In [1]:
```
import pandas as pd
```

In [2]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
```

In [3]:
```
pwd
```

Out[3]:
```
'C:\\Users\\USER'
```

In [4]:
```
cd C:\Users\USER\Downloads
```

```
C:\Users\USER\Downloads
```

In [5]:
```
df=pd.read_csv('creditcard.csv')
```

In [6]:
```
df.head()
```

Out[6]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0
1	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
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0

5 rows × 31 columns

In [7]:
```
df.drop('Time',axis=1,inplace=True)
```

In [8]:
```
df.head()
```

Out[8]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739

5 rows × 30 columns

In [9]:
```
cases=len(df)
non_fraud_count=len(df[df.Class==0])
fraud_count=len(df[df.Class==1])
```

In [10]:
```
print('Fraud Cases: {}'.format(len(df[df['Class'] == 1])))
print('Valid Transactions: {}'.format(len(df[df['Class'] == 0])))
```

```
Fraud Cases: 492
Valid Transactions: 284315
```

In [11]:
```
fraud_percentage=round(fraud_count/non_fraud_count*100,3)
fraud_percentage
```

Out[11]: 0.173

In [12]:
```
sc=StandardScaler()
```

In [13]:
```
amount=df['Amount'].values
df['Amount']=sc.fit_transform(amount.reshape(-1,1))
```

In [14]:
```
df['Amount']
```

Out[14]:
```
0         0.244964
1        -0.342475
2         1.160686
3         0.140534
4        -0.073403
              ...
284802   -0.350151
284803   -0.254117
284804   -0.081839
284805   -0.313249
284806    0.514355
Name: Amount, Length: 284807, dtype: float64
```

In [15]:
```
x=df.drop('Class',axis=1).values
y=df['Class'].values
```

In [16]:
```
from imblearn.over_sampling import SMOTE

smote = SMOTE()

# fit predictor and target variable
x_smote, y_smote = smote.fit_resample(x, y)
```

In [18]:
```
from collections import Counter
print('Original dataset shape', Counter(y))
print('Resample dataset shape', Counter(y_smote))
```

```
Original dataset shape Counter({0: 284315, 1: 492})
```

```
Resample dataset shape Counter({0: 284315, 1: 284315})
```

In [ ]:
```
```

In [19]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random
_state=0)
```

In [20]:
```
tree_model=DecisionTreeClassifier(max_depth=4, criterion='entropy')
```

In [21]:
```
tree_model.fit(x_train,y_train)
```

Out[21]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

In [23]:
```
tree_yhat=tree_model.predict(x_test)
tree_yhat
```

Out[23]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [33]:
```
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
knn_yhat=knn.predict(x_test)
```

In [ ]:
```
svm = SVC(kernel="rbf",random_state=0)
svm.fit(x_train, y_train)
svm_yhat = svm.predict(x_test)
```

In [37]:
```
lg=LogisticRegression()
lg.fit(x_train,y_train)
lg_yhat=lg.predict(x_test)
```

In [25]:
```
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score
```

In [39]:
```
accuracy=accuracy_score(y_test,lg_yhat)
```

```
accuracy
```

Out[39]:
```
0.999204147794436
```

In [26]:
```
accuracy=accuracy_score(y_test,tree_yhat)
```

In [27]:
```
accuracy
```

Out[27]:
```
0.9993679997191109
```

In [30]:
```
Precision=precision_score(y_test,tree_yhat)
```

In [31]:
```
report=classification_report(y_test,tree_yhat)
```

In [32]:
```
report
```

Out[32]:
```
'              precision    recall  f1-score   support\n\n           0
1.00      1.00      1.00     85296\n           1       0.85      0.76
0.81       147\n\n    accuracy                           1.00     85443
\n   macro avg       0.93      0.88      0.90     85443\nweighted avg
1.00      1.00      1.00     85443\n'
```

In [35]:
```
cMatrix=metrics.confusion_matrix(tree_yhat,y_test)
```

In [36]:
```
cMatrix
```

Out[36]:
```
array([[85277,    35],
       [   19,   112]], dtype=int64)
```

In [40]:
```
report=classification_report(y_test,lg_yhat)
```

In [41]:
```
report
```

Out[41]:
```
'              precision    recall  f1-score   support\n\n           0
1.00      1.00      1.00     85296\n           1       0.88      0.62
0.73       147\n\n    accuracy                           1.00     85443
```

```
\n   macro avg       0.94      0.81      0.86     85443\nweighted avg
1.00      1.00      1.00     85443\n'
```

In [ ]: