FRAUD DETECTION IN FINANCIAL TRANSACTION

July 20, 2024

[1]:
```
# The necessary libraries for data manipulation,machine learning are imported.
#These typically include pandas, sklearn, numpy.
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

[2]:
```
# Load the dataset
dataset = pd.read_csv("creditcard.csv")
```

[3]:
```
# Display the first few rows of the dataset
dataset.head()
```

[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	

	V8	V9	…	V21	V22	V23	V24	V25	\
0	0.098698	0.363787	…	-0.018307	0.277838	-0.110474	0.066928	0.128539	
1	0.085102	-0.255425	…	-0.225775	-0.638672	0.101288	-0.339846	0.167170	
2	0.247676	-1.514654	…	0.247998	0.771679	0.909412	-0.689281	-0.327642	
3	0.377436	-1.387024	…	-0.108300	0.005274	-0.190321	-1.175575	0.647376	
4	-0.270533	0.817739	…	-0.009431	0.798278	-0.137458	0.141267	-0.206010	

	V26	V27	V28	Amount	Class
0	-0.189115	0.133558	-0.021053	149.62	0
1	0.125895	-0.008983	0.014724	2.69	0
2	-0.139097	-0.055353	-0.059752	378.66	0
3	-0.221929	0.062723	0.061458	123.50	0
4	0.502292	0.219422	0.215153	69.99	0

[5 rows x 31 columns]

[4]:
```
# Display the last five rows of the dataset.
dataset.tail()
```

[4]:

	Time	V1	V2	V3	V4	V5	\
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	

	V6	V7	V8	V9	…	V21	V22	\
284802	-2.606837	-4.918215	7.305334	1.914428	…	0.213454	0.111864	
284803	1.058415	0.024330	0.294869	0.584800	…	0.214205	0.924384	
284804	3.031260	-0.296827	0.708417	0.432454	…	0.232045	0.578229	
284805	0.623708	-0.686180	0.679145	0.392087	…	0.265245	0.800049	
284806	-0.649617	1.577006	-0.414650	0.486180	…	0.261057	0.643078	

	V23	V24	V25	V26	V27	V28	Amount	\
284802	1.014480	-0.509348	1.436807	0.250034	0.943651	0.823731	0.77	
284803	0.012463	-1.016226	-0.606624	-0.395255	0.068472	-0.053527	24.79	
284804	-0.037501	0.640134	0.265745	-0.087371	0.004455	-0.026561	67.88	
284805	-0.163298	0.123205	-0.569159	0.546668	0.108821	0.104533	10.00	
284806	0.376777	0.008797	-0.473649	-0.818267	-0.002415	0.013649	217.00	

	Class
284802	0
284803	0
284804	0
284805	0
284806	0

[5 rows x 31 columns]

[5]:
```
# Display information about the dataset.
dataset.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
```

[6]:
```
# Check for any missing values in the dataset.
dataset.isnull().values.any()
```

[6]:
```
False
```

[7]:
```
# Distrubution of normal trasactions and fradulent transactions
dataset["Class"].value_counts()
```

[7]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

[8]:
```
# separating the data for analysis
normal = dataset[dataset.Class == 0]
fraud = dataset[dataset.Class == 1]
```

[9]:
```
print(normal)
```

```
            Time         V1         V2        V3        V4        V5  \
0            0.0  -1.359807  -0.072781  2.536347  1.378155 -0.338321
```

```
1            0.0   1.191857   0.266151  0.166480  0.448154  0.060018
2            1.0  -1.358354  -1.340163  1.773209  0.379780 -0.503198
3            1.0  -0.966272  -0.185226  1.792993 -0.863291 -0.010309
4            2.0  -1.158233   0.877737  1.548718  0.403034 -0.407193
…            …         …         …         …         …         …
284802  172786.0 -11.881118  10.071785 -9.834783 -2.066656 -5.364473
284803  172787.0  -0.732789  -0.055080  2.035030 -0.738589  0.868229
284804  172788.0   1.919565  -0.301254 -3.249640 -0.557828  2.630515
284805  172788.0  -0.240440   0.530483  0.702510  0.689799 -0.377961
284806  172792.0  -0.533413  -0.189733  0.703337 -0.506271 -0.012546

                V6        V7        V8        V9  …       V21       V22  \
0         0.462388  0.239599  0.098698  0.363787  … -0.018307  0.277838
1        -0.082361 -0.078803  0.085102 -0.255425  … -0.225775 -0.638672
2         1.800499  0.791461  0.247676 -1.514654  …  0.247998  0.771679
3         1.247203  0.237609  0.377436 -1.387024  … -0.108300  0.005274
4         0.095921  0.592941 -0.270533  0.817739  … -0.009431  0.798278
…              …         …         …         …  …         …         …
284802 -2.606837 -4.918215  7.305334  1.914428  …  0.213454  0.111864
284803  1.058415  0.024330  0.294869  0.584800  …  0.214205  0.924384
284804  3.031260 -0.296827  0.708417  0.432454  …  0.232045  0.578229
284805  0.623708 -0.686180  0.679145  0.392087  …  0.265245  0.800049
284806 -0.649617  1.577006 -0.414650  0.486180  …  0.261057  0.643078

                V23       V24       V25       V26       V27       V28  Amount  \
0         -0.110474  0.066928  0.128539 -0.189115  0.133558 -0.021053  149.62
1          0.101288 -0.339846  0.167170  0.125895 -0.008983  0.014724    2.69
2          0.909412 -0.689281 -0.327642 -0.139097 -0.055353 -0.059752  378.66
3         -0.190321 -1.175575  0.647376 -0.221929  0.062723  0.061458  123.50
4         -0.137458  0.141267 -0.206010  0.502292  0.219422  0.215153   69.99
…               …         …         …         …         …         …       …
284802    1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731    0.77
284803    0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   24.79
284804   -0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88
284805   -0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   10.00
284806    0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649  217.00

        Class
0           0
1           0
2           0
3           0
4           0
…           …
284802      0
284803      0
284804      0
284805      0
```

```
284806      0

[284315 rows x 31 columns]
```

[10]:
```
print(fraud)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
541        406.0 -2.312227  1.951992 -1.609851  3.997906 -0.522188 -1.426545
623        472.0 -3.043541 -3.157307  1.088463  2.288644  1.359805 -1.064823
4920      4462.0 -2.303350  1.759247 -0.359745  2.330243 -0.821628 -0.075788
6108      6986.0 -4.397974  1.358367 -2.592844  2.679787 -1.128131 -1.706536
6329      7519.0  1.234235  3.019740 -4.304597  4.732795  3.624201 -1.357746
…            …       …       …       …       …       …
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494
280143  169347.0  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.326536
280149  169351.0 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.003346
281144  169966.0 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.943548
281674  170348.0  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.096695

               V7        V8        V9  …       V21       V22       V23  \
541    -2.537387  1.391657 -2.770089  …  0.517232 -0.035049 -0.465211
623     0.325574 -0.067794 -0.270953  …  0.661696  0.435477  1.375966
4920    0.562320 -0.399147 -0.238253  … -0.294166 -0.932391  0.172726
6108   -3.496197 -0.248778 -0.247768  …  0.573574  0.176968 -0.436207
6329    1.713445 -0.496358 -1.282858  … -0.379068 -0.704181 -0.656805
…            …       …       …  …       …       …
279863 -0.882850  0.697211 -2.064945  …  0.778584 -0.319189  0.639419
280143 -1.413170  0.248525 -1.127396  …  0.370612  0.028234 -0.145640
280149 -2.234739  1.210158 -0.652250  …  0.751826  0.834108  0.190944
281144 -2.208002  1.058733 -1.632333  …  0.583276 -0.269209 -0.456108
281674  0.223050 -0.068384  0.577829  … -0.164350 -0.295135 -0.072173

              V24       V25       V26       V27       V28  Amount  Class
541     0.320198  0.044519  0.177840  0.261145 -0.143276    0.00      1
623    -0.293803  0.279798 -0.145362 -0.252773  0.035764  529.00      1
4920   -0.087330 -0.156114 -0.542628  0.039566 -0.153029  239.93      1
6108   -0.053502  0.252405 -0.657488 -0.827136  0.849573   59.00      1
6329   -1.632653  1.488901  0.566797 -0.010016  0.146793    1.00      1
…            …       …       …       …       …     …     …
279863 -0.294885  0.537503  0.788395  0.292680  0.147968  390.00      1
280143 -0.081049  0.521875  0.739467  0.389152  0.186637    0.76      1
280149  0.032070 -0.739695  0.471111  0.385107  0.194361   77.89      1
281144 -0.183659 -0.328168  0.606116  0.884876 -0.253700  245.00      1
281674 -0.450261  0.313267 -0.289617  0.002988 -0.015309   42.53      1

