



**MSc Computer Science**

**School of Computing, Engineering and Digital technologies**

**Machine Learning**

**ICA Element 2: Life Expectancy Rate Prediction Report**

**Name: Yaswanth Sai Chinthakayala**

**Student ID: W9640628**

## Table of Contents

1. Abstract.....	3
2. Introduction: .....	4
2.1 Dataset Description.....	4
3. Exploratory Data Analysis.....	4
3.1 Examining the data.....	4
3.2 Data pre-processing and Cleaning .....	7
3.3 Feature Scaling .....	11
3.4 Feature Engineering .....	12
3.5 Bi-Variate Analysis.....	12
4. Feature Selection .....	15
5. Experiments .....	17
5.1 Multi Linear Regression.....	17
5.2 KNN Regressor .....	17
5.3 Random Forest Regressor .....	18
5.4 Hyper Parameter Tuning .....	19
6. Results .....	20
7. Conclusion and Future work .....	21
8. References:.....	22

## 1. Abstract

The aim of this project is to predict the life expectancy rate of a country based on various features using a dataset that contains data from 2000 to 2015. Machine learning is used to solve this problem with the help of the dataset containing twenty-two features. Algorithms like K Nearest Neighbours, Multi Linear Regression and Random Forest regressor are used with hyper parameter tuning for obtaining best results. The results from the algorithms are good, all the algorithms predicted with 80 percent accuracy and the random forest regressor has the highest accuracy with 95 percent. These results can be used by their countries and researchers to analyse and adapt better lifestyle for higher life expectancy rate.

## 2. Introduction:

Life expectancy rate is defined as the average lifetime of a person. The life expectancy rate trend is increasing nowadays with the recent technological and medical advancements. But the average lifetime varies from country to country based on various factors. In this project, data visualization is used to draw conclusions from the dataset and predict the future life expectancy rate with machine learning.

Machine learning is an emerging technology that is being able to solve many difficult problems to humans. Using this technology, many researchers gain insights of the factors which affecting this life expectancy and help countries to overcome it. The dataset has various features related to diseases, schooling, income resources and the country status. These factors play an important role in predicting life expectancy.

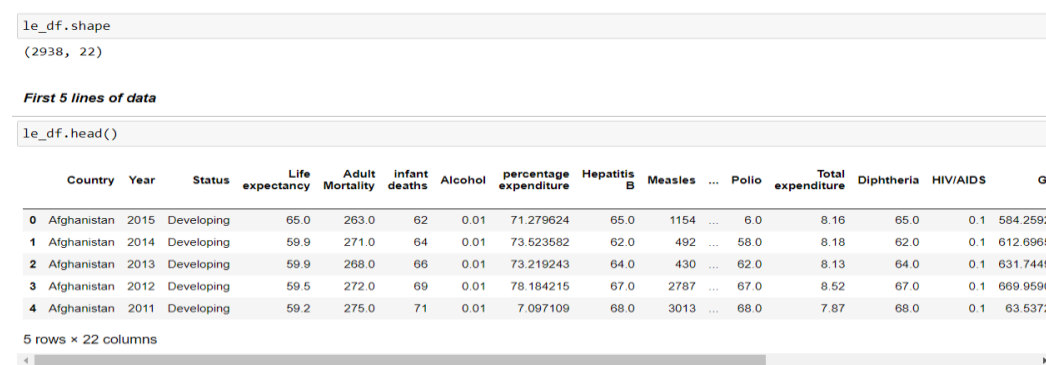
### 2.1 Dataset Description

This [Life expectancy dataset](#) is from Kaggle. This dataset contains 3000 rows and 22 columns. It has various features that include demographics, diseases, education and more columns for 193 countries from 2000 to 2015 range. The more information on this dataset found in the webpage.

## 3. Exploratory Data Analysis

### 3.1 Examining the data.

The first part of exploratory data analysis is to load the data file and checking the number of rows and columns and examining the first few lines of data.



```
le_df.shape
(2938, 22)
```

First 5 lines of data

```
le_df.head()
```

	Country	Year	Status	Life expectancy	Adult Mortality	Infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GC
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16	65.0	0.1	584.2592
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18	62.0	0.1	612.6965
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	0.1	631.7449
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	8.52	67.0	0.1	669.9590
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	7.87	68.0	0.1	63.5372

5 rows x 22 columns

Figure 1: Shape and head of the data

As shown in Figure 1, The number of rows for our dataset are 2938 and the columns are 22. The requirement for a dataset is at least 1000 rows and 5 columns should be present. This life expectancy dataset satisfies the requirement. The columns of the dataset are shown below:

```
le_df.columns
Index(['Country', 'Year', 'Status', 'Life expectancy ', 'Adult Mortality',
      'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
      'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',
      'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population',
      ' thinness 1-19 years', ' thinness 5-9 years',
      'Income composition of resources', 'Schooling'],
      dtype='object')
```

Figure 2: Columns of the dataset

The first thing that needs to be done is formatting the feature names. Most of the column names are not appropriate, they have leading spaces and naming is not correct. The names can be formatted by removing leading spaces.

*Renaming the columns (converting column names to snake case and removing leading spaces)*

```
: le_df = le_df.rename(columns = {'Country': 'country', 'Year': 'year', 'Status': 'status', 'Life expectancy ': 'life_expectancy', ' ',
      'infant deaths': 'infant_deaths', 'percentage expenditure': 'percentage_expenditure', 'Hepatitis B'
      'Measles ': 'measles', ' BMI ': 'BMI', 'under-five deaths ': 'under_five_deaths', 'Total expenditure'
      'Diphtheria ': 'diphtheria', 'Population': 'population', ' HIV/AIDS': 'HIV/AIDS', ' thinness 1-19 ye.
      ' thinness 5-9 years': 'thinness_5-9_years', 'Income composition of resources': 'income_composition_
```

Figure 3: Changing the column names.

