Supervised and Unsupervised Learning Algorithms for Life Expectancy Analysis

28/08/23

MAGDALENA BEATA SKOWERSKA

Contents

List of Figures

Introduction

The analysis of life expectancy data can provide valuable insights and help researchers, policymakers, and healthcare professionals better understand the factors influencing life expectancy and potentially identify patterns that might not be immediately apparent.

The aim of the project is to forecast the life expectancy by considering different factors such as: air pollution, diet, GDP per capita, lifestyle, education etc. and analyze common characteristics between different groups of countries identified.

The dataset used for this purpose is retrieved from the online community Kaggle, The Food and Agriculture Organization (FAO) and the Our World in Data.

The first part describes the dataset used in the analysis and some checks are performed to ensure its quality and suitability for applying the models. An evaluation of the different methods to select the most important predictors has been done to address the multicollinearity issue and to assess their impact on the improvement of the models performance.

In the second part, different predictive models are compared by using the following Supervised Learning techniques: Decision Trees, Random Forest and Boosting. A comparison between the Root Mean Square Error of all the models has been made to highlight which has the best performance.

The third part, analyzes the data by using Unsupervised Learning techniques such as: K-Means and Hierarchical Clustering. Through the analysis, it is possible to identify countries with similar characteristics where some interventions are needed to improve the quality of life, and so to increase the life expectancy considering the most influential factors.

The dataset

The final dataset combines data from different sources:

- Kaggle community from which there were retrieved: the Country Life Expectancy dataset (Kaggle, s.d.), health statistics of different countries (Kaggle, s.d.) and happiness statistics of different countries (Kaggle, s.d.). The following fields were taken into account:
    - Country
    - Schooling: indicates the education level of a country
    - Life expectancy (target variable in the supervised learning techniques) representing the average expected age of death per country
    - UHC Coverage: Universal health coverage (UHC) measures the access to the full range of quality health services people need, when and where they need them, without financial hardship. It covers the full continuum of essential health services, from health promotion to prevention, treatment, rehabilitation, and palliative care across the life course. The indicator is an index reported on a unitless scale of 0 to 100, which is computed as the geometric mean of 14 tracer indicators of health service coverage. The tracer indicators are as follows, organized by four components of service coverage: 1. Reproductive, maternal, newborn and child health 2. Infectious diseases 3. Noncommunicable diseases 4. Service capacity and access.
    - Tobacco usage: Prevalence of current tobacco use among persons aged 15 years and older (for this indicator age-standardized rate will be considered and it is used to allow for comparisons between populations with different age distributions)
    - Alcohol Abuse: total including both recorded and unrecorded alcohol consumption per capita among individuals aged 15 and older.
    - Ladder score: The ladder score is a measure of happiness or subjective well-being. It reflects the national average response to the question of life evaluations, where individuals rate their own life satisfaction on a scale. This score provides an overview of how people perceive their overall well-being and happiness.
    - Perceptions of corruption: The measure assesses perceptions of corruption. It is typically based on survey responses to questions related to the prevalence of corruption within the government and businesses. The national average of these responses provides insight into how corruption is perceived within a country.
    - GDP per capita

- The Food and Agriculture Organization website (FAO, s.d.), taking into account indicators representing the average amount of food available per person in a specific country over a given year:
    - Fruits Indicator
    - Fish Indicator
    - Meat Indicator
    - Sugar Indicator
    - Vegetables Indicator

- The Our World in Data from which the data retrieved is about Air Pollution (Our World in Data, s.d.):
    - Air Pollution: age-standardized data representing ambient and household air pollution attributable death rate per 100,00 population.

The data considered for the purposes of this report is limited to the year 2015, since this is the year for which all the data is available. By joining all the different datasets on the field *Country*, a final dataset of 99 observations and 14 independent variables is obtained.

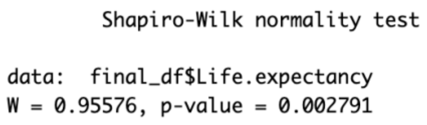
Data Checks

In this section, **Data Cleaning** is performed by checking and removing missing values. Four missing values were found: three in the column *Perceptions.of.corruption* and one in the column *Fish.Indicator*. Rows containing those values will be deleted.

The distribution of the data is analyzed to identify potential outliers. By using IQR Method no potential outlier has been identified.

Several checks on the data are conducted to ensure its quality and analyze its suitability for the regression model. These checks help to identify potential issues that could affect the reliability and accuracy of the regression results.

The response variable not normally distributed can affect the validity of statistical inference and the reliability of model predictions. The normality assumption check of the response variable *Life.expectancy* was done with the following results:



```
        Shapiro-Wilk normality test

data:  final_df$Life.expectancy
W = 0.95576, p-value = 0.002791
```

FIGURE 1 - SHAPIRO WILK TEST RESPONSE VARIABLE

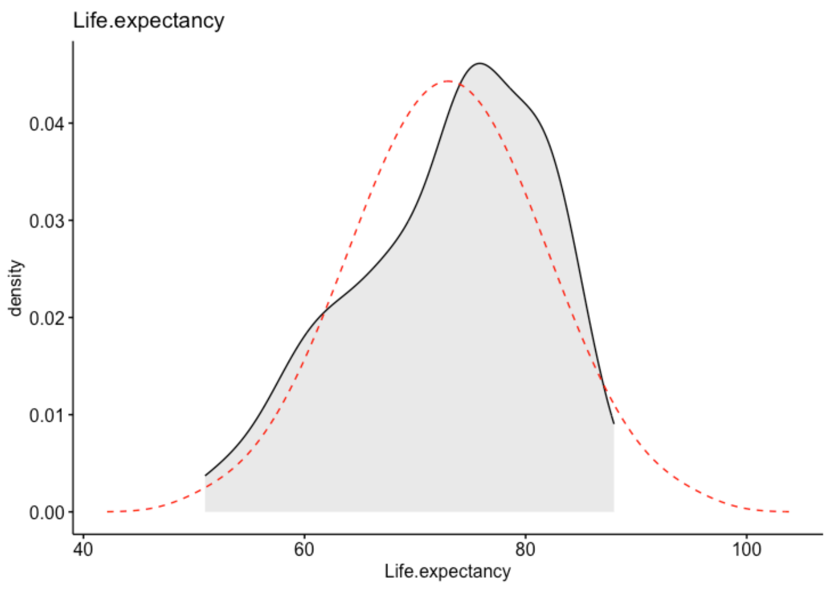
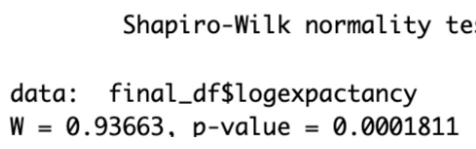


FIGURE 2 - LIFE EXPECTANCY DISTRIBUTION

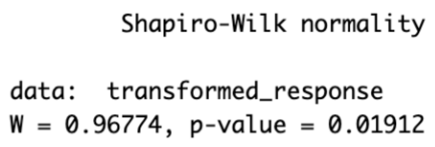
By performing the Shapiro-wilk test we obtain the p-value that is lower than 0.05, so the null hypothesis is rejected, indicating that the data does not follow a normal distribution.

The transformations of the response variable using mathematical functions like logarithm and Box-Cox transformation don't improve the results of the Shapiro-wilk test as indicated by following results (Fiugre 3) (Figure 4).



```
                Shapiro-Wilk normality test

data:  final_df$logexpactancy
W = 0.93663, p-value = 0.0001811
```

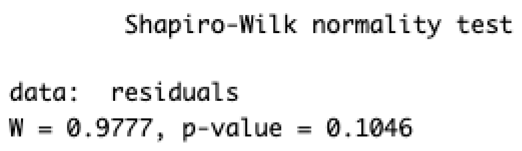
FIGURE 3 - SHAPIRO WILK TEST AFTER LOGARITHM TRANSFORMATION



```
                Shapiro-Wilk normality test

data:  transformed_response
W = 0.96774, p-value = 0.01912
```

FIGURE 4 - SHAPIRO WILK TEST AFTER BOX-COX TRANSFORMATION

Even if the response variable is not normally distributed, the residuals are normally distributed as can be seen from the Shapiro-wilk test result on the residuals (Figure 5).



```
             Shapiro-Wilk normality test

     data:  residuals
     W = 0.9777, p-value = 0.1046
```

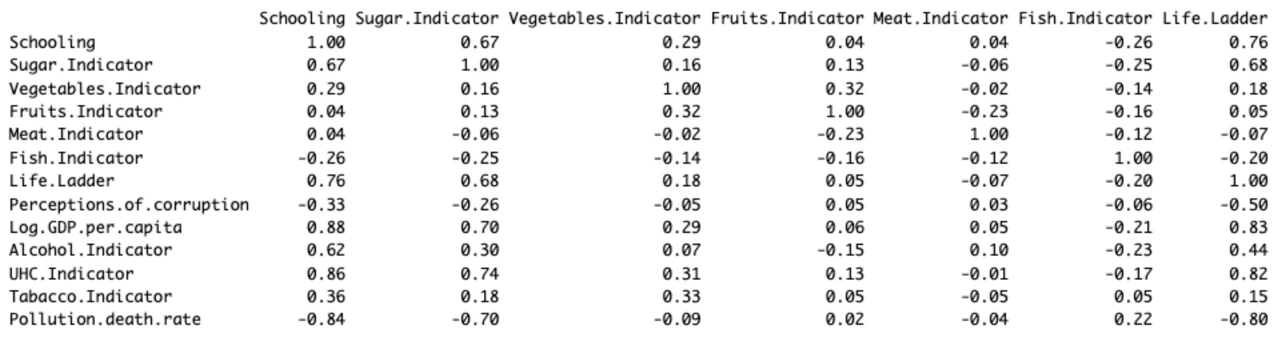
FIGURE 5 - SHAPIRO WILK TEST ON THE RESIDUALS

This result is important especially with smaller dataset where the central limit theorem (that often ensures an approximation the estimates distribution to the normal distribution even if the response variable itself is not perfectly normally distributed)) cannot be applied. In fact, the linear regression assumes that the residuals (the differences between observed and predicted values) are normally distributed.

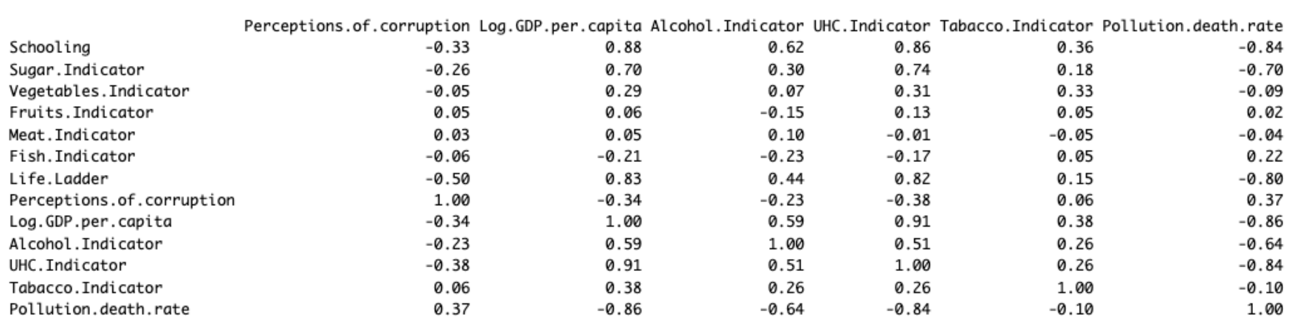
It is worth noting that there could be a multicollinearity issue which can lead to unstable coefficient estimates, and so complicated model interpretability, that lead to the violation of one of the linear regression assumption. In case of non-parametric models multicollinearity can lead to longer training times and increased model complexity.

Some common methods to check for multicollinearity among independent variables include calculating the correlation matrix between predictor variables and variance inflation factors (VIFs).

First, we check the correlation matrix between all the independent variables exept *Country* which is not considered an index for forecasting the *Life.expectancy* and will not be included in the analysis.



```
                          Schooling Sugar.Indicator Vegetables.Indicator Fruits.Indicator Meat.Indicator Fish.Indicator Life.Ladder
Schooling                      1.00            0.67                 0.29             0.04           0.04          -0.26        0.76
Sugar.Indicator                0.67            1.00                 0.16             0.13          -0.06          -0.25        0.68
Vegetables.Indicator           0.29            0.16                 1.00             0.32          -0.02          -0.14        0.18
Fruits.Indicator               0.04            0.13                 0.32             1.00          -0.23          -0.16        0.05
Meat.Indicator                 0.04           -0.06                -0.02            -0.23           1.00          -0.12       -0.07
Fish.Indicator                -0.26           -0.25                -0.14            -0.16          -0.12           1.00       -0.20
Life.Ladder                    0.76            0.68                 0.18             0.05          -0.07          -0.20        1.00
Perceptions.of.corruption     -0.33           -0.26                -0.05             0.05           0.03          -0.06       -0.50
Log.GDP.per.capita             0.88            0.70                 0.29             0.06           0.05          -0.21        0.83
Alcohol.Indicator              0.62            0.30                 0.07            -0.15           0.10          -0.23        0.44
UHC.Indicator                  0.86            0.74                 0.31             0.13          -0.01          -0.17        0.82
Tabacco.Indicator              0.36            0.18                 0.33             0.05          -0.05           0.05        0.15
Pollution.death.rate          -0.84           -0.70                -0.09             0.02          -0.04           0.22       -0.80
```



```
                          Perceptions.of.corruption Log.GDP.per.capita Alcohol.Indicator UHC.Indicator Tabacco.Indicator Pollution.death.rate
Schooling                                     -0.33               0.88              0.62          0.86              0.36                -0.84
Sugar.Indicator                               -0.26               0.70              0.30          0.74              0.18                -0.70
Vegetables.Indicator                          -0.05               0.29              0.07          0.31              0.33                -0.09
Fruits.Indicator                               0.05               0.06             -0.15          0.13              0.05                 0.02
Meat.Indicator                                 0.03               0.05              0.10         -0.01             -0.05                -0.04
Fish.Indicator                                -0.06              -0.21             -0.23         -0.17              0.05                 0.22
Life.Ladder                                   -0.50               0.83              0.44          0.82              0.15                -0.80
Perceptions.of.corruption                      1.00              -0.34             -0.23         -0.38              0.06                 0.37
Log.GDP.per.capita                            -0.34               1.00              0.59          0.91              0.38                -0.86
Alcohol.Indicator                             -0.23               0.59              1.00          0.51              0.26                -0.64
UHC.Indicator                                 -0.38               0.91              0.51          1.00              0.26                -0.84
Tabacco.Indicator                              0.06               0.38              0.26          0.26              1.00                -0.10
Pollution.death.rate                           0.37              -0.86             -0.64         -0.84             -0.10                 1.00
```

FIGURE 6 - CORRELATION MATRIX FOR ALL THE PREDICTORS WITH VALUES

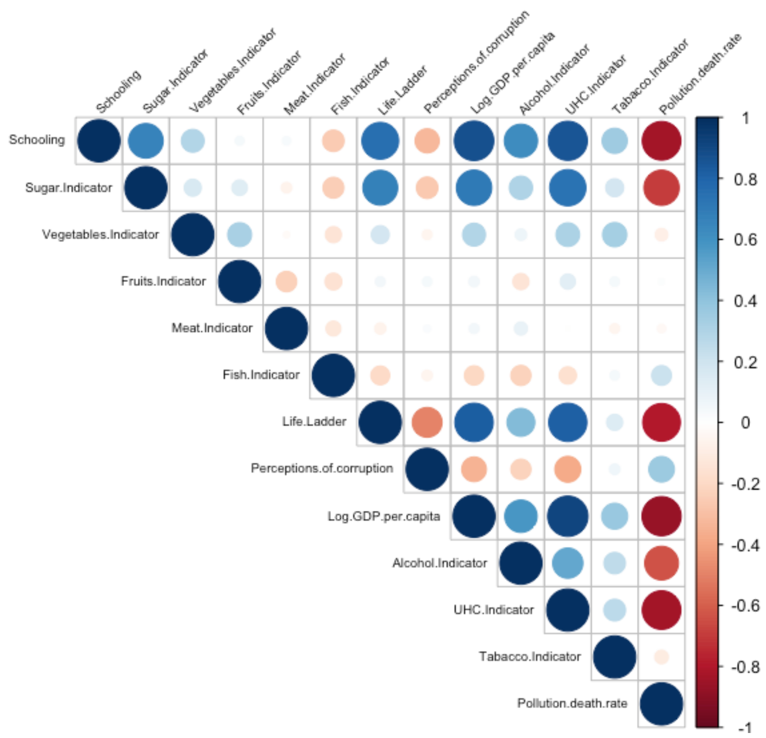


FIGURE 7 - CORRELATION MATRIX ALL PREDICTORS

Variance Inflation Factor (VIF) represents how much the variance of the estimated coefficient of a predictor variable is inflated due to correlations with other predictor variables. It is calculated based on the R-squared value obtained when regressing a predictor variable against all the other predictor variables in the model. VIF values greater than 2 may indicate multicollinearity.

```
                Schooling           Sugar.Indicator      Vegetables.Indicator          Fruits.Indicator            Meat.Indicator
                     TRUE                     FALSE                     FALSE                     FALSE                     FALSE
           Fish.Indicator               Life.Ladder Perceptions.of.corruption        Log.GDP.per.capita         Alcohol.Indicator
                    FALSE                      TRUE                     FALSE                      TRUE                     FALSE
            UHC.Indicator         Tabacco.Indicator      Pollution.death.rate
                     TRUE                     FALSE                      TRUE
```

FIGURE 8 - VIF RESULTS ALL PREDICTORS

As we can see from the analysis there are some variables that can be correlated, so an evalutation of different methods to select the most important predictors is needed. In this report, the considered and assessed methods are: best subset selection, forward stepwise selection and PCA. It's important taking into account the fact that these selection methods can lead to undesirable results. In fact, best subset selection and stepwise selection may yield inconsistent variable selection results because they may select one subset in one run and a different subset in another, depending on the specific data or order of evaluation. Also PCA is a linear technique, so it may not perform well if the underlying relationships in the data are nonlinear.

Best subset selection results (Figure 9):

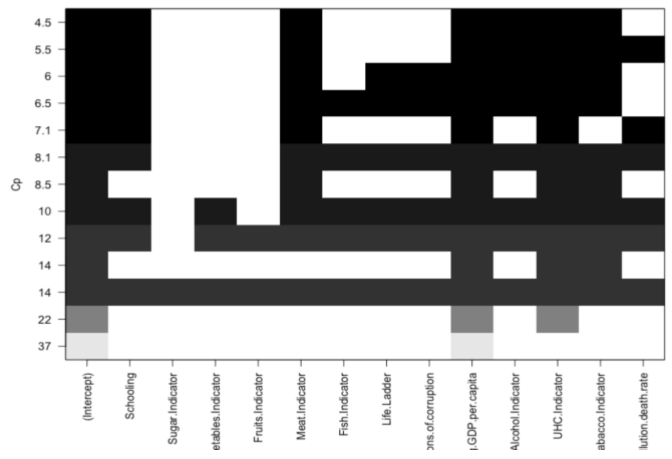


FIGURE 9 - BEST SELECTION RESULTS

Forward Stepwise Selection results (Figure 10):

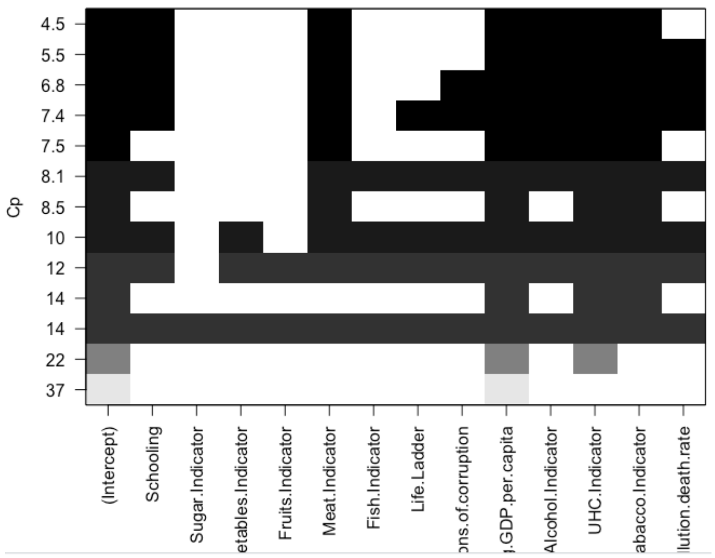


FIGURE 10 - FORWARD STEPWISE SELECTION RESULTS

Forward Stepwise Selection results using cross-validation to determine the optimal number of predictors to use in the model based on MSE (Figure 11):

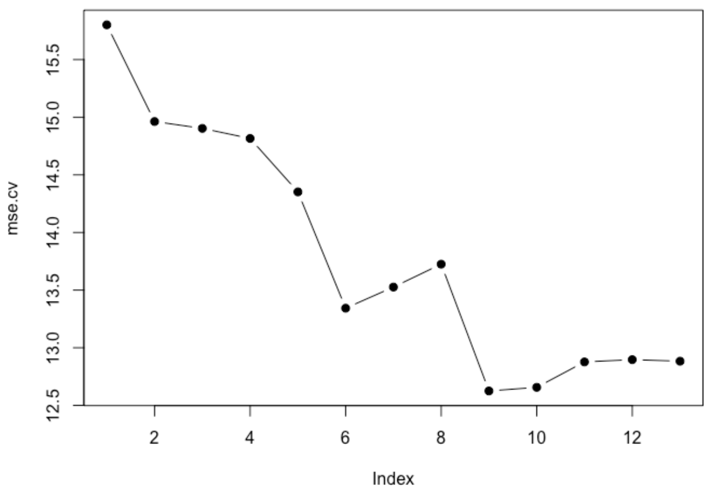


FIGURE 11- CROSS-VALIDATION FORWARD STEPWISE SELECTION RESULTS

On the basis of these results six features should be considered: “Schooling”, "Meat.Indicator", "Log.GDP.per.capita", "Alcohol.Indicator", "Tabacco.Indicator", “UHC.Indicator”.

With these predictors the calculated correlation matrix and VIF are (Figure 12) (Figure 13):

```
Schooling    Meat.Indicator  Alcohol.Indicator Log.GDP.per.capita  Tabacco.Indicator     UHC.Indicator
     TRUE             FALSE              FALSE               TRUE              FALSE              TRUE
```

FIGURE 12 - VIF RESULTS OF A SUBSET OF PREDICTORS

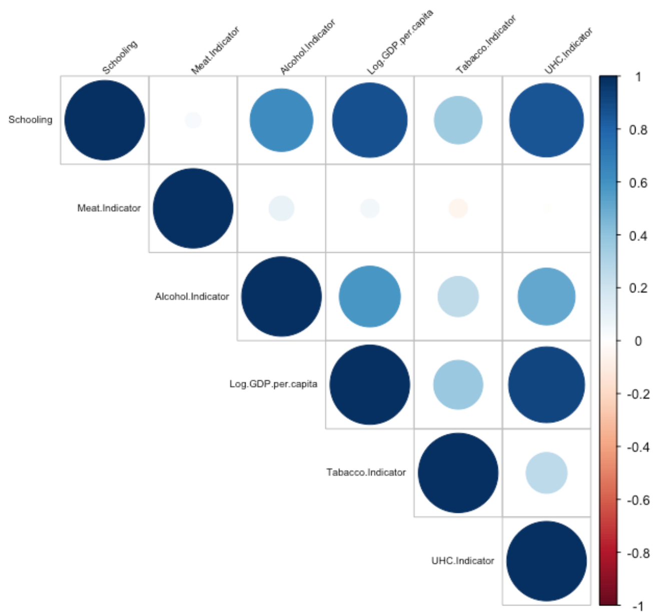


FIGURE 13 - CORRELATION MATRIX OF A SUBSET OF PREDICTORS

From the above results, some of the variables may be still correlated so the subset is considered: "Meat, Other", "Log.GDP.per.capita", "Alcohol Indicator", "Tabacco Indicator".

```
   Meat.Indicator  Alcohol.Indicator Log.GDP.per.capita  Tabacco.Indicator
            FALSE              FALSE              FALSE              FALSE
```

FIGURE 14 - VIF RESULTS OF A SMALLER SUBSET OF PREDICTORS

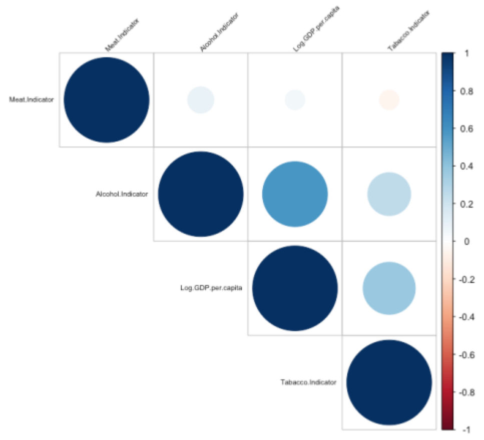
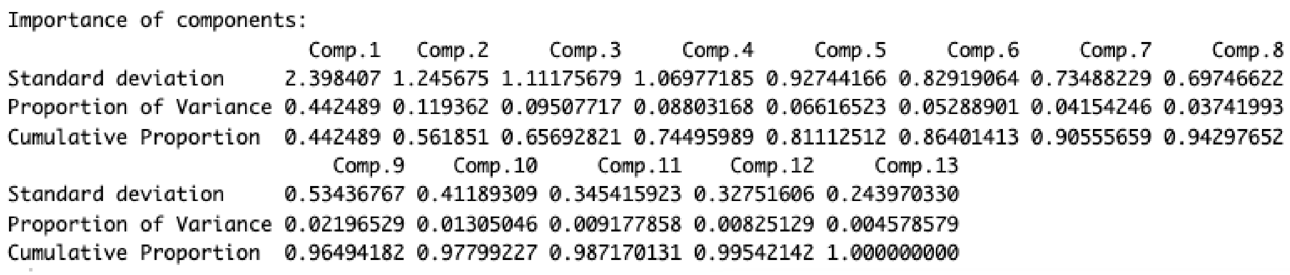


FIGURE 15 - CORRELATION MATRIX OF A SMALLER SUBSET OF PREDICTORS

An assessment of the most important features is done also by using Principal Component Analysis (PCA), a technique for reducing the dimensionality of data by identifying the principal components that capture the most significant variance in the data. Each principal component is a linear combination of the original variables. The loadings of the original variables on the principal components are coefficients that indicate the weights of each original variable in the linear combination that forms a principal component and can reveal which variables have the strongest influence on each component. This can help in identifying key features driving the variability in your data.

From the analysis it emerges that with four components around 74% of the total variance is explained and adding more components doesn't significantly increase the explained variance. This is also the number of components that can be selected by considering Kaiser-Guttman Criterion which is represented by the red line in Figure 18. This criterion suggests retaining components with eigenvalues greater than 1. Eigenvalues represent the amount of variance explained by each component and components with eigenvalues above 1 explain more variance than a single original variable.



```
Importance of components:
                          Comp.1   Comp.2     Comp.3     Comp.4     Comp.5     Comp.6     Comp.7     Comp.8
Standard deviation     2.398407 1.245675 1.11175679 1.06977185 0.92744166 0.82919064 0.73488229 0.69746622
Proportion of Variance 0.442489 0.119362 0.09507717 0.08803168 0.06616523 0.05288901 0.04154246 0.03741993
Cumulative Proportion  0.442489 0.561851 0.65692821 0.74495989 0.81112512 0.86401413 0.90555659 0.94297652
                           Comp.9    Comp.10     Comp.11    Comp.12     Comp.13
Standard deviation     0.53436767 0.41189309 0.345415923 0.32751606 0.243970330
Proportion of Variance 0.02196529 0.01305046 0.009177858 0.00825129 0.004578579
Cumulative Proportion  0.96494182 0.97799227 0.987170131 0.99542142 1.000000000
```

FIGURE 16 - IMPORTANCE OF COMPONENTS PCA

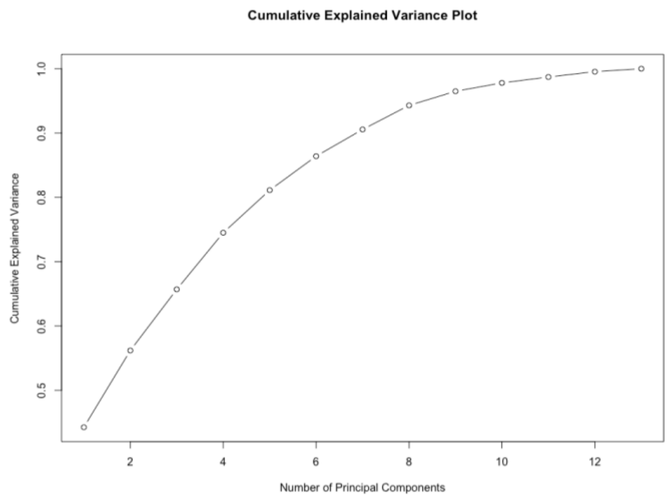


FIGURE 17 - CUMULATIVE EXPLAINED VARIANCE PCA

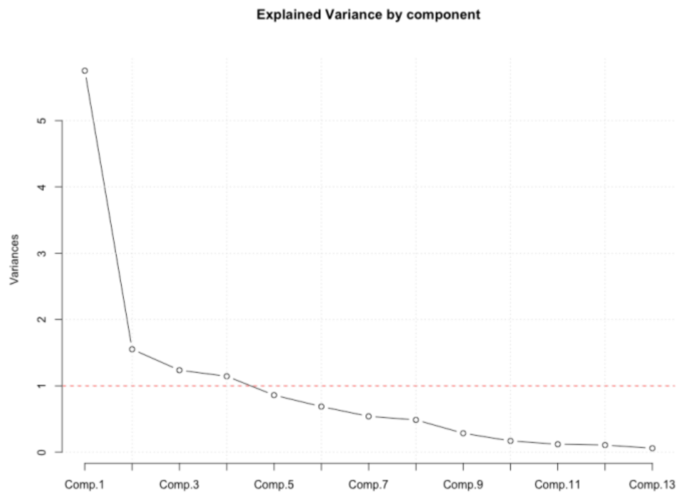
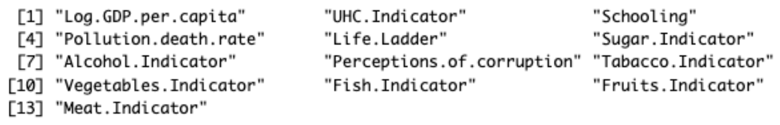


FIGURE 18 - EXPLAINED VARIANCE BY COMPONENT PCA

The screeplot above (Figure 18) displays the eigenvalues (or variances) of the principal components in descending order. It helps to visualize the amount of variance explained by each principal component and decide how many components to retain.

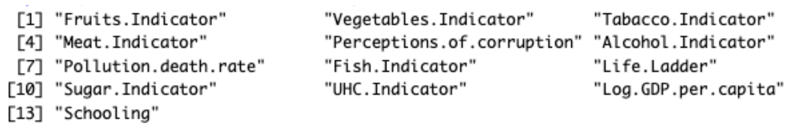
However, since in our analysis we want to use the original variables to build the models and understand how they are related to the response variable the PCA results will be used to analyze the quantitative variable contribution, or in other words the extent to which each original feature contributes to the formation of each principal component that can be expressed as loadings.

Below there is a list of the most influential variables on the first four PCA components in a descending order.



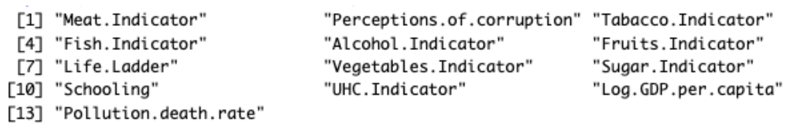
```
 [1] "Log.GDP.per.capita"       "UHC.Indicator"            "Schooling"
 [4] "Pollution.death.rate"     "Life.Ladder"              "Sugar.Indicator"
 [7] "Alcohol.Indicator"        "Perceptions.of.corruption" "Tabacco.Indicator"
[10] "Vegetables.Indicator"     "Fish.Indicator"           "Fruits.Indicator"
[13] "Meat.Indicator"
```

FIGURE 19 - INFLUENTIAL VARIABLES ON THE FIRST COMPONENT



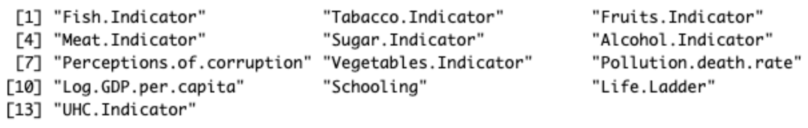
```
 [1] "Fruits.Indicator"         "Vegetables.Indicator"     "Tabacco.Indicator"
 [4] "Meat.Indicator"           "Perceptions.of.corruption" "Alcohol.Indicator"
 [7] "Pollution.death.rate"     "Fish.Indicator"           "Life.Ladder"
[10] "Sugar.Indicator"          "UHC.Indicator"            "Log.GDP.per.capita"
[13] "Schooling"
```

FIGURE 20 - INFLUENTIAL VARIABLES ON THE SECOND COMPONENT



```
 [1] "Meat.Indicator"           "Perceptions.of.corruption" "Tabacco.Indicator"
 [4] "Fish.Indicator"           "Alcohol.Indicator"        "Fruits.Indicator"
 [7] "Life.Ladder"              "Vegetables.Indicator"     "Sugar.Indicator"
[10] "Schooling"                "UHC.Indicator"            "Log.GDP.per.capita"
[13] "Pollution.death.rate"
```

FIGURE 21 - INFLUENTIAL VARIABLES ON THE THIRD COMPONENT



```
 [1] "Fish.Indicator"            "Tabacco.Indicator"        "Fruits.Indicator"
 [4] "Meat.Indicator"            "Sugar.Indicator"          "Alcohol.Indicator"
 [7] "Perceptions.of.corruption" "Vegetables.Indicator"     "Pollution.death.rate"
[10] "Log.GDP.per.capita"        "Schooling"                "Life.Ladder"
[13] "UHC.Indicator"
```

FIGURE 22 - INFLUENTIAL VARIABLES ON THE FOURTH COMPONENT

Below (Figure 23) there is a visualization of vectors representing the original variables on the two-dimensional plane where the principal components from a PCA analysis are the axis. The length and direction of the variable vectors indicate how strongly and in which direction the variables contribute to each principal component. The arrows are coloured on the basis of quantitative variable contribution from low values represented by green to higher values represented by blue. The angle between the arrows indicates the correlation between variables.

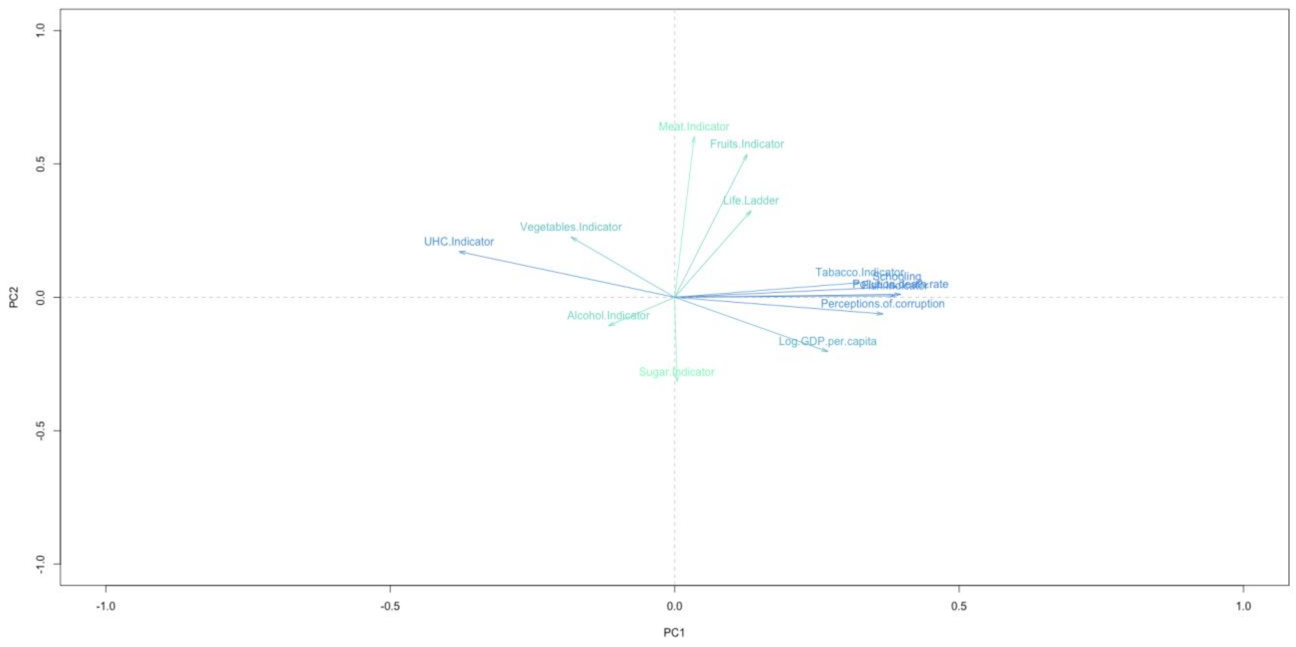


FIGURE 23 - BIDIMENSIONAL PCA PLOT

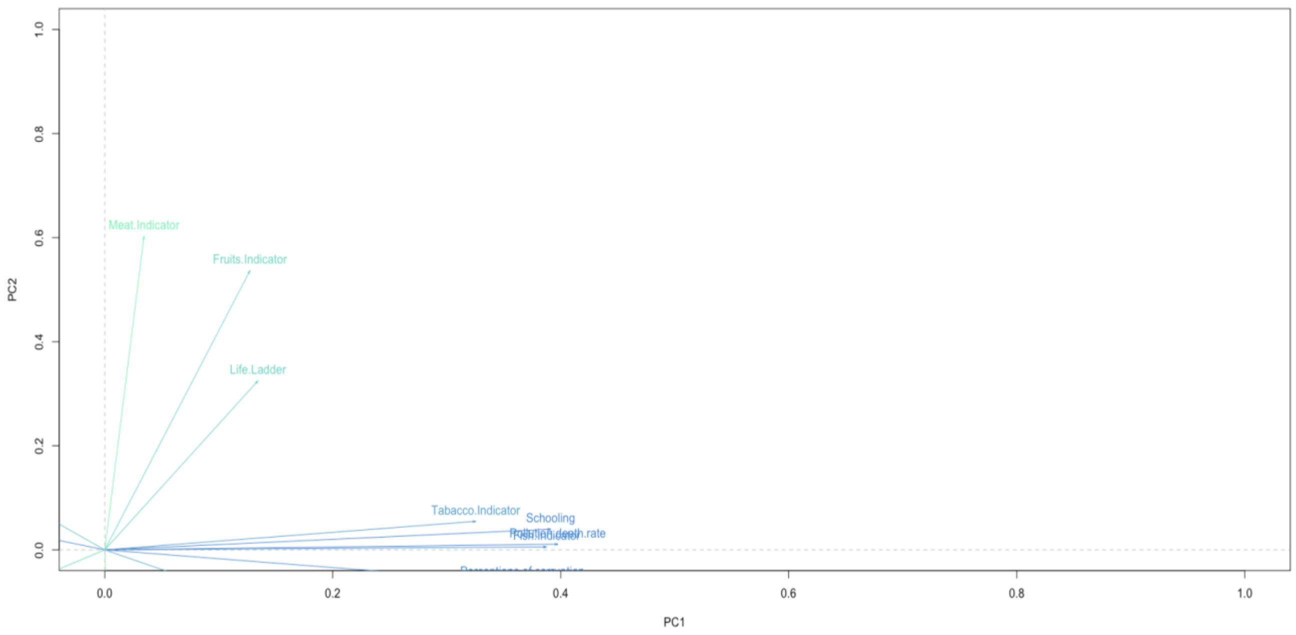
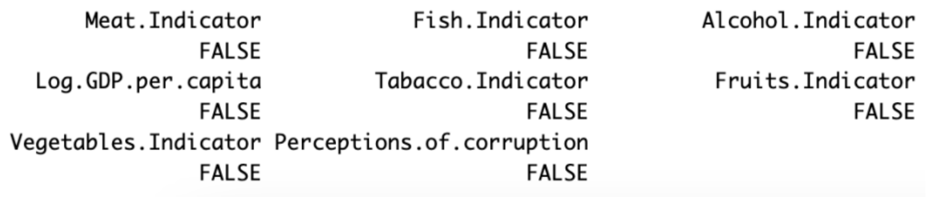


