SOLUTIONS

EXERCISE 1

Parameter combination table using the abalone data set:

```
> accuracies_table
   Degree  Cost CV.Accuracy Entire.DF.Accuracy
4       1 100.0    26.21499           27.65142
3       1  10.0    25.78406           26.88532
2       1   1.0    25.30524           26.19105
1       1   0.1    24.01245           24.63491
8       2 100.0    26.64592           31.79315
7       2  10.0    26.50227           28.82452
6       2   1.0    25.97558           27.43596
5       2   0.1    23.94063           24.87431
12      3 100.0    26.64592           33.68446
11      3  10.0    26.09528           29.56667
10      3   1.0    24.97007           26.71774
9       3   0.1    23.53364           24.61096
```

Combination with the highest cross-validation accuracy:

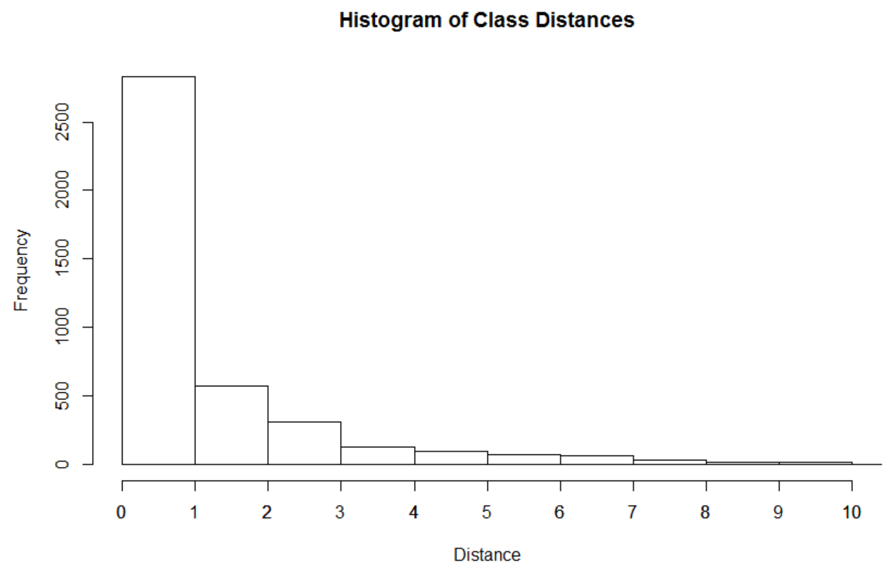
```
> max_combination
  Degree Cost CV.Accuracy Entire.DF.Accuracy
8      2  100    26.64592           31.79315
```

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:

```
> average_distance
[1] 1.513766
```

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):

```
> binary_classifier_table
           Description Dataset.Size Degree Cost Average.CV.Accuracy    Best.Accuracy
1    <=  9  vs  >=  10         4177      2  100    77.0868246748065 79.8659324874312
2     <=  7  vs  8 - 9         2096      1   10    82.0928753180662 83.0629770992366
3     <=  5  vs  6 - 7          839      1  0.1    87.0977353992849 87.7234803337306
4               8  vs  9       1257      2    1    62.9209758684699 64.7573587907717
5               6  vs  7        650      3    1    64.3076923076923 65.8461538461538
6  10 - 11  vs  >=  12         2081      2  100    65.8537562069518  70.783277270543
7  12 - 13  vs  >=  14          960      1   10    60.1388888888889           64.375
8            10  vs  11        1121      3   10    57.8873030032709 59.2328278322926
9            12  vs  13         470      2  100    56.8439716312057 59.5744680851064
```

EXERCISE 3

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

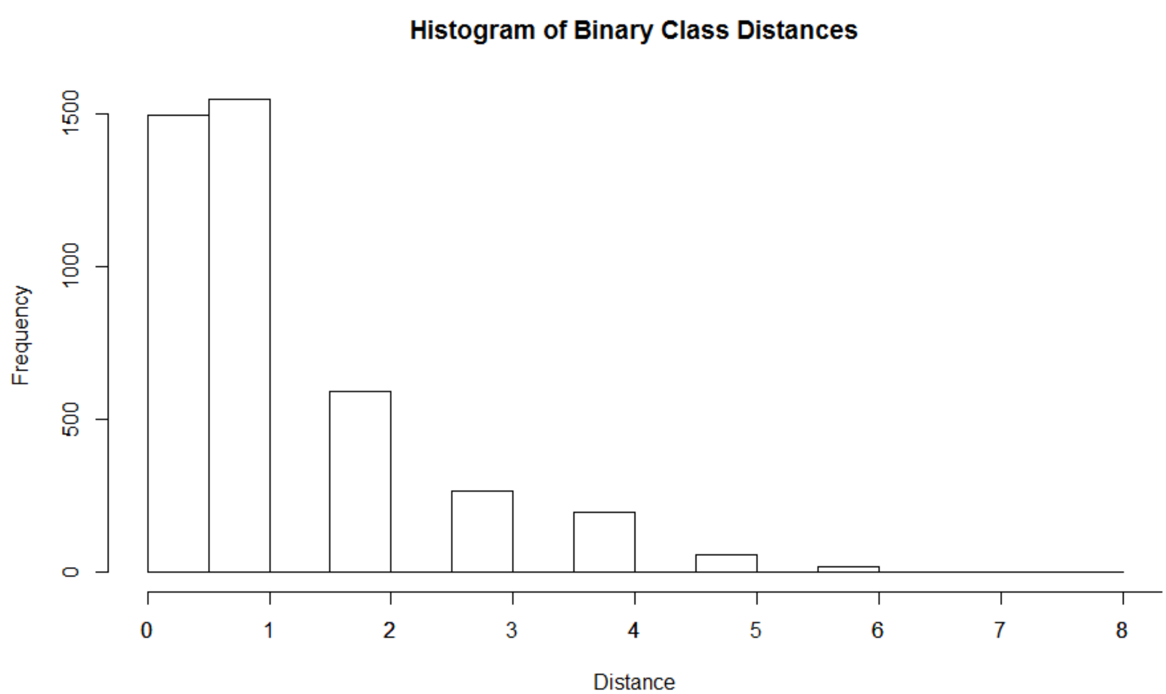
```
> binary_average_distance
[1] 1.141
```

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:

```
> binary_class_accuracy
[1] 36.62916
```

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:

```
> reg_accuracies_table
   Epsilon  Cost    CV.MSE Entire.DF.MSE
1     0.25 1e-01  1.957467      1.954661
2     0.50 1e-01  4.841288      4.635268
3     0.75 1e-01 10.620081     10.494145
4     1.00 1e-01 20.191924     19.753267
5     1.25 1e-01 31.890114     31.327224
6     1.50 1e-01 45.605031     45.503308
7     1.75 1e-01 64.058366     63.835242
8     0.25 1e+00  1.901948      1.880863
9     0.50 1e+00  3.009844      2.937969
10    0.75 1e+00  9.455004      9.389761
11    1.00 1e+00 18.188184     18.107956
12    1.25 1e+00 30.246685     30.152740
13    1.50 1e+00 45.629492     45.485435
14    1.75 1e+00 64.005422     63.828593
15    0.25 1e+01  1.857911      1.860247
16    0.50 1e+01  3.087263      2.941775
17    0.75 1e+01  9.435075      9.388801
18    1.00 1e+01 18.153996     18.114872
19    1.25 1e+01 30.258712     30.153416
20    1.50 1e+01 45.669981     45.492341
21    1.75 1e+01 63.988900     63.843066
22    0.25 1e+02  1.870768      1.868057
23    0.50 1e+02  3.070670      2.942062
24    0.75 1e+02  9.412851      9.390340
25    1.00 1e+02 18.184213     18.116082
26    1.25 1e+02 30.268778     30.152287
27    1.50 1e+02 45.597345     45.486227
28    1.75 1e+02 63.954225     63.826524
29    0.25 1e+03  1.861971      1.853047
30    0.50 1e+03  3.011352      2.946526
31    0.75 1e+03  9.424377      9.389048
32    1.00 1e+03 18.204606     18.116458
33    1.25 1e+03 30.275061     30.153726
34    1.50 1e+03 45.609684     45.506906
35    1.75 1e+03 64.016537     63.824236
```

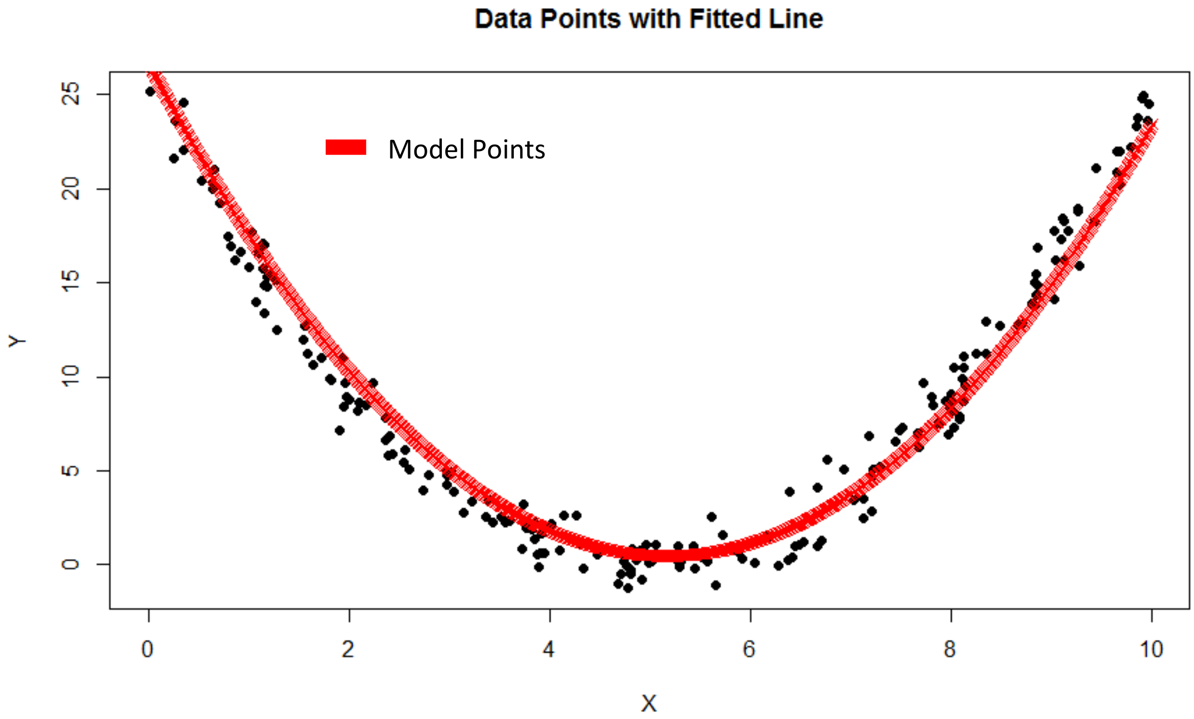
Combination resulting in the lowest* average CV MSE:

```
> reg_min_combination
   Epsilon Cost   CV.MSE Entire.DF.MSE
15    0.25   10 1.857911      1.860247
```

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:



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

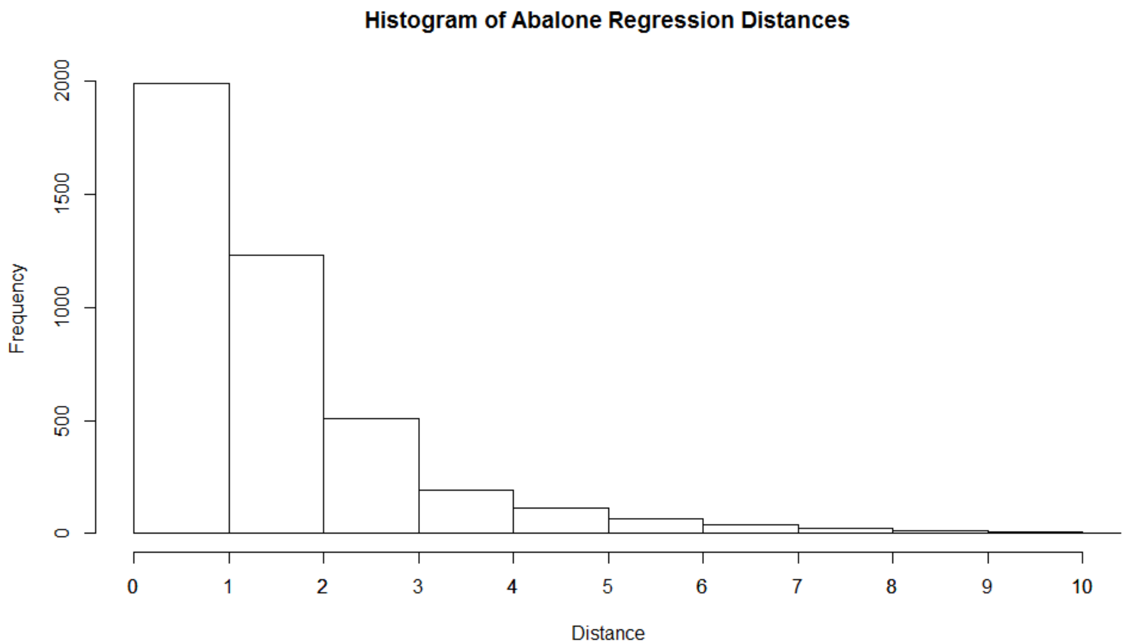
EXERCISE 6

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

```
> average_distance_reg_abalone
[1] 1.462579
```

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))
```