ds

June 10, 2020

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
pd.set_option('display.max_columns', 100)
df = pd.read_csv('creditcard.csv')
df.head()
```

[2]:
```
   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       V10       V11       V12       V13       V14  \
0  0.098698  0.363787  0.090794 -0.551600 -0.617801 -0.991390 -0.311169
1  0.085102 -0.255425 -0.166974  1.612727  1.065235  0.489095 -0.143772
2  0.247676 -1.514654  0.207643  0.624501  0.066084  0.717293 -0.165946
3  0.377436 -1.387024 -0.054952 -0.226487  0.178228  0.507757 -0.287924
4 -0.270533  0.817739  0.753074 -0.822843  0.538196  1.345852 -1.119670

        V15       V16       V17       V18       V19       V20       V21  \
0  1.468177 -0.470401  0.207971  0.025791  0.403993  0.251412 -0.018307
1  0.635558  0.463917 -0.114805 -0.183361 -0.145783 -0.069083 -0.225775
2  2.345865 -2.890083  1.109969 -0.121359 -2.261857  0.524980  0.247998
3 -0.631418 -1.059647 -0.684093  1.965775 -1.232622 -0.208038 -0.108300
4  0.175121 -0.451449 -0.237033 -0.038195  0.803487  0.408542 -0.009431

        V22       V23       V24       V25       V26       V27       V28  \
0  0.277838 -0.110474  0.066928  0.128539 -0.189115  0.133558 -0.021053
1 -0.638672  0.101288 -0.339846  0.167170  0.125895 -0.008983  0.014724
2  0.771679  0.909412 -0.689281 -0.327642 -0.139097 -0.055353 -0.059752
3  0.005274 -0.190321 -1.175575  0.647376 -0.221929  0.062723  0.061458
4  0.798278 -0.137458  0.141267 -0.206010  0.502292  0.219422  0.215153

   Amount  Class
```

```
0  149.62      0
1    2.69      0
2  378.66      0
3  123.50      0
4   69.99      0
```

[3]:
```
df.shape
```

[3]:
```
(284807, 31)
```

[4]:
```
df.describe()
```

[4]:
```
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  3.919560e-15  5.688174e-16 -8.769071e-15  2.782312e-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  -1.552563e-15  2.010663e-15 -1.694249e-15 -1.927028e-16 -3.137024e-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

                V10           V11           V12           V13           V14  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   1.768627e-15  9.170318e-16 -1.810658e-15  1.693438e-15  1.479045e-15
std    1.088850e+00  1.020713e+00  9.992014e-01  9.952742e-01  9.585956e-01
min   -2.458826e+01 -4.797473e+00 -1.868371e+01 -5.791881e+00 -1.921433e+01
25%   -5.354257e-01 -7.624942e-01 -4.055715e-01 -6.485393e-01 -4.255740e-01
50%   -9.291738e-02 -3.275735e-02  1.400326e-01 -1.356806e-02  5.060132e-02
75%    4.539234e-01  7.395934e-01  6.182380e-01  6.625050e-01  4.931498e-01
max    2.374514e+01  1.201891e+01  7.848392e+00  7.126883e+00  1.052677e+01

                V15           V16           V17           V18           V19  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   3.482336e-15  1.392007e-15 -7.528491e-16  4.328772e-16  9.049732e-16
std    9.153160e-01  8.762529e-01  8.493371e-01  8.381762e-01  8.140405e-01
min   -4.498945e+00 -1.412985e+01 -2.516280e+01 -9.498746e+00 -7.213527e+00
25%   -5.828843e-01 -4.680368e-01 -4.837483e-01 -4.988498e-01 -4.562989e-01
50%    4.807155e-02  6.641332e-02 -6.567575e-02 -3.636312e-03  3.734823e-03
```

```
75%     6.488208e-01  5.232963e-01  3.996750e-01  5.008067e-01  4.589494e-01
max     8.877742e+00  1.731511e+01  9.253526e+00  5.041069e+00  5.591971e+00

                V20           V21           V22           V23           V24  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   5.085503e-16  1.537294e-16  7.959909e-16  5.367590e-16  4.458112e-15
std    7.709250e-01  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min   -5.449772e+01 -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%   -2.117214e-01 -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%   -6.248109e-02 -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%    1.330408e-01  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max    3.942090e+01  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   1.453003e-15  1.699104e-15 -3.660161e-16 -1.206049e-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
```

[5]:
```
df.isnull().sum()
```

[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
```

[6]:
```
print('No fraud percent', df.Class.value_counts()[0]/len(df)*100)
print('Fraud percent', df.Class.value_counts()[1]/len(df)*100)
```

