Assignment1

September 28, 2018

```
In [1]: import pandas as pd
        import numpy as np
        %matplotlib inline
        import matplotlib.pyplot as plt
        data = pd.read_csv("creditcard.csv")
        data.describe()
```

```
/Users/kshitijdani/anaconda3/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2785:
  interactivity=interactivity, compiler=compiler, result=result)
```

```
Out[1]:                 Time             V7            V8             V9  \
        count  284909.000000  284909.000000  2.849090e+05  284909.000000
        mean    94826.986259       0.000171 -9.434918e-07      -0.000010
        std     47485.356111       1.238456  1.194284e+00       1.098634
        min         0.000000     -43.557242 -7.321672e+01     -13.434066
        25%     54215.000000      -0.554068 -2.086343e-01      -0.643099
        50%     84728.000000       0.040103  2.235024e-02      -0.051416
        75%    139310.000000       0.570497  3.273893e-01       0.597165
        max    172792.000000     120.589494  2.000721e+01      15.594995

                         V10            V11            V12            V13  \
        count  284909.000000  284909.000000  284909.000000  284909.000000
        mean        0.000002      -0.000179      -0.000015      -0.000082
        std         1.088858       1.020704       0.999100       0.995248
        min       -24.588262      -4.797473     -18.683715      -5.791881
        25%        -0.535465      -0.762624      -0.405605      -0.648564
        50%        -0.092926      -0.032868       0.139926      -0.013625
        75%         0.453998       0.739334       0.618108       0.662416
        max        23.745136      12.018913       7.848392       7.126883

                         V14            V15      ...                  V19  \
        count  284909.000000  284909.000000      ...        284909.000000
        mean       -0.000065      -0.000087      ...            -0.000051
        std         0.958601       0.915340      ...             0.814020
        min       -19.214325      -4.498945      ...            -7.213527
        25%        -0.425604      -0.582953      ...            -0.456307
```

```
50%         0.050601       0.047875       ...            0.003738
75%         0.493119       0.648726       ...            0.458867
max        10.526766       8.877742       ...            5.591971

                 V20            V21            V22            V23  \
count  284909.000000  284909.000000  284909.000000  284909.000000
mean       -0.000043       0.000012       0.000078       0.000004
std         0.771120       0.734477       0.725728       0.624450
min       -54.497720     -34.830382     -10.933144     -44.807735
25%        -0.211761      -0.228390      -0.542335      -0.161859
50%        -0.062497      -0.029435       0.006832      -0.011192
75%         0.133017       0.186431       0.528673       0.147643
max        39.420904      27.202839      10.503090      22.528412

                 V26            V27            V28         Amount  \
count  284909.000000  284909.000000  284909.000000  284909.000000
mean        0.000003      -0.000013      -0.000024      88.377411
std         0.482240       0.403712       0.330074     250.395279
min        -2.604551     -22.565679     -15.430084       0.000000
25%        -0.326979      -0.070856      -0.052972       5.600000
50%        -0.052114       0.001341       0.011233      22.000000
75%         0.240944       0.091033       0.078281      77.180000
max         3.517346      31.612198      33.847808   25691.160000

               Class
count  284909.000000
mean        0.001727
std         0.041520
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 23 columns]
```

In [2]:
```
data.dropna(inplace=True)
data=data.drop_duplicates()
data = data[data.V1 != '%']
data = data[data.V2 != '%']
data = data[data.V3 != '%']
data = data[data.V4 != '%']
data = data[data.V5 != '%']
data = data[data.V6 != '%']
data = data[data.V24 != '%']
data = data[data.V25 != '%']
data= data[data["V1"]!='?']
data= data[data["V2"]!='?']
```

```
data= data[data["V3"]!='?']
data= data[data["V4"]!='?']
data= data[data["V5"]!='?']
data= data[data["V6"]!='?']
data= data[data["V24"]!='?']
data= data[data["V25"]!='?']
data= data[data["V1"]!='.']
data= data[data["V2"]!='.']
data= data[data["V3"]!='.']
data= data[data["V4"]!='.']
data= data[data["V5"]!='.']
data= data[data["V6"]!='.']
data= data[data["V24"]!='.']
data= data[data["V25"]!='.']
data= data[data["V1"]!=']']
data= data[data["V2"]!=']']
data= data[data["V3"]!=']']
data= data[data["V4"]!=']']
data= data[data["V5"]!=']']
data= data[data["V6"]!=']']
data= data[data["V24"]!=']']
data= data[data["V25"]!=']']
data= data[data["V1"]!='[']
data= data[data["V2"]!='[']
data= data[data["V3"]!='[']
data= data[data["V4"]!='[']
data= data[data["V5"]!='[']
data= data[data["V6"]!='[']
data= data[data["V24"]!='[']
data= data[data["V25"]!='[']
data= data[data["V1"]!=';']
data= data[data["V2"]!=';']
data= data[data["V3"]!=';']
data= data[data["V4"]!=';']
data= data[data["V5"]!=';']
data= data[data["V6"]!=';']
data= data[data["V24"]!=';']
data= data[data["V25"]!=';']
data= data[data["V1"]!=',']
data= data[data["V2"]!=',']
data= data[data["V3"]!=',']
data= data[data["V4"]!=',']
data= data[data["V5"]!=',']
data= data[data["V6"]!=',']
data= data[data["V24"]!=',']
data= data[data["V25"]!=',']
data= data[data["V24"]!='/']
data= data[data["V25"]!='/']
```

```
        data= data[data["V3"]!='\'']
        data= data[data["V4"]!='\'']
        data= data[data["V5"]!='\'']
        data= data[data["V6"]!='\'']
        data= data[data["V24"]!='\'']
        data= data[data["V25"]!='\'']
        data.info()
```

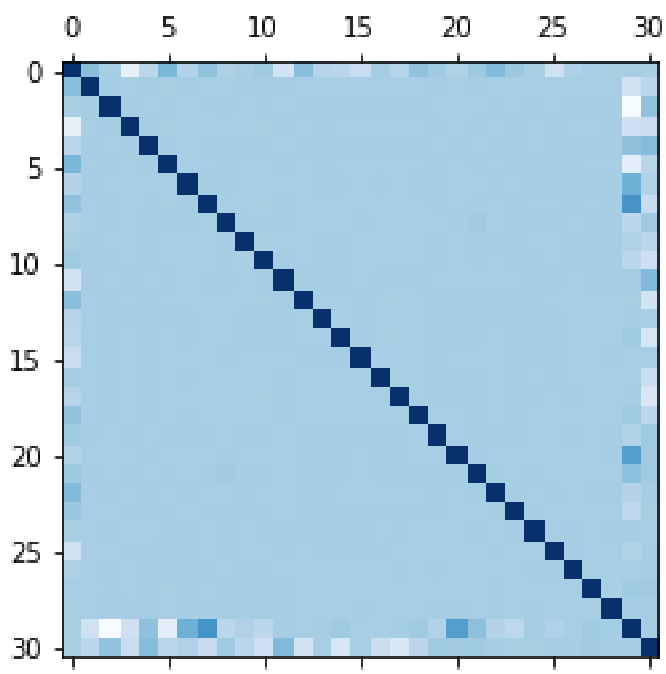
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 283798 entries, 0 to 284907
Data columns (total 31 columns):
Time      283798 non-null int64
V1        283798 non-null object
V2        283798 non-null object
V3        283798 non-null object
V4        283798 non-null object
V5        283798 non-null object
V6        283798 non-null object
V7        283798 non-null float64
V8        283798 non-null float64
V9        283798 non-null float64
V10       283798 non-null float64
V11       283798 non-null float64
V12       283798 non-null float64
V13       283798 non-null float64
V14       283798 non-null float64
V15       283798 non-null float64
V16       283798 non-null float64
V17       283798 non-null float64
V18       283798 non-null float64
V19       283798 non-null float64
V20       283798 non-null float64
V21       283798 non-null float64
V22       283798 non-null float64
V23       283798 non-null float64
V24       283798 non-null object
V25       283798 non-null object
V26       283798 non-null float64
V27       283798 non-null float64
V28       283798 non-null float64
Amount    283798 non-null float64
Class     283798 non-null int64
dtypes: float64(21), int64(2), object(8)
memory usage: 69.3+ MB
```

```
In [3]: data.V1 = data.V1.astype(float)
        data.V2 = data.V2.astype(float)
```

```
data.V3 = data.V3.astype(float)
data.V4 = data.V4.astype(float)
data.V5 = data.V5.astype(float)
data.V6 = data.V6.astype(float)
data.V24 = data.V24.astype(float)
data.V25 = data.V25.astype(float)
```

In [4]:
```
data.corr()
plt.matshow(data.corr(),cmap='Blues')
```

Out[4]: <matplotlib.image.AxesImage at 0x106dbf470>



In [5]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

import seaborn as sns

from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from sklearn.cluster import Birch
from sklearn.cluster import AgglomerativeClustering as AC

from sklearn import preprocessing
from collections import Counter
```

```python
from sklearn.metrics import accuracy_score
from sklearn.metrics import matthews_corrcoef
from sklearn.metrics import mean_squared_error
```

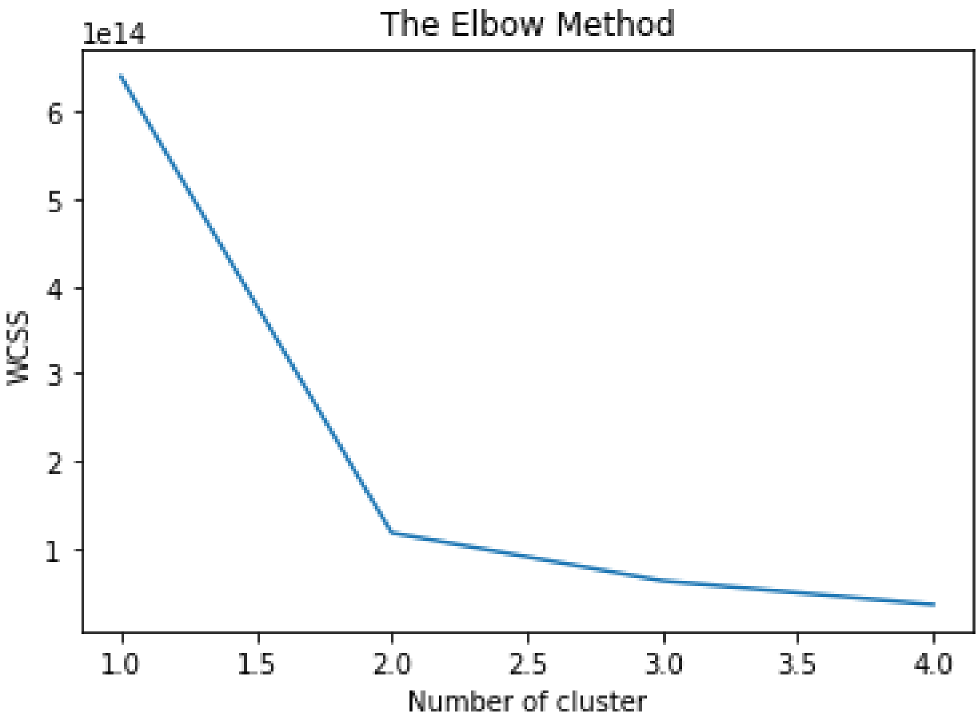
In [6]:
```python
Y = data.Class
pd.to_numeric(Y, downcast='signed')
#Storing all the class in a list of type int
```

Out[6]:
```
0         0
1         0
2         0
3         0
4         0
5         0
6         0
7         0
8         0
9         0
10        0
11        0
13        0
14        0
15        0
16        0
17        0
18        0
19        0
20        0
21        0
22        0
23        0
24        0
25        0
26        0
27        0
28        0
29        0
30        0
         ..
284869    0
284870    0
284871    0
284872    0
284874    0
284875    0
284876    0
284877    0
284878    0
```

```
284879    0
284881    0
284882    0
284883    0
284884    0
284886    0
284887    0
284888    0
284890    0
284891    0
284893    0
284895    0
284896    0
284898    0
284900    0
284901    0
284903    0
284904    0
284905    0
284906    0
284907    0
Name: Class, Length: 283798, dtype: int8
```

In [7]:
```
#K MEANS BEGINS
data = data.iloc[:,:30]
```

In [8]:
```
wcss = []
for i in range(1,5):
    kmeans = KMeans(n_clusters = i,init = 'k-means++',random_state = 0)
    kmeans.fit(data)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,5),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of cluster')
plt.ylabel('WCSS')
plt.show()
```




```
In [9]: kmeans = KMeans(n_clusters = 2,init = 'k-means++',random_state = 0)
        y2_kmeans = kmeans.fit_predict(data)
        Counter(y2_kmeans)
```

```
Out[9]: Counter({0: 152568, 1: 131230})
```

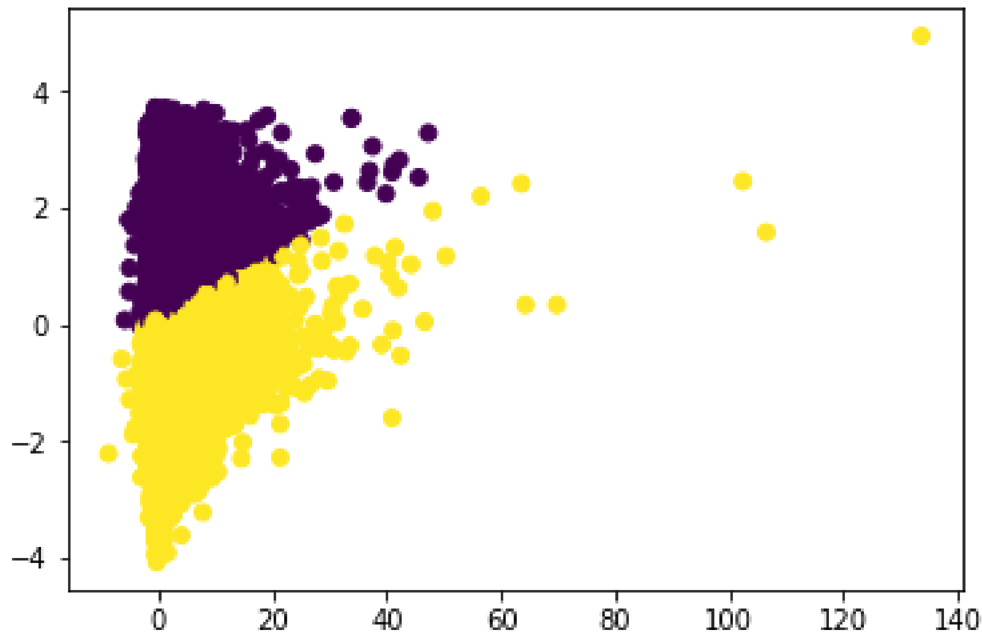
```
In [10]: acck2 = accuracy_score(Y,y2_kmeans)
         rootk2 = np.sqrt(mean_squared_error(Y,y2_kmeans))
         corrk2 = matthews_corrcoef(Y,y2_kmeans)
         print(acck2)
         print(rootk2)
         print(corrk2)
```

```
0.5370615719631568
0.680395787785935
-0.010000019791830071
```

```
In [11]: #Running PCA on the data
         data_Matrix = data.iloc[:,:30].values #storing dataframe values into a Matrix or ndarra
         pca = PCA(n_components=2)
         data_Matrix = StandardScaler().fit_transform(data_Matrix)
         principalComponents = pca.fit_transform(data_Matrix)
         pdf = pd.DataFrame(data = principalComponents, columns = ['pc 1', 'pc 2'])
```

In [12]:
```
#K means on the Data after further PCA
kmeans = KMeans(n_clusters = 2,init = 'k-means++',random_state = 0)
y2_kmeans_PCA = kmeans.fit_predict(pdf)
Counter(y2_kmeans_PCA)
%matplotlib inline
plt.figsize=(40,40)
plt.scatter(pdf.iloc[:,0],pdf.iloc[:,1],c=y2_kmeans_PCA)
```

