# HW8_EnsembleLearning

Rocco Matarazzo

2023-10-31

## Part 1) kNN

### Read in Data

First we'll read in the data.

```
# Set FileName
fileName <-"C:/Users/rnm50/OneDrive/Documents/1A North Carolina State University/Fall 2023/ST558 Data S

# Read data
heartData <- read_csv(fileName)
```

### Create New Variable

Using the mutate function, we can create a new variable.

```
# Create new variable and remove others
heartDataCleaned <-
  heartData %>%
  # Create new factor variable
  mutate(FactorHeartDisease = factor(HeartDisease)) %>%
  # Do not select these three column
  select(-HeartDisease, -ST_Slope, -ExerciseAngina)
```

### Creating Dummy Variables

Using the dummyVars and predict functions, we can create categorical variables for our dataset. Finally, we can bind the entire dataset together to get our final working dataset.

```
# Parsing out the binary variables
dummies <- dummyVars(FactorHeartDisease ~ ChestPainType + Sex + RestingECG, data = heartDataCleaned)

# Getting the categorical variables spread out
newVars <- (predict(dummies, newdata = heartDataCleaned))

# Column Binding them all together
heartDataFinal <- cbind(heartDataCleaned %>% select(-ChestPainType, -Sex, -RestingECG), newVars)
```

### Split Data into Testing and Training

Here we will split the data into training and testing sets. 80% of the data will be in the training set. There are several ways to do this. I made sure to set a seed so we get the same training set each time we re-run this code.

```r
# set seed for rerunning purposes
set.seed(51)

# actually splitting the data
# the 0.8 at the end implies 80% of data into training set
train <- sample(1:nrow(heartDataFinal), size = nrow(heartDataFinal)*0.8)
# taking the rows from heartDataFinal not in the training set
test <- dplyr::setdiff(1:nrow(heartDataFinal), train)

# applying those to dataset
heartDataTrain <- heartDataFinal[train, ]
heartDataTest <- heartDataFinal[test, ]
```

### Train kNN Model

Here we will train the kNN model using several tuning parameters.

```r
# Creating tuning grid for values 1 to 40
tuningGrid <- expand.grid(k = 1:40)

# Set train control method
ctrl <- trainControl(
  # Repeated cross validation
  method = "repeatedcv",
  # 10 folds
  number = 10,
  # 3 repeats
  repeats = 3)

knnFit <- train(FactorHeartDisease ~ .,
                data = heartDataTrain,
                method = "knn",
                preProcess = c("center","scale"),
                trControl = ctrl,
                tuneGrid = tuningGrid)
knnFit
```

```
## k-Nearest Neighbors
##
## 734 samples
##  15 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (15), scaled (15)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 660, 661, 660, 662, 661, 660, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.7666620  0.5311781
##    2  0.7579158  0.5132130
##    3  0.7832967  0.5648270
##    4  0.7933190  0.5845477
##    5  0.8019329  0.6016055
##    6  0.7982671  0.5947699
```

```
##    7  0.8010382  0.6000979
##    8  0.7906595  0.5793915
##    9  0.7946885  0.5873187
##   10  0.7929172  0.5836413
##   11  0.7978350  0.5932182
##   12  0.7946325  0.5861935
##   13  0.7874004  0.5720781
##   14  0.7910160  0.5786451
##   15  0.7887641  0.5746297
##   16  0.7906218  0.5784049
##   17  0.7942004  0.5853785
##   18  0.7946757  0.5860970
##   19  0.7964713  0.5901365
##   20  0.7923430  0.5814801
##   21  0.7991553  0.5952659
##   22  0.7968784  0.5906282
##   23  0.7955209  0.5879458
##   24  0.7968969  0.5903944
##   25  0.7919234  0.5805968
##   26  0.8041848  0.6052180
##   27  0.8023829  0.6013663
##   28  0.8019324  0.6004345
##   29  0.8051166  0.6064762
##   30  0.8064866  0.6091813
##   31  0.8101211  0.6163879
##   32  0.8096768  0.6155580
##   33  0.8105777  0.6172418
##   34  0.8101461  0.6162133
##   35  0.8083320  0.6125578
##   36  0.8096768  0.6150104
##   37  0.8137866  0.6231925
##   38  0.8146506  0.6247855
##   39  0.8160140  0.6276113
##   40  0.8173715  0.6303233
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 40.
```

As the output states, k = 40 was the chosen tuning parameter.

### Check Model Performance

We can evaluate how the model performs by using a confusion matrix. We must first acquire predicions!

```
# First get predictions on test set
testPredictions <- predict(knnFit, newdata = heartDataTest)

# Calculate the confusion matrix and other performance metrics
confusionMatrix(testPredictions,  heartDataTest$FactorHeartDisease)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 51 19
##          1 24 90
```

```
##
##                Accuracy : 0.7663
##                  95% CI : (0.6984, 0.8254)
##     No Information Rate : 0.5924
##     P-Value [Acc > NIR] : 5.144e-07
##
##                   Kappa : 0.511
##
##  Mcnemar's Test P-Value : 0.5419
##
##             Sensitivity : 0.6800
##             Specificity : 0.8257
##          Pos Pred Value : 0.7286
##          Neg Pred Value : 0.7895
##              Prevalence : 0.4076
##          Detection Rate : 0.2772
##    Detection Prevalence : 0.3804
##       Balanced Accuracy : 0.7528
##
##        'Positive' Class : 0
##
```

# Part 2) Ensemble

I did not print any of the initial fits in this section because of the length of their output.

### Classification Tree

First we'll do a classification tree.

```
# Creating tuning grid for values 0 to 0.1
 tuningGrid_CT <- expand.grid(cp = seq(0, 0.1, by = 0.001))

# Setting ensemble controls
ctrl_ensemble <- trainControl(
  # Repeated cross validation
  method = "repeatedcv",
  # 5 folds for computational ease
  number = 5,
  # 3 repeats
  repeats = 3)

CT_Fit <- train(FactorHeartDisease ~ .,
              data = heartDataTrain,
              method = "rpart",
              preProcess = c("center","scale"),
              trControl = ctrl_ensemble,
              tuneGrid = tuningGrid_CT)
```

### Bagged Tree

Next, we'll do a bagged tree.

```
# No tuning parameter here
```

```
# Setting ensemble controls
ctrl_ensemble <- trainControl(
  # Repeated cross validation
  method = "repeatedcv",
  # 5 folds for computational ease
  number = 5,
  # 3 repeats
  repeats = 3)

BT_Fit <- train(FactorHeartDisease ~ .,
                data = heartDataTrain,
                method = "treebag",
                preProcess = c("center","scale"),
                trControl = ctrl_ensemble)
```

### Random Forest

Next, a random forest. Tuning parameter 15 was the best value.

```
# Creating tuning grid for values 1 to 15
 tuningGrid_RF <- expand.grid(mtry = seq(1, 15))

# Setting ensemble controls
ctrl_ensemble <- trainControl(
  # Repeated cross validation
  method = "repeatedcv",
  # 5 folds for computational ease
  number = 5,
  # 3 repeats
  repeats = 3)

RF_Fit <- train(FactorHeartDisease ~ .,
                data = heartDataTrain,
                method = "rf",
                preProcess = c("center","scale"),
                trControl = ctrl_ensemble,
                tuneGrid = tuningGrid_RF)
```

