HW9

Trey Davidson

2024-11-07

# Question 1

![HW9 PDF](data:application/pdf;base64,)

HW9 PDF

# Question 7

###################  
  
  
# A.)   
carData <- Auto  
  
n <- nrow(carData)  
  
carData$MileageDistance <- ifelse(carData$mpg > median(carData$mpg), 1, 0)  
  
  
indexes <- getTrainTestIndex(n)  
  
traindata <- carData[indexes$train\_index, ]  
testdata <- carData[indexes$test\_index, ]  
  
traindata$MileageDistance <- as.factor(traindata$MileageDistance)  
testdata$MileageDistance <- as.factor(testdata$MileageDistance)  
  
  
# B.)  
lin\_costs <- rep(0.0, 4)  
  
cost\_values <- c(1, 10, 100, 1000)  
  
  
for (c in cost\_values)   
{  
 m1 <- svm(MileageDistance ~ . - mpg, kernel = "linear", cost = c, scale = FALSE, data = traindata)  
   
   
 predictions <- predict(m1, newdata = testdata)  
   
 confusion\_matrix <- confusionMatrix(predictions, testdata$MileageDistance)  
   
 accuracy <- confusion\_matrix$overall['Accuracy']  
 lin\_costs[c] <- accuracy  
  
}  
  
which.max(lin\_costs)

## [1] 1

# Question 8

# A.)  
data <- OJ  
  
datatrain <- data[1:800,]  
datatest <- data[800:nrow(data), ]  
  
  
# B.)  
m2 <- svm(datatrain$Purchase ~ ., data = datatrain, kernel = "linear", cost = 0.01)  
  
predictions <- predict(m2, newdata = datatest)  
  
confusion\_matrix <- confusionMatrix(predictions, datatest$Purchase)  
  
  
accuracy\_m2 <- confusion\_matrix$overall['Accuracy']  
  
test\_error\_m2 <- 1 - accuracy\_m2  
  
#C.)   
test\_error\_m2

## Accuracy   
## 0.2066421

# D.)  
m3 <- tune(svm, Purchase ~ ., data = datatrain, kernel = "linear", ranges = list(cost = c(0.01, 0.1, 1, 5, 10)))  
  
best\_cost <- m3$best.parameters$cost  
  
m3f <- svm(Purchase ~ ., data = datatrain, kernel = "linear", cost = best\_cost)  
  
predictions <- predict(m3f, newdata = datatest)  
  
confusion\_matrix\_m3f <- confusionMatrix(predictions, datatest$Purchase)  
  
accuracy\_m3f <- confusion\_matrix\_m3f$overall['Accuracy']  
  
test\_error\_m3f <- 1 - accuracy\_m3f  
  
test\_error\_m3f

## Accuracy   
## 0.2066421

# Between tuned and untuend linnear. Tuned wins barley  
  
  
# F.)  
m4 <- svm(Purchase ~ ., data = datatrain, kernel = "radial", cost = 0.01)  
  
  
predictions\_m4 <- predict(m4, newdata = datatest)  
  
  
confusion\_matrix\_m4 <- confusionMatrix(predictions\_m4, datatest$Purchase)  
  
  
accuracy\_m4 <- confusion\_matrix\_m4$overall['Accuracy']  
  
  
test\_error\_m4 <- 1 - accuracy\_m4  
  
  
test\_error\_m4

## Accuracy   
## 0.4243542

m5 <- tune(svm, Purchase ~ ., data = datatrain, kernel = "radial",   
 ranges = list(cost = c(0.01, 0.1, 1, 5, 10)))  
  
  
best\_cost\_m5f <- m5$best.parameters$cost  
  
  
  
  
m5f <- svm(Purchase ~ ., data = datatrain, kernel = "radial", cost = best\_cost\_m5f)  
  
  
predictions\_m5f <- predict(m5f, newdata = datatest)  
  
confusion\_matrix\_m5f <- confusionMatrix(predictions\_m5f, datatest$Purchase)  
  
  
accuracy\_m5f <- confusion\_matrix\_m5f$overall['Accuracy']  
  
test\_error\_m5f <- 1 - accuracy\_m5f  
  
test\_error\_m5f

## Accuracy   
## 0.1808118

# g.)  
  
m6 <- svm(Purchase ~ ., data = datatrain, kernel = "polynomial", cost = 0.01, degree = 2)  
  
  
predictions\_m6 <- predict(m6, newdata = datatest)  
  
  
confusion\_matrix\_m6 <- confusionMatrix(predictions\_m6, datatest$Purchase)  
  
  
accuracy\_m6 <- confusion\_matrix\_m6$overall['Accuracy']  
  
  
test\_error\_m6 <- 1 - accuracy\_m6  
  
test\_error\_m6

## Accuracy   
## 0.4243542

m7 <- tune(svm, Purchase ~ ., data = datatrain, kernel = "polynomial",   
 ranges = list(cost = c(0.01, 0.1, 1, 5, 10), degree = 2))  
  
  
best\_cost\_m7f <- m7$best.parameters$cost  
  
  
  
m7f <- svm(Purchase ~ ., data = datatrain, kernel = "polynomial", cost = best\_cost\_m7f, degree = 2)  
  
  
predictions\_m7f <- predict(m7f, newdata = datatest)  
  
  
confusion\_matrix\_m7f <- confusionMatrix(predictions\_m7f, datatest$Purchase)  
  
  
accuracy\_m7f <- confusion\_matrix\_m7f$overall['Accuracy']  
  
  
test\_error\_m7f <- 1 - accuracy\_m7f  
  
test\_error\_m7f

## Accuracy   
## 0.1771218

# The tuned model using degree 2 perfmoeed the best with an error rate of 18%.