In [1]:

```
import pandas as pd
import numpy as np
```

In [38]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
```

In [17]:

```
dataset1 = pd.read_csv('fraudTrain.csv')
```

In [18]:

```
dataset1
```

Out[18]:

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	a
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41
...	...	...	...	...	...	
1296670	1296670	2020-06-21 12:12:08	30263540414123	fraud_Reichel Inc	entertainment	15
1296671	1296671	2020-06-21 12:12:19	6011149206456997	fraud_Abernathy and Sons	food_dining	51
1296672	1296672	2020-06-21 12:12:32	3514865930894695	fraud_Stiedemann Ltd	food_dining	105
1296673	1296673	2020-06-21 12:13:36	2720012583106919	fraud_Reinger, Weissnat and Strosin	food_dining	74
1296674	1296674	2020-06-21 12:13:37	4292902571056973207	fraud_Langosh, Wintheiser and Hyatt	food_dining	4

1296675 rows × 23 columns

In [19]:

```
dataset2 = pd.read_csv('fraudTest.csv')
```

In [20]:

```
dataset2
```

Out[20]:

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt
0	0	2020-06-21 12:14:25	2291163933867244	fraud_Kirlin and Sons	personal_care	2.86
1	1	2020-06-21 12:14:33	3573030041201292	fraud_Sporer-Keebler	personal_care	29.84
2	2	2020-06-21 12:14:53	3598215285024754	fraud_Swaniawski, Nitzsche and Welch	health_fitness	41.28
3	3	2020-06-21 12:15:15	3591919803438423	fraud_Haley Group	misc_pos	60.05
4	4	2020-06-21 12:15:17	3526826139003047	fraud_Johnston-Casper	travel	3.19
...	...	...	...	...	...	...
555714	555714	2020-12-31 23:59:07	30560609640617	fraud_Reilly and Sons	health_fitness	43.77
555715	555715	2020-12-31 23:59:09	3556613125071656	fraud_Hoppe-Parisian	kids_pets	111.84
555716	555716	2020-12-31 23:59:15	6011724471098086	fraud_Rau-Robel	kids_pets	86.88
555717	555717	2020-12-31 23:59:24	4079773899158	fraud_Breitenberg LLC	travel	7.99
555718	555718	2020-12-31 23:59:34	4170689372027579	fraud_Dare-Marvin	entertainment	38.13

555719 rows × 23 columns

In [42]:

```
dataset1.info() , dataset2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1296675 entries, 0 to 1296674
Data columns (total 23 columns):
 #   Column                 Non-Null Count    Dtype  
---  ------                 --------------    -----  
 0   Unnamed: 0             1296675 non-null  int64  
 1   trans_date_trans_time  1296675 non-null  object 
 2   cc_num                 1296675 non-null  int64  
 3   merchant               1296675 non-null  object 
 4   category               1296675 non-null  object 
 5   amt                    1296675 non-null  float64
 6   first                  1296675 non-null  object 
 7   last                   1296675 non-null  object 
 8   gender                 1296675 non-null  object 
 9   street                 1296675 non-null  object 
 10  city                   1296675 non-null  object 
 11  state                  1296675 non-null  object 
 12  zip                    1296675 non-null  int64  
 13  lat                    1296675 non-null  float64
 14  long                   1296675 non-null  float64
 15  city_pop               1296675 non-null  int64  
 16  job                    1296675 non-null  object 
 17  dob                    1296675 non-null  object 
 18  trans_num              1296675 non-null  object 
 19  unix_time              1296675 non-null  int64  
 20  merch_lat              1296675 non-null  float64
 21  merch_long             1296675 non-null  float64
 22  is_fraud               1296675 non-null  int64  
dtypes: float64(5), int64(6), object(12)
memory usage: 227.5+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 555719 entries, 0 to 555718
Data columns (total 23 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   Unnamed: 0             555719 non-null  int64  
 1   trans_date_trans_time  555719 non-null  object 
 2   cc_num                 555719 non-null  int64  
 3   merchant               555719 non-null  object 
 4   category               555719 non-null  object 
 5   amt                    555719 non-null  float64
 6   first                  555719 non-null  object 
 7   last                   555719 non-null  object 
 8   gender                 555719 non-null  object 
 9   street                 555719 non-null  object 
 10  city                   555719 non-null  object 
 11  state                  555719 non-null  object 
 12  zip                    555719 non-null  int64  
 13  lat                    555719 non-null  float64
 14  long                   555719 non-null  float64
 15  city_pop               555719 non-null  int64  
 16  job                    555719 non-null  object 
 17  dob                    555719 non-null  object 
 18  trans_num              555719 non-null  object 
 19  unix_time              555719 non-null  int64  
 20  merch_lat              555719 non-null  float64
 21  merch_long             555719 non-null  float64
 22  is_fraud               555719 non-null  int64  
dtypes: float64(5), int64(6), object(12)
memory usage: 97.5+ MB
```

Out[42]:
```
(None, None)
```

In [23]:
```
print("Checking Duplicate values:",dataset1.duplicated().sum(), dataset2.duplicated().
print("Checking Null values:", dataset1.isna().sum().sum(), dataset2.isna().sum().sum(
```

