```
In [3]:
 import pandas as pd
 import numpy as np
 import matplotlib.pyplot as plt
 import seaborn as sns
 from matplotlib import gridspec
 In [7]:
 data = pd.read csv("kaggle credit.csv.zip")
 In [8]:
 data
Out[8]:
                        Time
                                                V1
                                                                    V2
                                                                                       ٧3
                                                                                                         V4
                                                                                                                           V5
                                                                                                                                              V6
                                                                                                                                                                V7
                                                                                                                                                                                  V8
                                                                                                                                                                                                     V9 ...
                                                                                                                                                                                                                                              V22
                                                                                                                                                                                                                           V21
                           0.0 \quad -1.359807 \quad -0.072781 \quad 2.536347 \quad 1.378155 \quad 0.338321 \quad 0.462388 \quad 0.239599 \quad 0.098698 \quad 0.363787 \quad \dots \quad 0.018307 \quad 0.277838 \quad 0.277838
                                                         0.0 1.191857
                           2.0 -1.158233
                                                          0.877737 \quad 1.548718 \quad 0.403034 \quad 0.407193 \quad 0.095921 \quad 0.592941 \quad 0.270533 \quad 0.817739 \quad \dots \quad 0.009431 \quad 0.798278
  284802 172786.0 11.881118 10.071785 9.834783 2.066656 5.364473 2.606837 4.918215 7.305334 1.914428 ... 0.213454 0.111864
  284803 172787.0 -0.732789 -0.055080 2.035030 0.738589 0.868229 1.058415 0.024330 0.294869 0.584800 ... 0.214205 0.924384
  284804 172788.0 1.919565 -0.301254 3.249640 0.557828 2.630515 3.031260 0.296827 0.708417 0.432454 ... 0.232045 0.578229
  284805 172788.0 -0.240440 0.530483 0.702510 0.689799 0.377961 0.623708 0.686180 0.679145 0.392087 ... 0.265245 0.800049
  284806 172792.0 -0.533413 -0.189733 0.703337 0.506271 0.012546 0.649617 1.577006 0.414650 0.486180 ... 0.261057 0.643078
 284807 rows × 31 columns
1
 In [9]:
 data.info
Out[9]:
 <bound method DataFrame.info of</pre>
                                                                                                                                            V1
                                                                                                                                                                    V2
                                                                                                                                                                                              V3
                                                                                                             Time
                                0.0 -1.359807 -0.072781 2.536347 1.378155 -0.338321
0
                                0.0 1.191857 0.266151 0.166480 0.448154 0.060018
                                1.0 -1.358354 -1.340163 1.773209 0.379780 -0.503198
                                1.0 -0.966272 -0.185226 1.792993 -0.863291 -0.010309
2.0 -1.158233 0.877737 1.548718 0.403034 -0.407193
3
 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
                                                                                                            V9 ...
                                                          V7
                                                                                   V8
                                                                                                                                               V21
                                                                                                                                                                       V22
                    0.462388 \quad 0.239599 \quad 0.098698 \quad 0.363787 \quad \dots \quad -0.018307 \quad 0.277838
 0
```

0 070000

0 000100

0 0 0 5 7 0 5

```
Τ
       -U.U8Z361 -U.U/88U3 U.U851UZ -U.Z554Z5 ... -U.ZZ5//5 -U.6386/Z

      1.800499
      0.791461
      0.247676 -1.514654
      ... 0.247998
      0.771679

      1.247203
      0.237609
      0.377436 -1.387024
      ... -0.108300
      0.005274

      0.095921
      0.592941
      -0.270533
      0.817739
      ... -0.009431
      0.798278

2
3
                                                . . .
                                     . . .
                                                                 . . .
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
                                                      ... 0.232045 0.578229
... 0.265245 0.800049
        3.031260 -0.296827
                               0.708417 0.432454
284805  0.623708 -0.686180  0.679145  0.392087
284806 -0.649617 1.577006 -0.414650 0.486180 ... 0.261057 0.643078
               V2.3
                          V2.4
                                     V2.5
                                                V2.6
                                                            V2.7
                                                                        V28 Amount \
        -0.110474 0.066928 0.128539 -0.189115 0.133558 -0.021053
0
        0.101288 -0.339846  0.167170  0.125895 -0.008983  0.014724
1
                                                                               2.69
        0.909412 -0.689281 -0.327642 -0.139097 -0.055353 -0.059752 378.66
        -0.190321 -1.175575 0.647376 -0.221929 0.062723 0.061458 123.50
       -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
              0
1
2
              0
              0
3
              0
284802
284803
284804
             0
284805
284806
             Ω
[284807 rows x 31 columns]>
In [10]:
data.shape
Out[10]:
(284807, 31)
In [11]:
data.describe()
Out[11]:
```

