Report on Part 2A

Preprocessing: Renamed common columns in DataFrame created from *world.csv* to align with column names in DataFrame from *life.csv.* A new DataFrame is created by performing inner merge between the 2 aforementioned DataFrames. All rows with NaN values in the "Life Expectancy" column (target variable) is removed as they will not contribute to training and testing the models. This new, preprocessed DataFrame is split into train and test data sets according to project specifications. After splitting, all objects in both *scaled_X_train* and *X_test* are converted to floats and filled with NaN if they are string values.

Imputation: All NaN values in both *scaled_X_train* and *scaled_X_test* are imputed with median of every feature in *scaled_X_train.* When building a model, we technically do not have access to the testing data. Hence we cannot compute a median for *scaled_X_test* as this acess to testing data will falsely increase the accuracy of the model. Both *scaled_X_train* and *scaled_X_test* are standardised with mean and variance of every feature in *scaled_X_train* for the same reason as above.

Algorithm Evaluation: Ranking in accuracy yields 5-NN (~87%) > 10-NN (~85%) > Decision Tree (~72%). They are both supervised learning algorithms, but Decision Tree builds its model on the training set before classifying the test set, whereas the K-NN algorithm performs on-the-spot learning. The latter is computationally expensive but its ability to model with complicated decision boundaries contributes to its higher accuracy over Decision Tree. The larger k is, the smoother the decision boundary, which increases the chance of underfitting the data rather than overfitting. 10-NN may not be as accurate as 5-NN as some different outputs are classified as constant regardless of their feature values.

Report on Part 2B

Preprocessing + Imputation + Split: Perform the same procedure as Part 2A, only difference is to keep a copy of *X_train* with country code for visualisation of cluster label for each country in training set later.

Interaction Term Pairs: Use *PolynomialFeatures* module to fit and transform *scaled_X_train.* Only allow interaction up to and including degree 2, do not allow features to multiply themselves, and do not include constant terms that act as intercept in a linear model. This produces 190 features from the original 20.

Elbow Method (to determine *k* for clustering): An ideal number of clusters is determined to be 7 via. the Elbow Method. The Elbow Method fits the dataframe to a k-means model with 1-20 clusters each time and append its inertia attribute (sum of squared dsitance of each sample to their closest cluster center) to produce *fig 1.* where it is shown that any number between 6-8 should produce reasonable results (adding another cluster does not significantly improve the total compactness of clustering). Higher cluster numbers are forfeited as they will have little effect on the result, and also in the best interest of computational power.

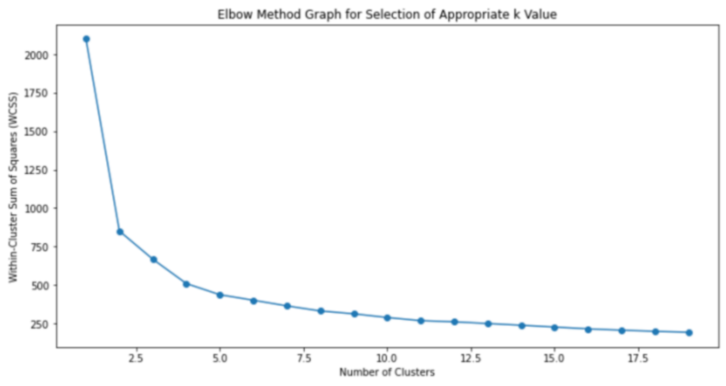

fig 1. Illustration of Elbow Method and optimal cluster number

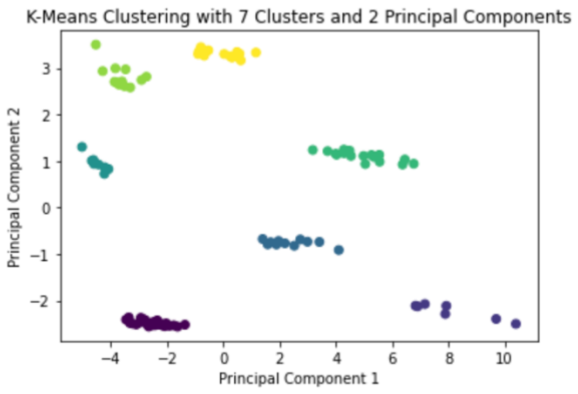

fig 2. K-Means Clustering with 7 Clusters and 2 PCA

K-Means Clustering: Use *KMeans* module with 7 clusters, fit this model to *scaled_X_train* and predict the closest cluster to each data point (country). Two representations of K-Means Clustering predictions are printed to *stdout*: A DataFrame of count of countries in each cluster (called *counts_df*), and a DataFrame of country codes and their cluster number (called *cluster_label_df*). The cluster graph (*fig. 2*) is sketched by collapsing the 20 original features to 2 dimensions that hold the most information, with *PCA* module. Note that initial cluster centroids are chosen at random each time the algorithm is run, hence the output graph may contain different clusters to *fig. 2*. but should still contain 6 clusters.

Feature Selection: Select 4 features from the generated dataset of 211 features in principled manner
Features are selected with the *ExtraTreeClassifier* module with a random state of 200 to ensure consistency between results. ExtraTrees (Extremely Randomised Trees) does not bootstrap observations and split nodes based on random splits in a random subset of the features, which introduces the randomness that reduces risk of overfitting the data. While constructing a forest of de-correlated Decision Trees for each feature, a Gini Importance value (normalised total reduction in Gini Index) is calculated. The features are ranked in descending order of their Gini Importance value and the highest 4 features are selected. Additionally, The top 10 most informative features are plotted in *fig. 3*. The top 4 features with highest *x-axis* value from *fig. 3* are selected for 5-NN classification later and printed to *stdout* with their relative importance.

Feature Selection: PCA

The Principal Components model from *PCA* module is fitted to and initially transforms *scaled_X_train*. This collapses the 211 features dataset to 4 dimentions that effectively combines all the features such that most valuable information of all features are still retained. Additionally, the 4 Principal Components are independent of one another. This output of 4 Principal Components based on *scaled_X_train* is saved in *principal_df* and printed to *stdout* for visualisation. It then transforms *scaled_X_test* with the 4 Principal Components observed from *scaled_X_train* data (as we technically do not have access to *scaled_X_test* data at the time of model-building) and produces *principal_test* DataFrame. Both *principal_df* and *principal_test* will be fed into 5-NN Classification algorithm later, former as input for training the 5-NN model and latter as input for testing it.

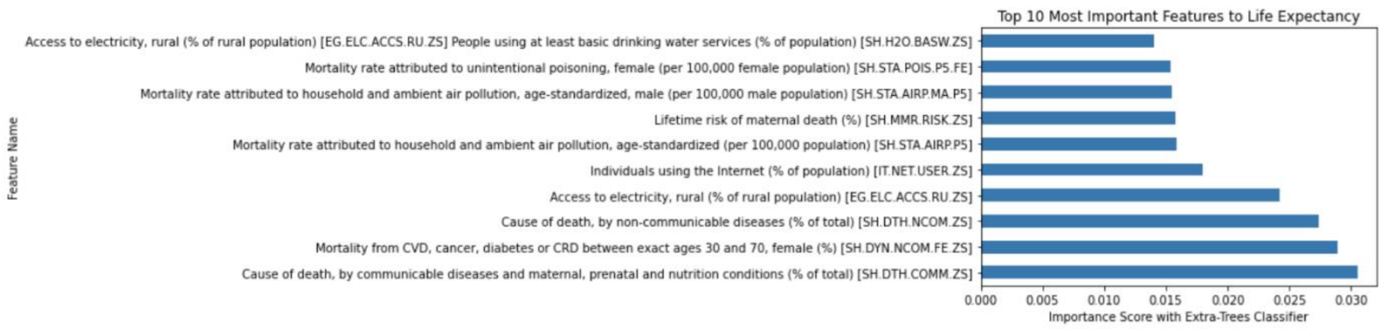

fig 3. Top 10 Most Important Features to Life Expectancy (Target Variable for 5-NN Classifier) According to Extra-Trees Classifier

Feature Selection: First 4 Features

The first 4 features of *scaled_X_train* (column D-G when *scaled_X_train* is opened in excel) is extracted as *first_4_features* DataFrame and will be fed into 5-NN Classification as training input later. The first 4 features of *scaled_X_test* (column D-G when *scaled_X_test* is opened in excel) is extracted as *first_4_features_test* DataFrame and will be fed into 5-NN Classification as testing input later.

Which of the 3 methods investigated in task 2B produced the best results for 5-NN classfication? Why?

Feature Engineering with selection via. Extra Tree Classifier produced the best accuracy (~87%) as interaction term pairs produced all possible quadratic combinations of features, and Extra Tree Classifier retained the 4 most important features out of 211 possibilities. Reducing from 211 features down to 4 dimensions still inevitably gave rise to a high loss of data despite PCA, hence its accuracy of (~76%). Taking the first 4 features produced the lowest accuracy (~72%) as expected, since this selection is not based on the relationship between the features and the target variable (Life Expectancy) but rather their column position in the raw file – which can be completely random and meaningless.

What other techniques can be implemented to improve classification accuracy with this data?

In terms of feature selection, the *SelectKBest* module from *sklearn* is an option that may improve the accuracy of classification by potentially selecting more influential features. In addition, for a numerical-input and categorical-response modelling problem as such, common techniques that may improve classification are mainly correlation-based, such as ANOVA correlation coefficient and Kendall's rank coefficient.

How reliable do you consider the classification model to be?

k-NN model performs well when the input data have the same scale, which points to a good degree of reliability for this particular model since its input data is cleaned, imputed and scaled. However, k-NN tends to struggle with a large number of inputs (like the data processed in this project)- especially in the case of a small *k* value where the model is high variance, hence a slight reduction in reliability. Ultimately, although k-NN is highly likely to overfit due to curse of dimensionality, several feature selection algorithms were implemented to reduce the dimensionality- hence the overall reliability is ideal.