In [2]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```python
df=pd.read_csv('C:\\Users\\ACER\\Downloads\\Fraud.csv')
```

In [4]:
```python
df.head()
```

Out[4]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

In [5]:
```python
df.tail()
```

Out[5]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C776919290	0.00	339682.13	1	0
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C1881841831	0.00	0.00	1	0
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C1365125890	68488.84	6379898.11	1	0
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C2080388513	0.00	0.00	1	0
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C873221189	6510099.11	7360101.63	1	0

In [6]:
```python
df.info
```

Out[6]:
```
<bound method DataFrame.info of          step      type      amount     nameOrig  oldbalanceOrg  \
0           1   PAYMENT     9839.64  C1231006815      170136.00
1           1   PAYMENT     1864.28  C1666544295       21249.00
2           1  TRANSFER      181.00  C1305486145         181.00
3           1  CASH_OUT      181.00   C840083671         181.00
4           1   PAYMENT    11668.14  C2048537720       41554.00
...       ...       ...         ...          ...            ...
6362615   743  CASH_OUT   339682.13   C786484425      339682.13
6362616   743  TRANSFER  6311409.28  C1529008245     6311409.28
6362617   743  CASH_OUT  6311409.28  C1162922333     6311409.28
6362618   743  TRANSFER   850002.52  C1685995037      850002.52
6362619   743  CASH_OUT   850002.52  C1280323807      850002.52

         newbalanceOrig     nameDest  oldbalanceDest  newbalanceDest  isFraud  \
0             160296.36  M1979787155            0.00            0.00        0
1              19384.72  M2044282225            0.00            0.00        0
2                  0.00   C553264065            0.00            0.00        1
3                  0.00    C38997010        21182.00            0.00        1
4              29885.86  M1230701703            0.00            0.00        0
...                 ...          ...             ...             ...      ...
6362615            0.00   C776919290            0.00       339682.13        1
6362616            0.00  C1881841831            0.00            0.00        1
6362617            0.00  C1365125890        68488.84      6379898.11        1
6362618            0.00  C2080388513            0.00            0.00        1
6362619            0.00   C873221189      6510099.11      7360101.63        1

         isFlaggedFraud
0                     0
1                     0
2                     0
3                     0
4                     0
...                 ...
6362615               0
6362616               0
6362617               0
6362618               0
6362619               0

[6362620 rows x 11 columns]>
```

In [7]:
```python
df.describe
```

Out[7]:
```
<bound method NDFrame.describe of          step      type      amount     nameOrig  oldbalanceOrg  \
0           1   PAYMENT     9839.64  C1231006815      170136.00
1           1   PAYMENT     1864.28  C1666544295       21249.00
2           1  TRANSFER      181.00  C1305486145         181.00
3           1  CASH_OUT      181.00   C840083671         181.00
4           1   PAYMENT    11668.14  C2048537720       41554.00
...       ...       ...         ...          ...            ...
6362615   743  CASH_OUT   339682.13   C786484425      339682.13
6362616   743  TRANSFER  6311409.28  C1529008245     6311409.28
6362617   743  CASH_OUT  6311409.28  C1162922333     6311409.28
6362618   743  TRANSFER   850002.52  C1685995037      850002.52
6362619   743  CASH_OUT   850002.52  C1280323807      850002.52

         newbalanceOrig     nameDest  oldbalanceDest  newbalanceDest  isFraud  \
0             160296.36  M1979787155            0.00            0.00        0
1              19384.72  M2044282225            0.00            0.00        0
2                  0.00   C553264065            0.00            0.00        1
3                  0.00    C38997010        21182.00            0.00        1
4              29885.86  M1230701703            0.00            0.00        0
...                 ...          ...             ...             ...      ...
6362615            0.00   C776919290            0.00       339682.13        1
6362616            0.00  C1881841831            0.00            0.00        1
6362617            0.00  C1365125890        68488.84      6379898.11        1
6362618            0.00  C2080388513            0.00            0.00        1
6362619            0.00   C873221189      6510099.11      7360101.63        1

         isFlaggedFraud
0                     0
1                     0
2                     0
3                     0
4                     0
...                 ...
6362615               0
6362616               0
6362617               0
6362618               0
6362619               0

[6362620 rows x 11 columns]>
```

In [8]:
```python
df.isnull().sum()
```

Out[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
```

In [9]:
```python
df['type'].unique()
```

Out[9]:
```
array(['PAYMENT', 'TRANSFER', 'CASH_OUT', 'DEBIT', 'CASH_IN'],
      dtype=object)
```

In [10]:
```python
type=df['type'].value_counts()
```

In [11]:
```python
type.index
```

Out[11]:
```
Index(['CASH_OUT', 'PAYMENT', 'CASH_IN', 'TRANSFER', 'DEBIT'], dtype='object')
```

In [12]:
```python
transaction=type.index
```

In [13]:
```python
quantity=type.values
```

In [14]:
```python
df['isFraud']=df['isFraud'].map({0:'No fraud',1:'fraud'})
```

In [15]:
```python
df
```

Out[15]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	No fraud	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	No fraud	0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	fraud	0
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	fraud	0
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	No fraud	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	fraud	0
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	fraud	0
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	fraud	0
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	fraud	0
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	fraud	0

6362620 rows × 11 columns

In [16]:
```python
type
```

Out[16]:
```
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: type, dtype: int64
```

In [17]:
```python
df.replace(to_replace=['CASH_OUT','PAYMENT','CASH_IN','TRANSFER','DEBIT'],value=[1,2,3,4,5],inplace=True)
```

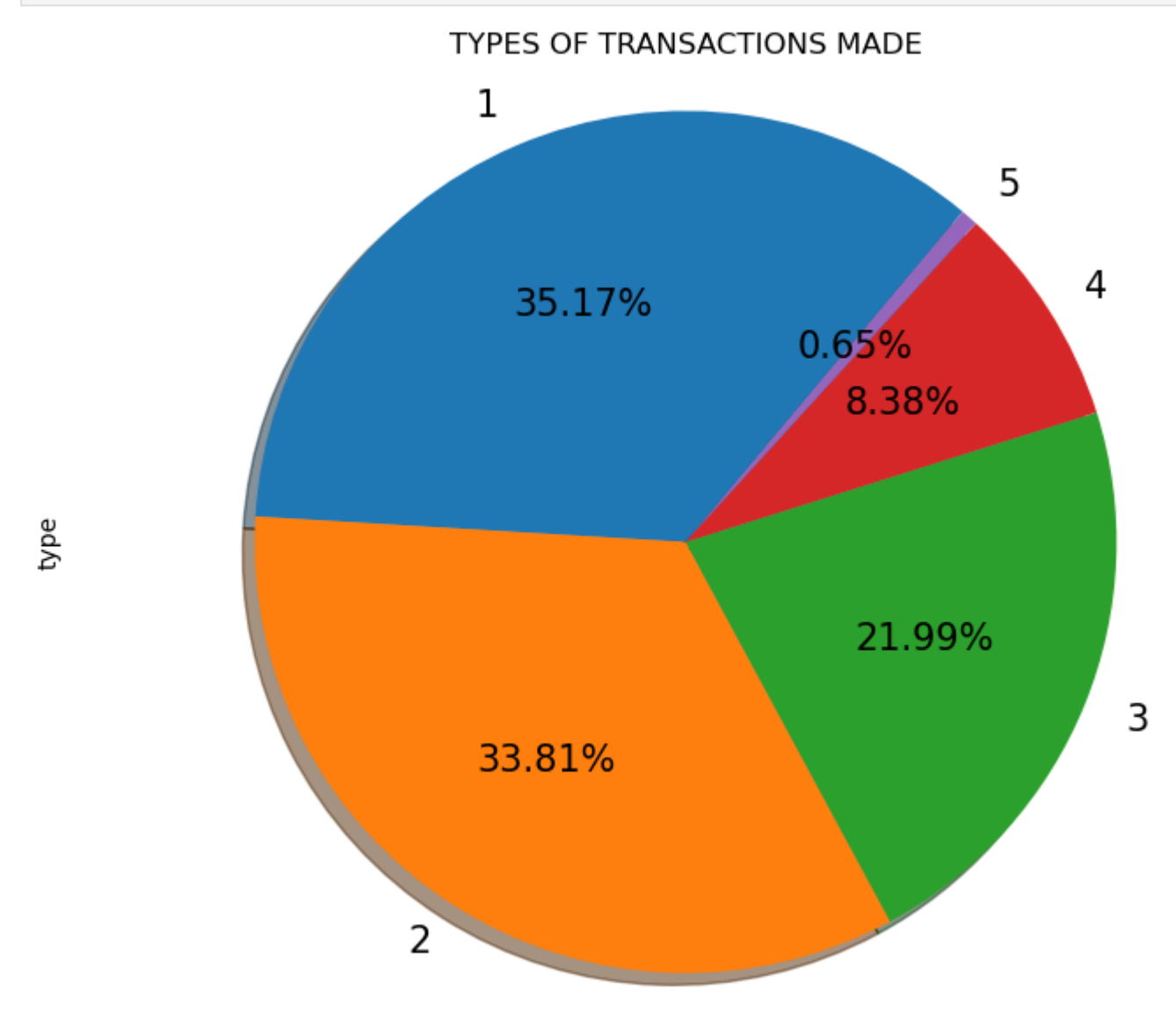
In [18]:
```python
df
```

Out[18]:

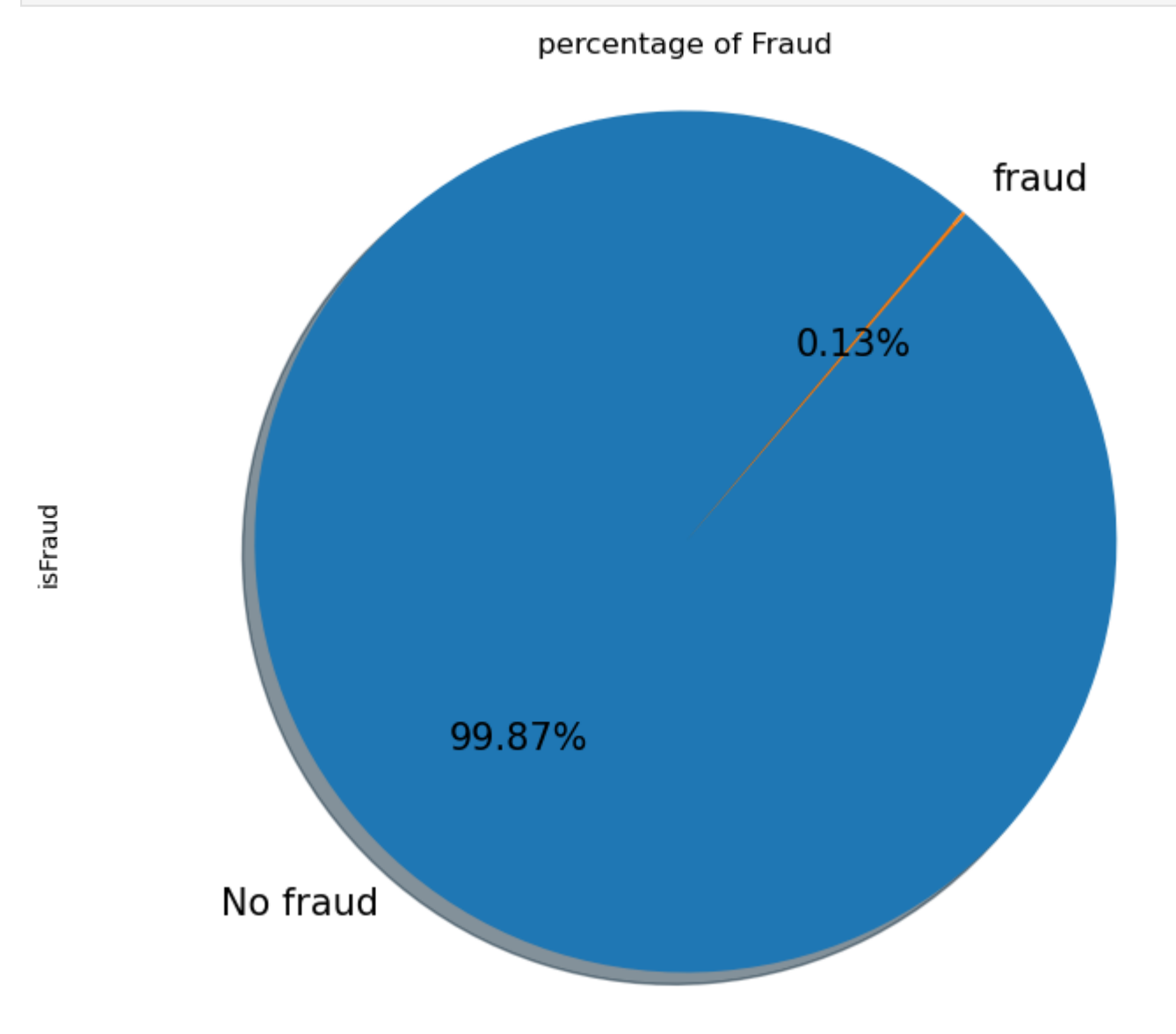
	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	2	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	No fraud	0
1	1	2	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	No fraud	0
2	1	4	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	fraud	0
3	1	1	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	fraud	0
4	1	2	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	No fraud	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	1	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	fraud	0
6362616	743	4	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	fraud	0
6362617	743	1	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	fraud	0
6362618	743	4	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	fraud	0
6362619	743	1	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	fraud	0

