Comparison of 3 Machine Learning Classifiers

Andrew Pesek

Abstract

There exists many classifiers used in performing a classification task, each with their own specialized uses and overall tradeoffs between accuracy and generalizability. In specific scenarios, some classifiers are naturally going to work better than others depending on the nature of the training data.

1. Introduction

There are several different methods of classification, each with varying degrees of accuracy in training and testing. Here I have experimented with 3 different classifiers to see which one ranks the best on the datasets chosen to use for this experiment.

2. Methods

In this experiment I've decided to use K-nearest neighbors with cross validation, decision tree with cross validation, and logistic regression for the classifiers. I'm using the k nearest neighbors from 1 to 10 in the knn classifier, a max depth from 1 to 10 in the decision tree classifier, and 10-fold cross validation for both of them. Logistic regression I am using on its own with 30,000 max iterations and no cross validation. Each classifier is tested on 20/80, 50/50, and 80/20 split of the data to use as training and then testing. Each partition of the data is then tested 3 times with the average accuracy recorded.

3. Experiments

After setting up each of the classifiers and cross validations, as well as some frequently used functions, I have imported each of the datasets I chose to use and organized them so they could be used in the main training algorithms.

3.1. Dataset Pre-processing

For each of the datasets I chose to use from the UCI database I have set them up as data frames so they could easily be displayed and manipulated for the purposes of the experimental setup. List of UCI repository datasets used:

- Wine https://archive.ics.uci.edu/ml/datasets/Wine
- Student Performance https://archive.ics.uci.edu/ml/datasets/Student+Performance
- Adult https://archive.ics.uci.edu/ml/datasets/Adult

Wines Dataset (A): For the first dataset I chose a multi-class dataset with chemical measurements of 3 different classes of wine. It's a smaller dataset with only continuous numerical values, so it should be easy to work with. The classes in this dataset are also very well defined.

Student Performance Dataset (B): For the second dataset I have a dataset of student demographics and performance scores from two different schools. There was no specified class for this dataset, so I chose which of the 2 schools as the class since it was the only attribute with a binary variable, and the classifiers should attempt to predict which of the schools the student attended based on the other variables. Here I have dropped the categorical data from the dataset I will end up using and keeping just the numerical values to make things easier.

Census Dataset (C): Lastly, for this dataset I've chosen a much larger dataset of US census information with the classification task of predicting whether the individual makes less than (0) or greater than (1) 50K a year, labeled under the 'income_>50K' column. Since most of the attributes are categorical I have one-hot encoded all of these columns. I then decided to drop all the columns with unknown categories (such as 'occupation_?'), since I believed they wouldn't add any useful information; and columns for native countries other than the US just to reduce the number of features and training time. Displayed in the table below are just the numerical feature columns, and the income class column ('income_>50K') the classifiers will be trained to predict.

3.2. Main Algorithm

After setting up the dataframes in preparation to display the accuracy results in, I have set up the main algorithm to train each of the classifiers. First, I convert the dataframes to numpy arrays then start by looping through each dataset. For each dataset (A, B, and C) it loops through each partition (20/80, 50/50, and 80/20), then for each partition through each trail (3x). In the code portion I have printed out messages to show the progress, and some of the heatmaps displaying cross validation for the knn classifier, shown in the following figures.. I am also using a similar method of cross-validation on the decision tree classifier. At the end each set of errors is averaged over each trial to be displayed in the dataframes.

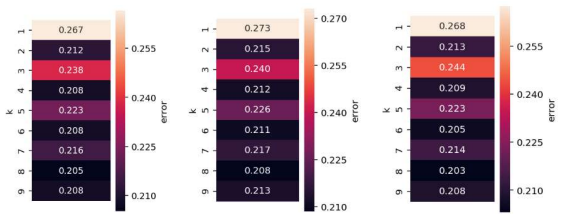

Figure 1. Heatmaps of cross validation error for each trial of training dataset C on the 80/20 partition

3.3. Results

	A Training	A Testing	B Training	B Testing	C Training	C Testing	Avg Testing
Knn	0.13 +/- 0.11	0.33 +/- 0.04	0.05 +/- 0.04	0.13 +/- 0.01	0.19 +/- 0.01	0.21 +/- 0.00	0.22
Decision Tree	0.00 +/- 0.00	0.17 +/- 0.04	0.05 +/- 0.05	0.13 +/- 0.01	0.13 +/- 0.01	0.15 +/- 0.00	0.15
Logistic Regression	0.03 +/- 0.02	0.10 +/- 0.03	0.03 +/- 0.02	0.17 +/- 0.03	0.21 +/- 0.00	0.20 +/- 0.00	0.15

Table 1. Results on 20/80 Partition

	A Training	A Testing	B Training	B Testing	C Training	C Testing	Avg Testing
Knn	0.13 +/- 0.10	0.31 +/- 0.02	0.10 +/- 0.01	0.13 +/- 0.01	0.19 +/- 0.00	0.21 +/- 0.00	0.22
Decision Tree	0.00 +/- 0.00	0.06 +/- 0.01	0.10 +/- 0.01	0.13 +/- 0.01	0.13 +/- 0.00	0.15 +/- 0.00	0.11
Logistic Regression	0.02 +/- 0.01	0.05 +/- 0.02	0.09 +/- 0.01	0.14 +/- 0.01	0.20 +/- 0.00	0.20 +/- 0.00	0.11

Table 2. Results on 50/50 Partition

	A Training	A Testing	B Training	B Testing	C Training	C Testing	Avg Testing
Knn	0.07 +/- 0.10	0.40 +/- 0.07	0.09 +/- 0.00	0.11 +/- 0.02	0.19 +/- 0.00	0.21 +/- 0.00	0.24
Decision Tree	0.02 +/- 0.01	0.15 +/- 0.08	0.09 +/- 0.03	0.11 +/- 0.01	0.14 +/- 0.00	0.15 +/- 0.00	0.13
Logistic Regression	0.01 +/- 0.00	0.08 +/- 0.06	0.11 +/- 0.01	0.13 +/- 0.03	0.20 +/- 0.00	0.20 +/- 0.01	0.13

Table 3. Results on 80/20 Partition

	20/80 Split	50/50 Split	80/20 Split
Knn	0.22	0.22	0.24
Decision Tree	0.15	0.11	0.13
Logistic Reg.	0.15	0.11	0.13

Table 4. Average Testing Error

4. Conclusions

From what I can tell, it seems that for each partition, decision tree and logistic regression classifiers worked about equally well, with knn doing considerably worse on both training and testing. Interestingly, the testing error was the best on the 50/50 split, but not by that much compared to the 80/20 split. Overall I'd say Decision Tree worked the best since its errors across the board were a little bit less variable, but in some cases were more variable than logistic regression. This is likely the case since I probably allowed a rather high max depth for decision tree and a high number of iterations for logistic regression, the resulting testing errors are probably the best they are going to get for this set up in particular. I suspect Knn was probably not the best classifier to use for the specific kinds of datasets I chose to use for this experiment. While I could increase the scope of the grid search, I don't believe it will change the results by very much.

Overall, I would conclude that decision tree and logistic regression are both very good classifiers for this particular setting, insofar as these specific datasets at least.

References

Caruana, Rich, and Alexandru Niculescu-Mizil. "An Empirical Comparison of Supervised Learning Algorithms." *proceedings of the 23rd International Conference on Machine Learning - ICML '06*, 2006, doi:10.1145/1143844.1143865. https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf

Forina, M. et al, PARVUS - An Extendible Package for Data Exploration, Classification and Correlation. Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy.

P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., *Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008)* pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7. http://www3.dsi.uminho.pt/pcortez/student.pdf

Ron Kohavi, "Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid", *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996. http://robotics.stanford.edu/~ronnyk/nbtree.pdf