Introduction to Machine Learning (CS1390/PHY1390)

Monsoon 2021 - Homework 1

Jenish Raj Bajracharya

Collaborators: None

Regression Analyses

Setting the Scene

1. Three new functions were added. Tangent: $y = tan(x)$, Log: $y = log(x)$, Inverse: $y = \frac{1}{x}$.
2. Plots: 3 plots were created
   (a) Fitting a degree n polynomial in training data set.
   (b) Actual value vs Predicted value. Note: The green line is the $y = x$ line because if we consider realizability then for every $x_i \in \{\text{testing set}\}$, $f(x_i) = y_i = h(x_i)$. Which is the line $y = x$.
   (c) Plotting the testing error and training error with respect to different degree of polynomial.

Function: Line

The domain and parameters of the functions were unchanged. 20 data points were split in the ration of 80:20 for training and testing.

Concept of over-fitting and under-fitting

From figure 1 we can see that the regression model in this case degree 1 is clearly under-fitting the training data as the model cannot represent the entire training data-set. Resulting in high bias but low variance which we can see from the second sub plot .

Similarly, figure 2 is over-fitting the data. All the points in the training data are represented bu using the polynomial of degree 20 but when we use the model with unseen data it results in high error. An overfitted model results in a low bias but with high variance as visible from the second plot as the red dots(training data) are concentrated while the blue dots are spread.

Figure 3 which uses a polynomial of degree 4 scan be said as a nearly perfect fit because the variance and bias is low as the points are scattered around the green line.