6362620 rows × 11 columns

In [19]:
```python
type_count = df.type.value_counts()
type_count.plot.pie(figsize=(9,7), autopct='%1.2f%%', shadow=True, fontsize=15,startangle=50)
plt.title('TYPES OF TRANSACTIONS MADE')
plt.axis('equal')
plt.show()
```



In [20]:
```python
isFraud_count = df.isFraud.value_counts()
isFraud_count.plot.pie(figsize=(9,7), autopct='%1.2f%%', shadow=True, fontsize=15,startangle=50)
plt.title('percentage of Fraud')
plt.axis('equal')
plt.show()
```



In [21]:
```python
 isFraud_count
```

Out[21]:
```
No fraud    6354407
fraud          8213
Name: isFraud, dtype: int64
```

In [22]:
```python
x=df[['type','amount','oldbalanceOrg','newbalanceOrig']]
```

In [23]:
```python
y=df.iloc[:,-2]
```

In [24]:
```python
y
```

Out[24]:
```
0          No fraud
1          No fraud
2             fraud
3             fraud
4          No fraud
             ...
6362615       fraud
6362616       fraud
6362617       fraud
6362618       fraud
6362619       fraud
Name: isFraud, Length: 6362620, dtype: object
```

In [25]:
```python
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [35]:
```python
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

In [36]:
```python
LR = LogisticRegression(random_state=42)
KN = KNeighborsClassifier()
DC = DecisionTreeClassifier(random_state=42)
RF = RandomForestClassifier(random_state=42)
```

In [37]:
```python
models = [LR,KN,DC,RF]
```

In [47]:
```python
def plot_confusion_matrix(y_test,prediction):
    cm_ = confusion_matrix(y_test,prediction)
    plt.figure(figsize = (6,4))
    sns.heatmap(cm_, cmap ='coolwarm', linecolor = 'white', linewidths = 1, annot = True, fmt = 'd')
    plt.title('Confusion Matrix')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.show()
```

In [48]:
```python
from sklearn.metrics import confusion_matrix
```

In [50]:
```python
def trainer(model,x_train,y_train,x_test,y_test):
    #fit your model
    model.fit(x_train,y_train)
    #predict on the fitted model
    prediction = model.predict(x_test)
    #print evaluation metric
    print('\nFor {}, Accuracy score is {} \n'.format(model.__class__.__name__,accuracy_score(prediction,y_test)))
    print(classification_report(y_test, prediction)) #use this later
    plot_confusion_matrix(y_test,prediction)
```

