Decision Tree and Naive Bayes

Joe Shymanski

February 25, 2022

1 Introduction

Decision Tree (DT) learners and the Naive Bayes (NB) algorithm are both incredibly powerful tools for classification problems of mapping an input X to an output Y. Of course, DTs are deterministic in their mapping and NB is probabilistic. This paper will report on the performance of the two algorithms using two different datasets.

2 Datasets

Two datasets were used in this investigation of DT and NB performance: Letters [2] and Zoo [1]. The Letters dataset is used for recognizing capital letters of several different fonts based on 16 major features. Each feature is given an intensity value from zero to 15. For the sake of this problem, the intensities from zero to seven were truncated to zero, while the values from eight to 15 were assigned a one. The Zoo dataset contains 16 features, most of which are Boolean, for categorizing different species of animals into seven overarching classes: mammals, birds, insects, fish, reptiles, amphibians, and invertebrates.

3 Models

Both models were implemented in Python using Pandas DataFrames as the primary data constructs. These objects come with a bit of overhead, so the algorithms may run much more slowly than an optimal implementation.

The DT learner utilizes a parameter for maximum tree depth to prevent overfitting. The NB algorithm takes advantage of Laplace smoothing to avoid probabilities of zero and to incorporate priors into the calculations.

4 Experimental Results

Dataset	Algorithm	Train Accuracy (%)	Test Accuracy (%)	Time (s)
Letters	DT	49.19	43.20	32.60
Letters	NB	35.58	30.85	7.80
Zoo	DT	100.00	94.05	0.20
Zoo	NB	95.42	93.83	0.05

Table 1: Algorithm performance results.

Table 1 reports the results of each algorithm on each dataset. 5-fold cross-validation was utilized. For the DTs, a maximum depth of five was used across all runs. For NB, the Laplace smoothing parameter k was set to one. All 101 examples were used with the Zoo dataset, but only 2000 of the Letters examples were seen. In order to reproduce the results, the seed for the NumPy random number generator was set to 100. All accuracy and times represent the average of all five runs from cross-validation.

5 Discussion

As is clear in Table 1, the DT learner is about four times slower than the NB algorithm. I know for a fact that the learner can become much more efficient if I were to use Python dictionaries instead of Pandas DataFrames for data lookup. However, the former may not scale as well as the latter and is certainly less straightforward to implement.

The results also show that the DT learner ended up outperforming the NB algorithm on each dataset. This surprised to me, as I would have expected a DT to perform rather poorly on an image recognition problem, unlike NB. However, quite the opposite held true. Perhaps a better implementation of the NB algorithm - one where a range of intensities can be considered - would have performed better.

Finally, it appears that both classifiers performed much better on the Zoo data than the Letters data. However, the even the worst performance on Letters was about eight times better than that of a random guesser, whereas the best Zoo performance was only seven times better. This is due to the fact that the Zoo dataset only had seven classes to choose from while the Letters dataset had 26.

6 Conclusion

The DT and NB predictors are quite effective at mapping (either directly or indirectly) an example X to label Y. The DT learner was a bit more effective for these problems but also cost a bit more to train and classify. NB still performed well, especially given the small number of samples seen in both datasets.

References

[1] Richard Forsyth. UCI machine learning repository, 1990.

[2] David J. Slate. UCI machine learning repository, 1991.