Andrey Lukin
CS 4641 Machine Learning
Project 1: Supervised Learning
Professor Charles Isbell

All the code used for this assignment can be found at the Github link here, with the necessary READMEs and CSVs

Supervised Learning

Supervised learning is an area of Machine Learning that is focused on the ability to predict a function based a set of provided inputs and outputs of a function. There are many different algorithms to predict this function, but for this comparative report we will be focusing on five different ones: Decision Trees, Decision Trees with Boosting, Neural Networks, Support Vector Machines, and k-Nearest Neighbors. To compare them to each other we will be running two of the same datasets with same parameters, and we will be comparing their train scores, test scores, and cross validation scores, as well as changing up different characteristics of the algorithms to see how they affect our scores. For this comparison we will be using two data sets, which are:

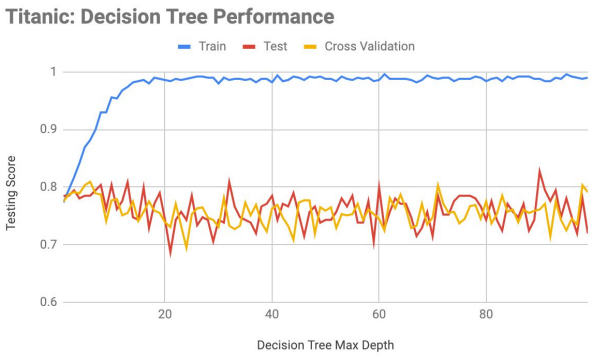
Titanic Survival Dataset:

The Titanic Survival Dataset contains data of almost a thousand different passengers on the Titanic, a historic boat that was known for its size during its time, and had a catastrophic accident when it hit an iceberg along the route and punctured its hull. With only 20% of the passengers and crew surviving the crash, historians try to look back to see what happened, and how this can be fixed for the future. As part of this dataset there are many different parameters that have been provided. For every passenger, there is a name, whether they survived or not, their sex, age, their “class”, the fare they paid to get on board, and other parameters that we will not be using for testing. We will be using the sex, age, class, and the fare of the ticket to predict whether or not the person would survive the Titanic crash. This could open up future discussion about gender equality in the past, and let us see potential class divisions of the time. The size of this dataset is only about 900 values.

The Rain in Australia Dataset:

The Rain in Australia Dataset contains the data of 142 thousand measurements regarding pressure, windspeed, rain, and sunlight at different locations in Australia. Parts of Australia are known for their dry periods, and as seen on a normal map, this pushed humans slowly towards specific locations in Australia. There is more than 80% of the Australian population near the coasts, making the majority of the continent without any population. With global warming become a bigger issue every year, it has become more important than ever to examine how it is progressing. With it happening on a global scale (as the name suggests) we have to try and **predict** the crazy effects that it will have on everyday life. This is why this model uses the humidity, max and min temperatures, and the pressure to predict the location. If we can compare locations to what they used to be, or predict to what they will become meteorologically, we can try and predict what will happen to the wildlife and humans there. The size of the data set is 56,000 values, and to do the report, only 1000 will be used to test, and then the full dataset will be used at the end.

Decision Trees



What decision trees are a learning algorithm that uses a tree data structure to categorize your dataset. To figure out what type of output you get with the trained function, you traverse the tree from the root node, and at every node answer questions regarding your data. As you get deeper into the tree, the questions become more specific. When you arrive at the leaf node, that will be the predicted output to the input you provided.

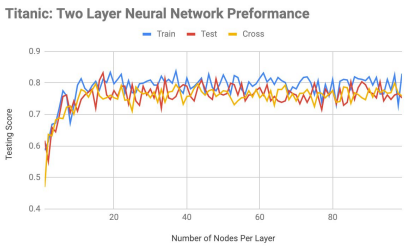
One important thing with decision trees is pruning, ensuring that the decision doesn't become too selective and the tree itself is well balanced. This way we can be sure that each question

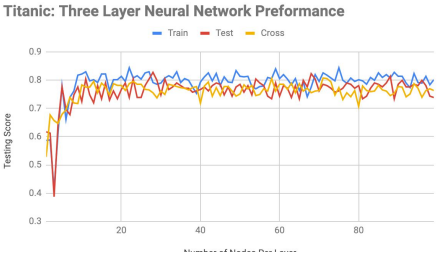
carries a large amount of impact on the final value, and we can ensure that there is no dead-weight in the questions.

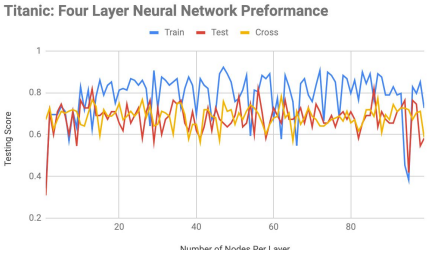
One main feature that is similar in both data sets is that the training score quickly grows to almost perfect, and stays there. What this demonstrates is overfitting: the tree's "questions" are becoming too specific, and are allowing the training dataset to perform considerably better than the testing one. To fix this, we need to pick the depth of the tree that has the highest testing/training score ratio. In the Titanic example, this would be anywhere from 2-3, but we have to be careful since the train score curve grows quickly there.

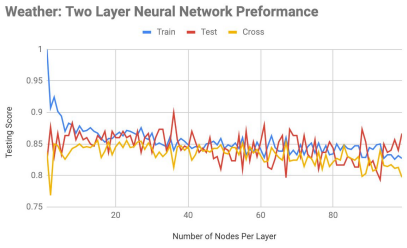
The weather model looks very similar to the Titanic one with one difference: it intersects in the beginning quiet a lot. This means the number of neighbors needed to not overfit it higher, and based on the graph we will be using 6 neighbors.

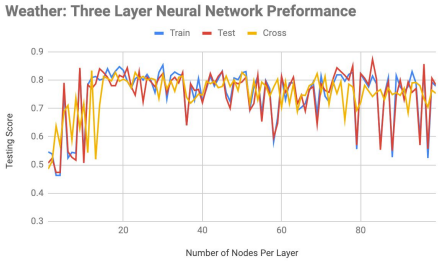
Neural Networks

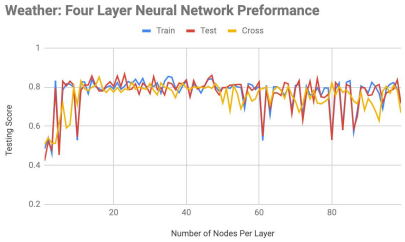












Neural networks are a supervised learning algorithm that mimics neurons. A neural network is made up of layers and nodes in those layers, which are connected from layer to layer (nodes in the same layer cannot be connected). With every neurons having a "minimum firing level", or the lowest value it must receive before it fires the input to the next neurons its connected to, this

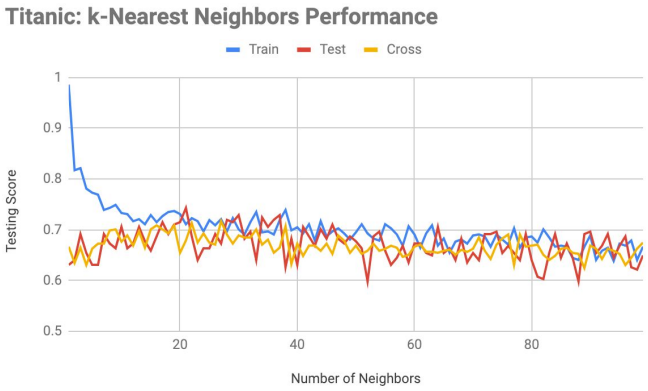
allows for a very complicated system to be formed. As discussed in lecture, a neural net can map out any function if there are enough layers and neurons in each layer.

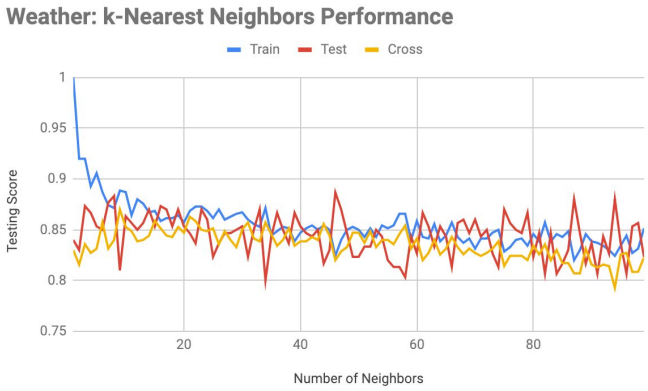
For this assignment, we are using the MLPClassifier of the **sklearn** Python library. This is an implementation of a multilayer perceptron, which utilizes backpropagation to improve the learning capabilities of the neural network.

As seen in the graphs above, using the data sets provided, there was testing to see the most optimal number of layers and neurons per layer to be used for the model. Every layer had the same number of neurons, and the different graphs represent the training, testing, and cross validation score as the number of nodes per layer increases. We see that the graphs have very asymptotic qualities, with all three values staying at about 78% accuracy for the best results. What is interesting though, is that the Titanic dataset becomes a lot more sporadic with four layers in the network. This might be because of overfitting; the values for the scores before a lot varied, and the difference between the train and test scores grows. With that in mind and based on the data, the Titanic model works best with a neural network made up of three hidden layers with about 8-10 neurons in each layer.

For the weather model, we see an interesting progression of how the neural networks are overfitting. As the number of layers grows from two, to three, and then to four, the graphs are becoming more and more variable. This is why we are sticking to 2 layers, 10 nodes at each layer (the 10 nodes were based on where the test and training scores converge).

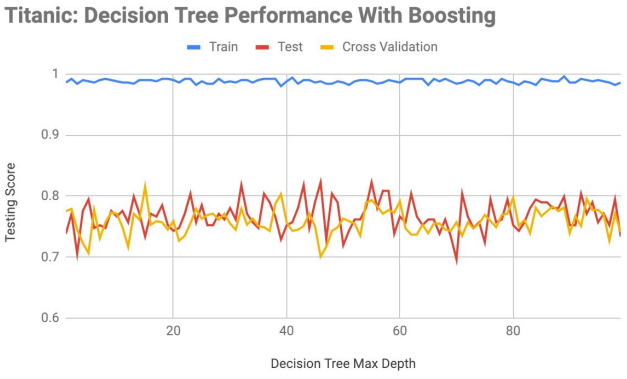
k-Nearest Neighbors

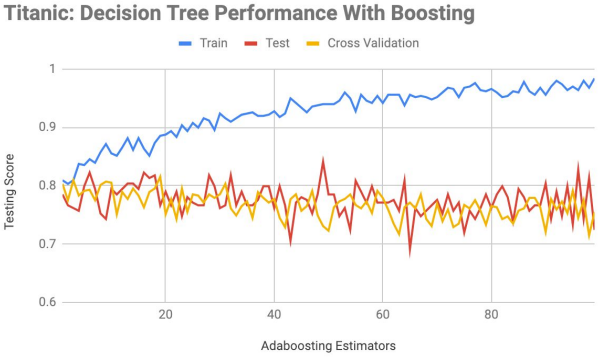




kNN is a pretty simple learning algorithm that utilizes the closest values, or “neighbors”, of the input to predict what the output is going to be. The “prediction” algorithm can be different, with linear or quadratic relationships or even a type of voting algorithm between the neighbors. For the purpose of this assignment, we will be using the **sklearn’s** KNeighborsClassifier. As part of these models, we play around with the different sizes of k, or the number of neighbors we look at to predict the output. With knn the problem of underfitting becomes more realistic: pick a k that is too large, and the output becomes too smooth and too “average”, outputting a safe average. On the other hand pick a k that is too small (like k=1), then the model becomes overfitted. Looking at the performance of the k-NN model with the Titanic dataset, we see that it converges at about 20 neighbors, with a testing score of about 70%. Because of that, we pick that as our k. We have a very similar convergence with the weather dataset, so we will pick 8 neighbors at the 87% score.

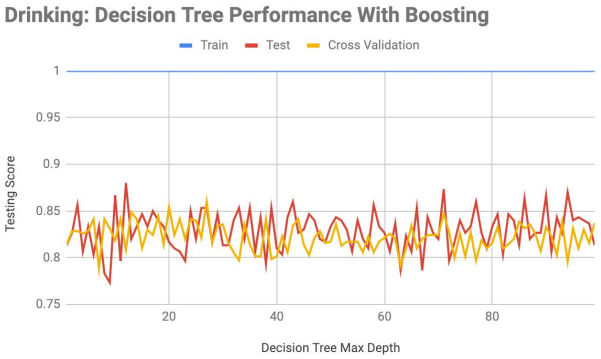
Boosting





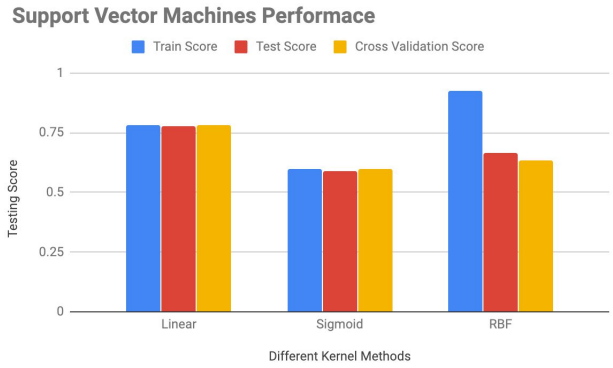
(The top right graph should say ‘Weather’, bottom left ‘Titanic’, and bottom right ‘Weather’) Boosting for decision trees is a method of adding separate weak learners, or small trees, together to create a much stronger leaner than before. There are many different algorithms to do this, but we are using the **sklearn’s** “Adaboost” or Adaptive Boosting.

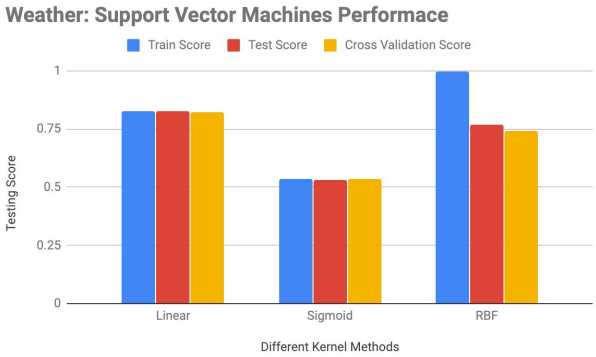
What we are trying to figure out with this dataset is the number of “estimators” that we should use in our model. The



estimators are another limit to ensure a more balanced decision tree. We first run it on the data sets, and see that right off the bat, the model is overfitting, and the number of estimators does not change anything. The train score is already almost at 1 with a decision tree of depth one. But, as we found out from a normal decision tree, we have a preferable depth. We use that depth, and provide it to the model we pass into the Adaboost function. Looking on the graphs, we want to have as high of a test/training score ratio, and find the maximum number of estimators before the model becomes overfitted. For Titanic, we pick 10, and similar to that we pick 10 for the

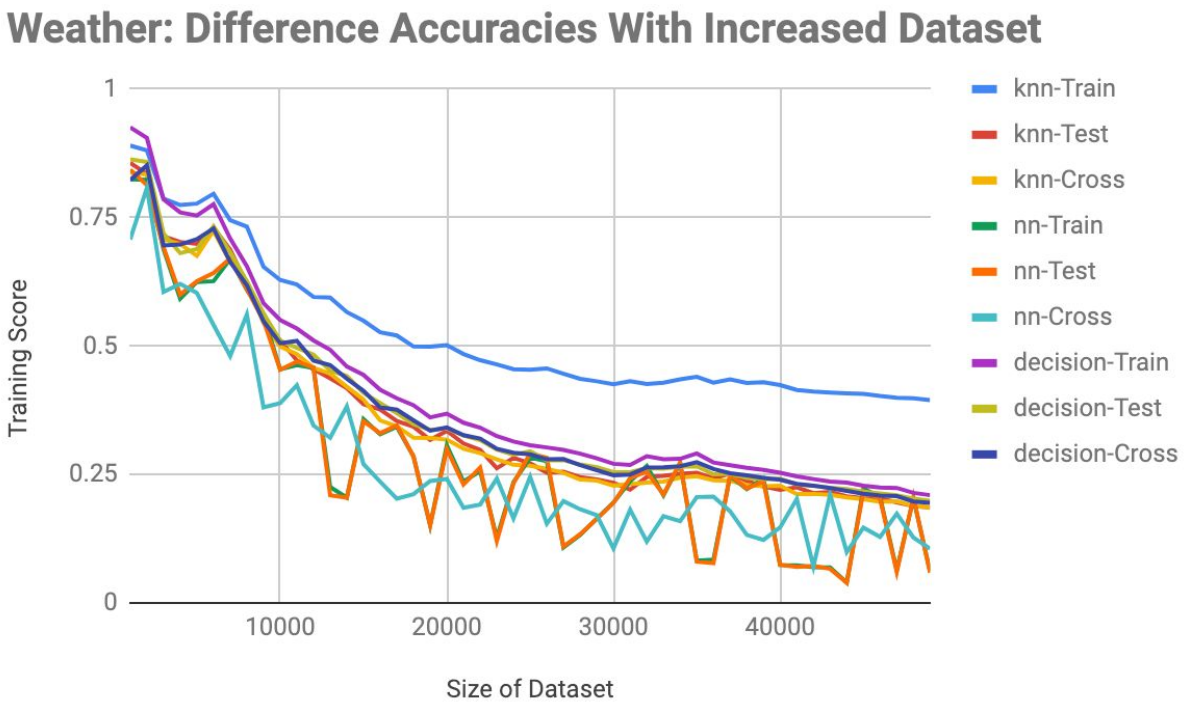
Support Vector Machines





A Support Vector Machine is a supervised learning algorithm that uses the provided "kernel method" or a type of function provided to it, and tries to find the parameters that the function will use to best fit the provided dataset. The kernel methods we will be using are a linear kernel, sigmoid kernel, and a RBF Gaussian kernel. In the Titanic dataset (on the left), we are comparing the score of the train, test and cross validation score. Here we see that even though RBF has the highest train score out of all, this demonstrates only the overfitting of the kernel method. What matters most is the cross validation and test scores, and with that we pick linear kernel as the winner. Interestingly enough, the SVM numbers for the weather model performed very similarly, and will have the same answers.

Conclusion



As part of the conclusion, the graph above demonstrates what happened to the different scores when run with 56000 data points from the weather dataset. It's interesting to see how all of scores slowly go down, and this might have been because of bad randomization of data on the modeling part.

In regards to the rest of the project, after looking over all of the different algorithms, the Titanic model performed best with the Neural Network, while the weather model performed best with k-Nearest Neighbors. The reason why titanic model might have performed best with the Neural Network is because of the high amount of parameters, while the k-NN performed better with the weather model because the data is more correlative and easier to predict. Out of everything in this project, I felt that the idea of overfitting was the most important: ensuring that even though you can increase the “score” of your trained model, this might not necessarily improve it overall.