Joshua Santillo
Dr. Isbell's CS 7/4641
February 4th, 2018

Analysis of Supervised Machine Learning Algorithms

A Note on Graphs

On all graphs, the blue represents cross validation performance and yellow represents testing performance. For all graphs, the left graph is Air Quality Data and the right graph is Tornado Data.

Supervised Learning

Supervised learning is the subset of machine learning that involves fitting a function to a set of inputs and outputs, as discussed in lecture.This is accomplished using a number of methods, including decision trees, k-nearest neighbors, artificial neural networks, and support vector machines, each of which will be explained below.

These algorithms are used here for classification, a subset of supervised learning involving the determination of using the fitted function to predict an output from a set of inputs. Because of the nature of the algorithms studied, the output should fit into one of an enumerable set of classes.

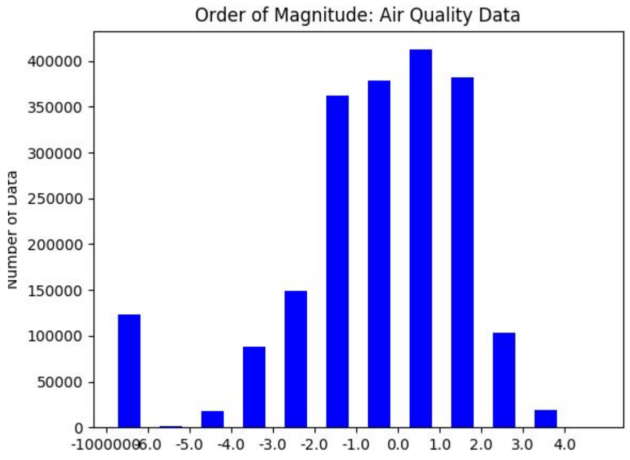
Each of these algorithms have a few common considerations to understand. The first is preference bias, meaning that each algorithm will favor certain hypothesis over others in the construction of a model, such as correctness, data structure size, or distances between data. These qualities guide the algorithm into classifying the data into one set over another. Another characteristic is bias, the tendency for an algorithm to make simplifying assumptions about the data that it's trained on, so high-bias algorithms are trained faster but will work best on data that does not defy the assumptions-- in other words, they perform best on datasets with consistent trends in data. Because of this, low-bias algorithms are preferred so they can handle any number of datasets, despite slower training time. The variance of an algorithm describes how much the model changes based on changes to the dataset. A good algorithm should have low variance, which implies that it's learning the underlying indicators of class in a dataset.

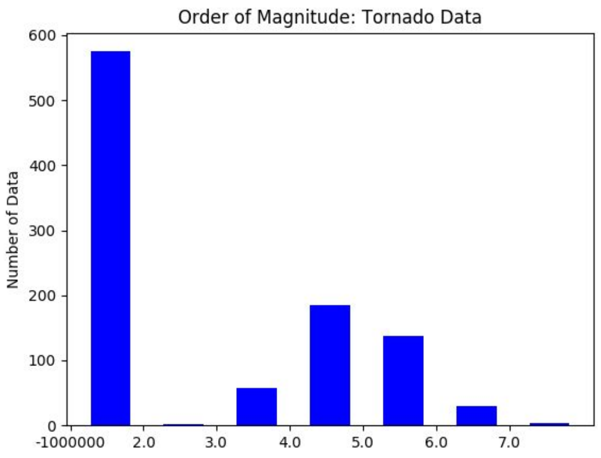
There are also some trends in performance caused by the way the algorithms interact with the datasets, the most important of which is overfitting. Overfitting is when the dataset is trained into believing qualities about the data that is not necessarily true, and when a model that has been overfitted to the training data is tested, the model performs poorly because it learns to assume things from the testing data.

For the purposes of this report, good performance is measured as having relatively high testing scores, but as will be discussed, good performance does not necessarily mean having an effective model. In addition, the cross-validation scores, presented in blue, act as a reference point to help determine where overfitting occurs. Also, each pair of graphs in a section dedicated to a classification algorithm corresponds to a hyperparameter under inspection.

Datasets

Both of the datasets studied are based in scientifically recorded measurements that are completely grounded in objectivity. This means that if there are inherent scientific or mathematical relationships between inputs and outputs, the classification algorithms should be able to model them. In order to make concrete categories out of highly varied data, the classes for both datasets are orders of magnitude, chosen for their hard cutoffs and ability to effectively categorize broad ranges of data. Also, note that -1,000,000 is a value given to all entries where the value is zero. Because this is an order of magnitude classifier for both datasets, log(0) is undefined, but can estimate it by giving it value -1000000 since it's a value small enough compared to the other values studied that it functions as intended. Presented below are the frequencies of each category for both datasets:





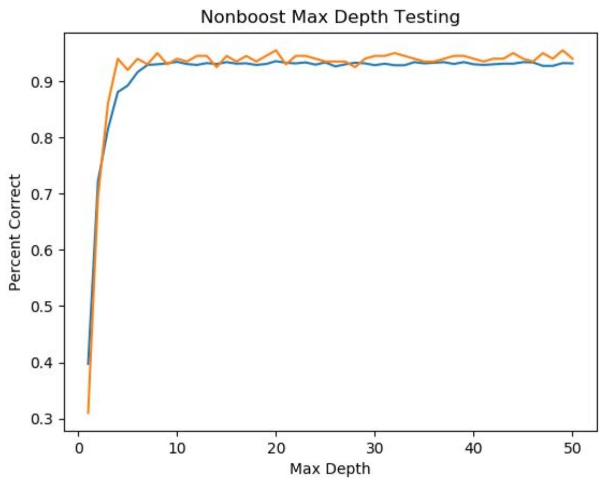
The EPA Air Quality Reporting Data is attempts to use higher percentiles to estimate a mean. In terms of testing performance of an algorithm, it is measured by how well it learns the relationship between existing statistical information and what the mean for that particular instance is. Also, there was a great deal of overfitting across most algorithms-hyperparameter combinations that was solved by reducing the size of the dataset from 8,000 to 2,000 rows.

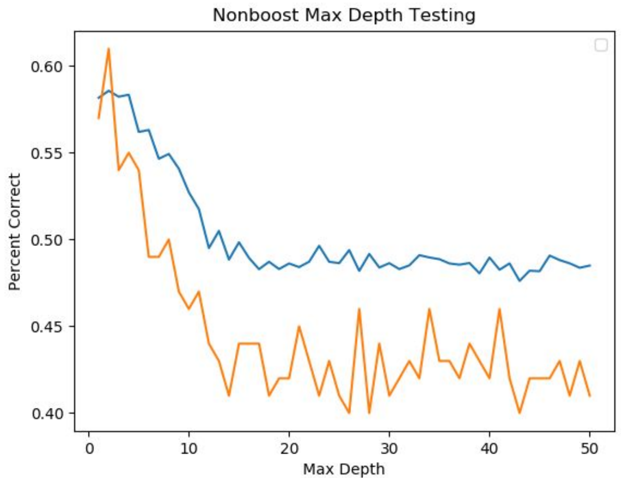
The tornado data is interesting because information such as the width, length, and longitude/latitude coordinates are used as inputs to determine the order of magnitude of damage in dollars lost that a tornado might cause without knowing the wind speed, meaning that all the information could be discovered solely with visual information captured from satellites. This could be used with tornado size predictors to help understand when to secure or evacuate areas as a storm is forming.

Between the two of these, both the number of rows and attributes in the EPA dataset were larger than in the tornado dataset, giving insight into the scaling problems of each algorithms performance when categorizing test data, as discussed for the affected algorithms. Additionally, because such a large portion of the Tornado dataset does end up having outputs of -1000000, and there is a relatively even distribution of the Air Quality dataset between four of its classes, it is expected that this example of centralization vs distribution will show how differently the algorithm performs with significantly different datasets.

Decision Trees

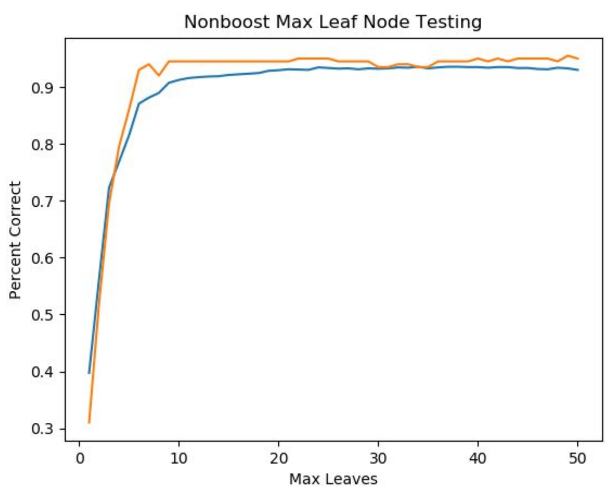
Decision trees are high-variance, low bias algorithms which classify data by using nodes as questions, edges as answers, and leaves as outputs. In this model, information about a piece of data is used to answer questions about it, moving further and further down the tree until a leaf node is reached. The deeper a tree, the more questions about the data are asked, and the more complicated it becomes. A non-boosted decision tree is employed to demonstrate the effects of changing the maximum depth of a tree on how well it performs, and then do the same but by changing the maximum number of leaf nodes that the tree is allowed to contain.

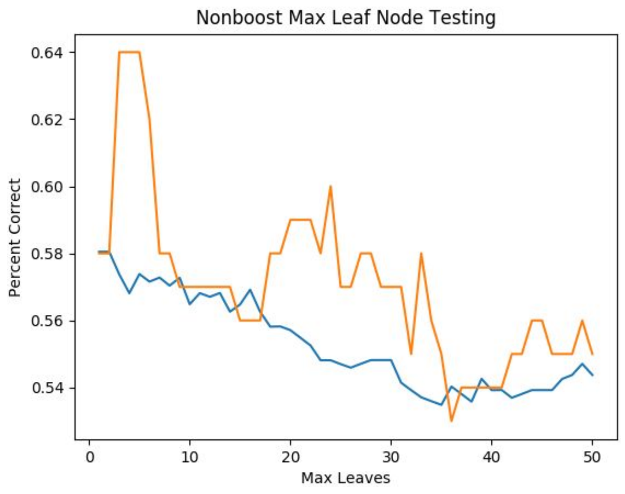




The left graph, showing that the EPA Air Quality mean predictor performs optimally with at least a maximum tree depth of five, also showing that any deeper trees do not improve performance more than a marginal, possibly coincidental amount. This happens so quickly because at a depth of 9, the splits are large enough to effectively divide the data into their correct categories. What is most likely is that the nodes of the decision tree are questions about the statistical attributes like percentiles and maximums, and using those to classify average air quality for the day.

In the graph of the Tornado data, with a shorter tree, the algorithm performed better. In fact, with a depth of under six, the algorithm tests optimally, which implies that the best splits happen earliest in the construction of the tree and a greater depth only serves to confound the categories. It’s also important to note that beyond a depth of four nodes deep, the data becomes more and more overfit to the training data, and performing less adequately on the testing data.



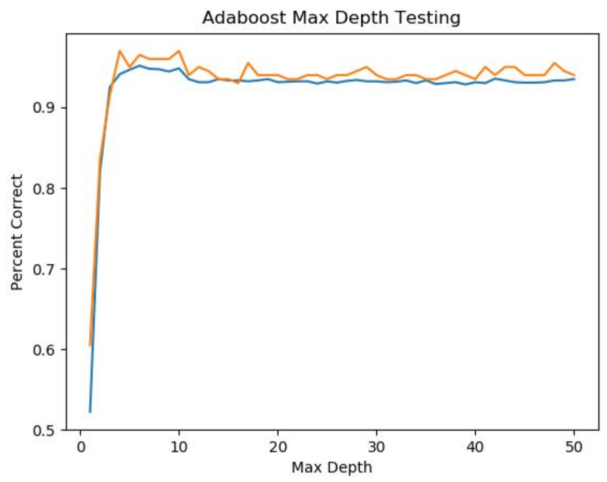


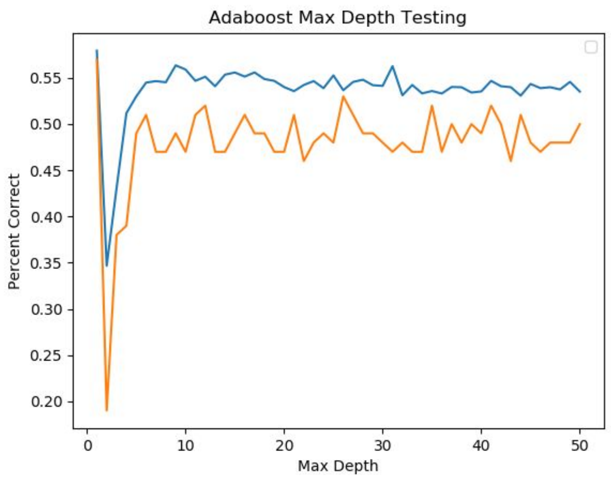
Here, in the Air Quality dataset, increasing the number of maximum leaves works well to improve both the cross-validation score and the testing score, with particularly diminishing returns after nine leaves. This is surprising because diminishing returns could be expected after at least eleven leaves, which is the number of classes, but instead, the classifier operates well enough after only nine because a vast majority of the dataset fits into nine categories. On the Tornado dataset, the model works best when the maximum number of leaves is close to the number of classes in the dataset, but further leaves only confuse and overcomplicate the model.

For each decision tree, because all of the questions had to do with floating-point number, passed in as input data, the decision tree was able to break down each attribute into ranges and decide what data was more likely to fall into a certain category. This was more difficult for Tornado data than the Air Quality data because the Air Quality data had generalizable relationships between the statistical data given and the mean for the day. The tornado data is more complicated because it has no way of picking up on knowledge such as the relationship between latitude and longitude being related. However, it was easier to train the model using two discrete values for latitude and longitude than coming up with a formula to give it a lat-long tuple to map it to numeric geographic information. The statistical data from Air Quality was more likely to have trends in the peak air pollution and where the mean might stand based on that because the tree is able to see a more clear relationship, such as higher maximums leading to higher averages.

