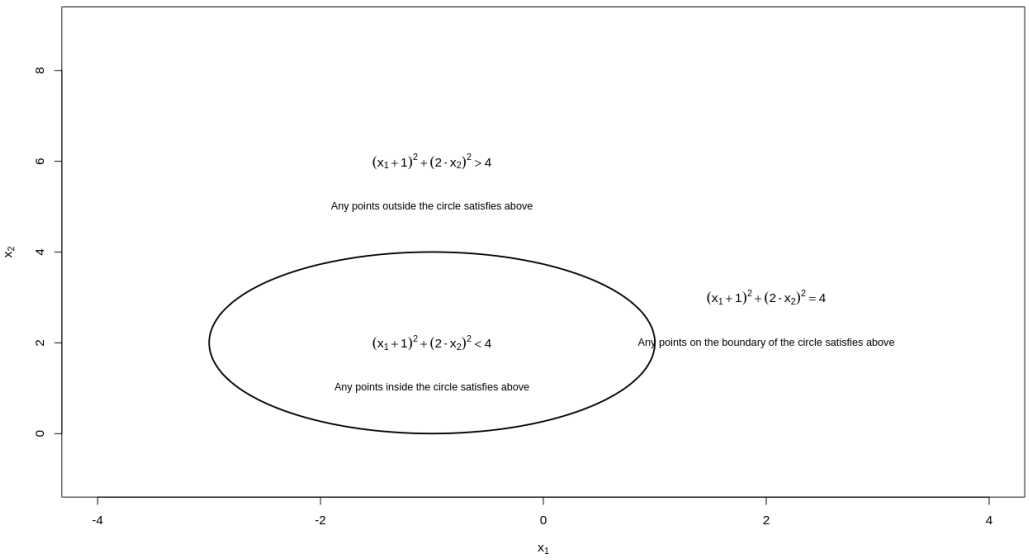
Solutions to Assignment 8

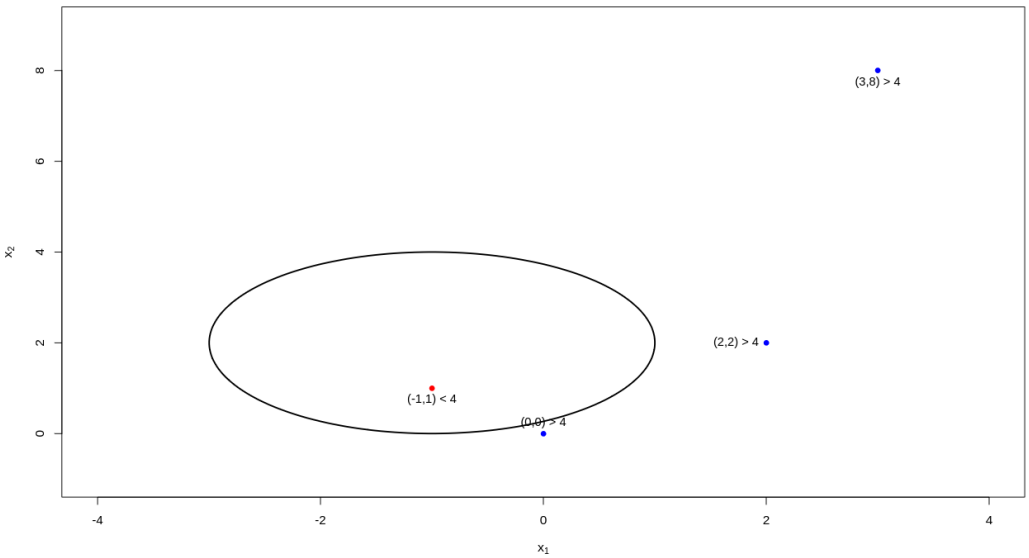
Rongfei Jin

March 28, 2025

1 ISLR

1.1 Conceptual 2





1.1.1 (d)

