MDL Assignment 1

Pooja Desur 2019101112

Manasvi Vaidyula 2019101012

Task 1

Linear Regression is one of the most common and widely practiced regression techniques. Regression problems help to find relationships among different variables and Linear Regression, being one of the simplest regression methods, is the easiest method to interpret results.

Given independent input variables $\mathbf{x} = (x_1, \ldots, x_r)$, with r being the number of features, while using Linear Regression, we assume a linear relationship between y (the output) and the features.

$$y = \beta_0 \; + \beta_1 x_1 \; + \; ... \; + \; \beta_r x_r \; + \; \epsilon$$

where ϵ is the irreducible error (refer to task 3).

Linear regression calculates the best estimated regression function

$$\hat{f}(x) = b_0 + b_1 x_1 \; + ... \; + \; b_r x_r$$

by calculating the predicted weights which are $b = b_0, b_1, \ldots, b_r$. The best-estimated regression function would give as close of a $\hat{f}(\mathbf{x}_i)$ for all observations i = 1, ..., n as to their actual value y_i.

LinearRegression() is a class in the python sklearn.linear_model package. By using .fit() on an instance of this class and giving the input values and actual output values (x and y) as arguments, the optimal weights b are calculated. It returns a model that can be used to predict output values $f(\mathbf{x})$ given an input $\mathbf{x}$ = (x_1, ..., x_r) by calling .predict() on the model.

Task 2

DEFINITIONS:

Bias:

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

$$Bias^2 = (E[\hat{f}(x)] - f(x))^2$$

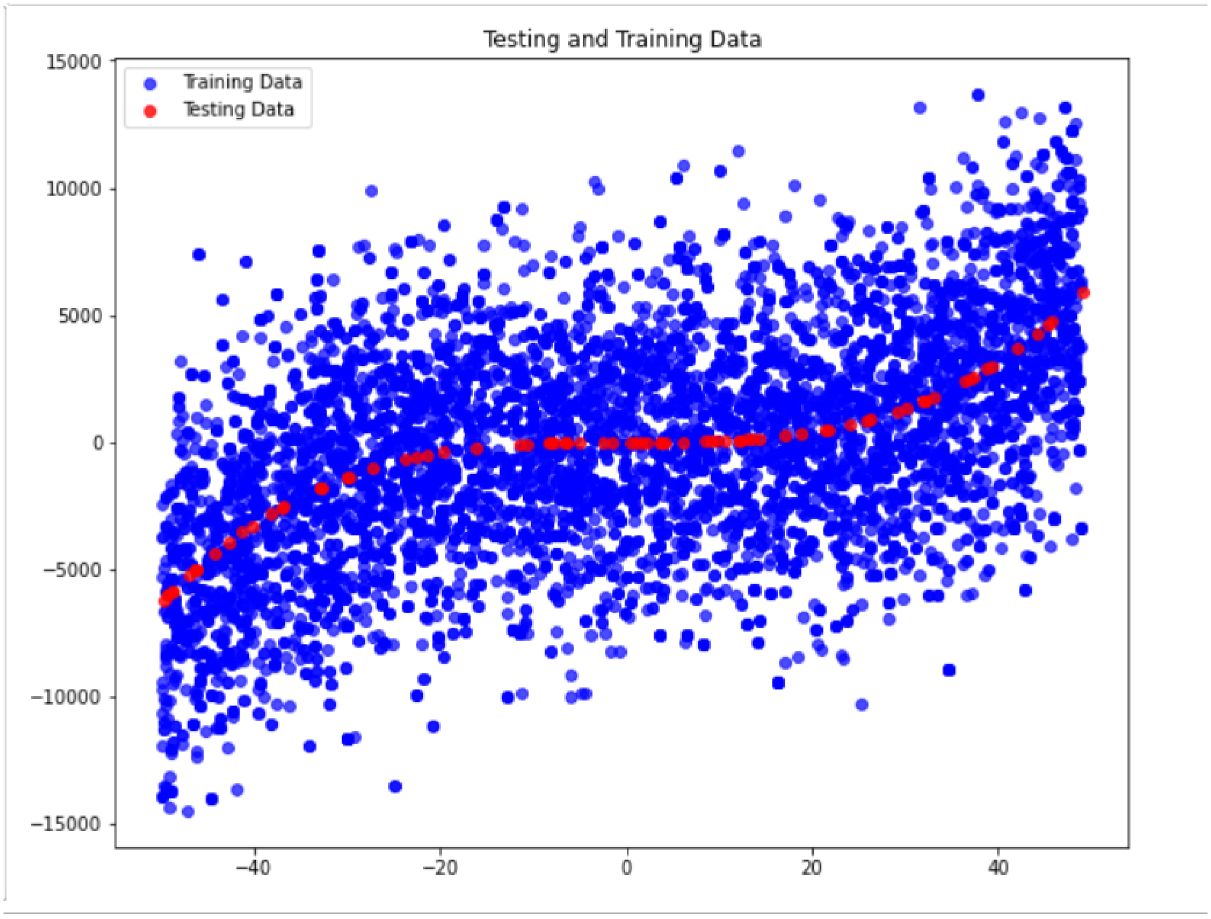
Variance:

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.

$$Variance = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$$

ALGORITHM:

- We first load and visualize the datasets(training and testing data) which had been provided to us.
- Test data appears to be following the trend of the training data.



- The training dataset is then partitioned into 10 and each partition is trained for polynomial models with complexity ranging from degree 1 to degree 20

- Each training partition has 800 data points, the given test data has 80 points.

```
# splitting training data into 10 partitions
x_partitioned = np.array_split(x_train,10)
y_partitioned = np.array_split(y_train,10)
```

- Each Model (ploynomial of a degree ranging from 1 to 20) is trained once on each of the 10 partitions.

- The model is trained using sklearn module : **preprocessing.PolynomialFeatures()** and l**inear model.LinearRegression().fit()**

```
## Training each model using sklearn module 10 times.
for deg in range(1,21):
    poly = PolynomialFeatures(deg)

    y_predicted  = np.zeros(shape=(10,80))

    for chunk in range(0,10):

        curr_X = poly.fit_transform(x_partitioned[chunk].reshape(-1,1))
        curr_Y = y_partitioned[chunk]

        reg = linear_model.LinearRegression()
        reg.fit(curr_X,curr_Y)

        X_test = poly.transform(x_test.reshape(-1,1))

        y = reg.predict(X_test)
        y_predicted[chunk] = y

    ##Calculate Mean of Predicted values of each point across the 10 partitions
    mean = np.mean(y_predicted,axis = 0)
```

- The `mean` matrix in the above code snippet, is the Mean of the Predicted values of each data point across the 10 realizations for each model. It is a (1×80) matrix.
- The performance metrics of the model are computed by calculating the bias and variance on each data point across the 10 different realizations of the model.

Bias and Variance: Using the bias and variance formulae given above we compute the bias and variance of each data point.

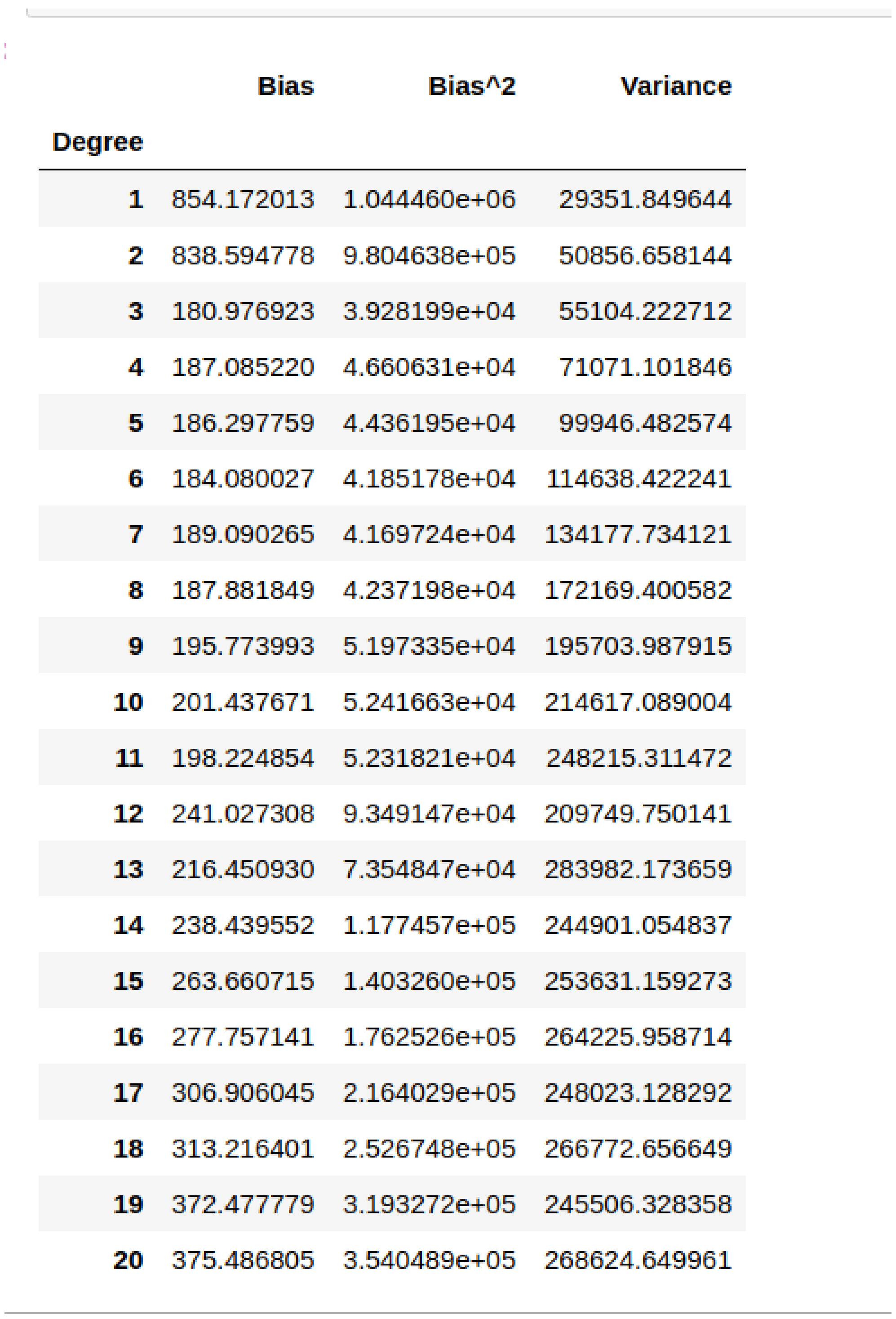
The bias and variance of the model are then calculated as the mean of the bias and variance values calculated for each point.

```
#BIAS:
curr_bias = np.mean(abs(mean - y_test))
bias[deg-1] = (curr_bias)

#BIAS squared:
```

```
curr_bias_sq = np.mean((mean - y_test)**2)
bias_sq[deg-1] = (curr_bias_sq)

#VARIANCE:
curr_variance = np.square(np.subtract(y_predicted,mean))
curr_variance = np.mean(curr_variance,axis = 0)
variance[deg-1]=(np.mean(curr_variance))
```

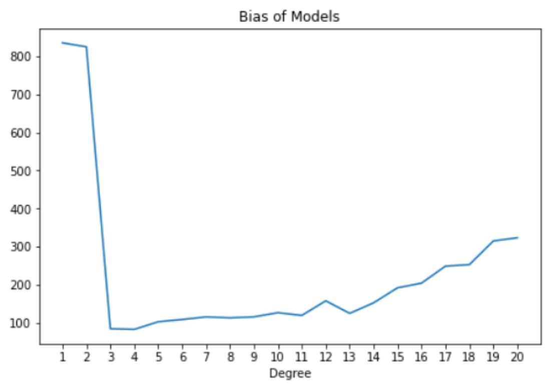
- We repeat the same for all 20 models (complexity ranging from degree 1 to 20) and store the values in the `bias_sq` ,`bias` and `variance` matrices.
- Note that while calculating the `bias` matrix we use the **absolute** difference between the average of the predicted data and the actual data so that negative and positive errors do not cancel each other out.

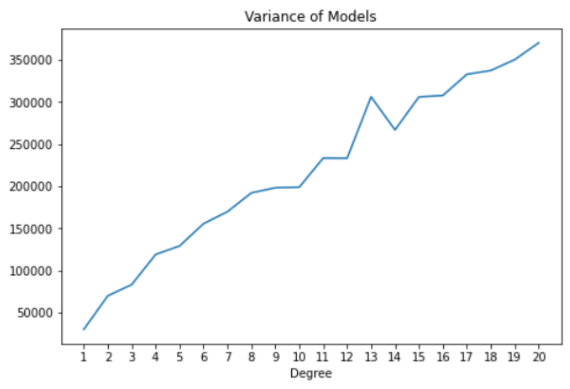
Tabulated Values for Bias and Variance



Degree	Bias	Bias^2	Variance
1	854.172013	1.044460e+06	29351.849644
2	838.594778	9.804638e+05	50856.658144
3	180.976923	3.928199e+04	55104.222712
4	187.085220	4.660631e+04	71071.101846
5	186.297759	4.436195e+04	99946.482574
6	184.080027	4.185178e+04	114638.422241
7	189.090265	4.169724e+04	134177.734121
8	187.881849	4.237198e+04	172169.400582
9	195.773993	5.197335e+04	195703.987915
10	201.437671	5.241663e+04	214617.089004
11	198.224854	5.231821e+04	248215.311472
12	241.027308	9.349147e+04	209749.750141
13	216.450930	7.354847e+04	283982.173659
14	238.439552	1.177457e+05	244901.054837
15	263.660715	1.403260e+05	253631.159273
16	277.757141	1.762526e+05	264225.958714
17	306.906045	2.164029e+05	248023.128292
18	313.216401	2.526748e+05	266772.656649
19	372.477779	3.193272e+05	245506.328358
20	375.486805	3.540489e+05	268624.649961

Graphs of Bias and Variance for all 20 models:





Bias :

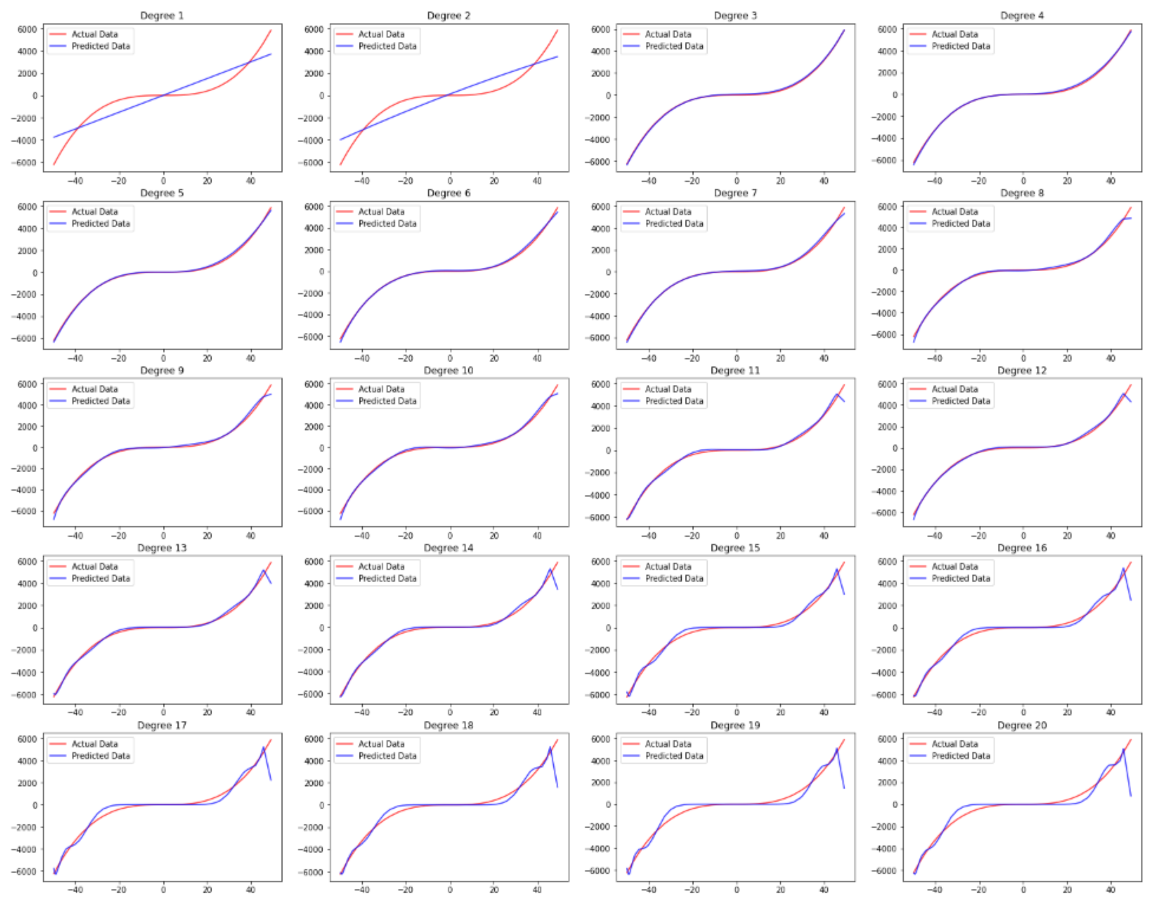
- At degrees less than 3 (linear and quadratic functions) the model underfits both the training data and testing data hence the high bias (which is the average difference between actual and predicted values)
- At degree 3 we see a steep decline in the bias as the cubic function fits the testing data almost perfectly.
- The given Test dataset is a simple cubic polynomial with very little noise. (Refer to Task 3)
- As the complexity of the model increases we see an increase in bias. This is because with the increase in number of features the model ends up overfitting the training data and accounts to the outliers as well. But since the testing data is a simple cubic function with very little noise the higher degree models do not perform well on the dataset giving greater errors in predicted values which is translated to the higher bias values we get.

Variance :

- The general trend of the variance is increasing with increase in degrees of the polynomials.
- As the complexity of the model increases with increase in degree leading to overfitting on each of the 10 realizations, the overall variance would increase.

- This is because as the degree of the polynomial increases and becomes more flexible, it also becomes susceptible to minor variations in the training dataset.
- Hence each time the model is trained, the increased flexibility causes the coefficients of the features to turn out significantly different due to differences in each realization of the training set. Hence the high variance in the predicted model of the test set.

Model Performance on Test Data



- These Plots show the performance of each of the 20 models on the test dataset. Because we have 10 different realizations of each model, we have used the average of the predictions over all the 10 sets.

- As we can see for lower degrees (1 and 2) the predicted values vary significantly from the actual values due to lack of flexibility in the model, hence giving us high error between predicted and actual values of test data.
- However, as the degree of the model increases we see that the predicted values closely follow the actual values thereby giving a low bias. (till degree 5)
- As the models get more complex however the predicted values start to not fit the actual test values well due to overfitting the training data.

Task 3

When predicting a value Y using a set of features on input variables $\mathbf{x}$ = (x_1, ..., x_r),

$$Y = f(x) \ + \epsilon$$

there will be some error ϵ that accounts for all the unmeasured influences on Y. This is termed as the **irreducible error**. It arises from the fact that $\mathbf{x}$ doesn't completely determine Y, and is a measure of the amount of noise in the data. It cannot be removed.

The total error for a given dataset (the mean square error) is

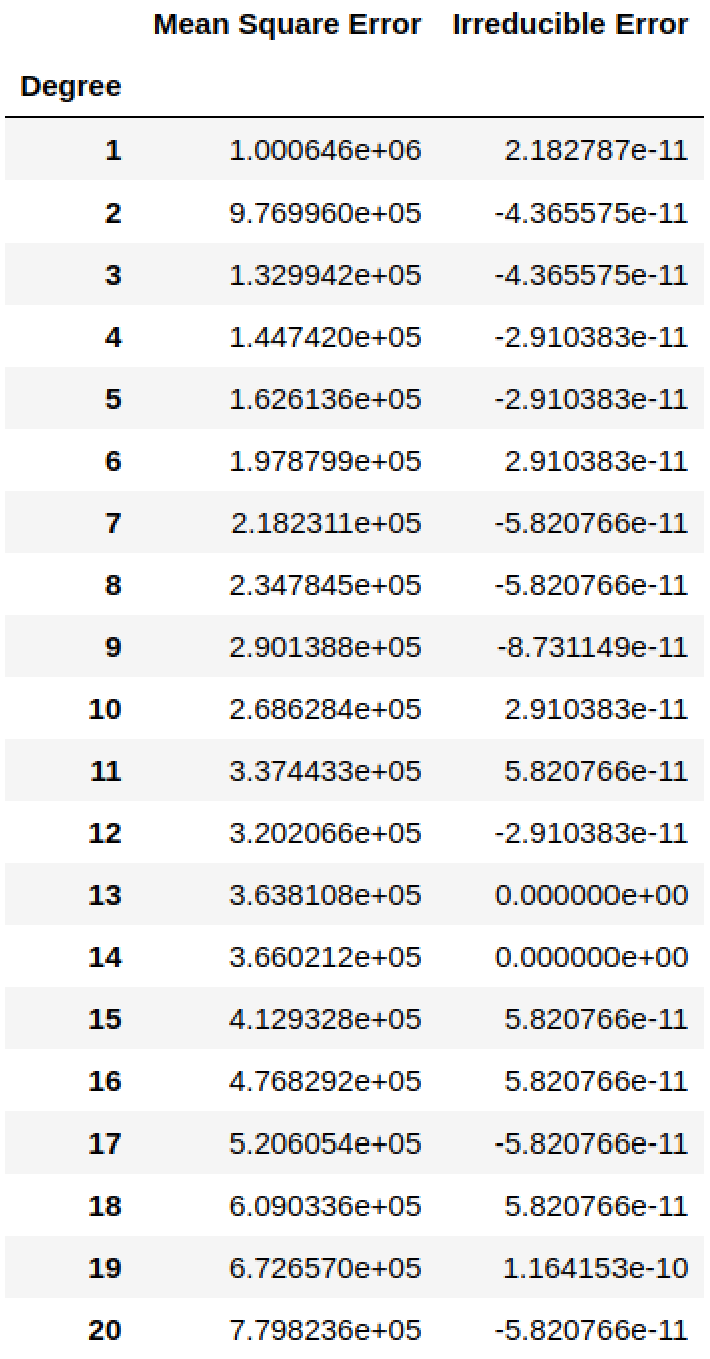
$$MSE = Bias^2 + \sigma_\epsilon^2 + Variance$$

and here, σ_ϵ^2 represents the irreducible error.

By calculating mean square error for test data points, we were able to find the irreducible error for each class function.

```
#MSE:
curr_mse = np.square(np.subtract(y_predicted,y_test))
curr_mse = np.mean(curr_mse,axis = 0)
mse[deg-1] = np.mean(curr_mse)

#IRE:
ire[deg-1] = mse[deg-1] - bias_sq[deg-1] - variance[deg-1]
```

Tabulated Values for Irreducible Error -

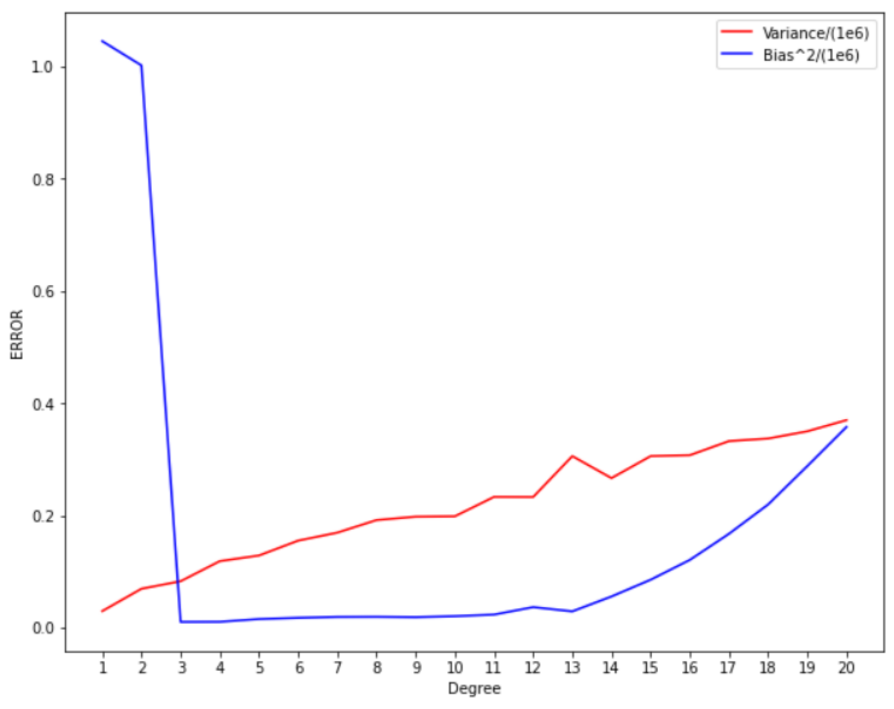


Degree	Mean Square Error	Irreducible Error
1	1.000646e+06	2.182787e-11
2	9.769960e+05	-4.365575e-11
3	1.329942e+05	-4.365575e-11
4	1.447420e+05	-2.910383e-11
5	1.626136e+05	-2.910383e-11
6	1.978799e+05	2.910383e-11
7	2.182311e+05	-5.820766e-11
8	2.347845e+05	-5.820766e-11
9	2.901388e+05	-8.731149e-11
10	2.686284e+05	2.910383e-11
11	3.374433e+05	5.820766e-11
12	3.202066e+05	-2.910383e-11
13	3.638108e+05	0.000000e+00
14	3.660212e+05	0.000000e+00
15	4.129328e+05	5.820766e-11
16	4.768292e+05	5.820766e-11
17	5.206054e+05	-5.820766e-11
18	6.090336e+05	5.820766e-11
19	6.726570e+05	1.164153e-10
20	7.798236e+05	-5.820766e-11

- The irreducible error for the given data is extremely negligible (of order 1e-11).
- As the class function varies, the irreducible error varies in the range of (-5∗1e-11 to +5∗1e-11) which are extremely minuscule amounts. As the irreducible error is caused due to noise in the data and does not depend on the models, it would remain the same across all class functions.
- Small fluctuations are due floating-point exception errors that arise while calculating the means.
- The negative irreducible error values we observe is because the fluctuations can be positive or negative with respect to the irreducible error which is close to 0.

Task 4

The $Bias^2 - Variance$ tradeoff graph -



- For degrees 1 and 2, the bias is extremely high meaning the class function for those degrees underfit the data. The model was not an effective representation of the training data. The variance was low as the model did not fit the train data well and was a bad fit across all realizations, and so the predictions were equally poor among all realizations leading to low variance.

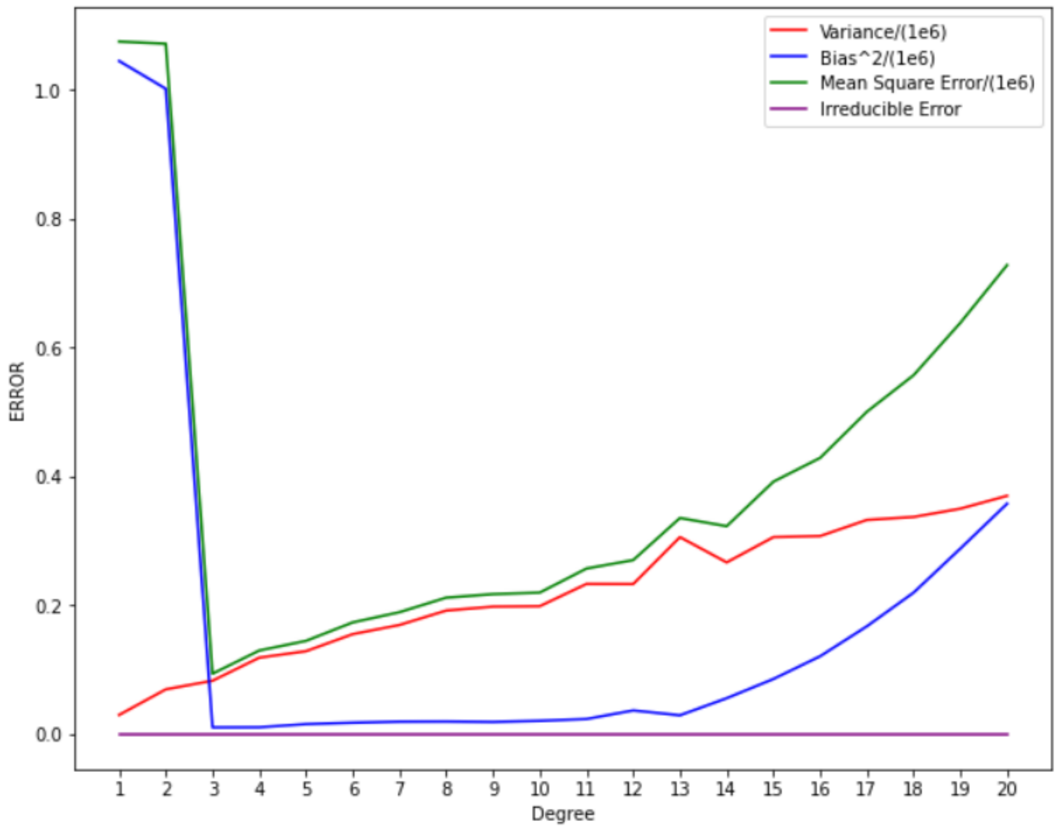
- **At degree 3, the Mean Square Error is at its lowest as shown in the graph below, meaning that degree 3 is the optimal model complexity**

- As the model complexity continues to increase (higher degrees till around 10), the $Bias^2$ curve tends to remain near zero, representing low bias values. The $Variance$ curve however tends upwards and continues to increase with higher model complexity. This is because as the degrees increase, the models begin overfitting to the training data, resulting in less accurate predictions on the test data. Due to overfitting among the different realizations for each class function, the variance, which is how much predictions for a given point vary between different realizations of the model, starts to increase and the curve tends upwards.

- As the complexity of the model increases, the models begin overfitting to the training data and so the test data which is found to be a simple cubic polynomial will face higher magnitude of errors due to the higher bias. Hence the $Bias^2$ tends upwards.

The graph below shows $Variance, Bias^2, Irreducible\ Error$ and $Mean\ Square\ Error.$ As explained in Task 3, Irreducible Error is negligible when compared to the other reducible errors.

Essentially MSE is the sum of $Bias^2$ and $Variance$, as depicted in the graph.


Variance/(1e6)
Bias^2/(1e6)
Mean Square Error/(1e6)
Irreducible Error
ERROR
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Degree