Credit card fraud

September 27, 2023

1 Credit Card Fraud Detection

[1]:
```
#importing the Dependencies

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
data=pd.read_csv('creditcard.csv')
```

Loading the dataset

[3]:
```
data.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]:
```
data.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]:
```
data.shape
```

[5]:
```
(284807, 31)
```

[6]:
```
data.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
```

As the non-null content for all the rows shows that there are no null values.

[7]:
```
data.describe()
```

[7]:

	Time	V1	V2	V3	V4	\
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	

	V5	V6	V7	V8	V9	\

```
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   9.604066e-16  1.487313e-15 -5.556467e-16  1.213481e-16 -2.406331e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

       …           V21           V22           V23           V24  \
count  …  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   …  1.654067e-16 -3.568593e-16  2.578648e-16  4.473266e-15
std    …  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min    … -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%    … -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%    … -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%    …  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max    …  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00

                V25           V26           V27           V28         Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000
mean   5.340915e-16  1.683437e-15 -3.660091e-16 -1.227390e-16      88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01     250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000

              Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
```

The .describe() provides the description of the dataset which has count, mean, standar deviation , min , max . The 25% 50% 75% descibes the amount of data is less than that value.

Another method to check the missing values!

[8]:
```
data.isnull().sum()
```

[8]:
```
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
```

[9]:
```
data.corr()
```

[9]:
```
              Time            V1            V2            V3            V4  \
Time     1.000000  1.173963e-01 -1.059333e-02 -4.196182e-01 -1.052602e-01
V1       0.117396  1.000000e+00  4.135835e-16 -1.227819e-15 -9.215150e-16
V2      -0.010593  4.135835e-16  1.000000e+00  3.243764e-16 -1.121065e-15
V3      -0.419618 -1.227819e-15  3.243764e-16  1.000000e+00  4.711293e-16
V4      -0.105260 -9.215150e-16 -1.121065e-15  4.711293e-16  1.000000e+00
V5       0.173072  1.812612e-17  5.157519e-16 -6.539009e-17 -1.719944e-15
V6      -0.063016 -6.506567e-16  2.787346e-16  1.627627e-15 -7.491959e-16
V7       0.084714 -1.005191e-15  2.055934e-16  4.895305e-16 -4.104503e-16
V8      -0.036949 -2.433822e-16 -5.377041e-17 -1.268779e-15  5.697192e-16
V9      -0.008660 -1.513678e-16  1.978488e-17  5.568367e-16  6.923247e-16
V10      0.030617  7.388135e-17 -3.991394e-16  1.156587e-15  2.232685e-16
```

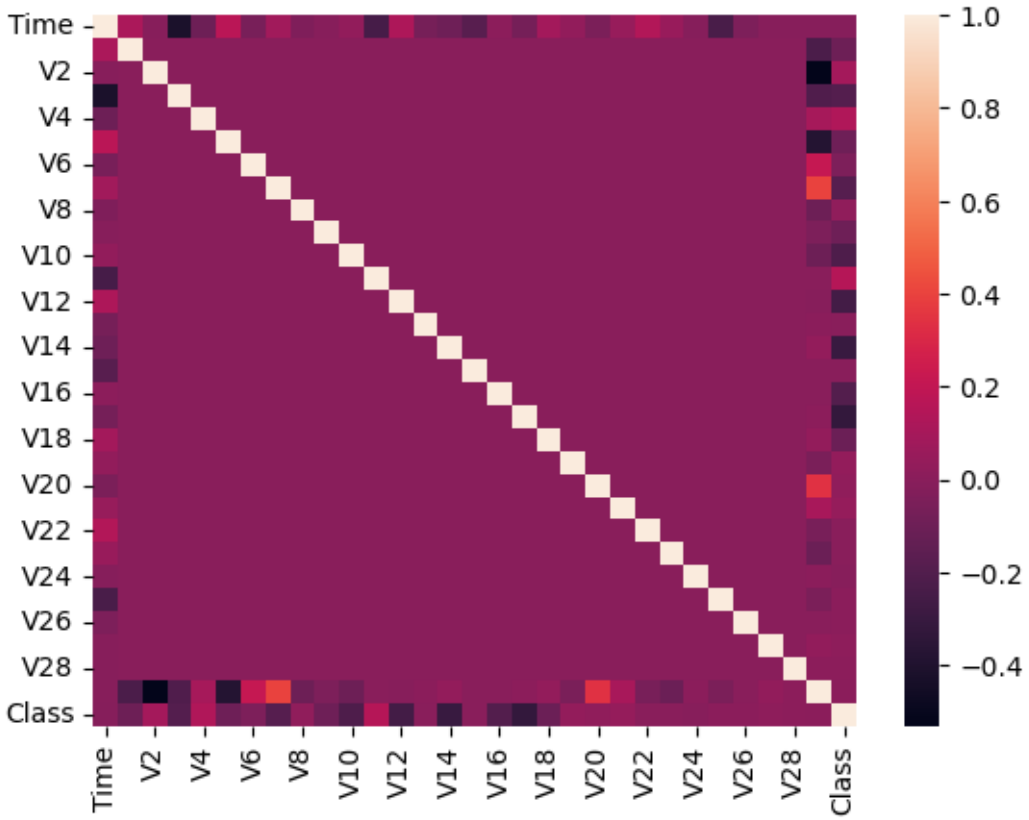
```
V11    -0.247689  2.125498e-16  1.975426e-16  1.576830e-15  3.459380e-16
V12     0.124348  2.053457e-16 -9.568710e-17  6.310231e-16 -5.625518e-16
V13    -0.065902 -2.425603e-17  6.295388e-16  2.807652e-16  1.303306e-16
V14    -0.098757 -5.020280e-16 -1.730566e-16  4.739859e-16  2.282280e-16
V15    -0.183453  3.547782e-16 -4.995814e-17  9.068793e-16  1.377649e-16
V16     0.011903  7.212815e-17  1.177316e-17  8.299445e-16 -9.614528e-16
V17    -0.073297 -3.879840e-16 -2.685296e-16  7.614712e-16 -2.699612e-16
V18     0.090438  3.230206e-17  3.284605e-16  1.509897e-16 -5.103644e-16
V19     0.028975  1.502024e-16 -7.118719e-18  3.463522e-16 -3.980557e-16
V20    -0.050866  4.654551e-16  2.506675e-16 -9.316409e-16 -1.857247e-16
V21     0.044736 -2.457409e-16 -8.480447e-17  5.706192e-17 -1.949553e-16
V22     0.144059 -4.290944e-16  1.526333e-16 -1.133902e-15 -6.276051e-17
V23     0.051142  6.168652e-16  1.634231e-16 -4.983035e-16  9.164206e-17
V24    -0.016182 -4.425156e-17  1.247925e-17  2.686834e-19  1.584638e-16
V25    -0.233083 -9.605737e-16 -4.478846e-16 -1.104734e-15  6.070716e-16
V26    -0.041407 -1.581290e-17  2.057310e-16 -1.238062e-16 -4.247268e-16
V27    -0.005135  1.198124e-16 -4.966953e-16  1.045747e-15  3.977061e-17
V28    -0.009413  2.083082e-15 -5.093836e-16  9.775546e-16 -2.761403e-18
Amount -0.010596 -2.277087e-01 -5.314089e-01 -2.108805e-01  9.873167e-02
Class  -0.012323 -1.013473e-01  9.128865e-02 -1.929608e-01  1.334475e-01

                V5            V6            V7            V8            V9  \
Time     1.730721e-01 -6.301647e-02  8.471437e-02 -3.694943e-02 -8.660434e-03
V1       1.812612e-17 -6.506567e-16 -1.005191e-15 -2.433822e-16 -1.513678e-16
V2       5.157519e-16  2.787346e-16  2.055934e-16 -5.377041e-17  1.978488e-17
V3      -6.539009e-17  1.627627e-15  4.895305e-16 -1.268779e-15  5.568367e-16
V4      -1.719944e-15 -7.491959e-16 -4.104503e-16  5.697192e-16  6.923247e-16
V5       1.000000e+00  2.408382e-16  2.715541e-16  7.437229e-16  7.391702e-16
V6       2.408382e-16  1.000000e+00  1.191668e-16 -1.104219e-16  4.131207e-16
V7       2.715541e-16  1.191668e-16  1.000000e+00  3.344412e-16  1.122501e-15
V8       7.437229e-16 -1.104219e-16  3.344412e-16  1.000000e+00  4.356078e-16
V9       7.391702e-16  4.131207e-16  1.122501e-15  4.356078e-16  1.000000e+00
V10     -5.202306e-16  5.932243e-17 -7.492834e-17 -2.801370e-16 -4.642274e-16
V11      7.203963e-16  1.980503e-15  1.425248e-16  2.487043e-16  1.354680e-16
V12      7.412552e-16  2.375468e-16 -3.536655e-18  1.839891e-16 -1.079314e-15
V13      5.886991e-16 -1.211182e-16  1.266462e-17 -2.921856e-16  2.251072e-15
V14      6.565143e-16  2.621312e-16  2.607772e-16 -8.599156e-16  3.784757e-15
V15     -8.720275e-16 -1.531188e-15 -1.690540e-16  4.127777e-16 -1.051167e-15
V16      2.246261e-15  2.623672e-18  5.869302e-17 -5.254741e-16 -1.214086e-15
V17      1.281914e-16  2.015618e-16  2.177192e-16 -2.269549e-16  1.113695e-15
V18      5.308590e-16  1.223814e-16  7.604126e-17 -3.667974e-16  4.993240e-16
V19     -1.450421e-16 -1.865597e-16 -1.881008e-16 -3.875186e-16 -1.376135e-16
V20     -3.554057e-16 -1.858755e-16  9.379684e-16  2.033737e-16 -2.343720e-16
V21     -3.920976e-16  5.833316e-17 -2.027779e-16  3.892798e-16  1.936953e-16
V22      1.253751e-16 -4.705235e-19 -8.898922e-16  2.026927e-16 -7.071869e-16
V23     -8.428683e-18  1.046712e-16 -4.387401e-16  6.377260e-17 -5.214137e-16
V24     -1.149255e-15 -1.071589e-15  7.434913e-18 -1.047097e-16 -1.430343e-16
```

```
V25     4.808532e-16  4.562861e-16 -3.094082e-16 -4.653279e-16  6.757763e-16
V26     4.319541e-16 -1.357067e-16 -9.657637e-16 -1.727276e-16 -7.888853e-16
V27     6.590482e-16 -4.452461e-16 -1.782106e-15  1.299943e-16 -6.709655e-17
V28    -5.613951e-18  2.594754e-16 -2.776530e-16 -6.200930e-16  1.110541e-15
Amount -3.863563e-01  2.159812e-01  3.973113e-01 -1.030791e-01 -4.424560e-02
Class  -9.497430e-02 -4.364316e-02 -1.872566e-01  1.987512e-02 -9.773269e-02

         …           V21           V22           V23           V24  \
Time     …  4.473573e-02  1.440591e-01  5.114236e-02 -1.618187e-02
V1       … -2.457409e-16 -4.290944e-16  6.168652e-16 -4.425156e-17
V2       … -8.480447e-17  1.526333e-16  1.634231e-16  1.247925e-17
V3       …  5.706192e-17 -1.133902e-15 -4.983035e-16  2.686834e-19
V4       … -1.949553e-16 -6.276051e-17  9.164206e-17  1.584638e-16