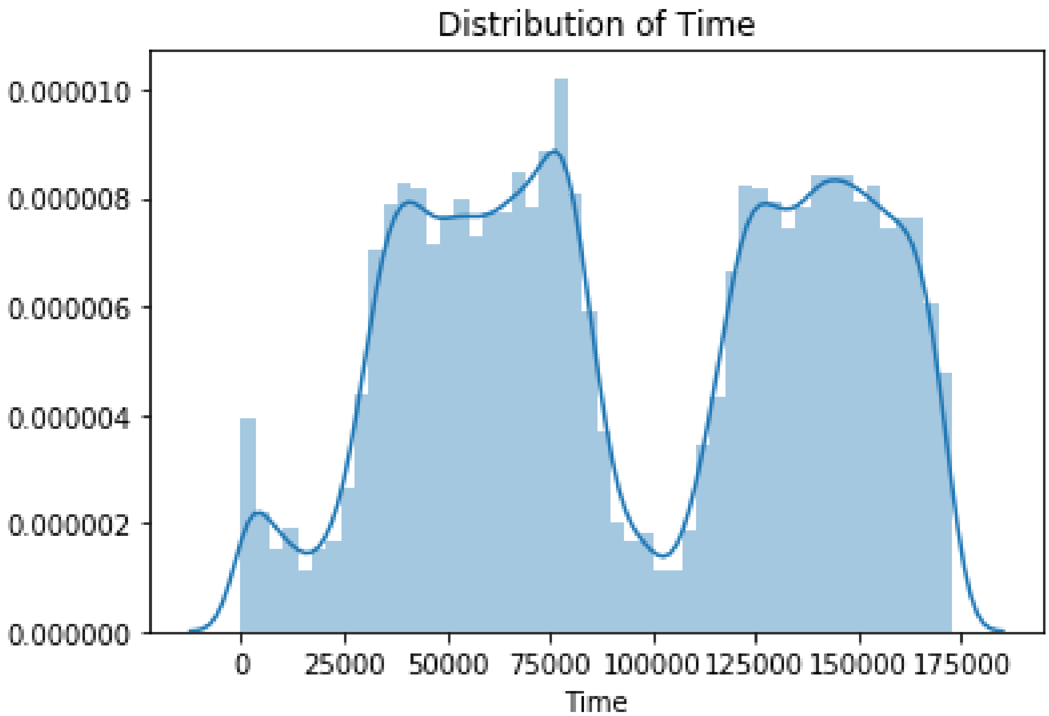
```
No fraud percent 99.82725143693798
Fraud percent 0.1727485630620034
```

[7]:
```
sns.countplot(df['Class'])
plt.title('No fraud 0 || Fraud 1')
```

[7]:
```
Text(0.5, 1.0, 'No fraud 0 || Fraud 1')
```

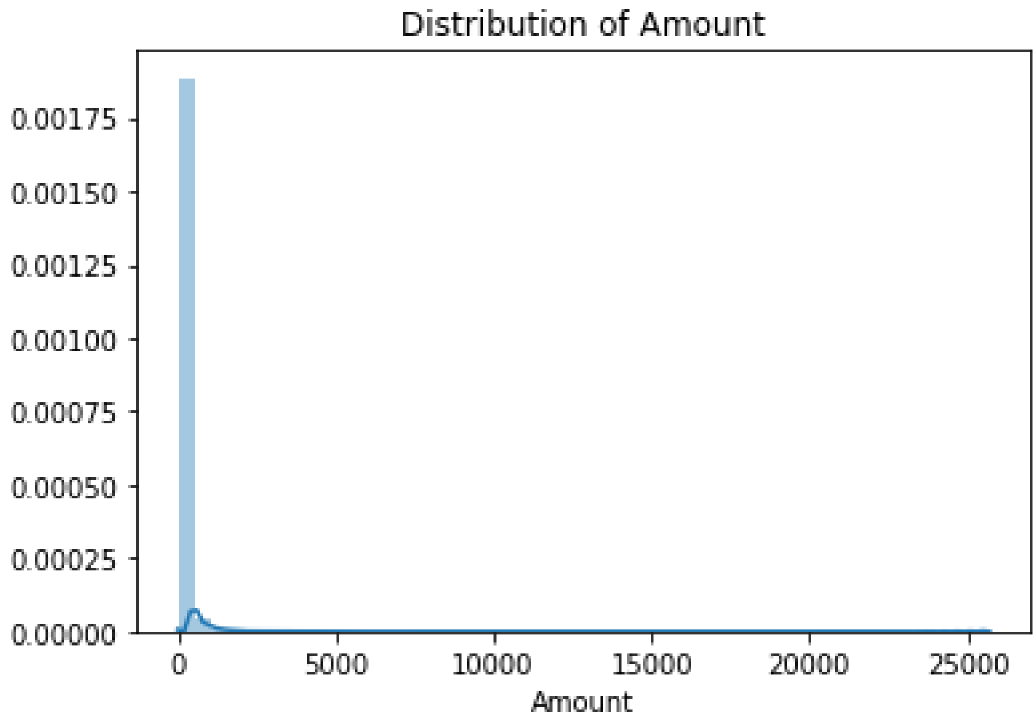
[8]:
```
sns.distplot(df['Time'])
plt.title('Distribution of Time')
```

[8]:
```
Text(0.5, 1.0, 'Distribution of Time')
```



[9]:
```
sns.distplot(df['Amount'])
plt.title('Distribution of Amount')
```

[9]:
```
Text(0.5, 1.0, 'Distribution of Amount')
```



[10]:
```
from sklearn.preprocessing import StandardScaler, RobustScaler

std_scaler = StandardScaler()
rob_scaler = RobustScaler()

df['scaled_amount'] = rob_scaler.fit_transform(df[['Amount']])
df['scaled_time'] = rob_scaler.fit_transform(df[['Time']])
```

