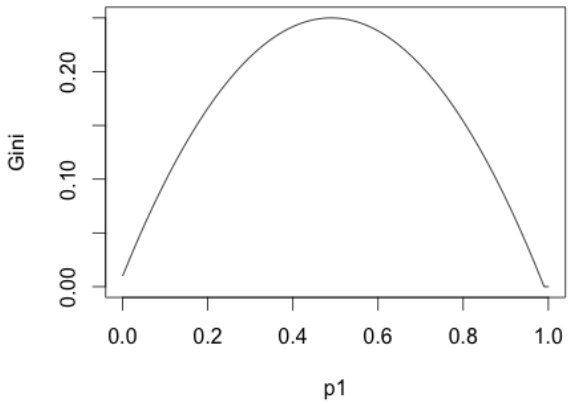
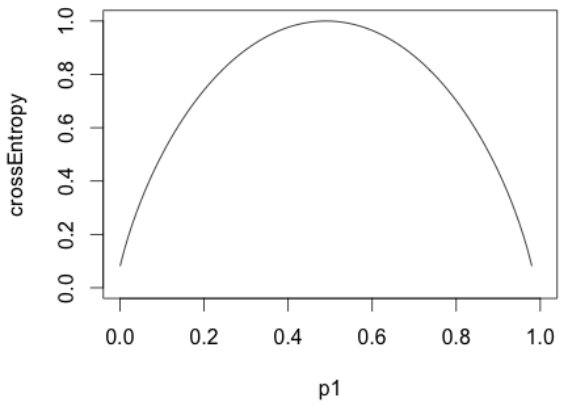
DS502 Statistical Method for Data Science
HW5

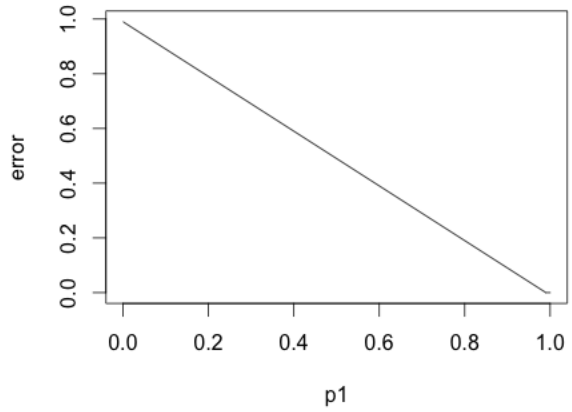
Congyuan Tang & Sachin Sudarshana

1. Section 8.4, Page 332, Question 3

Consider the Gini index, classification error, and cross-entropy in a simple classification setting with two classes. Create a single plot that displays each of the these quantities as a function of $\widehat{p}_{m1}$. The x-axis should display $\widehat{p}_{m1}$, ranging from 0 to 1, and the y-axis should display the value of the Gini index, classification error, and entropy.







2. Section 8.4, page 334, question 9

a. Splitting the data into training and testing

```
library(ISLR)
attach(OJ)
train=sample(1:nrow(OJ),800)
train_data=OJ[train,]
test_data=OJ[-train,]
```

b. Fitting a tree to the training data

```
library(tree)
tree_fit=tree(Purchase~.,data=train_data)
summary(tree_fit)
```

```
Classification tree:
tree(formula = Purchase ~ ., data = train_data)
Variables actually used in tree construction:
[1] "LoyalCH"       "PriceDiff"     "ListPriceDiff" "PctDiscMM"
Number of terminal nodes:  7
Residual mean deviance:  0.7609 = 603.4 / 793
Misclassification error rate: 0.165 = 132 / 800
```

The training error rate is **0.165**

c. Detailed text output

```
tree_fit
```

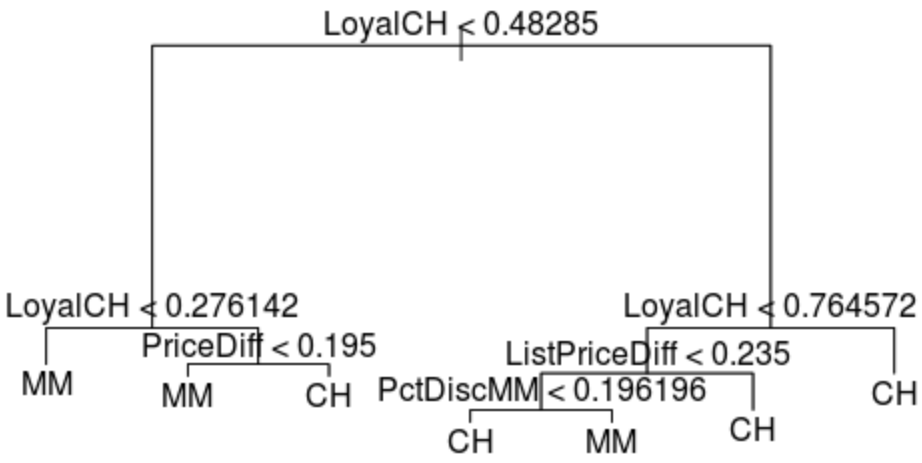
```
      node), split, n, deviance, yval, (yprob)
       * denotes terminal node

 1) root 800 1079.00 CH ( 0.59625 0.40375 )
   2) LoyalCH < 0.48285 309  323.10 MM ( 0.21683 0.78317 )
     4) LoyalCH < 0.276142 169   96.55 MM ( 0.08284 0.91716 ) *
     5) LoyalCH > 0.276142 140  185.70 MM ( 0.37857 0.62143 )
      10) PriceDiff < 0.195 64   67.24 MM ( 0.21875 0.78125 ) *
      11) PriceDiff > 0.195 76  105.30 CH ( 0.51316 0.48684 ) *
   3) LoyalCH > 0.48285 491  439.80 CH ( 0.83503 0.16497 )
     6) LoyalCH < 0.764572 234  280.30 CH ( 0.71368 0.28632 )
      12) ListPriceDiff < 0.235 92  127.40 MM ( 0.47826
0.52174)
        24) PctDiscMM < 0.196196 74  101.20 CH ( 0.56757 0.43243 ) *
```

```
        25) PctDiscMM > 0.196196 18   12.56 MM ( 0.11111 0.88889 ) *
      13) ListPriceDiff > 0.235 142  111.80 CH ( 0.86620 0.13380 ) *
     7) LoyalCH > 0.764572 257  108.70 CH ( 0.94553 0.05447 ) *
```

Let's consider the variable label '7'. The splitting variable is Loyalch, and the splitting value is 0.76. There are 257 points below in the subtree below this node. The deviance of all points in this region is 109. The prediction at this area is 'CH'. About 95% points in this node are 'CH', and 5% points in this node are 'MH'.
The star indicate that this is this is a terminal node.

d. Plotting the tree

```
plot(tree_fit)
text(tree_fit,pretty=1)
```



e. Confusion Matrix and Test error

```
predicted_tree=predict(tree_fit,test_data,type='class')
cm=table(test_data$Purchase,predicted_tree)
1-sum(diag(cm))/sum(cm)
```

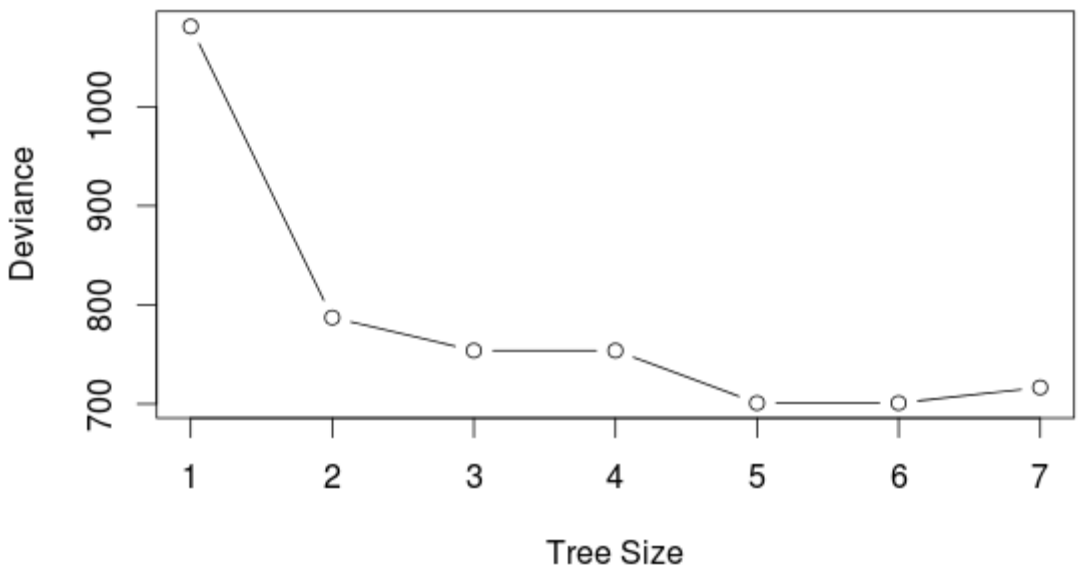
```
predicted_tree
     CH  MM
 CH 157  19
 MM  31  63

[1] 0.1851852
```

The test error was found out to be **0.185**

f. Optimal Tree

```
optimal_tree=cv.tree(tree_fit,FUN=prune.tree)
```

g. Plot



h. Lowest cross-validation classification error rate is in tree with **size 7**

i. Pruned tree

```
tree_pruned = prune.tree(tree_fit, best = 7)
```

j. Summary

```
Classification tree:
tree(formula = Purchase ~ ., data = train_data)
Variables actually used in tree construction:
[1] "LoyalCH"       "PriceDiff"     "ListPriceDiff" "PctDiscMM"
Number of terminal nodes:  7
Residual mean deviance:  0.7609 = 603.4 / 793
Misclassification error rate: 0.165 = 132 / 800
```

The misclassification error rate was found out to be **0.165 same as the unpruned tree**.

k. Test Error

```
predicted_pruned_tree = predict(tree_pruned,test_data,type='class')
table(predicted_pruned_tree,test_data$Purchase)
```

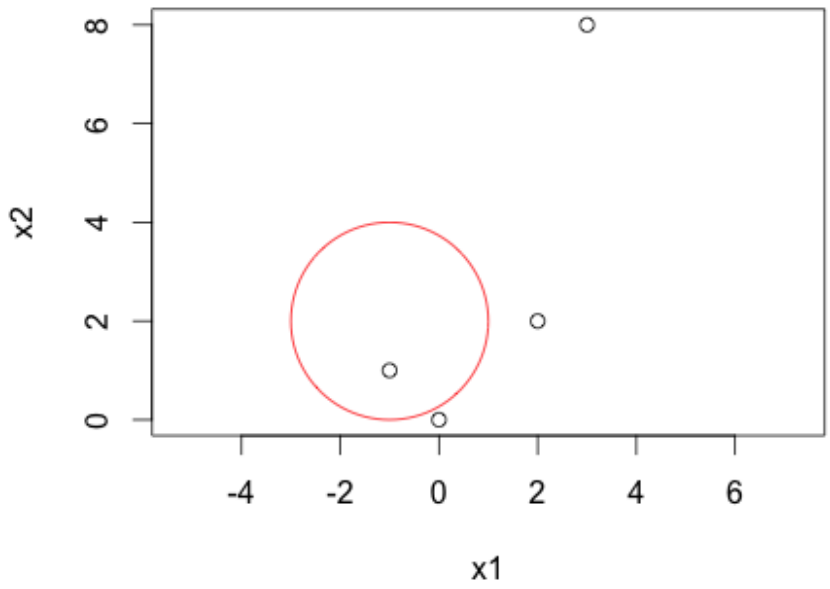
```
predicted_pruned_tree  CH  MM
                   CH 157  31
                   MM  19  63
```

The test error of pruned and unpruned tree is found out to be the same, **0.185**

3. Section 9.7, Page 368, question 2

We have seen that in p = 2 dimensions, a linear decision boundary takes the form $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$. We now investigate a non-linear decision boundary.

a. Sketch the curve
$$(1 + X_1)^2 + (2 - X_2)^2 = 4.$$
b. On your sketch, indicate the set of points for which
$$(1 + X_1)^2 + (2 - X_2)^2 > 4,$$
as well as the set of points for which
$$(1 + X_1)^2 + (2 - X_2)^2 \leq 4.$$
c. Suppose that a classifier assigns an observation to the blue class if
$$(1 + X_1)^2 + (2 - X_2)^2 > 4,$$
and to the red class otherwise. To what class is the observation (0, 0) classified? (-1, 1)? (2, 2)? (3, 8)?
d. Argue that while the decision boundary in (c) is not linear in terms of X_1 and X_2, it is linear in terms of X_1, X_1^2, X_2, and X_2^2.



