CS 4803/7646 – MLT (Machine Learning for Trading)

Project Name: Project 3

Student Name: Utkarsh Garg

GTID: 902904045

Graphs for ‘data-classification-prob.csv’

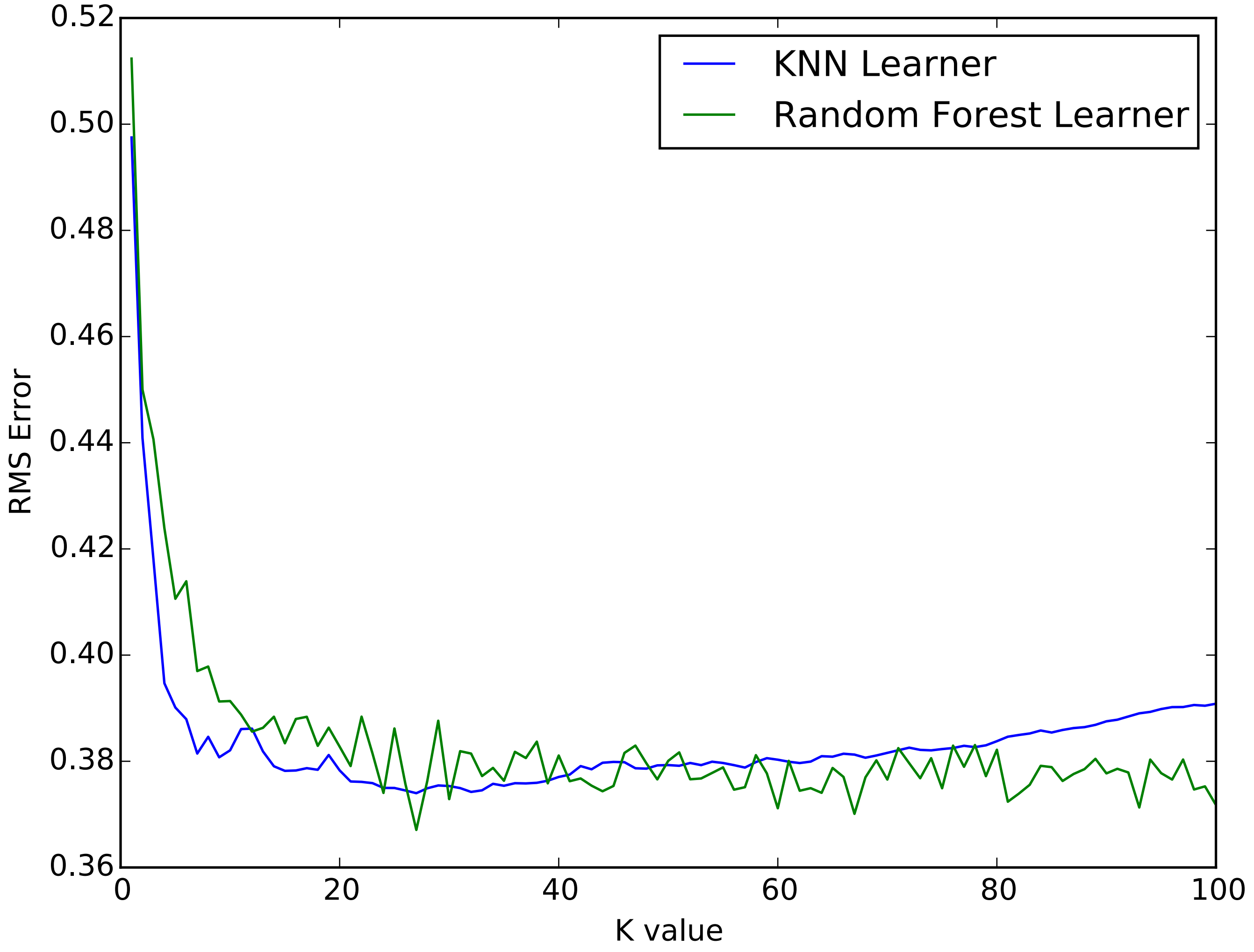


Chart 1. KNN versus Random Forest Learner,
RMS error plotted over K values/Number of trees.

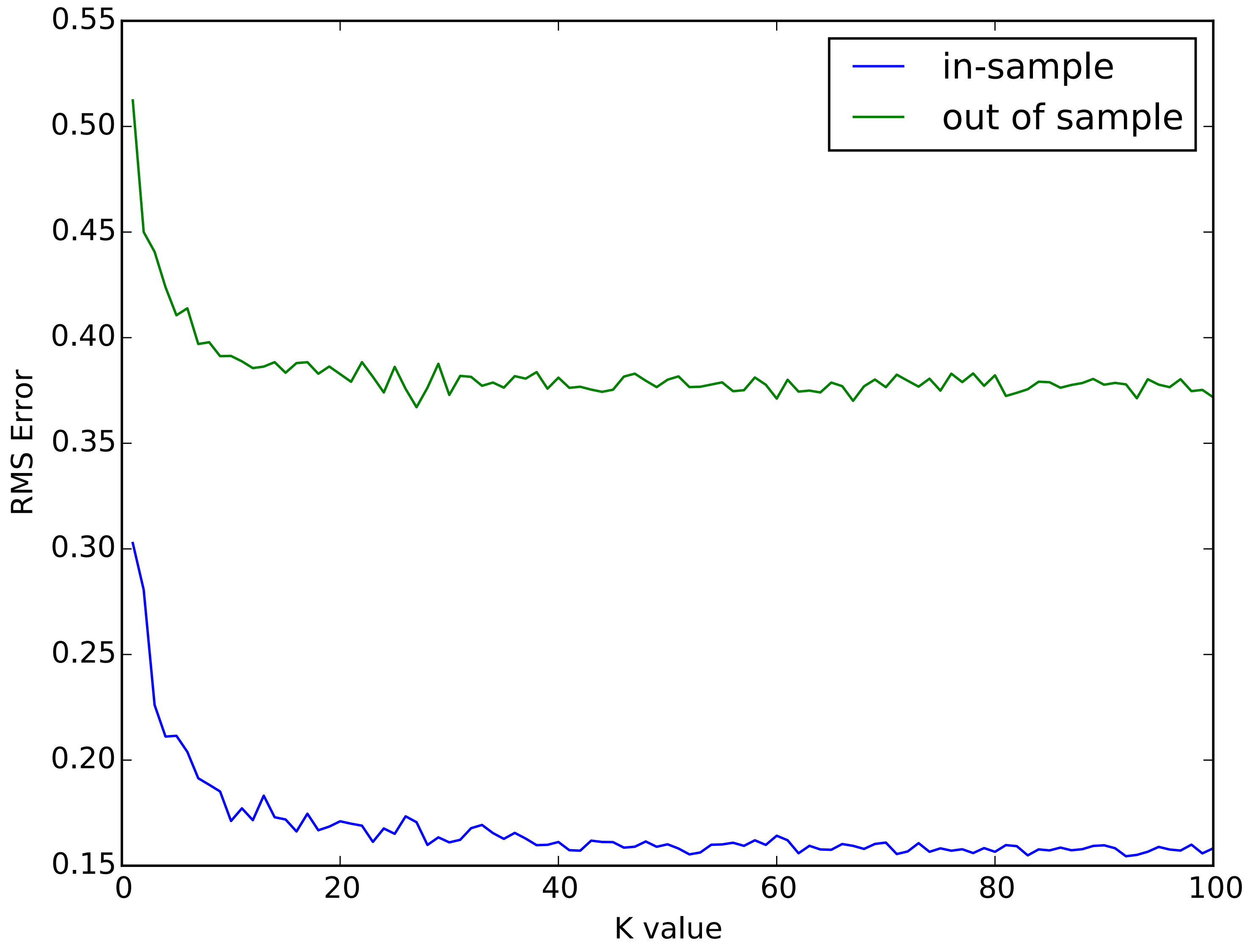


Chart 2. In-sample RMS error vs Out-of-Sample
RMS error for Random Forest Learner.

Graphs for 'data-ripple-prob.csv'

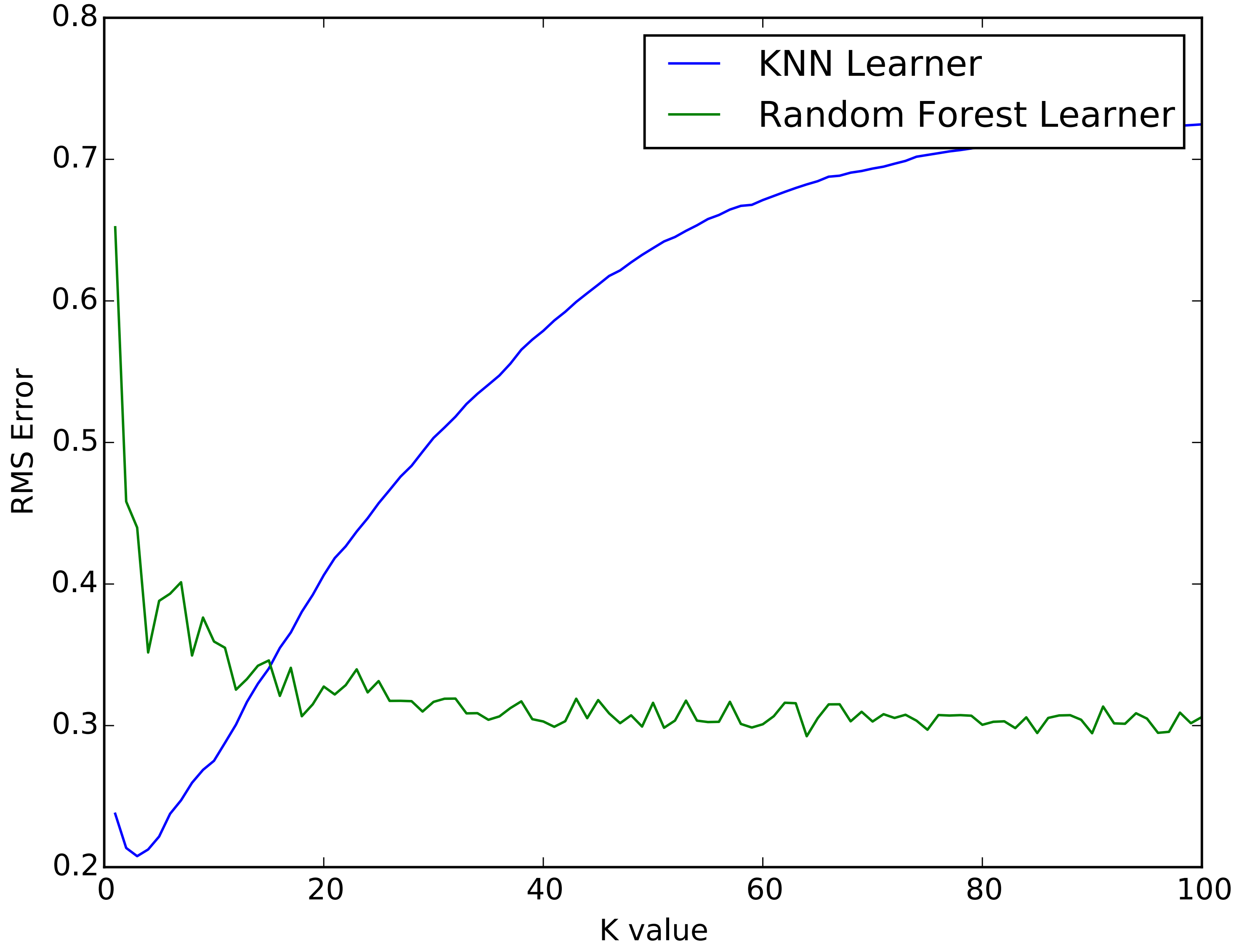


Chart 3. KNN versus Random Forest Learner,
RMS error plotted over K values/Number of trees.

