LIFE EXPECTANCY PREDICITON

(REGRESSION)

Mathematics for intelligent Systems-2

Project report submitted to the Amrita Vishwa Vidyapeetham in partial fulfillment of the requirement for the Degree of

BACHELOR of TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (AI)

For Semester-2 (2022)

SUBMITTED BY

Gajji Mohit Yadav – 21128

Lekha Sathvik Devabathini – 20040

Margana Girish Vardhan – 21143

Surya Vamsi Vema – 21162

Shinoy Yandra – 21168

Introduction

The term "life expectancy" basically refers to the number of years a person can expect to live. Life expectancy depends on serval factors. The objective of this project is to predict the life expectancy from these different features like the GDP of the country, percentage of vaccination, etc. The main focus is on the techniques of regression to predict the response based on the different features provided. The major focus will be on linear regression and the implementation of higher order regressions using the concepts of Linear regression. Various regression models will be trained based on the data given and will be analyzed for their accuracy.

The project should implement various visualizations in order to provide the user with better understanding of the given dataset. The project will be implemented in python using the Machine learning libraries such as pandas, NumPy and sk-learn. The visualizations are done using the Matplotlib library in python.

Various models will be compared and the best models will be selected as the final implantation for the project.

Data set

Source: **https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who**

The data was collected from WHO and United Nations website with the help of Deeksha Russell and Duan Wang.

Description of various columns in Data set:

Life Expectancy: Life Expectancy in age (Years)

Adult Mortality: Adult Mortality Rates of both sexes (number of people dying between 15 and 60 years per 1000 population)

Infant Deaths: Number of Infant Deaths per 1000 births

Alcohol: Alcohol, recorded per capita (15+) consumption (in liters of pure alcohol)

percentage expenditure: Expenditure on health as a percentage of Gross Domestic Product per capita(%)

Hepatitis B: Hepatitis B (HepB) immunization coverage among 1-year-olds (%)

Measles: number of reported Measles cases per 1000 population

BMI: Average Body Mass Index

Polio: Polio immunization coverage among 1-year-olds (%)

Diphtheria: Diphtheria immunization coverage among 1-year-olds (%)

HIV/AIDS: number of reported HIV/AIDS cases per 1000 population

GDP: Average Gross Domestic Product per capita(%)

Statistical description of the data in Dataset:

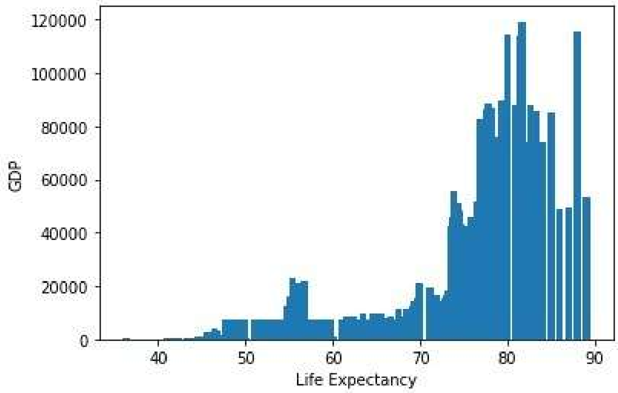
	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	Polio	Diphtheria	HIV/AIDS	GDP
count	2928.000000	2928.000000	2938.000000	2744.000000	2938.000000	2385.000000	2938.000000	2904.000000	2919.000000	2919.000000	2938.000000	2490.000000
mean	69.224932	164.796448	23.137412	4.602861	738.251295	80.940461	2419.592240	38.321247	82.550188	82.324084	1.742103	7483.158469
std	9.523867	124.292079	60.493282	4.052413	1987.914858	25.070016	11467.272489	20.044034	23.428046	23.716912	5.077785	14270.169342
min	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000	0.000000	1.000000	3.000000	2.000000	0.100000	1.681350
25%	63.100000	74.000000	0.000000	0.877500	4.685343	77.000000	0.000000	19.300000	78.000000	78.000000	0.100000	463.935626
50%	72.100000	144.000000	3.000000	3.755000	64.912906	92.000000	17.000000	43.500000	93.000000	93.000000	0.100000	1766.947595
75%	75.700000	228.000000	22.000000	7.702500	441.534144	97.000000	360.250000	56.200000	97.000000	97.000000	0.800000	5910.806335
max	89.000000	723.000000	576.000000	17.870000	19479.911610	99.000000	212183.000000	87.300000	99.000000	99.000000	50.600000	119172.741800

Visual Representations of Different features in Dataset:

Life expectancy Vs GDP:

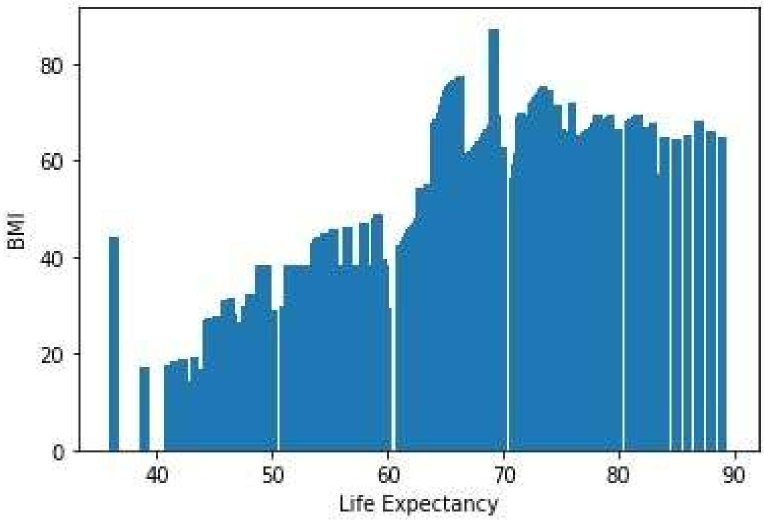
```
plt.bar(data['Life expectancy '], data ['GDP'])
plt.xlabel ("Life Expectancy")
plt.ylabel ("GDP")
```

```
Text(0, 0.5, 'GDP')
```



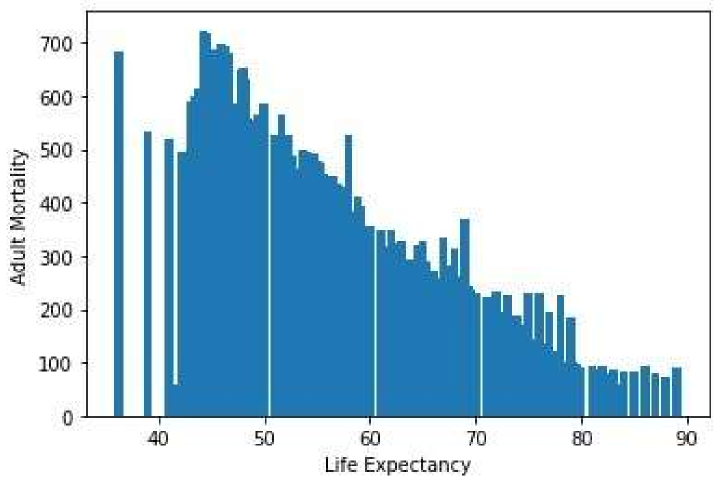
From the graph as the GDP increases life expectancy also increases also increases in most cases

Life Expectancy Vs BMI:



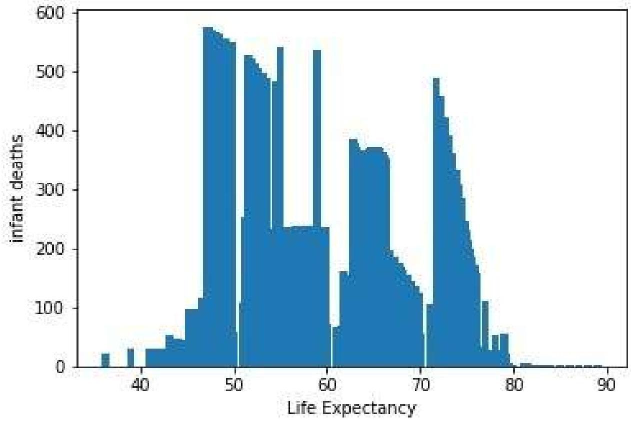
The Life expectancy is higher when the average body mass index is greater than 50

Life Expectancy Vs Adult Mortality



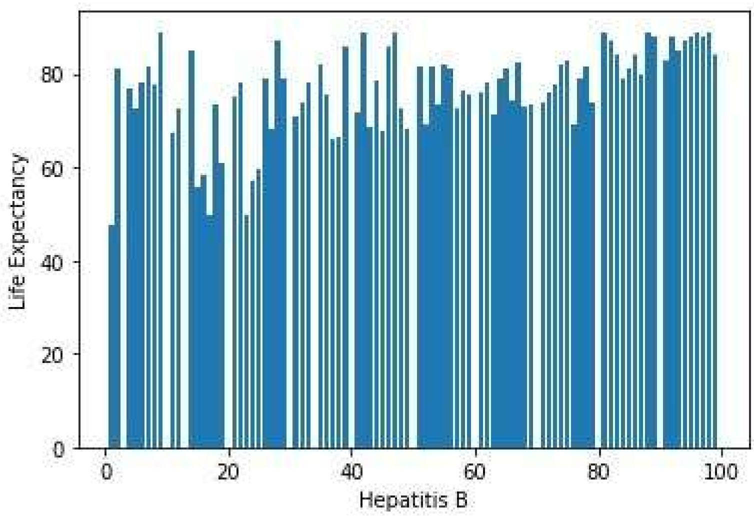
The adult mortality rate shows negative effects on the life expectancy

Life Expectancy Vs infant deaths



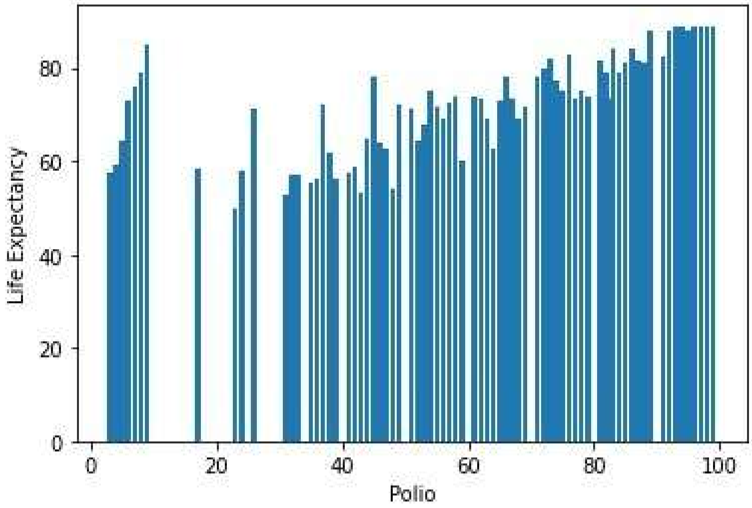
Despite of the expectations the infant mortality rate didn’t show any regular relation with life expectancy.

Hepatitis B vs Life Expectancy



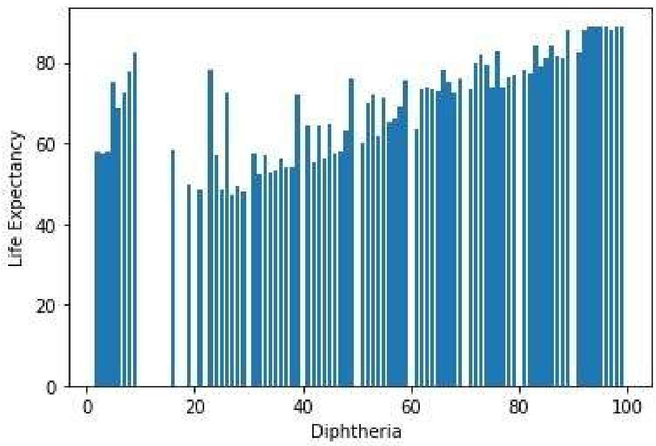
The life expectancy slightly increased with the increase in immunation percentage

Polio Vs Life Expectancy



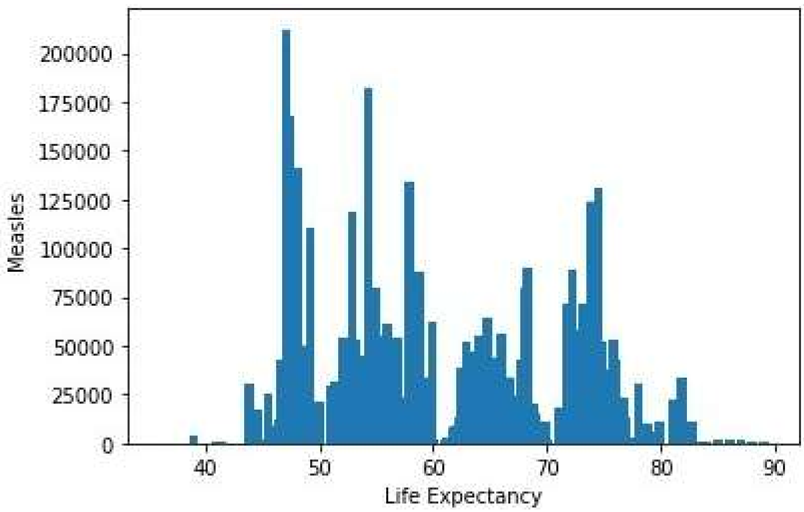
The life expectancy slightly increased with the increase in immunation percentage.

Diphtheria Vs Life Expectancy



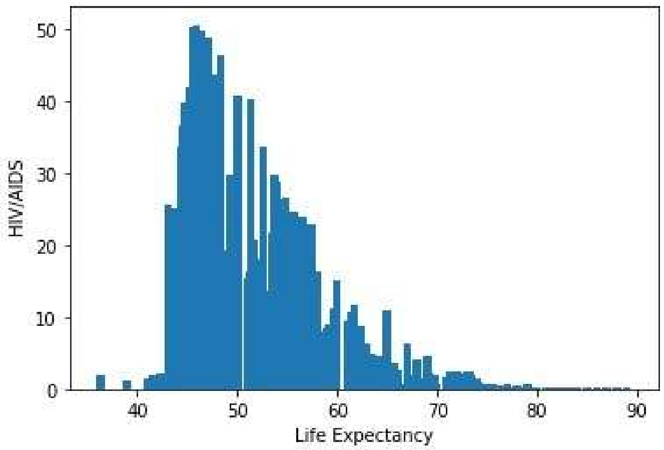
The life expectancy slightly increased with the increase in immunation percentage

Life Expectancy Vs Measles



The data in measles is lacking to show any good variation with life expectancy.

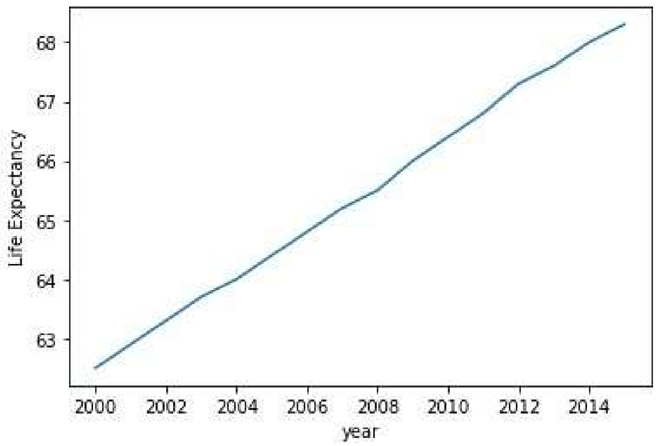
Life Expectancy Vs HIV/AIDS



The life expectancy decreased as the HIV/AIDS feature increased

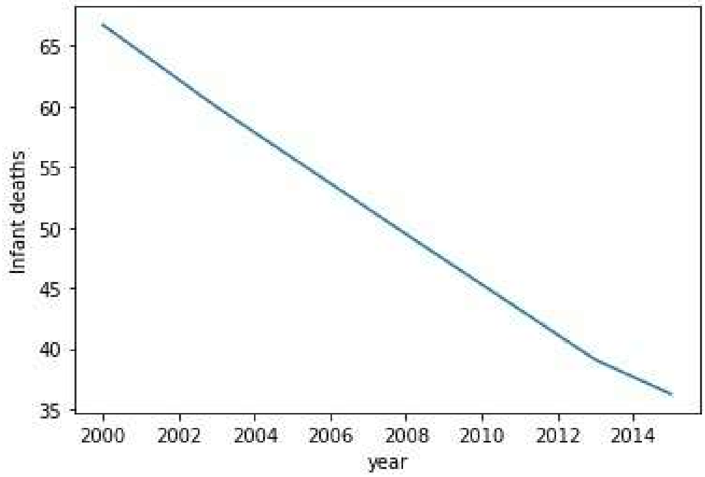
Analysis of Indian specific data:

Year Vs Life Expectancy:



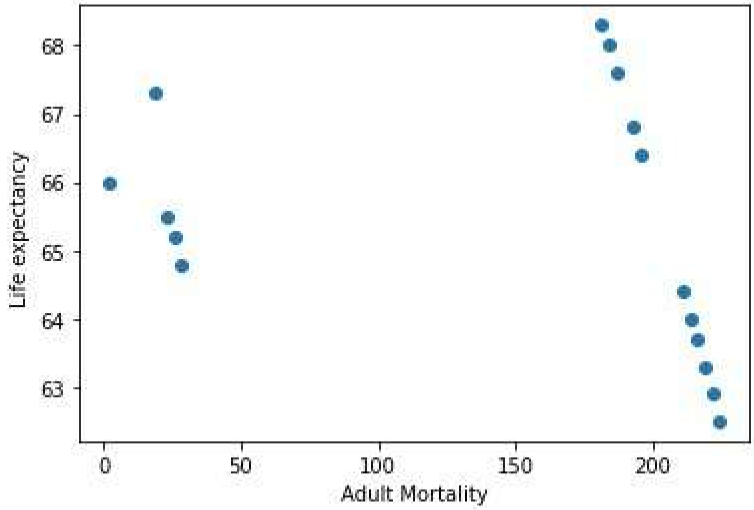
Life expectancy in India is increasing every year

Year Vs Infant deaths



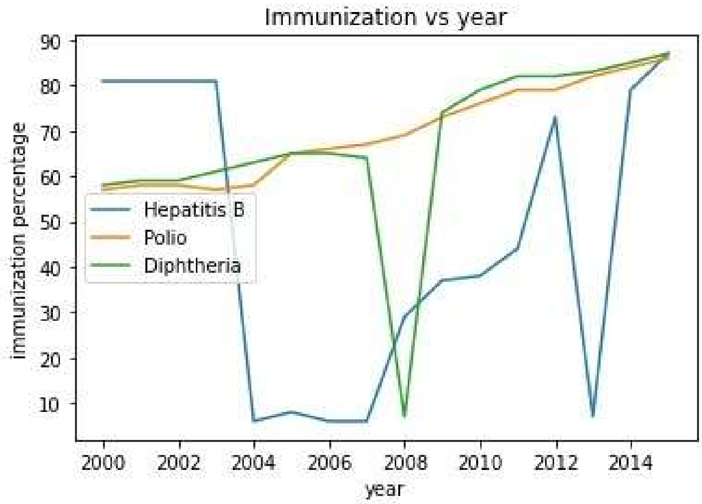
Infant deaths are decreasing every year

Adult Mortality VS Life Expectancy



Adult mortality data is not accurate enough to deduce any conclusions.

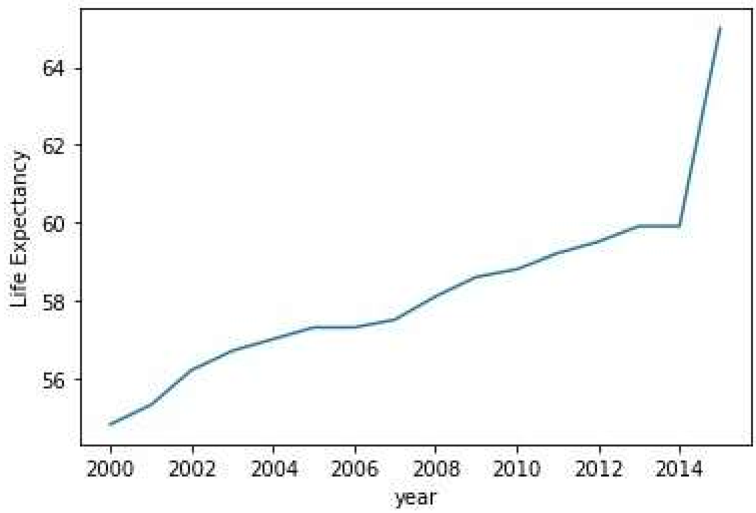
Immunization Vs year



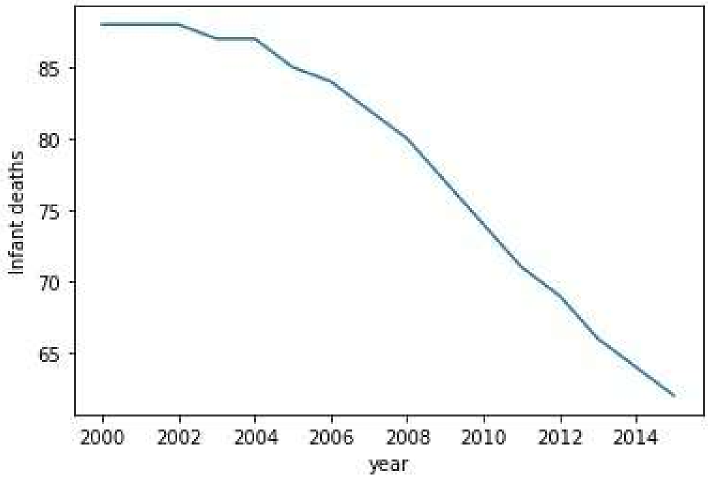
Immunization data Is not sufficiently accurate to get any observations.

Analysis of Afghanistan specific data:

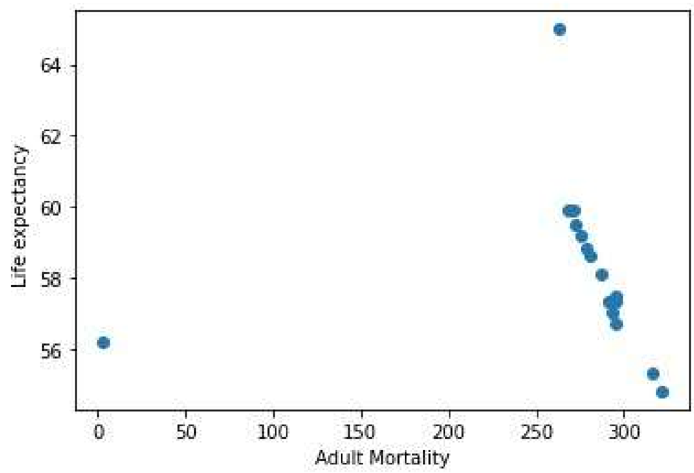
Life Expectancy Vs Year:



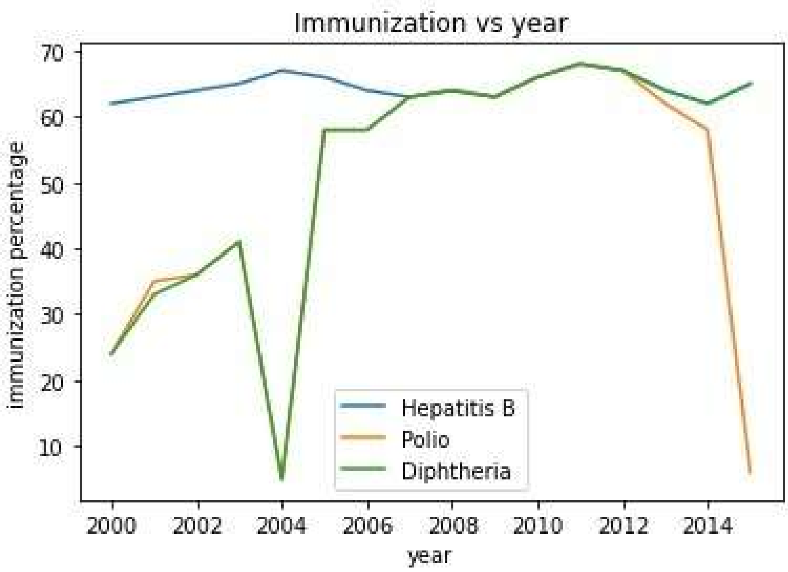
Infant Deaths Vs Year



Adult Mortality Vs Life expectancy



Immunization Vs year



Methodology

Data Cleaning:

```
data.columns
```

```
Index(['Country', 'Year', 'Life expectancy ', 'Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles ', ' BMI ', 'Polio', 'Diphtheria ', ' HIV/AIDS', 'GDP'],
      dtype='object')
```

```
data = data.drop(['Country'], axis= 'columns')
data.head()
```

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	Polio	Diphtheria	HIV/AIDS	GDP
0	2015	65.0	263.0	62.0	0.01	71.279624	65.0	1154	19.1	6.0	65.0	0.1	584.259210
1	2014	59.9	271.0	64.0	0.01	73.523582	62.0	492	18.6	58.0	62.0	0.1	612.696514
2	2013	59.9	268.0	66.0	0.01	73.219243	64.0	430	18.1	62.0	64.0	0.1	631.744976
3	2012	59.5	272.0	69.0	0.01	78.184215	67.0	2787	17.6	67.0	67.0	0.1	669.959000
4	2011	59.2	275.0	71.0	0.01	7.097109	68.0	3013	17.2	68.0	68.0	0.1	63.537231

```
data.columns
```

```
Index(['Year', 'Life expectancy ', 'Adult Mortality', 'infant deaths',
       'Alcohol', 'percentage expenditure', 'Hepatitis B', 'Measles ', ' BMI ',
       'Polio', 'Diphtheria ', ' HIV/AIDS', 'GDP'],
      dtype='object')
```

```
le = data['Life expectancy ']
print(le)
```

```
0       65.0
1       59.9
2       59.9
3       59.5
4       59.2
        ...
2933    44.3
2934    44.5
2935    44.8
2936    45.3
2937    46.0
Name: Life expectancy , Length: 2938, dtype: float64
```

Data importing and Cleaning

```
data = pd.read_csv('LifeExpectancyData.csv')
data.head()
```

	Country	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	Polio	Diphtheria	HIV/AIDS	GDP
0	Afghanistan	2015	65.0	263.0	62.0	0.01	71.279624	65.0	1154	19.1	6.0	65.0	0.1	584.259210
1	Afghanistan	2014	59.9	271.0	64.0	0.01	73.523582	62.0	492	18.6	58.0	62.0	0.1	612.696514
2	Afghanistan	2013	59.9	268.0	66.0	0.01	73.219243	64.0	430	18.1	62.0	64.0	0.1	631.744976
3	Afghanistan	2012	59.5	272.0	69.0	0.01	78.184215	67.0	2787	17.6	67.0	67.0	0.1	669.959000
4	Afghanistan	2011	59.2	275.0	71.0	0.01	7.097109	68.0	3013	17.2	68.0	68.0	0.1	63.537231

```
pd.isnull(data).sum()
```

```
Country                     0
Year                        0
Life expectancy            10
Adult Mortality            10
infant deaths               0
Alcohol                   194
percentage expenditure      0
Hepatitis B               553
Measles                     0
 BMI                       34
Polio                      19
Diphtheria                 19
 HIV/AIDS                   0
GDP                       448
dtype: int64
```

```
pd.isnull(data).sum()
```

```
Year                        0
Life expectancy            10
Adult Mortality            10
infant deaths               0
Alcohol                   194
percentage expenditure      0
Hepatitis B               553
Measles                     0
 BMI                       34
Polio                      19
Diphtheria                 19
 HIV/AIDS                   0
GDP                       448
dtype: int64
```

```
data['Life expectancy '] = data['Life expectancy '].fillna(np.mean(data['Life expectancy ']))
data['Adult Mortality'] = data['Adult Mortality'].fillna(np.mean(data['Adult Mortality']))
data['Alcohol'] = data['Alcohol'].fillna(np.mean(data['Alcohol']))
data['Hepatitis B'] = data['Hepatitis B'].fillna(np.mean(data['Hepatitis B']))
data[' BMI '] = data[' BMI '].fillna(np.mean(data[' BMI ']))
data['Polio'] = data['Polio'].fillna(np.mean(data['Polio']))
data['Diphtheria '] = data['Diphtheria '].fillna(np.mean(data['Diphtheria ']))
data['GDP'] = data['GDP'].fillna(np.mean(data['GDP']))
```

```
pd.isna(data).sum()
```

```
Year                      0
Life expectancy           0
Adult Mortality           0
infant deaths             0
Alcohol                   0
percentage expenditure    0
Hepatitis B               0
Measles                   0
 BMI                      0
Polio                     0
Diphtheria                0
 HIV/AIDS                 0
GDP                       0
dtype: int64
```

```
data.head()
```

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	Polio	Diphtheria	HIV/AIDS	GDP
0	2015	65.0	263.0	62.0	0.01	71.279624	65.0	1154	19.1	6.0	65.0	0.1	584.259210
1	2014	59.9	271.0	64.0	0.01	73.523582	62.0	492	18.6	58.0	62.0	0.1	612.696514
2	2013	59.9	268.0	66.0	0.01	73.219243	64.0	430	18.1	62.0	64.0	0.1	631.744976
3	2012	59.5	272.0	69.0	0.01	78.184215	67.0	2787	17.6	67.0	67.0	0.1	669.959000
4	2011	59.2	275.0	71.0	0.01	7.097109	68.0	3013	17.2	68.0	68.0	0.1	63.537231

```
independent_vars = data.drop(['Life expectancy '], axis = "columns")
independent_vars.head()
```

	Year	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	Polio	Diphtheria	HIV/AIDS	GDP
0	2015	263.0	62.0	0.01	71.279624	65.0	1154	19.1	6.0	65.0	0.1	584.259210
1	2014	271.0	64.0	0.01	73.523582	62.0	492	18.6	58.0	62.0	0.1	612.696514
2	2013	268.0	66.0	0.01	73.219243	64.0	430	18.1	62.0	64.0	0.1	631.744976
3	2012	272.0	69.0	0.01	78.184215	67.0	2787	17.6	67.0	67.0	0.1	669.959000
4	2011	275.0	71.0	0.01	7.097109	68.0	3013	17.2	68.0	68.0	0.1	63.537231

```
dependant_var = data['Life expectancy ']
dependant_var.head()
```

```
0    65.0
1    59.9
2    59.9
3    59.5
4    59.2
Name: Life expectancy , dtype: float64
```

```
x_train,x_test, y_train, y_test = train_test_split(independent_vars, dependant_var, train_size= 0.8)
x_train.head()
```

	Year	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	Polio	Diphtheria	HIV/AIDS	GDP
506	2005	76.0	2.0	8.00	6333.177967	14.000000	6	61.3	93.0	93.0	0.1	36189.588380
1660	2006	221.0	8.0	0.01	55.798370	68.000000	22	24.8	68.0	68.0	1.3	944.134851
651	2005	116.0	0.0	11.59	167.231990	80.940461	2	57.5	96.0	96.0	0.1	1224.245900
1103	2002	35.0	5.0	2.47	24.657618	80.940461	298	18.5	61.0	57.0	4.1	321.481324
2624	2008	344.0	14.0	1.33	69.359248	24.000000	187	2.4	8.0	81.0	4.8	513.391914

```
y_train.head()
```

```
506     81.0
1660    69.0
651     75.2
1103    52.8
2624    56.2
Name: Life expectancy , dtype: float64
```

```
x_test.head()
```

	Year	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	Polio	Diphtheria	HIV/AIDS	GDP
425	2006	361.0	24.0	4.50	21.249153	92.000000	784	14.5	88.0	92.0	3.8	165.879418
1249	2000	144.0	30.0	0.20	0.000000	67.000000	726	49.5	83.0	8.0	0.1	7483.158469
924	2005	11.0	0.0	9.95	4816.589613	80.940461	1	58.1	97.0	97.0	0.1	38969.171630
2442	2014	141.0	3.0	2.37	42.730828	99.000000	1686	22.7	99.0	99.0	0.1	382.549940
110	2001	141.0	1.0	2.86	53.193730	69.000000	69	47.4	97.0	94.0	0.1	694.435119

```
y_test.head()
```

```
425     54.1
1249    70.0
924     78.9
2442    74.7
110     72.6
Name: Life expectancy , dtype: float64
```

Implementation of regression Models

Models to predict Life expectancy from Adult Mortality rate

In [18]:
```
dict = {
    'Adult Mortality' : data['Adult Mortality'],
    'Life expectancy ' : data['Life expectancy ']

}
single_feature_data = pd.DataFrame(dict)
```

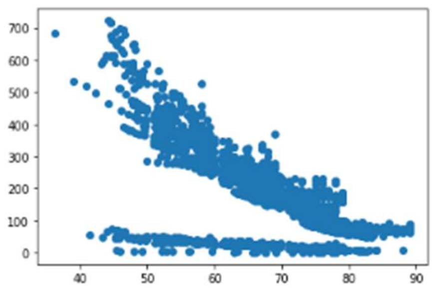
In [19]:
```
single_feature_data.head()
```

Out[19]:

	Adult Mortality	Life expectancy
0	263.0	65.0
1	271.0	59.9
2	268.0	59.9
3	272.0	59.5
4	275.0	59.2

In [20]:
```
plt.scatter(single_feature_data['Life expectancy ' ],single_feature_data['Adult Mortality'])
```

Out[20]: <matplotlib.collections.PathCollection at 0x1906432afe0>



In [21]:
```
X = single_feature_data.drop(['Life expectancy '], axis = 1)
Y= single_feature_data['Life expectancy ']
```

In [22]:
```
X_single_train, X_single_test, Y_single_train, Y_single_test = train_test_split(X,Y, train_size=0.8)
```

single variable Linear Regression Model

In [23]:
```
single_Linear_model = LinearRegression()
single_Linear_model.fit(X_single_train,Y_single_train)
```

Out[23]: LinearRegression()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [24]:
```
single_Linear_model.score(X_single_test, Y_single_test)
```

Out[24]:
```
0.49711848128726455
```

In [25]:
```
rmse_single_Linear_train  =  np.sqrt(mean_squared_error(Y_single_train , single_Linear_model.predict(X_single_train)))
mae_single_Linear_train = mean_absolute_error(Y_single_train , single_Linear_model.predict(X_single_train))
print("root mean squared error : ", rmse_single_Linear_train)
print("mean absolute error : ", mae_single_Linear_train)
```

```
root mean squared error :  6.868292353546761
mean absolute error :  4.839897628982059
```

In [26]:
```
rmse_single_Linear_test  =  np.sqrt(mean_squared_error(Y_single_test , single_Linear_model.predict(X_single_test)))
mae_single_Linear_test = mean_absolute_error(Y_single_test , single_Linear_model.predict(X_single_test))
print("root mean squared error : ", rmse_single_Linear_test)
print("mean absolute error : ", mae_single_Linear_test)
```

```
root mean squared error :  6.641210439357417
mean absolute error :  4.764130941304089 5
```

single variable quadratic Linear Regression Model

In [27]:
```
poly2 = PolynomialFeatures(degree= 2, include_bias= False)
quadratic_single_features_train = poly2.fit_transform(X_single_train)
quadratic_single_features_train.shape
```

Out[27]:
```
(2350, 2)
```

In [28]:
```
quadratic_single_features_test = poly2.fit_transform(X_single_test)
```

In [29]:
```
single_quadratic_model = LinearRegression()
single_quadratic_model.fit(quadratic_single_features_train,Y_single_train)
```

Out[29]: LinearRegression()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [30]:
```
single_quadratic_model.score(quadratic_single_features_test, Y_single_test)
```

Out[30]:
```
0.5104193171546023
```

In [31]:
```
rmse_single_quadratic_train  =  np.sqrt(mean_squared_error(Y_single_train , single_quadratic_model.predict(quadratic_single_features_train)))
mae_single_quadratic_train = mean_absolute_error(Y_single_train , single_quadratic_model.predict(quadratic_single_features_train))
print("root mean squared error : ", rmse_single_quadratic_train)
print("mean absolute error : ", mae_single_quadratic_train)
```

```
root mean squared error :  6.749141014978261
mean absolute error :  4.800973467428455
```

In [32]:
```
rmse_single_quadratic_test  =  np.sqrt(mean_squared_error(Y_single_test , single_quadratic_model.predict(quadratic_single_features_test)))
mae_single_quadratic_test = mean_absolute_error(Y_single_test , single_quadratic_model.predict(quadratic_single_features_test))
print("root mean squared error : ", rmse_single_quadratic_test)
print("mean absolute error : ", mae_single_quadratic_test)
```

```
root mean squared error :  6.552794390317189
mean absolute error :  4.737834643290052
```

single variable cubic Regression Model

In [33]:
```
poly3 = PolynomialFeatures(degree= 3, include_bias= False)
cubic_single_features_test = poly3.fit_transform(X_single_test)
cubic_single_features_train = poly3.fit_transform(X_single_train)
cubic_single_features_train.shape
```
Out[33]:
```
(2350, 3)
```

In [34]:
```
single_cubic_model = LinearRegression()
single_cubic_model.fit(cubic_single_features_train,Y_single_train)
```
Out[34]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [35]:
```
single_cubic_model.score(cubic_single_features_test, Y_single_test)
```
Out[35]:
```
0.5863687160978437
```

In [36]:
```
rmse_single_cubic_train  =  np.sqrt(mean_squared_error(Y_single_train , single_cubic_model.predict(cubic_single_features_train)))
mae_single_cubic_train = mean_absolute_error(Y_single_train , single_cubic_model.predict(cubic_single_features_train))
print("root mean squared error : ", rmse_single_cubic_train)
print("mean absolute error : ", mae_single_cubic_train)
```
```
root mean squared error :  6.196217335448624
mean absolute error :  4.142302888328426
```

In [37]:
```
rmse_single_cubic_test  =  np.sqrt(mean_squared_error(Y_single_test , single_cubic_model.predict(cubic_single_features_test)))
mae_single_cubic_test = mean_absolute_error(Y_single_test , single_cubic_model.predict(cubic_single_features_test))
print("root mean squared error : ", rmse_single_cubic_test)
print("mean absolute error : ", mae_single_cubic_test)
```
```
root mean squared error :  6.023114140287135
mean absolute error :  4.143872759800894
```

single variable biquadratic Regression Model

In [38]:
```
poly4 = PolynomialFeatures(degree= 4, include_bias= False)
biquadratic_single_features_test = poly4.fit_transform(X_single_test)
biquadratic_single_features_train = poly4.fit_transform(X_single_train)
biquadratic_single_features_train.shape
```
Out[38]:
```
(2350, 4)
```

In [39]:
```
single_biquadratic_model = LinearRegression()
single_biquadratic_model.fit(biquadratic_single_features_train, Y_single_train)
```
Out[39]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [40]:
```
single_biquadratic_model.score(biquadratic_single_features_test, Y_single_test)
```
Out[40]:
```
0.6129265863788559
```

In [41]:
```
rmse_single_biquadratic_train  =  np.sqrt(mean_squared_error(Y_single_train , single_biquadratic_model.predict(biquadratic_single_features_train)))
mae_single_biquadratic_train = mean_absolute_error(Y_single_train , single_biquadratic_model.predict(biquadratic_single_features_train))
print("root mean squared error : ", rmse_single_biquadratic_train)
print("mean absolute error : ", mae_single_biquadratic_train)
```
```
root mean squared error :  5.993028410832365
mean absolute error :  3.8534189823491887
```

In [42]:
```
rmse_single_biquadratic_test  =  np.sqrt(mean_squared_error(Y_single_test , single_biquadratic_model.predict(biquadratic_single_features_test)))
mae_single_biquadratic_test = mean_absolute_error(Y_single_test , single_biquadratic_model.predict(biquadratic_single_features_test))
print("root mean squared error : ", rmse_single_biquadratic_test)
print("mean absolute error : ", mae_single_biquadratic_test)
```
```
root mean squared error :  5.826544605228608
mean absolute error :  3.9140136292120955
```

Like this the models are implemented till seventh order polynomial regression for single variable regression and the errors are analyzed to find the best model for the given data

Models with all the avilable features

Linear regression model

In [58]:
```
model_Linear = LinearRegression()
model_Linear.fit(x_train, y_train)
```

Out[58]: LinearRegression()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [59]:
```
model_Linear.score(x_test, y_test)
```

Out[59]: 0.7541556984147635

In [60]:
```
rmse_Linear_train  =  np.sqrt(mean_squared_error(y_train , model_Linear.predict(x_train)))
mae_Linear_train = mean_absolute_error(y_train , model_Linear.predict(x_train))
print("root mean squared error : ", rmse_Linear_train)
print("mean absolute error : ", mae_Linear_train)
```

```
root mean squared error :  4.7376032868599705
mean absolute error :  3.5977758911595012
```

In [61]:
```
rmse_Linear_test  =  np.sqrt(mean_squared_error(y_test , model_Linear.predict(x_test)))
mae_Linear_test = mean_absolute_error(y_test , model_Linear.predict(x_test))
print("root mean squared error : ", rmse_Linear_test)
print("mean absolute error : ", mae_Linear_test)
```

```
root mean squared error :  4.707217562799239
mean absolute error :  3.4990816867014374
```

In [62]:
```
model_Linear.coef_
```

Out[62]:
```
array([ 1.17292660e-01, -2.63270102e-02, -1.54470867e-02,  3.54048103e-01,
        1.67336160e-04, -2.76259865e-02,  2.62064976e-06,  8.30754528e-02,
        4.02821815e-02,  7.08501456e-02, -4.82600137e-01,  7.82744222e-05])
```

Quadratic Regression Model

In [63]:
```
poly2 = PolynomialFeatures(degree= 2, include_bias= False)
quadeatic_features_train = poly2.fit_transform(x_train)
```

In [64]:
```
x_train.shape
```

Out[64]: (2350, 12)

In [65]:
```
quadeatic_features_train.shape
```

Out[65]: (2350, 90)

In [66]:
```
quadeatic_features_test = poly2.fit_transform(x_test)
```

In [67]:
```
model_Quadratic = LinearRegression()
model_Quadratic.fit(quadeatic_features_train, y_train)
```

Out[67]: LinearRegression()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [68]:
```
model_Quadratic.score(quadeatic_features_test, y_test)
```

Out[68]: 0.8534794416483015

In [69]:
```
rmse_Quadratic_train  =  np.sqrt(mean_squared_error(y_train , model_Quadratic.predict(quadeatic_features_train)))
mae_Quadratic_train = mean_absolute_error(y_train , model_Quadratic.predict(quadeatic_features_train))
print("root mean squared error : ", rmse_Quadratic_train)
print("mean absolute error : ", mae_Quadratic_train)
```

```
root mean squared error :  3.3826224503215
mean absolute error :  2.477233774559755
```

In [70]:
```
rmse_Quadratic_test  =  np.sqrt(mean_squared_error(y_test , model_Quadratic.predict(quadeatic_features_test)))
mae_Quadratic_test = mean_absolute_error(y_test , model_Quadratic.predict(quadeatic_features_test))
print("root mean squared error : ", rmse_Quadratic_test)
print("mean absolute error : ", mae_Quadratic_test)
```

```
root mean squared error :  3.6339879323734587
mean absolute error :  2.579126504443397
```

Cubic Regression Model

In [71]:
```
poly3 = PolynomialFeatures(degree= 3, include_bias= False)
cubic_features_train = poly3.fit_transform(x_train)
```

In [72]:
```
x_train.shape
```

Out[72]:
```
(2350, 12)
```

In [73]:
```
cubic_features_train.shape
```

Out[73]:
```
(2350, 454)
```

In [74]:
```
cubic_features_test = poly3.fit_transform(x_test)
```

In [75]:
```
cubic_features_test.shape
```

Out[75]:
```
(588, 454)
```

In [76]:
```
model_Cubic = LinearRegression()
model_Cubic.fit(cubic_features_train, y_train)
```

Out[76]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [77]:
```
model_Cubic.score(cubic_features_test, y_test)
```

Out[77]:
```
0.36513218735004627
```

In [78]:
```
rmse_Cubic_train  =  np.sqrt(mean_squared_error(y_train , model_Cubic.predict(cubic_features_train)))
mae_Cubic_train = mean_absolute_error(y_train , model_Cubic.predict(cubic_features_train))
print("root mean squared error : ", rmse_Cubic_train)
print("mean absolute error : ", mae_Cubic_train)
```

```
root mean squared error :  3.0931163295629394
mean absolute error :  2.2725924074041473
```

In [79]:
```
rmse_Cubic_test  =  np.sqrt(mean_squared_error(y_test , model_Cubic.predict(cubic_features_test)))
mae_Cubic_test = mean_absolute_error(y_test , model_Cubic.predict(cubic_features_test))
print("root mean squared error : ", rmse_Cubic_test)
print("mean absolute error : ", mae_Cubic_test)
```

```
root mean squared error :  7.564423808960748
mean absolute error :  3.7629817912337984
```

we observed a drastic drop in score from Quadratic to Linear So we will stop the increase in degree here as the model suffer from overfitting

Using PCA on over fitted Cubic regression model to observe the change.

n [80]:
```
pca = PCA(n_components=40)
cubic_features_train_pca = pca.fit_transform(cubic_features_train)
cubic_features_test_pca = pca.fit_transform(cubic_features_test)
```

n [81]:
```
cubic_features_train_pca.shape
```

ut[81]:
```
(2350, 40)
```

n [82]:
```
model_Cubic_pca = LinearRegression()
model_Cubic_pca.fit(cubic_features_train_pca, y_train)
```

ut[82]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

n [83]:
```
model_Cubic_pca.score(cubic_features_test_pca, y_test)
```

ut[83]:
```
-0.0027554661660789126
```

that didnt work out well so we can tell that the PCA decomposition cant solve overfitting.

Biquadratic Regression Model

In [84]:
```
poly4 = PolynomialFeatures(degree= 4, include_bias= False)
biquadratic_features_train = poly4.fit_transform(x_train)
biquadratic_features_train.shape
```

Out[84]: (2350, 1819)

In [85]:
```
biquadratic_features_test = poly4.fit_transform(x_test)
biquadratic_features_test.shape
```

Out[85]: (588, 1819)

In [86]:
```
model_Biquadratic = LinearRegression()
model_Biquadratic.fit(biquadratic_features_train, y_train)
```

Out[86]: LinearRegression()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [87]:
```
model_Biquadratic.score(biquadratic_features_test, y_test)
```

Out[87]: -468.8641691382015

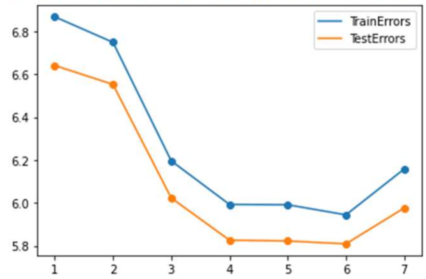
Results

Analysis of Performance of different models

For single variable regression

In [88]:
```
Degrees_of_regression = [1, 2, 3,4,5,6,7]
Train_errors = [rmse_single_Linear_train, rmse_single_quadratic_train, rmse_single_cubic_train, rmse_single_biquadratic_train, rmse_single_pentanomial
Test_errors = [rmse_single_Linear_test, rmse_single_quadratic_test, rmse_single_cubic_test, rmse_single_biquadratic_test, rmse_single_pentanomial_test
plt.plot(Degrees_of_regression,Train_errors, label = "TrainErrors")
plt.plot(Degrees_of_regression,Test_errors, label = "TestErrors")
plt.legend()
plt.scatter(Degrees_of_regression,Train_errors)
plt.scatter(Degrees_of_regression,Test_errors)
print(Train_errors)
```

[6.868292353546761, 6.749141014978261, 6.196217335448624, 5.993028410832365, 5.9915614330598, 5.9444227257083035, 6.158349584969995]

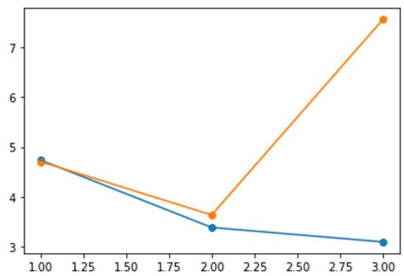


by analysing the errors the hexanomial regression is best for this single variable regression

For multivariable regression

In [89]:
```
Degrees_of_regression = [1, 2, 3]
Train_errors = [rmse_Linear_train, rmse_Quadratic_train, rmse_Cubic_train]
Test_errors = [rmse_Linear_test, rmse_Quadratic_test, rmse_Cubic_test]
plt.plot(Degrees_of_regression,Train_errors)
plt.plot(Degrees_of_regression,Test_errors)
plt.scatter(Degrees_of_regression,Train_errors)
plt.scatter(Degrees_of_regression,Test_errors)
```

Out[89]: <matplotlib.collections.PathCollection at 0x1906c9da1a0>



after a regression of degree 2 the model shows the problem of overfitting. So we can conclude that the quadratic regression will be the best model for the given regression scenario

Discussion

This dataset contains factors affecting life expectancy with the consideration of demographic variables, income composition, mortality rates, economic factors, immunization affect by formulating a regression model.

The dataset aims to answer the following key questions:

1. Do various predicting factors which has been chosen initially really affect the Life expectancy? What are the predicting variables affecting the life expectancy?
2. Should a country having a lower life expectancy value (<65) increase its healthcare expenditure in order to improve its average lifespan?
3. Does Life Expectancy have positive or negative correlation with eating habits, lifestyle, exercise, smoking, drinking alcohol etc.
4. Do densely populated countries tend to have lower life expectancy?