# ASSIGNMENT # 2

**TASK-**PERFORM LINEAR OR POLYNOMIAL REGRESSION IS UP TO YOU ON THE GIVEN DATASET AND PREDICT BRAIN WEIGHT FROM HEAD SIZE.

## CODE

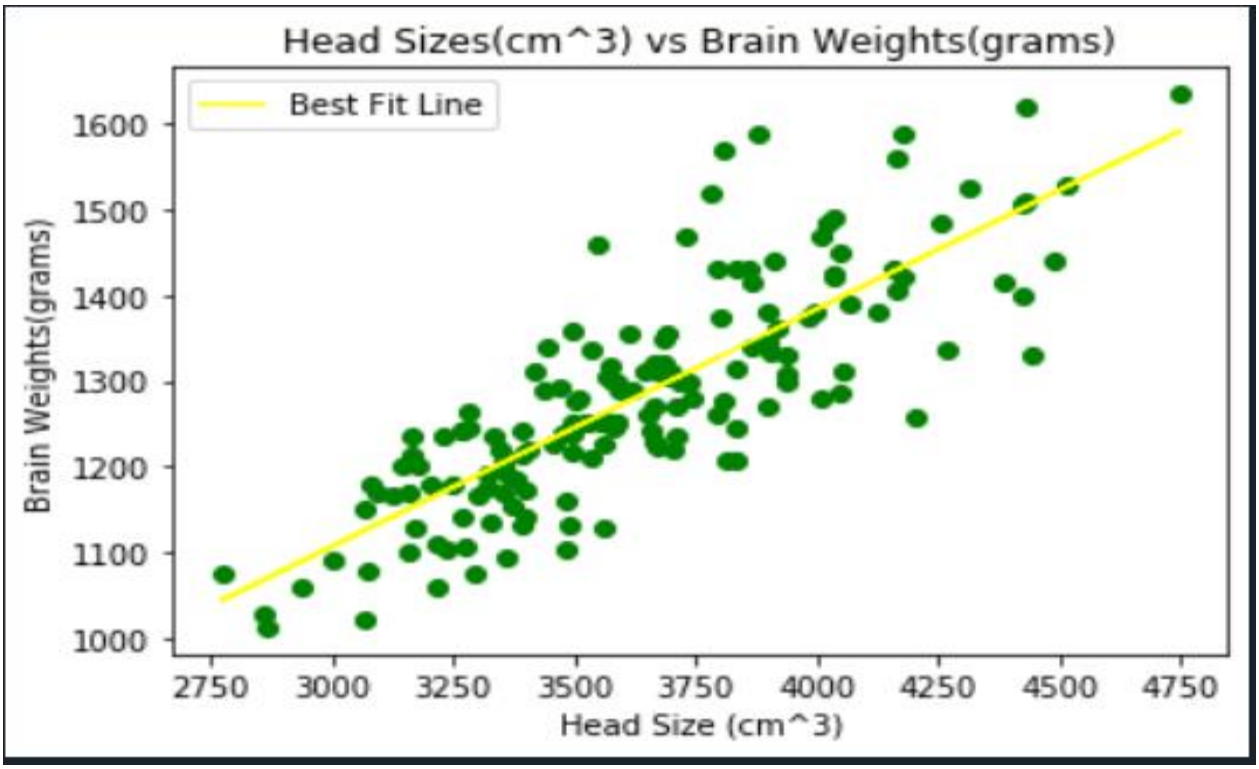
```
#Ayesha Nadeem Akhter
#Performing Linear Regression on the given dataset(Assignment-2)

#Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#Importing the dataset
data = pd.read_csv('dataset.csv')
HdSiz = data.iloc[:,2:3].values
BrnWt = data.iloc[:,3].values

#Partitioning the dataset into training and test dataset
from sklearn.model_selection import train_test_split
HdSiz_train, HdSiz_test, BrnWt_train, BrnWt_test =train_test_split(HdSiz,BrnWt,test_size=1/3,random_state = 22)

#Fitting the Simple Linear Regression to the Training dataset
from sklearn.linear_model import LinearRegression
RegVar = LinearRegression()
RegVar.fit(HdSiz_train, BrnWt_train)

#Predicting Values. Using the test dataset of HdSiz
PredVar = RegVar.predict(HdSiz_test)

#Visualizing the training dataset results
plt.scatter(HdSiz_train, BrnWt_train, color = 'green')
plt.plot(HdSiz_train, RegVar.predict(HdSiz_train), color = 'yellow', label = 'Best Fit Line')
plt.title('Head Sizes(cm^3) vs Brain Weights(grams)')
plt.xlabel('Head Size (cm^3)')
plt.ylabel('Brain Weights(grams)')
plt.legend()
plt.show()
```

```
#Visualizing the test dataset results
plt.scatter(HdSiz_test, BrnWt_test, color = 'red')
plt.plot(HdSiz_train, RegVar.predict(HdSiz_train), color = 'pink', label = 'Best Fit Line')
plt.title('Head Size (cm^3) vs Brain Weights (grams)')
plt.xlabel('Head Size (cm^3)')
plt.ylabel('Brain Weights (grams)')
plt.legend()
plt.show()
```

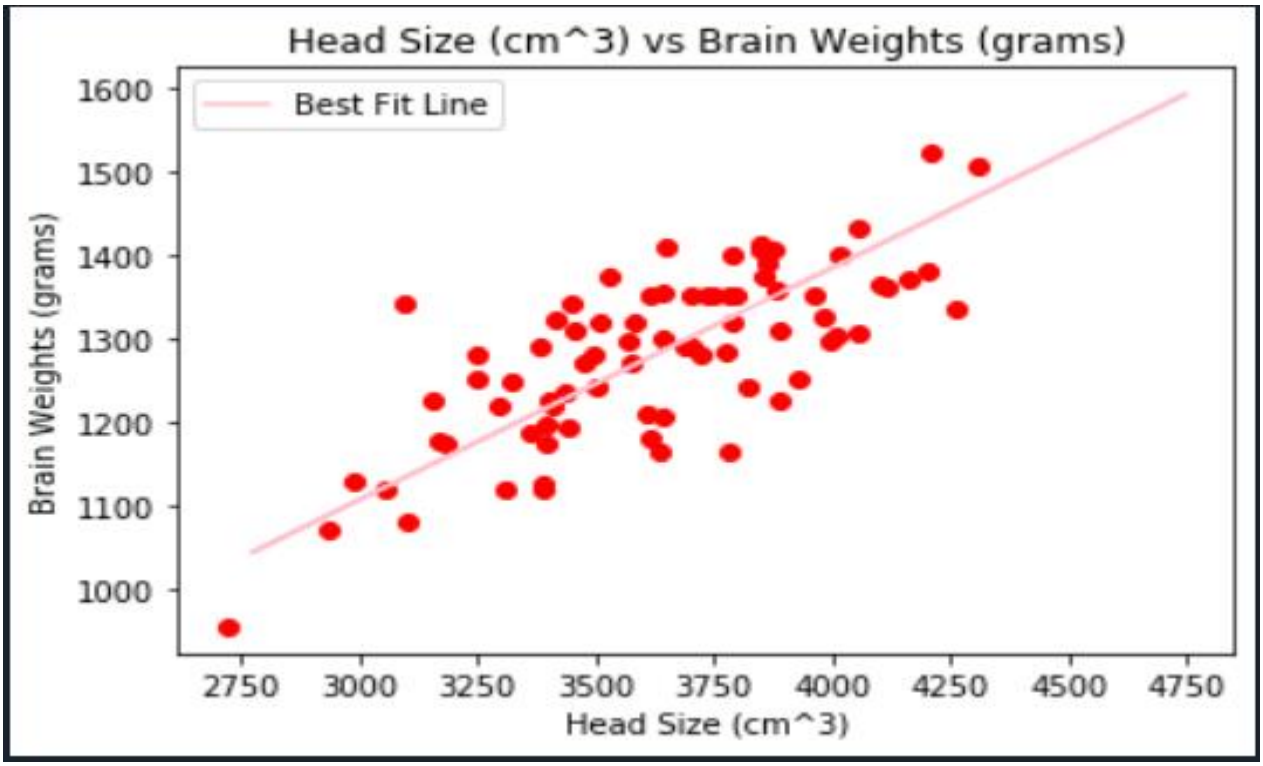
## VARIABLE EXPLORER

| Name | Type | Size | Value |
|---|---|---|---|
| BrnWt | Array of int64 | (237,) | [1530 1297 1335 ... 1104 1170 1120] |
| BrnWt_test | Array of int64 | (79,) | [1412 1506 1306 ... 1400 1240 1210] |
| BrnWt_train | Array of int64 | (158,) | [1432 1075 1237 ... 1340 1450 1270] |
| HdSiz | Array of int64 | (237, 1) | [[4512]<br>[3738]] |
| HdSiz_test | Array of int64 | (79, 1) | [[3850]<br>[4308]] |
| HdSiz_train | Array of int64 | (158, 1) | [[3796]<br>[3292]] |
| PredVar | Array of float64 | (79,) | [1342.48100825 1469.73276261 1398.32730219 ...<br>1324.42126145 1335.2571 ... |
| RegVar | linear_model._base.LinearRegression | 1 | LinearRegression object of sklearn.linear_model._base module |
| data | DataFrame | (237, 4) | Column names: Gender, Age Range, Head Size(cm^3), Brain Weight(grams) |

## TRAINING DATASET PLOT



## TEST DATASET PLOT



# CLASS-2 (MARCH 15, 2020) --- BY SIR HAMZA KHALID

**CLASS EXERCISE-** CREATE TWO RANDOM ARRAYS A AND B, AND MULTIPLY THEM. GET THEIR RESULT IN C AND ADD 1 TO EVERY ELEMENT OF C.

## CODE AND OUTPUT

```
import numpy as np

#Creating Random Arrays
A = np.random.randn(3,4)
B = np.random.randn(3,4)
print('Array A')
print('=======')
print(A)
print('\n')
print('Array B')
print('=======')
print(B)
print('\n')

#Multiplying the two arrays A and B
print ('Multiplication Of Arrays A and B')
print('==================================')
C = np.multiply(A,B)
print('\n')
print('The Resultant Array C')
print('=====================')
print(C)
print('\n')

#Defining an Array containing 1s
ones = np.ones((3,4))
#Adding Array 'ones' to Array C
res = np.add(ones,C)
print('Final Answer')
print('============')
print(res)
```

```
Array A
=======
[[ 0.6247778  -1.85747887 -0.30468914 -0.96706103]
 [ 0.04663542  0.1247227  -0.03883216 -1.13455803]
 [-0.46946509  2.00660126 -0.11383333 -0.70561045]]


Array B
=======
[[ 0.24606944 -0.65514671  1.79555821 -0.20349241]
 [ 0.27445076  1.43805898 -0.74712197 -0.69372648]
 [ 1.575412   -0.04985226 -0.84930071  1.20564998]]


Multiplication Of Arrays A and B
==================================


The Resultant Array C
=====================
[[ 0.15373872  1.21692116 -0.54708708  0.19678958]
 [ 0.01279913  0.1793586   0.02901236  0.78707294]
 [-0.73960094 -0.10003361  0.09667873 -0.85071922]]


Final Answer
============
[[1.15373872 2.21692116 0.45291292 1.19678958]
 [1.01279913 1.1793586  1.02901236 1.78707294]
 [0.26039906 0.89996639 1.09667873 0.14928078]]
```