To get more information on the dataset, the mean, variance and null values can also be evaluated.

```
le_df.info() #to see the data type and null values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
#   Column                                Non-Null Count  Dtype
---  ---                                -
0   country                              2938 non-null   object
1   year                                2938 non-null   int64
2   status                              2938 non-null   object
3   life_expectancy                     2928 non-null   float64
4   adult_mortality                     2928 non-null   float64
5   infant_deaths                       2938 non-null   int64
6   Alcohol                             2744 non-null   float64
7   percentage_expenditure              2938 non-null   float64
8   hepatitis_b                         2385 non-null   float64
9   measles                             2938 non-null   int64
10  BMI                                 2904 non-null   float64
11  under_five_deaths                   2938 non-null   int64
12  Polio                              2919 non-null   float64
13  total_expenditure                   2712 non-null   float64
14  diphtheria                          2919 non-null   float64
15  HIV/AIDS                           2938 non-null   float64
16  GDP                                 2490 non-null   float64
17  population                          2286 non-null   float64
18  thinness_1-19_years                 2904 non-null   float64
19  thinness_5-9_years                 2904 non-null   float64
20  income_composition_of_resources     2771 non-null   float64
21  Schooling                           2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

Figure 4: Information of the features.

From the figure 4, Most of the features data type is float and integer. Only country and status are objects. The status column contains 'developed' and 'developing' strings.

The describe dataset, gives you how each feature is distributed, what their max and min values and how the standard deviation is.

```
le_df.describe() #description of data
```

	year	life_expectancy	adult_mortality	infant_deaths	Alcohol	percentage_expenditure	hepatitis_b	measles	BMI	under_f
count	2938.000000	2928.000000	2928.000000	2938.000000	2744.000000	2938.000000	2385.000000	2938.000000	2904.000000	2938.000000
mean	2007.518720	69.224932	164.796448	30.303948	4.602861	738.251295	80.940461	2419.592240	38.321247	19.300000
std	4.613841	9.523867	124.292079	117.926501	4.052413	1987.914858	25.070016	11467.272489	20.044034	19.300000
min	2000.000000	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000	0.000000	1.000000	19.300000
25%	2004.000000	63.100000	74.000000	0.000000	0.877500	4.685343	77.000000	0.000000	19.300000	19.300000
50%	2008.000000	72.100000	144.000000	3.000000	3.755000	64.912906	92.000000	17.000000	43.500000	19.300000
75%	2012.000000	75.700000	228.000000	22.000000	7.702500	441.534144	97.000000	360.250000	56.200000	19.300000
max	2015.000000	89.000000	723.000000	1800.000000	17.870000	19479.911610	99.000000	212183.000000	87.300000	19.300000

Figure 5: Describing the data.

## 3.2 Data pre-processing and Cleaning

### Duplicates

One of the most important things to do in a dataset is data pre-processing. Initially, the duplicates are checked.

```
#returns the sum of duplicates  
le_df.duplicated().sum()
```

0

Figure 6: Duplicate check

As shown in figure 6, there are no duplicates in the dataset which is better. If there are any duplicates they can be dropped.

### Null Values

Null values are very difficult to deal with if they are more in number. If the dataset has fewer null values, they can be dropped. The percentage of null values for each column are shown in figure 7.

Let us calculate the null data percentage

```
(le_df.isnull().sum()/(len(le_df)))*100
```

country	0.000000
year	0.000000
status	0.000000
life_expectancy	0.340368
adult_mortality	0.340368
infant_deaths	0.000000
Alcohol	6.603131
percentage_expenditure	0.000000
hepatitis_b	18.822328
measles	0.000000
BMI	1.157250
under_five_deaths	0.000000
Polio	0.646698
total_expenditure	7.692308
diphtheria	0.646698
HIV/AIDS	0.000000
GDP	15.248468
population	22.191967
thinness_1-19_years	1.157250
thinness_5-9_years	1.157250
income_composition_of_resources	5.684139
Schooling	5.547992
dtype: float64	

Figure 7: Column wise Null percentage.

The null values are also checked using heatmap, which shows exactly where they are missing and if there is any huge missing section.

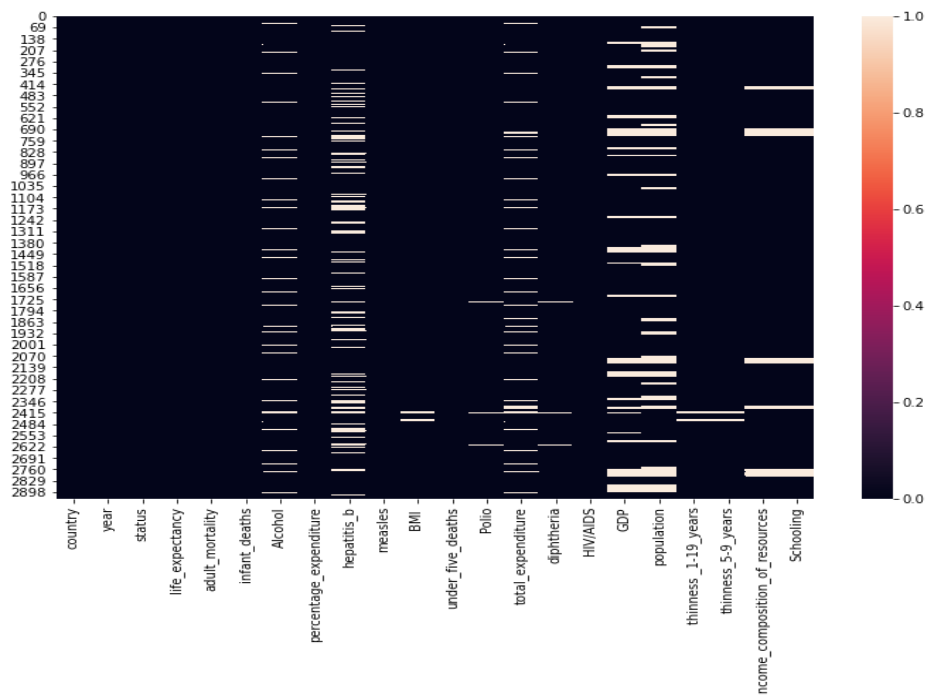


Figure 8: Null value heatmap

From the figure 7 and figure 8, a lot of null values are seen. The significant part of the data is missing in the columns of population, GDP and hepatitis\_b. The population has 22 percentage of missing data, before dropping them correlation matrix can be checked.

