Graduate Project

Shmuel Feld, Matt Glennon, Jay Salgado, Aban Khan

For this project we chose a "Heart Failure Prediction Dataset" from Kaggle (heart.csv). This dataset encompasses 11 clinical features for predicting heart disease events which includes: age, sex, chest pain type, resting BP, cholesterol, fasting glucose level, resting ECG, maximum heart rate, exercise induced angina, old peak, ST_slope and heart disease output. This dataset also was created from five different individual datasets: Cleveland, Hungarian, Switzerland, Long Beach VA and the Stalog Heart Data Set. Heart Failure Prediction Dataset is non-trivial due to the number of factors that influence heart failure. Additionally, there isn't always a linear relationship between the precursors to heart failure and its outcome as doctors increasingly point to environmental/diet stresses as an additive element to heart failure outcome. Such interactions become increasingly difficult to capture in models. Our dataset will demonstrate the differences between decision trees, neural networks and k-nearest neighbors clearly. Decision trees excel in following paths with clear categorical determinations which can be done thanks to the clinical features in this data set. Neural networks excel with building connections between non-linear relationships which can help for a dataset like this where a connection between resting blood pressure, max heart rate and heart disease, for example, may not be so linear. K-Nearest Neighbors excels with labeled data using a distance metric which may be beneficial for grouping based on density of clusters within our dataset. Our input features for this dataset includes age, sex, chest pain type, resting BP, cholesterol, fasting glucose level, resting ECG, maximum heart rate, exercise induced angina, old peak, ST_slope and heart disease output. We decided our data was as clean as it could be as each factor was important in determining the outcome of heart disease. Our target variable is a binary output where 0 indicates the absence of heart disease and 1 indicates the presence of heart disease.

We also felt it was an important data set to evaluate due to a team member's recent exposure to heart disease and failure. Using a dataset like this had an emotional impetus as much as an analytical one in analyzing a real problem that had real world implications.

K - Nearest Neighbors

For our use of the K-Nearest Neighbors (KNN) algorithm, our study was largely centered around testing different values for K, to find an optimal value for the accuracy of our model. First, we tested our algorithm on the employee data, iterating over a for-loop of K values and reporting the accuracy during the testing phase. Initially, even with a K value of 1, our model was 45.75% accurate. Then, as K increased, the accuracy also increased following a logarithmic growth pattern, up to a final value of 65.45%, which occurred when K=17. Further increases to K did not precipitate a higher accuracy value in our data which indicates that there is an upper bound for the optimal number of neighbors. Next, we tested this same model against true test data, the heart health dataset, where we first had to determine which values to give non-numeric data to calculate for our distance formula. For this, we swapped any text input to a numeric category, and simply checked to see if we had equal or unequal values. Then, for numeric data, we could compare these values against each other to see how close/far they are. In the end,

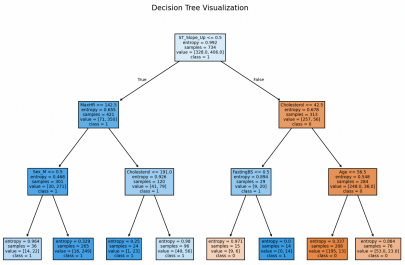
using this encoding, we found that the accuracy was initially around 41.85% when K=1, but did not initially grow with increases to K. However, then at K=7, the accuracy jumped to a value of 58.15%, remaining at this value for any further increases to K. Thus, our optimal K-value was 7, and our optimal accuracy was then reported to be 58.15%. Euclidean distance worked especially well for our categorical information which was converted into numerical data, as they only have a maximum distance of 2 as is, while also keeping track of the distance between our purely numerical data (Heart rate/ blood pressure).

Decision Trees

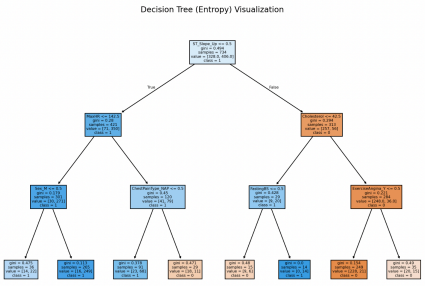
The chosen decision tree model was scikit-learn's DecisionTreeClassifier, selected for its ease of use in classification tasks. We chose not to pre-process the dataset for the decision tree model as the data provided was all relevant to the presence or absence of heart disease. This also gave us the opportunity to analyze the progression of the model. The dataset contains 734 patients all of whom have unique data and that fit into each classification. As a result, we compared outcomes of GINI index and entropy. We chose to use GINI index because it offers a slight edge in computational speed but more importantly produces comparably accurate, interpretable splits. On our chosen dataset, both Gini and entropy resulted in early splits on ST_Slope, MaxHR and Cholesterol which reduced node impurity early on, giving the model clear decision rules per node.

