NAME : KAUSTUBH JOSHI

COLLEGE: MAHARAJA SURAJMAL INSTITUTE OF DELHI , IP UNIVERSITY , DELHI

INTERNSHIP TITLE : PREDICTING LIFE EXPECTANCY USING MACHINE LEARNING – SB48574

PROJECT ID : SPS_PRO_215

PROJECT TITLE : PREDICTING LIFE EXPECTANCY USING MACHINE LEARNING

SBID : SB20200048574

EMAIL : kaustubh19j@gmail.com

GITHUB : https://github.com/SmartPracticeschool/llSPS-INT-1678-Predicting-Life-Expectancy-using-Machine-Learning

Abstract

In this project, we were asked to use the real world data set by WHO to predict the life expectancy of a person.

We were made to work with the real life datasets along with the machine learning libraries.

We were expected to gain experience using a common data-mining and machine learning libraries like pandas and NumPy and were expected to submit a report about the dataset and the algorithms used.

After performing the required tasks on a dataset of the life expectancy of various countries and years , herein lies my final report.

Keywords: Machine Learning, Regression Models, Classification, Prediction, Artificial Intelligence.

Introduction: Machine learning is a sub-domain of computer science which evolved from the study of pattern recognition in data, and also from the computational learning theory in artificial intelligence. It is the first-class ticket to most interesting careers in data analytics today

As data sources proliferate along with the computing power to process them, going straight to the data is one of the most straightforward ways to quickly gain insights and make predictions. Machine Learning can be thought of as the study of a list of sub-problems, viz: decision making, clustering, classification, forecasting, deep-learning, inductive logic programming, support vector machines, reinforcement learning, similarity and metric learning, genetic algorithms, sparse dictionary learning, etc. Supervised learning, or classification is the machine learning task of inferring a function from a labelled data

In Supervised learning, we have a training set, and a test set. The training and test set consists of a set of examples consisting of input and output vectors, and the goal of the supervised learning algorithm is to infer a function that maps the input vector to the output vector with minimal error. In an optimal scenario, a model trained on a set of examples will classify an unseen example in a correct fashion, which requires the model to generalize from the training set in a reasonable way.

In this project, we try and find patterns in a dataset , which is a sample of males in a heart-disease high risk region of South Africa, and attempt to throw various intelligently-picked algorithms at the data, and see what sticks. Problems and Issues in Supervised learning: Before we get started, we must know about how to pick a good machine learning algorithm for the given dataset. To intelligently pick an algorithm to use for a supervised learning task, we must consider the following factors

1. Heterogeneity of Data: Many algorithms like neural networks and support vector machines like their feature vectors to be homogeneous numeric and normalized. The algorithms that employ distance metrics are very sensitive to this, and hence if the data is heterogeneous, these methods should be the afterthought. Decision Trees can handle heterogeneous data very easily.

2. Redundancy of Data: If the data contains redundant information, i.e. contain highly correlated values, then it’s useless to use distance based

methods because of numerical instability. In this case, some sort of Regularization can be employed to the data to prevent this situation.

3. Dependent Features: If there is some dependence between the feature vectors, then algorithms that monitor complex interactions like Neural Networks and Decision Trees fare better than other algorithms. MACHINE LEARNING PROJECT 5

4. Bias-Variance Trade-off: A learning algorithm is biased for a particular input x if, when trained on each of these data sets, it is systematically incorrect when predicting the correct output for x, whereas a learning algorithm has high variance for a particular input x if it predicts different output values when trained on different training sets.

5. Curse of Dimensionality: If the problem has an input space that has a large number of dimensions, and the problem only depends on a subspace of the input space with small dimensions, the machine learning algorithm can be confused by the huge number of dimensions and hence the variance of the algorithm can be high.

6. Overfitting: The programmer should know that there is a possibility that the output values may constitute of an inherent noise which is the result of human or sensor errors. In this MACHINE LEARNING PROJECT 6 case, the algorithm must not attempt to infer the function that exactly matches all the data. Being too careful in fitting the data can cause overfitting, after which the model will answer perfectly for all training examples but will have a very high error for unseen samples. A practical way of preventing this is stopping the learning process prematurely, as well as applying filters to the data in the pre-learning phase to remove noises. Only after considering all these factors can we pick a supervised learning algorithm that works for the dataset we are working on.

Dataset: The dataset used is a life expectancy dataset released by the World Health Organisation.

The data set has the following features :

The data is saved as a csv file as LifeExpectancy.csv and it is read and stored in the life data variable. The Year column is dropped as it will not be used in the analysis. Below the first 5 rows are shown. The data contains 21 columns and 2938 rows with the header row. The table contains data about:

- Countries
- Status
- Life Expectancy
- Adult Mortality
- 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

PREPROCESSING AND CLEANING THE DATA SETS.

- Before the data can be imported using the machine learning libraries and can be trained, the data needs to be cleaned and pre-processed.

- All the null values in the data set need to be either set to 0 , deleted or set equal to the mean value.
- In the cleaning process, I have set the null values as 0 for the ease of calculation and maintaining the accuracy of the model.

IMPORTING THE REQUIRED LIBRARIES

```
In [4]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
```

LOADING THE PACKAGES

The following packages have been imported NumPy, Pandas, Matplotlib, SciPy, Seaborn. Sklearn is the most widely used package for the machine learning process. The fallowing subpackages have been used:

1. train_test_split
2. linear_model
3. model selection
4. metrics
5. tree
6. ensemble
7. pre-processing

IMPORTING THE DATASET

The required dataset in the csv file is imported as the panda data frame.

```
life.head()
```

	year	status	life	adultmortality	infantdeaths	alcohol	percentageexpenditure	hepatitisb	measles	bmi	...	polio	tota
0	2015	0	65.0	263	62	0.01	71.279624	65	1154	19.1	...	6	8.16
1	2014	0	59.9	271	64	0.01	73.523582	62	492	18.6	...	58	8.18
2	2013	0	59.9	268	66	0.01	73.219243	64	430	18.1	...	62	8.13
3	2012	0	59.5	272	69	0.01	78.184215	67	2787	17.6	...	67	8.52
4	2011	0	59.2	275	71	0.01	7.097109	68	3013	17.2	...	68	7.87

5 rows × 21 columns

DISPLAYING THE STRUCTURE OF THE DATA SET

```
In [7]: life.info()
```

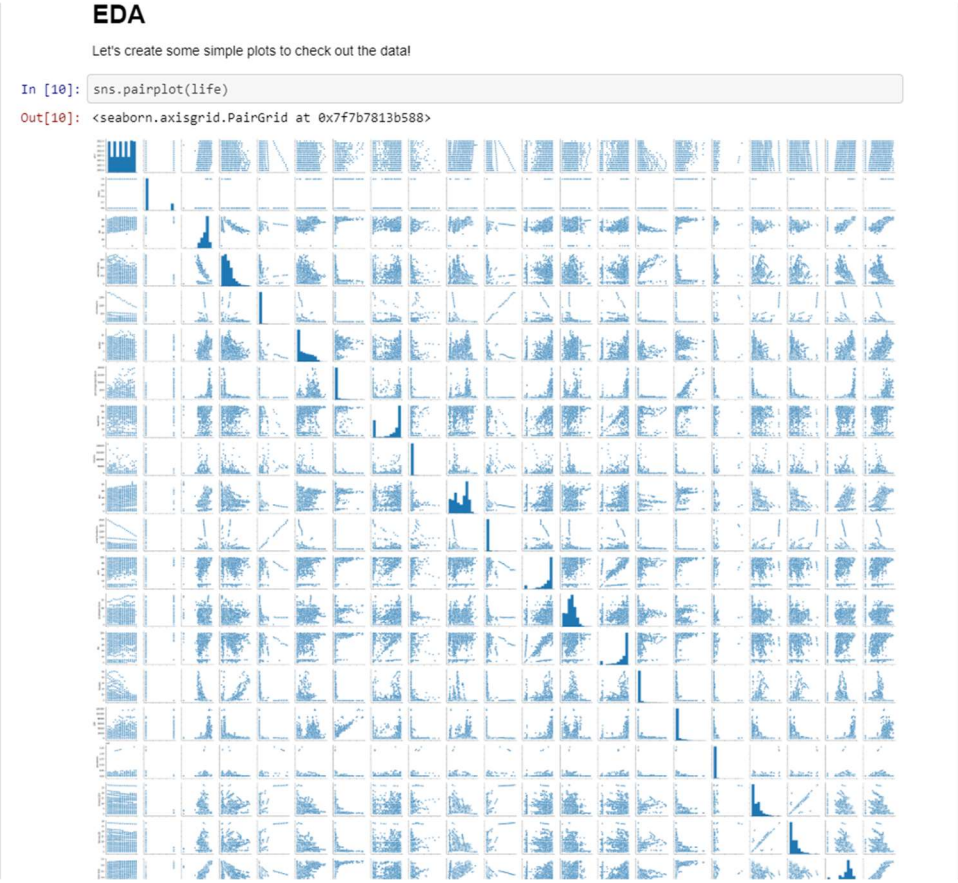
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 21 columns):
year                        2938 non-null int64
status                      2938 non-null int64
life                        2938 non-null float64
adultmortality              2938 non-null int64
infantdeaths                2938 non-null int64
alcohol                     2938 non-null float64
percentageexpenditure       2938 non-null float64
hepatitisb                  2938 non-null int64
measles                     2938 non-null int64
bmi                         2938 non-null float64
underfivedeaths             2938 non-null int64
polio                       2938 non-null int64
totalexpenditure            2938 non-null float64
dip                         2938 non-null int64
hiv/aids                    2938 non-null float64
gdp                         2938 non-null float64
population                  2938 non-null float64
thinness119                 2938 non-null float64
thinness59                  2938 non-null float64
incomecomp                  2938 non-null float64
schooling                   2938 non-null float64
dtypes: float64(12), int64(9)
memory usage: 482.1 KB
```

DISPLAYING THE COLUMNS OF THE DATA SET

```
In [9]: life.columns
```

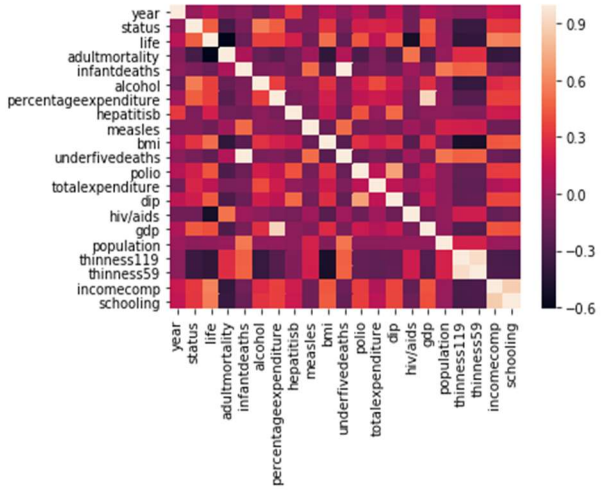
```
Out[9]: Index(['year', 'status', 'life', 'adultmortality', 'infantdeaths', 'alcohol',
               'percentageexpenditure', 'hepatitisb', 'measles', 'bmi',
               'underfivedeaths', 'polio', 'totalexpenditure', 'dip', 'hiv/aids',
               'gdp', 'population', 'thinness119', 'thinness59', 'incomecomp',
               'schooling'],
              dtype='object')
```

EDA



Now we will plot the correlation matrix visulizing it with a heatmap. The legend tells that the warmer colors show higher and positive correlation, while the colder low or negative.
There is a very high correlation between thinness of 5-9 year-old and that of 1-19 year-old. Also between population and infant deaths, under 5 deaths, another is between schooling and income composition of resources. On the other hand Life expectancy and Adult Mortality are very highly negatively correlated.

```
In [12]: sns.heatmap(life.corr())
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7b67908ef0>
```



TRAINING THE REGRESSION MODEL

Training a Linear Regression Model

Let's now begin to train out regression model! We will need to first split up our data into an X array that contains the features to train on, and a y array with the target variable, in this case the Price column. We will toss out the Address column because it only has text info that the linear regression model can't use.

X and y arrays

```
In [13]: X = life[['year', 'status','adultmortality', 'infantdeaths', 'alcohol','percentageexpenditure', 'hepatitis
         b', 'measles', 'bmi','underfivedeaths', 'polio', 'totalexpenditure', 'dip', 'hiv/aids','gdp', 'populatio
         n', 'thinness119', 'thinness59', 'incomecomp','schooling']]
         y = life['life']
```

Train Test Split

Now let's split the data into a training set and a testing set. We will train out model on the training set and then use the test set to evaluate the model.

```
In [14]: from sklearn.model_selection import train_test_split

In [15]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

Creating and Training the Model

```
In [16]: from sklearn.linear_model import LinearRegression

In [17]: lm = LinearRegression()

In [18]: lm.fit(X_train,y_train)
Out[18]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                  normalize=False)
```

Model Evaluation

Let's evaluate the model by checking out it's coefficients and how we can interpret them.

```
In [19]: # print the intercept
         print(lm.intercept_)
         54.32620483709512
```

PREDICTIONS FROM OUR MODEL

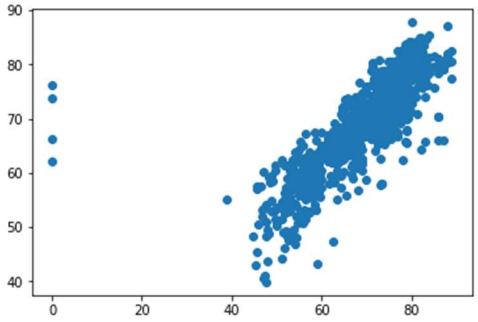
Predictions from our Model

Let's grab predictions off our test set and see how well it did!

In [21]:
```
predictions = lm.predict(X_test)
```

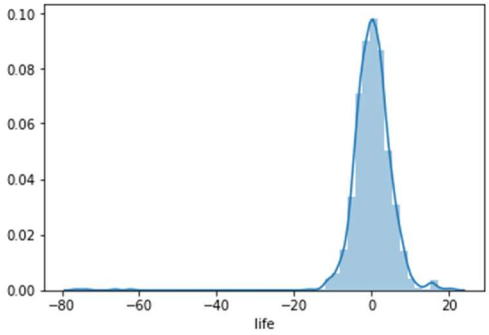
In [22]:
```
plt.scatter(y_test,predictions)
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x7f7b60110550>
```



Residual Histogram

In [23]:
```
sns.distplot((y_test-predictions),bins=50);
```



LINEAR REGRESSION WITH POLYNOMIAL FUNCTIONS

Regression Evaluation Metrics

Here are three common evaluation metrics for regression problems:

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:
$$\frac 1n\sum_{i=1}^n(y_i-\hat{y}_i)^2$$

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Comparing these metrics:

- **MAE** is the easiest to understand, because it's the average error.
- **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world.
- **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them.

In [24]:
```
from sklearn import metrics
```

In [26]:
```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 3.5394253196842085
MSE: 35.63069322049429
RMSE: 5.969145099634812
```

MEAN ABSOLUTE ERROR : 3.53942531968

ROOT MEAN SQUARE ERROR : 5.9691450996

After applying the regression algorithm:

1. MAE of 1.83
2. RMSE of 6.05

UI USING THE NODE RED

To integrate the ML model with the UI, we would be using the Node Red functionality provided by the IBM Watson Studio.

