In [1]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
train_location = "test_2019.csv"
train_df = pd.read_csv(train_location)
test_location = "train_2019.csv"
test_df = pd.read_csv(test_location)
train_df.head()
```

Out[2]:

| | Unnamed: 0 | trans_date_trans_time | cc_num | merchant | category | amt | first | last | gender | street | ... | lat | long | city_pop | job | dob | trans_num | unix_ti |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 2020-06-21 12:14:25 | 2291163933867244 | fraud_Kirlin and Sons | personal_care | 2.86 | Jeff | Elliott | M | 351 Darlene Green | ... | 33.9659 | -80.9355 | 333497 | Mechanical engineer | 1968-03-19 | 2da90c7d74bd46a0caf3777415b3ebd3 | 1371816 |
| **1** | 1 | 2020-06-21 12:14:33 | 3573030041201292 | fraud_Sporer-Keebler | personal_care | 29.84 | Joanne | Williams | F | 3638 Marsh Union | ... | 40.3207 | -110.4360 | 302 | Sales professional, IT | 1990-01-17 | 324cc204407e99f51b0d6ca0055005e7 | 1371816 |
| **2** | 2 | 2020-06-21 12:14:53 | 3598215285024754 | fraud_Swaniawski, Nitzsche and Welch | health_fitness | 41.28 | Ashley | Lopez | F | 9333 Valentine Point | ... | 40.6729 | -73.5365 | 34496 | Librarian, public | 1970-10-21 | c81755dbbbea9d5c77f094348a7579be | 1371816 |
| **3** | 3 | 2020-06-21 12:15:15 | 3591919803438423 | fraud_Haley Group | misc_pos | 60.05 | Brian | Williams | M | 32941 Krystal Mill Apt. 552 | ... | 28.5697 | -80.8191 | 54767 | Set designer | 1987-07-25 | 2159175b9efe66dc301f149d3d5abf8c | 1371816 |
| **4** | 4 | 2020-06-21 12:15:17 | 3526826139003047 | fraud_Johnston-Casper | travel | 3.19 | Nathan | Massey | M | 5783 Evan Roads Apt. 465 | ... | 44.2529 | -85.0170 | 1126 | Furniture designer | 1955-07-06 | 57ff021bd3f328f8738bb535c302a31b | 1371816 |

5 rows × 23 columns

In [3]:
```
train_df.isnull().sum()
```

Out[3]:
```
Unnamed: 0              0
trans_date_trans_time   0
cc_num                  0
merchant                0
category                0
amt                     0
first                   0
last                    0
gender                  0
street                  0
city                    0
state                   0
zip                     0
lat                     0
long                    0
city_pop                0
job                     0
dob                     0
trans_num               0
unix_time               0
merch_lat               0
merch_long              0
is_fraud                0
dtype: int64
```

In [4]:
```
def data_pre(X):

    del_col=['merchant','first','last','street','zip','unix_time','Unnamed: 0','trans_num','cc_num']
    X.drop(columns=del_col,inplace=True)

    X['trans_date_trans_time']=pd.to_datetime(X['trans_date_trans_time'])
    X['trans_date']=X['trans_date_trans_time'].dt.strftime('%Y-%m-%d')
    X['trans_date']=pd.to_datetime(X['trans_date'])
    X['dob']=pd.to_datetime(X['dob'])

    X["age"] = (X["trans_date"] - X["dob"]).dt.days //365
    X['trans_month']=X['trans_date'].dt.month
    X['trans_year']=X['trans_date'].dt.year

    X['gender']=X['gender'].apply(lambda x : 1 if x=='M' else 0)
    X['gender']=X['gender'].astype(int)

    X['lat_dis']=abs(X['lat']-X['merch_lat'])
    X['long_dis']=abs(X['long']-X['merch_long'])

    X=pd.get_dummies(X,columns=['category'])
    X=X.drop(columns=['city','trans_date_trans_time','state','job','merch_lat','merch_long','lat','long','dob','trans_date'])
    return X
```

In [5]:
```
train_df_pre=data_pre(train_df.copy())
train_df_pre.head()
test_df_pre=data_pre(test_df.copy())
test_df_pre.head()
```

Out[5]:

| | amt | gender | city_pop | is_fraud | age | trans_month | trans_year | lat_dis | long_dis | category_entertainment | ... | category_grocery_pos | category_health_fitness | category_home | category_kids_pets | category_misc_net | category_ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2.86 | 1 | 333497 | 0 | 52 | 6 | 2020 | 0.020491 | 0.265214 | False | ... | False | False | False | False | False | |
| **1** | 29.84 | 0 | 302 | 0 | 30 | 6 | 2020 | 0.870202 | 0.475569 | False | ... | False | False | False | False | False | |
| **2** | 41.28 | 0 | 34496 | 0 | 49 | 6 | 2020 | 0.177090 | 0.659611 | False | ... | False | True | False | False | False | |
| **3** | 60.05 | 1 | 54767 | 0 | 32 | 6 | 2020 | 0.242698 | 0.063961 | False | ... | False | False | False | False | False | |
| **4** | 3.19 | 1 | 1126 | 0 | 65 | 6 | 2020 | 0.706248 | 0.867734 | False | ... | False | False | False | False | False | |

5 rows × 23 columns

In [6]:
```
x_train=train_df_pre.drop('is_fraud',axis=1)
y_train=train_df_pre['is_fraud']
x_test=test_df_pre.drop('is_fraud',axis=1)
y_test=test_df_pre['is_fraud']
```

In [7]:
```
scaler = StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

In [8]:
```
logistic_regression=LogisticRegression()
logistic_regression.fit(x_train,y_train)
y_pred_logistic = logistic_regression.predict(x_test)
accuracy_logistic = accuracy_score(y_test, y_pred_logistic)
accuracy_logistic
print("\nClassification Report for Logistic Regression:\n", classification_report(y_test, y_pred_logistic))
```

```
Classification Report for Logistic Regression:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    553574
           1       0.00      0.00      0.00      2145

    accuracy                           1.00    555719
   macro avg       0.50      0.50      0.50    555719
weighted avg       0.99      1.00      0.99    555719
```

In [9]:
```
DecisionTree=DecisionTreeClassifier()
DecisionTree.fit(x_train,y_train)
y_pred_dt = DecisionTree.predict(x_test)
accuracy_dt = accuracy_score(y_test, y_pred_dt)
accuracy_dt
print("\nClassification Report for Decision Tree:\n", classification_report(y_test, y_pred_dt))
```

```
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    553574
           1       1.00      1.00      1.00      2145

    accuracy                           1.00    555719
   macro avg       1.00      1.00      1.00    555719
weighted avg       1.00      1.00      1.00    555719
```

In [10]:
```
random_forest = RandomForestClassifier(random_state=42,n_estimators=100)
random_forest.fit(x_train, y_train)
y_pred_rf = random_forest.predict(x_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
accuracy_rf
print("\nClassification Report for Random Forest:\n", classification_report(y_test, y_pred_rf))
```

```
Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    553574
           1       1.00      1.00      1.00      2145

    accuracy                           1.00    555719
   macro avg       1.00      1.00      1.00    555719
weighted avg       1.00      1.00      1.00    555719
```

In [ ]: