# COMP20008 Elements of Data Processing: Project 2.2

Ming Hui Tan

## 1 Comparing Classification Algorithms

### 1.1 Data Preprocessing

We begin by reading in **world.csv** and **life.csv** as dataframes and inner join both dataframes on the column 'Country Code' to discard any countries not present in both CSV files. Then, we drop any duplicate columns after merging the data and sort the data in ascending alphabetical order by 'Country Code'. To avoid sampling bias, we split the dataset into a training set comprising 70% of the data and a test set comprising the remaining 30% using the `train_test_split` function with a random state of 200 for reproducibility purpose. We then determine the set of features (`X`) as well as the class feature of the data (`y`).

To deal with missing values which exist in the form of string in the data, we first replace them with 'NaN' and then we perform median imputation for each feature in both training and testing dataset by imputing the missing value in each feature with only the median of the feature from the training data. This is done to avoid data leakage in the form of train-test contamination. The rationale of using median instead of mean is to minimize the effect of outliers which are present in the dataset. Then, we normalize each feature in the data to have 0 mean and unit variance with the mean and variance obtained from the training data using the library function `StandardScaler`. Finally, we extract the median used for imputation for each feature, as well as the mean and variance used for scaling, all rounded to three decimal places into **task2a.csv**.

### 1.2 Predicting Life Expectancy at Birth Using Classification Algorithms

We first train the KNeighborClassifier (we will refer to as KNN from now) using the preprocessed training data with 3 and 7 neighbors by fitting the KNN with the training dataset. The rest of the parameters are kept at their defaults. Then, we repeat the same process with DecisionTreeClassifier using the same training dataset and keeping all parameters at their defaults except random state where we set as 200 and max depth at 3.

Finally, we test the classifiers by predicting the testing dataset using the classifiers that we trained with the training data. To compare the performance of the classification algorithms, we use sklearn built-in accuracy score metric. We do this by comparing the actual value against the predicted value of the class features of the testing dataset.

### 1.3 Result

Overall, the result that we obtained is as follow:

Accuracy of decision tree: 0.709
Accuracy of k-nn (k=3): 0.673
Accuracy of k-nn (k=7): 0.727

Here, we observe that KNN with 7 clusters trumps the other classifiers in terms of accuracy. KNN with 3 clusters could be highly susceptible to noise points compared to KNN with 7 clusters, therefore resulting in relatively lower accuracy. DecisionTreeClassifier also has slightly lower accuracy than KNN with 7 clusters since DecisionTreeClassifier is more sensitive to noise as well.

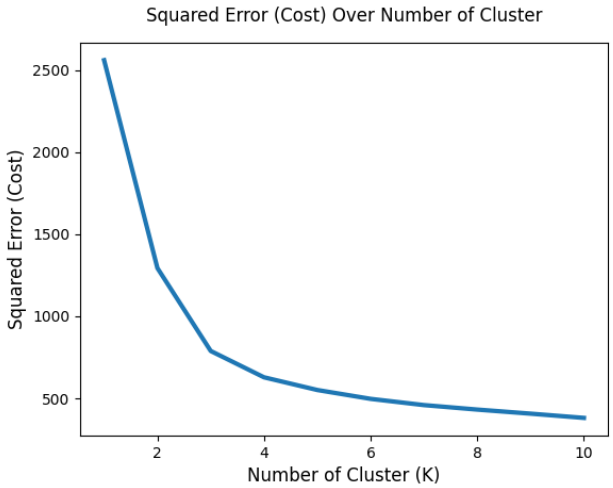
## 2 Feature Engineering and Selection

### 2.1 Data Preprocessing

For all three methods described below, we will perform the same data preprocessing steps as described in section 1.1 above excluding the normalization step to control all other variables so that we have the same dataset which allows us to determine if our new methods can boost the accuracy of the model.

For the first method (feature engineering), we first create interaction term pairs by multiplying two different features in both training and testing data. Since there are 20 features originally, generating features using this method

will yield us 190 additional features. We also use their abbreviations joined by '*' as the column name for the new columns. Apart from feature generation, we will also fit K-means clustering to the original imputed and scaled training data (the training data with 20 columns) and then use the resulting cluster labels of each data point as the values for a new feature called 'Cluster Label'. We then predict the 'Cluster Label' of the test data using the pre-fitted K-means clustering algorithm. To select the optimum value for the number of cluster for K-means algorithm, we use the **Elbow Method** where we plot the squared error of a number of clusters (which is represented by the inertia of the K-means after fitting) and obtain the number of clusters at which the curve begins to plateau. The graph that we obtain is as following:




Here, we can see that the point where the graph begins to plateau is at K = 3. Thus, by the elbow method, the optimal number of cluster in this case should be 3. In total, we have a dataset with 211 features (20 original features, 190 features generated by interaction term pair and 1 feature generated by clustering). Then, we normalize both the training and testing data that we engineered to mean 0 and unit variance with mean and variance from the training data using `StandardScaler`. Finally, we select the top 4 features with the best ANOVA F-values (we use this method instead of chi squared because the features in the data are continuous while the target feature is categorical and that there are negative values in the features after scaling) from the 211 features and use this as our dataset for this method.

As for the second method (PCA), we first normalized the preprocessed data since our features values are not uniform (eg. some feature has value from 0-100 while some has value from 1000-10000). Then, we simply take the first four principal components of the dataset by fitting the scaled training dataset with the PCA algorithm and then transform the testing dataset using the pre-fitted PCA algorithm. We end up with a dataset with four features comprise of the four principal components as follow.

| | pc1 | pc2 | pc3 | pc4 |
|---|---|---|---|---|
| 0 | 2.273725 | 0.105908 | -0.393829 | 0.770467 |
| 1 | -2.606225 | -1.201405 | -0.210355 | -0.166186 |
| 2 | -3.307587 | 0.460093 | -1.009576 | -0.189680 |
| 3 | 6.365711 | 1.825104 | -0.199269 | 0.121889 |
| 4 | -3.176539 | -0.255947 | -0.707322 | -0.304417 |

For the third method (first four features of the dataset), we simply extract the first four features from the dataset as our dataset (both training and testing set) for this method. After selecting, we normalize the selected data using the same method as before.

When we are done with the preprocessing and features selection stage, we end up with three different dataset. We then run each dataset with KNN classifier algorithm with 3 neighbors and compare their accuracy scores using sklearn built-in accuracy score metric.

## 2.2 Result

Overall, the result that we obtained is as follow:

Accuracy of feature engineering: 0.709
Accuracy of PCA: 0.727
Accuracy of first four features: 0.636

Here, we observe that feature engineering and PCA performed quite similarly with PCA having slightly higher accuracy. Using the first four features yield the least accuracy (lower than using 20 features) as expected since important features that are highly correlated with class features are likely to have been removed. The high accuracy of PCA could be attributed to its success in eliminating noise information while preserving the characteristic of the original dataset. Although some spatial information could be lost from using PCA, in our case, the lost spatial information could be noise, thus improving our KNN algorithm instead. As for feature engineering, we can see that it clearly performed better than using 20 features as in part 1 which is expected because we removed some noise in the data by selecting the 4 most correlated features only. However, since we are just adding more features by multiplying 2 different features from the original set of features. We are essentially creating replicas of the original features which then influence the distance computation of KNN algorithm more significantly. This results in more noise being generated in the dataset from the noise originally present in the dataset. Therefore, even after selecting the best 4 features, the accuracy is still below PCA's accuracy due to more noise in the dataset.

# 3 Discussions

## 3.1 Possible Improvements

We could use k-fold cross validation on the training data instead of just splitting the dataset into train and test every single time. This would account for the probability of being allocated a 'biased' set for training data each time. For instance, we could do k fold validation (with 30 times splits) and take the mean of the accuracy scores each time we test a model. This would provide a more stable and possibly higher accuracy score.

Besides, using a single value imputation (eg. median imputation) may introduce bias to our dataset and thus may sway the mean and variance of the data leading to a less accurate prediction. Instead of median imputation, we could do different imputation method such as Regression Imputation method where the predicted value is obtained by regressing the missing variable on other variables. This would allow us to preserve the distribution shape of the data. Thus, resulting in a more accurate and realistic imputed dataset.

Apart from that, the result could definitely be improved significantly using more advanced machine learning algorithms such as gradient boosting algorithms, logistic regressions or random forest classifier. Advanced algorithms such as random forest classifier may be less sensitive to noise data thus may result in higher accuracy prediction.

## 3.2 Reliability of Classification Model

Overall, the classification models that we use after various feature engineering and selections are quite reliable as there is no data leakage in the form of train-test contamination because all of our preprocessing steps are done solely on the training data and transform onto the testing data. This includes median imputation (takes only the median from the training data) as well as any normalization steps that we do using `StandardScaler`. All models are also fitted on training data and transform onto the testing data to prevent any cross contamination between the two.

We also achieve accuracy score of around 0.7. This means that on average, out of 10 new data, our models are able to classify 7 data correctly. However, the accuracy scores obtained are not very reliable as they are not stable. This is because we only record the accuracy score once after prediction. In order to get a more reliable accuracy score, we need to cross-validate our data to reduce any bias from random number choices.