```
import numpy as np
In [6]:
        import pandas as pd
        import sklearn
        import scipy
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.metrics import classification_report,accuracy_score
        from sklearn.ensemble import IsolationForest # IFA algorith (IsolationForest)
        from sklearn.neighbors import LocalOutlierFactor #LOF algorith (LocalOutlierFactor)
        from sklearn.svm import OneClassSVM # support-vector machines (SVM)
        # SVM:-are supervised learning models with associated learning algorithms that anal
        # analysis.
        import matplotlib.pyplot as plt
        %matplotlib inline
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import accuracy_score
        from sklearn.svm import SVC
        from sklearn.svm import SVR
        from sklearn.naive_bayes import GaussianNB
        from sklearn.tree import DecisionTreeClassifier
        # from sklearn.model_selection import train_test_split (this line has to import for
        import math
        from pylab import rcParams
        rcParams['figure.figsize'] = 14, 8
        RANDOM\_SEED = 42
        LABELS = ["Normal", "Fraud"]
        from sklearn.preprocessing import StandardScaler
In [7]: dataset = pd.read_csv(r"C:\Users\Deepak kumar sharma\OneDrive\Desktop\Jupyter Notel
        # above line dataset = pd.read_csv(r"path of the .csv file") then it will run pro
        dataset.head() # 1st 5 row of dataset
In [8]:
                                                           V5
           Time
                      V1
                               V2
                                        V3
                                                 V4
                                                                    V6
                                                                             V7
                                                                                      V8
Out[8]:
        0
             0.0 -1.359807 -0.072781 2.536347
                                             1.378155 -0.338321
                                                               0.462388
                                                                        0.239599
                                                                                  0.098698
                                                                                           0.
             0.0
                 1.191857
                           0.266151 0.166480
                                             0.448154
                                                      0.060018
                                                               -0.082361
                                                                        -0.078803
                                                                                  0.085102
                                                                                          -0.
        2
             1.0 -1.358354 -1.340163 1.773209
                                            0.379780 -0.503198
                                                               1.800499
                                                                         0.791461
                                                                                  0.247676
                                                                                          -1.
             1.0 -0.966272 -0.185226 1.792993
                                            -0.863291
                                                    -0.010309
                                                               1.247203
                                                                         0.237609
                                                                                  0.377436
                                                                                          -1.
             0.095921
                                                                        0.592941
                                                                                 -0.270533
                                                                                           0.
       5 rows × 31 columns
```

In [9]: dataset.info()

```
RangeIndex: 284807 entries, 0 to 284806
         Data columns (total 31 columns):
              Column Non-Null Count
                                      Dtype
         ---
             _____
                     -----
                                      ----
          0
              Time
                      284807 non-null float64
          1
              V1
                      284807 non-null float64
          2
              V2
                      284807 non-null float64
          3
             V3
                      284807 non-null float64
                      284807 non-null float64
          4
             V4
          5
              V5
                      284807 non-null float64
                      284807 non-null float64
              V6
          6
          7
              V7
                      284807 non-null float64
          8
              ٧8
                     284807 non-null float64
          9
              V/9
                     284807 non-null float64
          10 V10
                      284807 non-null float64
                      284807 non-null float64
             V11
          11
                     284807 non-null float64
          12 V12
          13 V13
                     284807 non-null float64
          14 V14
                     284807 non-null float64
          15 V15
                     284807 non-null float64
                     284807 non-null float64
          16 V16
                     284807 non-null float64
          17 V17
                     284807 non-null float64
          18 V18
          19 V19
                     284807 non-null float64
                     284807 non-null float64
          20 V20
                     284807 non-null float64
          21 V21
                      284807 non-null float64
          22 V22
                      284807 non-null float64
          23
             V23
          24 V24
                     284807 non-null float64
          25 V25
                     284807 non-null float64
          26 V26
                     284807 non-null float64
          27 V27
                     284807 non-null float64
          28 V28
                      284807 non-null float64
          29 Amount 284807 non-null float64
          30 Class
                     284807 non-null int64
         dtypes: float64(30), int64(1)
         memory usage: 67.4 MB
         x = dataset.iloc[: , 1:30].values
In [10]:
         y = dataset.iloc[:, 30].values
         print("Input Range : ", x.shape)
In [11]:
         print("Output Range : ", y.shape)
         Input Range: (284807, 29)
         Output Range: (284807,)
         print ("Class Labels : \n", y)
In [12]:
         Class Labels :
          [0 0 0 ... 0 0 0]
         dataset.isnull().values.any() # Good No Null Values!
In [13]:
         False
Out[13]:
         set_class = pd.value_counts(dataset['Class'], sort = True) # pd.value_counts() full
In [14]:
         # coloumn which is used //here outputs how many fraud and non fraud occur , sort()
         set_class.plot(kind = 'bar', rot=0) #1. Simple Bar Plot function
          # link:-https://dataindependent.com/pandas/pandas-bar-plot-dataframe-plot-bar/
         plt.title("Class Distribution of Transaction")
```

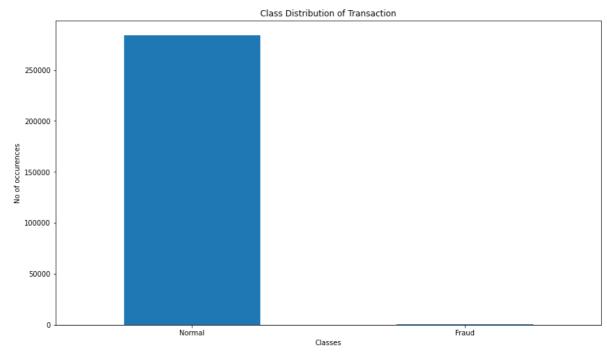
<class 'pandas.core.frame.DataFrame'>

```
plt.xticks(range(2), LABELS) # plt.xticks(ticks=none ,labels=none, **kways)//plt.;
plt.xlabel("Classes")

plt.ylabel("No of occurences")

#1. Simple Bar Plot
#In order to create a bar plot, you need to pass a X and Y values. X will be your of #bars. Y will be the value of your bars, or how high they are.
#Note: Rot = Rotation. When I specify rot=0, I'm telling pandas not to rotate my X
```

Out[14]: Text(0, 0.5, 'No of occurences')



```
In [15]: fraud_data = dataset[dataset['Class']==1]
          normal_data = dataset[dataset['Class']==0]
          # print(fraud_data.shape,normal_data.shape)
 In [ ]:
         fraud_data.Amount.describe()
In [16]:
                    492.000000
         count
Out[16]:
                    122.211321
         mean
         std
                    256.683288
         min
                      0.000000
         25%
                      1.000000
         50%
                      9.250000
         75%
                    105.890000
         max
                   2125.870000
         Name: Amount, dtype: float64
         normal_data.Amount.describe()
In [17]:
```

```
284315.000000
                                                              count
Out[17]:
                                                             mean
                                                                                                                                                 88.291022
                                                              std
                                                                                                                                           250.105092
                                                                                                                                                        0.000000
                                                              min
                                                               25%
                                                                                                                                                         5.650000
                                                               50%
                                                                                                                                                  22.000000
                                                              75%
                                                                                                                                                  77.050000
                                                                                                                               25691.160000
                                                              max
                                                             Name: Amount, dtype: float64
In [18]:
                                                               ## Correlation
                                                               import seaborn as sns
                                                               #get correlations of each features in dataset
                                                               corrmat = dataset.corr()
                                                               top_corr_features = corrmat.index
                                                               plt.figure(figsize=(20,20))
                                                               #plot heat map
                                                               g=sns.heatmap(dataset[top_corr_features].corr(),annot=True,cmap="RdYlGn")
                                                                          1 0.12 -0.012 -0.42 -0.11 0.17 -0.0630 0850 0370 00870 031 -0.25 0.12 -0.0660 0.099 -0.18 0.012-0.073 0.09 0.029-0.0510.045 0.14 0.051-0.016 -0.23 -0.045 0.045 0.0510 0.099 0.0110 0.012
                                                                                                    .1e-15.2e-15.2e-168e-167.5e-16e-152.4e-155e-1564e-177.1e-125.1e-125.4e-1-5e-16.5e-152e-157.9e-162e-187.5e-167.e-18.5e-163.e-163.e-163.e-162e-187.6e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-163.e-
                                                                                                                                     7e-165e-176e-169e-163e-166e-162e-166e-163e-128e-167e-181e-163e-156e-165e-165e-153e-167e-17.1e-15e-10.7e-19.1e-152e-16e-19.8e-1<sup>e</sup>0.21 0.1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     0.8
                                                            ≸ --0.119.2e-1161e-1157e-1
                                                                                                                                                  .7e-1755e-1761e-1767e-1769e-1765e-1765e-1766e-1766e-1763e-1764e-1766e-1767e-1761e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e-1769e
                                                                                                                                                                 4e-1157e-1154e-1154e-1152e-1152e-1154e-1159e-1156e-1157e-1152e-1153e-1153e-1155e-1359e-1169e-1184e-1181e-1158e-1163e-1163e-1159e-1164e-1159e-1164e-1159e-1164e-1159e-1164e-1159e-1164e-1159e-1164e-1159e-1164e-1159e-1164e-1159e-1164e-1159e-1164e-1159e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-1164e-116
                                                                                                                                                                              .2e-115.1e-1151e-1159e-172e-152.4e-1152e-1266e-115.5e-1256e-152e-162.2e-115.9e-1169e-168e-147.7e-19e-161.1e-1256e-115.4e-1455e-1266e-160.22 -0.044
                                                                                                                                                                                                 8e-1161e-115.5e-1174e-116.5e-1189e-127.6e-115.7e-169e-127.2e-115.6e-117.9e-1964e-142.e-146.9e-1464e-1764e-118.1e-1267e-1168e-1258e-160.4
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     - 0.6
                                                            e-126.8e-1265e-1268e-126.9e-126.6e-1461e-126.3e-1263e-1267e-1269e-142e-146.9e-142e-146.4e-147.e-1267e-1267e-1267e-126.2e-16-0.1 0.02
                                                            @ -0.0081/.5e-1@e-175.6e-1659e-1754e-1451e-1161e-1454e-1
                                                                                                                                                                                                                            6e-1164e-145.1e-1253e-1258e-125.1e-1252e-1251e-125e-146.4e-1263e-1169e-147.1e-1262e-1264e-1268e-145.9e-1267e-1171e-150.0440.098
                                                                      -0.0317.4e-174e-161.2e-1252e-145.2e-1669e-177.5e-1278e-1466e-1
                                                                          0.25<mark>2</mark>.1e-162e-161.6e-1355e-1352e-162e-151.4e-1255e-1364e-146.6e-1
                                                                                                                                                                                                                                                           4e-1 Qe-1 Q. 7e-1 153e-1 158e-1 152e-1 164e-1 15.4e-1 165e-1 167e-1 168e-1 169e-1 15.6e-1 Q. 6e-1 16.e-1 16.0001 0.15
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     - 0.4
                                                            2 - 0.122.1e-15.6e-1673e-15.6e-164e-15.4e-15.5e-188e-15.1e-158e-16.4e-1
                                                                                                                                                                                                                                                                         3e-1248e-127.4e-1253e-1256e-1 Ge-1 71.5e-1254e-1253e-1266e-1258e-1464e-125.7e-1254e-1267e-1264e-146.009$0.2
                                                            5 -0.0995e-14.7e-167e-1253e-166e-1266e-1266e-1266e-1256e-1256e-1257e-1258e-1272e-1
                                                                            ). 183. 5e-165e-179. 1e-1154e-186. 7e-1165e-1157e-1861e-116. 1e-1558e-1163e-15. 4e-1154e-116. 8e-1
                                                                                                                                                                                                                                                                                                                    3e-166e-185e-165e-185e-166e-147.2e-1869e-1465e-1862e-1268e-1161e-1152e-150.0030.004
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   - 0.2
                                                            9 -0.017/2e-10/2e-16/3e-16/6e-16/2e-15/6e-16/9e-15/3e-16/2e-16/5e-16/8e-16/3e-16/4e-15/3e-15
