Heart Disease Model

Laura Contreras

University of Denver

COMP 4448

11/14/2022

I. Introduction

Heart disease is one of the leading causes of death in the United States. 1 in every 5 deaths is related to some form of coronary heart disease. The most common form of heart disease is ‘coronary artery disease which can lead to a heart attack since it affects the blood flow (About Heart Disease, 2022). Plaque that builds up on the artery walls from high cholesterol, unhealthy diet, alcohol use, and smoking, are what reduce the supply of blood to the heart. Symptoms and conditions of heart disease vary by sex. Men are more likely to have chest pain, while women are more likely to have shortness of breath (Heart Disease, 2022).

A worrisome factor about heart disease is that it goes undiagnosed, and many people who pass away have no previous symptoms. In men, almost half of men who die from coronary heart disease did not have symptoms. Until a heart attack, failure, or arrhythmia occurs, a diagnosis of heart disease will remain unknown (About Heart Disease, 2022 & Heart Disease, 2022).

The purpose of the project is to build a model that properly predicts whether a patient has coronary heart disease. Because of how prevalent the disease is in the US, and globally, if a model can accurately predict the diagnosis, then doctors will be able to intervene earlier than later. By comparing 4 classification models on their F1-score, and MSE (mean squared error), an accurate model is selected.

The dataset utilized in the project comes from Kaggle (Heart Failure Prediction Dataset, 2021). The data was created by combining 5 different sets over 11 common features. Each set came from different areas (Long Beach, Clevland, Hungary, Switzerland, Irvine) and had never been combined with another dataset before. In total there are 918 respondents. The target variable

used is the binary ‘HeartDisease’ attribute, where 0 is no heart disease, and 1 is a yes for heart disease. There are 10 input variables that are categorical or continuous. An example of the continuous input variables is ‘Age’, RestingBP, and MaxHR. Some categorical variables in the dataset are ‘Sex’ and ‘ChestPainType’. Utilizing these attributes, 4 models were created and compared to determine the best at predicting whether a patient has heart disease or not.

II. Data Preprocessing

A. Data Preparation

Before starting the model building, the dataset was read into a data frame and cleaned. The column types were looked at and any categorical variables that were type object were converted to categories. The continuous variables were already in the proper data type. Also, no null values were found, so they did not need to be handled.

B. Data Exploration and Visualizations

In Table 1. looking at our description of the dataset for respondents with heart disease, we see that the minimum age is 31, while the highest age is 77. For max heart rate, we see that the highest rate is 195 beats per minute, and the lowest heart rate is 60 beats per minute. We have 508 respondents with heart disease. The average max heart rate was 127.66 beats per minute. When comparing the descriptive statistics to respondents without heart disease, the minimum age decreases, and the max heart rate increase. In Table 2. the minimum age is 28, while the max heart rate is 202 beats per minute.

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	508.000000	508.000000	508.000000	508.000000	508.000000	508.000000	508.0
mean	55.899606	134.185039	175.940945	0.334646	127.655512	1.274213	1.0
std	8.727056	19.828685	126.391398	0.472332	23.386923	1.151872	0.0
min	31.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	1.0
25%	51.000000	120.000000	0.000000	0.000000	112.000000	0.000000	1.0
50%	57.000000	132.000000	217.000000	0.000000	126.000000	1.200000	1.0
75%	62.000000	145.000000	267.000000	1.000000	144.250000	2.000000	1.0
max	77.000000	200.000000	603.000000	1.000000	195.000000	6.200000	1.0

Table 1. Description of Dataset for Respondents with Heart Disease

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	410.000000	410.000000	410.000000	410.000000	410.000000	410.000000	410.0
mean	50.551220	130.180488	227.121951	0.107317	148.151220	0.408049	0.0
std	9.444915	16.499585	74.634659	0.309894	23.288067	0.699709	0.0
min	28.000000	80.000000	0.000000	0.000000	69.000000	-1.100000	0.0
25%	43.000000	120.000000	197.250000	0.000000	134.000000	0.000000	0.0
50%	51.000000	130.000000	227.000000	0.000000	150.000000	0.000000	0.0
75%	57.000000	140.000000	266.750000	0.000000	165.000000	0.600000	0.0
max	76.000000	190.000000	564.000000	1.000000	202.000000	4.200000	0.0

