# MC3-Project-1

*Name: Xiaobing Li*
*Email: xli605@gatech.edu*

## Overview

In this project, I implement and evaluate three learning algorithms as Python classes: A KNN learner, a Linear Regression learner (provided) and a Bootstrap Aggregating learner. The classes are named KNNLearner, LinRegLearner, and BagLearner respectively. We are considering this a **regression** problem (not classification).

*Question 1.*

*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).*

## Method

To generate this dataset, I created 1000 points with Y values linearly correlated with X1 and X2 values plus Gaussian noises. I also added 10% outliers with significantly bigger Y values. The dataset is shown in Figure 1.

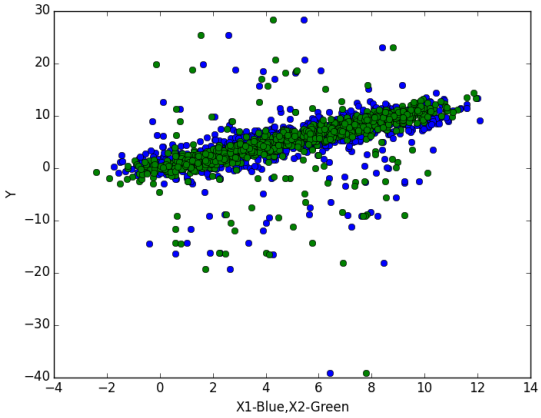



*Figure 1. A dataset best for LinRegLearner*

## Results

For the LinRegLearner, the RMSE for the train set is 4.25 and for the test set is 3.31. However, the KNNLearner performed much worse than the LinRegLearner. The RMSE for the train set is 3.50 but the RMSE for the test set is 4.72. A table is shown below.

| **LinRegLearner** | | **KNNLearner** | |
|---|---|---|---|
| **Train** | Test | Train | Test |
| RMSE: 4.24766577 | RMSE: 3.31088132 | RMSE: 3.49997359 | RMSE: 4.71625121 |

| | | | |
|---|---|---|---|
| corr: 0.583155247 | corr: 0.694374802 | corr: 0.743050563 | corr: 0.439386859 |

**Explanation**

LinRegLearner performs better than KNNLearner in this dataset because Linear Regression method is trying to find the least square error from the whole dataset, so it is not sensitive to outliers. KNN method is only checking the nearby data points so it is sensitive to outliers.

*Question 2.*

*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).*

**Method**

To generate this dataset, I created 800 Y values linearly correlated with X1 and X2 values with two slopes and 200 Y values correlated with X1 and X2 with two different slopes. Then randomly shuffle the dataset. That generates an non-linear relationship between Y and Xs. The dataset is shown in Figure 2.

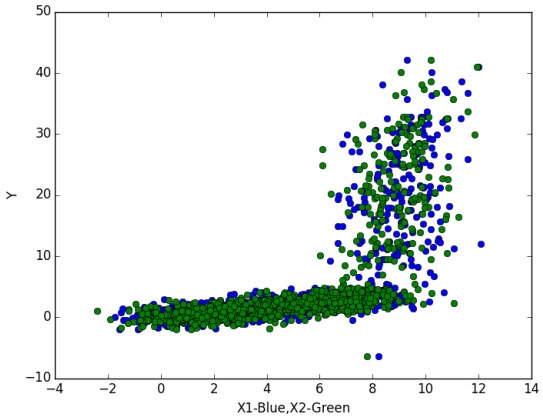



*Figure 2. A dataset best for KNNLearner*

**Results**

For the LinRegLearner, the RMSE for the train set is 6.22 and for the test set is 6.26. The KNNLearner performed better than the LinRegLearner. The RMSE for the train set is 3.71 and the RMSE for the test set is 5.50. A table is shown below.

| **LinRegLearner** | | **KNNLearner** | |
|---|---|---|---|
| **Train** | Test | Train | Test |
| | | | |

| | | | |
|---|---|---|---|
| RMSE:6.21588158<br>corr: 0.666334423 | RMSE: 6.26185865<br>corr: 0.702933315 | RMSE: 3.71337833<br>corr: 0.895505646 | RMSE: 5.50107217<br>corr: 0.786165590 |

**Explanation**

LinRegLearner performs worse than KNNLearner in this dataset because the Xs and Y are not linearly correlated. Still, Linear Regression method is trying to find the least square error for the whole dataset, however, I intended to make the dataset included two different distributions, so the performance of LinRegLearner is not good. KNN method is only checking the nearby data points so it can still find a good match even there are different distributions in the dataset.

