Mohamed Elenany

260892581

Deliverable 2

1. Problem Statement:

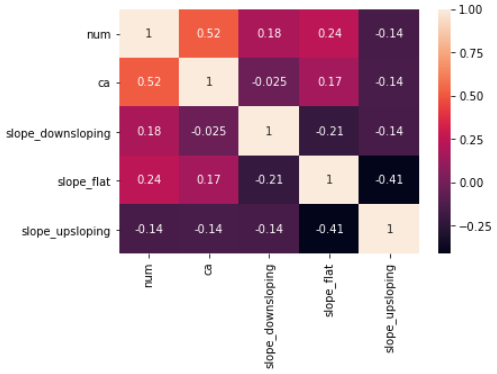
   I previously had a problem statement to predict the USD exchange rate of any country, however the results received were not good enough and better results were not feasible. As a result, my new problem statement is basically a classification problem to determine whether a patient has heart disease or not, and if yes, which stage it is (1,2,3,4).

2. Data Pre-processing:

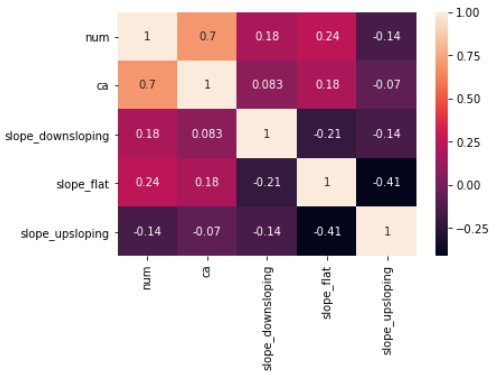
   The dataset I am using is a dataset offered from UCI that have multivariate data about patients and whether or not they have heart disease or not; the dataset can be found here. The dataset has 920 instances with 16 features (including the result of heart disease or not) where multiple rows have missing data. Each row represents a different patient with the labels of the rows including labels like age, resting blood pressure, cholesterol level, etc.

   The data had a lot of pre-processing steps to be take. I start by dropping the id and datasets columns as they are not beneficial to the result and will just add un-needed noise to the data if add. I will also remove the thal column as 53% of entries in this column are NaN, which means that having this data may not always be possible, thus it is best to remove it from our dataset. I also convert columns that have data such as True/false and Male/female to binary results to be able to be read by the model.

   The Slope and Ca columns both have 35% of their data to be missing, so I evaluate the correlation they have with the result to see if it beneficial to keep them or discard them. I recognize that the Ca column has a strong positive correlation, and the slope column has a weak correlation, thus it is best to keep them, and I fill the empty values with averages/modes of the result we have for each stage of heart disease. The result of this step is substantially improving the accuracy results for both models while keeping the correlation between the columns and the data the nearly similar. For the rest of the data, I discard all rows that have an empty entry and I one-hot-encoding of categorical data to get the data ready to be read by the models.

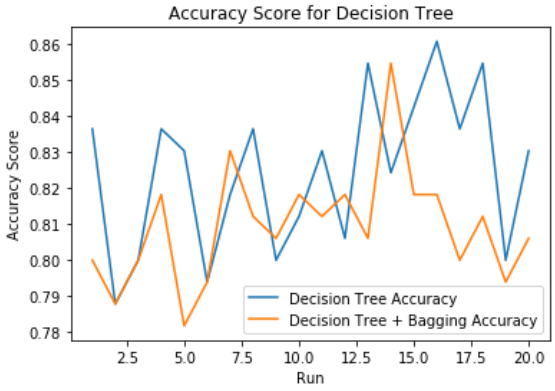


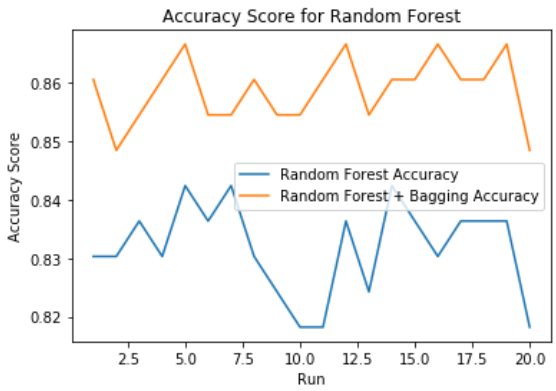
Before Pre-processing the ca and Slope column

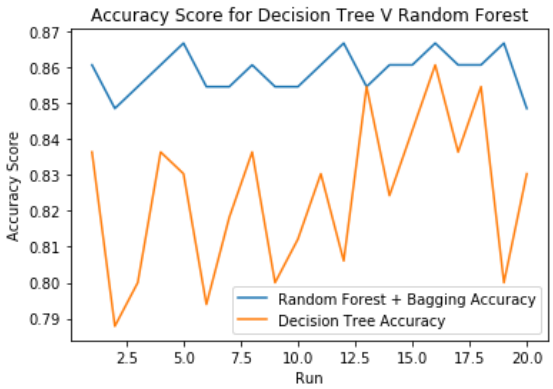


After Pre-processing the ca and Slope column

3. Machine Learning Model:
   I decided to try two classification models to apply on my dataset, and they are a decision tree and a random forest classifier. I used the implementations of those models offered in the sklearn library, and I ran the models on test sets after splitting my data into test and train sets. At the beginning, I received very low-test accuracies of 57% (RFC) and 48% (Decision Tree) while receiving very high train accuracy of 100% (decision tree) and 98%(RFC). Thus, I figured out that the data was overfitting; I approached this issue by trying to see if any of the dropped columns influenced the accuracy shown, and I found that in fact, the ca and slope columns that I previously dropped were very related to the result. Thus, I added them back to the data as explained in the previous section. As a result, this substantially improved the test accuracies for both while keeping the training accuracies the same. I ended up with accuracies for the test set of about 75% for both models.
   Another Problem that I faced was that the models were overfitting the data too much to the extent of getting 100% accuracy on training set. To solve this, I decided to conduct 5-fold-cross-validation in order to tune the hyperparameters for both models. This resulted in decreasing the train accuracy to about 80-85% and improving the test accuracy to about 80-83% for decision tree and 82-87% for random forests. The final step I took was attempting to decrease the variance in the data through 5-model bagging, and I took this step to try to get the best possible test accuracy. In the decision tree, bagging decreased the accuracy as 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 important features to decrease. Thus, each tree would make a decision that is more based on the noise of the data and not the important features, which resulted in lower accuracy on average. On the random forest, bagging resulted in improving the test accuracy as the random forests already deal well with noisy features, thus adding this source of randomness from the bagging's splitting of dataset and getting the result from 5 different forests actually improved the result received (as the variance of the data has now fell down due to the bagging process).
4. Preliminary Results:
   The metric that I am using to measure performance is an accuracy metric. From the first graph that is showing performance of decision tree, it is clear to see that on average of 20 different runs, the decision tree without bagging model (82.5% on average) performed better than that with bagging (80.9% on average). From the second graph showing performance of the random forest classifier, it is clear to see that on average of 20 runs, the random forest with bagging (85.9% on average) performed better than that with bagging (83.2% on average). Finally, looking at the 20 runs from random forest with bagging v decision tree without bagging, it is clear to see that the random forest with bagging (85.9% on average) did a lot better than the decision tree without bagging (82.5% on average). Thus, I have decided that I will be using predictions from the random forest with bagging for the final web application.







5. Next Steps:

There are many pros and cons to the tree models that I chose to use in this project. An advantage to think about is that the trees make predictions based on trends in certain feature values, and I think that one reason this project has high accuracy is because of this. Doctors use benchmarks and certain flags in a patient's report to determine if a patient has a heart disease or not, and the trees do a very similar job to reach its prediction. However, a disadvantage is that due to the random nature of the trees, the decision trees/random forest may look differently at each run, thus producing different accuracy with the same test data each time. The stochastic nature of trees can make it a bit tedious to interpret which model is working better.

For future work, I would recommend trying out a neural network or deep learning approach to see how the results would come out for this type of data. I believe that comparing the performance of trees against neural networks will be an interesting place to go next, especially with this dataset. I will not be making any further alterations to the model as enough complexity has been added to the model and the model is performing very well as seen from the results.