EE244 Final Project Report - Heart Disease Analysis and Prediction

Joshua Chen
861257903

Introduction:

With increasingly unaffordable healthcare, obesity rates at a record high, and physical health on the decline, Americans are at risk of a multitude of health complications, and they may not be aware. This project aims to determine important/unimportant attributes and a reliable classifier (acc ~ 95%) to identify those individuals at risk of heart complications. Such an application can help shed some light on the general public about their own health and if they should start taking their lifestyles more seriously.

This project explores the Heart Disease Data Set from the UCI Machine Learning Repository, and uses various machine learning models to predict an individual's risk of having heart disease. The purpose of using several models is for comparison to determine which prediction model is best suited for data sets like this one. Besides comparing the different models, hyperparameters are also tuned to determine their effects on their respective classifiers. In total there are three experimental runs: benchmark, omitted features, and tuned hyperparameters. For the benchmark run, all models are tested with their default hyperparameters. In omitted features, the least significant features determined from the benchmark run are removed and the models are tested again, but with the default hyperparameters. The final run involves using all features, but with each model's hyperparameters tuned to observe the effects on classification performance.

Related Work:

The experimental runs for this project involve the use of six sklearn classifiers. The sklearn classifiers I used are: Random forest, AdaBoost, Decision Tree, Logistic Regression, XGBoost, and KNN. Each one has its advantages and limitations depending on the nature of the data set. For logistic regression in particular, it is limited in its performance when the data set is too small or when some features provide overlapping information [1]. Both of those undesirable situations are applicable here for this data set since there are only about 300 data samples, and some of the 13 features do share similarities. It is hoped that the omitted features run will improve this classifier's performance as that run intends to drop some features.

From my own prior experience working with random forest, I have high hopes for this classifier since it, like other ensemble methods, has high generalization capabilities due to bagging and random feature selection [2]. This means that this classifier can give high classification performance with less memory (trees) required, and is robust to noise [3]. Additionally, this model is claimed to be just as reliable, or better than the AdaBoost model. The only main disadvantage of this classifier isn't applicable for this project, and that is, it can't handle dynamic data well [4].

The XGBoost classifier is also another ensemble model with recognition. Mostly praised for its scalability, performance, and popularity for machine learning competitions [5]. It differs from AdaBoost and is a massive improvement since it directly uses each decision tree's residual error to influence the next decision tree instead of going through the entire process of updating weights. Generally, it is much more efficient, flexible, and accurate than AdaBoost.

Technical Approach:

The first phase of this project is a preliminary data analysis to help me become more familiar with the data and analyze if there is any need to preprocess the data file before using it for machine learning applications. Shown below in Figure [1] are the data properties of the heart.csv file. All data types are numeric (int64 and float64), so there is no need to perform any label encoding. Next, it was determined that there were no missing values in the data set, so I was not required to do any further preprocessing of the data file.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Figure 1: heart.csv Properties

With the data file already ready for use, the next step in my approach was to perform the benchmark experimental run, where I tested out six sklearn classifier models with their default hyperparameters. The test accuracy results of this run serve as a datum to which I can compare the two modified experimental runs. Just after the benchmark run, I decided to extract the feature importance for the ensemble models to determine which features are the most and least influential in each model's decision making process. Once the least influential features were determined for each or all models, those features were then removed from the data set. This modified data set with only the "important features" was then passed to the second experimental run: omitted features.

In the omitted features experimental run, all of the classifiers' hyperparameters were kept in their default states, and the input data included only the more important features. The least influential factors that were removed from the data set are: fbs, restecg, and sex. Fbs stands for "Fasting blood sugar", restecg stands for "Resting electrocardiographic results", and sex is represented as either male or female.

In the final experimental run, the classifiers' hyperparameters are all tuned and each model is tested with a combination of different hyperparameter settings to observe the effects on classification performance. More details of each run and the feature importance analysis can be found in the next section: Experimental Results.

Experimental Results:

The results of the experimental runs were all quite decent. The general accuracy range of the models was about 80-90%, with KNN being below this metric and XGBoost being above. In all three experimental runs, XGBoost was consistently the best and most accurate classifier, reaching an accuracy metric of 93.4%.

Figure [2] below shows the results of the benchmark run. Here, all models were tested with their default hyperparameters. XGBoost has already reached its maximum value, with the random forest classifier not far behind. KNN is tested with 5 neighbors here with uniform weights, so its performance is low, not surprisingly.

	Model	Accuracy	CVS	Recall	Precision	F1
0	Random Forest Classifier	89.108911	79.609610	92.121212	88.372093	90.207715
1	AdaBoost Classifier	84.818482	78.468468	87.272727	85.207101	86.227545
2	Decision Tree Classifier	88.118812	72.327327	86.666667	91.082803	88.819876
3	Logistic Regression	85.808581	77.387387	90.909091	84.269663	87.463557
4	XGB Classifier	93.399340	79.009009	95.151515	92.899408	94.011976
5	KNN Classifier	71.947195	60.750751	74.545455	74.096386	74.320242

Figure 2: Benchmark Results

Now that the benchmark run has provided all necessary metrics and results, the next step is feature importance extraction. The following Figures [3-6] show the relative feature importance for every attribute for the ensemble models: Random forest, AdaBoost, Decision Tree, and XGBoost. Generally, the most important features are: cp, thal, chol, thalach, and ca. Respectively, these are chest pain type, Thal, serum cholesterol in mg/dl, maximum heart rate achieved, and number of major vessels (0-3) colored by fluoroscopy. This makes perfect sense since all of these attributes are directly related to cardiovascular health. Higher heart rate, more cholesterol, and existence of chest pain are indicators of heart complications and related issues. The least important features were determined to be: fbs, restecg, and sex. Respectively, these are

fasting blood sugar > 120 mg/dl, resting electrocardiographic results, and biological gender. Heart disease does not discriminate on the basis of sex, so it's understandable why that is unimportant in predicting heart disease. Overnight fasting blood sugar is not important in determining heart disease, but it is important in analyzing potential diabetics. What is interesting to note is that resting ECG is considered unimportant in the prediction, although it is directly related to detecting heart abnormalities such as irregular heart rate. Despite this, all three of these features are dropped from the data set to create a modified data set for use in the omitted features experimental run.

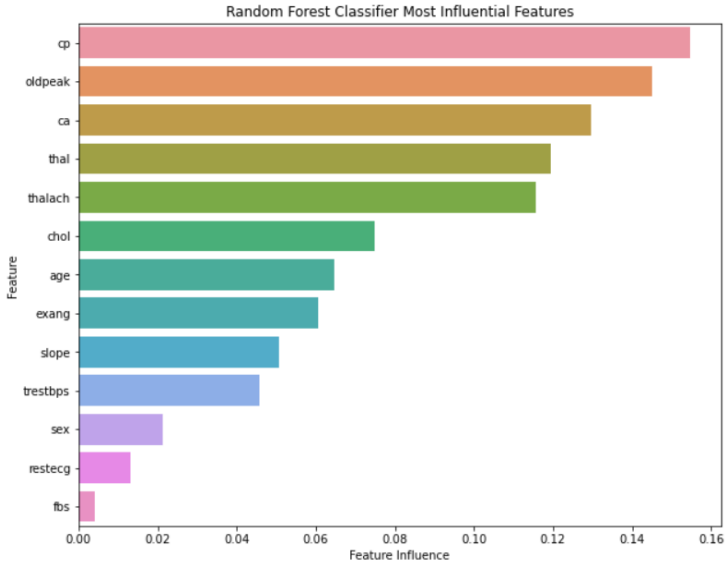

Figure 3: Feature Importance for Random Forest

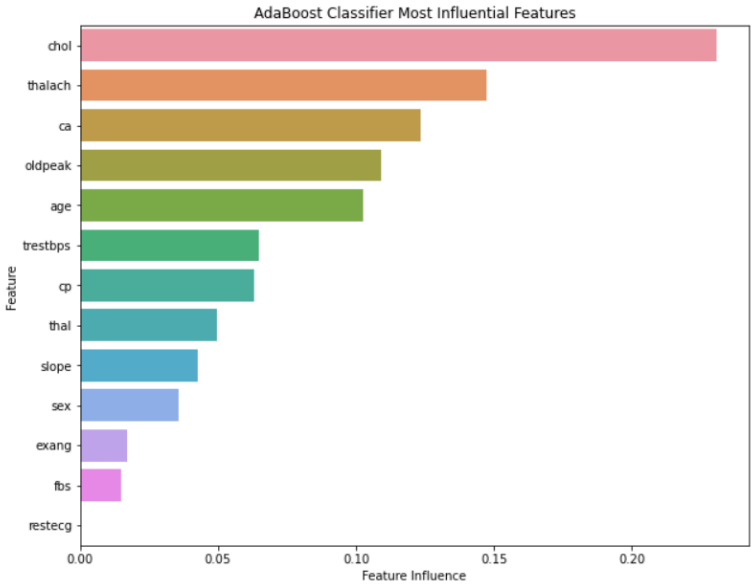

Figure 4: Feature Importance for AdaBoost

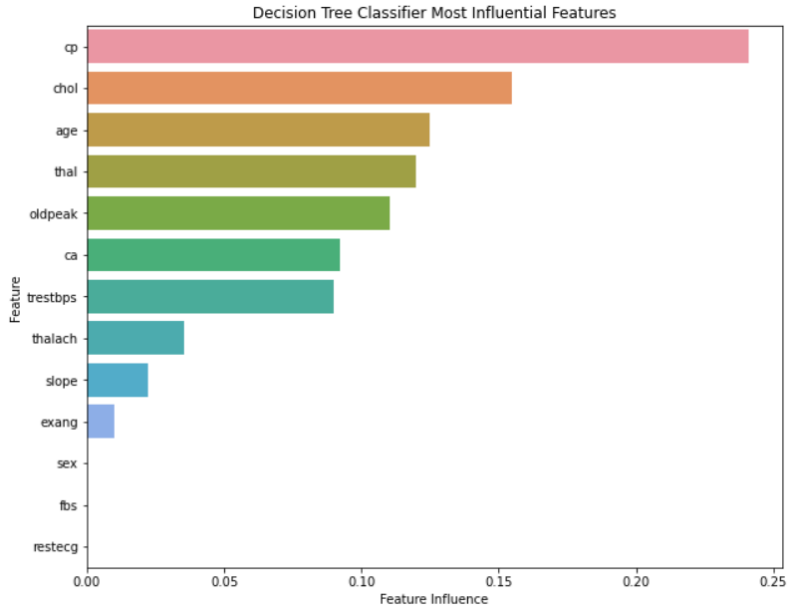

Figure 5: Feature Importance for Decision Tree

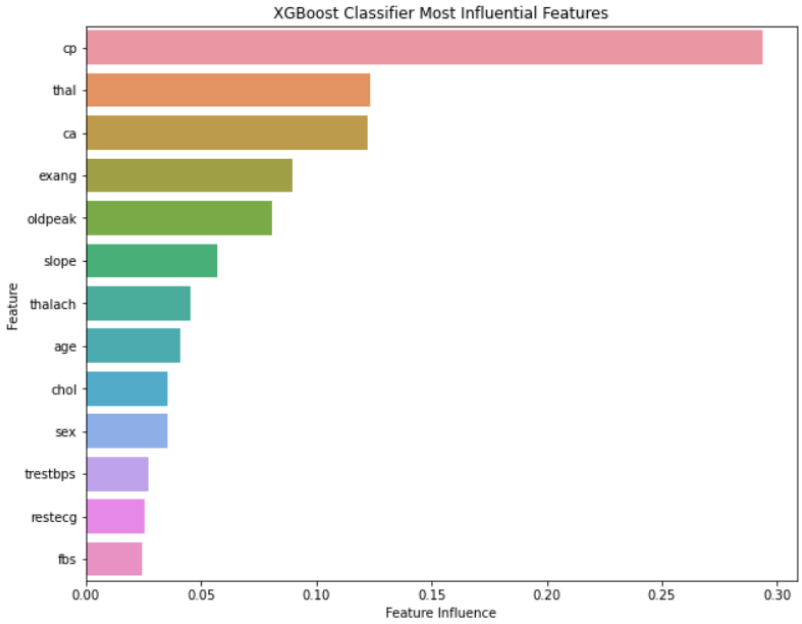

Figure 6: Feature Importance for XGBoost

Shown in Figure [7] are the test accuracy results for the omitted features run. Out of all models, only the AdaBoost has shown any improvement. KNN stayed the same, while the remaining four models suffered lower metrics than the benchmark. This signifies that although the less important features shouldn't be considered heavily, they shouldn't be completely ignored since they may carry some sensitive information that may help a classifier. Minimal correlation

or importance does not mean no correlation. It was at first surprising to me that all three of the dropped features were the least important for the Decision Tree classifier, yet the model only worsened.

	Model	Accuracy	CVS	Recall	Precision	F1
0	Random Forest Classifier	87.788779	79.564565	93.333333	85.555556	89.275362
1	AdaBoost Classifier	85.148515	80.135135	85.454545	87.037037	86.238532
2	Decision Tree Classifier	86.468647	72.312312	86.666667	88.271605	87.461774
3	Logistic Regression	84.818482	77.387387	90.303030	83.240223	86.627907
4	XGB Classifier	92.739274	76.786787	93.939394	92.814371	93.373494
5	KNN Classifier	71.947195	60.750751	74.545455	74.096386	74.320242

Figure 7: Omitted Features Experimental Results

	Model	Accuracy	CVS	Recall	Precision	F1
0	Random Forest Classifier	87.128713	79.609610	91.515152	85.795455	88.563050
1	AdaBoost Classifier	86.468647	76.246246	87.878788	87.349398	87.613293
2	Decision Tree Classifier	84.818482	69.579580	83.030303	88.387097	85.625000
3	Logistic Regression	85.148515	77.957958	90.303030	83.707865	86.880466
4	XGB Classifier	93.069307	79.549550	95.151515	92.352941	93.731343
5	KNN Classifier	75.247525	59.684685	73.939394	79.220779	76.489028

Figure 8: Tuned Hyperparameters Experimental Results

Figure [8] above shows the test accuracy results of the tuned hyperparameters experimental run. The only two models that saw improvement through changing hyperparameters were AdaBoost and KNN. The changes I made for these two were decreasing the number of trees from 75 to 50 for AdaBoost and decreasing the number of neighbors from 5 to 3 for KNN.

The logistic regression, random forest, and XGBoost all became slightly worse. For the two ensemble models, I decreased the number of trees substantially. The result of this action makes sense since with fewer trees, either model will not be able to train completely to the optimal extent. For logistic regression, I changed the norm criterion from L2 to L1 norm, so the accuracy change is minor but worse nonetheless. L1 is not true distance, so this result also makes sense.

The model that suffered worse from the tuning was the decision tree classifier. The only change was the information gain split criterion going from gini to entropy. The main difference between the gini impurity and entropy criterion is the computational cost. Generally, the gini is faster and more efficient because it has less complex computations than the entropy criterion.

Additionally, the gini impurity range is [0,0.5] while the entropy range is [0,1]. This means that the gini impurity criterion is better than the entropy criterion at selecting the best features, which explains why the decision tree classifier's performance worsened substantially.

Conclusions:

The most important takeaway from this project for me was the results of the omitted features run. I initially thought that by omitting unimportant features, the performance of all models would increase. That was not the case since even lesser features carry some useful or sensitive information for the classifiers to perform correctly. For the tuning hyperparameters run, all results are as expected with how I manipulated the combinations of each classifier's hyperparameters. Overall, the best models for classifying the heart disease data set are XGBoost and Random Forest, as both of these models have F1 scores exceeding 90, and metrics closest to the target accuracy metric of ~95%. The primary reason for these two is because both models have high generalizability and for XGBoost, scalability. Both models are adapted to handle small data sets that more conventional models like logistic regression will suffer with.

Acknowledgements:
Link to Dataset on UCI Machine Learning Repo:
https://archive.ics.uci.edu/ml/datasets/Heart+Disease

References:

[1] T. Juliana, M. William J (2016). "Logistic Regression Relating Patient Characteristics to Outcomes". JAMA. 316 (5): 533–4. doi:10.1001/jama.2016.7653. ISSN 0098-7484. OCLC 6823603312. PMID 27483067.

[2] Y. Mishina, M. Tsuchiya & H. Fujiyoshi, "Boosted random forests," 2014 International Conference on Computer Vision Theory and Applications (VISAPP), 2014, pp. 594-598.

[3] Breiman, L. (2001). "Random forests," *Machine learning*. Springer.

[4] A. Saffari, C. Leistner, J. Santner, M. Godec & H. Bischof, "On-line Random Forests," 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, 2009, pp.1393-1400, doi: 10.1109/ICCVW.2009.5457447.

[5] T. Chen, C. Guestrin, "XGBoost: A Scalable Tree Boosting System," 22nd ACM SIGKDD International Conference, 2016, doi: 10.1145/2939672.2939785.