                                                                                                                                                                                                                                                                                                                                 .5e-1254e-1253e-1252e-146.7e-1769e-128e-16-3e-146.3e-1253e-1668e-1256e-146.0039-0
                                                            5 -0.073.9e-1267e-1267e-1267e-1267e-1263e-126e-1262e-1263e-1261e-1255e-1352e-1366e-1256e-1256e-1255e-1351e-1266e-1266e-1265e-1366e-1265e-1366e-1266e-1266e-1265e-1366e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1266e-1
                                                                                                                                                                                                                                                                                                                                               .9e-1259e-1264e-1252e-1267e-1267e-1264e-1267e-1269e-1261e-1265e-107.0073<mark>-0.3</mark>
                                                            8 -0.05#.7e-185e-185e-186e-186e-186e-186e-180e-10e-10e-10e-185e-185e-186e-186e-182e-183e-185e-186e-186e-1
                                                            5 -0.0452.5e-165e-167e-17.9e-169e-168e-172e-16.9e-169e-162e-15.7e-163e-14e-166.4e-166e-14.7e-162e-164e-161.6e-1 1
                                                            👸 - 0.144.3e-186e-18.1e-1853e-1873e-14.7e-1899e-18e-167.1e-1864e-186e-186e-186e-187e-147e-1462e-187e-1488e-1862e-15e-15e-15 6e-1 🔞 1 🔭 3e-17e-16-5e-14.5e-187.6e-140.5e-187.6e-140.068.0008
                                                            -0.2
                                                            $\frac{7}{6}$ -0.01$\frac{1}{6}$.4e-1172e-1277e-1266e-145.1e-1251e-1254e-146e-145.4e-1264e-1260e-1454e-126e-146.5e-166e-142.4e-146e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e-1276e
                                                                        -0.239.6e-1465e-1161e-1461e-1468e-1466e-131.1e-1467e-1468e-1467e-1466e-151.6e-1466e-151.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-146.6e-14
                                                            👸 -0.041. 6e-17/1e-1152e-1452e-1452e-1452e-1452e-157e-157e-157e-159e-16e-161e-12.4e-158e-157e-158e-15.3e-159e-16e-16.9e-15.8e-15.8e-156e-155e-17/1e-153e-155e-156e-15
                                                            😤 -0.009241e-15.1e-1668e-125.8e-1666e-125.8e-1666e-125.8e-161e-145.9e-157.8e-161e-125.9e-151.8e-125.e-155.e-156e-125.4e-125.4e-125.4e-1666e-125.8e-166.6e-115.8e-161e-125.8e-161e-1
                                                                   -0.011-0.23 0.53 0.21 0.099 0.39 0.22 0.4 0.1 0.044 0.1 0.0000.0098 00530 0340 0030 00730 0360 0.056 0.34 0.11 0.065 0.110.00510 0480 00320 029 0.01 1 0.0056
                                                             8 -0.012 -0.1 0.091 0.19 0.13 -0.0950.044 0.19 0.02 -0.098 0.22 0.15 0.260.0046 0.3 0.0042 0.2 0.33 -0.11 0.035 0.02 0.040.00081.0028.0078.0038 0.0450.0180.0098 0.056
```

```
NameError
                                                  Traceback (most recent call last)
         Input In [27], in <cell line: 1>()
         ----> 1 xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, rand
         om_state = 0)
         NameError: name 'train_test_split' is not defined
In [28]: xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, random_state
         NameError
                                                  Traceback (most recent call last)
         Input In [28], in <cell line: 1>()
         ----> 1 xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, rand
         om_state = 0)
         NameError: name 'train_test_split' is not defined
In [19]: from sklearn.model_selection import train_test split
In [20]: xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, random_state
In [21]: print("xtrain.shape : ", xtrain.shape)
         print("xtest.shape : ", xtest.shape)
         print("ytrain.shape : ", ytrain.shape)
         print("ytest.shape : ", ytest.shape)
         xtrain.shape : (213605, 29)
         xtest.shape : (71202, 29)
         ytrain.shape : (213605,)
         ytest.shape : (71202,)
In [22]: #StandardScaler follows Standard Normal Distribution (SND). Therefore, it makes med
         stdsc = StandardScaler()
         xtrain = stdsc.fit_transform(xtrain)
         xtest = stdsc.transform(xtest)
         #The fit(data) method is used to compute the mean and std dev for a given feature s
         #The transform(data) method is used to perform scaling using mean and std dev calcu
         #The fit_transform() method does both fit and transform.
In [23]: | print("Training Set after Standardised : \n", xtrain[0])
         Training Set after Standardised:
          0.58133086 -0.40257892 -0.09319222 0.16481198 1.60036637 1.18028602
          -0.24273404 \quad 1.08764203 \quad -0.35935009 \quad -0.76863613 \quad -0.28881862 \quad -0.39536117
           0.13774039 -0.34055771 0.32484688 1.13026957 0.03716189 0.90724443
           0.61754959 0.39904973 -0.21031503 -0.2607924 -0.35356699]
In [35]: | dt_classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
         dt classifier.fit(xtrain, ytrain)
Out[35]:
                             DecisionTreeClassifier
         DecisionTreeClassifier(criterion='entropy', random_state=0)
In [36]: y_pred_decision_tree = dt_classifier.predict(xtest)
In [37]: print("y_pred_decision_tree : \n", y_pred_decision_tree)
         y_pred_decision_tree :
          [0 0 0 ... 0 0 0]
```

```
In [38]:
         com decision = confusion matrix(ytest, y pred decision tree)
         print("confusion Matrix : \n", com_decision)
         confusion Matrix :
          [[71052
                     30]
              25
                    95]]
         Accuracy_Model = ((com_decision[0][0] + com_decision[1][1]) / com_decision.sum())
In [39]:
         print("Accuracy_Decison : ", Accuracy_Model)
         Error_rate_Model= ((com_decision[0][1] + com_decision[1][0]) / com_decision.sum())
         print("Error_rate_Decison : ", Error_rate_Model)
         # True Fake Rate
         Specificity\_Model = (com\_decision[1][1] / (com\_decision[1][1] + com\_decision[0][1]))
         print("Specificity_Decison : ", Specificity_Model)
         # True Genuine Rate
         Sensitivity_Model = (com_decision[0][0] / (com_decision[0][0] + com_decision[1][0])
         print("Sensitivity_Decison : ", Sensitivity_Model)
         Accuracy_Decison : 99.92275497879272
         Error_rate_Decison : 0.07724502120726946
         Specificity_Decison: 76.0
         Sensitivity_Decison: 99.96482687789299
         svc_classifier = SVC(kernel = 'rbf', random_state =0)
In [24]:
         svc_classifier.fit(xtrain, ytrain)
Out[24]:
                  SVC
         SVC(random_state=0)
In [25]: y_pred2 = svc_classifier.predict(xtest)
In [26]: print("y_pred_randomforest : \n", y_pred2)
         y_pred_randomforest :
          [0 0 0 ... 0 0 0]
In [27]: cm2 = confusion_matrix(ytest, y_pred2)
         print("Confusion Matrix : \n\n", cm2)
         Confusion Matrix:
          [[71077
                      5]
              44
                    76]]
In [28]: # Validating the Prediction
         Accuracy_Model = ((cm2[0][0] + cm2[1][1]) / cm2.sum()) *100
         print("Accuracy_svc : ", Accuracy_Model)
         Error_rate_Model = ((cm2[0][1] + cm2[1][0]) / cm2.sum()) *100
         print("Error_rate_svc : ", Error_rate_Model)
         # True Fake Rate
         Specificity_Model= (cm2[1][1] / (cm2[1][1] + cm2[0][1])) *100
         print("Specificity_svc : ", Specificity_Model)
         # True Genuine Rate
         Sensitivity_Model= (cm2[0][0] / (cm2[0][0] + cm2[1][0])) *100
         print("Sensitivity_svc : ", Sensitivity_Model)
```

Accuracy_svc : 99.93118170837899 Error_rate_svc : 0.06881829162102188 Specificity_svc : 93.82716049382715 Sensitivity_svc : 99.93813360329578

```
In [32]:
         NameError
                                                   Traceback (most recent call last)
         Input In [32], in <cell line: 2>()
               1 # training the logistic model on training data data
         ---> 2 model=LogisticRegression()
         NameError: name 'LogisticRegression' is not defined
In [34]: from sklearn.linear_model import LogisticRegression
In [35]:
         # training the logistic model on training data data
         model=LogisticRegression()
In [36]: model.fit(xtrain, ytrain)
Out[36]: ▼ LogisticRegression
         LogisticRegression()
In [37]: # accuracy score on training data
         x_train_prediction=model.predict(xtrain)
         training_data_accuracy=accuracy_score(x_train_prediction,ytrain)
In [38]: print("accurcy on training data: ",training_data_accuracy)
         accurcy on training data: 0.99917136771143
In [40]: # accuracy score on test data
         x_test_prediction=model.predict(xtest)
         test_data_accuracy=accuracy_score(x_test_prediction,ytest)
In [41]: print("accurcy on test data: ",test_data_accuracy)
         accurcy on test data: 0.9992977725344794
In [ ]:
```