FIGURE 24 - BIDIMENSIONAL PCA PLOT OF ONE QUADRANT

PCA produces slightly different results from the other selection methods. By considering the three most influential variables on the first four components and adding them to the list of variables selected before, the calculated correlation matrix and VIF are (Figure 25) (Figure 25):



```
       Meat.Indicator            Fish.Indicator         Alcohol.Indicator
                FALSE                     FALSE                     FALSE
    Log.GDP.per.capita         Tabacco.Indicator          Fruits.Indicator
                FALSE                     FALSE                     FALSE
  Vegetables.Indicator Perceptions.of.corruption
                FALSE                     FALSE
```

FIGURE 25 - VIF RESULTS OF THE FINAL SUBSET OF PREDICTORS

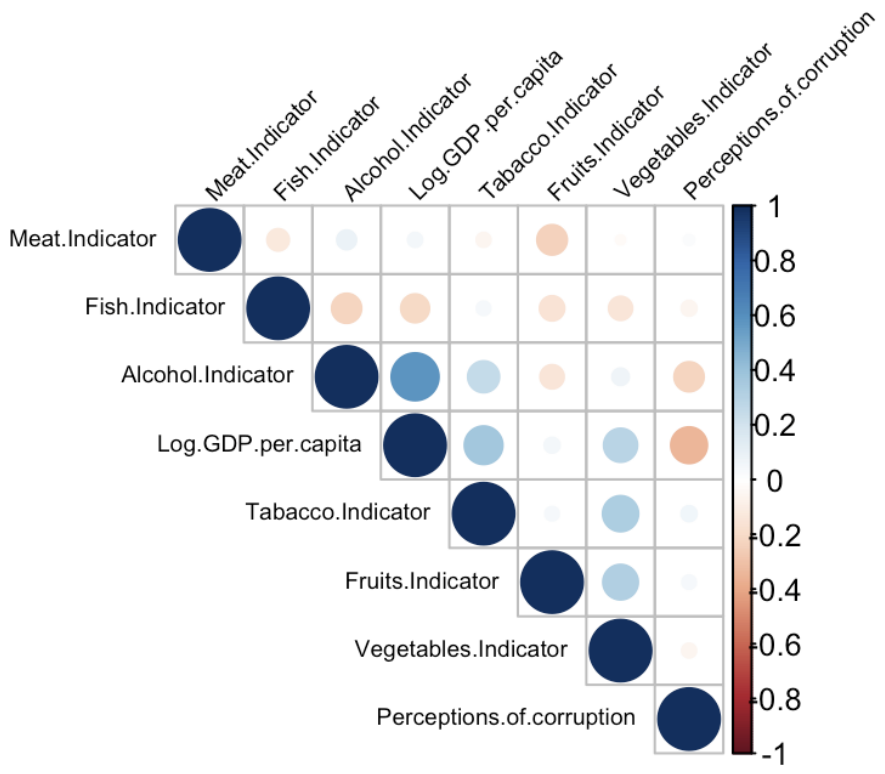


FIGURE 26 - CORRELATION MATRIX OF THE FINAL SUBSET OF PREDICTORS

Finally, the subset of the following selected variables will be considered and used to build and assess the models comparing them to the models where full dataset was used: "Life.expectancy", "Meat.Indicator", "Fish.Indicator","Alcohol.Indicator", "Log.GDP.per.capita", "Tabacco.Indicator", "Fruits.Indicator", "Vegetables.Indicator", "Perceptions.of.corruption".

By analyzing further the data, it can be shown that not all of the predictors depend linearly on Life expectancy (Figure 27). The features that seem to have a linear relationship are: Log.GDP.per.capita, Schooling, UHC indicator, Pollution.death.rate, Life.Ladder. For this reason, the linear regression and principal component regression (PCR – that can be used also in the presence of multicollinearity) cannot be applied since the linearity assumption on the relationship between the response variable and the selected predictors is violated.

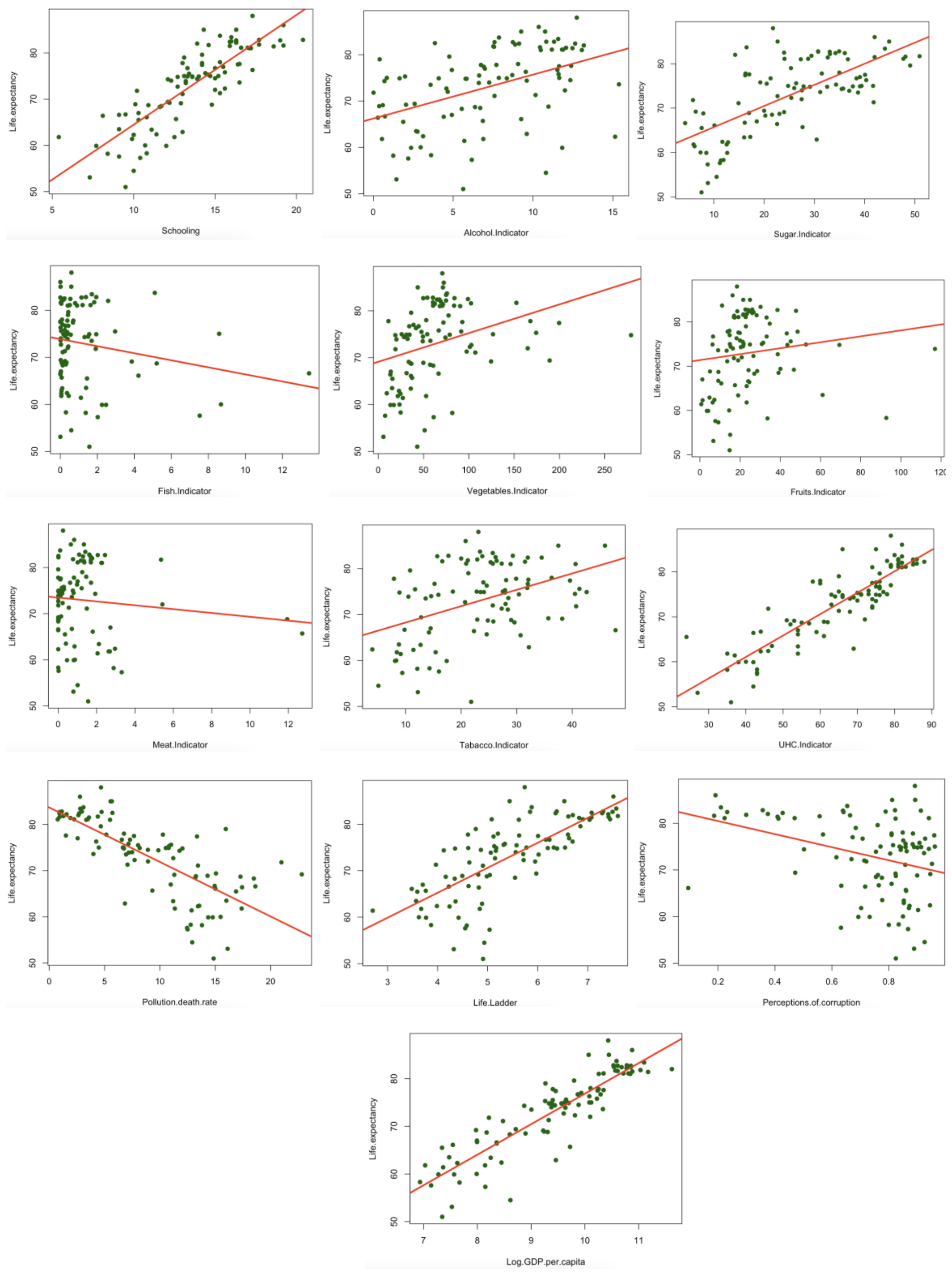


FIGURE 27 - LINEARITY CHECK

Non parametric model selection

In this section, different non-parametric models are considered and compared on the basis of their performance using root mean squared error (RMSE) as the evaluation metric (calculated by taking the square root of the average of the squared differences between the predicted values and the actual values).

Three different models are considered: Decision Tree, Random Forest and Boosting. Their performance will be calculated on both the entire dataset and the dataset with the selected features from the previous section.

Decision tree

In the analysis, there are considered two different decision tree algorithms corresponding to two different R functions: *rpart*, *ctree*.

The *rpart* function is based on binary recursive partitioning. The tree is built by recursively splitting the data into subsets based on the features that provide the best separation according to a chosen criterion, such as Gini impurity or cross-entropy, aiming to maximize the homogeneity of the target variable within each resulting subset after the split.

The *ctree* algorithm uses statistical tests, to assess the significance of a split based on the relationships between the predictor variables and the target variable. For example, for continuous predictor variables, the algorithm may use statistical tests like the Mann-Whitney U test or the Kolmogorov-Smirnov test to evaluate whether the distribution of the target variable differs significantly in different ranges or groups created based on the predictor variable's values. The algorithm calculates a p-value from the statistical test for each potential split and compares these p-values. A low p-value indicates that the split is statistically significant, implying that the predictor variable provides useful information for separating the data based on the target variable.

Considering all the features and using *rpart* function with the default splitting criterion (Gini impurity) the decision tree (Figure 28) obtained is characterized by a Test Root Mean Squared Error (RMSE) equal to 3.717.

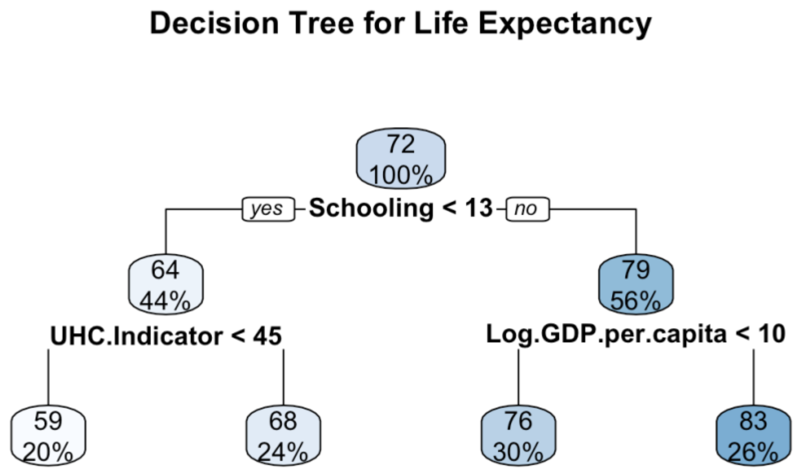


FIGURE 28 - DECISION TREE PLOT USING ALL PREDICTORS AND RPART() FUNCTION

Using the *ctree* function the decision tree with the Test Root Mean Squared Error (RMSE) equal to 3.798 and represented on Figure 29 is obtained. Using *ctree* function a split is implemented when the criterion exceeds the value given by mincriterion. In this case, mincriterion = 0.95 so the p-value must be smaller than 0.05 in order to split the node.

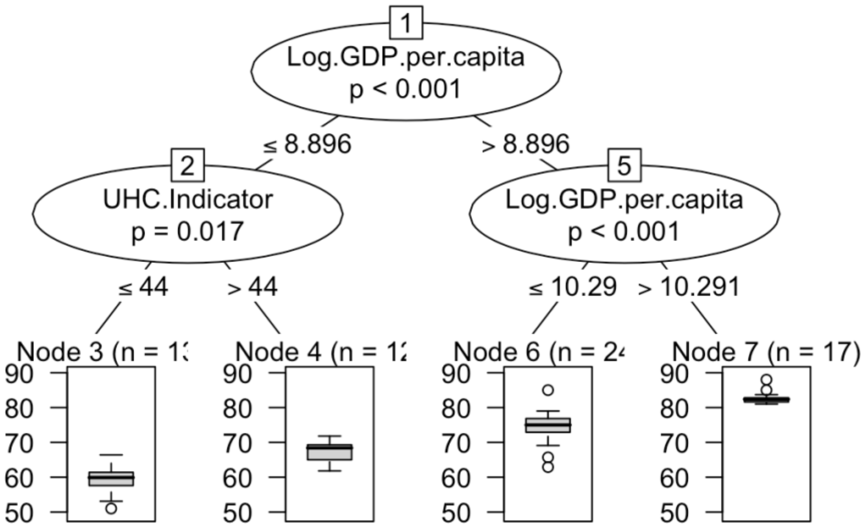


FIGURE 29 - DECISION TREE PLOT USING ALL PREDICTORS AND CTREE() FUNCTION

Considering only the selected features and using *rpart* function with the default splitting criterion (Gini impurity) the decision tree of the Figure 30 and characterized by a Root Mean Squared Error (RMSE) equal to 4.325 is obtained.

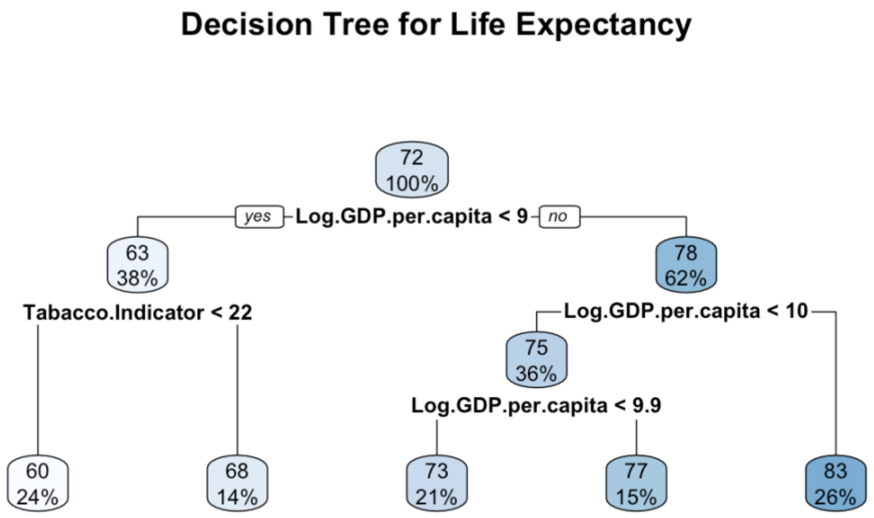


FIGURE 30 - DECISION TREE PLOT USING THE SELECTED PREDICTORS AND RPART() FUNCTION

Using the *ctree* function the decision tree of the Figure 31 characterized by a Root Mean Squared Error (RMSE) equal to 4.02 is obtained. In this case, mincriterion = 0.99 so the p-value must be smaller than 0.01 in order to split the node. The mincriterion value is increased with respect to the previous case to improve the accuracy.

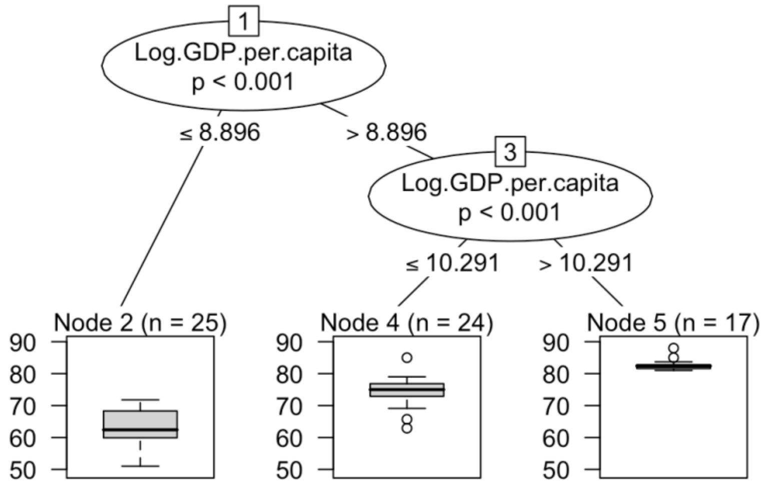


FIGURE 31 - DECISION TREE PLOT USING THE SELECTED PREDICTORS AND CTREE() FUNCTION

When building a decision tree, the algorithm chooses the features that best split the data to maximize information gain. Features that lead to the most significant reduction in uncertainty or impurity are selected at the higher levels of the tree. This naturally prioritizes the most informative features. Consequently, it can be noted that the value of the GDP per capita of a country has a significant impact on the life expectancy. It seems also that the performance of the algorithm is better by using all the features and in this case using all the features don't have a significant impact on the execution time of the algorithms. Considering this, other two important features are: the education level and the country healthcare service.

Random Forest

Random Forest is another algorithm considered in the analysis. A random forest is an ensemble method that consists of a collection of decision trees. It builds multiple decision trees, starting from a random subset of the original dataset sampled with replacement, and it combines their predictions to make a final prediction. It introduces randomness and reduce the variance leading to a better generalization of the model.

First, the model performance is analyzed on the basis of the number of randomly selected variables at each split (mtry) considering the dataset with all the features (Figure 32).

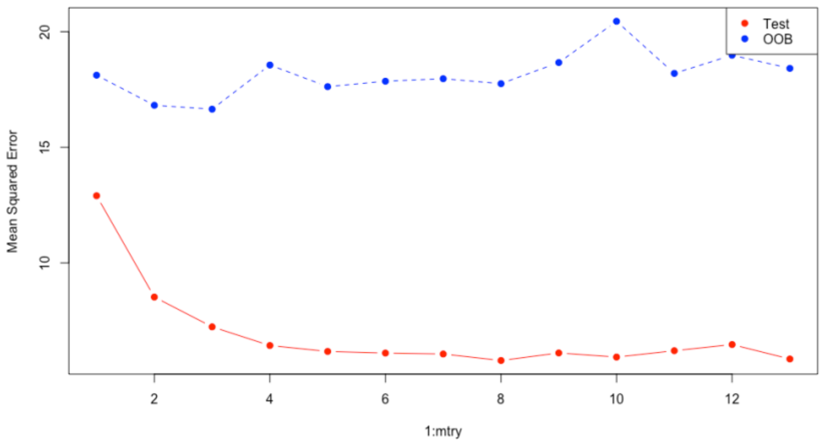
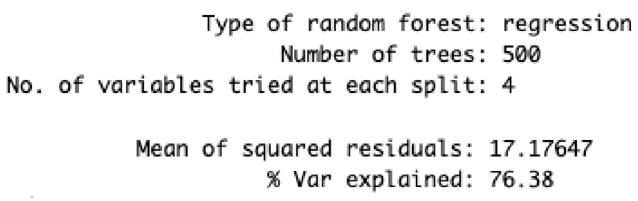


FIGURE 32 - MSE FOR DIFFERENT MTRY

By selecting four as the number of randomly selected variables at each split, the following result is obtained (Figure 33):



```
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 17.17647
                    % Var explained: 76.38
```

FIGURE 33 - RANDOM FOREST WITH ALL THE PREDICTORS

The Root Mean Squared Error (RMSE) on the test dataset is about 2.59.

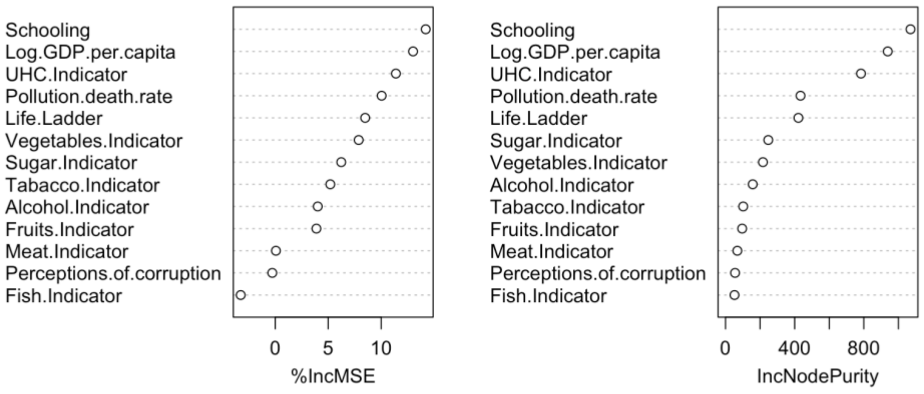


FIGURE 34 - RANDOM FOREST PREDICTORS IMPORTANCE WITH ALL CONSIDERED

In the following section, only the predictors selected in the previous section are considered and the same analysis is carried out (Figure 35).

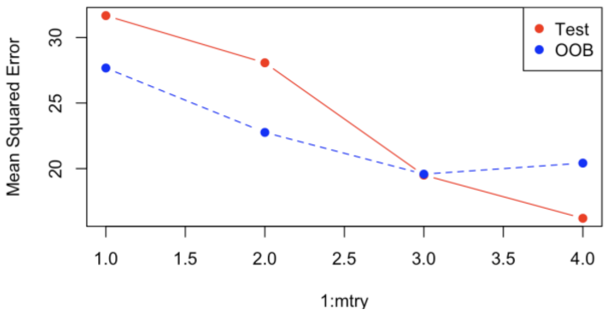
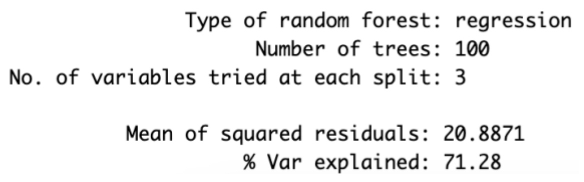


FIGURE 35 - MSE FOR DIFFERENT MTRY CONSIDERING THE SUBSET OF PREDICTORS

This time three is chosen as the number of randomly selected variables at each split, and the following result is obtained (Figure 36):



```
               Type of random forest: regression
                     Number of trees: 100
No. of variables tried at each split: 3

          Mean of squared residuals: 20.8871
                    % Var explained: 71.28
```

FIGURE 36 - RANDOM FOREST WITH A SUBSET OF PREDICTORS

The Root Mean Squared Error (RMSE) on the test dataset is about 4.52.

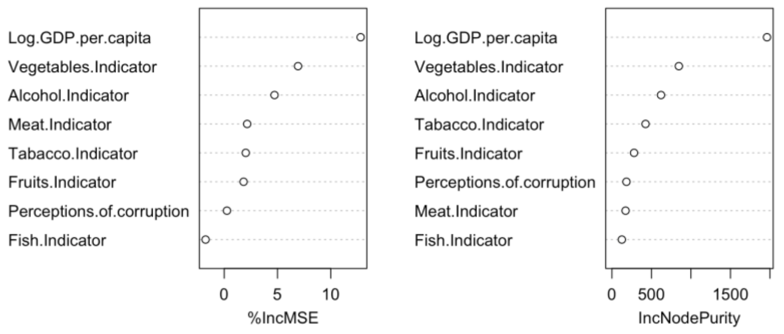


FIGURE 37 - RANDOM FOREST PREDICTORS IMPORTANCE CONSIDERING THE SUBSET

Also in this case by using all the features a better performance is obtained and similar results to those obtained with the decision tree algorithm are obtained in terms of the features importance (Figure 34).

Boosting

Boosting is a machine learning ensemble technique that combines the predictions of multiple weak models (typically decision trees) each new model focuses on correcting the errors of the previous model gradually minimizing the loss by adjusting the predictions in the direction that reduces the gradient (i.e., the errors) at a certain learning rate.

Since the response variable is not normally distributed and the gbm() function in R require a pre-defined loss functions such as Gaussian (for regression), in this analysis xgboost() function will be used which offer a greater flexibility and ability to capture complex relationships in the data. The model will be trained with 1000 trees (weak learners), a learning rate of 0.01 and a maximum tree depth of 4.

As before, the same analysis is carried out with all the features and then with the selected features only.

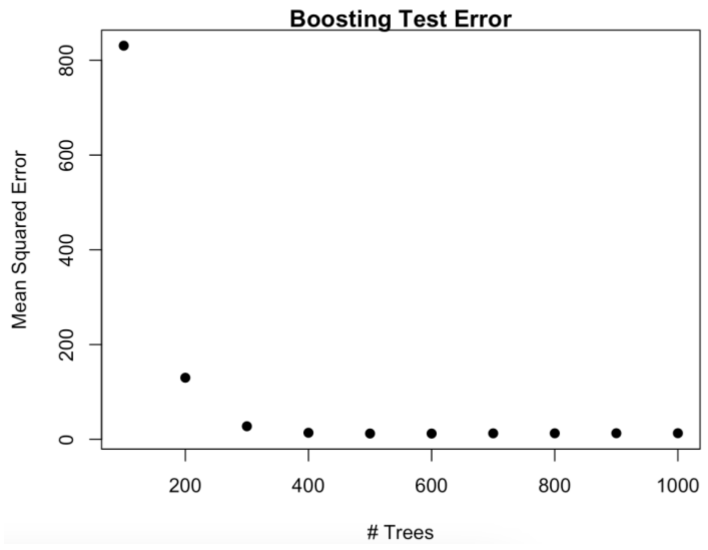


FIGURE 38 - BOOSTING FEATURE IMPORTANCE CONSIDERING ALL THE PREDICTORS

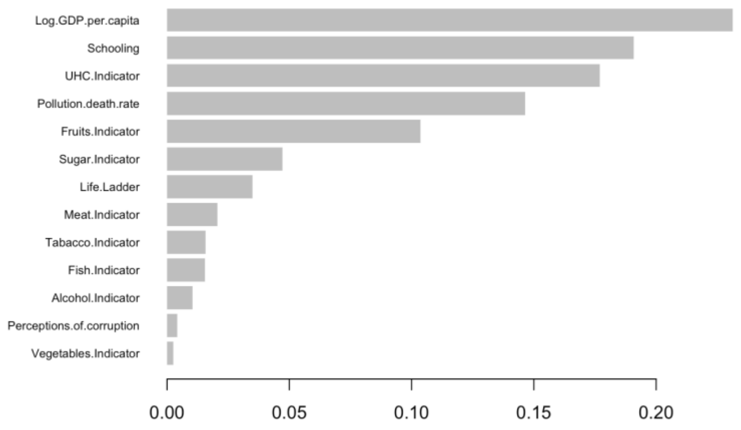


FIGURE 39 - MSE FOR DIFFERENT NUMBER OF TREES CONSIDERING ALL THE PREDICTORS

By analyzing the test error on the basis of the number of trees considered to predict new observations the lowest Test Error occurs with 600 trees when the Root Mean Squared Error (RMSE) is around 3.49.

Furthermore, in terms of features importance similar results are obtained as in the previous sections with the same first three features: Schooling, UHC.Indicator and GDP per capita.

By training the model with only the selected predictors the following results are obtained (Figure 40) (Figure 41):

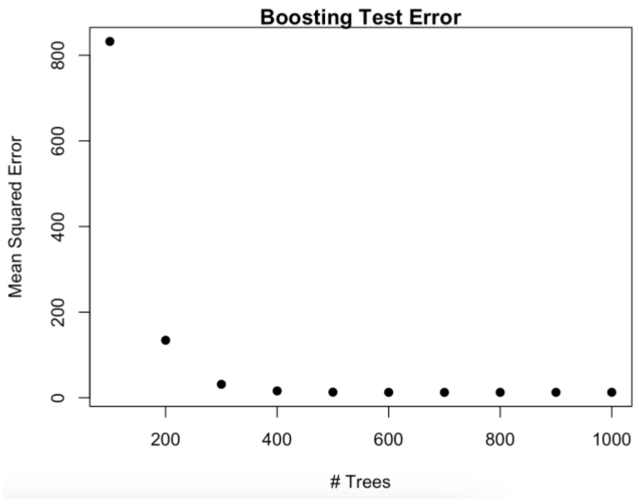


FIGURE 40 - BOOSTING FEATURE IMPORTANCE CONSIDERING THE SUBSET OF PREDICTORS

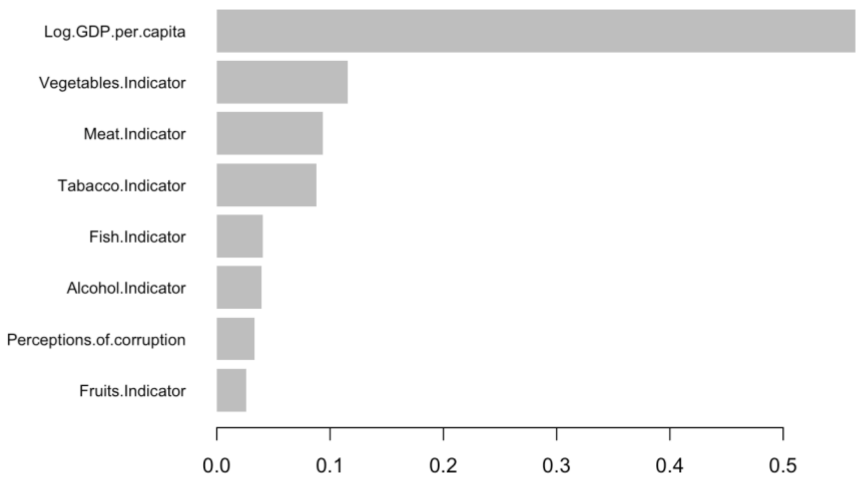


FIGURE 41 - BOOSTING MSE FOR DIFFERENT NUMBER OF TREES CONSIDERING THE SUBSET OF PREDICTORS

By analyzing the test error on the basis of the number of trees considered to predict new observations the lowest Test Error occurs with 700 trees when the Root Mean Squared Error (RMSE) is around 3.55. GDP per capita and Vegetables Indicator are still among the most important features. On the contrary Alcohol Indicator selected by the Random Forest trained with only the selected features was replaced by *Meat. Indicator*.

A better performance is obtained by using all the features to train the model.

Unsupervised learning

In this section, unsupervised learning models will be used to group different countries with similar characteristics into clusters and see how changes the life expectancy range between clusters. This analysis can help in identifying countries where some interventions are needed in order to increase the life expectancy age.

The unsupervised learning models used are:

- K-Means: the goal of k-means is to partition the data into distinct, non-overlapping groups, where each group represents a cluster. The algorithm tries to minimize the distance between the data points within the same cluster (intra-cluster distance) while maximizing the distance between different clusters (inter-cluster distance).
- Hierarchical Clustering: a method of grouping similar data points into clusters in a hierarchical manner. It creates a tree-like structure called a dendrogram, which illustrates the relationships between data points and clusters at different levels of granularity.

Before applying them, a data normalization is needed to ensure that features with different scales do not disproportionately influence the distance calculation. Also, the life expectancy variable is omitted during the model construction. It will be considered later to analyze the life expectancy ranges differences between the groups.

K-means

In order to apply K-means with the best number of cluster the within groups sum of squares (WSS) will be calculated for the different number of clusters. WSS quantifies the variation of data points within each cluster around the cluster's centroid. To select the number of cluster the elbow method will be used which consists in identifying the "elbow" point, where the rate of decrease in WSS starts to slow down, and adding more clusters beyond the elbow point doesn't significantly improve the model's fit. In this analysis, by using all the features two is identified as the best number of clusters.

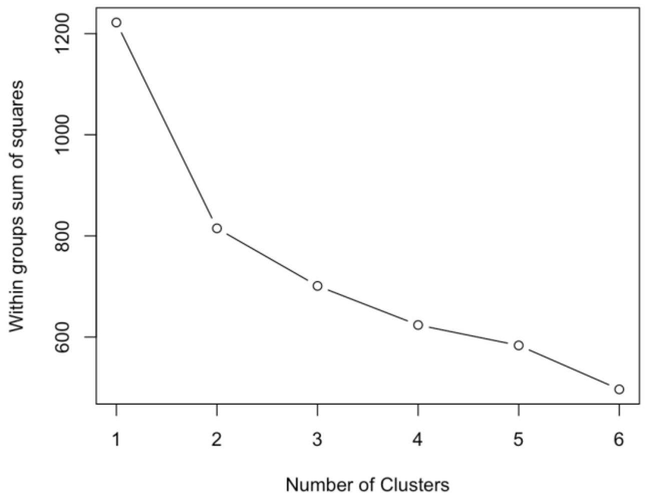


FIGURE 42 - K-MEANS WSS CONSIDERING ALL THE PREDICTORS

K-means has been run selecting two as the number of clusters (Figure 43).

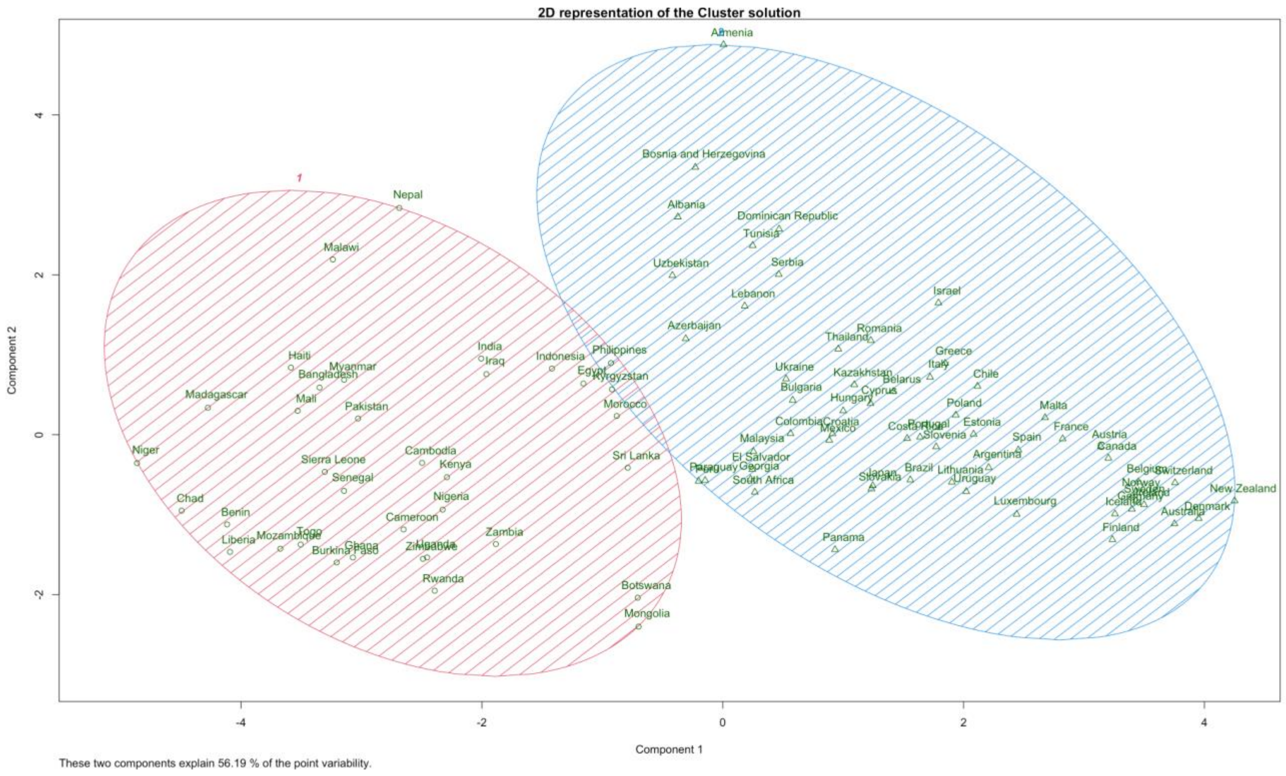
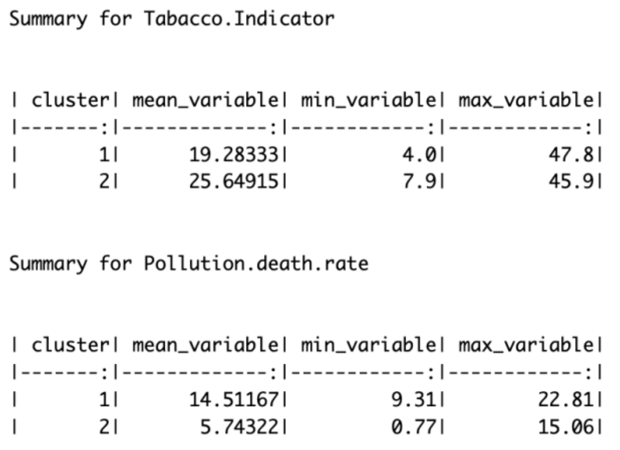


FIGURE 43 - K-MEANS CLUSTERS CONSIDERING ALL THE PREDICTORS

```
Summary for Life.expectancy


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      64.52778|         51.0|           79|
|       2|      78.14237|         62.9|           88|


Summary for Schooling


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      10.62778|          5.4|         14.8|
|       2|      15.38305|         12.1|         20.4|


Summary for Sugar.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      14.36889|         4.28|        35.42|
|       2|      31.91102|        14.23|        50.92|

Summary for Fish.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|     1.8152778|            0|        13.44|
|       2|     0.7979661|            0|         8.58|


Summary for Life.Ladder


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      4.362389|        2.702|        5.547|
|       2|      6.085458|        3.965|        7.603|


Summary for Perceptions.of.corruption


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|     0.7908056|        0.095|        0.946|
|       2|     0.6997288|        0.186|        0.962|
```

```
Summary for Vegetables.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      39.87222|         5.64|       124.59|
|       2|      77.27203|        11.26|       279.06|


Summary for Fruits.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      19.17556|         0.69|        92.73|
|       2|      26.83085|         4.76|       116.93|


Summary for Meat.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|     1.6536111|            0|        12.72|
|       2|     0.9267797|            0|         5.43|

Summary for Log.GDP.per.capita


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      8.153111|        6.935|        9.724|
|       2|     10.154424|        8.714|       11.617|


Summary for Alcohol.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      4.414194|        0.021|        15.14|
|       2|      8.792034|        0.850|        15.38|


Summary for UHC.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      48.50000|           24|           69|
|       2|      75.15254|           58|           88|
```



```
Summary for Tabacco.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      19.28333|          4.0|         47.8|
|       2|      25.64915|          7.9|         45.9|


Summary for Pollution.death.rate


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      14.51167|         9.31|        22.81|
|       2|       5.74322|         0.77|        15.06|
```

FIGURE 44 - SUMMARY FOR ALL THE PREDICTORS

From the analysis carried out it is possible to identify how the features distribution change between the two clusters and what are the countries present in each of them. The main differences between the two clusters are in:

- The mean of Life Expectancy that differs of about 14 years
- *UHC indicator* which is higher in the cluster with a higher life expectancy
- Alcohol, tabacco and sugar usages that are higher for the group characterized by higher range of life expectancy suggesting that some unhealthy habits do not have a high negative impact on life expectancy.
- The *Schooling* and *Life.Ladder* values that lead to consider the fact that higher education and being satisfied about life can make difference. This may be due to the fact that people with higher education tend to have more knowledge about the different diseases which can help in preventing their further development at early stages, and they usually live in a country with a good healthcare system which help them in the cure of the disease. Also, a positive impact of the *Life.Ladder* may suggest a psychological impact on the life expectancy and that people more positive about their life, with less concerns live a longer life.
- Vegetables and Fruits that are more consumed in countries with higher Life Expectancy, and the opposite happens for Meat and Fish consumed more in countries lower Life Expectancy.
- Higher death rate due to pollution where the life expectancy is lower

The same analysis is run with just the selected features from the previous section. The elbow point in this case is not distinctly evident (Figure 45). However, when experimenting with different numbers of clusters, it becomes apparent that using just two clusters results in less overlapping feature ranges.

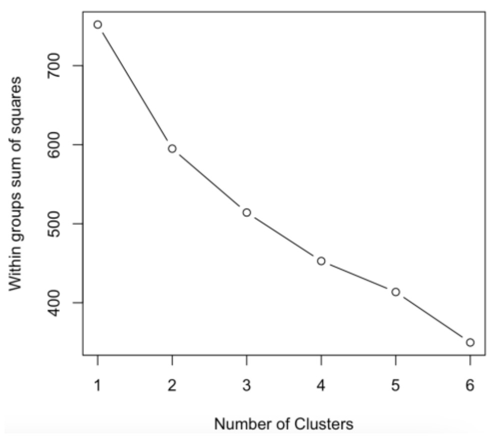


FIGURE 45 - K-MEANS WSS CONSIDERING THE SUBSET OF PREDICTORS

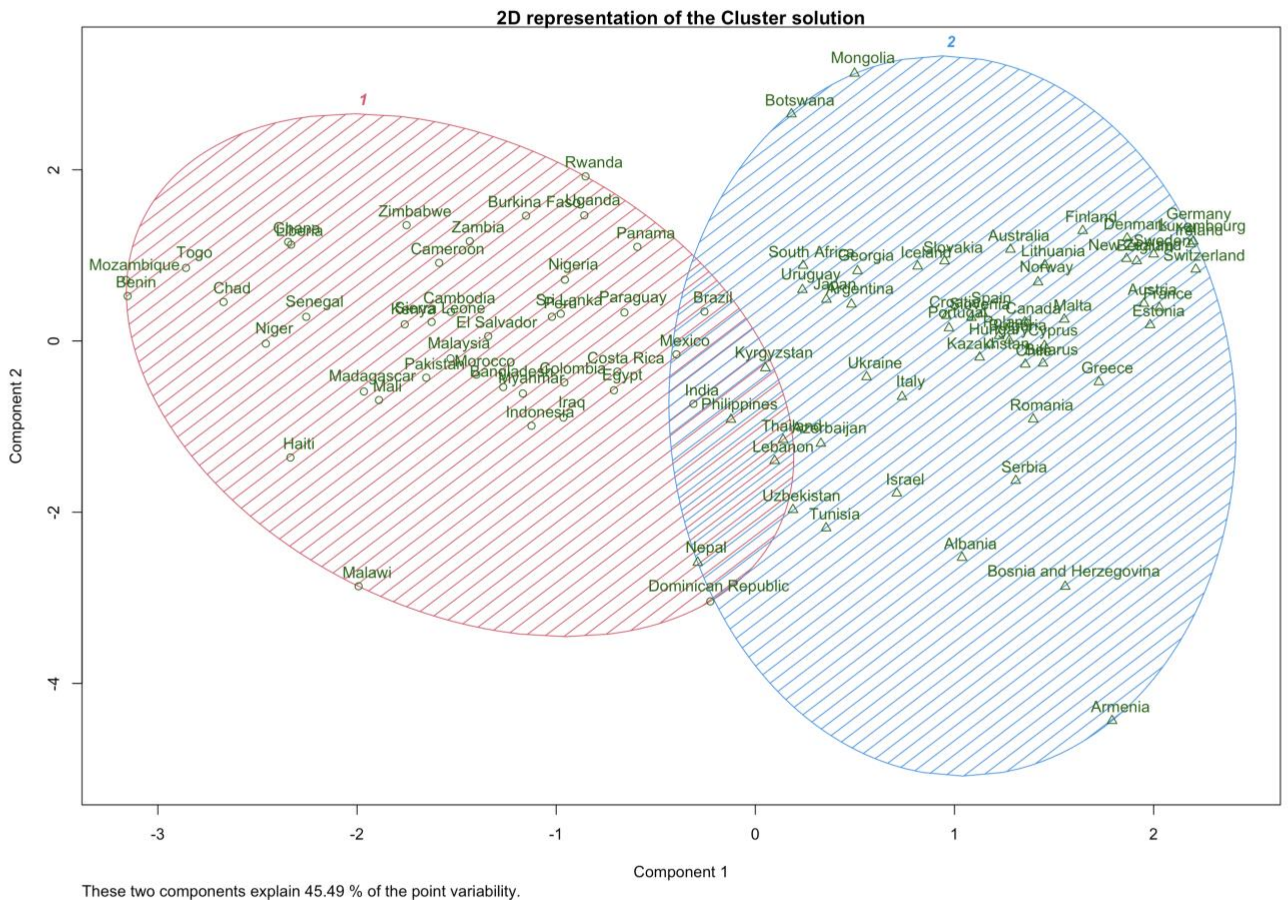
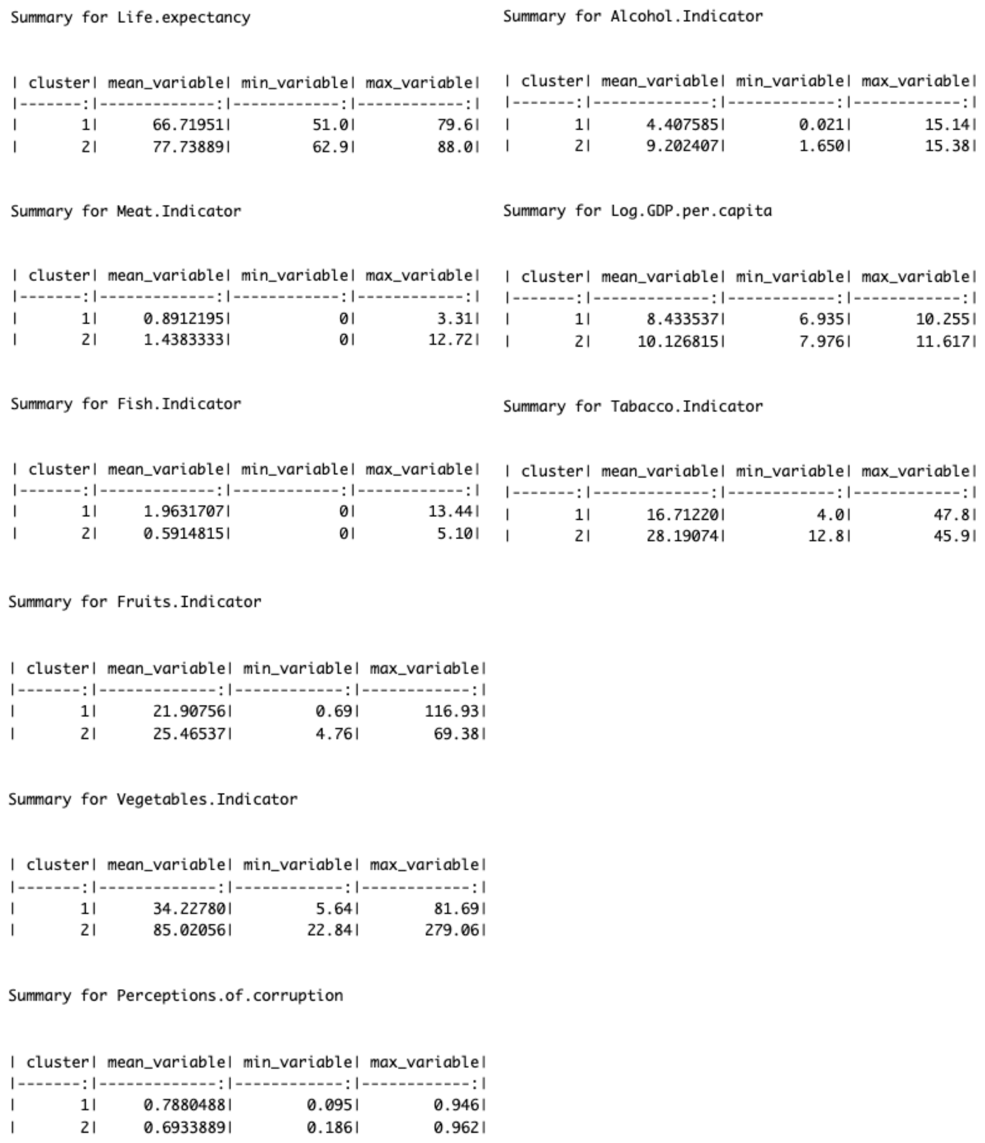


FIGURE 46 - K-MEANS CLUSTERS CONSIDERING THE SUBSET OF PREDICTORS



```
Summary for Life.expectancy


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      66.71951|         51.0|         79.6|
|       2|      77.73889|         62.9|         88.0|
```

```
Summary for Alcohol.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      4.407585|        0.021|        15.14|
|       2|      9.202407|        1.650|        15.38|
```

```
Summary for Meat.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|     0.8912195|            0|         3.31|
|       2|     1.4383333|            0|        12.72|
```

```
Summary for Log.GDP.per.capita


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      8.433537|        6.935|       10.255|
|       2|     10.126815|        7.976|       11.617|
```

```
Summary for Fish.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|     1.9631707|            0|        13.44|
|       2|     0.5914815|            0|         5.10|
```

```
Summary for Tabacco.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      16.71220|          4.0|         47.8|
|       2|      28.19074|         12.8|         45.9|
```

```
Summary for Fruits.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      21.90756|         0.69|       116.93|
|       2|      25.46537|         4.76|        69.38|
```

```
Summary for Vegetables.Indicator


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|      34.22780|         5.64|        81.69|
|       2|      85.02056|        22.84|       279.06|
```

```
Summary for Perceptions.of.corruption


| cluster| mean_variable| min_variable| max_variable|
|-------:|-------------:|------------:|------------:|
|       1|     0.7880488|        0.095|        0.946|
|       2|     0.6933889|        0.186|        0.962|
```

FIGURE 47 - SUMMARY FOR THE SUBSET OF PREDICTORS

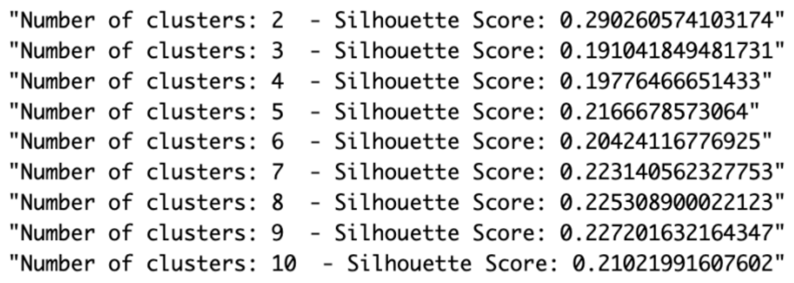
From the analysis carried out it is possible to identify a slightly different results from the ones obtained by using all the features. First, a slightly lower mean of life expectancy that now differs of about 10 years between the two clusters. The difference in mean values of all the features are similar between the two clusters to the ones obtained considering all the features, exept for the meat indicator which in this case is lower for the cluster having lower mean life expectancy. This may be due to the presence of some countries with a high meat indicator and that it may be not relevant in determining the life expectancy.

Hierarchical Cluster Analysis

In this report, hclust() function will be used to perform hierarchical clustering which implements an agglomerative approach. This approach starts with individual data points as separate clusters and iteratively merges the closest clusters until all data points are in a single cluster. It is the most common type of hierarchical clustering. The divisive approach instead starts with all data points in a single cluster and recursively divides clusters into smaller ones until each data point is in its own cluster.

A Silhouette Analysis will be performed for different numbers of clusters. This method provides a measure of how well-separated the clusters are. A silhouette score ranges from -1 to 1, where a higher value indicates that the object is well-matched to its own cluster and poorly matched to neighboring clusters.

First, all the features are used to perform the algorithm. During the clustering process ward.D was used as the linkage method used to calculate the distance between clusters.



```
"Number of clusters: 2  - Silhouette Score: 0.290260574103174"
"Number of clusters: 3  - Silhouette Score: 0.191041849481731"
"Number of clusters: 4  - Silhouette Score: 0.19776466651433"
"Number of clusters: 5  - Silhouette Score: 0.2166678573064"
"Number of clusters: 6  - Silhouette Score: 0.20424116776925"
"Number of clusters: 7  - Silhouette Score: 0.223140562327753"
"Number of clusters: 8  - Silhouette Score: 0.225308900022123"
"Number of clusters: 9  - Silhouette Score: 0.227201632164347"
"Number of clusters: 10  - Silhouette Score: 0.21021991607602"
```

FIGURE 48 - SILHOUETTE SCORE CONSIDERING ALL THE PREDICTORS

Because of the above results, two is selected as the number of clusters to build the model.

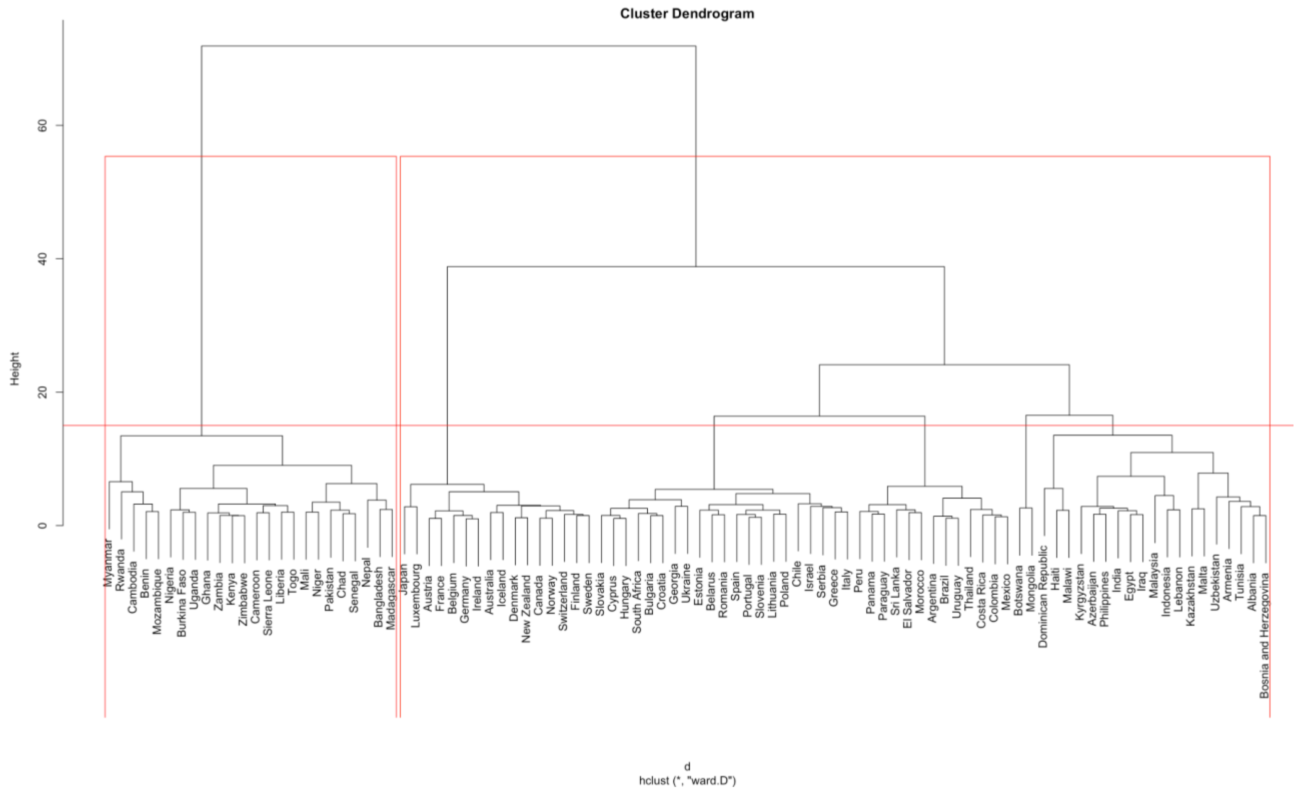
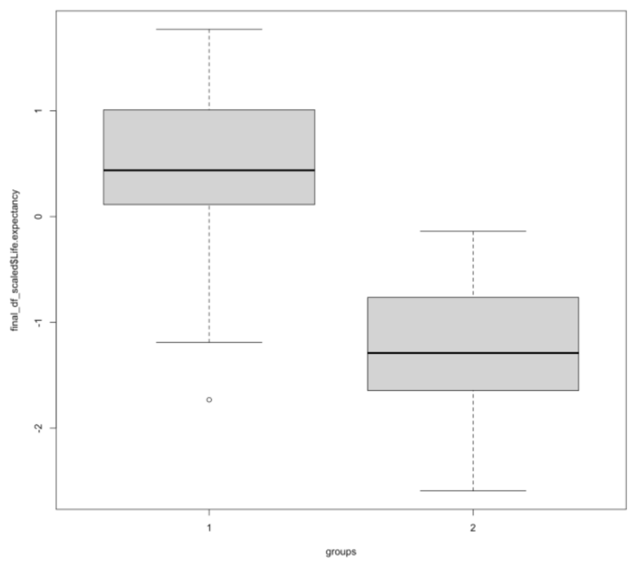
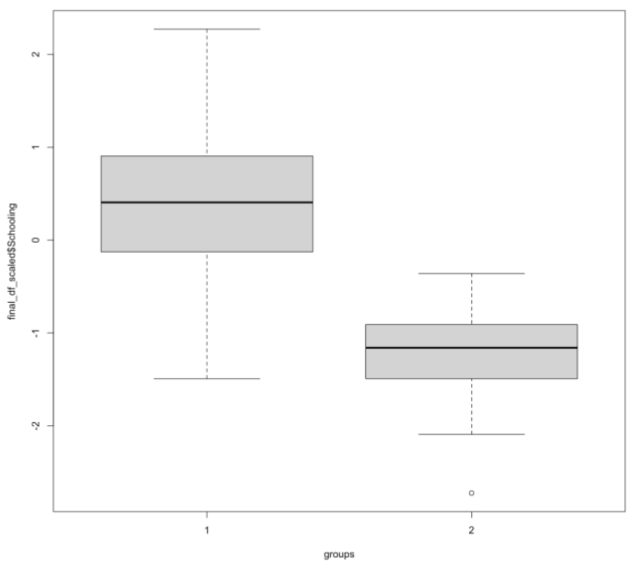
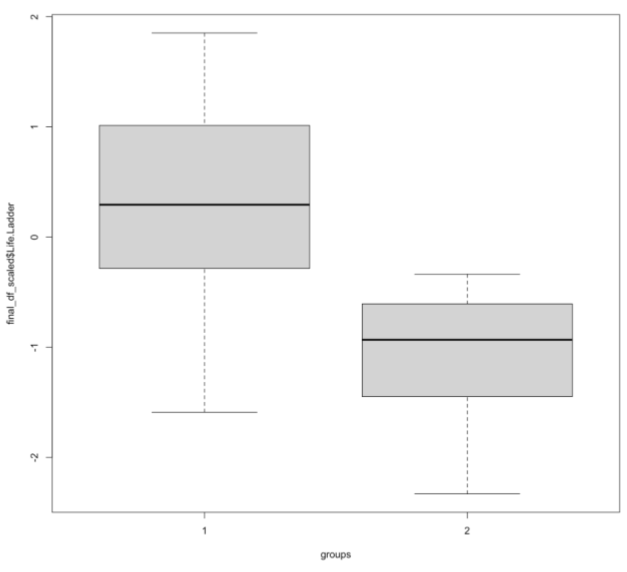
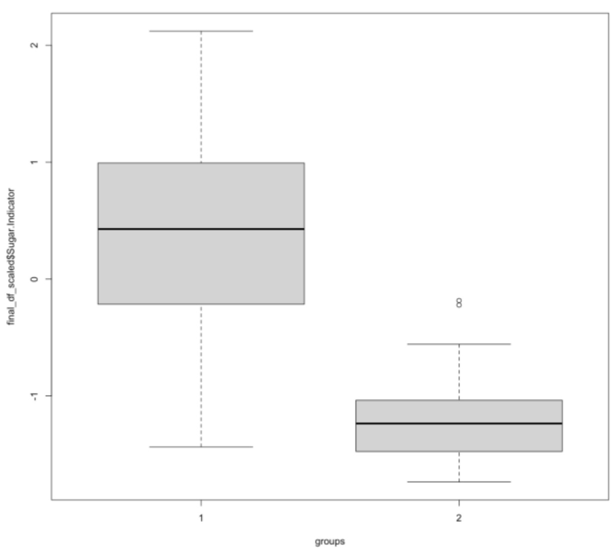


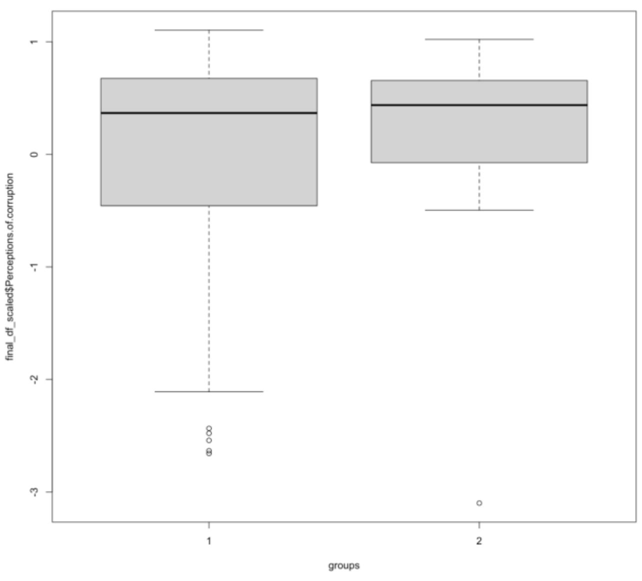
FIGURE 49 - HIERARCHICAL CLUSTERING CONSIDERING ALL THE PREDICTORS

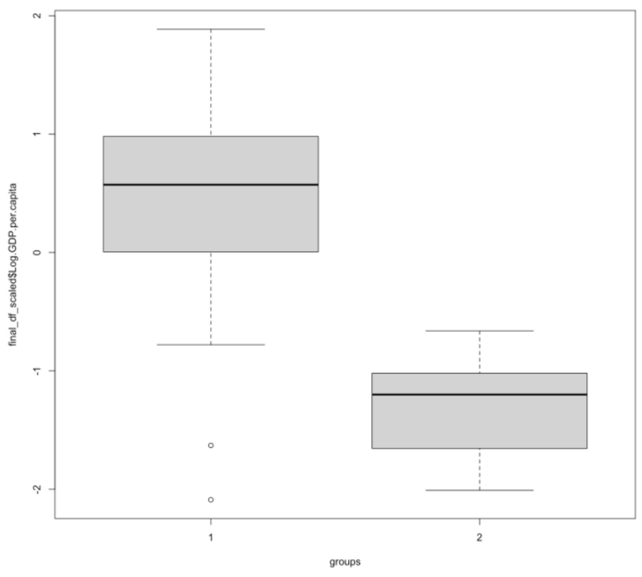


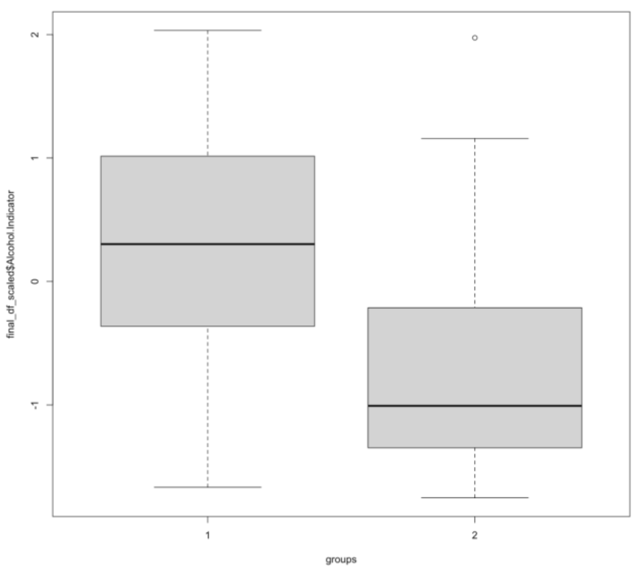


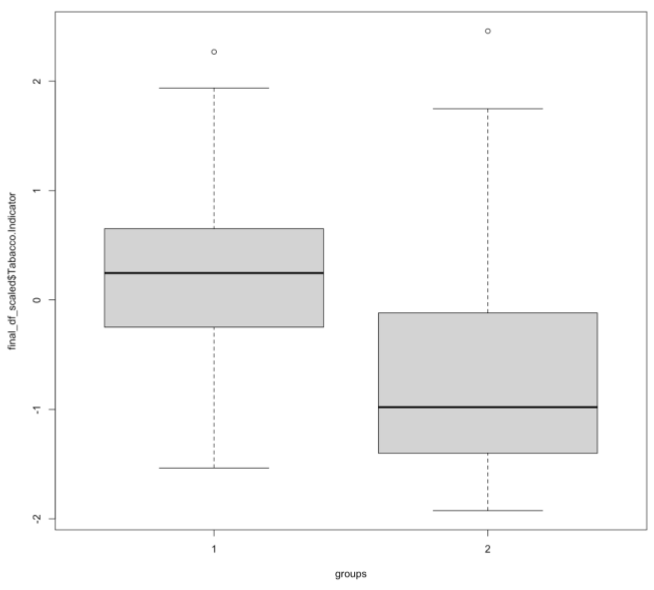

final_df_scaled$Life.Ladder
groups
1
2


final_df_scaled$Sugar.Indicator
groups
1
2


final_df_scaled$Perceptions.of.corruption
groups
1
2


final_df_scaled$Log.GDP.per.capita
groups
1
2


final_df_scaled$Alcohol.Indicator
groups
1
2


final_df_scaled$Tabacco.Indicator
groups
1
2

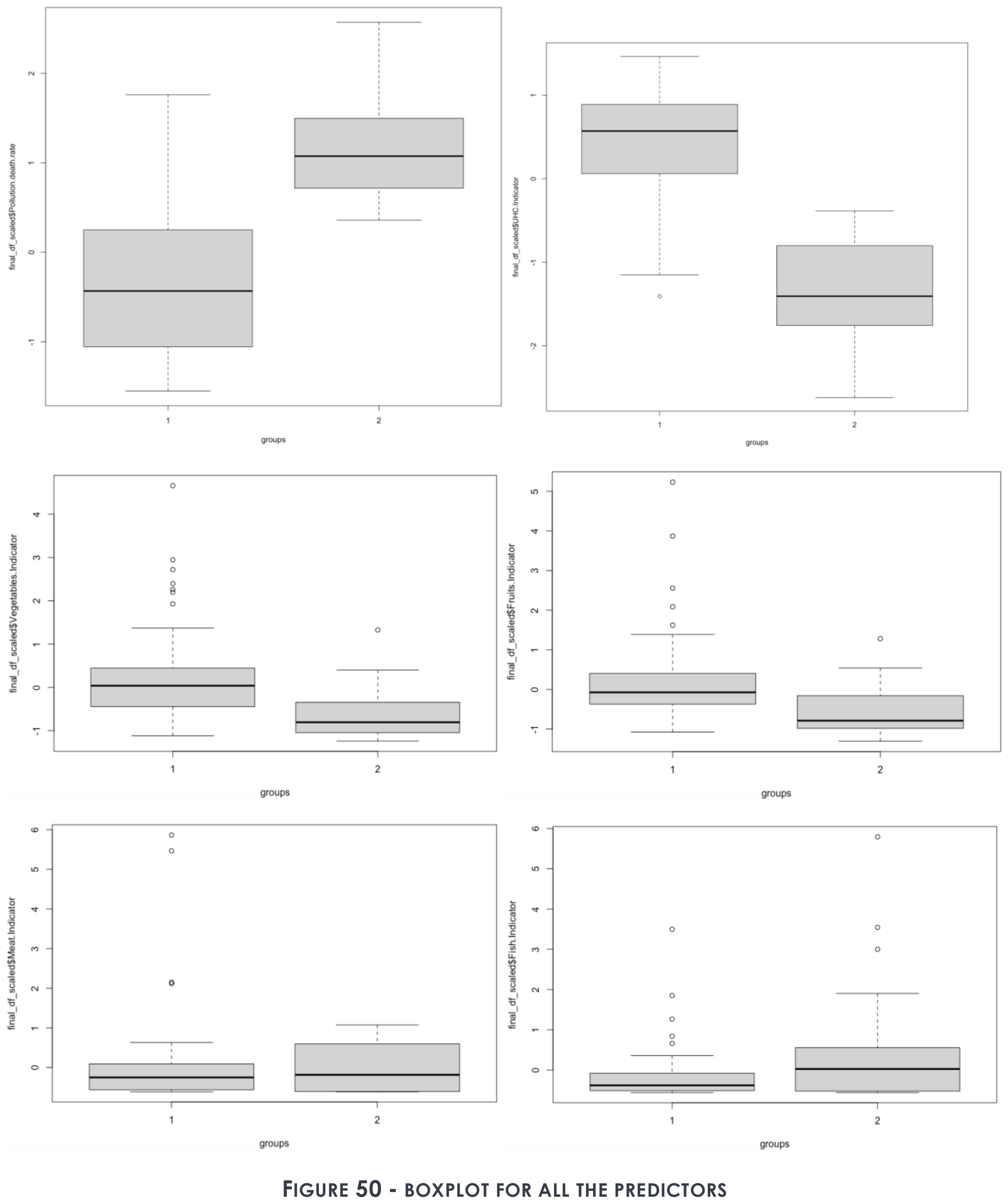
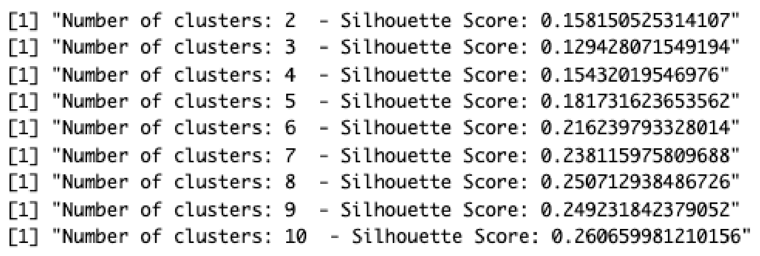


FIGURE 50 - BOXPLOT FOR ALL THE PREDICTORS

The Boxplot() function was used to analyze the distribution of the predictors in the two clusters. Similar conclusions can be drawn from this analysis as for the k-means clustering. Furthermore, it can be seen that when it comes to the indicators for Vegetables, Fruits, Meat, and Fish, there are some outliers. This might be a reason for the two different conclusions about the Meat.Indicator obtained in the previous section.

The same analysis is carried out with just the selected features.



```
[1] "Number of clusters: 2  - Silhouette Score: 0.158150525314107"
[1] "Number of clusters: 3  - Silhouette Score: 0.129428071549194"
[1] "Number of clusters: 4  - Silhouette Score: 0.15432019546976"
[1] "Number of clusters: 5  - Silhouette Score: 0.181731623653562"
[1] "Number of clusters: 6  - Silhouette Score: 0.216239793328014"
[1] "Number of clusters: 7  - Silhouette Score: 0.238115975809688"
[1] "Number of clusters: 8  - Silhouette Score: 0.250712938486726"
[1] "Number of clusters: 9  - Silhouette Score: 0.249231842379052"
[1] "Number of clusters: 10  - Silhouette Score: 0.260659981210156"
```

FIGURE 51 - SILHOUETTE SCORE CONSIDERING THE SUBSET OF PREDICTORS

Because of the Silhouette Score, ten is chosen as the number of clusters to build the final model. However, as can be seen below, it produces clusters with overlapping value ranges of life expectancy and all other features, making it difficult to analyze.

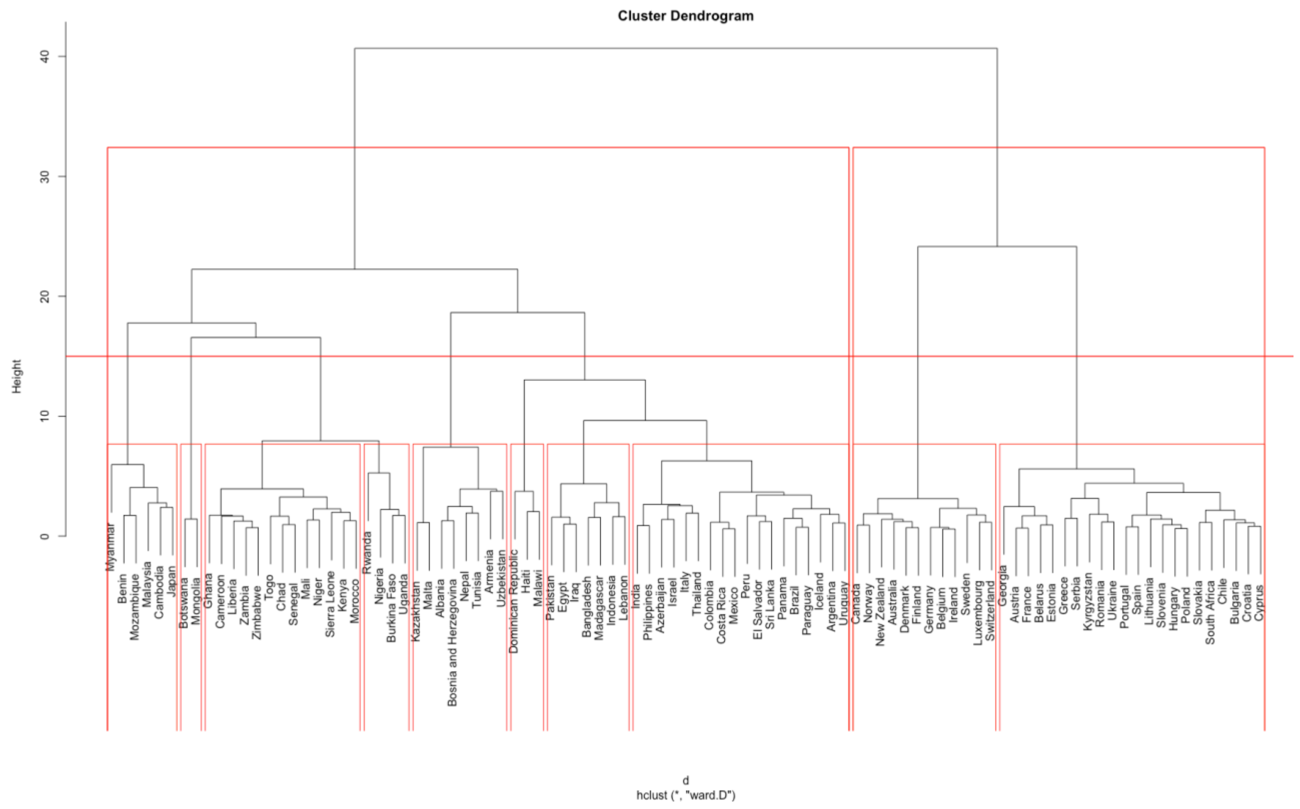


FIGURE 52 - HIERARCHICAL CLUSTERING CONSIDERING THE SUBSET OF PREDICTORS

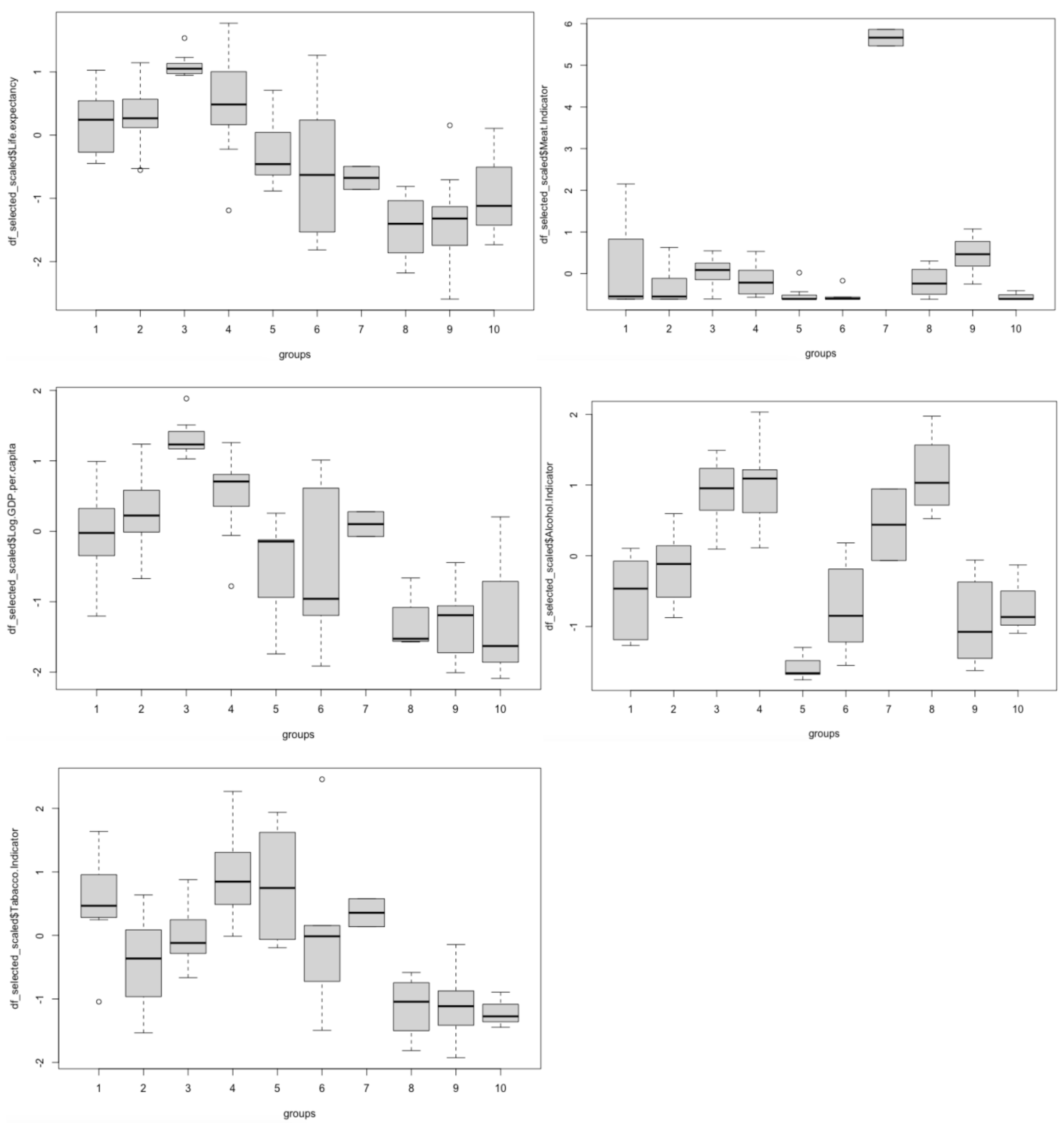


FIGURE 53 - BOXPLOT FOR THE SUBSET OF PREDICTORS

Conclusions

In conclusion, regarding the first objective of this report which consists in analyzing the different non-parametric models and their performance to predict the Life expectancy variable, the following RMSE results were obtained:

- Decision tree
  - With all the predictors by using ctree function: 3.798
  - With the selected predictors by using ctree function: 4.02
  - With all the predictors by using rpart function: 3.717
  - With the selected predictors by using rpart function: 4.325

- Random Forest
  - With all the predictors: 2.59
  - With the selected predictors: 4.52
- Boosting
  - With all the predictors: 3.49
  - With the selected predictors: 3.55

In general, by using ensemble methods the test error tends to be lower and this is probably due to the higher randomness introduce by the algorithms and consequently lower variance. Lower RMSEs are obtained with models trained with all the predictors. The best performance is obtained by using Random Forest with all the predictors. In this case, the significant features for determining the life expectancy are Schooling, GDP per capita and UHC Indicator.