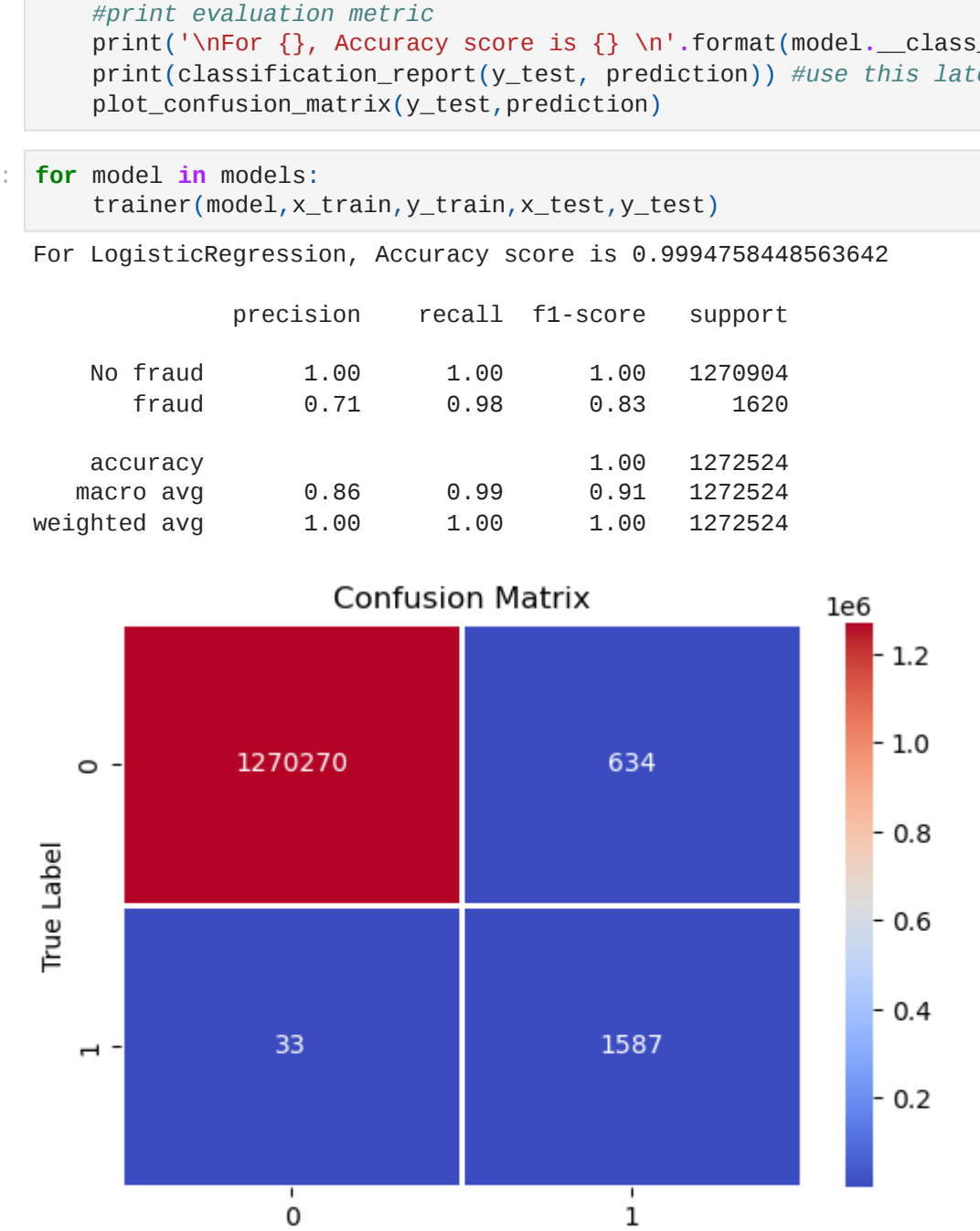
In [51]:
```python
for model in models:
    trainer(model,x_train,y_train,x_test,y_test)
```

```
For LogisticRegression, Accuracy score is 0.9994758448563642

              precision    recall  f1-score   support

    No fraud       1.00      1.00      1.00   1270904
       fraud       0.71      0.98      0.83      1620

    accuracy                           1.00   1272524
   macro avg       0.86      0.99      0.91   1272524
weighted avg       1.00      1.00      1.00   1272524
```

