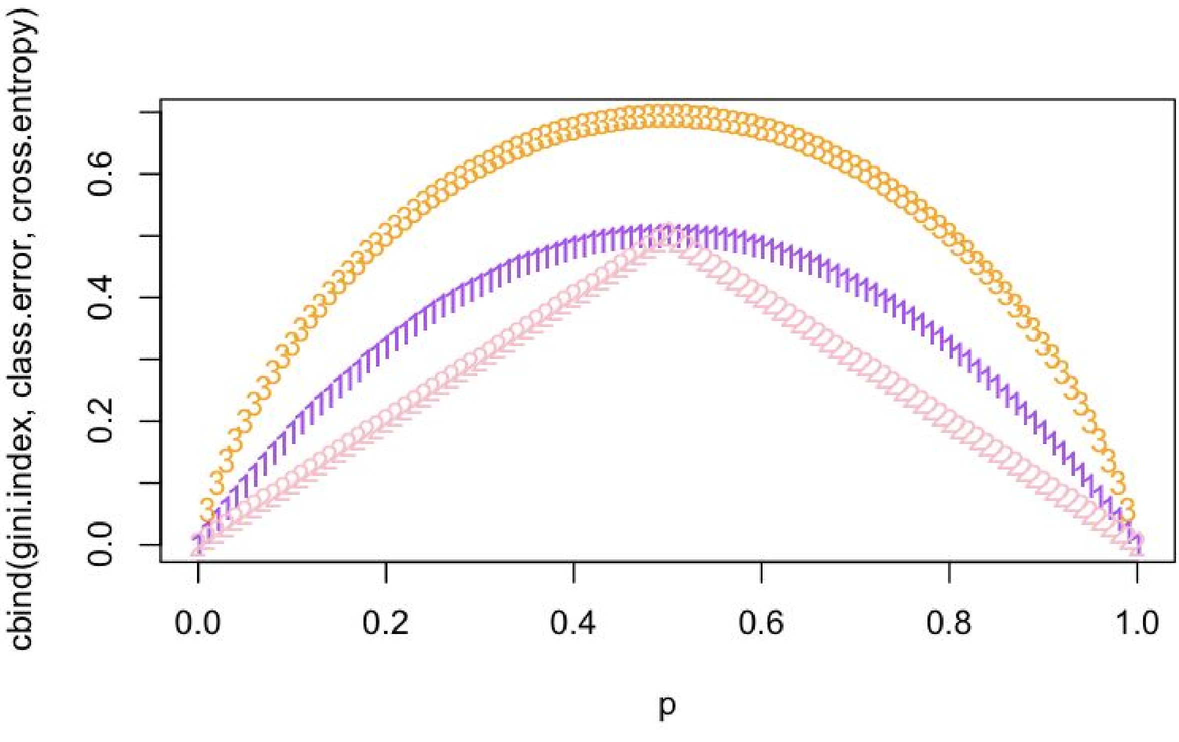
DS502 - HW5
Vandana Anand
Kratika Agrawal

Question 1:

```
p <- seq(0, 1, 0.01)
gini.index <- 2 * p * (1 - p)
class.error <- 1 - pmax(p, 1 - p)
cross.entropy <- - (p * log(p) + (1 - p) * log(1 - p))
matplot(p, cbind(gini.index, class.error, cross.entropy), col = c("purple",
 "pink", "orange"))
```



Question 2:

a. A training set containing a random sample of 800 observations,
and a test set containing the remaining observations was created.

b.

```
> #answer b
> library(tree)
> oj.tree = tree(Purchase~.,data=train_OJ)
> summary(oj.tree)

Classification tree:
tree(formula = Purchase ~ ., data = train_OJ)
Variables actually used in tree construction:
[1] "LoyalCH"     "PriceDiff"   "PriceMM"     "SalePriceMM"
Number of terminal nodes:  9
Residual mean deviance:  0.7247 = 573.2 / 791
Misclassification error rate: 0.1812 = 145 / 800
```

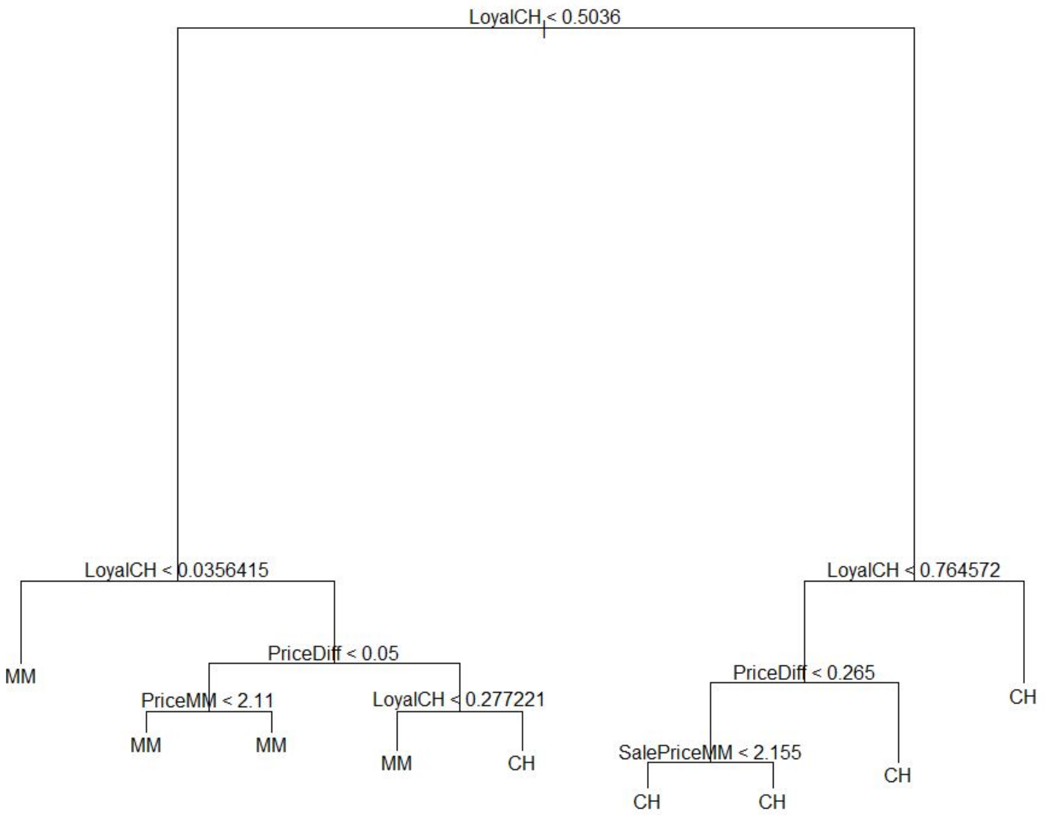
Tree summary suggests that variables “LoyalCH”, “PriceDiff”, “PriceMM”, “SalesPriceMM” are used in tree construction.
Tree has a training error rate of 18.12% and has 9 leaves (terminal nodes).

c.

```
> oj.tree
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 800 1068.00 CH ( 0.61250 0.38750 )
   2) LoyalCH < 0.5036 346  414.30 MM ( 0.28613 0.71387 )
     4) LoyalCH < 0.0356415 57    0.00 MM ( 0.00000 1.00000 ) *
     5) LoyalCH > 0.0356415 289  371.50 MM ( 0.34256 0.65744 )
      10) PriceDiff < 0.05 114  105.90 MM ( 0.17544 0.82456 )
        20) PriceMM < 2.11 89   94.84 MM ( 0.22472 0.77528 ) *
        21) PriceMM > 2.11 25    0.00 MM ( 0.00000 1.00000 ) *
      11) PriceDiff > 0.05 175  240.90 MM ( 0.45143 0.54857 )
        22) LoyalCH < 0.277221 62   66.24 MM ( 0.22581 0.77419 ) *
        23) LoyalCH > 0.277221 113  154.10 CH ( 0.57522 0.42478 ) *
   3) LoyalCH > 0.5036 454  365.70 CH ( 0.86123 0.13877 )
     6) LoyalCH < 0.764572 187  221.10 CH ( 0.72193 0.27807 )
      12) PriceDiff < 0.265 113  154.70 CH ( 0.56637 0.43363 )
        24) SalePriceMM < 2.155 102  141.20 CH ( 0.51961 0.48039 ) *
        25) SalePriceMM > 2.155 11    0.00 CH ( 1.00000 0.00000 ) *
      13) PriceDiff > 0.265 74   25.11 CH ( 0.95946 0.04054 ) *
     7) LoyalCH > 0.764572 267   91.71 CH ( 0.95880 0.04120 ) *
```

Let’s consider a possible terminal node
20) PriceMM < 2.11 89   94.84 MM (0.22472 0.77528) *
This is the node when PriceMM<2.11 with 89 observations belonging to this branch with the residual deviance of 94.84
The prediction is MM with 22.47% of observations taking MM value and is CH with 77.52% of observations taking CH value

d.



We can observe that LoyalCH is the most important variable in the tree as it is the root node and differentiates between customers loyal to brand CH or MM. When LoyalCH (loyalty for Citrus Hill) is less than 0.27, then the model predicts MM, or if LoyalCH is greater than 0.27, the customer chooses MM when it has a lower sales price or there isn't much price difference between the two.

e.

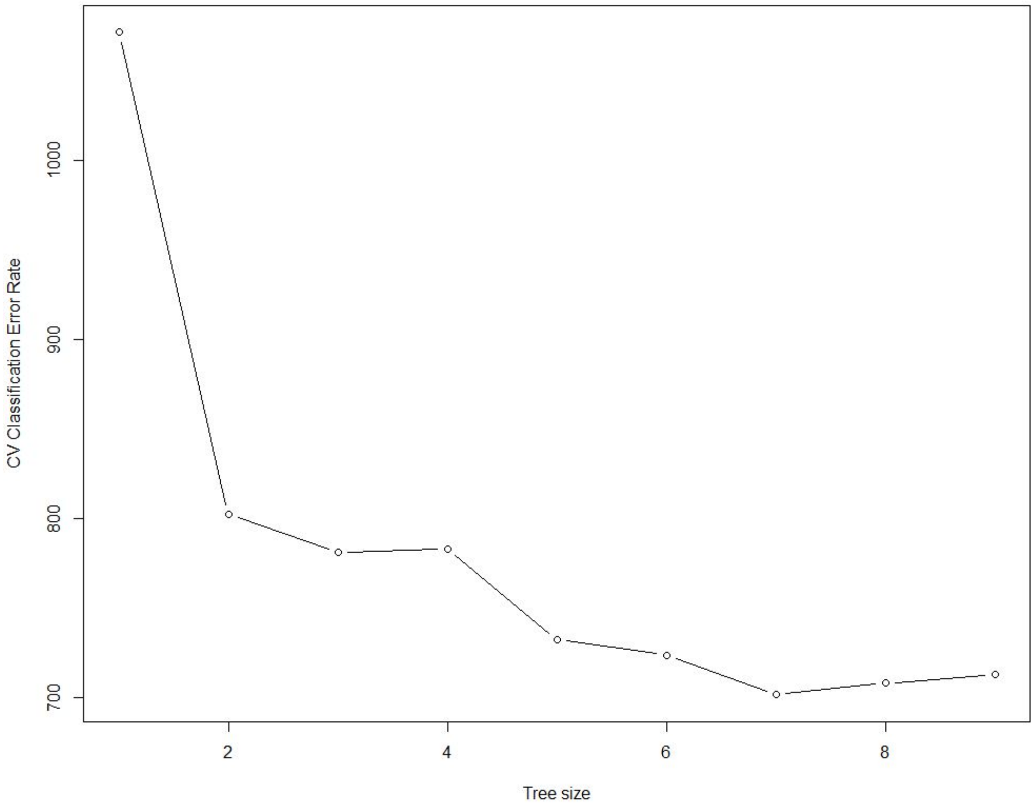
```
> table(test_OJ$Purchase,oj.pred)
    oj.pred
      CH  MM
  CH 148  15
  MM  31  76
```

Test Error Rate = 1-(148+76)/(148+15+31+76) = 0.1703

f.

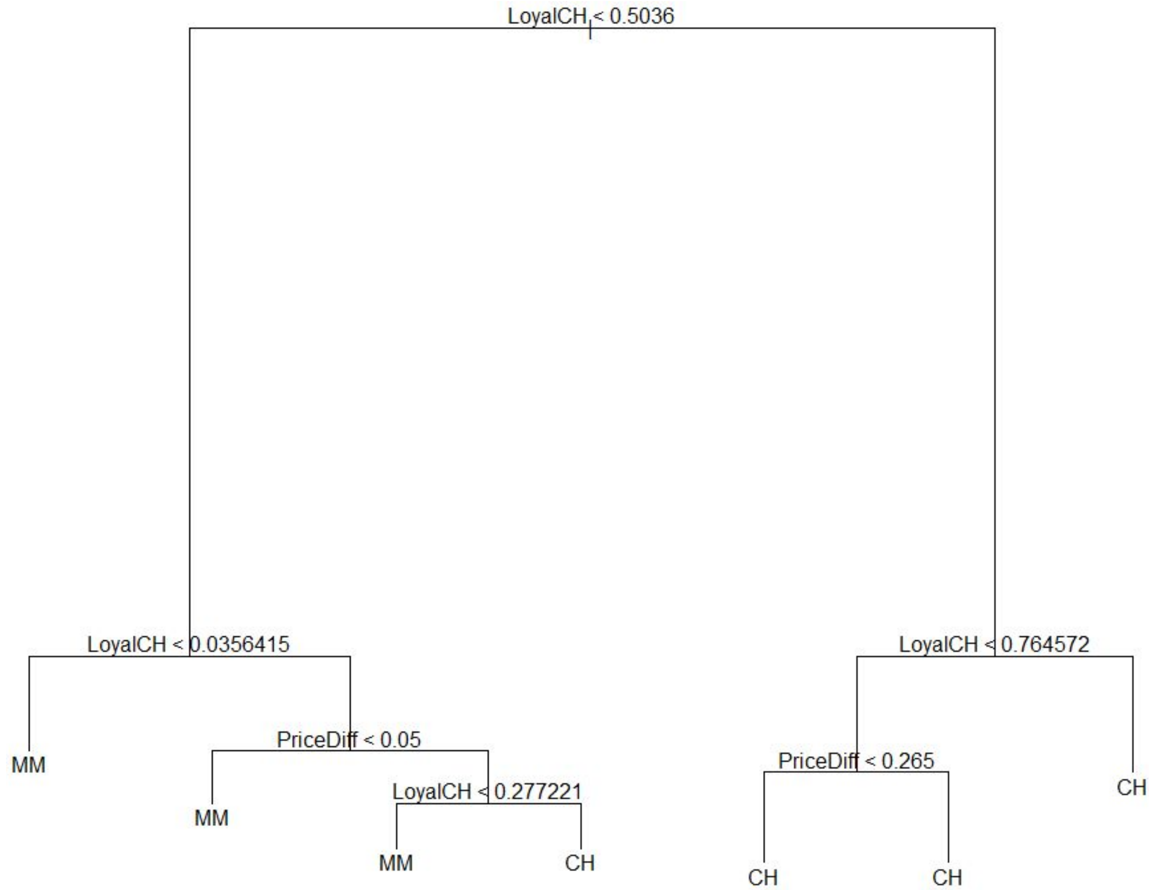
```
> #answer f
> oj.cv=cv.tree(oj.tree,FUN=prune.tree)
> oj.cv
$size
[1] 9 8 7 6 5 4 3 2 1

$dev
[1]  712.9012  708.0688  701.6374  723.8231  732.4337  782.7946  781.3453  802.2203 1071.1720

$k
[1]      -Inf  11.04325  13.40910  20.62771  24.65973  41.31745  42.77232  52.88065 288.25462

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```

g.



h. Cross-Validation Classification Error Rate is minimum for Tree Size=7

i. Pruned Tree for tree size=7



j.

```
> summary(oj.pruned)

Classification tree:
snip.tree(tree = oj.tree, nodes = c(10L, 12L))
Variables actually used in tree construction:
[1] "LoyalCH"   "PriceDiff"
Number of terminal nodes:  7
Residual mean deviance:  0.7537 = 597.7 / 793
Misclassification error rate: 0.1812 = 145 / 800
```

Training error rate for Pruned Tree remains to be the same 18.12% as for the un-pruned tree.

k.

```
> #answer k
> oj.pred.pruned=predict(oj.tree,test_OJ,type="class")
> table(test_OJ$Purchase,oj.pred.pruned)
     oj.pred.pruned
       CH  MM
  CH 148  15
  MM  31  76
```

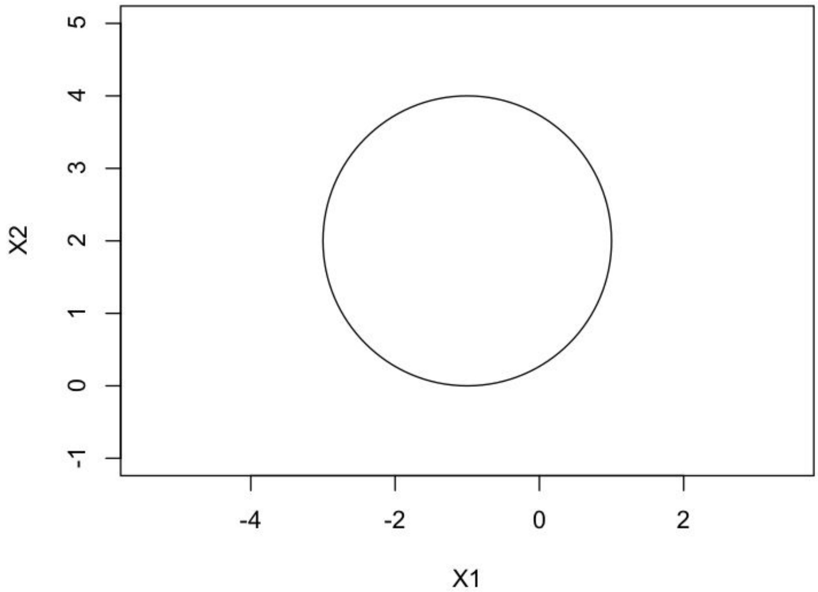
Test Error: 1 - (148+76)/270 = 17.03%

Test Error Rate also remains to be the same for both pruned tree and un-pruned tree.

Question 3:

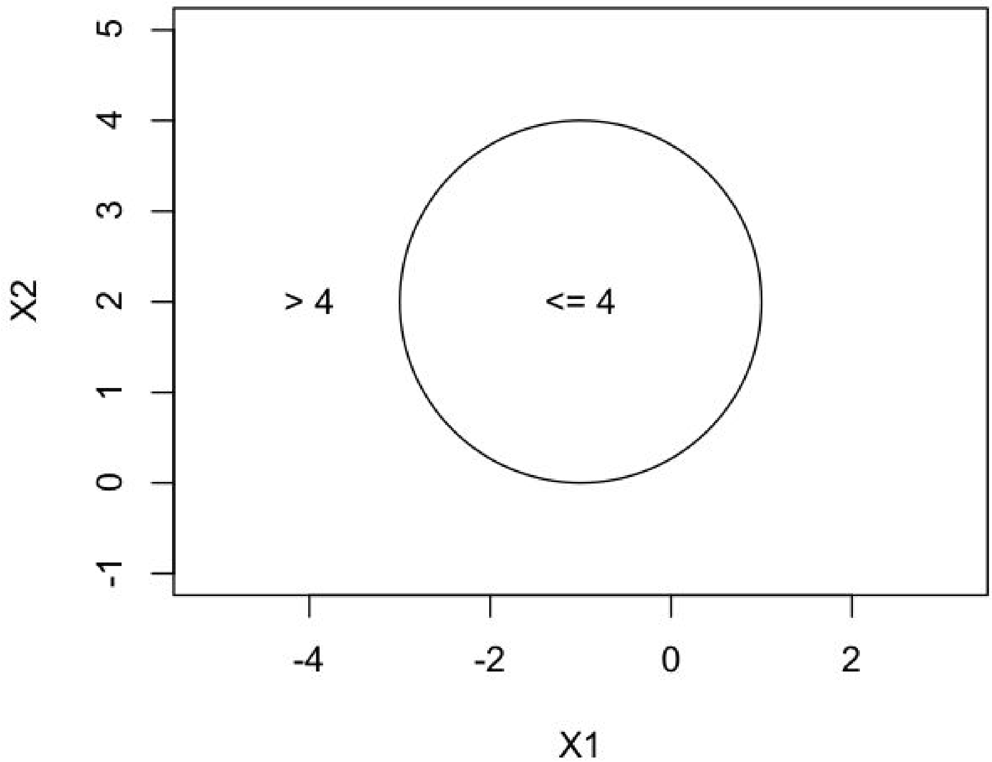
a)

```
plot(NA,NA, type='n', xlim=c(-4,2), ylim= c(-1,5), asp = 1, xlab="X1",
 ylab="X2")
symbols(c(-1), c(2), circles = c(2), add =TRUE, inches = FALSE)
```



b)

```
plot(NA,NA, type='n', xlim=c(-4,2), ylim= c(-1,5), asp = 1, xlab="X1",
 ylab="X2")
symbols(c(-1), c(2), circles = c(2), add =TRUE, inches = FALSE)
text(c(-1), c(2), '<= 4')
text(c(-4), c(2), '> 4')
```

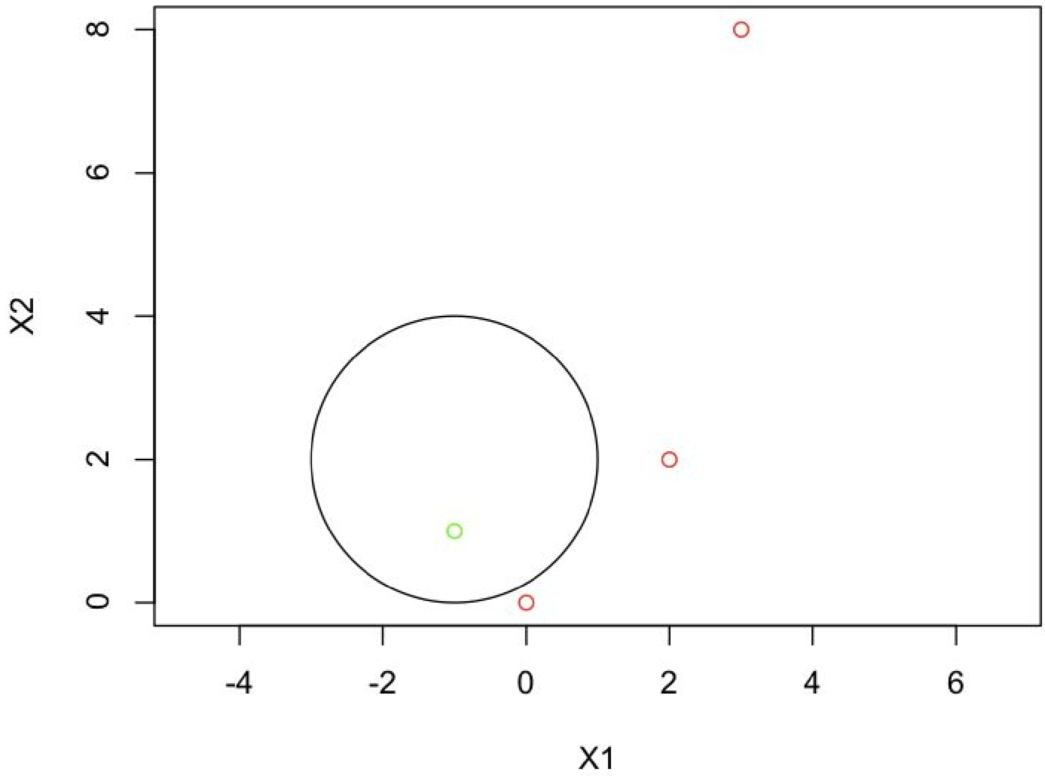


c)

We can replace X1 and X2 in the equation with the coordinates (0, 0), (−1, 1), (2, 2), (3, 8).

(1 + 0)^2 + (2 − 0)^2 = 5, 5 > 4 (red class)
(1 + -1)^2 + (2 − 1)^2 = 1, 1 < 4 (green class)
(1 + 2)^2 + (2 − 2)^2 = 9, 9 > 4 (red class)
(1 + 3)^2 + (2 − 8)^2 = 52, 52 > 4 (red class)

```
plot(c(0,-1, 2, 3), c(0,1,2,8), col = c("red", "green", "red", "red"),
type='p', asp = 1, xlab="X1", ylab="X2")
symbols(c(-1), c(2), circles = c(2), add =TRUE, inches = FALSE)
```

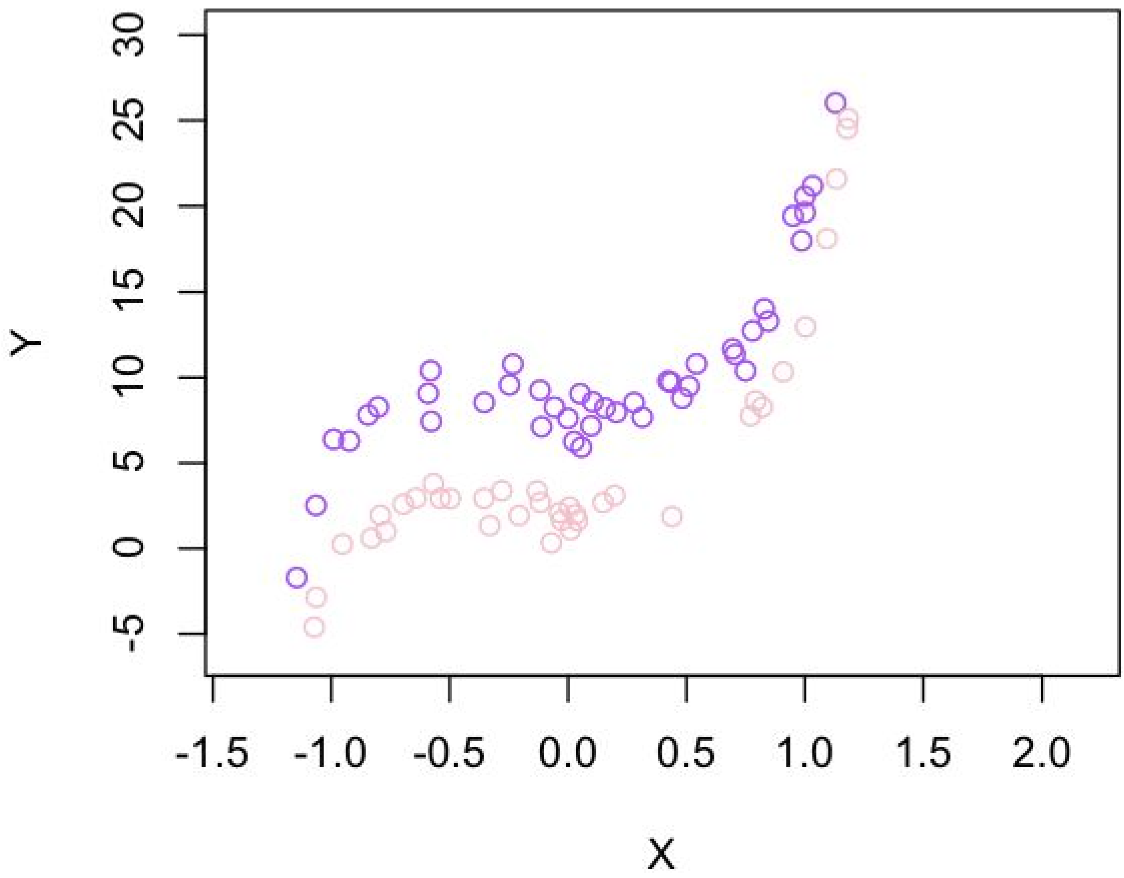


d) the decision boundary, (1 + X1)^ 2 + (2 − X2) ^ 2 > 4 is obviously not linear in terms of X1 and X2 because of the squared terms, but it is linear in terms of X1, (X1)^2, X2, and (X2)^2 since the equation can be expanded, which consists of a sum of quadratic terms:

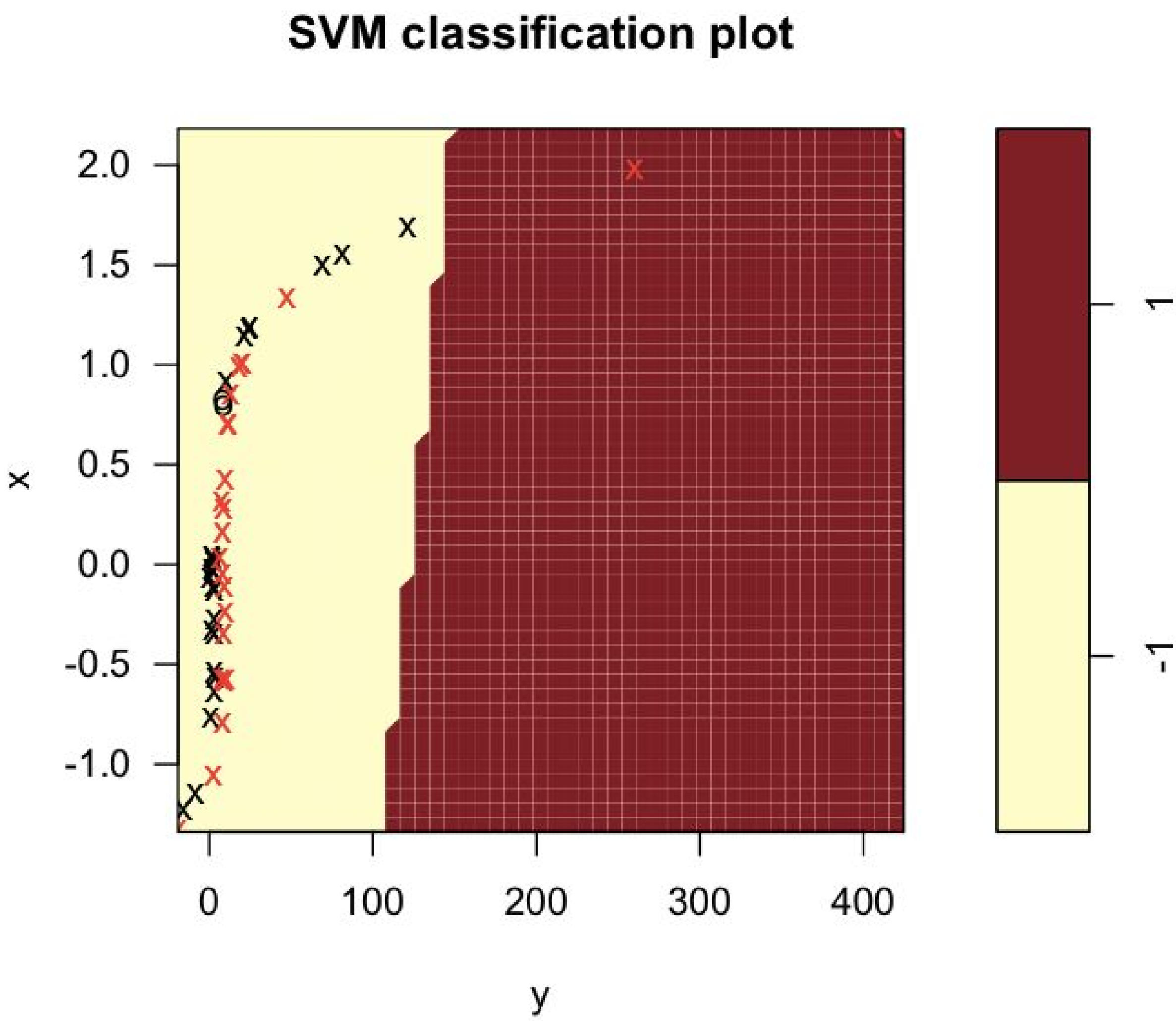
(1 + X1)^ 2 + (2 − X2) ^ 2 > 4
(1 + X1)(1 + X1) + (2 - X2)(2 - X2) > 4
1 + 2X1 + (X1)^2 + 4 - 4X2 + (X2)^2 > 4
5 + 2X1 - 4X2 + (X1)^2 + (X2)^2 > 4

Question 4:

```
# create the data separating the 2 classes
x=rnorm(100)
y= 8*x^5 + 4*x^2 +5 + rnorm(100)
class = sample(100,50)
y[class] = y[class] + 3
y[-class] = y[-class] - 3

plot(x[class], y[class], col = "purple", xlab = "X", ylab = "Y", ylim =
 c(-6,30))
points(x[-class], y[-class], col = "pink")
```



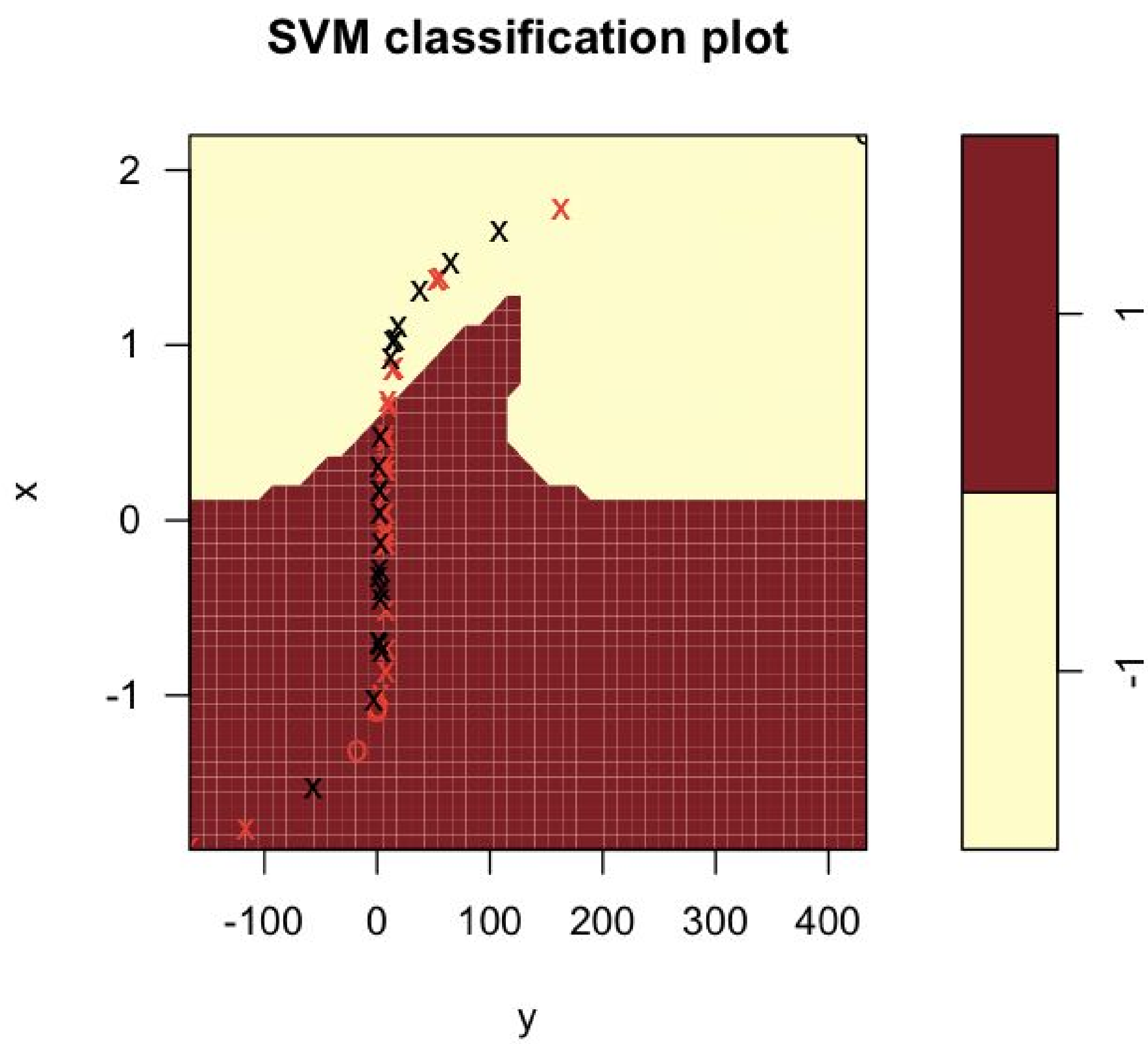

```
# fit the support vector classifier model
z = rep(-1,100)
z[class] = 1
data = data.frame(x=x, y=y, z=as.factor(z))
train = sample(100,50)
data.train=data[train,]
data.test=data[-train,]
svm.linear=svm(z~., data = data.train, kernel ="linear", cost=10)
plot(svm.linear, data.train)
```




```
> table(predict = predict(svm.linear, data.train), truth=data.train$z)
       truth
predict -1  1
     -1 26 22
     1   0  2
```

The support vector classifier made 22 classification errors on the training data.

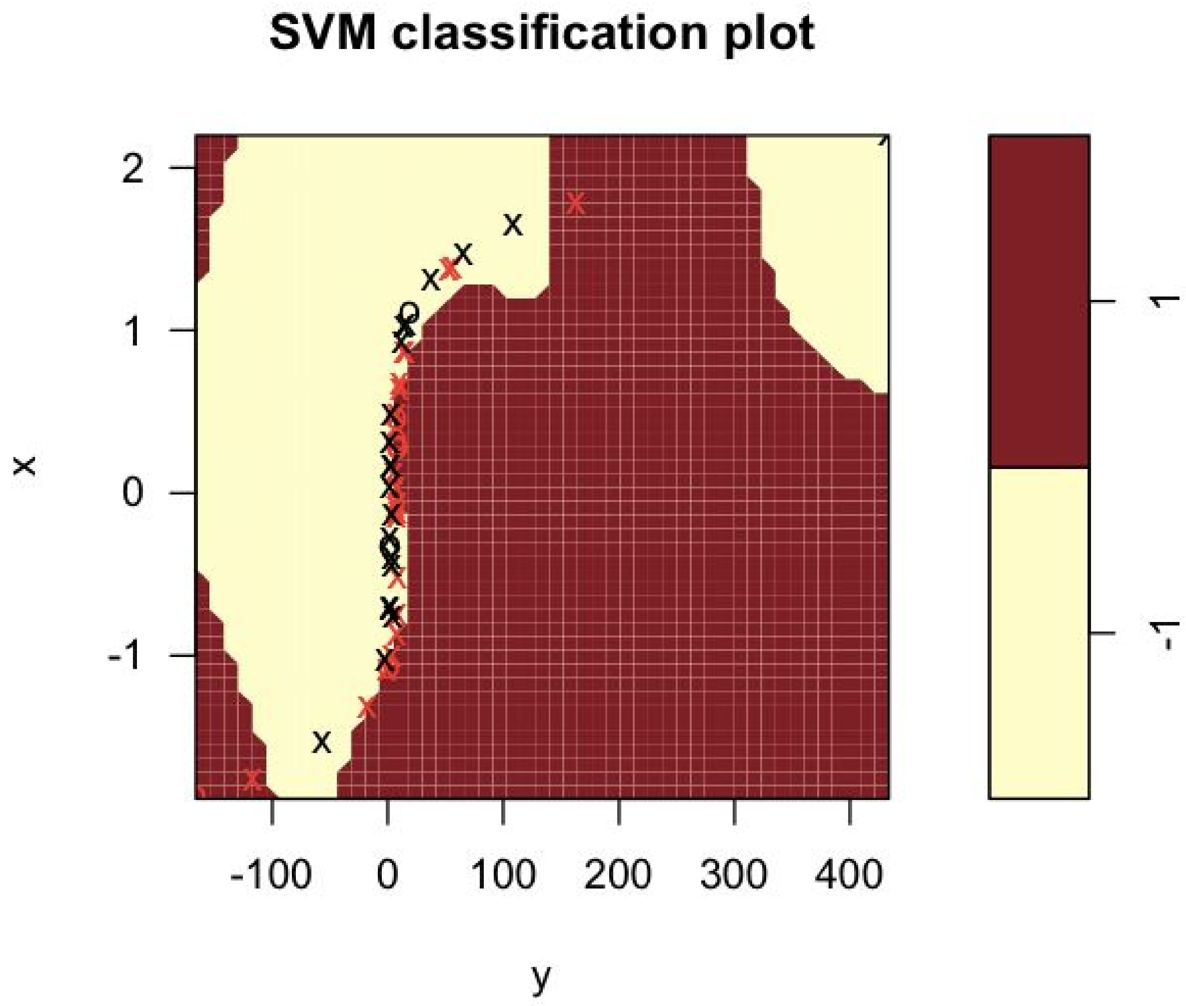
```
# fit the support vector machine with a polynomial kernel
svm.polynomial = svm(z~., data = data.train, kernel ="polynomial", cost =10)
plot(svm.polynomial, data.train)
```




```
> table(predict = predict(svm.polynomial, data.train), truth=data.train$z)
       truth
predict -1  1
     -1  8  6
     1  15 21
```

The support vector machine with a polynomial kernel made 21 classification errors on the training data.

```
# fit the support vector machine with a radial kernel and a gamma of 1
svm.radial = svm(z~., data = data.train, kernel ="radial", gamma=1, cost =10)
plot(svm.radial, data.train)
```



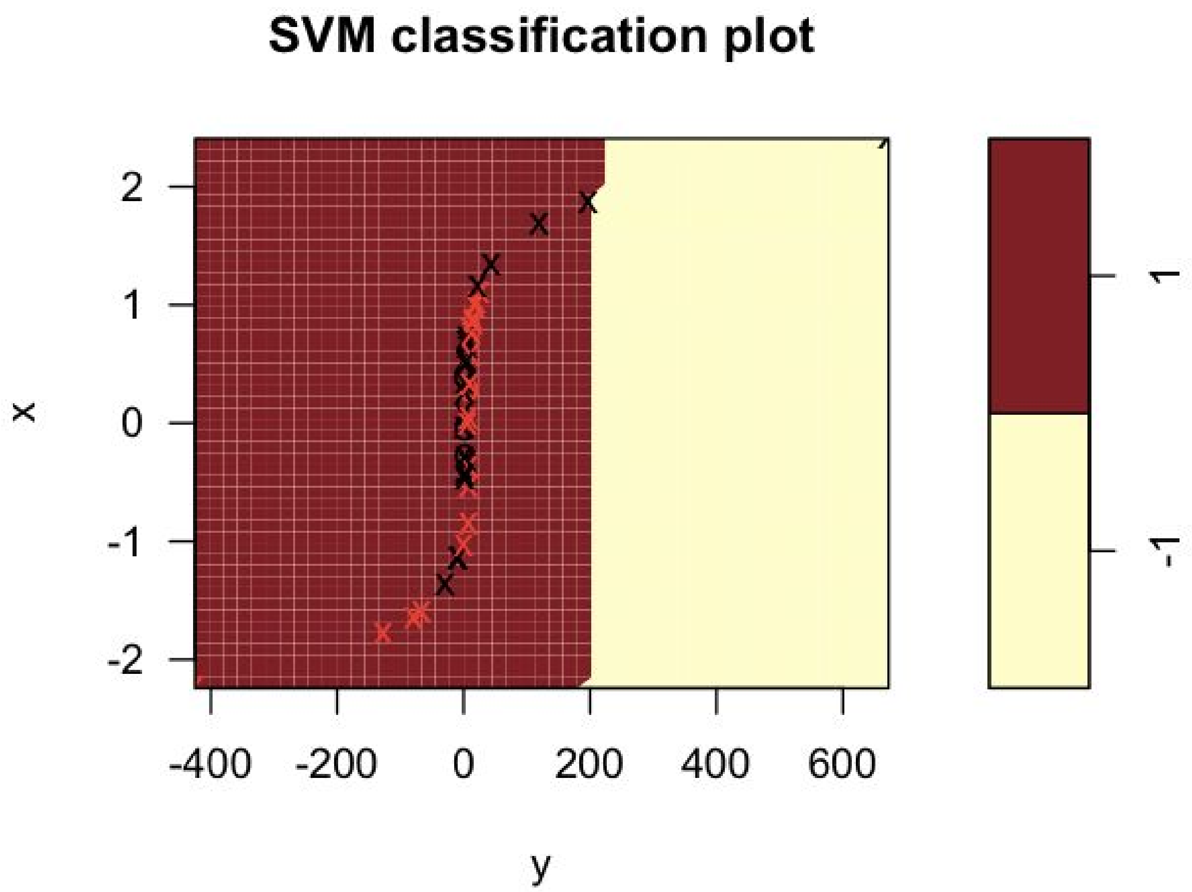

```
> table(predict = predict(svm.radial, data.train), truth=data.train$z)
       truth
predict -1  1
     -1 18  4
     1   5 23
```

The support vector machine with a radial kernel made 9 classification errors on the training data.

Thus, the SVM with polynomial and radial kernels outperforms the SVC because they have lower classification errors than the SVC on the training data.

Now we apply the model to the test data:

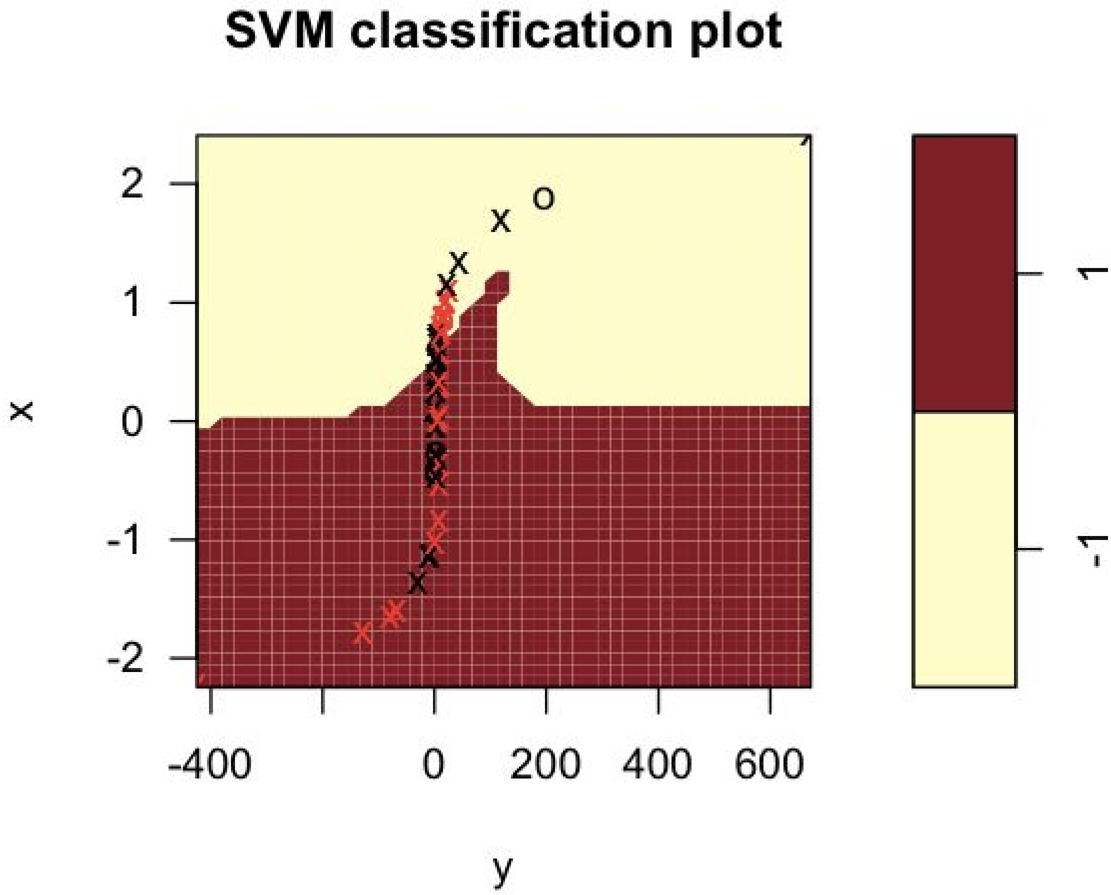
```
plot(svm.linear, data.test)
table(predict = predict(svm.linear, data.test), truth=data.test$z)
```