[492 rows x 31 columns]
```

[11]:
```
print(normal.shape)
print(fraud.shape)
```

```
(284315, 31)
(492, 31)
```

[12]:
```
normal.Amount.describe()
```

[12]:
```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

[13]:
```
fraud.Amount.describe()
```

[13]:
```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

[14]:
```
dataset.groupby("Class").mean()
```

[14]:
```
                Time        V1        V2        V3        V4        V5  \
Class
0      94838.202258  0.008258 -0.006271  0.012171 -0.007860  0.005453
1      80746.806911 -4.771948  3.623778 -7.033281  4.542029 -3.151225

               V6        V7        V8        V9  …       V20       V21  \
Class                                            …
0       0.002419  0.009637 -0.000987  0.004467  … -0.000644 -0.001235
1      -1.397737 -5.568731  0.570636 -2.581123  …  0.372319  0.713588

               V22       V23       V24       V25       V26       V27       V28  \
Class
0      -0.000024  0.000070  0.000182 -0.000072 -0.000089 -0.000295 -0.000131
1       0.014049 -0.040308 -0.105130  0.041449  0.051648  0.170575  0.075667

            Amount
```

```
Class
0       88.291022
1      122.211321

[2 rows x 30 columns]
```

[15]:
```
normal_sample = normal.sample(n=492)
```

[16]:
```
new_dataset = pd.concat([normal_sample,fraud],axis=0)
new_dataset
```

[16]:

	Time	V1	V2	V3	V4	V5	V6	\
163281	115814.0	1.951895	-0.468120	-0.465042	0.130377	-0.507516	-0.110569	
239899	150324.0	-1.230216	-1.119585	0.227748	-2.337372	-2.550650	1.733640	
58459	48398.0	0.945462	-1.134712	0.733303	0.564243	-1.360709	0.380627	
73922	55346.0	-2.165022	1.399798	-1.254162	1.115989	-0.760542	-1.121995	
77684	57163.0	1.085327	-0.069774	1.396814	1.371929	-0.956212	0.076432	
…	…	…	…	…	…	…	…	
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	

	V7	V8	V9	…	V21	V22	V23	\
163281	-0.639960	0.169497	0.872227	…	-0.218491	-0.660427	0.472225	
239899	0.385890	0.436121	-0.112403	…	-0.231337	-0.005166	-0.258115	
58459	-0.910520	0.358117	-0.336802	…	-0.110533	-0.109414	-0.124650	
73922	0.074055	1.010374	-0.876768	…	0.202071	0.688712	0.184993	
77684	-0.626901	0.205048	0.860754	…	-0.012451	0.204601	0.064112	
…	…	…	…	…	…	…	…	
279863	-0.882850	0.697211	-2.064945	…	0.778584	-0.319189	0.639419	
280143	-1.413170	0.248525	-1.127396	…	0.370612	0.028234	-0.145640	
280149	-2.234739	1.210158	-0.652250	…	0.751826	0.834108	0.190944	
281144	-2.208002	1.058733	-1.632333	…	0.583276	-0.269209	-0.456108	
281674	0.223050	-0.068384	0.577829	…	-0.164350	-0.295135	-0.072173	

	V24	V25	V26	V27	V28	Amount	Class
163281	0.729420	-0.694095	0.178417	-0.054998	-0.046461	16.98	0
239899	-1.379771	-0.337664	0.942816	-0.055021	-0.307165	440.00	0
58459	-0.052709	0.286733	-0.216320	0.048356	0.038892	144.81	0
73922	0.583680	-0.579458	-0.437558	0.285882	0.062842	89.99	0
77684	0.398888	0.299052	-0.399348	0.089570	0.038289	9.99	0
…	…	…	…	…	…	…	
279863	-0.294885	0.537503	0.788395	0.292680	0.147968	390.00	1
280143	-0.081049	0.521875	0.739467	0.389152	0.186637	0.76	1
280149	0.032070	-0.739695	0.471111	0.385107	0.194361	77.89	1

```
281144 -0.183659 -0.328168  0.606116  0.884876 -0.253700  245.00      1
281674 -0.450261  0.313267 -0.289617  0.002988 -0.015309   42.53      1

[984 rows x 31 columns]
```

[17]:
```
new_dataset["Class"].value_counts()
```

[17]:
```
Class
0    492
1    492
Name: count, dtype: int64
```

[18]:
```
## Separating features and target variable
x = new_dataset.drop(columns="Class",axis=1)
y = new_dataset["Class"]

print(x)
print(y)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
163281  115814.0  1.951895 -0.468120 -0.465042  0.130377 -0.507516 -0.110569
239899  150324.0 -1.230216 -1.119585  0.227748 -2.337372 -2.550650  1.733640
58459    48398.0  0.945462 -1.134712  0.733303  0.564243 -1.360709  0.380627
73922    55346.0 -2.165022  1.399798 -1.254162  1.115989 -0.760542 -1.121995
77684    57163.0  1.085327 -0.069774  1.396814  1.371929 -0.956212  0.076432
…            …         …         …         …         …         …
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494
280143  169347.0  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.326536
280149  169351.0 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.003346
281144  169966.0 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.943548
281674  170348.0  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.096695

               V7        V8        V9  …       V20       V21       V22  \
163281 -0.639960  0.169497  0.872227  … -0.184962 -0.218491 -0.660427
239899  0.385890  0.436121 -0.112403  … -0.839060 -0.231337 -0.005166
58459  -0.910520  0.358117 -0.336802  … -0.449984 -0.110533 -0.109414
73922   0.074055  1.010374 -0.876768  … -0.189768  0.202071  0.688712
77684  -0.626901  0.205048  0.860754  … -0.173585 -0.012451  0.204601
…            …         …         …  …       …         …         …
279863 -0.882850  0.697211 -2.064945  …  1.252967  0.778584 -0.319189
280143 -1.413170  0.248525 -1.127396  …  0.226138  0.370612  0.028234
280149 -2.234739  1.210158 -0.652250  …  0.247968  0.751826  0.834108
281144 -2.208002  1.058733 -1.632333  …  0.306271  0.583276 -0.269209
281674  0.223050 -0.068384  0.577829  … -0.017652 -0.164350 -0.295135

              V23       V24       V25       V26       V27       V28  Amount
163281   0.472225  0.729420 -0.694095  0.178417 -0.054998 -0.046461   16.98
239899  -0.258115 -1.379771 -0.337664  0.942816 -0.055021 -0.307165  440.00
```

```
58459  -0.124650 -0.052709  0.286733 -0.216320  0.048356  0.038892  144.81
73922   0.184993  0.583680 -0.579458 -0.437558  0.285882  0.062842   89.99
77684   0.064112  0.398888  0.299052 -0.399348  0.089570  0.038289    9.99
…            …         …         …         …         …         …       …
279863  0.639419 -0.294885  0.537503  0.788395  0.292680  0.147968  390.00
280143 -0.145640 -0.081049  0.521875  0.739467  0.389152  0.186637    0.76
280149  0.190944  0.032070 -0.739695  0.471111  0.385107  0.194361   77.89
281144 -0.456108 -0.183659 -0.328168  0.606116  0.884876 -0.253700  245.00
281674 -0.072173 -0.450261  0.313267 -0.289617  0.002988 -0.015309   42.53

[984 rows x 30 columns]
163281    0
239899    0
58459     0
73922     0
77684     0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

[19]:
```
print(y)
```

```
163281    0
239899    0
58459     0
73922     0
77684     0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

[20]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪stratify=y, random_state=2)
```

[21]:
```
print(x.shape, x_train.shape, x_test.shape)
```

```
(984, 30) (787, 30) (197, 30)
```

[22]:
```
print(y.shape, y_train.shape, y_test.shape)
```

```
(984,) (787,) (197,)
```

[23]:
```
from sklearn.linear_model import LogisticRegression

# Assuming x_train and y_train are already defined
model = LogisticRegression()
model.fit(x_train, y_train)
```

[23]:
```
LogisticRegression()
```

[24]:
```
from sklearn.metrics import accuracy_score

# Accuracy on training data
x_train_prediction = model.predict(x_train)
training_data_accuracy = accuracy_score(x_train_prediction, y_train)

print('Accuracy on training data:', training_data_accuracy)
```

```
Accuracy on training data: 0.940279542566709
```

from sklearn.metrics import accuracy_score

1 Accuracy on test data

x_test_prediction = model.predict(x_test) test_data_accuracy = accuracy_score(x_test_prediction, y_test)

print('Accuracy on test data:', test_data_accuracy)