Credit Card Farud Detection Using Logistic Regression

Aurthor :Irfan Ullah Khan

GITHUB PROFILE KAGGLE PROFILE LINKEDIN PROFILE

YOUTUBE PROFILE EMAIL CONTACT ME

Importing Necessary Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

```
df=pd.read_csv("/content/fraudTest.csv")
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50828 entries, 0 to 50827
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Unnamed: 0             50828 non-null  int64  
 1   trans_date_trans_time  50828 non-null  float64
 2   cc_num                 50828 non-null  float64
 3   merchant               50828 non-null  object 
 4   category               50828 non-null  object 
 5   amt                    50828 non-null  float64
 6   first                  50828 non-null  object 
 7   last                   50828 non-null  object 
 8   gender                 50828 non-null  object 
 9   street                 50828 non-null  object 
 10  city                   50828 non-null  object 
 11  state                  50827 non-null  object 
 12  zip                    50827 non-null  float64
 13  lat                    50827 non-null  float64
 14  long                   50827 non-null  float64
 15  city_pop               50827 non-null  float64
 16  job                    50827 non-null  object 
 17  dob                    50827 non-null  object 
 18  trans_num              50827 non-null  object 
 19  unix_time              50827 non-null  float64
 20  merch_lat              50827 non-null  float64
 21  merch_long             50827 non-null  float64
 22  is_fraud               50827 non-null  float64
dtypes: float64(11), int64(1), object(11)
memory usage: 8.9+ MB
```

```
df.describe()
```

	Unnamed: 0	trans_date_trans_time	cc_num	amt	zip	lat	
count	50828.000000	50828.000000	5.082800e+04	50828.000000	50827.000000	50827.000000	50827
mean	25413.500000	44011.977918	4.156839e+17	69.463839	48700.820351	38.529231	-9(
std	14672.924078	5.115022	1.306796e+18	151.240739	26784.263353	5.082601	1:
min	0.000000	44003.510010	6.041621e+10	1.000000	1257.000000	20.027100	-16!

```
df.head()
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last
0	0	44003.51001	2.290000e+15	fraud_Kirlin and Sons	personal_care	2.86	Jeff	Elliott
1	1	44003.51010	3.570000e+15	fraud_Sporer-Keebler	personal_care	29.84	Joanne	Williams
2	2	44003.51034	3.600000e+15	fraud_Swaniawski, Nitzsche and Welch	health_fitness	41.28	Ashley	Lopez
3	3	44003.51059	3.590000e+15	fraud_Haley Group	misc_pos	60.05	Brian	Williams
4	4	44003.51061	3.530000e+15	fraud_Johnston-Casper	travel	3.19	Nathan	Massey

5 rows × 23 columns

```
df.tail()
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last
50823	50823	44020.52016	5.820000e+11	fraud_Pollich LLC	home	153.71	Larry	House
50824	50824	44020.52041	3.550000e+15	fraud_Jast and Sons	food_dining	99.68	Kayla	Obrien
50825	50825	44020.52061	2.710000e+15	fraud_Towne, Greenholt and Koepp	shopping_net	10.29	Jenna	Brooks
50826	50826	44020.52139	2.250000e+15	fraud_Berge, Kautzer and Harris	personal_care	17.76	Margaret	Gibson
50827	50827	44020.52240	4.860000e+12	fraud_Pollich LLC	home	17.98	Elizabeth	Payne

5 rows × 23 columns

```
df
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first
0	0	44003.51001	2.290000e+15	fraud_Kirlin and Sons	personal_care	2.86	Jeff
1	1	44003.51010	3.570000e+15	fraud_Sporer-Keebler	personal_care	29.84	Joanne
2	2	44003.51034	3.600000e+15	fraud_Swaniawski, Nitzsche and Welch	health_fitness	41.28	Ashley
3	3	44003.51059	3.590000e+15	fraud_Haley Group	misc_pos	60.05	Brian
4	4	44003.51061	3.530000e+15	fraud_Johnston-Casper	travel	3.19	Nathan
...	...	...	...	...	...	...	...
50823	50823	44020.52016	5.820000e+11	fraud_Pollich LLC	home	153.71	Larry
50824	50824	44020.52041	3.550000e+15	fraud_Jast and Sons	food_dining	99.68	Kayla
50825	50825	44020.52061	2.710000e+15	fraud_Towne, Greenholt and Koepp	shopping_net	10.29	Jenna
50826	50826	44020.52139	2.250000e+15	fraud_Berge, Kautzer and Harris	personal_care	17.76	Margaret
50827	50827	44020.52240	4.860000e+12	fraud_Pollich LLC	home	17.98	Elizabeth

50828 rows × 23 columns

```
df.shape
```

```
(50828, 23)
```

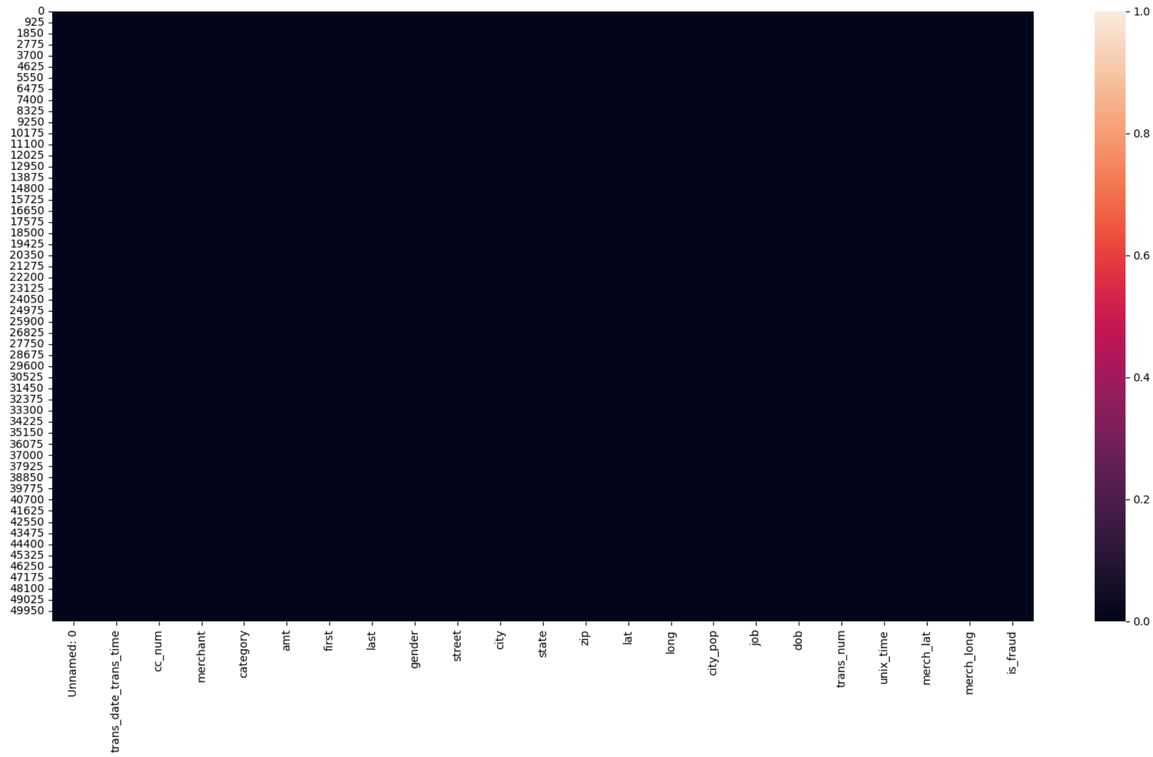
```
#Checking for null values through function and heatmap
df.isnull().any()
```

```
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                     True
zip                       True
lat                       True
long                      True
city_pop                  True
```

```
job                       True
dob                       True
trans_num                 True
unix_time                 True
merch_lat                 True
merch_long                True
is_fraud                  True
dtype: bool
```

