A Response to Caruana and Niculescu-Mizil's Comparison of Supervised Learning Algorithms

Lilyanne Kurth-Yontz lkurthyo@ucsd.edu

Abstract

The findings in this report are based from Rich Caruana and Alexandru Niculescu-Mizil's study, *An Empirical Comparison of Supervised Learning Algorithms*. In this report, I compare the classification accuracy scores from Caruana and Niculescu-Mizil's study to my own. To determine these scores, I choose three algorithms - decision trees, random forests, and k nearest neighbors. I also choose three datasets - one on frog calls, one on census data, and one on spam emails.

Introduction

Rich Caruana and Alexandru Niculescu-Mizil's study, *An Empirical Comparison of Supervised Learning Algorithms*, was published in 2006. This paper was a follow-up from King's STATLOG (1995), published more than 10 years earlier. Now that another 10 years have passed, including the debut of the scikit-learn package in June 2007, another study on classification methods is due. In this paper, I compute the classification accuracy scores of decision trees, random forests, and k nearest neighbors, and I compare them with the results of Caruana and Niculescu-Mizil's study.

Caruana and Niculescu-Mizil's paper describes decision trees as one of the worst models, random forests as one of the best, and k nearest neighbors as "mediocre".

Data & Problem Description

The datasets that I chose for this project are Frogs_MFCCs.csv*,* adult.csv, and spambase.csv. The first is a 7195-instance, 22-attribute set about anuran (frog) calls. The second is a 48,842-instance, 14-attribute set of census data. The third is a 4601-instance, 57-attribute set on spam emails.

The problem is determining which algorithm - decision trees, random forests, or k nearest neighbors - is best for predicting one trait from another. I expect decision trees to perform the worst, random forests to perform the best, and k nearest neighbors to perform somewhere in the middle, just like in Caruana and Niculescu-Mizil's study.

Method Description

I use Python, as well as the scikit-learn, numpy, and pandas libraries.

First, I load the .csv files using pandas. Using a label encoder, I convert the categorical values of each dataset into numerical values. Then I convert them into numpy arrays for more efficient processing.

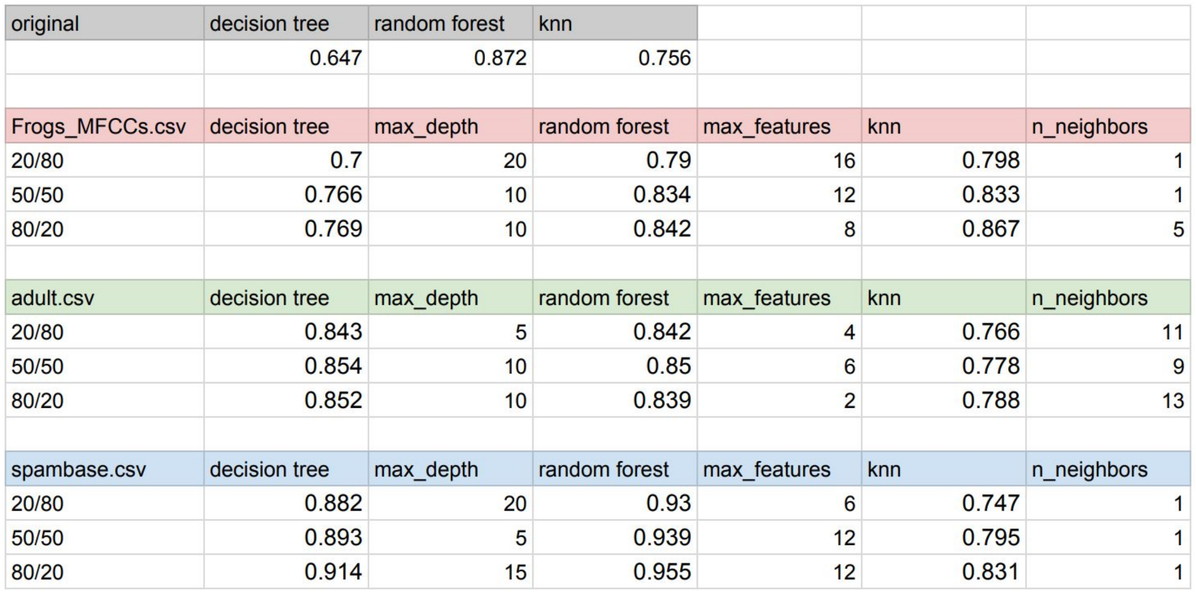
I start with setting up my decision tree classifier. Caruana and Niculescu-Mizil list BAYES, ID3, CART, CART0, C4, MML, SMML, etc as the tree models used in their study. However, scikit-learn only supports CART, so my experiments only incorporate CART. I chose my hyperparameters to be the depth of the tree. More specifically, I chose [1, 5, 10, 15, 20]. In creating a DecisionTreeClassifier() object, I set the criterion to “entropy”.

As for the random forest classifier, Caruana and Niculescu-Mizil tried both the Breiman-Cutler and Weka implementations. The scikit-learn documentation didn’t have explicit details on either, but the name “Breiman” came up repeatedly in the references, so I assume scikit-learn uses the Breiman-Cutler implementation. (This is good, since according to Caruana and Niculescu-Mizil, the Breiman-Cutler implementation yielded better results.) Caruana and Niculescu-Mizil used 1024 trees. Scikit-learn saves the number of trees in a parameter called “n_estimators”, which by default is 10. 1024 trees was too time-consuming for my computer, so I lowered this number to the default value. In the original study, [1, 2, 4, 6, 8, 12, 16, 20] was the size of the feature set considered at each split (scikit-learn calls this “max_features”). I used these values as my hyperparameters, except in the case of adult.csv. Since this dataset only had 14 parameters, I removed 16 and 20 from the parameter set. Just like in my decision tree, I set the RandomForestClassifier() object’s criterion to “entropy”.

For k nearest neighbors, Caruana and Niculescu-Mizil choose 26 evenly-spaced values of k, ranging from 1 to the size of the training set. However, 26 values of k was too time-consuming for my computer, so I used 13 values - [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25]. Even though Caruana and Niculescu-Mizil mention the kernel width, the scikit-learn documentation didn’t have any information on this parameter. So, I chose my 13 values of k (labeled in scikit-learn as “n_neighbors”) as my hyperparameters.

As for the process itself, each dataset is shuffled and split 20/80, 50/50, and 80/20. Each of these splits undergoes 5-fold cross-validation and grid search to find the optimal test accuracy. The code is structured by dataset, by algorithm, and by split, in that order.

Experiments



original	decision tree	random forest	knn			
	0.647	0.872	0.756			
Frogs_MFCCs.csv	decision tree	max_depth	random forest	max_features	knn	n_neighbors
20/80	0.7	20	0.79	16	0.798	1
50/50	0.766	10	0.834	12	0.833	1
80/20	0.769	10	0.842	8	0.867	5
adult.csv	decision tree	max_depth	random forest	max_features	knn	n_neighbors
20/80	0.843	5	0.842	4	0.766	11
50/50	0.854	10	0.85	6	0.778	9
80/20	0.852	10	0.839	2	0.788	13
spambase.csv	decision tree	max_depth	random forest	max_features	knn	n_neighbors
20/80	0.882	20	0.93	6	0.747	1
50/50	0.893	5	0.939	12	0.795	1
80/20	0.914	15	0.955	12	0.831	1

Overall, my random forest and k nearest neighbors accuracies match up with Caruana and Niculescu-Mizil's results, but there is a noticeable increase in decision tree test accuracy. Sometimes, random forests actually performed worse than trees and neighbors. Ultimately, my k nearest neighbors accuracy scores were the most faithful to the original study's result.

With Frogs_MFCCs.csv, decision trees performed the worst, and random forests and k nearest neighbors performed roughly the same. With adult.csv, decision trees and random forests performed equally, and k nearest neighbors performed the worst. With spambase.csv, random forests performed the best, decision trees performed okay, and k nearest neighbors performed the worst.

As mentioned previously, Caruana and Niculescu-Mizil used many different tree models in their study, while scikit-learn only uses one. Perhaps these researchers averaged their tree results over both well-performing and poor-performing models, and scikit-learn chose the best performing model. This theory could explain why my test accuracies are larger.

When looking at my results, I noticed that random forests performed exceptionally high under spambase.csv, which could imply that more attributes lead to higher test accuracies - at least, under random forest classifiers. This theory would explain why random forests in adult.csv, my dataset with the fewest attributes, performed on par with decision trees (the supposed "worst" of the three). It appears that the performance of these classifiers somewhat relies on the qualities of the dataset itself.

Conclusion

Caruana and Niculescu-Mizil's results are still relevant today, since their study can be more or less replicated with the use of scikit-learn. That being said, there is a noticeable increase in performance with scikit-learn's decision tree classifier. My findings also imply that there is no “one size fits all” rule for classifiers, and that a “good” or “bad” classifier is not necessarily “good” or “bad” in all cases.

References

Colonna, Juan Gabriel et al. (2017). Anuran Calls (MFCCs) Data Set [http://archive.ics.uci.edu/ml/datasets/Anuran+Calls+%28MFCCs%29]. Manaus, Brazil: Universidade Federal do Amazonas.

Hopkins, Mark et al. (1999). Spambase Data Set [http://archive.ics.uci.edu/ml/datasets/Spambase]. Palo Alto, CA: Hewlett-Packard Labs.

Kohavi, Ronny and Becker, Barry. (1996). Census Income Data Set [http://archive.ics.uci.edu/ml/datasets/Census+Income]. Silicon Graphics.

https://scikit-learn.org/stable/