Monira Khan

CS4641: Machine Learning

February 2nd, 2019

**Supervised Learning**

The purpose of this report is to analyze and describe five different supervised learning techniques: decision tree learning, neural networks, boosting, support vector machines, and k-nearest neighbors. I used the python library scikit-learn in order to easily implement all of these classifiers. The two datasets that all classifiers were run on are listed below.

* **Breast cancer dataset:**

This dataset consists 699 instances of data obtained from breast cancer databases from the University of Wisconsin hospitals. The instances are in chronological order due to the periodical reporting of clinical cases. For the purposes of this assignment, the data is shuffled before split into training and testing subsets. The first attribute of the instance is a unique id number which I kept out of the training and testing subsets because it does not benefit the classifier–in fact this attribute hinders the classifier’s performance. Each attribute afterward correlates to an aspect of a breast tissue cell. The aspects are: clump thickness, uniformity of the cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. The classification of the instance is either benign or malignant. Benign means that the cancer lacks the ability to infect the neighboring tissues–meaning the cancer is not dangerous. On the other hand, malignant means that the cancer is invasive and dangerous.

I think this dataset is interesting and extremely important because finding patterns and learning/analyzing this data could save lives by predicting cancers in patients before they get too dangerous. I believe that machine learning and computer science as a whole should be used for the greater good of helping the world, and studying cancer data is a very good start. I also found the relationship between the aspects of the cell and severity of cancer itself interesting.

* **Chess game dataset:**

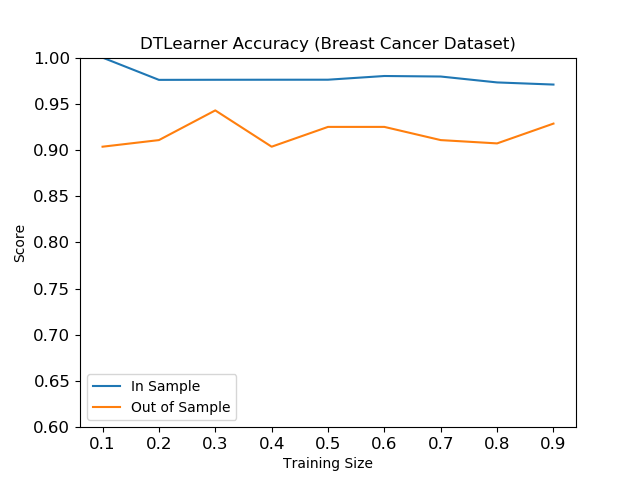
This dataset follows a chess game of king and rook team against a king and pawn team. The databased was generated and described by Alen Sharprio and supplied by the Turing Institute in Glasgow. The dataset is multivariate and categorical, but in order for the skit-learn library to classify the data, I had to convert the dataset into an integer dataset by setting the letter to a corresponding number. This dataset is composed of three thousand ninety-six instances with 36 attributes. The classification of the data is either white can win or white cannot win. The white team is the king and rook and the black team is the king and pawn team. The white is deemed to lose if the black pawn can safely advance. Each instance is a board-descriptions for this chess endgame. The 36 attributes correspond to the board. There is one board position per line.

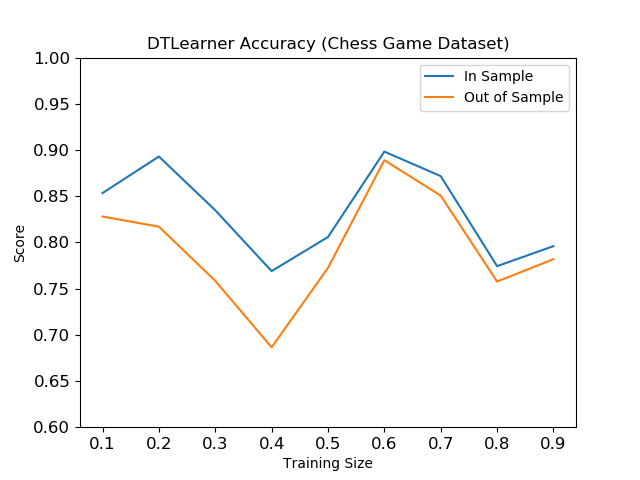
I personally found this database interesting because I have always been interested in the learning that took place when playing against a CPU on chess. The CPU always seem to win. Because I used to always play against the chess CPU, when I saw this dataset, I thought it would be very interesting to run classifiers on this type of data, because it gets me closer to creating my own machine learning chess bot.

All five algorithms’ hyperparameters were tuned with GridSearchCV in order to achieve optimal performance. With Grid Search, you feed the method a list of hyperparameters which it creates a grid and compares all of the algorithms performances according to the all the parameter combinations exhaustively. This results in an estimator especially tuned for the certain dataset.

**Decision Tree Classifier:** Decision trees are a non-parametric classifier that attempts to classify data points by rules it infers from its attributes. The rules work like a game of 21-questions where asking one question creates a rule–canceling out many options that the X can be classified as. The algorithm is logarithmic in the number of data points used to train the tree by this method of splitting up the data at every node. A disadvantage of decision trees, though, is overfitting to the training data by creating overtly complex trees, so the classifier cannot generalize to the testing data.

There were two ways that I have pruned the decision tree–using GridSearchCV and setting the max depth of the tree. Using the GridSearchCV is a form of pre-pruning the classifier. The model selection object takes parameters–max depth, max features, and min samples leaf–to compare and search for the optimized combination of the parameters for the training set. By choosing the best calibration of the parameters, the object is essentially pre-pruning the decision tree to be best fit for the data. This method resulted in the following learning curves:





Interestingly, the pre-pruning and hyperparameter tuning impacted the learner’s accuracy differently depending on the datasets. For the breast cancer dataset, the in sample accuracy was 100% with only 10% of the training data