In [31]:
```
import numpy as np
import pandas as pd
import matplotlib as plt
```

In [32]:
```
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It incl
udes your credentials.
# You might want to remove those credentials before you share the notebook.
client_2ab26ee219f04fe08de31d115315f066 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='kGqMn6Ir-jfdBfDTojxNJA5ppQy6ckU73Xs7VZWq9Wor',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3-api.us-geo.objectstorage.service.networklayer.co
m')

body = client_2ab26ee219f04fe08de31d115315f066.get_object(Bucket='predictingli
feexpectancy-donotdelete-pr-5dkqtwxuyvvnn2',Key='Life Expectancy Data.csv')['B
ody']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__,
body )

# If you are reading an Excel file into a pandas DataFrame, replace `read_csv`
by `read_excel` in the next statement.
df = pd.read_csv(body)
df.head()
```

Out[32]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0

5 rows × 22 columns

In [33]:
```
df.isnull().sum()
```

Out[33]:
```
Country                            0
Year                               0
Status                             0
Life expectancy                   10
Adult Mortality                   10
infant deaths                      0
Alcohol                          194
percentage expenditure             0
Hepatitis B                      553
Measles                            0
 BMI                              34
under-five deaths                  0
Polio                             19
Total expenditure                226
Diphtheria                        19
 HIV/AIDS                          0
GDP                              448
Population                       652
 thinness  1-19 years             34
 thinness 5-9 years               34
Income composition of resources  167
Schooling                        163
dtype: int64
```

In [34]:
```
df['Life expectancy '].fillna(df['Life expectancy '].mean(), inplace = True)
```

In [35]:
```
df['Adult Mortality'].fillna(df['Adult Mortality'].mean(), inplace = True)
```

In [36]:
```
df['Alcohol'].fillna(df.Alcohol.mean(), inplace = True)
```

In [37]:
```
df['Hepatitis B'].fillna(df['Hepatitis B'].mean(), inplace = True)
```

In [38]:
```
df[' BMI '].fillna(df[' BMI '].mean(),inplace = True)
```

In [39]:
```
df['Polio'].fillna(df['Polio'].mean(), inplace = True)
```

In [40]:
```
df['Total expenditure'].fillna(df['Total expenditure'].mean(), inplace = True)
```

In [41]:
```
df['Diphtheria '].fillna(df['Diphtheria '].mean(), inplace = True)
```

In [42]:
```
df['GDP'].fillna(df['GDP'].mean(), inplace = True)
```

In [43]:
```
df['Population'].fillna(df['Population'].mean(), inplace = True)
```

In [44]:
```
df[' thinness  1-19 years'].fillna(df[' thinness  1-19 years'].mean(), inplace
= True)
```

In [45]:
```
df[' thinness 5-9 years'].fillna(df[' thinness 5-9 years'].mean(), inplace = True)
```

In [46]:
```
df['Income composition of resources'].fillna(df['Income composition of resources'].mean(), inplace = True)
```

In [47]:
```
df['Schooling'].fillna(df['Schooling'].mean(), inplace = True)
```

In [48]:
```
df.isnull().sum()
```

Out[48]:
```
Country                          0
Year                             0
Status                           0
Life expectancy                  0
Adult Mortality                  0
infant deaths                    0
Alcohol                          0
percentage expenditure           0
Hepatitis B                      0
Measles                          0
 BMI                             0
under-five deaths                0
Polio                            0
Total expenditure                0
Diphtheria                       0
 HIV/AIDS                        0
GDP                              0
Population                       0
 thinness  1-19 years            0
 thinness 5-9 years              0
Income composition of resources  0
Schooling                        0
dtype: int64
```

In [49]:
```
# seperating dependent and independent featues

X = df.drop('Life expectancy ', axis = 1)
y = df['Life expectancy ']
X.head()
print(y.head())
print(X.head())
```

```
0    65.0
1    59.9
2    59.9
3    59.5
4    59.2
Name: Life expectancy , dtype: float64
       Country  Year      Status  Adult Mortality  infant deaths  Alcohol  \
0  Afghanistan  2015  Developing            263.0             62     0.01
1  Afghanistan  2014  Developing            271.0             64     0.01
2  Afghanistan  2013  Developing            268.0             66     0.01
3  Afghanistan  2012  Developing            272.0             69     0.01
4  Afghanistan  2011  Developing            275.0             71     0.01

   percentage expenditure  Hepatitis B  Measles    BMI   ...  Polio  \
0               71.279624         65.0      1154   19.1  ...    6.0
1               73.523582         62.0       492   18.6  ...   58.0
2               73.219243         64.0       430   18.1  ...   62.0
3               78.184215         67.0      2787   17.6  ...   67.0
4                7.097109         68.0      3013   17.2  ...   68.0

   Total expenditure  Diphtheria    HIV/AIDS         GDP  Population  \
0               8.16         65.0        0.1  584.259210  33736494.0
1               8.18         62.0        0.1  612.696514    327582.0
2               8.13         64.0        0.1  631.744976  31731688.0
3               8.52         67.0        0.1  669.959000   3696958.0
4               7.87         68.0        0.1   63.537231   2978599.0

    thinness  1-19 years   thinness 5-9 years  \
0                   17.2                 17.3
1                   17.5                 17.5
2                   17.7                 17.7
3                   17.9                 18.0
4                   18.2                 18.2

   Income composition of resources  Schooling
0                            0.479       10.1
1                            0.476       10.0
2                            0.470        9.9
3                            0.463        9.8
4                            0.454        9.5

[5 rows x 21 columns]
```

In [50]:
```
# count of categoricl features in country column

categ_feature=len(df['Country'].unique())
print(categ_feature)
```

```
193
```

In [51]:
```
cntdum=pd.get_dummies(X['Country'])
stadum=pd.get_dummies(X['Status'])
X.drop('Country',inplace=True,axis=1)
```

In [52]:
```
X.drop('Status',inplace=True,axis=1)
X=pd.concat([X,cntdum,stadum],axis=1)
```

In [53]:
```
#train-test split


from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X, y, test_size=0.30, random_st
ate=101)
```

In [54]:
```
#fitting model

from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,Y_train)
```

Out[54]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [55]:
```
#predicting values using trained model

y_pred=reg.predict(X_test)
```

In [60]:
```
#side by side comparison of first few predicted and true value

print("Actual Value    Predicted value")

k=0
for i in Y_test:
    for j in range(len(y_pred)):
        print(i,"          ",y_pred[k])
        k+=1
        break
    if k==15:
        break
```

```
Actual Value    Predicted value
62.5            52.68472476882823
53.6            54.34104169205216
83.3            83.51996572343921
64.3            63.431597490154445
73.5            73.49196823901343
72.7            72.45500156870435
68.2            69.13442748050466
81.1            80.2847937771104
59.7            60.64441361769582
81.4            82.89562630549699
63.8            64.65367974507853
72.2            71.33371754002849
74.9            75.6707066608675
75.4            75.75611990362734
48.2            48.551219573527874
```

In [57]:
```
from sklearn import metrics
```

In [58]:
```
# rmse error

print("RMSE ERROR: ",np.sqrt(metrics.mean_squared_error(Y_test, y_pred)))
```

```
RMSE ERROR:  1.9191954844716705
```

In [59]:
```
#accuracy of model

res=reg.score(X_test,Y_test)
print("Accuarcy : ",res*100)
```

```
Accuarcy :  95.7438870570211
```

In [ ]:
```

```