## Report on Feature Engineering and Selection

A decision tree is a tree like model which displays the decision and their possible consequences and is a way to display an algorithm only consisting of conditional control statements whereas in the k-nn classification elements are assigned to clusters by computing the centroids and the distance of the elements from the centroids. In task 2A, the k-nn algorithm (n=7) performed better than the decision tree algorithm. Following were the results –

```
Decision tree accuracy: 0.709
k-nn (k=3)    accuracy: 0.691
k-nn (k=7)    accuracy: 0.745
```

The dataset consisting of the 20 features and the target column (Life Expectancy) was first divided into training and testing sets (70% & 30% respectively) using the **sklearn** library's **train_test_split function**. This is done to build a model using the data whose output is already known i.e. the training set. The model is then applied to the testing set to observe its predictions. To avoid making the model complex, the depth of the decision tree is set to three. Both the algorithms are initialized with a random state parameter set to 200 to get the same output every time the code is executed. With the nearest neighbors set to 7, the accuracy scores for the predicted output is observed to be the highest(74.5%) amongst the three initialized algorithms. This can be observed due to the higher number of nearest neighbors which produce lower variance.

In task2b, feature engineering is implemented to generate new features from the existing features using interaction term pairs. First, the two datasets – world.csv and life.csv are merged based on the country code. The merged dataset is then processed to filter the nan values and then sorted alphabetically according to the country codes. This dataset now contains 20 original features. New features are created using these existing features by implementing **sklearn's PolynomialFeatures** function. This function is used to generate 190 new features (Fig 2). Next, k-means clustering is used to produce a new feature fclusterlabel (Fig 1) in which the cluster labels are stored as the column values. For this, the sum of squares within the clusters are calculated using **kmeans.inertia_** with the number of clusters initialized from 1 to 11. A graph is plotted using the number of clusters and the sum of squares. Here the elbow method is used to select the number of clusters for the KMeans clustering. The elbow point can be seen on the graph (Graph 1) at k=3 .

```
Feature engineered from clustering:
     fclusterlabel
0                0
1                0
2                0
3                1
4                0
..             ...
178              0
179              0
180              0
181              0
182              0

[183 rows x 1 columns]
```

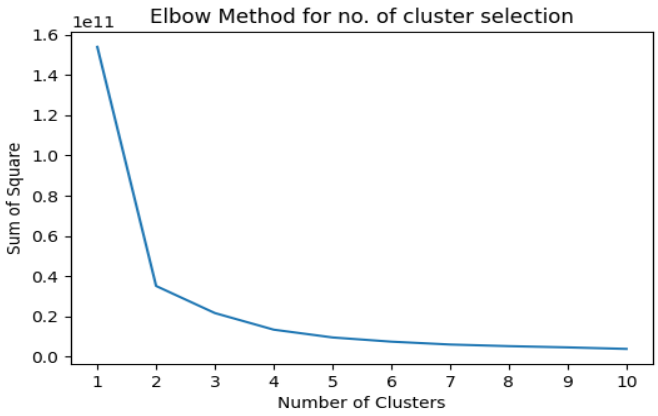
*Fig 1: Feature generation by clustering*

```
Dataset containing the old features and the new features engineered by interaction pairs:

           0            1         2  ...          207          208          209
0   97.700000    465.841583  87.941790  ...  1126.753439    77.210190   600.173428
1   40.624527   1949.114680  97.294588  ...   687.926850   144.178980  1432.474983
2  100.000000   3776.111694  45.645034  ...  8189.965669  6274.060451  2456.135856
3  100.000000  35465.405180  18.164904  ...  1227.798342  92640.579916   129.089135
4   99.966881  11006.029310  56.102271  ...  15284.840049  58217.637865  1647.068019
```

*Figure 2: Feature generation by interaction pairs*

From the figures 1 and 2, it can be seen how the interaction pairs are produced.




*Graph 1: Elbow Method*

Now, the new dataset consists of total 211 features. From this dataset, highly correlated features are dropped by first constructing a correlation matrix and filtering out the columns having a correlation equal to or more than 0.5. This process helps to reduce the computational costs of the algorithm as these highly correlated features are linearly dependent with other features and contribute very less to predict the output. Next, a chi square test is performed to select the four best features from the remaining features after the above filtering. This gives the following 4 features as the topmost out of all –

| | Specs | Score |
|---|---|---|
| 1 | Adjusted net national income per capita (curre... | 16016.397536 |
| 2 | Fixed broadband subscriptions (per 100 people)... | 12.598911 |
| 0 | Access to electricity (% of population) [EG.EL... | 9.583321 |
| 3 | Fixed telephone subscriptions (per 100 people)... | 3.950629 |

*Figure 3: 4 Best Features*

These features are then used for 3-NN classification and a model is built over the training set. Similar analysis is performed using the PCA method where first four principal components are selected for the 3-NN classification whereas the first four features from world.csv dataset are selected for the third method to perform the 3-NN. Following are the results –

```
Accuracy of feature engineering:  0.709
Accuracy of PCA:  0.782
Accuracy of first four features:  0.727
```

The PCA method is observed to produce the best results with an accuracy of 78.2% compared to the other two methods. PCA helps in dimensionality reduction which is very essential while working upon a large number of features. It helps to improve the outcome of a classifier. To improve the classification accuracy with this data, more relevant models such as regression can be applied, and the accuracies can then be compared. Tuning can also be used where the parameters used to train the model are changed. I believe that the classification model built in this process can be considered reliable as the prediction accuracies observed are in a good range reflecting a good model.