Abdullah Shumail

22108165

Fraud Detection model using Logistic Regression.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv(r"C:\Users\lenovo\Downloads\creditcard (1).csv")
df.head()
```

Out[1]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278

5 rows × 31 columns

In [2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

In [3]:
```
df.describe().T
```

Out[3]:

	count	mean	std	min	25%	50%	75%	max
Time	284807.0	9.481386e+04	47488.145955	0.000000	54201.500000	84692.000000	139320.500000	172792.000000
V1	284807.0	1.168375e-15	1.958696	-56.407510	-0.920373	0.018109	1.315642	2.454930
V2	284807.0	3.416908e-16	1.651309	-72.715728	-0.598550	0.065486	0.803724	22.057729
V3	284807.0	-1.379537e-15	1.516255	-48.325589	-0.890365	0.179846	1.027196	9.382558
V4	284807.0	2.074095e-15	1.415869	-5.683171	-0.848640	-0.019847	0.743341	16.875344
V5	284807.0	9.604066e-16	1.380247	-113.743307	-0.691597	-0.054336	0.611926	34.801666
V6	284807.0	1.487313e-15	1.332271	-26.160506	-0.768296	-0.274187	0.398565	73.301626
V7	284807.0	-5.556467e-16	1.237094	-43.557242	-0.554076	0.040103	0.570436	120.589494
V8	284807.0	1.213481e-16	1.194353	-73.216718	-0.208630	0.022358	0.327346	20.007208
V9	284807.0	-2.406331e-15	1.098632	-13.434066	-0.643098	-0.051429	0.597139	15.594995
V10	284807.0	2.239053e-15	1.088850	-24.588262	-0.535426	-0.092917	0.453923	23.745136
V11	284807.0	1.673327e-15	1.020713	-4.797473	-0.762494	-0.032757	0.739593	12.018913
V12	284807.0	-1.247012e-15	0.999201	-18.683715	-0.405571	0.140033	0.618238	7.848392
V13	284807.0	8.190001e-16	0.995274	-5.791881	-0.648539	-0.013568	0.662505	7.126883
V14	284807.0	1.207294e-15	0.958596	-19.214325	-0.425574	0.050601	0.493150	10.526766
V15	284807.0	4.887456e-15	0.915316	-4.498945	-0.582884	0.048072	0.648821	8.877742
V16	284807.0	1.437716e-15	0.876253	-14.129855	-0.468037	0.066413	0.523296	17.315112
V17	284807.0	-3.772171e-16	0.849337	-25.162799	-0.483748	-0.065676	0.399675	9.253526
V18	284807.0	9.564149e-16	0.838176	-9.498746	-0.498850	-0.003636	0.500807	5.041069
V19	284807.0	1.039917e-15	0.814041	-7.213527	-0.456299	0.003735	0.458949	5.591971
V20	284807.0	6.406204e-16	0.770925	-54.497720	-0.211721	-0.062481	0.133041	39.420904
V21	284807.0	1.654067e-16	0.734524	-34.830382	-0.228395	-0.029450	0.186377	27.202839
V22	284807.0	-3.568593e-16	0.725702	-10.933144	-0.542350	0.006782	0.528554	10.503090
V23	284807.0	2.578648e-16	0.624460	-44.807735	-0.161846	-0.011193	0.147642	22.528412
V24	284807.0	4.473266e-15	0.605647	-2.836627	-0.354586	0.040976	0.439527	4.584549
V25	284807.0	5.340915e-16	0.521278	-10.295397	-0.317145	0.016594	0.350716	7.519589
V26	284807.0	1.683437e-15	0.482227	-2.604551	-0.326984	-0.052139	0.240952	3.517346
V27	284807.0	-3.660091e-16	0.403632	-22.565679	-0.070840	0.001342	0.091045	31.612198
V28	284807.0	-1.227390e-16	0.330083	-15.430084	-0.052960	0.011244	0.078280	33.847808
Amount	284807.0	8.834962e+01	250.120109	0.000000	5.600000	22.000000	77.165000	25691.160000
Class	284807.0	1.727486e-03	0.041527	0.000000	0.000000	0.000000	0.000000	1.000000

In [4]:
```
df.shape
```

Out[4]:
```
(284807, 31)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

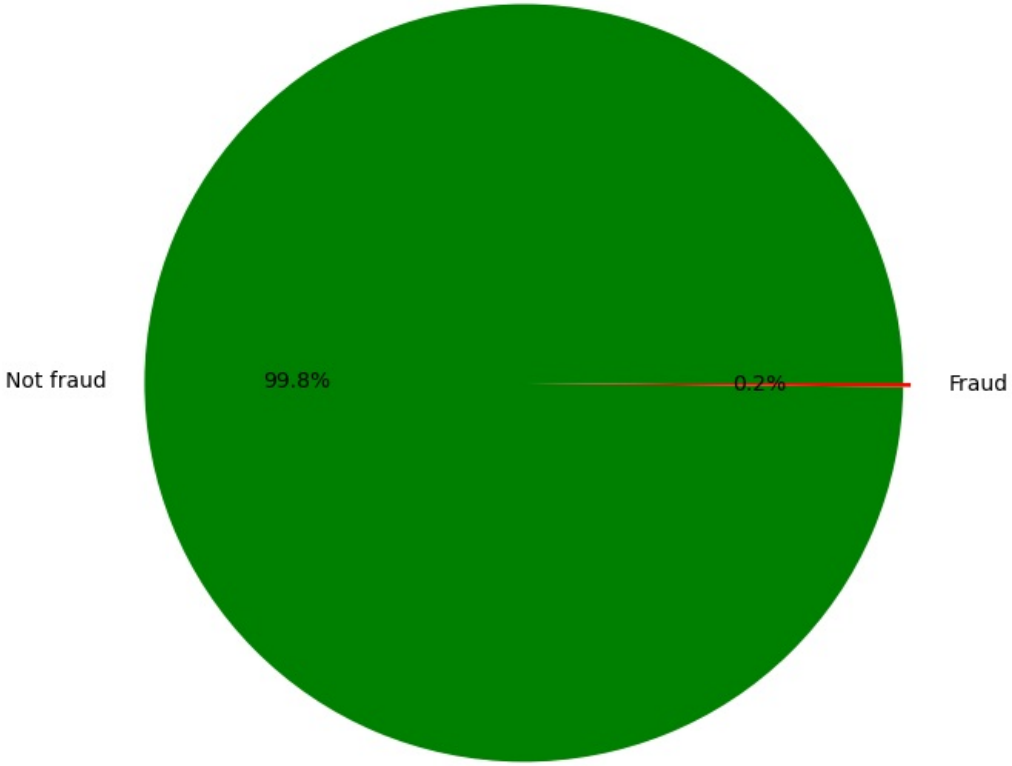
In [7]:
```
df.duplicated().sum()
```

Out[7]:
```
1081
```

In [8]:
```
df.drop_duplicates(inplace=True)
df['Class'].value_counts()
```

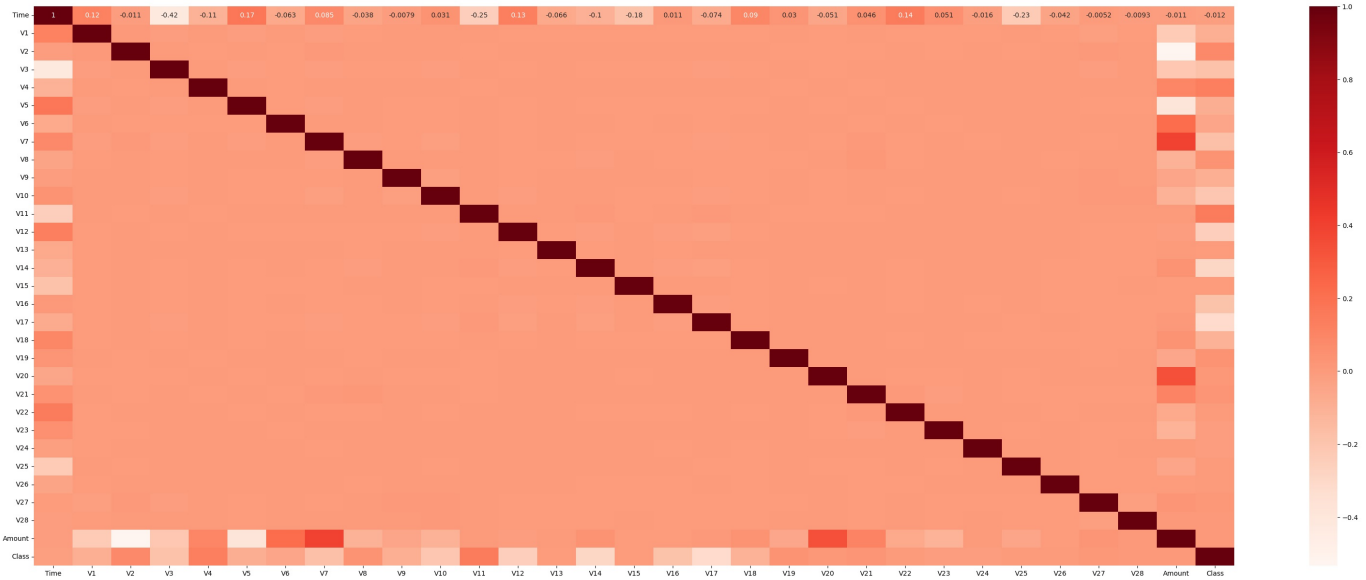
Out[8]:
```
Class
0    283253
1       473
Name: count, dtype: int64
```

In [9]:
```
plt.figure(figsize=(10,8))
labels = ['Not fraud','Fraud']
explode=[.01,.01]
color= 'green','red'
sizes= df.Class.value_counts().values
plt.pie(sizes,explode,labels,autopct='%1.1f%%',colors=color)
plt.show()
```



In [10]:
```
plt.figure(figsize=(40,15))
sns.heatmap(df.corr(),annot=True,cmap='Reds')
```

Out[10]: <Axes: >



In [11]:
```
x = df.iloc[:,0:-1]
y = df.iloc[:,-1]
x.shape
```

Out[11]: (283726, 30)

In [12]:
```
y.shape
```

Out[12]: (283726,)

In [13]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size= 0.75,random_state=101)
```

In [14]:
```
x_train.shape
```

Out[14]: (212794, 30)

In [15]:
```
y_train.shape
```

Out[15]:
```
(212794,)
```

In [16]:
```
x_test.shape
```

Out[16]:
```
(70932, 30)
```

In [17]:
```
y_test.shape
```

Out[17]:
```
(70932,)
```

In [18]:
```
from sklearn.linear_model import LogisticRegression
logit = LogisticRegression()
logit.fit(x_train,y_train)
```

```
C:\Users\lenovo\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs fai
led to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[18]:
```
▾ LogisticRegression
LogisticRegression()
```

In [19]:
```
y_predict_train=logit.predict(x_train)
y_predict_test=logit.predict(x_test)
```

In [20]:
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
```

In [21]:
```
cm_tr=confusion_matrix(y_train,y_predict_train)
cm_tr
```

Out[21]:
```
array([[212365,     72],
       [   144,    213]], dtype=int64)
```

In [22]:
```
cm_tst=confusion_matrix(y_test,y_predict_test)
cm_tst
```

Out[22]:
```
array([[70793,    23],
       [   39,    77]], dtype=int64)
```

In [23]:
```
cl_rep_train = classification_report(y_train,y_predict_train)
cl_rep_train
```

Out[23]:
```
'              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00    212437
\n           1       0.75      0.60      0.66       357\n\n    accuracy                           1.00    21279
4\n   macro avg       0.87      0.80      0.83    212794\nweighted avg       1.00      1.00      1.00    212794
\n'
```

In [24]:
```
print(f'classification report: {cl_rep_train}')
```

```
classification report:               precision    recall  f1-score   support

           0       1.00      1.00      1.00    212437
           1       0.75      0.60      0.66       357

    accuracy                           1.00    212794
   macro avg       0.87      0.80      0.83    212794
weighted avg       1.00      1.00      1.00    212794
```

In [25]:
```
cl_rep_tst= classification_report(y_test,y_predict_test)
cl_rep_tst
```

Out[25]:
```
'              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00     70816
\n           1       0.77      0.66      0.71       116\n\n    accuracy                           1.00     7093
2\n   macro avg       0.88      0.83      0.86     70932\nweighted avg       1.00      1.00      1.00     70932
\n'
```

In [26]:
```
print(f'classification report: {cl_rep_tst}')
```

```
classification report:               precision    recall  f1-score   support

           0       1.00      1.00      1.00     70816
           1       0.77      0.66      0.71       116

    accuracy                           1.00     70932
   macro avg       0.88      0.83      0.86     70932
weighted avg       1.00      1.00      1.00     70932
```

In [27]:
```
ac_train = accuracy_score(y_train,y_predict_train)
ac_train_per = ac_train*100
print(f'The accuracy for training data: {ac_train_per}')
```

```
The accuracy for training data: 99.89849337857271
```

In [28]:
```
ac_tst = accuracy_score(y_test,y_predict_test)
ac_tst_per = ac_tst*100
print(f'The accuracy for testing data: {ac_tst_per}')
```

```
The accuracy for testing data: 99.91259234196131
```

In [30]:
```
!pip install joblib
import joblib
```

```
Requirement already satisfied: joblib in c:\users\lenovo\anaconda3\lib\site-packages (1.2.0)
```

In [31]:
```
joblib.dump(logit, 'credit_card_fraud_detection_model.pkl')
```

Out[31]:
```
['credit_card_fraud_detection_model.pkl']
```

In [32]:
```
loaded_model = joblib.load('credit_card_fraud_detection_model.pkl')
```

In [33]:
```
predictions = loaded_model.predict(x_test)
print(predictions)
```

```
[0 0 0 ... 0 0 0]
```

In [34]:
```
import os

print(os.listdir())
```

```
['.anaconda', '.conda', '.condarc', '.continuum', '.ipynb_checkpoints', '.ipython', '.jupyter', '.matplotlib', '
.packettracer', '.vscode', '22108165ANN1B.ipynb', '3D Objects', 'anaconda3', 'AppData', 'Application Data', 'Cis
co Packet Tracer 8.2.2', 'Contacts', 'Cookies', 'credit_card_fraud_detection_model.pkl', 'dataset.csv', 'dataset
.txt', 'Desktop', 'Dev-C++.lnk', 'Documents', 'Downloads', 'Favorites', 'gawkk.txt', 'IntelGraphicsProfiles', 'L
inks', 'Local Settings', 'mobile_phone_price_data.csv', 'Music', 'My Documents', 'NetHood', 'NTUSER.DAT', 'ntuse
r.dat.LOG1', 'ntuser.dat.LOG2', 'NTUSER.DAT{a6ff422c-8561-11eb-93f8-89d69fd718b3}.TM.blf', 'NTUSER.DAT{a6ff422c-
8561-11eb-93f8-89d69fd718b3}.TMContainer00000000000000000001.regtrans-ms', 'NTUSER.DAT{a6ff422c-8561-11eb-93f8-8
9d69fd718b3}.TMContainer00000000000000000002.regtrans-ms', 'ntuser.ini', 'OneDrive', 'Personal Vault.lnk', 'Pict
ures', 'PrintHood', 'Recent', 'Saved Games', 'scikit_learn_data', 'Searches', 'SendTo', 'sendu', 'Start Menu', '
Templates', 'Untitled.ipynb', 'untitled.txt', 'Untitled1.ipynb', 'untitled1.txt', 'Untitled10.ipynb', 'Untitled1
1.ipynb', 'Untitled12.ipynb', 'Untitled13.ipynb', 'Untitled14.ipynb', 'Untitled15.ipynb', 'Untitled16.ipynb', 'U
ntitled17.ipynb', 'Untitled18.ipynb', 'Untitled19.ipynb', 'Untitled2.ipynb', 'Untitled20.ipynb', 'Untitled21.ipy
nb', 'Untitled22.ipynb', 'Untitled23.ipynb', 'Untitled24.ipynb', 'Untitled25.ipynb', 'Untitled26.ipynb', 'Untitl
ed27.ipynb', 'Untitled28.ipynb', 'Untitled3.ipynb', 'Untitled4.ipynb', 'Untitled5.ipynb', 'Untitled6.ipynb', 'Un
titled7.ipynb', 'Untitled8.ipynb', 'Untitled9.ipynb', 'Videos', 'yele']
```

In [36]:
```
from IPython.display import FileLink

FileLink('credit_card_fraud_detection_model.pkl')
```

Out[36]: credit_card_fraud_detection_model.pkl

In [ ]: