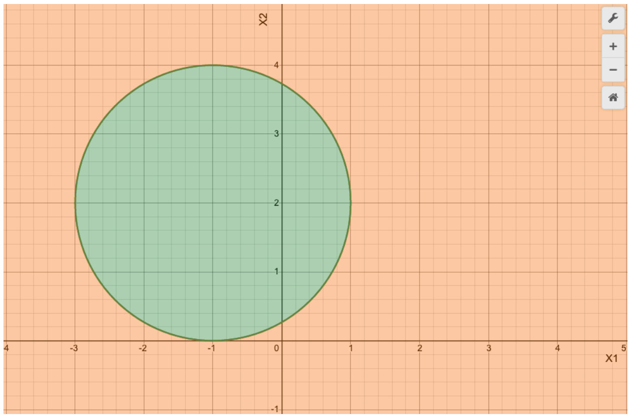
STATS 101C - Statistical Models and Data Mining - Homework 7

Darren Tsang, Discussion 4B

Produced on Wednesday, Dec. 02 2020 @ 05:26:36 AM

Question 1 (Exercise 2 from Section 9.7)

Part A and B



The green part are the points where $(1 + X_1)^2 + (2 - X_2)^2 \leq 4$. The orange part are the points where $(1 + X_1)^2 + (2 - X_2)^2 > 4$.

Part C

For (0, 0):

$$(1 + 0)^2 + (2 - 0)^2 = 5 > 4 \longrightarrow blue$$

For (-1, 1):

$$(1 + (-1))^2 + (2 - 1)^2 = 1 \not> 4 \longrightarrow red$$

For (2, 2):

$$(1 + 2)^2 + (2 - 2)^2 = 9 > 4 \longrightarrow blue$$

For (3, 8):

$$(1 + 3)^2 + (2 - 8)^2 = 52 > 4 \longrightarrow blue$$

Part D

We begin by expanding the equation of the decision boundary:

$$X_1^2 + 2X_1 + 1 + X_2^2 - 4X_2 + 4 - 4 > 0$$

It is not linear the predictor space $[X_1, X_2]$ because there is an X_1^2 and X_2^2 term.

On the other hand, it is linear in terms of the enlarged feature space $[X_1, X_1^2, X_2, X_2^2]$. We can set $X_1 = a$, $X_1^2 = b$, $X_2 = c$, and $X_2^2 = d$. Then, the decision boundary becomes $b + 2a + d - 4c + 1 > 0$, which is clearly linear.

Question 2 (Exercise 5 from Section 9.7)

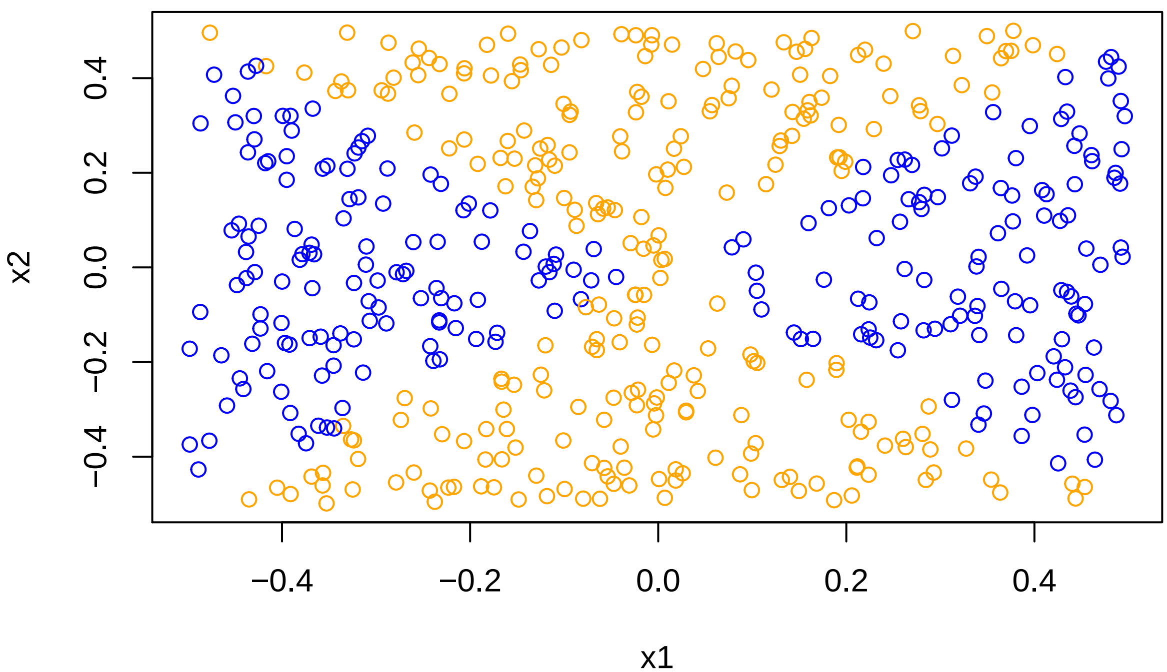
```
library(ggplot2)
library(e1071)
```

Part A

```
set.seed(1)
x1 <- runif(500) - 0.5
x2 <- runif(500) - 0.5
y <- 1*(x1^2-x2^2 > 0)
```

Part B

```
plot(x1, x2, col = ifelse(y, 'blue', 'orange'))
```



Part C

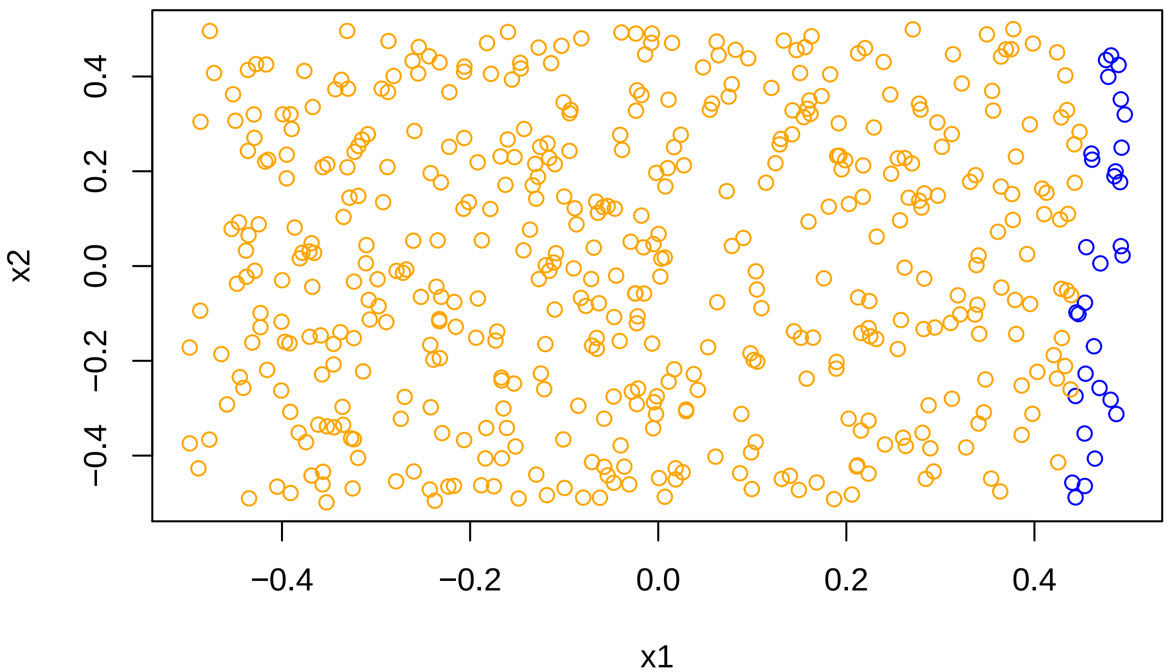
```
base.logistic <- glm(y ~ x1 + x2,
                     data = data.frame(x1, x2, 'y' = as.factor(y)),
                     family = "binomial")
```

Part D

```r
base.logistic.predictions <- predict(base.logistic, data.frame(x1, x2)) > 0
mean(base.logistic.predictions == y)
```

```
## [1] 0.57
```

```r
plot(x1, x2, col = ifelse(base.logistic.predictions, 'blue', 'orange'))
```

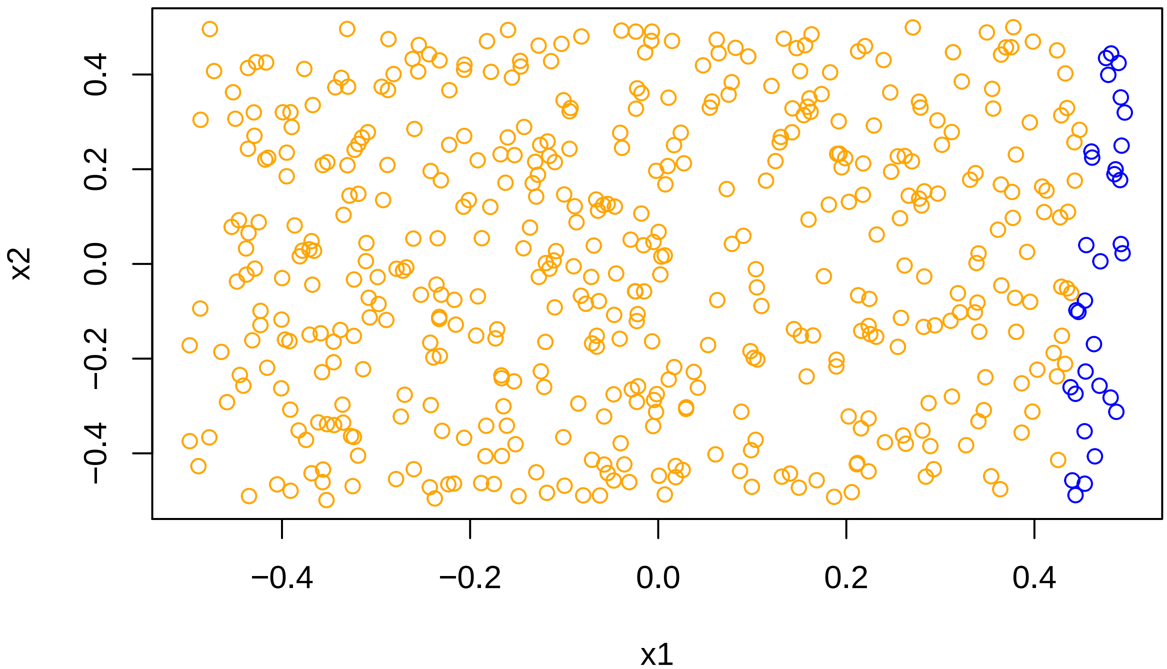


Part E and F

```r
multiplied.logistic <- glm(y ~ x1 * x2,
                           data = data.frame(x1, x2, 'y' = as.factor(y)),
                           family = "binomial")
multiplied.logistic.predictions <- predict(multiplied.logistic, data.frame(x1, x2)) > 0
mean(multiplied.logistic.predictions == y)
```

```
## [1] 0.572
```

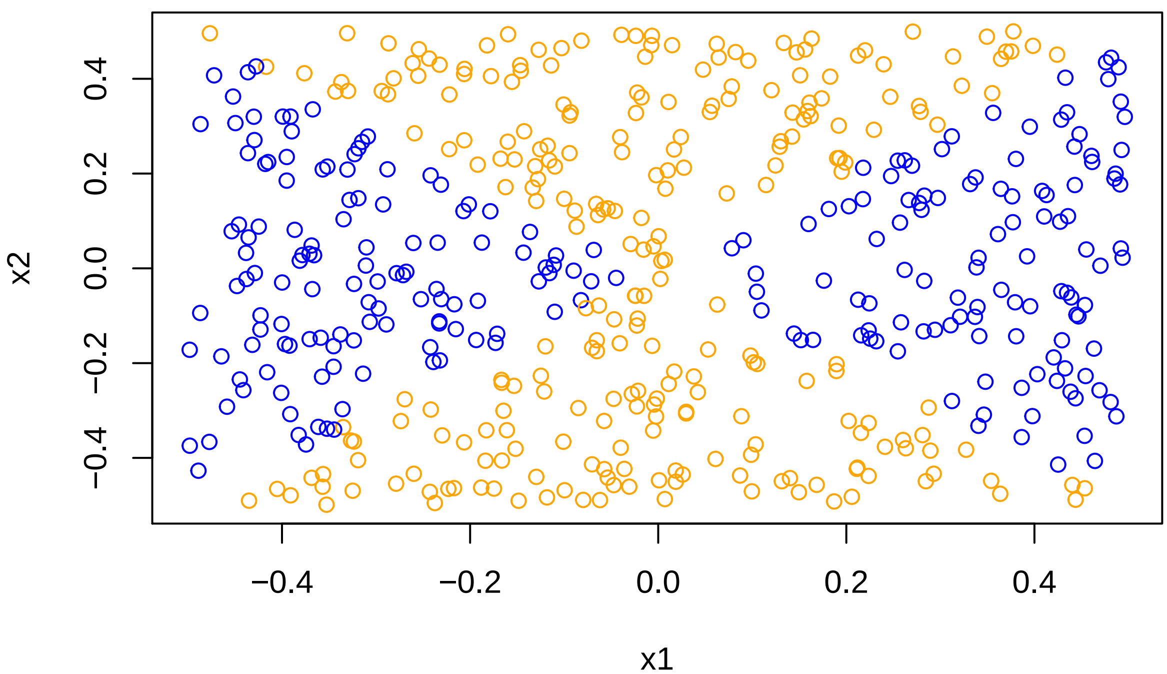
```r
plot(x1, x2, col = ifelse(multiplied.logistic.predictions, 'blue', 'orange'))
```




```r
squared.logistic <- glm(y ~ poly(x1, 2) + poly(x2, 2),
                        data = data.frame(x1, x2, 'y' = as.factor(y)),
                        family = "binomial")
squared.logistic.predictions <- predict(squared.logistic, data.frame(x1, x2)) > 0
mean(squared.logistic.predictions == y)
```

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

```r
plot(x1, x2, col = ifelse(squared.logistic.predictions, 'blue', 'orange'))
```



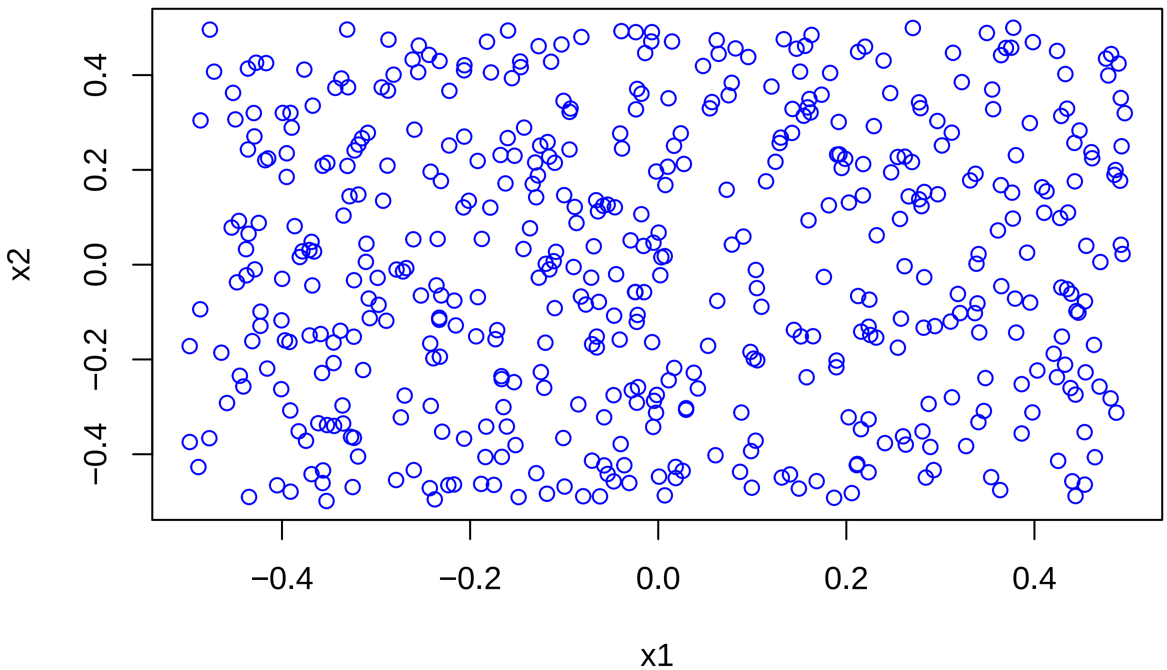
Part G

```
svm.linear <- svm(y ~ .,
                  data = data.frame(x1, x2, 'y' = as.factor(y)),
                  kernel = 'linear',
                  scale = FALSE,
                  gamma = 1,
                  cost = 1)

svm.linear.predictions <- predict(svm.linear, data.frame(x1, x2))
mean(svm.linear.predictions == y)
```

```
## [1] 0.522
```

```
plot(x1, x2, col = ifelse(svm.linear.predictions == 0, 'blue', 'orange'))
```



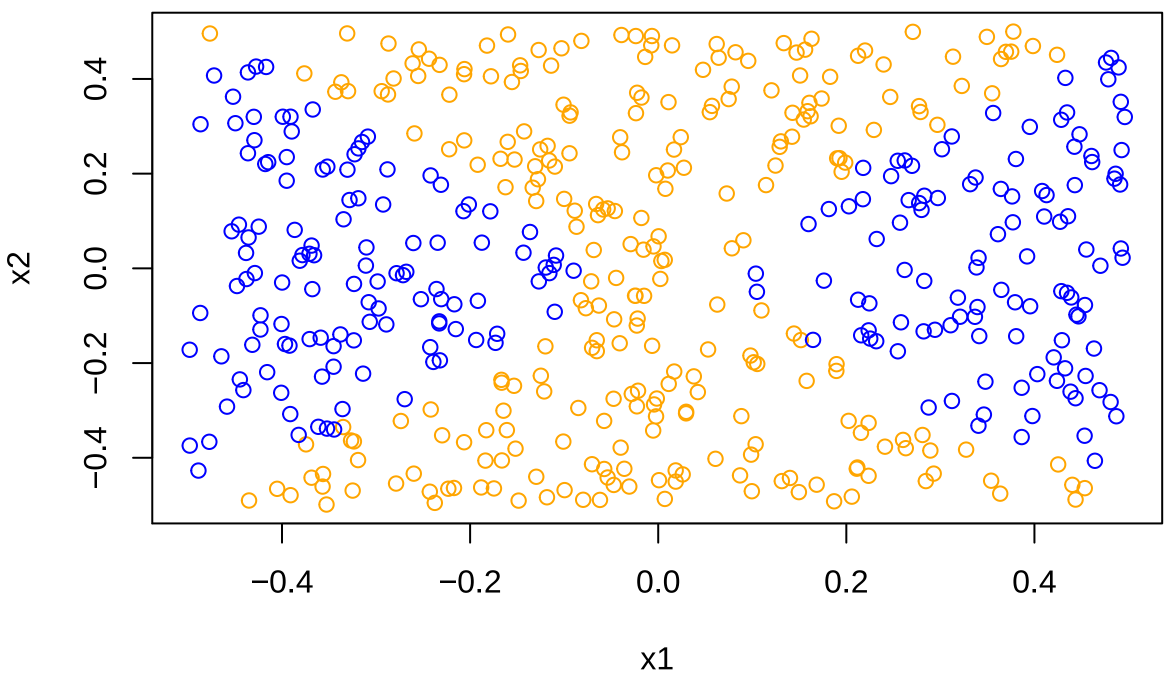
Part H

```
svm.radial <- svm(y ~ .,
                  data = data.frame(x1, x2, 'y' = as.factor(y)),
                  kernel = "radial",
                  gamma = 1,
                  cost = 1)

svm.radial.predictions <- as.integer(predict(svm.radial, data.frame(x1, x2))) - 1
mean(svm.radial.predictions == y)
```

```
## [1] 0.972
```

```r
plot(x1, x2, col = ifelse(svm.radial.predictions, 'blue', 'orange'))
```

