MC3 Project 1 Report

Vedanuj Goswami
GT ID : 903126228
Georgia Tech, Atlanta, GA

In this project we generate data sets and measure performance of *KNN Learner* and *Linear Regression Learner*. We also measure how varying values of k affect over fitting for *KNN Learner*, varying values of bags affect over fitting for *Bag Learner* and whether *Bagging* can eliminate or avoid over fitting over various values of k.

1. LinRegLearner vs KNNLearner using *best4linreg.py*

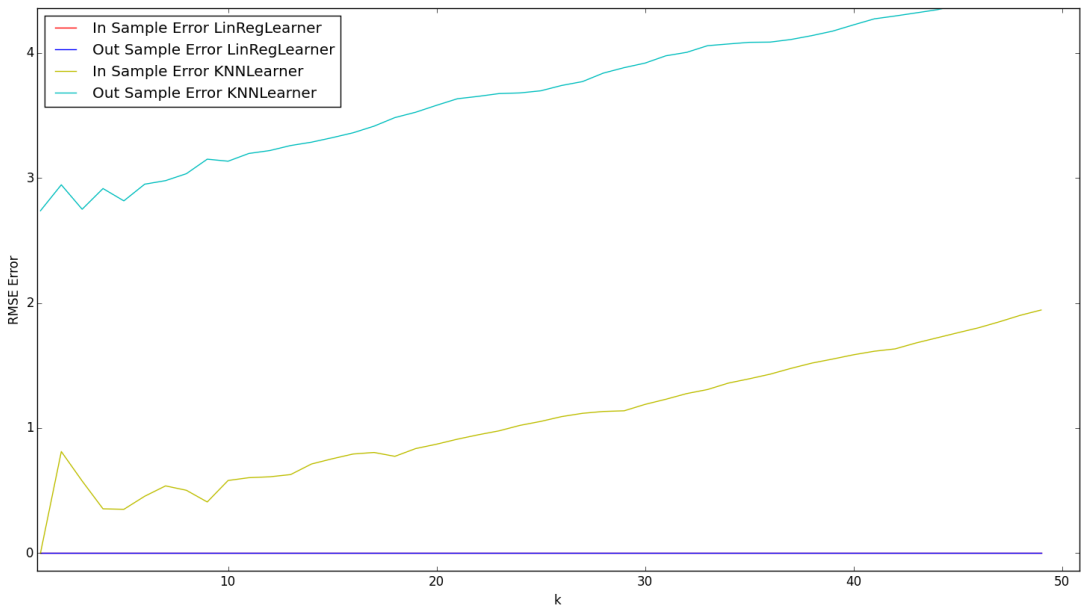

Fig. 1. *Linear Regression Learner* performing better than *KNN Learner* over data set generated by best4linreg.py

Linear regression performs better on linear data that can be fitted in a line or plane. The data set generating algorithm uses this simple logic to create a data set that fits a line. We

use the following formula to generate 1000 points with $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{Y}$ values.

$$\mathcal{Y} = \mathcal{X}_1 + 2 * \mathcal{X}_2 \tag{1}$$

where $\mathcal{X}_1$ varies from 1 to 10 and $\mathcal{X}_2$ varies from 1 to 100. The equation is just a linear equation that generates points on a line. Since the data is linear the *Linear Regression Learner* gives a correlation coefficient of 1.0 and RMS error of 0. However when running *KNN Learner* the correlation coefficient is always less than 1 and RMS error is always more than 0. This is because *KNN Learner* finds the mean of the k nearest data points which might not lie on the linear line that is used to create the data set. Hence on this data set *Linear Regression Learner* performs significantly better than *KNN Learner* for all values of **k**, which is evident from the Figure 1.

	Linear Regression Learner	**KNN Leaner with k = 3**
In Sample RMSE	5.04768310511e-14	0.576708412105
In Sample Correlation	1.0	0.999950264835
Out Sample RMSE	1.41943416558e-13	2.7506564873
Out Sample Correlation	1.0	0.999804341697

2. KNNLearner vs LinRegLearner using *best4KNN.py*

KNN Learner performs better than *Linear Regression Learner* when the data does not fit linearly or in a polynomial curve. We use a data set that is generated using two lines, having negative slopes with respect to each other. This means that Linear Regression Learner will try to fit one line over the other and when it tries to predict the remaining values, the error will be higher. This will not affect KNN Learner much as it will always try to find closest k points and take the average. We use the following algorithm to generate 1000 points with $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{Y}$ values.

$$\begin{aligned} \mathcal{Y} &= \mathcal{X}_1 + 2 * \mathcal{X}_2 \textbf{ if } 0 \leq \mathcal{X}_2 \leq 60 && (2) \\ &= -(\mathcal{X}_1 + 2 * \mathcal{X}_2) \textbf{ if } 61 \leq \mathcal{X}_2 \leq 100 && (3) \end{aligned}$$

This generates a randomly distributed set of values where for every 100 points, few points lie on a line and the rest lie on a line having negative slope with respect to the first line.

