Final Project

Introduction

Heart disease is the leading cause of death in the United States, causing about one in four deaths (CDC). It is also something people can greatly reduce their risk for through lifestyle changes and, in some cases, medicine. This is why predicting heart disease ahead of time is so important, and could have a real impact on increasing life expectancy across the world.

Using the UCI Machine Learning Repository dataset on heart disease, we can build a predictive model and examine important feature variables. The dataset contains 14 variables. Our 13 feature variables are: -Age

-Sex (0 = Female, 1 = Male)

-CP: Chest Pain Type (0 = Asymptomatic, 1 = Atypical Angina, 2 = Non-anginal pain, 3 = Typical Angina)

-trestbps: Resting Blood Pressure (in mm HG)

-chol: Serum Cholesterol (in mg/dl)

-fbs: Fasting blood sugar $>$ 120 mg/dl (0 = False, 1 = True)

-restecg: Resting electrocardiographic results (0 = Probable or definite left ventricular hypertrophy by Estes' criteria, 1 = Normal, 2 = ST-T wave abnormality)

-thalach: Max heart rate achieved

-exang: Exercised induced angina (0 = No, 1 = Yes)

-oldpeak: ST depression induced by exercise relative to rest

-slope: Slope of peak exercise in ST segment (0 = downsloping, 1 = flat, 2 = upsloping)

-ca: Number of major vessels colored by fluoroscopy

-thal: Thalium stress test result (1 = fixed defect, 2 = normal, 3 = reversible defect)

The 14th variable is our response, called "target". For this variable, a 0 represents heart disease and 1 for no heart disease. I am primarily interested in answering two questions: 1. How accurate of a model can we produce for prediciting heart disease? 2. What are the important feature variables?

Methods

First, the data was split 80/20 into a training set and a test set. For predictive modeling, I applied three different techniques.

1. KNN classification. I used 5-fold cross validation to find the best value of K.
2. Polynomial Kernel SVM. I used this because our classification is binary and non-linear. Cross-validation was used to determine the best values for cost and degree.
3. Random Forest. I included this because random forests is bootstrapping based, and the training set isn't particularly large. I also tested a range of values for 'mtry' to get the best possible model.

Each of these methods was evaluated using test misclassification rate.

Lastly, I created a pruned decision tree to interpret which variables were important in predicitng the target variable.

Results

```r
#Load in the data set
library(tidyverse)
```

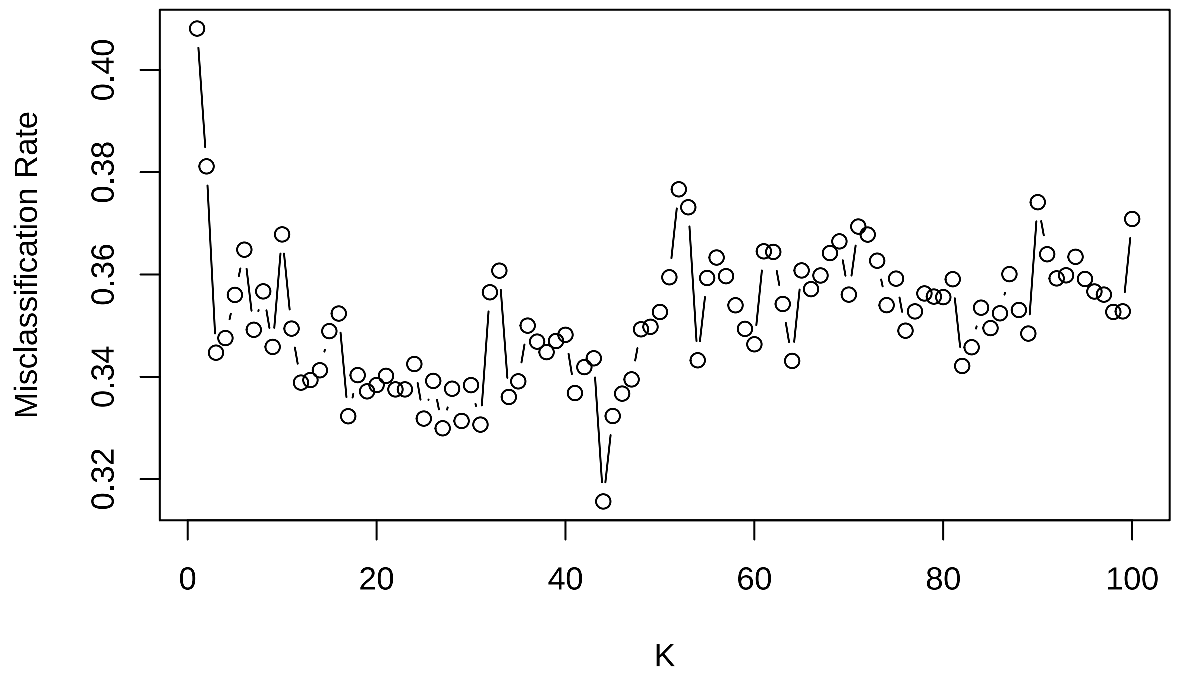
```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(readr)
heart <- read_csv("/Users/davidschultheiss/Downloads/heart.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   age = col_double(),
##   sex = col_double(),
##   cp = col_double(),
##   trestbps = col_double(),
##   chol = col_double(),
##   fbs = col_double(),
##   restecg = col_double(),
##   thalach = col_double(),
##   exang = col_double(),
##   oldpeak = col_double(),
##   slope = col_double(),
##   ca = col_double(),
##   thal = col_double(),
##   target = col_double()
## )
```

```r
#Removing some corrupted observations
heart = heart[-c(49, 93, 139, 164, 165, 252, 282), ]

#Factor Variable Transformation
heart$sex = factor(heart$sex)
heart$cp = factor(heart$cp)
heart$fbs = factor(heart$fbs)
heart$restecg = factor(heart$restecg)
heart$exang = factor(heart$exang)
heart$slope = factor(heart$slope)
heart$ca = factor(heart$ca)
heart$thal = factor(heart$thal)
heart$target = factor(heart$target)
```

```r
#Training & Test Data
set.seed(1)
train= sample(1:nrow(heart), .8*nrow(heart))
test = heart[-train, ]
training = heart[train, ]

#KNN
library(class)

set.seed(1)
folds = sample(1:5, nrow(heart), replace= T)
cv.misclass = matrix(0, nrow= 5, ncol= 100)

for (f in 1:5){
  cv.test = heart[folds == f, ]
  cv.train = heart[folds != f, ]
  for (k in 1:100) {
    knn.pred = knn(cv.train[ , -14], cv.test [ , -14], cv.train$target, k)
    cv.misclass[f, k] = mean(cv.test$target != knn.pred)
  }
}

cv.misrates = colMeans(cv.misclass)
plot(cv.misrates, xlab= 'K', ylab= 'Misclassification Rate', type= 'b')
```



```r
which.min(cv.misrates)
```

```
## [1] 44
```

```r
cv.misrates[44]
```

```
## [1] 0.3156216
```

```r
#SVM
library(e1071)
set.seed(1)

svm.tune = tune(svm, target~., data=training, kernel= 'polynomial', ranges=
                  list(cost= c(0.1, 1, 10, 100), degree= c(2, 3, 4, 5, 6)))
summary(svm.tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    10      2
##
## - best performance: 0.1990942
##
## - Detailed performance results:
##     cost degree     error dispersion
## 1    0.1      2 0.4695652 0.12854747
## 2    1.0      2 0.2286232 0.07964516
## 3   10.0      2 0.1990942 0.09507159
## 4  100.0      2 0.2414855 0.10828576
## 5    0.1      3 0.4695652 0.12854747
## 6    1.0      3 0.3086957 0.11617830
## 7   10.0      3 0.2115942 0.10382892
## 8  100.0      3 0.2250000 0.10811304
## 9    0.1      4 0.4695652 0.12854747
## 10   1.0      4 0.4527174 0.13003284
## 11  10.0      4 0.2876812 0.10790372
## 12 100.0      4 0.2164855 0.11266326
## 13   0.1      5 0.4695652 0.12854747
## 14   1.0      5 0.4653986 0.12507859
## 15  10.0      5 0.3557971 0.12267880
## 16 100.0      5 0.2668478 0.10669471
## 17   0.1      6 0.4695652 0.12854747
## 18   1.0      6 0.4695652 0.13003943
## 19  10.0      6 0.4273551 0.12722629
## 20 100.0      6 0.3173913 0.11251930
```

```r
svm.pred = predict(svm.tune$best.model, newdata= test)
mean(test$target != svm.pred)
```

```
## [1] 0.1
```

```r
#Random Forests
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

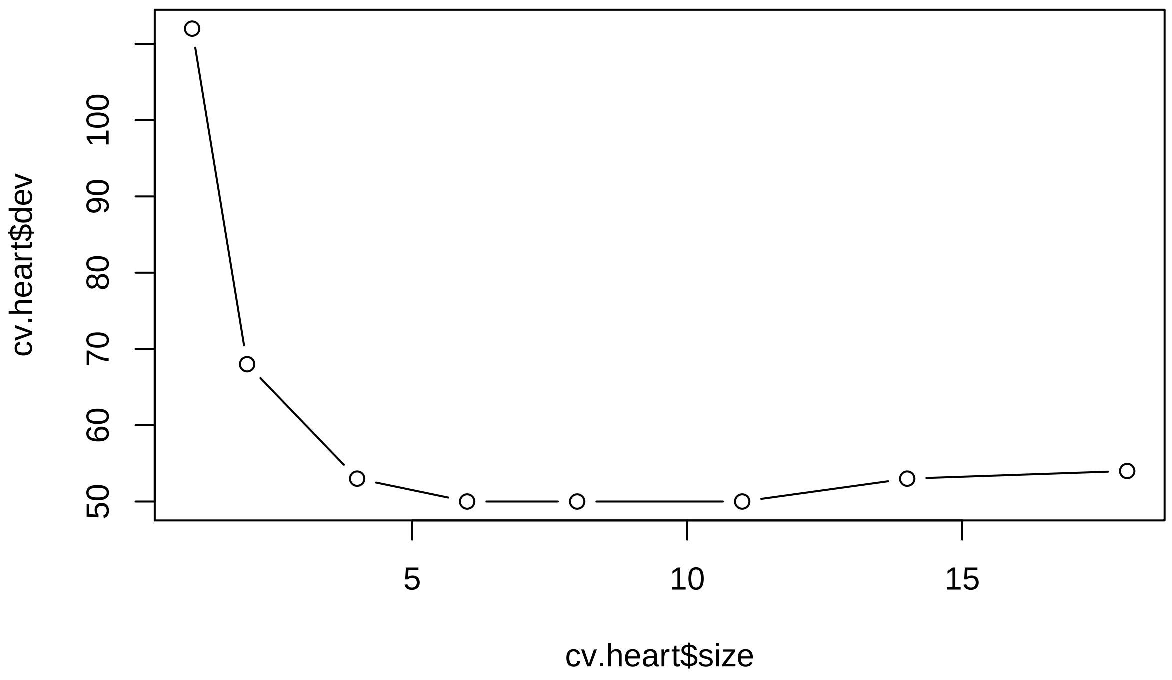
```r
set.seed(1)
rf.heart = randomForest(target~., data= training, mtry= 3, importance= F)
rf.pred = predict(rf.heart, newdata= test)
mean(test$target != rf.pred)
```

```
## [1] 0.1666667
```

```r
#Decision Tree
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

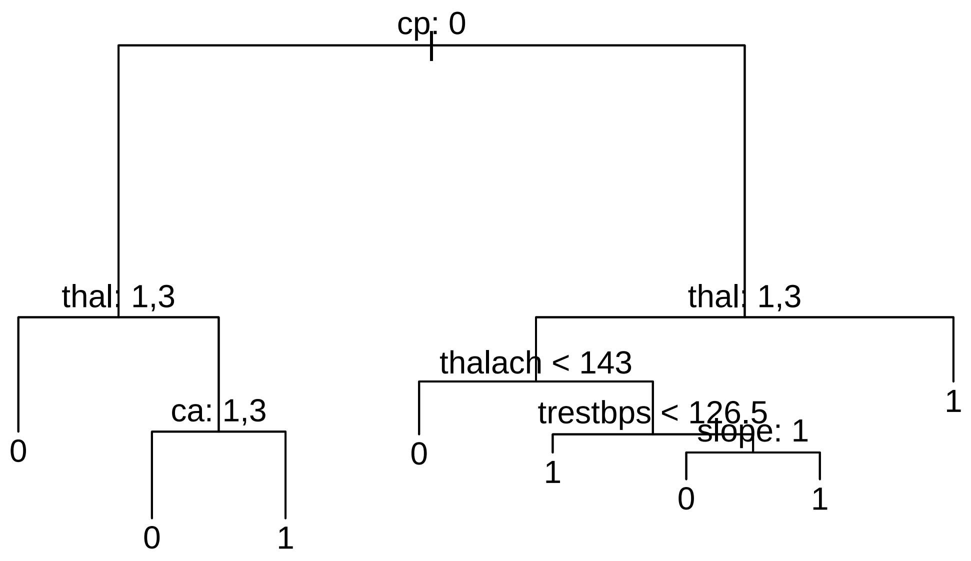
```r
tree.heart = tree(target~., data= training)

set.seed(1)
cv.heart = cv.tree(tree.heart, FUN= prune.misclass)
plot(cv.heart$size, cv.heart$dev, type= 'b')
```



```
which.min(cv.heart$size)
```

```
## [1] 8
```

```
prune.heart = prune.misclass(tree.heart, best= 8)
plot(prune.heart)
text(prune.heart, pretty=0)
```



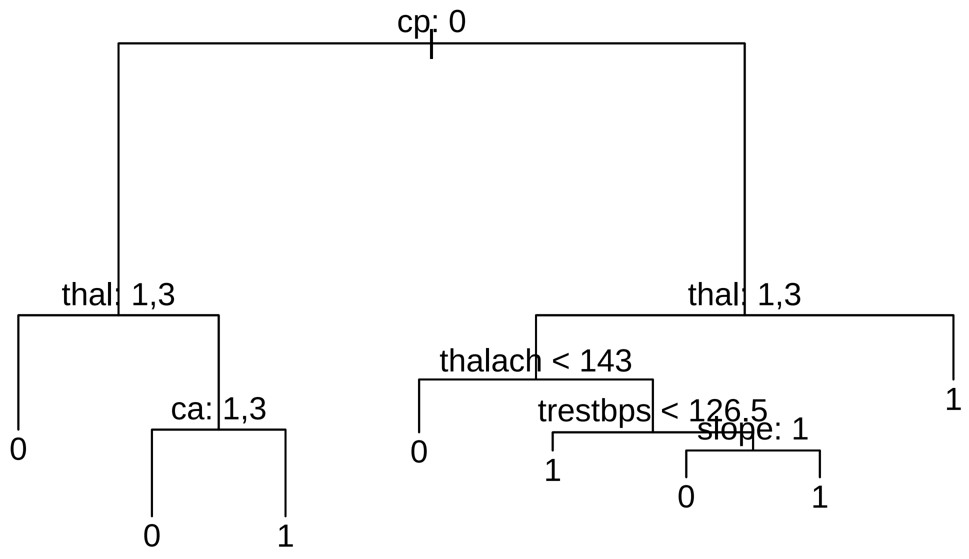
For KNN classification, K = 44 is found to be the best value for K. This gets a test misclassification rate of 31.6%. For SVM, cost of 10 with degree 2 provides the best model. This gets a test misclassification rate of 10%. For Random Forest, mtry = 3 was optimal value and gets a misclassification rate of 16.7%. For the descision tree, we find the optimal tree size to be 8. The pruned tree includes variables: cp, thal, thalach, ca, trestbps, and slope.

Discussion

Model accuracy for prediction was the best for SVM and worst for KNN. In the context of the data, SVM classified 90% of test observations correctly for heart disease. I was very impressed with this model performing as well as it did. However, even with 90% accuracy, a hospital would still need to run tests on the person to determine more accurately whether they had heart disease or not. 10% misclassification just isn't passable in medicine. Perhaps, with more data to train the model, prediction accuracy could rise to be a more viable diagnostic tool.

Pulling up the plot again for the decision tree:

```r
plot(prune.heart)
text(prune.heart, pretty=0)
```



The first node is whether someone's chest pain (cp) is asymptomatic. In other words, whether someone has chest pain caused by a diagnosed medical condition.

If chest pain is not caused by a diagnosed medical condition (asymptomatic), then you move to the left side of the tree. Another way of looking at it: if cp = 0, then you move left. If not, you move right. This process follows for the other nodes in the tree.

Overall, the tree gives us a good idea of the effects of some of these feature variables. Asymptomatic chest pain could signal heart disease when accompanied by an abnormal thalium stress test result (pretty much a heart blood flow test) or a fluoroscopy that colors 1 or 3 major vessels. It also suggests a maximum heart rate of under 143 and resting blood pressure over 126.5 mm HG could be signs of heart disease. Lastly, it shows that a flat peak exercise ST segment (something from a ECG stress test) could also be a sign of heart disease.

Conclusion

From our testing we find that the SVM model is the best for prediction of heart disease with the available data. The model yields a 10% misclassification rate, compared to 16.7% for Random Forests and 31.6% for KNN classification.

The decision tree gives us insights into important feature variables in the data; for instance, abnormal thalium stress test results are a significant red flag for heart disease.

Studies in the future could extend these models in different ways. The dataset used in this report contained 14 variables. These 14 were a subset of the original 76 in the dataset. I did not have access to the original dataset, so I could not experiment with the entire selection of variables. Including more variables could improve prediction accuracy. With this, attention should be paid to not overfit the data by creating an overly robust model.

The model could also be potentially improved by using more data. Having more data for the training set could increase predictive value.