Tyler Smith

826005315

ECEN 489 - Applied Data Science

Homework 2

Libraries =>

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
sns.set()
```

We are given unit variance (1) and zero mean Below the irreducible error is calculated

```
#Initializing the random error
mean, std = 0,1
#Number of values being tested
count = 3
#Creating the random error distribution
E = np.random.normal(mean, std, count)
#Values of the error at given instances
print('Irreducible Error at the 3 points',E)
```

```
Irreducible Error at the 3 points [ 4.42145873e-01 -1.34622794e+00 -1.2896579(
```

1) - A

Y = f(X) + E is the general form of a quantitive response for Y therefore the regression function f(X) = 1 + X, given that the model of Y = 1 + X + E.

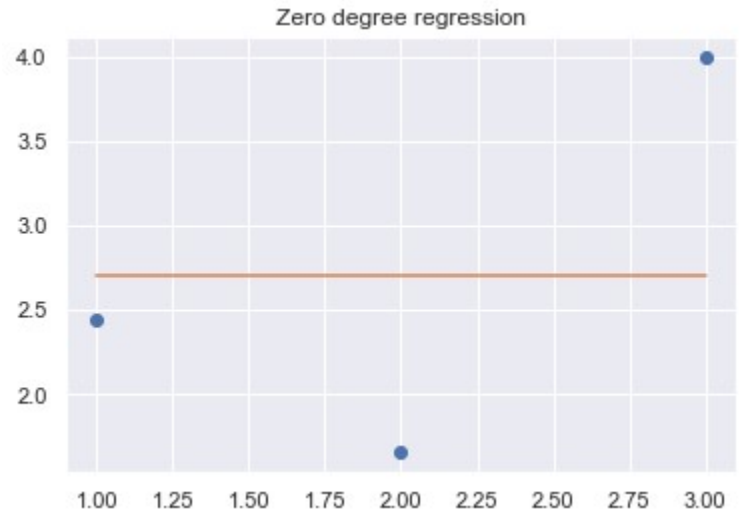
```
#creating x = 1,2,..count
X = np.arange(1,count+1,1,dtype=float)
#initializing Y
Y = np.zeros(count,dtype=float)
print('X = ',X)
```

1) - B

Below I will show fitting the data to a 0th degree regression

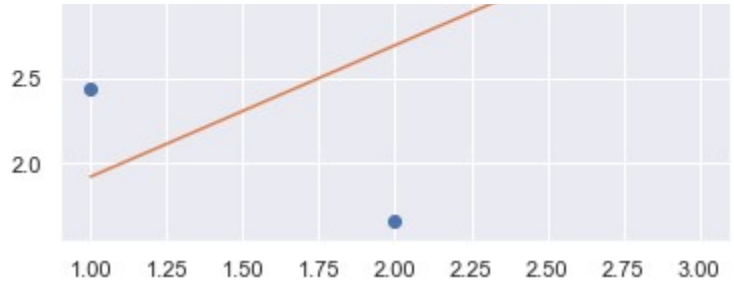
```
#This funtion produces a polynomial to fit to 0th degree
B_0_zero_degree = np.polyfit(X,Y,0)
#Plotting the scatterplot of the original data
plt.plot(X,Y,'o')
#Plotting the 0th degree regression
plt.plot(X,X*0 + B_0_zero_degree)
plt.title('Zero degree regression')
```

```
Text(0.5, 1.0, 'Zero degree regression')
```



Below I will show fitting the data to a 1st degree regression

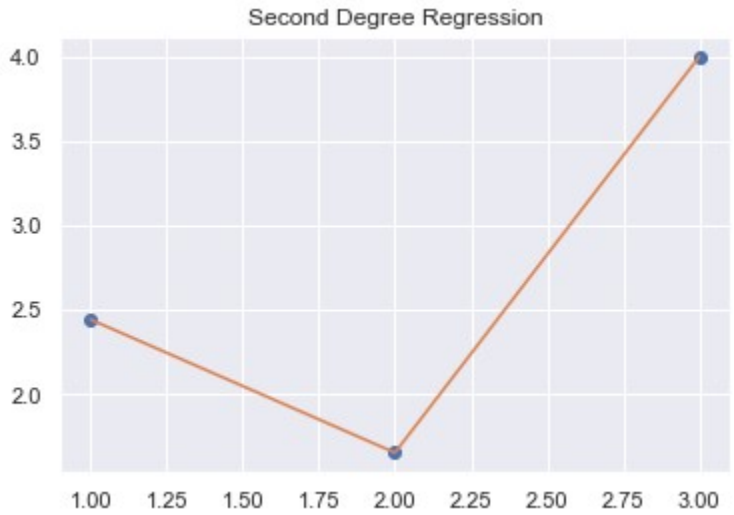
```
#generating the coefficents of the first order regresison
[B_1_1st,B_0_1st] = np.polyfit(X,Y,1)
#printing the coeggicents
print('B_0 first degree = ', B_0_1st )
print('B_1 first degree = ', B_1_1st)
#plotting the original data
plt.plot(X,Y,'o')
#potting the regression
```



Below I will shot fitting the data to a 2nd degree regression

```
#Generating the second degree regression
[B_2_2nd, B_1_2nd, B_0_2nd] = np.polyfit(X,Y,2)
#Printing the coefficents
print('B_0 second degree = ', B_0_2nd)
print('B_1 second degree = ', B_1_2nd)
print('B_2 second degree = ', B_2_2nd)
#Plotting the original data
plt.plot(X,Y,'o')
#Ploting the regression
plt.plot(X,((X**2) * B_2_2nd + X * B_1_2nd + B_0_2nd))
plt.title('Second Degree Regression')
```

```
B_0 second degree =  6.363831779915271
B_1 second degree =  -5.488341954672097
B_2 second degree =  1.5666560473555577
Text(0.5, 1.0, 'Second Degree Regression')
```



```
Y predicted =  2.442145872598732
Y real 2.4421458725987324
Y predicted =  1.6537720599933081
Y real 1.653772059993311
Y predicted =  3.9987103420990007
Y real 3.998710342099
training error for M2 =  3.106139814307734e-30
```

1) - C

Now creating the 1000 different test cases, This is just the Y and irreducible error vlaues. I will stores vlaues momentarily in arrays while computing them and then place them inside of a dataframe for ease of use later on

```
#This dataframe will store all the computed values for the response variable and tr
test_cases = pd.DataFrame(index = np.arange(0,1000),
                          columns = [
                          'E_1',
                          'E_2',
                          'E_3',
                          'Y_1',
                          'Y_2',
                          'Y_3',
                          'M_0_0th',
                          'M_0_1st',
                          'M_0_2nd',
                          'M_1_1st',
                          'M_1_2nd',
                          'M_2_2nd'
```

Now the constants for the regresisons will be calculated and stored in arrays

```
#initiating arrays to store the constants of the regressions
#0th degree
cnst_0 = np.zeros(set_amount, dtype = float)
#1st degree
cnst_1 = np.zeros((set_amount,2), dtype = float)
#2nd degree
cnst_2 = np.zeros((set_amount,3), dtype = float)
```

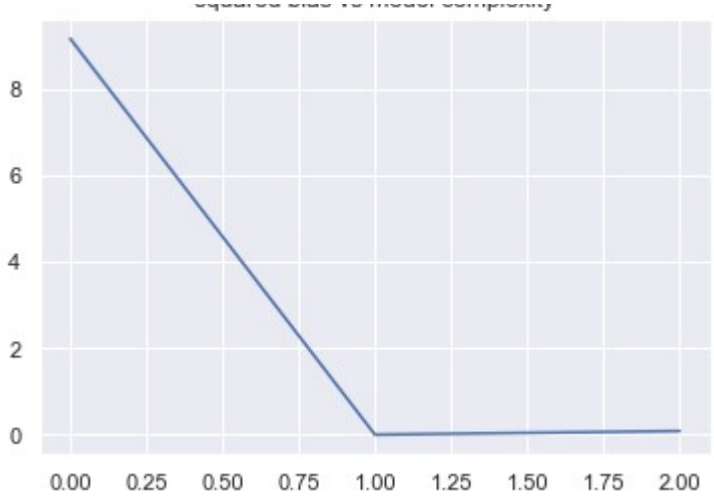
Calculating the regressions

```
#This will run for all training sets
for i in range(0,set_amount):
    #calculating 0th degree constant(s)
    cnst_0[i] = np.polyfit(X,Y_real[i,:],0) #b0_0
    #calculating 1st degree constant(s)
    cnst_1[i,:] = np.polyfit(X,Y_real[i,:],1) #b1_1, b0_1
    #calculating 2nd degree constant(s)
    cnst_2[i,:] = np.polyfit(X,Y_real[i,:],2) #b2_2, b1_2, b0_2
print(cnst 2[1,:])
```

```
        TE_1st[i] = TE_1st[i] + (Y_real[i,n] - (cnst_1[i,0]*X[n] + cnst_1[i,1]))**2
        ######################
        #training error for 2nd
        ######################
        TE_2nd[i] = TE_2nd[i] + (Y_real[i,n] - ((cnst_2[i,0]*(X[n]**2)) + (cnst_2[i
        ######################

    TE_0th[i] = TE_0th[i]/count
    TE_1st[i] = TE_1st[i]/count
    TE_2nd[i] = TE_2nd[i]/count
```

1) - D


8
6
4
2
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

```
for i in range(0,set_amount):
    IE_f5 = np.random.normal(mean, std, 5)
    Y_5 = 6 + IE_f5[4]
    diff_0[i] = (Y_5 - est_5_0[i])**2
    diff_1[i] = (Y_5 - est_5_1[i])**2
    diff_2[i] = (Y_5 - est_5_2[i])**2
```