SOLUTIONS

**EXERCISE 1**

**Parameter combination table using the abalone data set**:



**Combination with the highest cross-validation accuracy**:

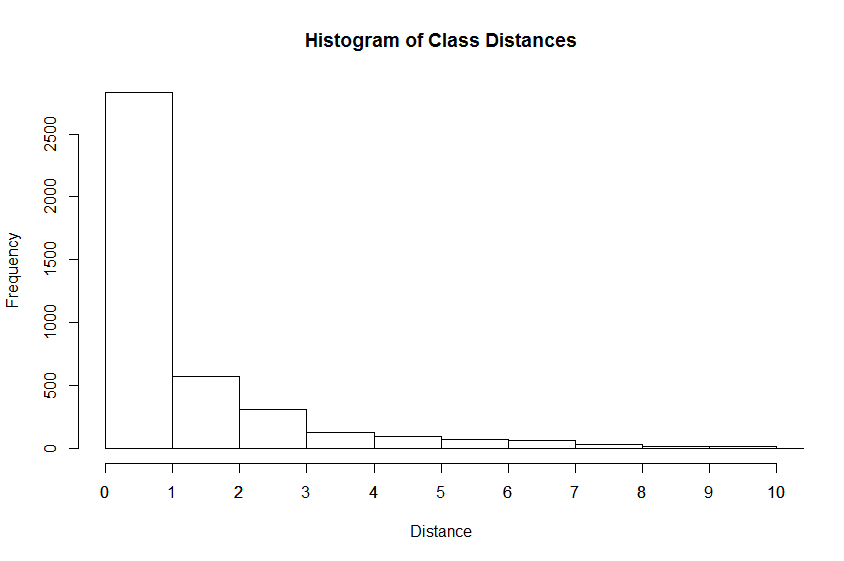


We see that the degree and cost values that produce the high cross-validation accuracy, 26.6, are 2 and 100, respectively.

**Average distance of the predicted class from the true class**:



**Histogram showing the frequency of how often a prediction is m away from the true number of rings:**



**EXERCISE 2**

**Table of best parameter combinations for each trained binary classifier (trained using abalone data set):**



**EXERCISE 3**

**Average distance of the predicted class from the true class of the binary-search learning algorithm applied to the abalone data set:**

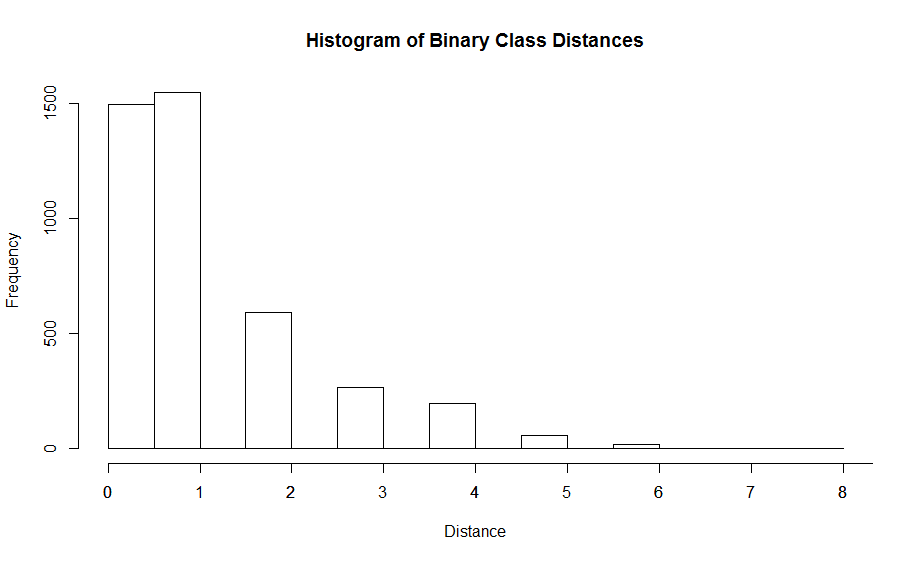
****

It is worth noting that the average distance using the binary search learning algorithm is about 0.4 lower.

