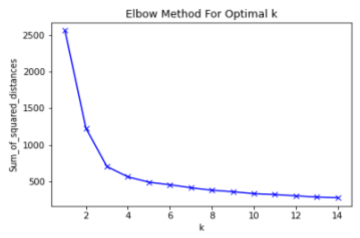
Task 2C

i. For k-nn, k=7 (accuracy of 0.727) performed best, and k=3 (accuracy of 0.673) weaker than k=7. The Decision Tree algorithm's performance was weaker than knn=7 but better than knn=3 on this dataset (accuracy of 0.709), hence k-nn performed better, with k=7. To perform the classification, 'life.csv' and 'world.csv' were both read and merged according to the country code, into a single dataframe, while ".." values were converted into NaN so that it can be processed accordingly. It's then sorted according to the alphabetical order of the country code. We extract the features and class label of the dataframe separately, then by using the *train_test_split* function from *sklearn*, the data was split into 70% training set and 30% testing set (using a random state of 200). Then, we impute the missing values in both the training and testing set using the median of the training set. The training and testing set were then normalised using the mean and variance from the training set, by *StandardScaler* from sklearn. We were able to obtain the mean and variance from the *StandardScaler*, which were written into '*task2a.csv*' along with the feature names and the median of each feature, with the format ('feature', 'median', 'mean', 'variance'). Finally, we use different algorithms from *sklearn* to train and test our data for classification. *DecisionTreeClassifier* was first used to build a decision tree according to the training set, then used on the testing set for classification, the accuracy score was then calculated. The second algorithm used was *KNeighborsClassifier*, which classifies according to k number of nearest data instances. The training set was first used to train the model, then used on the testing set for classification, and accuracy was calculated. This was used on both k=3 and k=7. As a result, three classification models were built. Higher k of k-nn performed better because it is less sensitive to the noises present in the dataset, also k-nn of k=7 performed better than the decision tree since the maximum depth allowed for the decision tree was only 3, thus making it less accurate.

ii. Firstly, the datasets from '*life.csv*' and '*world.csv*' were read in and merged with the same processes as Task 2a. The features and class labels were extracted, and training and testing sets were created using *train_test_split* from *sklearn*. Median imputation according to the training set was then performed on training and testing sets. We then normalised both sets using *StandardScaler* from *sklearn*. Feature engineering is then performed on the dataset. We first generated 1 cluster label for both testing and training sets (*f* clusterlabel) using k-means clustering trained by the training set (*KMeans* from *sklearn,*k=3), and we assigned each instance to the nearest cluster (0,1 or 2). Using *PolynomialFeatures* , 190 features were generated by all possible interaction term pairs from the original dataset. Combining the original features, feature by clustering, and the interaction term pair features, we calculate and obtain 4 features of highest Normalised Mutual Information using *normalized_mutual_info_score*. We then use the 4 features of this training set to train the k-nn classifier (*KNeighboursClassifier* from *sklearn*) and produce a prediction accuracy on the testing set of the same 4 features. The second type of feature engineering and selection was done via Principle Component Analysis (*PCA* from *sklearn*). The model was first created and trained by the training set (using n-components = 4) and transformed the training and testing sets into 4 sets of features for classification. K-nn was then performed and calculated the accuracy, similar to the first feature engineering approach. For the final approach, the first 4 features from the dataset were sliced and used to train and test the k-nn classifier as before, and accuracy scores were printed.

iii. To find the number of clusters to use for clustering labels (using k-means algorithm), the Elbow Method was adopted. After splitting the dataset, performing mean imputation and normalisation, the k-means algorithm (*KMeans* from *sklearn*) was used on the training set on range k from 1 to 15, and the sum of squared distances was calculated and plotted against k in *task2bgraph1.png*. Upon examining the plot, we could see that the 'elbow point' was located at k=3, where the rate of decrease of the sum of squared distances significantly dropped. Thus, we conclude that k=3 is the optimal value for the number of clusters used on the k-means algorithm.



task2bgraph1.png

iv. To select 4 features from the 211 features, the normalised mutual information of each feature was calculated (using *normalized_mutual_info_score* from *sklearn*), and 4 features with highest NMIs were then used. NMI was used since the chi-square feature on *sklearn* does not work well with negative numbers which were present in the dataset. NMI calculates the correlation of the feature and the class, so it is a good parameter to help us with selecting the best 4 features to predict the life expectancy.

v. Generally, the scores vary a bit when using different random states for splitting. However, the PCA method generally yielded the best results for classification using 3-NN (accuracy = 0.727). This is because the principal component analysis is effective with high dimensional data. It was able to extract the top 4 features from '*world.csv*' with the highest variability and reduced the dimension significantly while still providing good precision as from the original set. On the other hand, using interaction term pairs was less effective since multiplying different world development indicators was unlikely to generate any contextually useful information. Clustering labels using k-means were also unable to perform well on clusters of different sizes and density in the dataset. Hence, using the combination of both of these result labels gave the lowest accuracy scores for 3-NN classification (accuracy = 0.582) which was even lower than just simply taking the first 4 columns of the original feature, which at least still preserves some accuracy from the original dataset (accuracy = 0.636).

vi. We could try varying splitting proportions for out testing and training sets and choosing one that's not too high nor too low but still better for our accuracy. We could also use some domain knowledge on World Development Indicators and average lifespan to remove some similar or irrelevant features or features in a causal relationship, or select features of interest for classification. We could also then generate some linear weighted combination of features of interest (using domain knowledge) to help with the accuracy.

vii. I would say that my classification model is rather reliable. This is because the accuracy scores generated were quite satisfactory for most of the methods (PCA most reliable with 0.727 accuracy, and feature selection from 211 features least reliable with 0.582 accuracy). The data was split in a way that the proportion was reasonable. Median imputation was also performed based on the training data, and testing data was also normalised according to the training data. The testing sets were completely isolated from the training sets, meaning that the models were built solely on the training sets and completely independent of the testing sets. Informed choices on feature filtering were also made based on a standard parameter like NMI, and the elbow method on the number of clusters for k-means algorithm.