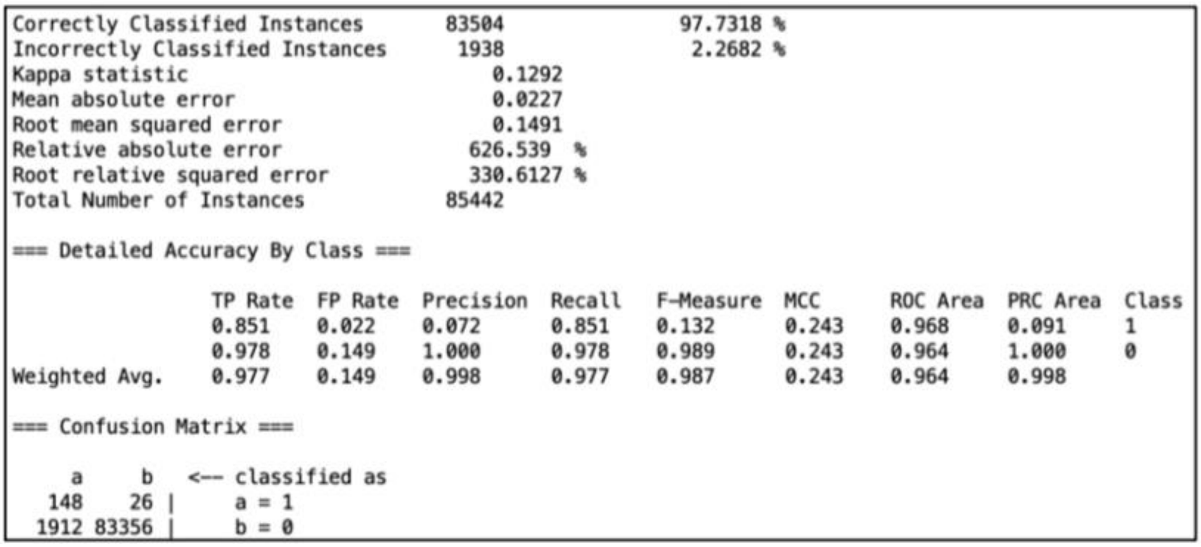
Steps:

1: Calculate the prior probability for given class labels

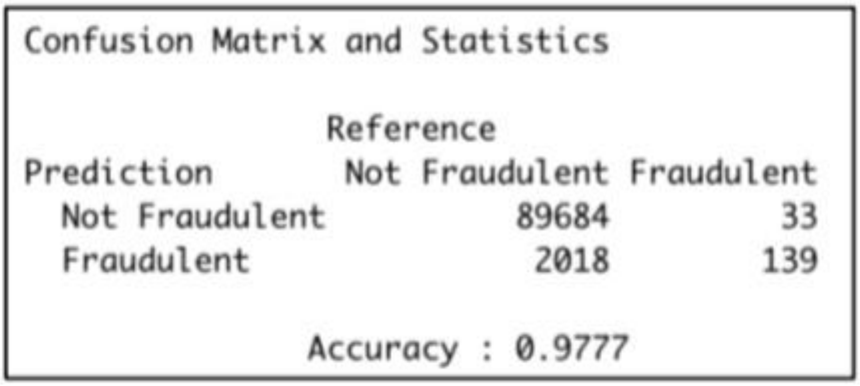
2: Find Likelihood probability with each attribute for each class

3: Put these value in Bayes Formula and calculate posterior probability.

4: See which class has a higher probability, given the input belongs to the higher probability class.