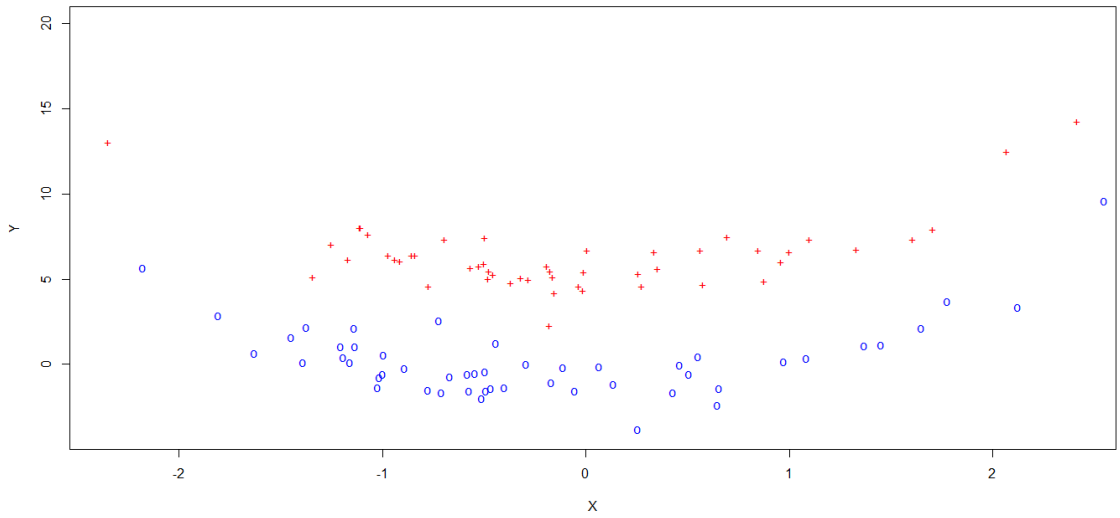
Inside and on the red circle, $(1 + X_1)^2 + (2 - X_2)^2 \leq 4$;
Outside the red circle, $(1 + X_1)^2 + (2 - X_2)^2 > 4$.

Clearly, we can see that only point (-1, 1) is in the circle, thus it's red; other points are blue.

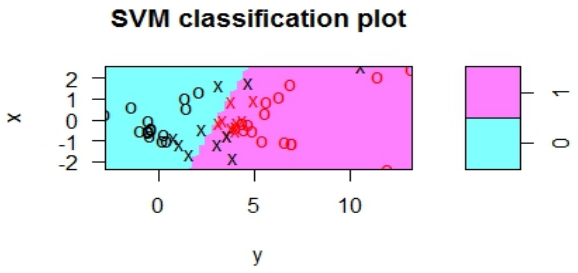
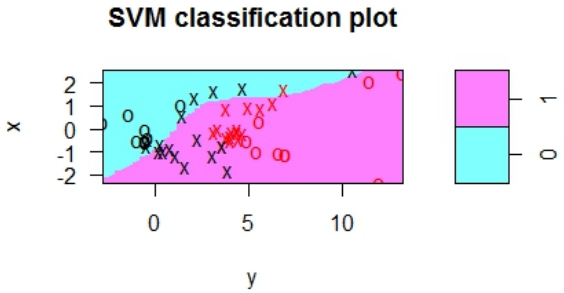
The coefficient of X_1, X_1^2, X_2, and X_2^2 are all linear, thus the decision boundary is linear in terms of X_1, X_1^2, X_2, and X_2^2.

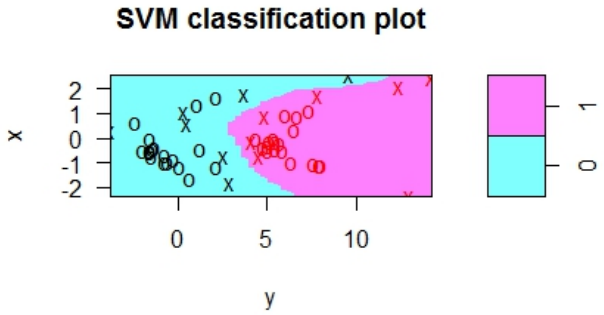
4. Section 9.7, Page 369, Question 4

The non-linear equation y=exp(x)+exp(-x) is used , and below is the graph between X and Y which shows nonlinear separation.



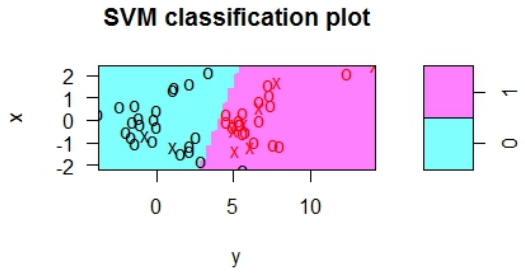
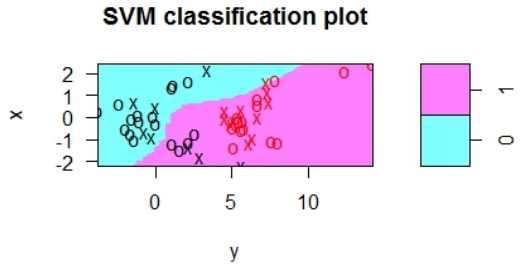
For the training dataset, the miscalculation error for support for support vector classifier, polynomial kernel and radial kernel were calculated and plotted.

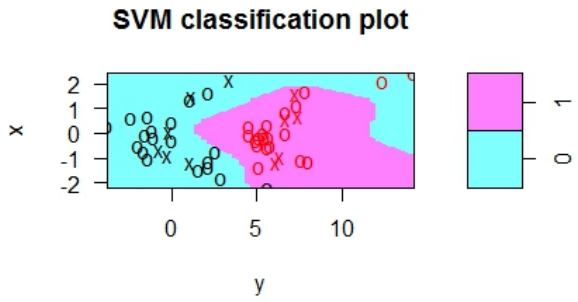
Support vector classifier: 0 1 0 23 2 1 0 25 This plot shows linear boundary, error = 4%	
polynomial kernel: 0 1 0 18 7 1 0 25 This plot shows non-linear boundary, and it is polynomial kernel of degree 3, misclassification error = 14%	

radial kernel 0 1 0 25 0 1 0 25 This plot shows non-linear boundary, and it is radial kernel of 1, misclassification error = 0%	

Among all the 3, radial kernel is the best classifier on training data, with zero misclassification error.

The same procedure was followed for testing data

Support vector classifier: 0 1 0 24 1 1 1 24 This plot shows linear boundary, error = 4%	
polynomial kernel: 0 1 0 17 8 1 0 25 This plot shows non-linear boundary, and it is polynomial kernel of degree 3, misclassification error = 16%	

radial kernel 0 1 0 25 0 1 1 24 This plot shows non-linear boundary, and it is radial kernel of 1, misclassification error = 2%	

The results were similar for testing data as well. The radial kernel method was the best followed by SVC and polynomial kernel.

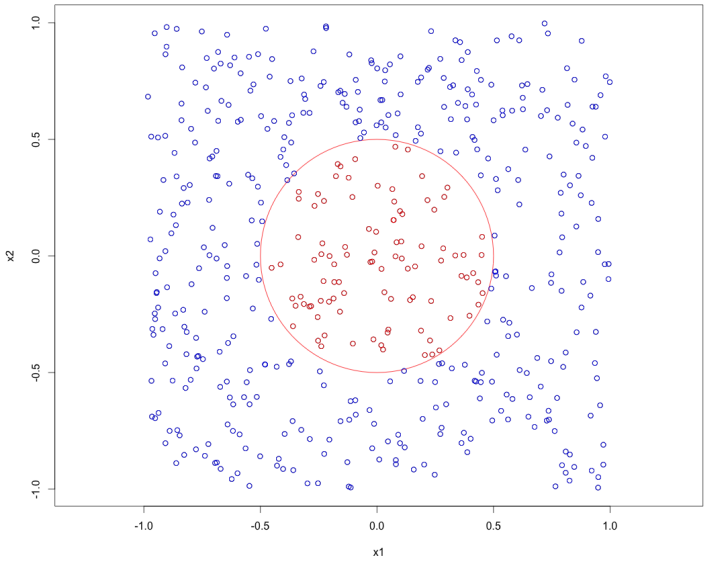
5. Section 9.7, Page 369-370, question 5

We have seen that we can fit an SVM with a non-linear kernel in order to perform classification using a non-linear decision boundary. We will now see that we can also obtain a non-linear decision boundary by performing logistic regression using non-linear transformations of the features.

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

```
> x1 = runif(500) * 2 - 1
> x2 = runif(500) * 2 - 1
> draw.circle(0,0,0.5,border = "red")
```

b. Plot the observations, colored according to their class labels. Your plot should display X1 on the X-axis, and X2 on the Y-axis.

