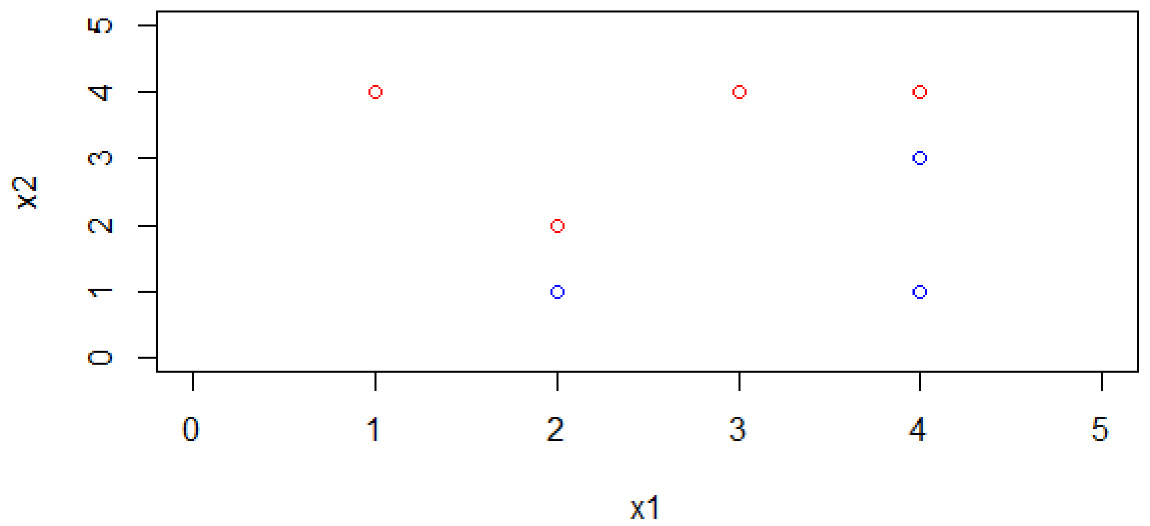
Jiayuan Guo - HW3 - R code

Chapter 9, Problem 3

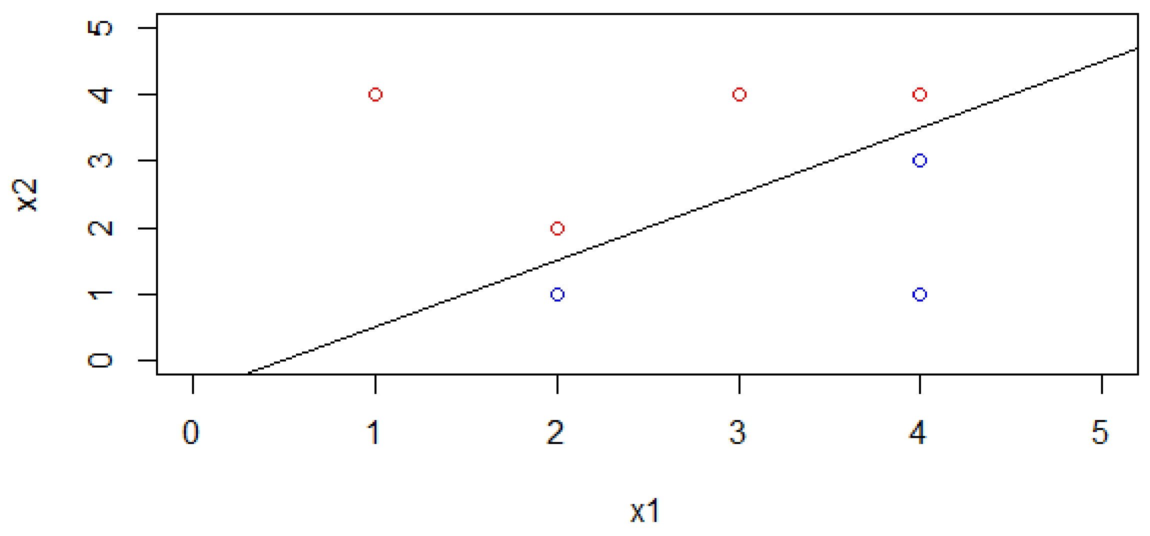
(a)

```
#Problem 3(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, xlim = c(0, 5), ylim = c(0, 5))
```



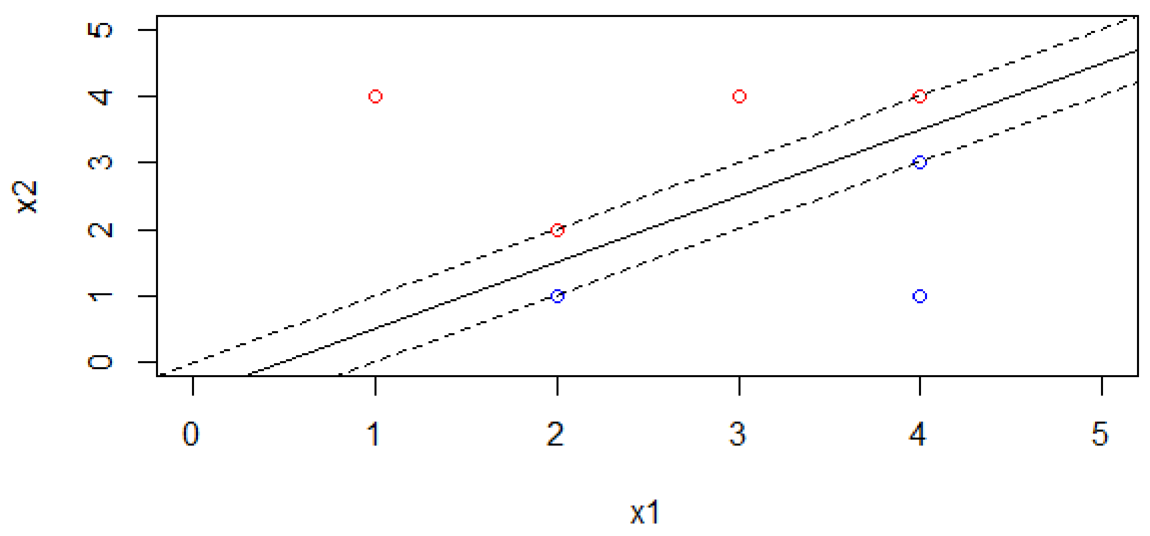
(b)

```
#Problem 3(b)
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
#Hyperplane equation is X2 = -0.5 + X1
abline(-0.5, 1)
```



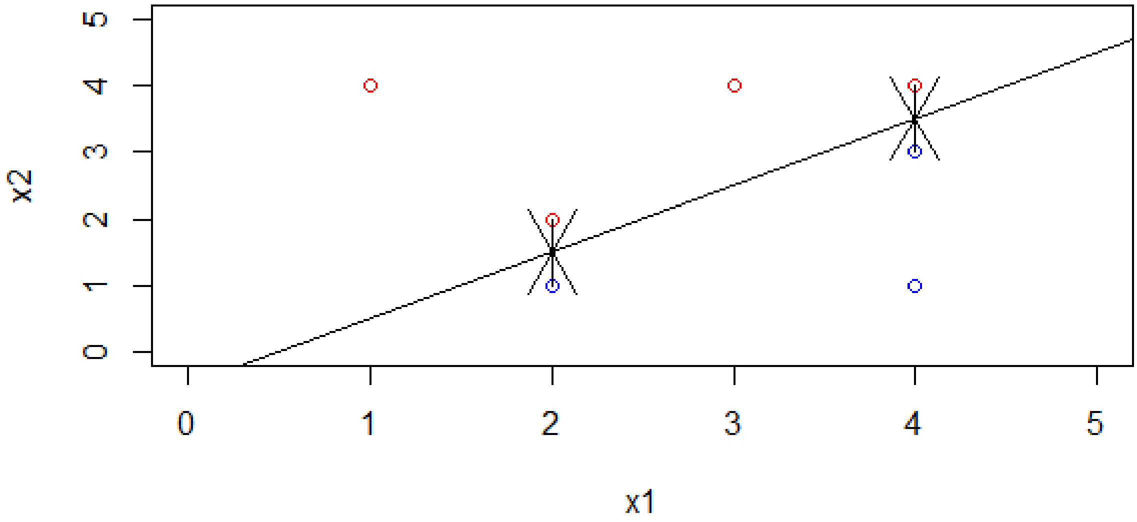
(d)

```
#Problem 3(d)
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.5, 1)
abline(-1, 1, lty = 2)
abline(0, 1, lty = 2)
```



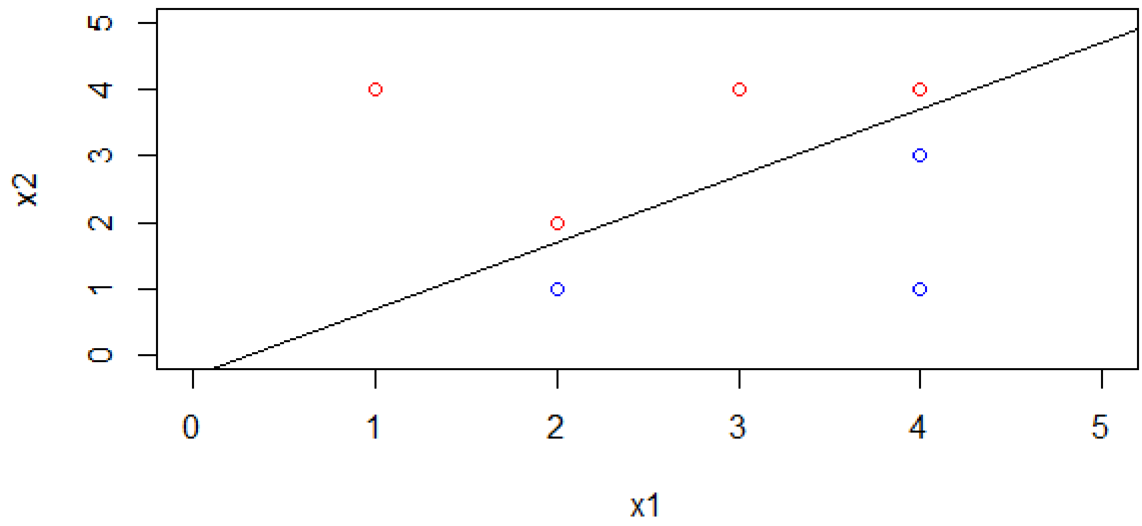
(e)

```
#Problem 3(e)
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.5, 1)
arrows(2, 1, 2, 1.5)
arrows(2, 2, 2, 1.5)
arrows(4, 4, 4, 3.5)
arrows(4, 3, 4, 3.5)
```



(g)

```
#Problem 3(g)
#Hyperplane X2 = -0.3 + X1 is not the optimal hyperplane seperating
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.3, 1)
```



(h)

```
#Problem 3(h)
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
#Additional observation is (2,4) blue
points(c(2), c(4), col = c("blue"))
```

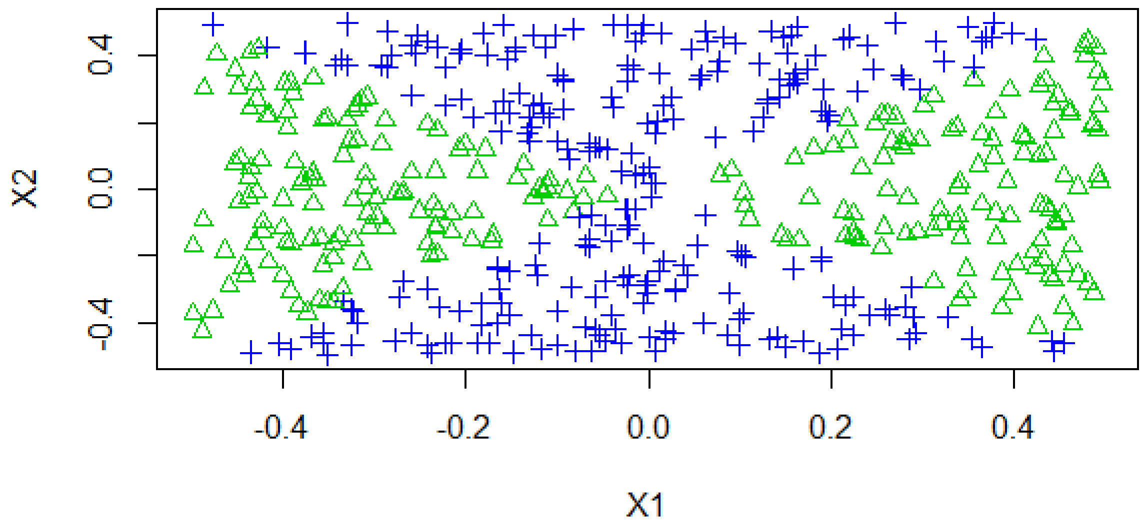
Chapter 9, Problem 5

(a)

```
#Problem 4(a)
set.seed(1)
x1 = runif(500) - 0.5
x2 = runif(500) - 0.5
y = 1 * (x1^2 - x2^2 > 0)
```

(b)

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



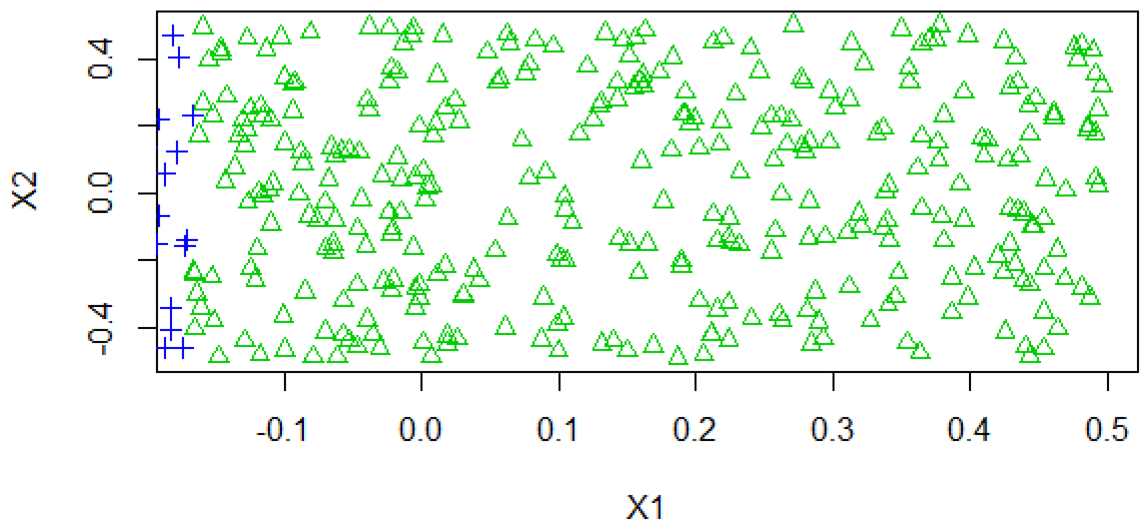
(c)

```
#Problem 4(c)
logit.fit = glm(y ~ x1 + x2, family = "binomial")
summary(logit.fit)
```

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

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-1.179   -1.139   -1.112    1.206    1.257

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.087260   0.089579  -0.974    0.330
x1           0.196199   0.316864   0.619    0.536
x2          -0.002854   0.305712  -0.009    0.993

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 692.18  on 499  degrees of freedom
Residual deviance: 691.79  on 497  degrees of freedom
AIC: 697.79

Number of Fisher Scoring iterations: 3
```

(d)

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



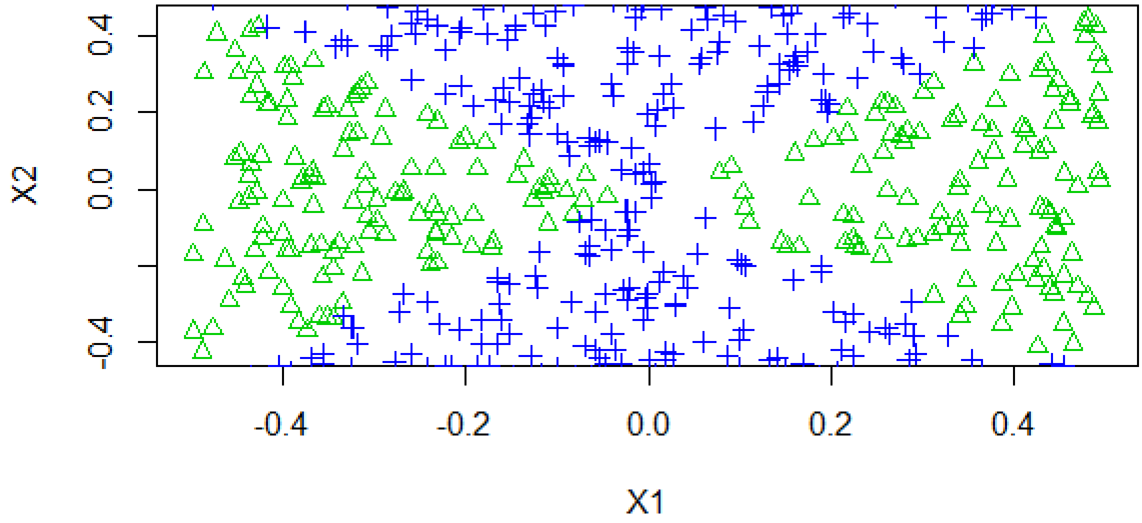
(e)

```
#Problem 4(e)
logitnl.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(logitnl.fit)
```

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

Deviance Residuals:
       Min           1Q       Median           3Q          Max
-8.240e-04  -2.000e-08  -2.000e-08   2.000e-08   1.163e-03

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    -102.2     4302.0  -0.024    0.981
poly(x1, 2)1   2715.3   141109.5   0.019    0.985
poly(x1, 2)2  27218.5   842987.2   0.032    0.974
poly(x2, 2)1   -279.7    97160.4  -0.003    0.998
poly(x2, 2)2 -28693.0   875451.3  -0.033    0.974
I(x1 * x2)     -206.4    41802.8  -0.005    0.996

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6.9218e+02  on 499  degrees of freedom
Residual deviance: 3.5810e-06  on 494  degrees of freedom
AIC: 12

Number of Fisher Scoring iterations: 25
```

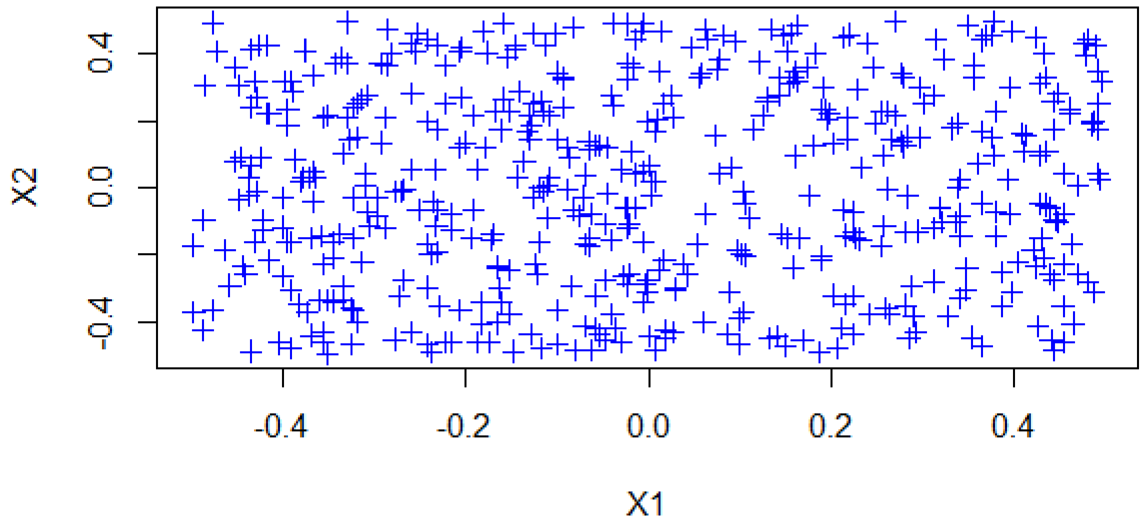
(f)

```
#Problem 5(f)
probs = predict(logitnl.fit, data, type = "response")
preds = rep(0, 500)
preds[probs > 0.47] = 1
plot(data[preds == 1, ]$x1, data[preds == 1, ]$x2, col = (4 - 1), pch = (3 - 1), xlab = "X1",
ylab = "X2")
points(data[preds == 0, ]$x1, data[preds == 0, ]$x2, col = (4 - 0), pch = (3 - 0))
```



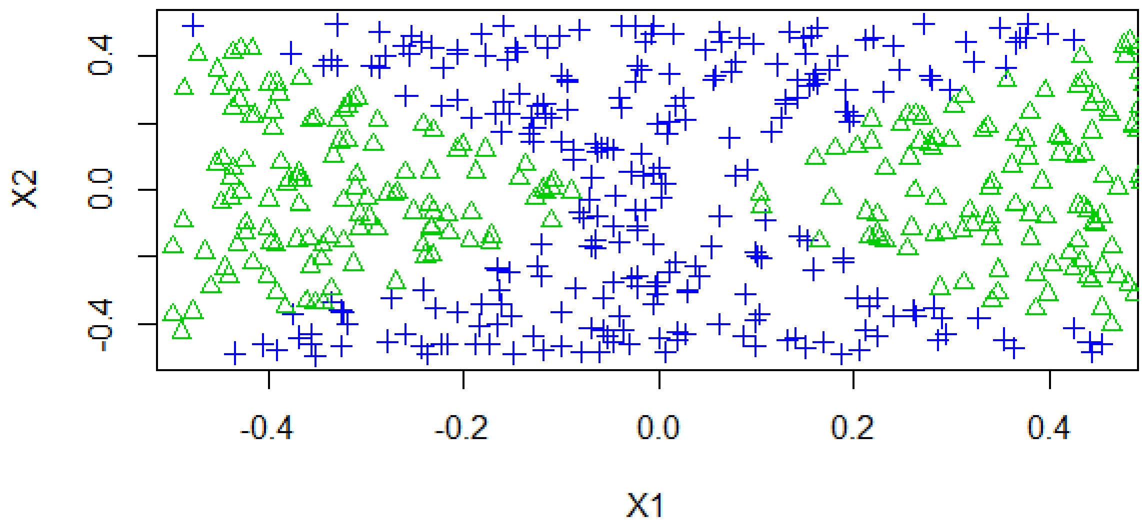
(g)

```
#Problem 5(g)
library(e1071)
data$y = as.factor(data$y)
svm.fit = svm(y ~ x1 + x2, data, kernel = "linear", cost = 0.01)
preds = predict(svm.fit, data)
plot(data[preds == 0, ]$x1, data[preds == 0, ]$x2, col = (4 - 0), pch = (3 - 0), xlab = "X1",
ylab = "X2")
points(data[preds == 1, ]$x1, data[preds == 1, ]$x2, col = (4 - 1), pch = (3 - 1))
```



(h)

```
#Problem 5(h)
data$y = as.factor(data$y)
svmnl.fit = svm(y ~ x1 + x2, data, kernel = "radial", gamma = 1)
preds = predict(svmnl.fit, data)
plot(data[preds == 0, ]$x1, data[preds == 0, ]$x2, col = (4 - 0), pch = (3 - 0), xlab = "X1",
ylab = "X2")
points(data[preds == 1, ]$x1, data[preds == 1, ]$x2, col = (4 - 1), pch = (3 - 1))
```



6. Using the Boston data set from the MASS package, fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore logistic regression, LDA, QDA and KNN models using various subsets of the predictors. Also try penalized logistic regression (ridge and lasso), as well as SVM using the optimal choices of tuning parameters for each method. Describe your findings.

```
#Problem 6
#Load the data
library(ISLR)
library(MASS)
library(caret)
data("Boston")
attach(Boston)
#Get the information about variables
help(Boston)
```

```
crim01 = rep(0, length(Boston$crim))
crim01[Boston$crim > median(Boston$crim)] <- 1
Boston = data.frame(Boston, crim01)
summary(Boston)
```

```
      crim                zn             indus            chas        
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
 1st Qu.: 0.08204   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              rad        
 Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130   Min.   : 1.000  
 1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100   1st Qu.: 4.000  
 Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207   Median : 5.000  
 Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795   Mean   : 9.549  
 3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188   3rd Qu.:24.000  
 Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.000  
      tax           ptratio          black            lstat            medv      
 Min.   :187.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.73   Min.   : 5.00  
 1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38   1st Qu.: 6.95   1st Qu.:17.02  
 Median :330.0   Median :19.05   Median :391.44   Median :11.36   Median :21.20  
 Mean   :408.2   Mean   :18.46   Mean   :356.67   Mean   :12.65   Mean   :22.53  
 3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23   3rd Qu.:16.95   3rd Qu.:25.00  
 Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.97   Max.   :50.00  
     crim01   
 Min.   :0.0  
 1st Qu.:0.0  
 Median :0.5  
 Mean   :0.5  
 3rd Qu.:1.0  
 Max.   :1.0  
```

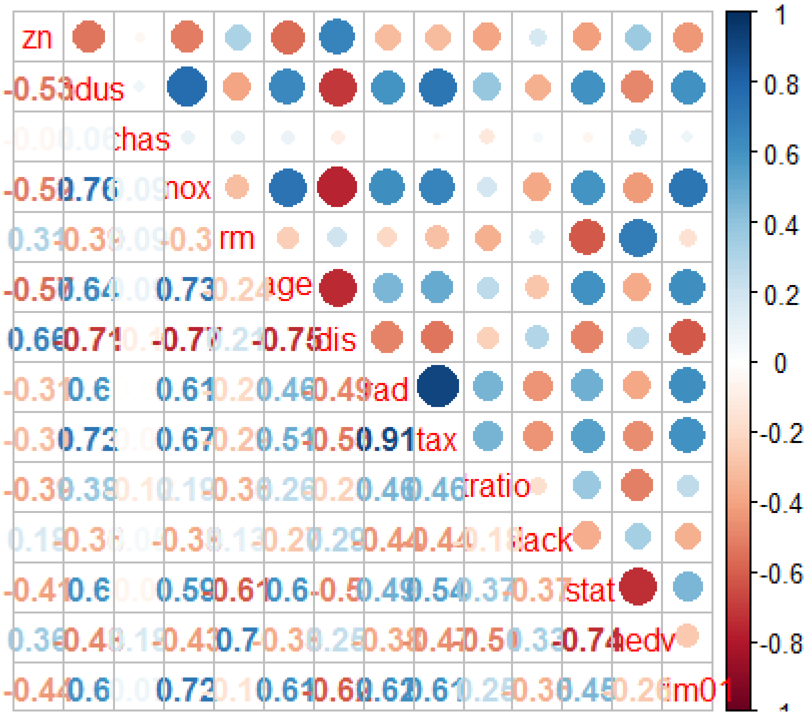
```
set.seed(1234)
train = sample(1:dim(Boston)[1], dim(Boston)[1]*.7, rep=FALSE)
test = -train
Boston.train = Boston[train, ]
Boston.test = Boston[test, ]
crim01.test = crim01[test]
```

Logistic Regression

```
#Logistic regression
fit.glm1 = glm(crim01 ~ . - crim01 - crim, data = Boston, family = binomial)
fit.glm1
```

```
Call:  glm(formula = crim01 ~ . - crim01 - crim, family = binomial,
    data = Boston)

Coefficients:
(Intercept)           zn        indus         chas          nox           rm
 -34.103704    -0.079918    -0.059389     0.785327    48.523782    -0.425596
        age          dis          rad          tax      ptratio        black
   0.022172     0.691400     0.656465    -0.006412     0.368716    -0.013524
      lstat         medv
   0.043862     0.167130

Degrees of Freedom: 505 Total (i.e. Null);  492 Residual
Null Deviance:       701.5
Residual Deviance: 211.9     AIC: 239.9
```

```
library(corrplot)
corrplot::corrplot.mixed(cor(Boston[, -1]), upper="circle")
```



```
fit.glm = glm(crim01 ~ nox + indus + age + rad, data = Boston, family = binomial)
probs = predict(fit.glm, Boston.test, type = "response")
pred.glm = rep(0, length(probs))
pred.glm[probs > 0.5] = 1
table(pred.glm, crim01.test)
```

```
        crim01.test
pred.glm  0  1
       0 65 15
       1  4 68
```

```
> mean(pred.glm != crim01.test)
[1] 0.125
```

The test error rate of logistic regression is 12.5%.

LDA

```
#LDA
fit.lda = lda(crim01 ~ nox + indus + age + rad , data = Boston)
pred.lda = predict(fit.lda, Boston.test)
table(pred.lda$class, crim01.test)
```

```
   crim01.test
     0  1
  0 66 20
  1  3 63
```

```
> mean(pred.lda$class != crim01.test)
[1] 0.1513158
```

The test error rate of LDA is 15.13%.

QDA

```
#QDA
fit.qda = qda(crim01 ~ nox + indus + age + rad , data = Boston)
pred.qda = predict(fit.qda, Boston.test)
table(pred.qda$class, crim01.test)
```

```
   crim01.test
     0  1
  0 68 27
  1  1 56
```

```
> mean(pred.qda$class != crim01.test)
[1] 0.1842105
```

The test error rate of QDA is 18.42%.

KNN

```r
#KNN
data = scale(Boston[,-c(1,15)])
set.seed(1234)
train = sample(1:dim(Boston)[1], dim(Boston)[1]*.7, rep=FALSE)
test = -train
#In KNN, we get training_y and testing_y seperately
training_data = data[train, c("nox" , "indus" , "age" , "rad")]
testing_data = data[test, c("nox" , "indus" , "age" , "rad")]
train.crime01 = Boston$crim01[train]
test.crime01= Boston$crim01[test]
library(class)
set.seed(1234)
knn_pred_y = knn(training_data, testing_data, train.crime01, k = 1)
table(knn_pred_y, test.crime01)
```

```
          test.crime01
knn_pred_y  0  1
         0 62  7
         1  7 76
```

```
> mean(knn_pred_y != test.crime01)
[1] 0.09210526
```

When k=1, the test error rate of KNN is 9.21%.

```r
knn_pred_y = NULL
error_rate = NULL
for(i in 1:dim(testing_data)[1]){
  set.seed(1234)
  knn_pred_y = knn(training_data,testing_data,train.crime01,k=i)
  error_rate[i] = mean(test.crime01 != knn_pred_y)
}
```

```r
### find the minimum error rate and corresponding k value
min_error_rate = min(error_rate)
print(min_error_rate)
K = which(error_rate == min_error_rate)
print(K)
```

```
> print(min_error_rate)
[1] 0.06578947
> print(K)
[1] 3
```

When k=3, we get the minimum test error rate of KNN, which is 6.57%