CS5525 Final Submission Report

Jennifer Appiah-Kubi, Rebecca DeSipio, Ajinkya Fotedar

12/11/2021

Contents

I. Introduction

The goal of this project is to predict the probability of having a heart attack using 13 variables given in the heart.csv data-set. With the response variable being of binary categorical form, classification methods were preferred. Subsequently, in this project, classification models chosen to analyze this data-set were: decision tree and its variants (random forest and bagging), support vector machines (SVM), logistic regression, and k-nearest neighbor (KNN). After exploring these various classification methods, we analyze and interpret the results of each method to determine which might be the "best" classifier.

The data used is the heart.csv data. It contains the following attributes:

- `age`
- `sex`
- `cp`: chest pain type (4 values)
- `trestbps`: resting blood pressure
- `chol`: serum cholesterol in mg/dl
- `fbs`: fasting blood sugar $>$ 120 mg/dl
- `restecg`: resting electrocardiograph results (values 0, 1, 2)
- `thalach`: maximum heart rate achieved
- `exang`: exercise induced angina

- `oldpeak`: ST depression induced by exercise relative to rest
- `slope`: the slope of the peak exercise ST segment
- `ca`: number of major vessels (0 - 3) colored by fluoroscope
- `thal`: thalassemia (blood disorder) 0 = normal; 1 = fixed defect; 2 = reversible defect
- `target`: 0 = less chance of heart attack; 1 = more chance of heart attack

Of the 14 attributes, the binary "target" variable is considered the response variable, dependent on the remainder of the attributes. For the remainder of this report, all models considered using the 13 predictor variables to perform prediction on "target".

II. Classification Methods Explored

Decision Trees

Decision trees are useful in classification to improve prediction accuracy. We explored bagging and random forest models. Figure 1 shows a classification tree used to predict whether someone is more or less likely to have a heart attack based on certain thresholds for each of the 14 predictors.

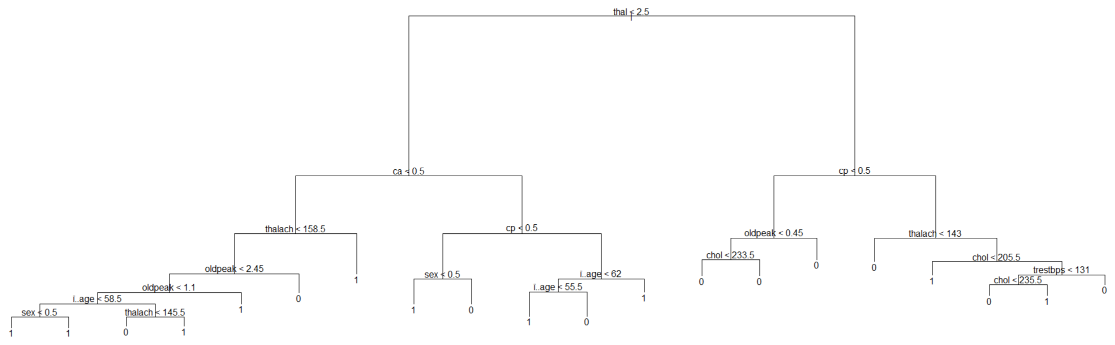


Figure 1: Classification tree for `heart` data-set.

Next, we decided to prune the classification tree, shown in figure 2. As we can see, the pruned tree reduced the number (add more here)

We can also look at the prediction accuracy (??) for both the original (see figure 3) and pruned trees and compare the outputted results.

Bagging:

Next, we used bagging, which is useful to reduce the variance and improve the prediction accuracy. The results of bagging are shown in figure 4. Given these results, we can conclude that thalassemia and chest pain type of the two most important attributes.

Random Forest:

Finally, random forest was used to improve upon bagging and further improve prediction accuracy, shown in figure 5. We can see that after performing random forest, the variables of importance have changed. While

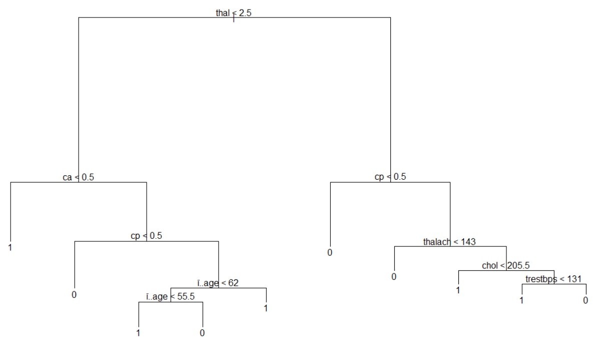
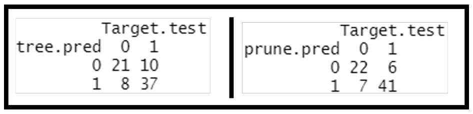


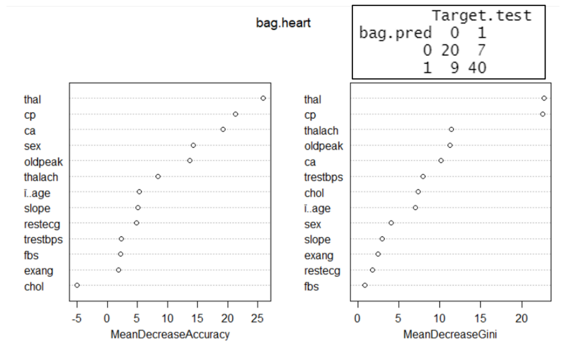
Figure 2: Pruned classification tree.



```
          Target.test
tree.pred  0  1
        0 21 10
        1  8 37
```

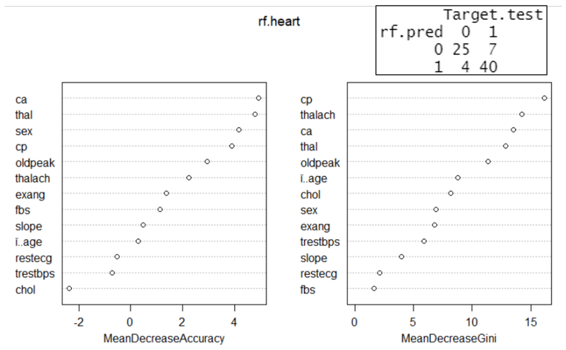
```
           Target.test
prune.pred  0  1
         0 22  6
         1  7 41
```

Figure 3: Prediction accuracy of the original tree (left) compared to the pruned tree (right).



```
        Target.test
bag.pred  0  1
       0 20  7
       1  9 40
```

Figure 4: The side-by-side plots show the variables of importance in heart attack predictability. The table in the top-right shows the prediction accuracy for bagging.



```
       Target.test
rf.pred  0  1
      0 25  7
      1  4 40
```

Figure 5: The side-by-side plots show the variables of importance in heart attack predictability. The table in the top-right shows the prediction accuracy for random forest.

thalassemia and chest pain type are still important, the random forest analysis shows that maximum heart rate achieved and number of major vessels are also of importance.

Random Forest Plots

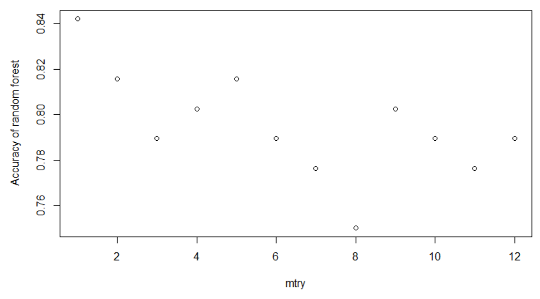
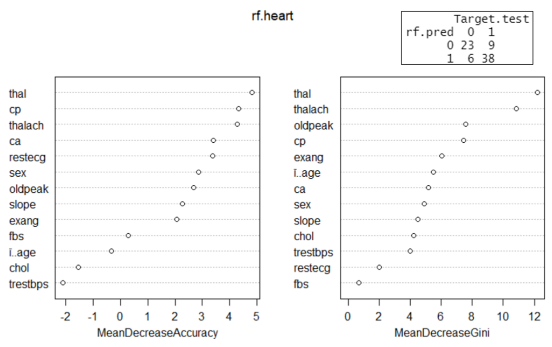


Figure 6: Accuracy plot.



```
        Target.test
rf.pred  0  1
      0 23  9
      1  6 38
```

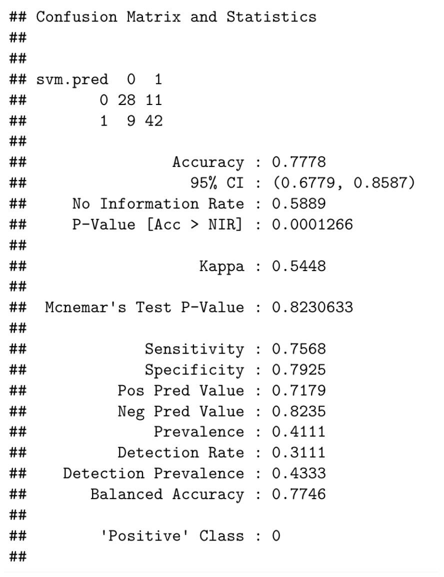
Figure 7: Random Forest variables of importance.

Support Vector Machines

SVM is a supervised classification method that separates data using *hyperplanes*, which act as a decision boundary between the various classes. Here we use a classifier for predicting whether a patient is suffering from any heart disease or not.

Model Training and Accuracy:

We first train a linear classifier with the help of a *train control* method that uses *repeated cross-validation*, and then calculate prediction accuracy (77.8%) using a *confusion matrix.*



```
## Confusion Matrix and Statistics
##
##
## svm.pred  0  1
##        0 28 11
##        1  9 42
##
##                Accuracy : 0.7778
##                  95% CI : (0.6779, 0.8587)
##     No Information Rate : 0.5889
##     P-Value [Acc > NIR] : 0.0001266
##
##                   Kappa : 0.5448
##
##  Mcnemar's Test P-Value : 0.8230633
##
##             Sensitivity : 0.7568
##             Specificity : 0.7925
##          Pos Pred Value : 0.7179
##          Neg Pred Value : 0.8235
##              Prevalence : 0.4111
##          Detection Rate : 0.3111
##    Detection Prevalence : 0.4333
##       Balanced Accuracy : 0.7746
##
##        'Positive' Class : 0
##
```

Figure 8: SVM Accuracy with Cost = 1

Choosing Different Costs:

In order to improve model performance, we play with *Cost (C)* values in our classifier. For this we define a grid with specific C values. We then train our model again using the new C values. Our prediction accuracy is the most (84.5%) when $C = 5$, reflected in the plot below.

Tuned Model:

Finally, we test the model for the same C values. We do this by using *predict* over the *tuned* training model and the testing data-set, and checking the accuracy using the *confusion matrix.* We note that this ends up giving a higher accuracy rate (78.9%).

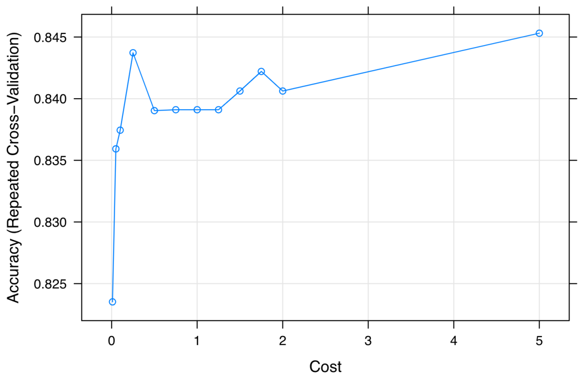


Figure 9: Accuracy Plot with Varying Costs

```
## Confusion Matrix and Statistics
##
##
## svm.pred.grid  0  1
##             0 28 10
##             1  9 43
##
##                Accuracy : 0.7889
##                  95% CI : (0.6901, 0.8679)
##     No Information Rate : 0.5889
##     P-Value [Acc > NIR] : 4.918e-05
##
##                   Kappa : 0.5658
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7568
##             Specificity : 0.8113
##          Pos Pred Value : 0.7368
##          Neg Pred Value : 0.8269
##              Prevalence : 0.4111
##          Detection Rate : 0.3111
##    Detection Prevalence : 0.4222
##       Balanced Accuracy : 0.7840
##
##        'Positive' Class : 0
##
```

Figure 10: Accuracy of the Tuned SVM Model

KNN

The K-nearest neighbors algorithm is a simple algorithm that closely follows the concept of the Bayes classifier. For a given data, with predictor variables, X, and a response variable, y with k classes, the Bayes classifier seeks to first establish a conditional probability of the classes of y given X. With a new observation, x_n, the prediction made is the class j with the highest conditional probability given x_n.

Establishing a conditional distribution for the data at hand may be impossible. Consequently, the KNN algorithm uses a frequentist approach to establish a conditional probability distribution for only subsets of the data. For a new observation x_n, the KNN algorithm finds the distance from x_n to each of the points in X, selecting the k nearest points. These are the k nearest neighbors of x_n. Let $y^{(kn)}$ be the classes of the k nearest neighbors of x_n. With this subset of data points, a conditional probability is calculated as in (1).

$$Pr(Y = j|X = x_n) = \frac{1}{k} \sum_{y_i \in y^{kn}} I(y_i = j) \tag{1}$$

In (1), I is the indicator function that evaluates to 1 if the condition holds, and 0 if otherwise. Thus, (1) determines a frequency, which also is the conditional probability given x_n for each class j of the response variable. The predicted class, therefore, is j such that Pr(Y=j|X=x_n) is maximum.

The KNN algorithm is applied to the normalized heart.csv data set, using an optimum value of $k = 12$. The confusion matrix obtained is as shown below, and the accuracy obtained is 0.789.

```
##          Target.test
## knn.heart  0  1
##         0 24 11
##         1  5 36
```

Figure 11: Results of KNN classfication with normalize data.

As k is a hyper-parameter, it must be determined a priori. Through a 5-fold cross-validation, different values of k are applied and the optimal is chosen. As observed from below, the best value of k is 5. This was used in the remainder of the KNN study.

Considering that the KNN algorithm works by finding the distance of the new data point from the k nearest points, it is likely that the effect of certain parameters which are on a higher scale will become dominant over others on a lower scale. This potentially affects the accuracy of prediction. It is therefore useful that the data is normalized prior to performing KNN. In this study, the Euclidean distance was used, and normalization applied as follows. Let X be the data set of parameters, and let X_{min} be the $1 \times p$ minimum vector of X. Also, let X_{max} be the $1 \times p$ column-wise maximum vector of X. Then, in this application, for each data point, X_i, normalization is performed according to (2).

$$X_{inorm} = \frac{X_i - X_{min}}{X_{max} - X_{min}} \tag{2}$$

Without normalization, the following prediction table was obtained for an optimal $k = 5$. The corresponding accuracy is 0.684.

```
##          Target.test
## knn.heart  0  1
##         0 19 14
##         1 10 33
```

Figure 12: Results of KNN prediction from unnormalized data.

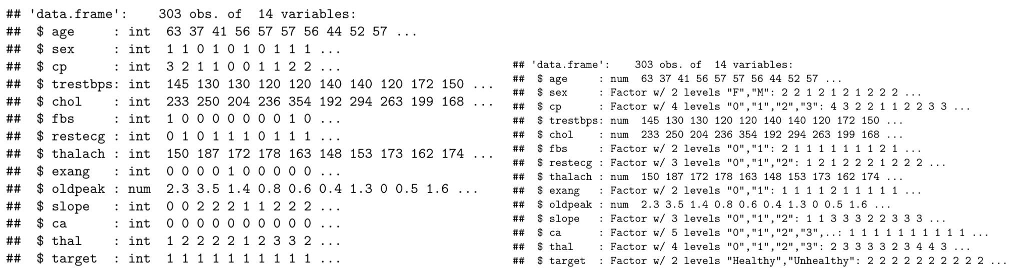
In an earlier section, the accuracy of the KNN algorithm applied to the normalized data was 0.789. Compared to the accuracy of the un-normalized data, one can see an improvement of over 15% accuracy. This underscores the need for normalization of data for especially distance-based algorithms such as the KNN.

Logistic Regression

LR predicts whether something is *True* or *False* instead of predicting something continuous. Here, we try to predict the probability that a person will get a heart disease or not.

Pre-processing Data:

For accurate predictions, we first process the raw data and convert a significant chunk of the variables into factors.



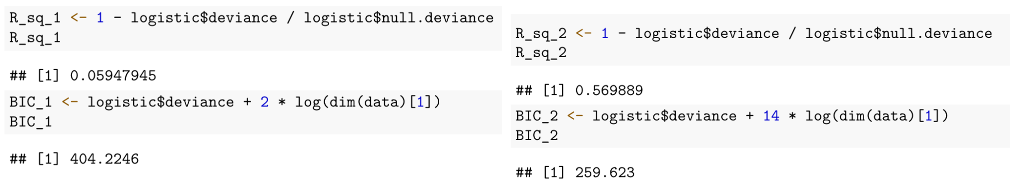
```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : int  1 1 0 1 0 1 0 1 1 1 ...
##  $ cp      : int  3 2 1 1 0 0 1 1 2 2 ...
##  $ trestbps: int  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 1 0 ...
##  $ restecg : int  0 1 0 1 1 1 0 1 1 1 ...
##  $ thalach : int  150 187 172 178 163 148 153 173 162 174 ...
##  $ exang   : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slope   : int  0 0 2 2 2 1 1 2 2 2 ...
##  $ ca      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ thal    : int  1 2 2 2 2 1 2 3 3 2 ...
##  $ target  : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : Factor w/ 2 levels "F","M": 2 2 1 2 1 2 1 2 2 2 ...
##  $ cp      : Factor w/ 4 levels "0","1","2","3": 4 3 2 2 1 1 2 2 3 3 ...
##  $ trestbps: num  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : num  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 2 1 ...
##  $ restecg : Factor w/ 3 levels "0","1","2": 1 2 1 2 2 2 1 2 2 2 ...
##  $ thalach : num  150 187 172 178 163 148 153 173 162 174 ...
##  $ exang   : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slope   : Factor w/ 3 levels "0","1","2": 1 1 3 3 3 2 2 3 3 3 ...
##  $ ca      : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ thal    : Factor w/ 4 levels "0","1","2","3": 2 3 3 3 3 2 3 4 4 3 ...
##  $ target  : Factor w/ 2 levels "Healthy","Unhealthy": 2 2 2 2 2 2 2 2 2 2 ...
```

Figure 13: Raw (left) and Processed (right) Data

Comparing Models:

We now compared the R^2, `AIC`, and `BIC` values of the two models - one with only age as the independent variable and another with all the variables. The second model is the better model since it has a higher R^2 and a lower `BIC`, which is an indicator of a better fit. We note that since the median age in our data-set is 55, it makes sense why it *is not* a statistically significant variable in our complex model (that covers all variables). With an `AIC` value of 225.6 (versus 396.8), the second model is again, a better fit.



```
R_sq_1 <- 1 - logistic$deviance / logistic$null.deviance
R_sq_1

## [1] 0.05947945

BIC_1 <- logistic$deviance + 2 * log(dim(data)[1])
BIC_1

## [1] 404.2246
```

```
R_sq_2 <- 1 - logistic$deviance / logistic$null.deviance
R_sq_2

## [1] 0.569889

BIC_2 <- logistic$deviance + 14 * log(dim(data)[1])
BIC_2

## [1] 259.623
```

Figure 14: Simple (left) and Complex (right) Models

Predicting Probability of Heart Disease:

We plot the probability of predicting whether a person in our data-set has a heart disease. The upper-right portion of the logistic curve (cyan) shows the probability a person *will* get a heart disease. The bottom-right portion of the logistic curve (light orange) shows the probability a person *will not* get a heart disease.

Model Accuracy:

Finally, we train the second model and predict its accuracy (77.8%) with a *confusion matrix*.

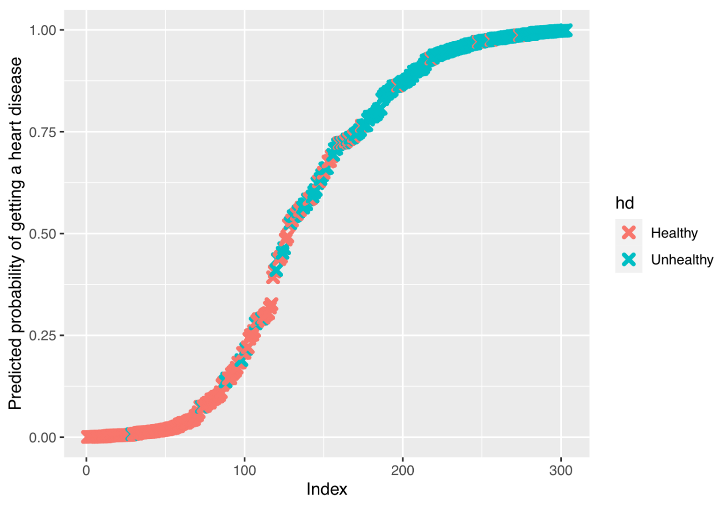


Figure 15: Predicted Heart Disease Probability

```
## Confusion Matrix and Statistics
##
##
## log.pred  0  1
##        0 28 11
##        1  9 42
##
##                Accuracy : 0.7778
##                  95% CI : (0.6779, 0.8587)
##     No Information Rate : 0.5889
##     P-Value [Acc > NIR] : 0.0001266
##
##                   Kappa : 0.5448
##
##  Mcnemar's Test P-Value : 0.8230633
##
##             Sensitivity : 0.7568
##             Specificity : 0.7925
##          Pos Pred Value : 0.7179
##          Neg Pred Value : 0.8235
##              Prevalence : 0.4111
##          Detection Rate : 0.3111
##    Detection Prevalence : 0.4333
##       Balanced Accuracy : 0.7746
##
##        'Positive' Class : 0
##
```

Figure 16: Logistic Regression Accuracy

III. Analysis

IV. Conclusion

V. Notes and References