V5       … -3.920976e-16  1.253751e-16 -8.428683e-18 -1.149255e-15
V6       …  5.833316e-17 -4.705235e-19  1.046712e-16 -1.071589e-15
V7       … -2.027779e-16 -8.898922e-16 -4.387401e-16  7.434913e-18
V8       …  3.892798e-16  2.026927e-16  6.377260e-17 -1.047097e-16
V9       …  1.936953e-16 -7.071869e-16 -5.214137e-16 -1.430343e-16
V10      …  1.177547e-15 -6.418202e-16  3.214491e-16 -1.355885e-16
V11      … -5.658364e-16  7.772895e-16 -4.505332e-16  1.933267e-15
V12      …  7.300527e-16  1.644699e-16  1.800885e-16  4.436512e-16
V13      …  1.008461e-16  6.747721e-17 -7.132064e-16 -1.397470e-16
V14      … -3.356561e-16  3.740383e-16  3.883204e-16  2.003482e-16
V15      …  6.605263e-17 -4.208921e-16 -3.912243e-16 -4.478263e-16
V16      … -4.715090e-16 -7.923387e-17  5.020770e-16 -3.005985e-16
V17      … -8.230527e-16 -8.743398e-16  3.706214e-16 -2.403828e-16
V18      … -9.408680e-16 -4.819365e-16 -1.912006e-16 -8.986916e-17
V19      …  5.115885e-16 -1.163768e-15  7.032035e-16  2.587708e-17
V20      … -7.614597e-16  1.009285e-15  2.712885e-16  1.277215e-16
V21      …  1.000000e+00  3.649908e-15  8.119580e-16  1.761054e-16
V22      …  3.649908e-15  1.000000e+00 -7.303916e-17  9.970809e-17
V23      …  8.119580e-16 -7.303916e-17  1.000000e+00  2.130519e-17
V24      …  1.761054e-16  9.970809e-17  2.130519e-17  1.000000e+00
V25      … -1.686082e-16 -5.018575e-16 -8.232727e-17  1.015391e-15
V26      … -5.557329e-16 -2.503187e-17  1.114524e-15  1.343722e-16
V27      … -1.211281e-15  8.461337e-17  2.839721e-16 -2.274142e-16
V28      …  5.278775e-16 -6.627203e-16  1.481903e-15 -2.819805e-16
Amount   …  1.059989e-01 -6.480065e-02 -1.126326e-01  5.146217e-03
Class    …  4.041338e-02  8.053175e-04 -2.685156e-03 -7.220907e-03

                V25           V26           V27           V28    Amount  \
Time   -2.330828e-01 -4.140710e-02 -5.134591e-03 -9.412688e-03 -0.010596
V1     -9.605737e-16 -1.581290e-17  1.198124e-16  2.083082e-15 -0.227709
V2     -4.478846e-16  2.057310e-16 -4.966953e-16 -5.093836e-16 -0.531409
V3     -1.104734e-15 -1.238062e-16  1.045747e-15  9.775546e-16 -0.210880
V4      6.070716e-16 -4.247268e-16  3.977061e-17 -2.761403e-18  0.098732
V5      4.808532e-16  4.319541e-16  6.590482e-16 -5.613951e-18 -0.386356
```

```
V6      4.562861e-16 -1.357067e-16 -4.452461e-16  2.594754e-16  0.215981
V7     -3.094082e-16 -9.657637e-16 -1.782106e-15 -2.776530e-16  0.397311
V8     -4.653279e-16 -1.727276e-16  1.299943e-16 -6.200930e-16 -0.103079
V9      6.757763e-16 -7.888853e-16 -6.709655e-17  1.110541e-15 -0.044246
V10    -2.846052e-16 -3.028119e-16 -2.197977e-16  4.864782e-17 -0.101502
V11    -5.600475e-16 -1.003221e-16 -2.640281e-16 -3.792314e-16  0.000104
V12    -5.712973e-16 -2.359969e-16 -4.672391e-16  6.415167e-16 -0.009542
V13    -5.497612e-16 -1.769255e-16 -4.720898e-16  1.144372e-15  0.005293
V14    -8.547932e-16 -1.660327e-16  1.044274e-16  2.289427e-15  0.033751
V15     3.206423e-16  2.817791e-16 -1.143519e-15 -1.194130e-15 -0.002986
V16    -1.345418e-15 -7.290010e-16  6.789513e-16  7.588849e-16 -0.003910
V17     2.666806e-16  6.932833e-16  6.148525e-16 -5.534540e-17  0.007309
V18    -6.629212e-17  2.990167e-16  2.242791e-16  7.976796e-16  0.035650
V19     9.577163e-16  5.898033e-16 -2.959370e-16 -1.405379e-15 -0.056151
V20     1.410054e-16 -2.803504e-16 -1.138829e-15 -2.436795e-16  0.339403
V21    -1.686082e-16 -5.557329e-16 -1.211281e-15  5.278775e-16  0.105999
V22    -5.018575e-16 -2.503187e-17  8.461337e-17 -6.627203e-16 -0.064801
V23    -8.232727e-17  1.114524e-15  2.839721e-16  1.481903e-15 -0.112633
V24     1.015391e-15  1.343722e-16 -2.274142e-16 -2.819805e-16  0.005146
V25     1.000000e+00  2.646517e-15 -6.406679e-16 -7.008939e-16 -0.047837
V26     2.646517e-15  1.000000e+00 -3.667715e-16 -2.782204e-16 -0.003208
V27    -6.406679e-16 -3.667715e-16  1.000000e+00 -3.061287e-16  0.028825
V28    -7.008939e-16 -2.782204e-16 -3.061287e-16  1.000000e+00  0.010258
Amount -4.783686e-02 -3.208037e-03  2.882546e-02  1.025822e-02  1.000000
Class   3.307706e-03  4.455398e-03  1.757973e-02  9.536041e-03  0.005632

           Class
Time   -0.012323
V1     -0.101347
V2      0.091289
V3     -0.192961
V4      0.133447
V5     -0.094974
V6     -0.043643
V7     -0.187257
V8      0.019875
V9     -0.097733
V10    -0.216883
V11     0.154876
V12    -0.260593
V13    -0.004570
V14    -0.302544
V15    -0.004223
V16    -0.196539
V17    -0.326481
V18    -0.111485
V19     0.034783
```

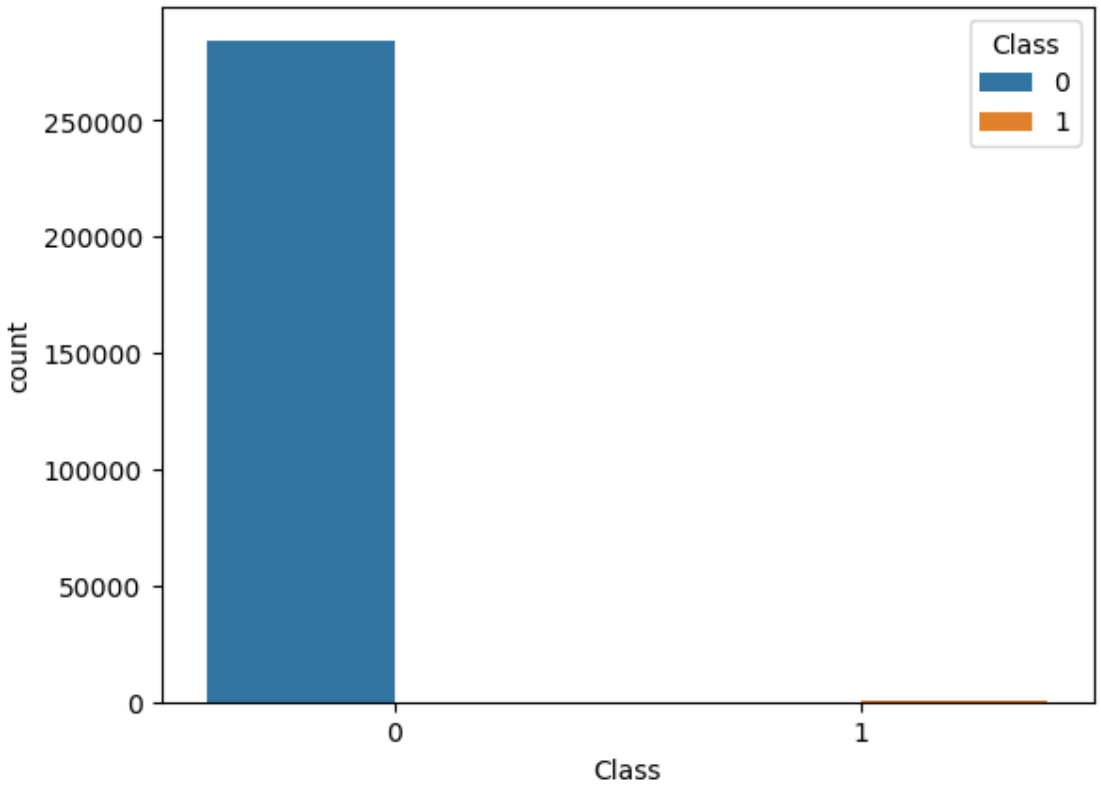
```
V20      0.020090
V21      0.040413
V22      0.000805
V23     -0.002685
V24     -0.007221
V25      0.003308
V26      0.004455
V27      0.017580
V28      0.009536
Amount   0.005632
Class    1.000000

[31 rows x 31 columns]
```

[10]:
```
sns.heatmap(data.corr())
plt.show()
```



Now we will check the no of datapoints for fraud and normal transaction

[12]:
```
sns.countplot(x=data['Class'],hue=data['Class'])
plt.show()
```



[13]:
```
data['Class'].value_counts()
```

[13]:
```
0    284315
1       492
Name: Class, dtype: int64
```

0–> represents Normal - Successfull transaction
1–> represents Fraud transaction

[ ]:
```

```

This data is Highly Unbalanced

[18]:
```
# Separating the data for analysis
legit=data[data.Class==0]
fraud=data[data.Class==1]
```

[16]:
```
legit
```

[16]:

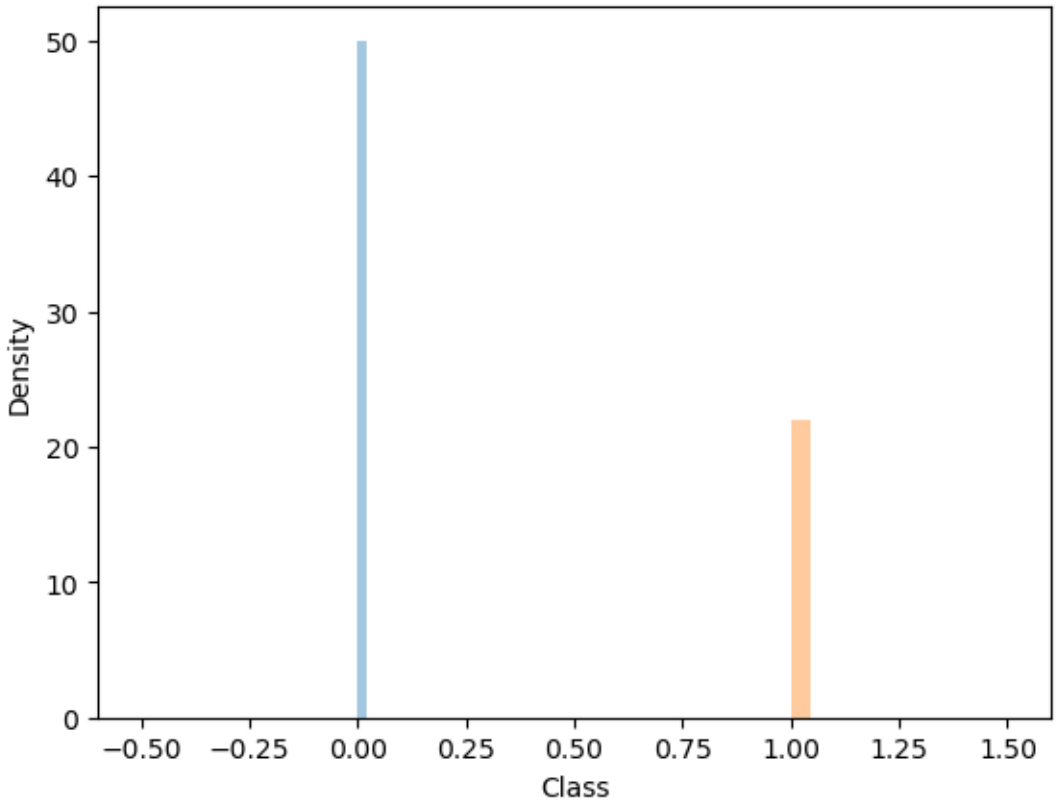
	Time	V1	V2	V3	V4	V5	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	

```
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]
```

[25]:
```
sns.distplot(legit['Class'])
sns.distplot(fraud['Class'])
plt.show()
```

```
C:\Users\morea\AppData\Local\Temp\ipykernel_17828\1492289659.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(legit['Class'])
C:\Users\morea\anaconda3\Lib\site-packages\seaborn\distributions.py:2511:
UserWarning: Dataset has 0 variance; skipping density estimate. Pass
`warn_singular=False` to disable this warning.
  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
C:\Users\morea\AppData\Local\Temp\ipykernel_17828\1492289659.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(fraud['Class'])
C:\Users\morea\anaconda3\Lib\site-packages\seaborn\distributions.py:2511:
UserWarning: Dataset has 0 variance; skipping density estimate. Pass
`warn_singular=False` to disable this warning.
  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
```



So after separating we are creating the a varible which will only contain the Successful transaction and Fraud transaction separately

[17]:
```
fraud
```

[17]:
```
            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]
```

[113]:
```
# Statistical measures of the data
legit.Amount.describe()
```

[113]:
```
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
```

[114]:
```
fraud.Amount.describe()
```

[114]:
```
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
```

[115]:
```
data.groupby(['Class']).mean()
```

[115]:
```
               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]
```

#comparing the values for both Transaction

here we get the difference between fraud and normal transaction

2 Under- Sampling

Building a sample dataset similar distribution of normal transaction and Fraudulent Transactions

[116]:
```
print(legit.shape)
print(fraud.shape)
```

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

No of fraud transaction is 492

[117]:
```
legit_sample=legit.sample(n=492)
```

Concatinating two data frames

[118]:
```
# creating new dataframe
new_data=pd.concat([legit_sample,fraud],axis=0)
# when we mention axis=0 the values will be added one by one row wise
```

```
new_data
```

[118]:
```
            Time        V1        V2        V3        V4        V5        V6  \
33439    37232.0 -0.501447  0.712651  1.169116 -1.225253  0.971812 -0.819341
204924  135505.0 -0.564957  0.811559  0.464239 -0.862621  0.613020 -0.476524
256198  157580.0  2.312670 -0.768481 -2.349595 -2.363262  0.198628 -0.955647
186601  127175.0  1.816406  0.016845 -0.197042  3.699612  0.051726  0.835552
111971   72456.0  1.099969 -0.148905 -0.218407 -0.037333 -0.160822 -0.805581
…            …         …         …         …         …         …         …
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  \
33439    1.357775 -0.444223 -0.575280  … -0.435153 -1.243069 -0.082468
204924   0.846742 -0.087449 -0.175232  … -0.191043 -0.719781 -0.038501
256198   0.118797 -0.517728  0.262263  … -0.400117 -0.337171  0.013451
186601  -0.487958  0.178191 -0.139639  …  0.138355  0.364099  0.008468
111971   0.388450 -0.195183 -0.351148  … -0.316952 -1.213102  0.033129
…             …         …         …  …         …         …         …
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
33439    -0.426980 -0.134262  0.490712 -0.206173 -0.138143   25.45      0
204924   -0.431617 -0.086103  0.183536 -0.463478 -0.019079    0.89      0
256198    0.090782  0.443126 -0.124141 -0.010520 -0.064766    9.85      0
186601   -1.033612 -0.133229  0.085461  0.001070 -0.031270   68.26      0
111971    0.031367  0.146987  0.666842 -0.127852  0.004517  103.97      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]
```

[119]:
```
new_data.head()
# Here we get the Normal data
```

[119]:
```
            Time        V1        V2        V3        V4        V5        V6  \
33439    37232.0 -0.501447  0.712651  1.169116 -1.225253  0.971812 -0.819341
204924  135505.0 -0.564957  0.811559  0.464239 -0.862621  0.613020 -0.476524
256198  157580.0  2.312670 -0.768481 -2.349595 -2.363262  0.198628 -0.955647
186601  127175.0  1.816406  0.016845 -0.197042  3.699612  0.051726  0.835552
111971   72456.0  1.099969 -0.148905 -0.218407 -0.037333 -0.160822 -0.805581

                V7        V8        V9  …       V21       V22       V23  \
33439    1.357775 -0.444223 -0.575280  … -0.435153 -1.243069 -0.082468
204924   0.846742 -0.087449 -0.175232  … -0.191043 -0.719781 -0.038501
256198   0.118797 -0.517728  0.262263  … -0.400117 -0.337171  0.013451
186601  -0.487958  0.178191 -0.139639  …  0.138355  0.364099  0.008468
111971   0.388450 -0.195183 -0.351148  … -0.316952 -1.213102  0.033129

               V24       V25       V26       V27       V28  Amount  Class
33439    -0.426980 -0.134262  0.490712 -0.206173 -0.138143   25.45      0
204924   -0.431617 -0.086103  0.183536 -0.463478 -0.019079    0.89      0
256198    0.090782  0.443126 -0.124141 -0.010520 -0.064766    9.85      0
186601   -1.033612 -0.133229  0.085461  0.001070 -0.031270   68.26      0
111971    0.031367  0.146987  0.666842 -0.127852  0.004517  103.97      0

[5 rows x 31 columns]
```

These are the random values we took from dataset

[120]:
```
new_data.tail()
# Here we get the Fraud data
```

[120]:
```
            Time        V1        V2        V3        V4        V5        V6  \
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  \
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
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
```

```
[5 rows x 31 columns]
```