```
print("Correlation between Life expectancy and population: ",le_df["life_expectancy"].corr(le_df["population"]))
print("Correlation between Life expectancy and hepatitis_b: ",le_df["life_expectancy"].corr(le_df["hepatitis_b"]))
print("Correlation between Life expectancy and GDP: ",le_df["life_expectancy"].corr(le_df["GDP"]))
```

```
Correlation between Life expectancy and population: -0.021538108386786478
Correlation between Life expectancy and hepatitis_b: 0.2567619476049244
Correlation between Life expectancy and GDP: 0.46145519262073825
```

Figure 9: Correlation values for most missing data.

The population has the least correlation, and it doesn't play any impact in predicting Life expectancy. So, the population column can be dropped. GDP and hepatitis\_b cannot be dropped because they have more correlation with target variable.

The missing value percentage is too high it cannot be dropped simply more than 600 rows of data will be lost (columns can be dropped but by deleting rows lot of data will be lost). Imputation is the best option to implement.



## Imputation:

- Mean Imputation: The null values are replaced by mean when the distribution is approximately normal.
- Median Imputation: The missing values are replaced by median if the distribution is skewed.
- Mode Imputation: This imputation only applies when the missing values are categorical.

For imputing data, the distribution needed to be observed carefully. From figure 10, it is seen that some of the features are approximately normal. It can be checked for all the features.

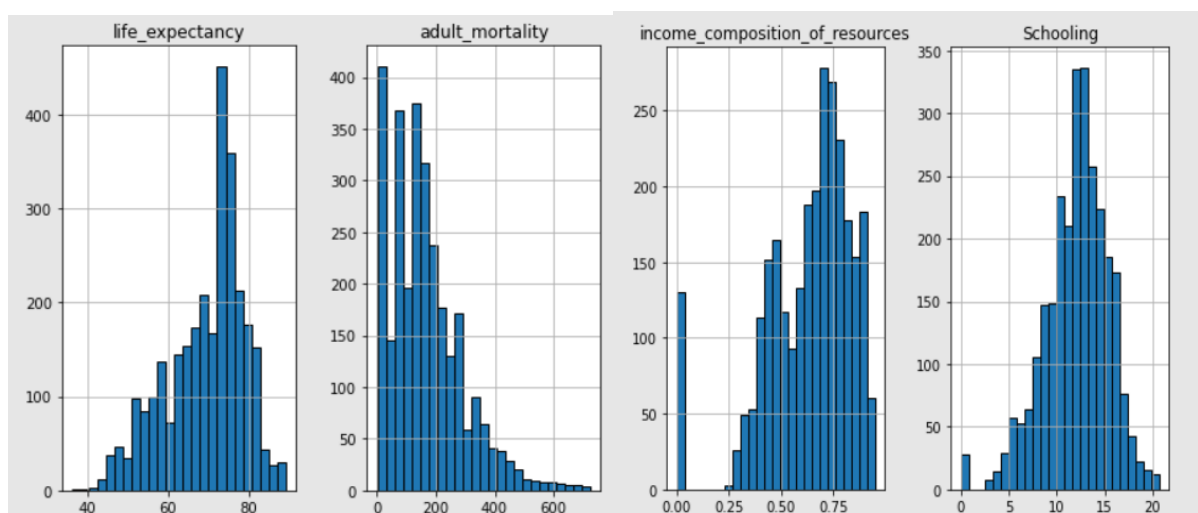


Figure 10: Feature distributions

The missing values can be replaced by the mean of each column.

***This line of code replaces the data***

```
: #null values are replaced by mean  
le_df = le_df.fillna(le_df.mean())
```

Figure 11: Mean Imputation

## Outliers

The data points which are significantly in different range from the original data are called outliers. There are many ways for outliers to be introduced to a dataset. Some of them are data entry errors, experimental errors and some are even intentional.

## Identifying outliers:

Some of the ways to identify outlier is z-score and interquartile range. Z-score works best for normal distribution. Interquartile range works for any values. This is shown:

