Credit Card Fraud Detection -Descriptive Analysis and Preprocessing

March 8, 2022

```
In [8]: import numpy as np
        # working with arrays, has functionsin domain of linear algebra, fourier transform, and
        import pandas as pd
        # pd is used for data processing
        import matplotlib.pyplot as plt
        #plt provides an implicit, MATLAB-like, way of plotting
        import seaborn as sns
        #sns is a data visualization library based on matplotlib

        data=pd.read_csv('/Users/eden_zoo/Desktop/Certificate in Data Analytics/CIND820 Capstone
```

1 Descriptive Analysis and Preprocessing

```
In [3]: print(data.head())
```

```
   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  \
0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928
1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846
2  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281
3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575
4 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267

        V25       V26       V27       V28  Amount  Class
0  0.128539 -0.189115  0.133558 -0.021053  149.62      0
1  0.167170  0.125895 -0.008983  0.014724    2.69      0
2 -0.327642 -0.139097 -0.055353 -0.059752  378.66      0
3  0.647376 -0.221929  0.062723  0.061458  123.50      0
4 -0.206010  0.502292  0.219422  0.215153   69.99      0
```

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

In [55]:
```python
print(data.shape)
# the dataset has 284807 rows/samples and 31 columns/attributes
data.describe()
```

```
(284807, 31)
```

Out[55]:
```
                 V1            V2            V3            V4            V5  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   3.919560e-15  5.688174e-16 -8.769071e-15  2.782312e-15 -1.552563e-15
std    1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00  1.380247e+00
min   -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00 -1.137433e+02
25%   -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01 -6.915971e-01
50%    1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02 -5.433583e-02
75%    1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01  6.119264e-01
max    2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01  3.480167e+01

                 V6            V7            V8            V9           V10  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   2.010663e-15 -1.694249e-15 -1.927028e-16 -3.137024e-15  1.768627e-15
std    1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00  1.088850e+00
min   -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01 -2.458826e+01
25%   -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01 -5.354257e-01
50%   -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02 -9.291738e-02
75%    3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01  4.539234e-01
max    7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01  2.374514e+01

           ...                V22           V23           V24           V25  \
count      ...       2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean       ...       7.959909e-16  5.367590e-16  4.458112e-15  1.453003e-15
std        ...       7.257016e-01  6.244603e-01  6.056471e-01  5.212781e-01
min        ...      -1.093314e+01 -4.480774e+01 -2.836627e+00 -1.029540e+01
25%        ...      -5.423504e-01 -1.618463e-01 -3.545861e-01 -3.171451e-01
50%        ...       6.781943e-03 -1.119293e-02  4.097606e-02  1.659350e-02
75%        ...       5.285536e-01  1.476421e-01  4.395266e-01  3.507156e-01
max        ...       1.050309e+01  2.252841e+01  4.584549e+00  7.519589e+00

                V26           V27           V28          Class    scaled_time  \
count  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000  284807.000000
mean   1.699104e-15 -3.660161e-16 -1.206049e-16       0.001727       0.118914
std    4.822270e-01  4.036325e-01  3.300833e-01       0.041527       0.557903
min   -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000      -0.994983
25%   -3.269839e-01 -7.083953e-02 -5.295979e-02       0.000000      -0.358210
50%   -5.213911e-02  1.342146e-03  1.124383e-02       0.000000       0.000000
```

```
75%    2.409522e-01  9.104512e-02  7.827995e-02       0.000000       0.641790
max    3.517346e+00  3.161220e+01  3.384781e+01       1.000000       1.035022

       scaled_amount
count  284807.000000
mean        0.927124
std         3.495006
min        -0.307413
25%        -0.229162
50%         0.000000
75%         0.770838
max       358.683155

[8 rows x 31 columns]
```