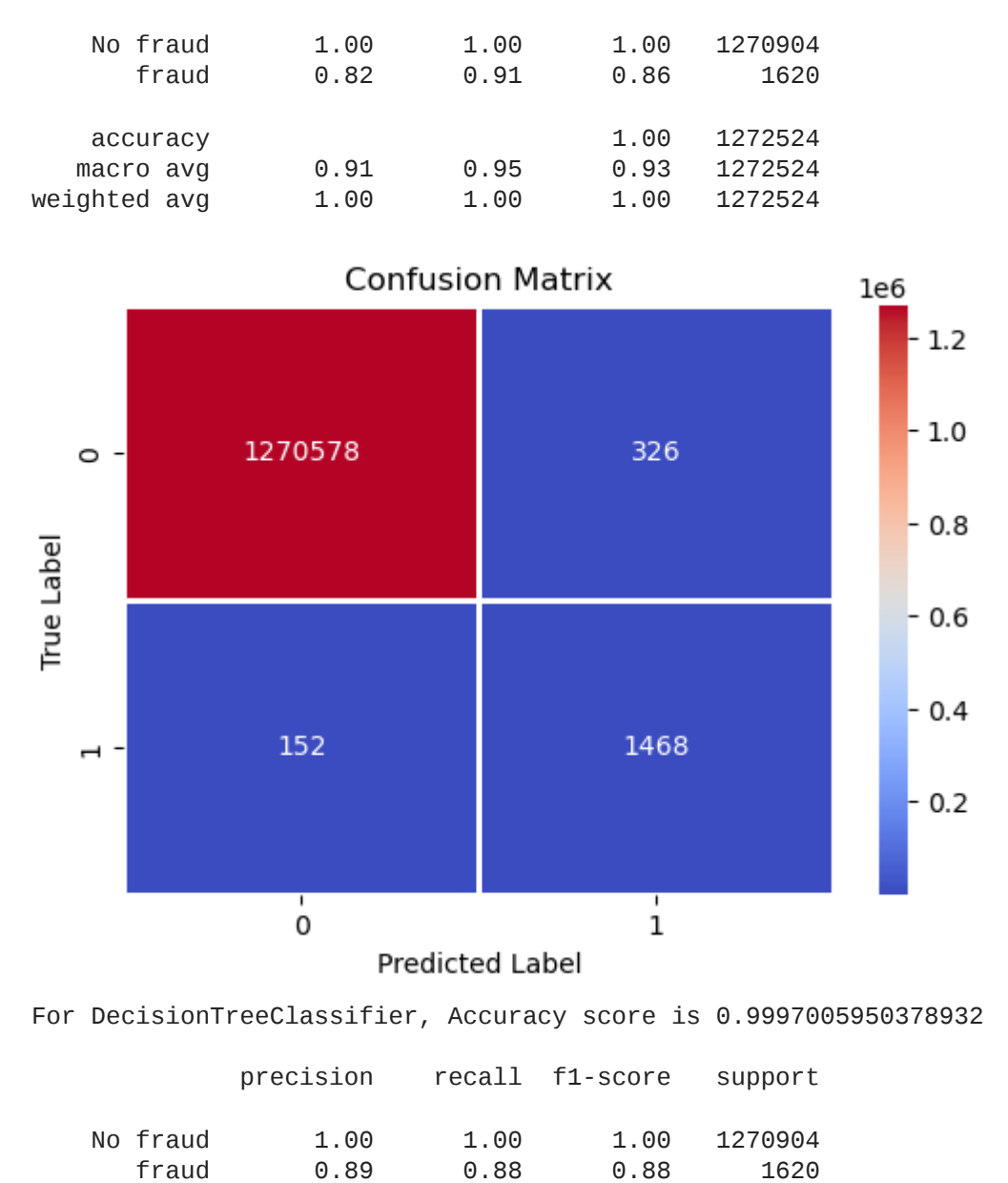



```
For KNeighborsClassifier, Accuracy score is 0.9996243685777243

              precision    recall  f1-score   support

    No fraud       1.00      1.00      1.00   1270904
       fraud       0.82      0.91      0.86      1620

    accuracy                           1.00   1272524
   macro avg       0.91      0.95      0.93   1272524
weighted avg       1.00      1.00      1.00   1272524
```