```
Q1 = le_df.quantile(0.25)
Q3 = le_df.quantile(0.75)
IQR = Q3 - Q1

# Identify the outliers for each column
outliers = ((le_df < (Q1 - 1.5 * IQR)) | (le_df > (Q3 + 1.5 * IQR))).sum()

# Calculate the percentage of outliers for each column
outlier_percentage = outliers / len(le_df) * 100
```

Figure 12: Outliers calculation using IQR.

The box plots for some of the features are shown in figure 13. It can be clearly seen that there are many outliers. There are some ways to deal with outliers. They are dropping them or converting them.

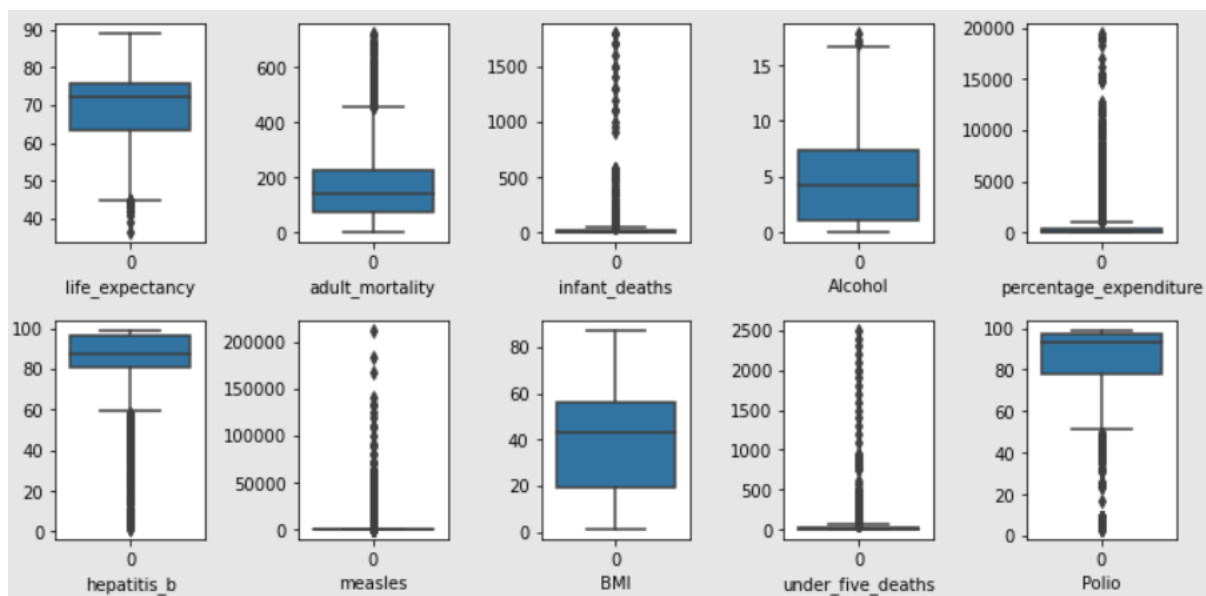


Figure 13: Box Plots

There are lot of outliers, without losing the data Winsorization can be used. It is the process of eliminating the extreme values, it will replace the outliers with the statistical data. After winsorization the outliers will be removed.

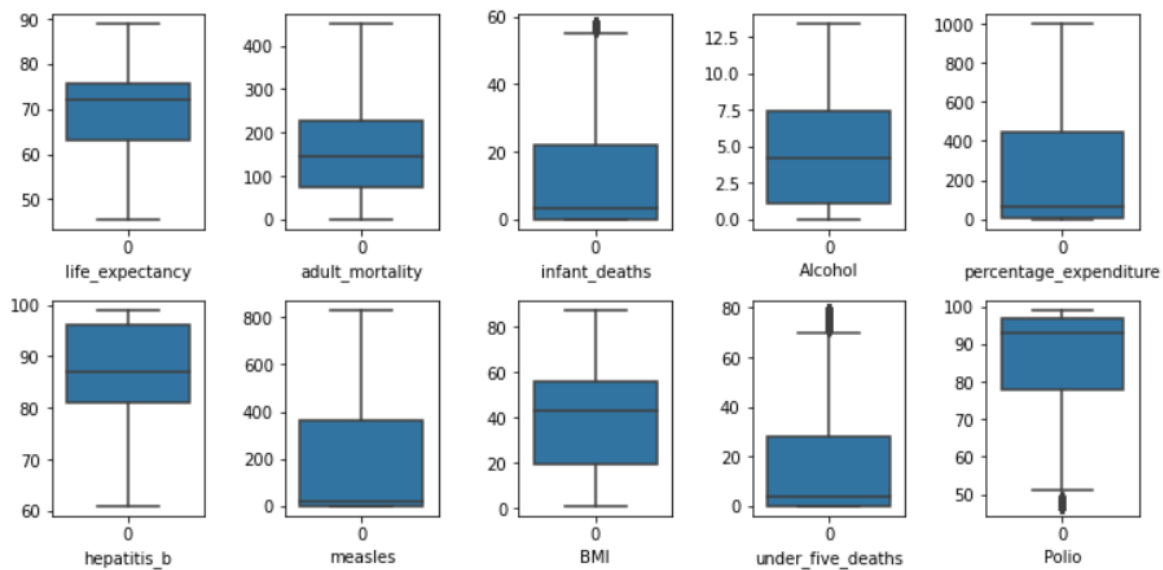


Figure 14: Box plot after Winsorization

### 3.3 Feature Scaling

Scaling is nothing but normalizing the features in the dataset. The real-world data will have different range of values for different features. Before feeding the data into a machine learning algorithm, it is the best practice to get all the features to same scale. There are different types of scalars like min-max scaler, standard scaler, and robust scaler.

In our dataset it is better to use the robust scalar which will deal efficiently even the data has outliers. In addition, some algorithms perform very efficiently with scaled data like linear regression and k nearest neighbours.

```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
scaler.fit(le_df[numeric_cols]) #fits the data
le_df[numeric_cols] = scaler.transform(le_df[numeric_cols]) #Transforms the data
```

Figure 15: Applying Robust Scaler

The dataset is now transformed and ready to use.

### 3.4 Feature Engineering

Feature Engineering is technique that is used to create new features with the existing feature. This helps in transforming the data and enhancing the model accuracy. In our dataset, status feature datatype is object, it can be converted to Boolean with 0 and 1 values. Now, this feature involves in deciding for target variable because of its data type.