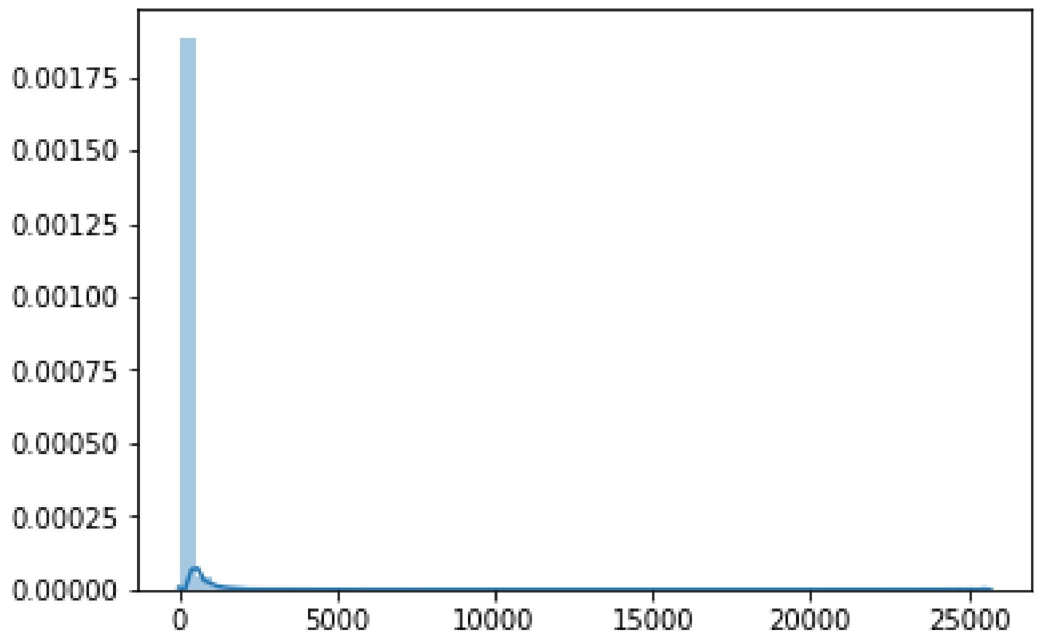
In [5]:
```
# Check data type and null values

data.dtypes.value_counts()
data[data.columns].isnull().sum()
# there is no missing value, because the dataset has gone through PCA
```

Out[5]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
```

```
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

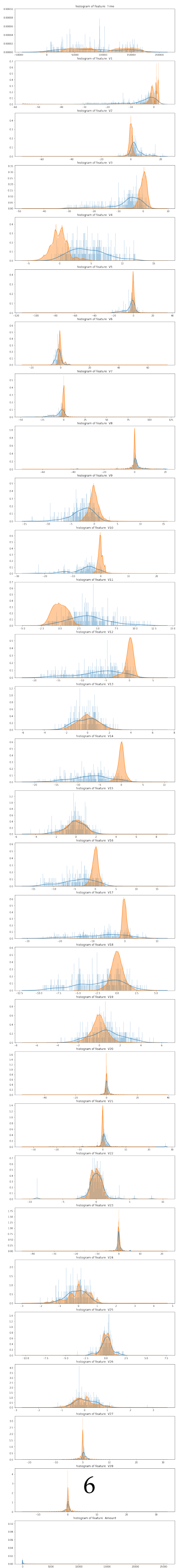
In [6]:
```python
amount=[data['Amount'].values]
sns.distplot(amount)
# Amount attribute is super skewed.
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb31f78e390>



In [7]:
```python
# Plot feature distributions. Orange is the majority, and blue is the minority.

from matplotlib import gridspec

features=data.iloc[:,0:30].columns

plt.figure(figsize=(12,30*4))
gs=gridspec.GridSpec(30,1)

for i,feature in enumerate(data[features]):
    ax=plt.subplot(gs[i])
    sns.distplot(data[feature][data.Class == 1],bins=500)
    sns.distplot(data[feature][data.Class == 0],bins=500)
    ax.set_xlabel('')
```

