Improving Classification accuracy by using KNN and Decision-Trees on 20newsgroup dataset

Bharath Shamasundar

Department of Computer Science

bbst59@mst.edu

24 February 2018

Abstract

In this project the 20newsgroup dataset is classified on the basis of the article category that each article belongs to. There are 20 article categories and each article belongs to one of them. We make use of the K-Nearest Neighbor and Decision Tree Classifiers in order to classify the articles on the basis of the category label. We also optimize the Nearest Neighbor classifier to find the optimal K for both the Accuracy score as well the F-measure.

1 Introduction and Problem to be Solved

Making meaning from raw data is one of the toughest challenges facing the computer science community. This not only requires excellent data preprocessing techniques but also one must have meaningful strategies to tackle the cleaned data. Such an interesting problem is present as part of the project here. The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The problem to be solved here how to classify the articles according to their article category.

There are 20 different article categories and 18828 articles present in the dataset. We use KNN and Decision Tree Classifiers in order to classify these articles.The rest of the report is organized as follows: Section 2 contains a brief overview of the data, Section 3 contains the procedure of classifying using KNN's and Decision Trees. Section 4 contains the optimization of KNN by selecting the best-K for both the accuracy score and the f-measure,along with comparisons of Best-KNN vs Decision Tree accuracies and scores vs K analysis. Section 5 deals with the accuracy comparisons for each of the article categories.The report ends in Section 6 with the conclusion and final remarks.

2 Data Overview

The 20newsgroup is a corpus document which contains a bunch of information but mainly deals with 18828 articles related to over 20 groups. Each of the category ID is the target label for the articles that are present in the dataset. The target ID ranges from 1-20.The categories along with the category ID are as follows:

1. alt.atheism
2. comp.graphics
3. comp.os.ms-windows.misc
4. comp.sys.ibm.pc.hardware
5. comp.sys.mac.hardware
6. comp.windows.x
7. misc.forsale
8. rec.autos
9. rec.motorcycles
10. rec.sport.baseball

11. rec.sport.hockey
12. sci.crypt
13. sci.electronics
14. sci.med
15. sci.space
16. soc.religion.christian
17. talk.politics.guns
18. talk.politics.mideast
19. talk.politics.misc
20. talk.religion.misc

Some of the categories are closely related, like for example (comp.sys.ibm.pc.hardware / comp.sys.mac.hardware) while some of them are not, like for example (sci.med and sci.space). The raw data has 4 main sub-headings and they are

- The target names meaning the various categories for the entire dataset
- The target values(in the range of the 20 categories)
- Data in terms of characters
- Data description for each article.

3 Classification using KNN and Decision Trees

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

1. In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

2. In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.Both for classification and regression, a useful technique can be to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor.The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.[2]

Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. [1]

3.1 KNN on the 20newsgroup Dataset

Scikit-Learn provides a nicely wrapped library in order to use KNN directly on the desired dataset. It is present under *sklearn.neighbors*. Randomly splitting the test-train dataset is not advised, since the performance is not consistent across the entire data. Hence we use the technique of cross-validation where the entire train data is split across multiple folds. One such fold will be reserved for testing purposes, whereas one other fold will be used to validate the train data. This happens on a rotation basis across the folds such that all folds will be used as test/train samples. The average of the cross fold score will the final accuracy or f-measure. We applied a 5-fold cross validation where 4 parts are used for train/validate and 1 part for testing. The average accuracy percentage for KNN without any optimization is 4.84% before feature selection and 4.95% after feature selection whereas the f-measure is 2.77% before feature selection and 4.26% after feature selection. This performance is poor and mainly because of the lack of optimization over K and also because of the data quality.

3.2 Decision Tree on the 20newsgroup dataset

Scikit-Learn provides a nicely wrapped library in order to use Decision Tree directly on the desires dataset. It is present under *sklearn.tree*.Randomly splitting the test-train dataset is not advised, since the performance is not consistent across the entire data. Hence we use the technique of cross-validation where the entire train data is split across multiple folds. One such fold will be reserved for testing purposes, whereas one other fold will be used to validate the train data. This happens on a rotation basis across the folds such that all folds will be used as test/train samples. The average of the cross fold score will the final accuracy or f-measure. We applied a 5-fold cross validation where 4 parts are used for train/validate and 1 part for testing. The average accuracy percentage for Decision Tree is 4.70% before feature selection and 4.97% after feature selection, whereas the f-measure is 4.51% before feature selection and 4.99% after

feature selection.

