Task 2 report

For task 2a, I combined the two csv “world” and “life” by country name to ensure any countries not presented in both world.csv and life.csv are discarded. Firstly, I changed all the missing value to nan value, and then I used the “life expectancy at birth” as class label, to separate 20 columns of data into training set and testing set with the ratio of 7:3, random state = 200. I then got the median of each column, replacing the nan values and testing the accuracy of 3 classification algorithms by scale all the data to standard scalar (removing the mean and scale to unit variance). As a result, the accuracy of decision tree is 0.709; the accuracy of k-nn (k=3) is 0.691; accuracy of k-nn (k=7) is 0.727. So, the best performance among those three algorithms is the k-nn (k=7) because it has the greatest accuracy value.

At the same time after I filled in all the median to their corresponding columns, I do calculate the mean and variance of each column (each feature), where the output “task 2a.csv” has 4 columns (name of feature, median, mean and variance respectively) and 20 rows (corresponding to the 20 features).

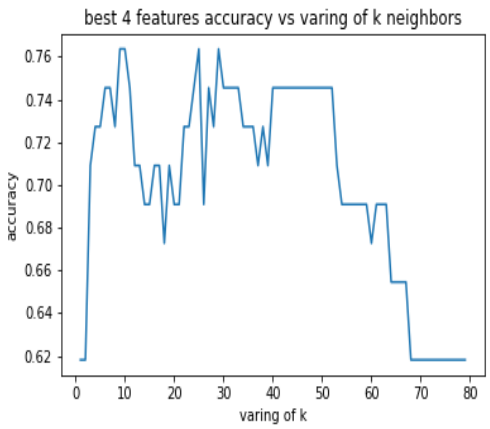
For task 2b, the preprocessing such as merge two csv by country code, replacing the missing value “..” to nan value and sorting data by country code methods are exactly the same as the task 2a. My class label is the “life expectancy at birth” and my data are the twenty features(columns). But this time, I do fill in the median to each column before splitting to two datasets because I need to create 190 more features for feature engineering. Basically, I multiply all the data in each column to all the rest 19 columns (apart itself) in order to create the new 190 features. There is one more feature by using the method of k-mean. The reason I chose the 3 clustering in k-mean is because the “life expectancy at birth” only contains three types of output data: low, median and high. In order to compare the accuracy of three methods, I split the training set and testing set by the ratio 7:3 with random state of 200 as well as scale them into standard scalar.

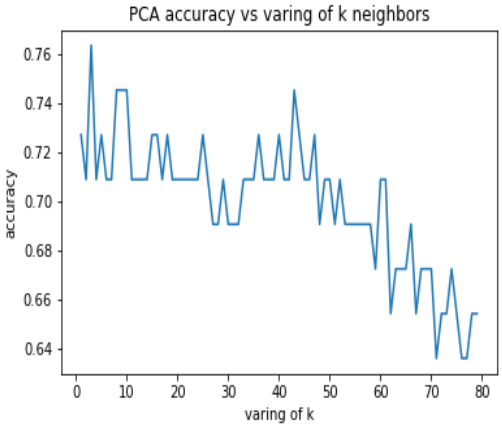
The first method is looking for the accuracy of the best 4 features. I used the function called “selectkbest” in sklearn library, so python would select the most significant 4 features to me. The accuracy of this feature engineering is 0.709, which is shown to be higher than the accuracy in task 2a as expected.

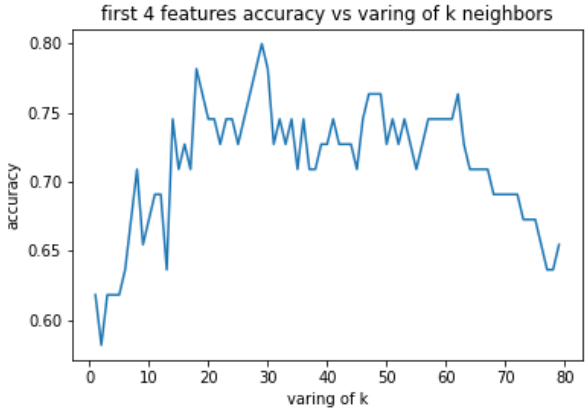
The second method PCA is a technique for reducing the dimensionality of datasets, increasing interpretability but at the same time minimizing information loss. Selecting the first 4 components which gives the accuracy of 0.764 – the greatest performance among all three methods. The reason of PCA performs better than feature engineering could be: PCA reduces the dimensions of data, eliminate the irrelevant features and reduce noises data, so first 4 components probably are the most efficient (useful) features. As the result, It is more useful to process the large dataset than feature engineering. However, feature engineering could perform better than PCA if choosing a better k value for knn.

The third method is selecting the first four features from the original dataset. This method is expecting of low accuracy because the first 4 features of dataset is probably a not good method of testing the accuracy because these features are not representative and we do not know how significant these features are. (They could be good features, but they also could be non-useful features). As a result, the accuracy is only 0.618, the lowest accuracy among three methods which is also expected.

The graphs below are applying the k-nearest neighbors to all three methods rather than only having 3 nearest neighbors. From these graphs, we can see there is significant decreasing trend when k is greater than 60, which gives a very important information that the large k may includes data from other classes, would reduce the accuracy of knn. In addition, it is shown that there are a few peaks between 10 to 30 of k, which means finding an appropriate value of k could increase the accuracy of feature selections.







In conclusion, I think it is pretty reliable to the classification model with the following two reasons:

Theoretically if k in knn is too small, its will be sensitive to noise points so the accuracy would be low; if k in knn is too big, the neighborhood may include from other classes, and the accuracy would be low also. This is perfectly shown by the graphs above.

The rank of accuracy of these three methods are all performed as expected.