Assignment 9

Maksim Nikiforov

10/19/2021

Contents

Part 1: kNN

1.

Read in the ***`heart.csv`*** *data file.*

We remove the `ST_Slope` column as instructed.

```r
# Read in space-separated data using "read_csv()" function.
# Remove column ST_Slope, convert HeartDisease to a factor
heartData <- read.csv(file = "./heart.csv") %>% select(-ST_Slope)
```

2.

Create dummy columns corresponding to the values of ***`Sex`****,* ***`ChestPainType`****, and* ***`RestingECG`*** *for use in our kNN fit.*

We can follow the example from section 3.1 in the provided caret vignette to create dummy variables. `dummyVars` ignores integers, so we specify the full set of variables with the formula `~ .` and create a new data frame, `newHeartData`.

```r
# Use dummyVars() and predict() to create new columns.
dummies <- dummyVars( ~ ., data = heartData)
newHeartData <- predict(dummies, newdata = heartData)
newHeartData <- as_data_frame(newHeartData)
```

```
## Warning: `as_data_frame()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
newHeartData$HeartDisease <- as.factor(newHeartData$HeartDisease)
```

3.

Now split the data set you've created into a training and testing set. Use `p = 0.8`*.*

We can create train and test partitions with the `createDataPartition` function.

```
# p = 0.8 places 80% of observations in the training set
set.seed(50)
heartIndex <- createDataPartition(newHeartData$HeartDisease, p = 0.8, list = FALSE)
heartTrain <- newHeartData[heartIndex, ]
heartTest <- newHeartData[-heartIndex, ]
```

4.

Finally, train the kNN model. Use repeated 10 fold cross-validation, with the number of repeats being 3. You should also preprocess the data by centering and scaling. Lastly, set the `tuneGrid` *so that you are considering values of k of 1, 2, 3, . . . , 40.*

The `caret` package allows us to pre-process data, fit and tune models on the training set, and predict on the test set.

```
#
kNNFit <- train(HeartDisease ~ ., data = heartTrain,
               method = "knn",
               preProcess = c("center", "scale"),
               trControl = trainControl(method = "repeatedcv",
                                        number = 10, repeats = 3),
               tuneGrid = data.frame(k = 1:40))
```

5.

Check how well your model does on the test set using the confusionMatrix() function.

```
# See confusion matrix on test set (correct/incorrect predictions)
confusionMatrix(data = heartTest$HeartDisease, reference = predict(kNNFit, newdata = heartTest))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 60 22
##          1 18 83
##
##                Accuracy : 0.7814
##                  95% CI : (0.7145, 0.839)
##     No Information Rate : 0.5738
##     P-Value [Acc > NIR] : 2.949e-09
##
##                   Kappa : 0.556
##
##  Mcnemar's Test P-Value : 0.6353
##
##             Sensitivity : 0.7692
##             Specificity : 0.7905
##          Pos Pred Value : 0.7317
##          Neg Pred Value : 0.8218
##              Prevalence : 0.4262
##          Detection Rate : 0.3279
##    Detection Prevalence : 0.4481
##       Balanced Accuracy : 0.7799
##
##        'Positive' Class : 0
##
```

Part 2: Ensemble

We'll look at predicting the same heart disease variable in this section as well, just instead of using kNN we'll use the following methods:

1.

A classification tree (use `method = rpart`*: tuning parameter is* `cp`*, use values 0, 0.001, 0.002, . . . , 0.1).*

We use `seq()` to generate values from 0 to 0.1. With this method, we see an accuracy of approximately 80.33%.

```r
# Create tuning parameter by generating regular sequence
rpartParam <- seq(from = 0, to = 0.1, by = 0.001)

# Classification tree fit
rpartFit <- train(HeartDisease ~ ., data = heartTrain,
               method = "rpart",
               preProcess = c("center", "scale"),
               trControl = trainControl(method = "repeatedcv",
                                        number = 10, repeats = 3),
               tuneGrid = data.frame(cp = rpartParam))

# See confusion matrix on test set (correct/incorrect predictions)
confusionMatrix(data = heartTest$HeartDisease, reference = predict(rpartFit, newdata = heartTest))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 57 25
##          1 11 90
##
##                Accuracy : 0.8033
##                  95% CI : (0.7382, 0.8583)
##     No Information Rate : 0.6284
##     P-Value [Acc > NIR] : 2.307e-07
##
##                   Kappa : 0.5958
##
##  Mcnemar's Test P-Value : 0.03026
##
##             Sensitivity : 0.8382
##             Specificity : 0.7826
##          Pos Pred Value : 0.6951
##          Neg Pred Value : 0.8911
##              Prevalence : 0.3716
##          Detection Rate : 0.3115
##    Detection Prevalence : 0.4481
##       Balanced Accuracy : 0.8104
##
##        'Positive' Class : 0
##
```

2.

A bagged tree (use ***`method = treebag`****: no tuning parameter).*

Here, we see a slightly lower accuracy of 79.23%.

```r
# Bagged tree fit
baggedTreeFit <- train(HeartDisease ~ ., data = heartTrain,
               method = "treebag",
               preProcess = c("center", "scale"),
               trControl = trainControl(method = "repeatedcv",
                                        number = 10, repeats = 3))

# See confusion matrix on test set (correct/incorrect predictions)
confusionMatrix(data = heartTest$HeartDisease,
                reference = predict(baggedTreeFit, newdata = heartTest))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 57 25
##          1 13 88
##
##                Accuracy : 0.7923
##                  95% CI : (0.7263, 0.8487)
##     No Information Rate : 0.6175
##     P-Value [Acc > NIR] : 2.954e-07
##
##                   Kappa : 0.5743
##
##  Mcnemar's Test P-Value : 0.07435
##
##             Sensitivity : 0.8143
##             Specificity : 0.7788
##          Pos Pred Value : 0.6951
##          Neg Pred Value : 0.8713
##              Prevalence : 0.3825
##          Detection Rate : 0.3115
##    Detection Prevalence : 0.4481
##       Balanced Accuracy : 0.7965
##
##        'Positive' Class : 0
##
```

3.

A random forest (use ***`method = rf`****: tuning parameter is* ***`mtry`****, use vales of 1, 2, . . . , 15).*

Here, we see an accuracy of ~80.87%, on part with the classification tree method.

```r
# Random forest fit
rfFit <- train(HeartDisease ~ ., data = heartTrain,
               method = "rf",
               preProcess = c("center", "scale"),
               trControl = trainControl(method = "repeatedcv",
                                        number = 10, repeats = 3),
               tuneGrid = data.frame(mtry = 1:15))

# See confusion matrix on test set (correct/incorrect predictions)
confusionMatrix(data = heartTest$HeartDisease,
                reference = predict(rfFit, newdata = heartTest))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 59 23
##          1 13 88
##
##                Accuracy : 0.8033
##                  95% CI : (0.7382, 0.8583)
##     No Information Rate : 0.6066
##     P-Value [Acc > NIR] : 9.382e-09
##
##                   Kappa : 0.5977
##
##  Mcnemar's Test P-Value : 0.1336
##
##             Sensitivity : 0.8194
##             Specificity : 0.7928
##          Pos Pred Value : 0.7195
##          Neg Pred Value : 0.8713
##              Prevalence : 0.3934
##          Detection Rate : 0.3224
##    Detection Prevalence : 0.4481
##       Balanced Accuracy : 0.8061
##
##        'Positive' Class : 0
##
```

4.

A boosted tree (use `method = gbm`: tuning parameters are `n.trees`, `interaction.depth`, `shrinkage`, and `n.minobsinnode`, use all combinations of `n.trees` of 25, 50, 100, 150, and 200, `interaction.depth` of 1, 2, 3, 4, `shrinkage` = 0.1, and `n.minobsinnode` = 10; Hint: use `expand.grid()` to create your data frame for `tuneGrid`).

With this method, our accuracy with the test data set is ~79.78%. ß

```r
# See confusion matrix on test set (correct/incorrect predictions)
confusionMatrix(data = heartTest$HeartDisease,
                reference = predict(boostedTreeFit, newdata = heartTest))
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction  0  1
##          0 58 24
##          1 15 86
##
##                Accuracy : 0.7869
##                  95% CI : (0.7204, 0.8438)
##     No Information Rate : 0.6011
##     P-Value [Acc > NIR] : 7.088e-08
##
##                   Kappa : 0.5646
##
##  Mcnemar's Test P-Value : 0.2002
##
##             Sensitivity : 0.7945
##             Specificity : 0.7818
##          Pos Pred Value : 0.7073
##          Neg Pred Value : 0.8515
##              Prevalence : 0.3989
##          Detection Rate : 0.3169
##    Detection Prevalence : 0.4481
##       Balanced Accuracy : 0.7882
##
##        'Positive' Class : 0
##
```