From both the above classifiers we can see that there is an improvement in the accuracy score as well as f-measure when there is feature selection applied on the dataset. This is mainly because when the number of features are far too many and sparse in nature the classifier does not enough data points to determine the category label, hence reducing the test accuracy. Whereas when the features are well defined in nature, the performance goes up as there is definitive boundaries that are present for which the classifier can determine it's bins. Also feature selection means less Data,leading to simpler Models, lower variance, and hence models generalize well.

4 Selecting the best-K

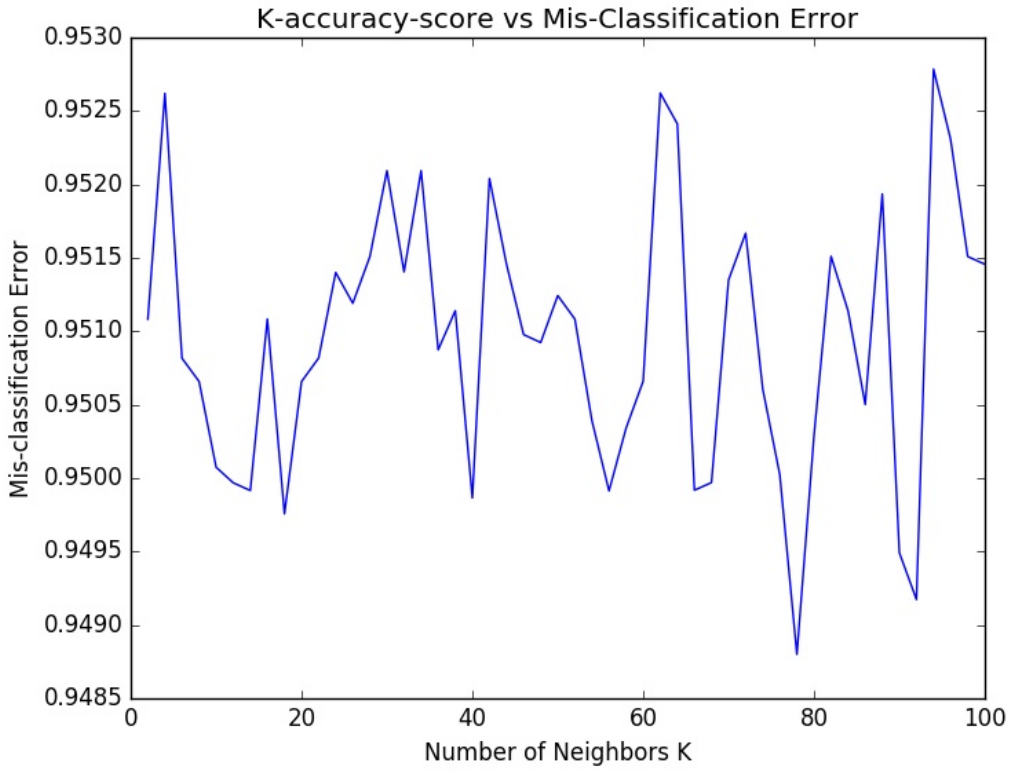

Figure 1: K vs Mis-classification error for Accuracy Score

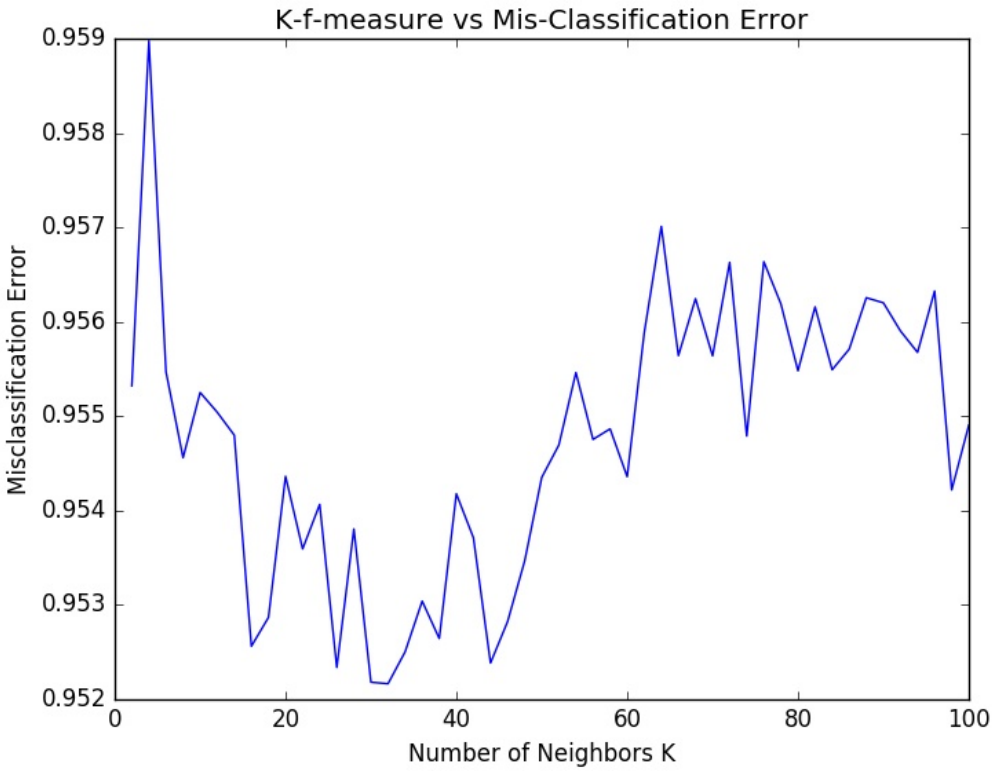

Figure 2: K vs Mis-classification error for F-Measure

From the previous section we certainly can come to the conclusion that we have not selected the best K, which is the number of neighbors in order to classify. Selecting the best K could lead to better performance measures. The steps used to iteratively find the best K is as follows:

1. Take a range of numbers based on the number of features. Here I went ahead with 1-100 since 100 is the maximum number of features that are present after feature selection

2. Once the range is set calculate Accuracy and f-measure score for each K in the range with a step size. The step size chosen by me here is 2.

3. Determine the mean squared error for each K in the list. The K having the lowest Mis-classification error will be the one which is the best K.

