REPORT

Life expectancy predicting model

Dixita Shukla

1. **INTRODUCTION**
   1. OVERVIEW

*Life expectancy is one of the most important factors in end-of-life decision making.*

*The project tries to create a model based on data provided by the World Health Organization (WHO) to evaluate the life expectancy for different countries in years. The data offers a timeframe from 2000 to 2015. The data originates from here: https://www.kaggle.com/kumarajarshi/life-expectancy-who/data The output algorithms have been used to test if they can maintain their accuracy in predicting the life expectancy for data they haven’t been trained.*

* 1. PURPOSE

*The purpose is to predict Life Expectancy by looking at the positive and negatively correlated factors to improve the Life Quality.*

*It serves as an example for countries to assess to improve life expectancy for their citizens.*

*When you are deciding when to start receiving retirement benefits, one important factor to take into consideration is how long you might live.*

1. **LITERATURE SURVEY**
   1. Existing Problem

*As we all know, Life expectancy is one of the most important factors in end-of-life decision making.*

*So, using the certain factors like Schooling, GDP, Adult Mortality Rate, Child Date, etc. life expectancy is predicted. All the factors are negatively or positively correlated.*

*When you are deciding when to start receiving retirement benefits, one important factor to take into consideration is how long you might live.*

* *A man turning age 65 on April 1, 2020 can expect to live, on average, until age 84.0.*
* *A woman turning age 65 on April 1, 2020 can expect to live, on average, until age 86.5.*
  1. Proposed Solution

*Using this model, life expectancy of a person can be predicted by taking some input features from the user.*

*Life Expectancy depends on the following features-*

* *Country*
* *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*

*By taking the information regarding the above factors, model will predict the life expectancy.*

*Want to know your life expectancy? You can use our simple Life Expectancy Predicting Model to get a rough estimate of how long you (or your spouse) may live. Knowing this information can help you make a more informed choice regarding when to collect Social Security retirement benefits.*

1. **Theoretical Analysis**
   1. Block Diagram

Country

**ML Model**

Adult Mortality Rate

Alcohol

Percentage Expenditure

Life Expectancy

Status

Income Composition of Resources

Schooling

Thinness 5-9 Years

Thinness 1-19 Years

Population

GDP

HIV/AIDS

Diphtheria

Total Expenditure

Polio

Under-5-Death

BMI

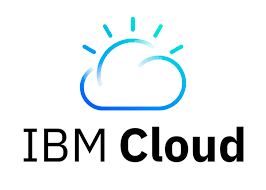
Measles

Hepatitis-B

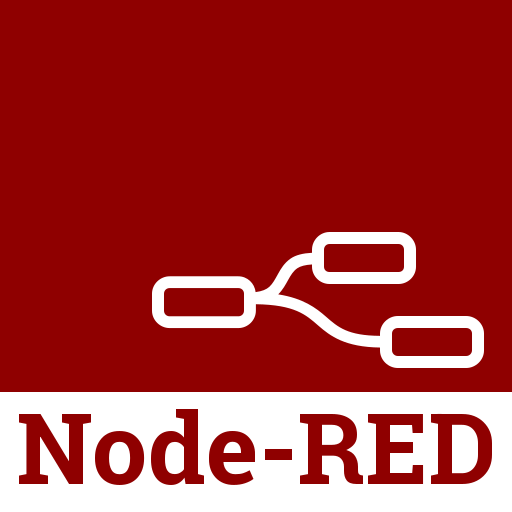
* 1. Software Designing

***IBM cloud*** *computing is a set of cloud computing services for business offered by the information technology company IBM.*

*It provides many services like Node-Red, Watson Studio, etc for storing and processing data.*



***Node Red*** *is used for creating the User Interface (UI) application. Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions.*

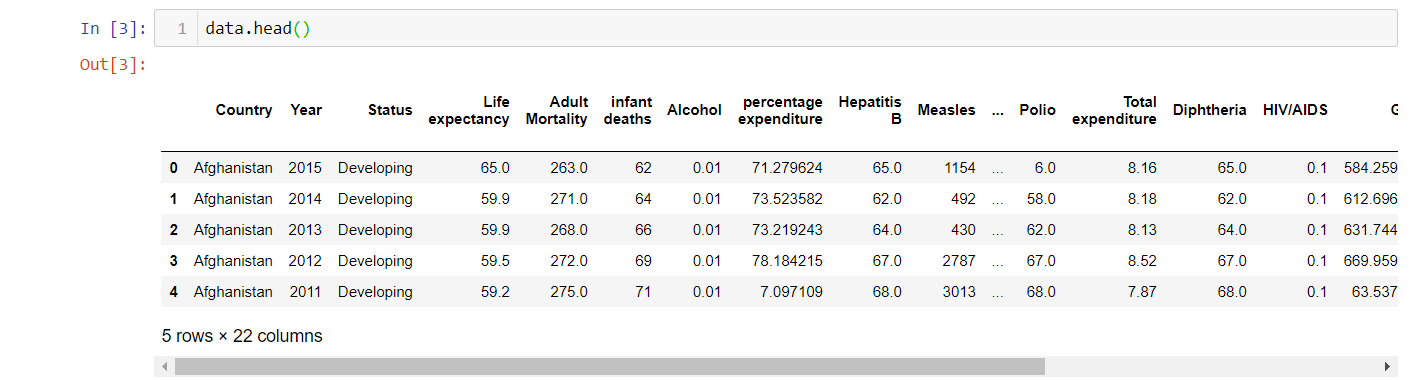


***Watson Studio*** *helps data scientists and analysts prepare data and build models at scale across any cloud. With its open, flexible multicloud architecture,***Watson Studio***provides capabilities that empower businesses to simplify enterprise data science and AI: Automate AI lifecycle management with AutoAI.*

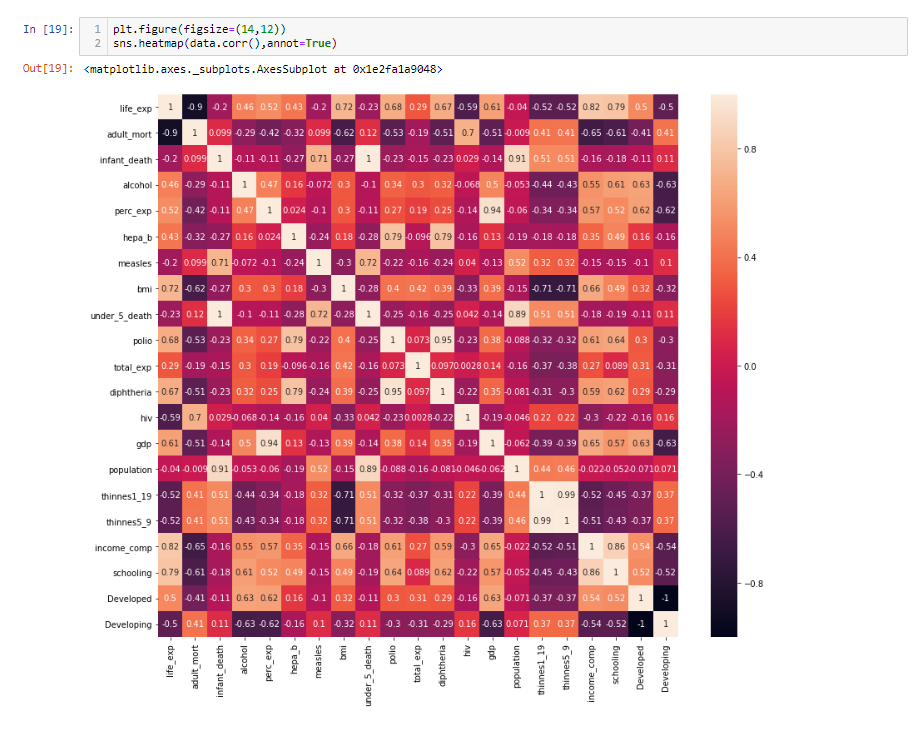
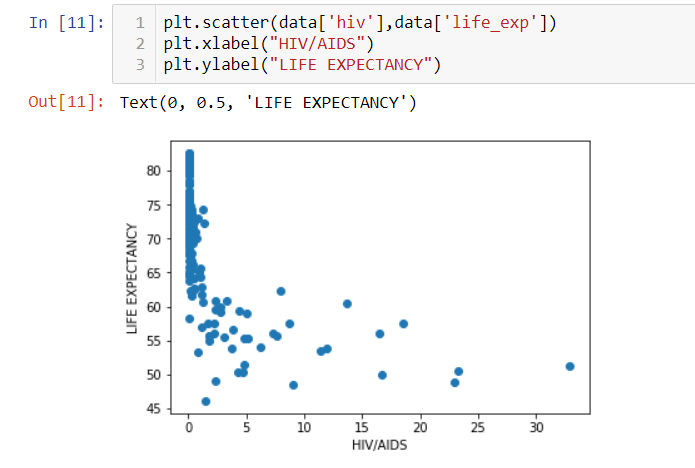
**

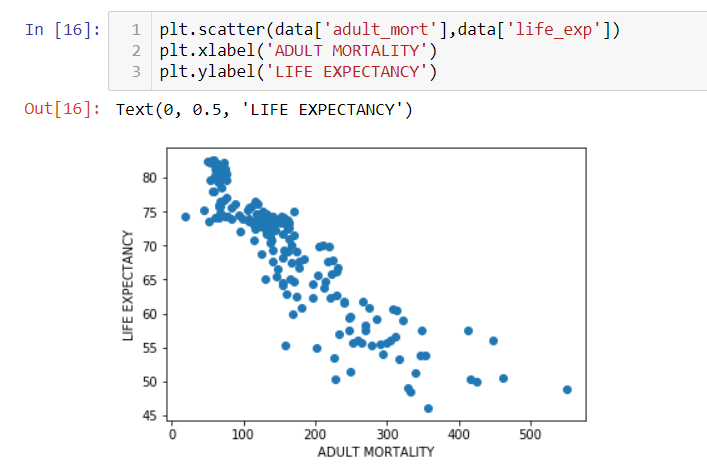
1. **Experimental Investigation**

*Data was collected from “* <https://www.kaggle.com/kumarajarshi/life-expectancy-who/data>” *and then pre-processed so that it is understood by the Machine Learning Algorithms Properly.*

**

*Exploratory Data Analysis was done, in order to visualise the dataset, and to check the correlation of different parameters on the ‘Life Expectancy’.*

**

**

*Then, different Regression Algorithms were applied and then accuracy is checked for each, so as to find the best fitted algorithm.*

*Fine Tuning was done, in order to find the best parameters so that we get the best possible accuracy.*

*The output algorithms have been used to test if they can maintain their accuracy in predicting the life expectancy for data they haven’t been trained. Four algorithms have been used:*

* *Linear Regression*
* *Ridge Regression*
* *Lasso Regression*
* *ElasticNet Regression*
* *Linear Regression with Polynomic Features*
* *Decision Tree Regression*
* *Random Forest Regression*

1. **Flowchart**

*Choose an adequate model and tune it to get the best performance possible*

*Develop a benchmark model*

*What is regularization and when is appropriate to use it*

*An overview of how a model learns*

*Split the data correctly*

*Differentiate between over and underfitting, defining what they are and explaining the best ways to avoid them*

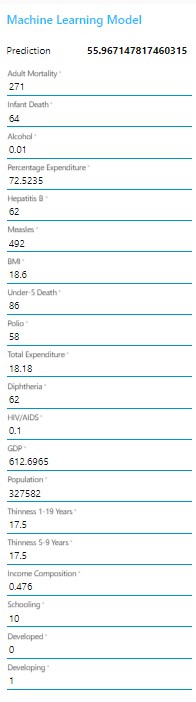
*Prepare the data (dealing with missing values, with categorial values…)*

*Set an evaluation protocol and the different protocols available*

*Gather data*

*Define adequately our problem*

1. **RESULT**

****

*By taking the inputs from the user of the certain parameters Life Expectancy has been predicted successfully.*

1. **Advantages and Disadvantages**

Advantages

* Life Expectancy can be predicted depending on certain parameters with great accuracy.
* Parameters which are increasing and decreasing the Life Expectancy can be known.
* *Knowing this information can help you make a more informed choice regarding when to collect Social Security retirement benefits.*

Disadvantages

* *Though, the accuracy of the model is very high. Still there is some chance that the does not give the exact Life Expectancy.*
* *It may create some tension when people got to know their wrong Life Expectancy age.*

1. **Application**

* *You can use our simple Life Expectancy Predicting Model to get a rough estimate of how long you (or your spouse) may live. Knowing this information can help you make a more informed choice regarding when to collect Social Security retirement benefits.*
* *When you are deciding when to start receiving retirement benefits, one important factor to take into consideration is how long you might live.*
* *Government can improve certain features on which the Life Expectancy depends, so the average life expectancy can be increased.*
* *Policy makers can benefit their customers by suggesting them the appropriate policies for them.*

1. **Conclusion**

*After comparing all the algorithms we can conclude the Lasso and the Elastic Net Regression offer which are the same:*

1. *Best Parameters: {‘alpha’: 0, ‘max\_iter’: 10}*
2. *R square on the test data of 92%*
3. *MAE of 1.83*
4. *MSE of 6.05*
5. **Future Scope**

* *Look at class within a particular country and see if these same factors are same in determining life expectancy for an individual*
* *Use twitter API to incorporate NLP analysis for a country to see how it relates to Life Expectancy.*
* *Increase the dataset size with continuing UN and Global Data to incorporate new added features like population, GDP, etc in order to test and clarify country groupings.*

1. **Bibliography**

* <https://www.kaggle.com/kumarajarshi/life-expectancy-who/data>
* *Introduction to Machine Learning with Python by Andreas C. Müller & Sarah Guido.*
* *SmartInternz Webinars*
* *Mentors*

**APPENDIX**

Source Code

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from nose.tools import \*

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import PolynomialFeatures

from sklearn.model\_selection import KFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import Ridge

from sklearn.linear\_model import Lasso

from sklearn.linear\_model import ElasticNet

from sklearn.metrics import make\_scorer

from scipy import stats

import seaborn as sns

who.to\_csv()

who.columns=['country','year','status','life\_exp','adult\_mort','infant\_death','alcohol','perc\_exp','hepa\_b','measles','bmi','under\_5\_death','polio','total\_exp','diphtheria','hiv','gdp','population','thinnes1\_19','thinnes5\_9','income\_comp','schooling']

who

who=who.drop('year',axis=1)

status=pd.get\_dummies(who.status)

who=pd.concat([who,status],axis=1)

who=who.drop(['status'],axis=1)

who.rename(columns={'Developing':0,'Developed':1})

who=who.groupby('country').mean()

who.head()

who.columns

plt.scatter(who['hiv'],who['life\_exp'])

plt.xlabel("HIV/AIDS")

plt.ylabel("LIFE EXPECTANCY")

plt.scatter(who['gdp'],who['life\_exp'])

plt.xlabel('GDP')

plt.ylabel('LIFE EXPECTANCY')

plt.scatter(who['bmi'],who['life\_exp'])

plt.xlabel('BMI')

plt.ylabel('LIFE EXPECTANCY')

plt.scatter(who['alcohol'],who['life\_exp'])

plt.xlabel('ALCOHOL')

plt.ylabel('LIFE EXPECTANCY')

plt.scatter(who['adult\_mort'],who['life\_exp'])

plt.xlabel('ADULT MORTALITY')

plt.ylabel('LIFE EXPECTANCY')

plt.scatter(who['schooling'],who['life\_exp'])

plt.xlabel('SCHOOLING')

plt.ylabel('LIFE EXPECTANCY')

plt.scatter(who['perc\_exp'],who['life\_exp'])

plt.xlabel('PERCENTAGE EXPENDITURE')

plt.ylabel('LIFE EXPECTANCY')

plt.figure(figsize=(14,12))

sns.heatmap(who.corr(),annot=True)

X=who.drop('life\_exp',axis=1)

y=who['life\_exp']

X.isnull().sum()

y.isnull().sum()

X.fillna(value=X.mean(),inplace=True)

y.fillna(value=y.mean(),inplace=True)

X.isnull().sum()

y.isnull().sum()

stats.describe(X[1:])

sc=MinMaxScaler()

X=sc.fit\_transform(X)

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,train\_size=0.7,test\_size=0.3)

lin\_reg=LinearRegression()

lin\_reg.fit(X\_train,y\_train)

print('R\_square score on the training data: ',lin\_reg.score(X\_train,y\_train))

print("Coefficients: ",lin\_reg.coef\_)

print("Mean Squared Error: ",mean\_squared\_error(y\_test,lin\_reg\_pred))

print("Absolute Squared Error: ",mean\_absolute\_error(y\_test,lin\_reg\_pred))

print("R\_square Score: ",r2\_score(y\_test,lin\_reg\_pred))

scoring=make\_scorer(r2\_score)

grid\_cv=GridSearchCV(Ridge(),param\_grid={'alpha':range(0,10),'max\_iter':[10,100,1000]},scoring=scoring,cv=5,refit=True)

grid\_cv.fit(X\_train,y\_train)

print("Best Parameters: "+str(grid\_cv.best\_params\_))

result=grid\_cv.cv\_results\_

print('R\_square score on the training data: ',grid\_cv.score(X\_train,y\_train))

print("R\_square Score: ",r2\_score(y\_test,grid\_cv.best\_estimator\_.predict(X\_test)))

print("Mean Squared Error: ",mean\_squared\_error(y\_test,lin\_reg\_pred))

print("Absolute Squared Error: ",mean\_absolute\_error(y\_test,lin\_reg\_pred))

scoring=make\_scorer(r2\_score)

grid\_cv1=GridSearchCV(Lasso(),param\_grid={'alpha':range(0,10),'max\_iter':[10,100,1000]},scoring=scoring,cv=5,refit=True)

grid\_cv1.fit(X\_train,y\_train)

print("Best Parameters: "+str(grid\_cv1.best\_params\_))

result=grid\_cv1.cv\_results\_

print('R\_square score on the training data: ',grid\_cv1.score(X\_train,y\_train))

print("R\_square Score: ",r2\_score(y\_test,grid\_cv1.best\_estimator\_.predict(X\_test)))

print("Mean Squared Error: ",mean\_squared\_error(y\_test,lin\_reg\_pred))

print("Absolute Squared Error: ",mean\_absolute\_error(y\_test,lin\_reg\_pred))

scoring=make\_scorer(r2\_score)

grid\_cv=GridSearchCV(ElasticNet(),param\_grid={'alpha':range(0,10),'max\_iter':[10,100,1000],'l1\_ratio':[0.1,0.4,0.8]},scoring=scoring,cv=5,refit=True)

grid\_cv.fit(X\_train,y\_train)

print("Best Parameters: "+str(grid\_cv.best\_params\_))

result=grid\_cv.cv\_results\_

print('R\_square score on the training data: ',grid\_cv.score(X\_train,y\_train))

print("R\_square Score: ",r2\_score(y\_test,grid\_cv.best\_estimator\_.predict(X\_test)))

print("Mean Squared Error: ",mean\_squared\_error(y\_test,lin\_reg\_pred))

print("Absolute Squared Error: ",mean\_absolute\_error(y\_test,lin\_reg\_pred))

quad\_reg=PolynomialFeatures(2,interaction\_only=True)

quad\_reg.fit(X\_train)

X\_train\_quad = quad\_reg.transform(X\_train)

X\_test\_quad=quad\_reg.transform(X\_test)

poly\_reg=LinearRegression()

poly\_reg.fit(X\_train\_quad,y\_train)

score\_poly=poly\_reg.score(X\_train\_quad,y\_train)

print("Accuracy: ",score\_poly)

poly\_reg\_predict=poly\_reg.predict(X\_test\_quad)

print("Mean Squared Error: ",mean\_squared\_error(y\_test,poly\_reg\_predict))

print("Mean Absolute Error: ",mean\_absolute\_error(y\_test,poly\_reg\_predict))

print("R\_Squared Score: ",r2\_score(y\_test,poly\_reg\_predict))

dt=DecisionTreeRegressor()

dt\_fit=dt.fit(X\_train,y\_train)

dt\_score=cross\_val\_score(dt\_fit,X\_train,y\_train,cv=5)

print("Mean Cross Validation Score: ",np.mean(dt\_score))

print("Score without CV: ",dt\_fit.score(X\_train,y\_train))

print("R\_square Score on the test Data: ",r2\_score(y\_test,dt\_fit.predict(X\_test)))

dt\_predict=dt.predict(X\_test)

scoring=make\_scorer(r2\_score)

grid\_cv=GridSearchCV(DecisionTreeRegressor(),param\_grid={'min\_samples\_split':range(2,10)},scoring=scoring,cv=5,refit=True)

grid\_cv.fit(X\_train,y\_train)

print("Best Parameters: ",str(grid\_cv.best\_params\_))

result=grid\_cv.cv\_results\_

print("R\_squared Score on Training Data: ",grid\_cv.best\_estimator\_.score(X\_train,y\_train))

print("R\_square Score: ",r2\_score(y\_test,grid\_cv.best\_estimator\_.predict(X\_test)))

print("Mean Squared Error: ",mean\_squared\_error(y\_test,dt\_predict))

print("Mean Absolute Error: ",mean\_absolute\_error(y\_test,dt\_predict))

rf=RandomForestRegressor()

rf\_fit=rf.fit(X\_train,y\_train)

rf\_score=cross\_val\_score(rf\_fit,X\_train,y\_train,cv=5)

print("Mean Cross Validation: ",np.mean(rf\_score))

print("Score without CV: ",rf\_fit.score(X\_train,y\_train))

print("R\_squared on the test data: ",r2\_score(y\_test,rf\_fit.predict(X\_test)))

rf\_predict=rf.predict(X\_test)

scoring=make\_scorer(r2\_score)

grid\_cv=GridSearchCV(RandomForestRegressor(),param\_grid={'min\_samples\_split':range(2,10)},scoring=scoring,cv=5,refit=True)

grid\_cv.fit(X\_train,y\_train)

result=grid\_cv.cv\_results\_

print("Best Parameters: ",str(grid\_cv.best\_params\_))

print("R\_squared Score on Training Data: ",grid\_cv.best\_estimator\_.score(X\_train,y\_train))

print("R\_square Score: ",r2\_score(y\_test,grid\_cv.best\_estimator\_.predict(X\_test)))

print("Mean Squared Error: ",mean\_squared\_error(y\_test,rf\_predict))

print("Mean Absolute Error: ",mean\_absolute\_error(y\_test,rf\_predict))

!pip install watson-machine-learning-client

from watson\_machine\_learning\_client import WatsonMachineLearningAPIClient

client=WatsonMachineLearningAPIClient(wml\_credentials)

model\_props={ client.repository.ModelMetaNames.AUTHOR\_NAME: "DIXITA",

client.repository.ModelMetaNames.AUTHOR\_EMAIL: "dixitashukla25@gmail.com",

client.repository.ModelMetaNames.NAME: "Life Expectancy Prediction"}

model\_artifact=client.repository.store\_model(grid\_cv,meta\_props=model\_props)

published\_model\_uid = client.repository.get\_model\_uid(model\_artifact)

published\_model\_uid

deployment = client.deployments.create(published\_model\_uid, name="Life Expectancy Prediction")

scoring\_endpoint = client.deployments.get\_scoring\_url(deployment)

scoring\_endpoint