Kevin Egocheaga

Final Report: Predicting Cardiovascular Disease

Problem Statement

Cardiovascular disease (CVD) is an umbrella term referring to conditions that affect the heart or blood vessels. CVD is the leading cause of death in both men and women and people of most racial (and ethnic) groups in the United States. It is estimated that one person dies every 34 seconds in the United States from CVD. Furthermore, CVD accounts for over a third of all deaths worldwide.

The purpose of this project was to create a model that predicts CVD given certain features. The nature of the problem is a difficult one, so performance(s) of the model were expected to be moderately accurate at best.

I kept feature engineering to a minimal. I did not add any additional variables to the analysis. However, I did hyperparameter tuning on all classification models used.

Data Wrangling

The original dataset consisted of 69,301 rows and 13 columns and with a mixture of continuous and discrete variables. A problem that I addressed right away was the modifying of the age, weight, height, and weight columns. For instance, the age column encrypted the age in days, which is hardly practical. These were modified according to practicality and preference.

Two other variables that needed work on were the systolic and diastolic blood pressure variables. Certain encoded values were impossibly high, while others were impossibly low. I used historical data to guide me. For instance, I found the highest recorded systolic blood pressure and delimited any values from exceeding it. Additionally, I did away with any diastolic readings that I reasonably suspected were either error recordings or impossibly low (i.e., readings that no living person can have).

As to the weight and height variables, there wasn’t a clear path to take. I took into account the z-score for the height variable. I got rid of all height values that were greater than or equal to three standard deviations from the mean.

Exploratory Data Analysis (EDA)

For the EDA part, I first saw the distribution of the CVD variable. The dataset was balanced, with about 50% of individuals having CVD. Also, I checked to see if any monotonic correlations hold among variables. Unsurprisingly, the two blood-pressure variables (from earlier) were positively moderately correlated with CVD, while the age and cholesterol variables were, too, positively correlated with CVD, but only weakly.

Afterward, I created histograms for most variables, seeing their distributions with respect to the dependent variable. These visual representations came out to be incredibly helpful, as we can see how CVD is distributed according to certain features. Here are some of the distributions that were particularly noteworthy:

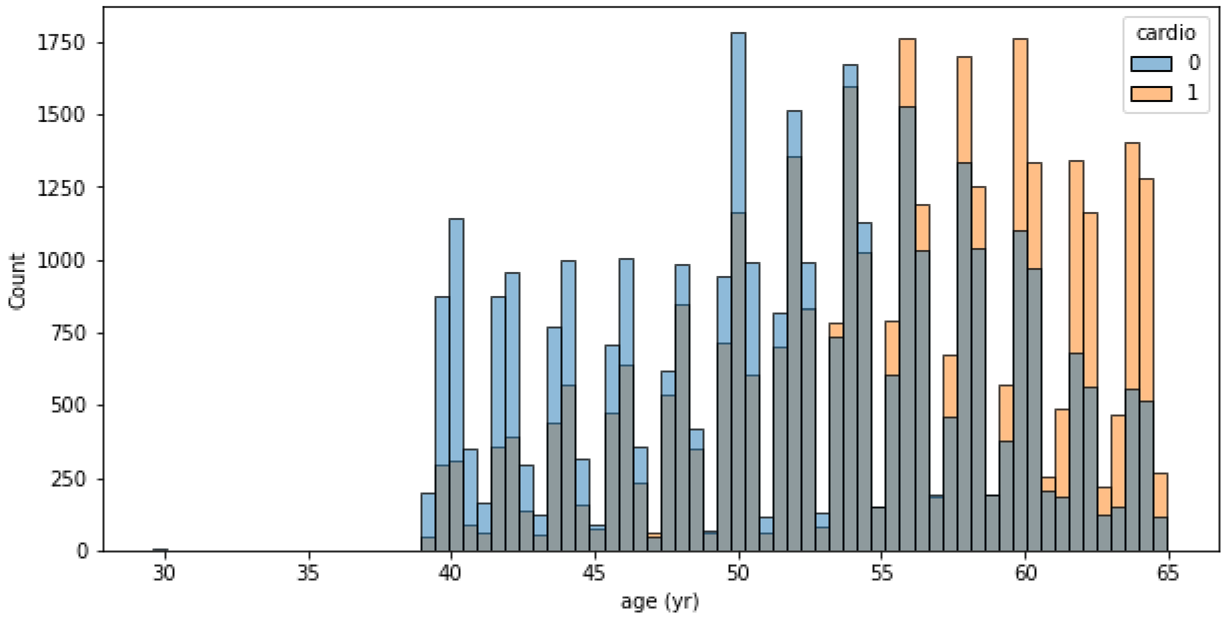


Figure 1: Histogram depicting age distribution colored by presence of CVD (1) or absence (0)

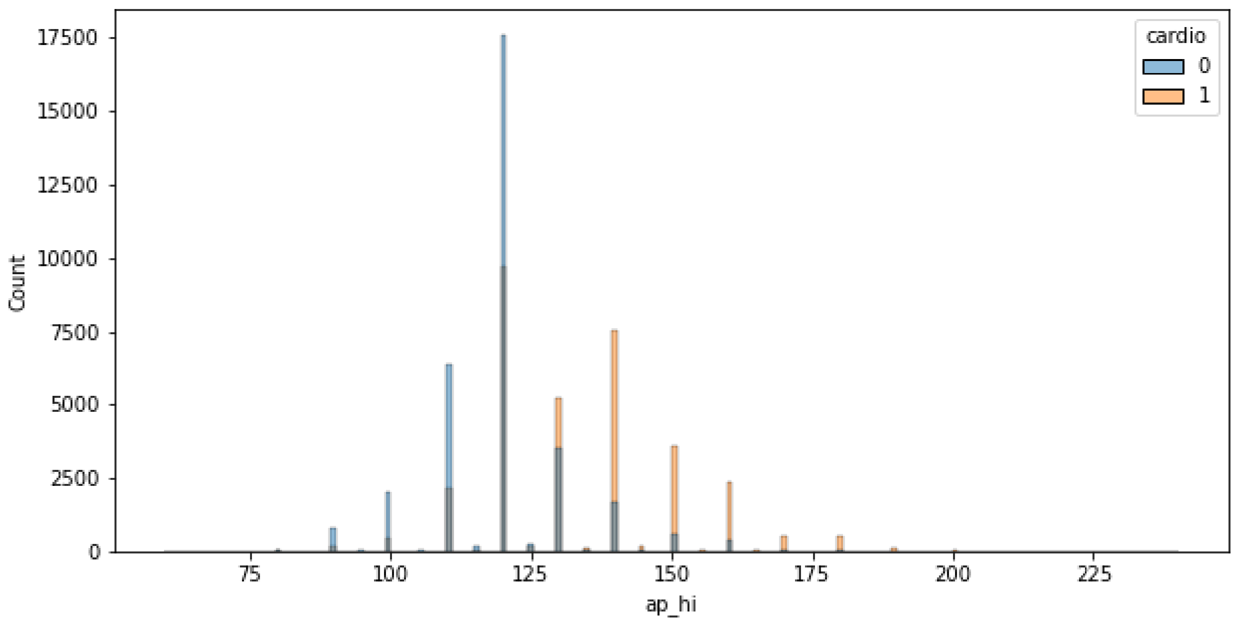


Figure 2: Histogram depicting systolic blood pressure colored by presence or absence of CVD

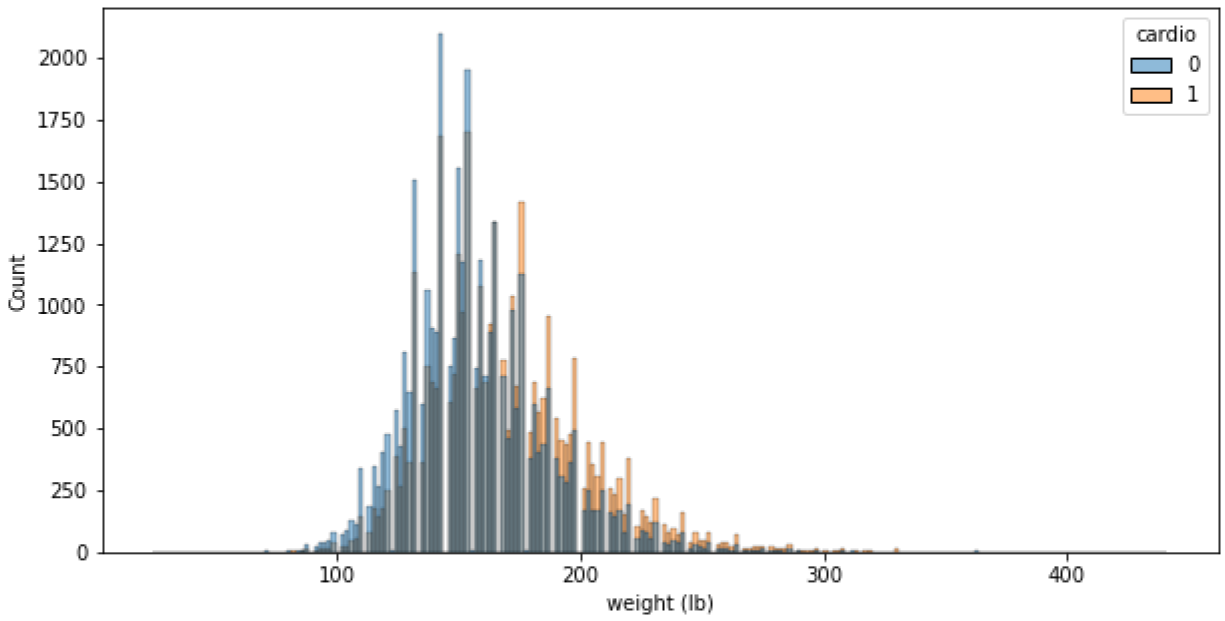


Figure 3: Histogram depicting weight distribution colored by presence or absence of CVD

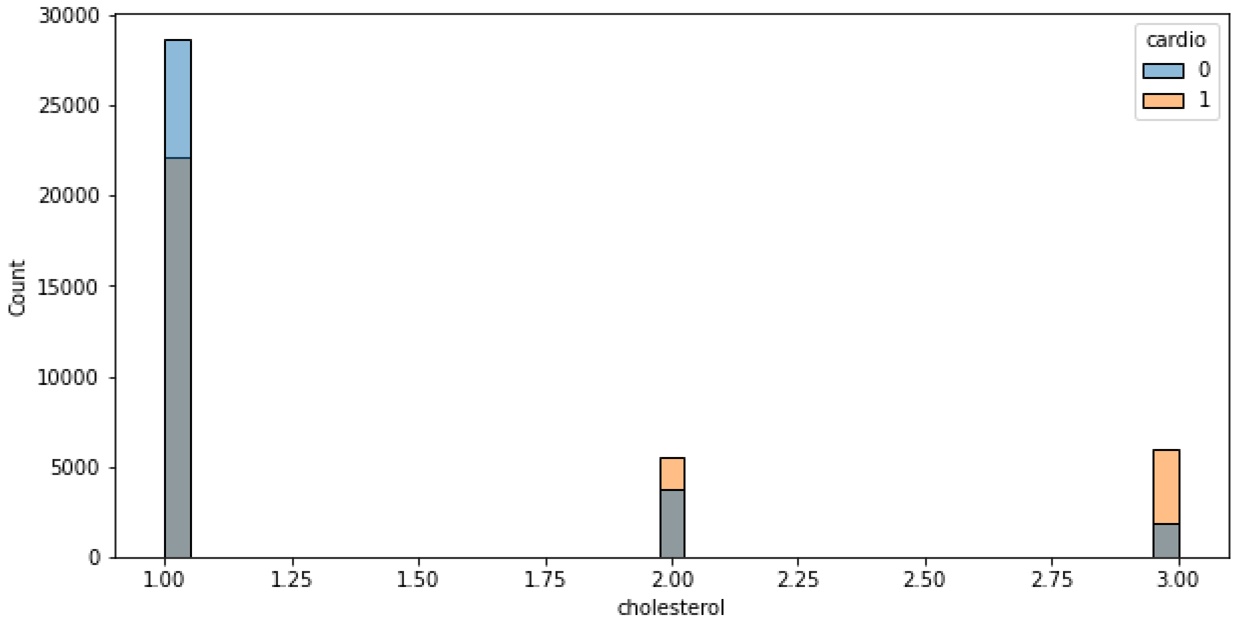


Figure 4: Bar graphs depicting cholesterol levels (1 = normal, 2 = above average, and 3 = well-above average) colored by presence or absence of CVD

As can be seen from the top two images, CVD is both more prevalent among the elderly and those with higher systolic blood pressure. As for the bottom two images, CVD is more prevalent among the heavier and those with higher cholesterol levels.

Preprocessing

First, I checked if there were any categorical variables that I may render as "dummy variables." I found none. Next, I scaled the data. This step is essential given that I will be using several classification models that are largely influenced by the scaling of the data. Lastly, I split my data into a training dataset and test dataset. This last step is to prevent data leakage.

Modeling

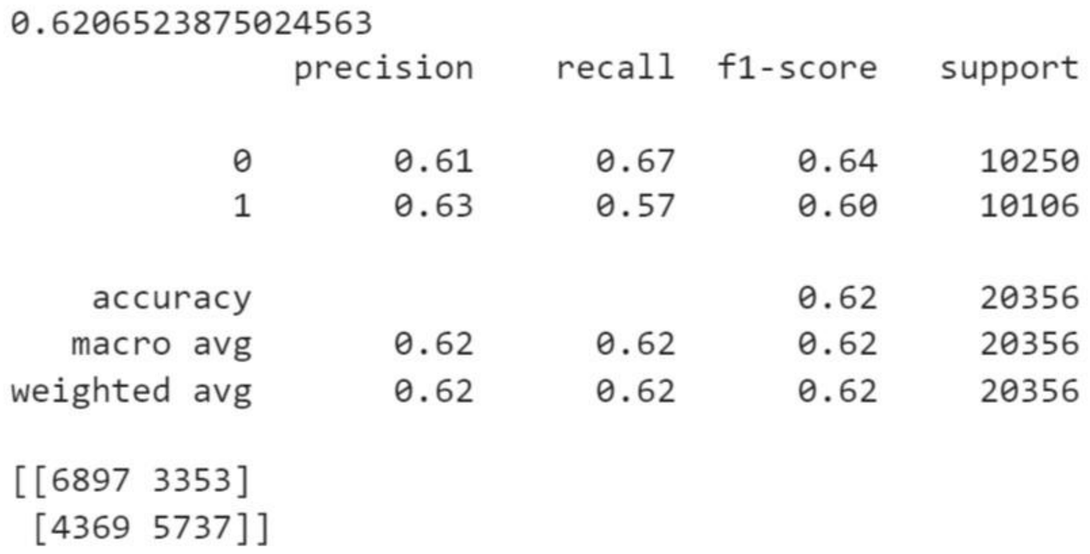
High-level Overview

As mentioned, I did hyperparameter tuning on all models. I started by instantiating a base model (i.e., a non-hyperparameter-tuned model or a default model) and fitting it onto the training data. Next, I predicted and checked its performance. (Note, this is a standard ML workflow.) I, then, got several metrics on the base model's performance –namely, I got the accuracy score, along with a classification report and confusion matrix report. The last two metrics serve to give a richer picture of how the model is performing, opposed to just noting merely its accuracy. After getting these metrics, I printed out the hyperparameters being used. This last step offers a lot of assistance; by printing out the hyperparameters, I can see which ones I might (or might not) choose to tune.

I then move onto instantiating a non-base model. Before fitting the model onto the training data, I choose my method of tuning the hyperparameters. I chose two methods: GridSearchCV and RandomSearch. (The former does an exhaustive search of the best parameters provided a grid of parameters and their possible values. This method is computationally expensive and can take a long time to complete. The latter does not do an exhaustive search but rather searches from a random selection of possible values of each parameter. This method is less computationally expensive and usually shorter than GridSearchCV.) After creating a grid of parameters, I call on my method that serves as a pipeline for the underlying model. Afterward, I fit the pipeline to the training data and predict. I get the performance metrics of each pipeline with a list of the optimal hyperparameters. After getting the list of optimal hyperparameters, I implement my non-base model with the optimal hyperparameters, giving the optimal model. Lastly, I verify the results of said optimal model, to see if it generalized well.

Low-level Overview

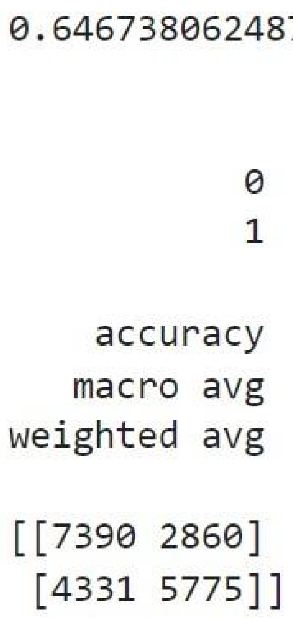
I chose four classification models: K-Nearest Neighbors (KNN), Logistic Regression, Decision Tree, and Random Forest. I used GridSearchCV for the first three models and used RandomSearch for Random Forest. Below, I'll present the performances of the base models, followed by the hyperparameter-tuned models.


```
0.6206523875024563
              precision    recall  f1-score   support

           0       0.61      0.67      0.64     10250
           1       0.63      0.57      0.60     10106

    accuracy                           0.62     20356
   macro avg       0.62      0.62      0.62     20356
weighted avg       0.62      0.62      0.62     20356

[[6897 3353]
 [4369 5737]]
```
Figure 5: Base KNN performance



