CS 4372

ASSIGNMENT 2 - DATA ANALYSIS REPORT

Names of students in your group:

Scott Vu - SMV210000

Philip Wallis - PTW190000

Number of free late days used: 1. ¾, ¾ Remaining

Note: You are allowed a **total** of 4 free late days for the **entire semester**. You can use at most 2 for each assignment. After that, there will be a penalty of 10% for each late day.

Please list clearly all the sources/references that you have used in this assignment.

Dataset Used:
Heart Disease
https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset/data

Libraries used:
pandas, matplotlib, seaborn, sklearn, xgboost, graphviz

https://pandas.pydata.org/docs/reference/index.html#api

https://matplotlib.org/stable/api/index.html

https://seaborn.pydata.org/api.html

https://scikit-learn.org/stable/modules/classes.html

https://xgboost.readthedocs.io/en/stable/python/python_api.html

https://graphviz.readthedocs.io/en/stable/api.html

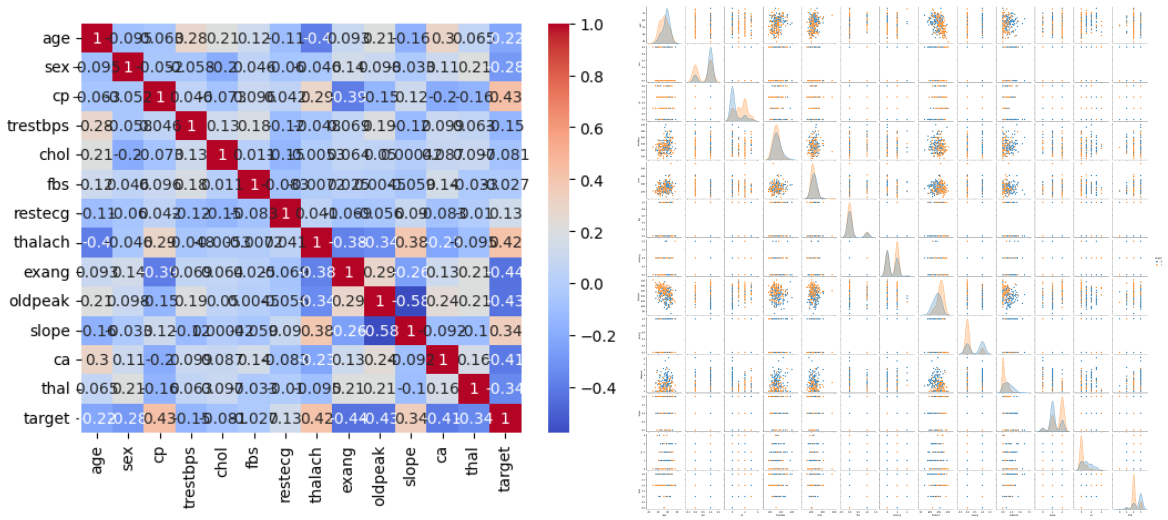
Introduction

In this assignment, we decided to use the *Heart Disease Dataset* from Kaggle (https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset/data), stored at https://personal.utdallas.edu/~smv210000/Heart%20Disease%20Dataset/heart.csv. We chose the dataset because it provides a categorization type task with various attributes. The heart disease dataset being widely used hints that it is a good dataset to use for categorical analysis. Using the GridSearchCV library to hyper-tune and the SKLearn/XGBoost library for model creation, we implemented a plain Decision Tree Classifier, Random Forest Classifier, Adaboost Classifier, and XGBoost Classifier. Using graphviz, we generated a tree image of what each classification stage looks like. By creating a ROC curve and Precision Recall Curve, as well as a classification report and a confusion matrix, we were able to better analyze each classifier and conclude which of the four classifiers works best with our data.

Preparing the Data

Before creating the models, we cleaned the data. Although this dataset is known to be cleaned, we made sure to redundantly drop null values and duplicate rows. We believed that every column was important, so there were no columns dropped.

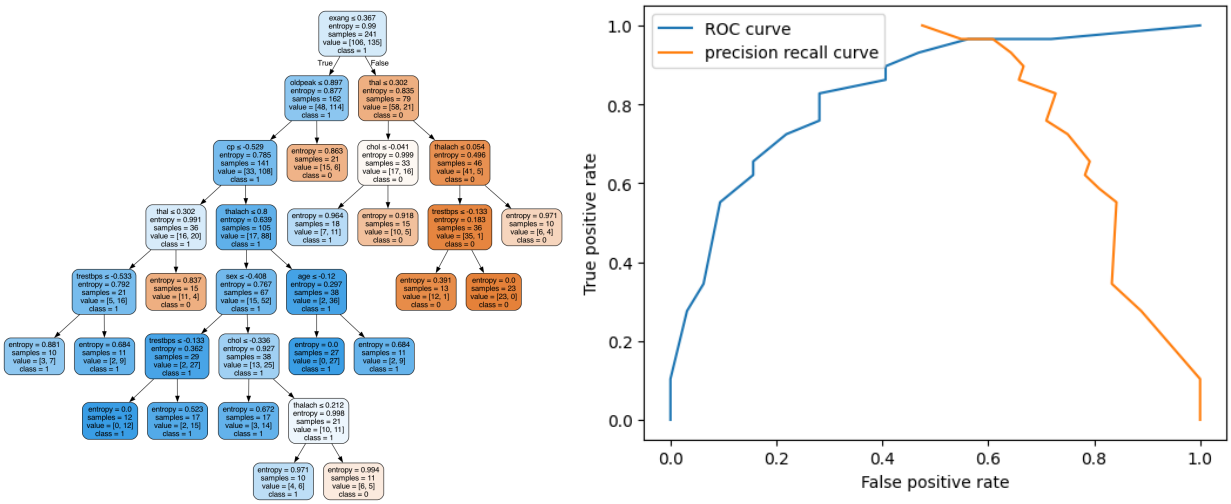
Using seaborn's heatmap function, we obtained a visualization of the heatmap of every attribute. At the very bottom row, you can see the "target" attribute, which is what we will be focusing on classifying.
Also, using the pairplot with the hue as the "target", we can see the difference between data points with heart disease and ones without.



Since "target" is our predicted variable, we extracted that value from the data as y, and kept the rest as independent variables.
Using SKLearn's function train_test_split, the datasets were split into a training set and a testing set. Using the test size of 0.2, we use 80% of the data for training and 20% of the data for testing. Also, the variables were scaled using sklearn's standard scaler.

Plain Decision Tree Classifier

To train various models to obtain the best-performing one, we fine-tuned the hyper-parameters in the creation of the classifier using the GridSearchCV library. The criterion, max depth, minimum samples split and leaf, maximum features and nodes were used in the parameter grid to create the model. Our definition of best model is the model with the highest Accuracy and our model had 82.16% accuracy in predicting the target.



The tree on the left shows the plain decision tree that was generated using the model, and the right graph shows the ROC curve and the precision recall curve. We can see that at around 0.6 false positive rate is where the two curves meet, which means at that point, both curves agree on the model's performance characteristics.

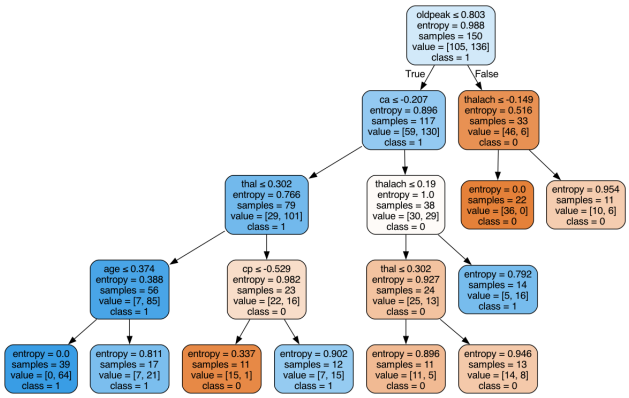
	precision	recall	f1-score	support
0	0.77	0.72	0.74	32
1	0.71	0.76	0.73	29
accuracy			0.74	61
macro avg	0.74	0.74	0.74	61
weighted avg	0.74	0.74	0.74	61

