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

In [2]:
```
train_data=pd.read_csv("fraudTrain.csv")
```

In [3]:
```
train_data.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
```

In [4]:
```
train_data.columns
```

Out[4]:
```
Index(['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category',
       'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip',
       'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud'],
      dtype='object')
```

In [5]:
```
train_data.drop(columns=['Unnamed: 0','cc_num','first', 'last', 'street', 'city', '
```

In [6]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
train_data["merchant"] = encoder.fit_transform(train_data["merchant"])
train_data["category"] = encoder.fit_transform(train_data["category"])
train_data["gender"] = encoder.fit_transform(train_data["gender"])
train_data["job"] = encoder.fit_transform(train_data["job"])
```

In [7]:
```
train_data
```

Out[7]:

	merchant	category	amt	gender	lat	long	city_pop	job	unix_tir
0	514	8	4.97	0	36.0788	-81.1781	3495	370	13253760
1	241	4	107.23	0	48.8878	-118.2105	149	428	13253760
2	390	0	220.11	1	42.1808	-112.2620	4154	307	13253760
3	360	2	45.00	1	46.2306	-112.1138	1939	328	13253760
4	297	9	41.96	1	38.4207	-79.4629	99	116	13253761
...	...	...	...	...	...	...	...	...	
1296670	499	0	15.56	1	37.7175	-112.4777	258	215	13718167
1296671	2	1	51.70	1	39.2667	-77.5101	100	360	13718167
1296672	599	1	105.93	1	32.9396	-105.8189	899	308	13718167
1296673	509	1	74.90	1	43.3526	-102.5411	1126	485	13718168
1296674	370	1	4.30	1	45.8433	-113.8748	218	467	13718168

1296675 rows × 12 columns

In [8]:
```
X = train_data.drop(columns=["is_fraud"], inplace = False)
Y = train_data["is_fraud"]
```

In [9]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_s
```

In [10]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.tree import DecisionTreeClassifier
```

In [11]:
```
model1 = DecisionTreeClassifier()
```

In [13]:
```
model1.fit(X_train, y_train)
y_pred1_1 = model1.predict(X_test)

accuracy = accuracy_score(y_test, y_pred1_1)
precision = precision_score(y_test, y_pred1_1)
recall = recall_score(y_test, y_pred1_1)
f1 = f1_score(y_test, y_pred1_1)
print(f"\n Accuracy: {accuracy}")
print(f" Precision: {precision}")
print(f" Recall: {recall}")
print(f" F1 Score: {f1}")
```

```
Accuracy: 0.9959666068984132
Precision: 0.6516497461928934
Recall: 0.6738845144356955
F1 Score: 0.6625806451612903
```

In [14]:
```
normal = train_data[train_data['is_fraud']==0]
fraud = train_data[train_data['is_fraud']==1]
```

In [15]:
```
normal_sample = normal.sample(n=fraud.shape[0])
```

In [16]:
```
new_data = pd.concat([normal_sample,fraud], ignore_index=True)
```

In [17]:
```
new_data.head()
```

Out[17]:

	merchant	category	amt	gender	lat	long	city_pop	job	unix_time	mer
0	93	9	256.66	1	37.6395	-97.1714	409656	223	1352492333	36.7
1	142	8	505.03	0	32.0758	-96.7010	1563	194	1367083222	32.7
2	316	1	32.41	0	27.4703	-81.4872	50835	178	1330532538	28.1
3	482	7	24.24	0	39.6747	-76.8941	11751	90	1368138130	39.8
4	395	4	134.12	1	40.8731	-96.1528	1517	120	1358487628	40.1

In [18]:
```
new_data['is_fraud'].value_counts()
X = new_data.drop('is_fraud', axis = 1)
y= new_data['is_fraud']
```

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_s
```

In [20]:
```
model2 = DecisionTreeClassifier()
model2.fit(X_train, y_train)
y_pred1_2 = model2.predict(X_test)
print(f"\n Accuaracy: {accuracy_score(y_test, y_pred1_2)}")
print(f"\n Precision: {precision_score(y_test, y_pred1_2)}")
print(f"\n Recall: {recall_score(y_test, y_pred1_2)}")
print(f"\n F1 Score: {f1_score(y_test, y_pred1_2)}")
```

```
Accuaracy: 0.9497169497169498

Precision: 0.9508196721311475

Recall: 0.9501965923984272

F1 Score: 0.9505080301540478
```

In [21]:
```
X = train_data.drop('is_fraud', axis = 1)
y= train_data['is_fraud']
```

In [22]:
```
from imblearn.over_sampling import SMOTE
```

In [23]:
```
from imblearn.over_sampling import SMOTE
```

In [24]:
```
X_ouver, y_ouver = SMOTE().fit_resample(X,y)
y_ouver.value_counts()
```

Out[24]:
```
is_fraud
0    1289169
1    1289169
Name: count, dtype: int64
```

In [25]:
```
X_train, X_test, y_train, y_test = train_test_split(X_ouver, y_ouver, test_size = 0
```

In [27]:
```
model3 = DecisionTreeClassifier()
model3.fit(X_train, y_train)
y_pred1_3 = model3.predict(X_test)
print(f"\n Accuaracy: {accuracy_score(y_test, y_pred1_3)}")
print(f"\n Precision: {precision_score(y_test, y_pred1_3)}")
print(f"\n Recall: {recall_score(y_test, y_pred1_3)}")
print(f"\n F1 Score: {f1_score(y_test, y_pred1_3)}")
```

```
Accuaracy: 0.982395649914286

Precision: 0.9776384719750634

Recall: 0.9873457520931912

F1 Score: 0.9824681344148319
```

