James Yang

DATA 558

Homework #3

1. In this problem, we'll see a (very!!) simple simulated example where a least squares linear model is “too flexible”.
   1. Y = 2.307-1.695x1 + 4.272x2 + ε

Table

Description automatically generated

* 1. ε = 0.395
     1. Bias: 0.0246
     2. Variance = 0
     3. EPE = Test MSE = E((y – f(x)^2) = 1.2363
     4. EPE = Test MSE = E((testing)^2) / 20 = 0.86109
     5. The expected predicted error was higher than the validation set test error but they seem to be very similar to each other. Since the model predicts 0, it should be very similar for all values.
  2. EPE 1d = sum((testing\_predicted – training)^2) = **64.13**

Code used:

Graphical user interface, text, application, email

Description automatically generated

* 1. The linear model had a higher variance and test error but a lower bias because of the overfitting in the model. This caused it to perform poorly when trying to predict new data points, but the model that predicted all 0’s was performing well because the output was already generated with the intention of having an expected value of 0. This meant that predicting 0 when X and Y were uncorrelated performed better with a more optimal testing error.

Table

Description automatically generatedChart, histogram

Description automatically generated

* 1. Validation error 2b = 0.9298
  2. Validation error 2c = 0.9336
  3. The estimated error of 2b was lower than the estimated error of 2c because we were fitting data that had correlation/relation to each other. If we could imagine data points that were fitted closer together in a more predictable trend line, then it would give you a more properly fitted regression line that has fewer outlying points. If we rerun this test multiple times, we can see that the validation (random correlation values) fluctuates a lot whereas the highest correlation values in 2c produce a more consistent error value. This explains why option 1 essentially gave us a meaningless estimate of the test error because there were always going to be random values that had little to no correlation with each other, thus making them almost completely random and misleading. Not only that, sometimes they can make the test error less than what its true value is because they might accidentally stumble in line with each other very closely by complete coincidence. Basically, they are phony points that disrupt any significance when it comes to the true error/model.

Using this code:

Graphical user interface, text

Description automatically generated

1. .
   1. I got my dataset from UCI ML Repo (<https://archive.ics.uci.edu/ml/datasets/Sports+articles+for+objectivity+analysis>). It represents attributes from 1000 sports articles with their link, objectivity, word count, number of adverbs, number of nouns, etc. It essentially analyzes 59 of attributes regarding the articles and how they are structured. In my case, I’m trying to see the response of the total word count based on how many adverbs, nouns, commas, semicolons, and the 54 other attributes that put together a sports article. These predictors are all small pieces of what puts together a sports article. How many quotes it contains, numbers/statistics found, and many other features help put together predictors for how many words are found within the article. Words are conditionalized as a string with a space. For example, “1 2 hello” would be 3 words in the case of this dataset.
   2. Using the same method as option 2 from the previous question, I split the observations into training and validation sets. Then on the training sets I found the correlations for the largest values for the correlation. I fit an L.S model to predict y using 59 features. My estimate came out to be **120.77** which means that the dataset may contain data with notable irregularities. Furthermore if I reduced the number of features to 30, my error came out to be a very high **1207.72.**

Graphical user interface, text, application

Description automatically generated

* 1. Using this code:

A picture containing logo

Description automatically generated

Chart, line chart

Description automatically generated

* 1. Using this code:

Text

Description automatically generated

The best lambda calculated was **52.308.**

The test error for ridge was **349.450.** I estimated the error the same way I have been doing in previous sections. I first created a ridge model using all the features and the total words count as the y and setting the lambda as 52.308. Then I predicted the ridge values on training data and calculated the testing error by summing the predicted – testing (same values from 3b) and squared them then divided by 300.

* 1. Using practically the same code as part c and d, except changing alpha = 1 we get these plots.

Chart, line chart

Description automatically generated

* 1. The best lambda calculated for Lasso was **15.248** and the test error received was **310.957**. I estimated the test error the same as I did in part d. I created a Lasso model which is the same as the Ridge model except with alpha = 1. Then I predicted the lasso based on training data and then calculated the error by summing the predicted model minus the testing model squared, and dividing by the number of elements 300.

Graphical user interface, text, application

Description automatically generated

* 1. Using this code:

Text

Description automatically generated

Graphical user interface, chart, line chart

Description automatically generated

After running this procedure a few times I have noticed that the best polynomial finding is approximately around 6. What this means is that the prediction that fits the regression best is not a simple linear model but rather a model that fits well with a 6th degree polynomial. The reason this may be the case is because the data is more complex and has a more curvy trajectory.

* 1. Using this code:

Text

Description automatically generated

Chart, line chart, histogram

Description automatically generated

I was able to produce this graph that looks very similar to the one found in figure 5.4. It appears that the best degree of polynomial is 7 which correlates well with the previous question using the validation set approach because they agree with each other. They are both saying that the model is complex and has a curvy trajectory. This model would not fit well with something as straight as a linear regression.

* 1. After using the caret library to generate 10-fold CVs, it produced a plot like this. Using this code:

Graphical user interface, text

Description automatically generated

Chart, line chart

Description automatically generated

The best degree polynomial it found to be was approximately 5. It aligns fairly well with the validation set and leave one out methods because it the dataset can not be fitted with a linear regression, so it needs more curvature in its regression. In this case it predicted that it only needed a 5th polynomial as opposed to the 6th and 7th polynomial predicted in the previous sections.

* 1. Using this code: (Code was too long to screenshot)

*error\_4d <- c()*

*for(i in 1:10) {*

*model\_4d <- lm(mpg ~ poly(Auto$horsepower, i)+ poly(Auto$cylinders, min(i, 4)) + poly(Auto$displacement, i) + poly(Auto$weight, i)+ poly(Auto$acceleration, i)+ poly(Auto$year, i)+ poly(Auto$origin, min(i, 2)), data=Auto)*

*predict\_4d <- predict(model\_4d, newdata = data.frame(Auto$horsepower))*

*error\_4d <- append(error\_4d, sum(model\_4d$residuals^2))*

*}*

*plot(1:10, error\_4d, type = "b", xlab = "Degree of Polynomial", ylab = "Training Set Mean Squared Error")*

I was able to produce this plot:

Chart, line chart

Description automatically generated

These values look awfully like both the textbook and what I have found in the previous sections (b and c). It makes sense that this linear regression has errors up to 4000+ though because as exhibited from the previous parts of the question, it needs to fit a curvature model with exponents up to 5-7 to reduce the error of the model.

* 1. Using the 10th polynomial,

Table

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

It will give you a very long report about each feature and their estimate, standard error, t value and p value. Something to note here is that the p-value is somewhat high for some of these attributes such as acceleration at the 10th polynomial and year at the 4th polynomial. This would indicate that they aren’t significant in the dataset in predicting MPG. It makes sense when looking at more of the features too because features at higher polynomial values are expected to be less statistically significant because they need to be more specific to the model.