CS 454
Introduction to Machine Learning and Artificial Neural Networks

Homework 1 Report

Polynomial Regression and Model Selection

Author	*Uygar KAYA*
Instructor	*Prof. Ethem ALPAYDIN*
Submission Date	*14.03.2022*

1 Introduction

In this assignment, we implement a Polynomial Regression, and we see how model selection can be done.

In this assignment, we have the one Test data set with 100 instances and 10 Training data set files each containing 25 instances. In each data set, the first column in each row is the Input - X and the second column is the desired Output - R.

1.1 Polynomial Regression

Polynomial Regression is a form of Linear regression known as a special case of Multiple linear regression which estimates the relationship as an nth degree polynomial. Polynomial Regression is sensitive to outliers so the presence of one or two outliers can also badly affect the performance.

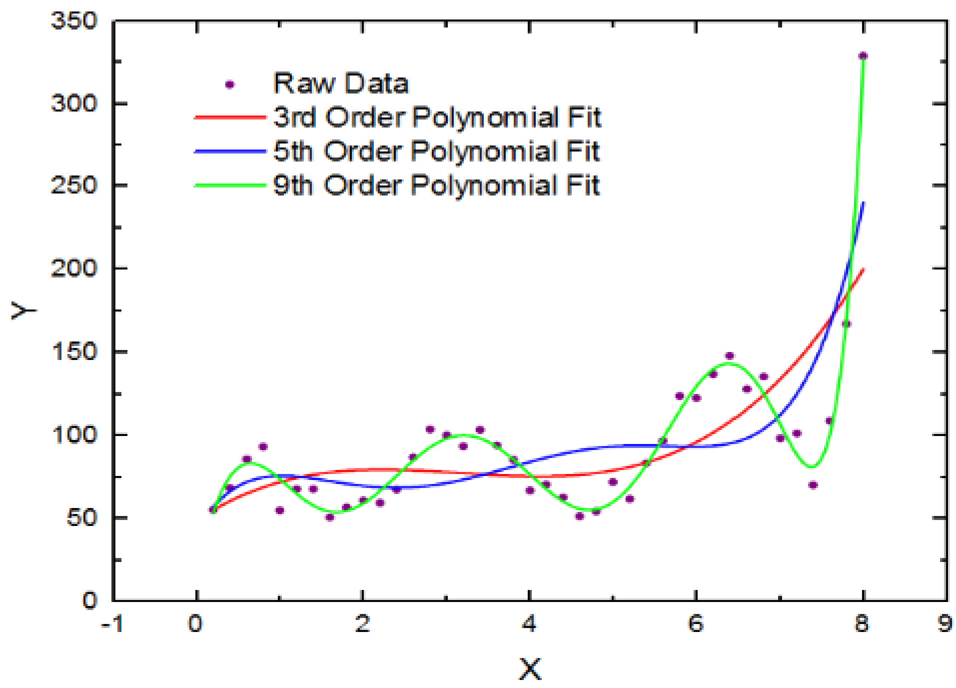


Figure 1: Example of Polynomial Regression [1]

2 Methodology

In this assignment, we apply a Polynomial Regression model methodology using Python Programming Language.

During this assignment, using only 5 different Python programming language libraries these are;

1. *NumPy*
2. *Pandas*
3. *Scikit-Learn*
4. *Matplotlib were used.*

NumPy library is used to polyfit function to find the polynomials and create the polynomial regression model. Pandas library is used to make a DataFrame. I used to Scikit-Learn library in order to use the mean_square_error function. Finally, I used the Matplotlib in order to plot and visualized the result data.

3 Implementation Details

In this assignment, firstly we import the necessary libraries which are Pandas, NumPy, Matplotlib and Scikit-Learn. After importing these libraries, we read the csv file with the help of the pandas library in Python and we have 10 sample training data .csv file & 1 testing data .csv file and after reading the 10 sample training data, I append these data to the list.

Importing Required Libraries

```python
import pandas as pd
import numpy as np

from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error
```

Reading Train & Test Files

```python
myTrainList = ['sample1.csv', 'sample2.csv', 'sample3.csv', 'sample4.csv', 'sample5.csv', 'sample6.csv', 'sample7.csv', 'sample8.csv', 'sample9.csv', 'sample10
trainDataFrame = []

for trainData in myTrainList:
    data = pd.read_csv(trainData, names=['X', 'R'],  header=None)
    trainDataFrame.append(data)
```

```python
testDataFrame = pd.read_csv('test.csv', names=['X', 'R'], header=None)
```

Figure 2: Libraries and Reading Training & Test Files

After reading the training & testing data, I separate the Input – X & Output – R list for each testing and training data. I separate the input and output data in order to use these values in the calculation for prediction and etc.

Seperate Input - X & Output - R

```python
train_x = []
train_r = []

for index in range(10):
    train_x.append(trainDataFrame[index]['X'].values)
    train_r.append(trainDataFrame[index]['R'].values)
```

```python
test_x = testDataFrame['X'].values
test_r = testDataFrame['R'].values
```

Figure 3: Separate the Data to Input & Output Data

After this process, I plot the whole training data and testing data in order to see the visualize distribution for the data.

Ploting the Whole Training & Testing Data

```python
plt.title('Training Data', fontweight="bold", style="italic")
plt.xlabel("Input - X", fontweight="bold", style="italic")
plt.ylabel("Output - R", fontweight="bold", style="italic")
plt.scatter(train_x, train_r, color = 'g')
plt.show()

plt.title('Testing Data', fontweight="bold", style="italic")
plt.xlabel("Input - X", fontweight="bold", style="italic")
plt.ylabel("Output - R", fontweight="bold", style="italic")
plt.scatter(test_x, test_r, color = 'b')
plt.show()
```

Figure 4: Plotting the Existing Whole Data

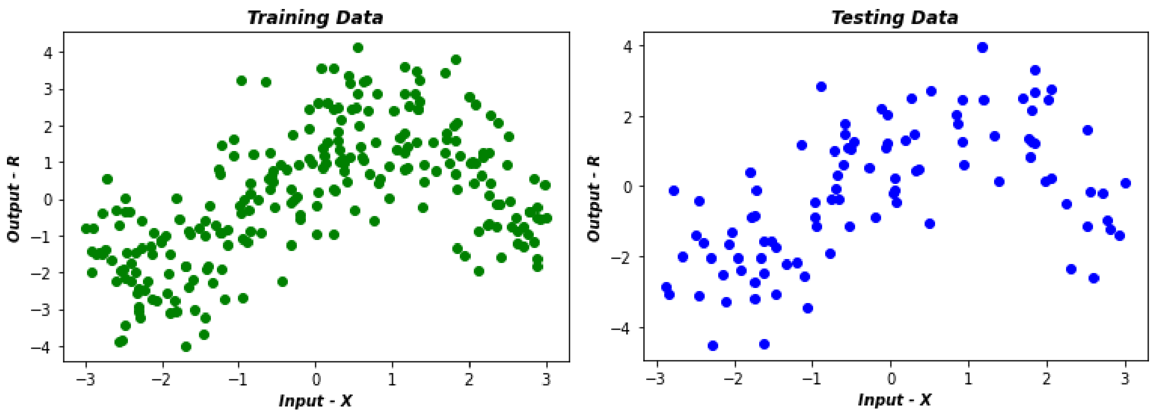


Figure 5: Visualizing Whole Training & Testing Data

3.1 Applying Polynomial Regression Model Strategy

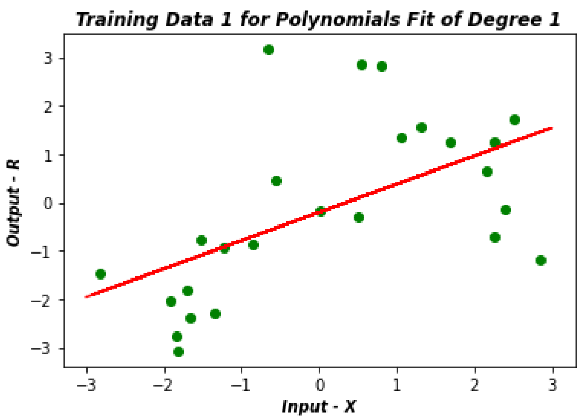
3.1.1 Polynomials Fit of Degree 1

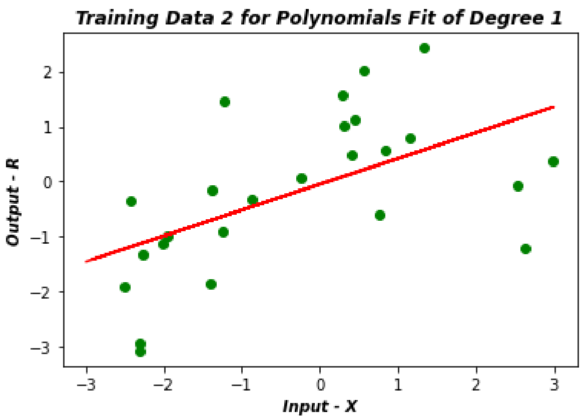
In this part, I fit polynomials of degree 1, 2, 3, 4, 5 and 6 to the ten data sets separately with using polyfit. For each degree, I have ten fits for the ten training sets. I plot these ten fits for each degree, and I calculate their mean square error values on the test set for each degree.

Polynomials Fit of Degree 1

```python
polyFit1 = []

for index in range(10):
    fit = np.polyfit(train_x[index], train_r[index], 1)
    polyFit1.append(fit)
```

```python
for index in range(10):
    plt.title('Training Data {} for Polynomials Fit of Degree 1'.format(index+1), fontweight="bold", style="italic")
    plt.xlabel("Input - X", fontweight="bold", style="italic")
    plt.ylabel("Output - R", fontweight="bold", style="italic")
    plt.scatter(train_x[index], train_r[index], color = 'g')
    plt.plot(train_x, np.poly1d(polyFit1[index])(train_x), color = 'r')
    plt.show()
```





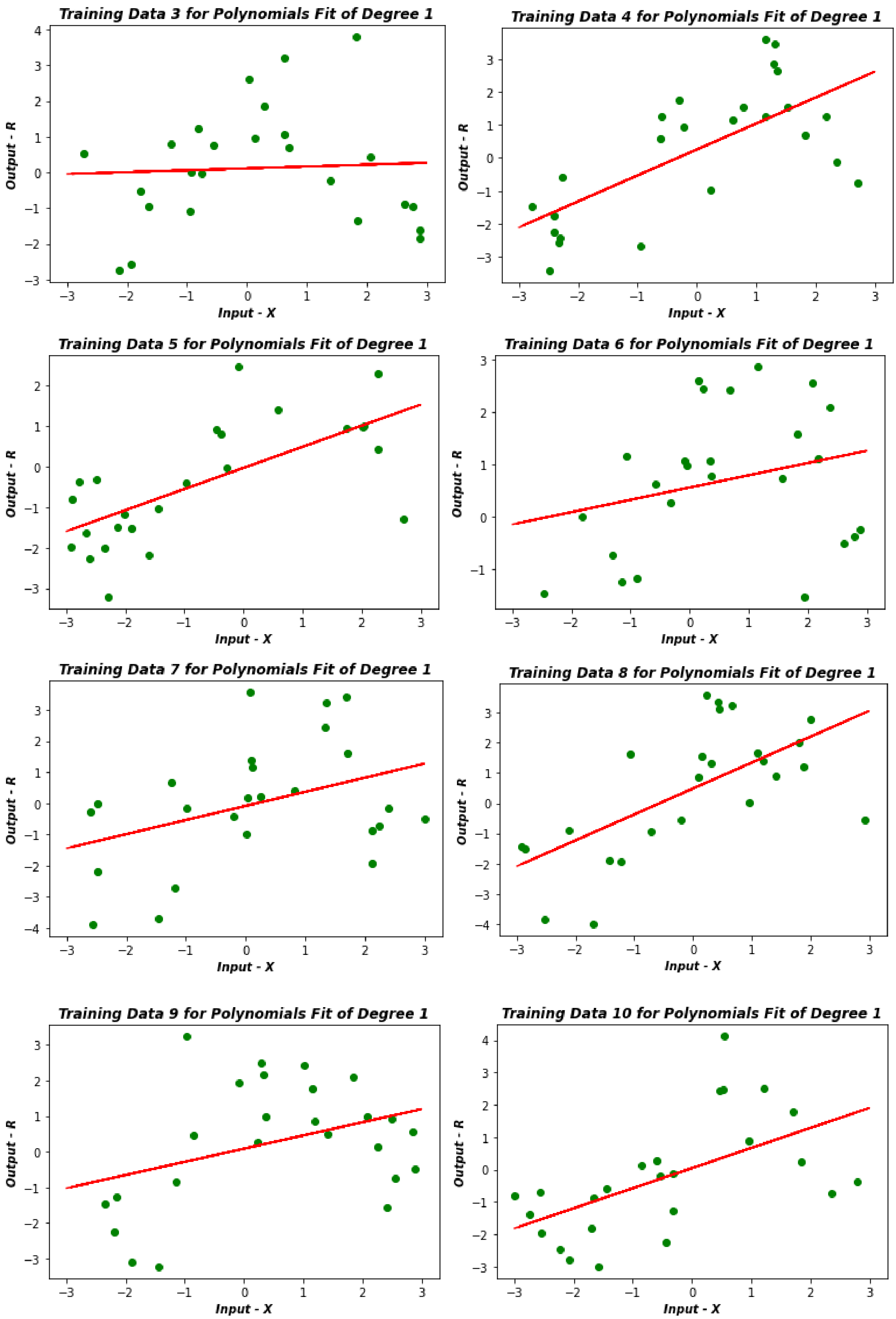


Figure 6: Plotting the Polynomials Fit of Degree 1

After plotting the ten fits for degree 1, I calculate the prediction for the degree 1 with using the formula y = mx + n which is m = polyFit1[indexx][0], x = test_x[index] and n = polyFit1[indexx][1].

Prediction for Each Model

```python
predictionList1 = []

for indexx in range(10):
    predictionList1.append([])
    for index in range(100):
        prediction = polyFit1[indexx][0] * test_x[index] + polyFit1[indexx][1]
        predictionList1[indexx].append(prediction)
```

Figure 7: Degree 1 Prediction for Each Model

After Calculated the prediction value, I use the mean_squared_error function from the Scikit-Learn and I calculate the MSE value for each 10 model for degree 1.

Calculating Mean Squared Error for Each Model

```python
MeanSquaredError1 = []

for index in range(10):
    mse = mean_squared_error(test_r, predictionList1[index])
    MeanSquaredError1.append(mse)
```

```python
for index, mse in enumerate(MeanSquaredError1):
    print('MSE for Model {} & Degree 1: {}'.format(index+1, mse))
```

```
MSE for Model 1 & Degree 1: 2.5672725281350113
MSE for Model 2 & Degree 1: 2.6095481409662464
MSE for Model 3 & Degree 1: 3.444971650701764
MSE for Model 4 & Degree 1: 2.7864268173314515
MSE for Model 5 & Degree 1: 2.583674322409223
MSE for Model 6 & Degree 1: 3.445554764623103
MSE for Model 7 & Degree 1: 2.6158264871464145
MSE for Model 8 & Degree 1: 3.077775079067153
MSE for Model 9 & Degree 1: 2.7511853969409197
MSE for Model 10 & Degree 1: 2.5888703878204917
```

Figure 8: Degree 1 MSE for Each Model

After making the MSE calculation for each model, I calculate the average mean square error for degree 1's model.

Calculating Avarage Mean Squared Error for Each Model

```python
total = 0
length = len(MeanSquaredError1)

for index in range(10):
    total += MeanSquaredError1[index]

averageMSE1 = total/length
print('Avarage MSE1:', averageMSE1)
```

```
Avarage MSE1: 2.8471105575141773
```

Figure 9: Average MSE for Degree 1

I repeat all the steps I have applied above for each degree.

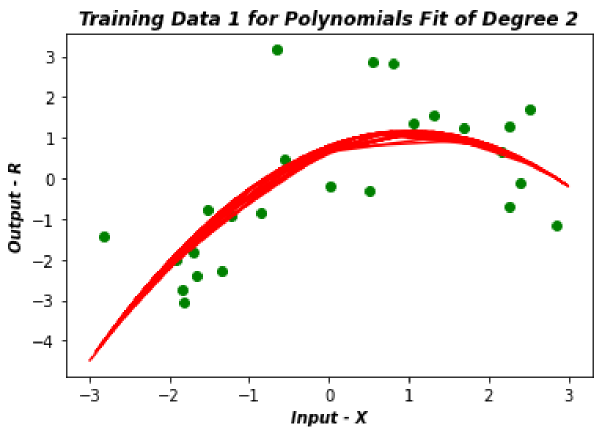
3.1.2 Polynomials Fit of Degree 2

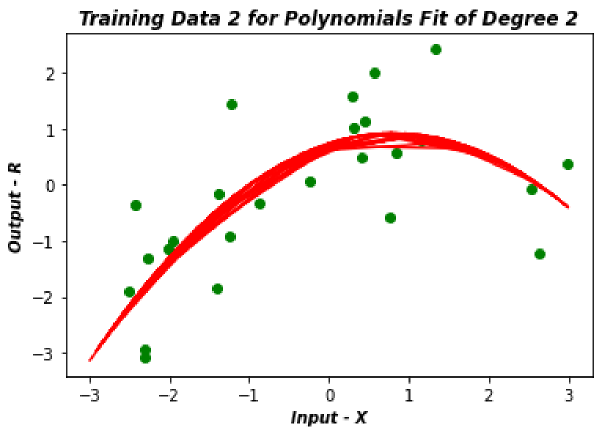
Polynomials Fit of Degree 2

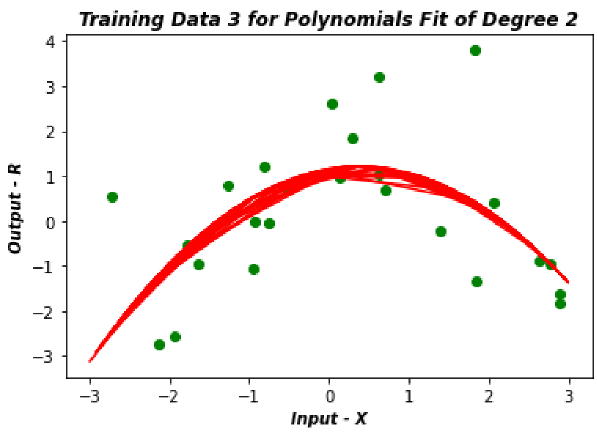
```python
polyFit2 = []

for index in range(10):
    fit = np.polyfit(train_x[index], train_r[index], 2)
    polyFit2.append(fit)
```

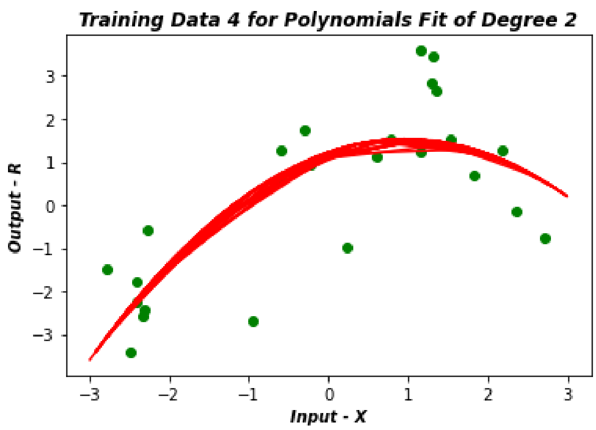
```python
for index in range(10):
    plt.title('Training Data {} for Polynomials Fit of Degree 2'.format(index+1), fontweight="bold", style="italic")
    plt.xlabel("Input - X", fontweight="bold", style="italic")
    plt.ylabel("Output - R", fontweight="bold", style="italic")
    plt.scatter(train_x[index], train_r[index], color = 'g')
    plt.plot(train_x, np.poly1d(polyFit2[index])(train_x), color = 'r')
    plt.show()
```

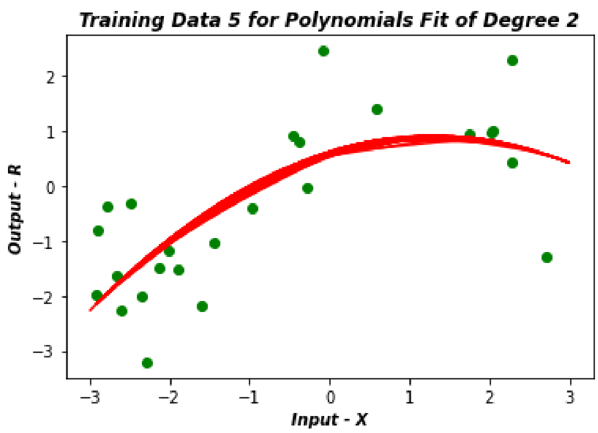
Figure 10: Polynomials Fit of Degree 2

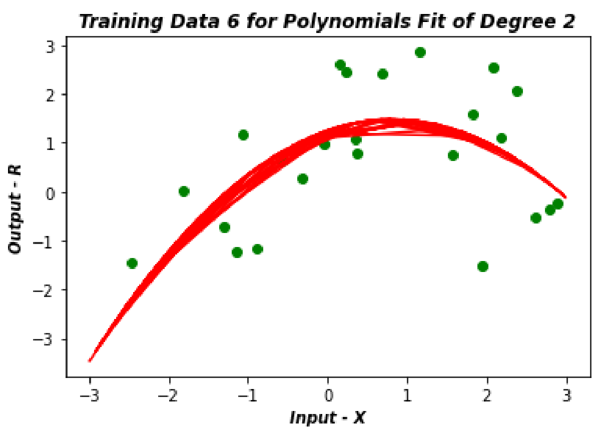

Training Data 1 for Polynomials Fit of Degree 2
Output - R
Input - X

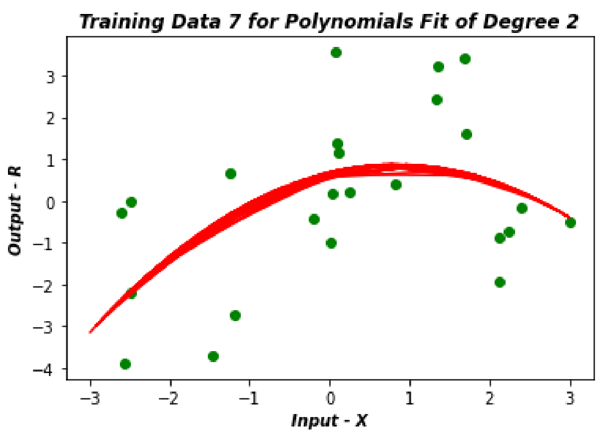

Training Data 2 for Polynomials Fit of Degree 2
Output - R
Input - X

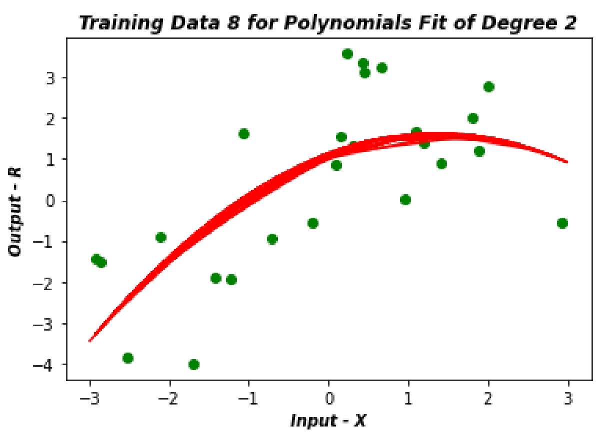

Training Data 3 for Polynomials Fit of Degree 2
Output - R
Input - X

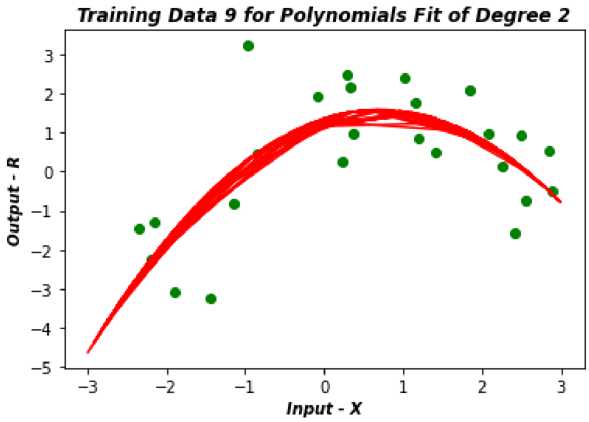

Training Data 4 for Polynomials Fit of Degree 2
Output - R
Input - X

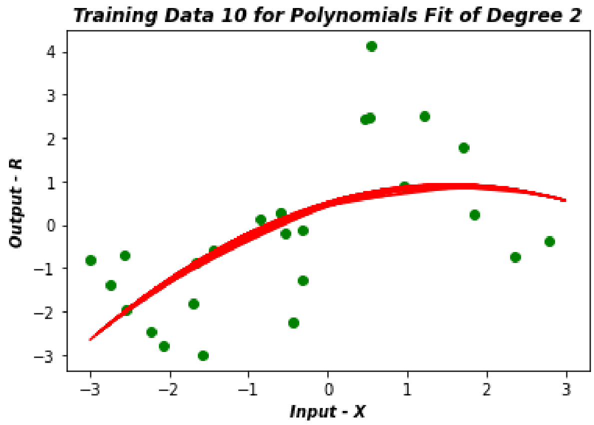

Training Data 5 for Polynomials Fit of Degree 2
Output - R
Input - X


Training Data 6 for Polynomials Fit of Degree 2
Output - R
Input - X


Training Data 7 for Polynomials Fit of Degree 2
Output - R
Input - X


Training Data 8 for Polynomials Fit of Degree 2
Output - R
Input - X





Prediction for Each model

```python
predictionList2 = []

for indexx in range(10):
    predictionList2.append([])
    for index in range(100):
        prediction = ((polyFit2[indexx][0]) * (test_x[index]**2)) + polyFit2[indexx][1] * test_x[index] + polyFit2[indexx][2]
        predictionList2[indexx].append(prediction)
```

[15] Python

Calculating Mean Squared Error for Each Model

```python
MeanSquaredError2 = []

for index in range(10):
    mse = mean_squared_error(test_r, predictionList2[index])
    MeanSquaredError2.append(mse)
```

[16] Python

```python
for index, mse in enumerate(MeanSquaredError2):
    print('MSE for Model {} & Degree 2: {}'.format(index+1, mse))
```

[17] Python

```
MSE for Model 1 & Degree 2: 1.9268225266724486
MSE for Model 2 & Degree 2: 2.0132604222643073
MSE for Model 3 & Degree 2: 2.344765798210042
MSE for Model 4 & Degree 2: 2.113691075554605
MSE for Model 5 & Degree 2: 2.210762486971338
MSE for Model 6 & Degree 2: 2.1331996586714745
MSE for Model 7 & Degree 2: 2.007064044199873
MSE for Model 8 & Degree 2: 2.208997102223812
MSE for Model 9 & Degree 2: 2.0692536648930995
MSE for Model 10 & Degree 2: 2.0895040591667096
```

Calculating Avarage Mean Squared Error for Each Model

```python
total = 0
length = len(MeanSquaredError2)

for index in range(10):
    total += MeanSquaredError2[index]

averageMSE2 = total/length
print('Avarage MSE2:', averageMSE2)
```

[18] Python

```
Avarage MSE2: 2.1117320838827713
```

Figure 11: Prediction – MSE and Average MSE for Degree 2

3.1.3 Polynomials Fit of Degree 3

Polynomials Fit of Degree 3

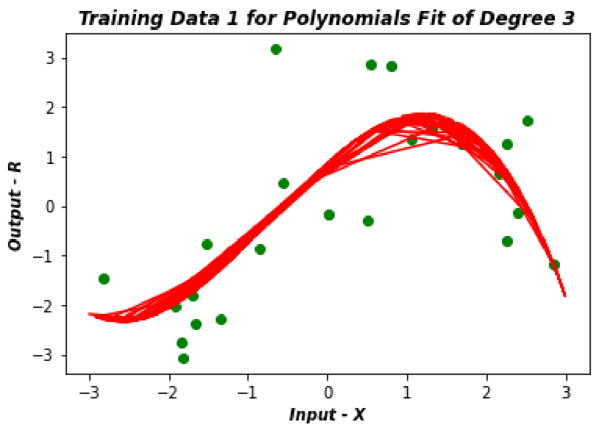
```python
polyFit3 = []

for index in range(10):
    fit = np.polyfit(train_x[index], train_r[index], 3)
    polyFit3.append(fit)
```

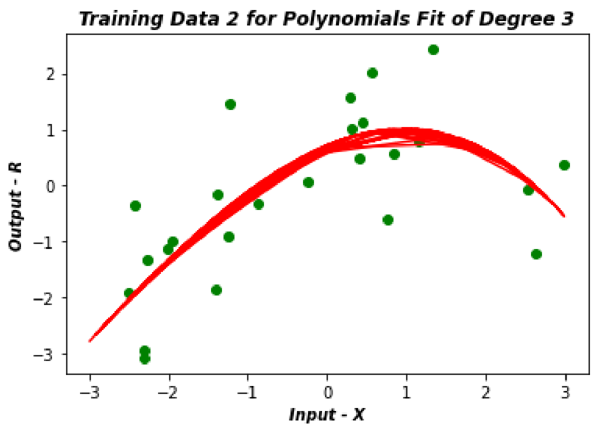
[19] Python

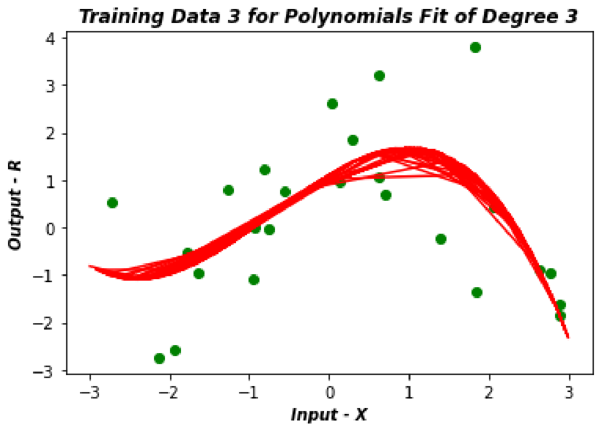
```python
for index in range(10):
    plt.title('Training Data {} for Polynomials Fit of Degree 3'.format(index+1), fontweight="bold", style="italic")
    plt.xlabel("Input - X", fontweight="bold", style="italic")
    plt.ylabel("Output - R", fontweight="bold", style="italic")
    plt.scatter(train_x[index], train_r[index], color = 'g')
    plt.plot(train_x, np.poly1d(polyFit3[index])(train_x), color = 'r')
    plt.show()
```

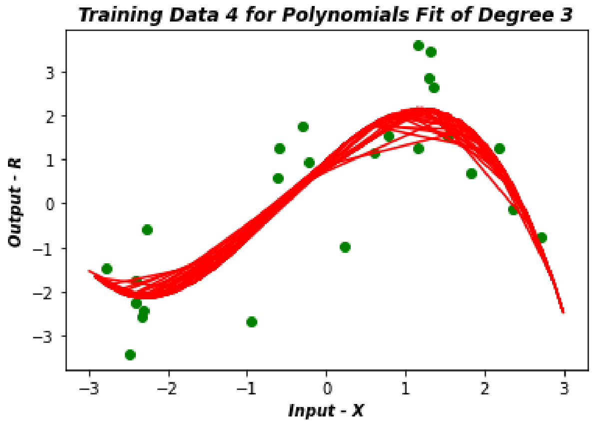
[20] Python

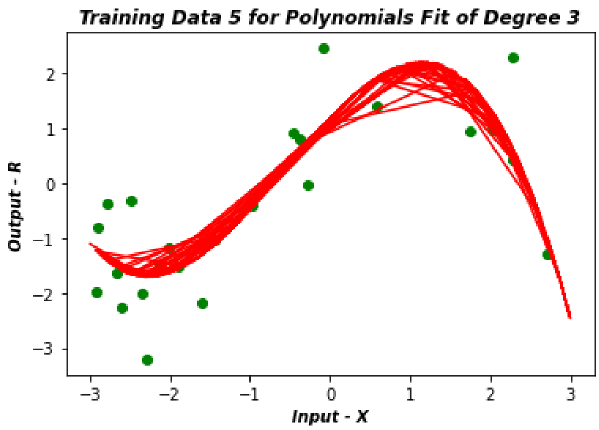
Figure 12: Polynomials Fit of Degree 3

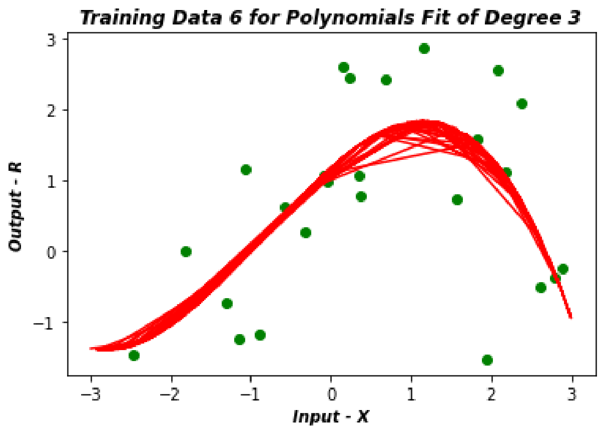

Training Data 1 for Polynomials Fit of Degree 3
Output - R
Input - X

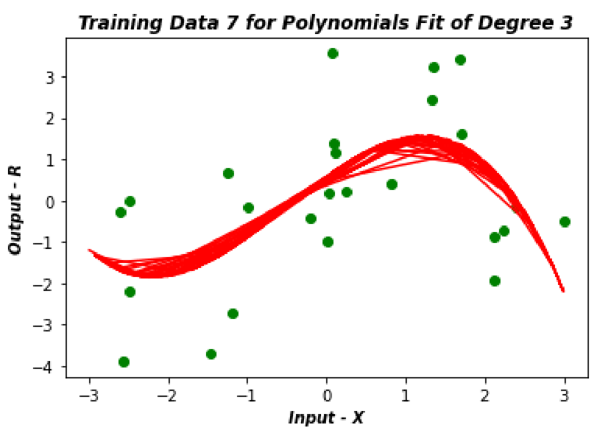

Training Data 2 for Polynomials Fit of Degree 3
Output - R
Input - X

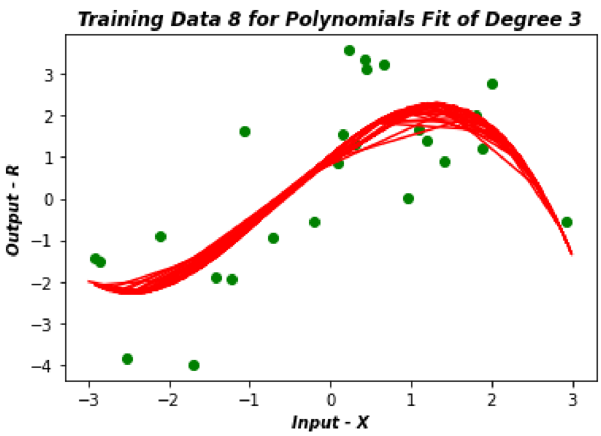

Training Data 3 for Polynomials Fit of Degree 3
Output - R
Input - X


Training Data 4 for Polynomials Fit of Degree 3
Output - R
Input - X


Training Data 5 for Polynomials Fit of Degree 3
Output - R
Input - X


Training Data 6 for Polynomials Fit of Degree 3
Output - R
Input - X


Training Data 7 for Polynomials Fit of Degree 3
Output - R
Input - X


Training Data 8 for Polynomials Fit of Degree 3
Output - R
Input - X

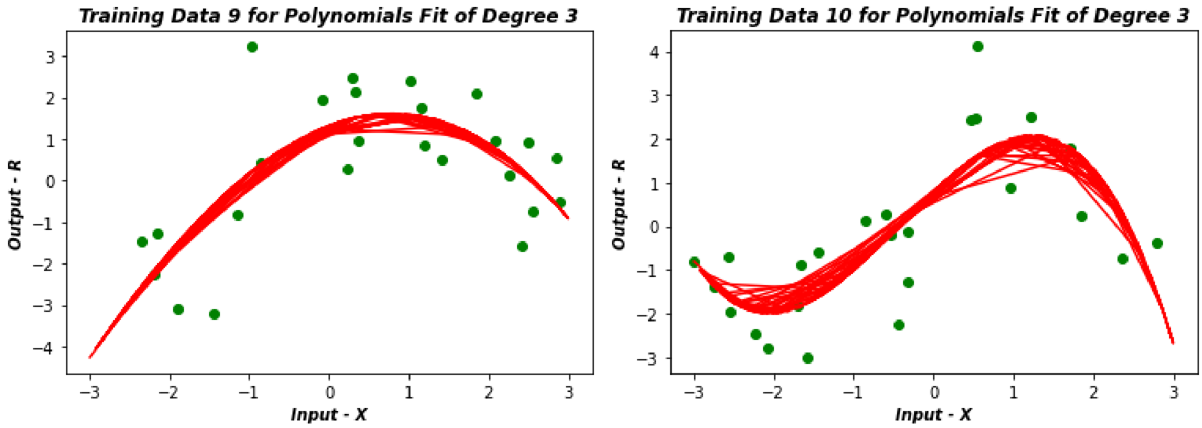


Figure 13: Plotting the Polynomials Fit of Degree 3

Prediction for Each model

```
predictionList3 = []

for indexx in range(10):
    predictionList3.append([])
    for index in range(100):
        prediction = ((polyFit3[indexx][0]) * (test_x[index]**3)) + ((polyFit3[indexx][1]) * (test_x[index]**2)) + polyFit3[indexx][2] * test_x[index] + polyFi
        predictionList3[indexx].append(prediction)
```

Calculating Mean Squared Error for Each Model

```
MeanSquaredError3 = []

for index in range(10):
    mse = mean_squared_error(test_r, predictionList3[index])
    MeanSquaredError3.append(mse)
```

```
for index, mse in enumerate(MeanSquaredError3):
    print('MSE for Model {} & Degree 3: {}'.format(index+1, mse))
```

```
MSE for Model 1 & Degree 3: 1.549724978735644
MSE for Model 2 & Degree 3: 1.914317046598652
MSE for Model 3 & Degree 3: 1.9885742712429322
MSE for Model 4 & Degree 3: 1.58161959952977
MSE for Model 5 & Degree 3: 1.671419153524317
MSE for Model 6 & Degree 3: 1.9520274442440086
MSE for Model 7 & Degree 3: 1.63047027109046
MSE for Model 8 & Degree 3: 1.6393188530181093
MSE for Model 9 & Degree 3: 1.963922707714174
MSE for Model 10 & Degree 3: 1.633880004453998
```

Calculating Avarage Mean Squared Error for Each Model

```
total = 0
length = len(MeanSquaredError3)

for index in range(10):
    total += MeanSquaredError3[index]

averageMSE3 = total/length
print('Avarage MSE3:', averageMSE3)
```

```
Avarage MSE3: 1.752527433015206
```

Figure 14: Prediction – MSE and Average MSE for Degree 3

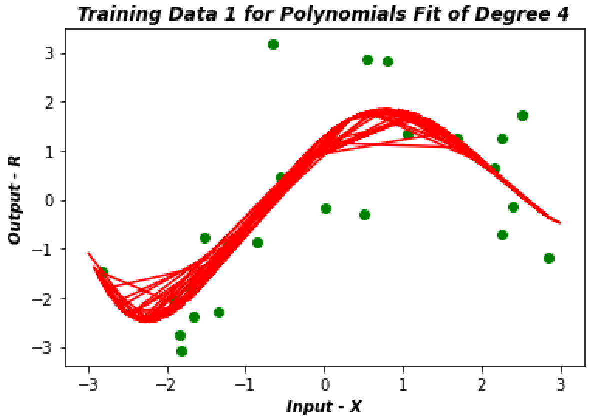
3.1.4 Polynomials Fit of Degree 4

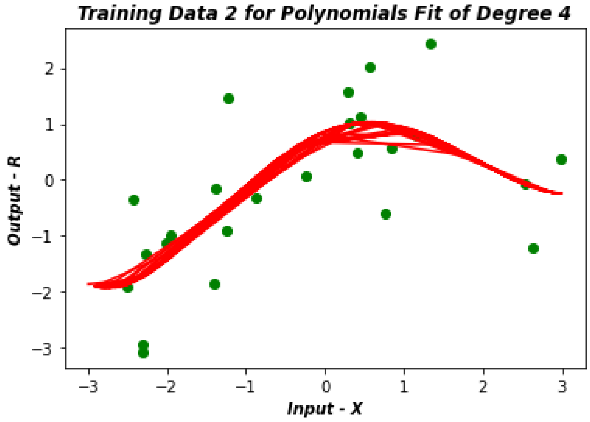
Polynomials Fit of Degree 4

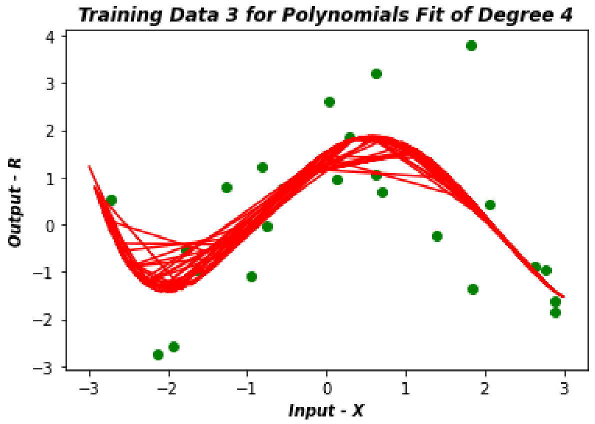
```
polyFit4 = []

for index in range(10):
    fit = np.polyfit(train_x[index], train_r[index], 4)
    polyFit4.append(fit)
```

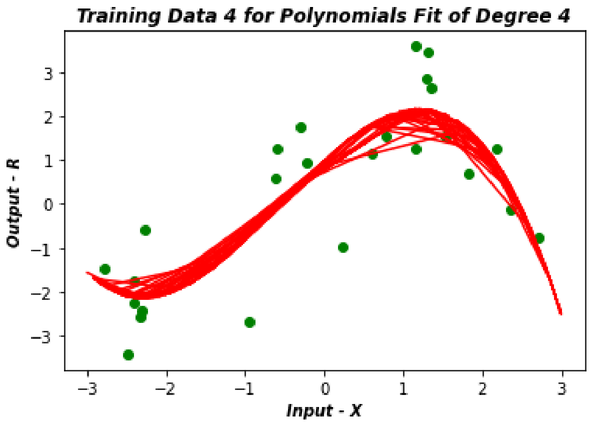
```
for index in range(10):
    plt.title('Training Data {} for Polynomials Fit of Degree 4'.format(index+1), fontweight="bold", style="italic")
    plt.xlabel("Input - X", fontweight="bold", style="italic")
    plt.ylabel("Output - R", fontweight="bold", style="italic")
    plt.scatter(train_x[index], train_r[index], color = 'g')
    plt.plot(train_x, np.poly1d(polyFit4[index])(train_x), color = 'r')
    plt.show()
```

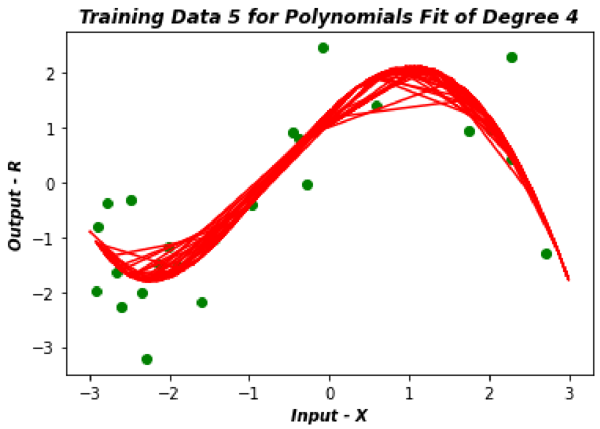
Figure 15: Polynomials Fit of Degree 4

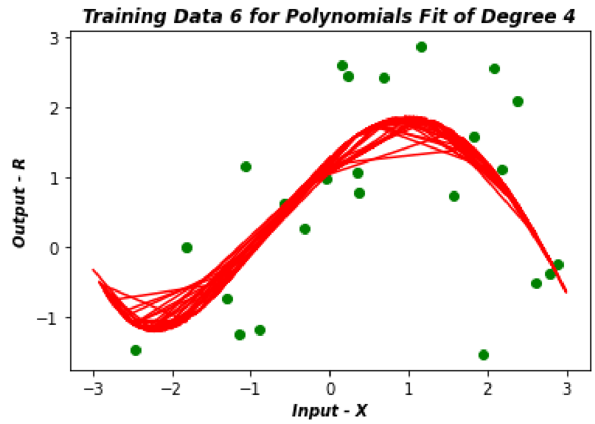

Training Data 1 for Polynomials Fit of Degree 4
Output - R
Input - X

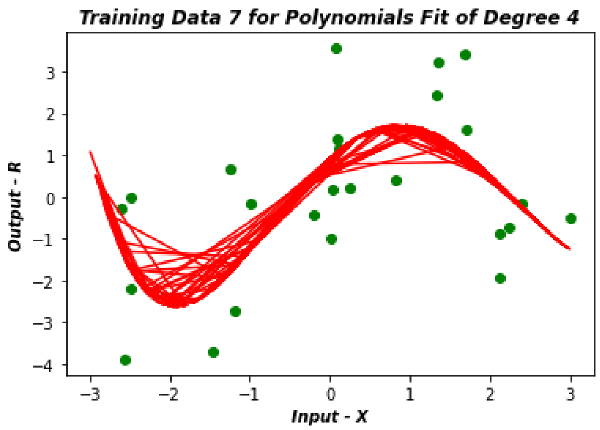

Training Data 2 for Polynomials Fit of Degree 4
Output - R
Input - X

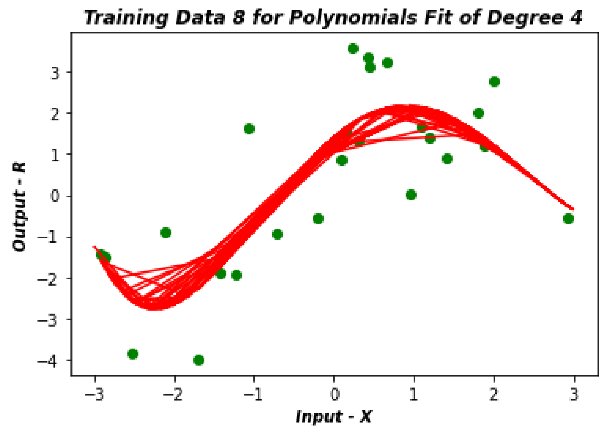

Training Data 3 for Polynomials Fit of Degree 4
Output - R
Input - X


Training Data 4 for Polynomials Fit of Degree 4
Output - R
Input - X


Training Data 5 for Polynomials Fit of Degree 4
Output - R
Input - X


Training Data 6 for Polynomials Fit of Degree 4
Output - R
Input - X


Training Data 7 for Polynomials Fit of Degree 4
Output - R
Input - X


Training Data 8 for Polynomials Fit of Degree 4
Output - R
Input - X

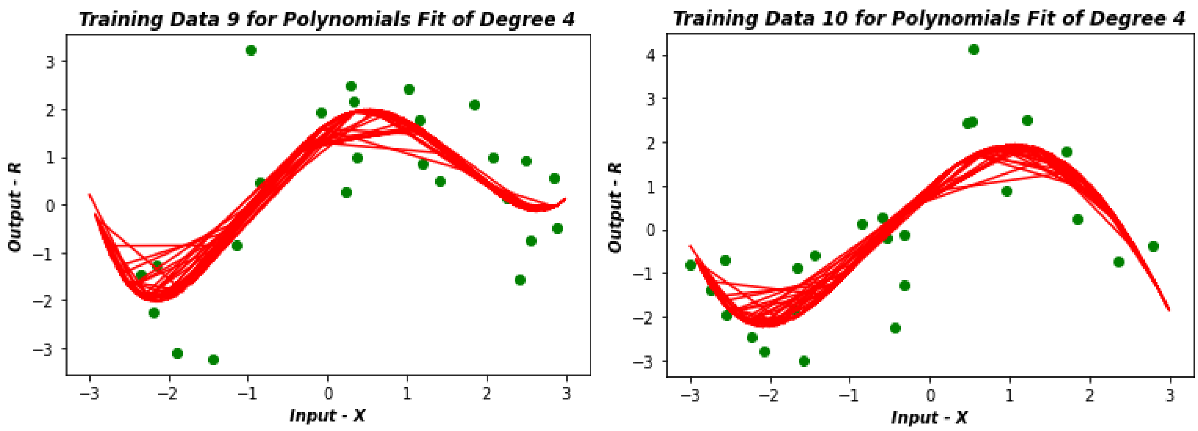


Figure 16: Plotting the Polynomials Fit of Degree 4

Prediction for Each model

```
predictionList4 = []

for indexx in range(10):
    predictionList4.append([])
    for index in range(100):
        prediction = ((polyFit4[indexx][0]) * (test_x[index]**4)) + ((polyFit4[indexx][1]) * (test_x[index]**3)) + ((polyFit4[indexx][2]) * (test_x[index]**2))
        predictionList4[indexx].append(prediction)
```

[27] Python

Calculating Mean Squared Error for Each Model

```
MeanSquaredError4 = []

for index in range(10):
    mse = mean_squared_error(test_r, predictionList4[index])
    MeanSquaredError4.append(mse)
```

[28] Python

```
for index, mse in enumerate(MeanSquaredError4):
    print('MSE for Model {} & Degree 4: {}'.format(index+1, mse))
```

[29] Python

```
MSE for Model 1 & Degree 4: 1.6018397980686208
MSE for Model 2 & Degree 4: 1.940790453490305
MSE for Model 3 & Degree 4: 2.137702797004289
MSE for Model 4 & Degree 4: 1.5831023434000848
MSE for Model 5 & Degree 4: 1.6399737224503093
MSE for Model 6 & Degree 4: 1.9575776282353972
MSE for Model 7 & Degree 4: 1.8094072474835579
MSE for Model 8 & Degree 4: 1.6869870519995684
MSE for Model 9 & Degree 4: 1.8893483405696891
MSE for Model 10 & Degree 4: 1.5818635254672426
```

Calculating Avarage Mean Squared Error for Each Model

```
total = 0
length = len(MeanSquaredError4)

for index in range(10):
    total += MeanSquaredError4[index]

averageMSE4 = total/length
print('Avarage MSE4:', averageMSE4)
```

[30] Python

```
Avarage MSE4: 1.7828592908169063
```

Figure 17: Prediction – MSE and Average MSE for Degree 4

3.1.5 Polynomials Fit of Degree 5

Polynomials Fit of Degree 5

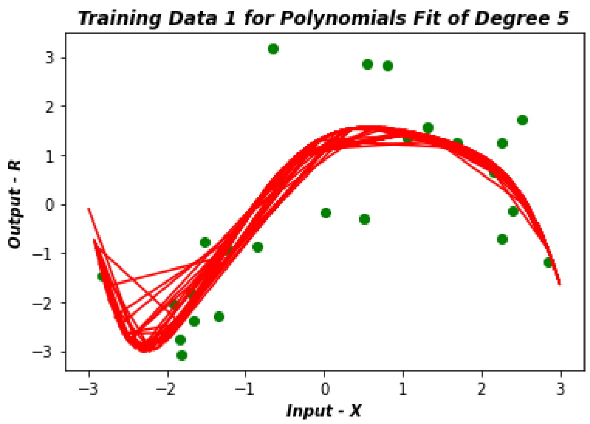
```
polyFit5 = []

for index in range(10):
    fit = np.polyfit(train_x[index], train_r[index], 5)
    polyFit5.append(fit)
```

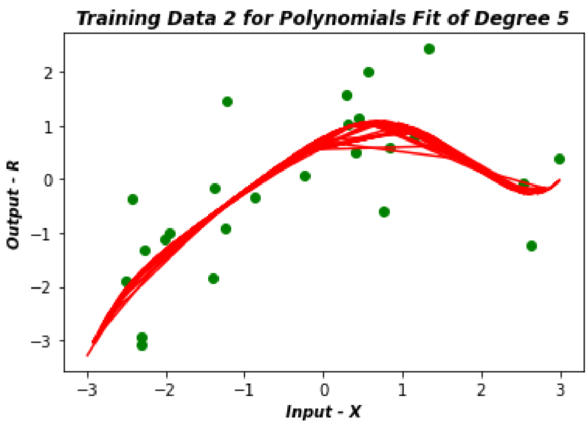
[31] Python

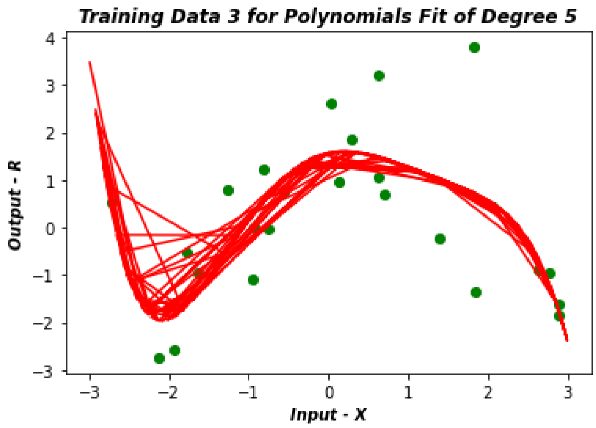
```
for index in range(10):
    plt.title('Training Data {} for Polynomials Fit of Degree 5'.format(index+1), fontweight="bold", style="italic")
    plt.xlabel("Input - X", fontweight="bold", style="italic")
    plt.ylabel("Output - R", fontweight="bold", style="italic")
    plt.scatter(train_x[index], train_r[index], color = 'g')
    plt.plot(train_x, np.poly1d(polyFit5[index])(train_x), color = 'r')
    plt.show()
```

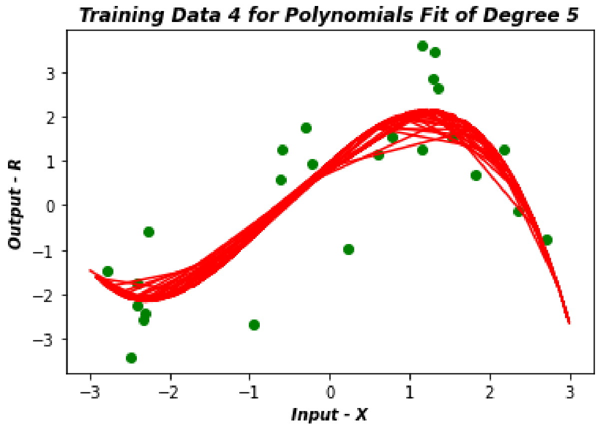
[32] Python

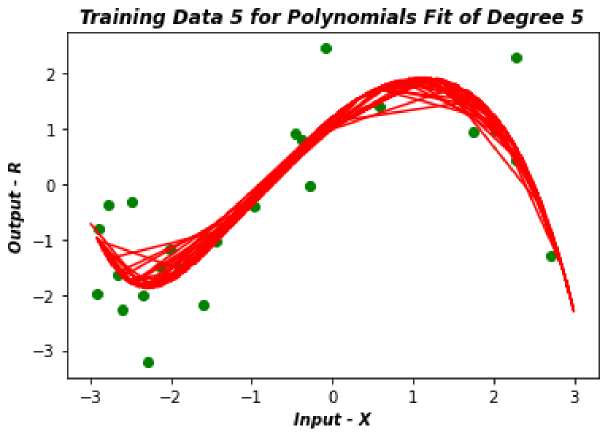
Figure 18: Polynomials Fit of Degree 5

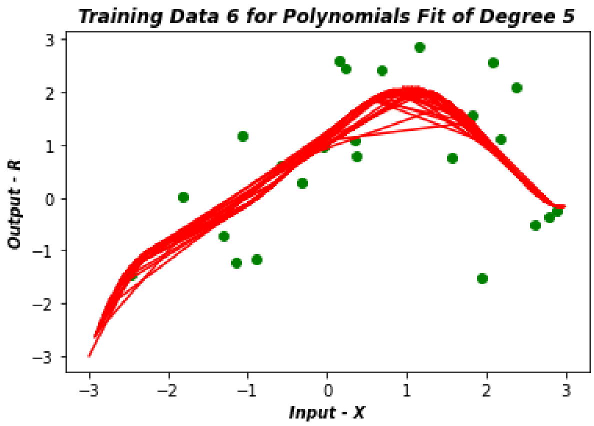

Training Data 1 for Polynomials Fit of Degree 5
Output - R
Input - X

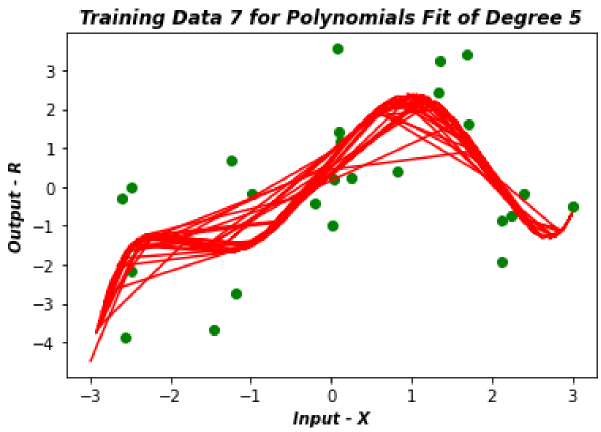

Training Data 2 for Polynomials Fit of Degree 5
Output - R
Input - X

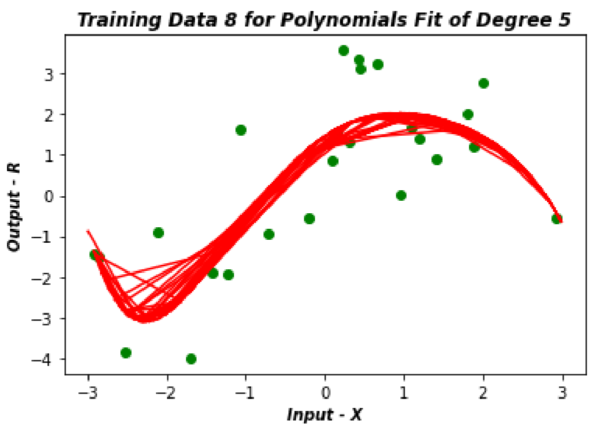

Training Data 3 for Polynomials Fit of Degree 5
Output - R
Input - X


Training Data 4 for Polynomials Fit of Degree 5
Output - R
Input - X


Training Data 5 for Polynomials Fit of Degree 5
Output - R
Input - X


Training Data 6 for Polynomials Fit of Degree 5
Output - R
Input - X


Training Data 7 for Polynomials Fit of Degree 5
Output - R
Input - X


Training Data 8 for Polynomials Fit of Degree 5
Output - R
Input - X

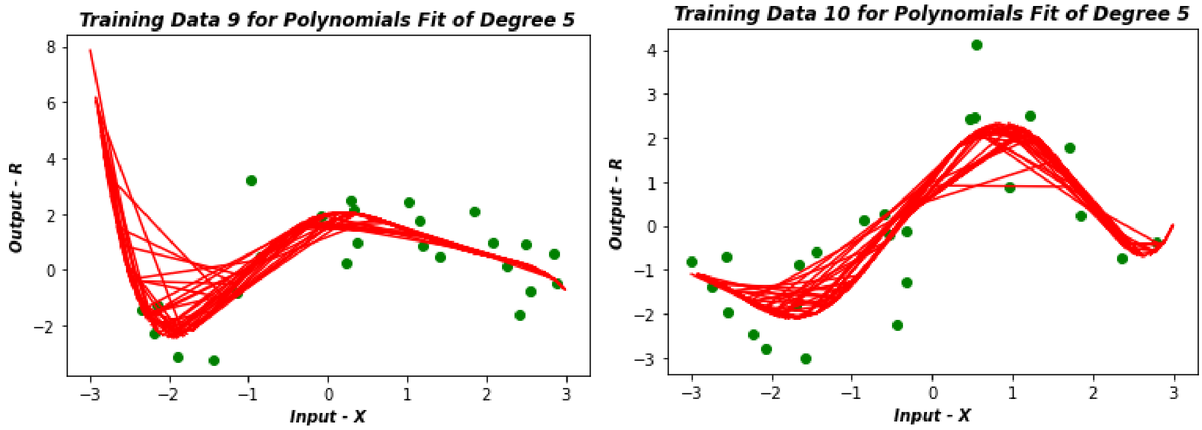


Figure 19: Plotting the Polynomials Fit of Degree 5

Prediction for Each model

```python
predictionList5 = []

for indexx in range(10):
    predictionList5.append([])
    for index in range(100):
        prediction = ((polyFit5[indexx][0]) * (test_x[index]**5)) + ((polyFit5[indexx][1]) * (test_x[index]**4)) + ((polyFit5[indexx][2]) * (test_x[index]**3))
        predictionList5[indexx].append(prediction)
```

[33] Python

Calculating Mean Squared Error for Each Model

```python
MeanSquaredError5 = []

for index in range(10):
    mse = mean_squared_error(test_r, predictionList5[index])
    MeanSquaredError5.append(mse)
```

[34] Python

```python
for index, mse in enumerate(MeanSquaredError5):
    print('MSE for Model {} & Degree 5: {}'.format(index+1, mse))
```

[35] Python

```
MSE for Model 1 & Degree 5: 1.7282666326675562
MSE for Model 2 & Degree 5: 1.985243595595827
MSE for Model 3 & Degree 5: 2.337159260799628
MSE for Model 4 & Degree 5: 1.5918528860579035
MSE for Model 5 & Degree 5: 1.661700393983935
MSE for Model 6 & Degree 5: 2.0026098865789197
MSE for Model 7 & Degree 5: 1.9507208342183355
MSE for Model 8 & Degree 5: 1.7963050169128718
MSE for Model 9 & Degree 5: 3.453596649561328
MSE for Model 10 & Degree 5: 1.7483061305547252
```

Calculating Avarage Mean Squared Error for Each Model

```python
total = 0
length = len(MeanSquaredError5)

for index in range(10):
    total += MeanSquaredError5[index]

averageMSE5 = total/length
print('Avarage MSE5:', averageMSE5)
```

[36] Python

```
Avarage MSE5: 2.025576128693103
```

Figure 20: Prediction – MSE and Average MSE for Degree 5

3.1.6 Polynomials Fit of Degree 6

Polynomials Fit of Degree 6

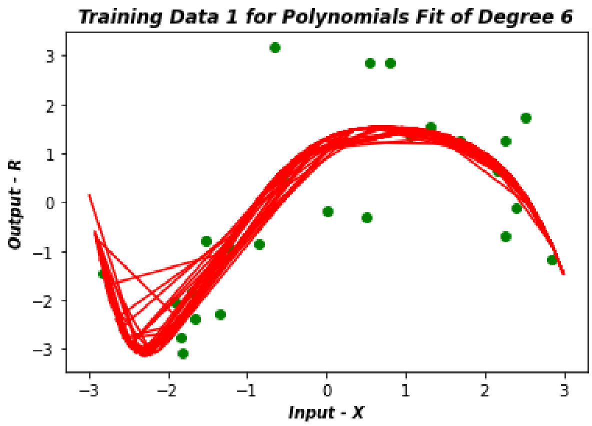
```python
polyFit6 = []

for index in range(10):
    fit = np.polyfit(train_x[index], train_r[index], 6)
    polyFit6.append(fit)
```

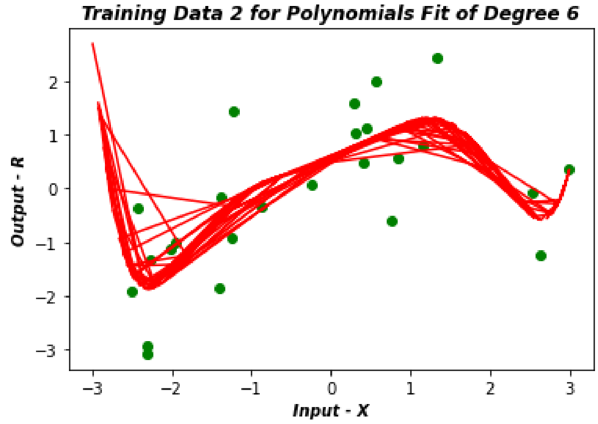
[37] Python

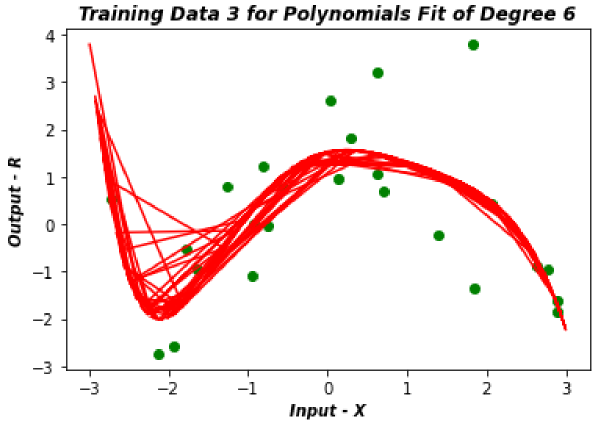
```python
for index in range(10):
    plt.title('Training Data {} for Polynomials Fit of Degree 6'.format(index+1), fontweight="bold", style="italic")
    plt.xlabel("Input - X", fontweight="bold", style="italic")
    plt.ylabel("Output - R", fontweight="bold", style="italic")
    plt.scatter(train_x[index], train_r[index], color = 'g')
    plt.plot(train_x, np.poly1d(polyFit6[index])(train_x), color = 'r')
    plt.show()
```

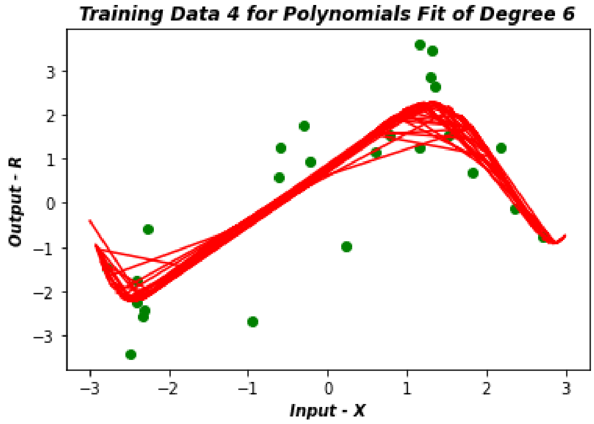
[38] Python

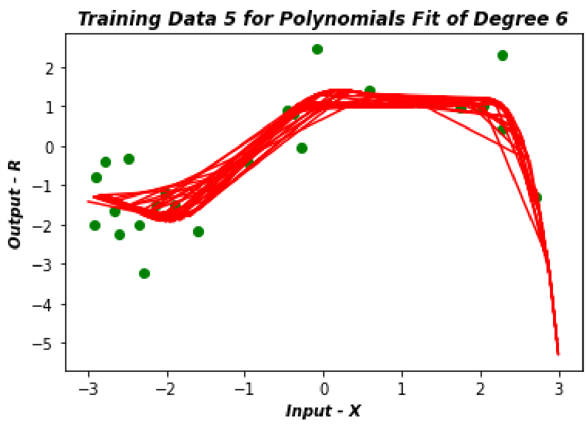
Figure 21: Polynomials Fit of Degree 6

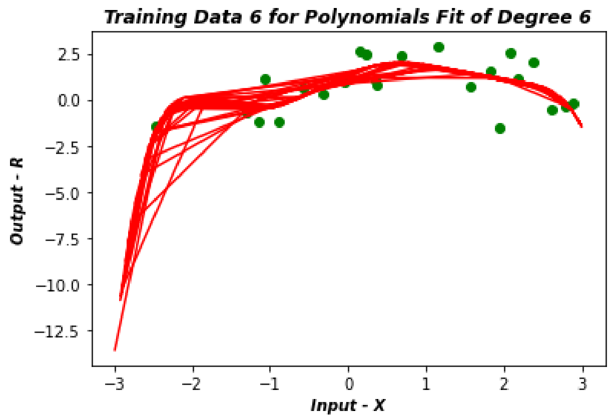

Training Data 1 for Polynomials Fit of Degree 6
Output - R
Input - X

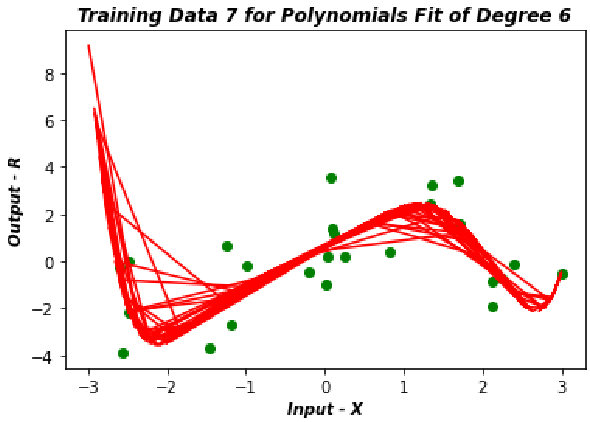

Training Data 2 for Polynomials Fit of Degree 6
Output - R
Input - X

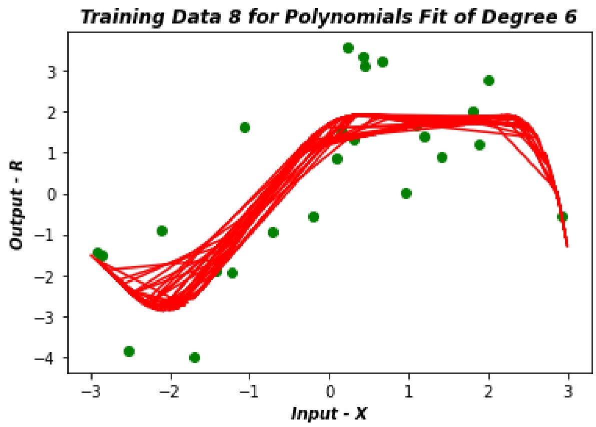

Training Data 3 for Polynomials Fit of Degree 6
Output - R
Input - X


Training Data 4 for Polynomials Fit of Degree 6
Output - R
Input - X


Training Data 5 for Polynomials Fit of Degree 6
Output - R
Input - X


Training Data 6 for Polynomials Fit of Degree 6
Output - R
Input - X


Training Data 7 for Polynomials Fit of Degree 6
Output - R
Input - X


Training Data 8 for Polynomials Fit of Degree 6
Output - R
Input - X

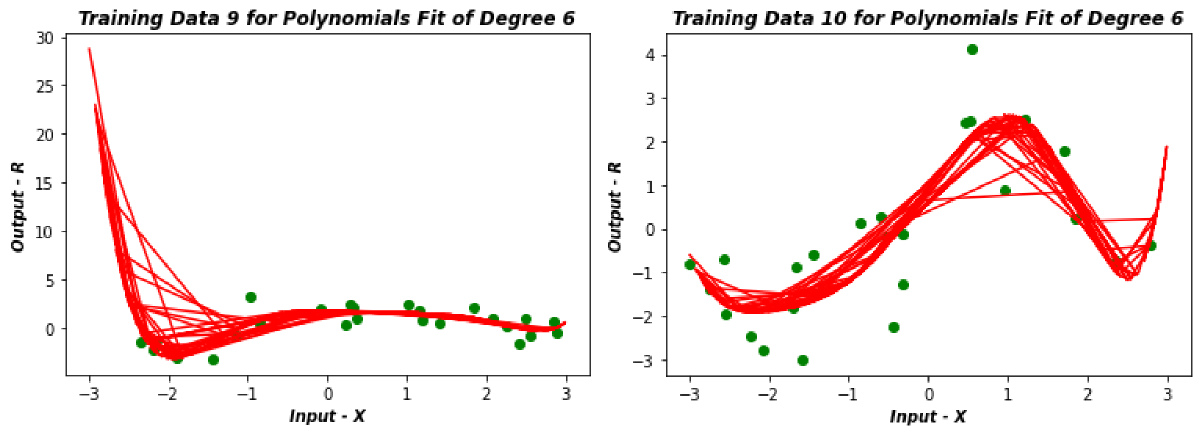


Figure 22: Plotting the Polynomials Fit of Degree 6

Prediction for Each model

```python
predictionList6 = []

for indexx in range(10):
    predictionList6.append([])
    for index in range(100):
        prediction = ((polyFit6[indexx][0]) * (test_x[index]**6)) + ((polyFit6[indexx][1]) * (test_x[index]**5)) + ((polyFit6[indexx][2]) * (test_x[index]**4))
        predictionList6[indexx].append(prediction)
```
[39] Python

Calculating Mean Squared Error for Each Model

```python
MeanSquaredError6 = []

for index in range(10):
    mse = mean_squared_error(test_r, predictionList6[index])
    MeanSquaredError6.append(mse)
```
[40] Python

```python
for index, mse in enumerate(MeanSquaredError6):
    print('MSE for Model {} & Degree 6: {}'.format(index+1, mse))
```
[41] Python

```
MSE for Model 1 & Degree 6: 1.7368982626137006
MSE for Model 2 & Degree 6: 2.060224387525662
MSE for Model 3 & Degree 6: 2.338943487710145
MSE for Model 4 & Degree 6: 1.6186765173254742
MSE for Model 5 & Degree 6: 2.040274014258989
MSE for Model 6 & Degree 6: 3.6910090331238945
MSE for Model 7 & Degree 6: 3.0369330511995387
MSE for Model 8 & Degree 6: 2.0784809330596787
MSE for Model 9 & Degree 6: 14.83564397067334
MSE for Model 10 & Degree 6: 1.8919263383224871
```

Calculating Avarage Mean Squared Error for Each Model

```python
total = 0
length = len(MeanSquaredError6)

for index in range(10):
    total += MeanSquaredError6[index]

averageMSE6 = total/length
print('Avarage MSE6:', averageMSE6)
```
[42] Python

```
Avarage MSE6: 3.5329009995812912
```

Figure 23: Prediction – MSE and Average MSE for Degree 6

4 Results

In this part, I plot the Average Mean Square Error for each degree.

Figure 24: Average Mean Square Error of Models with Different Degrees

As seen in Figure 1, the lowest error rate was found in the Degree 3 which is 1.7525. This demonstrates the importance of Degree 3 models in model selection.

5 Source Code

```python
import pandas as pd
import numpy as np

from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error

myTrainList = ['sample1.csv', 'sample2.csv', 'sample3.csv', 'sample4.csv',
'sample5.csv', 'sample6.csv', 'sample7.csv', 'sample8.csv', 'sample9.csv',
'sample10.csv']
trainDataFrame = []

for trainData in myTrainList:
    data = pd.read_csv(trainData, names=['X', 'R'],  header=None)
    trainDataFrame.append(data)

testDataFrame = pd.read_csv('test.csv', names=['X', 'R'], header=None)

train_x = []
train_r = []
for index in range(10):
    train_x.append(trainDataFrame[index]['X'].values)
    train_r.append(trainDataFrame[index]['R'].values)

test_x = testDataFrame['X'].values
test_r = testDataFrame['R'].values
```

```
plt.title('Training Data', fontweight="bold", style="italic")
plt.xlabel("Input - X", fontweight="bold", style="italic")
plt.ylabel("Output - R", fontweight="bold", style="italic")
plt.scatter(train_x, train_r, color = 'g')
plt.show()

plt.title('Testing Data', fontweight="bold", style="italic")
plt.xlabel("Input - X", fontweight="bold", style="italic")
plt.ylabel("Output - R", fontweight="bold", style="italic")
plt.scatter(test_x, test_r, color = 'b')
plt.show()

polyFit1 = []
for index in range(10):
    fit = np.polyfit(train_x[index], train_r[index], 1)
    polyFit1.append(fit)

for index in range(10):
    plt.title('Training Data {} for Polynomials Fit of Degree
1'.format(index+1), fontweight="bold", style="italic")
    plt.xlabel("Input - X", fontweight="bold", style="italic")
    plt.ylabel("Output - R", fontweight="bold", style="italic")
    plt.scatter(train_x[index], train_r[index], color = 'g')
    plt.plot(train_x, np.poly1d(polyFit1[index])(train_x), color = 'r')
    plt.show()

predictionList1 = []
for indexx in range(10):
    predictionList1.append([])
    for index in range(100):
        prediction = polyFit1[indexx][0] * test_x[index] + polyFit1[indexx][1]
        predictionList1[indexx].append(prediction)

MeanSquaredError1 = []
for index in range(10):
    mse = mean_squared_error(test_r, predictionList1[index])
    MeanSquaredError1.append(mse)

for index, mse in enumerate(MeanSquaredError1):
    print('MSE for Model {} & Degree 1: {}'.format(index+1, mse))

total = 0
length = len(MeanSquaredError1)
for index in range(10):
    total += MeanSquaredError1[index]

averageMSE1 = total/length
print('Avarage MSE1:', averageMSE1)
```

```
polyFit2 = []
for index in range(10):
    fit = np.polyfit(train_x[index], train_r[index], 2)
    polyFit2.append(fit)

for index in range(10):
    plt.title('Training Data {} for Polynomials Fit of Degree
2'.format(index+1), fontweight="bold", style="italic")
    plt.xlabel("Input - X", fontweight="bold", style="italic")
    plt.ylabel("Output - R", fontweight="bold", style="italic")
    plt.scatter(train_x[index], train_r[index], color = 'g')
    plt.plot(train_x, np.poly1d(polyFit2[index])(train_x), color = 'r')
    plt.show()

predictionList2 = []
for indexx in range(10):
    predictionList2.append([])
    for index in range(100):
        prediction = ((polyFit2[indexx][0]) * (test_x[index]**2)) +
polyFit2[indexx][1] * test_x[index] + polyFit2[indexx][2]
        predictionList2[indexx].append(prediction)
MeanSquaredError2 = []
for index in range(10):
    mse = mean_squared_error(test_r, predictionList2[index])
    MeanSquaredError2.append(mse)

for index, mse in enumerate(MeanSquaredError2):
    print('MSE for Model {} & Degree 2: {}'.format(index+1, mse))

total = 0
length = len(MeanSquaredError2)
for index in range(10):
    total += MeanSquaredError2[index]
averageMSE2 = total/length
print('Avarage MSE2:', averageMSE2)

polyFit3 = []
for index in range(10):
    fit = np.polyfit(train_x[index], train_r[index], 3)
    polyFit3.append(fit)

for index in range(10):
    plt.title('Training Data {} for Polynomials Fit of Degree
3'.format(index+1), fontweight="bold", style="italic")
    plt.xlabel("Input - X", fontweight="bold", style="italic")
    plt.ylabel("Output - R", fontweight="bold", style="italic")
    plt.scatter(train_x[index], train_r[index], color = 'g')
    plt.plot(train_x, np.poly1d(polyFit3[index])(train_x), color = 'r')
    plt.show()
```

```
predictionList3 = []
for indexx in range(10):
    predictionList3.append([])
    for index in range(100):
        prediction = ((polyFit3[indexx][0]) * (test_x[index]**3)) +
((polyFit3[indexx][1]) * (test_x[index]**2)) + polyFit3[indexx][2] *
test_x[index] + polyFit3[indexx][3]
        predictionList3[indexx].append(prediction)

MeanSquaredError3 = []
for index in range(10):
    mse = mean_squared_error(test_r, predictionList3[index])
    MeanSquaredError3.append(mse)

for index, mse in enumerate(MeanSquaredError3):
    print('MSE for Model {} & Degree 3: {}'.format(index+1, mse))

total = 0
length = len(MeanSquaredError3)
for index in range(10):
    total += MeanSquaredError3[index]

averageMSE3 = total/length
print('Avarage MSE3:', averageMSE3)

polyFit4 = []
for index in range(10):
    fit = np.polyfit(train_x[index], train_r[index], 4)
    polyFit4.append(fit)

for index in range(10):
    plt.title('Training Data {} for Polynomials Fit of Degree
4'.format(index+1), fontweight="bold", style="italic")
    plt.xlabel("Input - X", fontweight="bold", style="italic")
    plt.ylabel("Output - R", fontweight="bold", style="italic")
    plt.scatter(train_x[index], train_r[index], color = 'g')
    plt.plot(train_x, np.poly1d(polyFit4[index])(train_x), color = 'r')
    plt.show()

predictionList4 = []
for indexx in range(10):
    predictionList4.append([])
    for index in range(100):
        prediction = ((polyFit4[indexx][0]) * (test_x[index]**4)) +
((polyFit4[indexx][1]) * (test_x[index]**3)) + ((polyFit4[indexx][2]) *
(test_x[index]**2)) + polyFit4[indexx][3] * test_x[index] +
polyFit4[indexx][4]
        predictionList4[indexx].append(prediction)
```

```
MeanSquaredError4 = []
for index in range(10):
    mse = mean_squared_error(test_r, predictionList4[index])
    MeanSquaredError4.append(mse)

for index, mse in enumerate(MeanSquaredError4):
    print('MSE for Model {} & Degree 4: {}'.format(index+1, mse))

total = 0
length = len(MeanSquaredError4)
for index in range(10):
    total += MeanSquaredError4[index]

averageMSE4 = total/length
print('Avarage MSE4:', averageMSE4)

polyFit5 = []
for index in range(10):
    fit = np.polyfit(train_x[index], train_r[index], 5)
    polyFit5.append(fit)

for index in range(10):
    plt.title('Training Data {} for Polynomials Fit of Degree
5'.format(index+1), fontweight="bold", style="italic")
    plt.xlabel("Input - X", fontweight="bold", style="italic")
    plt.ylabel("Output - R", fontweight="bold", style="italic")
    plt.scatter(train_x[index], train_r[index], color = 'g')
    plt.plot(train_x, np.poly1d(polyFit5[index])(train_x), color = 'r')
    plt.show()

predictionList5 = []
for indexx in range(10):
    predictionList5.append([])
    for index in range(100):
        prediction = ((polyFit5[indexx][0]) * (test_x[index]**5)) +
((polyFit5[indexx][1]) * (test_x[index]**4)) + ((polyFit5[indexx][2]) *
(test_x[index]**3)) + ((polyFit5[indexx][3]) * (test_x[index]**2)) +
polyFit5[indexx][4] * test_x[index] + polyFit5[indexx][5]
        predictionList5[indexx].append(prediction)

MeanSquaredError5 = []
for index in range(10):
    mse = mean_squared_error(test_r, predictionList5[index])
    MeanSquaredError5.append(mse)

for index, mse in enumerate(MeanSquaredError5):
    print('MSE for Model {} & Degree 5: {}'.format(index+1, mse))
```

```
total = 0
length = len(MeanSquaredError5)
for index in range(10):
    total += MeanSquaredError5[index]

averageMSE5 = total/length
print('Avarage MSE5:', averageMSE5)

polyFit6 = []
for index in range(10):
    fit = np.polyfit(train_x[index], train_r[index], 6)
    polyFit6.append(fit)

for index in range(10):
    plt.title('Training Data {} for Polynomials Fit of Degree 
6'.format(index+1), fontweight="bold", style="italic")
    plt.xlabel("Input - X", fontweight="bold", style="italic")
    plt.ylabel("Output - R", fontweight="bold", style="italic")
    plt.scatter(train_x[index], train_r[index], color = 'g')
    plt.plot(train_x, np.poly1d(polyFit6[index])(train_x), color = 'r')
    plt.show()

predictionList6 = []
for indexx in range(10):
    predictionList6.append([])
    for index in range(100):
        prediction = ((polyFit6[indexx][0]) * (test_x[index]**6)) + 
((polyFit6[indexx][1]) * (test_x[index]**5)) + ((polyFit6[indexx][2]) * 
(test_x[index]**4)) + ((polyFit6[indexx][3]) * (test_x[index]**3)) + 
((polyFit6[indexx][4]) * (test_x[index]**2)) + polyFit6[indexx][5] * 
test_x[index] + polyFit6[indexx][6]
        predictionList6[indexx].append(prediction)

MeanSquaredError6 = []
for index in range(10):
    mse = mean_squared_error(test_r, predictionList6[index])
    MeanSquaredError6.append(mse)

for index, mse in enumerate(MeanSquaredError6):
    print('MSE for Model {} & Degree 6: {}'.format(index+1, mse))

total = 0
length = len(MeanSquaredError6)
for index in range(10):
    total += MeanSquaredError6[index]

averageMSE6 = total/length
print('Avarage MSE6:', averageMSE6)
```

```
averageMSE = [averageMSE1, averageMSE2, averageMSE3, averageMSE4, averageMSE5,
averageMSE6]
# labels = ['Average MSE 1', 'Average MSE 2', 'Average MSE 3', 'Average MSE
4', 'Average MSE 5', 'Average MSE 6']

for index in range(6):
    plt.title('Average Mean Squared Error for Each Degree', fontweight="bold",
style="italic")
    plt.xlabel("Degree", fontweight="bold", style="italic")
    plt.ylabel("Average Mean Squared Error", fontweight="bold",
style="italic")
    plt.scatter('Degree {}'.format(index+1), averageMSE[index])
```

6 References

[1] Example of overfitting with polynomial regression ... (n.d.). Retrieved March 14, 2022, from https://www.researchgate.net/figure/Example-of-overfitting-with-polynomial-regression-Increasing-the-order-of-the-polynomial_fig2_331733728