Table 2. Description of Dataset for Respondents without Heart Disease

In the distribution of heart disease by age (Fig. 1), we see that heart disease is more prevalent in older patients, while younger patients do not have heart disease as often. An interesting data point is at age 50, we see almost an equal amount of adults with or without heart disease (Fig 1.) In the sex attribute, heart disease is more prevalent in men. In females, while there are fewer respondents who are female, heart disease was not as common. Since 1 in 4 male deaths are from the disease, and 1 in 5 female deaths are from the disease, the distribution in the heart dataset shows that males are more likely to have heart disease than women (Heart Disease 2022, Fig. 2).

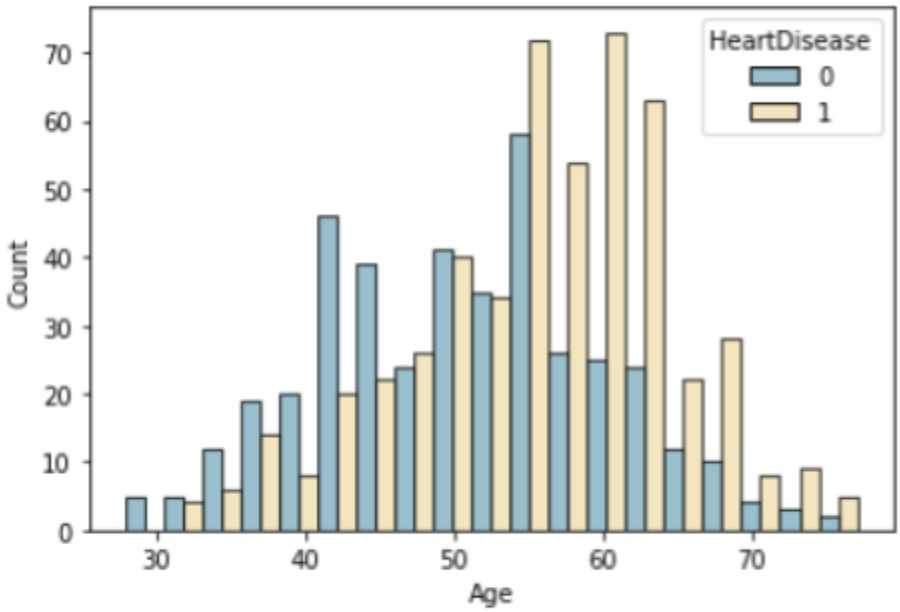

Figure 1. Histogram: Distribution of Age by Heart Disease

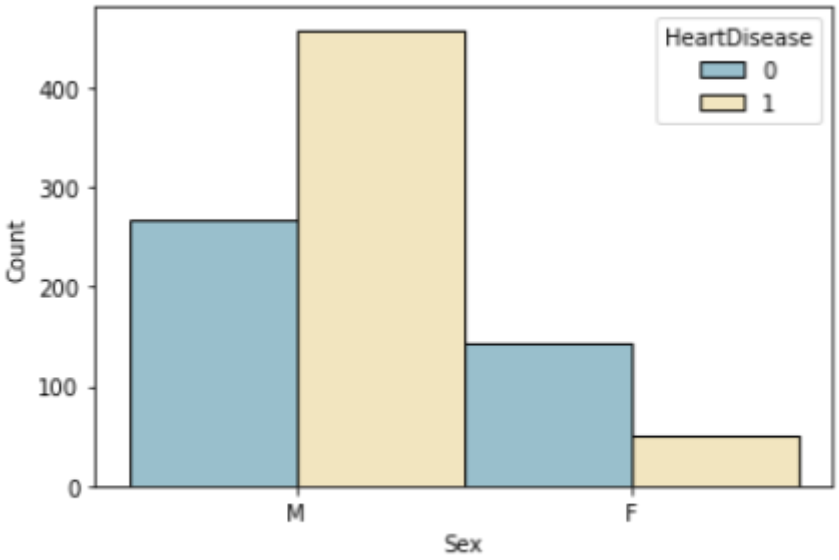

Figure 2. Distribution of sex by heart disease

In the MaxHR distribution (maximum heart rate), heart disease respondents had a lower range of max heart rates with a peak of around 130 (Fig 3). As the heart rate increased, the number of

respondents decreased significantly. Respondents without the disease had a much broader distribution with the majority having a maximum heart rate of 120-170 bpm. The peak was at 160 beats per minute (Fig 3).

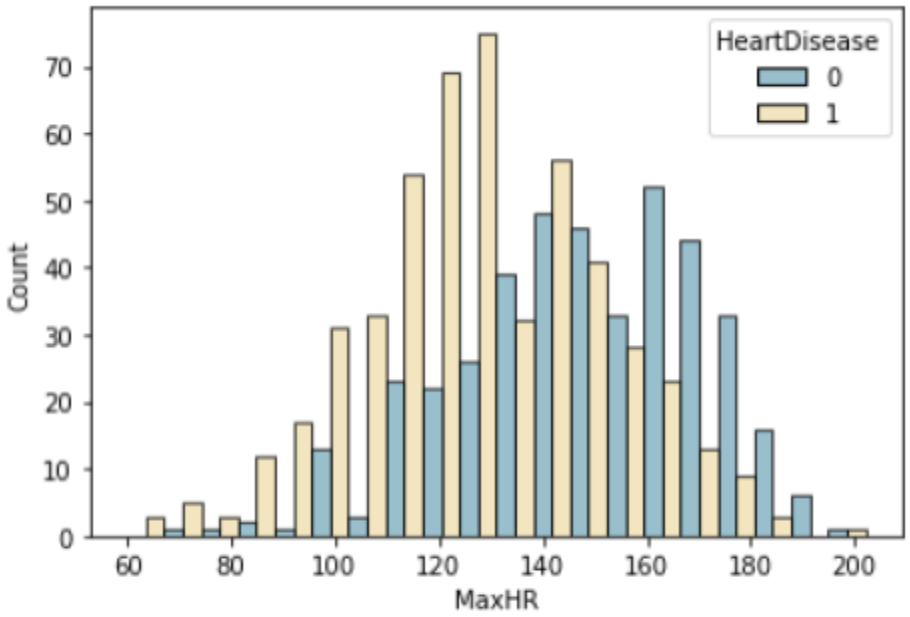

Figure 3. Distribution of maximum heart rate by heart disease

Figure 4. shows the correlation between the continuous variables and the target variable heart disease. Looking at the heatmap we have the highest positive correlations being ‘Oldpeak’ (0.4) and ‘Age’ (0.28), and ‘MaxHR’ has the strongest negative correlation (-0.4). Oldpeak represents the ST depression in an ECG (electrocardiogram that checks rhythm and electrical activity). When placing these three attributes in a 3D plot separated by heart disease, there is a clear relationship between the three (Fig. 5). For respondents with heart disease, there are more points where the age is higher, the maximum heart rate is lower, and the Oldpeak is lower than 0.4.

Without heart disease, the points are distributed where the age is lower, the maximum heart rate is higher, and Oldpeak is above 0.4 (Fig. 5).

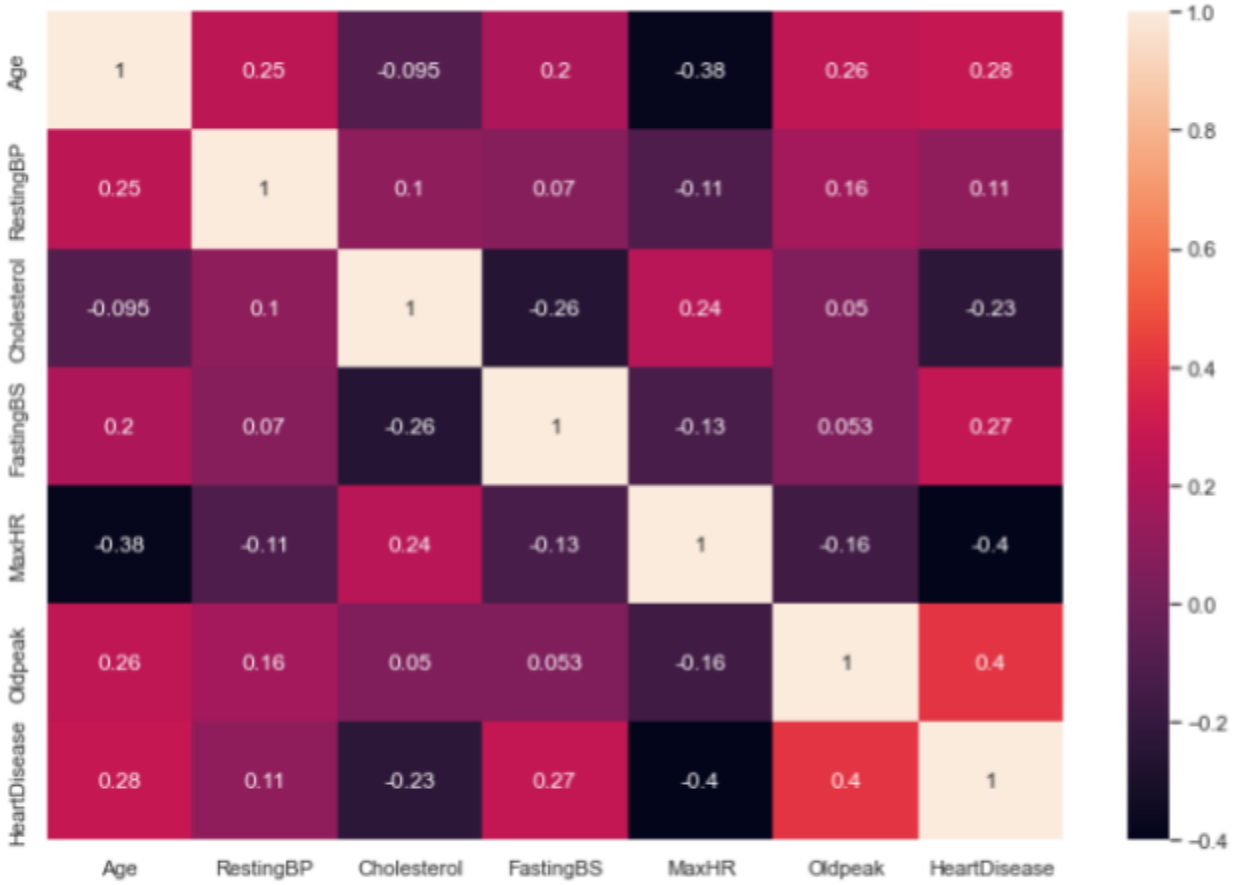

Figure 4. Correlation Heatmap of continuous variables

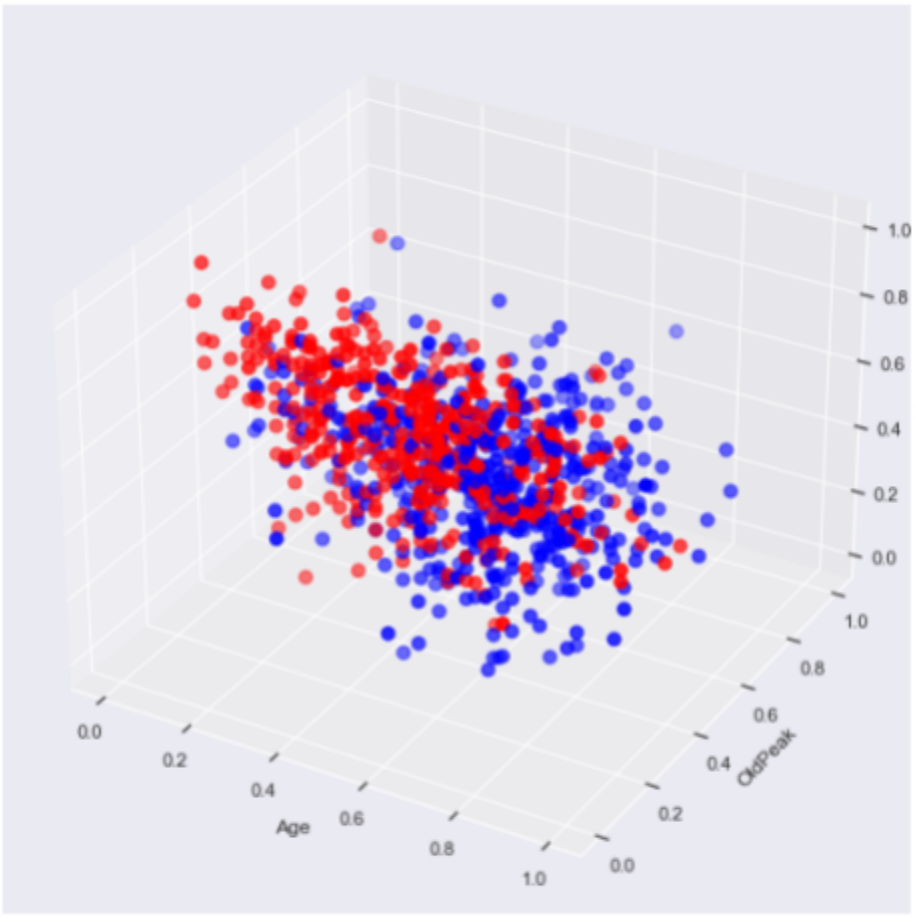

Figure 5. 3-D plot of age, maximum heart rate, and Oldpeak by heart heart disease

c. Data Splitting

Since the target variable was binary (0 or 1), the continuous variables were scaled using the MinMaxScaler from the sklearn package. Dummy variables were created for the 'Sex', 'ChestPainType', 'RestingECG', and 'ST_Slope', while the categorical variable 'ExerciseAngina' was mapped to 0 and 1 since its responses were 'yes' and 'no'. The data frame was then split on a 70/30 train test split, with 'HeartDisease' being the target variable. From the split, a test and train x and y sets were obtained.

III. Model Building and Evaluation

a. Model Building

Four algorithms were chosen to be compared, Logistic Regression, RandomForestClassifier, KNClassifier, and DecisionTreeClassifier. There were no specific parameters given, so the default parameters were used during the model fit. Through a for-loop, each model fit the train x and y set, obtained predictions on the train and test sets, then calculated the F1 score. The F1 was used since the target variable was binary and false positives/negatives are important in the prediction outcomes. A false negative or positive could prove costly in the real world if a patient is diagnosed incorrectly (Fig 6). Using a PrettyTable, each model with its corresponding train/test f1-score was placed into the table. In Table 3. we see that there is overfitting for all 4 models, as the train f-1 score was 1.0 for both DecisionTree and RandomForest, while the other two models had a score relatively close to the test f1 score (Table 3).

```
models = [DecisionTreeClassifier(), RandomForestClassifier(), LogisticRegression(), KNeighborsClassifier()]
```

```
# Creating Pretty Table
tbl = PrettyTable()
tbl.field_names = ["Model name", "Train F1", "Test F1"]
```

```
for model in models:
    model.fit(X_train, y_train)
    test_preds = model.predict(X_test)
    train_preds = model.predict(X_train)
    tstf1 = f1_score(y_test, test_preds)
    trnf1 = f1_score(y_train, train_preds)
    tbl.add_row([model, trnf1, tstf1])
```

Figure 6. Model Building

Model name	Train F1	Test F1
DecisionTreeClassifier()	1.0	0.7597402597402597
RandomForestClassifier()	1.0	0.897196261682243
LogisticRegression()	0.8758915834522112	0.8937499999999999
KNeighborsClassifier()	0.886039886039886	0.869281045751634

Table 3. Model Comparison, train and test F1 score

b. Model Hyperparameter Tuning and Model Selection

Each model went through hyperparameter tuning through a Grid Search Cross-Validation in the sklearn.model_selection package. Figure 7. shows the different parameters the GridSearchCV looked through for each model. To find the best_estimator the "f1" was utilized for the scoring parameter. The specific grid fit each model with parameters and the train datasets.

```
new_params = {'n_neighbors': range(1,50)}
grid_knc_new = GridSearchCV(estimator = KNeighborsClassifier(), param_grid = new_params, scoring = "f1")
grid_knc_new.fit(X_train, y_train)
print(grid_knc_new.best_estimator_)
```

```
KNeighborsClassifier(n_neighbors=49)
```

Figure 7. Model Tuning example

Utilizing the best_estimator that the GridSearchCV produced, a for loop was run again with the new algorithms containing the best parameters (Fig 8). The new model fit the train x and y set, and obtained predictions on the test set, then calculated the F1 score and the MSE score (mean squared error). Using a PrettyTable again, each model with its corresponding F-1 score and MSE score was placed into the table. To select the best model, a high F-1 score (close to 1) and a low

MSE score are needed. From the PrettyTable, RandomForestClassifier is the best model since it had an F-1 score of 0.8966 and an MSE of 0.1196 (Table 4).

```
models = [grid_dtc_new.best_estimator_, grid_rfc_new.best_estimator_, grid_lgr_new.best_estimator_,
          grid_knc_new.best_estimator_]
```

```
# Creating Pretty Table
tbl = PrettyTable()
tbl.field_names = ["Model name", "F1", "MSE"]
```

```
for model in models:
    model.fit(X_train, y_train)
    test_preds = model.predict(X_test)
    tstmse = mean_squared_error(y_test, test_preds, squared = True)
    f1 = f1_score(y_test, test_preds)
    tbl.add_row([model, f1, tstmse])
```

Figure 8. Code for model performance after tuning

Model name	F1	MSE
DecisionTreeClassifier(max_depth=3, max_features=8, random_state=40)	0.8481012658227848	0.17391304347826086
RandomForestClassifier(max_features=2, n_estimators=30, random_state=42)	0.896551724137931	0.11956521739130435
LogisticRegression(C=0.01)	0.890282131661442	0.12681159420289856
KNeighborsClassifier(n_neighbors=49)	0.8742138364779873	0.14492753623188406

Table 4. Tuned model performances

After selecting out RandomForest as the best model, further hyperparameter tuning was attempted by expanding on the parameters chosen in the first tuning. However, after running the GridSearchCV again, the same best_estimator was chosen (Fig 9).

```
params = [{'n_estimators': [3,5,10,20,25,30,35,40,45], 'max_features': [2,4,6,8,10,12,14,16,18]},
          {'bootstrap': [False], 'n_estimators': [3,5,10,20,25,30,35,40,45], 'max_features': [2,3,4,6,8,10,12,14,16,18,19]}]
```

```
grid_rfc_new = GridSearchCV(estimator = RandomForestClassifier(random_state=42), param_grid = new_params, scoring="f1")
grid_rfc_new.fit(X_train, y_train)
print(grid_rfc_new.best_estimator_)
```

```
RandomForestClassifier(max_features=2, n_estimators=30, random_state=42)
```

Figure 9. Further hypertuning on RandomForestClassifier code

c. Model Summary

The tuned RandomForestClassifier was selected as the best model for predicting whether a respondent has heart disease or not. In Table 5. the overall accuracy for the model is 0.88 with an overall F1 score of 0.8966. In Table 6, the confusion matrix shows that there are minimal false negatives and false positives. The recall for true positives is 0.87 and the precision is 0.92. The model correctly identifies heart disease 87% of the time, using recall (Table 5).

```
RandomForestClassifier(max_features=2, n_estimators=30, random_state=42)
MSE: 0.11956521739130435
Overall Accuracy: 0.8804347826086957
Overall F1: 0.896551724137931
```

	precision	recall	f1-score	support
0	0.83	0.89	0.86	112
1	0.92	0.87	0.90	164
accuracy			0.88	276
macro avg	0.87	0.88	0.88	276
weighted avg	0.88	0.88	0.88	276

Table 5. RandomForestClassifier Model Statistics

```
array([[100,  12],
       [ 21, 143]], dtype=int64)
```

Table 6. Confusion Matrix Actual vs Predicted

To further support the model, an AUC-ROC score was calculated and plotted. The AUC-ROC score is the area under the curve for the receiving operating characteristic. If the plot is above 50% of the area then it is a good model. Figure 10, that the ROC curve sits above 50% of the

area and the model is therefore a good fit for the data. The AUC-ROC score obtained is 0.8824 (Fig. 10) further supporting the RandomForestClassifier model.

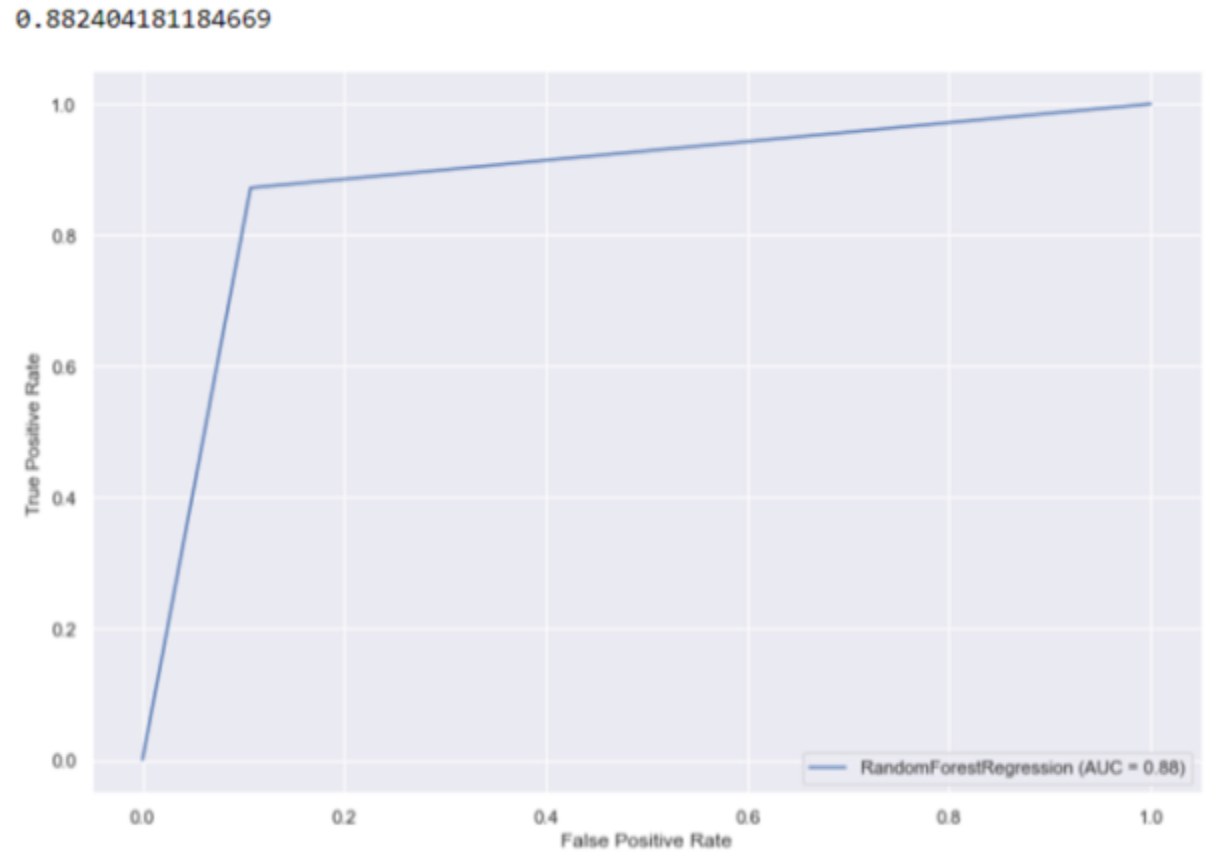

Figure 10. ROC Score and Curve

IV. Conclusion

By comparing the four models, KNClassifier, Logistic Regression, DecisiontreeClassifier, and RandomForestClassifier, it was found that the RandomForestClassifier was the best and recommended model for predicting whether a person has heart disease or not. The model had an accuracy of 0.88 and an overall F1 score of 0.8966.

A lesson learned during the hyper-tuning is that expanding on the parameters for a GridSearchCV can make the code run for a long period of time. It was attempted to expand further on the hyperparameters during the second tuning, however, the code ran for about 2 days before selecting the best estimator. If one is one has a timeline due date, it is important to realize how many different combinations the GridSearchCV will look through before running the code.

To further improve the accuracy and F1 score, outliers could be looked at and removed on continuous variables. Also, during hyperparameter tuning, expanding on different parameters could improve the model by selecting a better estimator.

References

Centers for Disease Control and Prevention. (2022, July 12). *About heart disease*. Centers for Disease Control and Prevention. Retrieved November 13, 2022, from https://www.cdc.gov/heartdisease/about.htm

Fedesoriano. (2021, September 10). *Heart failure prediction dataset*. Kaggle. Retrieved November 13, 2022, from https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

Mayo Foundation for Medical Education and Research. (2022, August 25). *Heart disease*. Mayo Clinic. Retrieved November 13, 2022, from https://www.mayoclinic.org/diseases-conditions/heart-disease/symptoms-causes/syc-20353118