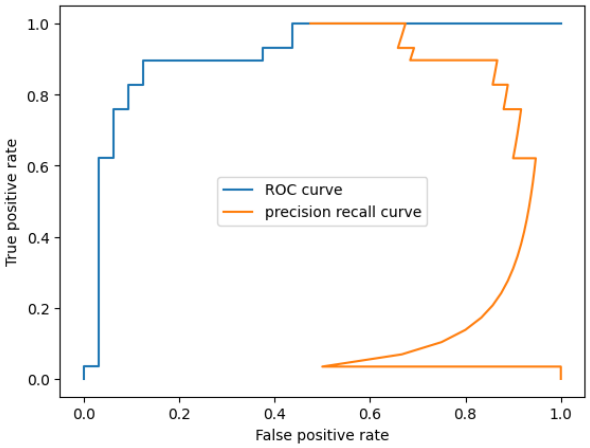
```
[[23  9]
 [ 7 22]]
```

The precision, recall, and f1 score all being about 0.7 lets us know that our model performs reasonably well, but due to the nature of the data, which is for medical use, these scores may be considered unusable.

Random Forest Classifier

For our Random Forest Classifier model, we utilized hyperparameter tuning to optimize various settings such as n estimators, criterion, max depth, features, nodes, and min samples split and nodes. This involved a significantly higher computational cost, taking approximately 90 minutes to train the model fully. Despite the computational expense, the Random Forest Classifier yielded the highest accuracy among our models at 86.31%.





This specific classifier model gave us the best accuracy at 86.31%, outperforming the other three models. The spike in the precision recall curve is due to the number of data we have, where outliers can be shown.

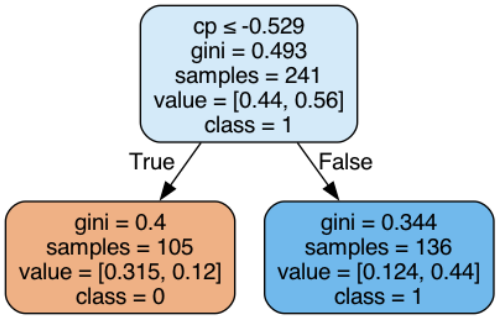
	precision	recall	f1-score	support
0	0.85	0.88	0.86	32
1	0.86	0.83	0.84	29
accuracy			0.85	61
macro avg	0.85	0.85	0.85	61
weighted avg	0.85	0.85	0.85	61

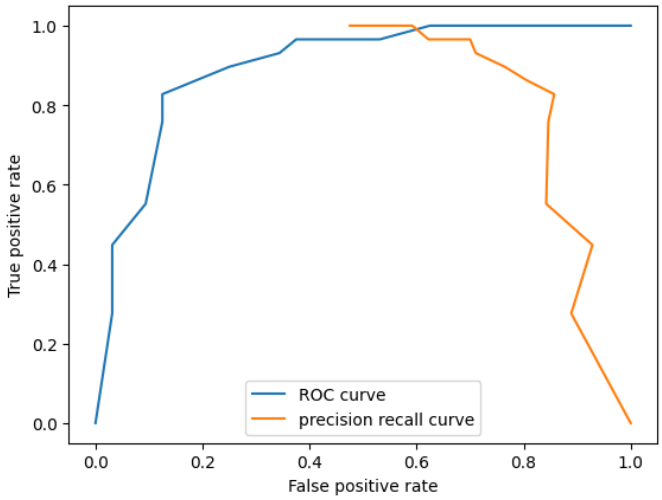
```
[[28  4]
 [ 5 24]]
```

The strength of the Random Forest model lies in its ability to manage high dimensionality and its inherent mechanism for feature selection. Unlike the Plain Decision Tree model, Random Forest lessens the impact of overfitting by averaging the results of numerous individual decision trees.

Adaboost Classifier

The Adaboost Classifier was another ensemble method we tried. By tuning the learning_rate and n_estimators parameters via GridSearchCV, we managed to achieve an accuracy of 82.99%. This was better than the Plain Decision Tree Classifier despite utilizing fewer hyperparameters.





Adaboost operates by focusing on instances that are difficult to classify, thereby incrementally improving the model's performance. However, it is worth mentioning that the model can be sensitive to noisy data and outliers, a limitation to consider depending on the application.

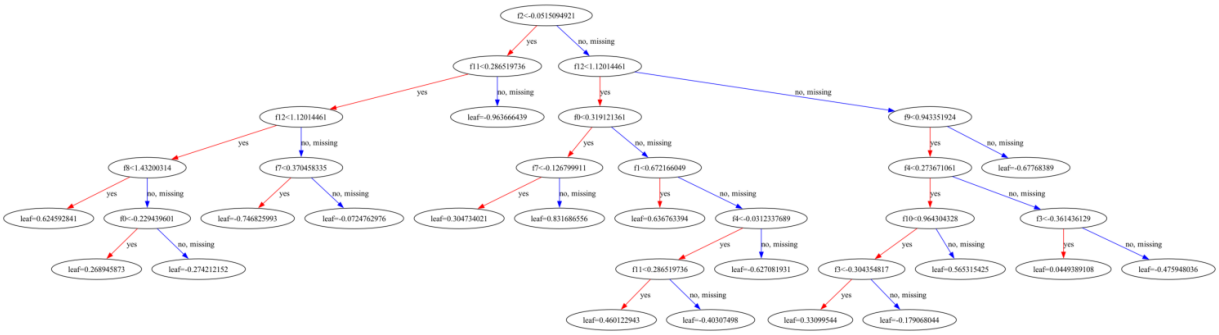
	precision	recall	f1-score	support
0	0.87	0.81	0.84	32
1	0.81	0.86	0.83	29
accuracy			0.84	61
macro avg	0.84	0.84	0.84	61
weighted avg	0.84	0.84	0.84	61

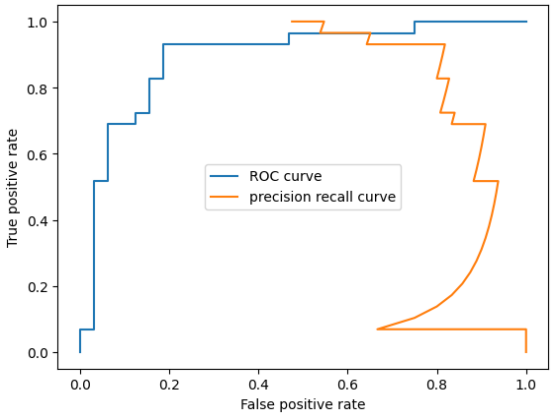
```
[[26  6]
 [ 4 25]]
```

In terms of performance metrics, the Adaboost model showcased strong precision, recall, and F1-scores across both classes. On average, we get values from 0.81 to 0.87. These values indicate that the model is robust, and the high f1-score tells us that the precision and recall is balanced in the model.

XGBoost Classifier

We also implemented the XGBoost Classifier, a gradient-boosted decision tree model. By tuning the learning_rate and n_estimators parameters via GridSearchCV, we managed to achieve an accuracy of 82.15%, which was comparable to the Adaboost Classifier. XGBoost offers excellent speed and performance; its importance lies in its ability to run efficiently on distributed systems and its flexibility in solving a wide range of machine learning problems.





	precision	recall	f1-score	support
0	0.78	0.88	0.82	32
1	0.84	0.72	0.78	29
accuracy			0.80	61
macro avg	0.81	0.80	0.80	61
weighted avg	0.81	0.80	0.80	61

```
[[28  4]
 [ 8 21]]
```

Breaking down the performance metrics, we observed that the XGBoost model showed a good balance between precision and recall. Although showing lower scores than adaboost, it shows that the model still performs moderately with scores of around 0.72 to 0.82.

Comparing the Models

Upon evaluation, the Random Forest Classifier emerged as the best-performing model in terms of accuracy. However, it's essential to note that while accuracy is an important metric, it may not be the only factor to consider, especially in the context of medical datasets where False Negatives and False Positives could have serious implications.

The Adaboost and XGBoost classifiers, despite their lower accuracy, offer advantages in terms of computational efficiency and are less prone to overfitting compared to a plain decision tree. Thus, the choice of model should be tailored to the specific requirements and constraints of your application.

Conclusion

Our exploration of various machine learning models to predict heart disease provided valuable insights into the strengths and limitations of each. While ensemble methods like Random Forest offered the highest accuracy, other models like Adaboost and XGBoost also demonstrated strong performance metrics and have their own sets of advantages, including speed and resistance to overfitting.

The project underscored the necessity of understanding the trade-offs between different models, especially when dealing with critical applications such as medical diagnoses. Fine-tuning hyperparameters played a significant role in optimizing our models, reinforcing the idea that machine learning is as much an art as it is a science.