MC3 – Project 1

Experiment and Report

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).

- **Data Generating Algorithm:**
  In order to create a dataset for the Linear Regression Learner to perform better than KNN Learner, I create a dataset that follow the linear model strictly. The dataset was created with a variable that kept increasing overtime with a minimal amount.
  1. The program will begin with a linear model coefficient with a random value (in my implementation, I used a random integer from 1 to 10).
  2. For every new variable, the program will generate the point follow the below function:

     Point = point_index*linear_coefficient + random_value

     Point_index: an count index that run from 1 to 1000
     linear_coefficient: the coefficient that is generated randomly from 1 to 10
     random_value: is the value that generated randomly from 0 to 1

  3. The process will be repeated until the amount of points was met. In this program, I only generated 1000 points.

  The purpose of this strategy is creating a dataset that follows the linear model strictly so that the error between the predicted value from the linear regression and the real value is very minimal.

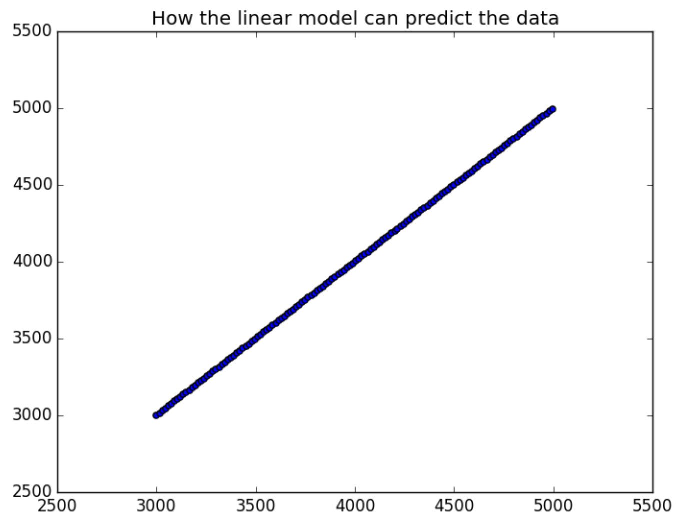

Chart 1 – Chart of the Linear Regression's prediction again its best dataset.

- **Why Linear Regression Learner performs better:**
  The Linear Regression Learner performs better than any other Learners because the created dataset was formatted in the linear format. The value in the dataset was increased with a constant positive rate. Since the dataset was created in a linearly format, the linear data model that was created by Linear Regression Learner can predict the value closer to the dataset and it can result a smaller error between the prediction and the real data.

		Linear Regression Learner	KNN Learner
In Sample Result	RMSE	0.34	1.53
	Correlation	0.99	0.99
Out of Sample Result	RMSE	0.35	1179.89
	Correlation	0.99	9.94

Table 1 – The RMSE and Correlation between Linear Regression and KNN Learners

The Table 1 has shown the Root Mean Squared Errors (RMSE) and Correlations between Linear Regression and KNN Learner. The Linear Regression has shown a strong performance with an RMSE of 0.34+ and strong correlation with a value of 0.99 for both in sample result and out of sample result. The KNN Learner has shown a significant lower performance with RMSE of 1.53 in sample result and 1179.89in out of sample result. Although the KNN Learner's correlation is high in sample of result with value of 0.99, the correlation is 9.94, which means there is no correlation at all between the predicted value and the testing data.

The chart 1 has shown how the Linear Model prediction again its best dataset. The linear line has shown all the prediction was corrected again the testing data in dataset.

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).

- **Data Generating Algorithm:**
  In order to create a dataset that allows the KNN Regression Learner to perform significantly better than the Linear Regression Learner, I created a dataset that follow a polynomial data model in order to ensure that the dataset will not follow a linear model. Since KNN Regression is a non-parametric approach, it will not follow any particular type of data but it will follow the average trending of historical. My approach to create this dataset is ensuring that the dataset is not following any particular linear model using COSINE line. The dataset is created follow the steps below:
  - Create a random coefficient (**A**)
  - Generate the variable using the function below:
    - **Y = cosin(x1/A + x2/A^2)**
      - Ex1: is a random variable from 0 to 100
      - x2: is a random variable from 0 to 100
  - Repeat the process until the number of points was met.

- **Why Linear Regression Learner performs better:**

  The KNN Regression does not follow any physical model but it focuses on the trend of the historical data. Because of that, the KNN Learner will provide the prediction based on how the historical data occurred before while the Linear Regression Learner tries fit everything into the data model that it created. Since the data model is a cosine line, the data was changed repeatedly from -1 and 1. By using the cosine line, the KNN learner is able to capture the trending and behavior of the data and make the correct prediction while the Linear Regression tried to put the wave line into a linear data model.

		Linear Regression Learner	KNN Learner
In Sample Result	RMSE	0.71	0.21
	Correlation	0.12	0.96
Out of Sample Result	RMSE	0.72	0.31
	Correlation	0.08	0.90

Table 2 – The RMSE and Correlation between Linear Regression and KNN Learners

The Table 2 has shown the Root Mean Squared Errors (RMSE) and Correlations between Linear Regression and KNN Learner. As I described above, since the dataset does not follow any linear model, the linear regression learner has performed significant worse with higher errors and lower correlation than the KNN Learner. The Chart 2 also shown how the KNN Learner can predict again the created dataset.

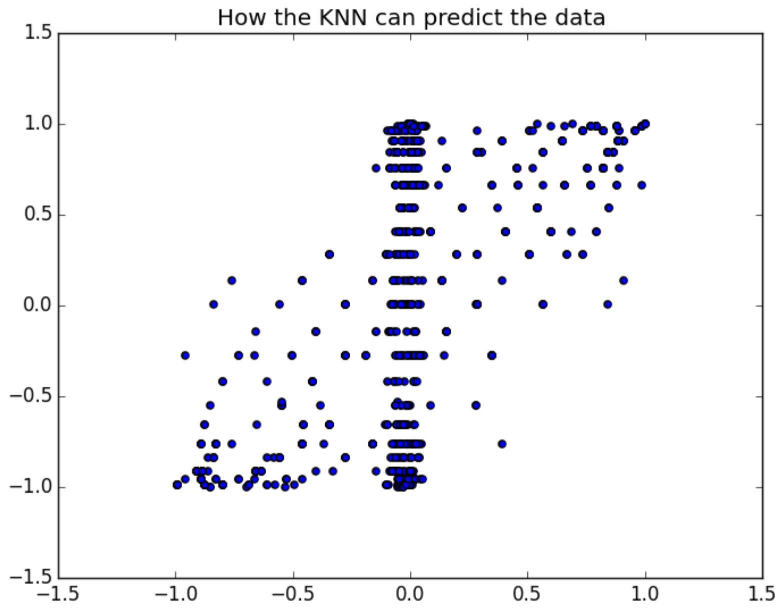

Chart 2 – Chart of the KNN Learner's prediction again its best dataset.

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

In the ripple dataset, each data point is a random float variable between -1.0 and 1.0 and the entire dataset does not follow any particular shape or model. The Chart 2 below has shown how the entire ripple dataset looks like.

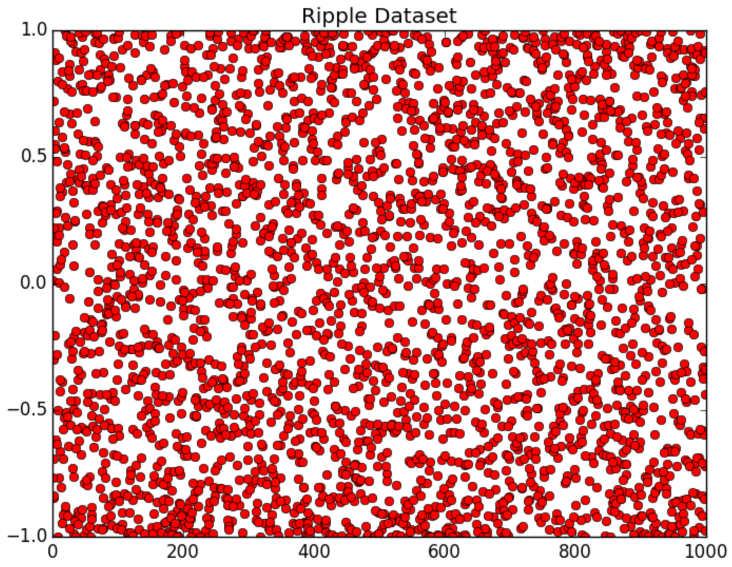

Chart 2 – Chart of the Ripple Dataset.

