ITCS 6156: Machine Learning
Assignment 3: k-Nearest Neighbor and Boosting

Submitted By: Ashwin Venkatesh Prabhu
UNCC ID: 800960400
Email: avenka11@uncc.edu

Collaborated with: Febin Zachariah
UNCC ID: 800961027
Email: fzachari@uncc.edu

k-Nearest Neighbors Algorithm

K Nearest Neighbor (k-NN) algorithm is widely used for predictive analytics. It can be used for both classification and regression predictive problems. Some characteristics of k-NN are, 1) It does not make any assumption on the underlying data distribution. This is a very useful feature, as practical data seldom obeys the theoretical assumptions made. 2) It is a lazy algorithm, which means it does not use training data points to make any generalization. There is no explicit training phase. Lack of generalization during the training phase means k-NN will keep all the training data and will use it during the testing phase, which makes for a costly testing phase, in terms of time as well as memory.

Assumptions:

1) k-NN assumes that the data is in a feature space. The data can be scalar or possibly even multidimensional vectors. Since the points are in feature space, they have a notion of distance. Even though Euclidean distance is the commonly used distance metric case, this need not be the required metric in case of some data points.
2) Each training data has a set of vectors and class label associated with each vector. In the simplest case, it will be positive or negative. But k-NN will work well with arbitrary number of classes.
3) The number "k" in this algorithm decides how many neighbors influence the classification. The value of "k" is usually an odd number if the number of classes is 2.

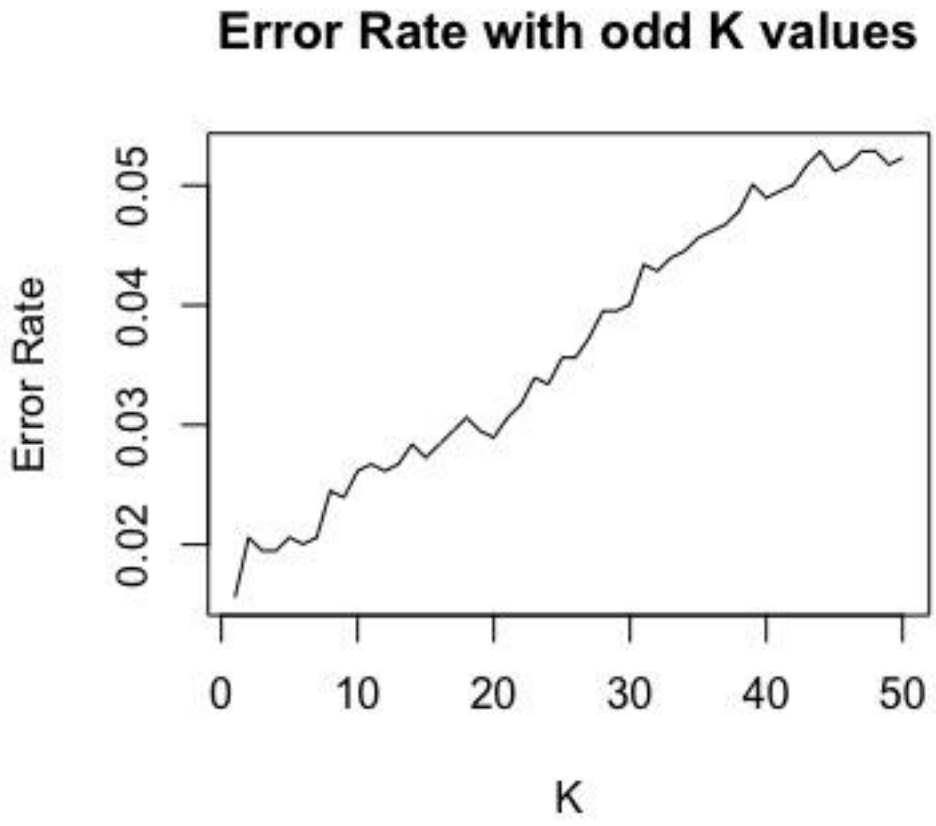
k-NN algorithm behaves differently with different values of k. 1) When k is 1, the algorithm is simple called nearest neighbor algorithm. Consider x to be the point to be labeled. Let y be the point nearest to x. Now as per the nearest neighbor algorithm, the label of y is assigned to x. This scenario holds good only if the number of data points is not very large, else this procedure will lead to a huge error. If the number of data points is very large, then chance of x and y having the same label becomes very high. 2) When k is K, then the algorithm tries to find the K nearest neighbors and do a majority voting. Typically, K will be odd in case of two class types. For example, if K = 7, then there will be 4 instances of class type 1 and 3 instances of class type 2. In this case, the new point will be labeled as class type 1. 3) When k = K and the neighbors are weighted. A very common thing to do is weighted k-NN where each point has a weight which is calculated using its distance. For example, the neighboring data points will have a higher weight than the farther points. With the increasing value of k, the accuracy also increase, but so does the computational cost.

