In my second part of my project, I start to learn Primary Component Analysis (PCA) to find out more feature information. PCA can be used to visualization and speed up the algorithm.

In PCA\_1.ipynb, I use All.csv as my data. I Import PCA package and choose 40 components as an example. Using the PCA algorithm, I choose the first two features as principal component 1 and principal component 2. The reason I choose the first two features is that PCA algorithm sorts out all features due to their importance. Using two features, it is easy to plot the whole data in the two-dimensional graph via matplotlib.

In PCA\_2. Ipynb, I use PCA to reduce feature from 79 to 40 as an example on the same dataset. I use Random Forest as my classifier, and import accuracy\_score on sklearn.metrics package to calculate the accuracy. As a result, I can easily compare the difference between the accuracy without PCA and the other with PCA. The conclusion is obvious that the accuracy without PCA is higher than the other. Maybe PCA results in this consequence, but I am not sure, so I continue to compare the result using different amounts of features.

In comparePCA1.ipynb, I compare them with different components. I use variance from 0.1 to 0.9 and n\_components from 10 to 79. On the graph, it is quite clear to see that generally with more features, the accuracy is high. But also I find out even with PCA (79) all features, the accuracy is lower. I am not sure what happens to this, so I will keep looking for reasons.

In comparePCA2.ipynb, I import PCA with various components to plot the data graph. I find out these graphs are identical because of the PCA algorithm. The algorithm will calculate out the most important feature on first column and so on. Therefore, no matter how many components are, the first two features are the same that generate the same graphs.