To design the UI, we need to import the flow of the UI.

Once, we have setup the flow, we need to integrate the ML model with it.

To integrate the ML Model with it we need to access the endpoint URL of our ML Model.

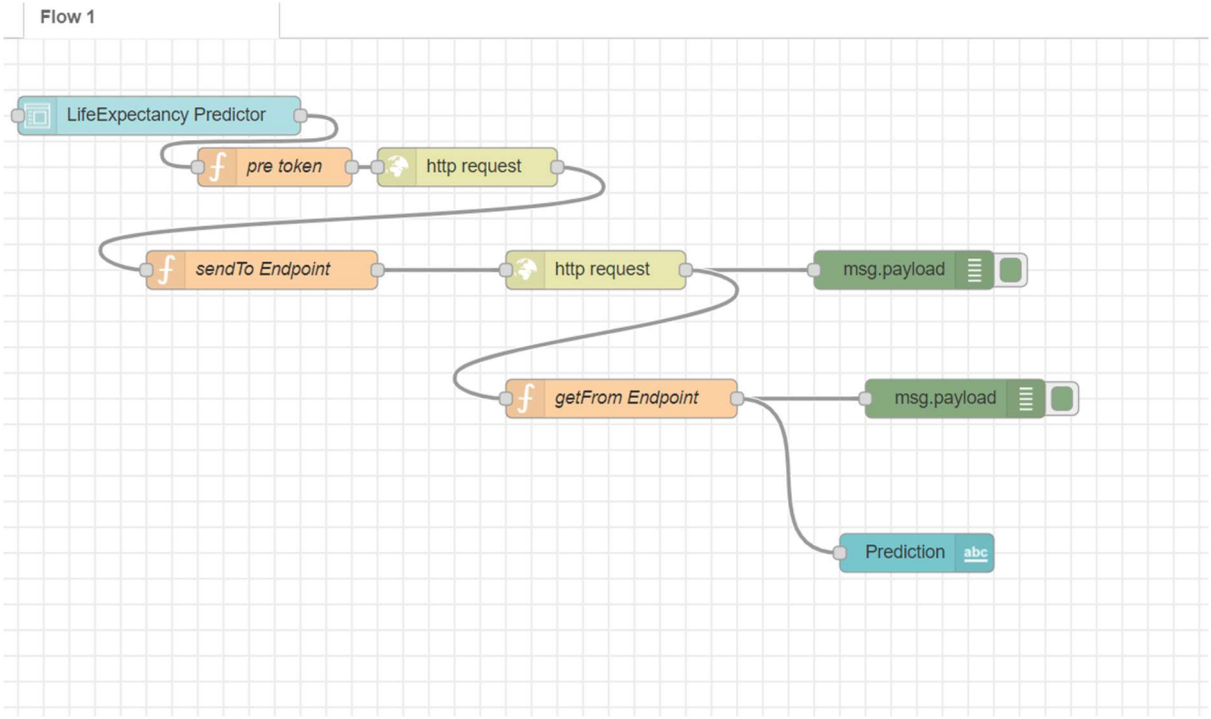
Components of the flow are :