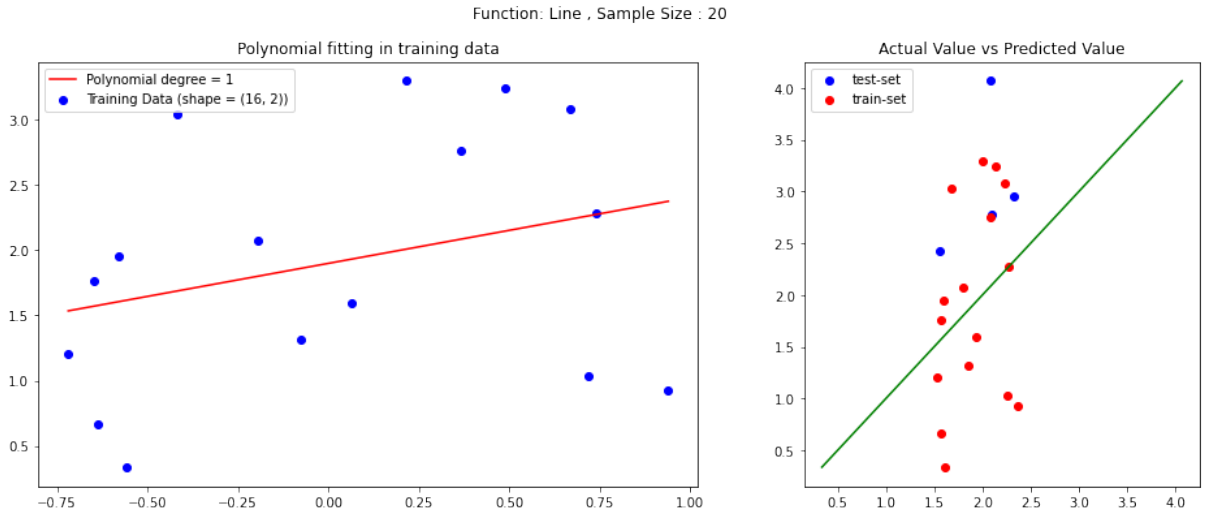


Figure 1: Underfitting the data

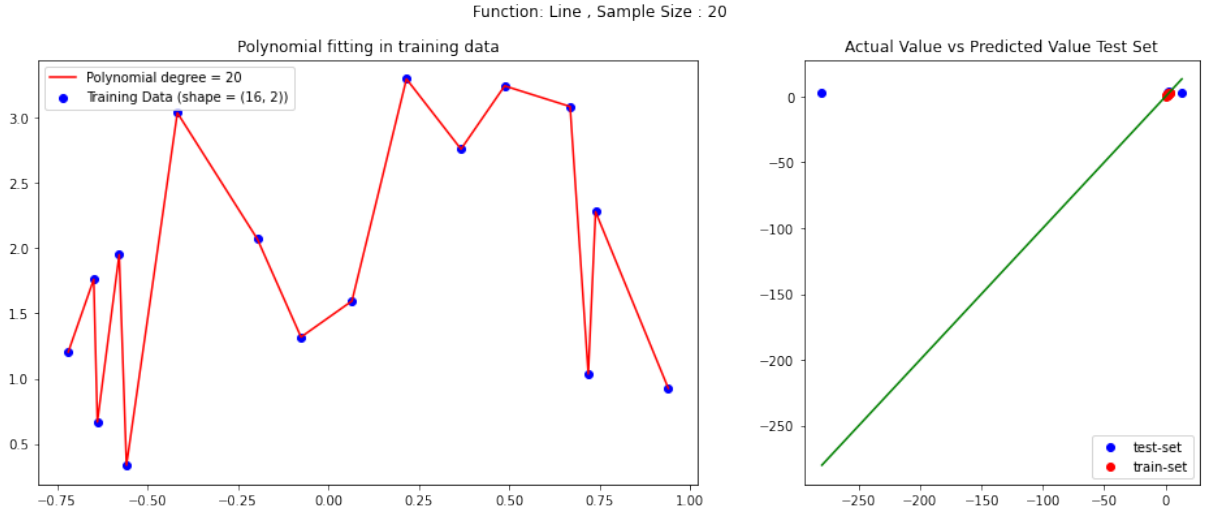


Figure 2: Overfitting the data

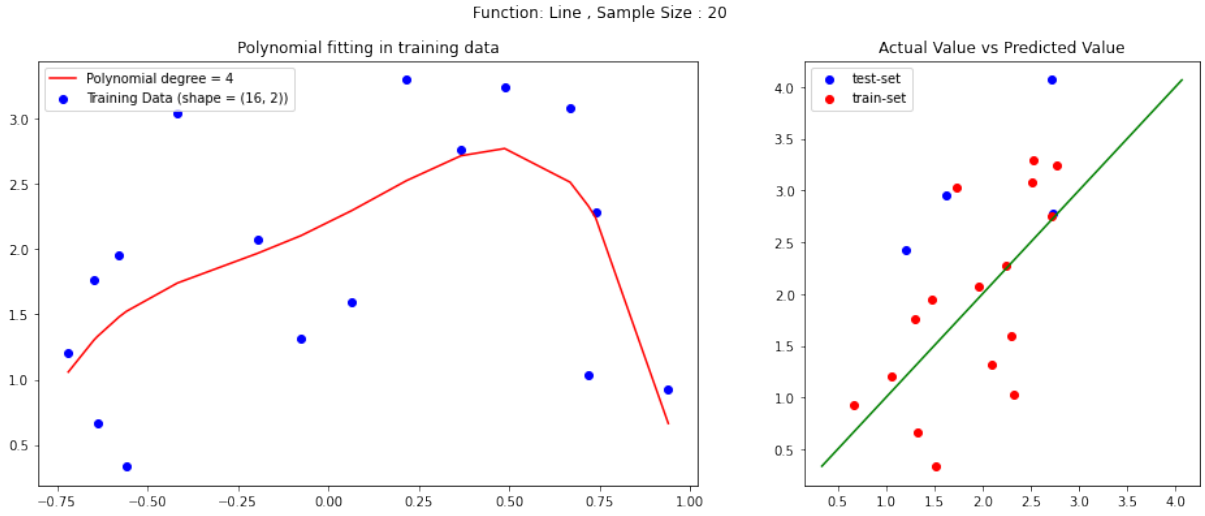


Figure 3: Nearly perfect fit

Concept of Testing Error and Training Error w.r.t polynomial degree

2 sets each with 100 sample set were extracted.

From figure 4 and 5 we can see that as the polynomial degree increases training error decreases while the testing error increase. For Set 0, degrees 12-17 polynomial fits the data as distance between the two errors are less. But for set 1 nothing can be concluded because both errors are inverse to each other.

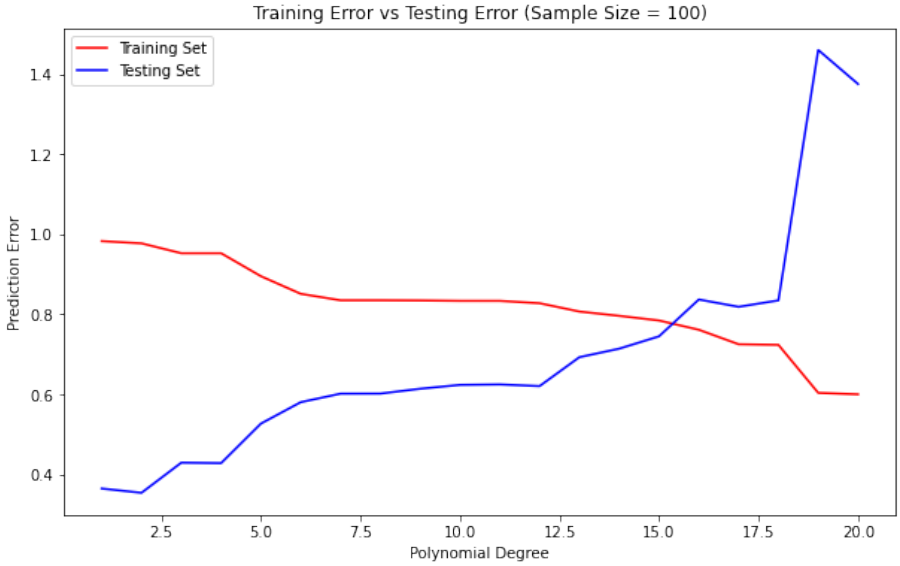


Figure 4: Set 0

Figure 5: Set 1

Function: Exponential

The parameters of the function were unchanged while the domain: $X \in \{-3, 3\}$. 50 data points were generated. Firstly the split ratio was 80:20 then I experimented with a split ration of 20:80 on the same set.

Concept on the size of the training data-set

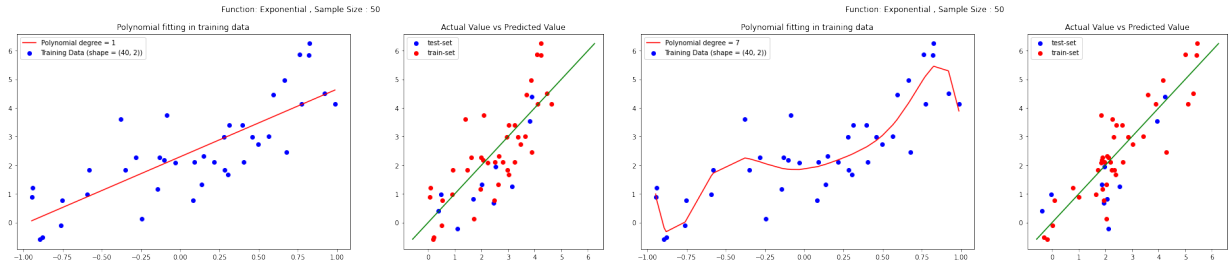


Figure 6: Polynomial degree 1

Figure 7: Polynomial degree 7

Figure 8: 80:20 split

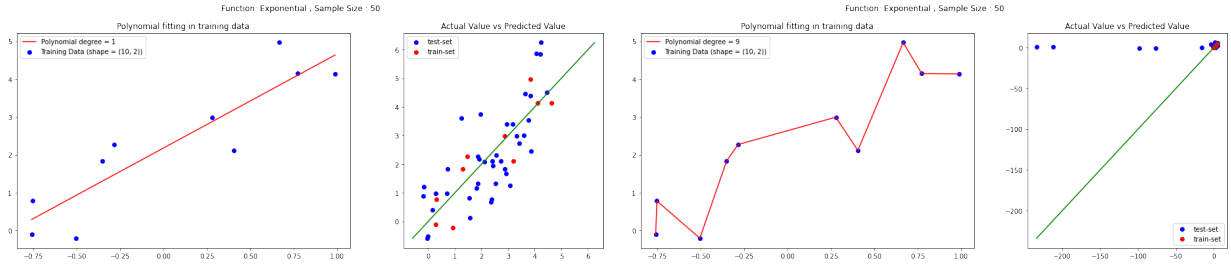


Figure 9: Polynomial degree 1

Figure 10: Polynomial degree 9

Figure 11: 20:80 split

In figure 8 the data is split with a ratio of 80:20 (training set and testing set). From the sub-figures 6 and 7 we can say that the polynomial of degree 1 is under fitting the data set while the degree 4 polynomial fits the data optimally. This is also clear from the actual vs predicted error figure. The actual vs predicted error figure has less spread and bias for polynomial degree 4 than degree 1.

In figure 11 the data is split with a ratio of 20:80 (training set and testing set).From the two sub-figures 9 and 10, we observe that when the training set is small compared to testing set, lower degree polynomial can perfectly fit the data while a higher degree might over-fit the data.

If we look at figures 13, the prediction error for testing data is very high compared to figure 12 regardless of the size of the polynomial being used to fit the data.

This verifies that the size of the training set should be sufficiently big enough to represent the whole data set.

Figure 12: 80:20 split

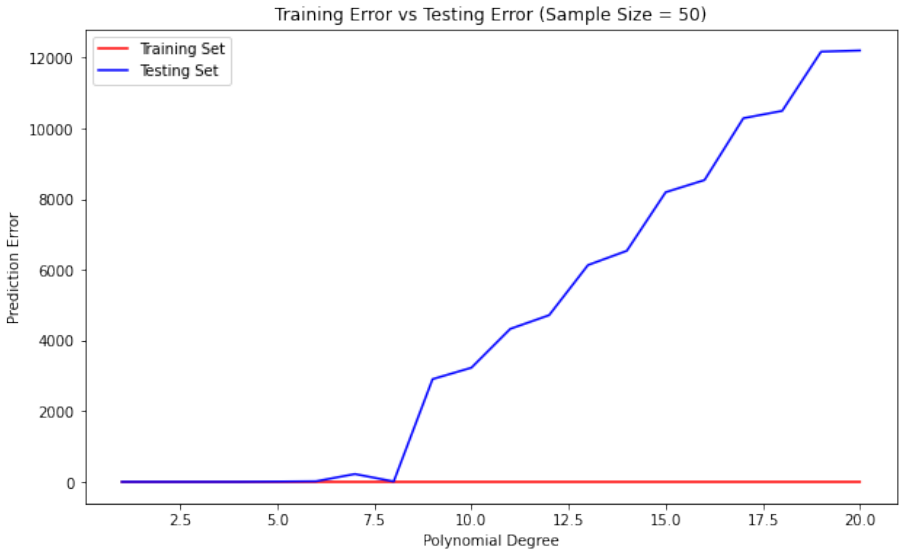


Figure 13: 20:80 split

Function: Trigonometric (sine, cosine, tangent)

For these three trigonometric functions the parameters were kept the same. Two set each of size 100 for domain: $X \in \{-3, 3\}$ were generated.

Sine

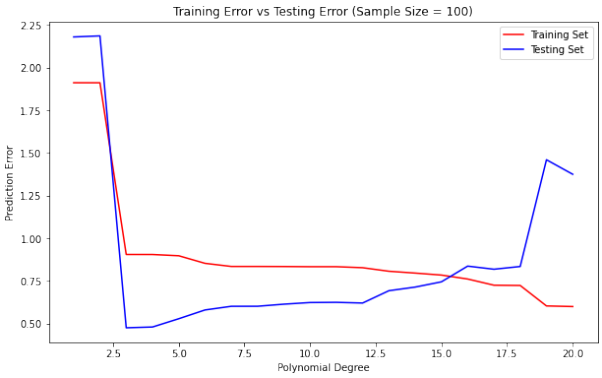


Figure 14: Set 0

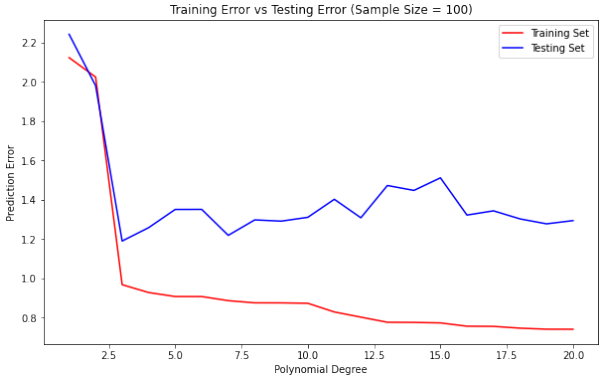


Figure 15: Set 1

For the sine function, both sets of data(figure 14 and 15) have the same trend. Lower degree polynomial results high bias, low variance. As the degree increases both the training error and testing error decreases which results in a approximately perfect model. As the model complexity further increases the testing error increases while the training error decreases because the model starts the over-fit the data. This results in low bias but high variance.
For set 0 polynomial degree $\in [6, 10]$ can be said as a nearly perfect fit because the distance between the training error and testing error is less. This is also visible in figure 16. The test-set and the train set are spread evenly along the $y = x$ line.For polynomial degree ≥ 10, we can say that the we have started to over-fit. This is visible from figure 17 as the training set are clustered around the $y = x$ line while the test sets are spread across.
Similarly for set 1, polynomials with degree $\in [5, 9]$ can fit the model (figure 18) well while degree ≥ 10 starts to increase the variance (figure 19).

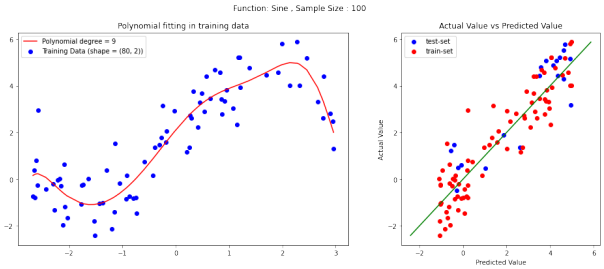


Figure 16: Poly Degree 9

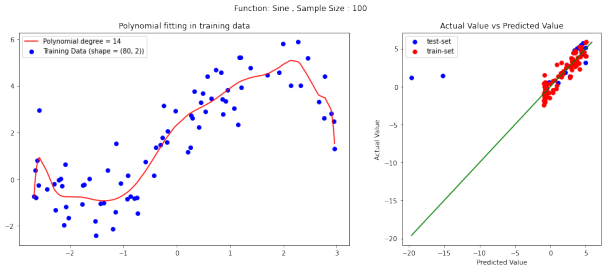


Figure 17: Poly Degree 14

Figure: Poly fit and actual vs predicted score in set 0

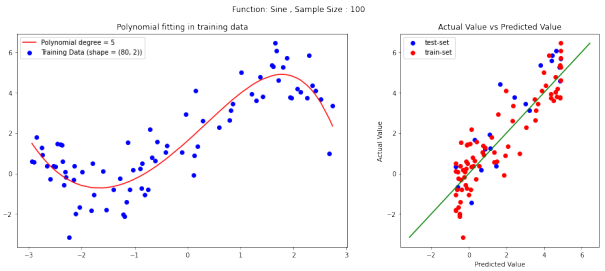


Figure 18: Poly Degree 5

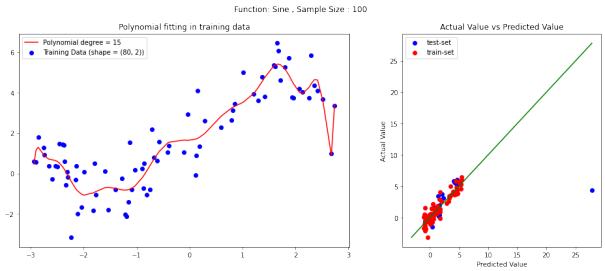


Figure 19: Poly Degree 15

Figure: Poly fit and actual vs predicted score in set 1

Cosine

Figure 20: Set 0

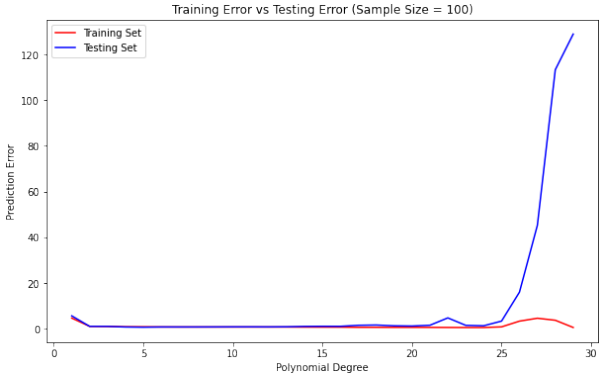


Figure 21: Set 1

. Unlike sine function, the two data-sets for cosine behave differently. For set 0 we can see that there is no significant difference in the test and training error at least till polynomial degree 30 (figure 20). But, for set 1 the gap increases for polynomial degree >25 (figure 21). This can further be verified from the actual vs predicted value plots(figures 24 and 25) . For

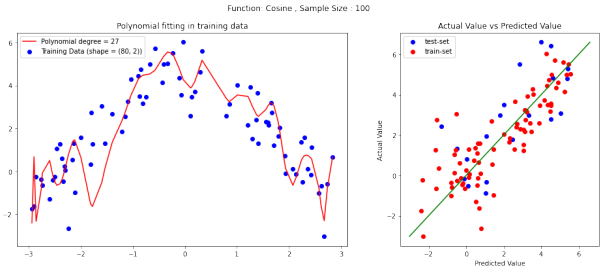


Figure 22: Set 0, Poly deg 27

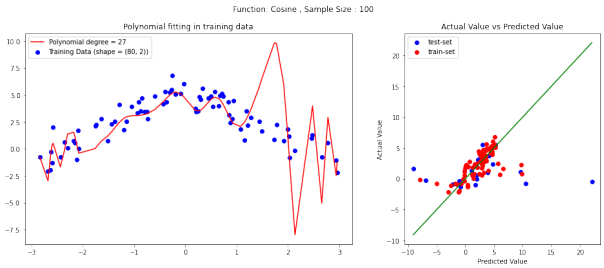


Figure 23: Set 1, Poly deg 27

both the sets polynomial degrees $\in [2, 20]$ can be said to be a good model. While for both the set degrees 1 results in under-fitting the data.

Tangent

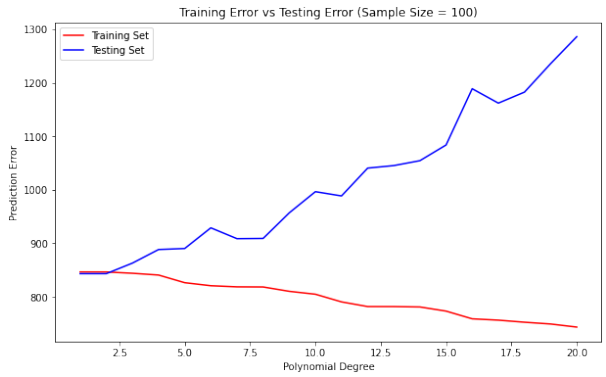


Figure 24: Set 0

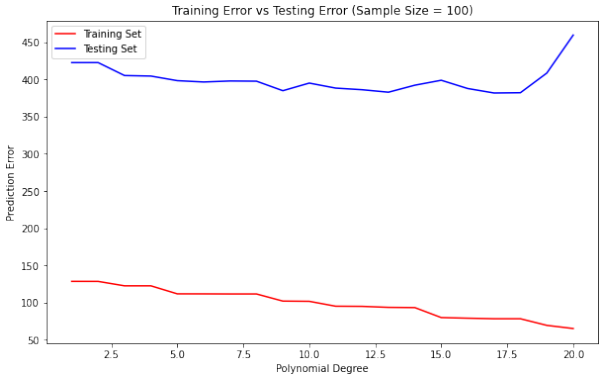


Figure 25: Set 1

For tangent function, we never find a perfect model for either sets because in both the sets the prediction error for either the training or test set is very high regardless of the polynomial degree. This is also visible from figures [26:29] as the model never perfectly fits the data resulting in high variance.

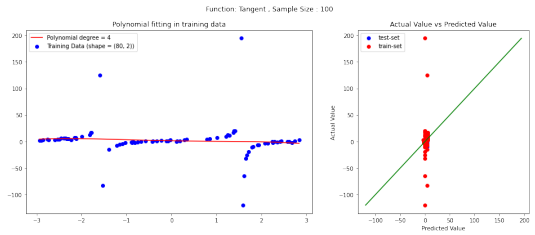


Figure 26: Poly Degree 4

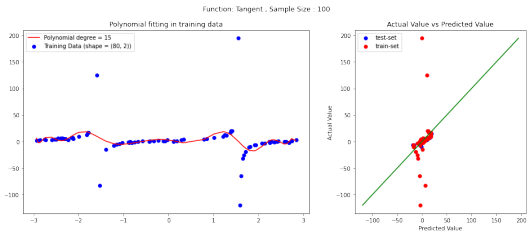


Figure 27: Poly Degree 15

Figure: Poly fit and actual vs predicted score in set 0

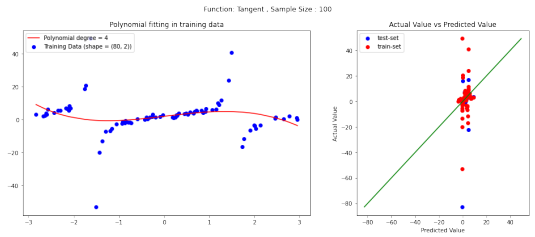


Figure 28: Poly Degree 4

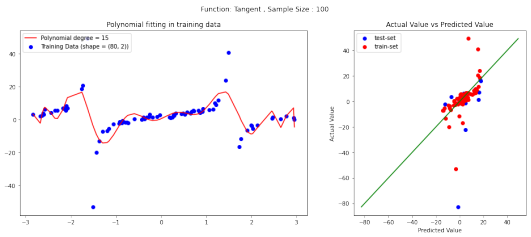


Figure 29: Poly Degree 15

Figure: Poly fit and actual vs predicted score in set 1

Function: Polynomial-n (Degree 4, 7)

For the two polynomial of degree 3 and 14 the parameters were kept the same. One set of data with size 100 were generated with variable domain.

From the previous examples, we can see the general trend in the variation of prediction error with respect to polynomial degree.
So, for the polynomial-n function I decided to **experiment with noise** in the data set.

