CS 7641 Assignment 1: Supervised Learning

Nimesh Chudasama
nimesh@gatech.edu

Abstract—This paper investigates the relationship between 5 supervised learning algorithms. In particular it highlights the differences between Decision Trees, Neural Networks, Boosting, Support Vector Machines, and k-NN on two datasets. This paper will highlight the key differences between each algorithms and how they behave on different datasets to gain a better understanding of each algorithm. Through analysis of their performance on different datasets this paper highlights how these algorithms perform on different datasets. All analysis was done through Python using sklearn to implement the algorithms and matplotlib to visualize their respective performance.

1 DATASETS INTRODUCTION

Two data sets were selected that I determined were interesting in a machine learning context. The two datasets differed heavily in terms of structure. This means that the structure was different in terms of number of instances, number of attributes, and a different type of classification problem.

Both datasets are from UCI's machine learning dataset repository. The models on each dataset were selected based on the best accuracy score. In each case, the test data was independent of the training data.

1.1 Abalone Dataset (Nash, 1994)

The Abalone dataset had 4177 instances, 8 attributes, and a Gaussian distribution with respect to age. Each instance had an associated age based on each 8 attribute. The age was converted into a multi-class classification with 6 labels of ages 0-5, 5-10, 10-15, 15-20, 20-25, and 25+.

As seen in the figure, most of the data was between the ages of 5-9 and 10-14 which means that this is an unbalanced dataset.

The reason for converting this regression problem into a classification problem was to do a direct comparison with the divorce dataset on how different dataset structures affect performance.

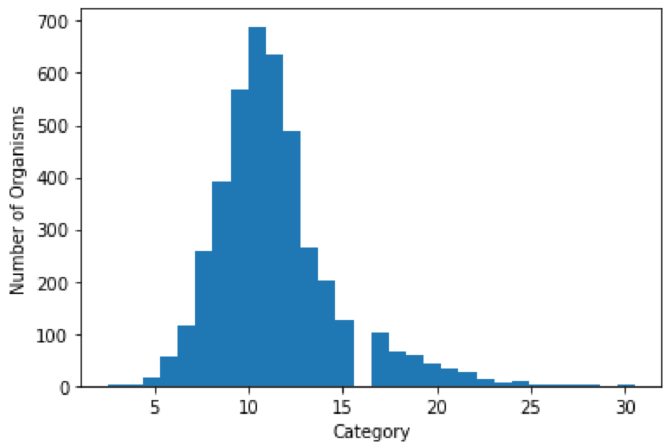

Figure 1—Abalone Dataset distribution [1]

1.2 Divorce Dataset (Yöntem, 2019)

The divorce dataset had 170 instances, and 54 attributes. Participants were asked to rate their significant other on a scale of 1-5 on common scenarios they might encounter in a relationship. Using this information they were put into a binary class of divorced or not.

This dataset was balanced 50/50 into divorced/not divorced.

2 DECISION TREE

Decision Trees split the data on an attribute and try to isolate each classification based on the attribute split.

In general, having a tree with a large depth would result in overfitting the model. This is because at every depth it splits at a specific attribute (sometimes the same attribute). This means that with continual splitting on training data attributes it overfits and is unable to generalize to the test data. The purpose of pruning is to find a depth that is optimal to new instances of data without overfitting the training data.

2.1 Abalone Dataset

As stated before, as the depth increases the training accuracy converges to 100%. This does not translate to the test data. For the test data, a decision tree with depth 4 maximizes the accuracy. After pruning the decision tree to a depth of 4, the model is general enough to obtain an accuracy of 72.24% on test data.

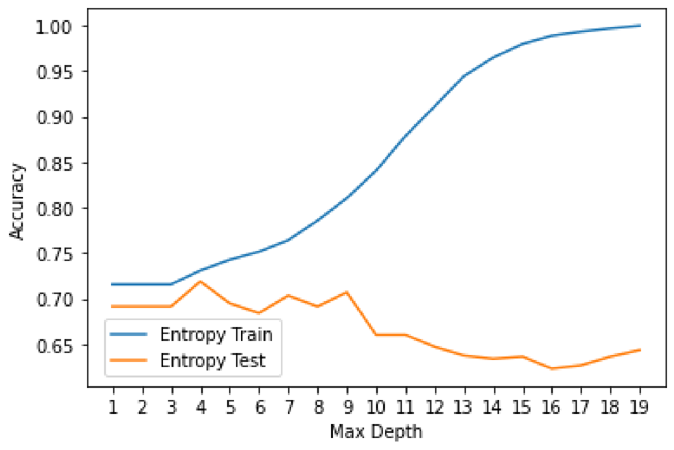

Figure 2—Abalone Decision Tree

2.2 Divorce Dataset

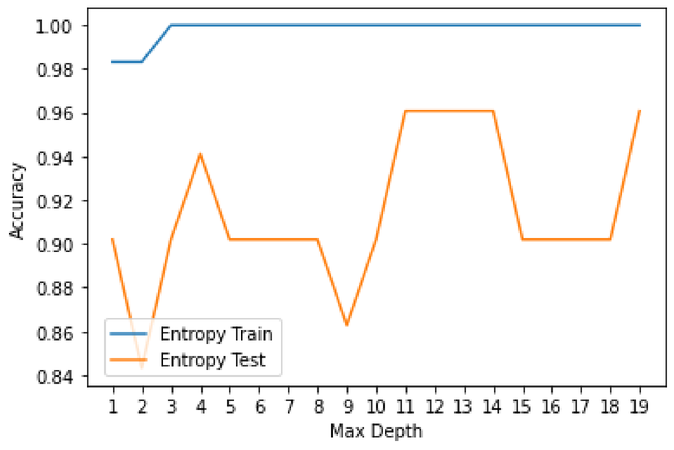

Figure 3—Divorce Decision Tree

After 3 splits on Divorce attributes, the decision tree can classify 100% of the training data correctly. The testing data seems best at a depth of 11. The reason that 11 is better than 12/13/14 is because it's a more general model that performs just as well as the more specific model. Using a pruned decision tree with a depth of 11 yields an accuracy of 96%.

2.3 Summary

These models both have different accuracy and there are a few explanations on why they perform differently.

The Abalone dataset is a multi classification problem. Guessing randomly on the Abalone, there's a 1/6 chance of obtaining a correct answer. In addition to this, there are less attributes to split the data on.

The Divorce dataset is a binary classification problem. Due to the binary nature of the problem, there's a 1/2 chance of guessing a correct answer. There are 54 attributes to split on and it seems that splitting on certain attributes increased accuracy substantially.

3 NEURAL NETWORK

Neural Networks use multiple perceptrons layers to compute a weight to fire each specific node. The weights are then set through backpropagation. In general, the more hidden layers that are present, the more weights that the algorithm has to train. This means for additional layers that you may add, training time greatly increases. Additional layers also risk overfitting the data. In this experiment different hidden layers were selected to determine how they affected the accuracy of the algorithm.

In addition to using different hidden layer sizes, GridSearchCV was used to find optimal hyperparameters by tuning the hidden layer sizes, activation function, and learning rate.

3.1 Abalone Dataset

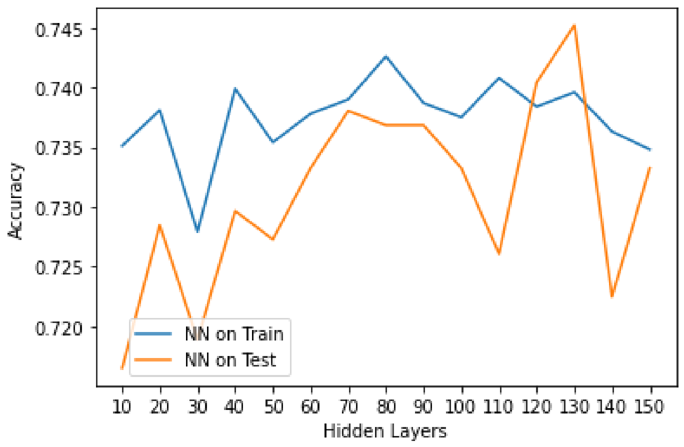

Figure 4—Abalone Neural Network

As the number of hidden layers increase the general trend of accuracy is inconclusive. Adding and removing layers did not seem to change the accuracy by much. It seemed like 130 neurons in the hidden layers resulted in an accuracy of 74.52%.

3.2 Divorce Dataset

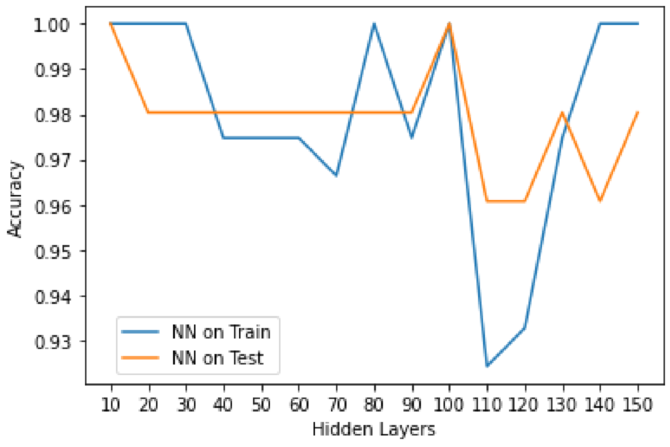

Figure 5—Divorce Neural Network

It seems like for a binary classification task, there wasn't much tuning required and the model quickly converged to an optimal solution with just a 10 perception hidden layer size with an accuracy of 100.0%.

3.3 Summary

It is not evident that adding more hidden layers significantly improve performance for classification tasks. Adding more layers will risk overfitting the data, increasing training times without improvement to the test accuracy.

Hyperparameter tuning through the use of GridSearchCV did not yield a different accuracy. For the Abalone dataset, using 150 hidden layers and a learning rate of 0.01 had an accuracy of 73.33%. On the divorce dataset the optimal hyperparameters were 10 hidden layers and a learning rate of 0.0001 with a result of 100.0% accuracy.

4 BOOSTING

Gradient Boosting is the algorithm incorporated. Gradient boosting uses the difference between the model's prediction and the actual label to adjust the model (Gorman, 2017). It's an ensemble learning algorithm of decision trees that is trained using gradient descent. The number of boosting stage is the number of trees used to predict the output. In general the more trees that you use, the longer and computationally expensive boosting is (Maklin, 2019).

Although it uses regression to predict the value, it is still useful because it can still put the value in a label. This means it can be used for classification problems. In the instance of the Abalone data, the labels were derived from discrete Age values which means this is an appropriate algorithm. For the Divorce data, the labels were binary already.

In addition to using the number of trees to be used in the ensemble, GridSearchCV was used to find the optimal depth the tree should use.

4.1 Abalone Dataset

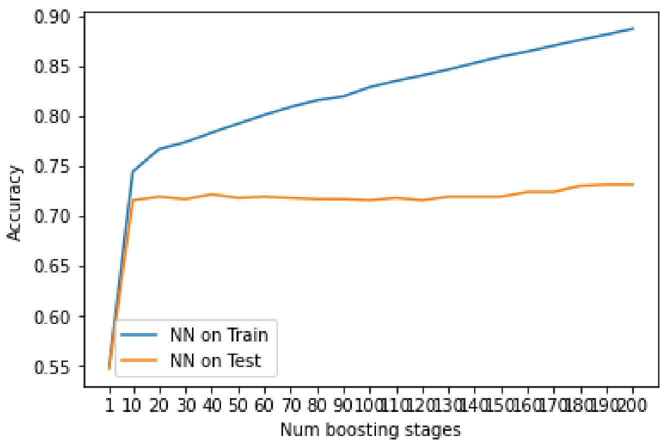

Figure 6—Abalone Boosting

After using 10 trees it seems like the Abalone dataset accuracy leveled off, meaning addition of trees to make a prediction did not yield better performance of the model. The maximal accuracy of the graph was at 73.08%.

4.2 Divorce Dataset

Since gradient descent is used and the label can only take on two different values. Using gradient descent it has to make a prediction of to the value of the output. In this case, it was able to achieve an accuracy of 96%. After 50 decisions trees were used in ensemble, the accuracy leveled off.

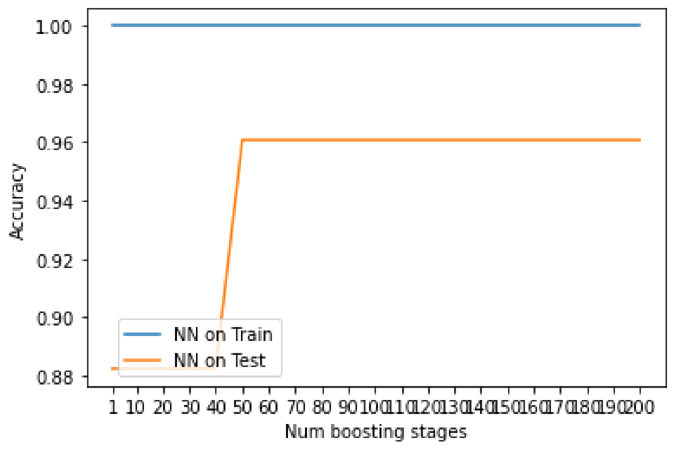

Figure 7—Divorce Boosting

4.3 Summary

Both algorithms had an "optimal" number of decision trees to use, and up until that number was reached performance was poor. Adding additional decision trees would result in overfitting the data and increasing training times.

Hyperparameter tuning using GridSearchCV was used to find the optimal depth of the decision tree and the number of estimators. For the Abalone dataset the optimal depth was 3 and the optimal number of estimators was 10 to yield an accuracy of 71.53%. For the Divorce dataset, the optimal depth was 3 and the optimal number of estimators was 60 to obtain an accuracy of 96.08%. Both of these hyperparameters were consistent with not overfitting the data (using too many decision trees) and using the optimal depth for the data.

5 SUPPORT VECTOR MACHINE

Support Vector Machines tries to find the optimal hyperplane between the data. In particular it finds the hyperplane with the greatest distance to between the points in an effort to generalize more than just choosing a random hyperplane that seperates the dataset.

In order to find different hyperplane, different kernels were used. We used polynomials of degrees 2 through 8, RBF, sigmoid and linear for the kernel to compare the results.

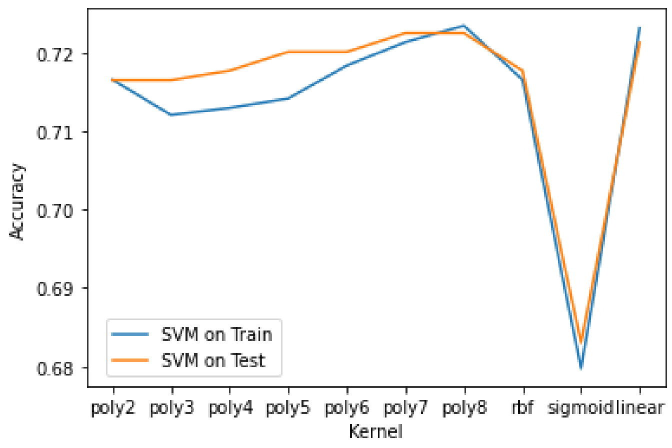

Figure 8—Abalone SVM

5.1 Abalone Dataset

The most optimal hyperplane was that with a polynomial 7 with an accuracy of 72.34%. SVM does not work well on multi classification problems and is better suited for binary classification problems. There are variations of SVM that can be used to to generalize for multiple classes, and as a result our accuracy tended to be in the low 70s.

5.2 Divorce Dataset

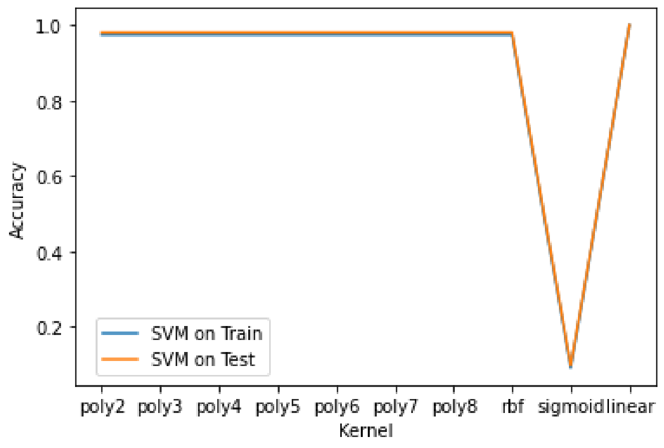

Figure 9—Divorce SVM

This dataset did seem to be linearly separable as every polynomial function and even the linear function was able to correctly classify the data with an accuracy of 100.0%.

5.3 Summary

Both datasets performed poorly on the sigmoid function. "The function maps any real value into another value between 0 and 1 (ml-cheatsheet, 2020)." Since Abalone was a multiclassification problem it's harder to find an "optimal hyperplane", whereas Divorce seemed to be a linearly seperable dataset so finding an optimal algorithm was not difficult.

6 K-NN

k-NN is a lazy learner. It predicts classes of values by looking at n training data points closest to our test value. In terms of how close a value is, we define a function. For our purpose we used uniform distance to determine how close a value is. We then iterated using different number of neighbors to test our algorithms (Aggarwal, 2020). In general the more neighbors present, the more general an algorithm is.

In addition to comparing the number of neighbors we used GridSearchCV to find optimal hyperparameters of k-NN.

6.1 Abalone Dataset

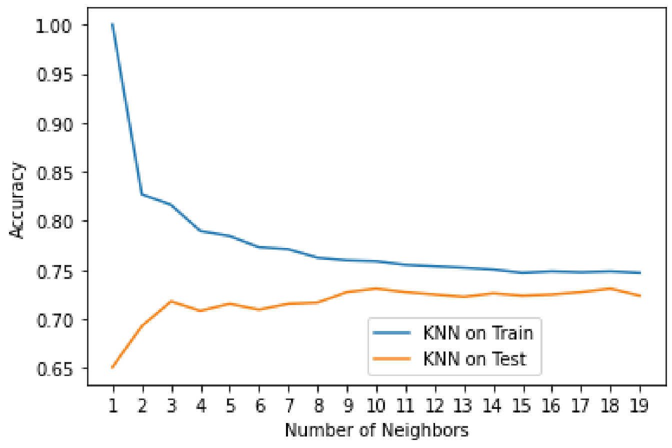

Figure 10—Abalone k-NN

It seemed that adding more neighbors to this dataset increased accuracy without affecting performance too much. The optimal number of neighbors according to the graph is 18 which maximizes the accuracy to 73.08%.

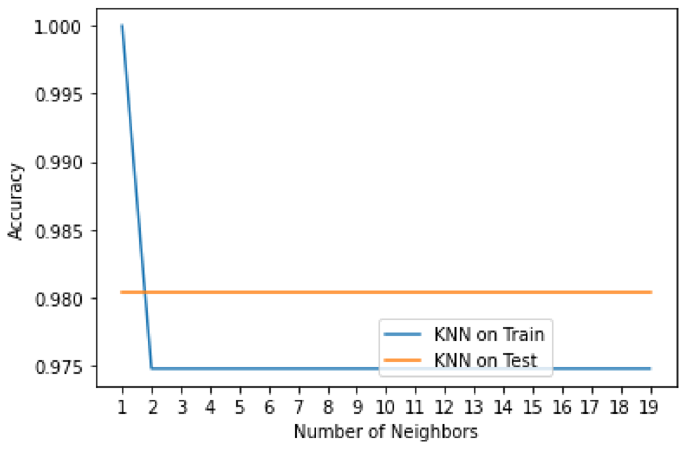

Figure 11—Divorce k-NN

6.2 Divorce Dataset

In the divorce dataset the test set performed better than the training set after 2 neighbors were included for an accuracy of 98.04%.

6.3 Summary

In both cases, in the training accuracy nearest neighbor of 1 yielded 100%. This is because the closest point of a training data is the training data we used to evaluate it. Therefore the accuracy of k-NN with neighbor 1 on train data will always be 1.

GridSearchCV was used on both datasets to find the optimal hyperparameters of the model. To find the optimal parameters we used different k neighbors, different weight measurement and a different algorithm to compute the nearest neighbors. For the Abalone dataset we found that the best number of neighbors was 10, using the distance metric was the optimal function to achieve and accuracy of 72.97%. For the Divorce dataset using 1 neighbor and uniform weights was the most optimal function to achieve an accuracy of 98.04%.

7 DIRECT COMPARISONS OF SUPERVISED LEARNING ALGORITHMS

Both the datasets had a similar distribution, but it seems like supervised learning performed better on the Divorce dataset, this might be because it was a binary classification task vs a multi classification task.

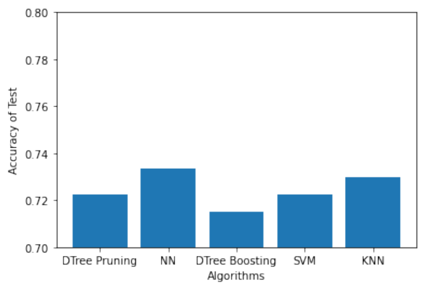

Figure 12—Abalone

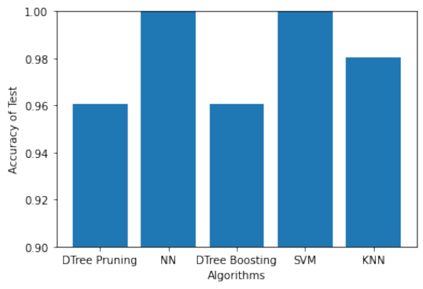

Figure 13—Divorce

7.1 Training Time

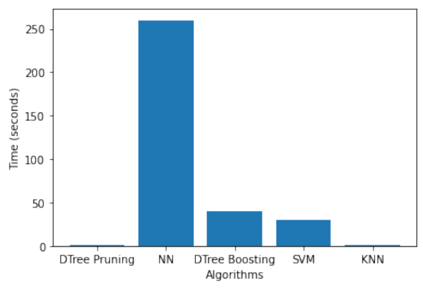

Figure 14—Abalone

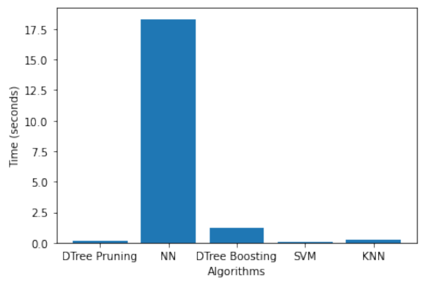

Figure 15—Divorce

Comparing all the algorithms it seems like decision trees, SVM and kNN took the least time to model the training data.

Neural networks took the most time because it had to use backpropagation to find optimal weights and Boosting took longer than others because it had to generate a lot of decision trees to get enough weak learners.

8 REFERENCES

[1] Aggarwal, Sangeet (2020). "K-Nearest Neighbors". In: *https://towardsdatascience.com/k-nearest-neighbors-94395f445221*.

[2] ml-cheatsheet (2020). "Logistic Regression — ML Glossary documentation". In: *https://ml-cheatsheet.readthedocs.io/en/latest/logistic*$_r$egression.html.

[3] Gorman, Ben (2017). "Gradient Boosting Explained". In: *https://www.gormanalysis.com/blog/gradien boosting-explained/*.

[4] Maklin, Cory (2019). “Gradient Boosting Decision Tree Algorithm Explained”. In: *https://towardsdatascience.com/machine-learning-part-18-boosting-algorithms-gradient-boosting-in-python-ef5ae6965be4*.

[5] Nash (1994). “The Population Biology of Abalone ($_H aliotis_s pecies) in Tasmania. I. Blackli$ In: *https://archive.ics.uci.edu/ml/datasets/Abalone/'%3EAbaloneDataset%3Cspan%3E*.

[6] Yöntem (2019). “DIVORCE PREDICTION USING CORRELATION BASED FEATURE SELECTION AND ARTIFICIAL NEURAL NETWORKS”. In: *https://dergipark.org.tr/en/p*