[11]:
```
df.drop(['Time','Amount'], axis=1, inplace=True)
scaled_amount = df['scaled_amount']
scaled_time = df['scaled_time']

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

# Amount and Time are Scaled!

df.head()
```

[11]:
```
   scaled_amount  scaled_time        V1        V2        V3        V4  \
0       1.783274    -0.994983 -1.359807 -0.072781  2.536347  1.378155
1      -0.269825    -0.994983  1.191857  0.266151  0.166480  0.448154
2       4.983721    -0.994972 -1.358354 -1.340163  1.773209  0.379780
3       1.418291    -0.994972 -0.966272 -0.185226  1.792993 -0.863291
```

```
4       0.670579    -0.994960 -1.158233  0.877737  1.548718  0.403034

         V5        V6        V7        V8        V9       V10       V11  \
0 -0.338321  0.462388  0.239599  0.098698  0.363787  0.090794 -0.551600
1  0.060018 -0.082361 -0.078803  0.085102 -0.255425 -0.166974  1.612727
2 -0.503198  1.800499  0.791461  0.247676 -1.514654  0.207643  0.624501
3 -0.010309  1.247203  0.237609  0.377436 -1.387024 -0.054952 -0.226487
4 -0.407193  0.095921  0.592941 -0.270533  0.817739  0.753074 -0.822843

        V12       V13       V14       V15       V16       V17       V18  \
0 -0.617801 -0.991390 -0.311169  1.468177 -0.470401  0.207971  0.025791
1  1.065235  0.489095 -0.143772  0.635558  0.463917 -0.114805 -0.183361
2  0.066084  0.717293 -0.165946  2.345865 -2.890083  1.109969 -0.121359
3  0.178228  0.507757 -0.287924 -0.631418 -1.059647 -0.684093  1.965775
4  0.538196  1.345852 -1.119670  0.175121 -0.451449 -0.237033 -0.038195

        V19       V20       V21       V22       V23       V24       V25  \
0  0.403993  0.251412 -0.018307  0.277838 -0.110474  0.066928  0.128539
1 -0.145783 -0.069083 -0.225775 -0.638672  0.101288 -0.339846  0.167170
2 -2.261857  0.524980  0.247998  0.771679  0.909412 -0.689281 -0.327642
3 -1.232622 -0.208038 -0.108300  0.005274 -0.190321 -1.175575  0.647376
4  0.803487  0.408542 -0.009431  0.798278 -0.137458  0.141267 -0.206010

        V26       V27       V28  Class
0 -0.189115  0.133558 -0.021053      0
1  0.125895 -0.008983  0.014724      0
2 -0.139097 -0.055353 -0.059752      0
3 -0.221929  0.062723  0.061458      0
4  0.502292  0.219422  0.215153      0
```

[12]:
```python
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.model_selection import KFold, StratifiedKFold

X = df.drop('Class', axis = 1)
y = df['Class']

sss = StratifiedKFold(n_splits=5, random_state=None, shuffle=False)

for train_index, test_index in sss.split(X, y):
    print("Train:", train_index, "Test:", test_index)
    original_Xtrain, original_Xtest = X.iloc[train_index], X.iloc[test_index]
    original_ytrain, original_ytest = y.iloc[train_index], y.iloc[test_index]
```

```
Train: [ 30473  30496  31002 ... 284804 284805 284806] Test: [    0     1     2
... 57017 57018 57019]
Train: [     0      1      2 ... 284804 284805 284806] Test: [ 30473  30496
31002 ... 113964 113965 113966]
```

```
Train: [     0      1      2 ... 284804 284805 284806] Test: [ 81609  82400
83053 ... 170946 170947 170948]
Train: [     0      1      2 ... 284804 284805 284806] Test: [150654 150660
150661 ... 227866 227867 227868]
Train: [     0      1      2 ... 227866 227867 227868] Test: [212516 212644
213092 ... 284804 284805 284806]
```

[13]:
```python
print(original_Xtrain.shape)
print(original_Xtest.shape)
```

```
(227846, 30)
(56961, 30)
```

[14]:
```python
df = df.sample(frac=1)

# amount of fraud classes 492 rows.
fraud_df = df.loc[df['Class'] == 1]
non_fraud_df = df.loc[df['Class'] == 0][:492]
```

