Installing important packages

```
!pip install matplotlib
!pip install pandas
!pip install numpy
!pip install seaborn
!pip install sklearn
!pip install pandoc
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: matplotlib in /home/madhu/.local/lib/python3.1
Requirement already satisfied: kiwisolver>=1.0.1 in /home/madhu/.local/lib/py
Requirement already satisfied: fonttools>=4.22.0 in /home/madhu/.local/lib/py
Requirement already satisfied: numpy>=1.17 in /home/madhu/.local/lib/python3.
Requirement already satisfied: pyparsing>=2.2.1 in /usr/lib/python3/dist-pack
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3
Requirement already satisfied: cycler>=0.10 in /home/madhu/.local/lib/python3
Requirement already satisfied: pillow>=6.2.0 in /usr/lib/python3/dist-package
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (fr
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in /home/madhu/.local/lib/python3.10/si
Requirement already satisfied: numpy>=1.21.0 in /home/madhu/.local/lib/python
Requirement already satisfied: pytz>=2020.1 in /usr/lib/python3/dist-packages
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (fr
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: numpy in /home/madhu/.local/lib/python3.10/sit
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: seaborn in /home/madhu/.local/lib/python3.10/s
Requirement already satisfied: numpy>=1.15 in /home/madhu/.local/lib/python3.
Requirement already satisfied: matplotlib>=2.2 in /home/madhu/.local/lib/pyth
Requirement already satisfied: pandas>=0.23 in /home/madhu/.local/lib/python3
Requirement already satisfied: scipy>=1.0 in /home/madhu/.local/lib/python3.1
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pyparsing>=2.2.1 in /usr/lib/python3/dist-pack
Requirement already satisfied: cycler>=0.10 in /home/madhu/.local/lib/python3
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3
Requirement already satisfied: kiwisolver>=1.0.1 in /home/madhu/.local/lib/py
Requirement already satisfied: pillow>=6.2.0 in /usr/lib/python3/dist-package
Requirement already satisfied: fonttools>=4.22.0 in /home/madhu/.local/lib/py
Requirement already satisfied: pytz>=2020.1 in /usr/lib/python3/dist-packages
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (fr
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: sklearn in /home/madhu/.local/lib/python3.10/s
Requirement already satisfied: scikit-learn in /home/madhu/.local/lib/python3
Requirement already satisfied: numpy>=1.17.3 in /home/madhu/.local/lib/python
Requirement already satisfied: joblib>=1.0.0 in /home/madhu/.local/lib/python
Requirement already satisfied: threadpoolctl>=2.0.0 in /home/madhu/.local/lib
Requirement already satisfied: scipy>=1.3.2 in /home/madhu/.local/lib/python3
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandoc in /home/madhu/.local/lib/python3.10/si
Requirement already satisfied: ply in /home/madhu/.local/lib/python3.10/site-
Requirement already satisfied: plumbum in /home/madhu/.local/lib/python3.10/s
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.linear_model import LinearRegression
```

Describing data

```
data2=pd.read_csv("Data/Life Expectancy Data.csv")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
last)
<ipython-input-1-1a792c391a69> in <module>
----> 1 data2=pd.read_csv("Data/Life Expectancy Data.csv")

NameError: name 'pd' is not defined
```

```
display(data2)
```

```
data2.describe()
```

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentag expenditur
count	2938.000000	2928.000000	2928.000000	2938.000000	2744.000000	2938.00000
mean	2007.518720	69.224932	164.796448	30.303948	4.602861	738.25129
std	4.613841	9.523867	124.292079	117.926501	4.052413	1987.91485
min	2000.000000	36.300000	1.000000	0.000000	0.010000	0.00000
25%	2004.000000	63.100000	74.000000	0.000000	0.877500	4.68534
50%	2008.000000	72.100000	144.000000	3.000000	3.755000	64.91290
75%	2012.000000	75.700000	228.000000	22.000000	7.702500	441.53414
max	2015.000000	89.000000	723.000000	1800.000000	17.870000	19479.91161

```
data2.dtypes
```

