Machine, Data and Learning - Assignment 1

Ayushman Panda (2020121007)
Pratishtha Abrol (2020121002)

Algorithm followed:

Splitting the data:

The dataset given already had data divided into train and test datasets. The train dataset was further divided into 10 equal sets to build 10 different classifiers for each degree class.

The dataset is loaded into the program using the pickle.load() function.

The 200 models were created by putting into a loop for model index within the degree number. We note each model's output, and coefficient of determination. Each model is plotted to see it's output against the test data.

Calculating Bias and Variance:

> Bias:

Bias is the difference between the average prediction of the model and actual value of the test dataset. Models that are overfitted are characterised by a low bias, and those underfitted have a high bias. As more parameters are added, that is, with increase in model complexity the bias decreases.

Formula :

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

> Variance:

Variance is the variability of the model prediction for a given datapoint, that is, error due to small fluctuations in the training set.

Formula :

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

> Bull's Eye graphs for Variance and Bias:

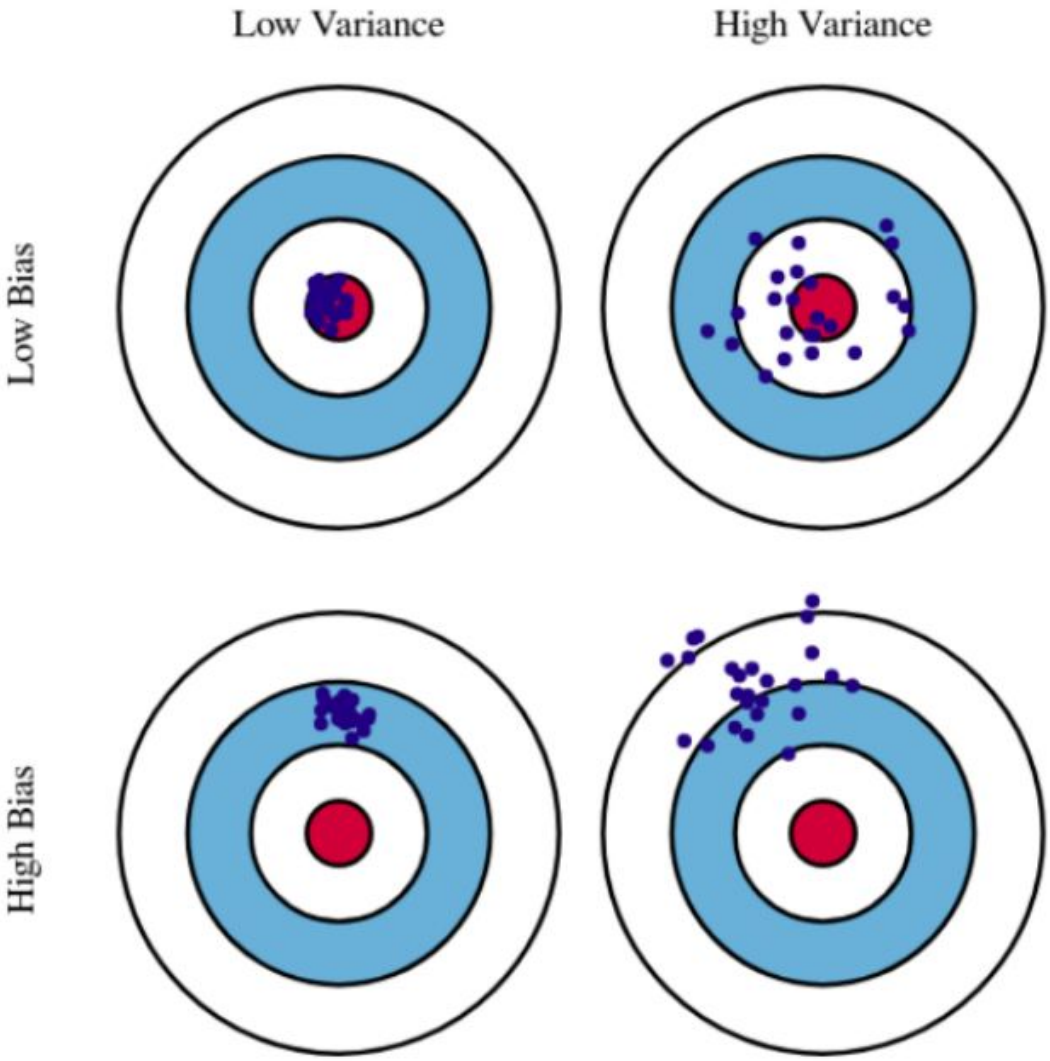


Figure 1: Graphical illustration of bias and variance.

Main Function:

The models are trained on the 10 different datasets for polynomials ranging from x to x^{20}
For each model, the training is done using *sklearn.linear_model.LinearRegression.fit()*
Once trained, we test it on the test dataset.

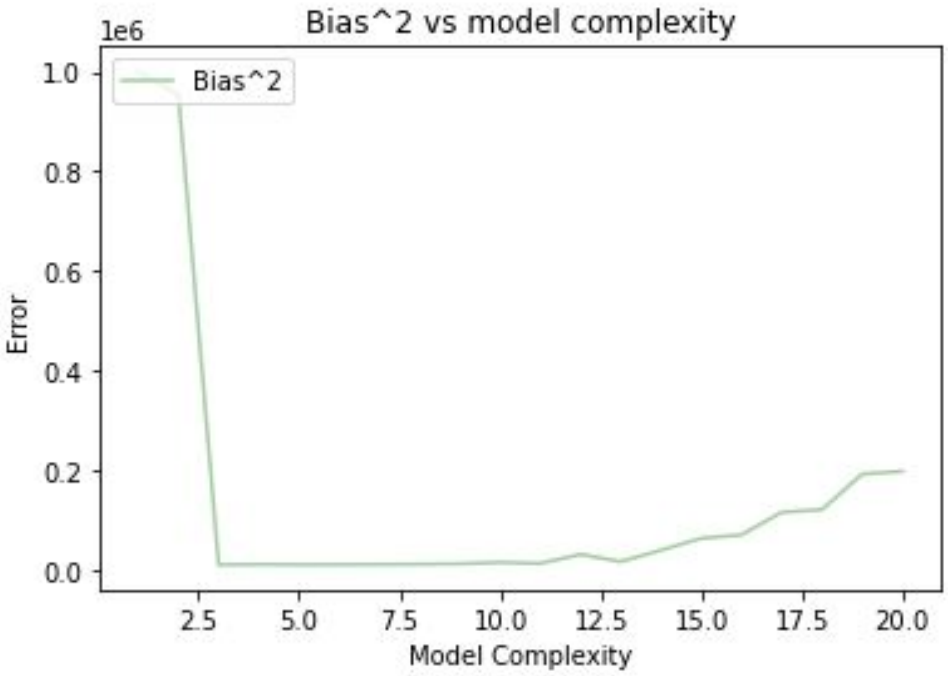
```
all_bias=[]
all_bias_sq=[]
all_var=[]
all_err=[]
for degree in range (1,21):

    local_err=[]
    output=[]
    print("\n\n\t\t\t\t\tPolynomial of degree: ",degree)
    for model_index in range (10):
        print("\n\n\t\t\t\t\t\tModel: ",model_index+1)
        print("\n")
        predict = higherdegreemodel(degree, train_dataset[model_index][["X"]], train_dataset[model_index][["y"]])
        title = "Degree: %s, Model: %s" % (degree, model_index+1)
        plot(title, predict)
        local_err.append(np.mean(np.square(test[["y"]] - predict)).y)
        output.append(predict)

    predicted_point_mean = np.mean(output, axis=0)
    all_bias.append((np.mean(abs(predicted_point_mean - test[["y"]]))).y)
    all_bias_sq.append((np.mean((predicted_point_mean - test[["y"]])**2)).y)
    predicted_point_var = np.var(output, axis=0)
    all_var.append(np.mean(predicted_point_var))
    all_err.append(np.mean(local_err))
    print("\n
-------------------------------------------------------------------------------------------------------------------------\n")
    print("\nMean Bias of degree ", degree, ": ", (np.mean(abs(predicted_point_mean - test[["y"]]))).y)
    print("\nMean Variance of degree ", degree, ": ", np.mean(predicted_point_var))
    print("\n
-------------------------------------------------------------------------------------------------------------------------\n")
```

Results:

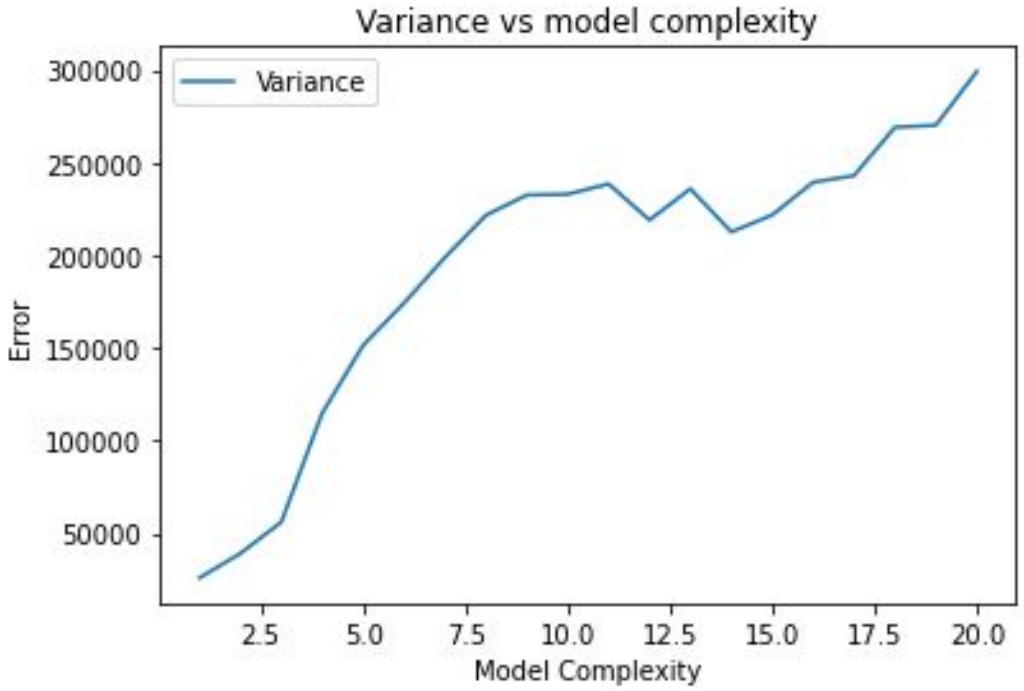
1. Bias and Variance tabulation

	Degree	Mean_Bias	Mean_var
0	1	819.717436	25999.093010
1	2	810.763391	39105.833813
2	3	68.510851	56095.893210
3	4	81.339711	114907.291530
4	5	78.958414	151434.027824
5	6	78.364801	174226.745435
6	7	86.916374	198849.561271
7	8	90.333728	221548.875061
8	9	92.752719	232473.437639
9	10	98.159691	232927.885933
10	11	91.069563	238457.966715
11	12	125.167477	219051.479089
12	13	91.632107	235831.418303
13	14	130.174551	212545.346977
14	15	166.459218	221715.110773
15	16	170.416978	239357.861272
16	17	236.714659	242994.334500
17	18	239.125004	269053.408511
18	19	304.867541	270103.513645
19	20	305.449999	299028.015761

