Project Report

Main Objective and Business Value:

The project's objective is to predict the presence of heart disease in patients. It is designed for use by medical professionals, primarily relying on the patient's medical data collected during checkup examinations conducted by a doctor. This tool would offer significant value to doctors in their clinical practice, as diagnosing heart disease is often a challenging and complex task, involving expensive additional tests and reports. Therefore, having such a tool to assist in identifying and forecasting patient diagnoses would be highly beneficial and lead to substantial cost savings.

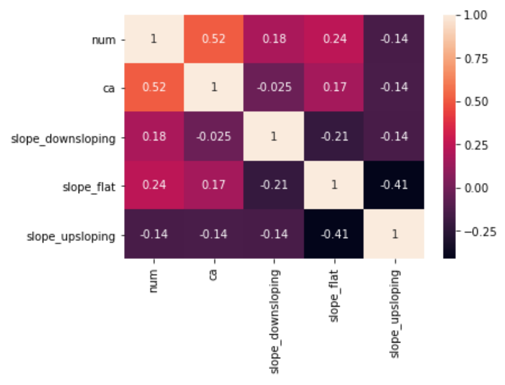
Data Pre-processing:

The dataset I am using is provided by UCI and contains multivariate data about patients and whether or not they have heart disease. You can find the dataset here. It consists of 920 instances with 16 features, including the presence or absence of heart disease. Many rows in this dataset contain missing data. Each row represents a different patient, and the labels of the rows include attributes such as age, resting blood pressure, cholesterol level, and more.

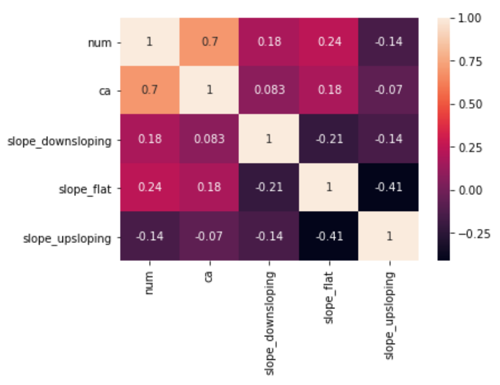
The data required several preprocessing steps. To begin, I removed the "id" and "datasets" columns as they do not provide meaningful information for the analysis and could introduce unnecessary noise into the data. Additionally, I excluded the "thal" column since 53% of its entries are NaN, indicating that this data may not always be available. Therefore, it is best to exclude it from our dataset. I also converted columns with values like True/False and Male/Female into binary results to make them compatible with the model.

Both the "Slope" and "Ca" columns have 35% of their data missing. To decide whether to keep or discard them, I evaluated their correlation with the outcome. I observed a strong positive correlation in the "Ca" column and a weak correlation in the "Slope" column. Consequently, it is advisable to retain them in the dataset. I filled the empty values with averages or modes based on the result we have for each stage of heart disease. This step significantly improved the accuracy of both models while maintaining the correlation between the columns and the data almost unchanged.

For the remaining data, I removed rows with missing entries and applied one-hot encoding to handle categorical data, preparing the data for the models' analysis.



Before Pre-processing the ca and Slope column



After Pre-processing the ca and Slope column

Machine Learning Model:

I decided to apply two classification models to my dataset: a decision tree and a random forest classifier. I used the implementations of these models provided by the scikit-learn library and ran them on test sets after splitting my data into training and test sets. Initially, I obtained very low test accuracies of 57% for the Random Forest Classifier (RFC) and 48% for the Decision Tree, while achieving very high training accuracies of 100% for the Decision Tree and 98% for RFC. This indicated overfitting in the data.

To address this issue, I investigated whether any of the previously dropped columns influenced the accuracy. I found that the "ca" and "slope" columns, which I had previously removed, were closely related to the result. Consequently, I added them back to the dataset, as explained in the previous section. This resulted in a substantial improvement in the test accuracies for both models, while maintaining the training accuracies at the same high level. I ended up with test accuracies of about 75% for both models.

Another problem I encountered was excessive overfitting of the models to the extent of achieving 100% accuracy on the training set. To mitigate this, I decided to perform 5-fold cross-validation to tune the hyperparameters for both models. This led to a decrease in the training accuracy to around 80-85% and an improvement in the test accuracy to approximately 80-83% for the Decision Tree and 82-87% for the Random Forest.

The final step I took was an attempt to reduce the variance in the data through 5-model bagging. I did this to achieve the best possible test accuracy. In the Decision Tree, bagging decreased the accuracy because splitting the training data into 5 sets and training each tree on a different set may have caused the importance of some of the more critical features to decrease. Consequently, each tree made decisions more influenced by the noise in the data rather than the important features, resulting in lower average accuracy.

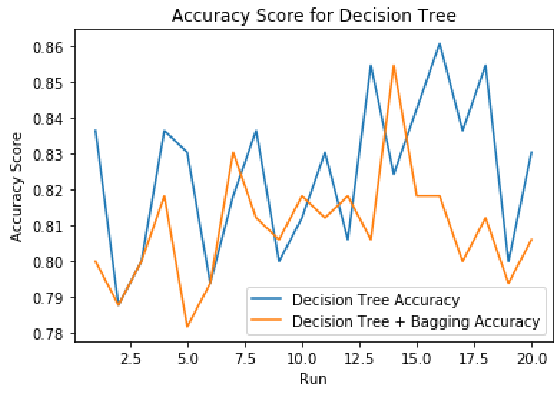
For the Random Forest, bagging improved the test accuracy because Random Forests are already adept at handling noisy features. Thus, adding the additional source of randomness from bagging's data splitting and getting results from 5 different forests actually improved the accuracy, as the variance in the data decreased due to the bagging process.

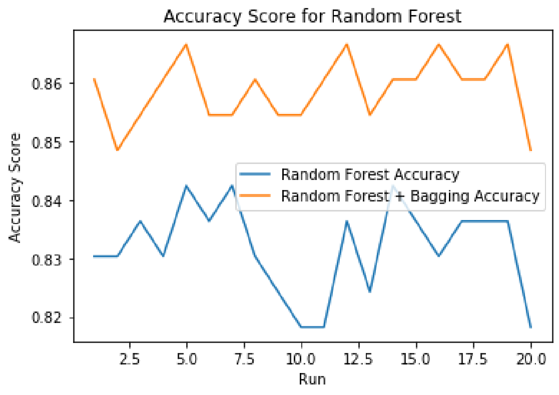
Preliminary Results:

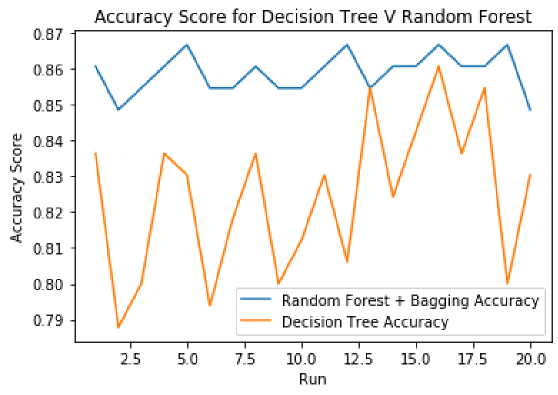
The metric I am using to measure performance is accuracy. From the first graph, which displays the performance of the decision tree, it is evident that, on average over 20 different runs, the decision tree without bagging achieved better results (82.5% on average) compared to the decision tree with bagging (80.9% on average).

In the second graph, which shows the performance of the random forest classifier, it is clear that, on average over 20 runs, the random forest with bagging outperformed (85.9% on average) the random forest without bagging (83.2% on average).

Finally, when comparing the 20 runs of the random forest with bagging to the decision tree without bagging, it is apparent that the random forest with bagging (85.9% on average) significantly outperformed the decision tree without bagging (82.5% on average). Therefore, I have decided to use predictions from the random forest with bagging for the final web application.







Next Steps:

There are several pros and cons to the tree models I chose to use in this project. One advantage to consider is that these trees make predictions based on trends in certain feature values. I believe that one reason this project achieved high accuracy is because of this characteristic. In clinical practice, doctors use benchmarks and specific indicators in a patient's report to determine whether the patient has heart disease or not. The tree models perform a similar job by analyzing the data to reach their predictions.

However, a disadvantage is that due to the random nature of the trees, decision trees and random forests may yield different results in each run, leading to variations in accuracy when using the same test data. The stochastic nature of these models can make it somewhat challenging to interpret and compare their performance consistently.

For future work, I would recommend exploring a neural network or deep learning approach to see how the results compare with this type of data. Comparing the performance of tree models against neural networks could be an interesting direction to take, especially with this dataset. I do not plan to make any further alterations to the model, as it already incorporates a sufficient level of complexity and is performing very well, as evident from the results.