Regarding the second objective of this report, which consists of finding a model to determine the group of countries with characteristics that lead to a higher life expectancy and those countries where some interventions are needed to improve it, similar results were obtained with both k-means and hierarchical clustering. For K-means, two clusters were built from both the entire dataset and the dataset with just the selected most important features. On the other hand, for hierarchical clustering, two clusters were built considering the entire dataset, and ten considering the selected most important features.

A better classification, based on the difference in life expectancy means across clusters, is achieved by considering the entire dataset. Generally, when examining the entire dataset and constructing two clusters, it becomes evident that factors such as a country's healthcare system, higher education, and life satisfaction positively influence life expectancy. Surprisingly, habits like alcohol, tobacco, and sugar consumption do not appear to have a significant negative impact. Conversely, a country's pollution levels appear to lower life expectancy. Additionally, initial analysis suggests that countries with higher life expectancy tend to consume more vegetables and fruits, while those with lower life expectancy lean towards higher consumption of meat and fish. However, it's worth noting that these variables may contain outliers, which could affect the accuracy of this observation. In fact, in case of the K-means run on the subset of predictors an opposite resulti s obtained for the Meat.Indicator which is lower for the cluster having lower life expectancy.

Appendix

In this section, there is the R code used in this report.

```
#setwd("~/Documents/university/Statistical learning")

library(dplyr)
library(tidyr)
library(outliers)
library(MASS)
library(ggpubr)
library(corrplot)
library(ggplot2)
library(car)
library(rpart.plot)
library(party)
library("partykit")
library(rpart)
library(leaps)
library(randomForest)
require(randomForest)
require(gbm)
library(cluster)
library(knitr)

#read data
df_life_expectancy = read.csv("life_expectancy.csv")
df_diet= read.csv("diet.csv")
df_happiness=read.csv("world-happiness-report.csv")
df_alcohol=read.csv("AlcoholSubstanceAbuse.csv")
df_uhc=read.csv("uhcCoverage.csv")
df_tabacco=read.csv("tobaccoAge15.csv")
df_pollution=read.csv("share-deaths-air-pollution.csv")


#Select year=2015
df_life_expectancy=df_life_expectancy[df_life_expectancy$Year=="2015",]
df_happiness=df_happiness[df_happiness$year=="2015",]
df_uhc=df_uhc[df_uhc$Period=="2015",]

#Select year=2015 and the indicators values both sexes
df_alcohol=df_alcohol[df_alcohol$Period=="2015" & df_alcohol$Dim1=="Both
sexes",]
df_tabacco=df_tabacco[df_tabacco$Period=="2015" & df_tabacco$Dim1=="Both
sexes",]
df_pollution=df_pollution[df_pollution$Year=="2015",]


#Select the appropriate columns
subset_fields <- c("Country", "Schooling", "Life.expectancy")
df_life_expectancy <- df_life_expectancy[,subset_fields]

subset_fields <- c("Country.name", "Life.Ladder",
"Perceptions.of.corruption", "Log.GDP.per.capita")
df_happiness <- df_happiness[, subset_fields]

subset_fields <- c("Location", "First.Tooltip")
df_alcohol <- df_alcohol[,subset_fields]
```

```r
subset_fields <- c("Location", "First.Tooltip")
df_uhc <- df_uhc[, subset_fields]

subset_fields <- c("Location", "First.Tooltip")
df_tabacco <- df_tabacco[, subset_fields]

subset_fields <- c("Area", "Item", "Value")
df_diet <- df_diet[, subset_fields]

subset_fields <- c("Entity",
"Deaths...Cause..All.causes...Risk..Air.pollution...Sex..Both...Age..Age.sta
ndardized..Percent.")
df_pollution <- df_pollution[, subset_fields]


#rename the columns
names(df_diet)[names(df_diet) == "Area"] <- "Country"
names(df_diet)[names(df_diet) == "Item"] <- "Diet.Item"
names(df_happiness)[names(df_happiness) == "Country.name"] <- "Country"
names(df_alcohol)[names(df_alcohol) == "Location"] <- "Country"
names(df_alcohol)[names(df_alcohol) == "First.Tooltip"] <-
"Alcohol.Indicator"
names(df_uhc)[names(df_uhc) == "Location"] <- "Country"
names(df_uhc)[names(df_uhc) == "First.Tooltip"] <- "UHC.Indicator"
names(df_tabacco)[names(df_tabacco) == "Location"] <- "Country"
names(df_tabacco)[names(df_tabacco) == "First.Tooltip"] <-
"Tabacco.Indicator"
names(df_pollution)[names(df_pollution) == "Entity"] <- "Country"
names(df_pollution)[names(df_pollution) ==
"Deaths...Cause..All.causes...Risk..Air.pollution...Sex..Both...Age..Age.sta
ndardized..Percent."] <- "Pollution death rate"
names(df_pollution)[names(df_pollution) == "Pollution death rate"] <-
"Pollution.death.rate"


#transform diet dataframe row values of a column into new columns
df_diet <- pivot_wider(df_diet, id_cols = Country, names_from = "Diet.Item",
values_from = Value)

#rename the columns
names(df_diet)[names(df_diet) == "Meat, Other"] <- "Meat.Indicator"
names(df_diet)[names(df_diet) == "Fruits, other"] <- "Fruits.Indicator"
names(df_diet)[names(df_diet) == "Vegetables, other"] <-
"Vegetables.Indicator"
names(df_diet)[names(df_diet) == "Marine Fish, Other"] <- "Fish.Indicator"
names(df_diet)[names(df_diet) == "Sugar (Raw Equivalent)"] <-
"Sugar.Indicator"

#Reorder the columns so that the Life.expectancy is the first one
df_life_expectancy <- df_life_expectancy[,c("Life.expectancy", "Schooling",
"Country")]

#join datasets by country
final_df <- df_life_expectancy %>%
  inner_join(df_diet, by = "Country") %>%
  inner_join(df_happiness, by = "Country") %>%
  inner_join(df_alcohol, by = "Country") %>%
  inner_join(df_uhc, by = "Country") %>%
  inner_join(df_tabacco, by = "Country") %>%
  inner_join(df_pollution, by = "Country")

############################################
#Data Cleaning

#Count missing values in each column
missing_count <- colSums(is.na(final_df))

#Total count of missing values in the entire data frame
total_missing_count <- sum(missing_count)

print(total_missing_count)
print(missing_count)

#Remove rows with missing values
final_df <- na.omit(final_df)

# Create a dataframe without the country column
final_df_no_country=final_df %>% select_if(is.numeric)

#Normality assumption check
shapiro.test(final_df_no_country$Life.expectancy)

ggdensity(final_df_no_country, x = "Life.expectancy", fill = "lightgray",
title = "Life.expectancy") +
  stat_overlay_normal_density(color = "red", linetype = "dashed")

#Log transform the response variable
logexpactancy=log(final_df_no_country$Life.expectancy)
shapiro.test(logexpactancy)

#Box-cox transformation of the response variable
response_variable<-final_df_no_country$Life.expectancy
boxcox_output <- boxcox(lm(response_variable ~ 1))
#Extract the optimal lambda value
lambda <- boxcox_output$x[which.max(boxcox_output$y)]
#Apply the Box-Cox transformation using the optimal lambda
transformed_response <- if (abs(lambda) < 1e-6) log(response_variable) else
(response_variable^lambda - 1) / lambda
shapiro.test(transformed_response)

# Normality assumption check for the residuals
lm_model <- lm(Life.expectancy ~., data = final_df_no_country)
residuals <- residuals(lm_model)
shapiro_test_result <- shapiro.test(residuals)
print(shapiro_test_result)

# Linearity assumption check
plot(Life.expectancy~Schooling, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
mod<-lm(Life.expectancy~Schooling, data=final_df_no_country)
abline(mod, col="red", lwd=3)
```

```
plot(Life.expectancy~Alcohol.Indicator, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
mod<-lm(Life.expectancy~Alcohol.Indicator, data=final_df_no_country)
abline(mod, col="red", lwd=3)

plot(Life.expectancy~Sugar.Indicator, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
mod<-lm(Life.expectancy~Sugar.Indicator, data=final_df_no_country)
abline(mod, col="red", lwd=3)

plot(Life.expectancy~Fish.Indicator, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
mod<-lm(Life.expectancy~Fish.Indicator, data=final_df_no_country)
abline(mod, col="red", lwd=3)

plot(Life.expectancy~Vegetables.Indicator, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
mod<-lm(Life.expectancy~Vegetables.Indicator, data=final_df_no_country)
abline(mod, col="red", lwd=3)

plot(Life.expectancy~Fruits.Indicator, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
mod<-lm(Life.expectancy~Fruits.Indicator, data=final_df_no_country)
abline(mod, col="red", lwd=3)

plot(Life.expectancy~Meat.Indicator, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
mod<-lm(Life.expectancy~Meat.Indicator, data=final_df_no_country)
abline(mod, col="red", lwd=3)

plot(Life.expectancy~Tabacco.Indicator, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
mod<-lm(Life.expectancy~Tabacco.Indicator, data=final_df_no_country)
abline(mod, col="red", lwd=3)

plot(Life.expectancy~UHC.Indicator, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
mod<-lm(Life.expectancy~UHC.Indicator, data=final_df_no_country)
abline(mod, col="red", lwd=3)

plot(Life.expectancy~Pollution.death.rate, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
mod<-lm(Life.expectancy~Pollution.death.rate, data=final_df_no_country)
abline(mod, col="red", lwd=3)

plot(Life.expectancy~Life.Ladder, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
mod<-lm(Life.expectancy~Life.Ladder, data=final_df_no_country)
abline(mod, col="red", lwd=3)

plot(Life.expectancy~Perceptions.of.corruption, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
mod<-lm(Life.expectancy~Perceptions.of.corruption, data=final_df_no_country)
abline(mod, col="red", lwd=3)

plot(Life.expectancy~Log.GDP.per.capita, col="darkgreen", pch=19,
cex=1,data=final_df_no_country)
```

```
mod<-lm(Life.expectancy~Log.GDP.per.capita, data=final_df_no_country)
abline(mod, col="red", lwd=3)


# Outliers detection
Q1 <- quantile(final_df_no_country$Life.expectancy, 0.25)
Q3 <- quantile(final_df_no_country$Life.expectancy, 0.75)
IQR <- Q3 - Q1
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR
potential_outliers <- which(final_df_no_country$Life.expectancy <
lower_bound | final_df_no_country$Life.expectancy > upper_bound)
print(potential_outliers)

boxplot(final_df_no_country$Life.expectancy)

############################################
#Multicollinearity checks

# Create a dataframe with only the predictors
df_predictors <- final_df_no_country[, !(names(final_df_no_country) %in%
c("Life.expectancy"))]

cor_matrix <- cor(df_predictors)
round(cor_matrix,2)
symnum(cor_matrix, abbr.colnames = FALSE)

corrplot(cor_matrix, type = "upper",
         tl.col = "black", tl.srt = 45, tl.cex = 0.6)

full.model <- lm(Life.expectancy ~., data = final_df_no_country)
sqrt(vif(full.model)) > 2

############################################################################
#
#Evaluation of the different methods to select the most important predictors

######################
# Best subset selection
regfit.full=regsubsets(Life.expectancy~.,data=final_df_no_country, nvmax=13)
reg.summary=summary(regfit.full)

# Identify the best number of variables
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp")
which.min(reg.summary$cp)
plot(regfit.full,scale="Cp")

###########################
#forward stepwise selection
regfit.fwd=regsubsets(Life.expectancy~.,data=final_df_no_country,method="for
ward", nvmax=13)
summary(regfit.fwd)
plot(regfit.fwd,scale="Cp")

#Model Selection Using cross-validation for forward stepwise selection
set.seed(11)
folds=sample(rep(1:5,length=nrow(final_df)))
```

```
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  mat[,names(coefi)]%*%coefi
}

cv.errors=matrix(NA,5,13)

for(k in 1:5){

best.fit=regsubsets(Life.expectancy~.,data=final_df_no_country[folds!=k,],nv
max=13,method="forward")

  for(i in 1:13){
    pred=predict(best.fit,final_df_no_country[folds==k,],id=i)
    cv.errors[k,i]=mean((final_df_no_country$Life.expectancy[folds==k]-
pred)^2)
  }
}
mse.cv=apply(cv.errors,2,mean)
plot(mse.cv,pch=19,type="b")

#Create a dataframe with the most important variables
df_selected <- final_df[, c("Life.expectancy","Schooling", "Meat.Indicator",
"Alcohol.Indicator", "Log.GDP.per.capita", "Tabacco.Indicator",
"UHC.Indicator")]

#check multicollinearity
full.model <- lm(Life.expectancy ~., data = df_selected)
sqrt(vif(full.model)) > 2

#Create a dataframe with the most important variables and without the
response variable
df_predictor_sel <- df_selected[, !(names(df_selected) %in%
"Life.expectancy")]
cor_matrix1 <- cor(df_predictor_sel)
round(cor_matrix1,2)
symnum(cor_matrix1, abbr.colnames = FALSE)

corrplot(cor_matrix1, type = "upper",
         tl.col = "black", tl.srt = 45, tl.cex = 0.6)


# Create a dataframe with the selection of the most important features on
the basis of VIF results
df_selected <- final_df[, c("Life.expectancy", "Meat.Indicator",
"Alcohol.Indicator", "Log.GDP.per.capita", "Tabacco.Indicator")]

#check multicollinearity
full.model <- lm(Life.expectancy ~., data = df_selected)
sqrt(vif(full.model)) > 2

df_predictor_sel <- df_selected[, !(names(df_selected) %in%
"Life.expectancy")]
cor_matrix1 <- cor(df_predictor_sel)
```

```
round(cor_matrix1,2)
symnum(cor_matrix1, abbr.colnames = FALSE)

corrplot(cor_matrix1, type = "upper",
         tl.col = "black", tl.srt = 45, tl.cex = 0.6)

########################################
# PCA

# Create a dataframe with a scaled variables values - Data Normalization
final_df_scaled <- as.data.frame(scale(final_df_no_country))
final_df_scaled <- final_df_scaled %>% mutate(final_df$Country)

# Set the row names for a dataframe
new_row_names <- final_df$Country
rownames(final_df_scaled) <- new_row_names

PCA_res <- princomp(final_df_scaled[, -c(1, ncol(final_df_scaled))], cor=T)
summary(PCA_res)
screeplot(PCA_res, type = "lines", npcs = min(nrow(final_df_scaled) - 1,
ncol(final_df_scaled) - 2), main = "Explained Variance by component")
abline(h = 1, col = "red", lty = 2)  # Line at eigenvalue 1 (Kaiser
criterion)
grid(col = "lightgray", lty = "dotted")  # Add gridlines

# Calculate and plot cumulative explained variance
eigenvalues <- PCA_res$sdev^2
cumulative_variance <- cumsum(eigenvalues) / sum(eigenvalues)
print(cumulative_variance)
plot(cumulative_variance, type = "b", xlab = "Number of Principal
Components", ylab = "Cumulative Explained Variance", main = "Cumulative
Explained Variance Plot")

#Individuate the most influential features on each  component
# Get the loadings matrix
loadings_matrix <- PCA_res$loadings
print(loadings_matrix)

# Specify the principal component index to get the loadings for the
specified component
component_index <- 1
loadings <- loadings_matrix[, component_index]
sorted_loadings <- sort(abs(loadings), decreasing = TRUE)
influential_variable_indices <- names(sorted_loadings)
print(influential_variable_indices)

# Specify the principal component index to get the loadings for the
specified component
component_index <- 2
loadings <- loadings_matrix[, component_index]
sorted_loadings <- sort(abs(loadings), decreasing = TRUE)
influential_variable_indices <- names(sorted_loadings)
print(influential_variable_indices)


# Specify the principal component index to get the loadings for the
specified component
```

```
component_index <- 3
loadings <- loadings_matrix[, component_index]
sorted_loadings <- sort(abs(loadings), decreasing = TRUE)
influential_variable_indices <- names(sorted_loadings)
print(influential_variable_indices)

# Specify the principal component index to get the loadings for the
specified component
component_index <- 4
loadings <- loadings_matrix[, component_index]
sorted_loadings <- sort(abs(loadings), decreasing = TRUE)
influential_variable_indices <- names(sorted_loadings)
print(influential_variable_indices)


# Calculate the absolute loadings and calculate color intensity based on
loadings
absolute_loadings <- abs(PCA_res$loadings)

# Define a color palette
color_palette <- colorRampPalette(c("#7dfabb", "#071aeb"))(100)
color_intensity <- cut(absolute_loadings, breaks = 100)

# Create a plot
plot(NA, NA, xlim = c(-1, 1), ylim = c(-1, 1), xlab = "PC1", ylab = "PC2",
type = "n")
arrows(0, 0, PCA_res$loadings[, 1], PCA_res$loadings[, 2], length = 0.1,
angle = 15, col = color_palette[color_intensity])
text(PCA_res$loadings[, 1], PCA_res$loadings[, 2], labels =
influential_variable_indices, pos = 3, col = color_palette[color_intensity])
# Add x and y axes
abline(v = 0, h = 0, col = "gray", lty = 2)

# Create an plot
plot(NA, NA, xlim = c(0, 1), ylim = c(0, 1), xlab = "PC1", ylab = "PC2",
type = "n")
arrows(0, 0, PCA_res$loadings[, 1], PCA_res$loadings[, 2], length = 0.05,
angle = 15, col = color_palette[color_intensity])
text(PCA_res$loadings[, 1], PCA_res$loadings[, 2], labels =
influential_variable_indices, pos = 3, col = color_palette[color_intensity])
