CS 7646 001 – MC3 P1

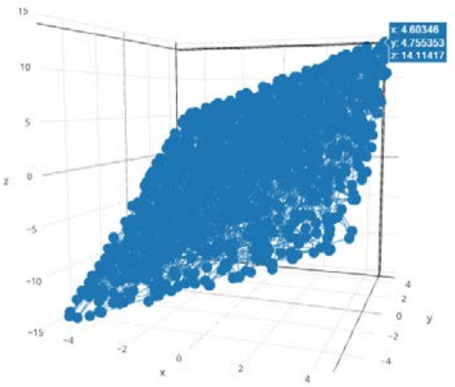
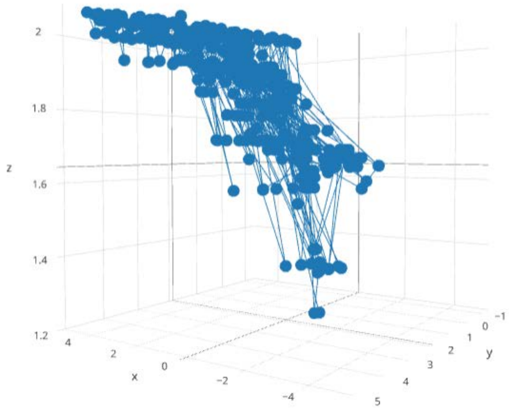
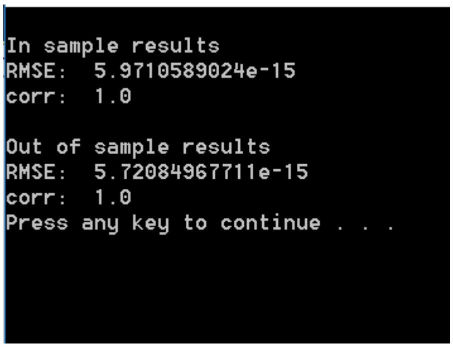
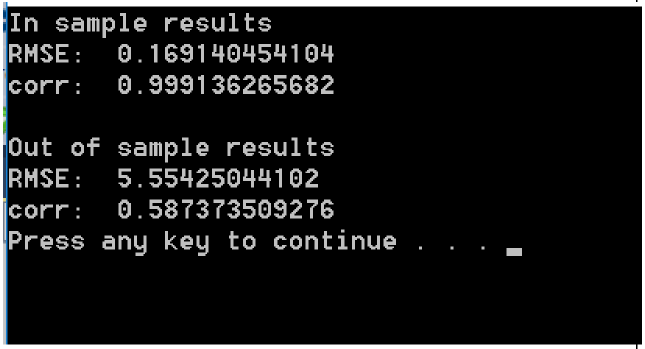
Gatech Id : nghosh9

a. Create your own dataset generating code (call it best4linreg.py) that creates data that performs significantly better with LinRegLearner than KNNLearner. Explain your data generating algorithm, and explain why LinRegLearner performs better. Your data should include at least 2 dimensions in X, and at least 1000 points. (Don't use bagging for this section).

I used points from the function below and sorted them in ascending order of **Z**. Sorting is an important step in generating the data so that KNN does not have any "trained" data points in the region we are testing the algorithm. (last 400 data points in the whole set)

Z = X + 2Y

The graph below shows the points plotted for this equation between (-5,5) for both x and y

Results with LinRegLearner	**Results with KNNLearner**
	
 In sample results RMSE: 5.9710589024e-15 corr: 1.0 Out of sample results RMSE: 5.72084967711e-15 corr: 1.0 Press any key to continue . . .	 In sample results RMSE: 0.169140454104 corr: 0.999136265682 Out of sample results RMSE: 5.55425044102 corr: 0.587373509276 Press any key to continue . . .

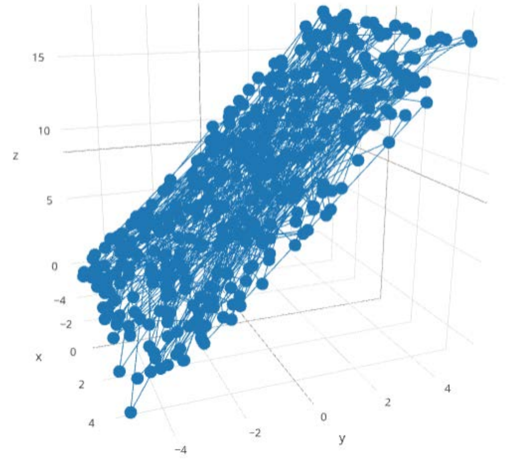
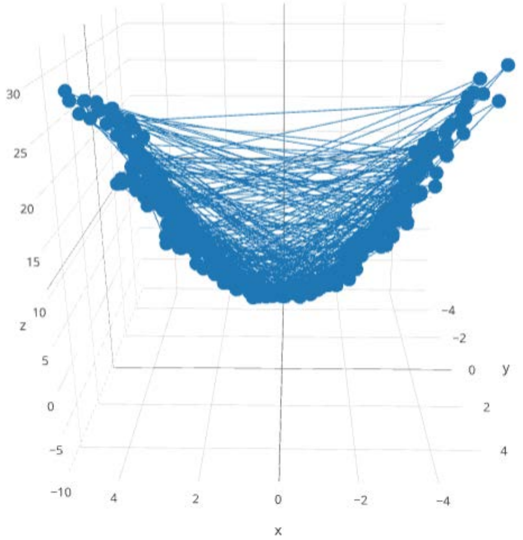
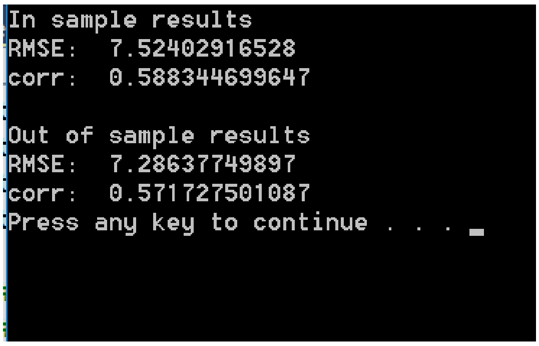
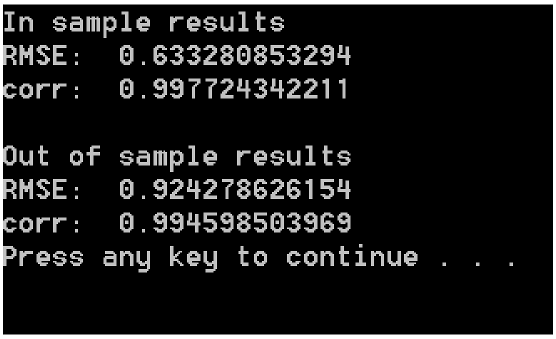
The Linear Regression learner performed really well on the dataset as it inherently identified the parametric nature of the data points.

As we see above the KNN learner performed really poorly on the dataset with a RMS of 5.5 and a correlation of only .58 . This is because the KNN learner failed to identify the parametric nature of the dataset and also because there was no overlap between the training and testing results (z values). The trick of using sorted data may be defeated if n-fold cross validation is used to train/test the data.

b. Create your own dataset generating code (call it best4KNN.py) that creates data that performs significantly better with KNNLearner than LinRegLearner. Explain your data generating algorithm, and explain why KNNLearner performs better. Your data should include at least 2 dimensions in X, and at least 1000 points. (Don't use bagging for this section).

For this is used points from the function below.

Z = X^2 + 2*Y

Results with LinRegLearner	**Results with KNNLearner**
	
	

```
In sample results
RMSE:  7.52402916528
corr:  0.588344699647

Out of sample results
RMSE:  7.28637749897
corr:  0.571727501087
Press any key to continue . . .
```

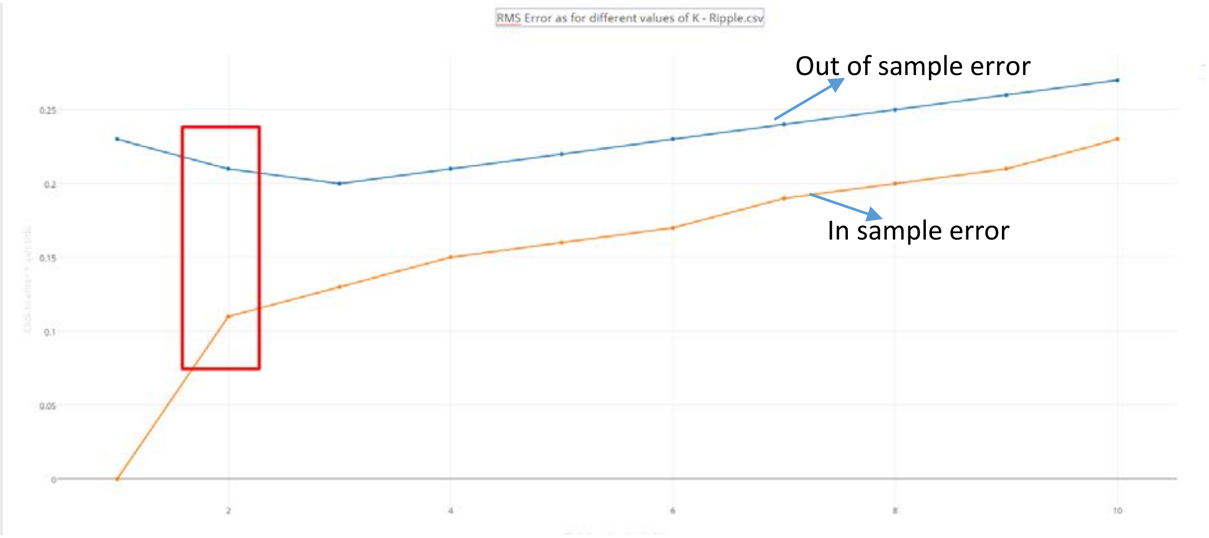
```
In sample results
RMSE:  0.633280853294
corr:  0.997724342211

Out of sample results
RMSE:  0.924278626154
corr:  0.994598503969
Press any key to continue . . .
```

In this case the KNN learner performed really well as it was able to identify nearest neighbours from the training data when predicting values. The Linear regression learner is unable to perform well as it only predicts linear coefficients for its model. In this case it predicts the coefficients as

$$\mathbf{Z=-.09x + 1.82y + 8}$$

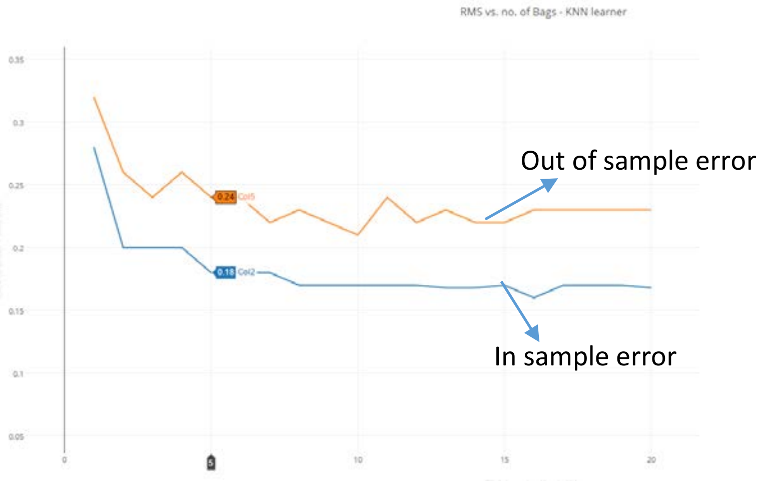
c. Consider the dataset ripple with KNN. For which values of K does overfitting occur? (Don't use bagging).

From the graph below (RMS vs. K) for insample and out of sample error it is clear that over fitting occurs for values of k <= 2. It is around this point that the in sample rms error keeps decreasing while the out of sample error starts to increase.



d. Now use bagging in conjunction with KNN with the ripple dataset. How does performance vary as you increase the number of bags? Does over fitting occur with respect to the number of bags?

The graph below shows the performance of bagging (rms error) as **no. of bags** increases



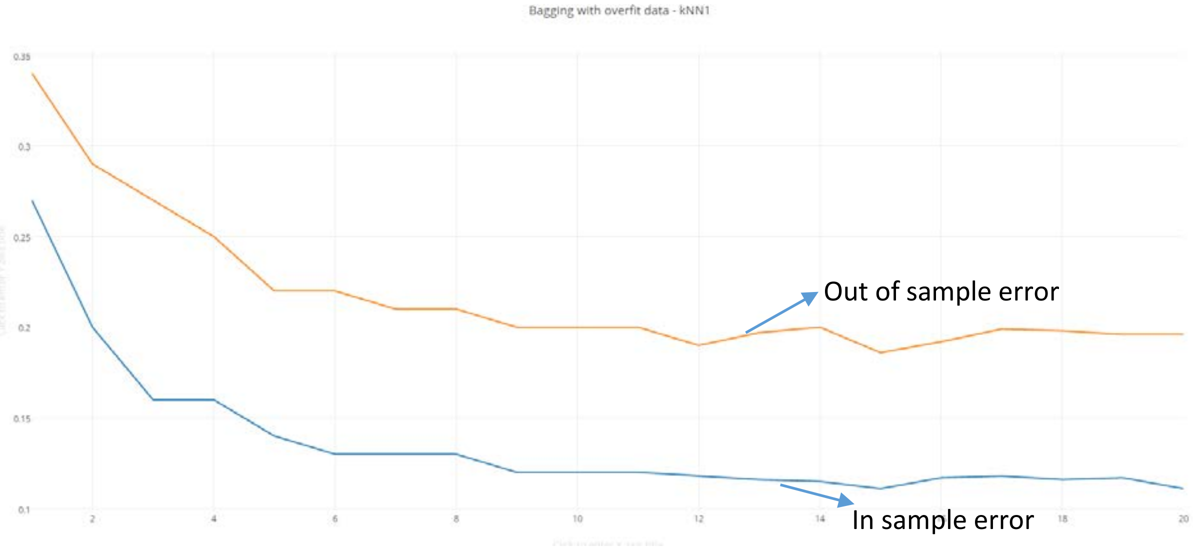
It is seen that the out of sample RMS error steadily decreases till around 7 bags and then steadies around .22-.23. This is expected as the error should reduce with the averaging of results that larger number of bags leads to.

Over fitting – No, there is no over fitting as we increase the number of bags, the RMS error steadies around the .22 mark and stays there even as we keep increasing the number of bags.

e. Can bagging reduce or eliminate over fitting with respect to K for the ripple dataset?

Yes, bagging will reduce the over fitting caused due to an incorrect choice of K. This is because for no of bags > 1 , every bag will contain a different dataset which will lead to different nearest neighbours for a point and help in averaging out the results.

Let's check for k = 1 in the ripple dataset which clearly will over fit the data. We will use bag sizes from 1 to 20 to see results.



As expected the RMS error decreases as the no. of bags increase and then steadies to a value of . This is where we clearly know that the KNN model has been over fit with a k of 1. So bagging does help reduce the effects of overfitting. For very large number of bags it will almost eliminate the effects of overfitting.