Polynomial-4

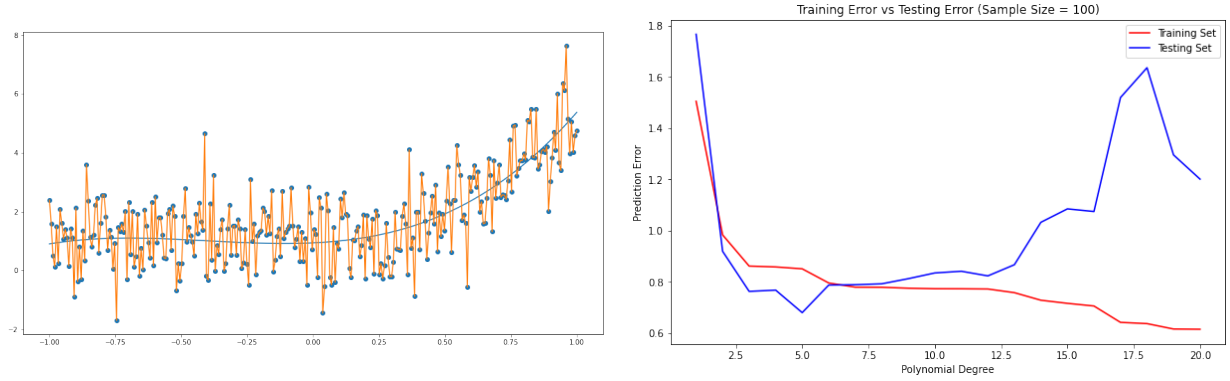


Figure 30: Noisy Data-set

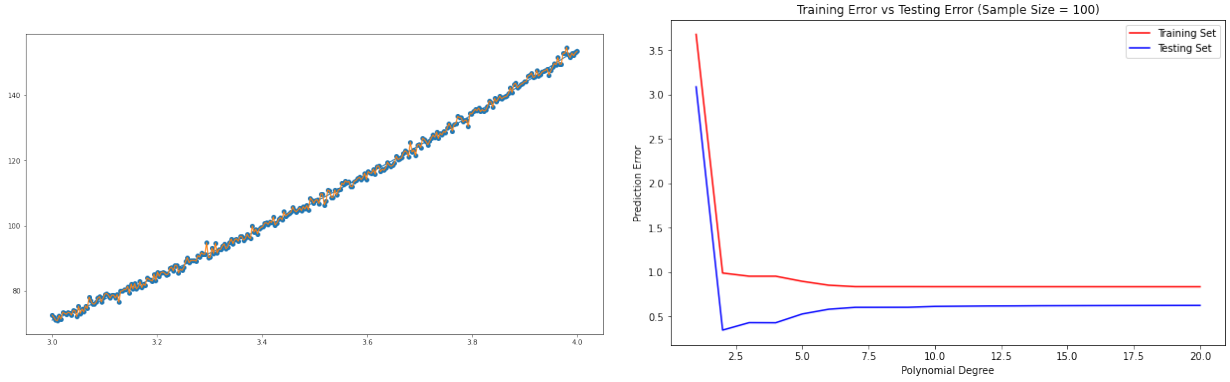


Figure 31: Less Noisy Data-set

For a noisy data-set(figure 30), the same trend for the prediction error holds. But for non noisy data-set (figure 31) the trend no longer hold true because for polynomial degree ≥ 4 we observe that the testing error and training error remains constant.

Polynomial-7

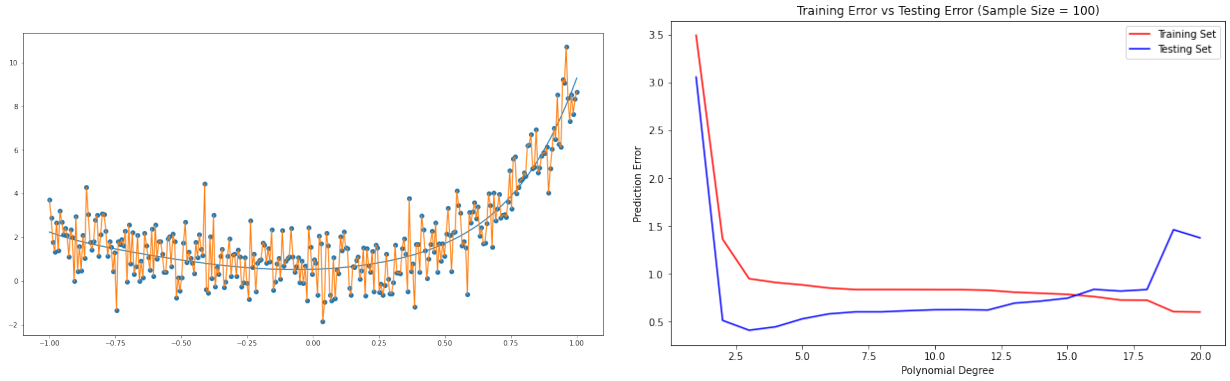


Figure 32: Noisy Data-set

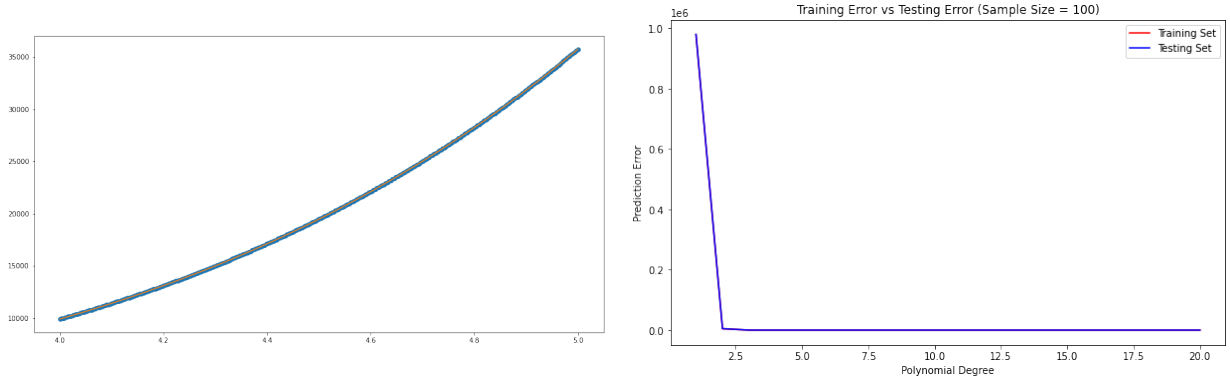


Figure 33: Less Noisy Data-set

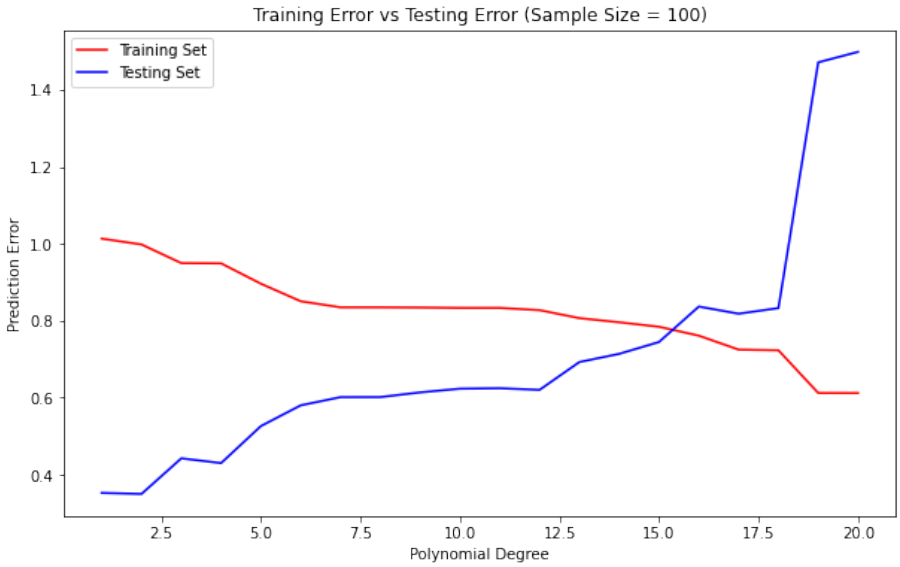
Similar to polynomial-4 the less noisy data-set in polynomial-7 perfectly fits the model compared noisy data-set.

