Machine, Data and Learning - Assignment 1

Report

Submitted by
Gokul Vamsi Thota, 2019111009
T H Arjun, 2019111012

Task 1: *LinearRegression.fit()*:

The LinearRegression().fit() method's main purpose is to facilitate the training of the model with the training data set, so that it tries to choose the best-fit based on the same dataset used for training. In turn, this fitting helps in testing the model, by making it predict output values for the testset.

This method takes two parameters: X and Y (optional third parameter: sample_weight). The parameter X is array-like, sparse-matrix of shape (n samples, n features), which typically represents the training data. This parameter X might vary based on the regression to be performed, such as, simple linear regression, multiple linear regression, polynomial regression etc. The parameter Y also has a similar array shape, of the form (n samples), or (n samples, n targets). Y comprises the target values for the corresponding parameter X (in the same training set). The sample_weight parameter depicts individual weights for each sample with the help of an array of form (n samples), but its default is set to None.

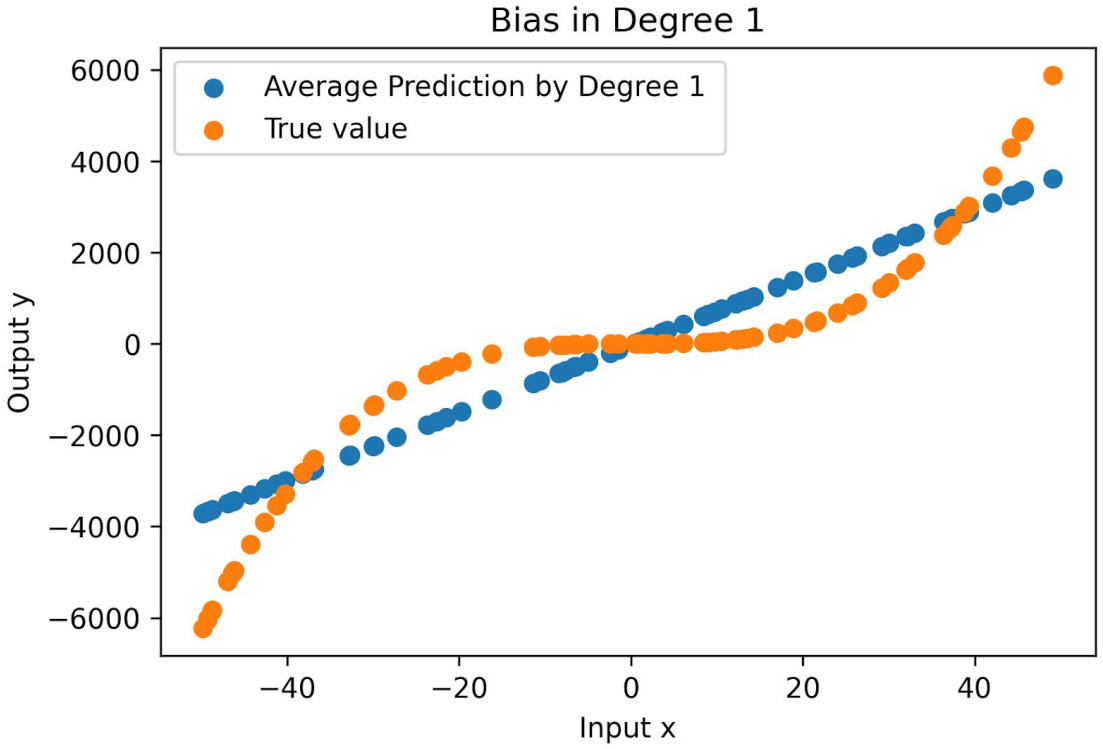
On receiving these necessary parameters, this method fits the linear model, by calculating the necessary coefficient values in an attempt to map the input variables X (say) in the training dataset to its corresponding output variables Y (say) (to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation). Thus, these coefficient values can be used to predict the behavior of the model for different testset data. This method eventually returns an instance of self, now the model is trained to predict values of output variables based on input variables.

