In [1]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score

from sklearn import metrics
from collections import Counter
```

In [2]:
```python
import os
os.chdir("C:\\Users\\kumar\\OneDrive\\Desktop\\Machine Learning")
```

In [3]:
```python
try:
    train_df = pd.read_csv("C:\\Users\\kumar\\OneDrive\\Desktop\\Machine Learning\\fraudTrain.csv")
    test_df = pd.read_csv("C:\\Users\\kumar\\OneDrive\\Desktop\\Machine Learning\\fraudTest.csv")
except:
    train_df = pd.read_csv('fraudTrain.csv')
    test_df = pd.read_csv('fraudTest.csv')
```

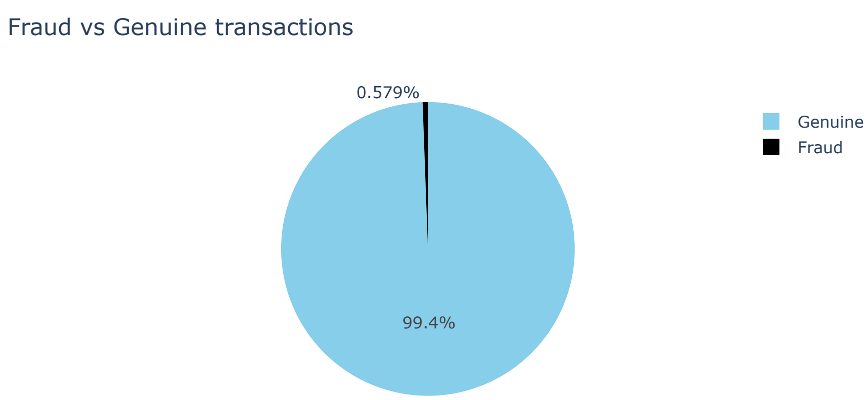
In [4]:
```python
train_df.head()
```

Out[4]:

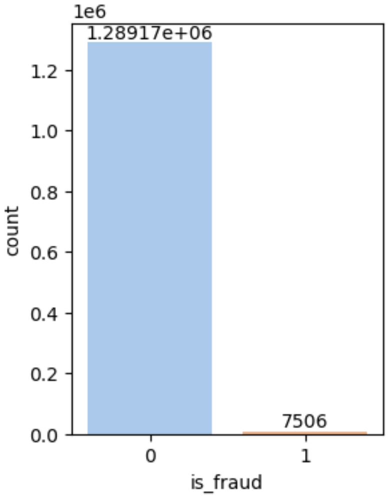
	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	...	lat	long	city_
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	...	36.0788	-81.1781	3
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	...	48.8878	-118.2105	
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	...	42.1808	-112.2620	4
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	9443 Cynthia Court Apt. 038	...	46.2306	-112.1138	1
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	408 Bradley Rest	...	38.4207	-79.4629	

5 rows × 23 columns

In [5]:
```python
fig = px.pie(values=train_df['is_fraud'].value_counts(), names=["Genuine","Fraud"] , width=700, height=400, color_discrete_sequence=[
             ,title="Fraud vs Genuine transactions")
fig.show()
```



In [6]:
```python
plt.figure(figsize=(3,4))
ax = sns.countplot(x='is_fraud',data=train_df,palette="pastel")
for i in ax.containers:
    ax.bar_label(i,)
```



In [7]:
```
print('Genuine:', round(train_df['is_fraud'].value_counts()[0]/len(train_df) * 100,2), '% of the dataset')
print('Frauds:', round(train_df['is_fraud'].value_counts()[1]/len(train_df) * 100,2), '% of the dataset')
```

```
Genuine: 99.42 % of the dataset
Frauds: 0.58 % of the dataset
```

In [8]:
```
train_df.info(),test_df.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[8]:
```
(None, None)
```

In [9]:
```
train_df.isnull().sum(),test_df.isnull().sum()
```

Out[9]:
```
(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,
 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 [10]:
```
drop_columns = ['Unnamed: 0','cc_num','merchant','trans_num','unix_time','first','last','street','zip']
train_df.drop(columns=drop_columns,inplace=True)
test_df.drop(columns=drop_columns,inplace=True)
```

In [11]:
```
print(train_df.shape)
print(test_df.shape)
```