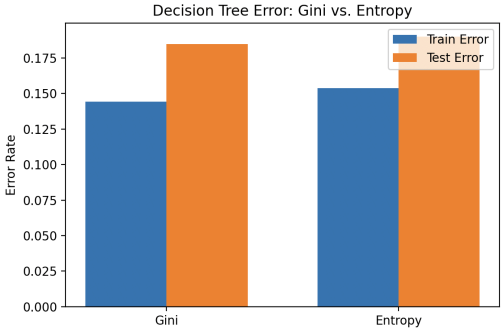
The graphs below present the results of these experiments and highlight the effects of each parameter on model accuracy.

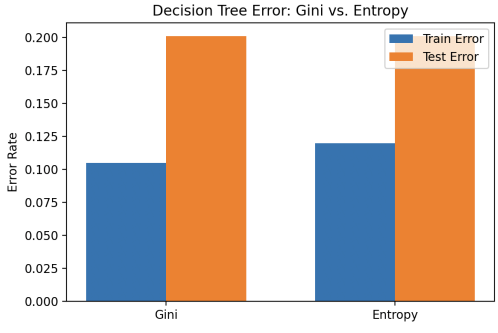
This decision tree was visualized off the GINI index.



This decision tree was visualized off the entropy.





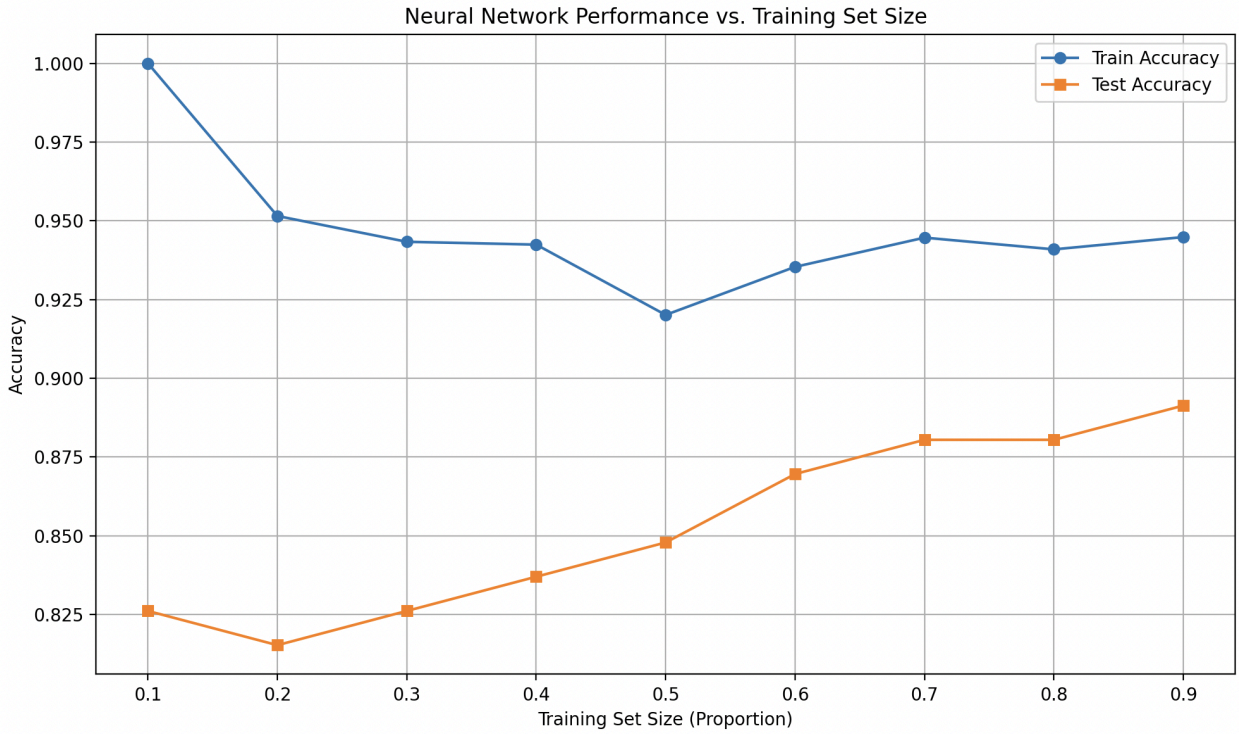


This is the visualization of the decision tree training error versus testing error. This training error versus testing error graph demonstrates that both trees operating on GINI and entropy progressed similarly. The training sample