```
#developing countries : 1  
#developed countries : 0  
le_df=pd.get_dummies(le_df,columns=["status"],drop_first=True)
```

Figure 16: Feature Engineering

### 3.5 Bi-Variate Analysis

**Life expectancy and Status\_developing.**

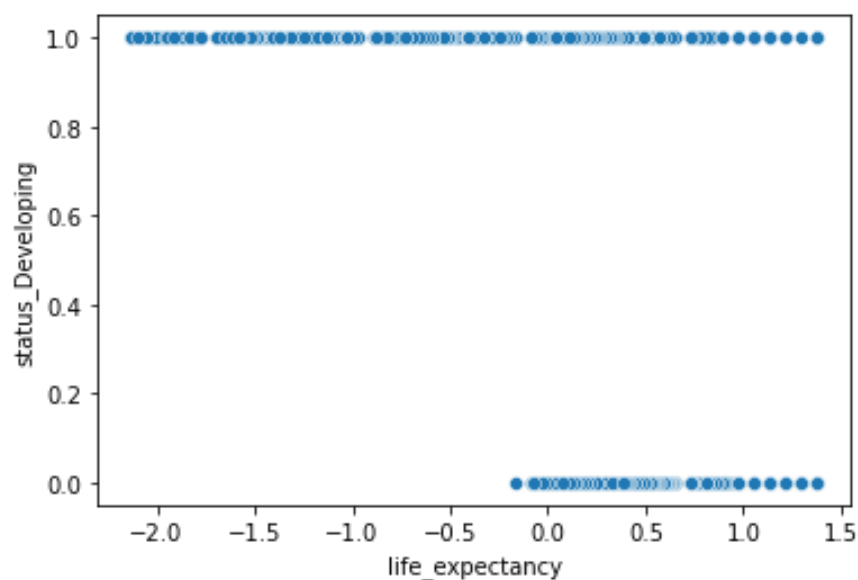


Figure 17: Relation between life expectancy and status developing.

From the Figure 17, It is clearly visible that the developed countries have more life expectancy than the developed countries.

## Life expectancy and Income Composition of resources

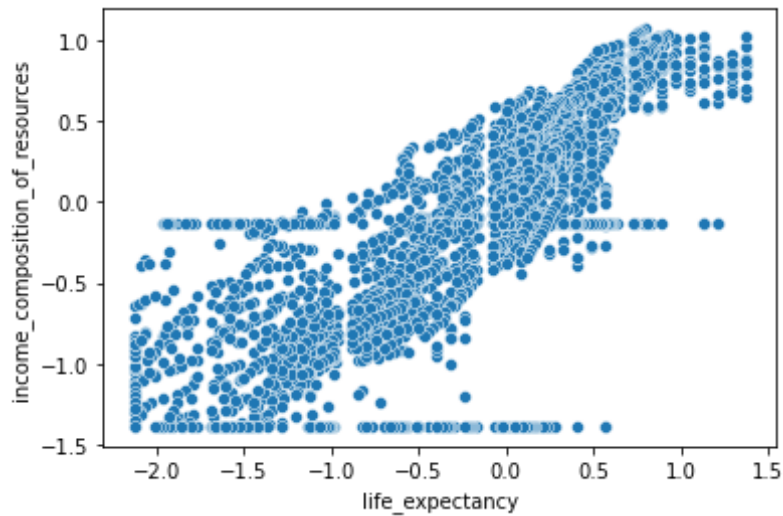


Figure 18: Life expectancy and Income

From the figure 18, Linear relationship can be seen for income and life expectancy rate. As the income increases the people live longer.

## Life expectancy and HIV/AIDS

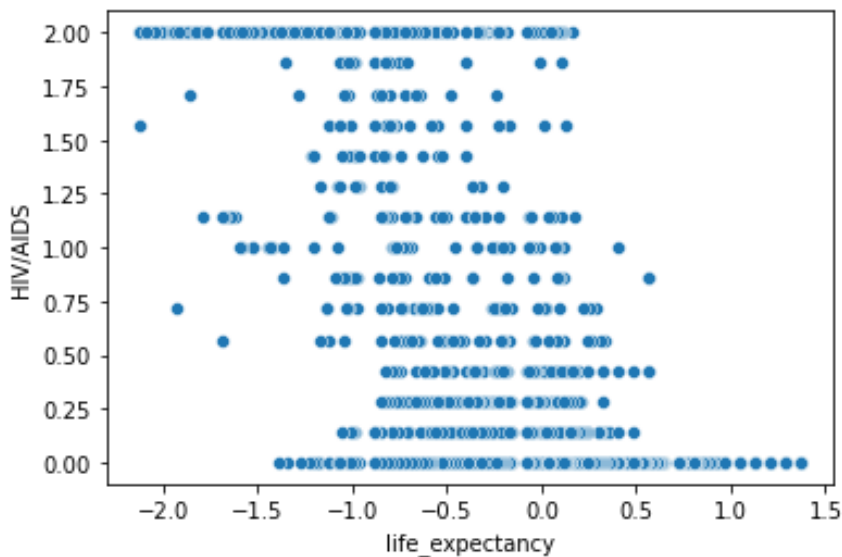


Figure 19: Life expectancy and HIV/AIDS

The trend from figure 19, is not as previous features but, it can be said that the people with no HIV or AIDS live longer than the people who are affected.

## Life expectancy and adult mortality

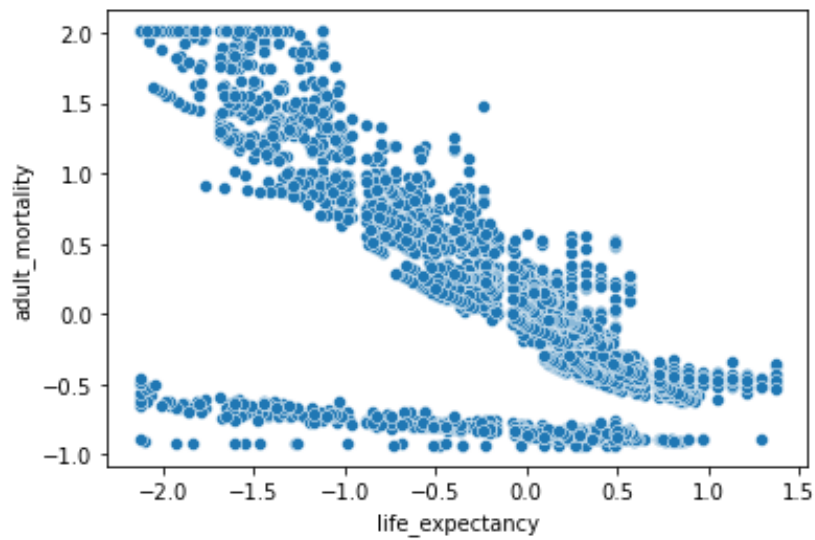


Figure 20: Life expectancy and adult mortality.

Both features negatively correlate to each other. As the life expectancy increases, the adult mortality decreases.

Similarly, all features can be compared to other features using pair plot. The part of pair plot is shown in the below figure.

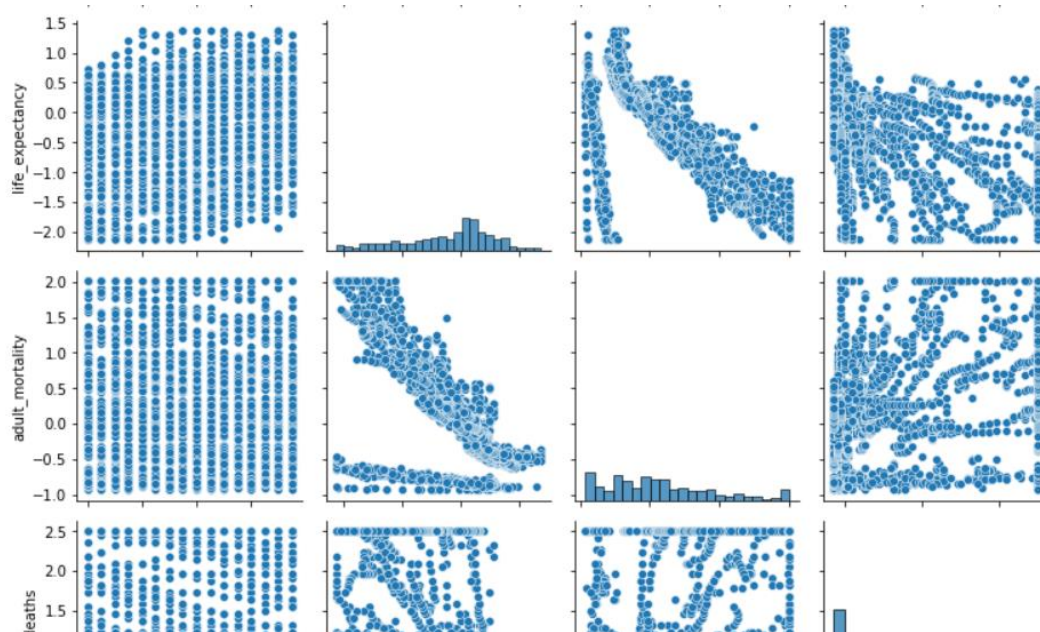