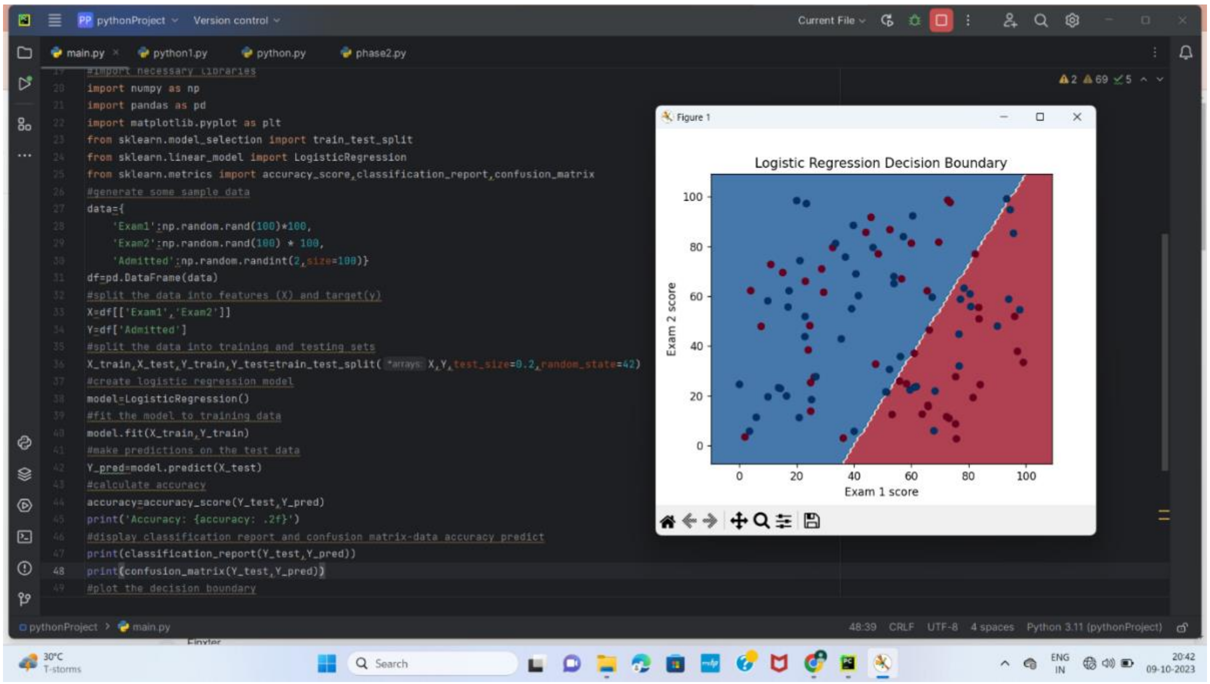
```
Correctly Classified Instances       83504               97.7318 %
Incorrectly Classified Instances      1938                2.2682 %
Kappa statistic                          0.1292
Mean absolute error                      0.0227
Root mean squared error                  0.1491
Relative absolute error                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
```



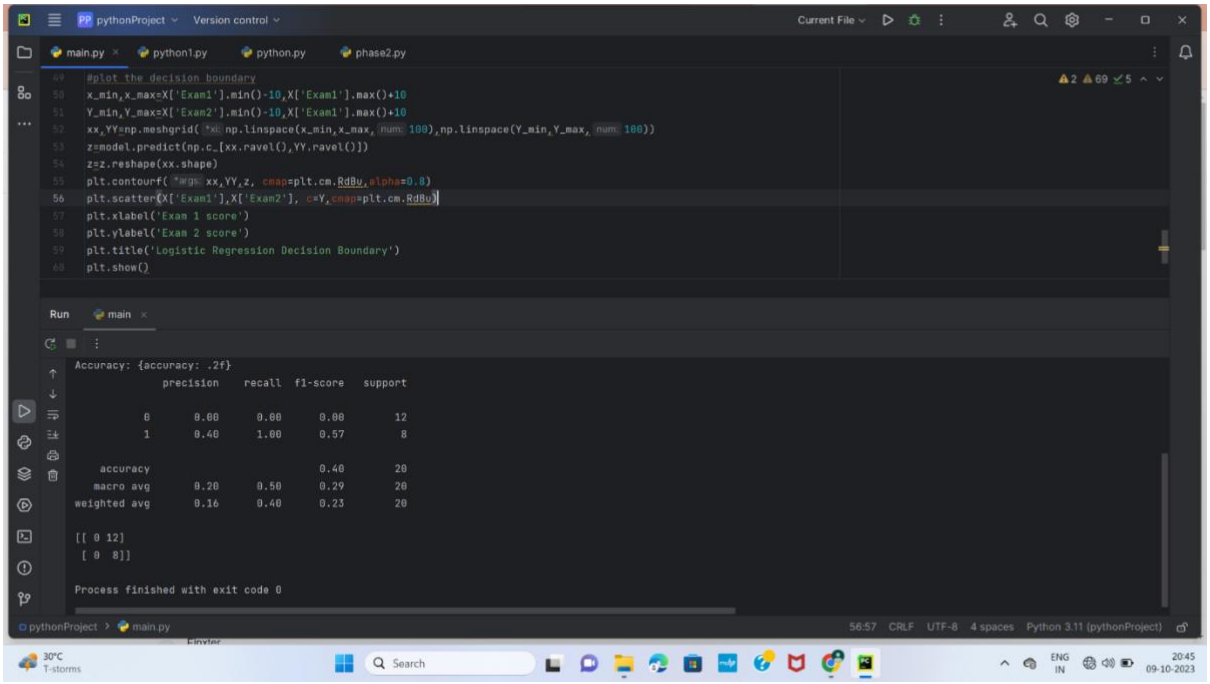
```
Confusion Matrix and Statistics

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

               Accuracy : 0.9777
```