### Boosted Tree

And finally a boosted tree! As the output says, thefinal tuning parameters were n.trees = 200, interaction.depth = 1, shrinkage = 0.1 and n.minobsinnode = 10.

```
# Create tuning grid for boosted tree (several variables!)
tuningGrid_boostedTrees <- expand.grid(n.trees = c(25, 50, 100, 200),
                               interaction.depth = c(1,2,3),
                               shrinkage = c(0.1),
                               n.minobsinnode = c(10))

# Setting ensemble controls
ctrl_ensemble <- trainControl(
  # Repeated cross validation
  method = "repeatedcv",
  # 5 folds for computational ease
  number = 5,
```

```
  # 3 repeats
  repeats = 3)

BoostedTree_Fit <- train(FactorHeartDisease ~ .,
              data = heartDataTrain,
              method = "gbm",
              preProcess = c("center","scale"),
              trControl = ctrl_ensemble,
              tuneGrid = tuningGrid_boostedTrees)
```

### Testing Each Model

Here we'll test each model using the testing dataset.

```
# Save all predictions
CT_predict <- predict(CT_Fit, newdata = heartDataTest)
BT_predict <- predict(BT_Fit, newdata = heartDataTest)
RF_predict <- predict(RF_Fit, newdata = heartDataTest)
BoostedTree_predict <- predict(BoostedTree_Fit, newdata = heartDataTest)
```

Now we can run a confusion matrix on each method.

```
# Run confusion matrices
confusionMatrix(CT_predict,  heartDataTest$FactorHeartDisease)
```

### Classificaiton Tree Confusion Matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 51 24
##          1 24 85
##
##                Accuracy : 0.7391
##                  95% CI : (0.6694, 0.801)
##     No Information Rate : 0.5924
##     P-Value [Acc > NIR] : 2.263e-05
##
##                   Kappa : 0.4598
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.6800
##             Specificity : 0.7798
##          Pos Pred Value : 0.6800
##          Neg Pred Value : 0.7798
##              Prevalence : 0.4076
##          Detection Rate : 0.2772
##    Detection Prevalence : 0.4076
##       Balanced Accuracy : 0.7299
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(BT_predict,  heartDataTest$FactorHeartDisease)
```

### Bagged Tree Confusion Matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 58 18
##          1 17 91
##
##                Accuracy : 0.8098
##                  95% CI : (0.7455, 0.8638)
##     No Information Rate : 0.5924
##     P-Value [Acc > NIR] : 2.548e-10
##
##                   Kappa : 0.6069
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7733
##             Specificity : 0.8349
##          Pos Pred Value : 0.7632
##          Neg Pred Value : 0.8426
##              Prevalence : 0.4076
##          Detection Rate : 0.3152
##    Detection Prevalence : 0.4130
##       Balanced Accuracy : 0.8041
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(RF_predict,  heartDataTest$FactorHeartDisease)
```

### Random Forest Confusion Matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 56 13
##          1 19 96
##
##                Accuracy : 0.8261
##                  95% CI : (0.7634, 0.8779)
##     No Information Rate : 0.5924
##     P-Value [Acc > NIR] : 8.489e-12
##
##                   Kappa : 0.6353
##
##  Mcnemar's Test P-Value : 0.3768
##
##             Sensitivity : 0.7467
```

```
##             Specificity : 0.8807
##          Pos Pred Value : 0.8116
##          Neg Pred Value : 0.8348
##              Prevalence : 0.4076
##          Detection Rate : 0.3043
##    Detection Prevalence : 0.3750
##       Balanced Accuracy : 0.8137
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(BoostedTree_predict,  heartDataTest$FactorHeartDisease)
```

**Boosted Tree Confusion Matrix**

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 59 15
##          1 16 94
##
##                Accuracy : 0.8315
##                  95% CI : (0.7695, 0.8826)
##     No Information Rate : 0.5924
##     P-Value [Acc > NIR] : 2.54e-12
##
##                   Kappa : 0.6504
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7867
##             Specificity : 0.8624
##          Pos Pred Value : 0.7973
##          Neg Pred Value : 0.8545
##              Prevalence : 0.4076
##          Detection Rate : 0.3207
##    Detection Prevalence : 0.4022
##       Balanced Accuracy : 0.8245
##
##        'Positive' Class : 0
##
```

The classification tree was the least accurate of these four models, and slightly less accurate than the kNN model. The boosted tree was the most accurate model.