was about 80% of the data provided while the test sample was about 20% or 184 patients. This was done at a model depth of 3. However, when increased to 5 we noticed less error within the training error, however, the test error margin increased.

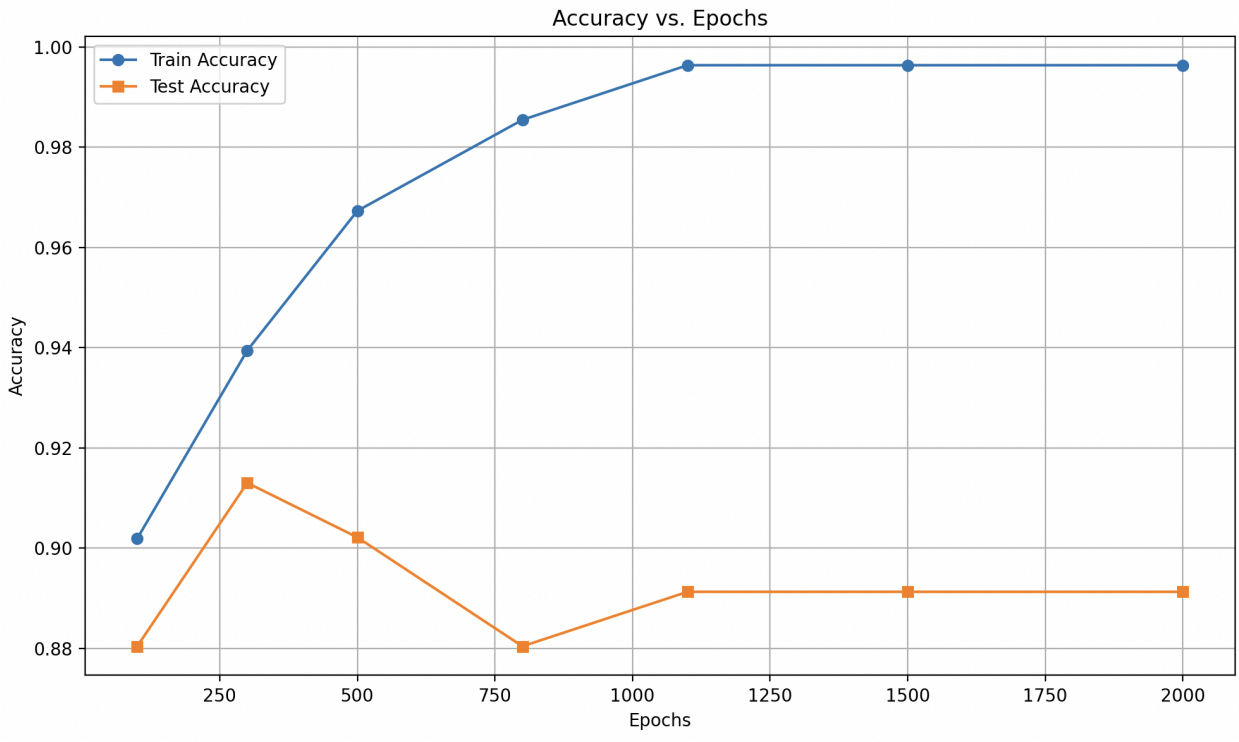
The accuracy of our decision tree was 81% with an F1-score of 80% and recall of 77%. We chose to use cross-validation because it allows us to generalize our output based on how many folds we choose to pursue, in this case 5. It also helps better control the complexity of our data and prevent overfitting.

Neural Networks

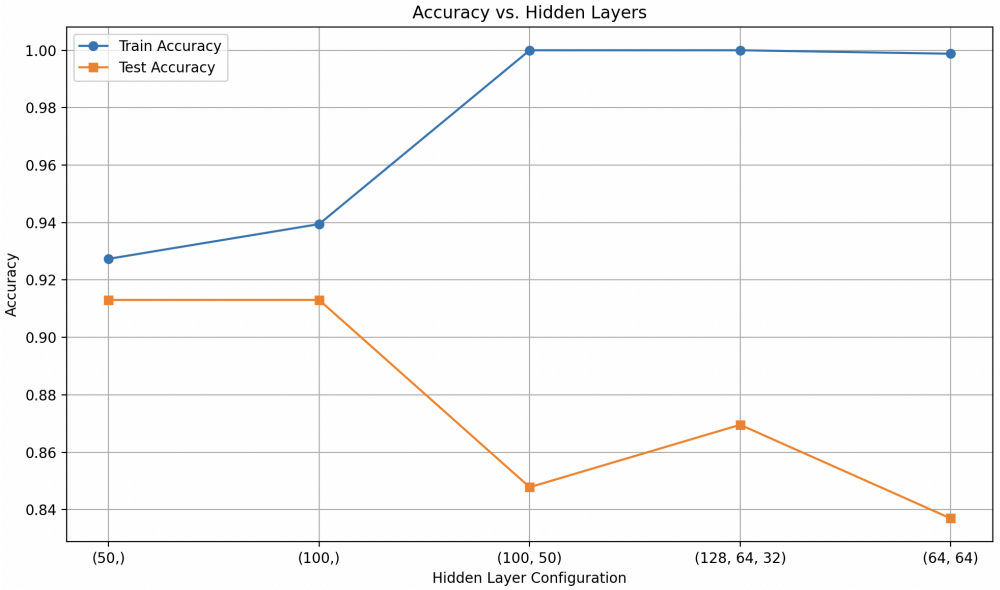
The chosen neural network model was scikit-learn's MLPClassifier, selected for its simplicity, accessibility, and flexibility in tuning hyperparameters such as training set size, number of training epochs, and hidden layer architecture. A range of configurations was tested to determine the optimal setup for achieving high generalization performance on the test data. The graphs below present the results of these experiments and highlight the effects of each parameter on model accuracy.



When evaluating training, and test accuracy based on training set size, the results were as expected. As the training set size increases, the training accuracy declines as the data becomes more diverse. Conversely, the test accuracy improved as the training set increased, demonstrating that the model is generalizing better with larger training samples, showing that more data reduces overfitting.



As the number of training epochs increased, both training and test accuracy initially improved; however, test accuracy eventually began to decline while training accuracy continued rising toward 100%. This behavior is an example of overfitting, where after a certain point, the model continues to optimize on the training data without corresponding gains on unseen data. Beyond 300 epochs, test performance stops improving despite the model continuing to learn, suggesting the model begins memorizing instead of generalizing.



As the network is made to be deeper or wider, training accuracy consistently increases, reaching 100% with more complex architectures. However, test accuracy peaks with a simple single-layer configuration. This trend indicates overfitting and suggests that, for this dataset, simpler architectures provide better generalization.

Top configuration

Hidden layer = 1 layer -> 100 neurons
Activation function = ReLU
Epochs = 300
Train Accuracy: 0.939
Test Accuracy: 0.913

To validate our model's generalization performance, we performed a 5-fold cross-validation on the top model configuration shown above. The mean accuracy across folds was 0.814 with a standard deviation of 0.037. This is slightly lower than the fixed test set accuracy of 0.913. This suggests that the hold-out test set may have been easier than average.

Determining the Best-Performing Algorithm Based on Accuracy

Based on the reported accuracies, the Neural Network performed best, achieving an optimal accuracy of 91.3%, followed by the Decision Tree at 81%, and K-Nearest Neighbor (KNN) at 58.15%.

In this context, "best" is defined by classification accuracy, which measures the proportion of correctly predicted instances out of all instances. Since all models were evaluated using the same metric, the Neural Network is considered the best-performing algorithm on this dataset. Its higher accuracy suggests it was more effective at capturing the underlying patterns and generalizing from the data, potentially due to its capacity to model complex relationships through multiple layers and weighted connections.