Concept of noise in ML

From the previous two examples, we see that a nose less data can be easily modeled and results in zero error. But, one must know that in real life data is never noise free and the distribution of the data is never known. Thus comes machine learning, where one should balance the bias and variance in a model and make predictions.

Function: Log and Inverse

Together with the tangent function log and inverse are two new functions that were created. The log and inverse functions both use domain $\in [0.1, 1]$ because $y = \log(0)$ and $y = \frac{1}{0}$ are infinite.
Both these function show the same trend as other functions.



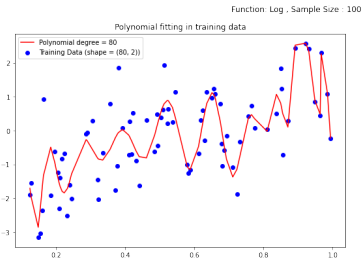


Figure 34: Overfit

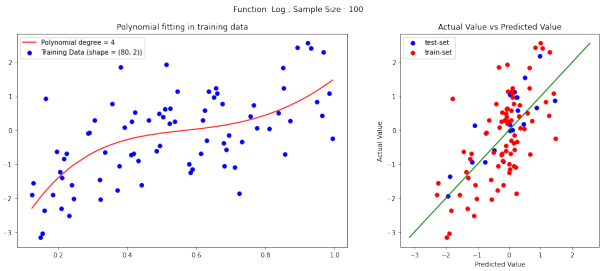


Figure 35: Good Fit

Figure 36: Function: Log

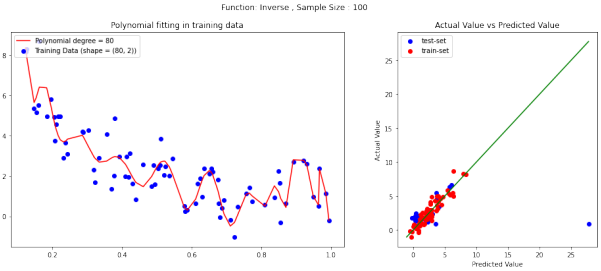


Figure 37: Overfit

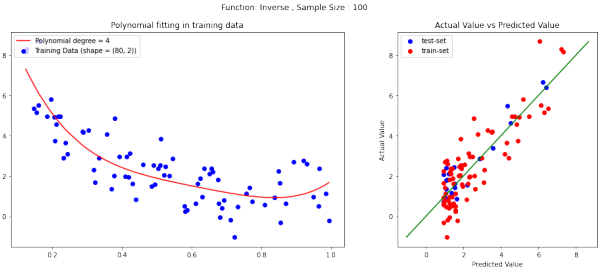


Figure 38: Good Fit

Figure 39: Function: Inverse

K-Nearest Neighbours

Setting the Scene

1. Sampling the data: Code for sampling data is submitted along with the notebook. For sampling I used a pseudo-random number generator to index the array.

2. Splitting the data: I used the *test_train_split* function from Sklearn library. Firstly the data was split in 80:20 ratio then in 20:80 ratio. For each fraction of split the sample size of 70 was generated.

3. Error Function: For calculating error, I created my own error function. Using MSE for this data-set would result in wrong errors because MSE uses distance between two variables. In this case, if our predictor predicts 2 instead of 0 then the distance is 4. Similarly if our predictor predicts 1 the distance is 1. Thus, we have 2 different error weights for the same error resulting in anomaly. So, I used the average error to compute the error. The error is defined such that:

$$E(x) = \begin{cases} 1 & \text{if } h(x) \neq f(x) \\ 0 & \text{otherwise} \end{cases}$$

$$\implies L_D(h) = \frac{1}{n} \sum_{i=1}^{n} E(x_i)$$

From figures 40 and 41 we see that the plot does not follow the previous trend we encountered in regression. Rather, for knn the nearly perfect model is given by n whose error is lowest and the distance between the training error and test error is less. So, for set 0, we can argue that $n = 11$ gives near perfect fit because the the distance (δ) between the training and test set errors are lowest compared to other n. And the absolute training and testing error are also low. This results in model with low bias and low variance.
Similarly for set 1, $n = 8$ can be considered a good fit because the error rate is low and the δ is also low compared to other parts of the graph. Note: the intersection does not represent any n.

Recall in regression the 20:80 split (figure 13) resulted in the prediction error of testing set very high compared to training set. A similar trend holds for knn as well. But here we observe that as n increases both testing error and training error increases regardless of the set (figure 42 and 43). This shows that for a model to be able to learn appropriate number of training data must be chosen.