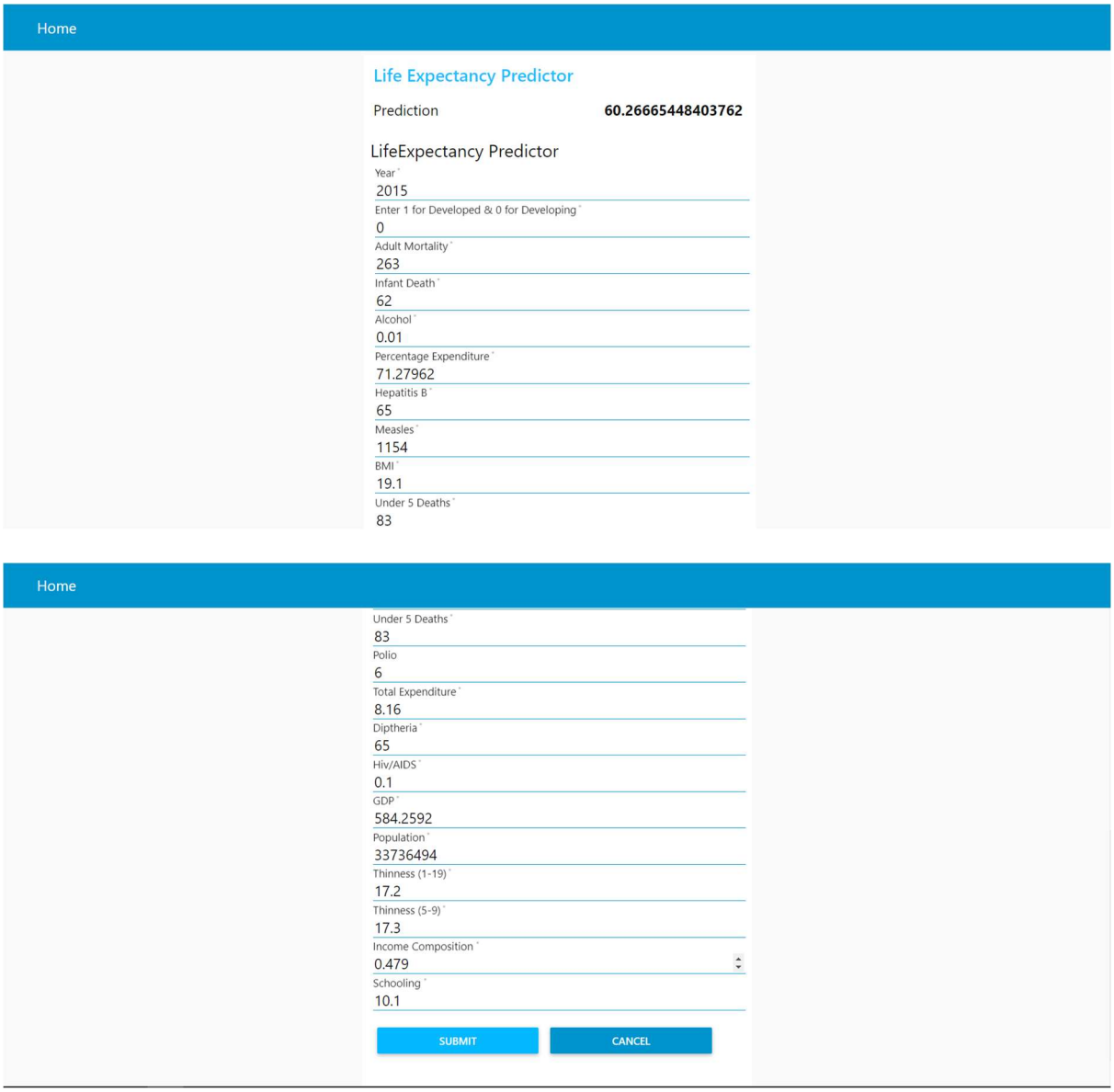
Form : The form contains all the elements of the UI. All the labels are associated with a variable.

http requests : To setup the flow, we need two http requests. The first http request requires a token to connect to the machine learning service of the Watson studio .

The second http request helps us in integrating the model using the endpoint URL.

Once the flow has been setup, we deploy the model.





For Afghanistan : 2015

Actual Value : 65

Predicted value : 60.2666

Error Percentage : (65-60.2666)/65 = 7.28%

CONCLUSION

- Error Percentage : 7.28%
- MAE of 1.83
- RMSE of 6.05

SOURCES

1. https://www.kaggle.com/kumarajarshi/life-expectancy-who/data
2. SmartInternz Webinars and Labs
3. Stack Overflow
4. Slack