CS7641 – Machine Learning

Assignment 1 – Supervised Learning

Mason J. Kelchner  
mason.kelchner@gatech.edu

# 

# INTRODUCTION

A variety of supervised machine learning algorithms were applied to two classification data set problems to investigate how they perform under various circumstances, how tuning the algorithms for training affects their performs, and benefits and drawbacks of each as they are applied to a particular dataset. The techniques investigated were Decision Trees with pruning, Boosted Decisition Trees, Neural Networks implemented using a Multi-Layer Perceptron structure, Support Vector Machine, and K-Nearest Neighbors. All algorithms were implemented using the Python open source module sklearn [1].

The two chosen classification problem datasets were for a prediction of credit default and for mushroom classification. These are problems faced in industry today as credit card companies or any company that provides loans faces the risk of their lendees default on their credit, causing a financial loss to the lender. It would be important to implement algorithms to predict whether some parameters describing a lendee or credit applicant would be a higher risk associated with default. This would inform either approving or declining credit or applying a particular interest rate to a loan. Secondly, the mushroom classification problem is very important to foragers and suppliers of mushrooms for a variety of their purposes. In particular, some mushrooms are edible and can be very valuable while other mushrooms which can look very similar to their popular and edible relatives are in fact not edible and in some cases very poisonous to humans. It would advantageous for those interested in obtaining mushrooms to implement supervised machine learning algorithms which can identify mushroom based on various parameters associated with each. Both of these problems are well suited for supervised machine learning classification. The benefits and drawbacks of each algorithm from above as they are applied to both dataset will be discussed in this analysis.

# Experiments

Each classification problem described previously for credit default prediction and mushroom identification were used for evaluating the performance of each supervised learning algorithm. Each algorithm was trained on a subset of the entire data set, the training set, and tested without learning on a test size. For all models, a test size of 20% of the data set was set aside while the models learned on 80% of the data set. Some cross validation was included and will be described in those algorithms’ sections. A random seed was set for all experiments, testing splits, and models for reproducibility. However, the wall clock training time for the algorithms will always very between runs, computer hardware, and architecture. Though, for each seed or run, the trend of training time should qualitatively remain the same. Each algorithm had hyperparameters which were varied to understand their effects on the models performance. Model performance for each data set was evaluated by plotting learning curves, validation curves, and training times for each model configuration. Additionally, the sklearn class GridSearchCV was used to determine the best performing algorithm defined in the state space defined by the hyperparameters of each tested algorithm where best was defined as the accuracy score of the algorithm on the test data set [1].

## Decision Tree with Pruning

The first experiment for both data sets involved training a classification decision tree with pruning (DT) by using the sklearn class DecisionTreeClassifier [1]. For both data sets, two splitting methods were used for the decision trees: random and best feature selection where best was using the maximum information gain for a split. Additionally, the number of minimum samples at a leaf, the maximum depth of the tree, and the cost complexity pruning parameter were varied using the input parameters min\_samples\_leaf, max\_depth, and ccp\_alpha, respectively, for both decision tree splitting types for both data sets. The performance of all these decision trees on the training and test data for both data sets as well as the training time is shown in Figure 1.

Figure 1 Decision Tree with pruning using random and best feature splitting for variable hyperparameters

## Boosted Decision Tree

The second experiment for both data sets involved training a classification decision tree this time incorporating boosting (BDT) using a combination of weak learners using decision trees. The hyperparameters varied for this model were the number of weak learners, the learning rate, the max depth of the trees, and the minimum number of samples at a leaf. For this experiment, the sklearn class GradientBoostingClassifier was used [1]. These hyperparameters were varied over a range and each of the models’ performance for the training and test data sets as well as the training time is shown in Figure 2.

Figure 2 Decision Tree incorporating boosting for variable hyperparameters

## K-Nearest Neighbors

The third experiment for both data sets used a K-Nearest Neighbors (KNN) algorithm for classification for both data sets. Training a KNN algorithm is simple as the training is constant while the query or prediction time is dependent on the data set or sample space. Therefore, for this experiment training time was not a performance metric. The sklearn class KNeighborsClassifier was used for this experiment [1]. For both data sets the number of nearest neighbors (k) was varied and the accuracy of each algorithms’ predictions are shown in Figure 3.

Figure 3 K-Nearest Neighbors algorithm with variable k

## Support Vector Machines

The fourth experiment for both data sets used a Support Vector Machine algorithm using the sklearn class LinearSVC [1]. The hyperparameters tuned for this experiment included XXXXXXXXXXXXXXX. For each model and data set their performance and training time is shown in Figure 4.

Figure 4 Support Vector Machine performance

## Neural Network – Multi-Layer Perceptron

The fifth and final experiment for both data sets applied a neural network algorithm using the sklearn class MLPClassifier which uses a multi-layer perceptron [1].

# Summary

# References

|  |  |
| --- | --- |
| [1] | F. Pedregosa, Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011. |