IMPORT LIBRARIES

```
import pandas as pd
```

LOAD DATASETS

```
data=pd.read_csv('FastagFraudDetection.csv')
```

```
data.head()
```

	Transaction_ID	Timestamp	Vehicle_Type	FastagID	TollBoothID	Lane_Type	Vehicle_Dimensions	Transaction_Amount	Amount_paid	Geogr
0	1	1/6/2023 11:20	Bus	FTG-001-ABC-121	A-101	Express	Large	350	120	13. 7
1	2	1/7/2023 14:55	Car	FTG-002-XYZ-451	B-102	Regular	Small	120	100	13. 7
2	3	1/8/2023 18:25	Motorcycle	NaN	D-104	Regular	Small	0	0	13. 7
3	4	1/9/2023 2:05	Truck	FTG-044-LMN-322	C-103	Regular	Large	350	120	13. 7
4	5	1/10/2023 6:35	Van	FTG-505-DEF-652	B-102	Express	Medium	140	100	13. 7

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Transaction_ID        5000 non-null   int64
 1   Timestamp             5000 non-null   object
 2   Vehicle_Type          5000 non-null   object
 3   FastagID              4451 non-null   object
 4   TollBoothID           5000 non-null   object
 5   Lane_Type             5000 non-null   object
 6   Vehicle_Dimensions    5000 non-null   object
 7   Transaction_Amount    5000 non-null   int64
 8   Amount_paid           5000 non-null   int64
 9   Geographical_Location 5000 non-null   object
 10  Vehicle_Speed         5000 non-null   int64
 11  Vehicle_Plate_Number  5000 non-null   object
 12  Fraud_indicator       5000 non-null   object
dtypes: int64(4), object(9)
memory usage: 507.9+ KB
```

```
data.describe()
```

	Transaction_ID	Transaction_Amount	Amount_paid	Vehicle_Speed
count	5000.000000	5000.00000	5000.000000	5000.000000
mean	2500.500000	161.06200	141.261000	67.851200
std	1443.520003	112.44995	106.480996	16.597547
min	1.000000	0.00000	0.000000	10.000000
25%	1250.750000	100.00000	90.000000	54.000000
50%	2500.500000	130.00000	120.000000	67.000000
75%	3750.250000	290.00000	160.000000	82.000000
max	5000.000000	350.00000	350.000000	118.000000

```
data.isnull().sum()
```

```
Transaction_ID            0
Timestamp                 0
Vehicle_Type              0
```

```
FastagID                549
TollBoothID               0
Lane_Type                 0
Vehicle_Dimensions        0
Transaction_Amount        0
Amount_paid               0
Geographical_Location     0
Vehicle_Speed             0
Vehicle_Plate_Number      0
Fraud_indicator           0
dtype: int64
```

```
data.dropna(subset=['FastagID'],inplace=True)
```

```
data.describe()
```

	Transaction_ID	Transaction_Amount	Amount_paid	Vehicle_Speed
count	4451.000000	4451.000000	4451.000000	4451.000000
mean	2466.227140	180.927881	158.684565	67.884745
std	1428.941144	103.004437	99.857565	16.632295
min	1.000000	0.000000	0.000000	10.000000
25%	1254.500000	110.000000	100.000000	55.000000
50%	2405.000000	140.000000	120.000000	67.000000
75%	3702.500000	300.000000	180.000000	82.000000
max	5000.000000	350.000000	350.000000	118.000000

DATA PREPROCESSING

```
pd.concat([data['Vehicle_Type'],data['TollBoothID'],data['Lane_Type'],data['Vehicle_Dimensions']]).unique()
```

```
array(['Bus ', 'Car', 'Truck', 'Van', 'Sedan', 'SUV', 'Motorcycle',
       'A-101', 'B-102', 'C-103', 'D-106', 'Express', 'Regular', 'Large',
       'Small', 'Medium'], dtype=object)
```

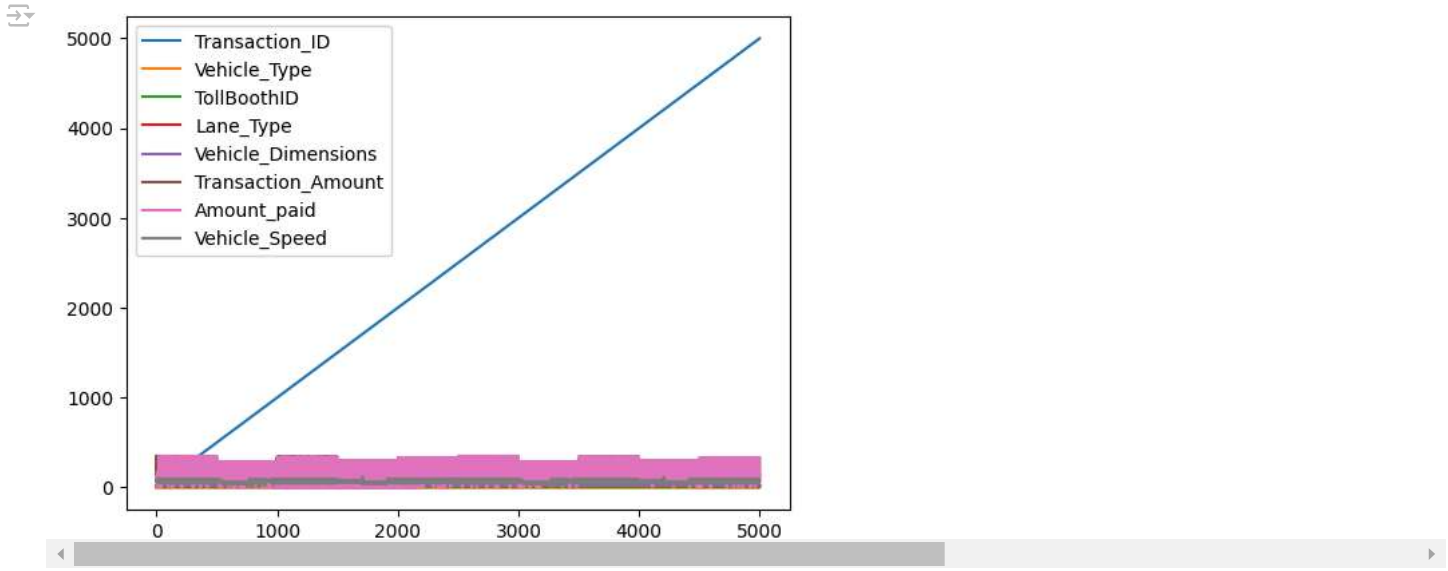
```
veh_type={'Bus ':1, 'Car':2, 'Truck':3, 'Van':4, 'Sedan':5, 'SUV':6, 'Motorcycle':7}
toll_id={'A-101':11, 'B-102':12, 'C-103':13, 'D-106':14}
lane={'Express':21, 'Regular':22}
veh_dim={'Large':31,'Small':32, 'Medium':33}
```

```
data['Vehicle_Type'].replace(veh_type,inplace=True)
data['TollBoothID'].replace(toll_id,inplace=True)
data['Lane_Type'].replace(lane,inplace=True)
data['Vehicle_Dimensions'].replace(veh_dim,inplace=True)
```

DATA VISUALIZATION

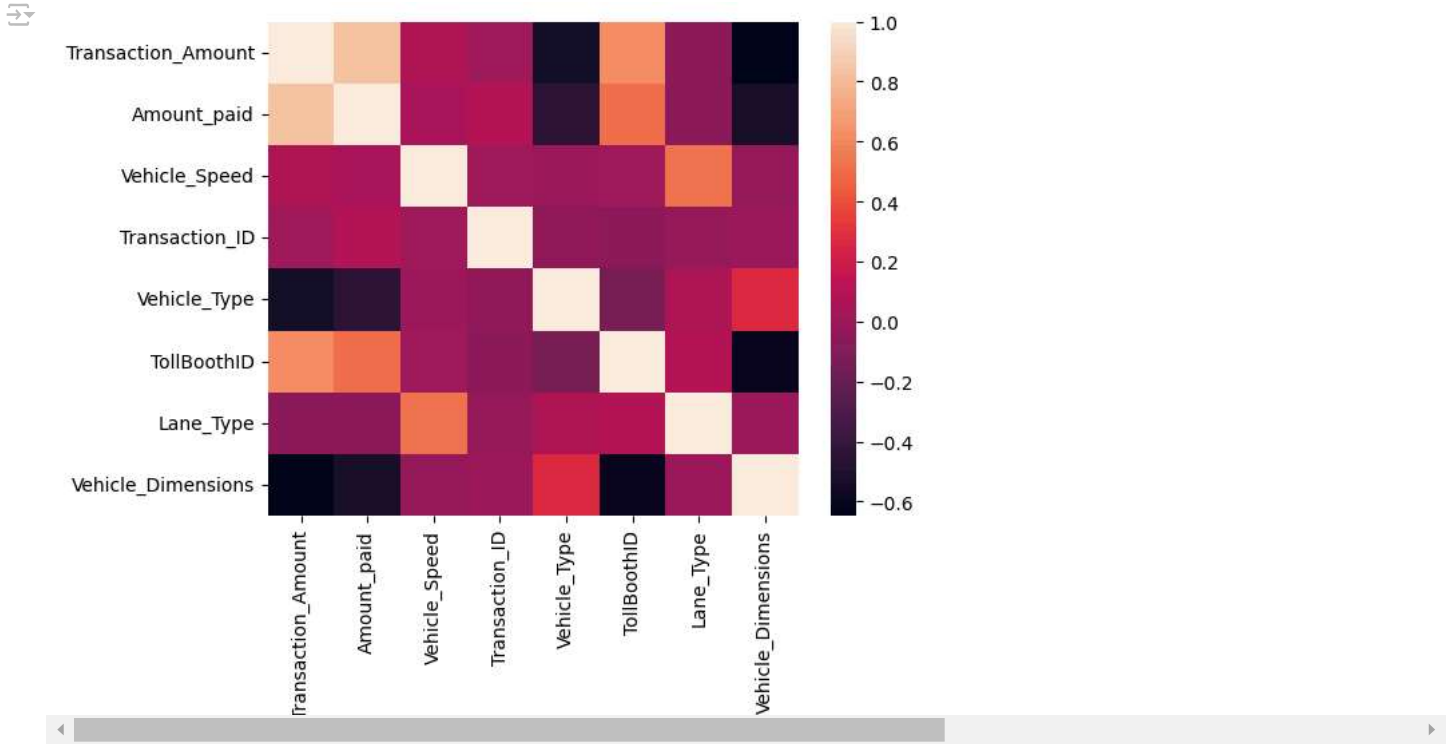
```
import matplotlib.pyplot as plt
```

```
data.plot()
plt.show()
```

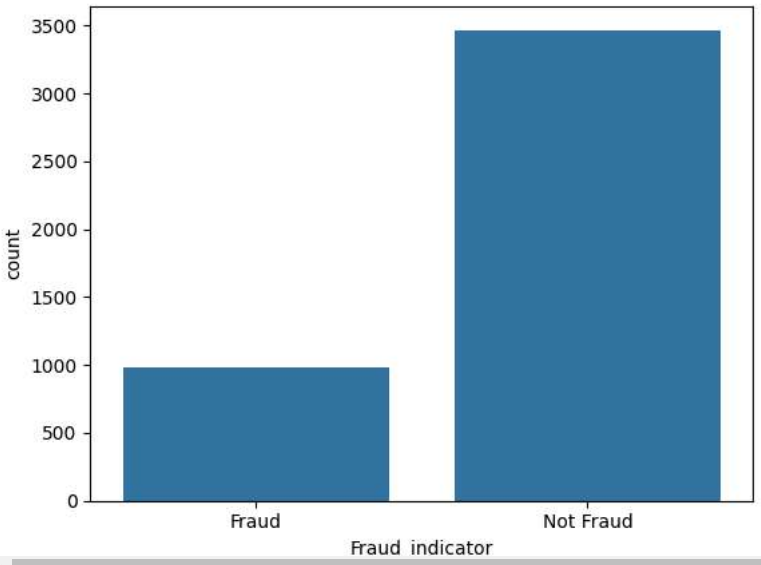



```
import seaborn as sb
```

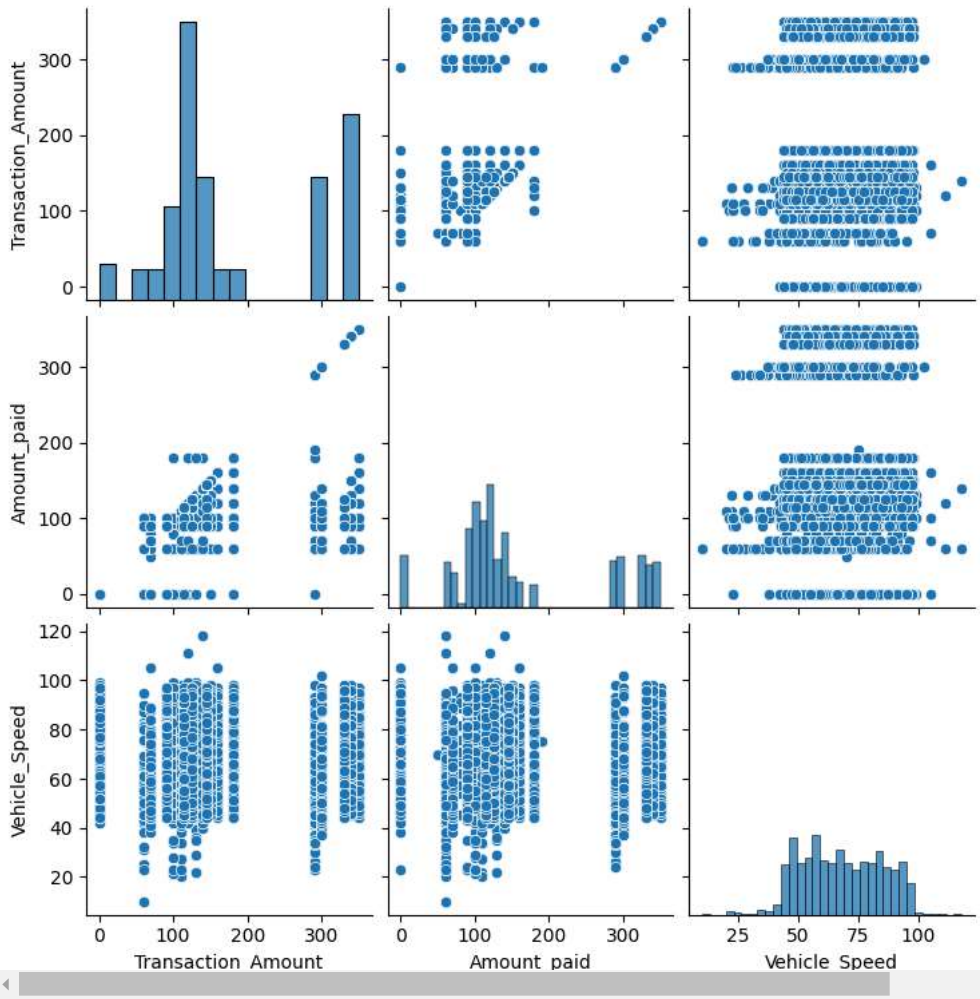
```
correlation_matrix = data[['Transaction_Amount', 'Amount_paid', 'Vehicle_Speed','Transaction_ID','Vehicle_Type','TollBoothID','Lane_Type','V
sb.heatmap(correlation_matrix)
plt.show()
```




```
sb.countplot(x='Fraud_indicator',data=data)
plt.show()
```




```
sb.pairplot(data, vars=['Transaction_Amount', 'Amount_paid', 'Vehicle_Speed'])
plt.show()
```



SEPERATE X AND Y

```
x=data.drop(['FastagID','Geographical_Location','Vehicle_Plate_Number','Fraud_indicator','Timestamp'],axis=1)
y=data.Fraud_indicator
x.shape
```

```
(4451, 8)
```

DATASET SPLITTING

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

MODEL SELECTION

```
from sklearn.tree import DecisionTreeClassifier as dt

model1= dt()
```

TRAINING

```
model1.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

PREDICTION

```
pred1=model1.predict(x_test)
pred1
```

```
array(['Not Fraud', 'Not Fraud', 'Not Fraud', ..., 'Fraud', 'Not Fraud',
       'Not Fraud'], dtype=object)
```

MODEL EVALUATION

```
from sklearn.metrics import (accuracy_score,classification_report,confusion_matrix)

print('decision tree  accuracy is',accuracy_score(y_test,pred1))
```

```
decision tree  accuracy is 0.999251497005988
```

```
print('classification report\n',classification_report(y_test,pred1))
```

```
classification report
                  precision    recall  f1-score   support

           Fraud       1.00      1.00      1.00       275
       Not Fraud       1.00      1.00      1.00      1061

        accuracy                           1.00      1336
       macro avg       1.00      1.00      1.00      1336
    weighted avg       1.00      1.00      1.00      1336
```

```
confusion_matrix(y_test,pred1)
```

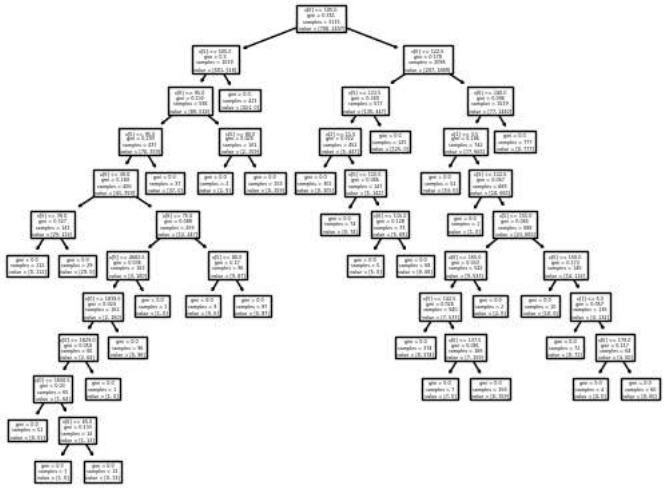
```
array([[ 274,    1],
       [   0, 1061]])
```

DECISION TREE VISUALIZATION

```
from sklearn.tree import plot_tree

plot_tree(model1)
```

```
Text(0.32075471698113206, 0.875, 'x[5] <= 105.0\ngini = 0.5\nsamples = 1019\nvalue = [501, 518]'),
Text(0.2830188679245283, 0.7916666666666666, 'x[6] <= 95.0\ngini = 0.232\nsamples = 598\nvalue = [80, 518]'),
Text(0.20754716981132076, 0.7083333333333334, 'x[5] <= 95.0\ngini = 0.293\nsamples = 437\nvalue = [78, 359]'),
Text(0.16981132075471697, 0.625, 'x[6] <= 30.0\ngini = 0.184\nsamples = 400\nvalue = [41, 359]'),
Text(0.07547169811320754, 0.5416666666666666, 'x[5] <= 30.0\ngini = 0.327\nsamples = 141\nvalue = [29, 112]'),
Text(0.03773584905660377, 0.4583333333333333, 'gini = 0.0\nsamples = 112\nvalue = [0, 112]'),
Text(0.11320754716981132, 0.4583333333333333, 'gini = 0.0\nsamples = 29\nvalue = [29, 0]'),
Text(0.2641509433962264, 0.5416666666666666, 'x[6] <= 75.0\ngini = 0.088\nsamples = 259\nvalue = [12, 247]'),
Text(0.18867924528301888, 0.4583333333333333, 'x[0] <= 4682.5\ngini = 0.036\nsamples = 163\nvalue = [3, 160]'),
Text(0.1509433962264151, 0.375, 'x[0] <= 1839.5\ngini = 0.024\nsamples = 162\nvalue = [2, 160]'),
Text(0.11320754716981132, 0.2916666666666667, 'x[0] <= 1829.0\ngini = 0.059\nsamples = 66\nvalue = [2, 64]'),
Text(0.07547169811320754, 0.20833333333333334, 'x[0] <= 1650.5\ngini = 0.03\nsamples = 65\nvalue = [1, 64]'),
Text(0.03773584905660377, 0.125, 'gini = 0.0\nsamples = 51\nvalue = [0, 51]'),
Text(0.11320754716981132, 0.125, 'x[6] <= 65.0\ngini = 0.133\nsamples = 14\nvalue = [1, 13]'),
Text(0.07547169811320754, 0.041666666666666664, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
Text(0.1509433962264151, 0.041666666666666664, 'gini = 0.0\nsamples = 13\nvalue = [0, 13]'),
Text(0.1509433962264151, 0.20833333333333334, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
Text(0.18867924528301888, 0.2916666666666667, 'gini = 0.0\nsamples = 96\nvalue = [0, 96]'),
Text(0.22641509433962265, 0.375, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
Text(0.33962264150943394, 0.4583333333333333, 'x[5] <= 80.0\ngini = 0.17\nsamples = 96\nvalue = [9, 87]'),
Text(0.3018867924528302, 0.375, 'gini = 0.0\nsamples = 9\nvalue = [9, 0]'),
Text(0.37735849056603776, 0.375, 'gini = 0.0\nsamples = 87\nvalue = [0, 87]'),
Text(0.24528301886792453, 0.625, 'gini = 0.0\nsamples = 37\nvalue = [37, 0]'),
Text(0.3584905660377358, 0.7083333333333334, 'x[5] <= 80.0\ngini = 0.025\nsamples = 161\nvalue = [2, 159]'),
Text(0.32075471698113206, 0.625, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
Text(0.39622641509433965, 0.625, 'gini = 0.0\nsamples = 159\nvalue = [0, 159]'),
Text(0.3584905660377358, 0.7916666666666666, 'gini = 0.0\nsamples = 421\nvalue = [421, 0]'),
Text(0.6415094339622641, 0.875, 'x[6] <= 122.5\ngini = 0.178\nsamples = 2096\nvalue = [207, 1889]'),
Text(0.5471698113207547, 0.7916666666666666, 'x[5] <= 122.5\ngini = 0.349\nsamples = 577\nvalue = [130, 447]'),
Text(0.5094339622641509, 0.7083333333333334, 'x[2] <= 11.5\ngini = 0.022\nsamples = 452\nvalue = [5, 447]'),
Text(0.4716981132075472, 0.625, 'gini = 0.0\nsamples = 305\nvalue = [0, 305]'),
Text(0.5471698113207547, 0.625, 'x[5] <= 115.0\ngini = 0.066\nsamples = 147\nvalue = [5, 142]'),
Text(0.5094339622641509, 0.5416666666666666, 'gini = 0.0\nsamples = 74\nvalue = [0, 74]'),
Text(0.5849056603773585, 0.5416666666666666, 'x[6] <= 115.0\ngini = 0.128\nsamples = 73\nvalue = [5, 68]'),
Text(0.5471698113207547, 0.4583333333333333, 'gini = 0.0\nsamples = 5\nvalue = [5, 0]'),
Text(0.6226415094339622, 0.4583333333333333, 'gini = 0.0\nsamples = 68\nvalue = [0, 68]'),
Text(0.5849056603773585, 0.7083333333333334, 'gini = 0.0\nsamples = 125\nvalue = [125, 0]'),
Text(0.7358490566037735, 0.7916666666666666, 'x[6] <= 240.0\ngini = 0.096\nsamples = 1519\nvalue = [77, 1442]'),
Text(0.6981132075471698, 0.7083333333333334, 'x[1] <= 3.5\ngini = 0.186\nsamples = 742\nvalue = [77, 665]'),
Text(0.660377358490566, 0.625, 'gini = 0.0\nsamples = 53\nvalue = [53, 0]'),
Text(0.7358490566037735, 0.625, 'x[5] <= 112.5\ngini = 0.067\nsamples = 689\nvalue = [24, 665]'),
Text(0.6981132075471698, 0.5416666666666666, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
Text(0.7735849056603774, 0.5416666666666666, 'x[5] <= 155.0\ngini = 0.065\nsamples = 688\nvalue = [23, 665]'),
Text(0.6981132075471698, 0.4583333333333333, 'x[6] <= 165.0\ngini = 0.033\nsamples = 542\nvalue = [9, 533]'),
Text(0.660377358490566, 0.375, 'x[5] <= 142.5\ngini = 0.026\nsamples = 540\nvalue = [7, 533]'),
Text(0.6226415094339622, 0.2916666666666667, 'gini = 0.0\nsamples = 374\nvalue = [0, 374]'),
Text(0.6981132075471698, 0.2916666666666667, 'x[6] <= 137.5\ngini = 0.081\nsamples = 166\nvalue = [7, 159]'),
Text(0.660377358490566, 0.20833333333333334, 'gini = 0.0\nsamples = 7\nvalue = [7, 0]'),
Text(0.7358490566037735, 0.20833333333333334, 'gini = 0.0\nsamples = 159\nvalue = [0, 159]'),
Text(0.7358490566037735, 0.375, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
Text(0.8490566037735849, 0.4583333333333333, 'x[6] <= 150.0\ngini = 0.173\nsamples = 146\nvalue = [14, 132]'),
Text(0.8113207547169812, 0.375, 'gini = 0.0\nsamples = 10\nvalue = [10, 0]'),
Text(0.8867924528301887, 0.375, 'x[1] <= 5.5\ngini = 0.057\nsamples = 136\nvalue = [4, 132]'),
Text(0.8490566037735849, 0.2916666666666667, 'gini = 0.0\nsamples = 72\nvalue = [0, 72]'),
Text(0.9245283018867925, 0.2916666666666667, 'x[6] <= 170.0\ngini = 0.117\nsamples = 64\nvalue = [4, 60]'),
Text(0.8867924528301887, 0.20833333333333334, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]'),
Text(0.9622641509433962, 0.20833333333333334, 'gini = 0.0\nsamples = 60\nvalue = [0, 60]'),
Text(0.7735849056603774, 0.7083333333333334, 'gini = 0.0\nsamples = 777\nvalue = [0, 777]')]
```



KNN CLASSIFIER

```
from sklearn.neighbors import KNeighborsClassifier as kn
```

```
model2= kn(n_neighbors=5)
```

```
model2.fit(x_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

```
pred2=model2.predict(x_test)
```

```
print('KNN accuracy is',accuracy_score(y_test,pred2))
```

```
KNN accuracy is 0.9011976047904192
```

```
print('classification report\n',classification_report(y_test,pred2))
```

```
classification report
                precision    recall  f1-score   support

         Fraud       0.93      0.56      0.70       275
     Not Fraud       0.90      0.99      0.94      1061

      accuracy                           0.90      1336
     macro avg       0.92      0.77      0.82      1336
  weighted avg       0.90      0.90      0.89      1336
```

```
confusion_matrix(y_test,pred2)
```

```
array([[ 154,  121],
       [  11, 1050]])
```

SVM MODEL

```
from sklearn.svm import SVC
```

```
model3=SVC()
```

```
model3.fit(x_train,y_train)
```

```
SVC
SVC()
```

```
pred3=model3.predict(x_test)
```

```
pred3
```

```
array(['Not Fraud', 'Not Fraud', 'Not Fraud', ..., 'Not Fraud',
       'Not Fraud', 'Not Fraud'], dtype=object)
```

```
print('SVM accuracy is',accuracy_score(y_test,pred3))
```

```
SVM accuracy is 0.8488023952095808
```

```
print('classification report\n',classification_report(y_test,pred3))
```

```
classification report
                precision    recall  f1-score   support

         Fraud       1.00      0.27      0.42       275
     Not Fraud       0.84      1.00      0.91      1061

      accuracy                           0.85      1336
```

```
         macro avg       0.92      0.63      0.67      1336
      weighted avg       0.87      0.85      0.81      1336
```

```
confusion_matrix(y_test,pred3)
```

```
array([[  73,  202],
       [   0, 1061]])
```

ADABOOST

```
from sklearn.ensemble import AdaBoostClassifier as ab
```

```
model4=ab(n_estimators=5)
```

```
model4.fit(x_train,y_train)
```

```
AdaBoostClassifier
AdaBoostClassifier(n_estimators=5)
```

```
pred4=model4.predict(x_test)
pred4
```

```
array(['Not Fraud', 'Not Fraud', 'Not Fraud', ..., 'Fraud', 'Not Fraud',
       'Not Fraud'], dtype=object)
```

```
print('AdaBoost accuracy is',accuracy_score(y_test,pred4))
```

```
AdaBoost accuracy is 0.9461077844311377
```

```
print('classification report\n',classification_report(y_test,pred4))
```

```
classification report
```