[15]:
```python
fraud_df.head()
```

[15]:
```
        scaled_amount  scaled_time        V1        V2         V3        V4  \
9179        -0.293440    -0.840776 -2.880042  5.225442 -11.063330  6.689951
215132       9.798225     0.649197 -2.921944 -0.228062  -5.877289  2.201884
275992       8.555858     0.964990 -2.027135 -1.131890  -1.135194  1.086963
74507        1.515266    -0.341569 -7.427924  2.948209  -8.678550  5.185303
238366      -0.279466     0.763026  0.754316  2.379822  -5.137274  3.818392

              V5        V6         V7        V8        V9        V10  \
9179   -5.759924 -2.244031 -11.199975  4.014722 -3.429304 -11.561950
215132 -1.935440  0.631141  -1.245106  1.511348 -1.899987  -6.428231
275992 -0.010547  0.423797   3.790880 -1.155595 -0.063434   1.334414
74507  -4.761090 -0.957095  -7.773380  0.717309 -3.682359  -8.403150
238366  0.043203 -1.285451  -1.766684  0.756711 -1.765722  -3.263007

              V11        V12       V13        V14       V15        V16  \
9179    10.446847 -15.479052  0.734442 -13.883779  0.821440 -11.911483
215132   4.229154  -5.292314 -0.888087  -7.672250  0.547571  -4.307060
275992   1.032016  -0.722023 -1.533240   0.334119  0.297479  -0.429392
74507    5.705206  -8.640746 -1.602925  -9.466139  0.137324  -7.303243
238366   3.592797  -2.772349 -0.074534  -6.281094  0.165978  -2.679171

              V17       V18       V19       V20       V21       V22       V23  \
9179   -18.103004 -6.837835  3.126929  1.191444  2.002883  0.351102  0.795255
215132  -5.701174 -1.772803 -0.193132  2.230735  1.441622  0.895528  1.385511
275992  -0.824644  0.489668  0.873344  0.033804 -0.315105  0.575520  0.490842
74507  -12.448039 -4.332834  2.352030 -0.123085 -0.299847  0.610479  0.789023
238366  -1.385557  0.249057  2.353453  0.369663  0.397058  0.141165  0.171985
```

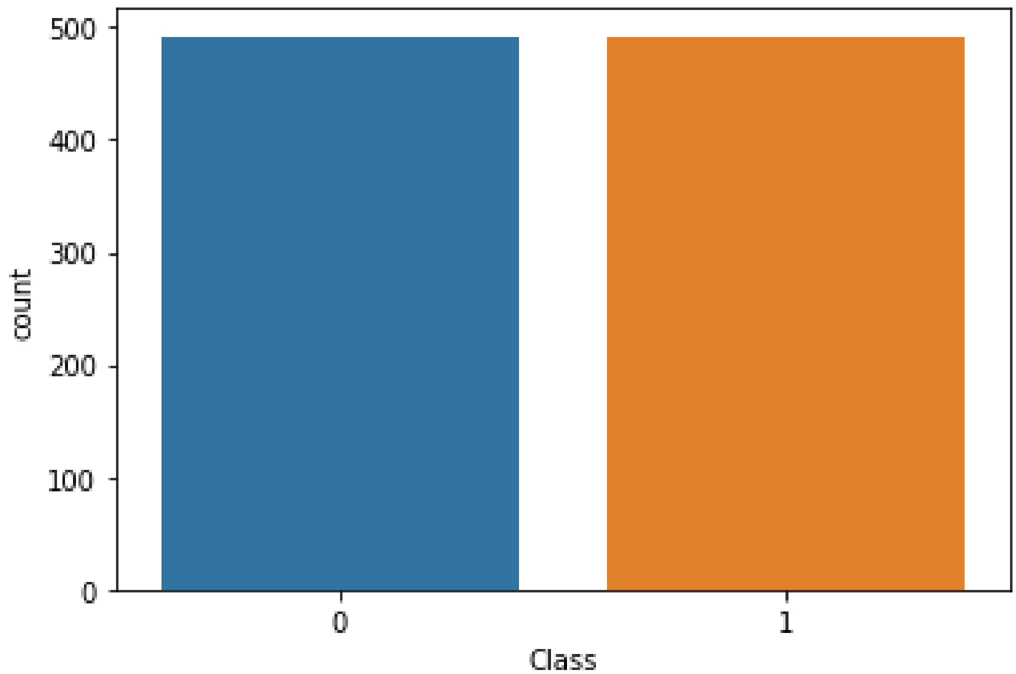
```
             V24       V25       V26       V27       V28  Class
9179   -0.778379 -1.646815  0.487539  1.427713  0.583172      1
215132 -2.028024  0.509131  0.172643  0.726781  0.234514      1
275992  0.756502 -0.142685 -0.602777  0.508712 -0.091646      1
74507  -0.564512  0.201196 -0.111225  1.144599  0.102280      1
238366  0.394274 -0.444642 -0.263189  0.304703 -0.044362      1
```

[16]:
```python
normal_distributed_df = pd.concat([fraud_df, non_fraud_df])
# Shuffle dataframe rows
new_df = normal_distributed_df.sample(frac=1, random_state=42)

new_df.head()
```