```
> table(predict = predict(svm.linear, data.test), truth=data.test$z)
       truth
predict -1  1
     -1  1  0
     1  26 23
```

The support vector classifier made 26 classification errors on the testing data.

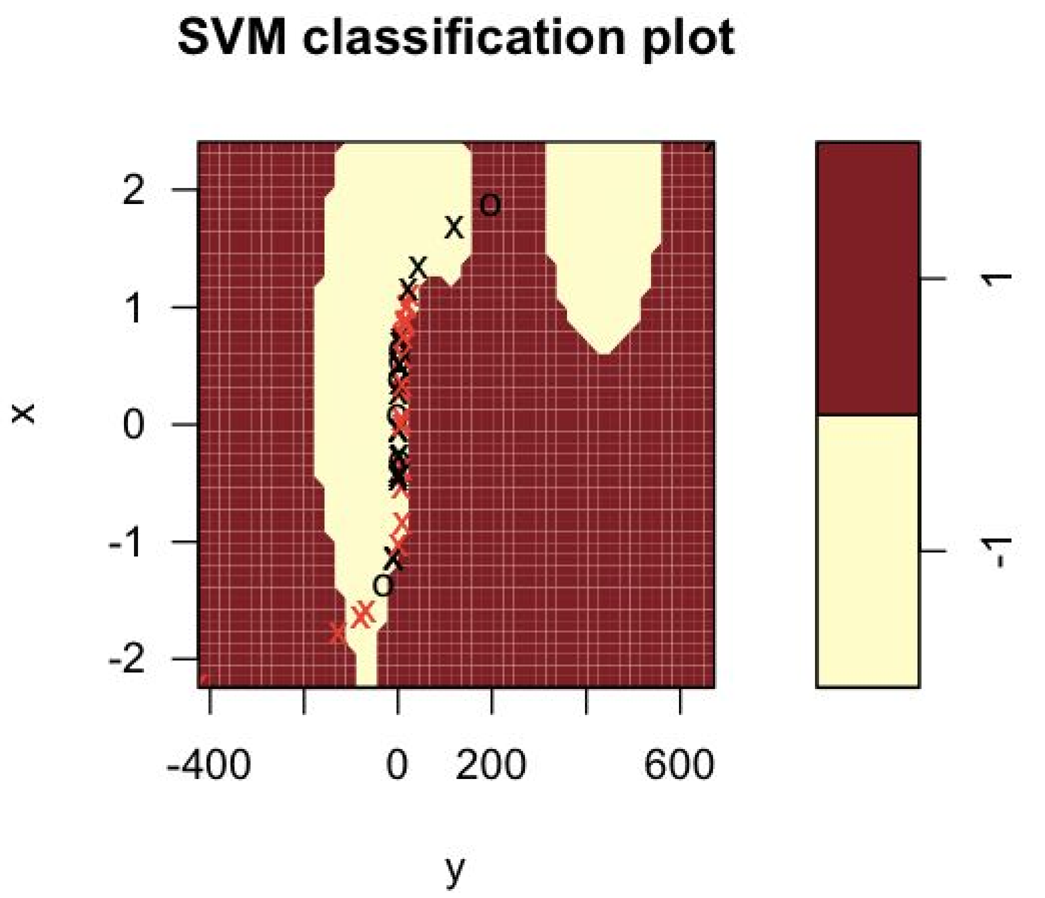
```
plot(svm.polynomial, data.test)
table(predict = predict(svm.polynomial, data.test), truth=data.test$z)
```




```
> table(predict = predict(svm.polynomial, data.test), truth=data.test$z)
       truth
predict -1  1
     -1  8 10
     1  19 13
```

The support vector machine with a polynomial kernel made 29 classification errors on the testing data.

```
plot(svm.radial, data.test)
table(predict = predict(svm.radial, data.test), truth=data.test$z)
```




```
> table(predict = predict(svm.radial, data.test), truth=data.test$z)
       truth
predict -1  1
     -1 15  8
     1  12 15
```

The support vector machine with a radial kernel made 20 classification errors on the testing data.

Overall, because the SVM radial kernel made the least amount of classification errors on the test data, it is the best model.

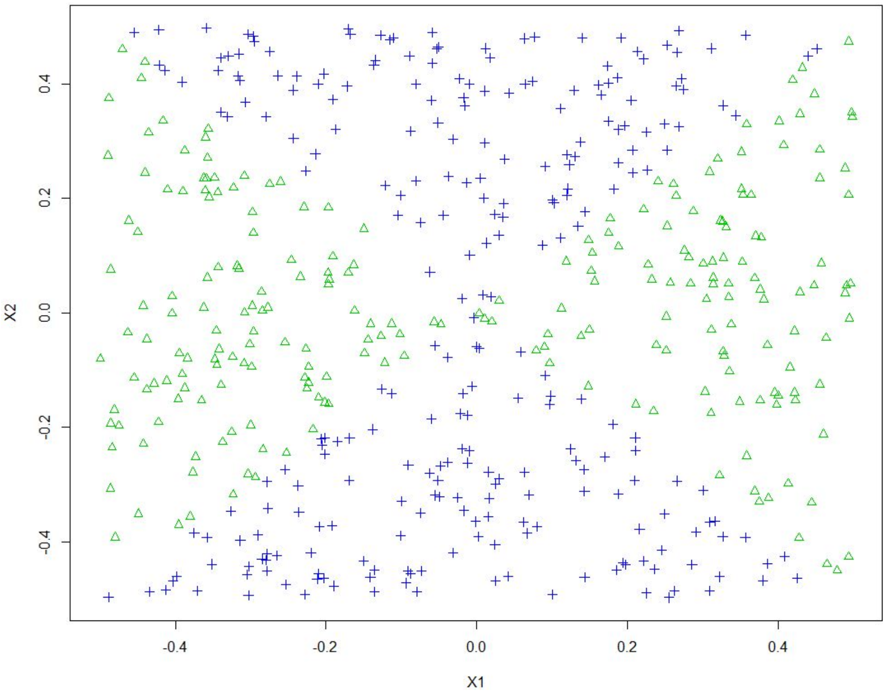
Question 5:

a. a data set with n = 500 and p = 2, such that the observations belong to two classes with a quadratic decision boundary between them was made using:

> x1=runif (500) -0.5
> x2=runif (500) -0.5
> y=1*((x1)^2-(x2)^2 > 0)

b.

```
> #answer b
> plot(x1,x2,xlab="X1",ylab="X2",col=(4-y),pch=(3-y))
```

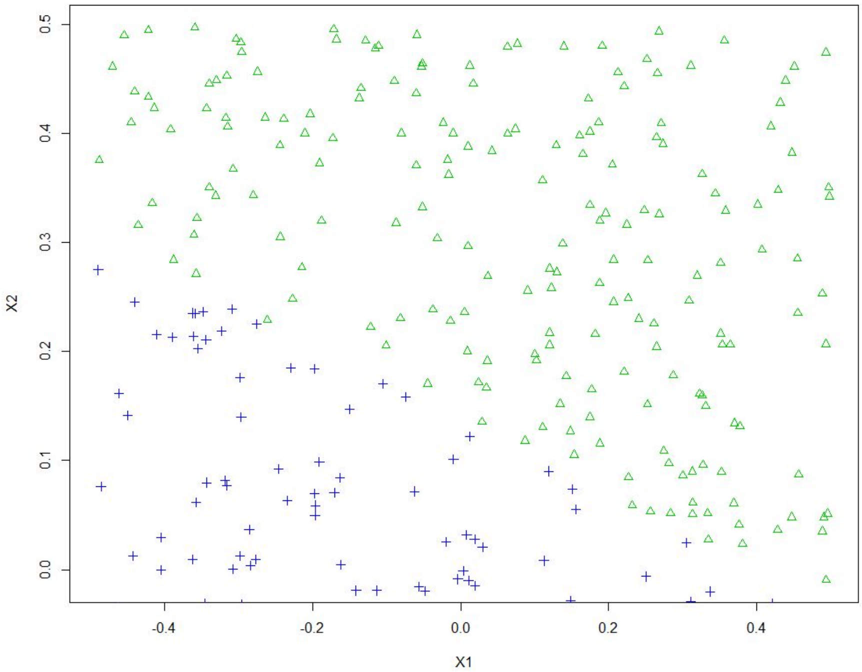


c.

```
> #answer c
> lr.fit = glm(y~x1+x2,family="binomial")
> summary(lr.fit)

Call:
glm(formula = y ~ x1 + x2, family = "binomial")

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-1.174   -1.126   -1.063    1.239    1.299

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.15180    0.08987  -1.689   0.0912 .
x1           0.07975    0.32300   0.247   0.8050
x2           0.23196    0.30306   0.765   0.4440
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 690.26  on 499  degrees of freedom
Residual deviance: 689.59  on 497  degrees of freedom
AIC: 695.59

Number of Fisher Scoring iterations: 3
```

d.

```
> #answer d
> data = data.frame(x1 = x1, x2 = x2, y = y)
> data.probs = predict(lr.fit, data, type = "response")
> data.preds = rep(0, 500)
> data.preds[data.probs > 0.47] = 1
> plot(data[data.preds == 1, ]$x1, data[data.preds == 1, ]$x2, col = (4 - 1), pch = (3 - 1), xlab = "X1",
 ylab = "X2")
> points(data[data.preds == 0, ]$x1, data[data.preds == 0, ]$x2, col = (4 - 0), pch = (3 - 0))
```

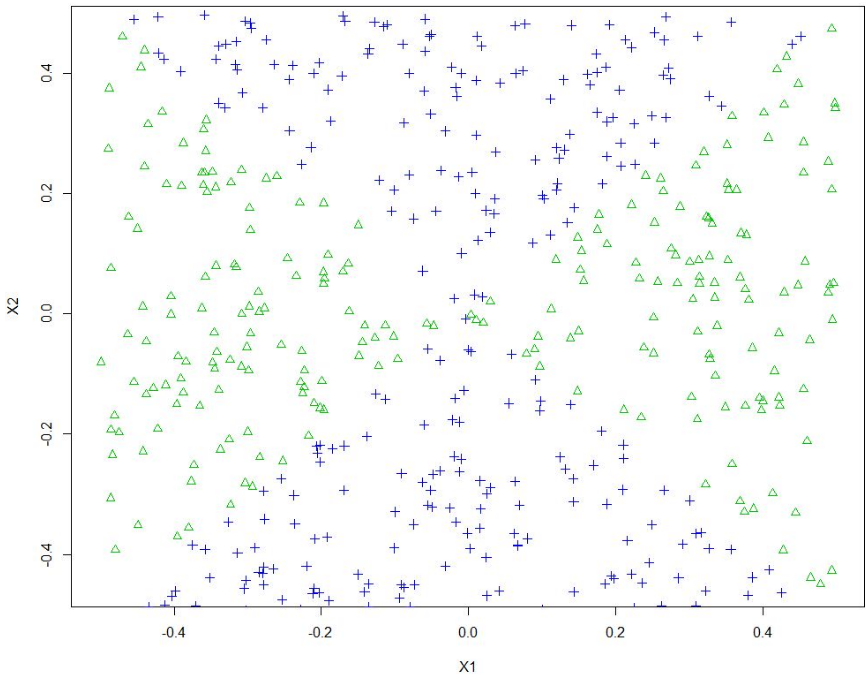


The decision boundary seems to be Linear.

e.

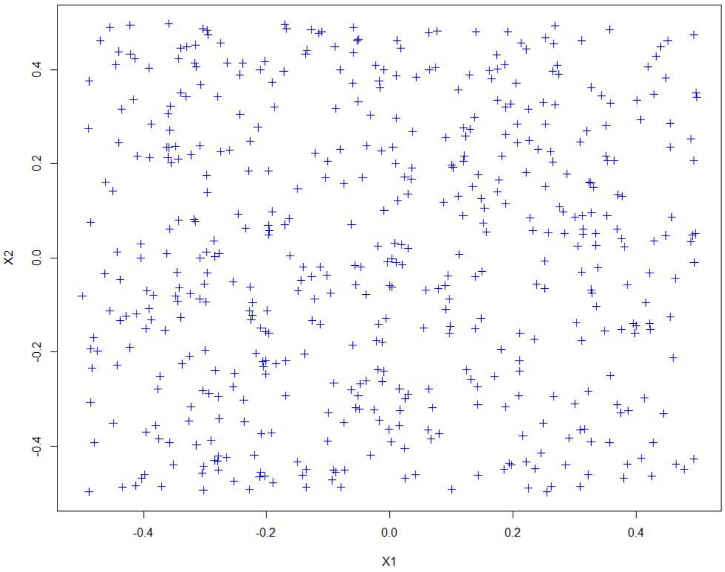
```
> #answer e
> lr.nl.fit = glm(y ~ poly(x1, 2) + poly(x2, 2) + I(x1 * x2), family = "binomial")
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(lr.nl.fit)

Call:
glm(formula = y ~ poly(x1, 2) + poly(x2, 2) + I(x1 * x2), family = "binomial")

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.01002   0.00000   0.00000   0.00000   0.01249

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      -1344.3    12358.9  -0.109    0.913
poly(x1, 2)1      -308.2   204045.7  -0.002    0.999
poly(x1, 2)2  237386.3  1452287.1   0.163    0.870
poly(x2, 2)1     2068.2   218704.4   0.009    0.992
poly(x2, 2)2 -262961.8  1604540.6  -0.164    0.870
I(x1 * x2)       -4992.2   114092.0  -0.044    0.965

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6.9026e+02  on 499  degrees of freedom
Residual deviance: 3.5346e-04  on 494  degrees of freedom
AIC: 12

Number of Fisher Scoring iterations: 25
```

f.



g.

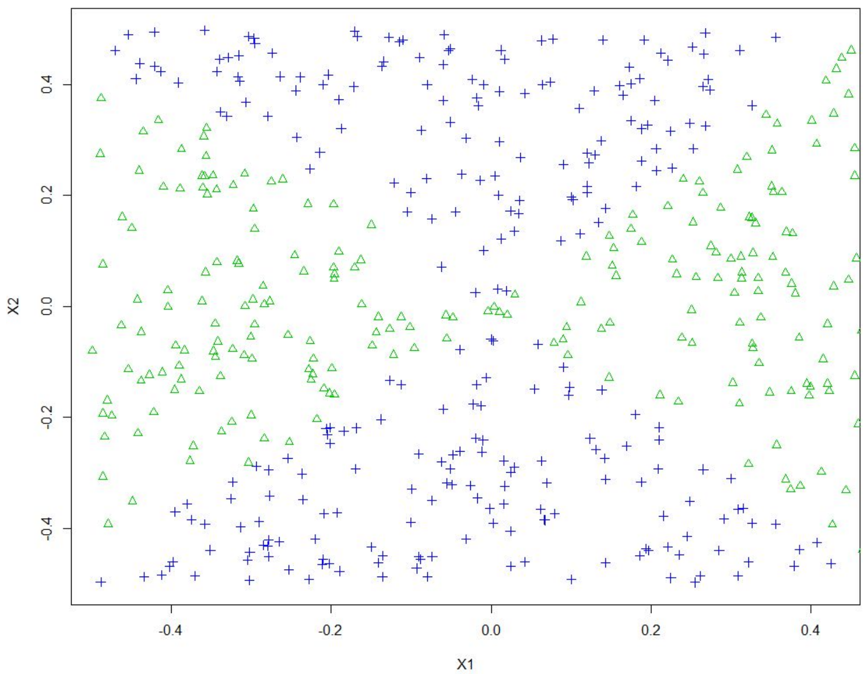
```
> #answer g
> library(e1071)
Warning message:
package ‘e1071’ was built under R version 3.6.3
> data$y = as.factor(data$y)
> svm.fit = svm(y~x1+x2, data, kernel = "linear", cost = 0.01)
> summary(svm.fit)

Call:
svm(formula = y ~ x1 + x2, data = data, kernel = "linear", cost = 0.01)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01

Number of Support Vectors:  465

 ( 231 234 )


Number of Classes:  2

Levels:
 0 1
```



SVM has classified all points under the same class.

h.

```
> #answer h
> data$y = as.factor(data$y)
> svm.nl.fit = svm(y~x1+x2, data, kernel = "radial", gamma = 1)
> summary(svm.nl.fit)

Call:
svm(formula = y ~ x1 + x2, data = data, kernel = "radial", gamma = 1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  154

 ( 75 79 )


Number of Classes:  2

Levels:
 0 1
```



i. We observe that SVM with linear kernel and logistic regression without any interaction, aren't able to find a distinguishable non-linear decision boundary. However, when we use SVM with non-linear kernel and logistic regression with interaction, they perform really well in distinguishing two classes by finding a non-linear decision boundary. Using non-linear kernel based SVM and logistic regression requires hyperparameter tuning for obtaining a decision boundary which is close to the ground truth.