Figure 21: Part of Pair plot.

## 4. Feature Selection

The correlation matrix is the best way to find correlation of a feature with any other column. The correlation matrix for this dataset is shown in the figure 22. With the help of figure 22 some observations can be made. Features that have more than 60 percent of correlation with life expectancy is:

1. Adult mortality
2. under\_five\_deaths
3. HIV/AIDS
4. Income composition of resources
5. Schooling

All these features cannot be used directly. Some of them needs to be dropped and some of them needs to be considered.

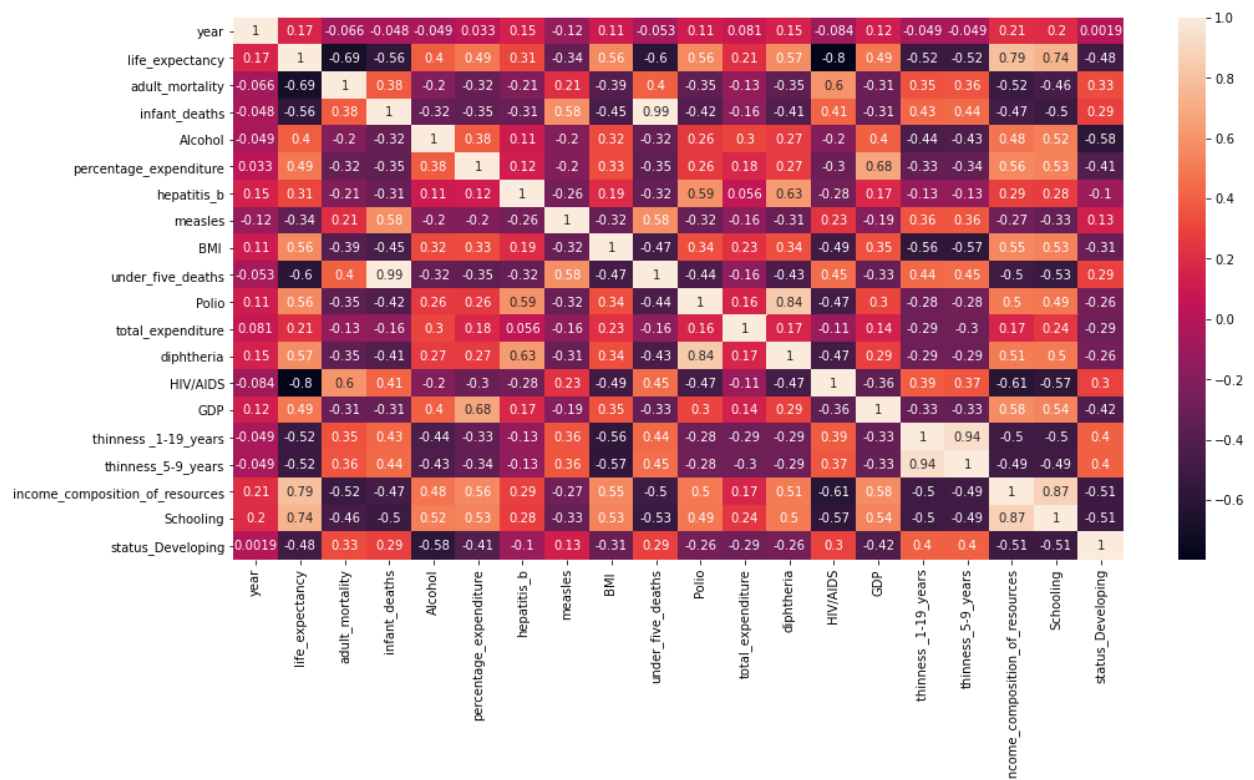


Figure 22: Correlation matrix

**Note:**

If two features have more correlation with each other. One of those features has correlation with target variable then, one of them should be dropped to avoid over fitting and complex problems. For example, Schooling and Income composition of resources have more correlation that is 0.87. Also, life expectancy has correlation of 0.74 and 0.79 for schooling and income respectively. So, the schooling should be dropped.

In the similar way some features can be dropped. Other feature selection methods also can be added to be more accurate.

**Feature Importance using P Value:**

p-value means probability value. This value is defined as how likely the event is occurred by chance. The p-value selected features are shown in the below figure.