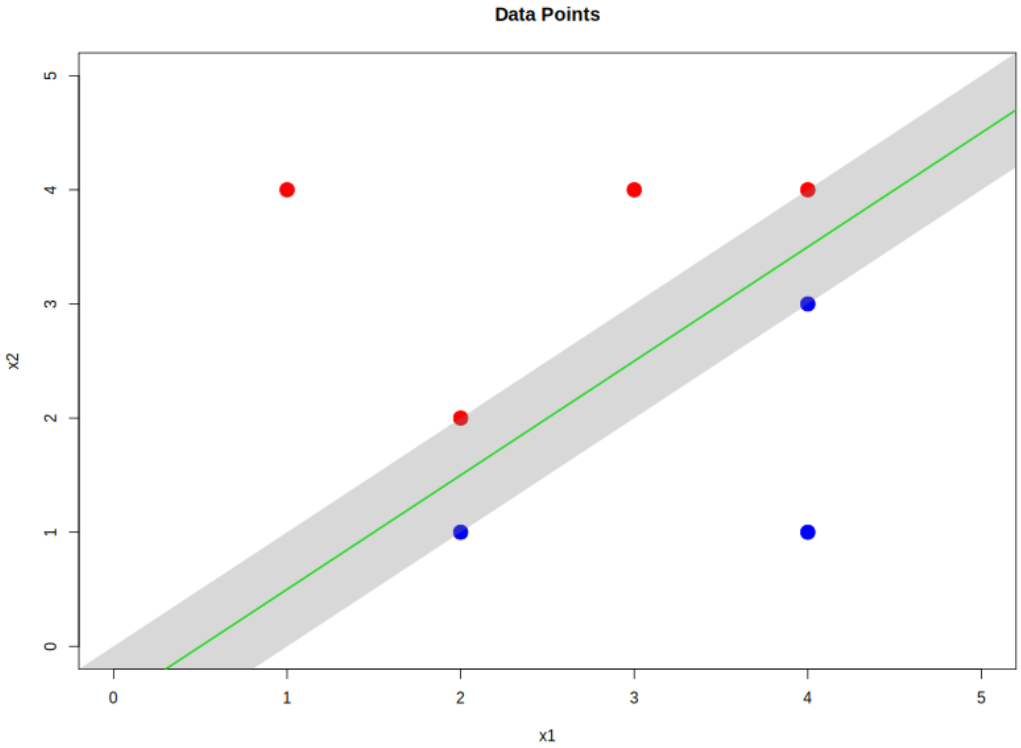
One can expandsn the expression as follow

$$\begin{aligned}(1+X_1)^2+(2-X_2)&=4\\1+2X_1+X_1^2+4-4X_2+X_2^2&=4\\1+2X_1+X_1^2-4X_2+X_2^2&=0\end{aligned}$$

The left hand side is not a linear combination of X_1 and X_2, thus it is not a linear

1.2 Conceptual 3

(a), (d)



The width of the gray rectangular region indicates the margin

(c), (b)

The decision boundary is given by $0.5 + X_1 - X_2 = 0$

Classifiy to Red if $0.5 + X_1 - X_2 > 0$, otherwise classify to Blue.