**Training accuracy of the binary-search learning algorithm applied to the abalone data set:**

****

**Histogram showing the frequency of how often a prediction is m rings away from the true value:**



**EXERCISE 4**

**Table of 35 parameter combination results for eps-regression performed on the Exercise-4 data set:**

****

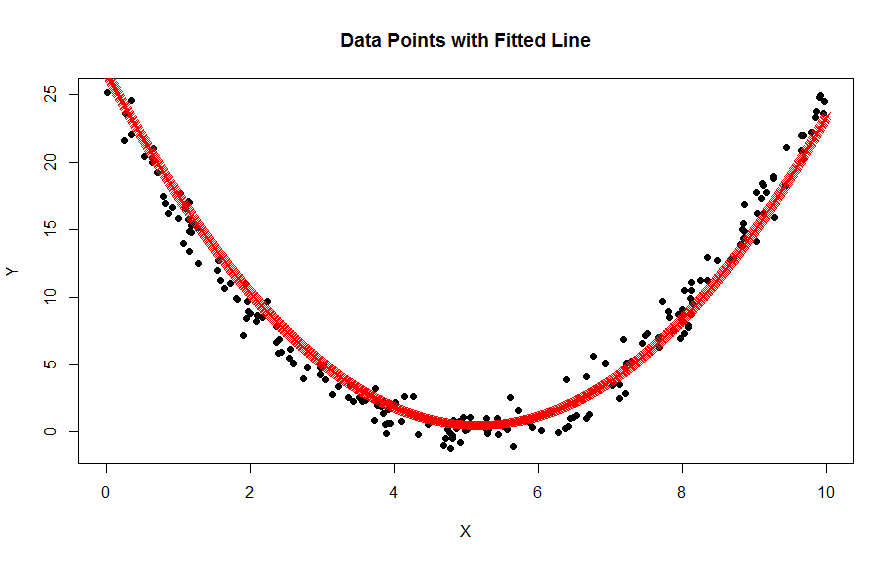
**Combination resulting in the lowest\* average CV MSE:**

****

We see that the epsilon and cost values that produce the lowest error, 1.857911, are 0.25 and 10, respectively.

**EXERCISE 5**

**Graph of plotted data points against curve provided by the best SVM from the previous exercise:**



Model Points

The **predicted line** was found by creating a sequence of 1000 points evenly spaced between 0 and 10.

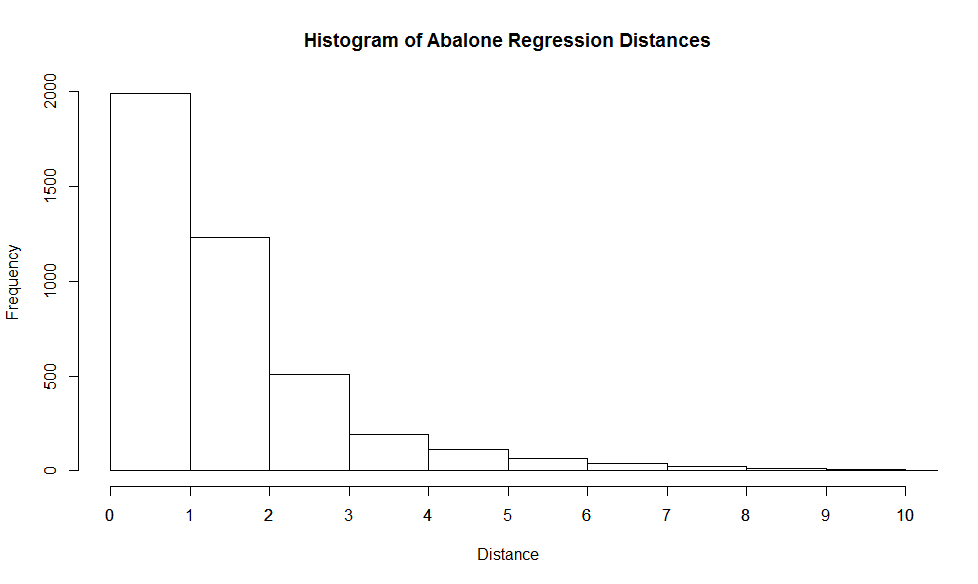
**EXERCISE 6**

**Average distance of the predicted class from the true class using the d, C, and e parameters which produce the least MSE:**

****

Just to be clear, a table was not asked for in this exercise.