[121]:
```
new_data['Class'].value_counts()
```

[121]:
```
0    492
1    492
Name: Class, dtype: int64
```

[122]:
```
new_data.groupby(['Class']).mean()
```

[122]:
```
                Time        V1        V2        V3        V4        V5  \
Class
0      94765.343496 -0.111169 -0.101239 -0.004216 -0.060914 -0.008734
1      80746.806911 -4.771948  3.623778 -7.033281  4.542029 -3.151225

              V6        V7        V8        V9  …       V20       V21  \
Class                                           …
0     -0.027341  0.021106  0.088718 -0.114013  …  0.008093  0.003672
1     -1.397737 -5.568731  0.570636 -2.581123  …  0.372319  0.713588

             V22       V23       V24       V25       V26       V27       V28  \
Class
0      0.023651 -0.007984 -0.056204  0.027500  0.068467 -0.031843 -0.021496
1      0.014049 -0.040308 -0.105130  0.041449  0.051648  0.170575  0.075667

            Amount
Class
0       94.903272
1      122.211321

[2 rows x 30 columns]
```

Hence the mean for both fraud and normal its similar, hence nature of dataset is not changed this will help out model to check whetheher the data is normal or fraudulant

[123]:
```
x=new_data.drop(columns=['Class'],axis=1)
y=new_data['Class']
```

[124]:
```
x
```

[124]:
```
            Time        V1        V2        V3        V4        V5        V6  \
33439    37232.0 -0.501447  0.712651  1.169116 -1.225253  0.971812 -0.819341
204924  135505.0 -0.564957  0.811559  0.464239 -0.862621  0.613020 -0.476524
256198  157580.0  2.312670 -0.768481 -2.349595 -2.363262  0.198628 -0.955647
186601  127175.0  1.816406  0.016845 -0.197042  3.699612  0.051726  0.835552
111971   72456.0  1.099969 -0.148905 -0.218407 -0.037333 -0.160822 -0.805581
…            …         …         …         …         …         …         …
```

```
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  \
33439   1.357775 -0.444223 -0.575280  …  0.137070 -0.435153 -1.243069
204924  0.846742 -0.087449 -0.175232  … -0.274283 -0.191043 -0.719781
256198  0.118797 -0.517728  0.262263  … -0.482962 -0.400117 -0.337171
186601 -0.487958  0.178191 -0.139639  … -0.144392  0.138355  0.364099
111971  0.388450 -0.195183 -0.351148  …  0.141449 -0.316952 -1.213102
…            …         …         …  …         …         …         …
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
33439  -0.082468 -0.426980 -0.134262  0.490712 -0.206173 -0.138143   25.45
204924 -0.038501 -0.431617 -0.086103  0.183536 -0.463478 -0.019079    0.89
256198  0.013451  0.090782  0.443126 -0.124141 -0.010520 -0.064766    9.85
186601  0.008468 -1.033612 -0.133229  0.085461  0.001070 -0.031270   68.26
111971  0.033129  0.031367  0.146987  0.666842 -0.127852  0.004517  103.97
…            …         …         …         …         …         …       …
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]
```

[125]:
```
y
```

[125]:
```
33439     0
204924    0
256198    0
186601    0
111971    0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
```

```
Name: Class, Length: 984, dtype: int64
```

[126]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

[127]:
```
x_train.shape
```

[127]:
```
(787, 30)
```

[128]:
```
x_test.shape
```

[128]:
```
(197, 30)
```

[129]:
```
x.shape
```

[129]:
```
(984, 30)
```

We are using Logisctic regression

[130]:
```
reg = LogisticRegression()
```

[131]:
```
reg
```

[131]:
```
LogisticRegression()
```

[132]:
```
# Training the Logistic regression model

reg.fit(x_train,y_train)
```

[132]:
```
LogisticRegression()
```

x_train contains all the features of the training data
y_train contains the corresponding label i.e 0 and 1

[133]:
```
y_pred=reg.predict(x_train)
```

[134]:
```
accuracy_score(y_train,y_pred)
```

[134]:
```
0.9453621346886912
```

so the accuracy of our model based on training the data is 94% using Logistic regression

[136]:
```
y_test_pred=reg.predict(x_test)
accuracy_score(y_test,y_test_pred)
```

[136]:
```
0.949238578680203
```

This is the accuracy of our model based on testing the data which is 94% using LR

In this model the train and testing accuracy is quite close to score then this model is fitting very well hence model perforemd well

[138]:
```
clf= SVC()
```

[139]:
```
clf
```

[139]:
```
SVC()
```

[140]:
```
clf.fit(x_train,y_train)
```

[140]:
```
SVC()
```

[145]:
```
y_train_pred=clf.predict(x_train)
```

[146]:
```
accuracy_score(y_train,y_train_pred)
```

[146]:
```
0.5285895806861499
```

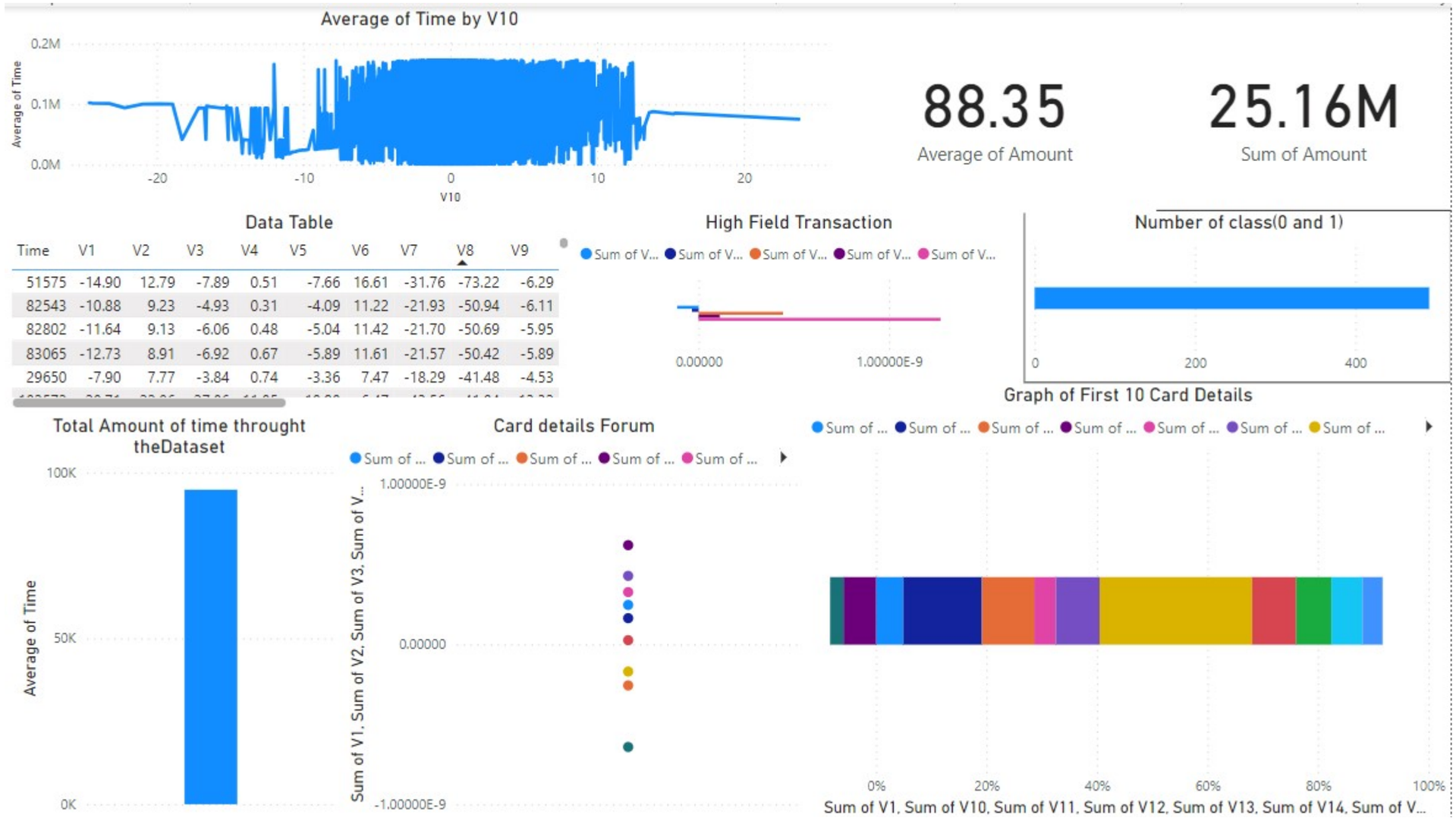
[143]:
```
y_pred=clf.predict(x_test)
```

[144]:
```
accuracy_score(y_test,y_pred)
```

[144]:
```
0.5329949238578681
```

so the using SVM in this is not reliable as training and testing are fitting well but the score is less as compared to LR

[ ]:
```

```


Average of Time by V10
0.2M
0.1M
0.0M
Average of Time
-20
-10
0
10
20
V10
88.35
Average of Amount
25.16M
Sum of Amount
Data Table
Time V1 V2 V3 V4 V5 V6 V7 V8 V9
51575 -14.90 12.79 -7.89 0.51 -7.66 16.61 -31.76 -73.22 -6.29
82543 -10.88 9.23 -4.93 0.31 -4.09 11.22 -21.93 -50.94 -6.11
82802 -11.64 9.13 -6.06 0.48 -5.04 11.42 -21.70 -50.69 -5.95
83065 -12.73 8.91 -6.92 0.67 -5.89 11.61 -21.57 -50.42 -5.89
29650 -7.90 7.77 -3.84 0.74 -3.36 7.47 -18.29 -41.48 -4.53
High Field Transaction
Sum of V... Sum of V... Sum of V... Sum of V... Sum of V...
0.00000
1.00000E-9
Number of class(0 and 1)
0
200
400
Graph of First 10 Card Details
Total Amount of time throught theDataset
100K
50K
0K
Average of Time
Card details Forum
Sum of ... Sum of ... Sum of ... Sum of ... Sum of ...
1.00000E-9
0.00000
-1.00000E-9
Sum of V1, Sum of V2, Sum of V3, Sum of V...
Sum of ... Sum of ... Sum of ... Sum of ... Sum of ... Sum of ... Sum of ...
0%
20%
40%
60%
80%
100%
Sum of V1, Sum of V10, Sum of V11, Sum of V12, Sum of V13, Sum of V14, Sum of V...