```
Checking Duplicate values: 0 0
Checking Null values: 0 0
```

In [24]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [25]:
```
# data preprocessing for Training Data

train_data = dataset1[['gender', "category", "amt", "city", "job", "is_fraud"]]

train_data.loc[:, 'category'] = le.fit_transform(train_data["category"])
train_data.loc[:, 'city'] = le.fit_transform(train_data["city"])
train_data.loc[:, 'job'] = le.fit_transform(train_data["job"])
train_data.loc[:, 'gender'] = train_data["gender"].map({'M': 0, 'F': 1})
```

In [26]:
```
train_data.head(10)
```

Out[26]:

	gender	category	amt	city	job	is_fraud
0	1	8	4.97	526	370	0
1	1	4	107.23	612	428	0
2	0	0	220.11	468	307	0
3	0	2	45.00	84	328	0
4	0	9	41.96	216	116	0
5	1	2	94.63	223	479	0
6	1	3	44.54	351	29	0
7	0	2	71.65	236	127	0
8	1	9	4.27	474	375	0
9	1	4	198.39	149	329	0

In [27]:
```
# data preprocessing for Testing Data

test_data = dataset2[['gender',"category","amt","city", "job","is_fraud"]]

test_data.loc[:, 'category'] = le.fit_transform(test_data["category"])
test_data.loc[:, 'city'] = le.fit_transform(test_data["city"])
test_data.loc[:, 'job'] = le.fit_transform(test_data["job"])
test_data.loc[:, 'gender'] = test_data["gender"].map({'M': 0, 'F': 1})
```

In [28]:
```
test_data.head(10)
```

Out[28]:

	gender	category	amt	city	job	is_fraud
0	0	10	2.86	157	275	0
1	1	10	29.84	16	392	0
2	1	5	41.28	61	259	0
3	0	9	60.05	764	407	0
4	0	13	3.19	247	196	0
5	1	7	19.55	90	361	0
6	1	5	133.93	117	455	0
7	1	10	10.37	725	124	0
8	0	12	4.37	503	13	0
9	1	1	66.54	624	41	0

In [29]:

```python
X_train = train_data[['gender', "category", "amt", "city", "job"]]
Y_train = train_data["is_fraud"]
```

In [30]:

```python
X_test = test_data[['gender', "category", "amt", "city", "job"]]
Y_test = test_data["is_fraud"]
```

In [31]:

```python
log_reg = LogisticRegression()
log_reg.fit(X_train, Y_train)
```

Out[31]:

```
▾ LogisticRegression
LogisticRegression()
```

In [33]:

```python
decision_tree = DecisionTreeClassifier(random_state=50)
decision_tree.fit(X_train, Y_train)
```

Out[33]:

```
▾          DecisionTreeClassifier
DecisionTreeClassifier(random_state=50)
```

In [34]:

```python
random_forest = RandomForestClassifier(random_state=42)
random_forest.fit(X_train, Y_train)
```

Out[34]:

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

In [39]:

```python
# Logistic Regression
y_pred_log_reg = log_reg.predict(X_test)
print(f"Accuracy: {accuracy_score(Y_test, y_pred_log_reg)}")
print(classification_report(Y_test, y_pred_log_reg))
```

```
Accuracy: 0.9955013235106231
              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 [40]:

```
# Decision Tree
y_pred_tree = decision_tree.predict(X_test)
print(f"Accuracy: {accuracy_score(Y_test, y_pred_tree)}")
print(classification_report(Y_test, y_pred_tree))
```

```
Accuracy: 0.9959026054534756
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    553574
           1       0.48      0.59      0.52      2145

    accuracy                           1.00    555719
   macro avg       0.74      0.79      0.76    555719
weighted avg       1.00      1.00      1.00    555719
```

In [41]:

```
# Random Forest
y_pred_forest = random_forest.predict(X_test)
print(f"Accuracy: {accuracy_score(Y_test, y_pred_forest)}")
print(classification_report(Y_test, y_pred_forest))
```

```
Accuracy: 0.9974987358719065
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    553574
           1       0.71      0.61      0.65      2145

    accuracy                           1.00    555719
   macro avg       0.85      0.80      0.83    555719
weighted avg       1.00      1.00      1.00    555719
```

In [ ]: