Group 9 Paper

Henrique Magalhaes Rio , Oriana Meldrum, Jorie Alvis

12/11/2021

Introduction

In 2019 659,041 Americans died of heart disease. Four out of five heart disease deaths are due to heart attacks and strokes, and one-third of these deaths occur prematurely in people under 70 years of age. Heart failure is a common event caused by heart disease and this dataset we used contains 11 variables that can be used to predict a possible heart disease.

Motivation

People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors) need early detection and management so machine learning techniques are of great clinical interest for early detection.

For this project, we proposed to use the various machine learning methods learned this semester in CSU's DSCI 445 Machine Learning class that can be applied to a classification problem.

Methods

Logistic Regression

```
##
## Call:
## glm(formula = HeartDisease ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6823  -0.3298   0.1593   0.4214   2.6144
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.688185   1.814047  -0.379  0.70442
## Age               0.019083   0.018443   1.035  0.30081
## SexM              1.449234   0.363402   3.988 6.66e-05 ***
## ChestPainTypeATA -2.200799   0.415679  -5.294 1.19e-07 ***
## ChestPainTypeNAP -2.294483   0.378043  -6.069 1.28e-09 ***
## ChestPainTypeTA  -1.468140   0.617820  -2.376  0.01749 *
## RestingBP        -0.002964   0.008709  -0.340  0.73361
## Cholesterol      -0.004616   0.001424  -3.241  0.00119 **
## FastingBS1        1.218883   0.374243   3.257  0.00113 **
## RestingECGNormal -0.125716   0.359027  -0.350  0.72622
## RestingECGST     -0.306034   0.479711  -0.638  0.52350
## MaxHR             0.001609   0.006653   0.242  0.80896
```

```
## ExerciseAnginaY   0.953829   0.337967   2.822  0.00477 **
## Oldpeak           0.315591   0.165962   1.902  0.05723 .
## ST_SlopeFlat      1.470516   0.565530   2.600  0.00932 **
## ST_SlopeUp       -1.189753   0.597455  -1.991  0.04644 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 754.15  on 550  degrees of freedom
## Residual deviance: 335.54  on 535  degrees of freedom
## AIC: 367.54
##
## Number of Fisher Scoring iterations: 6
```

```
##          True
## predicted   0   1
##         0 136  20
##         1  35 176
```

```
## [1] "Accuracy : 0.850136239782016"
```

We started by fitting a logistic regression model with all the variables, and from the results above we see that there are a few non-significant variables and important ones, it is worth noting that SexM variable is is statistically significant and positive, which interesting as it means that on average men are more likely to get heart disease than women given that the other variables are the same, also, ST_SlopeFlat and ST_SlopeUp are statistically significant and the show different signs which means that people with ST_SlopeFlat have a higher Likelihood of having heart Disease when compared to those that have ST_SlopeUp, and this is consistent with science as ST_Slope is a important factor in the diagnosis of a heart Disease. Overall, for a simple method that also allows for inference, it performed relatively well with an accuracy of 0.85.

LDA

Our next method for our binary classification problem was Linear Discriminate Analysis (LDA). LDA is used to find a linear combination of factors to sperate 2 or more classes. Typically, LDA uses only numeric predictors but through dummy variables, LDA can use categorical variables. This allows for limited inferences to be made from LDA.

```
LDA_model = lda(HeartDisease~., data=train)
LDA_model
```

```
## Call:
## lda(HeartDisease ~ ., data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.4337568 0.5662432
##
## Group means:
##        Age      SexM ChestPainTypeATA ChestPainTypeNAP ChestPainTypeTA
## 0 50.34310 0.6359833       0.37656904        0.3305439      0.05020921
## 1 55.67308 0.8878205       0.04807692        0.1217949      0.04166667
##   RestingBP Cholesterol FastingBS1 RestingECGNormal RestingECGST    MaxHR
## 0  130.1088    225.8577  0.1046025        0.6359833    0.1506276 150.2050
## 1  133.7147    178.8462  0.3205128        0.5769231    0.1955128 127.7276
##   ExerciseAnginaY   Oldpeak ST_SlopeFlat ST_SlopeUp
```

```
## 0        0.125523 0.3974895    0.1841004  0.7782427
## 1        0.625000 1.2467949    0.7628205  0.1506410
##
## Coefficients of linear discriminants:
##                              LD1
## Age                0.0097004227
## SexM               0.6347331404
## ChestPainTypeATA -1.2381070900
## ChestPainTypeNAP -1.1851491518
## ChestPainTypeTA  -0.6753130586
## RestingBP        -0.0009882450
## Cholesterol      -0.0021443061
## FastingBS1        0.5252252999
## RestingECGNormal -0.0355744966
## RestingECGST     -0.0835470224
## MaxHR             0.0004668529
## ExerciseAnginaY   0.5605434782
## Oldpeak           0.1780991082
## ST_SlopeFlat      0.7064569196
## ST_SlopeUp       -0.8582220592
```

```
preds = predict(LDA_model, test)
overall_LDA = sum(ifelse(preds$class == test$HeartDisease, 1, 0))/nrow(test)

table(preds$class, "True"=test$HeartDisease)
```

```
##    True
##       0   1
##   0 138  20
##   1  33 176
```

The LDA model achived a test accuracy of 0.856. Even though we can't make strong infrences from our LDA, we can look at the means of each factor of each class. We see that the Resting BP factor has little class mean difference.

KNN

The next method we used was K-Nearest Neighbors. K-nearest Neighbors is useful for both regression and classification. An object will be classified by a vote of the nearest k neighbors. The object is assigned to the class most common to those k neighbors. For this project, k was selected by cross validation. The best accuracy comes from a k of 23 with and accuracy of 0.6878024. Technically both k=1 and k=4 have higher accuracy, but we are avoiding k=1 using as any outlines would create a skew and in certain types of classification problems, our research shower it was better to avoid using an odd number as this can tie votes. With an accuracy rate of only 0.6757493 had the lowest overall accuracy rate.

```
## [1] 0.6757493
```

	0	1
0	93	41
1	78	155

```
## [1] 0.3242507
```

LASSO

After KNN, we used the least absolute shrinkage and selection operator(LASSO). When using the LASSO method, it was important to select family = "binomial" for the classification problem. The tuning lambda value was selected through cross validation. LASSO performed relatively well with an accuracy of 0.85.

```
train_matrix <- model.matrix(HeartDisease ~ ., data = train)
test_matrix <- model.matrix(HeartDisease ~ ., data = test)
grid <- 10^seq(10, -2, length=100)
lasso <- glmnet(train_matrix, train$HeartDisease, alpha = 1, lambda = grid, thresh = 1e-12, family = "bi
#plot(lasso)
cv_lasso  <- cv.glmnet(train_matrix, train[, "HeartDisease"], alpha=1,family = "binomial")
#plot(cv_lasso)
best_lambda_lasso <- cv_lasso$lambda.min
lasso_prediction <- predict(lasso, newx = test_matrix, s = best_lambda_lasso)
lasso_MSE <- mean((lasso_prediction - test[, "HeartDisease"])^2)
num_zeros <-cv_lasso$nzero[which.min(cv_lasso$cvm)]
# MSE values to use:
paste("Tuning Lambda",best_lambda_lasso)
```

```
## [1] "Tuning Lambda 0.00626936793624135"
```

```
#lasso_MSE
#paste("Number of zeros",num_zeros)
#Confusion Matrix
predicted_lasso <- predict(lasso, newx= test_matrix, s=best_lambda_lasso)
predicted_lasso <- ifelse(predicted_lasso>0.5,1,0)
confm_lasso<-table(predicted=predicted_lasso,True=test$HeartDisease)
confm_lasso
```

```
##          True
## predicted   0   1
##         0 147  30
##         1  24 166
```

```
#Over all correct:
overall_lasso <- (confm_lasso[1,1]+confm_lasso[2,2])/sum(confm_lasso)
kable(confm_lasso) #confusion matrix
```

	0	1
0	147	30
1	24	166

```
overall_lasso #over all correct
```

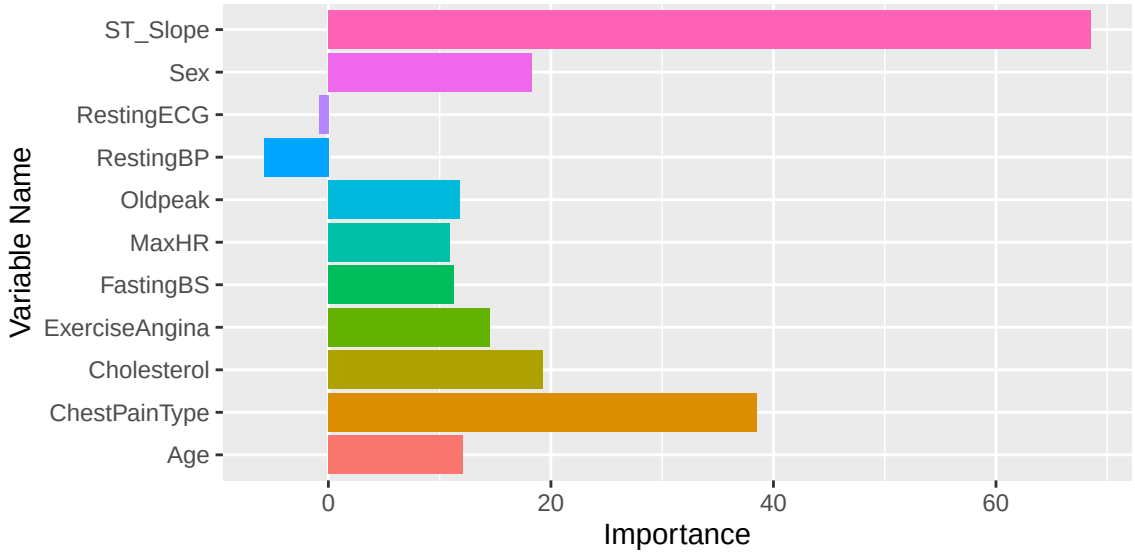
```
## [1] 0.852861
```

Bagging

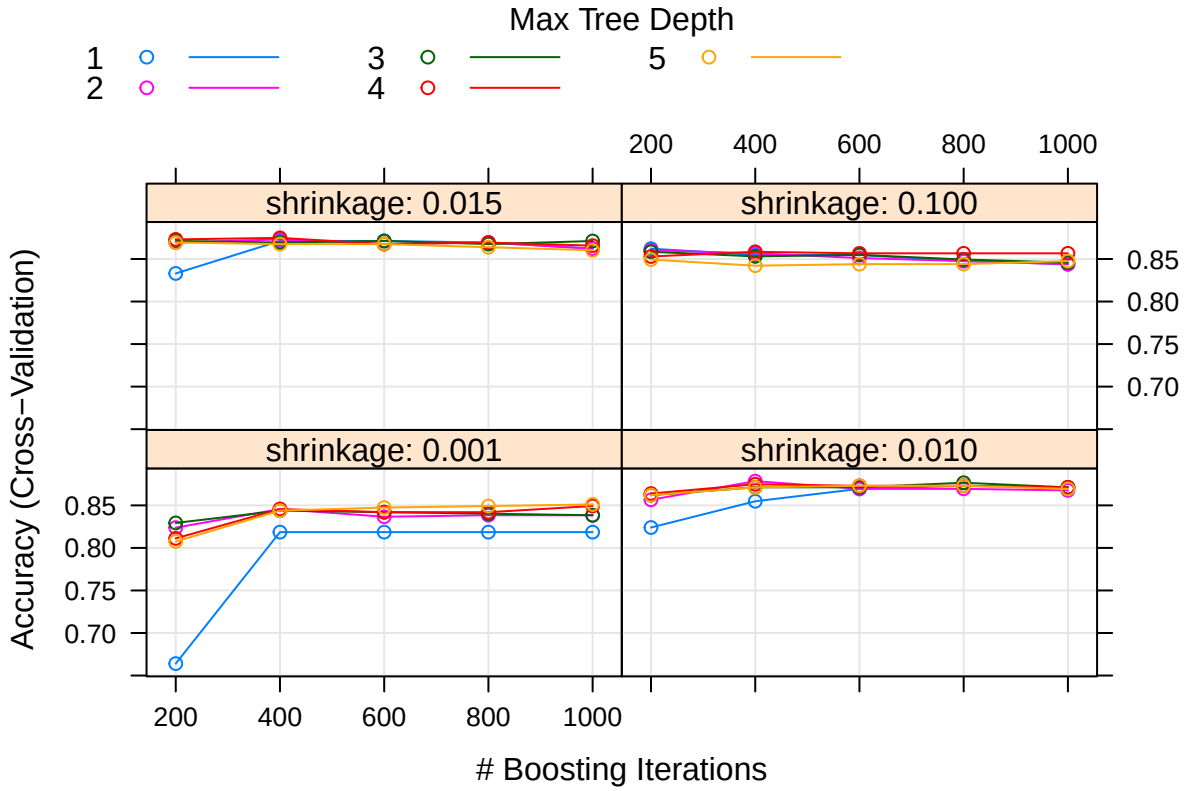
The bagging model was preformed with 500 trees and identified the most important variable as being ST_Slope followed by Chest Pain Type. Interestingly, resting ECG and Resting BP both had negative importance. This means if we remove those two features, we would improve model performance. As other methods found resting ECG and Resting BP to be among the least important variable, it would be worth taking the time to remove them and see if it improves the accuracy rates for our data. Bagging performed rather disappointingly with an accuracy of 0.84, the lowest of all methods except KNN.