### ➢ Logistic Regression

Logistic regression also does not require independent variables to be linearly related, nor does it require equal variance within each group, which also makes it a less stringent procedure for statistical analysis. As a result, logistic regression was used to predict the probability of fraudulent credit cards. Assumptions and Limitations of Logistic Regression. Logistic regression analysis uses maximum likelihood estimation to predict group membership. However, to interpret the results of the prediction of group membership with precision and accuracy, a preliminary analysis of the cleaned dataset was conducted to observe if the assumptions of logistic regression were met. The above techniques are used for the detection of fraud transactions involved with the banks. After the classification, the intensity of the individual cards is inquired and calculated, which prepares and classifies the credit card trend and spending based anomalies. The regression models are used to perform the operation on the given data stream obtained from the credit card company for the detection of the credit card frauds by analysing the spending behaviour of the customers.



```
#import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
#generate some sample data
data={
    'Exam1':np.random.rand(100)*100,
    'Exam2':np.random.rand(100) * 100,
    'Admitted':np.random.randint(2,size=100)}
df=pd.DataFrame(data)
#split the data into features (X) and target(y)
X=df[['Exam1','Exam2']]
Y=df['Admitted']
#split the data into training and testing sets
X_train,X_test,Y_train,Y_test=train_test_split( X,Y,test_size=0.2,random_state=42)
#create logistic regression model
model=LogisticRegression()
#fit the model to training data
model.fit(X_train,Y_train)
#make predictions on the test data
Y_pred=model.predict(X_test)
#calculate accuracy
accuracy=accuracy_score(Y_test,Y_pred)
print('Accuracy: {accuracy: .2f}')
#display classification report and confusion matrix-data accuracy predict
print(classification_report(Y_test,Y_pred))
print(confusion_matrix(Y_test,Y_pred))
#plot the decision boundary
```



```
#plot the decision boundary
x_min,x_max=X['Exam1'].min()-10,X['Exam1'].max()+10
Y_min,Y_max=X['Exam2'].min()-10,X['Exam1'].max()+10
xx,YY=np.meshgrid( np.linspace(x_min,x_max, 100),np.linspace(Y_min,Y_max, 100))
z=model.predict(np.c_[xx.ravel(),YY.ravel()])
z=z.reshape(xx.shape)
plt.contourf( xx,YY,z, cmap=plt.cm.RdBu,alpha=0.8)
plt.scatter(X['Exam1'],X['Exam2'], c=Y,cmap=plt.cm.RdBu)
plt.xlabel('Exam 1 score')
plt.ylabel('Exam 2 score')
plt.title('Logistic Regression Decision Boundary')
plt.show()
```

```
Accuracy: {accuracy: .2f}
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        12
           1       0.40      1.00      0.57         8

    accuracy                           0.40        20
   macro avg       0.20      0.50      0.29        20
weighted avg       0.16      0.40      0.23        20

[[ 0 12]
 [ 0  8]]

Process finished with exit code 0
```