```
plt.figure(figsize = (20,10))
sns.heatmap(df.isnull())
```

```
<Axes: >
```



```
df.dtypes
```

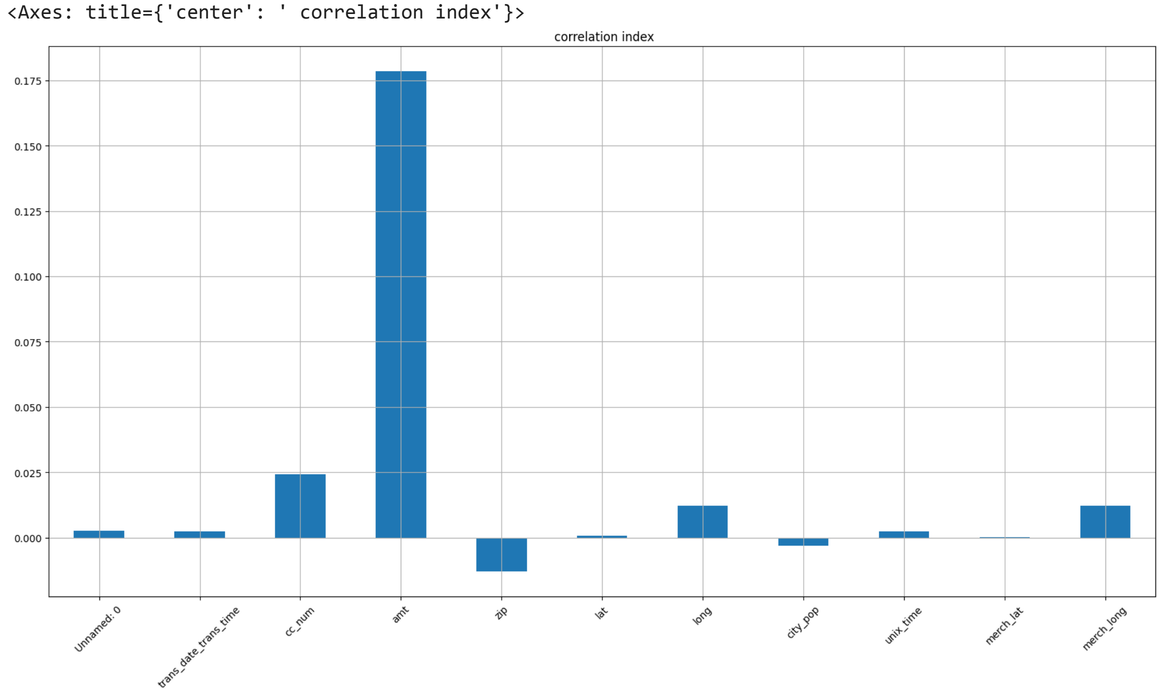
```
Unnamed: 0                 int64
trans_date_trans_time    float64
cc_num                   float64
merchant                  object
category                  object
```

```
amt                        float64
first                       object
last                        object
gender                      object
street                      object
city                        object
state                       object
zip                        float64
lat                        float64
long                       float64
city_pop                   float64
job                         object
dob                         object
trans_num                   object
unix_time                  float64
merch_lat                  float64
merch_long                 float64
is_fraud                   float64
dtype: object
```

```
dataset_2 = df.drop(columns = 'is_fraud')
```

```
dataset_2.corrwith(df['is_fraud']).plot.bar(figsize = (20,10), title = ' correlation index', rot = 45, grid = True)
```

```
<Axes: title={'center': ' correlation index'}>
```



```
corr =  df.corr()
plt.figure(figsize = (35,15))
sns.heatmap(corr, annot = True, cmap = 'coolwarm', linewidth = 2)
```

    <Axes: >

```
#findind the features with high correlation
high_corr = df.corr()
high_corr_features = high_corr.index[abs(high_corr['is_fraud'] > 0.5)]
```

```
high_corr_features
```

    Index(['is_fraud'], dtype='object')

```
x = df.drop(columns = 'is_fraud')
```

```
y = df['is_fraud']
```

```
x
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	
0	0	44003.51001	2.290000e+15	fraud_Kirlin and Sons	personal_care	2.86	Jeff	
1	1	44003.51010	3.570000e+15	fraud_Sporer-Keebler	personal_care	29.84	Joanne	W
2	2	44003.51034	3.600000e+15	fraud_Swaniawski, Nitzsche and Welch	health_fitness	41.28	Ashley	
3	3	44003.51059	3.590000e+15	fraud_Haley Group	misc_pos	60.05	Brian	W
4	4	44003.51061	3.530000e+15	fraud_Johnston-Casper	travel	3.19	Nathan	M
...	...	...	...	...	...	...	...	
50823	50823	44020.52016	5.820000e+11	fraud_Pollich LLC	home	153.71	Larry	
50824	50824	44020.52041	3.550000e+15	fraud_Jast and Sons	food_dining	99.68	Kayla	
50825	50825	44020.52061	2.710000e+15	fraud_Towne, Greenholt and Koepp	shopping_net	10.29	Jenna	
50826	50826	44020.52139	2.250000e+15	fraud_Berge, Kautzer and Harris	personal_care	17.76	Margaret	
50827	50827	44020.52240	4.860000e+12	fraud_Pollich LLC	home	17.98	Elizabeth	

50828 rows × 22 columns

```
y
```

```
0        0.0
1        0.0
2        0.0
3        0.0
4        0.0
        ...
50823    0.0
50824    0.0
50825    0.0
50826    0.0
50827    NaN
Name: is_fraud, Length: 50828, dtype: float64
```

```
df.groupby('is_fraud').mean()
```

is_fraud	Unnamed: 0	trans_date_trans_time	cc_num	amt	zip	lat	lo
0.0	25410.566530	44011.976974	4.136740e+17	67.756165	48722.974913	38.529015	-90.1396
1.0	26019.857143	44012.171027	9.189346e+17	495.575911	43175.931034	38.583064	-87.5121

```
x=df.drop('is_fraud',axis=1)
```

```
y=df['is_fraud']
```

```
y.shape
```

```
(50828,)
```

```
x.shape
```

```
(50828, 22)
```

```
missing_values = y.isnull().sum()
if missing_values.any():
    print("There are missing values in 'y'.")
```

```
There are missing values in 'y'.
```

```
# Option 2: Impute missing values
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="mean")
y = imputer.fit_transform(y.to_frame().values)
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y, random_state=42)
```

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
x_train = x_train.drop('trans_date_trans_time', axis=1)
```

```
print(x_train.head())
```

```
       Unnamed: 0        cc_num     amt      zip      lat     long  city_pop  \
9216         9216  6.010000e+15   64.76  23937.0  36.9688 -78.5615    1970.0
42868       42868  2.230000e+15   75.18  39769.0  33.3570 -89.0473    1923.0
40853       40853  3.590000e+15    8.36  21750.0  39.6991 -78.1762    3766.0
22311       22311  6.010000e+15   57.48  73754.0  36.3850 -98.0727    1078.0
4832         4832  3.030000e+13  133.29  72165.0  35.5762 -91.4539     111.0

            dob                         trans_num     unix_time  merch_lat  \
9216   02-09-85  205b896137189f145f86261f6b00e75d  1.372018e+09  37.300259
42868  16-01-60  68661acb2f7b6bf0acad21f563cf4a01  1.373091e+09  33.787372
40853  14-02-84  bd0445c0bf554893fd3ccd1adfd530dc  1.373044e+09  40.087332
22311  06-07-52  81b20a651bb36ef406895e74245fa110  1.372471e+09  36.388727
4832   13-06-00  2ffeb77b10ee6744f85a6602a471faaf  1.371924e+09  35.665908

       merch_long
9216   -78.493307
42868  -88.930328
40853  -77.672439
22311  -97.967321
4832   -90.630391
```

```
print(x_train.dtypes)
```

```
Unnamed: 0      int64
cc_num        float64
amt           float64
zip           float64
lat           float64
long          float64
city_pop      float64
dob            object
trans_num      object
unix_time     float64
merch_lat     float64
merch_long    float64
dtype: object
```

```
x_train = x_train.drop('trans_num', axis=1)
```

```
print(x_train.columns)
print(x_test.columns)
```

```
Index(['Unnamed: 0', 'cc_num', 'amt', 'zip', 'lat', 'long', 'city_pop',
       'unix_time', 'merch_lat', 'merch_long'],
      dtype='object')
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'],
      dtype='object')
```

```
missing_cols = set(x_train.columns) - set(x_test.columns)
if missing_cols:
    raise ValueError(f"Missing columns in x_test: {missing_cols}")
```

```
x_test.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'],
      dtype='object')
```

```
print("Columns in x_train:", x_train.columns)
print("Columns in x_test:", x_test.columns)
```

```
Columns in x_train: Index(['Unnamed: 0', 'cc_num', 'amt', 'zip', 'lat', 'long', 'city_pop',
       'unix_time', 'merch_lat', 'merch_long'],
      dtype='object')
Columns in x_test: 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'],
      dtype='object')
```

```
# Check and align columns in x_train and x_test
common_columns = set(x_train.columns) & set(x_test.columns)
x_train = x_train[common_columns]
x_test = x_test[common_columns]

# Now, fit and predict
model = LogisticRegression()
model.fit(x_train, y_train)
ypred = model.predict(x_test)
```