	0	1
0	138	24
1	33	172

```
## [1] 0.8446866
```



Bosting



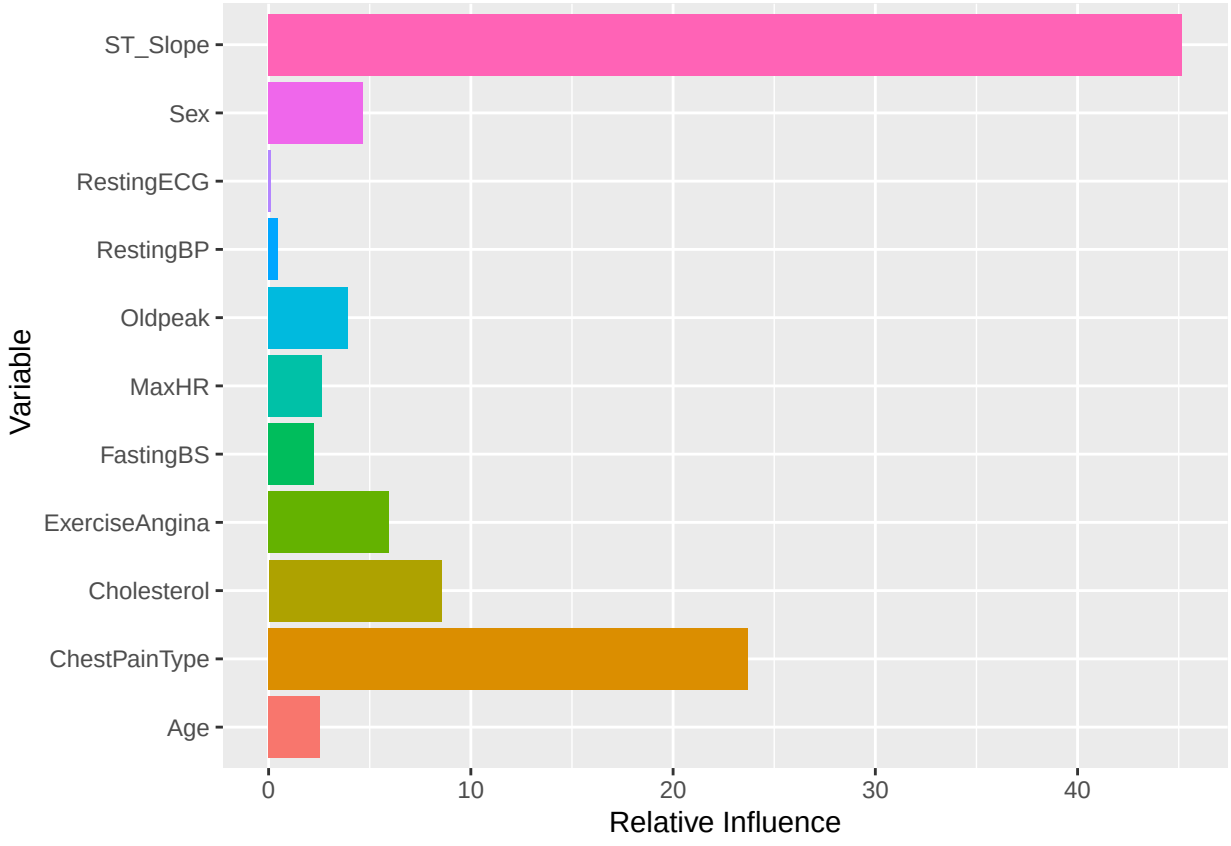

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 32     400                 2      0.01             10
```

In order to tune tree boosting method I used a 5 fold cross-validation on the training data set, from the plot it seems that there is a spike on the on the accuracy when using shrinkage with 0.01, 400 trees and a interaction depth of 2. It should also be noted that several other ranges of the tune parameters were tested but for simplicity sake on the range on the plot was left which included the best one overall.

```
##           actual
## predicted   0   1
##         0 141  15
##         1  30 181
```

```
## [1] "accuraccy: 0.877384196185286"
```

Fitting the model with the tuning parameters previously discussed, we get a much better accuracy when compared to the other methods but at the cost of having lost the ability of doing inference in this data set. We can take a look at the importance graph below, in which we can see that ST_Slope, ChestPainType, and Cholesterol are the most important variables, which is somewhat consistent with most of the models used so far. Also Consistent with the other models, we can see that RestingECG and RestingBP are the least important variables for boosting.



Random Forest

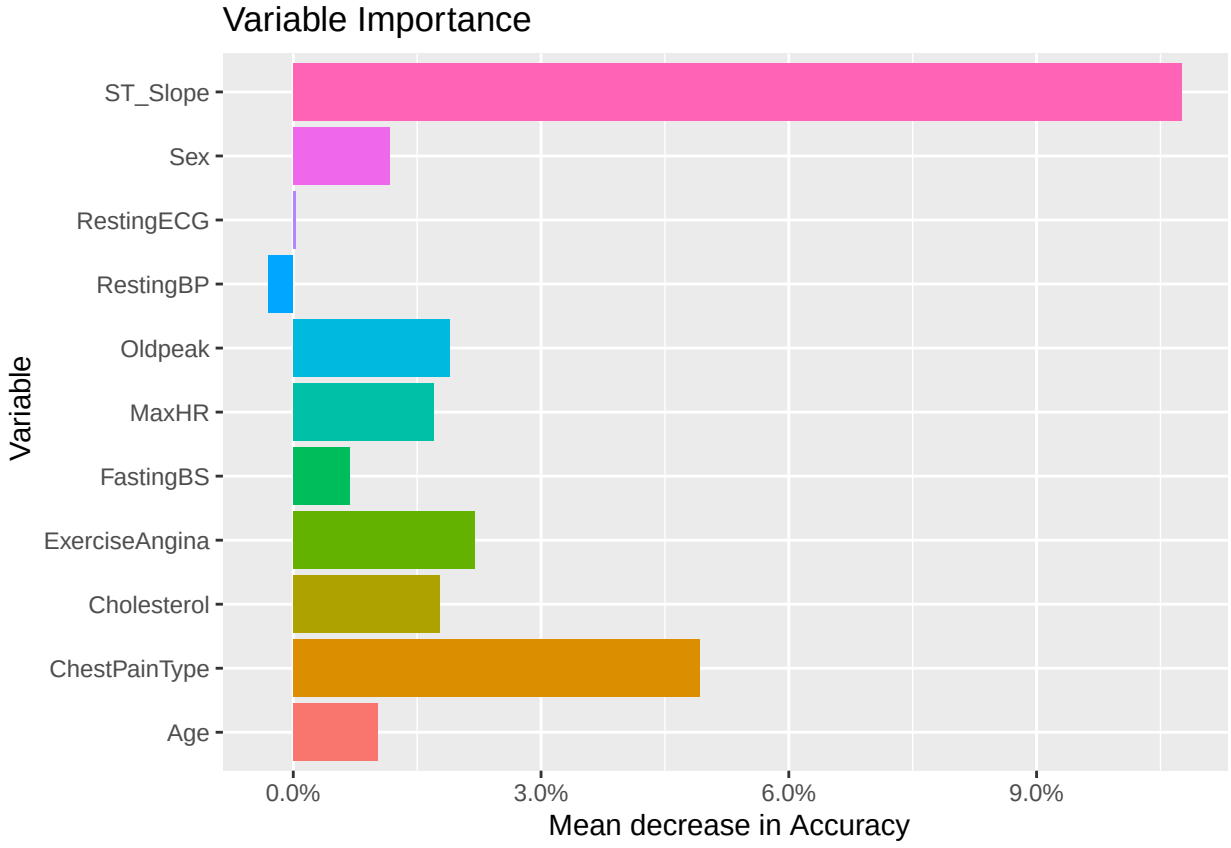
```
##           True
## predicted   0   1
##         0 141  17
##         1  30 179
```

```
## [1] " Test Accuracy : 0.871934604904632"
```

```
##
```

```
## Call:
##  randomForest(formula = as.factor(HeartDisease) ~ ., data = train,      mtry = sqrt(ncol(train)), imp
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 13.61%
## Confusion matrix:
##     0   1 class.error
## 0 194  45  0.18828452
## 1  30 282  0.09615385
```

Next, we fitted a Random Forest, in this case I chose the number of predictors randomly sampled at each iteration to be $m = \sqrt{p}$, as this was what was recommended by the class book. For random forests, the test accuracy relatively good at 0.8719 which is good, but not better than some of te other methods. Looking at the variable importance, we have some similar results to the other tree based methods, as ST_Slope, ChestPainType, and Cholesterol seem to be the most important variables. However, in random forests we got a negative value for the RestingBP, which is somewhat consistent with the other methos as RestingBP also did not seem to be a important variable in derterminining heart disease, however, in this case it means that for random forests is seems that including RestingBP decreases the accuracy of the method.



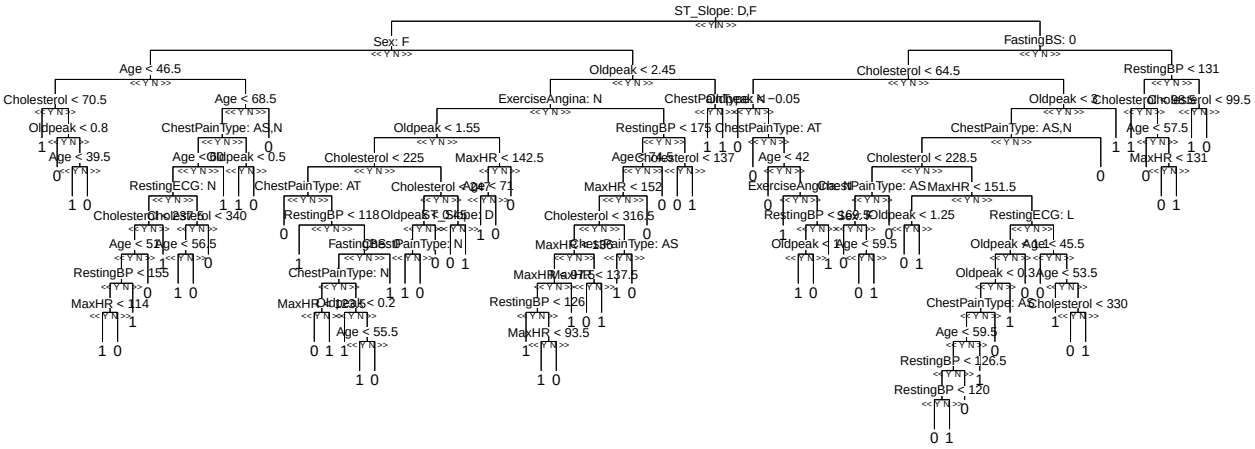
Representative Tree

While working on the random forests, I began to search for a way to visualize at least one of the trees in the ensemble, in order to more visualization and perhaps get a deeper insight into the method. While I found a way to do it in stack overflow, I also found a recommendation of a paper by Banerjee, et al. called "Identifying representative trees from ensembles" (2012), which was published in the journal of statistical

medicine. What we found really interesting is that the example application which was very similar to our application as it was classification problem using kidney cancer data, and there they discuss the importance of not losing the ability to do inference while still having high accuracy.

In the paper Banerjee, et al. discuss how to choose the most representative tree in an ensemble, which is done by finding the average distance between one tree and all other trees in the ensemble. Banerjee, et al. discusses several types of distance in this paper only one of them was used, as it was already implemented in a R package. This measure captures the similarities between the predictions, in that 2 trees are similar if they have the same predictions for all subjects, this is defined by Banerjee, et al as : $d_2(T_1, T_2) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_{1i} - \hat{y}_{2i})^2$, if the trees perfectly similar, the distance $d_2(T_1, T_2)$ is equal 0. For for finding the representative tree we used the package "reptree", and using it on this random forests we got the following tree:

```
## [1] "Constructing distance matrix..."
## [1] "Finding representative trees..."
```



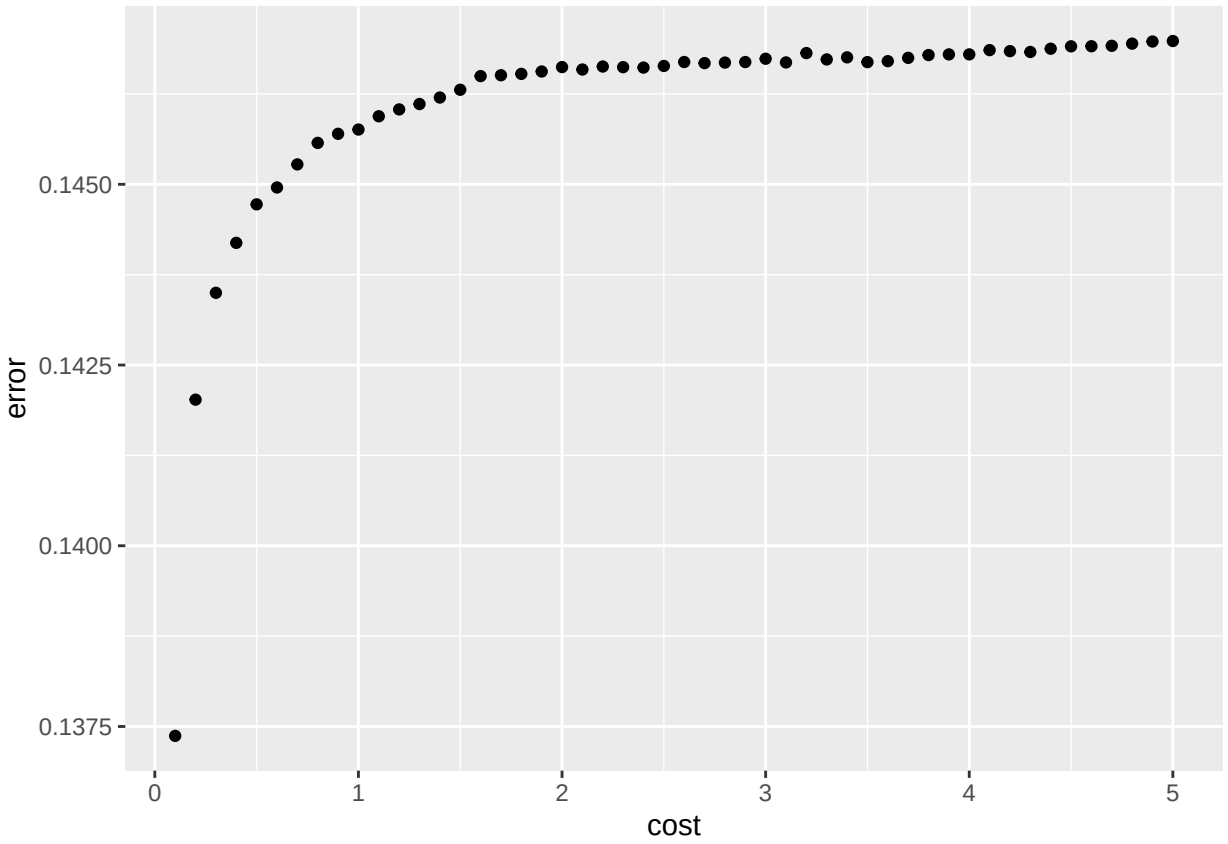
In the tree we can see ST_Slope is the root node, which was expected since most models considered it the most important variable, however, it is interesting that in the next nodes we have FastingBS and Sex:F both variables were not considered very important by the random forest model. This made us somewhat skeptical about this algorithm, however, it does make sense as we are looking for a representative tree not the most accuracate tree, if we had more time we would've liked to implement our own algorithm based of the paper in order to further validate the results.

SVM

The last model we used was SVM. We wanted to using the tuning method to best optimize our SVM. We tested three kernals for our SVM: linear, polynomial, and radial. We tuned the cost parameter for each, the degree parameter for the polynomial kernal, and the gamma parameter for the radial kernal.

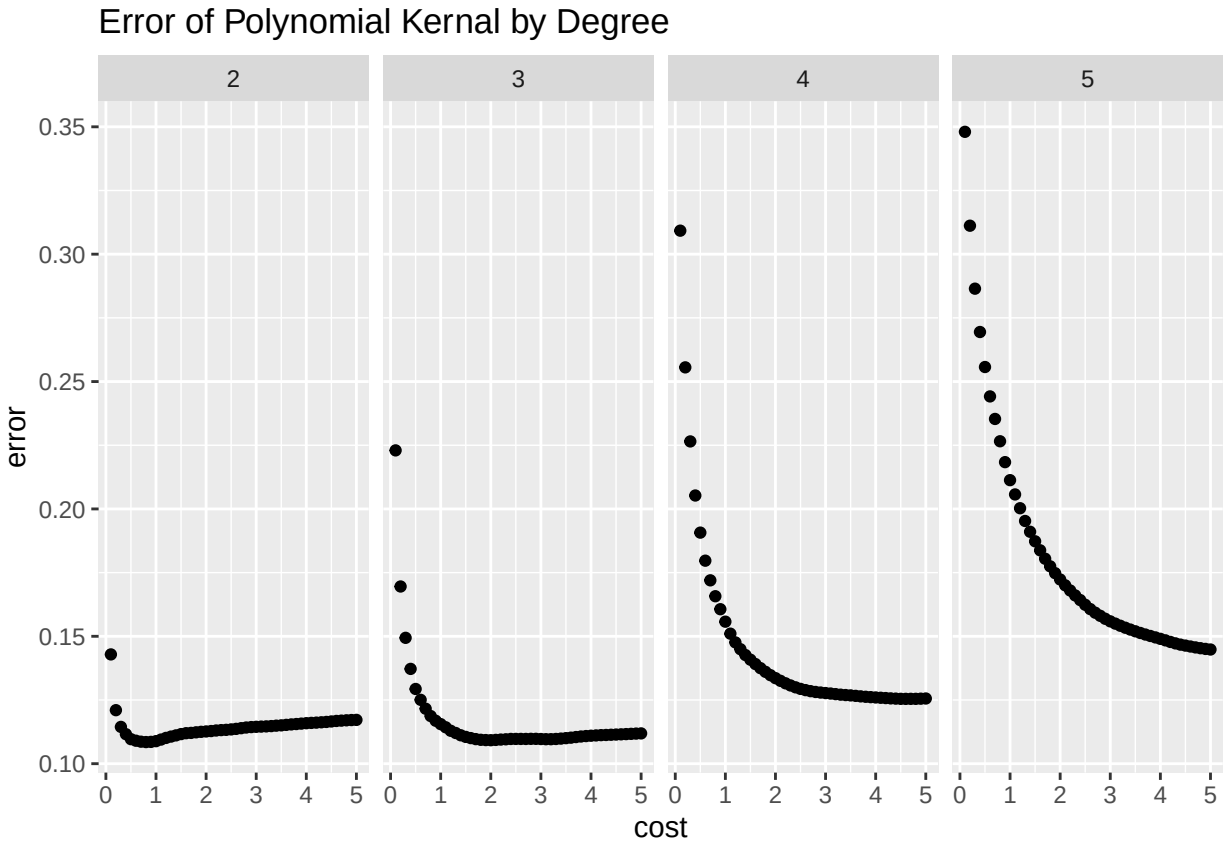
```
svm_linear_cv = tune(svm, HeartDisease~., scale=TRUE, kernel="linear", data=train,
                     ranges = list(cost = seq(0.1, 5, by = 0.1)))

ggplot(data=svm_linear_cv$performances) + geom_point(aes(x=cost, y=error))
```




```
svm_poly_cv = tune(svm, HeartDisease~., scale=TRUE, kernel="polynomial", data=train,
                   ranges = list(cost = seq(0.1, 5, by = 0.1),
                                 degree = c(2,3,4, 5)))

ggplot(data=svm_poly_cv$performances) + geom_point(aes(x=cost, y=error)) +
  facet_grid(~degree) + labs(title="Error of Polynomial Kernal by Degree")
```




```
train <- read.csv("train.csv")
test <- read.csv("test.csv")
train$ChestPainType <- as.factor(train$ChestPainType)
train$Sex <- as.factor(train$Sex)
train$FastingBS <- as.factor(train$FastingBS)
train$RestingECG <- as.factor(train$RestingECG)
train$ExerciseAngina <- as.factor(train$ExerciseAngina)
train$ST_Slope <- as.factor(train$ST_Slope)
test$ChestPainType <- as.factor(test$ChestPainType)
test$Sex <- as.factor(test$Sex)
test$FastingBS <- as.factor(test$FastingBS)
test$RestingECG <- as.factor(test$RestingECG)
test$ExerciseAngina <- as.factor(test$ExerciseAngina)
test$ST_Slope <- as.factor(test$ST_Slope)

SVM_radial = svm(HeartDisease~., data=train, kernel="radial", gamma=0.25, cost=0.4,type="C")

preds = predict(SVM_radial, test, type="class")
acc = sum(ifelse(preds == test$HeartDisease, 1, 0))/nrow(test)

table(preds,test$HeartDisease)
```

```
##
## preds   0   1
##     0 133  12
##     1  38 184
```

```
overall_SVM <- acc
```

We found that the radial kernel with cost of 0.4 and gamma parameter of 0.25 gave us the best training accuracy. Our testing accuracy for our SVM model was 0.864.

Conclusion

```
library(knitr)
df1 = data.frame(Method = c("Logistic Regression","LDA","KNN","LASSO","Bagging","Random Forest","Boosti

kable(df1)
```

Method	Accuracy
Logistic Regression	0.8501362
LDA	0.8555858
KNN	0.6757493
LASSO	0.8528610
Bagging	0.8446866
Random Forest	0.8719346
Boosting	0.8773842
SVM	0.8637602

We found that all of our models, except KNN, had an accuracy within a few points. It was interesting to note from LDA that our prior probabilities of our two classes closely follow the CDCs probability of getting heart disease. ST-Slope was shown to be the most important predictor in most of our models and we estimate this is the reason for the similar accuracy between models. So choosing the best model depends a lot on whether or not we value more inference or prediction, if we only care about inference, boosting is probably the best method otherwise, something like the logistic regression might be more useful for inference, however, if both are really important a using a random forest with a representative tree might be the best way.

Future Proposal

It would be interesting to re-try this project, removing the ST-Slope value. As we learned from industry professionals, the ST Slope is nearly always associated with some type of heart rhythm anomaly, and thus is frequently present in cases of heart disease. Many of our models indicated the ST Slope as the number one predictor for heart disease due to this strong association. We also would've liked to be able to make more inference from our models. A model like LDA may have one of the highest accuracy, but we could make very little inference about our data from this model.

References

Banerjee M, Ding Y, Noone AM. Identifying representative trees from ensembles. Stat Med. 2012 Jul 10;31(15):1601-16. doi: 10.1002/sim.4492. Epub 2012 Feb 3. PMID: 22302520.