In [32]:
```
test_data=pd.read_csv("fraudTest.csv")
test_data
```

Out[32]:

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	categor
0	0	2020-06-21 12:14:25	2291163933867244	fraud_Kirlin and Sons	personal_ca
1	1	2020-06-21 12:14:33	3573030041201292	fraud_Sporer-Keebler	personal_ca
2	2	2020-06-21 12:14:53	3598215285024754	fraud_Swaniawski, Nitzsche and Welch	health_fitne
3	3	2020-06-21 12:15:15	3591919803438423	fraud_Haley Group	misc_pc
4	4	2020-06-21 12:15:17	3526826139003047	fraud_Johnston-Casper	trav
...	...	...	...	...	
555714	555714	2020-12-31 23:59:07	30560609640617	fraud_Reilly and Sons	health_fitne
555715	555715	2020-12-31 23:59:09	3556613125071656	fraud_Hoppe-Parisian	kids_pe
555716	555716	2020-12-31 23:59:15	6011724471098086	fraud_Rau-Robel	kids_pe
555717	555717	2020-12-31 23:59:24	4079773899158	fraud_Breitenberg LLC	trav
555718	555718	2020-12-31 23:59:34	4170689372027579	fraud_Dare-Marvin	entertainme

555719 rows × 23 columns

In [33]:
```
test_data.drop(columns=['Unnamed: 0','cc_num','first', 'last', 'street', 'city', 's
encoder = LabelEncoder()
test_data["merchant"] = encoder.fit_transform(test_data["merchant"])
test_data["category"] = encoder.fit_transform(test_data["category"])
test_data["gender"] = encoder.fit_transform(test_data["gender"])
test_data["job"] = encoder.fit_transform(test_data["job"])
```

In [34]:
```
test_data
```

Out[34]:

	merchant	category	amt	gender	lat	long	city_pop	job	unix_tim
0	319	10	2.86	1	33.9659	-80.9355	333497	275	137181686
1	591	10	29.84	0	40.3207	-110.4360	302	392	137181687
2	611	5	41.28	0	40.6729	-73.5365	34496	259	137181689
3	222	9	60.05	1	28.5697	-80.8191	54767	407	137181691
4	292	13	3.19	1	44.2529	-85.0170	1126	196	137181691
...	...	...	...	...	...	...	...	...	
555714	507	5	43.77	1	40.4931	-91.8912	519	460	138853434
555715	264	7	111.84	1	29.0393	-95.4401	28739	198	138853434
555716	496	7	86.88	0	46.1966	-118.9017	3684	294	138853435
555717	75	13	7.99	1	44.6255	-116.4493	129	58	138853436
555718	125	0	38.13	1	35.6665	-97.4798	116001	276	138853437

555719 rows × 12 columns

In [35]:
```
X_test = test_data.drop(columns=["is_fraud"], inplace = False)
Y_test = test_data["is_fraud"]
```

In [36]:
```
y_pred1 = model1.predict(X_test)
print(f"\n Accuaracy: {accuracy_score(Y_test, y_pred1)}")
print(f"\n Precision: {precision_score(Y_test, y_pred1)}")
print(f"\n Recall: {recall_score(Y_test, y_pred1)}")
print(f"\n F1 Score: {f1_score(Y_test, y_pred1)}")
```

```
Accuaracy: 0.993163811206743

Precision: 0.29985479186834463

Recall: 0.5776223776223777

F1 Score: 0.39477457384100684
```

In [37]:
```
y_pred2 = model2.predict(X_test)
print(f"\n Accuaracy: {accuracy_score(Y_test, y_pred2)}")
print(f"\n Precision: {precision_score(Y_test, y_pred2)}")
print(f"\n Recall: {recall_score(Y_test, y_pred2)}")
print(f"\n F1 Score: {f1_score(Y_test, y_pred2)}")
```

```
Accuaracy: 0.9495068550832345

Precision: 0.06647930676837632

Recall: 0.9263403263403264

F1 Score: 0.12405569082849473
```

In [38]:
```
y_pred3 = model3.predict(X_test)
print(f"\n Accuaracy: {accuracy_score(Y_test, y_pred3)}")
print(f"\n Precision: {precision_score(Y_test, y_pred3)}")
print(f"\n Recall: {recall_score(Y_test, y_pred3)}")
print(f"\n F1 Score: {f1_score(Y_test, y_pred3)}")
```

```
Accuaracy: 0.9649607085595417

Precision: 0.042654278625349734

Recall: 0.37668997668997667

F1 Score: 0.07663125948406677
```

In [39]:
```
import numpy as np
from scipy import stats
combined_predictions = np.vstack((y_pred1, y_pred2, y_pred3)).T
final_predictions = stats.mode(combined_predictions, axis=1)[0].flatten()
```

In [40]:
```
print(f"\n Accuaracy: {accuracy_score(Y_test, final_predictions)}")
print(f"\n Precision: {precision_score(Y_test, final_predictions)}")
print(f"\n Recall: {recall_score(Y_test, final_predictions)}")
print(f"\n F1 Score: {f1_score(Y_test, final_predictions)}")
```

```
Accuaracy: 0.9909684570799271

Precision: 0.24023861171366595

Recall: 0.6195804195804195

F1 Score: 0.3462289957014459
```

In [ ]:

In [ ]: