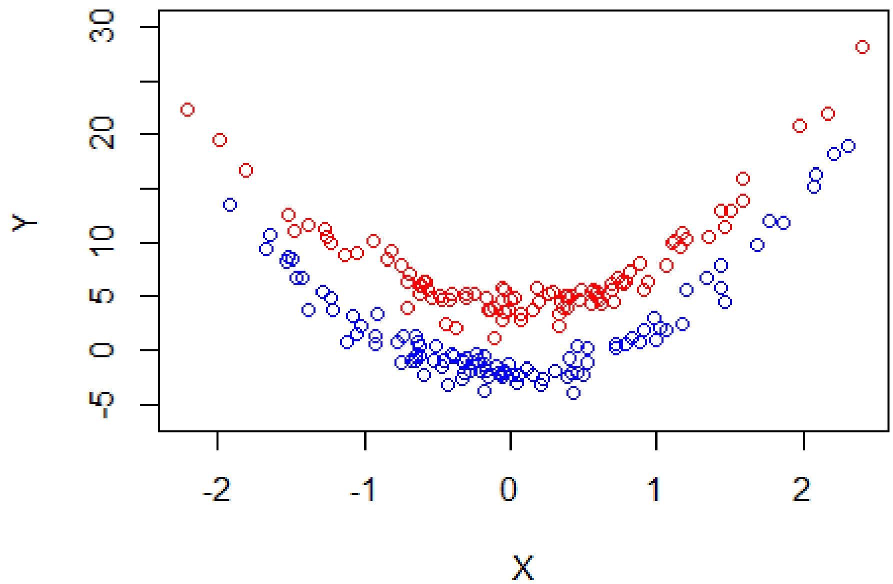
HW4-MATH4323

anthonycastillo ID:1670011

2022-10-17

Question 1 (a):
You are tested positive for cancer but actually is a case of false positive. The patient will then start chemotherapy which could cause damage to the patient even more.
Question 1 (b):
If passing through airport security and the metal detector detects metal but non is actually present this is a case of false positive. If it were actually a false negative it could be a security risk and could put people in danger at the airport.
Question 1 (c):
Whether a pregnancy test is false positive and false negative could alter someone life forever
Question 2 (a):

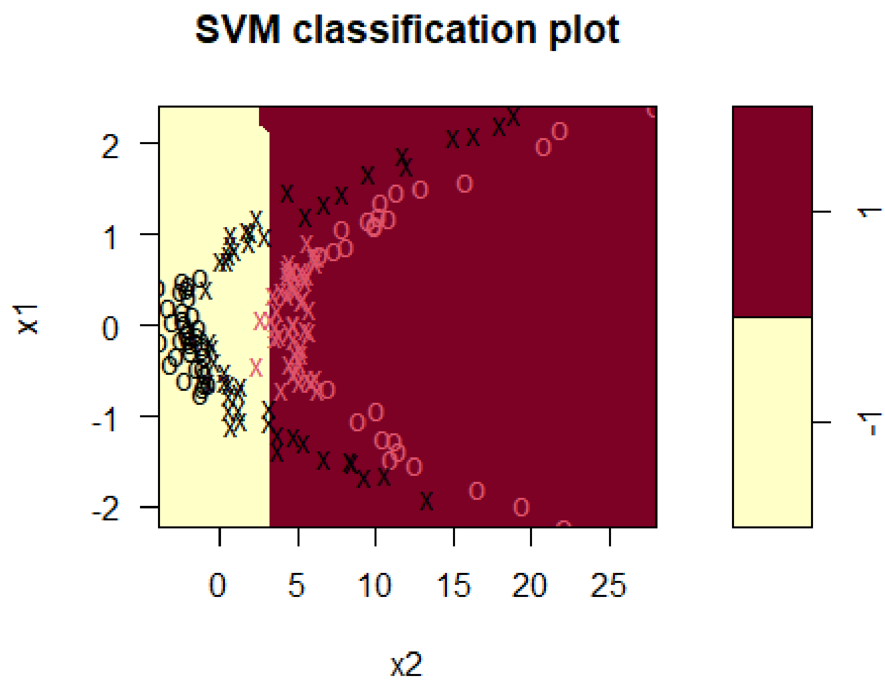
```
set.seed(1)
x1 <- rnorm(200)
x2 <- 4 * x1^2 + 1 + rnorm(200)
y <- as.factor(c(rep(1,100), rep(-1,100)))
x2[y==1] <- x2[y==1]+3
x2[y==-1] <- x2[y==-1]-3
plot(x1[y==1],x2[y==1], col = "red", xlab = "X", ylab = "Y", ylim = c(-6,30))
points(x1[y==-1],x2[y==-1], col = "blue")
```



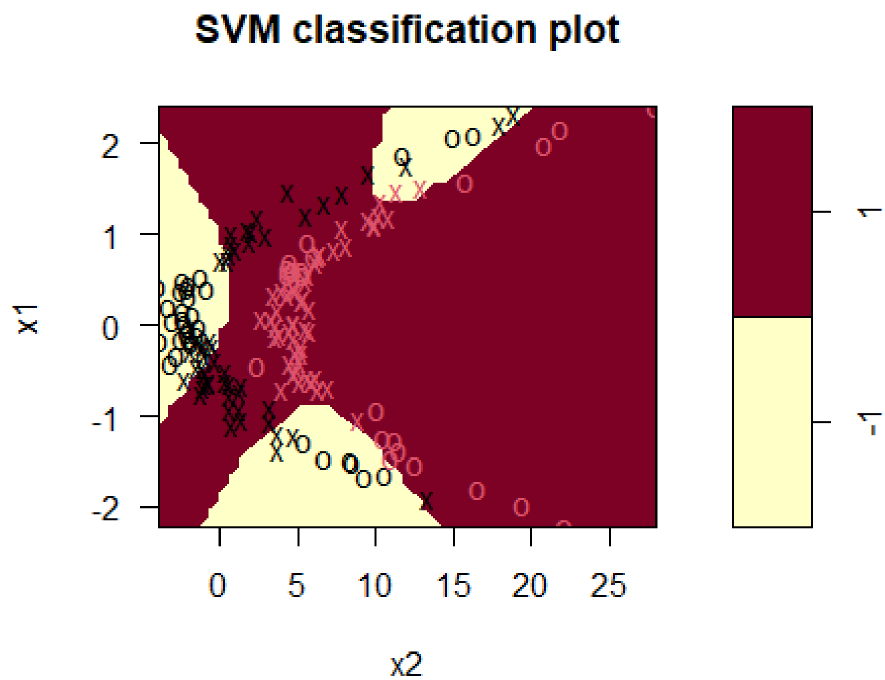

```
dat <- data.frame(x1,x2,y)

#separate data 80/20
set.seed(1)
train <- sample(200,200*.8)
dat.train <- dat[train,]
dat.test <- dat[-train,]
```

Question 2 (b):

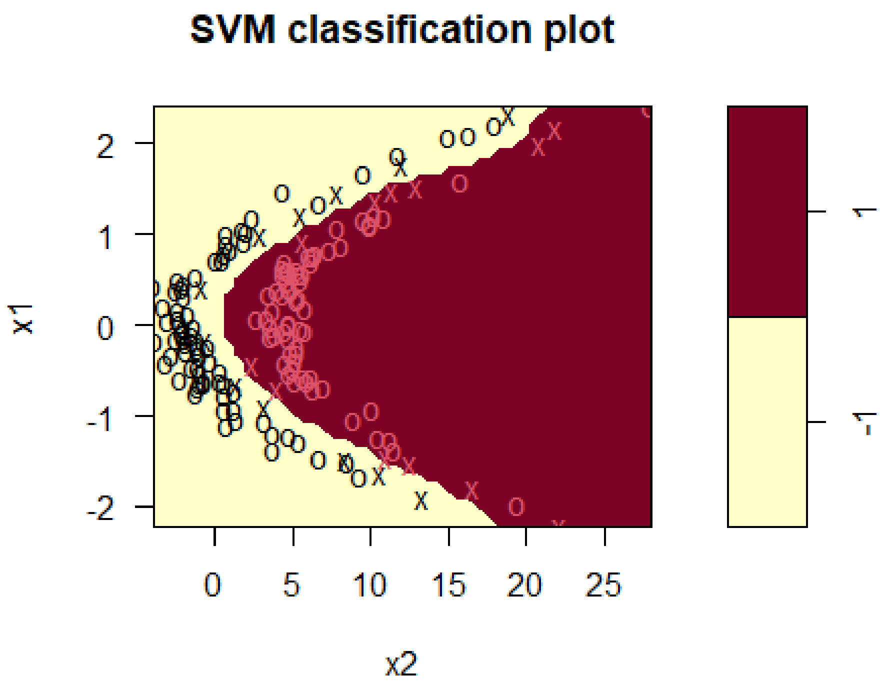
```
library(e1071)

## Warning: package 'e1071' was built under R version 4.1.3

set.seed(1)
#linear model
linear.tune <- tune(method = svm, y~.,
                    data = dat.train,
                    kernel = "linear",
                    ranges = list(cost=c(0.001,0.01,0.1,1,5,10,10,100)))
print(linear.tune$best.parameters)

##   cost
## 5    5

plot(linear.tune$best.model,dat.train, main = "Optimal Linear Model")
```




```
#polynomial model
set.seed(1)
poly.tune <- tune(method = svm,
                  y~., data = dat.train,
                  kernel = "polynomial",
                  ranges = list(degree = seq(2,5),
                                cost = c(0.001,0.01,0.1,1,5,10,10,100)))
print(poly.tune$best.parameters)

##    degree cost
## 22      3   10

plot(poly.tune$best.model,dat.train, main = "Optimal Polynomial Model")
```




```r
#radial model
set.seed(1)
radial.tune <- tune(method = svm,
                    y~., data = dat.train,
                    kernel = "radial",
                    ranges = list(cost=c(0.001,0.01,0.1,1,5,10,10,100),
                                  gamma = c(0.5,1,2,3,4)))
print(radial.tune$best.parameters)

##   cost gamma
## 6   10   0.5

plot(radial.tune$best.model,dat.train, main = "Optimal Radial Model")
```



Question 2 (c):

```
#linear model training error
linear.pred <- predict(linear.tune$best.model,newdata = dat.train)
mean(linear.pred != dat.train$y)

## [1] 0.1625

#polynomial training error
poly.pred <- predict(poly.tune$best.model, newdata = dat.train)
mean(poly.pred != dat.train$y)

## [1] 0.24375

#radial training error
rad.pred <- predict(radial.tune$best.model, newdata = dat.train)
mean(rad.pred != dat.train$y)

## [1] 0

#linear test error
l.test.pred <- predict(linear.tune$best.model, newdata = dat.test)
mean(l.test.pred != dat.test$y)

## [1] 0.175
```

```
#polynomial test error
p.test.pred <- predict(poly.tune$best.model, newdata = dat.test)
mean(p.test.pred != dat.test$y)

## [1] 0.15

#radial test error
r.test.pred <- predict(radial.tune$best.model, newdata = dat.test)
mean(r.test.pred != dat.test$y)

## [1] 0
```

The best training error is 0 coming from the radial model
The best test error is 0 coming from the radial model
Question 3 (a):

```
library("ISLR")

## Warning: package 'ISLR' was built under R version 4.1.3

newAuto <- data.frame(Auto)
newAuto$mpg01 <- ifelse(newAuto$mpg > median(newAuto$mpg),1,0)
newAuto$mpg01 <- as.factor(newAuto$mpg01)
```

Question 3 (b):

```
set.seed(1)
auto.tune <- tune(method = svm, mpg01~.,
                  data = newAuto,
                  kernel = "linear",
                  ranges = list(cost=c(0.01,0.1,1,5,10,100)))
#best cost value
print(auto.tune$best.parameters)

##   cost
## 3    1

#test error/cross validation error
print(auto.tune$best.performance)

## [1] 0.01025641

#train error
auto.pred <- predict(auto.tune$best.model)
mean(auto.pred != newAuto$mpg01)

## [1] 0.00255102
```

Question 3 (c):

```r
set.seed(1)
auto.poly.tune <- tune(method = svm, mpg01~.,
                       data = newAuto,
                       kernel = "polynomial",
                       ranges = list(cost =c(0.01,0.1,1,5,10,100), degree =
seq(2,4)))
#best cost value and degree
print(auto.poly.tune$best.parameters)

##   cost degree
## 6  100      2

#test error/cross-validation error
print(auto.poly.tune$best.performance)

## [1] 0.3013462

#training error
auto.poly.pred <- predict(auto.poly.tune$best.model)
mean(auto.poly.pred != newAuto$mpg01)

## [1] 0.2831633
```

Question 3 (d):

```r
set.seed(1)
auto.radial.tune <- tune(method = svm, mpg01~.,
                         data = newAuto,
                         kernel = "radial",
                         ranges = list(cost = c(0.01,0.1,1,5,10,100),
                                       gamma = c(0.01,0.1,1,5)))
#best cost value and gamma
print(auto.radial.tune$best.parameters)

##   cost gamma
## 6  100  0.01

#test error/cross-validation error
print(auto.radial.tune$best.performance)

## [1] 0.01282051

#training error
auto.radial.pred <- predict(auto.radial.tune$best.model)
mean(auto.radial.pred != newAuto$mpg01)

## [1] 0.00255102
```

Question 3 (e):
The best cross-validation error is from the linear model with the error of 0.01025641
The best training error is from the linear model and radial both having the error of

0.00255102
Question 4 (a):

```
library(ISLR)
set.seed(1)
oj.rows <- nrow(OJ)
oj.sample <- sample(oj.rows,800)
oj.train <- OJ[oj.sample,]
oj.test <- OJ[-oj.sample,]
```

Question 4 (b):

```
set.seed(1)
oj.linear.tune <- tune(method = svm, Purchase~.,
                       data = oj.train,
                       kernel = "linear",
                       ranges = list(cost = c(0.01,0.05,0.1,0.5,1,10)))
#training error
oj.lin.pred <- predict(oj.linear.tune$best.model, newdata = oj.train)
mean(oj.lin.pred != oj.train$Purchase)

## [1] 0.165

#test error
oj.test.pred <- predict(oj.linear.tune$best.model,newdata = oj.test)
mean(oj.test.pred != oj.test$Purchase)

## [1] 0.1555556
```

Question 4 (c):

```
set.seed(1)
oj.poly.tune <- tune(method = svm, Purchase~.,
                     data = oj.train,
                     kernel = "polynomial",
                     ranges = list(degree = c(3), cost =
c(0.01,0.05,0.1,0.5,1,10)))
#training error
oj.poly.pred <- predict(oj.poly.tune$best.model, newdata = oj.train)
mean(oj.poly.pred != oj.train$Purchase)

## [1] 0.15375

#test error
oj.pt.pred <- predict(oj.poly.tune$best.model, newdata = oj.test)
mean(oj.pt.pred != oj.test$Purchase)

## [1] 0.2222222
```

Question 4 (d):

```
set.seed(1)
oj.radial.tune <- tune(method = svm, Purchase~.,
                       data = oj.train,
                       kernel = "radial",
                       ranges = list(gamma = 2, cost =
c(0.01,0.05,0.1,0.5,1,10)))
#training error
oj.rad.pred <- predict(oj.radial.tune$best.model,newdata = oj.train)
mean(oj.rad.pred != oj.train$Purchase)

## [1] 0.10375

#test error
oj.rt.pred <- predict(oj.radial.tune$best.model,newdata = oj.test)
mean(oj.rt.pred != oj.test$Purchase)

## [1] 0.1962963
```

Question 4 (e):
The best training error is 0.10375 coming from the radial model
The best test error is 0.1555556 coming from the linear model
Question 5 (a):

```
set.seed(1)
iris.row <- nrow(iris)
iris.sample <- sample(1:iris.row, iris.row*0.80)
iris.train <- iris[iris.sample,]
iris.test <- iris[-iris.sample,]
```

Question 5 (b):

```
#linear model
set.seed(1)
iris.lin.tune <- tune(method = svm, Species~.,
                      data = iris.train,
                      kernel = "linear",
                      ranges = list(cost = c(0.01,0.1,1,5,10,100)))
#optimal linear model
print(iris.lin.tune$best.parameters)

##   cost
## 3    1

#polynomial model
set.seed(1)
iris.poly.tune <- tune(method = svm, Species~.,
```

```
                           data = iris.train,
                           kernel = "polynomial",
                           ranges = list(cost = c(0.01,0.1,1,5,10,100), degree =
seq(2,5)))
#optimal polynomial model
print(iris.poly.tune$best.parameters)

##    cost degree
## 11   10      3

#radial model
set.seed(1)
iris.rad.tune <- tune(method = svm, Species~.,
                      data = iris.train,
                      kernel = "radial",
                      ranges = list(cost = c(0.01,0.1,1,5,10,100), gamma =
c(0.01,0.1,1,5)))
#optimal polynomial model
print(iris.rad.tune$best.parameters)

##   cost gamma
## 5   10  0.01
```

Question 5 (c):

```
#training error linear
iris.lin.pred <- predict(iris.lin.tune$best.model,iris.train)
mean(iris.lin.pred != iris.train$Species)

## [1] 0.01666667

#training error polynomial
iris.poly.pred <- predict(iris.poly.tune$best.model,iris.train)
mean(iris.poly.pred != iris.train$Species)

## [1] 0.025

#training error radial
iris.rad.pred <- predict(iris.rad.tune$best.model,iris.train)
mean(iris.rad.pred != iris.train$Species)

## [1] 0.03333333

#test error linear
iris.lin.pred.test <- predict(iris.lin.tune$best.model,iris.test)
mean(iris.lin.pred.test != iris.test$Species)

## [1] 0

#test error polynomial
iris.poly.pred.test <- predict(iris.lin.tune$best.model,iris.test)
mean(iris.poly.pred.test != iris.test$Species)
```

```
## [1] 0

#test error radial
iris.rad.pred.test <- predict(iris.lin.tune$best.model,iris.test)
mean(iris.rad.pred.test != iris.test$Species)

## [1] 0
```

The best training error is 0.01666667 coming from the linear model
The best test error is 0 from all three models
Question 5 (d):
The two types that can be used for classification is One-versus-One and One-versus-All classifications.The one-vs-one will construct SVMs for all possible pairs of classes while the one-vs-all will fit 2 different SVM classifiers each class versus the rest.
The svm() function will use the one-versus-one approach if more than 2 classes.