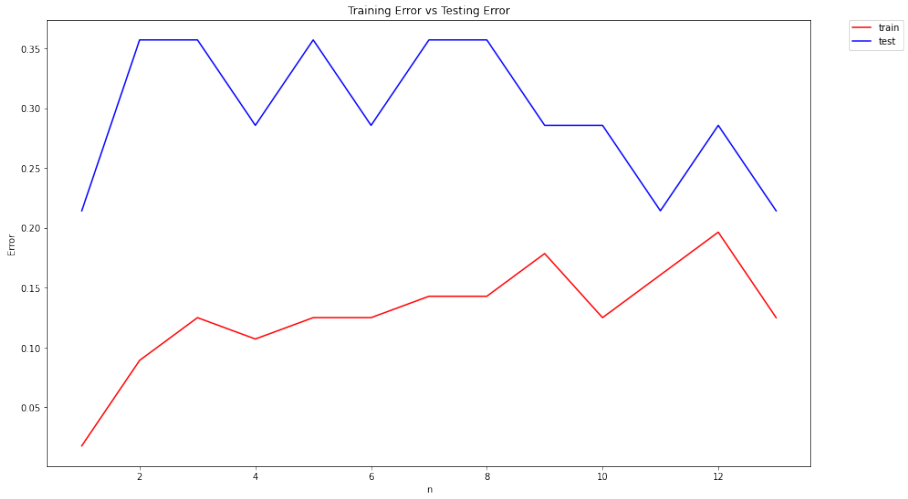


Figure 40: Set 0

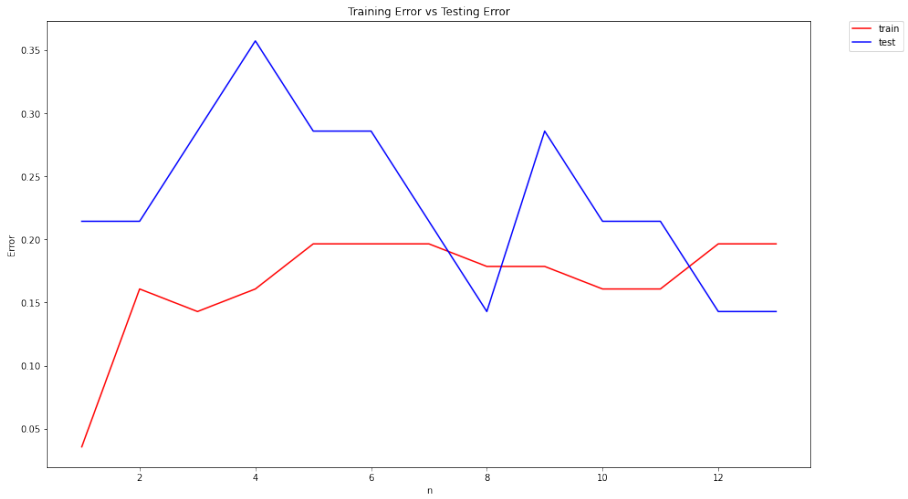


Figure 41: Set 1

80:20 split

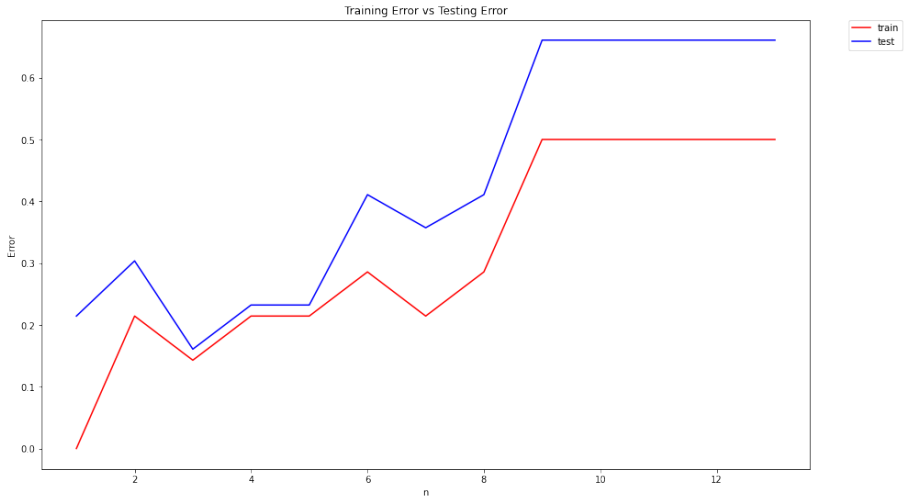


Figure 42: Set 0

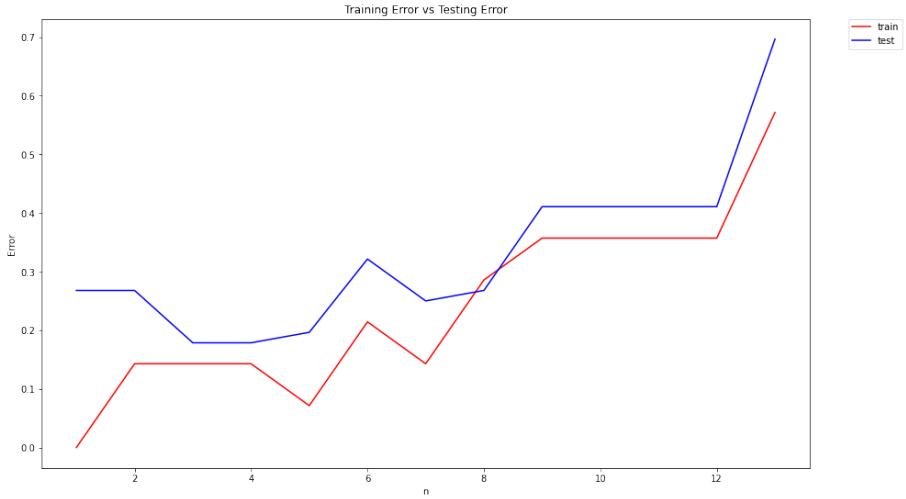


Figure 43: Set 1

20:80 split