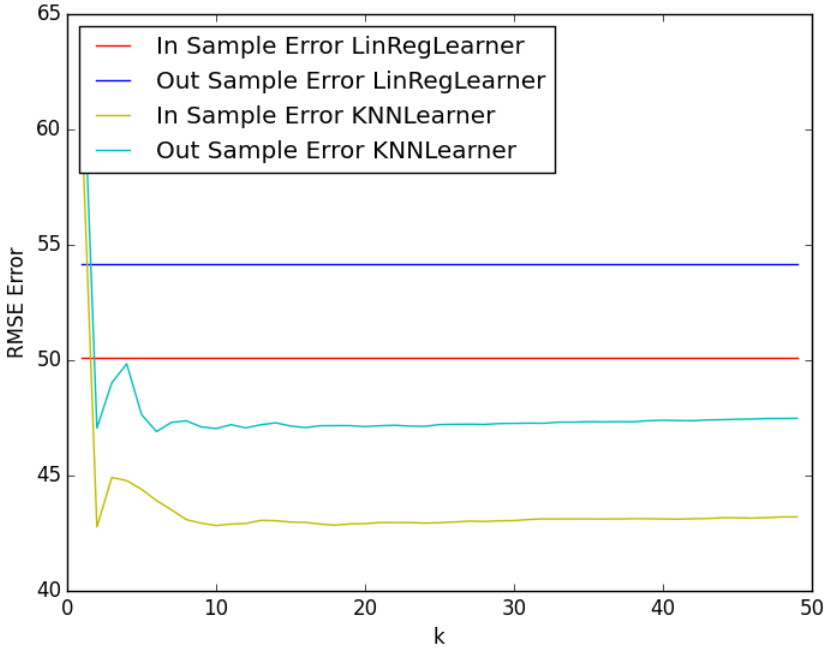

Fig. 2. *KNN Learner* performing better than *Linear Regression Learner* over data set generated by best4KNN.py

We observe in Figure 2 that for varying values of k on this data set, *KNN Learner* always performs significantly better than *Linear Regression Learner*.

	Linear Regression Learner	**KNN Leaner with k = 3**
In Sample RMSE	50.0819138536	44.9144103742
In Sample Correlation	0.528850891169	0.652603224954
Out Sample RMSE	54.1443472593	49.0172163632
Out Sample Correlation	0.517145256341	0.634670591113

3. KNNLearner Overfitting

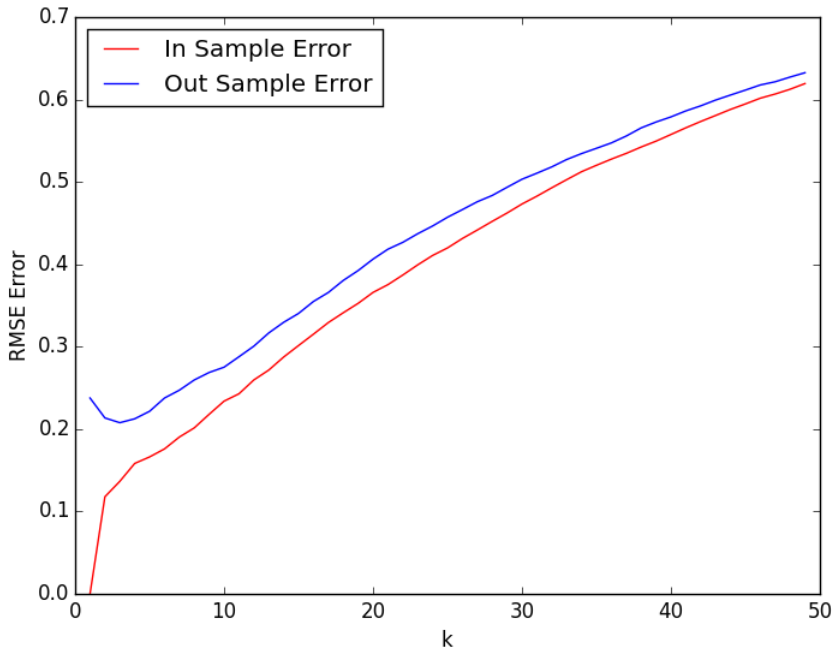

Fig. 3. *KNN Learner* on varying the value of k on ripple.csv data set

We observe that for values of k lower than 3 over fitting occurs as in sample error reduces but out sample error increases for values lower than k = 3. Value of k is taken from 1 to 50.

4. Bagging with KNN with respect to number of bags

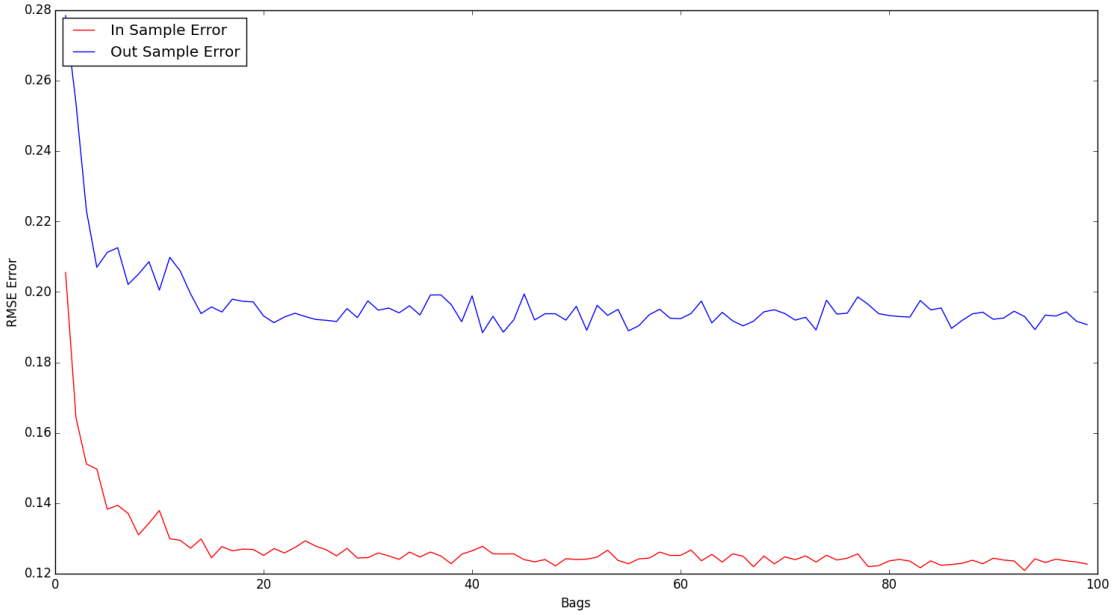

Fig. 4. *BagLearner* on increasing the number of bags reduces error.

We observe that as the number of bags increases the in sample and out sample error decreases. Hence performance improves as the number of bags are increased.

Bagging helps to avoid over fitting. We can observe this from the Figure 4 where we see that in sample and out of sample error both decreases as we increase the number of bags from 1 to 100.

5. Bagging with varying k and Overfitting

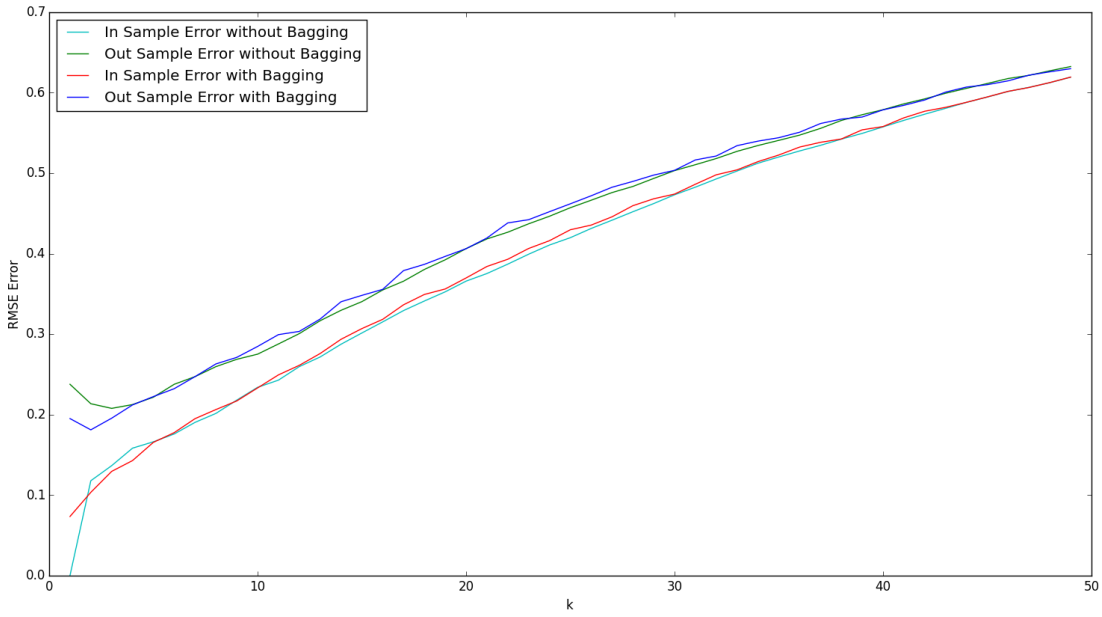

Fig. 5. *KNN* performance with and without Bagging. We observe that Bagging helps to reduce over fitting.

We observe that over fitting reduces if we use bagging. In Figure 5 we see that when Bagging is used over fitting occurs for values less than 2. However for KNN without Bagging over fitting occurs for values lower than 3. Hence we can conclude that bagging helps to reduce over fitting with respect to k.