```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
import nltk
import re
from nltk.stem.porter import PorterStemmer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
df = pd.read_csv('/content/drive/My Drive/Subway/fraudTrain.csv')
```

```
df.columns
```

```
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')
```

```
df.shape
```

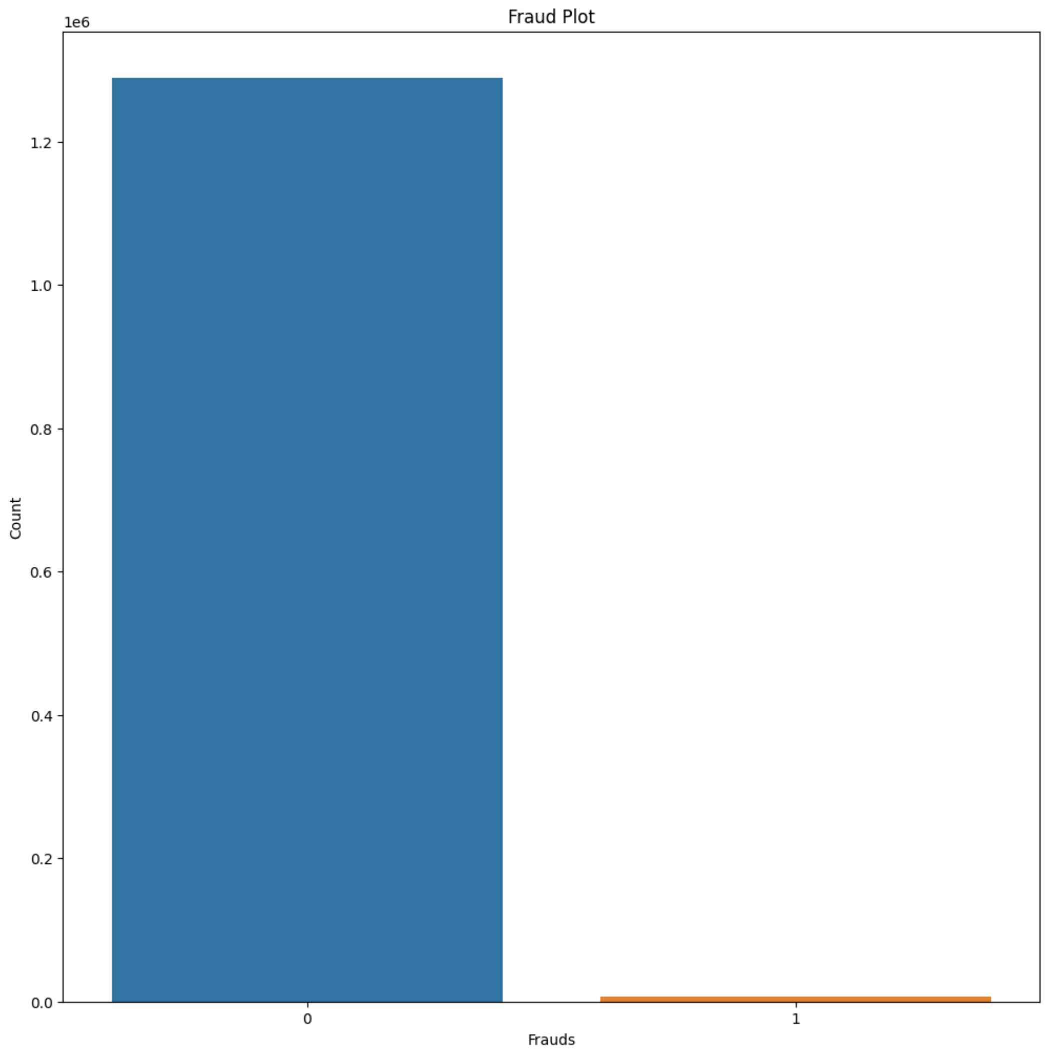
```
(1296675, 23)
```

```
df.head(5)
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	l
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Ba
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanc
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	W
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Ga

5 rows × 23 columns

```
plt.figure(figsize=(12,12))
sns.countplot(x='is_fraud', data=df)
plt.xlabel('Frauds')
plt.ylabel('Count')
plt.title('Fraud Plot')
plt.show()
```



```
frauds = list(df['is_fraud'].unique())
frauds.sort()
frauds
```

```
[0, 1]
```

```
df.isna().any()
```

```
Unnamed: 0               False
trans_date_trans_time    False
cc_num                   False
merchant                 False
category                 False
amt                      False
first                    False
last                     False
gender                   False
street                   False
city                     False
state                    False
zip                      False
lat                      False
long                     False
city_pop                 False
job                      False
dob                      False
trans_num                False
unix_time                False
merch_lat                False
merch_long               False
is_fraud                 False
dtype: bool
```

```
df.drop('Unnamed: 0', axis=1, inplace=True)
```

```
import pandas as pd
from sklearn.preprocessing import OrdinalEncoder
from sklearn.naive_bayes import CategoricalNB
from sklearn import metrics

data = df
le = OrdinalEncoder()
data[['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']] = le.fit_transform(
    data[['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']])

Features = ['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']
X = data[Features]
Y = data['is_fraud']

clf_nb = CategoricalNB()
clf_nb.fit(X,Y)

# y_pred = clf.predict(X)
# print(Y)
# print(y_pred)

# print("Accuracy : " , metrics.accuracy_score(y_pred,Y) * 100)
```

```
▾ CategoricalNB
CategoricalNB()
```

```
clf_lr = LogisticRegression(max_iter=1000)
clf_lr.fit(X, Y)

# y_pred = clf_lr.predict(X)
# print(Y)
# print(y_pred)

# print("Accuracy : ", metrics.accuracy_score(y_pred, Y) * 100)
```

```
▾        LogisticRegression
LogisticRegression(max_iter=1000)
```

```
from sklearn.svm import SVC


clf_svm = SVC(kernel='linear')
clf_svm.fit(X, Y)
```

```
y_pred = clf_nb.predict(X)
print(Y)
print(y_pred)

print("Accuracy : " , metrics.accuracy_score(y_pred,Y) * 100)
```

```
0          0.0
1          0.0
2          0.0
3          0.0
4          0.0
          ...
1296670    0.0
1296671    0.0
1296672    0.0
1296673    0.0
1296674    0.0
Name: is_fraud, Length: 1296675, dtype: float64
[0. 0. 0. ... 0. 0. 0.]
Accuracy :  92.29274876125476
```

```
y_pred = clf_lr.predict(X)
print(Y)
print(y_pred)

print("Accuracy : " , metrics.accuracy_score(y_pred,Y) * 100)
```

```
0          0.0
1          0.0
2          0.0
3          0.0
4          0.0
          ...
1296670    0.0
1296671    0.0
1296672    0.0
1296673    0.0
1296674    0.0
Name: is_fraud, Length: 1296675, dtype: float64
[0. 0. 0. ... 0. 0. 0.]
Accuracy :  99.39792160718761
```

```
y_pred = clf_svm.predict(X)
print(Y)
print(y_pred)

print("Accuracy : " , metrics.accuracy_score(y_pred,Y) * 100)
```

```
tt = pd.read_csv('/content/drive/My Drive/Subway/fraudTest.csv')

tt[['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']] = le.fit_transform(
    tt[['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']])

XX= tt[Features]
YY = tt['is_fraud']
```

```
yy_pred = clf_nb.predict(XX)
print(YY)
print(yy_pred)

print("Accuracy : " , metrics.accuracy_score(yy_pred,YY) * 100)
```

```
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0
         ...
555714    0.0
555715    0.0
555716    0.0
555717    0.0
555718    0.0
Name: is_fraud, Length: 555719, dtype: float64
[0. 0. 0. ... 0. 0. 0.]
Accuracy :  72.92966409282388
```

```
yy_pred = clf_lr.predict(XX)
print(YY)
print(yy_pred)

print("Accuracy : " , metrics.accuracy_score(yy_pred,YY) * 100)
```

```
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0
         ...
555714    0.0
555715    0.0
555716    0.0
555717    0.0
555718    0.0
Name: is_fraud, Length: 555719, dtype: float64
[0. 0. 0. ... 0. 0. 0.]
Accuracy :  99.4983075979047
```

```
yy_pred = clf_svm.predict(XX)
print(YY)
print(yy_pred)
print("Accuracy : " , metrics.accuracy_score(yy_pred,YY) * 100)
```

26m 40s completed at 17:14