	Time	V1	V2	V3	V4	V5	V6	V7	1
count	284807.000000	2.848070e+05	2.848070e+						
mean	94813.859575	3.919560e-15	5.688174e-16	-8.769071e- 15	2.782312e-15	-1.552563e- 15	2.010663e-15	-1.694249e- 15	-1.927028
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+
min	0.000000	5.640751e+01	7.271573e+01	- 4.832559e+01	5.683171e+00	1.137433e+02	2.616051e+01	4.355724e+01	7.321672e+
25%	54201.500000	-9.203734e- 01	-5.985499e- 01	-8.903648e- 01	-8.486401e- 01	-6.915971e- 01	-7.682956e- 01	-5.540759e- 01	-2.086297
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e- 02	-5.433583e- 02	-2.741871e- 01	4.010308e-02	2.235804e-
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+

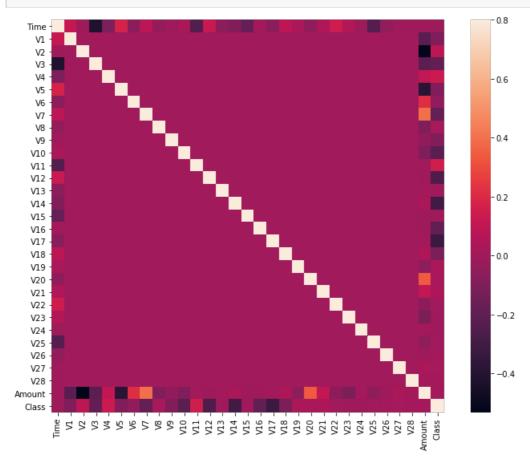
8 rows × 31 columns

```
In [13]:
  ######## determine the number of fraud case in the data set #############
  fraud = data[data['Class']==1]
 valid = data[data['Class']==0]
 outlierfraction = len(fraud)/float(len(valid))
 print(outlierfraction)
 print('Fraud Cases:{}'.format(len(data[data['Class']==1])))
 print('Valid Transactions:{}'.format(len(data[data['Class']==0])))
0.0017304750013189597
Fraud Cases: 492
Valid Transactions: 284315
In [ ]:
 \# Only 0.17% of all transactions are fraudulent. The data is highly imbalanced.
  # Let's apply our unbalanced models first, and if we don't get good accuracy, we can find a way to
 balance this dataset.
In [14]:
 print('Amount details of Fraud Transaction')
  fraud.Amount.describe() ####### we can see in this avarage monetary transaction there is more f
  raudulent transactions #######
Amount details of Fraud Transaction
Out[14]:
                                            492.000000
count
                                                122.211321
mean
                                                256.683288
std
min
                                                          0.000000
                                                         1.000000
25%
50%
                                                        9.250000
75%
                                               105.890000
                                             2125.870000
max
Name: Amount, dtype: float64
In [16]:
 ####### plotting correlation matrix #########
  plt.figure(figsize=(30,30))
 sns.heatmap(data.corr(), annot=True, cmap="RdYlGn", annot kws={"size":15})
Out[16]:
 <matplotlib.axes._subplots.AxesSubplot at 0x288ed486148>
                       0.12 - 0.011 - 0.42 - 0.11 - 0.17 - 0.0630.085 - 0.037 \cdot 0.097 \cdot 0.031 - 0.25 - 0.12 - 0.0660.099 - 0.18 \cdot 0.012 - 0.073 \cdot 0.09 \cdot 0.029 \cdot 0.051 \cdot 0.045 \cdot 0.14 \cdot 0.051 \cdot 0.016 - 0.23 - 0.043 \cdot 0.050 \cdot 0.094 \cdot 0.014 \cdot 0.01
                          1 .7e-17.4e-158e-154e-174e-1Qe-159.5e-177.e-154e-174e-174e-154e-15.1e-164e-15.1e-164e-15.8e-166e-16.9e-158e-16e-16.8e-176e-157.8e-176e-177.8e-176e-177.8e-1768e-176e-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-1769-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177.8e-177
                                                   .5e-115.1e-1-2e-165e-164e-164.4e-2527e-1478e-1965e-145.6e-1969e-115.5e-268e-1459e-197.9e-1266e-1255e-197.3e-1364e-127.5e-1151e-145.1e-1483e-1276e-145.5e-1157e-1
                                                    1 8.4e-164e-154e-1254e-124.4e-146.2e-163e-1255e-1272e-16.9e-1863e-151e-1162e-1856e-124e-1256e-18.4e-166e-124.6e-1251e-19.4e-1878e-1652e-1652e-1652e-1679e-1
                                                                  1 .9e-127e-166e-153e-1569e-161e-151e-351e-355e-355e-355e-375e-369e-364e-165e-157e-362e-16e-16.1e-16e-17.2e-354e-16.2e-364e-37.9e-10.099 0.13
  g -0.063.4e-15.6e-16.4e-13.7e-1769e-16 1 .4e-15.7e-1769e-16 1 .4e-15.7e-1769e-16 1 .4e-15.7e-1769e-16 1 .4e-15.7e-1769e-16 1 .4e-15.7e-1769e-16 1 .4e-176.7e-1769e-176 1 .4e-176.7e-176 1 .4e-176.7e-1769e-176 1 .4e-176.7e-1769e-176 1 .4e-176.7e-1769e-176 1 .4e-176.7e-1769e-176 1 .4e-176.7e-1769e-176 1 .4e-176.7e-1769e-176 1 .4e-176.7e-176 1 .4e-1
  g -0.0370.5e-147.4e-1574e-1562e-156e-15.7e-157e-171.2.9e-1661e-172e-166.3e-177.4e-151e-150e-165e-165.e-1461e-155e-161e-155e-151e-155e-150e-159e-15.8e-1564e-1562e-1564e-1562e-1569e-16.5e-160.1 0.02
  0.0317.4e-147.8e-163e-161e-165.6e-169e-169e-179.1e-127.8e-1 1 .6e-1164e-189e-1266e-1356e-1357e-139e-127.7e-177.1e-1851e-165.7e-189e-164e-127.9e-1266e-1361e-145.7e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e-127.1e
           0.25<mark>.</mark>4e-165e-16.5e-17.1e-176e-169e-16.1e-13e-164.7e-1766e-16-1 1.2e-159e-156e-14.8e-16.2e-16-1-6e-1-6.2e-15.3e-159e-158e-1772e-162e-14.6e-161e-1766e-161e-17600010.15
```

```
0.122.4e-16.6e-1262e-15.7e-1368e-1261e-1165e-16.3e-127.4e-1154e-1362e-1
                                                                                                                                                                                               .3e-1148e-1269e-1265e-126.9e-1269e-1263e-117.3e-1262e-126.9e-1164e-1469e-1261e-126.8e-1263e-1763e-1-6.009
   0.06<del>0</del>.1e-1869e-16.9e-1165e-1266e-1263e-1269e-127.4e-1267e-1269e-1169e-125.3e-1
                                                                                                                                                                                                            .8e-14.7e-164e-16.2e-162e-16.6e-168e-1885e-127.7e-157.9e-265e-161e-127.1e-1465e-161e-150.005-10.004
   0.09<mark>9</mark>.4e-126.5e-1263e-126.5e-137.6e-1365e-136.7e-1261e-1263e-1266e-1266e-127.8e-1268e-13
                                                                                                                                                                                                                          .2e-17/9e-147/6e-159e-17/1.1e-1168e-1166e-137/4e-1766e-127/3e-128/6e-1466e-1183e-1265e-150/034
     0.183.3e-1268e-1261e-126.5e-1261e-1264e-1189e-172e-161.6e-1756e-146.8e-1269e-146.7e-1462e-1
                                                                                                                                                                                                                            1 1
                                                                                                                                                                                                                                         .3e-1532e-1765e-176.6e-1862e-1769e-187.9e-1161e-146.6e-1869e-1768e-176.3e-1761e-1-50.0030.0047
 0.0126.3e-1469e-117.2e-146.9e-1265e-1265e-1269e-146e-146.3e-1167e-1462e-1265e-1264e-1269e-117.3e-15
                                                                                                                                                                                                                                                        .9e-153e-151e-153.9e-166.9e-1669e-167.5e-146.3e-1666e-1662e-1768e-1666e-166.0039-0.
  0.0735e-1-69.9e-1266e-1-74.4e-1164e-1266e-1161e-1-15.5e-1565e-1267e-126.7e-1-166.9e-1262e-1262e-1262e-1
                                                                                                                                                                                                                                                                      .6e-1359e-1269e-1768e-1264e-1664e-135.5e-1478e-1469e-1268e-115.2e-1260073-0.3
  0.092,9e-1266e-1564e-155e-161e-1258e-15.1e-1461e-162e-164e-166e-165,9e-162e-19e-17.5e-168e-15.6e-1
                                                                                                                                                                                                                                                                                      .4e-1459e-1161e-1257e-137.6e-1161e-1263e-1862e-1264e-1268e-160.036 -0.11
 0.029.8e-1965e-177.6e-176.7e-1761e-1767e-176.9e-176.3e-1161e-1767e-177.2e-1763e-177.6e-1161e-1766e-116e-158.9e-176.4e-1
                                                                                                                                                                                                                                                                                                   9e-164e-169.7e-167e-1361e-177.4e-1566e-115.1e-1164e-1-15.0560.035
 .0.0511e-169.3e-1464e-1462e-1261e-1169e-1167e-116.1e-1463e-1161e-1253e-1163e-1253e-118.8e-1362e-1369e-146.9e-1469e-1
                                                                                                                                                                                                                                                                                                                 .1e-151e-15e-161.6e-15.5e-16e-161.4e-151e-160.34 0.02
0.0451.8e-1864e-173e-171e-161.4e-1166e-1169e-116.4e-1166e-1271e-116.9e-1662e-1665e-1276e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.9e-127.
                                                                                                                                                                                                                                                                                                                                .9e-16.1e-1163e-126.8e-1469e-141e-15.1e-160.11 0.04
 0.147.5e-127.5e-1466e-1261e-1561e-15.4e-1261e-1555e-1269e-16.7e-158e-127.9e-126.7e-1874e-15.9e-169e-19.4e-126.7e-197.7e-1261e-159e-1
0.0519.8e-1161e-1261e-176e-176.6e-176.2e-1273e-1269e-1269e-1268e-1262e-1164e-145.9e-1766e-177.1e-1265e-1364e-146.6e-156/e-1361e-1361e-1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          -0.2
 0.016.4e-16.1e-16.4e-17.2e-16.3e-16.3e-16.3e-12.6e-17.8e-12.6e-14.6e-17.2e-16.9e-15.5e-16.6e-16.6e-16.3e-16.5e-17.1e-18.1e-17.1e-18.1e-17.6e-116.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16.3e-16
     D. 23<mark>9</mark>. 8e-1453e-1478e-1564e-1566e-151e-155e-15.4e-1161e-126.9e-1466e-1561e-1461e-147.6e-1569e-146.6e-1468e-15.3e-1464e-145.5e-1468e-151e-1458e-1
-0.011-0.23 -0.53 -0.21 0.099 -0.39 0.22 0.4 -0.1 -0.044 -0.1 0.000D.0095.00530.0340.0030.00730.036-0.056 0.34 0.11 -0.065-0.110.00510.0480.003D.029 0.01
-0.012 -0.1 0.091 -0.19 0.13 -0.0950.044 -0.19 0.02 -0.098-0.22 0.15 -0.26 0.0046 -0.3 -0.0042 -0.2 -0.33 -0.11 0.035 0.02 0.040.00081.0029.0072.003B.00450.0180.009B.0056
```