Since KNN will take a mean of a K variables, the value of K that will definitely allow the over fitting to occur is 1 because with K=1, the regression will follow each data point and it will try to capture every data point into their model without providing the relationship between each data point together. The over fitting may occur in this case since the learner will follow the data and do really well during the Training in sample result and it will do worse during the Test in the out of sample result. The table 3 below has shown how the KNN Learner behave again the ripple dataset with K = 2 and K = 1.

		KNN Learner (K=2)	KNN Learner (K=1)
In Sample Result	RMSE	0.12	0.00
	Correlation	0.99	1.00
Out of Sample Result	RMSE	0.21	0.24
	Correlation	0.95	0.94

Table 3 – The RMSE and Correlation between Linear Regression and KNN Learners using 'ripple' dataset

As we can all see, in the sample result, the KNN Learner provided no error and strong correlation between the predicted results and the dataset with K =1. However, in out of sample result, the KNN Learner provides an error of 0.24 and a weaker correlation. Comparing with K=2, we can see the strong improvement from In Sample Result with lower RMSE and higher Correlation while in the out of sample result, the performance got worse with slightly higher RMSE and lower correlation. The error occurs because the KNN Learner did not capture the relationship between data points in the training dataset.

<u>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?</u>

After use Bagging method in conjunction with KNN Learner with K = 1, the performance was improved significantly with the decreased RMSE in the out of sample result from 0.7 to 0.2. Using Bagging method also allows RMSE in the Sample Result to capture the relationship between data point with a value of 0.070+.

In order to understand the effective of using number of bags in the Bagging method, I have run the method multiple times with different amount of bags. Table 4 below can show how the amount of bags can affect to the performance of the learner.

		20 bags	**25 bags**	**50 bags**	**100 bags**
In Sample Result	RMSE	0.078	0.073	0.072	0.070
	Correlation	0.993	0.994	0.995	0.995
Out of Sample Result	RMSE	0.200	0.191	0.194	0.188
	Correlation	0.958	0.962	0.961	0.963

Table 4 – The RMSE and Correlation of the Bagging method using KNN Learner with K = 1

As we can all see, the more bags we use in the Bagging method, the higher performance we can get with lower RMSE and higher Correlation using the KNN Learner with K = 1.

<u>Can bagging reduce or eliminate over fitting with respect to K for the ripple dataset?</u>

Yes, the Bagging method definitely reduce the over fitting with respect to K for the ripple dataset. However, it still cannot eliminate the over fitting totally. The model will need to be trained multiple times with different value of K and bags in order to find the optimal solution for the ripple dataset. The higher amount of bags I used the lower errors and higher correlation that I can get.

In order to test it clearly, I used 20 and 25 bags in the bagging method since it provided a significant change in the previous test and I test it with KNN method using K = 3, K=2, and K=1.

		20 bags and K = 3	**20 bags and K = 2**	**20 bags and K = 1**	**25 bags and K = 3**	**25 bags and K = 2**	**25 bags and K = 1**
In Sample Result	RMSE	0.13	0.11	0.08	0.13	0.11	0.08
	Correlation	0.99	0.99	0.99	0.99	0.99	0.99
Out of Sample Result	RMSE	0.20	0.19	0.20	0.19	0.19	0.19
	Correlation	0.96	0.96	0.96	0.96	0.96	0.96

Table 5 – The RMSE and Correlation of the Bagging method using KNN Learner with various values of K and bag.

As the Table 5 has shown, by using 20 bags in the bagging method with K=3, K=2, and K=1, the performance has been improved significant with a slightly over fitting occurs. When K was moved from 2 to 1, the RMSE was decreased significantly in sample result but RMSE was slightly increased out of sample result. However, when the number of bags was increased from 20 to 25, the over-fitting problem was decreased. Instead of slightly increasing when K was moved from 2 to 1, the RMSE was unchanged in the out of sample comparing to the significantly decreasing of the RMSE in the sample result.