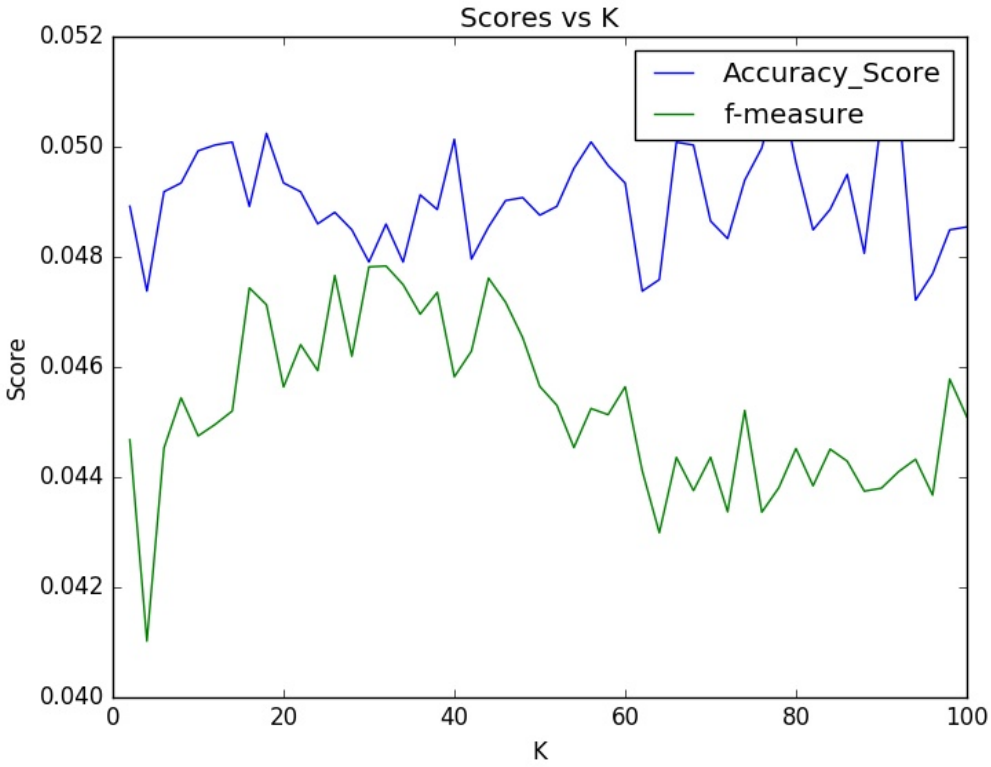

Figure 3: Accuracy and F-measure variation with number of K's

The mis-classification graphs are as shown in Figure 1 and 2. From these graphs one can determine that the best K for accuracy score is 78, whereas the best K for f-measure is 32 and when KNN is applied with those K's respectively the accuracy score is 5.12% and the f-measure is 4.60%. Figure 3 shows as to how with varying number of K's the accuracy as well as the f-measure scores respond. From this graph it is clear that the optimal K's were chosen to be correct.

Figure 4 shows the comparison of KNN after applying the best-K against Decision Tress. From the graph one can notice that even though there is an increase in the accuracy score percentage, the f-measure seems better off without using the best-K. This is mainly because KNN's are "Lazy Learners" as they do not build any classification model and only base their prediction on test

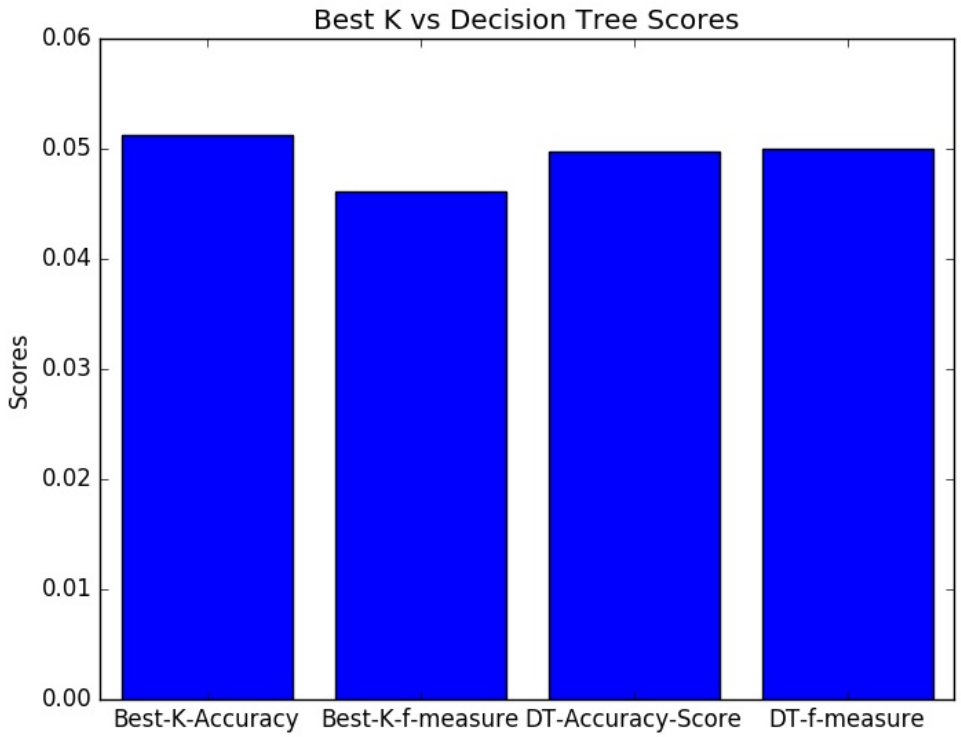

Figure 4: KNN with best-K vs Decision Tree Score Comparison

observations whereas Decision Trees are "Eager Learners" because they first build a classification model on the training dataset before being able to actually classify an unseen observation from test dataset.

5 Accuracy Comparison for Each Article Category

Figure 5 shows the accuracy comparison for each article category for KNN with the best K vs Decision Tree. From the graph we can see that there are nine article categories for which KNN performs better and ten where the decision tree performs better. In one article category they both seem to be performing on par. The article categories where KNN performs better are the following:

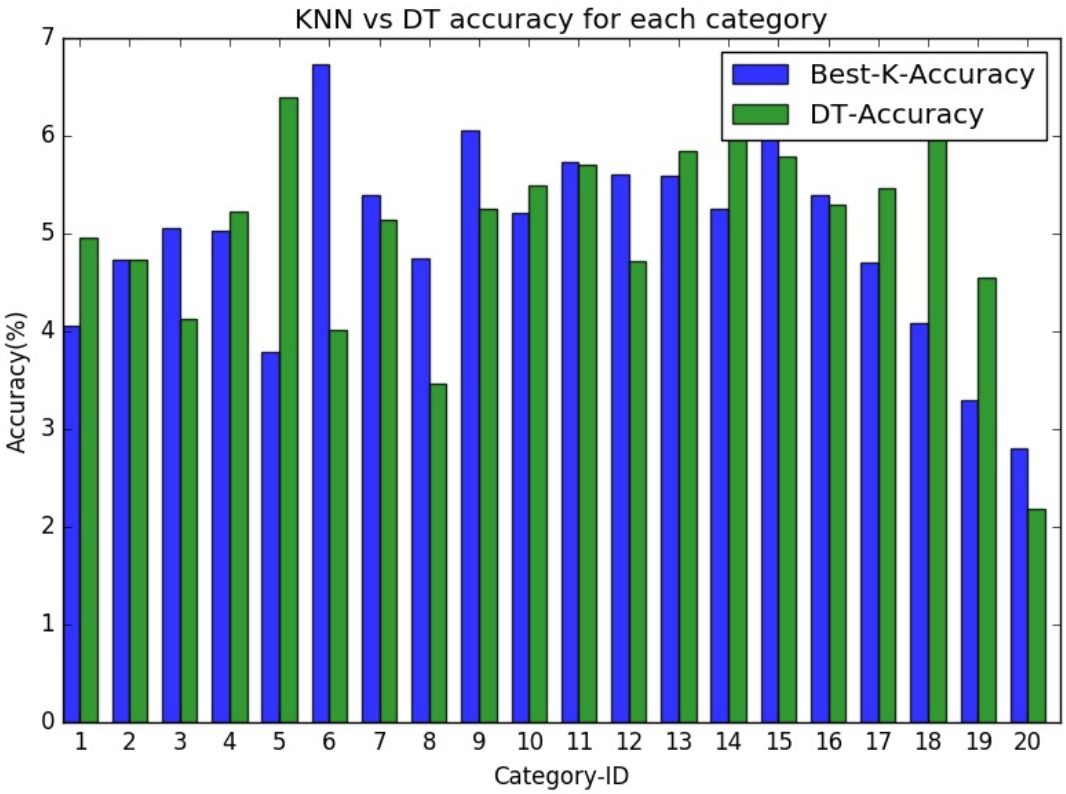

Figure 5: KNN vs Decision comparison for each article category

1. comp.os.ms-windows.mis
2. comp.windows.x
3. misc.forsale
4. rec.autos
5. rec.motorcycles
6. rec.sport.hockey
7. sci.crypt
8. sci.space
9. soc.religion.christian

In comp.graphics category both the classifiers perform on par. The article categories where Decision Trees perform better are as follows:

1. alt.atheism
2. comp.sys.ibm.pc.hardware
3. comp.sys.mac.hardware
4. rec.sport.baseball
5. sci.electronics
6. sci.med
7. talk.politics.guns
8. talk.politics.mideast
9. talk.politics.misc
10. talk.religion.misc

Overall we can say that the Decision Tree Classifier performs better since there are 10 categories where it performs better than KNN.

6 Conclusion and Final Remark

In this project we used the KNN and Decision tree classifier in order to classify the article categories of the 20newsgroup dataset. Overall both the classifiers perform extremely poorly in terms of accuracies. This is mainly because of the sparse dataset and also low quality data being present in the articles. The decision tree classifier performs better on 10 article categories, even after finding the optimal-K for KNN. One solution to improve KNN would be increase the range of k beyond 100. But having said that the concluding remarks for KNN vs DT are as follows:

1. KNN is computationally expensive as it needs frequent look-ups, whereas Decision trees has an in-memory classification model.

2. KNN does not support incremental learning as noticed for different K's, whereas Decision Tree does.

3. KNN makes for better clustering as it takes into account the distance metrics whereas DT is a better classifier as it makes true or false decisions at each tree level making it more classifier efficient.This can also be noticed from the results as Decision Trees were better in 10 article categories when compared to KNN's 9.

References

[1] Wikipedia contributors. Decision tree learning — wikipedia, the free encyclopedia, 2018. [Online; accessed 24-February-2018].

[2] Wikipedia contributors. K-nearest neighbors algorithm — wikipedia, the free encyclopedia, 2018. [Online; accessed 24-February-2018].