## ➢ Linear Regression:

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables. Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

$$y = a0 + a1x + \varepsilon$$

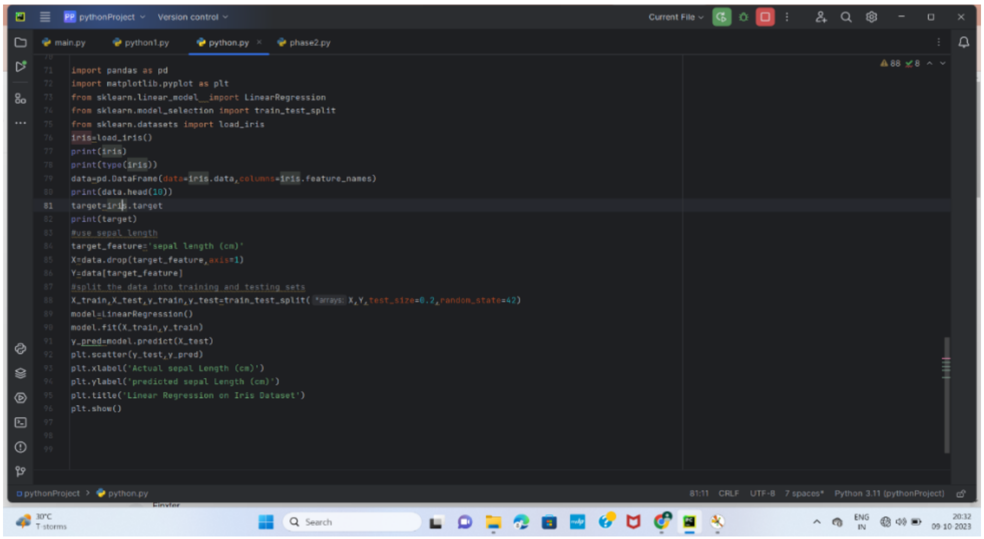
Where, Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

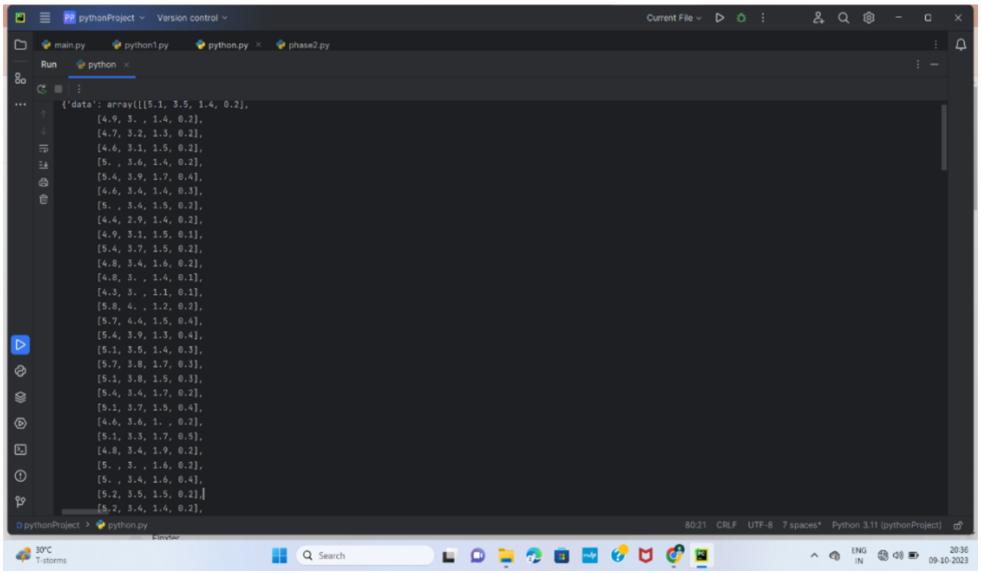
a0= intercept of the line (Gives an additional degree of freedom)

a1 = Linear regression coefficient (scale factor to each input value).