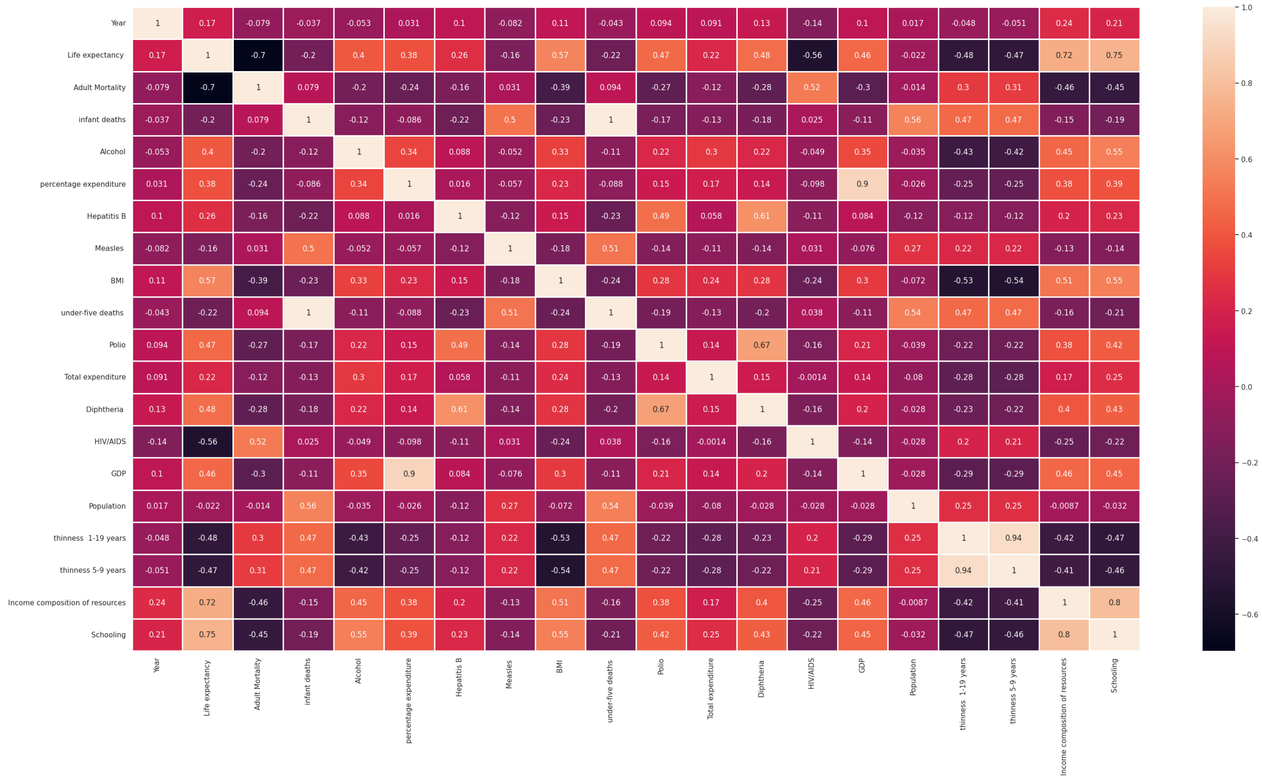
```
Country                            object
Year                                int64
Status                             object
Life expectancy                   float64
Adult Mortality                   float64
```

```
infant deaths                        int64
Alcohol                            float64
percentage expenditure             float64
Hepatitis B                        float64
Measles                              int64
 BMI                               float64
under-five deaths                    int64
Polio                              float64
Total expenditure                  float64
Diphtheria                         float64
 HIV/AIDS                          float64
GDP                                float64
Population                         float64
 thinness  1-19 years              float64
 thinness 5-9 years                float64
Income composition of resources    float64
Schooling                          float64
dtype: object
```

```
data2.corr()['Life expectancy ']#["Life expectancy "]
# plt.show()
```

```
Year                               0.170033
Life expectancy                    1.000000
Adult Mortality                   -0.696359
infant deaths                     -0.196557
Alcohol                            0.404877
percentage expenditure             0.381864
Hepatitis B                        0.256762
Measles                           -0.157586
 BMI                               0.567694
under-five deaths                 -0.222529
Polio                              0.465556
Total expenditure                  0.218086
Diphtheria                         0.479495
 HIV/AIDS                         -0.556556
GDP                                0.461455
Population                        -0.021538
 thinness  1-19 years             -0.477183
 thinness 5-9 years               -0.471584
Income composition of resources    0.724776
Schooling                          0.751975
Name: Life expectancy , dtype: float64
```

```
sn.set(rc={'figure.figsize':(35,18)})
mn = sn.heatmap(data=data2.corr(),linewidths=.75,annot=True)
plt.show()
```

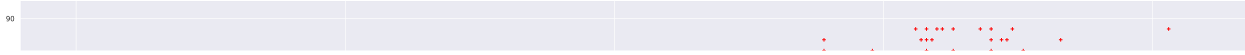


Here, "Schooling" is the most co-related feature with the output feature "Life expectancy". Hence it is selected for the univariate linear regression.

Boxplot and scatterplot for most related feature

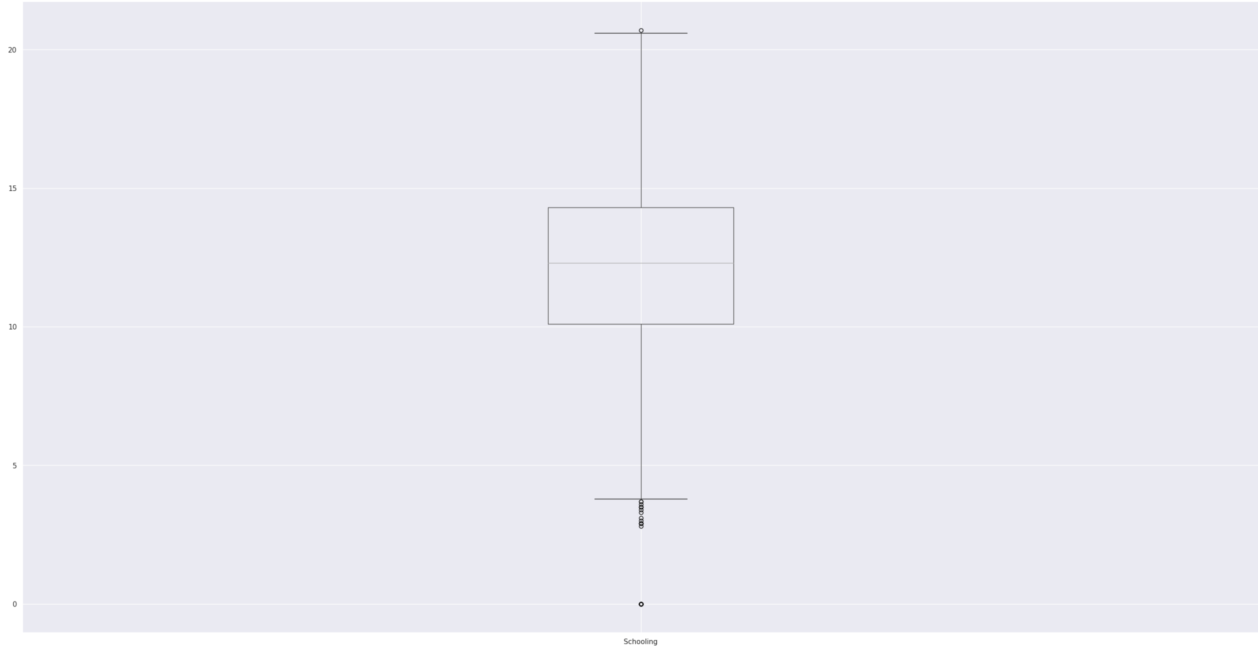
```
%matplotlib inline
plt.xlabel("Schooling")
plt.ylabel("Life expectancy ")
plt.scatter(data2["Schooling"],data2["Life expectancy "],color='red',marker='+')
```

<matplotlib.collections.PathCollection at 0x7fc954cc5930>




```
data2.boxplot(column='Schooling')
```

<AxesSubplot:>



Checking number of missing values in each column

```
data2[data2.columns].isna().sum()
```

```
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
```

Data preprocessing

Data preprocessing is done to convert raw data set into the dataset that is suitable for machine learning model. It includes various steps such as encoding, removing/replacing null values with mean/median/mode/default values, feature selection, etc. Encoding categorical data into numbers using replace for the column of "Status". For categorical data to make it comparable with other data, mapping from labels to numbers is done.

```
data2_processed=data2.copy()
# print(np.array(data2_processed.columns))
status_mapping={'Developing':0,'Developed':1}
data2_processed["Status"]=data2["Status"].replace(status_mapping,inplace=False)
data2_processed=data2_processed.copy()
status_mapping={'2000':0,'2001':1,'2002':2,'2003':3,'2004':4,'2005':5,'2006':6,'20
data2_processed["Year"]=data2["Year"].replace(status_mapping,inplace=False)
```

Columns containing missing values are replaced with the mean of the feature values to

```
print(data2.columns)
data2_processed=data2_processed.copy()
```

```
data2_processed['Life expectancy '].fillna(int(data2_processed['Life expectancy ']
data2_processed['Adult Mortality'].fillna(int(data2_processed['Adult Mortality'].me
data2_processed['Alcohol'].fillna(int(data2_processed['Alcohol'].mean()), inplace=T
data2_processed['Hepatitis B'].fillna(int(data2_processed['Hepatitis B'].mean()), i
data2_processed[' BMI '].fillna(int(data2_processed[' BMI '].mean()), inplace=True
data2_processed['Polio'].fillna(int(data2_processed['Polio'].mean()), inplace=True
data2_processed['Total expenditure'].fillna(int(data2_processed['Total expenditure
data2_processed['Diphtheria '].fillna(int(data2_processed['Diphtheria '].mean()), i
data2_processed['GDP'].fillna(int(data2_processed['GDP'].mean()), inplace=True)
data2_processed[' thinness  1-19 years'].fillna(int(data2_processed[' thinness  1-1
data2_processed[' thinness 5-9 years'].fillna(int(data2_processed[' thinness 5-9 ye
data2_processed['Income composition of resources'].fillna(int(data2_processed['Inco
data2_processed['Schooling'].fillna(int(data2_processed['Schooling'].mean()), inpla
data2_processed['Population'].fillna(int(data2_processed['Population'].mean()), inp
data2_processed[data2_processed.columns].isna().sum()
```

```
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')
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
```

```
data2_processed.head()
```

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percenta expenditu
0	Afghanistan	2015	0	65.0	263.0	62	0.01	71.279
1	Afghanistan	2014	0	59.9	271.0	64	0.01	73.523
2	Afghanistan	2013	0	59.9	268.0	66	0.01	73.219

One hot encoding

Get.dummies function from pandas used to encode the values of column "Country".

```
categorical_columns1=['Country']
data2_processed_onehotencoded = pd.get_dummies(data2_processed,categorical_columns
# print(data2_processed_onehotencoded.head())
print(data2_processed_onehotencoded.columns)
```

```
Index(['Year', 'Status', 'Life expectancy ', 'Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles ', ' BMI ',
       ...
       'Country_United Republic of Tanzania',
       'Country_United States of America', 'Country_Uruguay',
       'Country_Uzbekistan', 'Country_Vanuatu',
       'Country_Venezuela (Bolivarian Republic of)', 'Country_Viet Nam',
       'Country_Yemen', 'Country_Zambia', 'Country_Zimbabwe'],
      dtype='object', length=214)
```

```
data2_processed_onehotencoded.corr()['Life expectancy ']#["Life expectancy "]
# plt.show()
```

```
Year                                          0.169527
Status                                        0.481999
Life expectancy                               1.000000
Adult Mortality                              -0.696358
infant deaths                                -0.196514
                                                ...
Country_Venezuela (Bolivarian Republic of)    0.032409
Country_Viet Nam                              0.043209
Country_Yemen                                -0.041737
Country_Zambia                               -0.119240
Country_Zimbabwe                             -0.145852
Name: Life expectancy , Length: 214, dtype: float64
```

Heatmap

Functions definitions

```
def line_function(X,W,b):
    val=np.dot(W,X.T)+b
    return val

def cost_fn_linear_regression(X,y,W,b):
    n=len(y)
    cost=(1/(2*n))*sum(np.square(np.subtract(line_function(X,W,b),y)))
    return cost

def gradient_uni(X,y,learningRate,N):
    W=0
    b=0
    costs=[]
    m=X.shape[0]
    for i in range(N):
        y_pred=W*X+b
        costs.append(cost_fn_linear_regression(X,y,W,b))
        dW=-2/n*(np.sum(X*(y-y_pred)))
        db=-2/n*(np.sum((y-y_pred)))
        W=W-(learningRate*dW)
        b=b-(learningRate*db)
    return W,b,costs



def cost_function(X,y,W,n):
    a=(np.matmul(X,W.T))-y
    cost=1/(2*n)*np.matmul(a.T,a)
#     print(cost)
    return cost

def gradient_multi(train_set_X,y,learningRate,N):
    W = np.zeros(train_set_X.shape[1])
    n=train_set_X.shape[0]
    costs=[]
    for i in range(N):
        cost=cost_function(train_set_X,y,W,n)
        costs.append(cost)
        ans=(np.matmul(train_set_X,W.T))-y
        W =W-(learningRate*(1/n)*np.matmul(train_set_X.T,ans))
    return W,costs

def train_closed_uni(train_set_X,n):
    W=0
    b=0
    sum_x=sum(train_set["Schooling"])
    sum_y=sum(train_set["Life expectancy "])
    sum_xy=sum(train_set["Life expectancy "]*train_set["Schooling"])
    sum_xsquare=sum(train_set["Schooling"]**2)
    A=[[n,sum_x],[sum_x, sum_xsquare]]
    B=[[sum_y],[sum_xy]]
    res=[[0],[0]]
    C=np.linalg.inv(A)# find inverse
    for i in range(len(C)):
        for j in range(len(B[0])):
```

```
            for k in range(len(B)):
                res[i][j]=res[i][j]+(C[i][k]*B[k][j])
    b=res[0][0]
    W=res[1][0]
    return W,b

def train_closed_multi(train_set_X):
    X_train= np.c_[np.ones((train_set_X.shape[0],1)),train_set_X]
    X_test = np.c_[np.ones((test_set_X.shape[0],1)),test_set_X]
#     print(X_train)
#     print(X_test)
    W=(np.linalg.inv(X_train.T @ X_train) @ (X_train.T @ train_set_y))
    return X_test @ W

def predict(val,W,b):
    return W*val+b

def predict_val(x):
    y_pred=[]
    for i in x :
        y_pred.append(predict(i))
    return y_pred

def mean_squared_error(y_pred,test_set_y):
    return np.square(np.subtract(test_set_y,y_pred)).mean()

def mean_absolute_error(y_pred,test_set_y):
    return np.abs(y_pred-test_set_y).mean()

def standardize(df,col_name):
    df[col_name]=(df[col_name]-df[col_name].mean())/df[col_name].std()

def normalize(df,col_name):
    df[col_name]=(df[col_name]-df[col_name].min())/(df[col_name].max()-df[col_name
```

Data Standardization

```
data2standardized=data2_processed_onehotencoded.copy()
for column in data2standardized.columns:
#     if column!='Life expectancy ':
#         standardize(data2standardized,column)
    normalize(data2standardized,column)
display(data2standardized)
```

	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure
0	1.000000	0.0	0.544592	0.362881	0.034444	0.000000	0.003659
1	0.933333	0.0	0.447818	0.373961	0.035556	0.000000	0.003774
2	0.866667	0.0	0.447818	0.369806	0.036667	0.000000	0.003759
3	0.800000	0.0	0.440228	0.375346	0.038333	0.000000	0.004014
4	0.733333	0.0	0.434535	0.379501	0.039444	0.000000	0.000364
...	...	...	...	...	...	...	...
2933	0.266667	0.0	0.151803	1.000000	0.015000	0.243561	0.000000
2934	0.200000	0.0	0.155598	0.988920	0.014444	0.226764	0.000000

Train-test split

2937	0.000000	0.0	0.184061	0.019668	0.013333	0.093505	0.000000

```
data2standardized=data2standardized.iloc[:,2:21]
cols=data2standardized.columns
train_size = int(0.8 * len(data2standardized))
train_set =data2standardized[:train_size]
test_set = data2standardized[train_size:]
X_train_uni=np.array(train_set['Schooling'].tolist())
X_test_uni=np.array(test_set['Schooling'].tolist())
X_check=train_set
y_check=train_set['Life expectancy ']
train_set_y=np.array(train_set['Life expectancy '].tolist())
train_set_X=train_set.drop(columns=['Life expectancy ']).to_numpy()
test_set_y=np.array(test_set['Life expectancy '].tolist())
test_set_X=test_set.drop(columns=['Life expectancy ']).to_numpy()
print(train_set_X.shape)
print(train_set_y.shape)
print(test_set_X.shape)
print(test_set_y.shape)
```

```
(2350, 18)
(2350,)
(588, 18)
(588,)
```

Multivariate Logistic Regression

Train using Closed form

```
 y_pred=np.array(train_closed_multi(train_set_X))
#  print(y_pred)
```

Checking Accuracy

Mean Squared Error

```
print(mean_squared_error(y_pred,test_set_y))
```

```
0.006522631386155024
```

Mean Absolute Error

```
print(mean_absolute_error(y_pred,test_set_y))
```

```
0.05851788721971853
```

Train using Gradient Descent

↳ 3 cells hidden

Mean Squared Error

```
print(mean_squared_error(y_pred,test_set_y))
```

```
0.021292422056380968
```

Mean Absolute Error

```
print(mean_absolute_error(y_pred,test_set_y))
```

```
0.10722253665327912
```

Checking with built-in functions

```
LR = LinearRegression()
# X=pd.DataFrame(train_set_X,cols)
# y=pd.DataFrame(train_set_y,'Life expectancy ')
X_sample=X_check.sample(frac=1,random_state=11)
Y_sample=y_check.sample(frac=1,random_state=11)
train_size = int(0.8*len(train_set_X))
```

```
X_train = X_sample[:train_size]
Y_train = Y_sample[:train_size]
X_test = X_sample[train_size:]
Y_test = Y_sample[train_size:]
LR.fit(X_train,Y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred=LR.predict(X_test)
mse = np.square(np.subtract(Y_test,y_pred)).mean()
print(mse)
```

```
2.3562318666427754e-32
```

Univariate Linear Regression

Train using Closed Form

```
n=len(train_set_X)
W,b=train_closed_uni(train_set_X,n)
y_pred=predict(test_set_y,W,b)
```

Mean Squared Error

```
print(mean_squared_error(y_pred,test_set_y))
```

```
0.003066501706701539
```

Mean Absolute Error

```
print(mean_absolute_error(y_pred,test_set_y))
```

```
0.04608357459707135
```

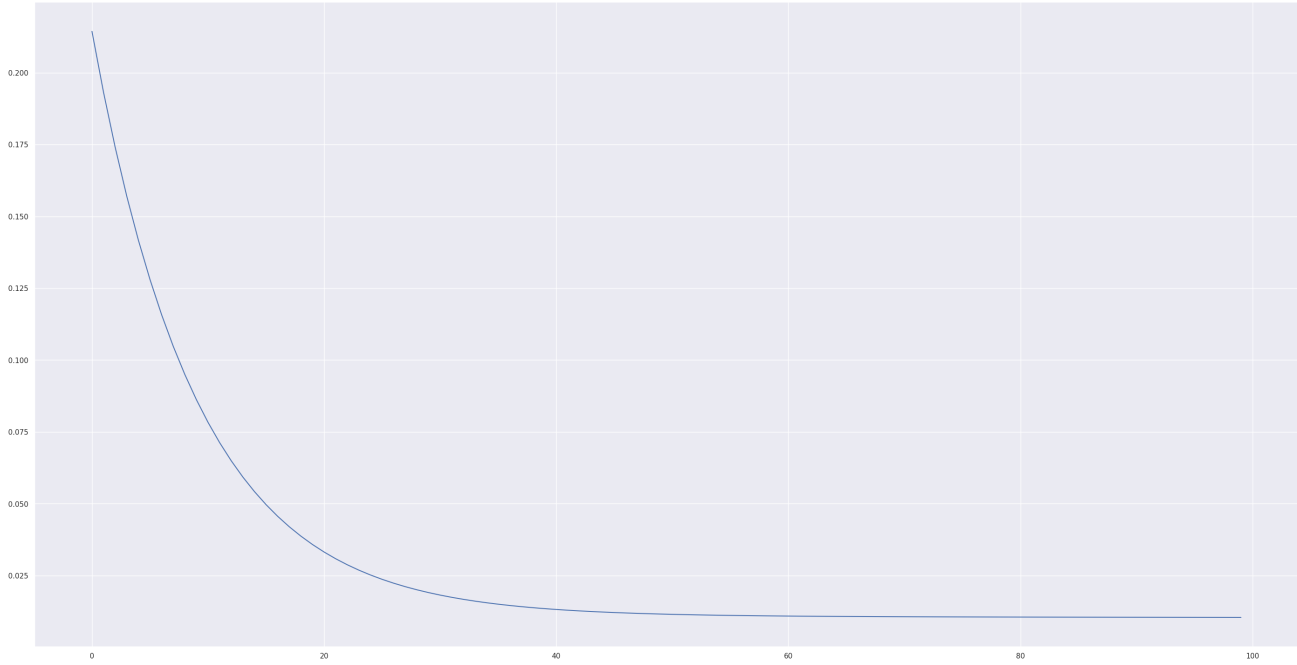
Train using Gradient Descent

```
W,b,costs=gradient_uni(X_train_uni,train_set_y,learningRate=0.02 ,N=100)
print(W,b)
```

```
0.3201326828027143 0.4474045272420688
```

```
plt.plot(range(len(costs)),costs)
```

```
[<matplotlib.lines.Line2D at 0x7fc94fa54be0>]
```



Predict on test data

```
# print(X_test_uni.shape)
y_pred=predict(X_test_uni,W,b)
# print(y_pred.shape)
```

Mean Squared Error

```
print(mean_squared_error(y_pred,test_set_y))
```

```
0.026812909075378186
```

Mean Absolute Error

```
print(mean_absolute_error(y_pred,test_set_y))
```

```
0.12587476249854784
```