In [17]:

```
corrmat = data.corr()
fig = plt.figure(figsize=(12,9))
sns.heatmap(corrmat,vmax = .8, square = True)
plt.show()
```



in [].

```
# In the heat map, we can clearly see that most of the features are not correlated with other feat
ures, but there are some features that are positively or negatively correlated with each other. Fo
r example, V2 and V5 are strongly negatively correlated with the Amount function.
# We also see some correlation with the V20 and Amount.
In [23]:
\#\#\#\#\#\#\# dividing X and Y from the dataset \#\#\#\#\#\#\#\#
X= data.drop(['Class'],axis = 1)
Y =data['Class']
print (X.shape)
print (Y.shape)
xData = X.values
ydata = Y.values
(284807, 30)
(284807,)
In [26]:
###### using skitlearn to split data into training and testing #################
from sklearn.model selection import train test split
xTrain, xTest, yTrain, yTest = train test split(xData, ydata, test size = 0.2, random state = 42)
In [28]:
####### using Random Forest classifier #############
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(xTrain,yTrain)
ypred = rfc.predict(xTest)
In [32]:
######## evaluating each classifier ##########
from sklearn.metrics import classification report, accuracy score
from sklearn.metrics import precision score, recall score
from sklearn.metrics import f1 score,matthews corrcoef
from sklearn.metrics import confusion matrix
In [43]:
n outliers = len(fraud)
n errors = (ypred != yTest).sum()
print('The model used in Random Forest Classifier')
acc = accuracy score(yTest,ypred)
print('accuracy score {}'.format(acc))
prec = precision score(yTest,ypred)
                                      ##### Precision quantifies the number of positive class
predictions that actually belong to the positive class(correctly predicted positive observation to
the total predicted positive observation.
print('precision score {}'.format(prec))
                                  ##### Recall quantifies the number of positive class
rec = recall_score(yTest, yPred)
predictions made out of all positive examples in the dataset. true positives divided by the total
number of true positives and false negatives.
print('recall score {}'.format(rec))
f1 = f1_score(yTest,ypred)
                                   ##### F-Measure provides a single score that balances both the
concerns of precision and recall in one number.
print('f1 score {}'.format(f1))
                                      ####### as a measure of the quality of binary (two-class)
MCC = matthews corrcoef(yTest,ypred)
classifications,
print('matthews_corrcoef {}'.format(MCC))
```

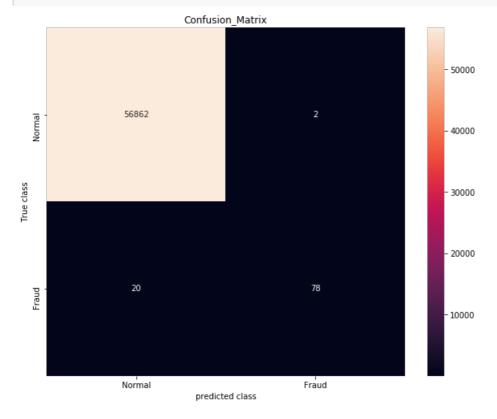
```
The model used in Random Forest Classifier accuracy_score 0.9996137776061234 precision_score 0.975 recall_score 0.7959183673469388 f1_score 0.8764044943820225 matthews_corrcoef 0.8807418913871203
```

In [37]:

```
########## confusion matrix ########

LABELS = ['Normal','Fraud']
conf_matrix = confusion_matrix(yTest,ypred)

plt.figure(figsize = (10,8))
sns.heatmap(conf_matrix,xticklabels=LABELS,yticklabels=LABELS, annot=True, fmt='d');
plt.title('Confusion_Matrix')
plt.xlabel('predicted class')
plt.ylabel('True class')
plt.show()
```