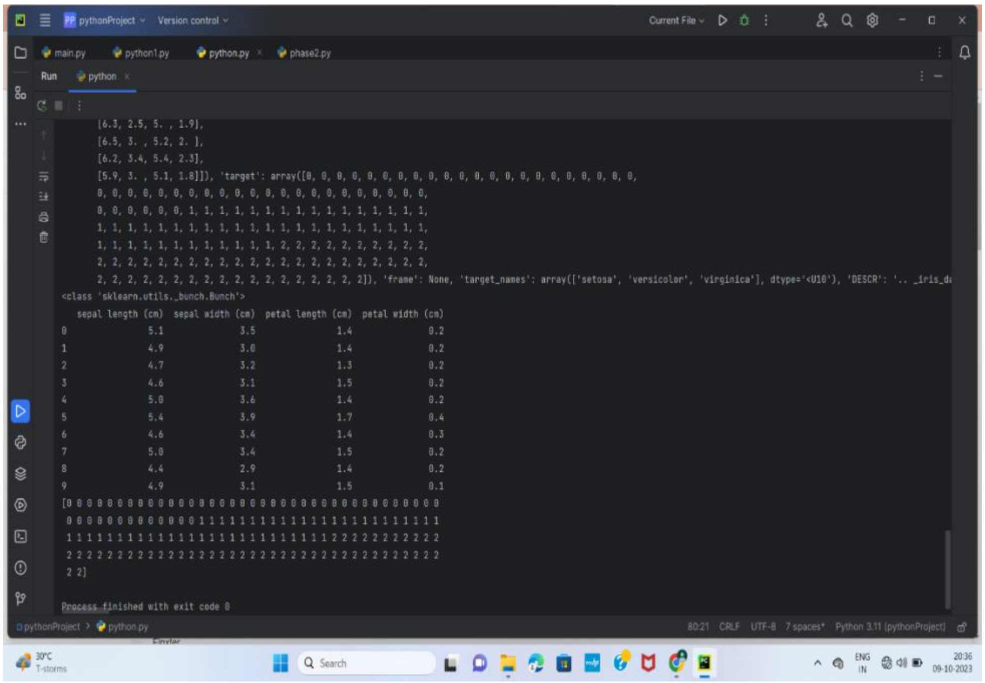
ε = random error



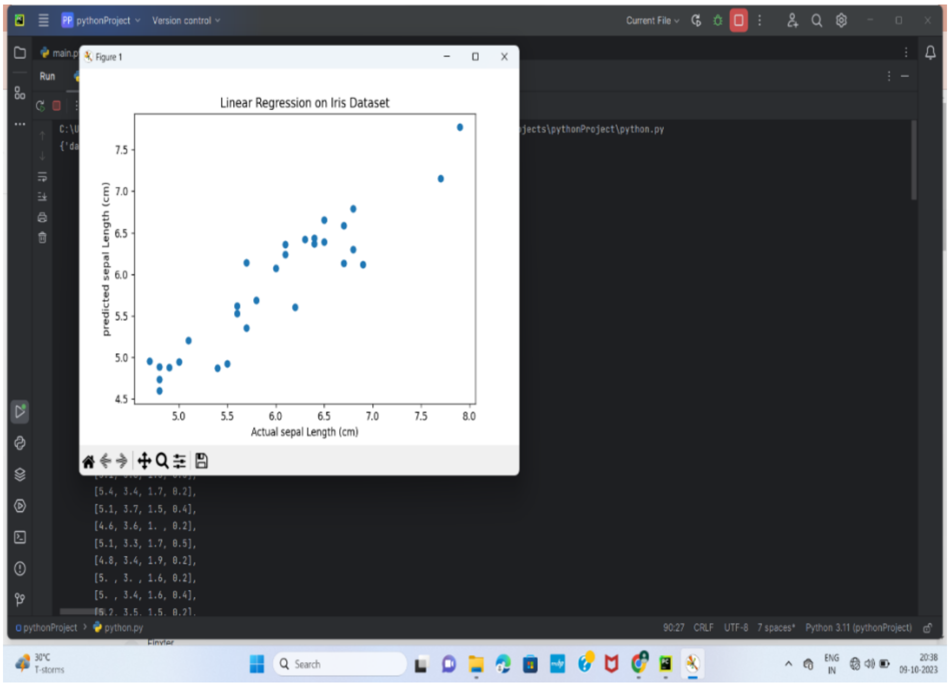
```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model  import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
iris=load_iris()
print(iris)
print(type(iris))
data=pd.DataFrame(data=iris.data,columns=iris.feature_names)
print(data.head(10))
target=iris.target
print(target)
#use sepal length
target_feature='sepal length (cm)'
X=data.drop(target_feature,axis=1)
Y=data[target_feature]
#split the data into training and testing sets
X_train,X_test,y_train,y_test=train_test_split( *arrays: X,Y,test_size=0.2,random_state=42)
model=LinearRegression()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
plt.scatter(y_test,y_pred)
plt.xlabel('Actual sepal Length (cm)')
plt.ylabel('predicted sepal Length (cm)')
plt.title('Linear Regression on Iris Dataset')
plt.show()
```



```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
```



```
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]), 'frame': None, 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'), 'DESCR': '.. _iris_d
<class 'sklearn.utils._bunch.Bunch'>
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
5                5.4               3.9                1.7               0.4
6                4.6               3.4                1.4               0.3
7                5.0               3.4                1.5               0.2
8                4.4               2.9                1.4               0.2
9                4.9               3.1                1.5               0.1
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]

Process finished with exit code 0
```