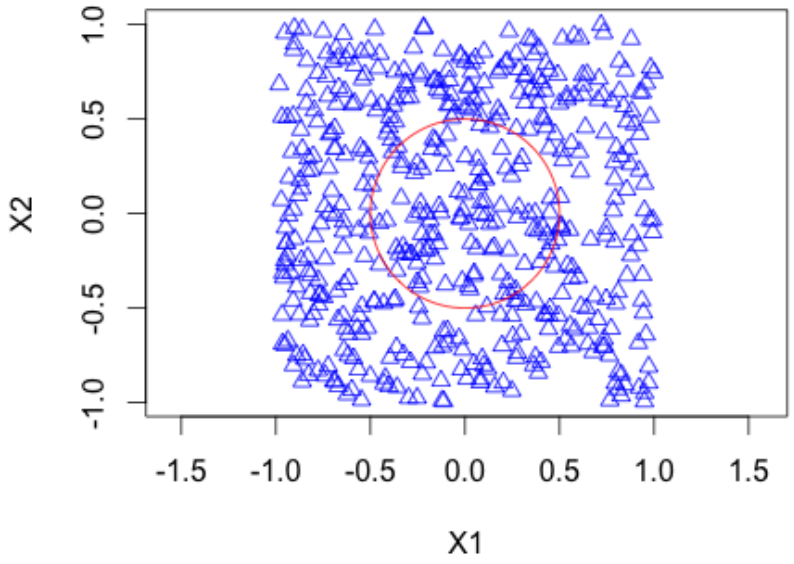


c. Fit a logistic regression model to the data, using X1 and X2 as predictors.

```
> logit.fit = glm(y ~ x1 + x2, family = "binomial")
> summary(logit.fit)
Call:
glm(formula = y ~ x1 + x2, family = "binomial")
Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.8531  0.5968  0.6359  0.6787  0.7552

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.4214     0.1133  12.543   <2e-16 ***
x1          -0.1518     0.1979  -0.767    0.443
x2           0.1698     0.1985   0.855    0.392
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 491.97  on 499  degrees of freedom
Residual deviance: 490.65  on 497  degrees of freedom
AIC: 496.65

Number of Fisher Scoring iterations: 4
```

According to the output, only “intercept” is statistically significant.

d. Apply this model to training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should be linear.



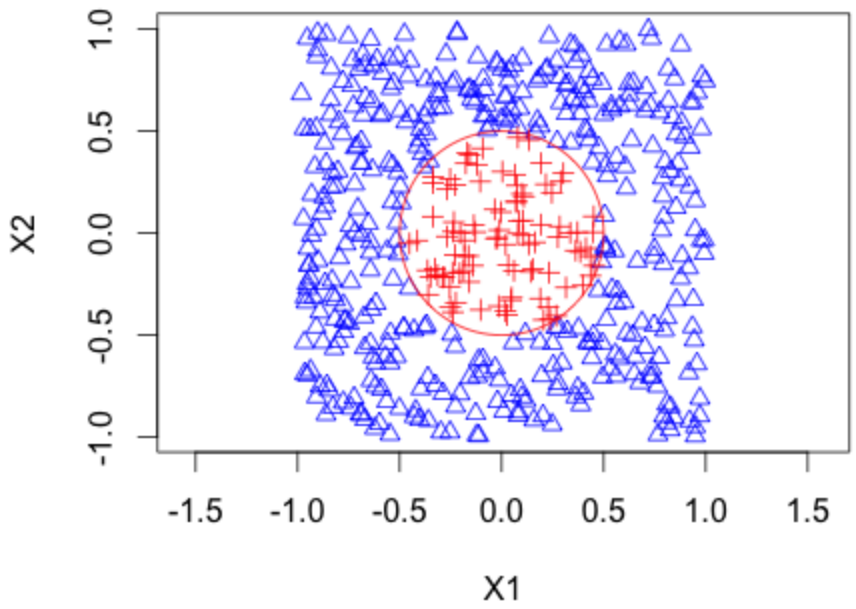
Apparently, the model predicted the output will be all “blue”s. The decision boundary in this case could be viewed as linear.

e. Now fit a logistic regression model to the data using non-linear functions of X1 and X2 as predictors.

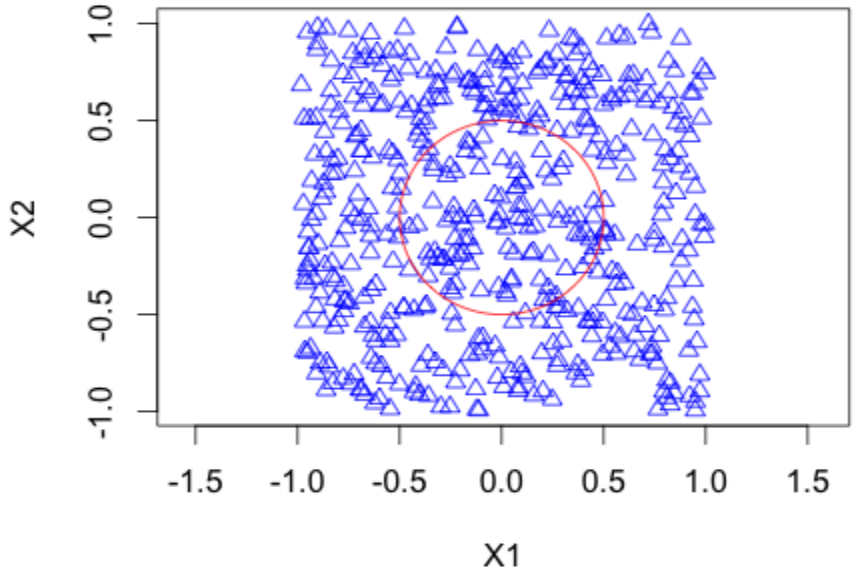
```
> logitnl.fit <- glm(y ~ poly(x1, 2) + poly(x2, 2) + I(x1 * x2), family = "binomial")

> summary(logitnl.fit)

Call:

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

Deviance Residuals:

      Min         1Q      Median         3Q        Max

-5.066e-04   2.000e-08   2.000e-08   2.000e-08   5.577e-04

Coefficients:

             Estimate Std. Error z value Pr(>|z|)

(Intercept)     556.93   30165.22   0.018    0.985

poly(x1, 2)1   -720.36   85975.38  -0.008    0.993

poly(x1, 2)2   9066.91  497797.14   0.018    0.985

poly(x2, 2)1     39.58   63261.72   0.001    1.000

poly(x2, 2)2   8525.88  461840.72   0.018    0.985

I(x1 * x2)       35.64   19761.86   0.002    0.999


(Dispersion parameter for binomial family taken to be 1)


Null deviance: 4.9197e+02  on 499  degrees of freedom

Residual deviance: 9.4790e-07  on 494  degrees of freedom

AIC: 12


Number of Fisher Scoring iterations: 25
```

According to the output, none of the variables is statistically significant.

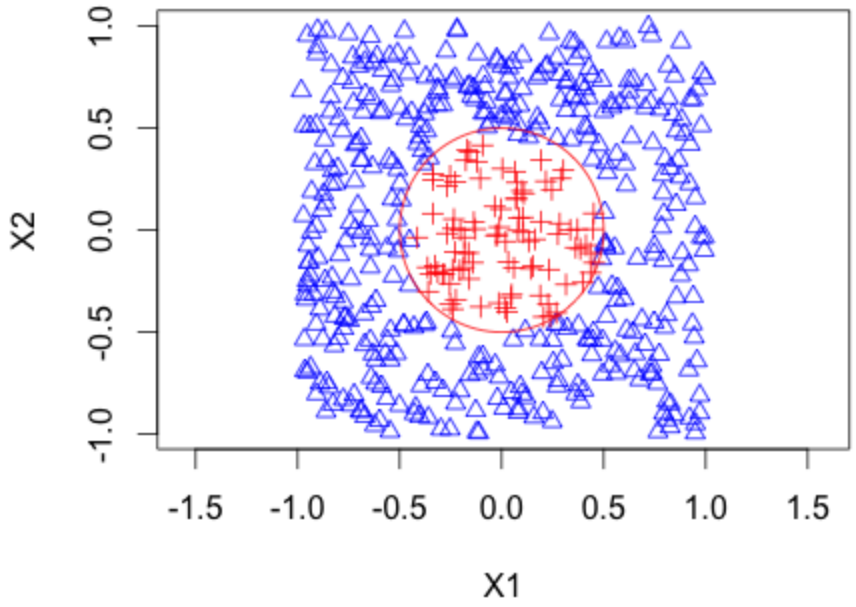
f. Apply this model to training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should obviously be non-linear.



g. Fit a support vector classifier to the data with X_1 and X_2 as predictors. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.

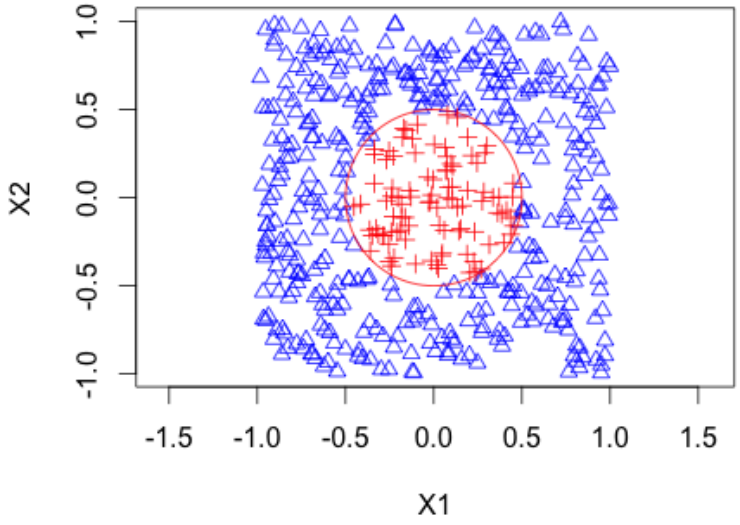


h. Fit a SVM using a non-linear kernel to the data with X_1 and X_2 as predictors. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.



i. Comments

From the above results and graphs we can clearly see that non-linear kernel SVM is just as powerful as non-linear logistic regression. They both have significantly better performance classifying non-linear boundary binary classification problems. The linear kernel in SVC and logistic regression performs badly in this classification problems. Interestingly, in SVM, with the increase of gamma parameter, the accuracy is increasing. The graph in (h) part is under gamma = 1, but when gamma = 10000, the result is like below:



The performance is actually better (than gamma = 1) !