Optical Recognition of Handwritten Digits

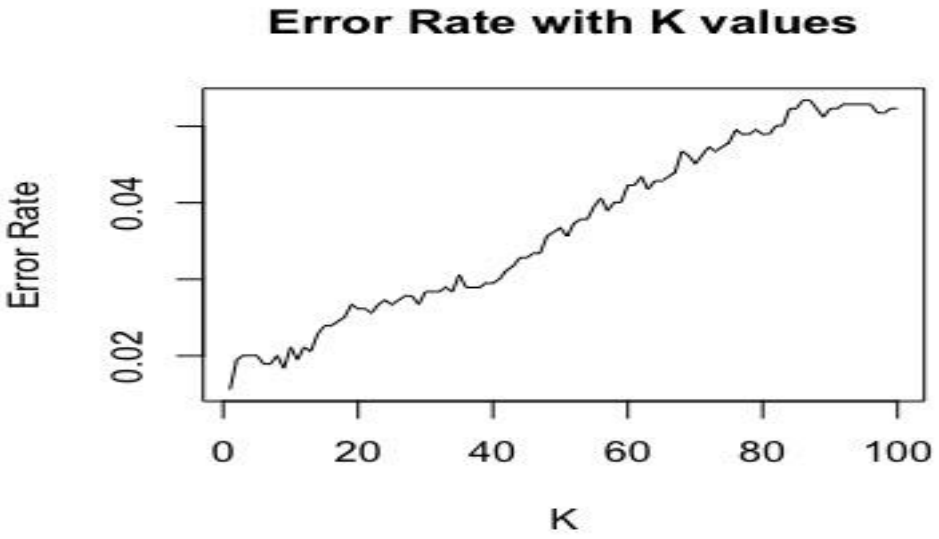
Implementation Details:

1) The program is written using R programming language
2) The program using "knn" method from the "class" library to execute k-NN algorithm on the dataset.
3) The dataset for train and test data does not have a header. So, the headers for the feature variables are names V1 to V64. The header for class variable is named V65.