```
    ax.set_title('histogram of feature: '+str(feature))
plt.show()
```

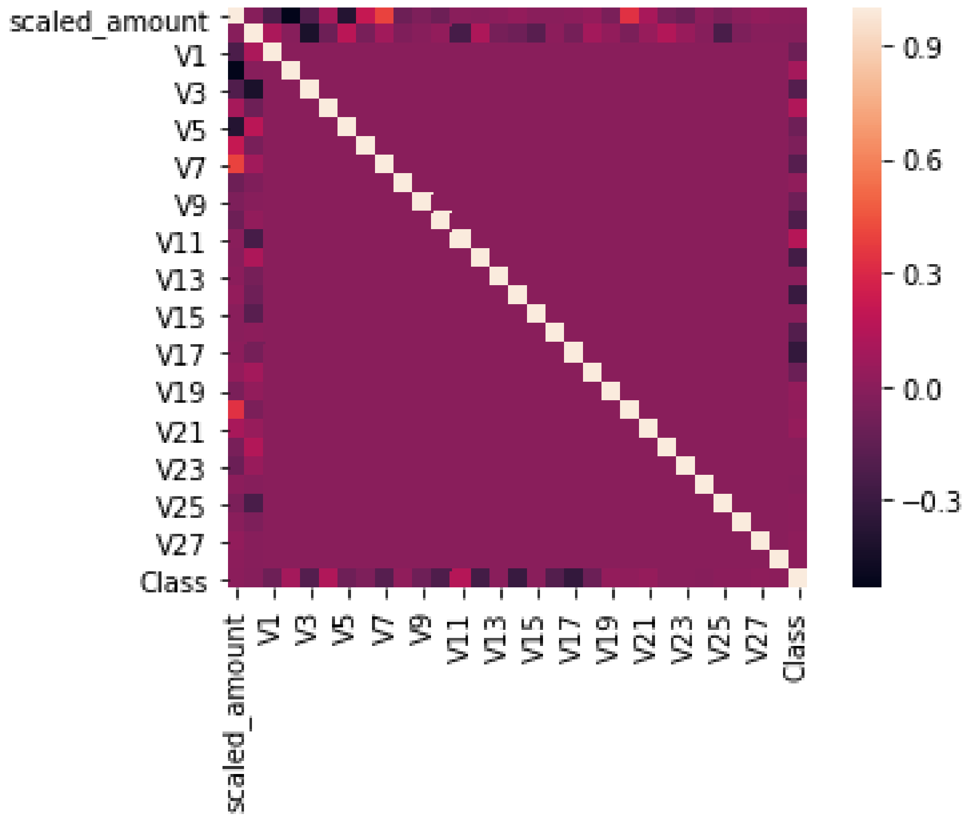

histogram of feature: Time
histogram of feature: V1
histogram of feature: V2
histogram of feature: V3
histogram of feature: V4
histogram of feature: V5
histogram of feature: V6
histogram of feature: V7
histogram of feature: V8
histogram of feature: V9
histogram of feature: V10
histogram of feature: V11
histogram of feature: V12
histogram of feature: V13
histogram of feature: V14
histogram of feature: V15
histogram of feature: V16
histogram of feature: V17
histogram of feature: V18
histogram of feature: V19
histogram of feature: V20
histogram of feature: V21
histogram of feature: V22
histogram of feature: V23
histogram of feature: V24
histogram of feature: V25
histogram of feature: V26
histogram of feature: V27
histogram of feature: V28
histogram of feature: Amount

In [9]:
```
#Histograms above show that there is very few outlier compared to the data size
#Histograms above show that in majority of attributes, the fraud distribution line fitte

#Rescaling amount and time features. Looking at the histograms, V1-V28 all seem scaled,
from sklearn.preprocessing import RobustScaler

rbst_scaler=RobustScaler() # robustscaler is less prone to outliers

data['scaled_time']=rbst_scaler.fit_transform(data['Time'].values.reshape(-1,1))
data['scaled_amount']=rbst_scaler.fit_transform(data['Amount'].values.reshape(-1,1))

data.drop(['Time','Amount'],axis=1,inplace=True)

scaled_time=data['scaled_time']
scaled_amount=data['scaled_amount']

data.drop(['scaled_time','scaled_amount'],axis=1,inplace=True)
data.insert(0,'scaled_amount',scaled_amount)
data.insert(1,'scaled_time',scaled_time)
```

In [10]:
```
# Check whether this dataset is imbalanced
fraud=data[data['Class']==1]
legitimate=data[data['Class']==0]
fraud_percentage = "{:.3%}".format(len(fraud)/float(len(legitimate)))
print('Fraud percentage in the dataset is ', fraud_percentage)

#Target percentage is only 0.173%, which means the dataset is highly unbalanced and req
```

```
Fraud percentage in the dataset is  0.173%
```

In [11]:
```
#correlation heatmap
corrmat=data.corr()
sns.heatmap(corrmat,square=True)
plt.show()
```


scaled_amount
V1
V3
V5
V7
V9
V11
V13
V15
V17
V19
V21
V23
V25
V27
Class
0.9
0.6
0.3
0.0
−0.3