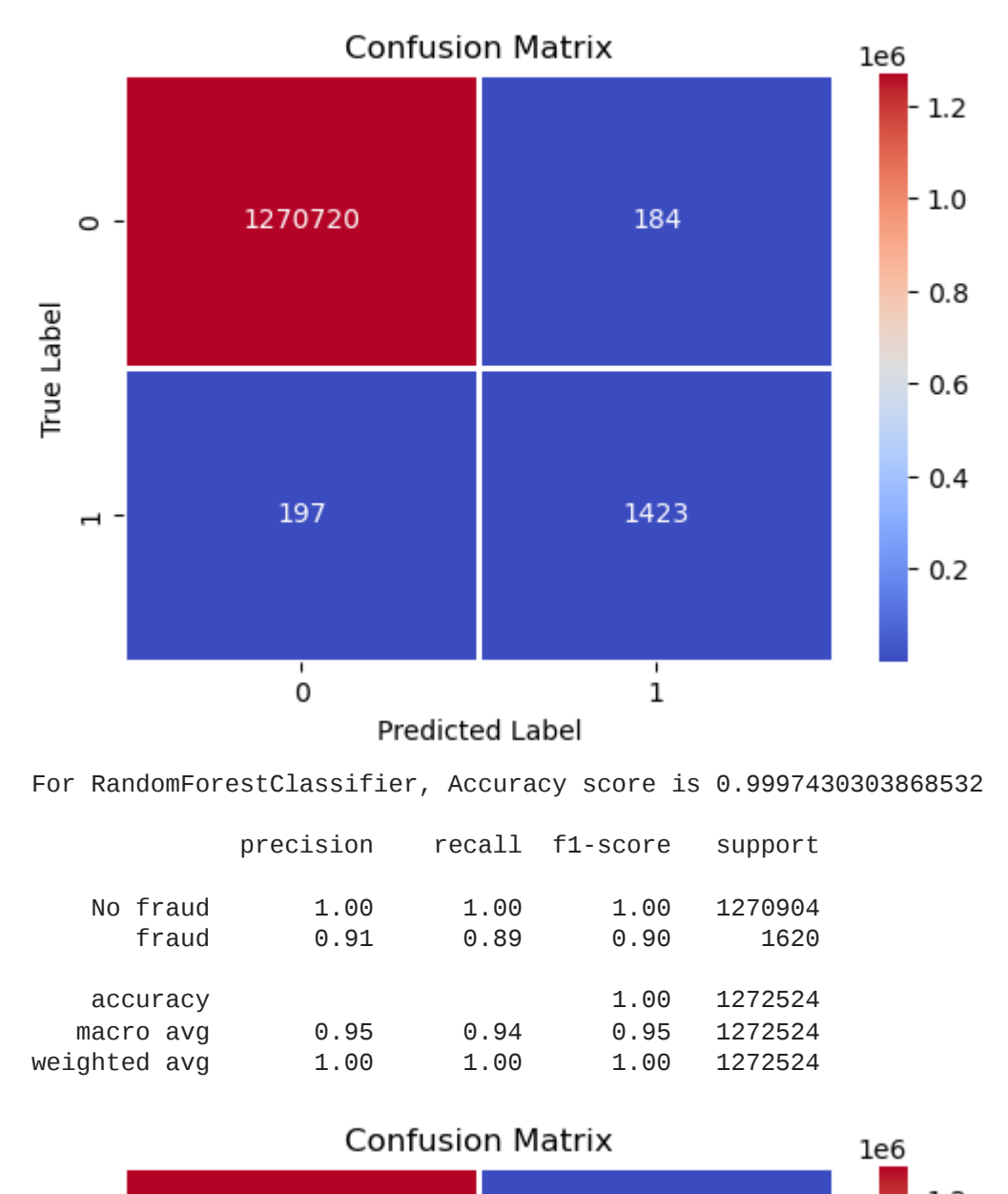



```
For DecisionTreeClassifier, Accuracy score is 0.9997005950378932

              precision    recall  f1-score   support

    No fraud       1.00      1.00      1.00   1270904
       fraud       0.89      0.88      0.88      1620

    accuracy                           1.00   1272524
   macro avg       0.94      0.94      0.94   1272524
weighted avg       1.00      1.00      1.00   1272524
```