In [38]:

```
# Let's check the rest of the algorithms
# Importing Libraries
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report
from sklearn import metrics
```

In [42]:

```
###### decision tree #######
# Decision Tree
```

```
decision tree = DecisionTreeClassifier()
decision_tree.fit(xTrain, yTrain)
# predictions
yPred = decision tree.predict(xTest)
acc decision tree = round(decision tree.score(xTrain,yTrain)*100,2)
acc decision tree
Out[42]:
100.0
In [44]:
n outliers2 = len(fraud)
n errors2 = (ypred != yTest).sum()
print('Decision Tree')
acc2 = accuracy score(yTest,ypred)
print('accuracy score {}'.format(acc))
                                       ##### Precision quantifies the number of positive class
prec2 = precision score(yTest,ypred)
predictions that actually belong to the positive class(correctly predicted positive observation to
the total predicted positive observation.
print('precision score {}'.format(prec))
                                   ##### Recall quantifies the number of positive class
rec2 = recall_score(yTest, yPred)
predictions made out of all positive examples in the dataset. true positives divided by the total
number of true positives and false negatives.
print('recall score {}'.format(rec))
f1 = f1 score(yTest,ypred)
                                   ##### F-Measure provides a single score that balances both the
concerns of precision and recall in one number.
print('f1_score {}'.format(f1))
MCC2 = matthews corrcoef(yTest,ypred)
                                        ####### as a measure of the quality of binary (two-class)
classifications,
print('matthews corrcoef {}'.format(MCC))
Decision Tree
accuracy_score 0.9996137776061234
precision score 0.975
recall score 0.7959183673469388
fl score 0.8764044943820225
matthews corrcoef 0.8807418913871203
In [46]:
######### Random Forest ##########
random forest = RandomForestClassifier(n estimators=100)
random forest.fit(xTrain, yTrain)
y_pred = random_forest.predict(xTest)
random forest.score(xTrain, yTrain)
acc random forest = round(random forest.score(xTrain, yTrain) * 100, 2)
acc_random_forest
Out[46]:
100.0
In [47]:
# Gaussian Naive Bayes
gaussian = GaussianNB()
gaussian.fit(xTrain, yTrain)
y pred = gaussian.predict(xTest)
acc gaussian = round(gaussian.score(xTrain, yTrain) * 100, 2)
acc gaussian
print(classification_report(yTest, y_pred))
print("accuracy:", metrics.accuracy_score(yTest, y_pred))
```

precision recall f1-score support

```
0 1.00 0.99 1.00 56864
1 0.15 0.63 0.24 98
accuracy 0.99 56962
macro avg 0.57 0.81 0.62 56962
weighted avg 1.00 0.99 1.00 56962
```

accuracy: 0.9930128857835048

In [48]:

```
# Perceptron
perceptron = Perceptron()
perceptron.fit(xTrain, yTrain)
y_pred = perceptron.predict(xTest)
acc_perceptron = round(perceptron.score(xTrain, yTrain) * 100, 2)
acc_perceptron
print(classification_report(yTest, y_pred))
print("accuracy:",metrics.accuracy_score(yTest, y_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864 98
accuracy macro avg weighted avg	0.50 1.00	0.50 1.00	1.00 0.50 1.00	56962 56962 56962

accuracy: 0.9981917769741231

In [49]:

```
# Linear SVC
linear_svc = LinearSVC()
linear_svc.fit(xTrain, yTrain)
y_pred = linear_svc.predict(xTest)
acc_linear_svc = round(linear_svc.score(xTrain, yTrain) * 100, 2)
acc_linear_svc
print(classification_report(yTest, y_pred))
print("accuracy:",metrics.accuracy_score(yTest, y_pred))
```

C:\Users\Dipsikha\anaconda3\lib\site-packages\sklearn\svm_base.py:977: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.

"the number of iterations.", ConvergenceWarning)

	precision	recall	il-score	support
0 1	1.00 0.73	1.00 0.19	1.00 0.31	56864 98
accuracy macro avg weighted avg	0.86 1.00	0.60	1.00 0.65 1.00	56962 56962 56962

accuracy: 0.9984902215512096

In [50]:

```
# Stochastic Gradient Descent
sgd = SGDClassifier()
sgd.fit(xTrain, yTrain)
y_pred = sgd.predict(xTest)
acc_sgd = round(sgd.score(xTrain, yTrain) * 100, 2)
acc_sgd
print(classification_report(yTest, y_pred))
print("accuracy:",metrics.accuracy_score(yTest, y_pred))
```

precision recall f1-score support

```
U 1.00 1.00 1.00 56864
1 0.00 0.00 0.00 98
accuracy 1.00 56962
macro avg 0.50 0.50 0.50 56962
weighted avg 1.00 1.00 1.00 56962
```

accuracy: 0.9981917769741231

In [51]:

```
# K Nearest Neighbor:
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(xTrain, yTrain)
y_pred = knn.predict(xTest)
acc_knn = round(knn.score(xTrain, yTrain) * 100, 2)
acc_knn
print(classification_report(yTest, y_pred))
print("accuracy:",metrics.accuracy_score(yTest, y_pred))
```

	precision	recall	f1-score	support
0 1	1.00	1.00	1.00 0.17	56864 98
accuracy macro avg weighted avg	1.00	0.55 1.00	1.00 0.58 1.00	56962 56962 56962

accuracy: 0.9984375548611355

In [52]:

```
# Support Vector Machines
svc = SVC()
svc.fit(xTrain, yTrain)
y_pred = svc.predict(xTest)
acc_svc = round(svc.score(xTrain, yTrain) * 100, 2)
acc_svc
print(classification_report(yTest, y_pred))
print("accuracy:",metrics.accuracy_score(yTest, y_pred))
```

support	f1-score	recall	precision	
56864 98	1.00	1.00	1.00	0 1
56962 56962 56962	1.00 0.50 1.00	0.50 1.00	0.50 1.00	accuracy macro avg weighted avg

accuracy: 0.9982795547909132

C:\Users\Dipsikha\anaconda3\lib\site-packages\sklearn\metrics_classification.py:1221:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
 _warn_prf(average, modifier, msg_start, len(result))

In [53]:

```
# AdaBoost
from sklearn.ensemble import AdaBoostClassifier
Ada = AdaBoostClassifier(random_state=1)
Ada.fit(xTrain, yTrain)
y_pred = Ada.predict(xTest)
acc_add = round(Ada.score(xTrain, yTrain) * 100, 2)
acc_add
print(classification_report(yTest, y_pred))
print("accuracy:",metrics.accuracy_score(yTest, y_pred))
```

```
1.00 1.00
         0
              1.00
                                         56864
         1
                0.86
                        0.72
                                 0.78
                                            98
                                  1.00
                                          56962
   accuracy
  macro avg
                0.93
                         0.86
                                  0.89
                                          56962
                1.00
                         1.00
                                 1.00
                                          56962
weighted avg
```

accuracy: 0.9993153330290369

In [54]:

```
import xgboost as xgb

XGB = xgb.XGBClassifier(random_state = 1)

XGB.fit(xTrain,yTrain)
y_pred = XGB.predict(xTest)
acc_XGB = round(XGB.score(xTrain,yTrain)*100,2)
acc_XGB
print(classification_report(yTest,y_pred))
print("accuracy:",metrics.accuracy_score(yTest,y_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.99	0.80	0.88	98
accuracy			1.00	56962
macro avg	0.99	0.90	0.94	56962
weighted avg	1.00	1.00	1.00	56962

accuracy: 0.9996313331694814

In [55]:

Out[55]:

MODEL

SCORE

SCOKE	
100.00	Random Forest
100.00	Decision Tree
100.00	AdaBoost
99.93	XGBoost
99.86	KNN
99.85	Support Vector Machines
99.82	Logistic Regression
99.82	Perceptron
99.82	Stochastic Gradient Decent
99.35	Naive Bayes

In []: