Preliminary Project Report

Introduction

For this project, we are going to build a machine learning algorithm that can accurately predict whether a patient has heart disease. We will be using the Heart Disease UCI dataset from Kaggle, found here. This dataset has 303 samples and 14 features. As the dataset is already in a CSV file, we were able to read the data directly into a dataframe using a library like Pandas.

We are going to use linear SVM and classification trees to predict if a patient has heart disease. We chose these machine learning algorithms because linear SVM is a parametric model and classification trees are non-parametric, and it may be useful to see which performs better. We plan to use Python for data analysis and modeling with Numpy, Pandas, and sklearn libraries.

In this preliminary report, we will show the results from preprocessing the data, running some of the algorithms mentioned above, and current progress with validation techniques.

Preprocessing

The dataset originally contained every positive entry for disease followed by every negative entry. This order may have created bias during training and cross validation, so we randomized the order of every row of data while keeping the data within each row and column consistent. We also found that two rows had null values for the “thal” feature. We decided to remove these rows because there is still a large enough amount of data without them and the two rows may have hindered our accuracy otherwise. Our progress here can be seen in the “preprocess.py”. “preprocess.py” is also the python file to run if you would like to see the results so far.

Algorithms

So far, we have implemented Linear SVM for heart disease classification and are working on classification trees. We used sklearn imported through Python to construct both models. The svm.py file has three functions for producing predictions. The first function serves as a test to show that the heart dataset can be passed to the function and the svm classifier will choose the columns to model on and predict for dummy data points. The second function works with kfold.py which passes the training data and returns the model built from the training data. The third function returns the predictions for the model and test data passed in. We have also started to implement validation trees for heart disease classification, and the algorithm can produce classifications and predictions between variables even if they are currently unoptimized. We have also been able to form visual representations of the trees using graphviz and DOT files. Our progress for these algorithms can be seen in “svm.py” and “validationtree.py”.

Cross-validation

We are working on an approach for k-fold cross validation with a k value of 15. Randomizing the rows in the preprocessing step was important, so that the k-fold cross-validation algorithm didn't exclusively train/test on positive or negative samples. "kfold.py" separates the data into training and testing datasets, passes the training data to svm.py and validationtree.py to return the models, then uses the prediction function in svm.py and validationtree.py to return predictions for the test data for each model. kfold.py returns the accuracies for both classification trees and svm over all 15 folds. Our progress towards implementing bootstrapping with 30 rounds has had similar progress but we are still working on making the validation testing work with both algorithms. The code for our progress on the k-fold cross validation can be seen in "kfold.py", and the code for our progress on bootstrapping can be seen in "bootstrapping.py".