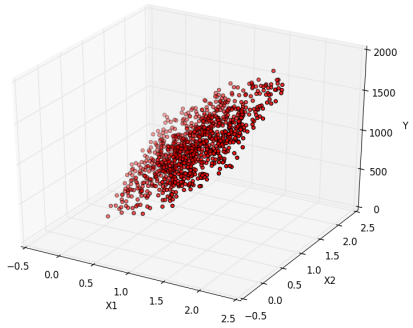
MC3_P1 Report
Tyler Bobik

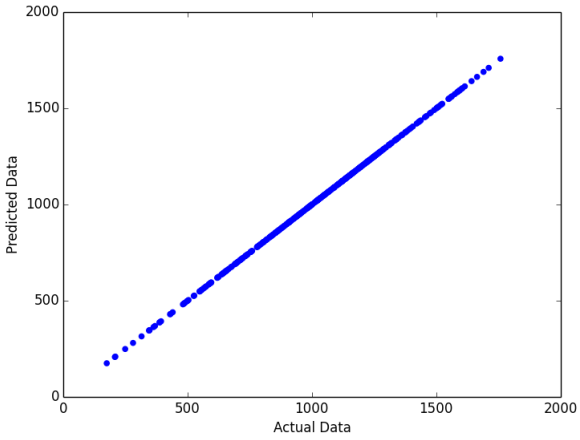
Best4linreg

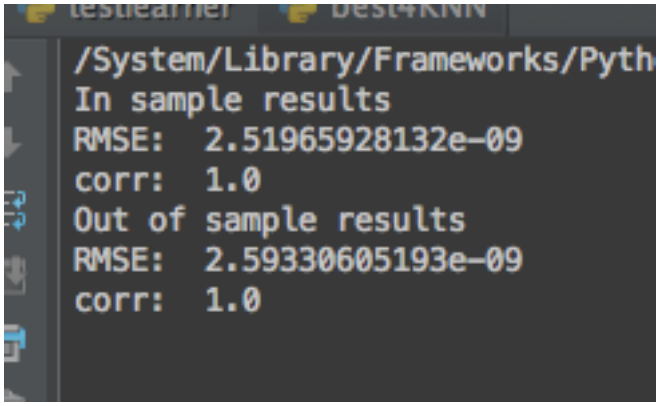
I created a data set that performs better with linear regression then with KNN. My dataset consists of two X dimensions and 1 Y dimension. To ensure that the linear equation would perform better I started with with a random uniform sample from 0 to 1 with 1000 values for my X1 and X2 dimension. I then added a bit of random noise to the data. To create my Y dimension, I multiplied each X dimension by 500 and added them to each other. This caused the linear equation to perform better then the KNN because linear regression is able to calculate the relationship between Y and X1,X2 dimensions without being thrown off by the noise. The KNN does not perform as well because it gets thrown off by the noise and random variation in the X dimension data.

You can see this in the 3D model of the data set I came up with with noisy but uniformly distributed data:

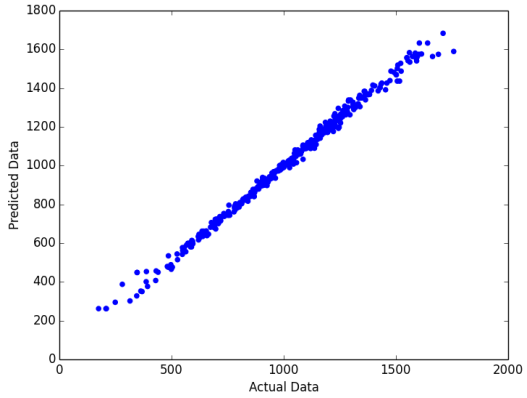


Here is the out of sample results when running the best4linreg.csv with Linear Regression where you can see it fits the data well a perfect correlation and has a very low RSME:





Here is the out of sample results when running the best4linreg.csv with KNN where you can see it does not perform as well as Linear Regression. You can see that by the data not being displayed as linearly and has a very high RSME compared to the Linear Regression.

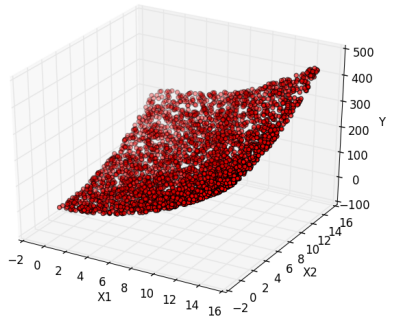



```
/System/Library/Frameworks
In sample results
RMSE:   12.8075548739
corr:   0.998963781593
Out of sample results
RMSE:   24.5480519051
corr:   0.997074444962
```