**Histogram showing frequency of how often a prediction is m rings away from the true number of rings:**



APPENDIX

**EXERCISE 1**

# EXERCISE 1

# READ IN ABALONE DATA SET AND SET THE COLUMN NAMES BASED ON DATA DESCRIPTION

abalone\_df <- read.table("abalone.data", sep = ",", header = FALSE)

colnames(abalone\_df) <- c("Sex", "Length", "Diam", "Height", "Whole",

"Shucked", "Viscera", "Shell", "Rings")

# INITIALIZE LEARNING PARAMETERS

degrees <- c(1:3)

costs <- c(10 ^ (-1:2))

cross\_fold <- 5

# KEEP TRACK OF BEST CV ACCURACY AND THE BEST RESPECTIVE PREDICTIONS

best\_cv\_accuracy <- 0

best\_predictions <- data.frame()

# INITIALIZE TABLE (DATA FRAME TO BE APPENDED)

accuracies\_table <- data.frame("Degree" = integer(), "Cost" = numeric(),

"CV Accuracy" = numeric(), "Entire DF Accuracy" = numeric(),

stringsAsFactors = FALSE)

# BUILD MODEL FOR EVERY COMBINATION OF

for (d in degrees) {

for (c in costs) {

# MODEL WITHOUT CROSS VALIDATION

current\_model\_entire\_df <- svm(Rings ~ ., data = abalone\_df,

kernel = "polynomial", degree = d, type = "C-classification", cost = c)

predictions <- predict(current\_model\_entire\_df, abalone\_df[, - length(abalone\_df)])

accuracy\_total <- 100 \* mean(predictions == abalone\_df[, length(abalone\_df)])

# MODEL WITH 5-FOLD CROSS VALIDATION

current\_model <- svm(Rings ~ ., data = abalone\_df, kernel = "polynomial",

degree = d, type = "C-classification", cost = c, cross = cross\_fold)

accuracies\_table[nrow(accuracies\_table) + 1,] <- c(d, c, current\_model$tot.accuracy, accuracy\_total)

if (current\_model$tot.accuracy > best\_cv\_accuracy) {

best\_cv\_accuracy <- current\_model$tot.accuracy

best\_predictions <- predict(current\_model, abalone\_df[,-length(abalone\_df)])

}

}

}

# SORT BY INCREASING COMPLEXITY

accuracies\_table <- accuracies\_table[order(-accuracies\_table$Cost),]

accuracies\_table <- accuracies\_table[order(accuracies\_table$Degree),]

# FIND COMBINATION WITH HIGHEST CV ACCURACY

max\_combination <- accuracies\_table[which.max(accuracies\_table$CV.Accuracy),]

# AVERAGE DISTANCE OF THE PREDICTED CLASS FROM THE TRUE CLASS USING BEST PARAMS

best\_pred\_int <- as.integer(best\_predictions)

class\_distances <- abs(best\_pred\_int - abalone\_df$Rings)

average\_distance <- mean(class\_distances)

# HISTOGRAM SHOWING FREQUENCY OF M DISTANCES FROM TRUE CLASS

par(bg = 'white')

hist(class\_distances, main = "Histogram of Class Distances", xlab = "Distance", ylab = "Frequency", xlim = c(0, 10))

axis(side=1, at=seq(0,10,1), labels=seq(0,10,1))

**EXERCISE 2**

#EXERCISE 2

# MAKE ALL LESS THAN 5 = 5

abalone\_df$Rings[abalone\_df$Rings < 5] <- 5

# MAKE ALL GREATER THAN 14 = 14

abalone\_df$Rings[abalone\_df$Rings > 14] <- 14

# INITIALIZE BOUNDS FOR BINARY CLASSIFIER

f1 <- list(c(0:9), c(10:30))

f2 <- list(c(0:7), c(8:9))

f3 <- list(c(0:5), c(6:7))

f4 <- list(c(8), c(9))

f5 <- list(c(6), c(7))

f6 <- list(c(10:11), c(12:30))

f7 <- list(c(12:13), c(14:30))

f8 <- list(c(10), c(11))

f9 <- list(c(12), c(13))

# LIST OF ALL BOUNDS

classifier\_bounds <- list(f1, f2, f3, f4, f5, f6, f7, f8, f9)

# TABLE (DATA FRAME) WHERE CLASSIFIER TRAINING RESULTS WILL RESIDE

binary\_classifier\_table <- data.frame("Description" = character(), "Dataset Size" = integer(), "Degree" = numeric(), "Cost" = numeric(),

"Average CV Accuracy" = numeric(), "Best Accuracy" = numeric(), stringsAsFactors = FALSE)

# LIST OF ALL QUERY NODE MODELS

binary\_classifier\_models <- list()

for (bound in classifier\_bounds) {

# CALCULATE BEST PARAMATERS GIVEN THE BINARY PARTITION BOUNDS AND APPEND TO CLASSIFIER PARAMS DATAFRAME

negative\_class <- bound[[1]]

positive\_class <- bound[[2]]

# CREATE DESCRIPTION BASED ON BOUNDS

description <- ""

if (length(negative\_class) > 2) {

description <- paste(description, "<= ", negative\_class[length(negative\_class)])

} else if (length(negative\_class) == 2) {

description <- paste(description, negative\_class[1], "-", negative\_class[2])

} else {

description <- paste(description, negative\_class[1])

}

description <- paste(description, " vs ")

if (length(positive\_class) > 2) {

description <- paste(description, ">= ", positive\_class[1])

} else if (length(positive\_class) == 2) {

description <- paste(description, positive\_class[1], "-", positive\_class[2])

} else {

description <- paste(description, positive\_class[1])

}

df <- abalone\_df

# CREATE SUBSET BASED ON BOUNDS

df <- df[df$Rings %in% negative\_class | df$Rings %in% positive\_class,]

# CHANGE PROBLEM TO A BINARY CLASSIFIER BASED ON BOUNDS

df$Rings[df$Rings %in% negative\_class] <- -1

df$Rings[df$Rings %in% positive\_class] <- 1

average\_accuracy <- 0

best\_accuracy <- 0

best\_d <- 0

best\_c <- 0

cross\_fold <- 5

num\_of\_combos <- length(costs) \* length(degrees)

best\_model <- 0

for (d in degrees) {

for (c in costs) {

# BUILD MODEL FOR GIVEN COMBINATION OF D AND C

current\_model <- svm(Rings ~ ., data = df, kernel = "polynomial",

degree = d, type = "C-classification", cost = c, cross = cross\_fold)

average\_accuracy = average\_accuracy + current\_model$tot.accuracy

# FIND BEST ACCURACY

if (current\_model$tot.accuracy > best\_accuracy) {

best\_accuracy <- current\_model$tot.accuracy

best\_d <- d

best\_c <- c

best\_model <- current\_model

}

}

}

# APPEND BINARY CLASSIFIER TABLE: DESCRIPTION, TRAINING SET SIZE, BEST D, BEST C, AVERAGE CV ACC, BEST ACC

binary\_classifier\_table[nrow(binary\_classifier\_table) + 1,] <- c(description, nrow(df), best\_d, best\_c, average\_accuracy / num\_of\_combos, best\_accuracy)

# APPEND BINARY CLASSIFIER MODEL LIST: THESE ARE THE QUERY NODES

binary\_classifier\_models[[length(binary\_classifier\_models) + 1]] <- best\_model

}

**EXERCISE 3**

# EXERCISE 3

# PREDICT CLASSIFICATIONS USING TRAINED BINARY CLASSIFIER QUERA NODES

final\_predictions <- c()

for (i in 1:nrow(abalone\_df)) {

# <= 9 or >= 10

model <- binary\_classifier\_models[[1]]

prediction <- predict(model, abalone\_df[i, - length(abalone\_df)])

if (prediction == 1) {

# prediction >= 10

# 10-11 or >= 12

model <- binary\_classifier\_models[[6]]

prediction <- predict(model, abalone\_df[i, - length(abalone\_df)])

if (prediction == 1) {

# prediction >= 12

# 12-13 or >= 14

model <- binary\_classifier\_models[[7]]

prediction <- predict(model, abalone\_df[i, - length(abalone\_df)])

if (prediction == 1) {

# final prediction is 14

final\_predictions <- c(final\_predictions, 14)

} else {

# 12 or 13

model <- binary\_classifier\_models[[9]]

prediction <- predict(model, abalone\_df[i, - length(abalone\_df)])

if (prediction == 1) {

# final prediction is 13

final\_predictions <- c(final\_predictions, 13)

} else {

# final prediction is 12

final\_predictions <- c(final\_predictions, 12)

}

}

} else {

# prediction 10-11

model <- binary\_classifier\_models[[8]]

prediction <- predict(model, abalone\_df[i, - length(abalone\_df)])

if (prediction == 1) {

# final prediction is 11

final\_predictions <- c(final\_predictions, 11)

} else {

# final prediction is 10

final\_predictions <- c(final\_predictions, 10)

}

}

} else if (prediction == -1){

# prediction <= 9

# <= 7 or 8-9

model <- binary\_classifier\_models[[2]]

prediction <- predict(model, abalone\_df[i, - length(abalone\_df)])

if (prediction == 1) {

# prediction 8-9

model <- binary\_classifier\_models[[4]]

prediction <- predict(model, abalone\_df[i, - length(abalone\_df)])

if (prediction == 1) {

# final prediction is 9

final\_predictions <- c(final\_predictions, 9)

} else {

# final prediction is 8

final\_predictions <- c(final\_predictions, 8)

}

} else {

# prediction <= 7

# <= 5 or 6-7

model <- binary\_classifier\_models[[3]]

prediction <- predict(model, abalone\_df[i, - length(abalone\_df)])

if (prediction == 1) {

# prediction 6-7

model <- binary\_classifier\_models[[5]]

prediction <- predict(model, abalone\_df[i, - length(abalone\_df)])

if (prediction == 1) {

# final prediction is 7

final\_predictions <- c(final\_predictions, 7)

} else {

# final prediction is 6

final\_predictions <- c(final\_predictions, 6)

}

} else {

# final prediction is 5

final\_predictions <- c(final\_predictions, 5)

}

}

}

}

# TRAINING ACCURACY OF BINARY CLASSIFIER

binary\_class\_accuracy <- mean(final\_predictions == abalone\_df$Rings) \* 100

# AVERAGE DISTANCE OF PREDICTED CLASS FROM TRUE CLASS

binary\_class\_distances <- abs(final\_predictions - abalone\_df$Rings)

binary\_average\_distance <- mean(binary\_class\_distances)

hist(binary\_class\_distances, main = "Histogram of Binary Class Distances",

xlab = "Distance", ylab = "Frequency", xlim = c(0, 8))

axis(side = 1, at = seq(0, 10, 1), labels = seq(0, 10, 1))

**EXERCISE 4**

# EXERCISE 4

# READ IN DATA SET

reg\_df <- read.csv("Exercise-4.csv")

# INITIALIZE TRAINING PARAMETERS

reg\_cross\_fold <- 10

reg\_costs <- c(10 ^ (-1:3))

epsilons <- c(0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75)

# KEEP TRACK OF PARAMS THAT PRODUCE LOWEST MSE

lowest\_mse <- 1000

best\_reg\_c <- 0

best\_reg\_e <- 0

best\_reg\_model <- 0

# TABLE(DATAFRAME) WHERE RESULTS FROM EACH COMBINATION RESIDE

reg\_accuracies\_table <- data.frame("Epsilon" = integer(), "Cost" = numeric(),

"CV MSE" = numeric(), "Entire DF MSE" = numeric(), stringsAsFactors = FALSE)

for (c in reg\_costs) {

for (e in epsilons) {

# BUILD MODEL AND CALCULATE MSE OVER ENTIRE DATASET

reg\_model\_entire <- svm(Y ~ X, data = reg\_df, kernel = "polynomial",

degree = 2, type = "eps-regression", epsilon = e, cost = c, cross = 100)

# BUILD MODEL USING 10-FOLD CROSSVALIDATION

reg\_model <- svm(Y ~ X, data = reg\_df, kernel = "polynomial",

degree = 2, type = "eps-regression", epsilon = e, cost = c, cross = 10)

# APPEND TABLE WITH CURRENT RESULTS

reg\_accuracies\_table[nrow(reg\_accuracies\_table) + 1,] <- c(e, c, reg\_model$tot.MSE, reg\_model\_entire$tot.MSE)

if (reg\_model$tot.MSE < lowest\_mse) {

lowest\_mse = reg\_model$tot.MSE

best\_reg\_c <- c

best\_reg\_e <- e

best\_reg\_model <- reg\_model

}

}

}

# SORT TABLE WITH INCREASING COMPLEXITY

reg\_accuracies\_table <- reg\_accuracies\_table[order(reg\_accuracies\_table$Epsilon),]

reg\_accuracies\_table <- reg\_accuracies\_table[order(reg\_accuracies\_table$Cost),]

# COMBINATION FOR HIGHEST CV ACCURACY

reg\_min\_combination <- reg\_accuracies\_table[which.min(reg\_accuracies\_table$CV.MSE),]

**EXERCISE 5**

# EXERCISE 5

# PREDICTED LINE USING 1000 EVENLY SPACED POINTS FROM 0 TO 10

plot(reg\_df, pch = 16)

test\_data <- data.frame(seq(0, 10, length.out = 1000))

colnames(test\_data) <- c("X")

predicted <- predict(best\_reg\_model, test\_data)

points(test\_data$X, predicted, col = "red", pch = 4)

title("Data Points with Fitted Line")

**EXERCISE 6**

# EXERCISE 6

# INITIALIZE LEARNING PARAMETERS

reg\_cross\_fold <- 5

reg\_costs <- c(10 ^ (-1:3))

epsilons <- c(0.25, 0.5, 0.75)

degrees <- c(1:3)

# KEEP TRACK OF PARAMS THAT PRODUCE LOWEST MSE

best\_reg\_c\_abalone <- 0

best\_reg\_e\_abalone <- 0

best\_reg\_d\_abalone <- 0

best\_reg\_model\_abalone <- 0

lowest\_mse\_abalone <- 1000

best\_reg\_model\_abalone <- 0

for (c in reg\_costs) {

for (e in epsilons) {

for (d in degrees) {

reg\_model <- svm(Rings ~ ., abalone\_df, kernel = "polynomial", degree = d, type = "eps-regression", epsilon = e, cost = c, cross = reg\_cross\_fold)

if (reg\_model$tot.MSE < lowest\_mse\_abalone) {

lowest\_mse\_abalone = reg\_model$tot.MSE

best\_reg\_c\_abalone <- c

best\_reg\_e\_abalone <- e

best\_reg\_d\_abalone <- d

best\_reg\_model\_abalone <- reg\_model

}

}

}

}

# AVERAGE DISTANCE OF THE PREDICTED CLASS FROM THE TRUE CLASS

predicted\_reg\_abalone <- predict(best\_reg\_model\_abalone, abalone\_df[, - length(df)])

reg\_abalone\_distances <- abs(predicted\_reg\_abalone - abalone\_df[, length(df)])

average\_distance\_reg\_abalone <- mean(reg\_abalone\_distances)

# HISTOGRAM OF FREQUENCY OF DISTANCES FROM THE TRUE CLASS

hist(reg\_abalone\_distances, main = "Histogram of Abalone Regression Distances",

xlab = "Distance", ylab = "Frequency", xlim = c(0, 10))

axis(side = 1, at = seq(0, 10, 1), labels = seq(0, 10, 1))