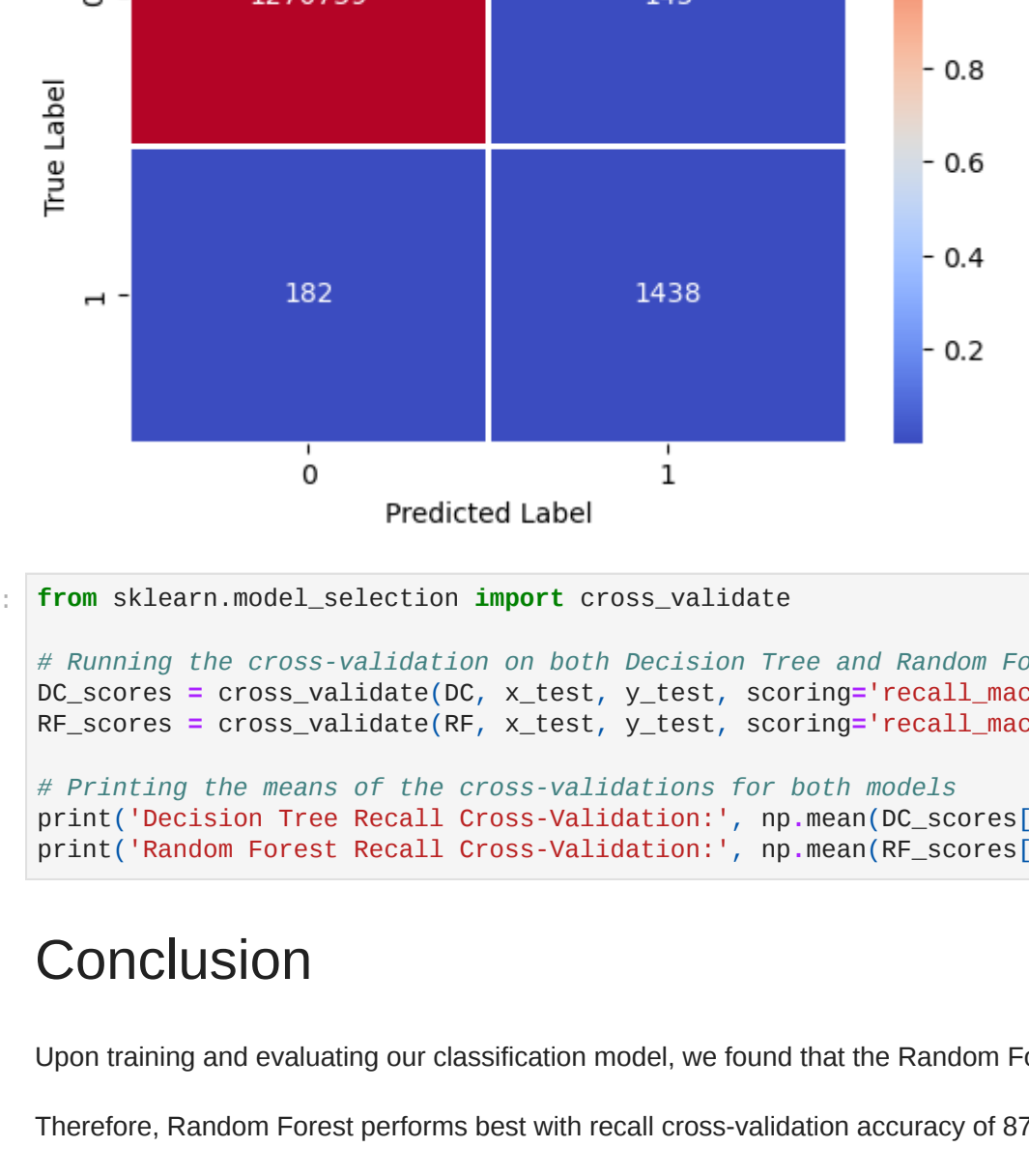



```
For RandomForestClassifier, Accuracy score is 0.9997430303868532

              precision    recall  f1-score   support

    No fraud       1.00      1.00      1.00   1270904
       fraud       0.91      0.89      0.90      1620

    accuracy                           1.00   1272524
   macro avg       0.95      0.94      0.95   1272524
weighted avg       1.00      1.00      1.00   1272524
```



In [ ]:
```python
from sklearn.model_selection import cross_validate

# Running the cross-validation on both Decision Tree and Random Forest models; specifying recall as the scoring metric
DC_scores = cross_validate(DC, x_test, y_test, scoring='recall_macro')
RF_scores = cross_validate(RF, x_test, y_test, scoring='recall_macro')

# Printing the means of the cross-validations for both models
print('Decision Tree Recall Cross-Validation:', np.mean(DC_scores['test_score']))
print('Random Forest Recall Cross-Validation:', np.mean(RF_scores['test_score']))
```

Conclusion

Upon training and evaluating our classification model, we found that the Random Forest model performed the best by a narrow margin.

Therefore, Random Forest performs best with recall cross-validation accuracy of 87% which is important for our problem statement where false negative is our priority

Recommendation

Transaction History and Frequency - if unaccounted transactions occurs frequently we should confirm genuinity of the transaction with the customer

Repeated wrong PIN or Password - We should halt the transaction and alert the customer immediately.

Make customers to change PIN or password often

Instruct user to use own mobile or computers while doing transactions to avoid phishing attacks

Increased cybersecurity for banking websites and mobile applications

Two factor authentication for transaction

Ensure that blossom bank hire a data engineer that will ensure the dataset is accurate, balanced for proper EDA as there are too many outliers in this data set. This will enable the business to build machime learning models that predict outcomes more accurately with better performance.