abline(v = 0, h = 0, col = "gray", lty = 2)


#Select the most important variables considering PCA results
df_selected <- final_df[, c("Life.expectancy", "Meat.Indicator",
"Fish.Indicator","Alcohol.Indicator", "Log.GDP.per.capita",
"Tabacco.Indicator", "Fruits.Indicator", "Vegetables.Indicator",
"Perceptions.of.corruption")]

#check multicollinearity
full.model <- lm(Life.expectancy ~., data = df_selected)
sqrt(vif(full.model)) > 2

df_predictor_sel <- df_selected[, !(names(df_selected) %in%
"Life.expectancy")]
cor_matrix1 <- cor(df_predictor_sel)
round(cor_matrix1,2)
```

```
symnum(cor_matrix1, abbr.colnames = FALSE)

corrplot(cor_matrix1, type = "upper",
         tl.col = "black", tl.srt = 45, tl.cex = 0.6)



#####################################################
# Decision trees

# Decision tree with all the predictors
# Set a seed for reproducibility
set.seed(12)
# Create a random index to split the data (70% for training, 30% for
testing) and  create the training and testing datasets
train_index <- sample(1:nrow(final_df_no_country), 0.7 *
nrow(final_df_no_country))
train_data <- final_df_no_country[train_index, ]
test_data <- final_df_no_country[-train_index, ]

# Fit a decision tree model using the Life.expectancy as the target variable
and the other columns as features
decision_tree_model <- rpart(Life.expectancy ~ ., data = train_data)
rpart.plot(decision_tree_model, main = "Decision Tree for Life Expectancy")
predictions <- predict(decision_tree_model, newdata = test_data, type =
"vector")
actual_labels <- test_data$Life.expectancy

# Calculate RMSE by comparing predicted and actual labels
rmse <- sqrt(mean((predictions - actual_labels)^2))
print(paste("Root Mean Squared Error (RMSE):", rmse))

# ctree
ptree<-ctree(Life.expectancy ~ ., data=train_data, control =
ctree_control(mincriterion=0.95, minsplit=0, minbucket=0))
plot(ptree)
predictions <- predict(ptree, newdata = test_data)

# Calculate RMSE by comparing predicted and actual labels
rmse <- sqrt(mean((predictions - actual_labels)^2))
print(paste("Root Mean Squared Error (RMSE):", rmse))

# Decision tree with the selected predictors
# Set a seed for reproducibility
set.seed(12)
train_index <- sample(1:nrow(df_selected), 0.7 * nrow(df_selected))
train_data <- df_selected[train_index, ]
test_data <- df_selected[-train_index, ]

# Fit a decision tree model using the Life.expectancy as the target variable
and the other columns as features
decision_tree_model <- rpart(Life.expectancy ~ ., data = train_data)
rpart.plot(decision_tree_model, main = "Decision Tree for Life Expectancy")
predictions <- predict(decision_tree_model, newdata = test_data, type =
"vector")
actual_labels <- test_data$Life.expectancy
```

```
# Calculate RMSE by comparing predicted and actual labels
rmse <- sqrt(mean((predictions - actual_labels)^2))
print(paste("Root Mean Squared Error (RMSE):", rmse))

# ctree
ptree<-ctree(Life.expectancy ~ ., data=train_data, control =
ctree_control(mincriterion=0.99, minsplit=0, minbucket=0))
plot(ptree)
predictions <- predict(ptree, newdata = test_data)

# Calculate RMSE by comparing predicted and actual labels
rmse <- sqrt(mean((predictions - actual_labels)^2))
print(paste("Root Mean Squared Error (RMSE):", rmse))

##########################################
# Random Forest

set.seed(30)
# Create a random index to split the data (70% for training, 30% for
testing)
train_index <- sample(1:nrow(final_df_no_country), 0.7 *
nrow(final_df_no_country))

oob.err=double(13)
test.err=double(13)
for(mtry in 1:13){

fit=randomForest(Life.expectancy~.,data=final_df_no_country,subset=train_ind
ex,mtry=mtry,ntree=500)
  oob.err[mtry]=fit$mse[100]
  pred=predict(fit,final_df_no_country[-train_index,])
  test.err[mtry]=with(final_df_no_country[-
train_index,],mean((Life.expectancy-pred)^2))
  cat(mtry," ")
}

matplot(1:mtry,cbind(test.err,oob.err),pch=19,col=c("red","blue"),type="b",y
lab="Mean Squared Error")
legend("topright",legend=c("Test","OOB"),pch=19,col=c("red","blue"))

#Random forest with the selected mtry
set.seed(30)
train_data <- final_df_no_country[train_index, ]
test_data <- final_df_no_country[-train_index, ]
rf.data = randomForest(Life.expectancy ~ ., data=train_data, ntree=500,
mtry=4, importance=TRUE)
rf.data
varImpPlot(rf.data)

predictions <- predict(rf.data,train_data[,-1])
actual_labels <- train_data$Life.expectancy
rmse <- sqrt(mean((predictions - actual_labels)^2))
print(paste("Train Root Mean Squared Error (RMSE):", rmse))

predictions <- predict(rf.data,test_data[,-1])
actual_labels <- test_data$Life.expectancy
rmse <- sqrt(mean((predictions - actual_labels)^2))
```

```
print(paste("Test Root Mean Squared Error (RMSE):", rmse))


# Random forest considering only the selected predictors
set.seed(30)
train_index <- sample(1:nrow(df_selected), 0.7 * nrow(df_selected))

oob.err=double(4)
test.err=double(4)
for(mtry in 1:4){

fit=randomForest(Life.expectancy~.,data=df_selected,subset=train_index,mtry=
mtry,ntree=100)
  oob.err[mtry]=fit$mse[100]
  pred=predict(fit,df_selected[-train_index,])
  test.err[mtry]=with(df_selected[-train_index,],mean((Life.expectancy-
pred)^2))
  cat(mtry," ")
}

matplot(1:mtry,cbind(test.err,oob.err),pch=19,col=c("red","blue"),type="b",y
lab="Mean Squared Error")
legend("topright",legend=c("Test","OOB"),pch=19,col=c("red","blue"))

#Random forest with the selected mtry
set.seed(30)
train_data <- df_selected[train_index, ]
test_data <- df_selected[-train_index, ]
rf.data = randomForest(Life.expectancy ~ ., data=train_data, ntree=100,
mtry=3, importance=TRUE)
rf.data
varImpPlot(rf.data)

predictions <- predict(rf.data,train_data[,-1])
actual_labels <- train_data$Life.expectancy
rmse <- sqrt(mean((predictions - actual_labels)^2))
print(paste("Train Root Mean Squared Error (RMSE):", rmse))

predictions <- predict(rf.data,test_data[,-1])
actual_labels <- test_data$Life.expectancy
rmse <- sqrt(mean((predictions - actual_labels)^2))
print(paste("Test Root Mean Squared Error (RMSE):", rmse))

##########################################
# Boosting

#Xboost with all the predictors
set.seed(8)
train_index <- sample(1:nrow(final_df_no_country), 0.7 *
nrow(final_df_no_country))
train_data <- final_df_no_country[train_index, ]
test_data <- final_df_no_country[-train_index, ]

dtrain <- xgb.DMatrix(data = as.matrix(train_data[,-1]), label =
train_data[,1])

params <- list(
```

```r
  objective = "reg:squarederror",
  eta = 0.01,
  max_depth = 4,
  nrounds = 1000
)

# Train the XGBoost model
xgb_model <- xgboost(data = dtrain, params = params, nrounds =
params$nrounds)

# Generate predictions on the test dataset for various numbers of trees
n.trees <- seq(from = 100, to = 1000, by = 100)
predmat <- matrix(NA, nrow = nrow(test_data), ncol = length(n.trees))

for (i in seq_along(n.trees)) {
  predmat[, i] <- predict(xgb_model, newdata = as.matrix(test_data[, -1]),
ntreelimit = n.trees[i])
}

berr <- apply((predmat - test_data$Life.expectancy)^2, 2, mean)
plot(n.trees, berr, pch = 19, ylab = "Mean Squared Error", xlab = "# Trees",
main = "Boosting Test Error")

# Find the lowest Root Mean Squared Error and the corresponding number of
trees
lowest_berr <- min(berr)
best_num_trees <- n.trees[which.min(berr)]
rmse <- sqrt(lowest_berr)
cat("Lowest Root Mean Squared Error:", rmse, "\n")
cat("Number of Trees for Lowest Error:", best_num_trees, "\n")

feature_importance <- xgb.importance(model = xgb_model)
xgb.plot.importance(importance_matrix = feature_importance)

# XGBoost considering only 4 predictors
set.seed(8)
train_index <- sample(1:nrow(df_selected), 0.7 * nrow(df_selected))
train_data <- df_selected[train_index, ]
test_data <- df_selected[-train_index, ]

dtrain <- xgb.DMatrix(data = as.matrix(train_data[,-1]), label =
train_data[,1])

params <- list(
  objective = "reg:squarederror",
  eta = 0.01,
  max_depth = 4,
  nrounds = 1000
)

# Train the XGBoost model
xgb_model <- xgboost(data = dtrain, params = params, nrounds =
params$nrounds)

# Generate predictions on the test dataset for various numbers of trees
n.trees <- seq(from = 100, to = 1000, by = 100)
predmat <- matrix(NA, nrow = nrow(test_data), ncol = length(n.trees))
```

```r
for (i in seq_along(n.trees)) {
  predmat[, i] <- predict(xgb_model, newdata = as.matrix(test_data[, -1]),
ntreelimit = n.trees[i])
}

berr <- apply((predmat - test_data$Life.expectancy)^2, 2, mean)
plot(n.trees, berr, pch = 19, ylab = "Mean Squared Error", xlab = "# Trees",
main = "Boosting Test Error")

# Find the lowest Root Mean Squared Error and the corresponding number of
trees
lowest_berr <- min(berr)
best_num_trees <- n.trees[which.min(berr)]
rmse <- sqrt(lowest_berr)
cat("Lowest Root Mean Squared Error:", rmse, "\n")
cat("Number of Trees for Lowest Error:", best_num_trees, "\n")

feature_importance <- xgb.importance(model = xgb_model)
xgb.plot.importance(importance_matrix = feature_importance)



# UNSUPERVISED LEARNING

#Clusters

#K-Means
#Data Normalization
final_df_scaled <- as.data.frame(scale(final_df_no_country))
final_df_scaled <- final_df_scaled %>% mutate(final_df$Country)

# Calculate the mean and standard deviation of the predictors from the
original data frame
original_mean <- sapply(final_df_no_country, mean)
original_sd <- sapply(final_df_no_country, sd)

# Set the row names for a dataframe
new_row_names <- final_df$Country
rownames(final_df_scaled) <- new_row_names

# Data Normalization for the dataframe with only the selected most important
predictors
df_selected_scaled <- as.data.frame(scale(df_selected))

# Set the row names for a dataframe
new_row_names <- final_df$Country
rownames(df_selected_scaled) <- new_row_names



#K-Means with all the features
wssplot <- function(data, nc=15, seed=230){
  wss <- (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares")}

wssplot(final_df_scaled[,-c(1, ncol(final_df_scaled))], nc=6)

set.seed(230)
k.means.fit <- kmeans(final_df_scaled[,-c(1, ncol(final_df_scaled))],
centers=2)
table(k.means.fit$cluster, final_df$Life.expectancy)

clusplot(final_df_scaled[,-c(1, ncol(final_df_scaled))],
k.means.fit$cluster,
         main='2D representation of the Cluster solution',
         color=TRUE, shade=TRUE,labels = 2, lines = 0)

cluster_assignments <- k.means.fit$cluster
final_df_scaled$cluster <- cluster_assignments

# List of variables/columns in the dataset (excluding the 'cluster' and
'final_df$Country' columns) to undo the scaling for each column
variable_names <- names(final_df_scaled)[!names(final_df_scaled) %in%
c("cluster","final_df$Country")]

for (col_name in variable_names) {
  scaled_col <- final_df_scaled[[col_name]]
  unscaled_col <- scaled_col * original_sd[col_name] +
original_mean[col_name]
  final_df_scaled[[col_name]] <- unscaled_col
}

cluster_summaries <- list()

# Loop through each variable and calculate mean, min, and max by cluster
for (variable_name in variable_names) {
  variable_summary <- final_df_scaled %>%
    group_by(cluster) %>%
    summarize(
      mean_variable = mean(.data[[variable_name]]),
      min_variable = min(.data[[variable_name]]),
      max_variable = max(.data[[variable_name]])
    )

  # Append the summary to the list
  cluster_summaries[[variable_name]] <- variable_summary
}

# Print or inspect the results
for (variable_name in variable_names) {
  cat("Summary for", variable_name, "\n")
  print(kable(cluster_summaries[[variable_name]], format = "markdown"))
  cat("\n\n")
}


#K-Means with just the selected features
wssplot <- function(data, nc=15, seed=1234){
  wss <- (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares")}

wssplot(df_selected_scaled[, -1], nc=6)

set.seed(230)
k.means.fit <- kmeans(df_selected_scaled[, -1], 2)
table(k.means.fit$cluster, df_selected_scaled$Life.expectancy)

clusplot(df_selected_scaled[, -1], k.means.fit$cluster,
         main='2D representation of the Cluster solution',
         color=TRUE, shade=TRUE,
         labels=2, lines=0)

cluster_assignments <- k.means.fit$cluster
df_selected_scaled$cluster <- cluster_assignments

# List of variables/columns in the dataset (excluding the 'cluster' column)
to undo the scaling for each column
variable_names <- names(df_selected_scaled)[!names(df_selected_scaled) %in%
c("cluster","final_df$Country")]

#original_mean <- sapply(final_df %>% select_if(is.numeric), mean)  #
Replace with actual mean values
#original_sd <- sapply(final_df%>% select_if(is.numeric), sd)  # Replace
with actual standard deviation values

for (col_name in variable_names) {
  scaled_col <- df_selected_scaled[[col_name]]
  unscaled_col <- scaled_col * original_sd[col_name] +
original_mean[col_name]
  df_selected_scaled[[col_name]] <- unscaled_col
}

cluster_summaries <- list()

# Loop through each variable and calculate mean, min, and max by cluster
for (variable_name in variable_names) {
  variable_summary <- df_selected_scaled %>%
    group_by(cluster) %>%
    summarize(
      mean_variable = mean(.data[[variable_name]]),
      min_variable = min(.data[[variable_name]]),
      max_variable = max(.data[[variable_name]])
    )

  # Append the summary to the list
  cluster_summaries[[variable_name]] <- variable_summary
}

# Print or inspect the results
for (variable_name in variable_names) {
  cat("Summary for", variable_name, "\n")
  print(kable(cluster_summaries[[variable_name]], format = "markdown"))
  cat("\n\n")
}


########################################
# Hierarchical Clustering

#Data Normalization
final_df_scaled <- as.data.frame(scale(final_df_no_country))
final_df_scaled <- final_df_scaled %>% mutate(final_df$Country)

# Set the row names for a dataframe
new_row_names <- final_df$Country
rownames(final_df_scaled) <- new_row_names

# Data Normalization for the dataframe with only the selected most important
predictors
df_selected_scaled <- as.data.frame(scale(df_selected))

# Set the row names for a dataframe
new_row_names <- final_df$Country
rownames(df_selected_scaled) <- new_row_names

#Hierarchical Clustering with all the features
set.seed(230)
d <- dist(final_df_scaled[, -c(1, ncol(final_df_scaled))], method =
"euclidean")
for (k in 2:10) {
  hc <- hclust(d,method="ward.D")
  cluster_assignments <- cutree(hc, k)
  silhouette_scores <- silhouette(cluster_assignments, d, FUN=mean)
  silhouette_avg <- mean(silhouette_scores[,"sil_width"])
  print(paste("Number of clusters:", k, " - Silhouette Score:",
silhouette_avg))
}

H.fit <- hclust(d, method="ward.D")
plot(H.fit)
abline(h=15, col="red")
groups <- cutree(H.fit, k=2) # cut tree into 2 clusters

# Draw dendogram with red borders around the 2 clusters
rect.hclust(H.fit, k=2, border="red")

boxplot(final_df_scaled$Life.expectancy ~ groups)
boxplot(final_df_scaled$Schooling ~ groups)
boxplot(final_df_scaled$Life.Ladder ~ groups)
boxplot(final_df_scaled$Sugar.Indicator ~ groups)
boxplot(final_df_scaled$Perceptions.of.corruption ~ groups)
boxplot(final_df_scaled$Log.GDP.per.capita ~ groups)
boxplot(final_df_scaled$Alcohol.Indicator ~ groups)
boxplot(final_df_scaled$Tabacco.Indicator ~ groups)
boxplot(final_df_scaled$Pollution.death.rate ~ groups)
boxplot(final_df_scaled$UHC.Indicator ~ groups)
boxplot(final_df_scaled$Vegetables.Indicator ~ groups)
boxplot(final_df_scaled$Fruits.Indicator ~ groups)
boxplot(final_df_scaled$Meat.Indicator ~ groups)
boxplot(final_df_scaled$Fish.Indicator ~ groups)

#Hierarchical Clustering with the selected features
set.seed(230)
d <- dist(df_selected_scaled[, -1], method = "euclidean")
for (k in 2:10) {
  hc <- hclust(d, method = "ward.D")
  cluster_assignments <- cutree(hc, k)
  silhouette_scores <- silhouette(cluster_assignments, d, FUN = mean)
  silhouette_avg <- mean(silhouette_scores[, "sil_width"])
  print(paste("Number of clusters:", k, " - Silhouette Score:", 
silhouette_avg))
}

H.fit <- hclust(d, method = "ward.D")
plot(H.fit)
abline(h = 15, col = "red")
groups <- cutree(H.fit, k = 10) # cut tree into 10 clusters
rect.hclust(H.fit, k = 10, border = "red")

boxplot(df_selected_scaled$Life.expectancy ~ groups)
boxplot(df_selected_scaled$Meat.Indicator ~ groups)
boxplot(df_selected_scaled$Log.GDP.per.capita ~ groups)
boxplot(df_selected_scaled$Alcohol.Indicator ~ groups)
boxplot(df_selected_scaled$Tabacco.Indicator ~ groups)
```

Bibliography

FAO. (s.d.). Retrieved from https://www.fao.org/faostat/en/#data/FBS

Kaggle. (s.d.). Retrieved from https://www.kaggle.com/datasets/amirhosseinmirzaie/countries-life-expectancy

Kaggle. (s.d.). Retrieved from https://www.kaggle.com/datasets/utkarshxy/who-worldhealth-statistics-2020-complete?resource=download

Kaggle. (s.d.). Tratto da https://www.kaggle.com/code/erglbozkurt/world-happiness-expanatory-data-analysis/input?select=world-happiness-report.csv

Our World in Data. (s.d.). Tratto da https://ourworldindata.org/air-pollution