CS 7641

Supervised Learning

Dylan Lawrence

The Data

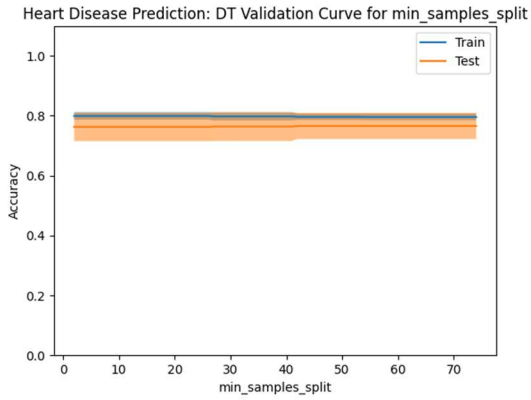
For my dataset I have chosen to use a dataset of patients tested for heart disease. This dataset is from Kaggle.com. I used this dataset for both classification problems. The first classification problem we will be exploring is predicting whether a patient tested positive for heart disease. For the second question we will only work the data from people who did test positive for heart disease, and we will be trying to predict the sex of the patient.

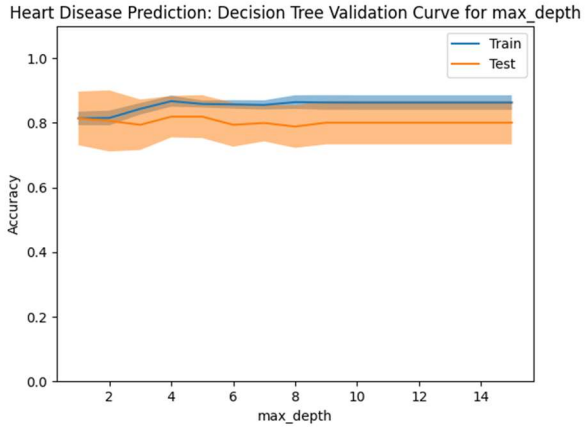
The dataset has 918 observations, with no missing values. The columns for this dataset are Age, Sex, ChestPainType, RestingBP, Cholesterol, FastingBS, RestingECG, MaxHR, ExerciseAngina (Chest Pain After Exercise), Oldpeak (Causes a depression in a graph when measuring electrical signatures in the heart, specifically when switching from resting to exercise or vice versa.), ST_Slope (The direction of the depression), HeartDisease (Whether or not the person has heart disease). As you can see many of these variables are categorical, and they were stored as text. During the data cleaning process, I converted them to numerical values.

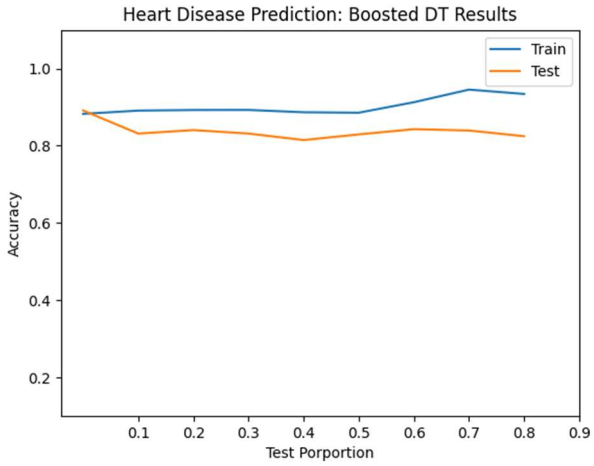
This question is interesting because heart disease is one of the leading causes of premature death. It will be interesting to see what it's indicators are, and if those indicators vary between men and women.

Question 1: Predicting Heart Disease

Decision Tree







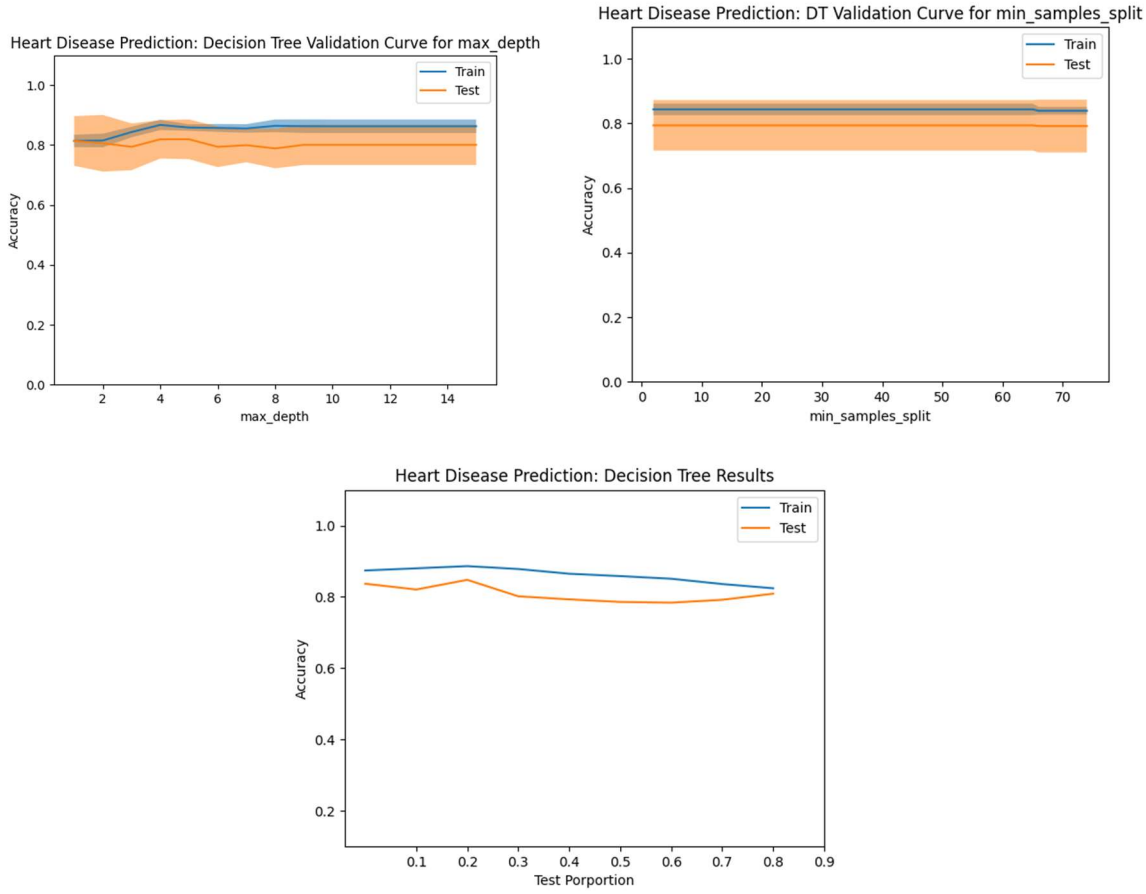
Starting with the decision tree I decided to begin by exploring the hyperparameters min_samples_split and max_depth, since in my previous learning about decision trees they were the most influential variables. When constructing my trees I used 'gini' to determine which variable to split on. Gini represents the probability that a randomly selected observation from node is labeled incorrectly.

This dataset, min_samples_split did not have a large impact on the accuracy of the decision tree. After exploring the initial tree produced, I realized this is because the tree was not creating any nodes that were separating small numbers of observations. This likely has to do with the strong correlation some of the variables have with whether the patient had heart disease.

Max_depth was also relatively uninfluential in the decision tree. This one I had a harder time figuring out as with a high max depth, the tree would go deep, but it did not have a large impact on accuracy (even before a value of 10, where it flattens). This could potentially be due to an error on me, but I imagine it likely has to do with the strong correlations again, particularly between RestingECG, Age, and MaxHR.

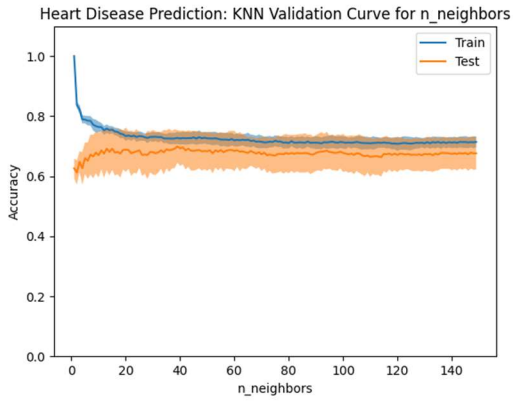
Finally, when testing the decision tree on different values of test_size, I found that the most accurate test_size was .2. To prune the trees I used the ccp alpha method that comes with scikit-learn. We had an accuracy of 86% on the test set, and a hair under 91% on the training set. This does not seem to indicate overfitting to me.

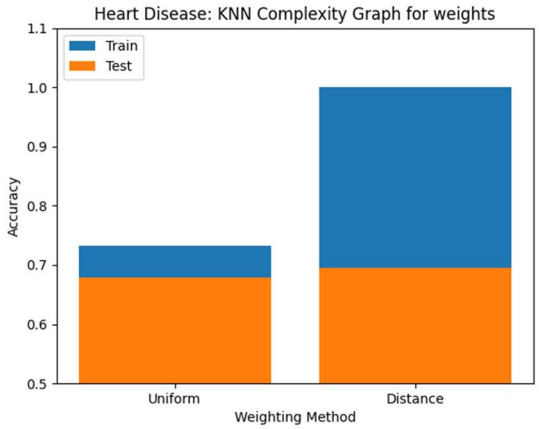
Boosted Decision Tree

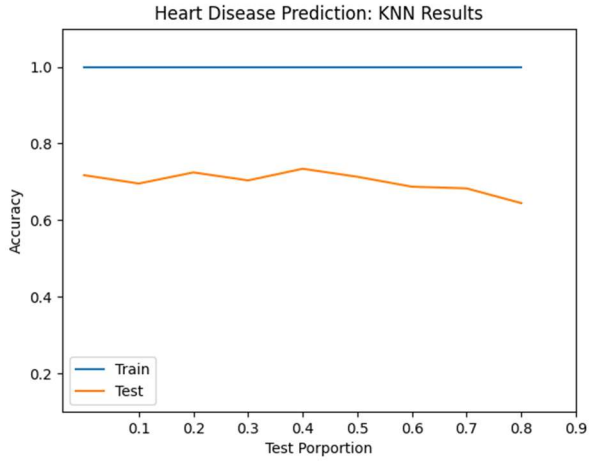


For boosted decision trees I decided to optimize the same hyper parameters as regular decision trees. The results for max_depth was very similar, and the results for min_sample_size are strange. For some reason I was having issues with the second where the training test would fit perfectly, yet I did not seem to have a major overfitting issue.

K Nearest Neighbors



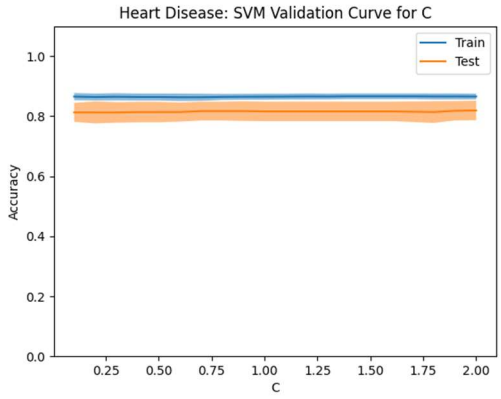


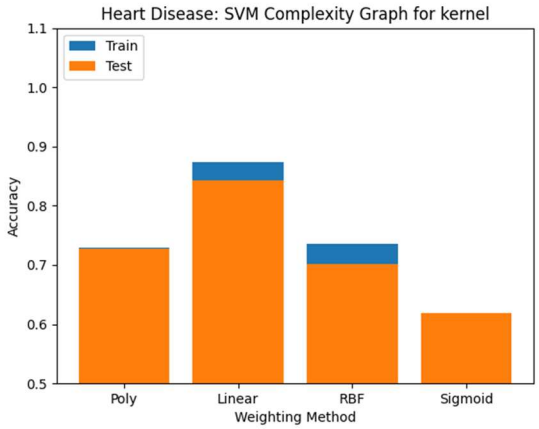


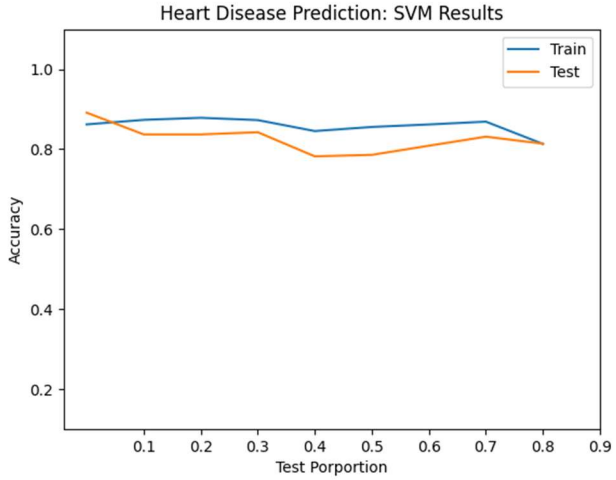
KNN did not work particularly well as a model on this data. I optimized for n_neighbors and weights. For weights there was two options, all neighbors would be weighted equally "uniform" or closer neighbors carry more weight "distance". N_neighbors wasn't particularly interesting, however with a low value it heavily overfits. The best accuracy was achieved with a value of 38.

Weights were much more interesting. Even though distance appears like it would overfit based on training results alone, it had a slightly higher accuracy of ~70% than uniform, which sat around 68%.

Support Vector Machines





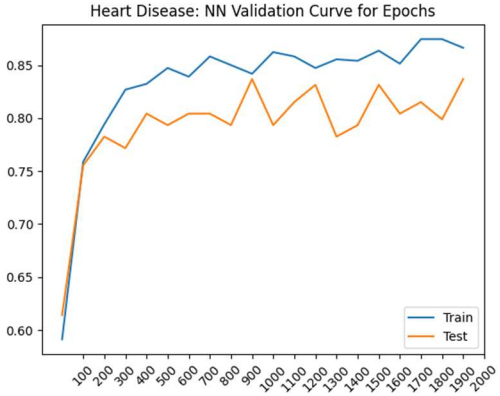


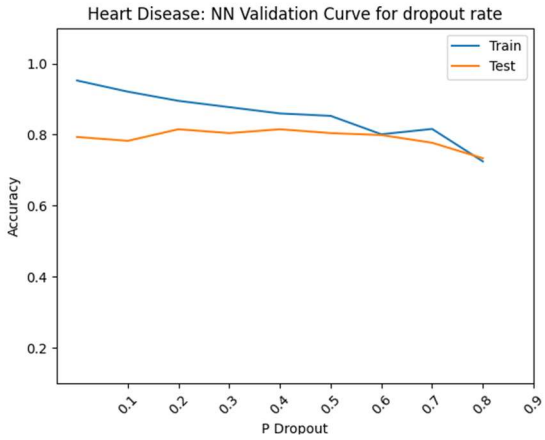
For support vector machines I decided to test both the Kernel, and C. C is a measure of how heavily the model is punished for misclassifying an observation. A high C value is an easy way to overfit.

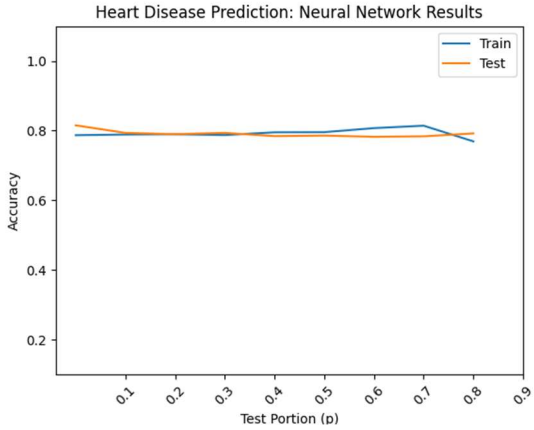
However, when testing C from a value of .1 to 2, I noticed that there is very little if any change. What this tells us regarding the data is that when placed in a theoretical space such as a KNN or SVM model, observations with heart disease and ones without do not sit near each other.

For the Kernels I used the 4 main ones that come with Scipy. To keep it short I will only discuss the clear winner. Linear is the simples kernel tested, the model draws a straight line to separate the two outcomes as clearly as possible.

Neural Networks







For Neural Networks I tested the number of epochs, the probability that a value would be dropped after going through the first linear layer. As you can see the number of epochs had a very volatile impact on the accuracy of the model. I tried to test with high epochs, but my computer was simply unable to complete them in a reasonable amount of time.

Dropout rate had an interesting impact on the outcome of the model. The ideal dropout was .2, however the higher the rate, until about .9, the less overfitting there was. While this is interesting, I don't think we want to sacrifice accuracy just to reduce overfitting.

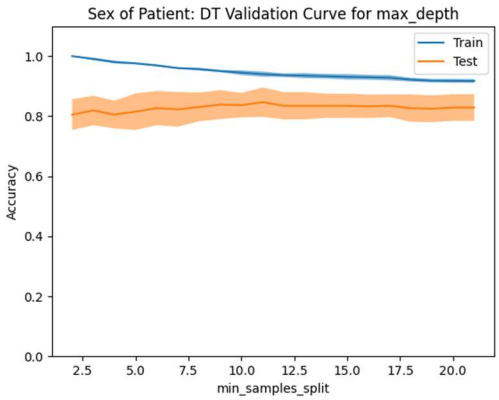
Another interesting detail about the Neural Networks is that this one was hardly affected by the test_rate. This tells me that the NN was able to extrapolate a lot of information from just a small amount of data. However, the ideal rate was .2

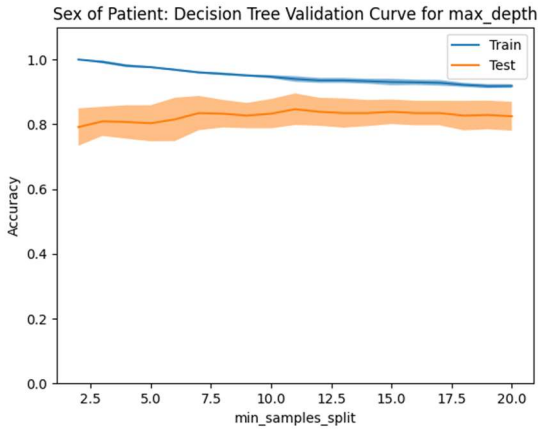
Question 2: Predicting Sex

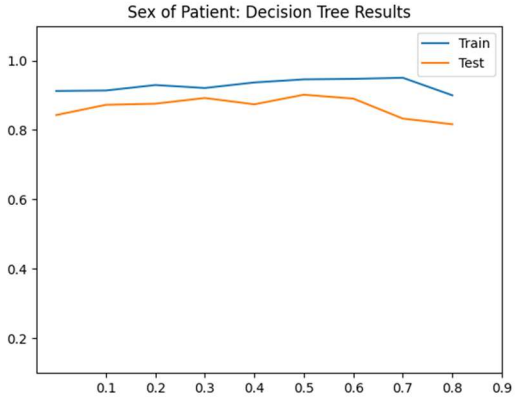
Due to not being permitted much space for the number of graphs required. I have decided to group all of question 2 together. First I cleaned the data again, but this time we are only keeping people who have heart disease, and trying to predict their sex. The results here were largely the same, however I noticed that I was getting higher accuracy with more basic models. I

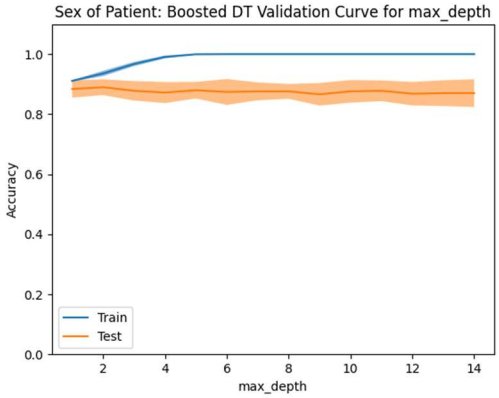
will call out a few that I find interesting. The main ones are again KNN, and SVM. The models in this question had the same outcome where the number of neighbors, or degrees (this SVM used a polynomial kernel) did not have a large impact on accuracy. I believe this is largely due to the indicators of heart disease differing between men and women, but that is for another project.

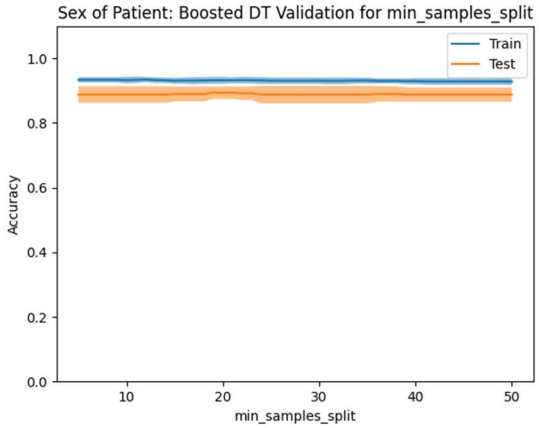
Additionally, my neural network was behaving strangely. I am unsure why the number of epochs made no difference in test results, but after trying to debug for a few hours I believe this largely came down to something on my end I was unable to debug.

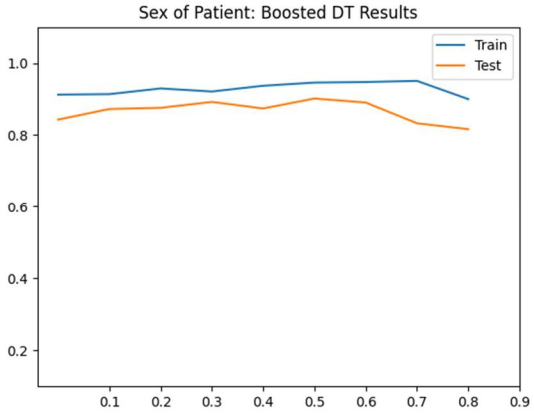


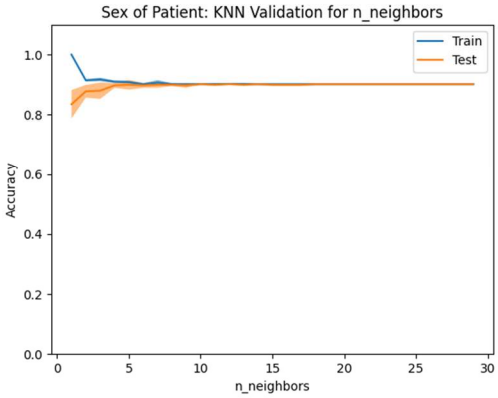


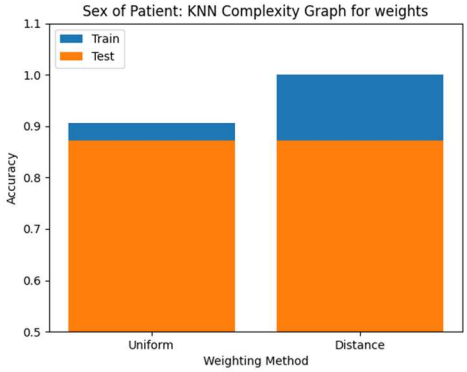


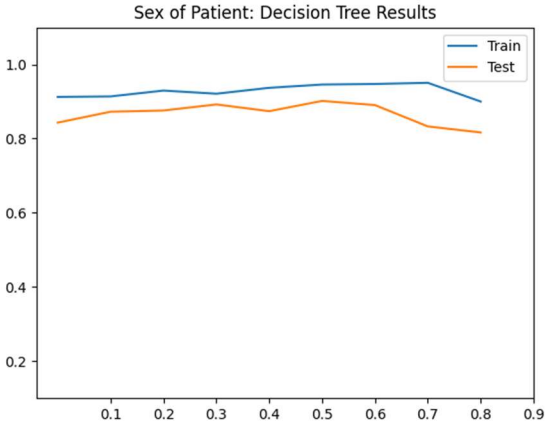

Sex of Patient: Boosted DT Validation Curve for max_depth
Accuracy
max_depth
Train
Test

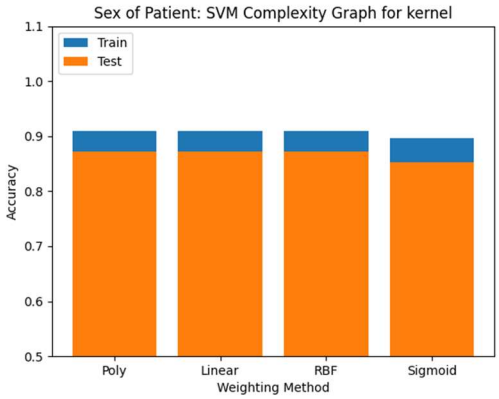

Sex of Patient: Boosted DT Validation for min_samples_split
Accuracy
min_samples_split
Train
Test

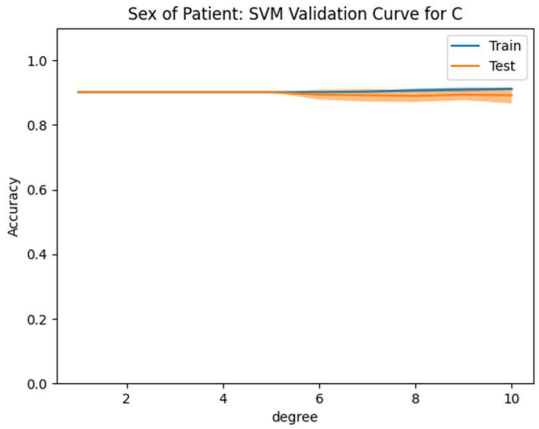

Sex of Patient: Boosted DT Results
Train
Test

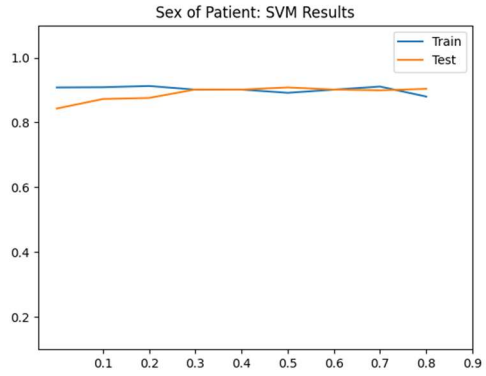

Sex of Patient: KNN Validation for n_neighbors
Accuracy
n_neighbors
Train
Test

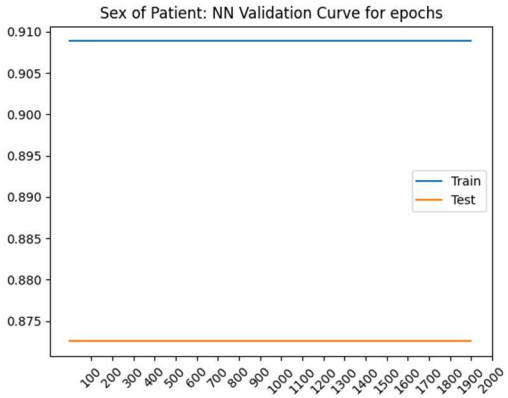

Sex of Patient: KNN Complexity Graph for weights
Accuracy
Uniform
Distance
Weighting Method
Train
Test

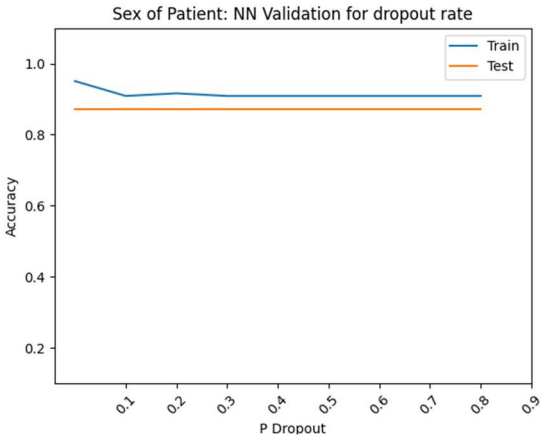

Sex of Patient: Decision Tree Results
Train
Test

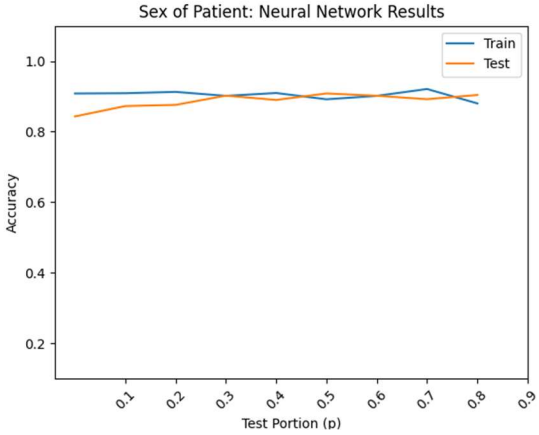

Sex of Patient: SVM Complexity Graph for kernel
Train
Test
Accuracy
1.1
1.0
0.9
0.8
0.7
0.6
0.5
Poly
Linear
RBF
Sigmoid
Weighting Method


Sex of Patient: SVM Validation Curve for C
Train
Test
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
10
degree


Sex of Patient: SVM Results
Train
Test
1.0
0.8
0.6
0.4
0.2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9







References

I only referenced course content and the following sites:

https://scikit-learn.org/stable/getting_started.html

https://pytorch.org/docs/stable/index.html