Out[12]:
```
<matplotlib.collections.PathCollection at 0x107f71128>
```



In [13]:
```
acck2PCA = accuracy_score(Y,y2_kmeans_PCA)
rootk2PCA = np.sqrt(mean_squared_error(Y,y2_kmeans_PCA))
corrk2PCA = matthews_corrcoef(Y,y2_kmeans_PCA)
print(acck2PCA)
print(rootk2PCA)
print(corrk2PCA)
```

```
0.44954509897885114
0.7419264795255315
0.009269966655346039
```

In [14]:
```
#DBSCAN BEGINS
```

In [15]:
```
# trying with eps 1000
```

```
dbscan = DBSCAN(eps=1000, metric='euclidean', min_samples=2)
dbsc = dbscan.fit(data)
dbsc.labels_
Counter(dbsc.labels_)
```

```
Out[15]: Counter({0: 283710,
                  -1: 52,
                  1: 2,
                  2: 2,
                  3: 2,
                  4: 3,
                  5: 2,
                  6: 2,
                  7: 2,
                  8: 6,
                  9: 3,
                  10: 2,
                  11: 4,
                  12: 2,
                  13: 2,
                  14: 2})
```

```
In [16]: acc1 = accuracy_score(Y,dbsc.labels_)
         root1 = np.sqrt(mean_squared_error(Y,dbsc.labels_))
         corr1 = matthews_corrcoef(Y,dbsc.labels_)
         print("Accuracy of this DBSCAN eps=1000 is =",acc1)
         print("Root Mean Square of this DBSCAN is eps=1000 =",root1)
         print("Correlation of this DBSCAN eps= 1000 is =",corr1)
```

```
Accuracy of this DBSCAN is = 0.9980232418833114
Root Mean Square of this DBSCAN is = 0.10535389869725545
Correlation of this DBSCAN is = -0.0003679599397427888
```

```
In [17]: #trying with eps 3000
         dbscan = DBSCAN(eps=3000, metric='euclidean', min_samples=2)
         dbsc = dbscan.fit(data)
         Counter(dbsc.labels_)
```

```
Out[17]: Counter({0: 283789, -1: 9})
```

```
In [18]: acc2 = accuracy_score(Y,dbsc.labels_)
         root2 = np.sqrt(mean_squared_error(Y,dbsc.labels_))
         corr2 = matthews_corrcoef(Y,dbsc.labels_)
         print("Accuracy of this DBSCAN eps=3000 is =",acc2)
         print("Root Mean Square of this DBSCAN eps=3000 is =",root2)
         print("Correlation of this DBSCAN eps=3000 is =",corr2)
```

```
0.9983016088908307
0.041211540970572766
```

```
-0.0001150486698459505
```

```
In [19]: data_Matrix = pdf.values
         print(pdf)
```

```
            pc 1      pc 2
0       0.404136 -2.540137
1      -0.412908 -2.057401
2       1.822147 -2.519876
3       0.288260 -1.771123
4      -0.007495 -1.473696
5      -0.369992 -2.068312
6      -0.411072 -1.527528
7      -0.177335 -0.985514
8       0.051368 -1.197490
9      -0.378148 -1.827973
10     -0.365996 -2.102834
11     -0.348534 -0.546772
12      0.238508 -2.113637
13     -0.275389 -1.840064
14     -0.376412 -1.712603
15     -0.375988 -1.891926
16     -0.353394 -1.996895
17     -0.421530 -1.310044
18     -0.704072 -0.966131
19     -0.388322 -1.969127
20      0.832699 -2.029719
21     -0.253208 -2.275659
22     -0.411191 -1.893688
23     -0.186862 -2.226331
24     -0.582423 -1.594547
25     -0.364142 -1.723576
26     -0.200072 -1.820705
27     -0.338510 -1.782789
28     -0.188003 -1.948741
29     -0.346213 -2.088231
...          ...       ...
283768 -0.402112  1.291517
283769 -0.459158  1.209373
283770 -0.284059 -0.573600
283771 -0.483792  1.655096
283772 -0.567340  1.771319
283773  0.369267  1.058107
283774  0.320054 -0.011699
283775 -0.429496  1.048531
283776 -0.351298  1.404671
283777 -0.240683  1.245686
```

```
283778  0.233759  0.780762
283779  0.076234 -0.726941
283780 -0.599343  1.215895
283781  8.910680  2.463743
283782 -0.400134  0.315170
283783 -0.489503  0.078338
283784 -0.304837  1.025289
283785 -0.426268  1.019449
283786 -0.546534  0.832710
283787 -0.559449  0.832299
283788 -0.517698  1.590062
283789  0.144147  1.588167
283790 -0.294506  0.832103
283791  1.105910  2.814396
283792 -0.440424  0.766042
283793 -0.509090  1.033644
283794  4.033980  1.212895
283795 -0.081447  1.858988
283796 -0.414194  1.347461
283797 -0.526025  1.674971

[283798 rows x 2 columns]
```

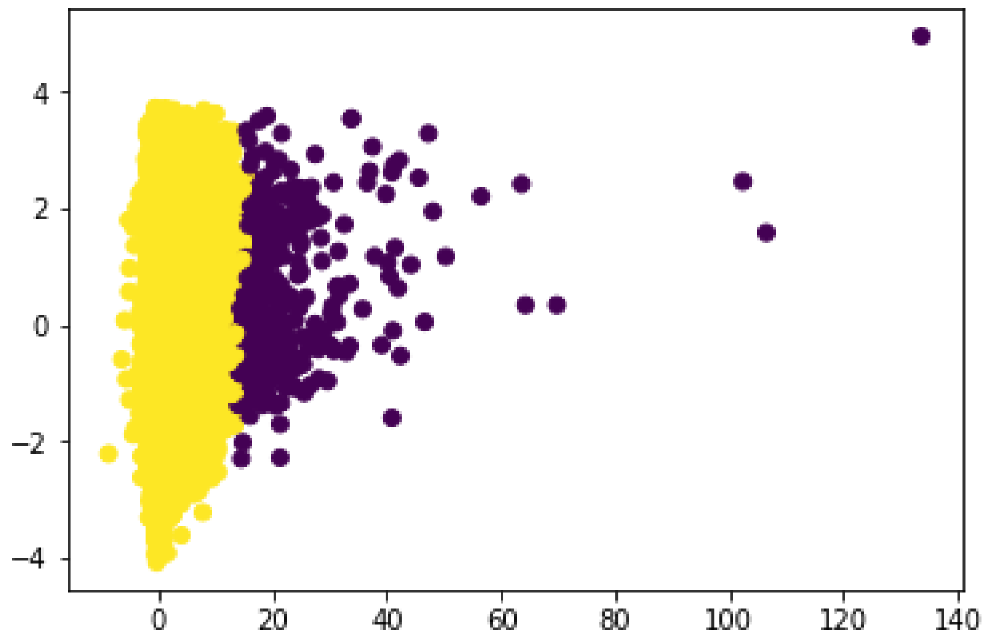
In [24]:
```
#Performing BIRCH CLUSTERING

BRC = Birch(branching_factor=50, n_clusters=2, threshold=0.5)
data_Matrix_predict = BRC.fit_predict(data_Matrix)
accBRC = accuracy_score(Y,data_Matrix_predict)
rootBRC = np.sqrt(mean_squared_error(Y,data_Matrix_predict))
corrBRC = matthews_corrcoef(Y,data_Matrix_predict)
```

In [23]:
```
print("Accuracy of Birch is = ",accBRC)
print("Root mean square of Birch is = " , rootBRC)
print("Correlation between class and predicted class is = ",corrBRC)
plt.scatter(pdf.iloc[:,0],pdf.iloc[:,1],c=data_Matrix_predict)
```

```
Accuracy of Birch is =  0.0030761316147400616
Root mean square of Birch is =  0.9984607495466509
Correlation between class and predicted class is =  0.001535040106108969
```

Out[23]:
```
<matplotlib.collections.PathCollection at 0x1a72073898>
```


4
2
0
−2
−4
0
20
40
60
80
100
120
140