```
: print("Top five features that plays important role in this is:\n",)
for i in pvalue_selected_features:
    print (i)
```

Top five features that plays important role in this is:

```
adult_mortality
under_five_deaths
diphtheria
HIV/AIDS
income_composition_of_resources
```

*Figure 23: P value selected features*

**Train Test Split:**

The above five features can be used for training and target variable is life expectancy. The whole data is divided into 70 percent train data and 30 percent test data.

**Splitting the data into training and testing**

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0)
```

*Figure 24: Splitting Data*



## 5. Experiments

### 5.1 Multi Linear Regression

The regression predicts values that are continuous in nature. If two features are linearly correlated then, linear regression can be used to predict the values. In multi linear regression, instead of one feature multiple features are used to predict the continuous value.

```
Model_LR = LinearRegression()  
Model_LR.fit(X_train,y_train)
```

```
LinearRegression()
```

```
#predicted values  
y_pred= Model_LR.predict(X_test)
```

```
print ("The Intercept is:", Model_LR.intercept_)  
print ("The Coefficients are:", Model_LR.coef_)
```

```
The Intercept is: 0.10601666219413705  
The Coefficients are: [-0.20948156 -0.12279811  0.08063233 -0.33891046  0.40547406]
```

Figure 25: Multi Linear Regression

The model is fit to training data, then the values are predicted. In the figure 25, the intercept and coefficients are displayed. The evaluation metrics for the same model is:

Evaluation Metrics for Multi Linear Regression:

```
R-Squared (R2) score is: 0.8318435746156746  
Mean Squared Error (MSE) is: 0.09633977342888247  
Mean Absolute Error (MAE) is 0.23810052070298857
```

Figure 26: Multi linear regression evaluation metrics

The R2 score is 0.83, which is very good. Next, the same data will be tested using other algorithms.

### 5.2 KNN Regressor

KNN means k nearest neighbours, mostly this nearest neighbours model is used for classification problems, in this data set, knn regressor model is used. This supervised model is distance-based algorithm. In this algorithm the distance between test points and train data points are calculated. Then mean or median is used to compute the predicted values.

```

model_knn = KNeighborsRegressor()
model_knn.fit(X_train,y_train) #fitting the model

KNeighborsRegressor()

y_pred = model_knn.predict(X_test)

```

Figure 27: KNN Model

The evaluation metrics for this knn regression model is:

Evaluation Metrics for KNN Regression:

R-Squared (R2) score is: 0.9183995869255586  
Mean Squared Error (MSE) is: 0.04675031173698879  
Mean Absolute Error (MAE) is 0.14403117584068714

Figure 28: KNN Evaluation metrics

The R2 score using KNN is 0.91, which is way more than multi linear regression. MSE and MAE also improved with respect to multi linear regression.

### 5.3 Random Forest Regressor

In random forest the data is split into many decision trees for training the model. Random forest regressor can work for both regression and classification problems like a decision tree. First, it starts with the root, then it will follow the flow of their splits until the leaf node is reached. At this point we obtain the result. This dataset is trained with the random forest regressor, and the results are obtained.

```

: model_RF = RandomForestRegressor()
: model_RF.fit(X_train,y_train)

: RandomForestRegressor()

: y_pred = model_RF.predict (X_test)

```

Figure 29: Random Forest Regressor

The evaluation metrics are:

### Evaluation Metrics for Random Forest Regression:

```
R-Squared (R2) score is: 0.9518111806645049
Mean Squared Error (MSE) is: 0.027608222082364128
Mean Absolute Error (MAE) is 0.10702149429322154
```

Figure 30: Random Forest Evaluation Metrics

The results using random forest regressor is 0.95, greater than other algorithms. MAE and MSE are also better than other algorithms.

## 5.4 Hyper Parameter Tuning

Grid search cross validation is used for tuning parameters. This grid search cv tests for all the parameter range that is in the parameter grid and then outputs the best parameters and best score.

### KNN Parameter Tuning

The parameter grid is defined and then grid search is called with 5-fold cross validation as shown in Figure 31.

```
param_grid = {'n_neighbors': [3, 5, 7, 9, 11, 13, 15, 17, 19],
              'weights': ['uniform', 'distance'],
              'p': [1, 2, 3],
              'metric': ['euclidean', 'manhattan', 'chebyshev']}

knn = KNeighborsRegressor()

#Perform grid search for knn model using parameter grid with 5 cross validation
grid_search = GridSearchCV(knn, param_grid, cv=5)

grid_search.fit(X_train,y_train) #Fitting the model.
```

Figure 31: KNN Parameter Tuning

```
print ("The best parameters after performing grid search is:" , grid_search.best_params_)
The best parameters after performing grid search is: {'metric': 'manhattan', 'n_neighbors': 5, 'p': 1, 'weights': 'distance'}

print ("The best score after performing grid search is:", grid_search.best_score_)
The best score after performing grid search is: 0.9438312196107799
```

Figure 32: KNN Best Params

The score after hyper parameter tuning is 0.94 which increased from normal applied parameters.

## Random Forest Parameter Tuning

```
grid_search = GridSearchCV(RandomForestRegressor(random_state=0),
                           {
                               'n_estimators':np.arange(5,100,5),
                               'max_features':np.arange(0.1,1.0,0.05),
                           },cv=5, scoring="r2",verbose=1,n_jobs=-1)
grid_search.fit(X_train,y_train)
```

Figure 33: Random Forest Grid Search

```
print("The best parameters found in grid search is:", grid_search.best_params_)
The best parameters found in grid search is: {'max_features': 0.6000000000000002, 'n_estimators': 90}

print("The best score with hyper parameter tuning is:",grid_search.best_score_)
The best score with hyper parameter tuning is: 0.9556893989853144
```

Figure 34: Random Forest Best Parameters

The best score after hyper parameter tuning random forest is 0.955 which increased from normal applied parameters.

## 6. Results

This dataset is trained in three models. The evaluation metrics for each model is shown in the Table 1.

Table 1: Results

Model	R2 Score	MSE	MAE	Hyper Parameter Tuning Score
Multi Linear Regression	0.831	0.096	0.238	None
KNN Regressor	0.918	0.046	0.144	0.943
Random Forest Regressor	0.951	0.027	0.107	0.955

The results are good for all the algorithms. The best algorithm that predicted with most score is Random Forest Regressor. The best score is 0.955 for random forest regressor using hyper parameter tuning.

## 7. Conclusion and Future work

To Conclude, life expectancy is an important indicator and is influenced by many factors such as schooling, diseases, adult mortality and more. From all the data analysis it is obviously seen that many people with higher income are living long life. Their income is high because of their education. Other conclusion to draw is people with diseases are not living longer. Understanding all these can help governments taking respective measures for a better world.

There is a lot of scope for the future work, it can include impact of food, accommodation, and climate. Other areas like people's religion, beliefs effect on life expectancy can also be studied. By collecting more data, there are lot of opportunities for future work.

## 8. References:

“Kaggle: Life Expectancy (WHO)” [online] Available at: <https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who> [Accessed 1 May 2023].

Kumar, A. (2020). *Python - Replace Missing Values with Mean, Median & Mode*. [online] Data Analytics. Available at: <https://vitalflux.com/pandas-impute-missing-values-mean-median-mode/> [Accessed 1 May 2023].

Suresh, A. (2020). *How to Remove Outliers for Machine Learning* [online] Analytics Vidhya. Available at: <https://medium.com/analytics-vidhya/how-to-remove-outliers-for-machine-learning-24620c4657e8> [Accessed 1 May 2023].

Chong, J. (2021). *What is Feature Scaling & Why is it Important in Machine Learning?* [online] Medium. Available at: <https://towardsdatascience.com/what-is-feature-scaling-why-is-it-important-in-machine-learning-2854ae877048> [Accessed 1 May 2023].

Patel, H. (2021). *What is Feature Engineering — Importance, Tools and Techniques for Machine Learning*. [online] Medium. Available at: <https://towardsdatascience.com/what-is-feature-engineering-importance-tools-and-techniques-for-machine-learning-2080b0269f10> [Accessed 1 May 2023].

Simplilearn.com. (2022). *What is Multiple Linear Regression in Machine Learning? / Simplilearn*. [online] Available at: <https://www.simplilearn.com/what-is-multiple-linear-regression-in-machine-learning-article> [Accessed 1 May 2023].

Teixeira-Pinto, A. (n.d.). *2 K-nearest Neighbours Regression / Machine Learning for Biostatistics*. [online] [bookdown.org](https://bookdown.org/tpinto). Available at: <https://bookdown.org/tpinto/home/Regression-and-Classification/k-nearest-neighbours-regression.html>

Beheshti, N. (2022). *Random Forest Regression*. [online] Medium. Available at: <https://towardsdatascience.com/random-forest-regression-5f605132d19d>