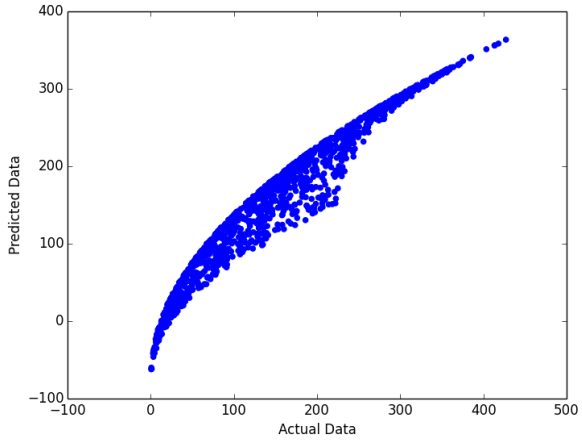
Best4KNN

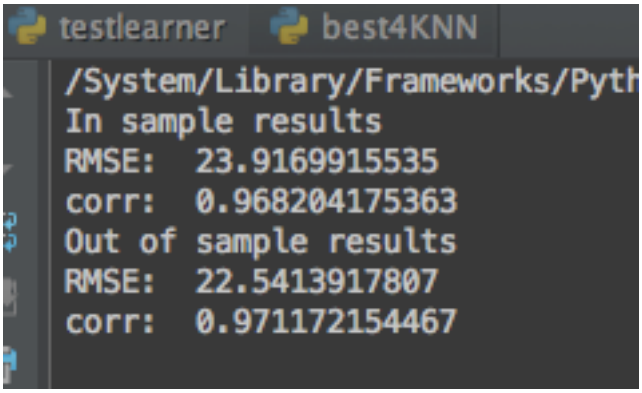
I created a data set that performs better with KNN then with Linear Regression. My dataset consists of two X dimensions and 1 Y dimension. For this data set I made it so that the random sample I generated for each X dimension was not uniform and used 3000 data points and would not give Linear Regression an advantage. I multiplied the X1 and X2 data by 15 do create a wider distribution. Next I took the X1 and X2, squared them, added them together and used this data to create my Y dimension. Squaring each X1 and X2 and having it equal the Y dimension caused the Y to be skewed to the left of the X dimension points making it hard for the linear regression to fit a line through that data with the X dimensions. Also because it is curved and linear regression uses a straight line it doesn't fit as well as the KNN which uses the Euclidean distance and is a better fit for data that does not fit a straight line.

Here is the 3D plot of my best4KNN csv data that I pass in, You can see the Y predict is very heavily skewed:

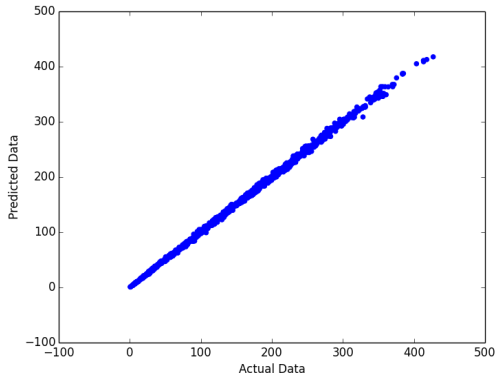


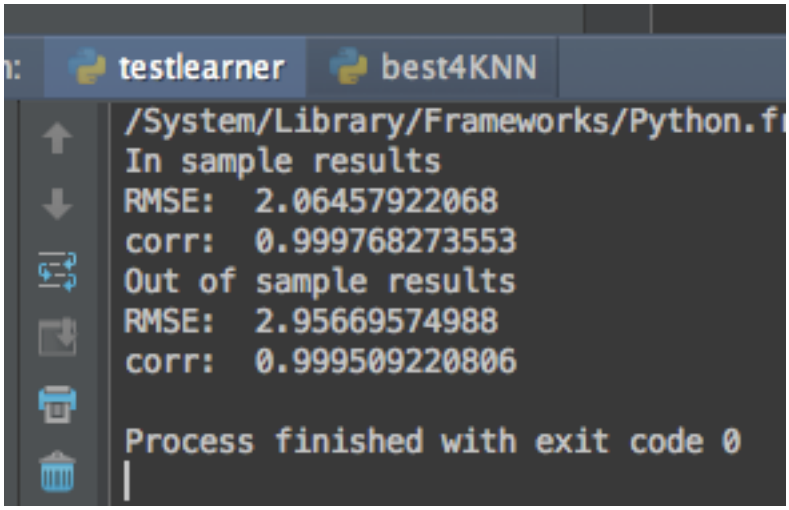
Here is the out of sample results when running the best4KNN.csv with Linear Regression where you can see it it does not fit the data well, has a lower correlation and has a very high RSME:



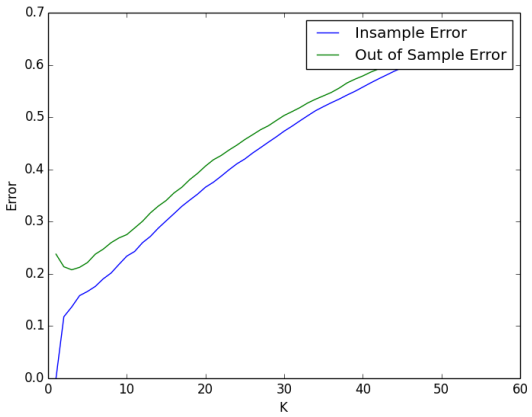


Here is the out of sample results when running the best4KNN.csv with KNN where you can see it performs much better then Linear Regression. You can see the data fits much better and the RMSE is much lower and has a higher correlation then using Linear Regression.



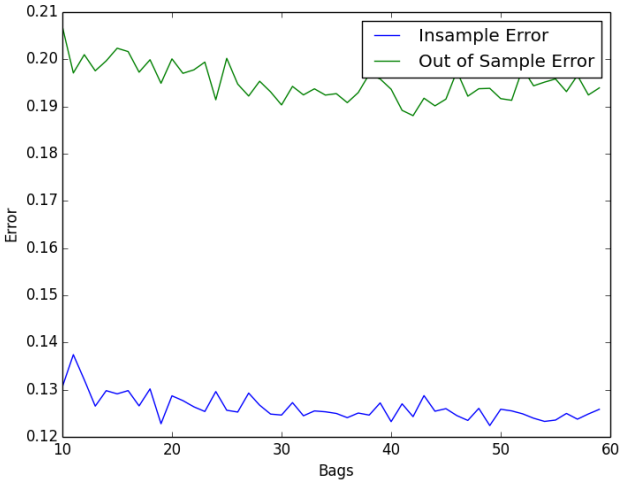


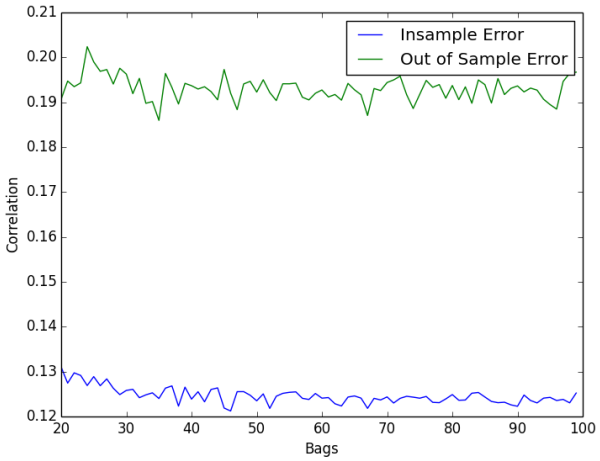
Overfitting with dataset ripple with KNN. For which values of K does overfitting occur?



As we reduce K down to 1, our in sample error approaches zero, it actually becomes 0 when k is equal to 1. Also, as we decrease K our out of sample error decreases but at around k=3 it begins to increase. As out of sample error increases and in sample error decreases is where overfitting occurs. So I would say the values for K that overfitting occurs for our data is when k is less then or equal to 3.

How does performance vary as you increase the number of bags for ripple.csv? Does overfitting occur with respect to the number of bags?

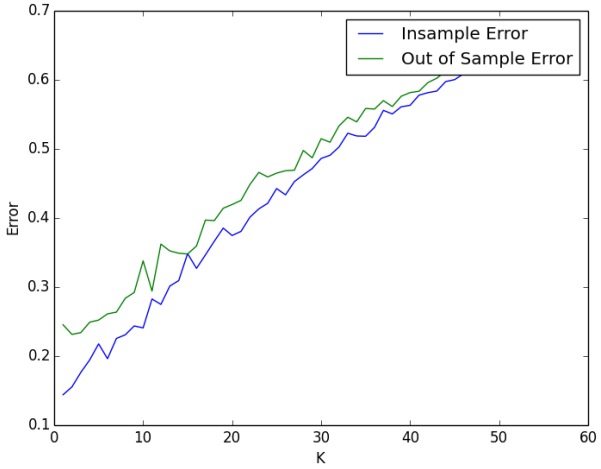




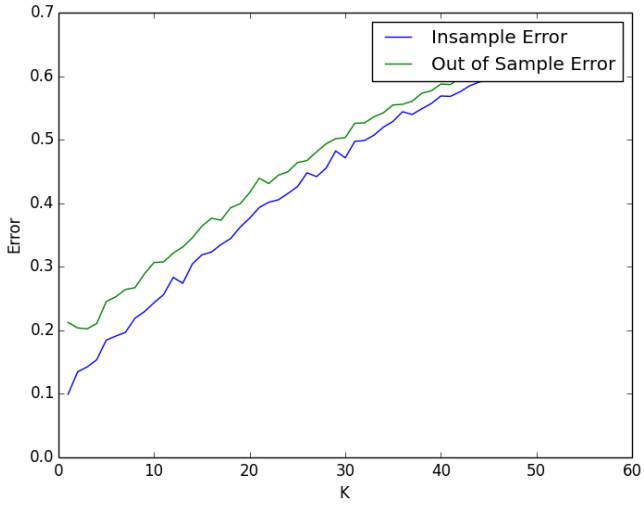
When comparing the insample, out of sample error with K = 3 in respect to the number of bags used, it seems that the overall error is decreasing and performance improving when the bags are greater then 25 but spikes at around 40 bags but that values is still better then just using a KNN learner with no bags. Also when the bags are over 30 there seems to be less overfitting as the bags increase. This is because the ensemble is much more smooth then any of the individual learners by them self.

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

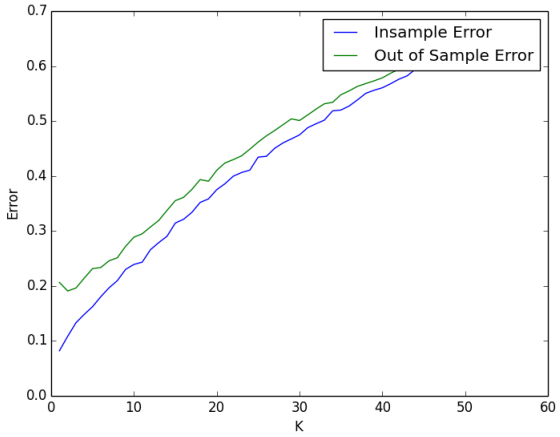
Bags=2; K from: 0 to 60



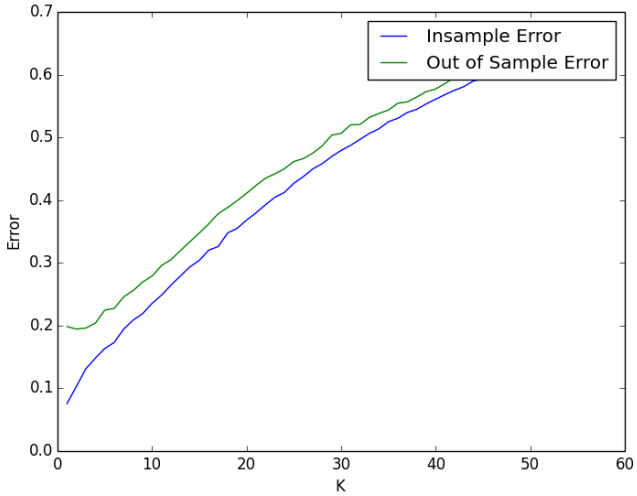
Bags=4; K from: 0 to 60



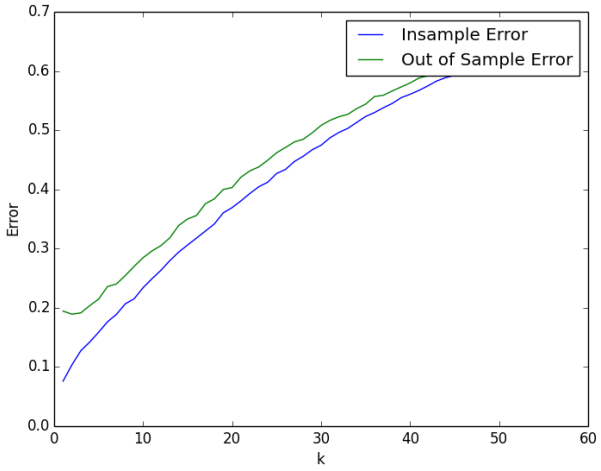
Bags=10; K from: 0 to 60



Bags=20; K from: 0 to 60



Bags=40; K from: 0 to 60



From these graphs it is clear that bagging can reduce overfitting with respect to K for the ripple dataset. You can see the overfitting is greater when the number of bags equal 2 and k ranges from 0 to 60, overfitting occurs when k is around k <= 3 and the overfitting is greater here then when there are 4 bags on the next graph. When the bags increase to 20 and k ranges from 0 to 60, overfitting remains the same when K <= 3 for when the bags are 4, 10, 20 and 40. Overfitting seems to be the same size regardless of if there are 4 or 40 bags for when k<=3. As we increase K the model is less likely to overfit. The reason for this is because overfitting is more likely to occur on 2 to 1 bags of KNN learners vs an ensemble of greater KNN learners and the reason is smoothing.