Use Overfitting To Evaluate Different Models

The process of Machine Learning and using [Overfitting to evaluate Linear Regression Model and Non-linear Regression .]()

- Please compare the following two Regression Models to see which one has more serious overfitting issue.
  - [Linear Regression Model 1]()
  - [Non-Linear Regression Model 2]()
- Suppose we collect a set of sample data and [distribute]() the sample data by
  -
  - `Training phase: 50%`
  - `Validation phase: 25%`
  - `Test phase: 25%`

Training Phase				**Validation Phase**				**Test Phase**	
Real Data Set 1 50% of the collcted data		**[Model 1: Linear Regression]()**	**[Model 2: Non-Linear Regression]()**	**Real Data Set 2 25% of the collcted data**		**[Model 1: Linear Regression]()**	**[Model 2: Non-Linear Regression]()**	**Real Data Set 3 25% of the collcted data**	**The better model ([Model 1]() or [Model 2]()) selected from the Validation Phase based on the analysis of [overfitting]() will be used to calculate $\hat{y}$**
▪ **After calculating a1, b1, a2, b2 in Training Phase, the values are not changed with the new Real Data Sets in Validation Phase and Test Phase.** ▪ **Only $\hat{y}$ values are changed with the new Real Data Sets.**									
x	**y**	**$\hat{y}$=a1 + b1 * x**	**$\hat{y}$=a2 + b2 * x^2**	**x**	**y**	**$\hat{y}$=a1 + b1 * x**	**$\hat{y}$=a2 + b2 * x^2**	**x**	**$\hat{y}$=a1 + b1 * x or $\hat{y}$=a2 + b2 * x^2**
1	**1.8**	**1.37**	1.75	**1.5**	**1.7**	**1.28**	1.41	**1.4**	**1.88**
2	**2.4**	**2.23**	2.15	**2.9**	**2.7**	**2.54**	2.31	**2.5**	**2.46**
3.3	**2.3**	**3.35**	3.08	**3.7**	**2.5**	**3.26**	3.08	**3.6**	**3.36**
4.3	**3.8**	**4.22**	4.10	**4.7**	**2.8**	**3.61**	3.53	**4.5**	**4.34**
5.3	**5.3**	**5.08**	5.39	**5.1**	**5.5**	**4.51**	4.87	**5.4**	**5.53**
1.4	**1.5**	**1.71**	1.88	**X**	**X**	**X**	**X**	**X**	**X**
2.5	**2.2**	**2.66**	2.45	**X**	**X**	**X**	**X**	**X**	**X**
2.8	**3.8**	**2.92**	2.67	**X**	**X**	**X**	**X**	**X**	**X**
4.1	**4.0**	**4.04**	3.88	**X**	**X**	**X**	**X**	**X**	**X**
5.1	**5.4**	**4.91**	5.12	**X**	**X**	**X**	**X**	**X**	**X**

Note:

- Real Data Set 1 can be used to determine the formulas for Model 1: Linear Regression and Model 1: Linear Regression. That is, to determine the valuese of a1, b1, a2, and b2 in the following formulas:
    -
    - ŷ=a1 + b1 * x
    - ŷ=a2 + b2 * x^2

    - After the formulas are determined, you can use the formulas to calculate the ŷ values in the following phases:
        - Training Phase
        - Validation Phase
        - Test Phase
    - Note: The values of "x" in "ŷ=a1 + b1 * x" and "ŷ=a2 + b2 * x^2" are the same as the "x" list on the "Real Data Set".
- Optional: You may want to implement the following 3 programs:
    - Program 1: To implement Linear Regression Model 1
      Note:
        - This program is to use RealData Set 1 to determine a1 and b1 based on Model 1.
        - The program can be used to fill part of the blank spaces in above table.
    - Program 2: Non-Linear Regression Model 2
      Note:
        - This program is to use RealData Set 1 to determine a2 and b2 based on Model 2.
        - The program can be used to fill part of the blank spaces in above table.
    - Program 3: Calculate MSE

Ans:

Training Phase - Liner Regression Model:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1, 2, 3.3, 4.3, 5.3, 1.4, 2.5, 2.8, 4.1, 5.1])
y = np.array([1.8, 2.4, 2.3, 3.8, 5.3, 1.5, 2.2, 3.8, 4.0, 5.4])

n = x.size

ss_xy = np.sum(x*y)
ss_xx = np.sum(x*x)
s_x = np.sum(x)
s_y = np.sum(y)
```

```
b = (n*ss_xy-s_x*s_y)/(n*ss_xx-s_x*s_x)
a = (s_y - b*s_x)/n

print ("Sum of x*y", ss_xy)
print ("Sum of x*x", ss_xx)
print ("Sum of x", s_x)
print ("Sum of y", s_y)
print ("Slope_b", b)
print ("Intercept_a", a)

def plot_regression_line(x, y, a, b):
    # plotting the actual points as scatter plot
    plt.scatter(x, y, color = "m", marker = "o", s = 30)

    # predicted response vector
    y_pred = a + b*x

    # plotting the regression line
    plt.plot(x, y_pred, color = "g")

    # putting labels
    plt.xlabel('x')
    plt.ylabel('y')

    # function to show plot
    plt.show()

def main():
    for i in range(0, n):
      y_bar = a + b * x[i]
      print("Y_Value: ", y_bar)

    # plotting regression line
    plot_regression_line(x, y, a, b)

if __name__ == "__main__":
            main()
```

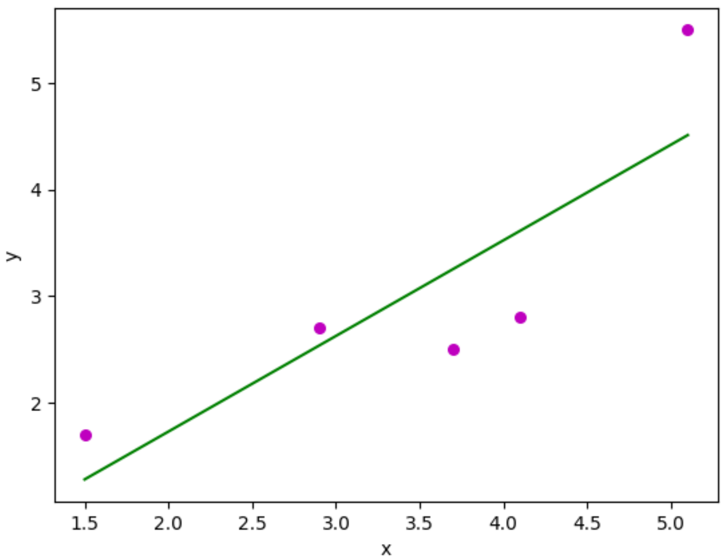
```
D:\Files\UNewFiles\NPU-File\MSCS\2021Summer\CS550\
Sum of x*y 120.79999999999998
Sum of x*x 121.34
Sum of x 31.800000000000004
Sum of y 32.5
Slope_b 0.8631776810447154
Intercept_a 0.5050949742778048
Y_Value:  1.3682726553225202
Y_Value:  2.2314503363672356
Y_Value:  3.3535813217253656
Y_Value:  4.216759002770081
Y_Value:  5.079936683814796
Y_Value:  1.7135437277404062
Y_Value:  2.663039176889593
Y_Value:  2.9219924812030076
Y_Value:  4.044123466561137
Y_Value:  4.907301147605853

Process finished with exit code 0
```

Non-Linear Model:

```
D:\Files\UNewFiles\NPU-File\MSCS\2021Summer\CS550\Assi
Sum of x_bar 121.34
Sum of y 32.5
Sum of x_bar * y 509.76199999999994
Sum of x_bar * x_bar 2329.9862
Slope_b 0.13456241139124608
Intercept_a 1.61721970017862
Y_Value:  1.751782111569866
Y_Value:  2.1554693457436045
Y_Value:  3.08260436022929
Y_Value:  4.105278686802761
Y_Value:  5.397077836158722
Y_Value:  1.8809620265054623
Y_Value:  2.458234771373908
Y_Value:  2.672189005485989
Y_Value:  3.879213835665466
Y_Value:  5.11718802046493

Process finished with exit code 0
```

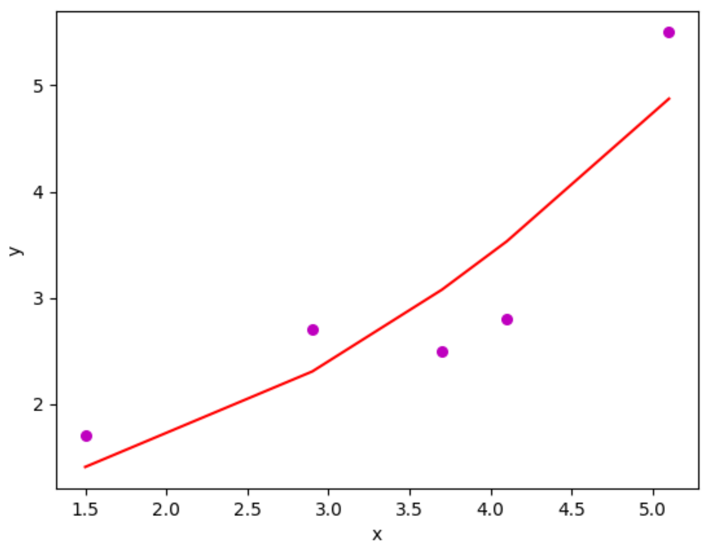
Validation Phase - Liner Regression Model:

```
D:\Files\UNewFiles\NPU-File\MSCS\2021Summe
Sum of x*y 59.16
Sum of x*x 67.16999999999999
Sum of x 17.3
Sum of y 15.2
Slope_b 0.8982494529540502
Intercept_a -0.06794310722101393
Y_Value:  1.2794310722100615
Y_Value:  2.5369803063457317
Y_Value:  3.255579868708972
Y_Value:  3.6148796498905917
Y_Value:  4.513129102844642

Process finished with exit code 0
```



Non-Liner Model:

```
D:\Files\UNewFiles\NPU-File\MSCS\2021Summer\CS
Sum of x_bar 67.16999999999999
Sum of y 15.2
Sum of x_bar * y 250.87999999999997
Sum of x_bar * x_bar 1222.3028999999997
Slope_b 0.14591184777999153
Intercept_a 1.079820236923594
Y_Value:  1.408121894428575
Y_Value:  2.306938876753323
Y_Value:  3.0773534330316785
Y_Value:  3.532598398105251
Y_Value:  4.874987397681173

Process finished with exit code 0
```



MSE – Python Program

y = [11,20,19,17,10]
y_bar = [12,18,19.5,18,9]

```
summation = 0  #variable to store the summation of differences
n = len(y) #finding total number of items in list
for i in range (0,n):  #looping through each element of the list
  difference = y[i] - y_bar[i]  #finding the difference between observed and predicted value
  squared_difference = difference**2  #taking square of the differene
  summation = summation + squared_difference  #taking a sum of all the differences
MSE = summation/n  #dividing summation by total values to obtain average
print ("The Mean Square Error is: ", MSE)
```

Model_1 – Training Phase = 0.28129
Model_2 – Training Phase = 0.2348

Model_1 – Validation Phase = 0.48316
Model_2 – Validation Phase = 0.3

Compare:

Model_1 = 0.48316 / 0.28129 = 1.717

Model_2 = 0.3 / 0.2348 = 1.277

So:

Model_2 is better model