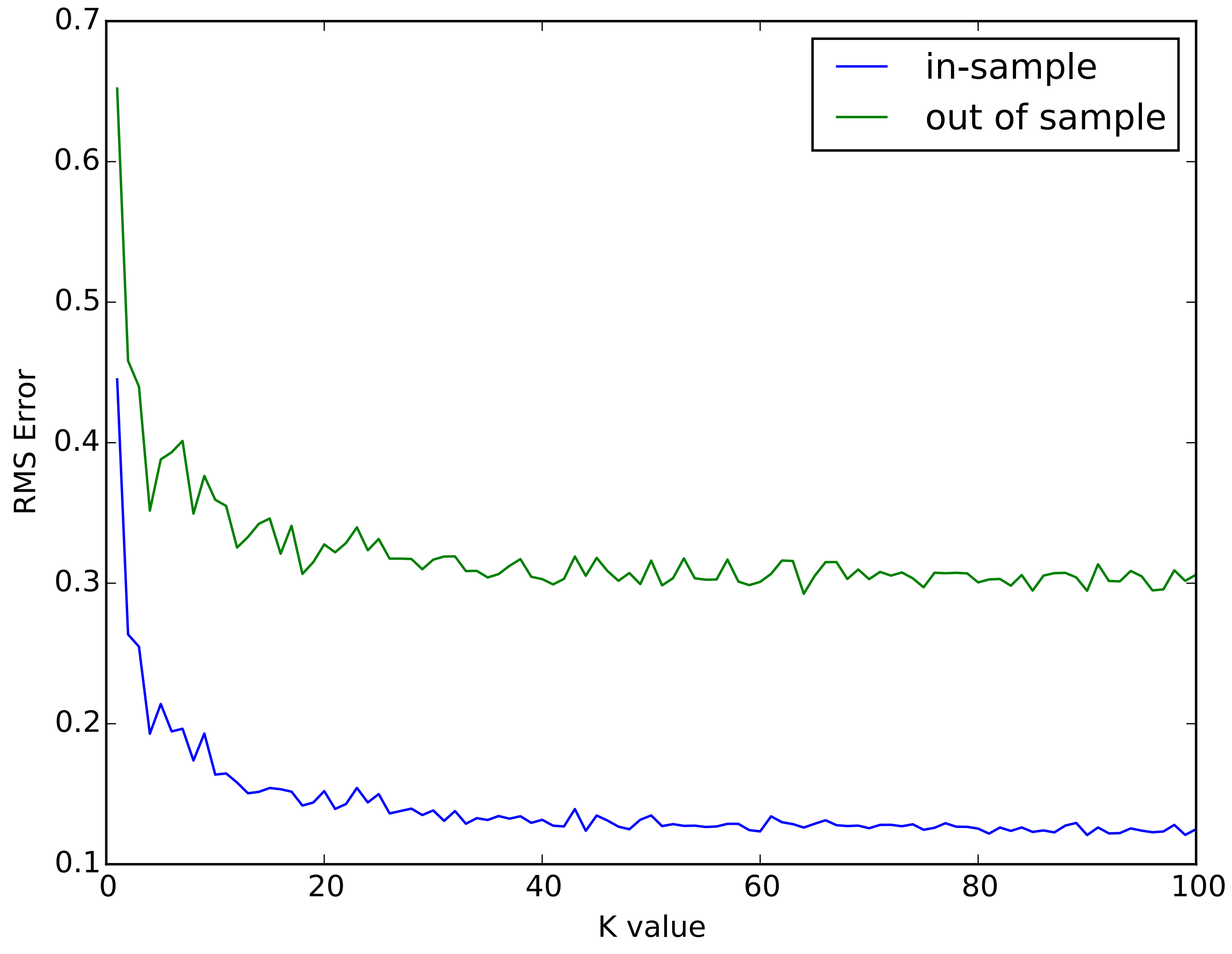


Chart 4. In-sample RMS error vs Out-of-Sample RMS error for Random Forest Learner.

Which approach (KNN vs Random Forests) is more subject to overfitting and why?

- KNN is more subject to overfitting simply because the performance of random forests increases with the number of trees, as seen from charts 1 and 3, whereas KNN starts overfitting with decreasing K. As seen from charts 2 and 4, both in-sample RMS and out-of-sample RMS error decrease with increasing number of trees, and there is no point beyond which, they oppose each other, and thus, there is no clear evidence of overfitting.

Did you see improved performance using more trees in one data set or the other (or both)?

- Yes, performance improved with increasing number of trees in both datasets. Since I am using bagging in my implementation and the data set is relatively large, with a small number of trees, some data might never be trained or be trained only once, when working with a small number of trees. With a large number of trees, it is ensured that all data is used for training, giving a better model.

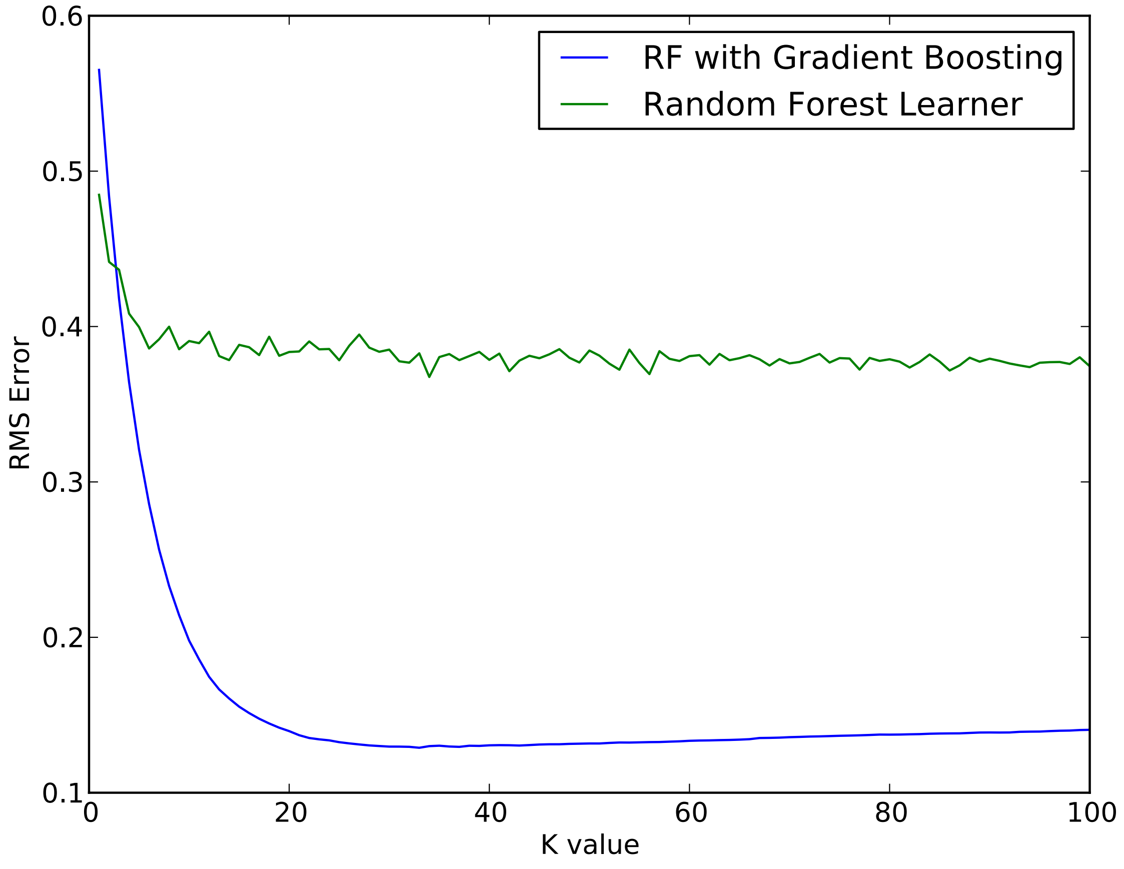
If there was a difference, explain why you think the improvement is better for one data set?

- The correlation coefficient for ripple data goes upto ~ 0.95 whereas the correlation coefficient for classification data goes only up to 0.89. The difference can be attributed to the data contained in the two file: for Y values, whereas the classification file can have only 3 distinct values, the ripple file can have infinitely many, and are varied, thus the training is better. This difference in the data set is the reason for the difference in improvement.

Now that you have compared KNN, linear regression and Random Forests, which approach do you think is best, and why? Does it depend on the dataset? Why?

- I think the approach depends on the data set. For the classification data set I would use Random Forest since its overall RMS error is much lower than KNN and hence the performance is better. For the Ripple data set however, I would use KNN with a k value ~ 3 since that has an RMS much lower than the rest of the graph. There is no overall winner, the choice of algorithm strongly depends on the dataset.

```
CS 4803/7646 - MLT (Machine Learning for Trading)

Project Name: Project 3 - Extra Credit

Student Name: Utkarsh Garg

GTID: 902904045

Implemented Boosting using scikit learn library, specifically,
Gradient Boosting Regressor.

Improvement:

Classification Data:
     Minimum RMS without boosting ~ 0.38
     Minimum RMS with boosting  ~ 0.13

There is also an overall improvement, as can be seen from the graph.
```



Graph 1. RF with and without boosting, on classification data.

```
Ripple Data:
     Minimum RMS without boosting ~ 0.3
     Minimum RMS with boosting  ~ 0.19
```

PARAMETERS FOR BOOSTING (kept same for both data sets. K is the number of trees):

```
GradientBoostingRegressor(n_estimators=k, learning_rate=0.1,
max_depth=5, random_state=0, loss='ls').fit(Xtrain, Ytrain)
```

Extra Code for Boosting:

```
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import GradientBoostingRegressor


gradientRmsError = np.zeros([100])


#RMS for gradient boost

est = GradientBoostingRegressor(n_estimators=k, learning_rate=0.1,
max_depth=5, random_state=0, loss='ls').fit(Xtrain, Ytrain)
gradientBoostmse = mean_squared_error(Ytest, est.predict(Xtest))
gradientRmsError[k-1] = gradientBoostmse


linename = ['RF with Gradient Boosting', 'Random Forest Learner']
createComparisonPlot('K value', 'RMS Error', kArray, gradientRmsError,
rfRmsError, 'RMSComparisonGradient.pdf', linename)
```