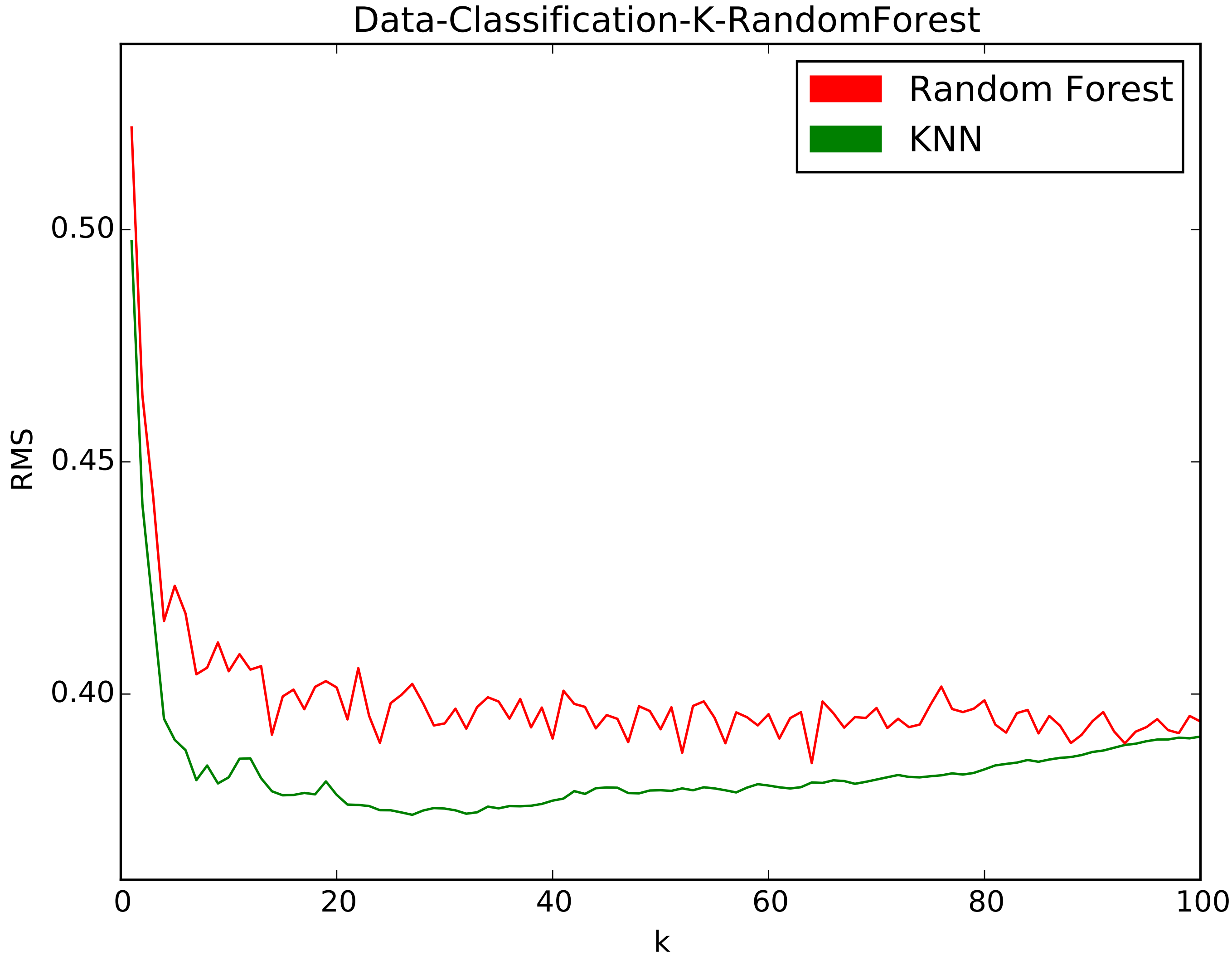
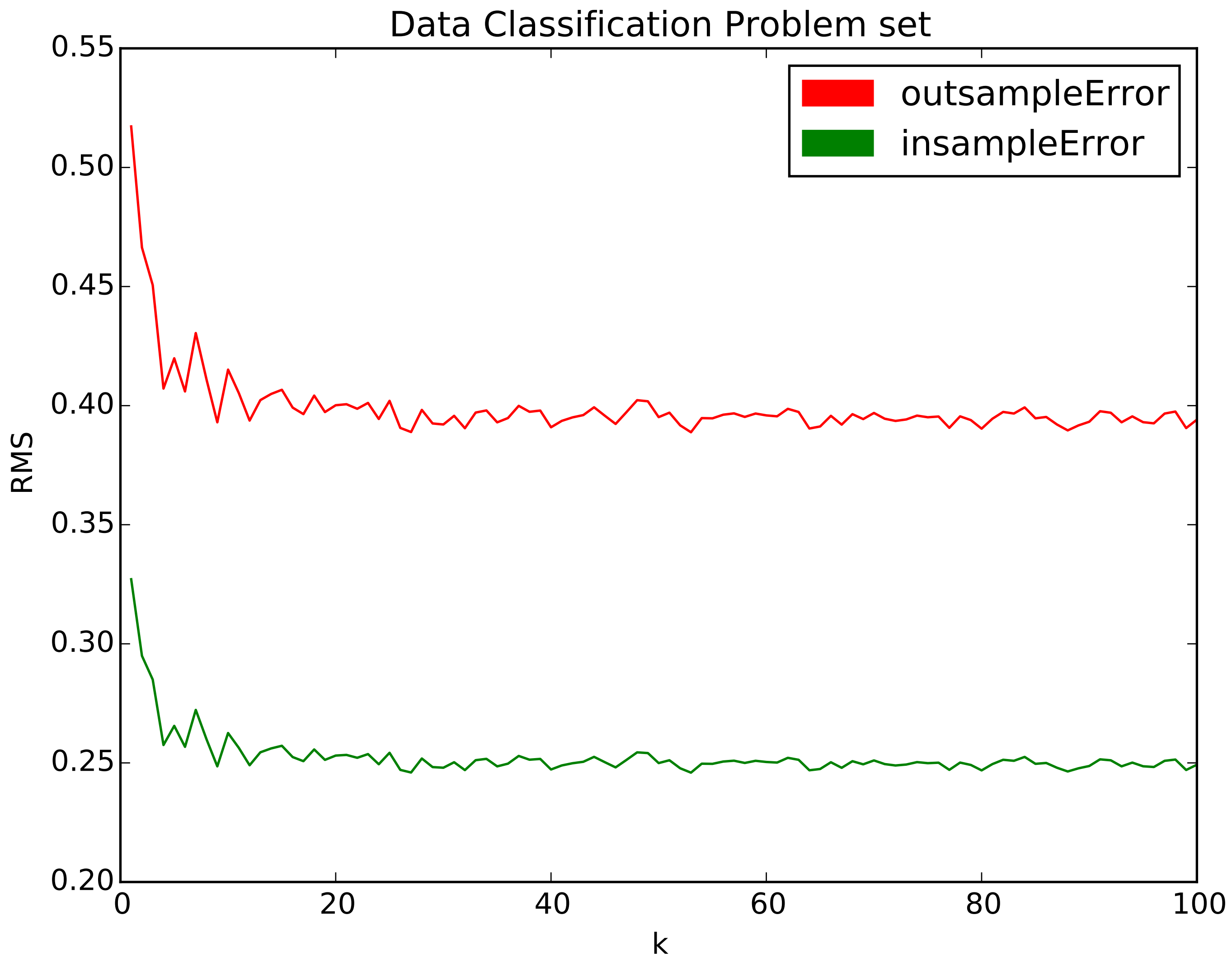
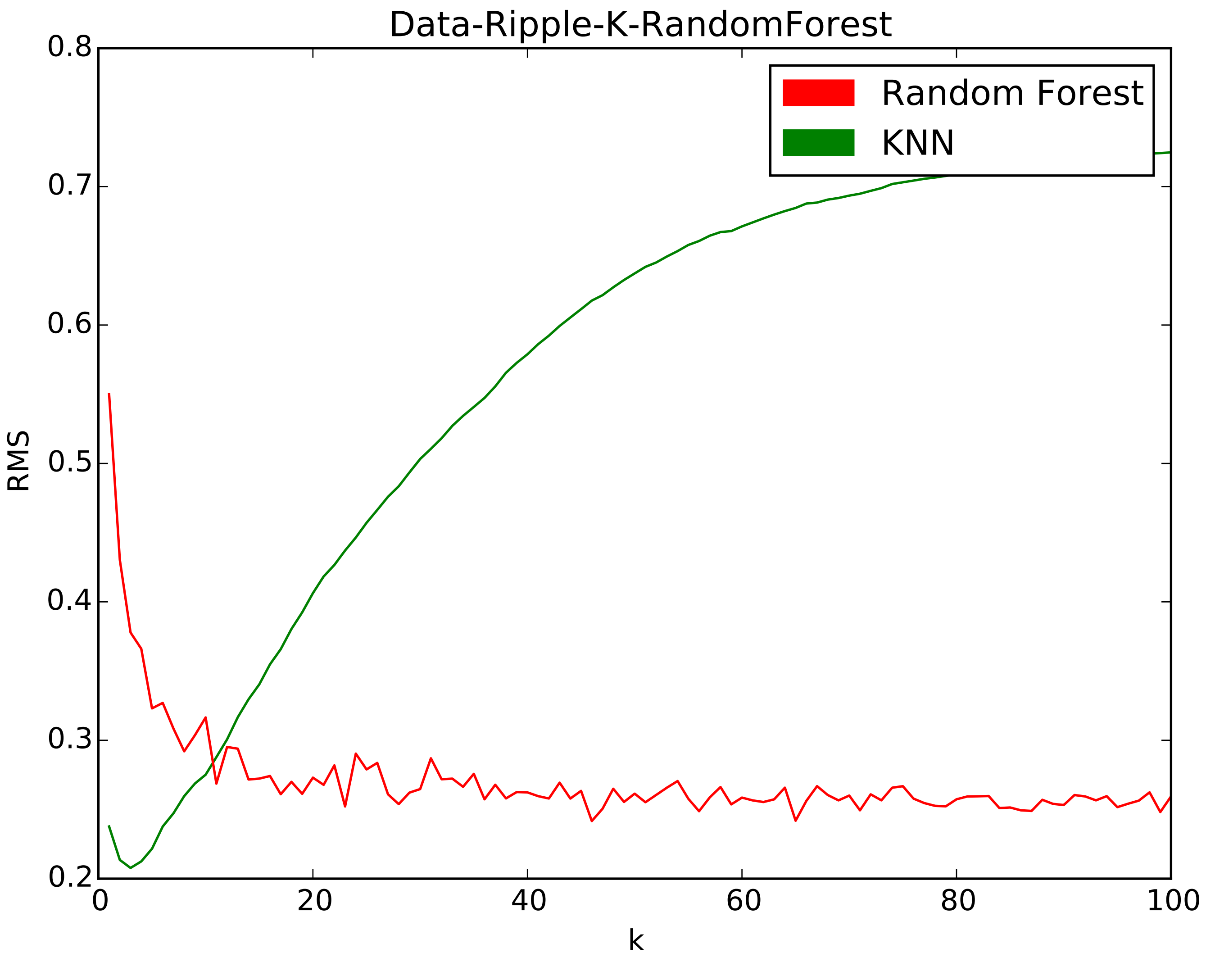
Akshat Harit
903090915

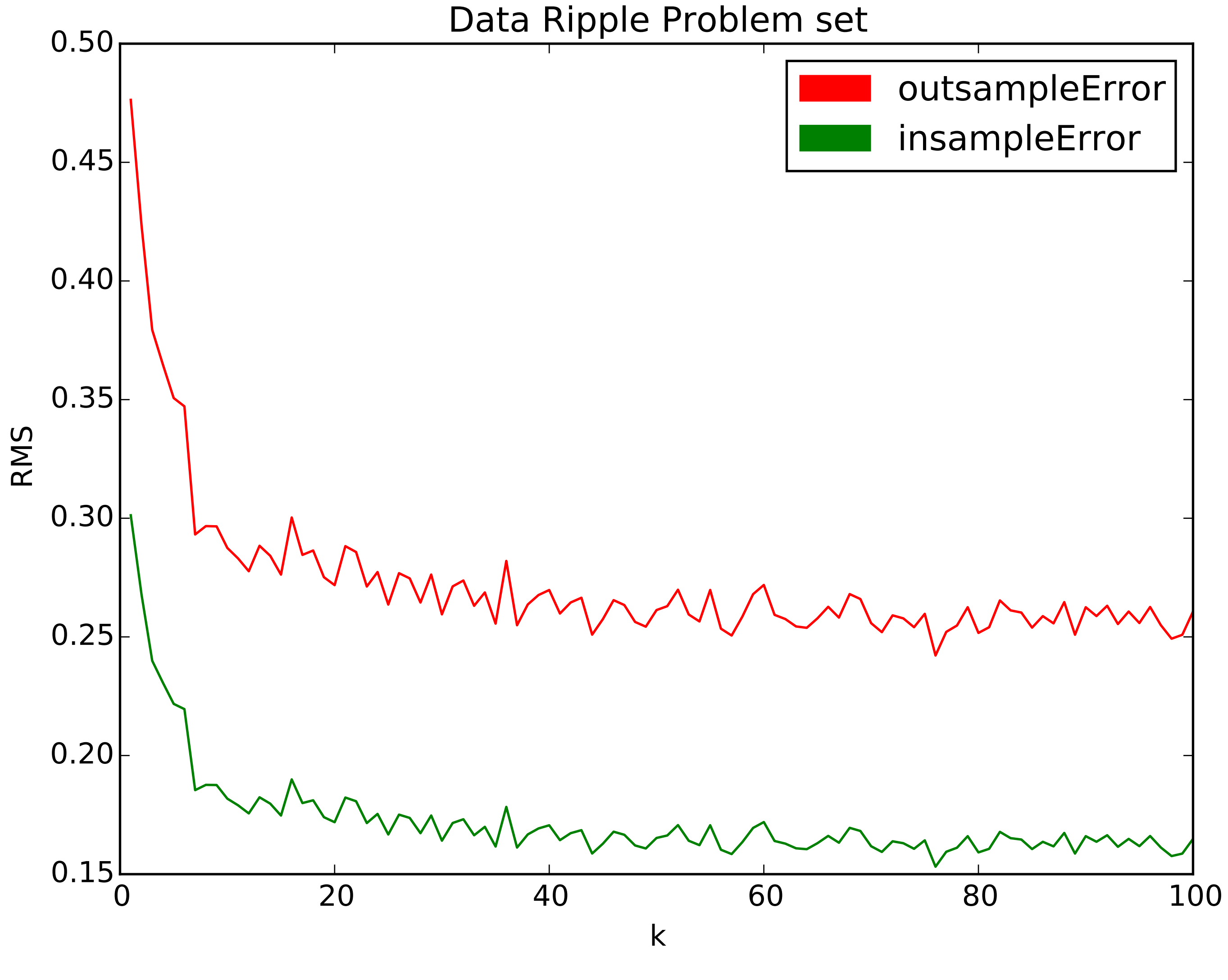
1. **Which approach (KNN vs Random Forests) is more subject to overfitting and why?**
   KNN is more subject to over fitting. This is because Random forests by nature take a random set of data per tree. So no tree has the complete data and hence over fitting is not easily found. Possibly a very large number of trees will be required for Random Forest to over fit. KNN meanwhile for low values of K, due to just outputting average of nearby pints over fits easily
2. **Did you see improved performance using more trees in one data set or the other (or both)?**
   Till a certain number of trees, performance improved when adding more trees in the forest. However, after a point this performance gain wasn't that much even with adding trees. Also due to inherent randomness in trees, there is no guarantee that the same number of trees will give the same performance in another iteration. Furthermore, increasing number of trees doesn't monotonically increase performance, again due to randomness.
3. **If there was a difference, explain why you think the improvement is better for one data set.**
   For ripple data set we observer that the error drop is larger. Also there seems to be a trend that increasing more trees would further decrease error. For classification data set, we observe both lesser error decrease, and less effect of larger number of trees. This could be because of the fact that ripple data set has different y values, thus randomization and then averaging out still has scope of improvement for increasing trees. Whereas classification data set has only -1,0,1 as output i.e. leaf nodes, so further addition of trees doesn't seem to increase accuracy that much.
4. **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 data set? Why?**

   No particular approach can be used for all data sets. Even in these data sets, we see that for RMS error, KNN outperforms random forest till 100 trees. All the methods i.e linear regression, KNN and random forests are better at producing different models. For example, linear regression is very good, as the name suggests for linear models. KNN and random forests fit non-linear models. In literature, random forest is generally better for classification tasks. KNN is in general better for regression. However KNN and random forests also have different training, query times. Linear regression and random forest need time for training, where if online training is required this isn't a very good approach. There are ways to use them online, but KNN can directly be used for online training. For querying however, the position is reversed. KNN requires the maximum amount for querying. Also if we are dealing with very large data sets, as each query for KNN needs to parse the whole data, querying time can be very slow, unless parallel algorithms aren't used. KNN however isn't trivial to parallelize. Linear regression and random forests have characteristics that give fast query times. Also random forest can take advantage of parallelization easily.


Data-Classification-K-RandomForest
🟥 Random Forest
🟩 KNN
RMS
0.50
0.45
0.40
0
20
40
60
80
100
k


Data Classification Problem set
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
RMS
0
20
40
60
80
100
k
🟥 outsampleError
🟩 insampleError


Data-Ripple-K-RandomForest
Random Forest
KNN
RMS
k
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
20
40
60
80
100


Data Ripple Problem set
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
RMS
0
20
40
60
80
100
k
outsampleError
insampleError