6. Section 10.7, Page 414-415, Question 4

a. Information is insufficient to make a conclusion in this case. The minimal intercluster dissimilarity might be the same as the maximal intercluster dissimilarity or vice versa. If they are the same, then they share the same height.
b. They would fuse at the same height as they are single leaf observations, kind of linkage doesn't affect this situation, since the min or max of their distance would be the same when doing the single or complete linkage fusing on these single observation clusters.

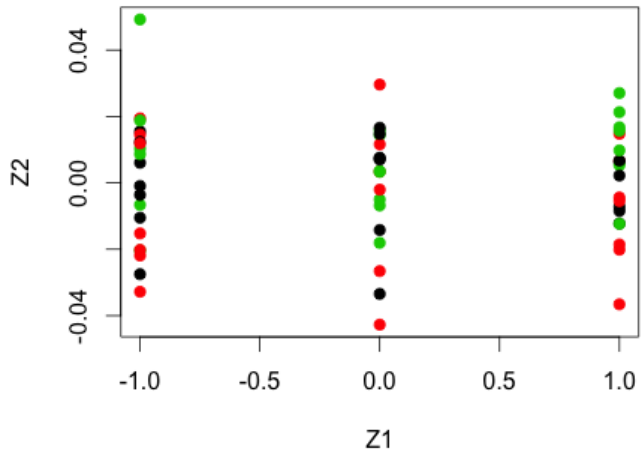
7. Section 10.7, Page 417, question 10

In this problem, you will generate simulated data, and then perform PCA and K-means clustering on the data.

a. Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables.

```
set.seed(1992)
data = matrix(rep(0, 20*3*50), ncol = 50)
data = data.frame(data)
for (i in 1:50) {
  data[1:20, i] = rnorm(20, mean = 1, sd = 1)
  data[21:40, i] = rnorm(20, mean = 2, sd = 2)
  data[41:60, i] = rnorm(20, mean = 3, sd = 3)
}
label = c(rep(1,20), rep(2,20), rep(3,20))
data = data.frame(data, label)
```

b. Perform PCA on the 60 observations and plot the first two principal component score vectors. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, then continue on to part (c). If not, the return to part (a) and modify the simulation so that there is greater separation between the three classes. Do not continue to part (c) until the three classes show at least some separation in the first two principal component score vectors.



The clusters are still well separated in Z1 dimension.

c. Perform K-means clustering of the observations with K=3. How well do the clusters that you obtained in K-means clustering compare to the true class labels?

> km.out <- kmeans(x, 3, nstart = 20)

> table(label, km.out$cluster)

label	1	2	3
1	20	0	0
2	0	0	20
3	0	20	0

The classification result here is perfectly accurate.

d. Perform K-means clustering with K=2. Describe your results.

> km.out <- kmeans(data, 2, nstart = 20)

> table(label, km.out$cluster)

label 1 2

1 0 20

2 0 20

3 20 0

The result here is pretty weird, but seems accurate.

e. Now perform K-means clustering with K=4, and describe your results.

> km.out <- kmeans(data, 4, nstart = 20)

> table(label, km.out$cluster)

label 1 2 3 4

1 0 20 0 0

2 0 0 0 20

3 13 0 7 0

The third cluster is splitted into two clusters.

f. Now perform K-means clustering with K=3 on the first two principal component score vectors, rather than on the raw data. That is, perform K-means clustering on the 60x2 matrix of which the first column is the first principal component score vector, and the second column is the second principal component score vector. Comment on the results.

> km.out <- kmeans(pr.out$x[, 1:2], 3, nstart = 20)

> table(label, km.out$cluster)

label 1 2 3

1 20 0 0

```
2  0 20  0
3  0  0 20
```

The result is perfectly accurate again.

g. Using the “scale()” function, perform K-means clustering with K=3 on the data after scaling each variable to have standard deviation one. How do these results compare to those obtained in (b) ? Explain.

```
> km.out <- kmeans(scale(data), 3, nstart = 20)
> table(label, km.out$cluster)

label  1  2  3
   1  5  6  9
   2  9  6  5
   3  4  6 10
```

The result is worse. This is probably because the scale action degenerate the information contained in our previously faked data, and the model performs worse.