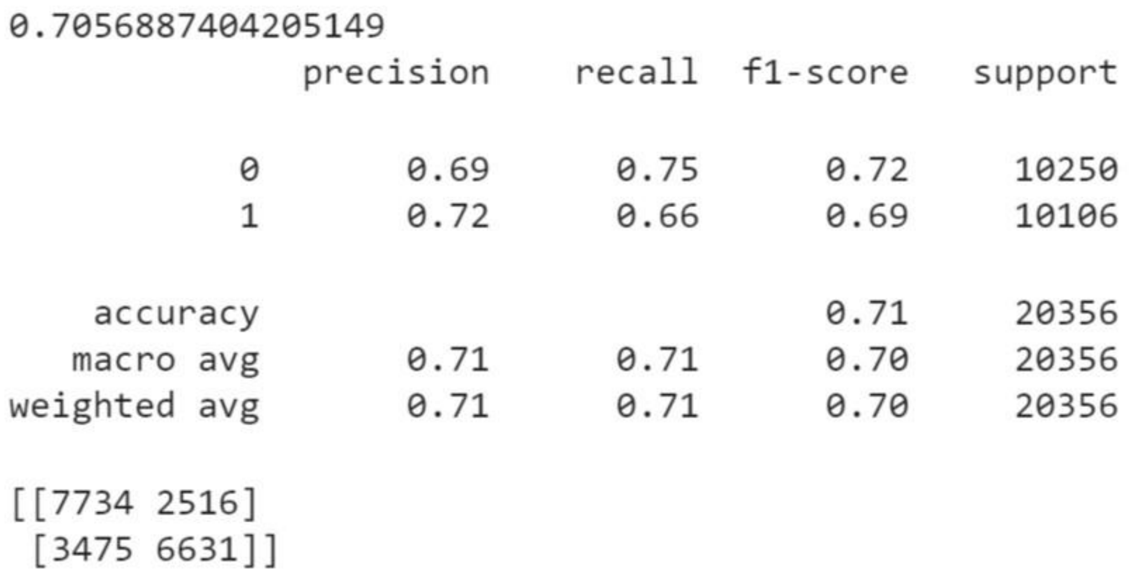
```
0.6467380624877186
              precision    recall  f1-score   support

           0       0.63      0.72      0.67     10250
           1       0.67      0.57      0.62     10106

    accuracy                           0.65     20356
   macro avg       0.65      0.65      0.64     20356
weighted avg       0.65      0.65      0.64     20356

[[7390 2860]
 [4331 5775]]
```

Figure 6: Optimized KNN performance metrics



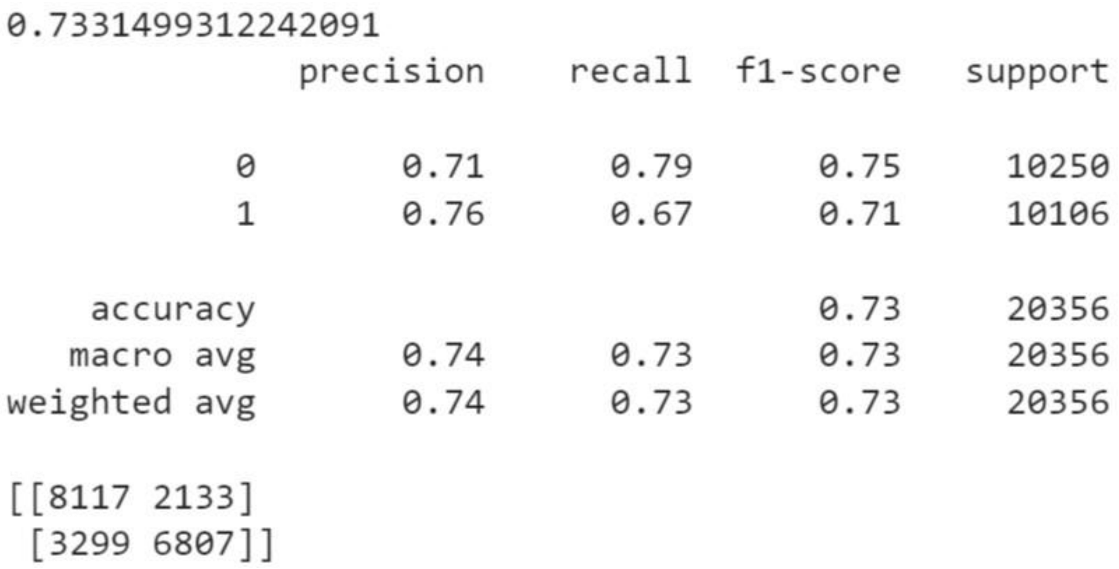
```
0.7056887404205149
              precision    recall  f1-score   support

           0       0.69      0.75      0.72     10250
           1       0.72      0.66      0.69     10106

    accuracy                           0.71     20356
   macro avg       0.71      0.71      0.70     20356
weighted avg       0.71      0.71      0.70     20356

[[7734 2516]
 [3475 6631]]
```

Figure 7: Base logistic regression performance metrics



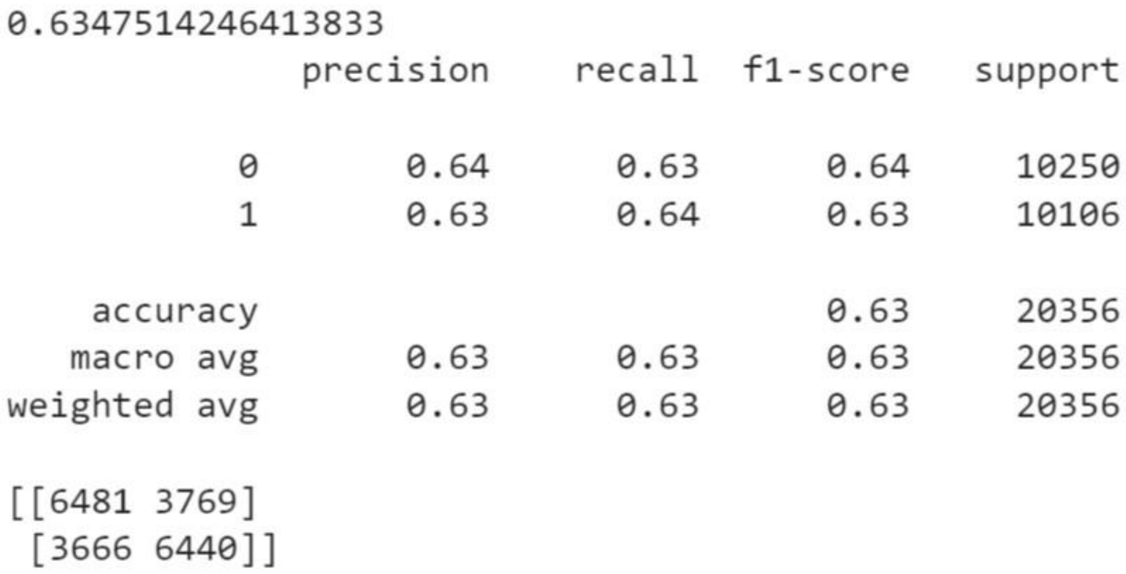
```
0.7331499312242091
              precision    recall  f1-score   support

           0       0.71      0.79      0.75     10250
           1       0.76      0.67      0.71     10106

    accuracy                           0.73     20356
   macro avg       0.74      0.73      0.73     20356
weighted avg       0.74      0.73      0.73     20356

[[8117 2133]
 [3299 6807]]
```

Figure 8: Optimized logistic regression performance metrics



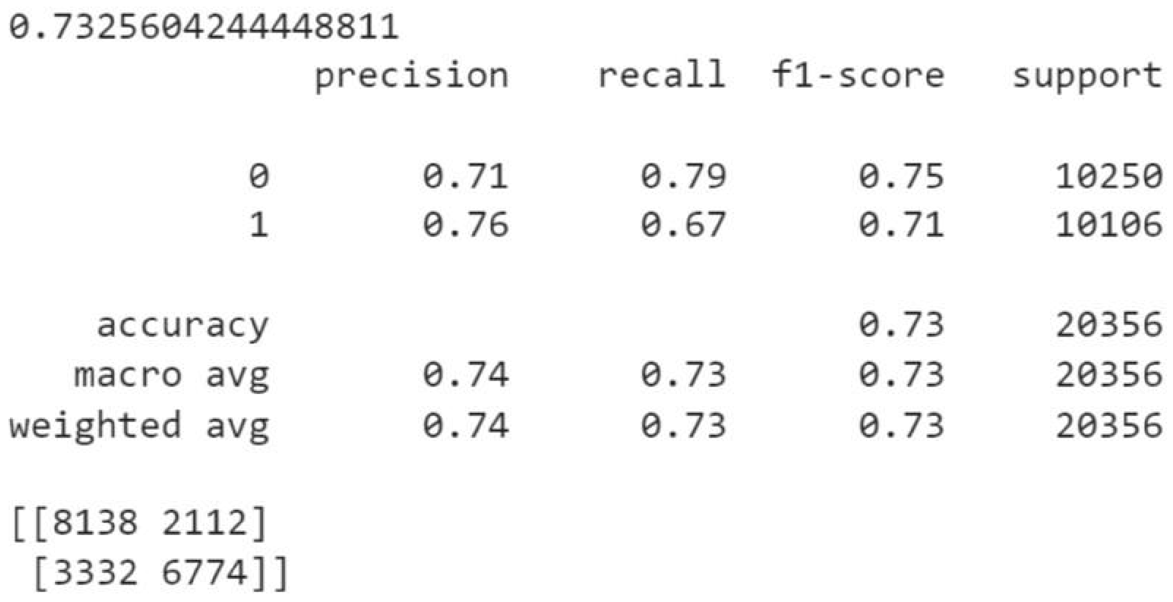
```
0.6347514246413833
              precision    recall  f1-score   support

           0       0.64      0.63      0.64     10250
           1       0.63      0.64      0.63     10106

    accuracy                           0.63     20356
   macro avg       0.63      0.63      0.63     20356
weighted avg       0.63      0.63      0.63     20356

[[6481 3769]
 [3666 6440]]
```

Figure 9: Base decision tree performance metrics



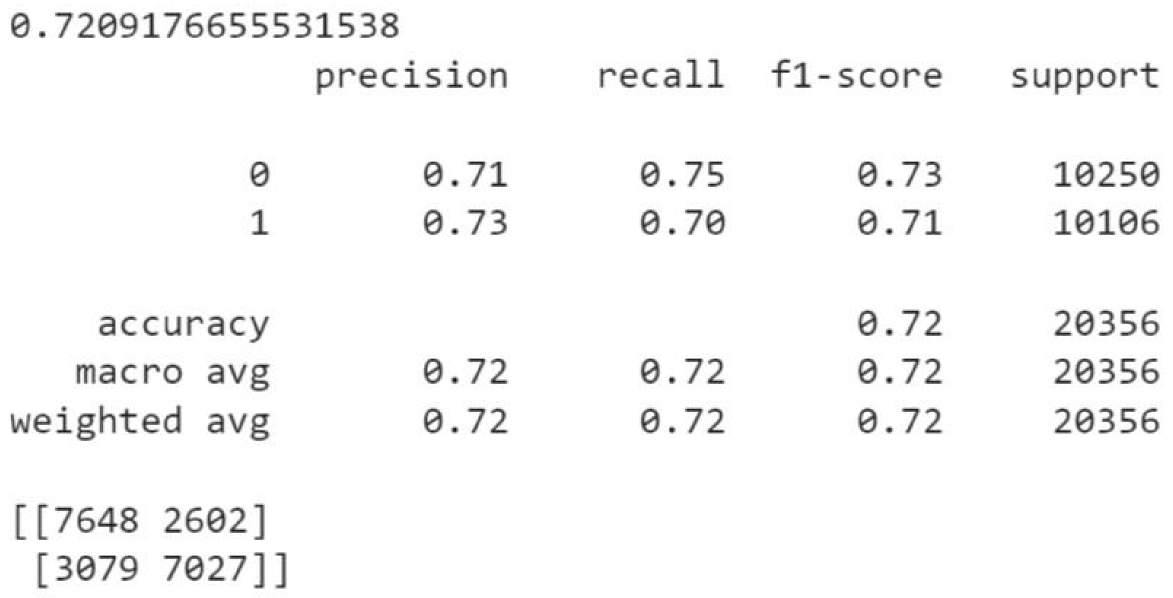
```
0.7325604244448811
              precision    recall  f1-score   support

           0       0.71      0.79      0.75     10250
           1       0.76      0.67      0.71     10106

    accuracy                           0.73     20356
   macro avg       0.74      0.73      0.73     20356
weighted avg       0.74      0.73      0.73     20356

[[8138 2112]
 [3332 6774]]
```

Figure 10: Optimized decision tree performance metrics



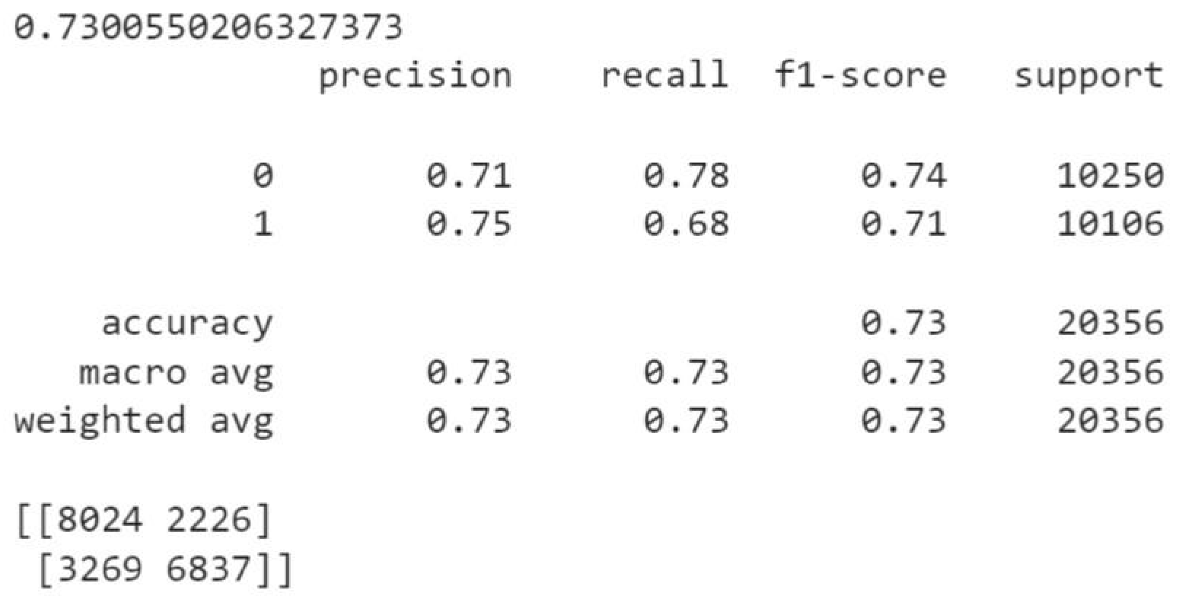
```
0.7209176655531538
              precision    recall  f1-score   support

           0       0.71      0.75      0.73     10250
           1       0.73      0.70      0.71     10106

    accuracy                           0.72     20356
   macro avg       0.72      0.72      0.72     20356
weighted avg       0.72      0.72      0.72     20356

[[7648 2602]
 [3079 7027]]
```

Figure 11: Base random forest performance metrics



```
0.7300550206327373
              precision    recall  f1-score   support

           0       0.71      0.78      0.74     10250
           1       0.75      0.68      0.71     10106

    accuracy                           0.73     20356
   macro avg       0.73      0.73      0.73     20356
weighted avg       0.73      0.73      0.73     20356

[[8024 2226]
 [3269 6837]]
```

Figure 12: Optimized random forest performance metrics

Results

As can be seen, optimizing the models did improve performance. In certain circumstances, performance only improved slightly. But this result aligns well with my earlier prediction that moderate accuracy is to be expected at best given the nature of the problem. Furthermore, as to performance, it appears that the optimized logistic model had the highest accuracy, while the KNN model, both base and optimized, performed poorly in this respect. However, the model that predicted CVD the most accurately was the base random forest model.

The main objective of this analysis was to have a model predict CVD, and it appears that a base random forest model will do. Yet, this result should not be taken as simply the truth. Although prediction of CVD was my main objective (i.e., discerning true positives was given priority), for another analyst or organization the objective may be different. An alternatively legitimate objective may be to desire the highest accuracy score. In such a case, the optimized logistic regression model is the best. But for my concerns, the base random forest model does the best job.

Takeaways

The nature of the problem is difficult, and leaves a lot of ambiguity. I hope, however, that my simple analysis shows that predicting cardiovascular disease with simple means and minimal complexity can be achieved with moderate success.