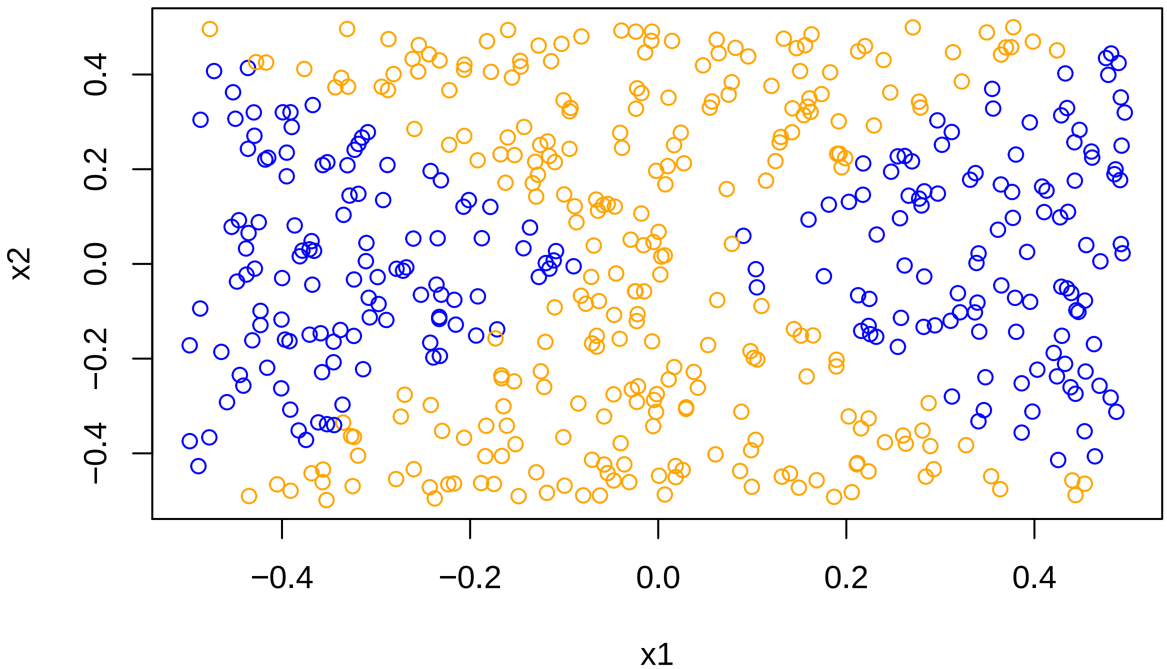



```r
svm.poly <- svm(y ~ .,
                data = data.frame(x1, x2, 'y' = as.factor(y)),
                kernel = "polynomial",
                degree = 2,
                gamma = 1,
                cost = 1)

svm.poly.predictions <- as.integer(predict(svm.poly, data.frame(x1, x2))) - 1
mean(svm.poly.predictions == y)
```

```
## [1] 0.972
```

```r
plot(x1, x2, col = ifelse(svm.poly.predictions, 'blue', 'orange'))
```



Part I

```
res <- rbind('logistic w/ x1 + x2' = mean(base.logistic.predictions == y),
             'logistic w/ x1*x2' = mean(multiplied.logistic.predictions==y),
             'logistic w/ x1^2 + x2^2' = mean(squared.logistic.predictions==y),
             'svm linear' = mean(svm.linear.predictions==y),
             'svm radial' = mean(svm.radial.predictions==y),
             'svm polynomal (n = 2)' = mean(svm.poly.predictions == y))
colnames(res) <- c("Accuracy")
res
```

```
##                         Accuracy
## logistic w/ x1 + x2        0.570
## logistic w/ x1*x2          0.572
## logistic w/ x1^2 + x2^2    1.000
## svm linear                 0.522
## svm radial                 0.972
## svm polynomal (n = 2)      0.972
```

We can see above that using logistic regression with $X_1^2 + X_2^2$ achieves the best accuracy of 100%. This makes sense because in reality, that was the formula used to determine which data points belonged to which class. Then, SVM with radial and polynomial with degree 2 both had accuracy of 97.2%. The rest of the methods I tried didn't do a good job at all; they just predicted most, if not all, of the points to be in the same class.

Question 3 (Exercise 8 from Section 9.7)

```
library(ISLR)
data(OJ)
```

Part A

```
set.seed(999)
i <- sample(seq(1, dim(OJ)[1]), 800)
```

```
OJ.train <- OJ[i, ]
dim(OJ.train)
```

```
## [1] 800  18
```

```
OJ.test <- OJ[-i, ]
dim(OJ.test)
```

```
## [1] 270  18
```

Part B

```
svm.linear <- svm(Purchase ~ .,
                  data = OJ.train,
                  kernel = 'linear',
                  scale = TRUE,
                  cost = .01)
summary(svm.linear)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "linear", cost = 0.01,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  447
##
##  ( 223 224 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

There are 447 support vectors in total, with 223 of them being CH and 224 of them being MM.

Part C

```
svm.linear.training.error <- mean(predict(svm.linear, OJ.train) != OJ.train$Purchase)
svm.linear.training.error
```

```
## [1] 0.17375
```

```
svm.linear.testing.error <- mean(predict(svm.linear, OJ.test) != OJ.test$Purchase)
svm.linear.testing.error
```

```
## [1] 0.137037
```

Part D

```
set.seed(999)
linear.tune.out <- tune(svm,
                        Purchase ~ .,
                        data = OJ.train,
                        kernel = "linear",
                        ranges = list(cost = c(seq(.01, 10, by = .1), 10)))
```

```
linear.tune.out$best.parameters
```

```
##   cost
## 2 0.11
```

Part E

```
linear.best.svm <- linear.tune.out$best.model
```

```
linear.best.svm.training.error <- mean(predict(linear.best.svm, OJ.train) != OJ.train$Purchase)
linear.best.svm.training.error
```

```
## [1] 0.1725
```

```
linear.best.svm.testing.error <- mean(predict(linear.best.svm, OJ.test) != OJ.test$Purchase)
linear.best.svm.testing.error
```

```
## [1] 0.1407407
```

Part F

```r
radial.svm <- svm(Purchase ~ .,
                  data = OJ.train,
                  kernel = 'radial',
                  scale = TRUE,
                  cost = .01)
summary(radial.svm)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "radial", cost = 0.01,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##
## Number of Support Vectors:  637
##
##  ( 318 319 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

There are 637 support vectors in total, with 318 of them being CH and 319 of them being MM.

```r
radial.svm.training.error <- mean(predict(radial.svm, OJ.train) != OJ.train$Purchase)
radial.svm.training.error
```

```
## [1] 0.3975
```

```r
radial.svm.testing.error <- mean(predict(radial.svm, OJ.test) != OJ.test$Purchase)
radial.svm.testing.error
```

```
## [1] 0.3666667
```

```r
set.seed(999)
radial.tune.out <- tune(svm,
                        Purchase ~ .,
                        data = OJ.train,
                        kernel = "radial",
                        ranges = list(cost = c(seq(.01, 10, by = .1), 10)))
```

```r
radial.tune.out$best.parameters
```

```
##    cost
## 16 1.51
```

```r
radial.best.svm <- radial.tune.out$best.model
```

```r
radial.best.svm.training.error <- mean(predict(radial.best.svm, OJ.train) != OJ.train$Purchase)
radial.best.svm.training.error
```

```
## [1] 0.15375
```

```r
radial.best.svm.testing.error <- mean(predict(radial.best.svm, OJ.test) != OJ.test$Purchase)
radial.best.svm.testing.error
```

```
## [1] 0.1407407
```

Part G

```r
poly.svm <- svm(Purchase ~ .,
                data = OJ.train,
                kernel = 'polynomial',
                degree = 2,
                scale = TRUE,
                cost = .01)
summary(poly.svm)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "polynomial",
##     degree = 2, cost = 0.01, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.01
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  644
##
##  ( 318 326 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

There are 644 support vectors in total, with 318 of them being CH and 326 of them being MM.

```r
poly.svm.training.error <- mean(predict(poly.svm, OJ.train) != OJ.train$Purchase)
poly.svm.training.error
```

```
## [1] 0.39375
```

```r
poly.svm.testing.error <- mean(predict(poly.svm, OJ.test) != OJ.test$Purchase)
poly.svm.testing.error
```

```
## [1] 0.362963
```

```r
set.seed(999)
poly.tune.out <- tune(svm,
                      Purchase ~ .,
                      data = OJ.train,
                      kernel = "polynomial",
                      degree = 2,
                      ranges = list(cost = c(seq(.01, 10, by = .1), 10)))
```

```r
poly.tune.out$best.parameters
```

```
##    cost
## 40 3.91
```

```r
poly.best.svm <- poly.tune.out$best.model
```

```r
poly.best.svm.training.error <- mean(predict(poly.best.svm, OJ.train) != OJ.train$Purchase)
poly.best.svm.training.error
```

```
## [1] 0.15625
```

```r
poly.best.svm.testing.error <- mean(predict(poly.best.svm, OJ.test) != OJ.test$Purchase)
poly.best.svm.testing.error
```

```
## [1] 0.1666667
```

Part H

```r
res <- rbind("linear" = c(linear.best.svm.training.error, linear.best.svm.testing.error),
             "radial" = c(radial.best.svm.training.error, radial.best.svm.testing.error),
             "poly" = c(poly.best.svm.training.error, poly.best.svm.testing.error))
colnames(res) <- c("Training Error", "Testing Error")
res
```

```
##        Training Error Testing Error
## linear        0.17250     0.1407407
## radial        0.15375     0.1407407
## poly          0.15625     0.1666667
```

Since radial has both the lowest training and testing error, radial seemed to give the best results on our data.