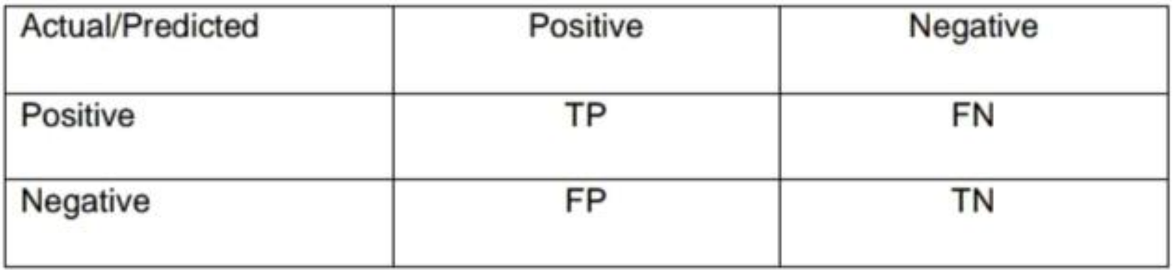


```
C:\U                                                jects\pythonProject\python.py
{'da
```

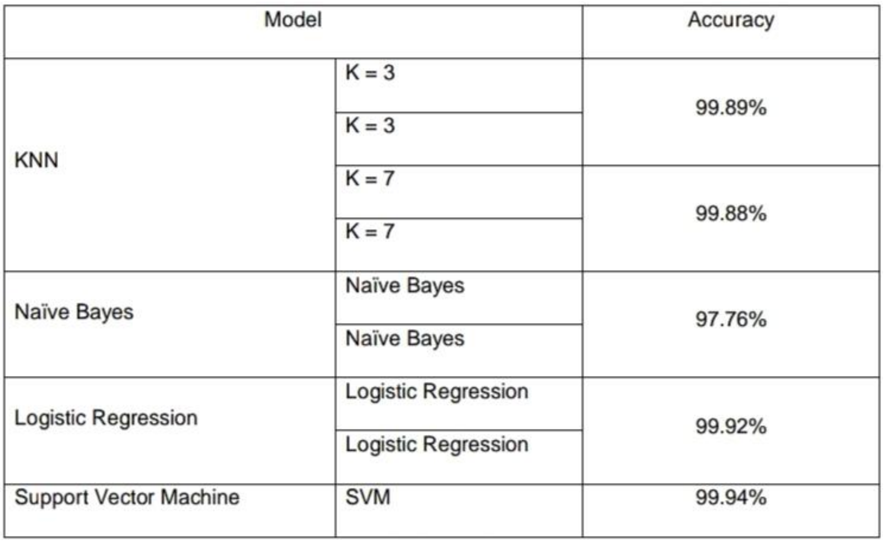
```
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
```

## ➢ Accuracy and Other Metrics for Prediction evaluation

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 |



| 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% |