[16]:
```
        scaled_amount  scaled_time        V1        V2        V3        V4  \
132973       0.741703    -0.052844 -5.659842 -6.318881  0.877500  2.528836
231978      -0.195626     0.731987 -2.064240  2.629739 -0.748406  0.694992
47020        1.294907    -0.489338  0.990268 -0.063841 -0.399402  1.091776
252124      -0.296653     0.833774 -1.928613  4.601506 -7.124053  5.716088
6971        24.979809    -0.888497 -3.499108  0.258555 -4.489558  4.853894

              V5        V6        V7        V8        V9       V10       V11  \
132973  4.084124 -3.350876 -3.092932  1.075469 -0.107346 -0.232361 -1.292089
231978  0.418178  1.392520 -1.697801 -6.333065  1.724184 -0.887242 -1.594258
47020   0.197857 -0.179447  0.404442  0.021850 -0.345870  0.222890  0.969673
252124  1.026579 -3.189073 -2.261897  1.185096 -4.441942 -6.646154  3.827868
6971   -6.974522  3.628382  5.431271 -1.946734 -0.775680 -1.987773  4.690396

             V12       V13        V14       V15       V16       V17       V18  \
132973 -0.262224 -1.255812   0.745268 -0.257177  0.801618 -0.086481 -0.417798
231978 -0.338775 -0.978065  -3.688826 -1.487083  0.526946  2.347023  1.691220
47020  -0.018758 -1.685849   1.120864  0.414373 -0.082660 -0.321059  0.001748
252124 -6.518649  0.251137 -12.456706 -0.649166 -1.283145 -2.718560 -0.085466
6971   -6.998042  1.454012  -3.738023  0.317742 -2.013543 -5.136135 -1.183822

             V19       V20       V21       V22       V23       V24       V25  \
132973 -1.947073  1.659535  0.770734 -0.414649  0.214456  0.115022 -0.967188
231978 -0.736111 -1.424486  6.215514 -1.276909  0.459861 -1.051685  0.209178
47020  -0.277157 -0.042541  0.109229  0.029449 -0.245834 -0.328147  0.689290
252124 -2.097385  0.328796  0.602291 -0.541287 -0.354639 -0.701492 -0.030973
6971    1.663394 -3.042626 -1.052368  0.204817 -2.119007  0.170279 -0.393844

             V26       V27       V28  Class
132973  0.582231  0.077374 -0.929111      0
231978 -0.319859  0.015434 -0.050117      1
47020  -0.254508 -0.029366  0.010297      0
252124  0.034070  0.573393  0.294686      1
6971    0.296367  1.985913 -0.900452      1
```

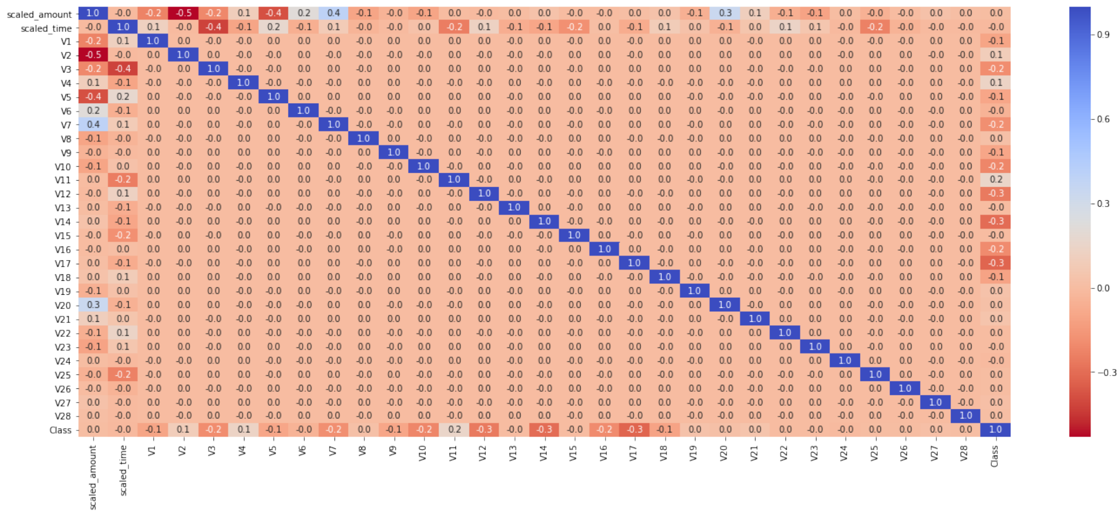
[17]:
```
sns.countplot(new_df['Class'])
```

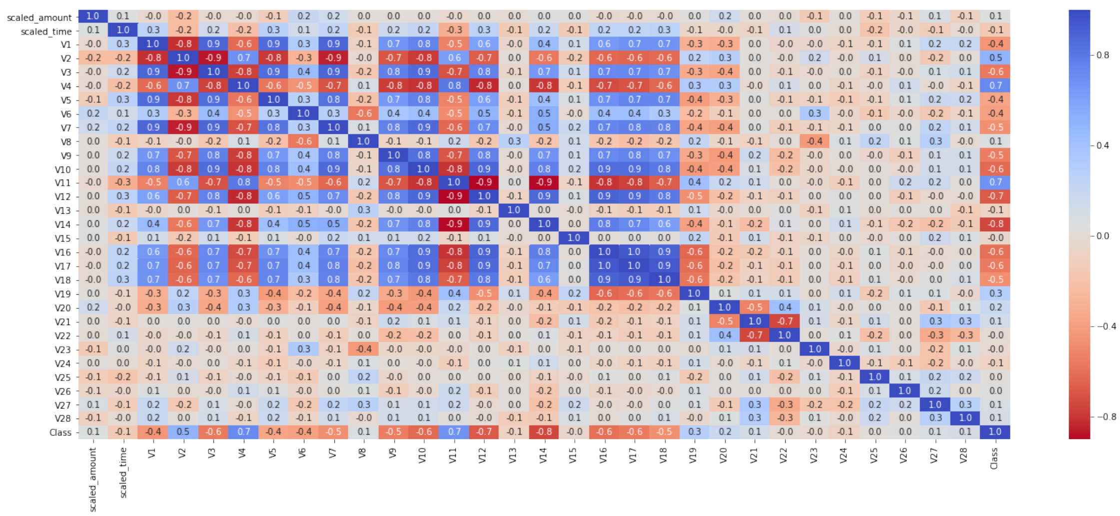
[17]: <matplotlib.axes._subplots.AxesSubplot at 0x1195d1278>



[18]:
```
f, (ax1, ax2) = plt.subplots(2, 1, figsize=(24,20))

corr = df.corr()
sns.heatmap(corr, ax = ax1, cmap='coolwarm_r', annot = True, fmt = '.1f')

new_corr = new_df.corr()
sns.heatmap(new_corr, ax = ax2, cmap='coolwarm_r', annot = True, fmt = '.1f')
```

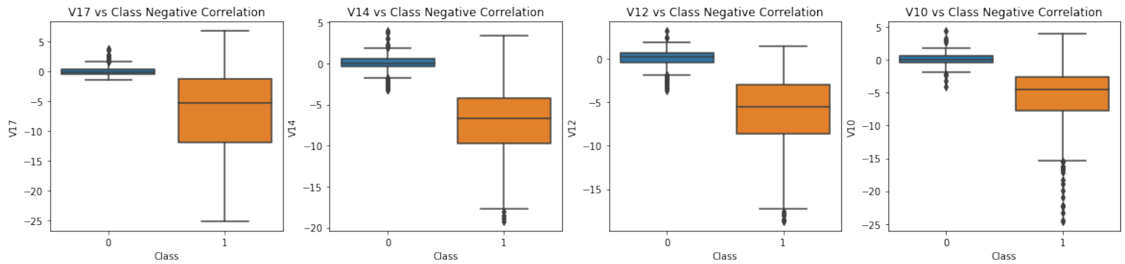
[18]: <matplotlib.axes._subplots.AxesSubplot at 0x11ba78208>



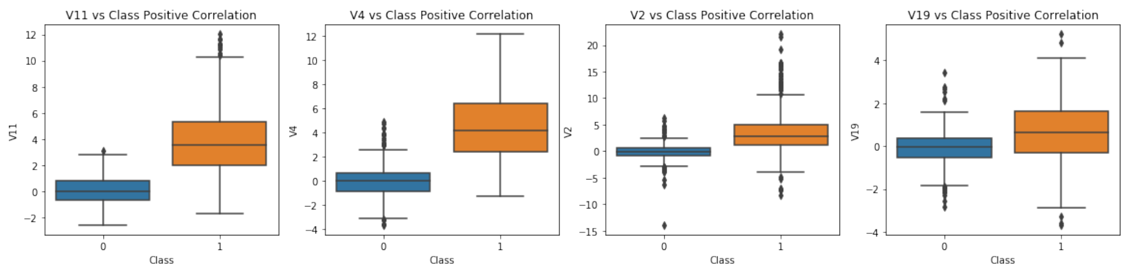


[19]:
```
f, axes = plt.subplots(1, 4, figsize = (20, 4))
sns.boxplot(x = 'Class', y = 'V17', data = new_df, ax = axes[0])
axes[0].set_title('V17 vs Class Negative Correlation')

sns.boxplot(x = 'Class', y = 'V14', data = new_df, ax = axes[1])
axes[1].set_title('V14 vs Class Negative Correlation')

sns.boxplot(x = 'Class', y = 'V12', data = new_df, ax = axes[2])
axes[2].set_title('V12 vs Class Negative Correlation')

sns.boxplot(x = 'Class', y = 'V10', data = new_df, ax = axes[3])
axes[3].set_title('V10 vs Class Negative Correlation')
```

[19]:
```
Text(0.5, 1.0, 'V10 vs Class Negative Correlation')
```



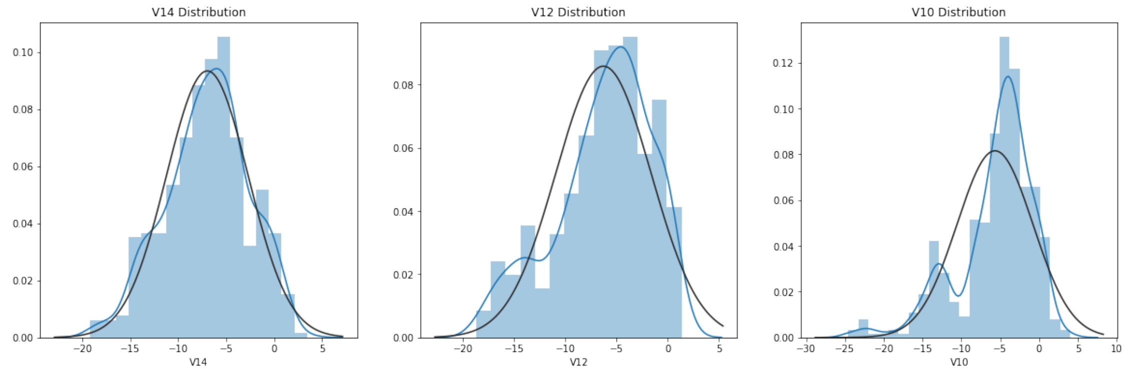
[20]:
```
f, axes = plt.subplots(ncols=4, figsize=(20,4))

# Positive correlations (The higher the feature the probability increases that
↪it will be a fraud transaction)
sns.boxplot(x="Class", y="V11", data=new_df, ax=axes[0])
axes[0].set_title('V11 vs Class Positive Correlation')

sns.boxplot(x="Class", y="V4", data=new_df, ax=axes[1])
axes[1].set_title('V4 vs Class Positive Correlation')


sns.boxplot(x="Class", y="V2", data=new_df, ax=axes[2])
axes[2].set_title('V2 vs Class Positive Correlation')


sns.boxplot(x="Class", y="V19", data=new_df, ax=axes[3])
axes[3].set_title('V19 vs Class Positive Correlation')

plt.show()
```



[21]:
```
# check anormaly
# check normality

from scipy.stats import norm
f, (ax1, ax2, ax3) = plt.subplots(ncols = 3, figsize = (20, 6))

sns.distplot(new_df[new_df['Class']==1]['V14'], fit = norm, ax =ax1)
```

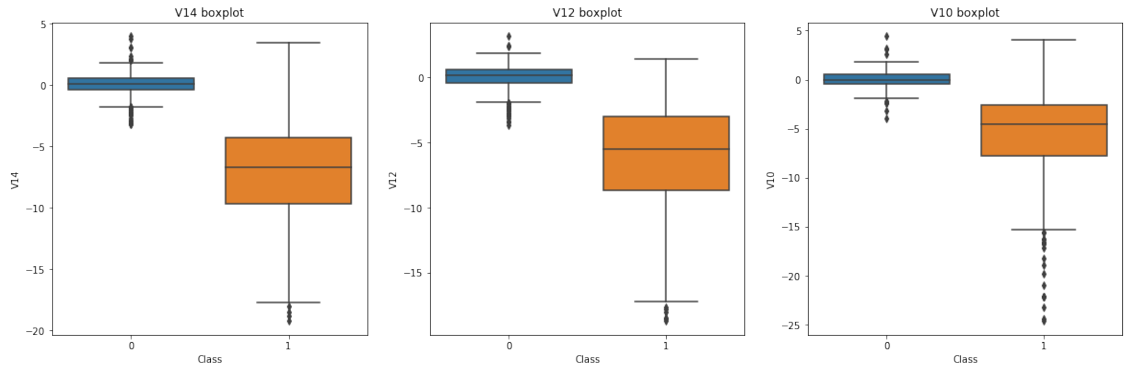
```python
ax1.set_title('V14 Distribution')

sns.distplot(new_df[new_df['Class']==1]['V12'], fit = norm, ax =ax2)
ax2.set_title('V12 Distribution')

sns.distplot(new_df[new_df['Class']==1]['V10'], fit = norm, ax =ax3)
ax3.set_title('V10 Distribution')
```

[21]: Text(0.5, 1.0, 'V10 Distribution')



[22]:
```python
f,(ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(20,6))

sns.boxplot(x="Class", y="V14", data=new_df,ax=ax1)
ax1.set_title('V14 boxplot')

sns.boxplot(x="Class", y="V12", data=new_df,ax=ax2)
ax2.set_title('V12 boxplot')

sns.boxplot(x="Class", y="V10", data=new_df,ax=ax3)
ax3.set_title('V10 boxplot')
```

[22]: Text(0.5, 1.0, 'V10 boxplot')



[23]:
```
v14_fraud = new_df['V14'].loc[new_df['Class'] == 1].values
q25, q75 = np.percentile(v14_fraud, 25), np.percentile(v14_fraud, 75)
v14_iqr = q75-q25
v14_cut_off = v14_iqr*1.5
v14_lower, v14_upper = q25 - v14_cut_off, q75 + v14_cut_off
outliers = [x for x in v14_fraud if x < v14_lower or x > v14_upper]

new_df = new_df.loc[(new_df['V14']>= v14_lower) & (new_df['V14']<= v14_upper)]
```

[24]:
```
v12_fraud = new_df['V12'].loc[new_df['Class'] == 1].values
q25, q75 = np.percentile(v12_fraud, 25), np.percentile(v12_fraud, 75)
v12_iqr = q75 - q25

v12_cut_off = v12_iqr * 1.5
v12_lower, v12_upper = q25 - v12_cut_off, q75 + v12_cut_off
print('V12 Lower: {}'.format(v12_lower))
print('V12 Upper: {}'.format(v12_upper))
outliers = [x for x in v12_fraud if x < v12_lower or x > v12_upper]
print('V12 outliers: {}'.format(outliers))
print('Feature V12 Outliers for Fraud Cases: {}'.format(len(outliers)))
new_df = new_df.drop(new_df[(new_df['V12'] > v12_upper) | (new_df['V12'] <␣
↪v12_lower)].index)
print('Number of Instances after outliers removal: {}'.format(len(new_df)))
print('----' * 44)
```

```
V12 Lower: -17.3430371579634
V12 Upper: 5.776973384895937
V12 outliers: [-18.683714633344298, -18.047596570821604, -18.553697009645802,
-18.4311310279993]
Feature V12 Outliers for Fraud Cases: 4
Number of Instances after outliers removal: 975
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------
```

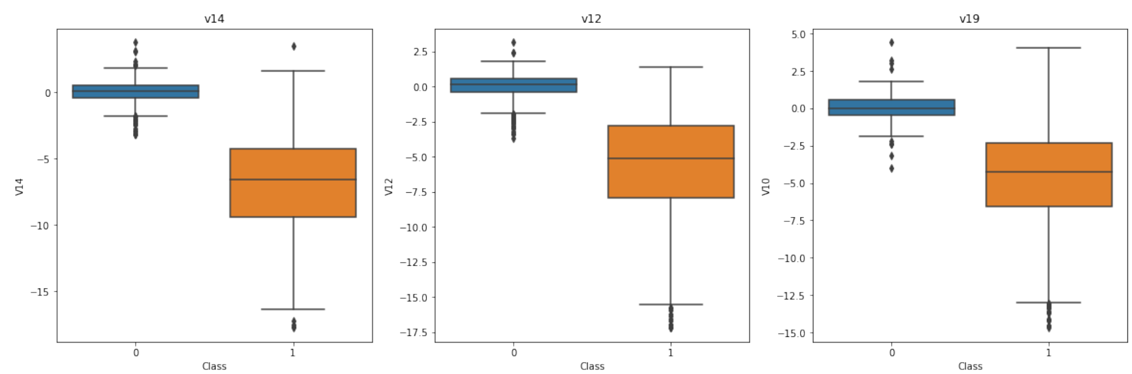
[25]:
```
v10_fraud = new_df['V10'].loc[new_df['Class'] == 1].values
q25, q75 = np.percentile(v10_fraud, 25), np.percentile(v10_fraud, 75)
v10_iqr = q75 - q25

v10_cut_off = v10_iqr * 1.5
v10_lower, v10_upper = q25 - v10_cut_off, q75 + v10_cut_off
print('V10 Lower: {}'.format(v10_lower))
print('V10 Upper: {}'.format(v10_upper))
outliers = [x for x in v10_fraud if x < v10_lower or x > v10_upper]
print('V10 outliers: {}'.format(outliers))
print('Feature V10 Outliers for Fraud Cases: {}'.format(len(outliers)))
new_df = new_df.drop(new_df[(new_df['V10'] > v10_upper) | (new_df['V10'] <␣
↪v10_lower)].index)
```

```
print('Number of Instances after outliers removal: {}'.format(len(new_df)))
```

```
V10 Lower: -14.89885463232024
V10 Upper: 4.920334958342141
V10 outliers: [-22.1870885620007, -20.949191554361104, -16.6496281595399,
-15.563791338730098, -17.141513641289198, -16.7460441053944,
-15.346098846877501, -18.2711681738888, -24.5882624372475, -18.9132433348732,
-15.2399619587112, -15.124162814494698, -19.836148851696, -15.2318333653018,
-14.9246547735487, -16.3035376590131, -15.2399619587112, -22.1870885620007,
-15.1237521803455, -14.9246547735487, -15.563791338730098, -16.2556117491401,
-23.2282548357516, -22.1870885620007, -22.1870885620007, -24.403184969972802,
-16.6011969664137]
Feature V10 Outliers for Fraud Cases: 27
Number of Instances after outliers removal: 948
```

[26]:
```
fg, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize = (20, 6))
sns.boxplot(x = "Class", y = 'V14', data = new_df, ax = ax1)
ax1.set_title('v14')

sns.boxplot(x = "Class", y = 'V12', data = new_df, ax = ax2)
ax2.set_title('v12')

sns.boxplot(x = "Class", y = 'V10', data = new_df, ax = ax3)
ax3.set_title('v19')
```

[26]: Text(0.5, 1.0, 'v19')



[27]:
```
X = new_df.drop('Class', axis=1)
y = new_df['Class']
```

[28]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
↪random_state = 42)
```

```
X_train = X_train.values
X_test = X_test.values
y_train = y_train.values
y_test = y_test.values
```

[35]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import collections
from sklearn.model_selection import cross_val_score

classifiers = {
    'LogisticRegression':LogisticRegression(),
    'KNearest' : KNeighborsClassifier(),
    'Support Vector Classifier': SVC(),
    'DecisionTreeClassifier' : DecisionTreeClassifier()
}

for key, classifier in classifiers.items():
    classifier.fit(X_train, y_train)
    training_score = cross_val_score(classifier, X_train, y_train, cv =5)
    print(key, ':', round(training_score.mean(),3)*100)
```

```
LogisticRegression : 92.9
KNearest : 92.5
Support Vector Classifier : 93.30000000000001
DecisionTreeClassifier : 90.10000000000001
```

[ ]:
```

```

[44]:
```
a = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
b = np.array([0, 0, 1, 1])
skf = StratifiedKFold(n_splits=2)

#StratifiedKFold(n_splits=2, random_state=None, shuffle=False)
for train_index, test_index in skf.split(a, b):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = a[train_index], a[test_index]
    y_train, y_test = b[train_index], b[test_index]
    print(X_train, X_test)
```

```
TRAIN: [1 3] TEST: [0 2]
[[3 4]
 [7 8]] [[1 2]
 [5 6]]
TRAIN: [0 2] TEST: [1 3]
```

```
[[1 2]
 [5 6]] [[3 4]
 [7 8]]
```

[ ]:

```

```