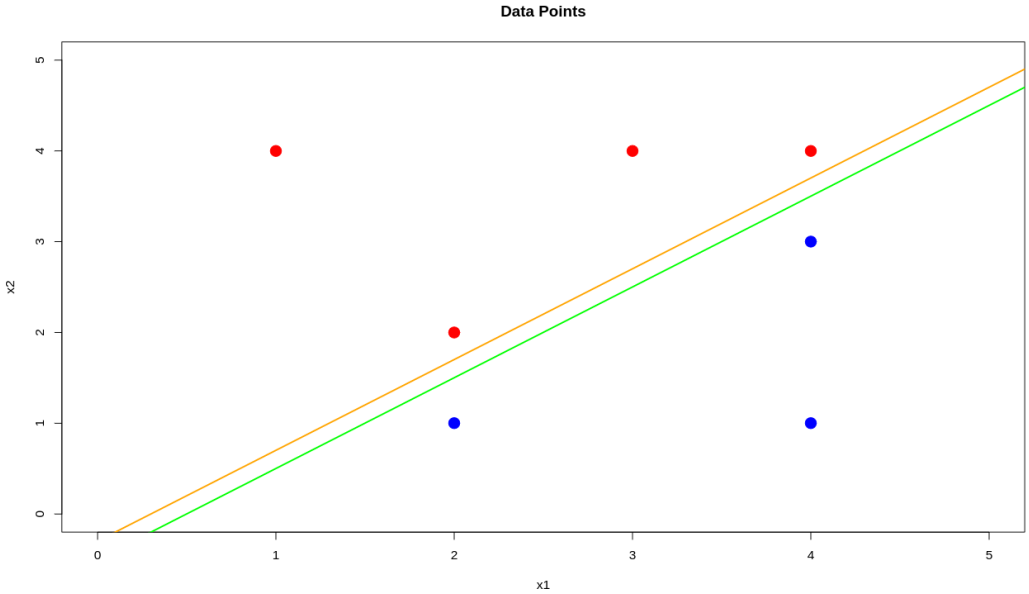
(e)

The support vectors are (2,1), (2,2), (4,3), (4,4)

(f)

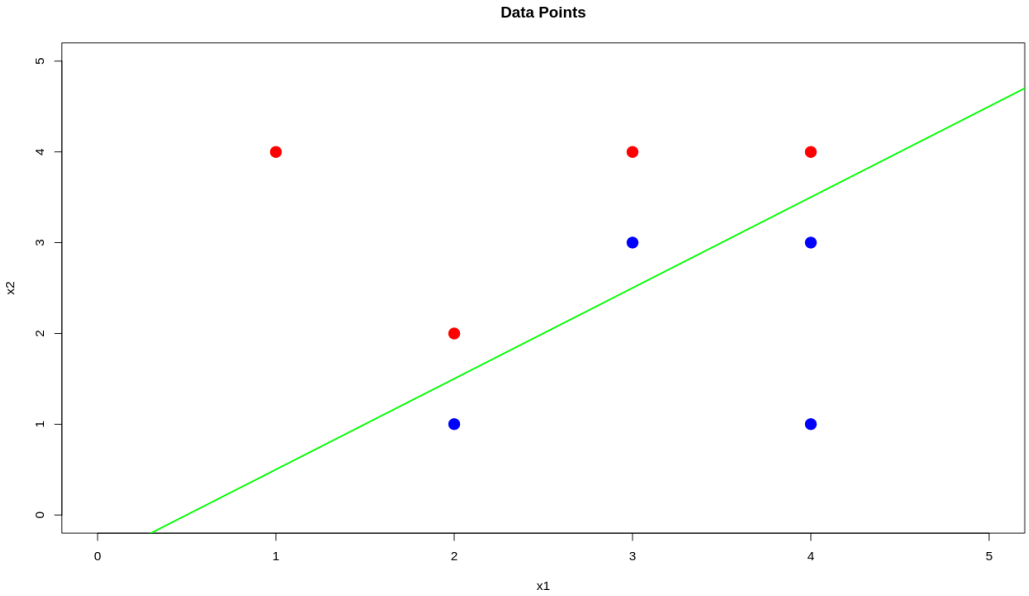
The maximal margin hyperplane is only determined by the support vectors, thus the decision boundary will not change if we slightly move the 7th observation

(g)

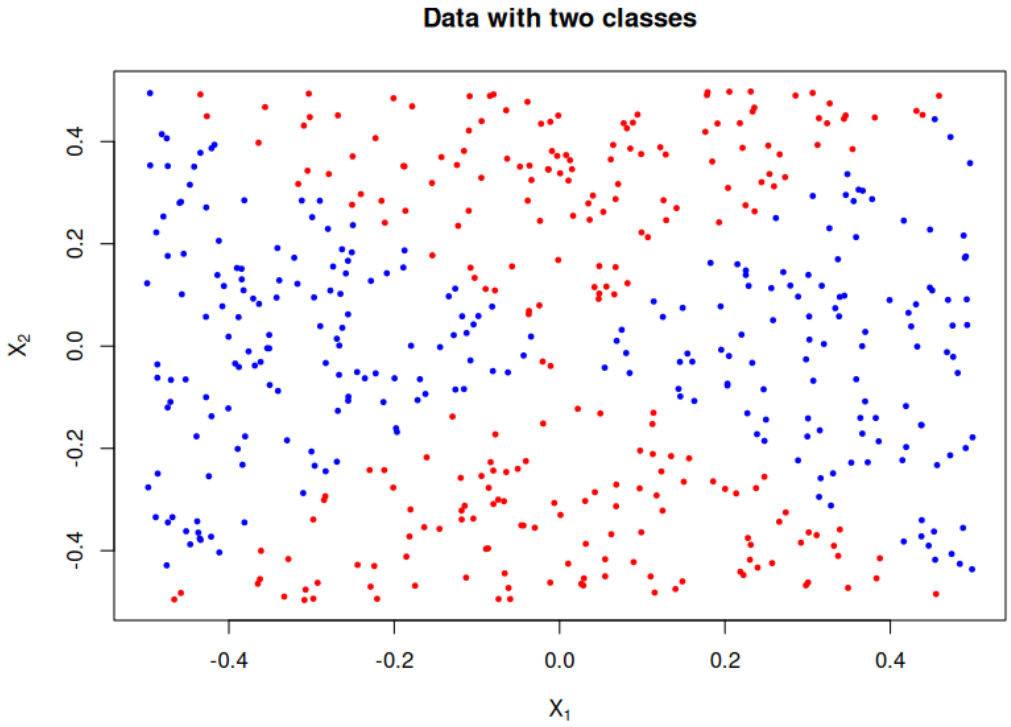


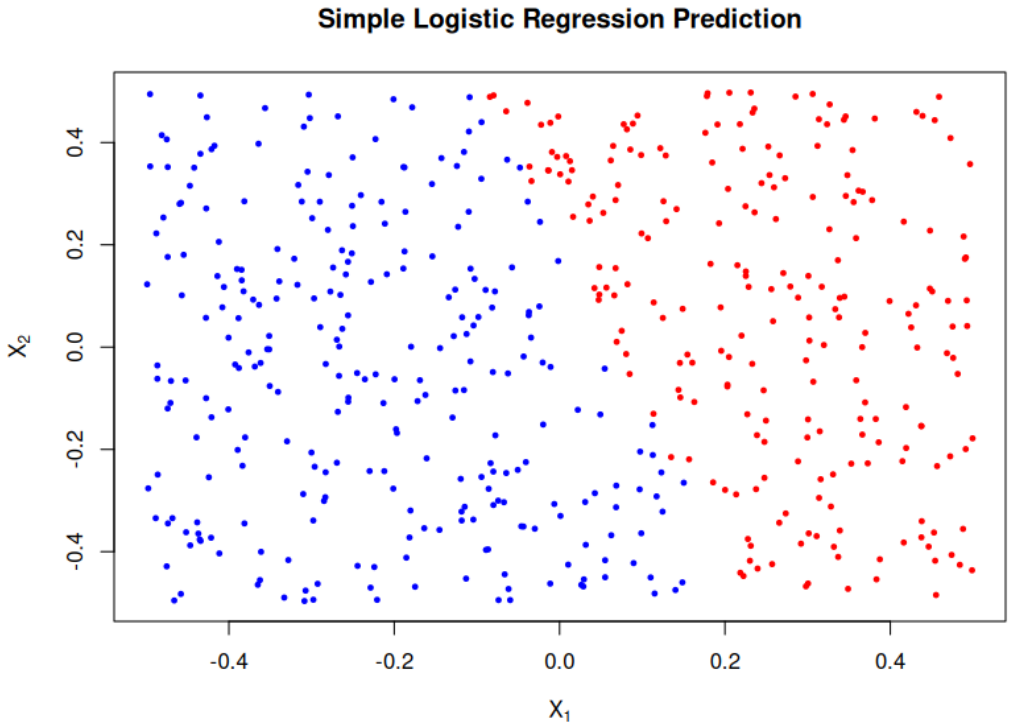
The orange line is the non-optimal hyperplane

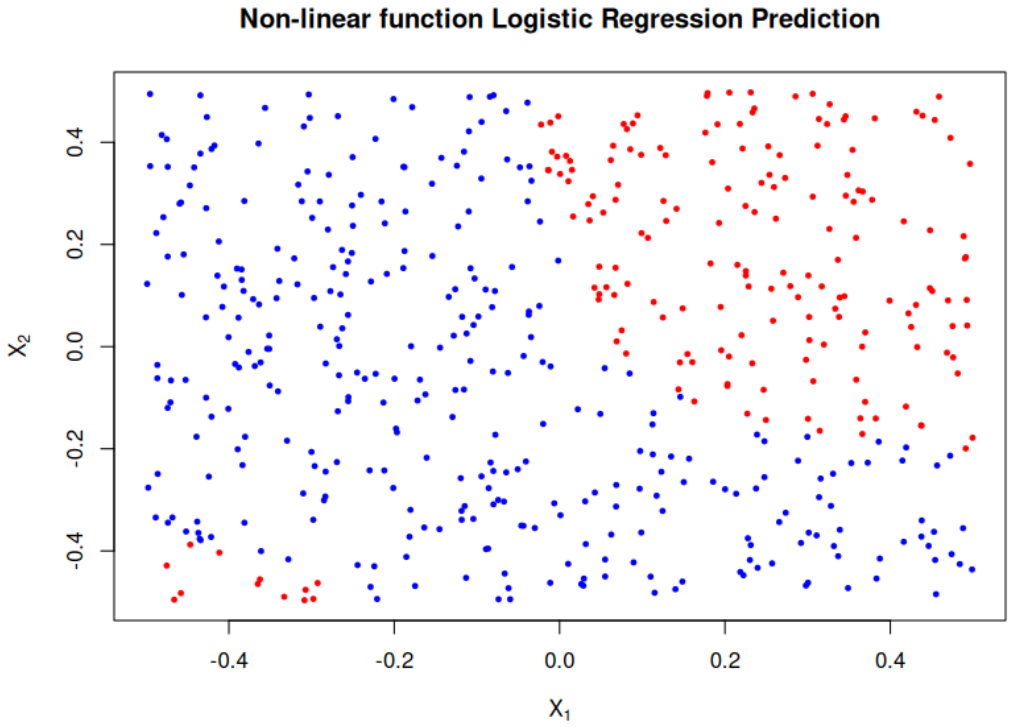
(f)

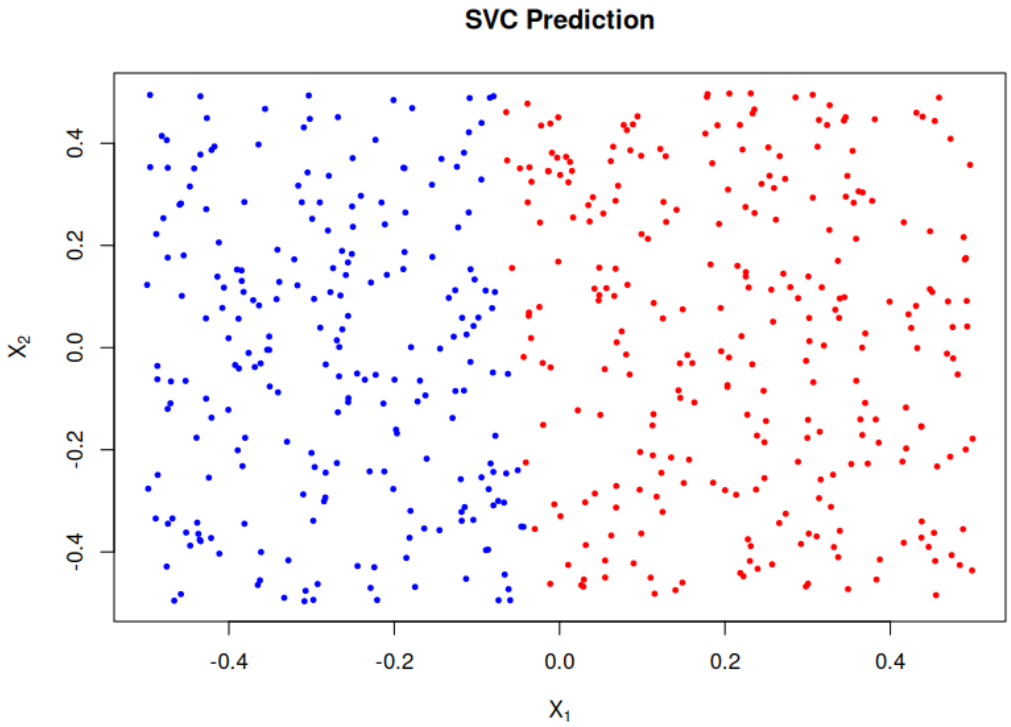


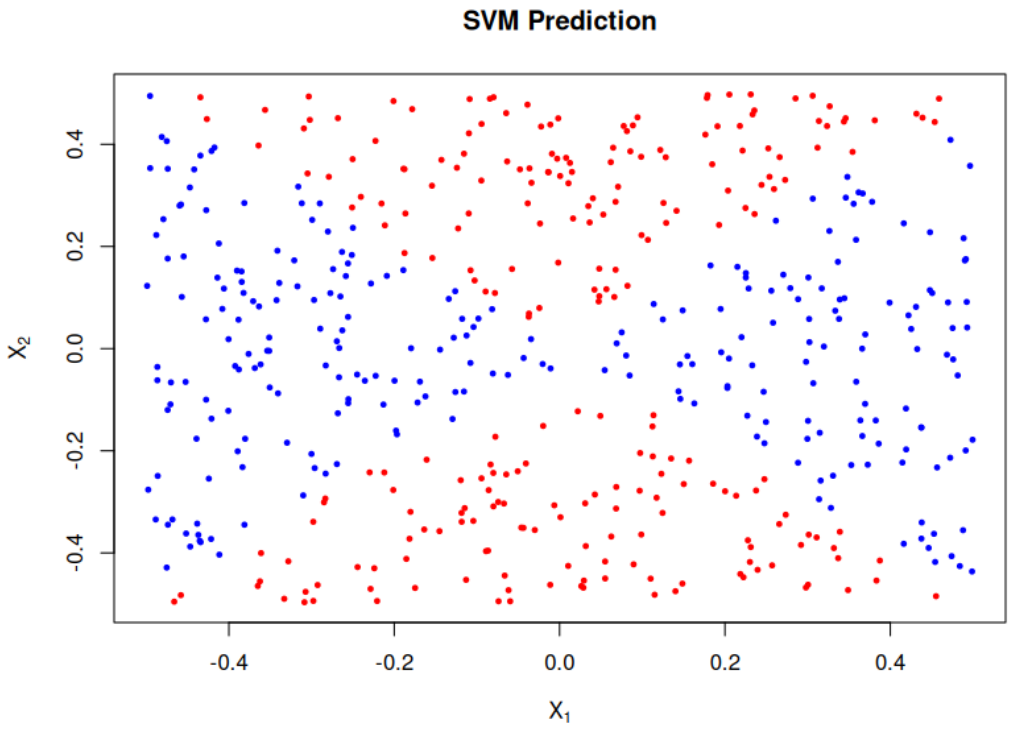
1.3 Applied 5


Data with two classes
X2
X1
-0.4
-0.2
0.0
0.2
0.4


Simple Logistic Regression Prediction
X2
X1
-0.4
-0.2
0.0
0.2
0.4


Non-linear function Logistic Regression Prediction
X2
X1
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4


SVC Prediction
X2
X1
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4



The logistic regression models, either linear or non-linear feature, are able to classify some of the data, but not all of them. The decision boundary is not optimal as it does not separate the data well.

The SVM with linear kernel is not able to classify the data well, as the data is not linearly separable.

The SVM with radial kernel is able to classify the data well, as it can capture the non-linear relationship between the features.

```
library(e1071) # for SVM
x1 <- runif (500) - 0.5
x2 <- runif (500) - 0.5
y <- 1 * (x1^2 - x2^2 > 0)

data <- data.frame(x1 = x1, x2 = x2, y = as.factor(y))

png(filename = "assignment8/figs/q5-1.png", width = 800, height = 600, res = 100)
plot(x1, x2, col = ifelse(y == 0, "red", "blue"), pch = 19, cex = 0.5,
     xlab = expression(X[1]), ylab = expression(X[2]), main = "Data with two classes")
dev.off()

logistic_model <- glm(y ~ x1 + x2, data = data, family = binomial)
pred_1 <- predict(logistic_model, type = "response")
pred_1 <- ifelse(pred_1 > 0.5, 1, 0)

png(filename = "assignment8/figs/q5-2.png", width = 800, height = 600, res = 100)
plot(x1, x2, col = ifelse(pred_1 == 0, "red", "blue"), pch = 19, cex = 0.5,
     xlab = expression(X[1]), ylab = expression(X[2]), main = "Simple Logistic Regression Prediction")
# save the plot
dev.off()



logistic_model_2 <- glm(y ~ x1 + x2 + I(x1*x2) + I(x1)^2, data = data, family = binomial)
pred_2 <- predict(logistic_model_2, type = "response")
```

```r
pred_2 <- ifelse(pred_1 > 0.5, 1, 0)
png(filename = "assignment8/figs/q5-3.png", width = 800, height = 600, res = 100)
plot(x1, x2, col = ifelse(pred_2 == 0, "red", "blue"), pch = 19, cex = 0.5,
     xlab = expressiopredictions <- predict(svm_model, data)n(X[1]), ylab = expression(X[2]), main = "Non-lin
# save the plot
dev.off()

# support vector classifier
svc_model <- svm(y ~ .,
                 data = data,
                 kernel = "linear",
                 cost = 10)
pred_svc <- predict(svc_model, data)
png(filename = "assignment8/figs/q5-4.png", width = 800, height = 600, res = 100)
plot(x1, x2, col = ifelse(pred_svc == 0, "red", "blue"), pch = 19, cex = 0.5,
     xlab = expression(X[1]), ylab = expression(X[2]), main = "SVC Prediction")
# save the plot
dev.off()



svm_model <- svm(y ~ .,
                 data = data,
                 kernel = "radial",
                 gamma = 0.5,
                 cost = 10)

pred_svm <- predict(svm_model, data)
png(filename = "assignment8/figs/q5-5.png", width = 800, height = 600, res = 100)
plot(x1, x2, col = ifelse(pred_svm == 0, "red", "blue"), pch = 19, cex = 0.5,
     xlab = expression(X[1]), ylab = expression(X[2]), main = "SVM Prediction")
# save the plot
dev.off()
# Save the models for later use
```

1.4 Applied 8

Based on the summaries, both radial and polynomial d=2 basi utilized over 630 vectors where the linear models utilized much less vectors around 340.

Based on the test errors the tuned linear model performed the best with 0.15, while the polynomial models performed the worst

```
library(ISLR2)
library(e1071)

str(OJ)

# sample 800 rows from the data
set.seed(123)
training_rows <- sample(1:nrow(OJ), 800, replace = FALSE)
oj_training <- OJ[training_rows, ]
oj_testing <- OJ[-training_rows, ]

# svc with cost = 0.01
svc_model <- svm(Purchase ~ .,
                 data = oj_training,
                 kernel = "linear",
                 cost = 0.01)

summary(svc_model)

compute_errors <- function(model, training, testing) {
  pred <- predict(model, training)
  mat <- table(pred, training$Purchase)
  error_rate <- 1 - sum(diag(mat)) / sum(mat)

  pred_test <- predict(model, testing)
  mat_test <- table(pred_test, testing$Purchase)
  error_rate_test <- 1 - sum(diag(mat_test)) / sum(mat_test)

  return(list(training_error = error_rate, testing_error = error_rate_test))
}


tuned_svc_model <- tune(svm, Purchase ~ .,
                        data = oj_training,
                        kernel = "linear",
                        ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10)))


# Radial version
radial_model_1 <- svm(Purchase ~ .,
                      data = oj_training,
                      kernel = "radial",
                      cost = 0.01
                      )
summary(radial_model_1)

# tune radial model
tuned_radial_model_1 <- tune(svm, Purchase ~ .,
                             data = oj_training,
                             kernel = "radial",
                             ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10)))
```

```r
# polynomial version d=2
poly_model <- svm(Purchase ~ .,
                  data = oj_training,
                  kernel = "polynomial",
                  cost = 0.01,
                  degree = 2
                  )
summary(poly_model)

tuned_poly_model <- tune(svm, Purchase ~ .,
                         data = oj_training,
                         kernel = "polynomial",
                         ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10)))


compute_errors(svc_model, oj_training, oj_testing)
compute_errors(tuned_svc_model$best.model, oj_training, oj_testing)
compute_errors(radial_model_1, oj_training, oj_testing)
compute_errors(tuned_radial_model_1$best.model, oj_training, oj_testing)
compute_errors(poly_model, oj_training, oj_testing)
compute_errors(tuned_poly_model$best.model, oj_training, oj_testing)
```

2 ESL

2.1 Ch 4, Exercise 7

The criterion enforces the value $\beta^T x_i + \beta_0$ to be the distance to the hyperplane

However, the constraint does not solve the problem because the max margin hyperplane is not necessarily the one that minimizes the distance to the hyperplane.

2.2 Ch 12, Exercise 1

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}\xi_i \tag{1}$$

$$\text{subject to } \xi_i \geq 1 - y_i(\beta^T x_i + \beta_0), \quad \xi_i \geq 0, \quad i = 1, \ldots, n \tag{2}$$

$$\min_{\beta,\beta_0} \sum_{i=1}^{N}[1 - y_i f(x_i)]_+ + \frac{\lambda}{2}\|\beta\|^2$$

let $\lambda = \frac{1}{C}$

$$\min_{\beta,\beta_0} \sum_{i=1}^{N}[1 - y_i f(x_i)]_+ + \frac{1}{2C}\|\beta\|^2 \tag{3}$$

$$\iff \min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}[1 - y_i f(x_i)]_+ \tag{4}$$

$$\tag{5}$$

Now any β, β_0 that satifies (4) will satisfy (1)
Now consider $\xi_i > 1 - y_i f(x_i)$ by the KKT conditions we have

$$a_i[y_i(\beta^T x_i + \beta_0) - (1 - \xi_i)] = 0$$

$$\mu_i = 0$$

Since $\xi_i > 1 - y_i f(x_i)$ we have $a_i = 0$

$$L_p = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}a_i[1 - y_i f(x_i) - \xi_i]$$

Therefore the primal function minimization problem above is equivalent to (1)