Boosting

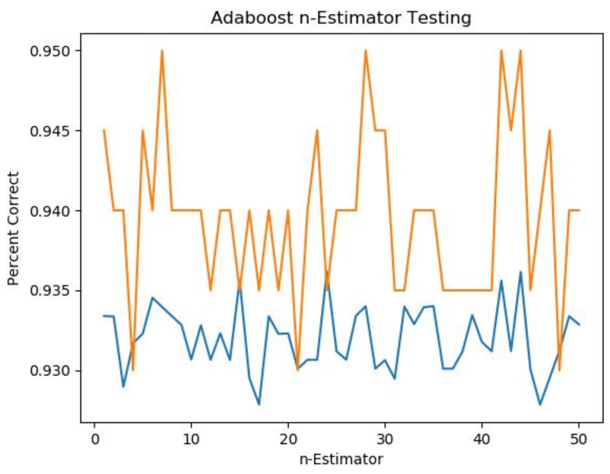
Boosting is the process combining weak learners about the data. This is achieved by determining which examples are difficult to classify by taking incorrectly classified data and combining weak rules used to classify them to create more accurate, strong classifiers. This allows us to have fewer classifying steps, and offering more utility in terms of performance. Here, boosting is applied to the same decision tree algorithm used above, and below are the boosted decision trees for Air Quality and Tornado data respectively:

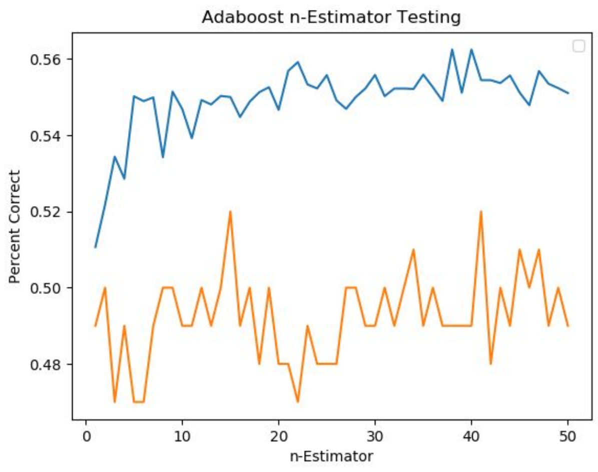




The performance of Air Quality data is at its peak at a depth between eight and ten nodes deep, but the testing performance declines as the tree grows deeper. When that performance declines, the cross-validation also declines, but performs more consistently than the testing performance. There are times when the testing score is worse than the cross-validation score, but it also means that the false assumptions that are made about the data only defy the assumption for a small set of examples.

The Tornado dataset is notably different. In this instance, the highest that testing performance exhibited is greater than 55%, but beyond a max depth of one, the boosting reaches optimal performance only beyond maximum depth 7. However, for all max depths, the test scores are not better than the cross-validation scores, which means that the model learned from data that does not generalize to the dataset.



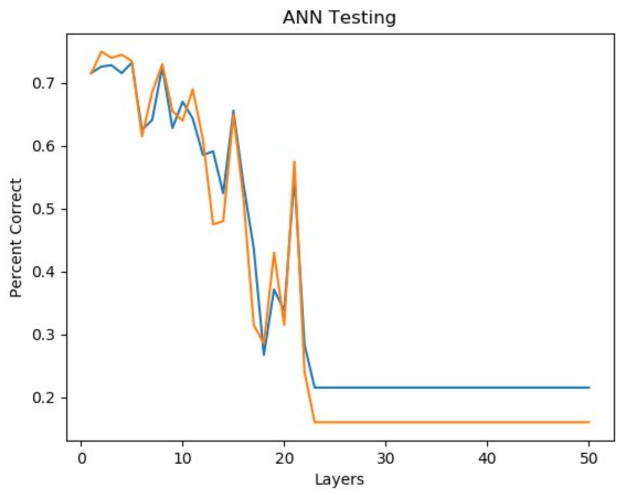


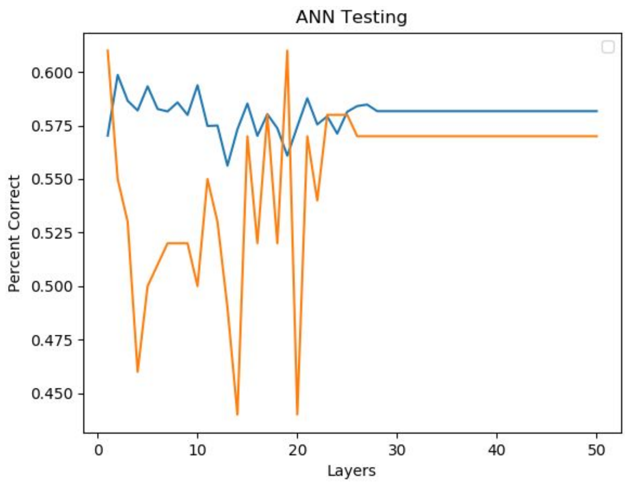
Here, for the Air Quality dataset, increasing the n-estimator causes very little change in the performance of the classifier. This is likely because there are few weak learners, or enough strong learners to effectively classify elements. Boosting would combine weak learners into strong learners, but here, combining the weak learners has marginal effects on the performance of the algorithm.

For the Tornado dataset, the model was always overfit by a hefty margin. The cross-validation improved as n in n-estimator increased, while the test score improved to a lesser degree. The improvement in performance means that there are, in this case, sets of weak learners that adaboost has combined to make fewer, strong learners.

Artificial Neural Networks

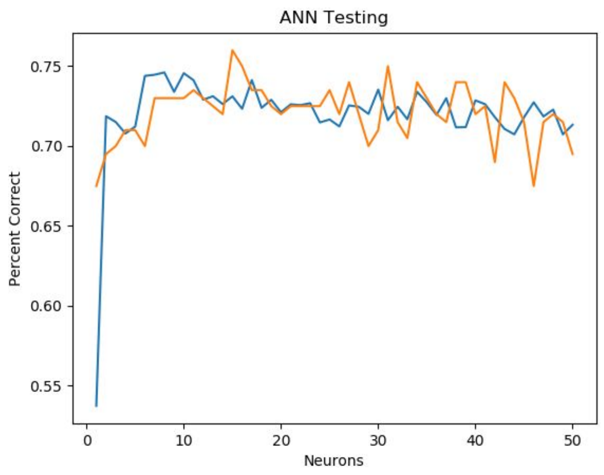
Artificial neural networks are composed of layers of perceptrons, and with larger datasets and larger sets of layers, it is one of the more time-costly classification algorithms analyzed. In a neural network, nodes, called perceptrons, are organized into layers. On each neural network, there exists one input layer and one output layer, but any number of hidden layers, including a networks without hidden layers. Each of these layers are composed of any number of perceptrons, one or more, each of which may receive signals from prior layers, which the perceptron weighs based on how important the perceptron perceives the incoming signal to be. When the incoming signals on a perceptron reach a particular threshold, the perceptron activates an outgoing signal, which is an incoming signal for the next layer of perceptrons. The reason for the performance of a network can be difficult to determine because it is difficult to analyze a trained neural network because the network has so many moving parts automated by the training process. Here, both the number of layers and the number of nodes in a layer are used as hyperparameters to analyze the performance of an artificial neural network.

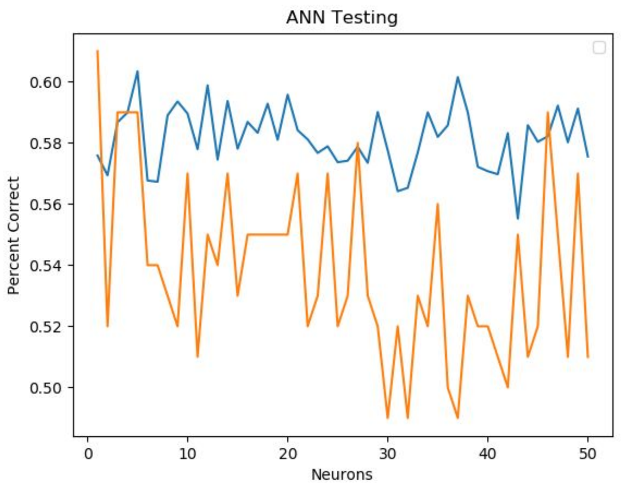




The performance for an artificial neural network on the Air Quality estimation problem was not as good as it was for decision trees, but it was still effective at lower number of hidden layers. Performance suffered beyond ten hidden layers, and change in results was non-existent beyond 25 hidden layers, when the model was dramatically overfit to the data.

For the Tornado data, the model was consistently overfitted beyond a single hidden layer, which was the point that the network performs optimally. After that, the model performs poorly until 16 hidden layers, and no change is observable after 25 hidden layers where again, the model is overfit to the training data. Increasing score can be attributed to the decrease in variance that accompanies a higher number of hidden layers, especially since so many of the outputs for the Tornado dataset are in one particular category.





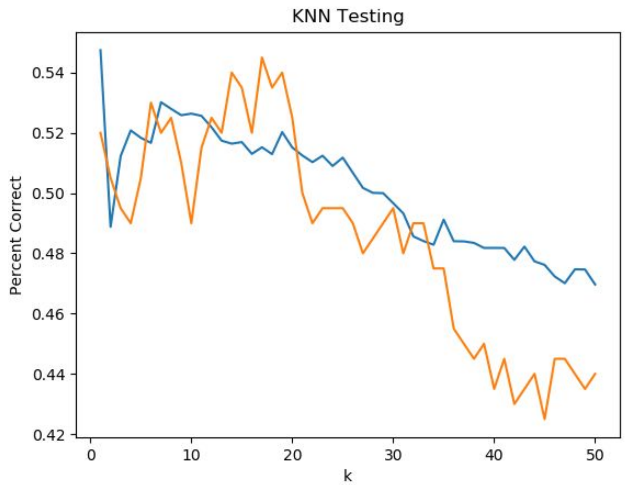
When the number of neurons is varied, the Air Quality data is modelled well after six neurons per layer. Beyond twenty neurons per layer, the model begins to confound the fit and performance declines slightly.

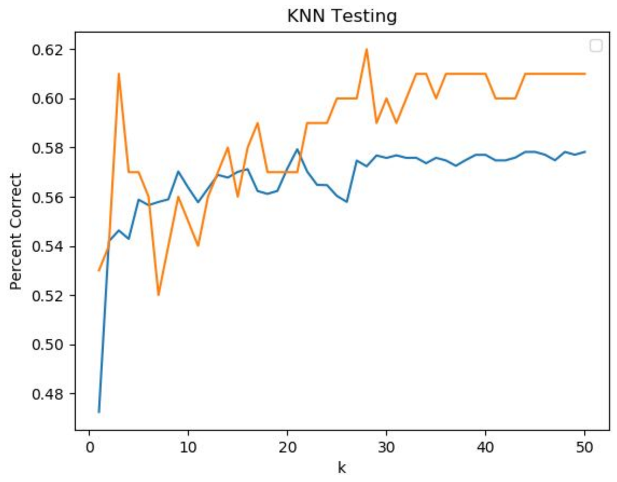
For the Tornado dataset, the model overfits after one neuron and seems to decline from there, but begins to increase after 30 neurons. For this set and for the last, the cross-validation score was fairly consistent for all numbers of neurons.

In our neural network, as discussed, it's difficult to determine what would cause certain behaviors or tendencies. However, because it's a large set of perceptrons with incoming weights re-evaluated by backpropagation, it only prefers to choose to give large weights to signals that tend to give correct results and weak weights to signals that tend to give incorrect results.

K-Nearest Neighbors

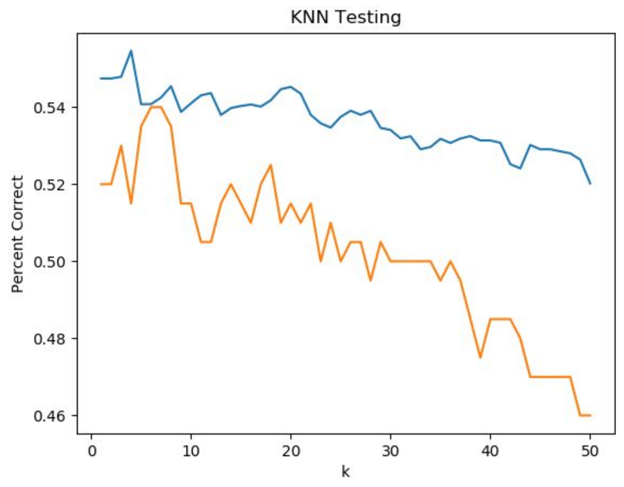
K-nearest neighbors finds the closest k data to the input in question, using an appropriate distance function, and applying the mode of all k of them to determine an output for the input in question. Larger values of k lead to high bias and low variance, since for large k, there are k-1 other data weighted alongside each other point in k, meaning the model does not change significantly and therefore has low variance, but tends to have higher bias because k's which do not fit the set do not have much influence because they are likely to have greater distance from the data in question. The first analysis of k-nearest neighbors was done counting all nearest neighbors equally. This means that regardless of the difference between the datum in question and the near neighbors, all neighbors are weighted equally. This is useful when data comes in discoverable and distinct clusters. In both datasets, this is not the case. For the second analysis, a euclidean distance function is applied to find the k-nearest neighbors and use those distances to the k neighbors for determining how much to weight the data. Good performance in this configuration of k-nearest neighbors operates under the assumption that closer data will have y-values similar to the point in question, and that as they become further away, the data become more and more dissimilar.

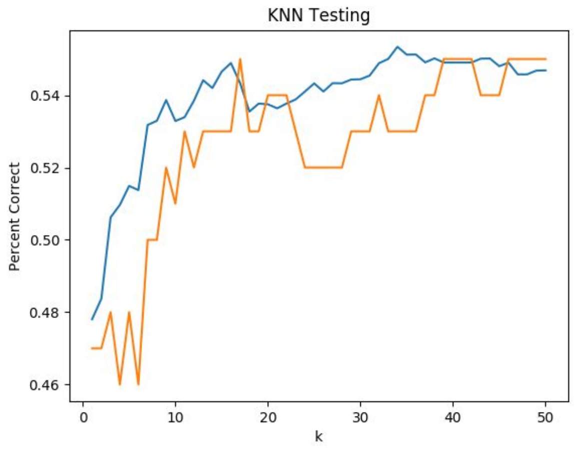




For the Air Quality dataset, the model performs best when it makes assumptions from a single nearest neighbor. Most tests beyond k=20 show an overfitted model that dramatically declines in performance, but between fifteen and twenty, the model performs best. This implies that in this fifteen-to-twenty range, the model gets a wide enough sample set to effectively determine an output, “effectively” being a term relative to this set of tests.

For the Tornado dataset, as the model increases n, the performance increases, but has marginal improvements beyond 32, which makes a lot of sense because there is one output that applies to most of the dataset, so if the model chooses that one constantly, it turns out that it is correct over sixty percent of the time, which seems like a major disadvantage of KNN as a modelling system.





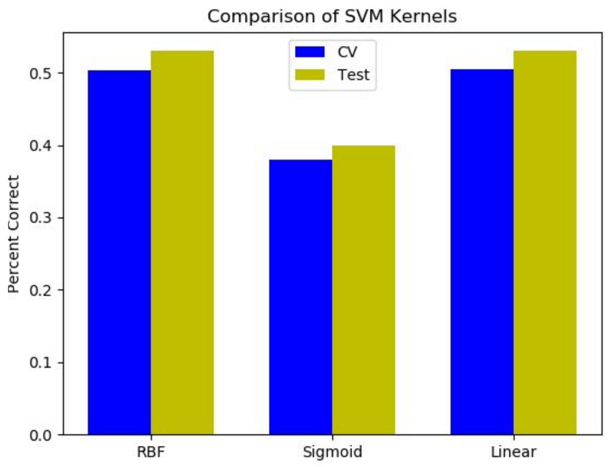
These graphs show a Euclidean distance version of KNN, where the distances are weighted. For both datasets, it appears that, since the cross-validation scores are higher than the testing scores, the model is overfitted to the training data.

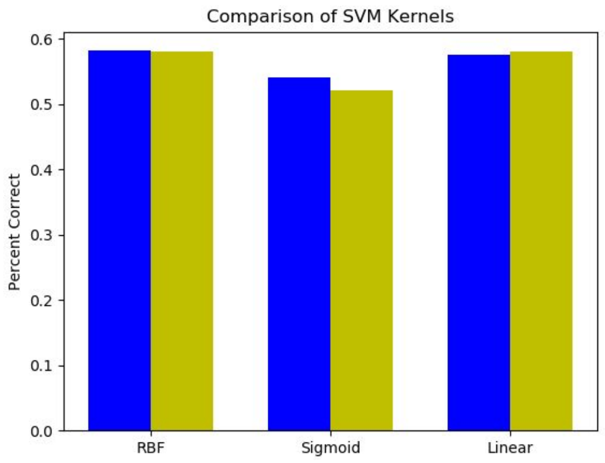
For the Air Quality dataset, as k increases, performance declines gradually. This means that the best representatives to use are closest to the input in question, and points further away only confuse the model. This implies that the nearest neighbors do the best job of modelling the data, but at below sixty percent correct, it still is not a good way of analyzing the data.

For the Tornado dataset, as k grows, performance improved, again because the more input it has, the more likely it is to output the category that occurs most often in the dataset. It is easy to misperceive the improving performance as the model improving, but on the contrary, the model is only choosing to output the mode of the dataset, giving us an extremely simplified model of the existing data because it works for a majority of datapoints.

Support Vector Machines

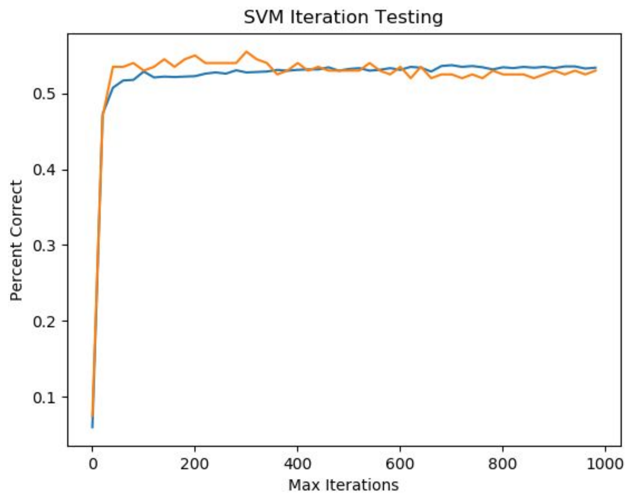
Support Vector Machines are a way of setting boundaries between classes, and these boundaries by generating lines that hope to maximize the margin, a distance, between the classes. For any input, whatever boundary it falls into is how the support vector machine classifies it. In our first two graphs, the performance of different kernels are compared with unlimited iterations on boundary definition. The kernels are transformations on the data so that the set can be represented linearly. In the second two graphs, a maximum is imposed on the number of iterations that SVM is allowed to test lines that define boundaries before termination

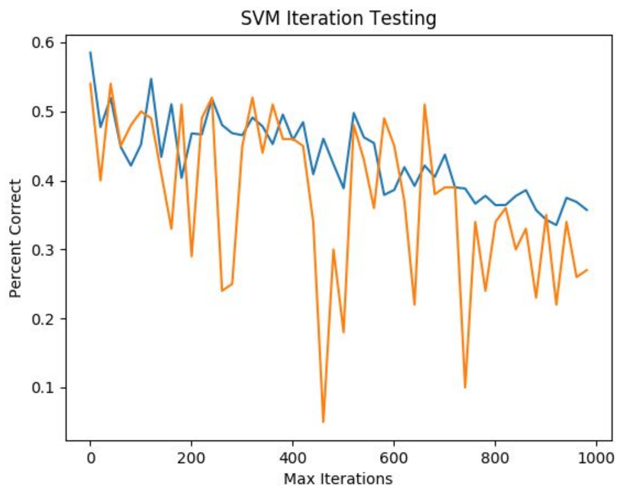




For the Air Quality data, no kernel worked particularly well in linearizing the data. However, the sigmoid kernel works particularly worse than the other two. Despite these insights, little information exists about how well this data can be modelled in space.

In the Tornado data, similar results occur because no kernel worked well, but they all worked better for this dataset than on the Air Quality dataset. This is because the Tornado data had fewer attributes, i.e., fewer dimensions, and because of that it could calculate larger margins between classes.





Here, the maximum number of iterations is limited and use a linear kernel, and for the Air Quality data it's almost inconsequential after 80 iterations, implying that this dataset is very quick to generate lines to its maximum potential. After 400 iterations, it begins to decline slightly and overfits to the training data.

In the Tornado dataset, increasing the number of iterations causes the performance to decline and eventually begin overfitting. Surprisingly, this graph shows that the cross-validation scores are often higher than the training scores, whereas in the bar graph, the cross-validation score was just lower than the training score. The default max iterations that the bar graph ran on used no limit, so the only assumption to make is that the bar graph chose to use the optimal number of maximum iterations for the training score.

Conclusions

This exercise was very interesting in terms of performance, because the Air Quality dataset was extremely predictable using decision trees, with or without using boosted ensembles as long as the max depth and max leaves were both past a surprisingly low threshold. This implies that the decision tree alone used particularly strong learners that had extremely effective splits. However, the only other algorithm that performed adequately on that dataset was an artificial neural network, which suffered from increased variance with large number of hidden layers and perceptrons, as evidenced by the training score dropping below the cross-validation score. In addition, the reason that it performed poorly on k-nearest neighbors and support vector machines was because those two both use distance in space, and the algorithms couldn't spatially map the data in a way that makes sense in terms of relating inputs to outputs. This dataset clearly suffers from the curse of dimensionality, as evidenced by the poor performance in k-nearest neighbors and support vector machines, but also by the decision tree's ability to perform well by determining which questions about the data give the most effective splits and have the greatest information gain.

For the Tornado dataset the impact of having such a large proportion of the output in only one category was obvious. It was extremely challenging to eek out scores above 50%, but it was helpful to remove dimensions which only caused higher confusion across all algorithms, such as day of the week and day of the month. The higher-bias tests gave the highest test results, but that result came from the model consistently categorizing inputs as "-1,000,000" and being correct between 50% and 60% of the time. It wasn't able to effectively categorize, but it could achieve peak performance by guessing the mode of the outputs. On high-variance tests, the algorithms were susceptible to overtraining and couldn't generalize to the entire dataset. It was a result of the algorithm seeing many instances of -1,000,000 in training and applying arbitrary rules to classify it. These results imply that factors such as latitude, longitude, tornado width, and tornado length have very little to do with the destructive force of a tornado, which is legitimately surprising. The size of a tornado would seem like a strong candidate for how much destructive potential it has, but the results of these tests make clear that wind speed is hands-down the most important factor in estimating losses caused by tornadoes.

Citations

Tornado Dataset
http://www.spc.noaa.gov/climo/online/
Air Quality Dataset
https://www.kaggle.com/epa/air-quality/data