Solutions Exercises Statistical Learning - Week 4

Julian D. Karch

February 23, 2024

1 1

```r
# Load data
library(readr)
student_mat <- read_delim("student-mat.csv",
    delim = ";", escape_double = FALSE, trim_ws = TRUE)
student_mat$y <- as.factor(student_mat$G1>11)
# Set the number of folds
k <- 5

# Calculate the size of each fold
fold_size <- floor(nrow(student_mat) / k)


# Define the folds
folds <- rep(1:k, each = fold_size)
#in case sample size n is not dividable by number of folds k
if (length(folds) < nrow(student_mat)) {
  folds[(length(folds) + 1):nrow(student_mat)] <-
    sample(1:k, nrow(student_mat) - length(folds))
}

# Initialize variables for storing results
predictions <- logical(nrow(student_mat))

# Shuffle the folds (equivalent to shuffling the data)
set.seed(123)
folds <- sample(folds)


# Perform k-fold cross-validation
for (i in 1:k) {
  # Determine the test indices for the current fold
```

```r
  test_indices <- folds == i

  # Create the training and testing sets
  train_set <- student_mat[!test_indices, ]
  test_set <- student_mat[test_indices, ]


  # Fit the model and make predictions
  fitted_model <- glm(y ~ studytime +  schoolsup +
                        romantic, data = train_set,
                        family = "binomial")
  probs <- predict(fitted_model, test_set, type = "response")
  classes <- rep(FALSE, nrow(test_set))
  classes[probs > .5] <- TRUE
  predictions[test_indices] <- classes
}

# Calculate the overall mean squared error
accuracy <-  mean(predictions == student_mat$y)
print(accuracy)
```

```
## [1] 0.6
```

```r
## Not part of solution:
## the same but with caret package
library(caret)
set.seed(123)
lr_model <- train(y ~ studytime + schoolsup + romantic,
                  data = student_mat,
                  method = "glm",
                  family = "binomial",
                  trControl = trainControl(method = "cv", number = 5))

print(lr_model$results)
```

```
##   parameter  Accuracy    Kappa AccuracySD    KappaSD
## 1      none 0.6150049 0.134515 0.03764649 0.08981714
```

```r
## close but not exactly the same
## because the caret package and our code do not
## use the same folds
```

2 2+3

```r
# Split data into training and test sets
library(caret)
set.seed(123)
train_idx <- sample(1:nrow(student_mat),
                    size = round(0.8*nrow(student_mat)),
                    replace = FALSE)
train_set <- student_mat[train_idx,]
test_set <- student_mat[-train_idx,]

# Define the training control
train_control <- trainControl(method = "cv", number = 10)

# cross-validate the knn model with candidate ks
set.seed(123) #set seed to make sure it is the same as for LDA
knn_model <- train(y ~ studytime + schoolsup + romantic,
                   data = train_set,
                   method = "knn",
                   tuneGrid = data.frame(k = c(1,3,5,7,9)),
                   trControl = train_control,
                   preProcess = c("center","scale"))

# Print the optimal k and corresponding accuracy
print(knn_model$bestTune)

##   k
## 5 9

print(knn_model$results)

##   k  Accuracy      Kappa AccuracySD    KappaSD
## 1 1 0.5726815 0.09404293 0.03151248 0.06302629
## 2 3 0.5759073 0.09962504 0.02178731 0.04628213
## 3 5 0.5791331 0.10994746 0.03614458 0.07116845
## 4 7 0.5822581 0.11734242 0.03880197 0.07665785
## 5 9 0.5822581 0.11734242 0.03880197 0.07665785

# cross-validate lda
lda_model <- train(y ~ studytime + schoolsup + romantic,
                   data = train_set,
                   method = "lda",
                   trControl = train_control)

print(lda_model$results)

##   parameter  Accuracy     Kappa AccuracySD   KappaSD
## 1      none 0.6041331 0.1361032 0.05193908 0.1152138
```

```r
# So lda was most accurate

# alternative using the resample function from the caret package,
# which was explained in the  reading material and
# offers more ways to compare models
resamp <- resamples(list(knn = knn_model, lda = lda_model))
summary(resamp)

##
## Call:
## summary.resamples(object = resamp)
##
## Models: knn, lda
## Number of resamples: 10
##
## Accuracy
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## knn 0.5161290 0.5670363 0.5871976 0.5822581 0.5937500 0.6451613    0
## lda 0.5483871 0.5703125 0.5937500 0.6041331 0.6129032 0.7187500    0
##
## Kappa
##            Min.    1st Qu.    Median      Mean  3rd Qu.      Max. NA's
## knn -0.02649007 0.09531391 0.1180556 0.1173424 0.161124 0.2302483    0
## lda  0.00913242 0.06204864 0.1185808 0.1361032 0.145724 0.3949580    0

summary(diff(resamp))

##
## Call:
## summary.diff.resamples(object = diff(resamp))
##
## p-value adjustment: bonferroni
## Upper diagonal: estimates of the difference
## Lower diagonal: p-value for H0: difference = 0
##
## Accuracy
##     knn    lda
## knn        -0.02187
## lda 0.3671
##
## Kappa
##     knn    lda
## knn        -0.01876
## lda 0.7101

# we again would conclude the lda was more accurate
```

```r
# but now also see that there is not significant difference
# retrain selected lda and get predictions on test set
fitted_lda <- train(y ~ studytime + schoolsup + romantic,
                    data = train_set, method = "lda",
                    preProcess = c("center","scale"),
                    trControl = trainControl(method = "none"))
cf <- confusionMatrix(predict(fitted_lda,test_set), test_set$y)
print(cf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    44   17
##      TRUE      8   10
##
##                Accuracy : 0.6835
##                  95% CI : (0.5692, 0.7837)
##     No Information Rate : 0.6582
##     P-Value [Acc > NIR] : 0.3653
##
##                   Kappa : 0.2354
##
##  Mcnemar's Test P-Value : 0.1096
##
##             Sensitivity : 0.8462
##             Specificity : 0.3704
##          Pos Pred Value : 0.7213
##          Neg Pred Value : 0.5556
##              Prevalence : 0.6582
##          Detection Rate : 0.5570
##    Detection Prevalence : 0.7722
##       Balanced Accuracy : 0.6083
##
##        'Positive' Class : FALSE
##
```

3 4

```r
## data generation copied from assignment
library(MASS)
set.seed(123)
n <- 200
```

```r
p <- 10^4
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
y <- rnorm(n)
data_set <- as.data.frame(cbind(y,X))
names(data_set) <- c("y", paste0("x",1:p))


# Calculate the correlation between y and each predictor
corr_y <- cor(data_set[, "y"], data_set[,-1])
# Select the 5 variables with highest absolute correlation with y
top_vars <- order(abs(corr_y), decreasing = TRUE)[1:5]
# Add 1 to top_vars to account for the fact that y is in the
# first column of data_set
top_vars <- top_vars + 1
# Create a new data set with only the columns of y and
# the selected top_vars
data_set_top <- data_set[,c(1,top_vars)]
# Fit a linear regression model using glm
# with the selected variables
lm_model <- glm(y ~ ., data = data_set_top)
# Perform 10-fold cross-validation using cv.glm
library(boot)
cv_model <- cv.glm(data_set_top, lm_model, K = 10)
# Extract the estimate of the mean squared error
# from the cv_model object
cv_model$delta[1]
```

```
## [1] 0.8315507
```

The estimated MSE is thus substantially lower than the lowest possible MSE. This is caused by the fact that we used cross-validation the wrong way (see reading material). The correct way would be to do the variable section within the cross-validation procedure.