Predicting Heart Disease - A Springboard Capstone Report *by Sheena Sharma*

Heart disease is currently the leading cause of death in the United States[[1]](#footnote-1). In addition to the devastating loss and emotional pain caused by heart disease, it is also a large financial burden. It is estimated that heart disease cases contribute to most of the $320+ billion yearly direct healthcare expenditure[[2]](#footnote-2). To curtail the staggering numbers of lost life and the financial burden of this disease, it would be vastly beneficial to be able to predict whether or not a person will develop heart disease and to what severity. Current predictive models use a few key factors such as blood pressure, total cholesterol, and LDL cholesterol to predict the incidence of heart disease[[3]](#footnote-3), however these models are unable to predict the severity of heart disease. Better predictions of the development of heart disease and its severity will have huge social and economic advantages and thus in this project I aimed to build a model that can predictive heart disease and the severity of heart disease. Due to the vast social and economic advantages such a model, there may be many stakeholders. For the purposes of this project I anticipated the client to be a hospital, who’s goal is to develop targeted tests and treatment to those likely to develop heart disease and not spend additional resources testing and treating those who will very likely not develop heart disease. In this report I will summarize the technical details of building a classifier to predict heart disease and its severity, including how I cleaned/wrangled the data and how I came to the conclusion of using a Random Forest Classification algorithm. To conclude, I summarize take-home messages of this work and provide recommendations to the client. Finally, I address limitations on this work and future directions.

To begin with, I accessed data via the UCI repository. There were 3 datasets, each condition patient data from 3 unique hospitals. Each dataset contained about 200 unique rows of data and the same 76 attributes. The first step was loading each dataset into a pandas data frame in the appropriate format and eliminating obvious meaningless features. The majority of the 76 features could not be used to develop a predictive model. Some features contained a patient name, social security number, or patient id number. Other feature columns were completely blank and other feature columns were labeled as ‘Do Not Use,’ ‘Irrelevant,’ ‘unsure if data was collected consistently,’ or ‘unsure of units’. I threw all such feature columns out. Also, some features had no explanation of what the feature was. In these cases, the feature columns were denoted by a series of letters or numbers without any indication as to what type of data the feature column contained. I threw these out as well because it would not be useful to tell a client ‘Here are some unknown features that can predict heart disease.’ After the initial data wrangling and cleaning I was left with 20 remaining features. The features were a mix of categorical variables continuous variables.

Upon exploring the data, I found that in each feature column there are quite a few missing values (denoted by -9) and a few values that were 0. Adding some ‘jitter’ to the plots were useful in exploring categorical data. The added jitter helps to few the spread of data rather than using a traditional plot in which all participant’s data would be overlapping in the case of categorical variables.

Neither a value of -9 or 0 make sense for the continuous data as you cannot have a 0 or -9 age or heart rate or blood pressure. I found that histograms were particularly useful for seeing the negative values.

I tried a few different techniques of dealing with the missing values:

Option 1 - Regular data) I used all data including -9s and 0s. I did not make any decisions on data imputation or feature selection and instead just wanted to see how negative values and 0 values effect the models. Sometimes missing data can be informative. For instance, it could be that people for which there is no maximum heart rate measure were unable to undergo the cardiac stress test due to other health reasons that could be relevant to predicting heart disease. However, the data is missing at random as far as I know.

Option 2 – OneHotEncoder) First, I thresholded the data such that if a feature contained more than 50% missing values or 0s then I removed that feature completely. Some sources suggest that if more than 50% of the data is missing then imputing that much missing data will not add any strength to the model[[4]](#footnote-4). Next, I used the OneHotEncoder to encode the categorical variables. This takes out any nominal ordering of categorical variables (1,2,3,4) that may be interpreted as a meaningful order by a given model. For the continuous data, I checked if there were any relationships between the features that could inform the way missing values are imputed. For example, if rest heart rate and max heart rate were linearly related then I could use a linear model to impute the missing resting heart rate values based on the max heart rate values. Additionally, if two features are highly correlated then one feature could be eliminated as multi-collinearity can impact the predictive power of some models. For instance, Decision Trees choose a single feature to maximize information gain and thus suffer when two features are highly correlated. In this case I found that no two features were highly correlated. Therefore, I imputed the missing values with the mean of each feature.

Option 3 - Label Coded) I thresholded the data the same way as above. If more than 50% of the values of a feature were missing, then that column was dropped. All continuous features were imputed the same way as in option 2 (with the mean value). For categorical variables, the original label coding was kept (0,1,2,3). Missing values in categorical features were imputed using the mode of each categorical feature.

Option 4 - Normalized data) The data were thresholded and imputed as described in option 3. All continuous data were normalized. Some classifiers, like the support vector machine, require normalized data to make more accurate predictions.

Next I chose a handful of classifiers to train and chose evaluation metrics. Each of the four data sets were split into training data and test data and the training data was subsequently used to train each classifier and compare evaluation metrics. This is a supervised learning data set as the outcome labels are given. I chose the following supervised learning techniques: logistic regression, knn, decision tree, random forest, and support vector machine. For evaluation I chose to look at a variety of metrics. The accuracy of the estimator on training data is an important measure as it indicates how well each model predicts the training data. I also computed a cross\_validation score (mean of 5 folds) for each classifier. The objective of cross validation is to choose different partitions of the training data and test the classifier on each partition so that the classifier is not biased to just one partition. Cross validation helps with overfitting.

I also looked at precision, recall, the area under the ROC curve, and confusion matrices for each classifier. The area under the ROC curve is a commonly used metric in binary class problems. In the case of this multiclass problem, a one v rest paradigm was used to create 5 binary problems (Class 0 vs All other classes, Class 1 vs All other classes, etc) and the area under the 5 resulting ROC curves was calculated. We want to maximize the area under the ROC curve, which occurs with maximal true positives and minimal false positives (see model example from scikit learn below).

In a multiclass problem such as this one in which we create 5 binary classifiers, the area under the curve may not be the best metric to use as it can give us very high, and potentially, misleading areas under the curve. For instance, in the case of the Class 4 v rest classifier, the area under the ROC curve is very high (near 0.9) because the hundreds of samples that were accurately classified as not belonging to class 4 were classified as True Positives, even in the case of mislabeling within the rest group (for instance if a class 3 instance was classified as class 2). Therefore, in addition to the area under the ROC curve, I also looked at metrics such as precision, recall and the F1 score (harmonic mean of precision and recall), which are particularly useful in cases on class imbalance. In this case the dataset not balanced – Class 0 (no heart disease) makes up 40% of the dataset whereas class 4 (most severe form of heart disease) only makes up 4% of the dataset.

One key difference between the calculation for area under the ROC curve (where TP = TP/TP + FN and FP = FP/FP + TN) and the F1 score (where precision = TP/TP + FP and recall = TP/TP + FN), is that the area calculation includes true negatives whereas the F1 score does not. Therefore the area calculation is more likely to be skewed by imbalanced datasets than the F1 score as described above (we have a lot of TN as with the Class 4 v rest classifier which can potentially skew the area under the ROC curve metric).

I tried each of the 5 classifiers (logistic, knn, decision tree, random forest, and svm) with each of the 4 cleaned datasets options. **Ultimately, all dataset options gave similar scores, however I found that that the precision and recall were consistently higher for the random forest model with dataset 2 – OneHotEncoder**. Below is a brief description of the categorical and continuous features in dataset 2.

|  |  |
| --- | --- |
| Sex – Male or Female | Ex\_bp1 – peak systolic exercise blood pressure |
| Age – Patient Age | Ex\_bp2 - peak diastolic exercise blood pressure |
| Rest\_bp – Resting blood pressure (mmHg) | Cp\_type – chest pain type |
| Chol – serum cholesterol (mg/dl) | Fbs – fasting blood sugar < 120 mg/dl (1 =True, 0 =False) |
| Max\_hr – Maximum heart rate | Rest\_ecg - resting electrocardiographic results |
| Rest\_hr – Resting heart rate | Exang - exercise induced angina (1=True; 0 =False) |
| Slope – slope of peak exercise (1=up, 2=flat, 3=down) | Oldpeak - ST depression induced by exercise |

Now, I will describe how I came to the conclusion of using a random forest algorithm.

The first classifier I tried was the logistic regression. The logistic regression cannot handle a multiclass problem out-of-the-box. I had to binarize the data using a One v. All (or One v. Rest) system. With one v. all, it is essential to iterate through each class and label it as positive while the remaining classes all receive a negative weight. In this case, to train a multiclass logistic classifier, 5 unique classifiers were trained. A standard L2 penalty was chosen at this early stage along with a lbfgs solver, which is capable of handling multiclass datasets. On each of the four data sets the logistic regression had a low accuracy score and a cross validation score (around 50%), suggesting high bias.

Next I tried a decision tree model. I initially thought this would be the most ideal model due to its ease of handling both categorical and continuous variables along with its easy interpretability. However, I found that the decision tree model had high variance and was thus overfitting the data as evidenced by the high accuracy score (~.99) but low cross\_val score (~ .48).

Random forests can help combat decision trees from overfitting as they introduce 2 sources of randomization: 1) Drawing a random bootstrap sample with replacement from the original dataset from which to grow a decision tree and 2) Drawing a random sample of features without replacement from the original dataset for each decision tree. I tried the random forest model on all datasets and found that while the cross\_val score was about the same as for the logistic regression (~.48), the precision and recall scores were consistently higher for each class when using Random Forest compared to Logistic Regression. In this case I want to maximize the precision scores in order to maximize the amount of times we are accurately predicting a heart disease label.

I also tried k-nearest neighbors and support vector machine models. The cross\_val score was again similar for these models compared to Random Forest (~.48). However, the precision and recall scores were overall lower for these two models compared to Random Forest.

Looking at all evaluation metrics for all classifiers across all 4 datasets led me to choose the **random forest model with dataset 2 – OneHotEncoder**. However, it is important to note that in this case no dataset provided vastly better than any other datasets. Dataset 2 had some slight advantages over the other data sets when used with the Random Forest model (marginally higher cross\_val and F1 scores). Below is an example of evaluation metrics for the Random Forest model (left) and the Logistic Regression model (Right) on dataset 2. This example shows that while cross\_val scores are similar for both models, the precision scores are consistently higher for each class when the Random forest model is used.

At this point, after choosing the Random Forest model on dataset 2, OneHotEncoder, I sought to optimize the hyperparameters of the Random Forest. With this model, there aren’t too many hyperparameters to worry about and we don’t have to worry as much about overfitting and thus specifying an optimal number of nodes, the way that we do with decision trees. However, to prevent overfitting a chose a selection of decision trees (n\_estimators). This data set is small so we may not need as many trees as we would for a larger, more complex data set. Other key hyperparameters I chose to optimize were the maximum number of features, and the information gain criteria (gini or entropy). Gini and entropy are two measures of information gain and decision trees/random forests operate by maximimizing the information gain at each node. Although the calculations for gini and entropy are different, the two measures provide very similar information and thus changing the criterion should not greatly effect the accuracy of the model.

After optimizing the hyperparameters, I tested the optimized random forest on the test data, or hold-out dataset. The prediction accuracy of each class in the test data set appeared high ( > 90%), however it is important to note that accuracy is not the best metric in the case of an imbalanced multiclass problem. To assess the prediction accuracy of each class, we must binarize the data in a One v Rest fashion. In doing so, we often end up with a small positive class and a huge negative class (As is the case with class 4 where 4 instances in the test set belong to class 4 and 120 do not). In this case, the prediction accuracy is really high **even though the model did not accurately predict any of the 4 positive instances because the model was weighted by all of the negative instance. By predicting that all samples were in the negative class we achieved a high prediction accuracy for class 4 because technically only 4 samples were not classified correctly – the 4 positive instance that we really want to predict accurately.**

Since prediction accuracy is a misleading measure in this case, I again turned to precision, recall and F1 scores. Below is a blood of the F1 scores for each class for the training and test data sets. The image shows that that F1 score is reasonably high for class 0 for both the training and test data sets (~ 0.8), but that the F1 score for all other classes in the test data are really low.

To further investigate how many times the model predicted true positives and true negatives, I calculated a confusion matrix. Again here we can see that the model is doing a pretty good job of predicting Class 0, but not the other classes.

**Therefore, I conclude that the model performs well for predicting Class 0, but not the other classes**. To this end, I binarized the data (no heart disease = 0, heart disease = 1) in order to investigate key features that can predict the occurrence of heart disease. Below is a decision tree of the results. This decision tree has allowed me to come to the following recommendations: Those with asymptomatic chest pain, below the age of 56.6, with a maximum heart rate of less than or equal to 96.5 will likely develop heart disease.

Using the same dataset in a binary format (heart disease or no heart disease), some have shown about a 65% accuracy in predicting the incidence of heart disease. It’s important to keep this ‘baseline’ prediction accuracy in mind to set realistic expectations for the outcome of a model. In order to continue to improve a model that can accurately predict heart disease and its severity, it would be useful to have a larger dataset, with more features.

[5] http://www.becomingadatascientist.com/2015/01/19/data-science-practice-classifying-heart-disease/

1. CardioSmart: https://www.cardiosmart.org/Heart-Basics/CVD-Stats [↑](#footnote-ref-1)
2. CDC foundation: http://www.cdcfoundation.org/pr/2015/heart-disease-and-stroke-cost-america-nearly-1-billion-day-medical-costs-lost-productivity [↑](#footnote-ref-2)
3. Wilson, P. W. F., R. B. D'agostino, D. Levy, A. M. Belanger, H. Silbershatz, and W. B. Kannel. "Prediction of Coronary Heart Disease Using Risk Factor Categories." Circulation 97, no. 18 (1998): 1837-847. doi:10.1161/01.cir.97.18.1837. [↑](#footnote-ref-3)
4. AnalyticsVidhya: https://www.analyticsvidhya.com/blog/2015/07/dimension-reduction-methods/ [↑](#footnote-ref-4)