*Question 3.*

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

**Method**

To answer this question, I applied KNNLearner to the ripple dataset with a series of K values. RMSEs for the train set and test set were plotted to the K values. The result is shown in Figure 3.

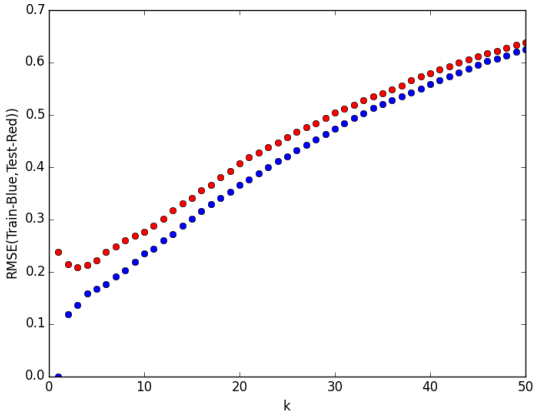



*Figure 3. RMSE vs K*

**Results**

As we can see from the figure, the RMSE for the train set (blue dots) keeps decreasing with K value decreasing. However, the RMSE for the test set starts to increase when k is smaller than 3. So k=2 is where the overfitting occurs.

*Question 4.*

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

**Method**

To answer this question, I applied BagLearner to the ripple dataset with a series of bag numbers. RMSEs for the train set and test set were plotted to the bag numbers. The result is shown in Figure 4.

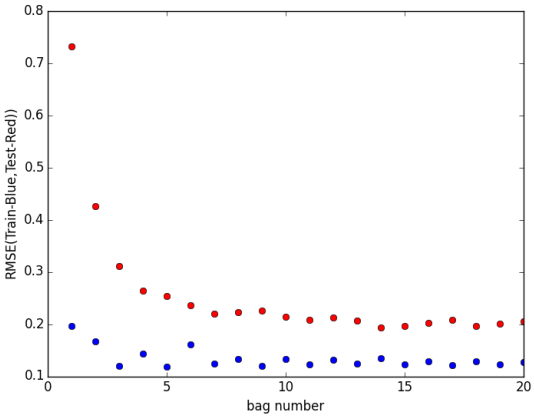



*Figure 4. RMSE vs bag numbers*

**Results**

As we can see from the figure, the RMSE for the train set (blue dots) keeps decreasing then stays in a certain level with bag numbers. At the same time, the RMSE for the test set also keep decreasing until reaching a certain level. The overfitting doesn't occur in this method.

*Question 5.*

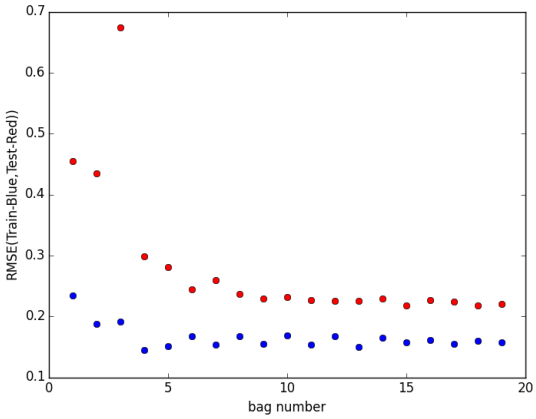
*Can bagging reduce or eliminate overfitting with respect to K for the ripple dataset?*

**Method**

My result for Question 4 seems support the idea in this question. Bagging can reduce overfitting for the ripple dataset. To confirm this idea, I used an overfitting k value k=2 in the BagLearner to see its performance. The result is shown in Figure 5.

**Result**

Even for an overfitting k value, the BagLearner still keeps reducing the RMSE. The RMSE can eventually decrease to almost the same level as k=3, so we can conclude that bagging did reduce the overfitting.




*Figrue 5. Bagging with K=2*