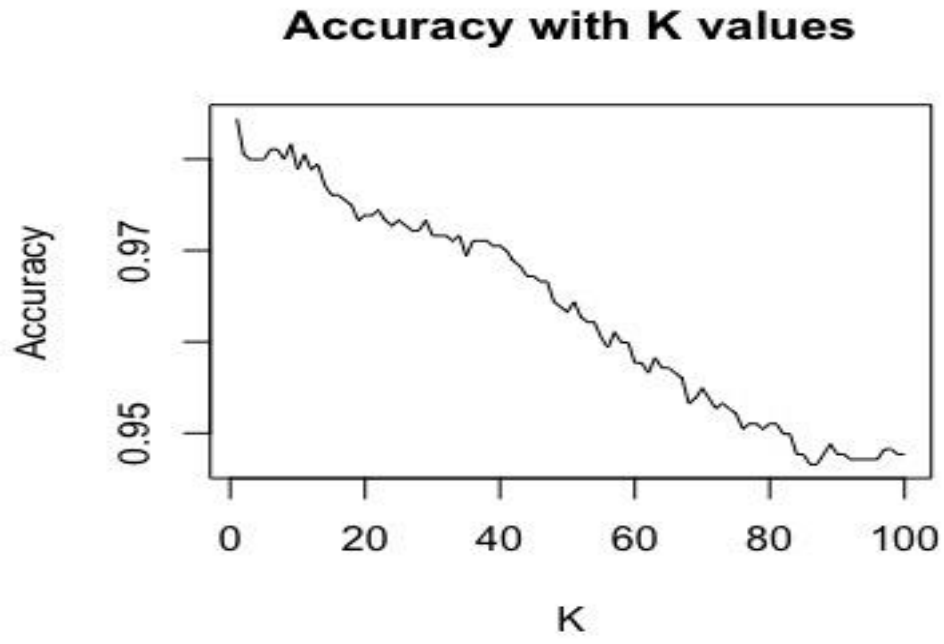
4) The train data is imported and stored in “train_data” variable
5) The test data is imported and stored in “test_data” variable
6) “knn” method is used execute k-NN algorithm. As parameters to the method, the train dataset, test dataset and the class variables for the train dataset is passed. This method is executed in a for loop with k value ranging from 1 to 100. This k value is passed as parameter to the “knn” method at every iteration.
7) There is one more parameter named “l” passed to “knn” method which needs a numeric value. Default value is 0. The value of this parameter defines the minimum vote for definite decision. The value of “l” needs to be less than k-1. Having tried different values of “l” for this dataset, default value of “l” which is 0, gives the best result.
8) At every iteration, the accuracy achieved is calculated and the stored in a list. A graph is plotted for error rate for k values ranging from 1 to 100.
9) Error rate graph for only odd values of k (k=1,3,5,…,99). There is not much fluctuation in error rate with slight increase from <0.01 when k=1 to 0.05 when k = 99.



10) Error rate graph for all values of all values of k from 1 to 100. There is not much fluctuation in error rate with slight increase from <0.01 when k=1 to 0.05 when k = 100.



11) Accuracy graph for all values of all values of k from 1 to 100. There is not much fluctuation in accuracy except for a slight decrease as the k value increases.



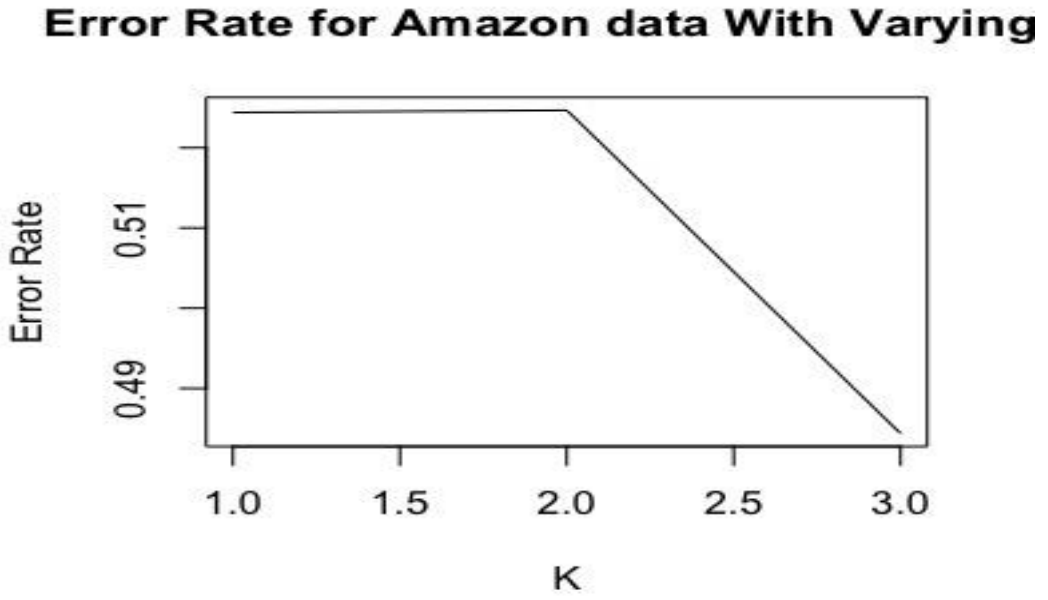
12) Conclusion: Since we have more than two class types in the current dataset, the k value is not restricted to an odd number. As for the accuracy, it is supposed to increase as the value of k increases, but here, we can see a slight decrease in the accuracy as the k value increases. Nevertheless, the accuracy of prediction is still very close to 95% even at k = 100, which is very high.

Amazon Baby Product Review Dataset

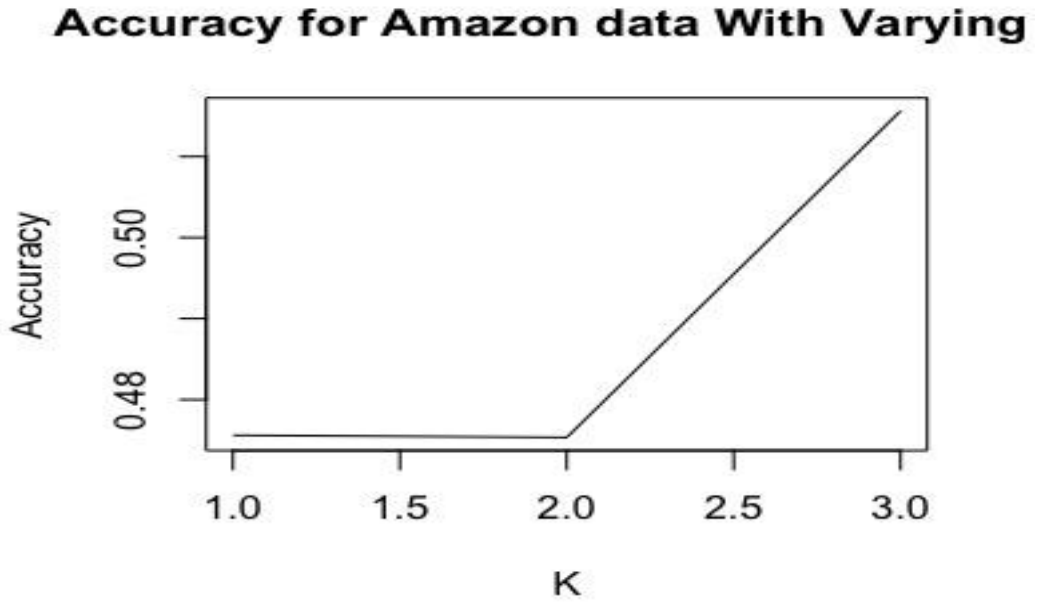
Implementation details:

1) The program is written using R programming language.
2) The program using "knn" method from the "class" library to execute k-NN algorithm on the dataset.
3) The dataset has three headers namely, "name", "review", "rating".
4) For previous two assignments (decision trees and neural networks), we have used a sentiment score analysis method defined using "plyr" and "stringr" library from R, which calculates the positive word count, negative word count and sentiment score of the reviews. But this method threw an error: "RError: Too many ties in knn" when used for knn for amazon dataset. Have also tried different values of k. On further analysis, we realized that the issue was with our sentiment analysis method as it was producing not so accurate sentiment scores which led to too many similar values while knn tried to calculate the similarity among the nearest neighbors. Also, tried to run the same code in High Performance Cluster, which gave a segmentation fault error. The conclusion was that there was a need for a more accurate sentiment analysis method, and we had two options – Rsentiment (from R) and Vader Sentiment (from python nltk library). Vader Sentiment gave use better output, and hence decided to use Vader sentiment to calculate the sentiment score for the train and test dataset.
5) The SentimentIntensityAnalyzer method takes the text review as input and output four values, namely, "pos" (positive score of the review), "neg" (negative score of the review), "neu" (neutral score of the review), "compound" (normalized value of the sum of all sentiment scores)
6) After calculating the sentiment analysis for the train dataset (stored in "sentiment_train.csv"), and test dataset (stored in "sentiment_test.csv"), the new files are used as input for predictive analysis.
7) "knn" method is used execute k-NN algorithm. As parameters to the method, the train dataset, test dataset and the class variables for the train dataset is passed. This method is executed in a for loop with k value ranging from 1 to 3. This k value is passed as parameter to the "knn" method at every iteration. A "k" value more than 3 gives "too many ties in knn" error.
8) There is one more parameter named "l" passed to "knn" method which needs a numeric value. Default value is 0. The value of this parameter defines the minimum vote for definite decision. The value of "l" needs to be less than k-1. Having tried different values of "l" for this dataset, default value of "l" which is 0, gives the best result.
9) At every iteration, the accuracy achieved is calculated and the stored in a list. A graph is plotted for error rate for k values ranging from 1 to 3.

10) Error rate graph for all values of all values of k from 1 to 3. We can see from the below graph that the error rate is steady for k = 1,2, but decreases for k = 3.



11) Accuracy graph for all values of all values of k from 1 to 3. We can see from the below graph that the accuracy is steady for k = 1,2, but increases for k = 3



12) Conclusion: Since we have more than two class types in the current dataset, the k value is not restricted to an odd number. As for the accuracy, the value of k is steady for k = 1, 2, but increases slightly for k = 3. The accuracy for k = 3 is 51.98%.

Boosting

Have used two algorithms to demonstrate the boosting technique here. They are, 1) C5.0, which is widely used algorithm when it comes to decision trees. C5.0 is an advancement of C4.5, which is basically an extension of ID3 algorithm. 2) Gradient Boosting, which is basically about "boosting" many weak predictive models into a strong one in the form of an ensemble of weak models. A weak predict model is any model which works a little better than the random guess.

Optical Recognition of Handwritten Digits

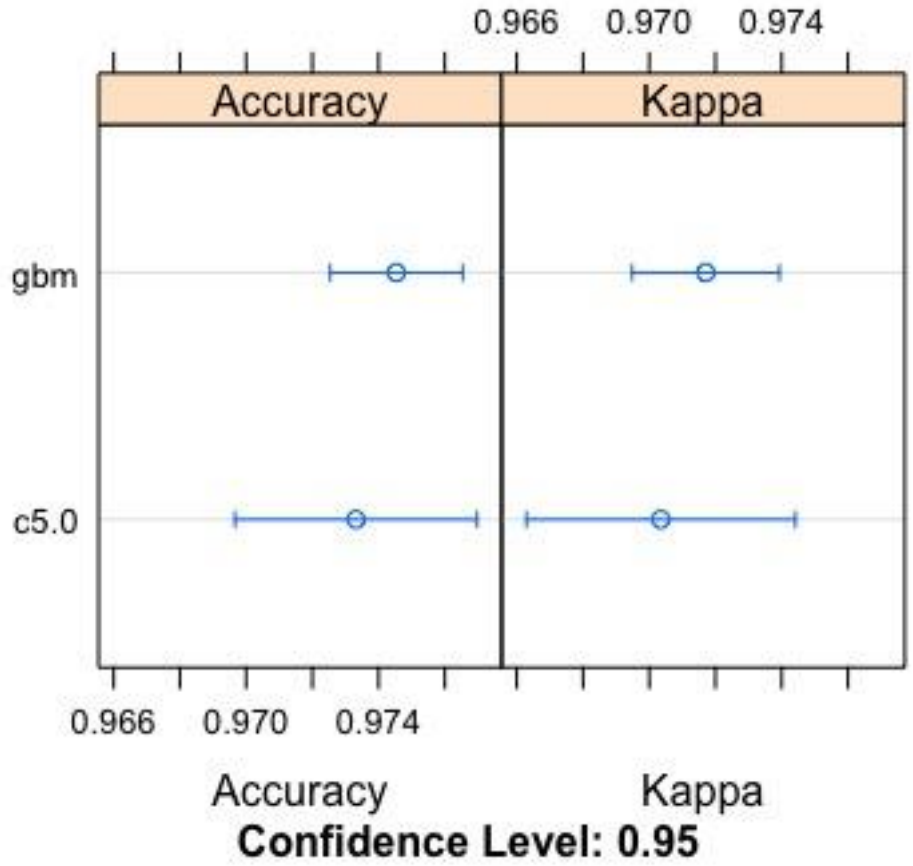
Implementation details:

1) The program is written using R programming language
2) The program uses "class", "caret", "caretEnsemble", "C50" libraries to implement the boosting technique
3) The dataset for train and test data does not have a header. So, the headers for the feature variables are names V1 to V64. The header for class variable is named V65.
4) The train data is imported and stored in "train_data" variable
5) The test data is imported and stored in "test_data" variable
6) "train" method is used to train the data. The parameters passed to this method are, formula (V65~.), training data, method ("C5.0", "gbm") for C5.0 and Gradient boosting, metric ("Accuracy". By default, the value is "RMSE" and "Rsquared" for regression, "Accuracy" and "Kappa" for classification), trControl (This parameter will decide how the train function will act. A trainControl object is passed. This object is created with three parameters, method ("repeatedcv" for repeated training and test splits), number (5, specifies the number of folds/number of resampling iterations), repeats (3, for repeated k-fold cross validation only: number of complete sets of folds to compute)
7) Two separate models are trained for C5.0 algorithm (model stored in variable "model_c50") and Gradient Boosting algorithm (model stored in variable "model_gbm").

8) By using “resamples” method, we can find the accuracy of the trained models. The model generated using gradient boosting algorithm is slightly better with approximately >97% accuracy. There is not much difference between accuracies of the two algorithms.

```
> summary(boosting_results)

Call:
summary.resamples(object = boosting_results)

Models: c5.0, gbm
Number of resamples: 15

Accuracy
       Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
c5.0 0.9607  0.9693 0.9751 0.9733  0.9777 0.9843    0
gbm  0.9685  0.9725 0.9738 0.9745  0.9771 0.9804    0

Kappa
       Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
c5.0 0.9564  0.9658 0.9724 0.9704  0.9753 0.9826    0
gbm  0.9650  0.9695 0.9709 0.9717  0.9746 0.9782    0
```

9) Accuracy for the boosting model is represented below:



10) “Predict” method is used for predictive analysis using the models created on the test data.

11) Gradient boosting gives a better accuracy, 95.63%, than the boosted C5.0 algorithm, 95.49%.
12) Conclusion: As the definition states, boosting algorithm takes a set of weak learners and combines them to create an ensemble, which is a strong learner. This is clearly visible by looking at the accuracy of C5.0 decision tree algorithm, which is an astounding 95%, which is much better than 75%, the accuracy which was there for the decision tree algorithm implemented in Assignment 1.

Amazon Baby Product Review Dataset

Implementation details:

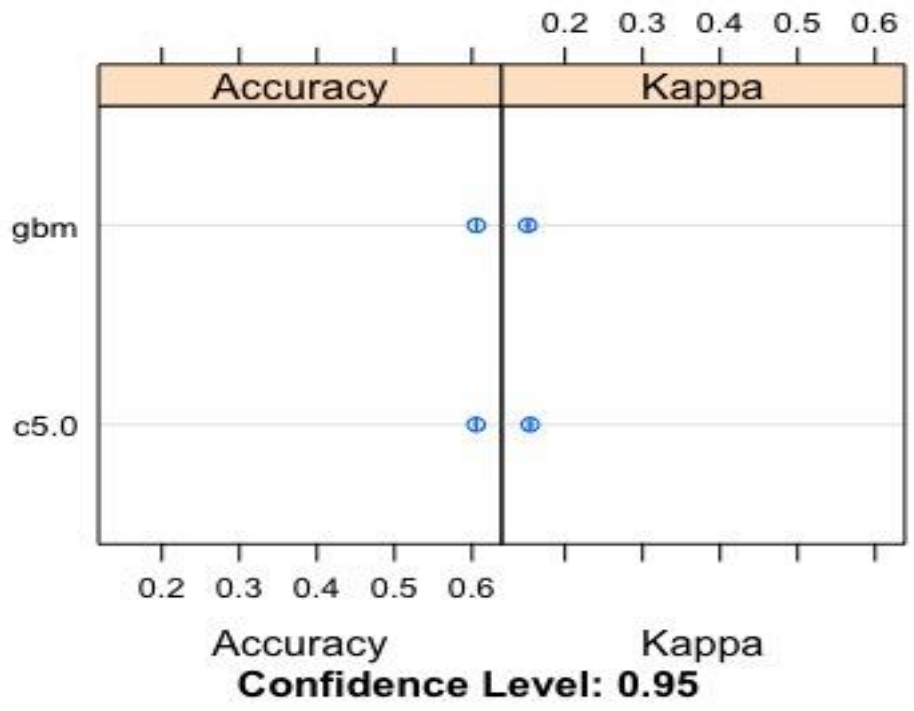
1) The program is written using R programming language.
2) The program uses "class", "caret", "caretEnsemble", "C50" libraries to implement the boosting technique
3) The dataset has three headers namely, "name", "review", "rating".
4) For previous two assignments (decision trees and neural networks), we have used a sentiment score analysis method defined using "plyr" and "stringr" library from R, which calculates the positive word count, negative word count and sentiment score of the reviews. But this method threw an error: "RError: Too many ties in knn" when used for knn for amazon dataset. Have also tried different values of k. On further analysis, we realized that the issue was with our sentiment analysis method as it was producing not so accurate sentiment scores which led to too many similar values while knn tried to calculate the similarity among the nearest neighbors. Also, tried to run the same code in High Performance Cluster, which gave a segmentation fault error. The conclusion was that there was a need for a more accurate sentiment analysis method, and we had two options – Rsentiment (from R) and Vader Sentiment (from python nltk library). Vader Sentiment gave use better output, and hence decided to use Vader sentiment to calculate the sentiment score for the train and test dataset.
5) The SentimentIntensityAnalyzer method takes the text review as input and output four values, namely, "pos" (positive score of the review), "neg" (negative score of the review), "neu" (neutral score of the review), "compound" (normalized value of the sum of all sentiment scores)
6) After calculating the sentiment analysis for the train dataset (stored in "sentiment_train.csv"), and test dataset (stored in "sentiment_test.csv"), the new files are used as input for predictive analysis.
7) The train data is imported and stored in "train_data" variable
8) The test data is imported and stored in "test_data" variable
9) "train" method is used to train the data. The parameters passed to this method are, formula (V65~.), training data, method ("C5.0", "gbm") for C5.0 and Gradient boosting, metric ("Accuracy". By default, the value is "RMSE" and "Rsquared" for regression, "Accuracy" and "Kappa" for classification), trControl (This parameter will decide how the

train function will act. A trainControl object is passed. This object is created with three parameters, method ("repeatedcv" for repeated training and test splits), number (5, specifies the number of folds/number of resampling iterations), repeats (3, for repeated k-fold cross validation only: number of complete sets of folds to compute)

10) Two separate models are trained for C5.0 algorithm (model stored in variable "model_c50") and Gradient Boosting algorithm (model stored in variable "model_gbm").
11) By using "resamples" method, we can find the accuracy of the trained models. Both the models have approximately 60% accuracy.

```
> summary(boosting_results)

Call:
summary.resamples(object = boosting_results)

Models: c5.0, gbm
Number of resamples: 15

Accuracy
       Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
c5.0 0.6028  0.6038 0.6057 0.6053  0.6065 0.6073    0
gbm  0.6042  0.6051 0.6055 0.6058  0.6066 0.6080    0

Kappa
       Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
c5.0 0.1483  0.1544 0.1549 0.1552  0.1569 0.1599    0
gbm  0.1452  0.1508 0.1524 0.1520  0.1527 0.1582    0
```

12) Accuracy for the boosting model is represented below:



13) "Predict" method is used for predictive analysis using the models created on the test data.

14) Accuracy of the models over test data is also same with C5.0 giving 60.10% accuracy and Gradient Boosting giving 60.12% accuracy.
15) Conclusion: As expected, boosting technique using C5.0 and Gradient Boosting for amazon also shows some improvement in accuracy level even though not a significant improvement as desired.

References

1) https://www.researchgate.net/publication/275828927_VADER_A_Parsimonious_Rule-based_Model_for_Sentiment_Analysis_of_Social_Media_Text
2) http://www.nltk.org/_modules/nltk/sentiment/vader.html
3) https://github.com/nltk/nltk/tree/develop/nltk/sentiment
4) https://www.linkedin.com/pulse/sentiment-analysis-using-vader-muthuraj-kumaresan
5) https://mathbabe.org/2013/04/04/k-nearest-neighbors-dangerously-simple/
6) https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/
7) http://machinelearningmastery.com/machine-learning-ensembles-with-r/