```
(1296675, 14)
(555719, 14)
```

In [12]:
```
train_df['trans_date_trans_time']=pd.to_datetime(train_df['trans_date_trans_time'])
train_df['trans_date']=train_df['trans_date_trans_time'].dt.strftime('%Y-%m-%d')
train_df['trans_date']=pd.to_datetime(train_df['trans_date'])
train_df['dob']=pd.to_datetime(train_df['dob'])
```

In [13]:
```
test_df['trans_date_trans_time']=pd.to_datetime(test_df['trans_date_trans_time'])
test_df['trans_date']=test_df['trans_date_trans_time'].dt.strftime('%Y-%m-%d')
test_df['trans_date']=pd.to_datetime(test_df['trans_date'])
test_df['dob']=pd.to_datetime(test_df['dob'])
```

In [14]:
```
train_df["age"] = train_df["trans_date"]-train_df["dob"]
train_df["age"]=train_df["age"].astype('timedelta64[Y]')
```

In [15]:
```
test_df["age"] = test_df["trans_date"]-test_df["dob"]
test_df["age"]=test_df["age"].astype('timedelta64[Y]')
```

In [16]:
```
train_df['trans_month'] = pd.DatetimeIndex(train_df['trans_date']).month
train_df['trans_year'] = pd.DatetimeIndex(train_df['trans_date']).year
```

In [17]:
```
train_df['latitudinal_distance'] = abs(round(train_df['merch_lat']-train_df['lat'],3))
train_df['longitudinal_distance'] = abs(round(train_df['merch_long']-train_df['long'],3))
```

In [18]:
```
test_df['latitudinal_distance'] = abs(round(test_df['merch_lat']-test_df['lat'],3))
test_df['longitudinal_distance'] = abs(round(test_df['merch_long']-test_df['long'],3))
```

In [19]:
```
drop_columns = ['trans_date_trans_time','city','lat','long','job','dob','merch_lat','merch_long','trans_date','state']
train_df.drop(columns=drop_columns,inplace=True)
test_df.drop(columns=drop_columns,inplace=True)
```

In [20]:
```
train_df.gender=train_df.gender.apply(lambda x: 1 if x=="M" else 0)
test_df.gender=test_df.gender.apply(lambda x: 1 if x=="M" else 0)
```

In [21]:
```
train_df = pd.get_dummies(train_df, columns=['category'], prefix='category')
test_df = pd.get_dummies(test_df, columns=['category'], prefix='category')
```

```
test_df = test_df.reindex(columns=train_df.columns, fill_value=0)
```

In [22]:
```
train_df.head()
```

Out[22]:

	amt	gender	city_pop	is_fraud	age	trans_month	trans_year	latitudinal_distance	longitudinal_distance	category_entertainment	...	category_grocery_p
0	4.97	0	3495	0	30.0	1	2019	0.068	0.870	0	...	
1	107.23	0	149	0	40.0	1	2019	0.271	0.024	0	...	
2	220.11	1	4154	0	56.0	1	2019	0.970	0.108	1	...	
3	45.00	1	1939	0	51.0	1	2019	0.804	0.447	0	...	
4	41.96	1	99	0	32.0	1	2019	0.254	0.830	0	...	

5 rows × 23 columns

In [23]:
```
test_df.head()
```

Out[23]:

	amt	gender	city_pop	is_fraud	age	trans_month	trans_year	latitudinal_distance	longitudinal_distance	category_entertainment	...	category_grocery_p
0	2.86	1	333497	0	52.0	0	0	0.020	0.265	0	...	
1	29.84	0	302	0	30.0	0	0	0.870	0.476	0	...	
2	41.28	0	34496	0	49.0	0	0	0.177	0.660	0	...	
3	60.05	1	54767	0	32.0	0	0	0.243	0.064	0	...	
4	3.19	1	1126	0	64.0	0	0	0.706	0.868	0	...	

5 rows × 23 columns

In [24]:
```
X_train = train_df.drop('is_fraud', axis=1)
y_train = train_df['is_fraud']
X_test = test_df.drop('is_fraud', axis=1)
y_test = test_df['is_fraud']
```

In [25]:
```
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X_train, y_train = smote.fit_resample(X_train, y_train)
```

In [26]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [27]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
```

In [28]:
```
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
```

Out[28]:
```
DecisionTreeClassifier(random_state=42)
```

In [29]:
```
y_pred = clf.predict(X_test)
```

In [30]:
```
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00    553574
           1       0.33      0.72      0.45      2145

    accuracy                           0.99    555719
   macro avg       0.67      0.86      0.73    555719
weighted avg       1.00      0.99      0.99    555719
```

In [31]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [32]:
```
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```

Out[32]:
```
RandomForestClassifier(random_state=42)
```

In [33]:
```
y_pred = clf.predict(X_test)
```

In [34]:
```
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    553574
           1       0.56      0.79      0.65      2145

    accuracy                           1.00    555719
   macro avg       0.78      0.89      0.83    555719
weighted avg       1.00      1.00      1.00    555719
```

In [35]:
```
import xgboost as xgb
```

In [36]:
```
clf = xgb.XGBClassifier(
    learning_rate=0.1,
    n_estimators=100,
    max_depth=3,
    objective='binary:logistic',
    random_state=42
)
```

In [37]:
```
clf.fit(X_train, y_train)
```

Out[37]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=3, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...)
```

In [38]:
```
y_pred = clf.predict(X_test)
```

In [39]:
```
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    553574
           1       0.16      0.85      0.27      2145

    accuracy                           0.98    555719
   macro avg       0.58      0.92      0.63    555719
weighted avg       1.00      0.98      0.99    555719
```

In [ ]: