MSSC 6250 Machine Learning Homework 4

Support Vector Machines, Tree Methods, Unsupervised Learning

Henri Medeiros Dos Reis

- Deadline: **Friday, April 28 11:59 PM**
- Homework presentation date: **Tuesday, May 2**
- Please submit your work in **one PDF** file to **D2L** > **Assessments** > **Dropbox**. *Multiple files or a file that is not in pdf format are not allowed.*
- Any relevant code should be attached.
- Read **ISL** Chapter 8, 9, and 12.

Exercises required for all students

1. **ISL** Sec. 8.4: 12 (Don't do BART)

Solution:

The chosen data set is Boston, since it is of easy access, and we have worked on it previously.

```
set.seed(1)
library(ISLR2)
```

```
Warning: package 'ISLR2' was built under R version 4.2.2
```

```
summary(Boston)
```

```
      crim                zn             indus            chas        
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
 1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
 Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
 Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
```

```
 Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
      nox               rm             age              dis        
 Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
 1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
 Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
 Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
 3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
 Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
      rad              tax           ptratio          lstat      
 Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   : 1.73  
 1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.: 6.95  
 Median : 5.000   Median :330.0   Median :19.05   Median :11.36  
 Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :12.65  
 3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:16.95  
 Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :37.97  
      medv      
 Min.   : 5.00  
 1st Qu.:17.02  
 Median :21.20  
 Mean   :22.53  
 3rd Qu.:25.00  
 Max.   :50.00  
```

Let's try to predict - the per capita crime rate by town. And use MSE as the metric to measure our performance.

```r
train <- sample(nrow(Boston), 0.8 * nrow(Boston))
test <- -train
```

Linear regression:

```r
lm.fit <- lm(crim ~ zn + nox + dis + rad + ptratio + medv,
             data = Boston[train,])

lm.prediction <- predict(lm.fit, newdata = Boston[test,])
(mse <- mean((Boston[test,1]-lm.prediction)2))
```

```
[1] 64.77735
```

Linear regression gave a MSE of 64.77735.

Boosting:

```r
library(gbm)
```

```
Warning: package ’gbm’ was built under R version 4.2.3
```

```
Loaded gbm 2.1.8.1
```

```
gbm.fit <- gbm::gbm(crim ~ zn + nox + dis + rad + ptratio + medv,
                    data = Boston[train,],
                    distribution = "gaussian", n.trees = 300,
                    shrinkage = 0.5, bag.fraction = 0.8,
                    cv.folds = 10)

gbm.prediction <- predict(gbm.fit, newdata = Boston[test,])
```

```
Using 18 trees...
```

```
(mse <- mean((Boston[test,1]-gbm.prediction)2))
```

```
[1] 53.76935
```

Boosting gave a MSE around 54. Which was better than the linear regression model.

Bagging:

```
library(randomForest)
```

```
Warning: package ’randomForest’ was built under R version 4.2.3
```

```
randomForest 4.7-1.1
```

```
Type rfNews() to see new features/changes/bug fixes.
```

```
bag.fit <- randomForest(crim ~ zn + nox + dis + rad + ptratio + medv,
                         data = Boston[train,],
                       mtry = 5)
bag.prediction <- predict(bag.fit, newdata = Boston[test,])
(mse <- mean((Boston[test,1]-bag.prediction)2))
```

```
[1] 53.84921
```

Bagging gave a MSE around 52. Which was better than both previous models.

Random forests:

```
rngF.fit <- randomForest(crim ~ zn + nox + dis + rad + ptratio + medv,
                         data = Boston[train,],
                        mtry = 2)
rngF.prediction <- predict(rngF.fit, newdata = Boston[test,])
(mse <- mean((Boston[test,1]-rngF.prediction)2))
```

```
[1] 52.41303
```

Random forests also gave a MSE around 52.

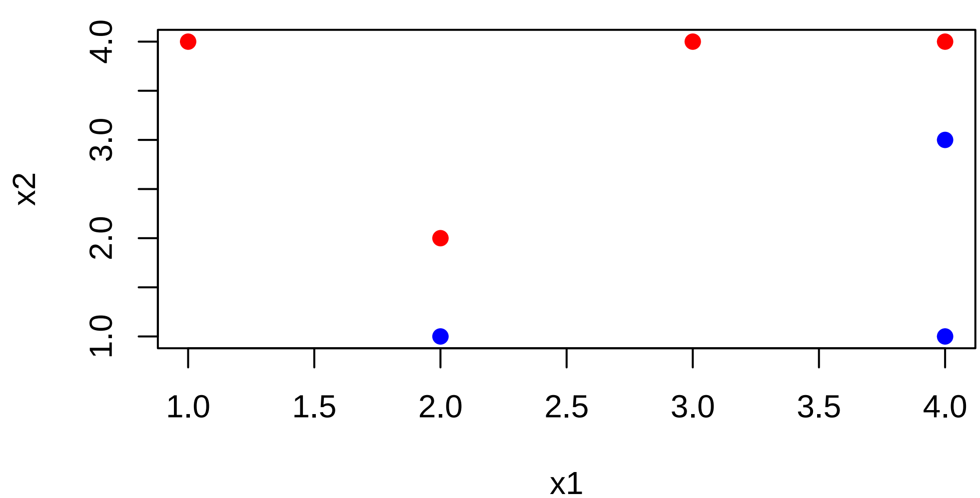
Therefore, both Random forests and Bagging were the models that resulted in the best performance in this data set.

2. **ISL** Sec. 9.7: 3

Solution:

a-)

```
x1 = c(3,2,4,1,2,4,4)
x2 = c(4,2,4,4,1,3,1)
colors = c("red", "red", "red", "red", "blue", "blue", "blue")
plot(x1,x2,col=colors,pch=19)
```



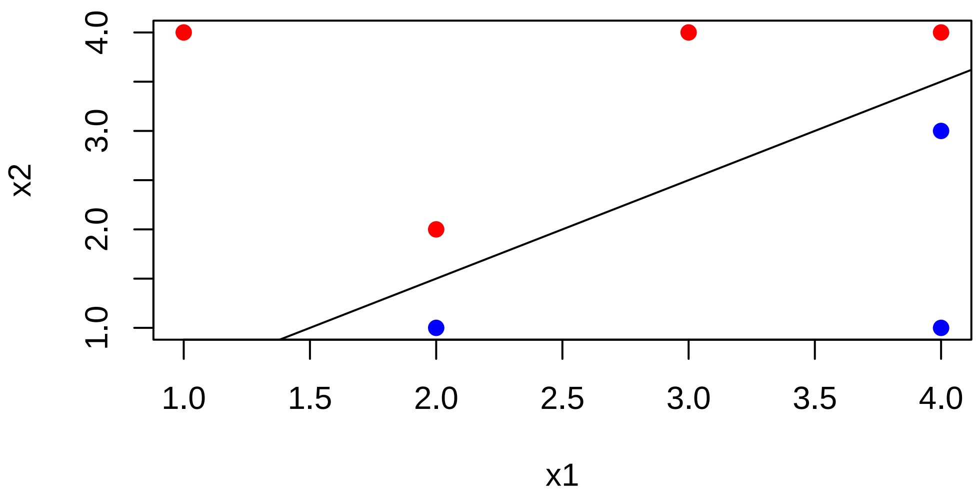
b-) Since we are using margin my_classifier. Then we need to look at observations #2, #3 and #5, #6.Since they are the closest to the boundry.

$$(2,2),(4,4)(2,1),(4,3)$$

Then just use the equation of the line and rewrite it.

$$=> (2,1.5),(4,3.5)b = (3.5-1.5)/(4-2) = 1a = X_2 - X_1 = 1.5 - 2 = -0.5$$

```
plot(x1,x2,col=colors,pch=19)
abline(-0.5, 1)
```



c-)

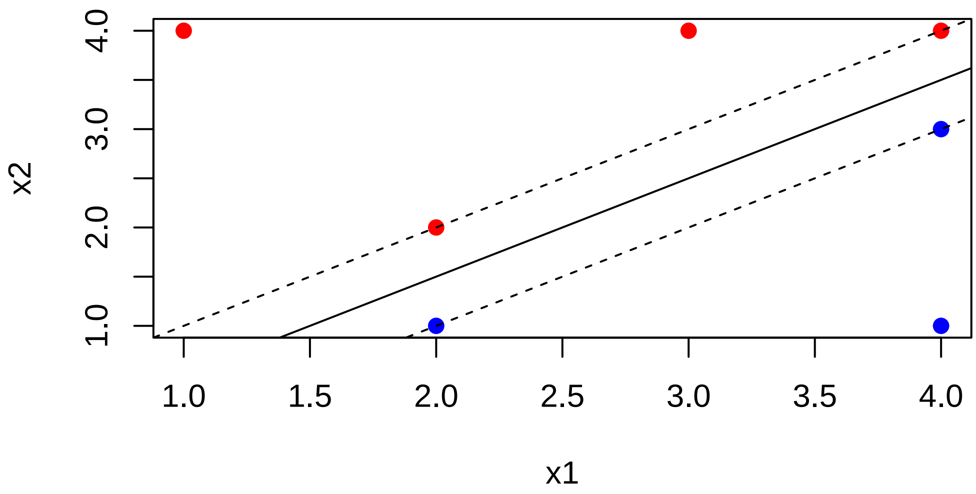
The values for βs: $\beta_0 = 0.5$, $\beta_1 = 1$, and $\beta_0 = -1$

So the rules are:

$0.5 - X_1 + X_2 > 0 \Rightarrow$ Blue and $0.5 - X_1 + X_2 \leq 0 \Rightarrow$ Red

d-)

```
plot(x1,x2,col=colors,pch=19)
abline(-0.5, 1)
abline(-1, 1, lty=2)
abline(0, 1, lty=2)
```



e-) There are four support vector for the maximal margin my_classifier.

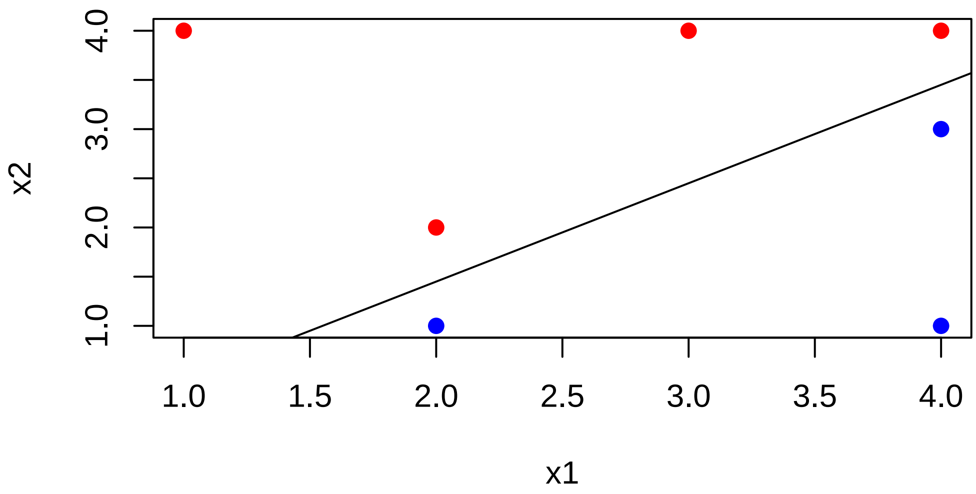
The observations 2,3,5 and 6.

f-)

Since the seventh observation is not a support vector, any small change to that observation would not affect the hyperplane.

g-)

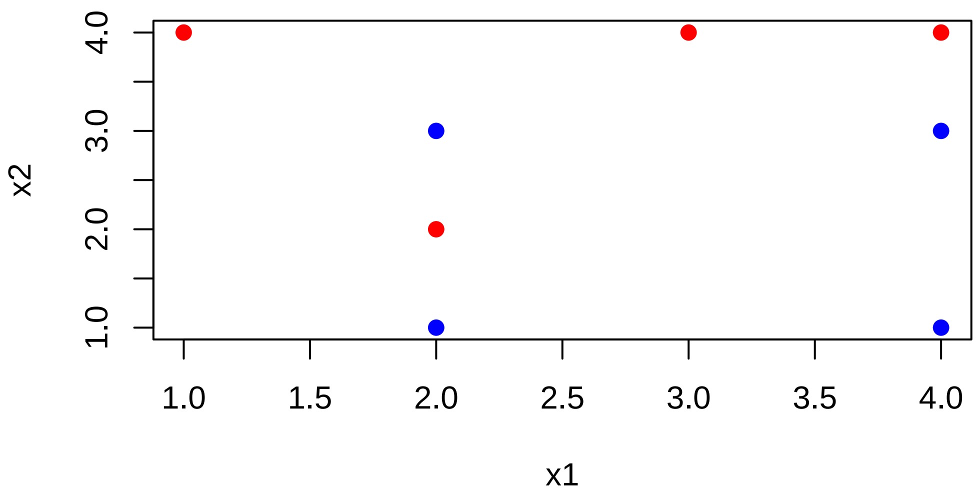
```
plot(x1,x2,col=colors,pch=19)
abline(-0.55, 1)
```



$-0.55 - X_1 + X_2 > 0$

h-)

```
plot(x1,x2,col=colors,pch=19)
points(c(2), c(3), col=c("blue"),pch=19)
```



3. **ISL** Sec. 9.7: 5

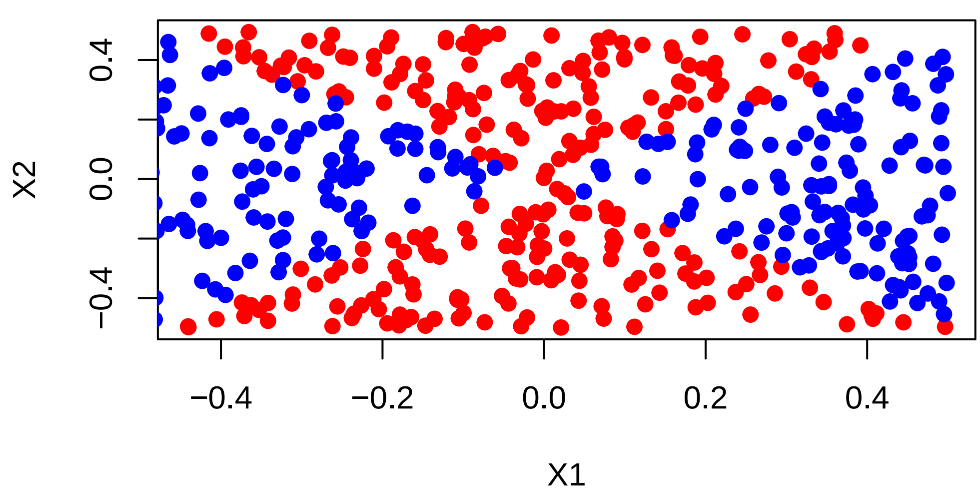
Solution:

a-)

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

b-)

```
plot(x1[y==0], x2[y==0], col="red", xlab="X1", ylab="X2", pch=19)
points(x1[y==1], x2[y==1], col="blue", pch=19)
```



c-)

```
glm.fit=glm(y~. ,family='binomial', data=data.frame(x1,x2,y))
glm.fit
```

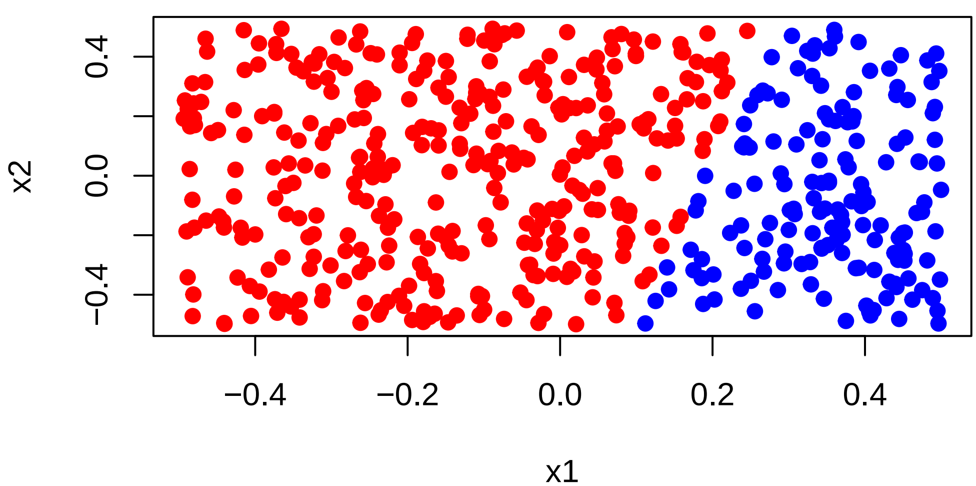
```
Call:  glm(formula = y ~ ., family = "binomial", data = data.frame(x1,
    x2, y))

Coefficients:
(Intercept)           x1           x2
    -0.1487       0.7817      -0.1381
```

```
Degrees of Freedom: 499 Total (i.e. Null);  497 Residual
Null Deviance:      690.8
Residual Deviance: 684.4    AIC: 690.4
```

d-)

```
glm.pred=predict(glm.fit,data.frame(x1,x2))
plot(x1,x2,col=ifelse(glm.pred>0,'blue','red'),pch=19)
```



As expected, by comparing the graphs we can see this model does not do a good job.

e-) Let's try a polynomial of degree 2.

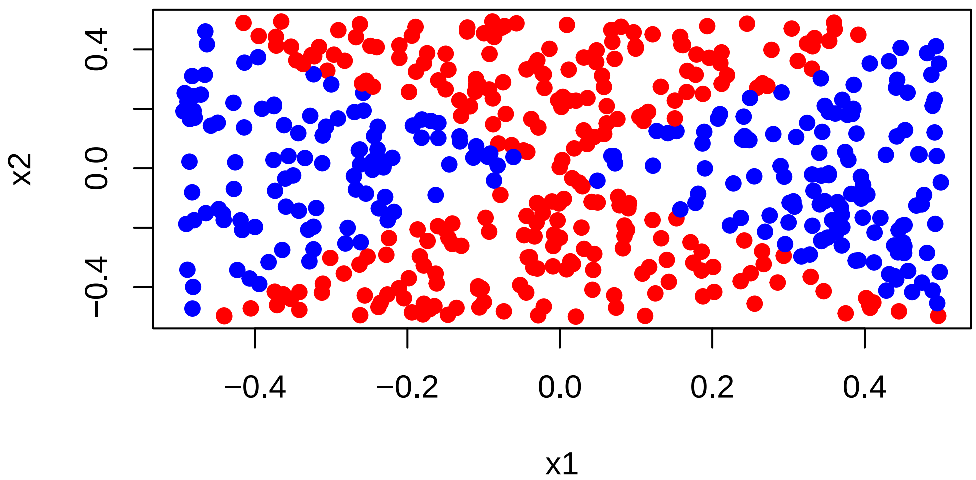
```
glm.fit2=glm(y~poly(x1,2)+poly(x2,2) ,family='binomial', data=data.frame(x1,x2,y))
```

```
Warning: glm.fit: algorithm did not converge
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

f-)

```
glm.pred=predict(glm.fit2,data.frame(x1,x2))
plot(x1,x2,col=ifelse(glm.pred>0,'blue','red'),pch=19)
```

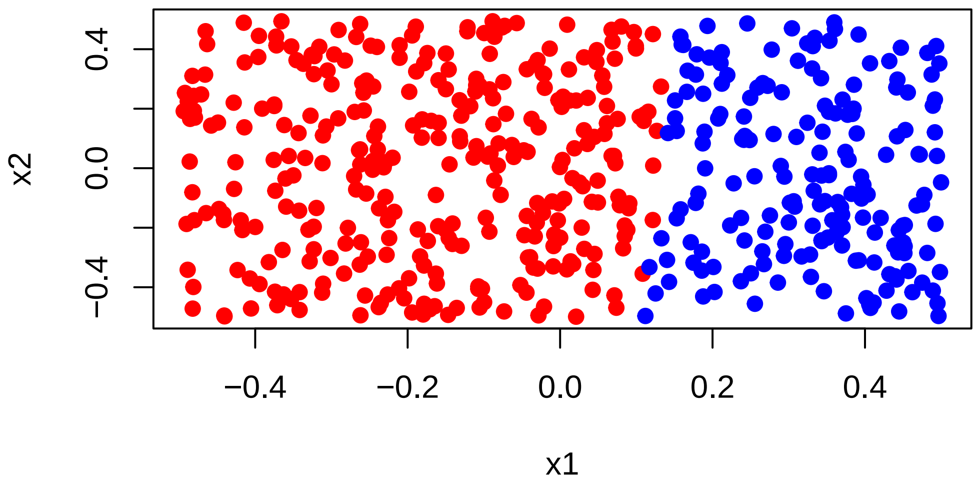


g-)

```r
library(e1071)
```

```
Warning: package 'e1071' was built under R version 4.2.3
```

```r
svm.fit=svm(y~.,data=data.frame(x1,x2,y=as.factor(y)),kernel='linear')
svm.pred=predict(svm.fit,data.frame(x1,x2),type='response')
plot(x1,x2,col=ifelse(svm.pred!=0,'blue','red'),pch=19)
```

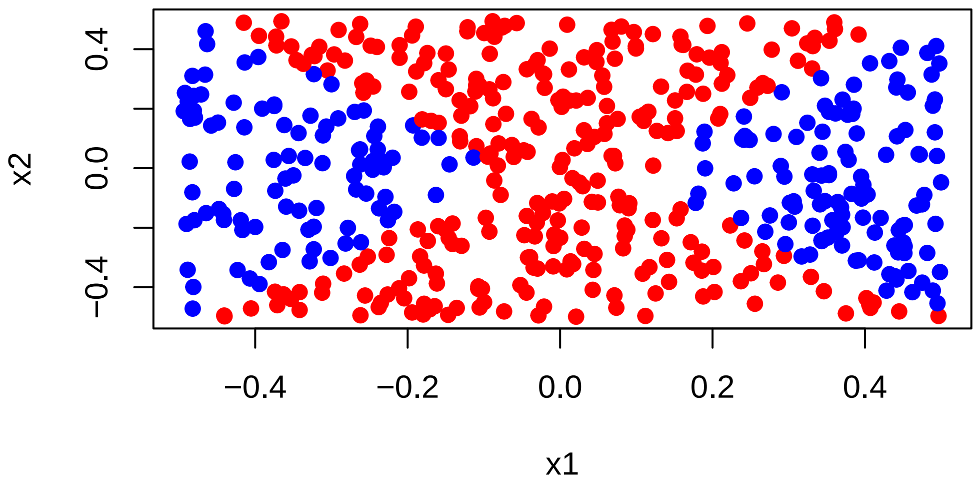


As expected, by comparing the graphs we can see this model does not do a good job, since the relation is not linear.

h-)

Let's try a polynomial of degree 2.

```
svm.fit=svm(y~.,data=data.frame(x1,x2,y=as.factor(y))
            ,kernel='polynomial',degree=2)
svm.pred=predict(svm.fit,data.frame(x1,x2),type='response')
plot(x1,x2,col=ifelse(svm.pred!=0,'blue','red'),pch=19)
```



As expected, by comparing the graphs we can see this model does not do a good job, since the relation is not linear. And we know the relationship is quadratic.

i-)

This experiment demonstrates the effectiveness of SVMs with non-linear kernels for locating non-linear boundaries. SVMs using linear kernels and logistic regression with no interactions both fall short in locating the decision boundary. Logistic regression appears to have the same power as radial-basis kernels when interaction factors are included. However, choosing the proper interaction terms requires some manual work and fine adjustment.

4. **ISL** Sec. 12.6: 10

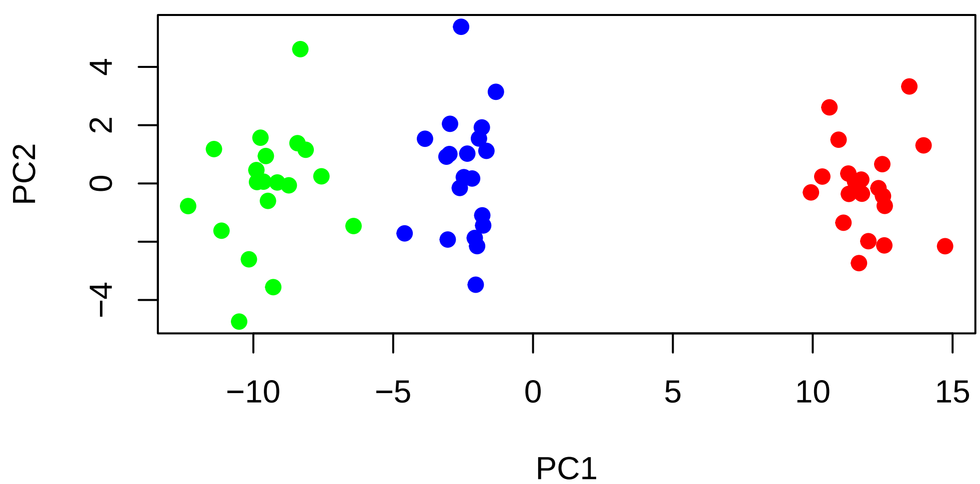
Solution:

a-)

```
data = matrix(c(rnorm(20 * 50, mean = 4),
            rnorm(20 * 50, mean = 2),
            rnorm(20 * 50, mean = 5)), ncol = 50, byrow = TRUE)
my_class <- rep(0,60)
my_class[1:20] <- "blue"
my_class[21:40] <- "red"
my_class[41:60] <- "green"
```

b-)

```r
pcs = prcomp(data)
plot(pcs$x[,1:2],col=my_class,pch=19)
```



c-)

```r
kmeans.result = kmeans(data, centers=3, nstart = 100)
table(my_class, kmeans.result$cluster)
```

```
my_class  1  2  3
   blue   0  0 20
   green  0 20  0
   red   20  0  0
```

As we can see all observations are correctly my_classified. This is expected as the observations in the three my_classes are well separated, and we happen to know that there are 3 my_classes.

d-)

```r
kmeans.result = kmeans(data, centers=2, nstart = 100)
table(my_class, kmeans.result$cluster)
```

```
my_class  1  2
   blue  20  0
   green 20  0
   red    0 20
```

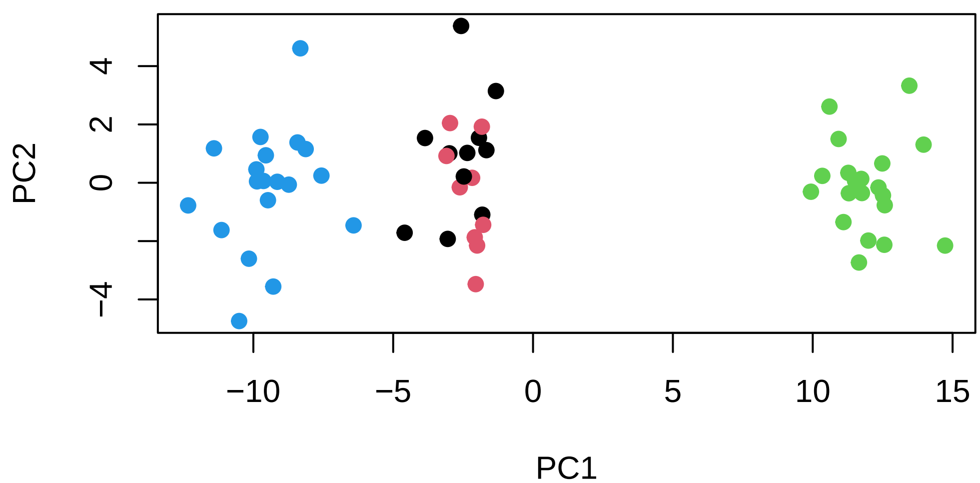
From looking at the plot of the my_classes, it makes sense that all the green and blue are my_classified together, since the distance from the red is much larger.

e-)

```
kmeans.result = kmeans(data, centers=4, nstart = 100)
table(my_class, kmeans.result$cluster)
```

```
my_class  1  2  3  4
   blue  11  9  0  0
   green  0  0  0 20
   red    0  0 20  0
```

```
plot(pcs$x[,1:2],col=kmeans.result$cluster,pch=19)
```



Since we already knew that we had 3 my_classes, using 4 clusters would dived into more clusters than necessary. And we can see that it happened. The "middle" cluster got separated into two.

f-)

```
kmeans.res2 = kmeans(pcs$x[,1:2],centers=3, nstart = 100)
table(kmeans.res2$cluster,my_class)
```

```
   my_class
    blue green red
```

```
   1    20      0    0
   2     0     20    0
   3     0      0   20
```

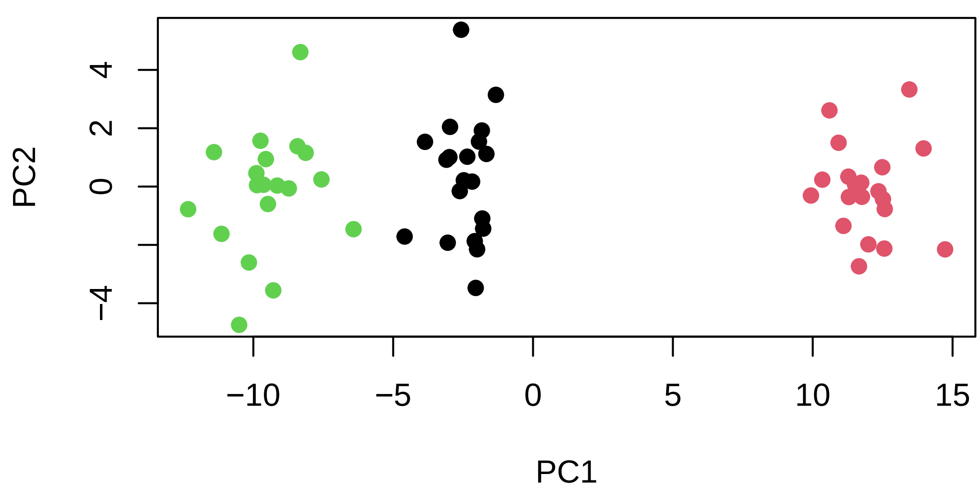
The results show that it perfectly separated the clusters.

g-)

```
kmeans.result = kmeans(scale(data), centers=3, nstart = 100)
table(my_class, kmeans.result$cluster)
```

```
my_class  1  2  3
   blue  20  0  0
   green  0  0 20
   red    0 20  0
```

```
plot(pcs$x[,1:2],col=kmeans.result$cluster,pch=19)
```



The outcomes are the same as part (b), where the assigned clusters are flawlessly mapped to the original my_classes.Since there was no overlapping in the simulated data. However, results from data sets with overlapping observations would probably differ.

Exercises required for MSSC PhD students

None.

Optional Exercises

None.