Task 2: Analysis of Bias And Variance

Degree	Bias	Bias Square	Variance
1	819.536837	671640.627183	31446.640822
2	810.419904	656780.421087	43644.615397
3	68.868145	4742.821342	52881.438577
4	76.103286	5791.710095	66135.768280
5	75.791341	5744.327348	78029.114438
6	76.678197	5879.545953	117180.313054
7	85.211834	7261.056695	137420.226662
8	91.660264	8401.604088	152262.052291
9	94.327970	8897.766009	205138.596591
10	103.893264	10793.810393	201351.578983
11	97.237432	9455.118111	193563.521702
12	129.826059	16854.805722	218101.710815
13	99.199317	9840.504469	194508.024436
14	131.794409	17369.766158	178200.423456
15	173.234170	30010.077747	173314.665927
16	177.578003	31533.947048	171030.374842
17	249.132495	62067.000050	178875.624953
18	250.484016	62742.242348	184509.213076
19	318.146435	101217.154353	197829.216203
20	318.143280	101215.146490	216365.514813

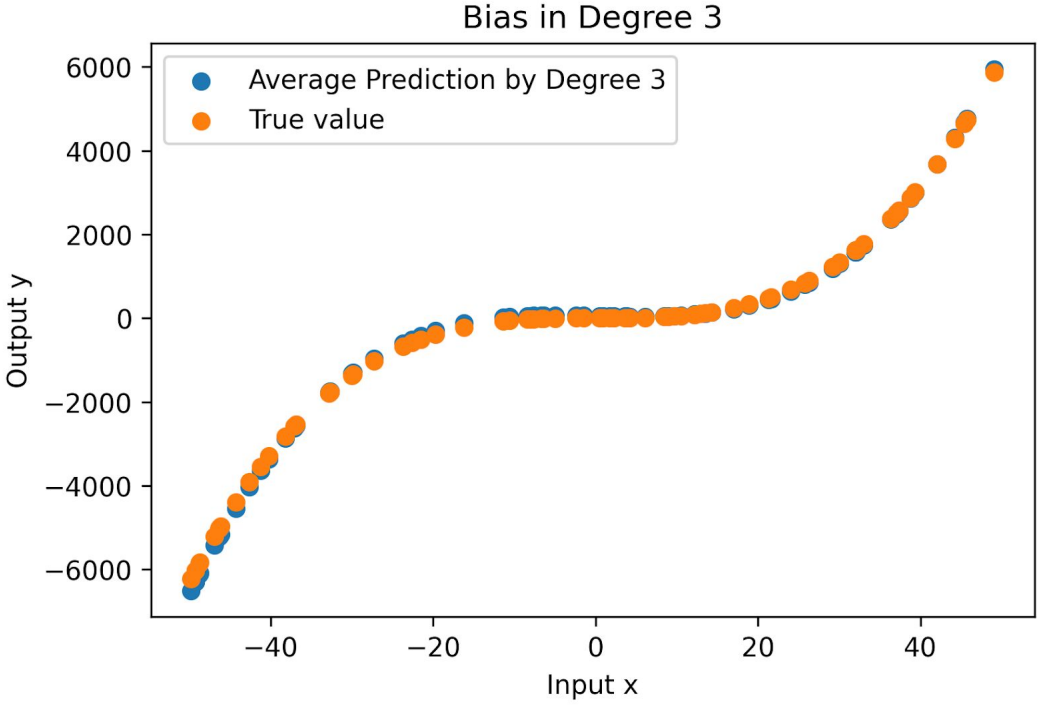
Bias is the difference between the average prediction of our model and the correct value which we are trying to predict for a data point. Identifying the mapping between input and output variables is essential in training the model. The more erroneous the predicted mapping between input and output variables, the higher the bias, and vice-versa. This explains the high bias for lower degree polynomials in the table. Because the given data cannot be represented in those degrees, which leads to oversimplification in representation (underfitting). This leads to differences in average prediction of the model and the correct value (which is depicted by the high bias values in the below plot).

This is shown in the plot below for function of degree 1 (worst case observed):



The test data (shown in blue) is represented by a curve. We tried to model this curve using the function f(x)= mx+c. This led to oversimplification due to insufficient training of the model because of which values predicted by our model were far off from the true value of the test data, leading to bias (it explains the high bias at the top of the table).

Now checking at the plot below for degree 3 (best case observed):



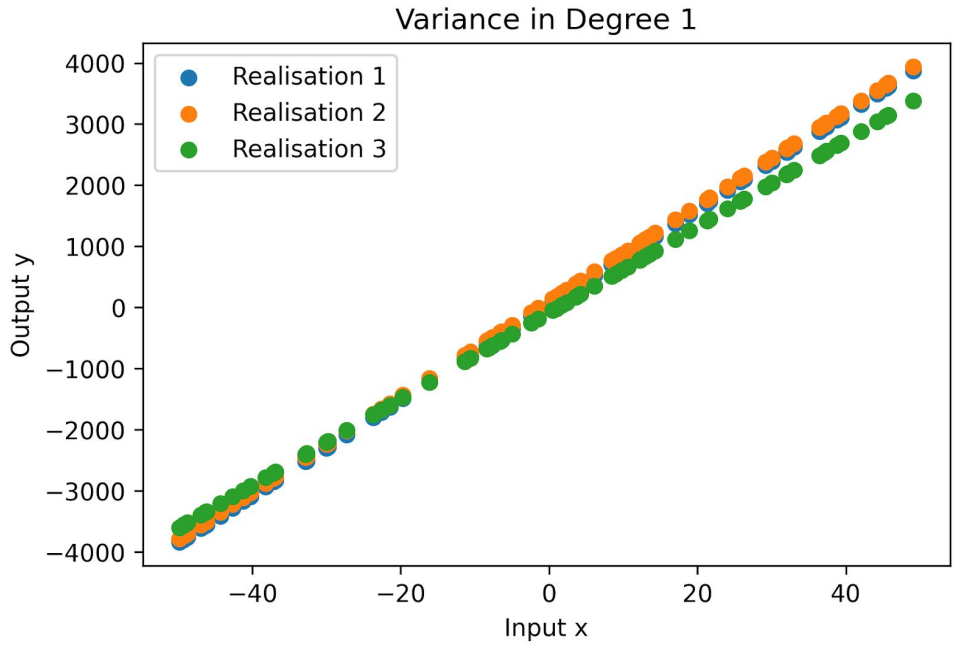
The degree 3 function was able to fit the test data in the best way among all other functions, thus the predicted values of the model were in the proximity of true value of test data, which explains the low bias values observed.

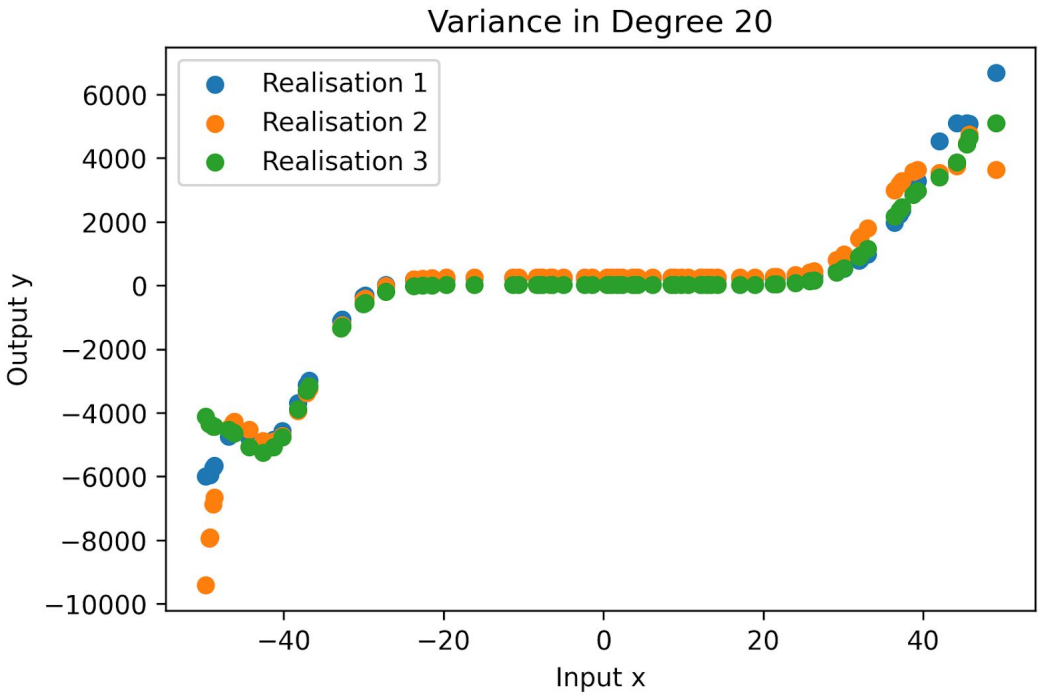
In the functions of relatively higher degrees, the bias observed is slightly higher because a big change in the output variables is observed even if the input variables changed by a very small amount, leading to overshooting and improper fitting of data.

Generally, high bias implies underfitting, meaning the model is not able to capture all the details of the relationship between input and output, probably due to insufficient training. Similarly, low bias implies overfitting where the model is too accurate for the current training set that when the data is changed, the output varies by a significant amount. So we obtain different outputs across different realisations of the same model.

This leads us to Variance. Variance is the variability of a model prediction for a given data point. It depicts the variation in a model's prediction for a given data point, if the model is constructed / trained differently several times. In the above table, we can observe that the variance increases as we move downwards (increasing degree of functions). This is because of overfitting (as explained above) which makes it work only for a specific data set. It loses generality and predicts variable outputs for various realisations of the same model, over different training sets. This further explains the increasing variance downwards in the table (as degree increases), because of loss of generality due to overfitting.

This can be observed in below plots:





The plot for function with Degree 1, has low variance (best case observed), that is, the predictions over various realisations of the model produced similar outputs. It is highly generalised, leading to underfitting, where it is unable to capture all the details, thus explaining high bias and low variance observed.

Now in plot for degree 20 (worst case observed), The model was fitted very specifically to the training set, that it showed varying outputs across various realisations of the model, which explains high variance observed.

Task 3: Irreducible error

Degree	Bias	Bias Square	Variance	Mean Squared Error	Irreducible Error
1	819.536837	671640.627183	31446.640822	1.036087e+06	5.093170e-11
2	810.419904	656780.421087	43644.615397	9.972201e+05	-1.018634e-10
3	68.868145	4742.821342	52881.438577	6.224556e+04	-1.455192e-11
4	76.103286	5791.710095	66135.768280	7.503330e+04	1.455192e-11
5	75.791341	5744.327348	78029.114438	8.673356e+04	-1.455192e-11
6	76.678197	5879.545953	117180.313054	1.265308e+05	-2.910383e-11
7	85.211834	7261.056695	137420.226662	1.477539e+05	0.000000e+00
8	91.660264	8401.604088	152262.052291	1.642435e+05	0.000000e+00
9	94.327970	8897.766009	205138.596591	2.181367e+05	0.000000e+00
10	103.893264	10793.810393	201351.578983	2.191869e+05	2.910383e-11
11	97.237432	9455.118111	193563.521702	2.091344e+05	0.000000e+00
12	129.826059	16854.805722	218101.710815	2.534959e+05	0.000000e+00
13	99.199317	9840.504469	194508.024436	2.172396e+05	0.000000e+00
14	131.794409	17369.766158	178200.423456	2.217571e+05	-2.910383e-11
15	173.234170	30010.077747	173314.665927	2.424477e+05	0.000000e+00
16	177.578003	31533.947048	171030.374842	2.490420e+05	5.820766e-11
17	249.132495	62067.000050	178875.624953	3.051981e+05	0.000000e+00
18	250.484016	62742.242348	184509.213076	3.191331e+05	2.910383e-11
19	318.146435	101217.154353	197829.216203	4.057135e+05	0.000000e+00
20	318.143280	101215.146490	216365.514813	4.337857e+05	5.820766e-11

The irreducible error is defined and calculated as:
Irreducible Error =
Average Mean Squared Error - (Average Bias)^2 - Average variance,
where average is defined over all the models trained for a particular class of polynomials.

The values displayed in the table for irreducible error are positive and negative for different classes of polynomials, but the values are always tending to 0. The very small negative numbers are because of the float precision errors in calculations.

In general, in an ideal scenario such as here, Irreducible Error will be a constant (which ideally tends to 0) over the various models and polynomials if it is taken on a particular data point, since it represents the error that arises due to the noise in the data. This is because the output might be dependent on other factors different from our input, which our model cannot capture / map. It might also be due to measurement errors which cannot be controlled or reduced explicitly.

This is noise present in the data. This error cannot be reduced by building better models, as this error is independent of the model. Hence, ideally it is a constant which tends to 0. This constant value of irreducible error, is the limiting value of Mean Squared Error. The irreducible error is independent of inputs and training of a model in an ideal scenario.

Task 4: Bias Variance Tradeoff

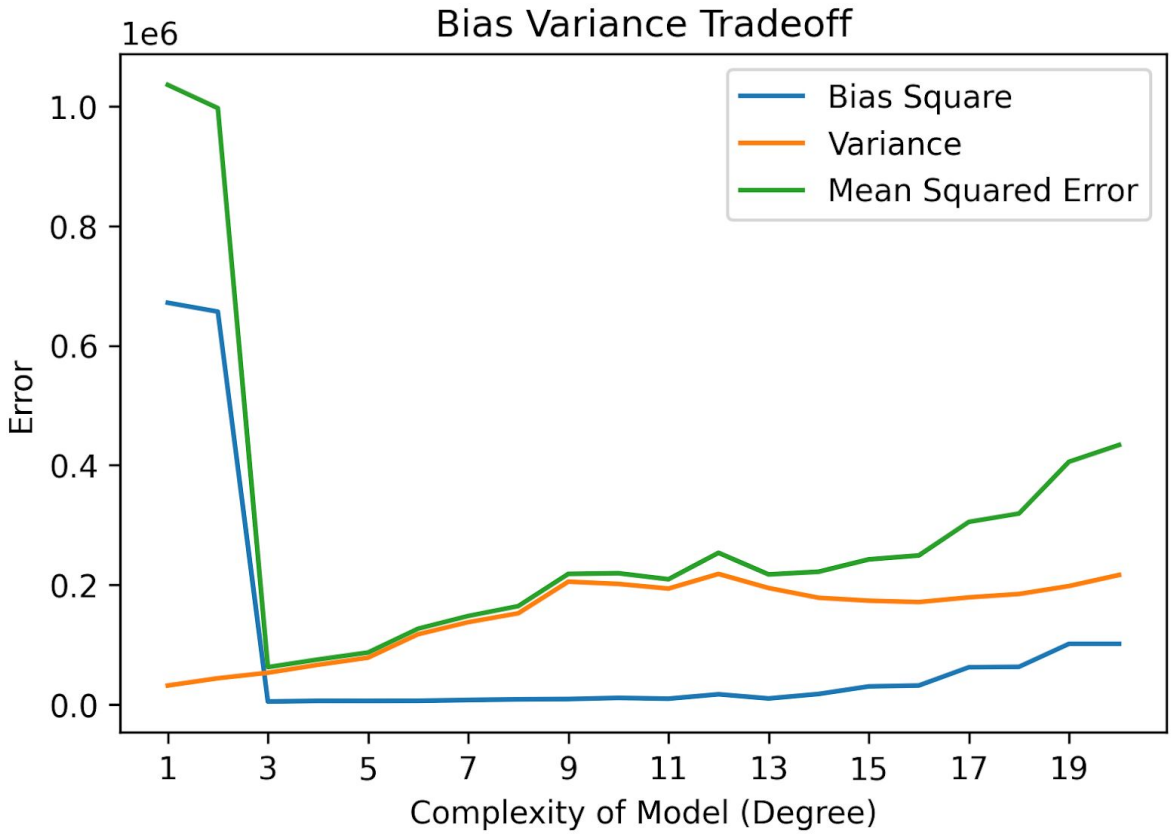


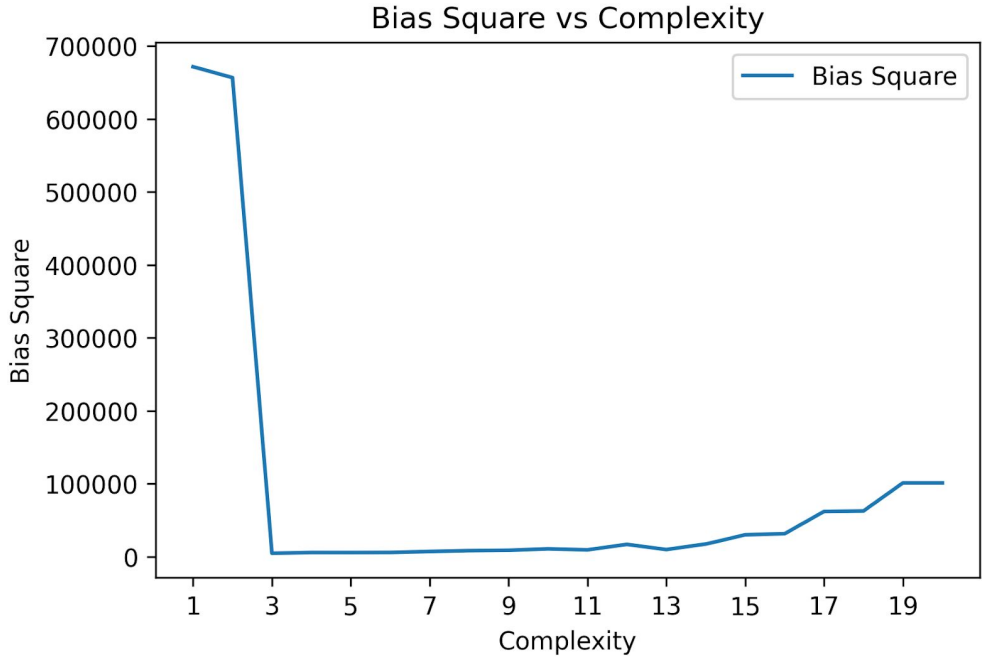
Fig. Plot of Bias Square, Variance and Mean Squared Error vs Complexity

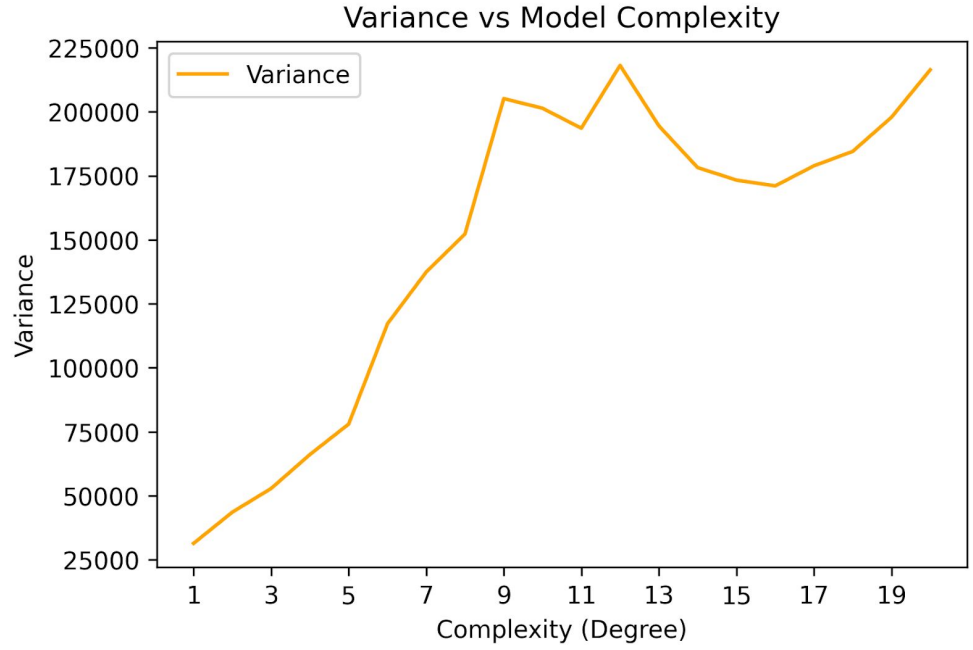
On analysing the definitions of bias and variance, we realize that both the quantities, if present in abundance, will decrease accuracy of the model, that is, increase model's error. As we desire low error, we need low variance and low bias. But this is not easy to attain. A low-biased method fits training data very well. If we test the performance of the model on a different test set, we'll get significantly different outputs from true value. This is a typical example of a scenario in which overfitting occurs, where we observe high variance and low bias. The less biased a method is, the greater is its ability to fit data well. The greater this ability, the higher the variance. Hence, the lower the bias, the greater the variance.

The reverse also holds true. The greater the bias, the lower the variance. A high-bias method builds simplistic models that generally doesn't fit the training data well. As we change inputs, that is, if we train the model differently, the output doesn't change much and the model is not able to capture all details. This indicates high bias and low variance, a typical scenario in which underfitting occurs. We desire a model which produces low bias. Generally, a simple model performs poorly, that is, it gives a very varied prediction of output variables with respect to their true value.

If we aim for low variance, and design a simplistic model to accommodate the training dataset, it would cause high bias because of not being able to fit data well. If we desire low bias and design a model specific to the training set, it would fail on different test datasets due to loss of generality, and there would be a high variance observed in terms of accuracy.
Thus, both bias and variance are inversely proportional. We need to strike a balance between the two to build an ideal model. This is known as the Bias-Variance tradeoff. We can't have both low bias and low variance, so we aim for an optimal solution which incorporates both.

The observed plots below show this inverse relations:





(The explanation about the nature of the above two plots is mentioned in detail in 'Analysis of Bias and Variance' section above.)

Total Mean Square Error is the sum of Variance, Bias square and Irreducible Error (all these terms being positive). Irreducible Error does not change across models. It is a constant. Hence, that is the limit of reducing total error.

As Variance and Bias square are inversely proportional, we need to optimize both such that Total Mean Square Error is minimal. As the first plot shows, Mean Squared Error follows the trend of a bell curve. The minima of that graph depicts the perfect values for bias and variance, which represent an ideal model.

In this assignment, we found the perfect spot for the data given to be at degree 3, where the Mean Square Error is minimum. In the plot it is clear that the minima is at degree 3 and the corresponding values of bias and variance at degree 3 is the sweet spot for data. Hence, the best fit is observed with degree 3 in the given polynomials, because the tradeoff between bias and variance is most optimal here.

Based on the graph, we can also comment that the class of polynomials with degree 1 have low variance, but high bias, because of insufficient training and tendency of being much more generalized than needed. Similarly, the class of polynomials with degree 20 are observed to have very high variance, but relatively low bias, because of the extremely specific nature of the trained model to its corresponding training dataset, and loss of generality.