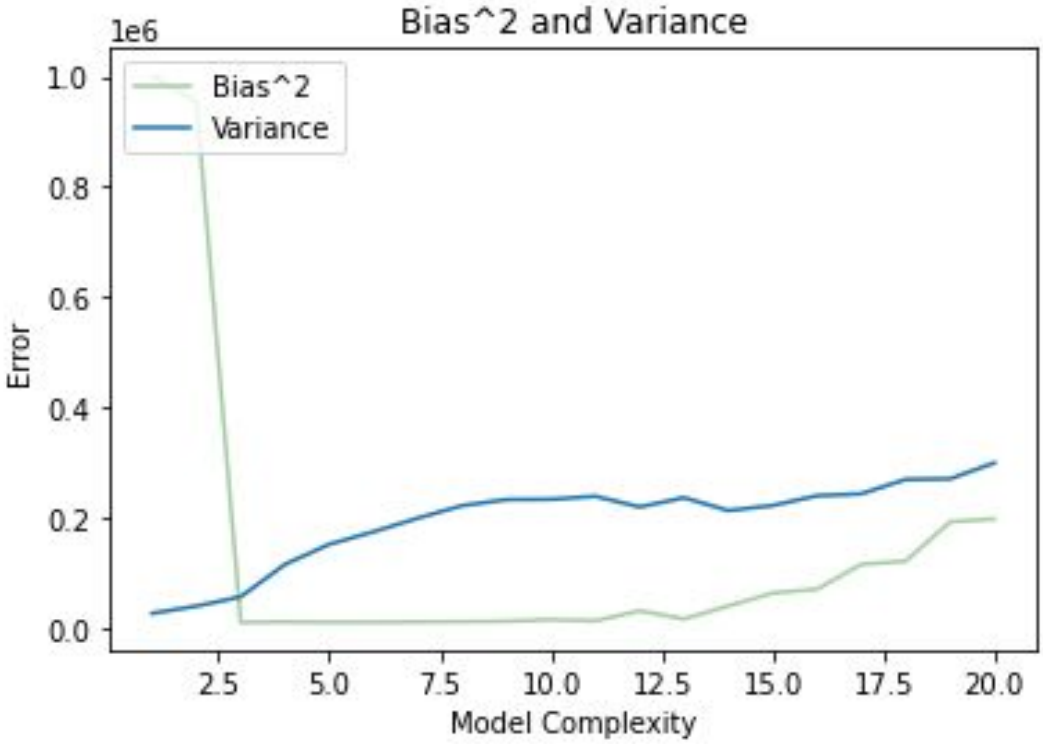
2. Bias Graph



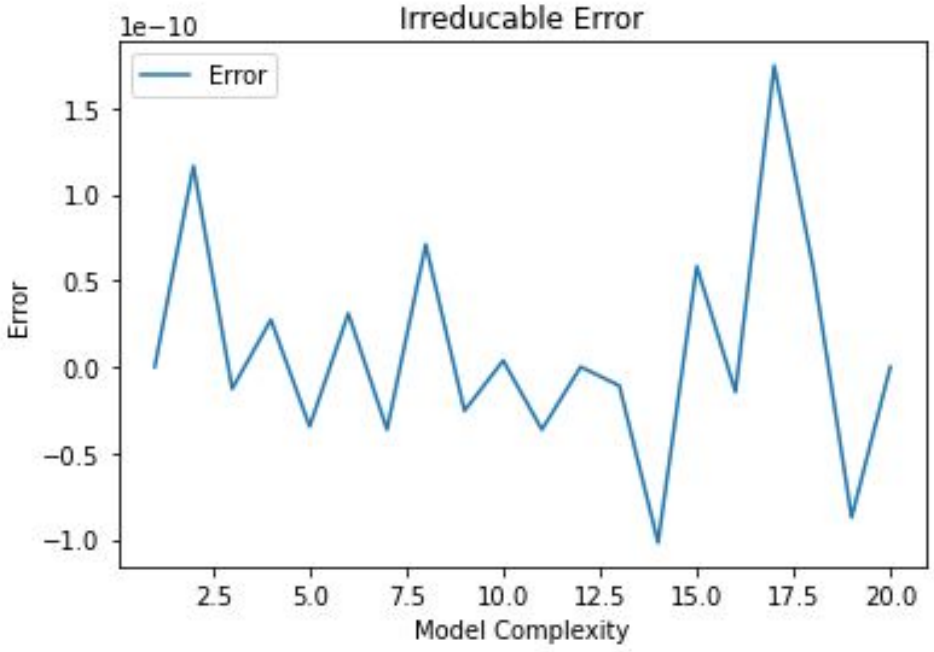
3. Variance Graph



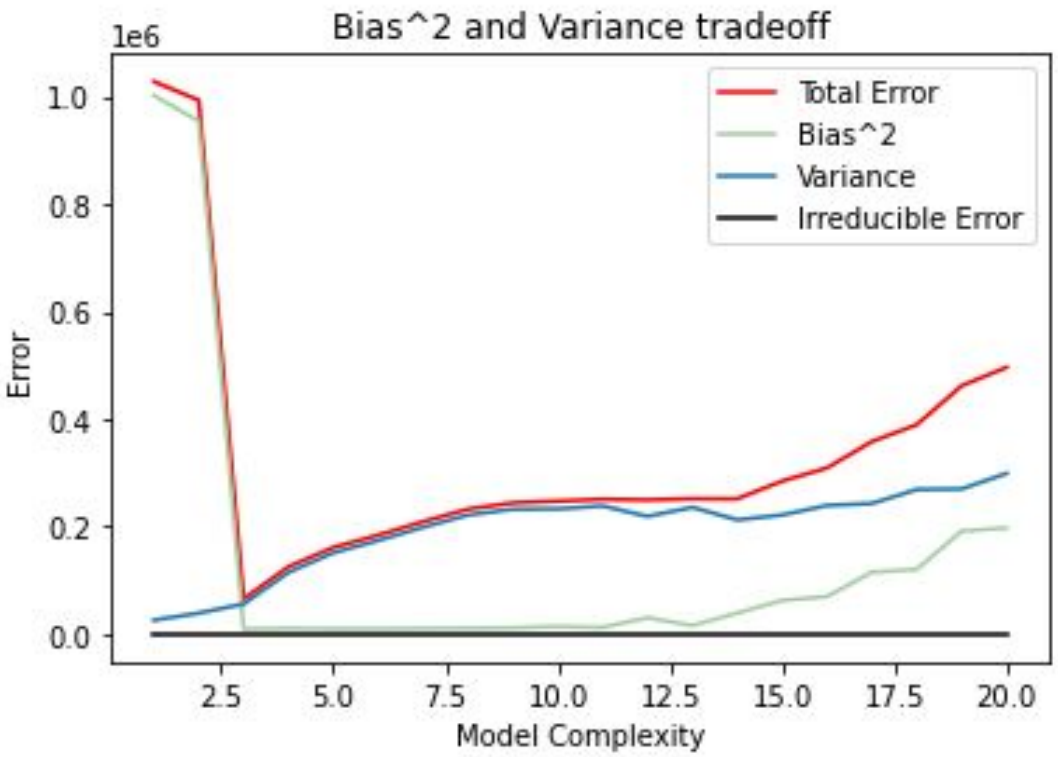
4. Bias vs Variance



5. Irreducible Error



6. Bais-Variance Tradeoff with Total error(MSE) and irreducible error



Observations:

From the above graphs we see that with an increase in model complexity, bias decreases and variance increases.

- With increase in number of features, or complexity of the model, here, by increasing the degree of the function, the model extracts more information from the training set and works well with them. This is characterised by low bias with increase in model complexity. However, at the same time, the model does not generalise well and performs poorly on the test data. This is reflected in the increase in variance with increase in complexity.
- Lower degree polynomial functions have a high bias but low variance. The model may not perform well even on existing training data because of less number of features provided.
- As we see here, the bias, variance and the total error or mean square error is collectively minimised at degree 3 polynomial function, it is the optimum number of features to fit this particular dataset.