wjm9w1xir

January 28, 2025

[2]:

```
#importing the necessary packages
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,␣
 ↪GradientBoostingClassifier, AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.metrics.cluster import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:

```
#loading the dataset
data = pd.read_csv(r"C:\Users\91703\Downloads\Online fraud detection.csv")
```

[4]:

```
data.head()
```

[4]:

```
   step      type    amount     nameOrig  oldbalanceOrg  newbalanceOrig  \
0     1   PAYMENT   9839.64  C1231006815       170136.0       160296.36
1     1   PAYMENT   1864.28  C1666544295        21249.0        19384.72
2     1  TRANSFER    181.00  C1305486145          181.0            0.00
3     1  CASH_OUT    181.00   C840083671          181.0            0.00
4     1   PAYMENT  11668.14  C2048537720        41554.0        29885.86

      nameDest  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
0  M1979787155             0.0             0.0        0               0
1  M2044282225             0.0             0.0        0               0
2   C553264065             0.0             0.0        1               0
3    C38997010         21182.0             0.0        1               0
```

```
4  M1230701703             0.0             0.0        0               0
```

[5]:
```
data.tail()
```

[5]:
```
         step      type     amount     nameOrig  oldbalanceOrg  \
1048570    95  CASH_OUT  132557.35  C1179511630      479803.00   
1048571    95   PAYMENT    9917.36  C1956161225       90545.00   
1048572    95   PAYMENT   14140.05  C2037964975       20545.00   
1048573    95   PAYMENT   10020.05  C1633237354       90605.00   
1048574    95   PAYMENT   11450.03  C1264356443       80584.95   

         newbalanceOrig     nameDest  oldbalanceDest  newbalanceDest  isFraud  \
1048570       347245.65   C435674507       484329.37       616886.72        0   
1048571        80627.64   M668364942            0.00            0.00        0   
1048572         6404.95  M1355182933            0.00            0.00        0   
1048573        80584.95  M1964992463            0.00            0.00        0   
1048574        69134.92   M677577406            0.00            0.00        0   

         isFlaggedFraud  
1048570               0  
1048571               0  
1048572               0  
1048573               0  
1048574               0  
```

[6]:
```
data.describe()
```

[6]:
```
               step        amount  oldbalanceOrg  newbalanceOrig  \
count  1.048575e+06  1.048575e+06   1.048575e+06    1.048575e+06   
mean   2.696617e+01  1.586670e+05   8.740095e+05    8.938089e+05   
std    1.562325e+01  2.649409e+05   2.971751e+06    3.008271e+06   
min    1.000000e+00  1.000000e-01   0.000000e+00    0.000000e+00   
25%    1.500000e+01  1.214907e+04   0.000000e+00    0.000000e+00   
50%    2.000000e+01  7.634333e+04   1.600200e+04    0.000000e+00   
75%    3.900000e+01  2.137619e+05   1.366420e+05    1.746000e+05   
max    9.500000e+01  1.000000e+07   3.890000e+07    3.890000e+07   

       oldbalanceDest  newbalanceDest       isFraud  isFlaggedFraud  
count    1.048575e+06    1.048575e+06  1.048575e+06       1048575.0  
mean     9.781600e+05    1.114198e+06  1.089097e-03             0.0  
std      2.296780e+06    2.416593e+06  3.298351e-02             0.0  
min      0.000000e+00    0.000000e+00  0.000000e+00             0.0  
25%      0.000000e+00    0.000000e+00  0.000000e+00             0.0  
50%      1.263772e+05    2.182604e+05  0.000000e+00             0.0  
75%      9.159235e+05    1.149808e+06  0.000000e+00             0.0  
max      4.210000e+07    4.220000e+07  1.000000e+00             0.0  
```

[7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 11 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   step            1048575 non-null  int64
 1   type            1048575 non-null  object
 2   amount          1048575 non-null  float64
 3   nameOrig        1048575 non-null  object
 4   oldbalanceOrg   1048575 non-null  float64
 5   newbalanceOrig  1048575 non-null  float64
 6   nameDest        1048575 non-null  object
 7   oldbalanceDest  1048575 non-null  float64
 8   newbalanceDest  1048575 non-null  float64
 9   isFraud         1048575 non-null  int64
 10  isFlaggedFraud  1048575 non-null  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 88.0+ MB
```

[8]:
```
data.isnull().sum()
```

[8]:
```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

[9]:
```
data.dtypes
```

[9]:
```
step                int64
type               object
amount            float64
nameOrig           object
oldbalanceOrg     float64
newbalanceOrig    float64
nameDest           object
oldbalanceDest    float64
newbalanceDest    float64
```

```
isFraud             int64
isFlaggedFraud      int64
dtype: object
```

[10]:
```
#Normalizing numerical columns
numeric_cols = ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',␣
 ↪'newbalanceDest']
scaler = StandardScaler()
data[numeric_cols] = scaler.fit_transform(data[numeric_cols])
```

[11]:
```
data.head()
```

[11]:
```
   step      type    amount     nameOrig  oldbalanceOrg  newbalanceOrig  \
0     1   PAYMENT -0.561738  C1231006815      -0.236855       -0.243832
1     1   PAYMENT -0.591840  C1666544295      -0.286956       -0.290673
2     1  TRANSFER -0.598194  C1305486145      -0.294045       -0.297117
3     1  CASH_OUT -0.598194   C840083671      -0.294045       -0.297117
4     1   PAYMENT -0.554837  C2048537720      -0.280123       -0.287183

      nameDest  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
0  M1979787155       -0.425883       -0.461062        0               0
1  M2044282225       -0.425883       -0.461062        0               0
2   C553264065       -0.425883       -0.461062        1               0
3    C38997010       -0.416661       -0.461062        1               0
4  M1230701703       -0.425883       -0.461062        0               0
```

[12]:
```
#doing the Label Encoder to transform the below columns
le = LabelEncoder()
data['type'] = le.fit_transform(data['type'])
data['nameOrig'] = le.fit_transform(data['nameOrig'])
data['nameDest'] = le.fit_transform(data['nameDest'])
```

[13]:
```
#Splitting data into training and testing sets:
X = data.drop('isFraud', axis=1) # features
y = data['isFraud'] # target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[14]:
```
#  LOGISTIC REGRESSION #
logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train, y_train)
```

[14]:
```
LogisticRegression(max_iter=1000)
```

[15]:
```
y_pred_logreg = logreg.predict(X_test)
print(y_pred_logreg)
```

```
[0 0 0 … 0 0 0]
```

[16]:

```
print("Logistic Regression Model Performance:")
print("Accuracy:", accuracy_score(y_test, y_pred_logreg))
print("Classification Report:\n", classification_report(y_test, y_pred_logreg))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_logreg))
```

```
Logistic Regression Model Performance:
Accuracy: 0.9990367880218392
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00    209491
           1       0.96      0.10      0.19       224

    accuracy                           1.00    209715
   macro avg       0.98      0.55      0.59    209715
weighted avg       1.00      1.00      1.00    209715

Confusion Matrix:
 [[209490      1]
 [   201     23]]
```

[17]:

```
# DESICION TREE CLASSIFIER #
DT = DecisionTreeClassifier(random_state=42)
DT.fit(X_train, y_train)
```

[17]:

```
DecisionTreeClassifier(random_state=42)
```

[18]:

```
#Model evaluation
y_pred_DT = DT.predict(X_test)
print(y_pred_DT)
```

```
[0 0 0 … 0 0 0]
```

[19]:

```
print("Decision Tree Model Performance:")
print("Accuracy:", accuracy_score(y_test, y_pred_DT))
print("Classification Report:", classification_report(y_test, y_pred_DT))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_DT))
```

```
Decision Tree Model Performance:
Accuracy: 0.9995994564051213
Classification Report:                precision    recall  f1-score   support

           0       1.00      1.00      1.00    209491
           1       0.80      0.84      0.82       224

    accuracy                           1.00    209715
```

```
   macro avg       0.90      0.92      0.91    209715
weighted avg       1.00      1.00      1.00    209715

Confusion Matrix:
 [[209443     48]
 [    36    188]]
```

[20]:

```
# RANDOM FOREST CLASSIFIER #
RF = RandomForestClassifier(n_estimators=100, random_state=42)
RF.fit(X_train, y_train)
```

[20]:

```
RandomForestClassifier(random_state=42)
```

[21]:

```
#Model evaluation
y_pred_RF = RF.predict(X_test)
print(y_pred_RF)
```

```
[0 0 0 … 0 0 0]
```

[22]:

```
print("Random Forest Model Performance:")
print("Accuracy:", accuracy_score(y_test, y_pred_RF))
print("Classification Report:", classification_report(y_test, y_pred_RF))
print("Confusion Matrix:", confusion_matrix(y_test, y_pred_RF))
```

```
Random Forest Model Performance:
Accuracy: 0.9997711179457835
Classification Report:               precision    recall  f1-score   support

           0       1.00      1.00      1.00    209491
           1       0.95      0.83      0.89       224

    accuracy                           1.00    209715
   macro avg       0.98      0.91      0.94    209715
weighted avg       1.00      1.00      1.00    209715

Confusion Matrix: [[209482      9]
 [    39    185]]
```