hw7

Question #1a

```
options(repos = list(CRAN="http://cran.rstudio.com/"))
install.packages("ggplot2")
```

```
##
## The downloaded binary packages are in
##  /var/folders/9r/f60_frl94wzd1l2rhq76j3sm0000gn/T//RtmpmXVZMS/downloaded_packages
```

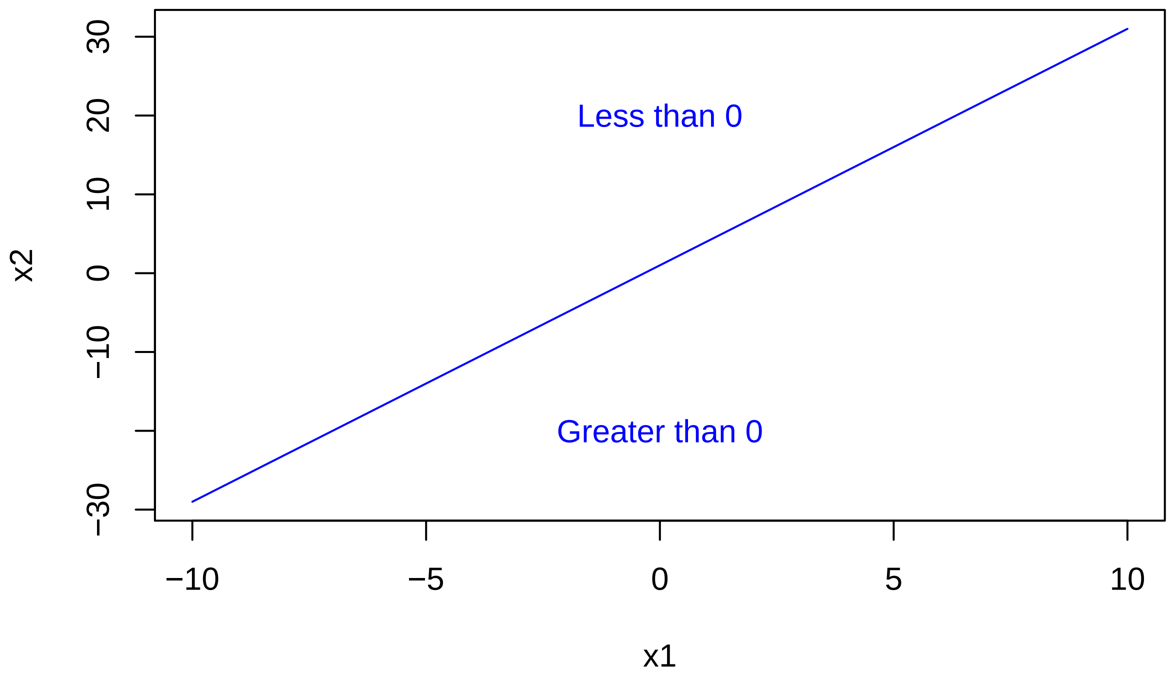
```
install.packages("latex2exp")
```

```
##
## The downloaded binary packages are in
##  /var/folders/9r/f60_frl94wzd1l2rhq76j3sm0000gn/T//RtmpmXVZMS/downloaded_packages
```

```
library(ggplot2)
library(latex2exp)
```

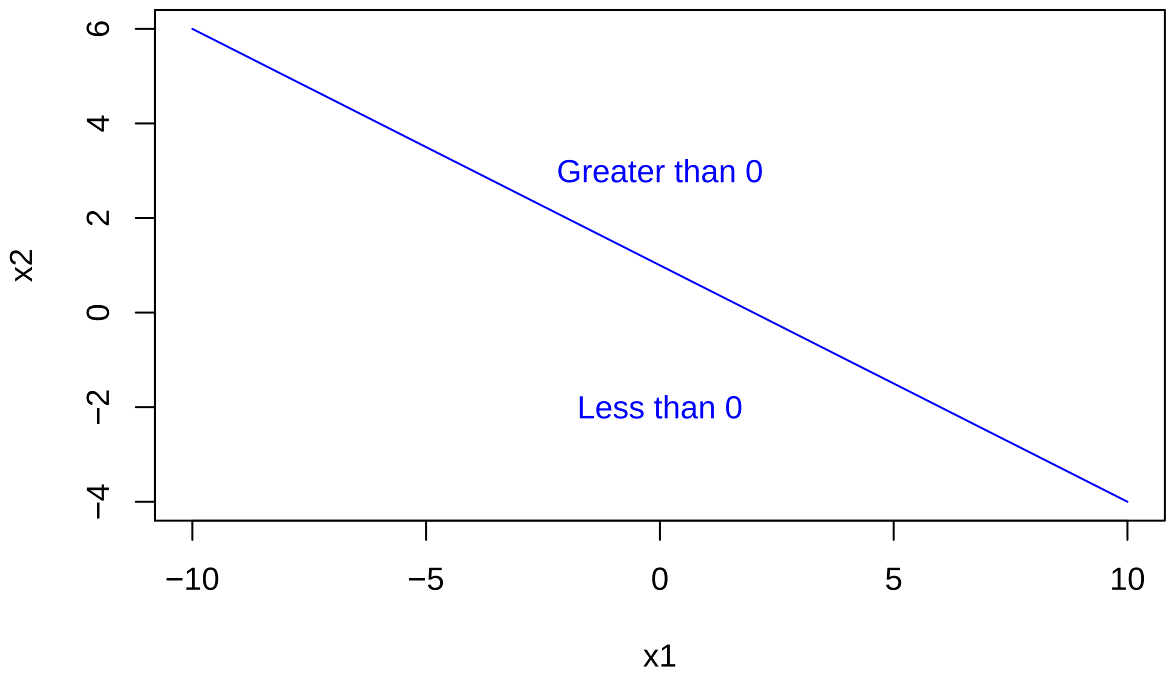
```
## Warning: package 'latex2exp' was built under R version 4.0.5
```

```
x1 <- -10:10
x2 <- 3*x1 + 1

plot(x1, x2, type = "l", col = "blue")
text(c(0), c(-20), "Greater than 0", col = "blue")
text(c(0), c(20), "Less than 0", col = "blue")
```



Question #1b

```r
x1 <- -10:10
x2 <- (-1/2)*x1 + 1

plot(x1, x2, type = "l", col = "blue")
text(c(0), c(3), "Greater than 0", col = "blue")
text(c(0), c(-2), "Less than 0", col = "blue")
```



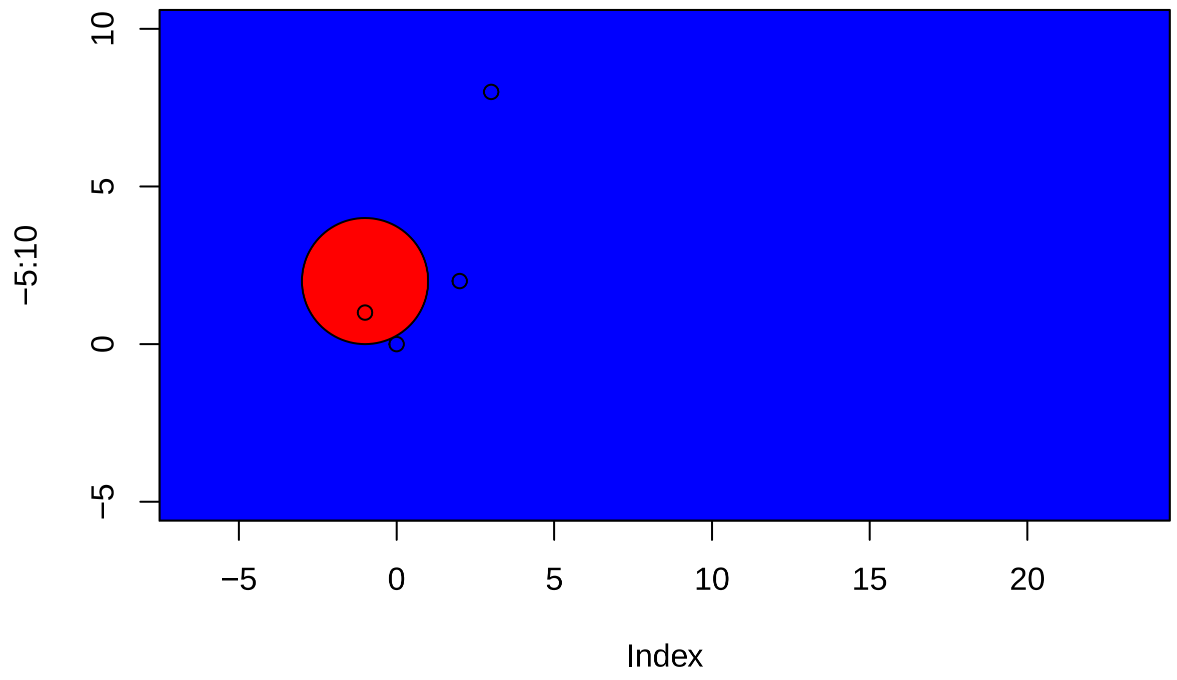
Question #2

(a)

```
install.packages('plotrix')
```

```
##
## The downloaded binary packages are in
##  /var/folders/9r/f60_frl94wzd1l2rhq76j3sm0000gn/T//RtmpmXVZMS/downloaded_packages
```

```
library(plotrix)
plot.new()
rect(par("usr")[1], par("usr")[3], par("usr")[2],
     par("usr")[4], col="blue")
par(new=TRUE)
plot(-5:10, type="n", asp=1)
draw.circle(-1,2,2, col='red')
points(x = c(0, -1, 2, 3), y = c(0, 1, 2, 8), col = 'black')
```



(b) The blue region is where the set of points would be >4 and the red region is where the set of points would be <= 4.

(c) The points are shown on the plot. (0, 0) is in the blue class. (-1, 1) is in the red class. (2, 2) and (3, 8) are in the blue class.

(d) It is linear in terms of X_1, X_1^2, X_2, X_2^2 because the equation could be expanded into $2X_1$ - $4X_2$ + X_1^2 + X_2^2 + 1 = 0 which is linear in terms of the listed variables.

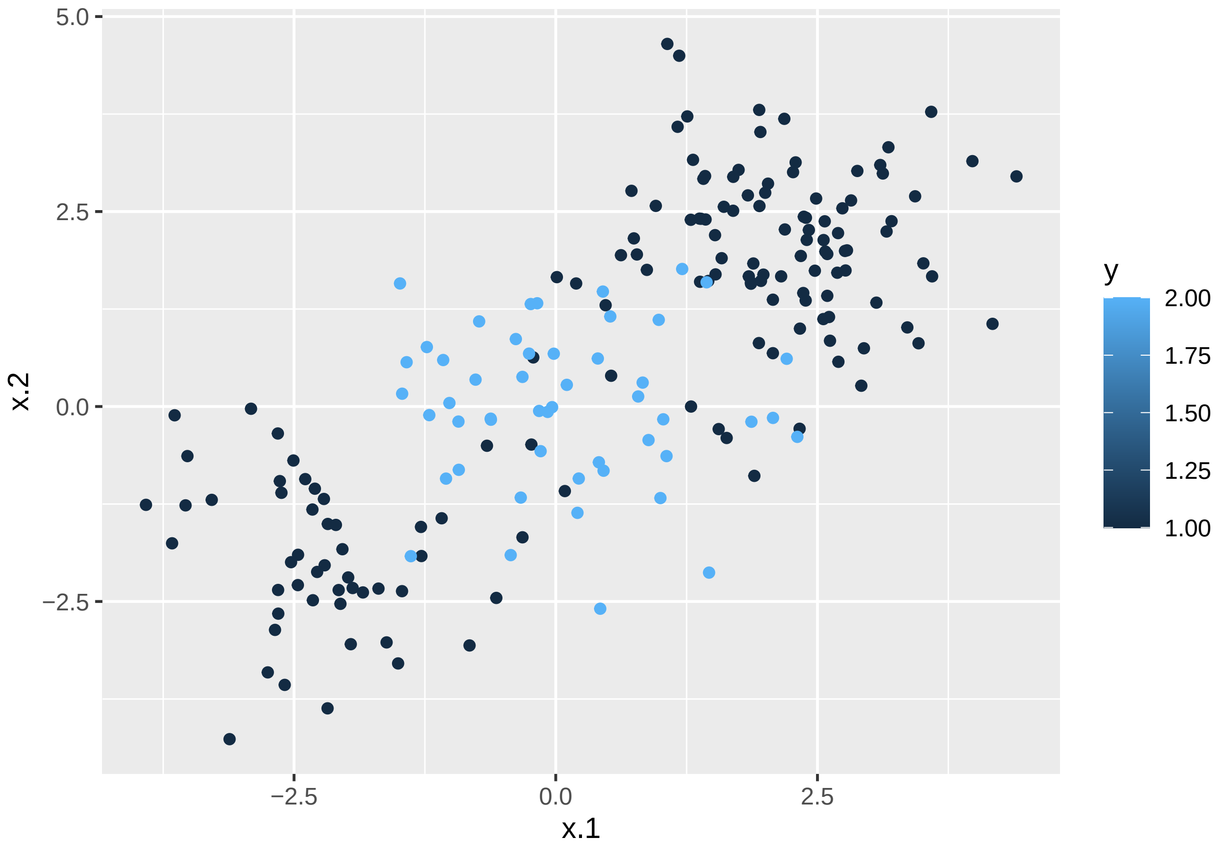
Question 3

###a

```
train = read.csv("SVM_train.csv")
head(train)
```

```
##   X      x.1        x.2 y
## 1 1 1.373546  2.4094018 1
## 2 2 2.183643  3.6888733 1
## 3 3 1.164371  3.5865884 1
## 4 4 3.595281  1.6690922 1
## 5 5 2.329508 -0.2852355 1
## 6 6 1.179532  4.4976616 1
```

```r
library(ggplot2)
ggplot(data=train, aes(x=x.1, y=x.2, color=y))+ geom_point()
```



The class labeled 2 is mostly centered around 0,0 and the other class, 1, is spread out on the upper right and bottom left side of the other class. The two classes are somewhat visually separable, but not perfectly separable: there are points that are almost on top of each other and would require over fitting to perfectly predict. The decision boundry will definitely not be linear, and closer to a circle.

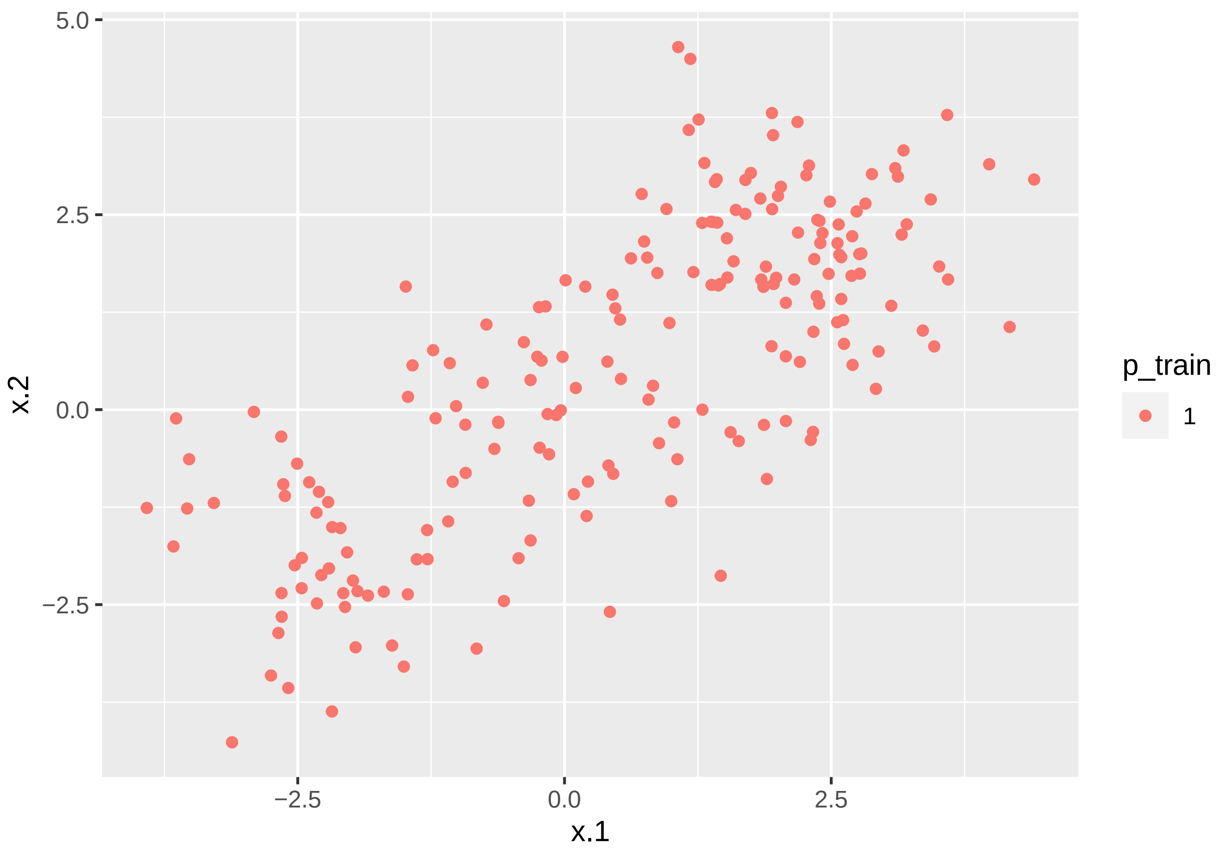
###b

```r
#install.packages("e1071")
library("e1071")
library("ggplot2")
col<-c("x.1", "x.2", "y")
train = train[,col]
train[,3] <- sapply(train[,3], as.factor)
svmfit_1 <- svm(y ~., data = train, kernel = "linear", cost = .001, scale = FALSE)

tuned <- tune(svm, y ~., data = train, kernel = "linear",
              ranges = list(cost=c(0.001,0.01,.1,1,10,100)))
# Will show the optimal cost parameter
summary(tuned)
```

```
##
## Parameter tuning of 'svm':
##
```

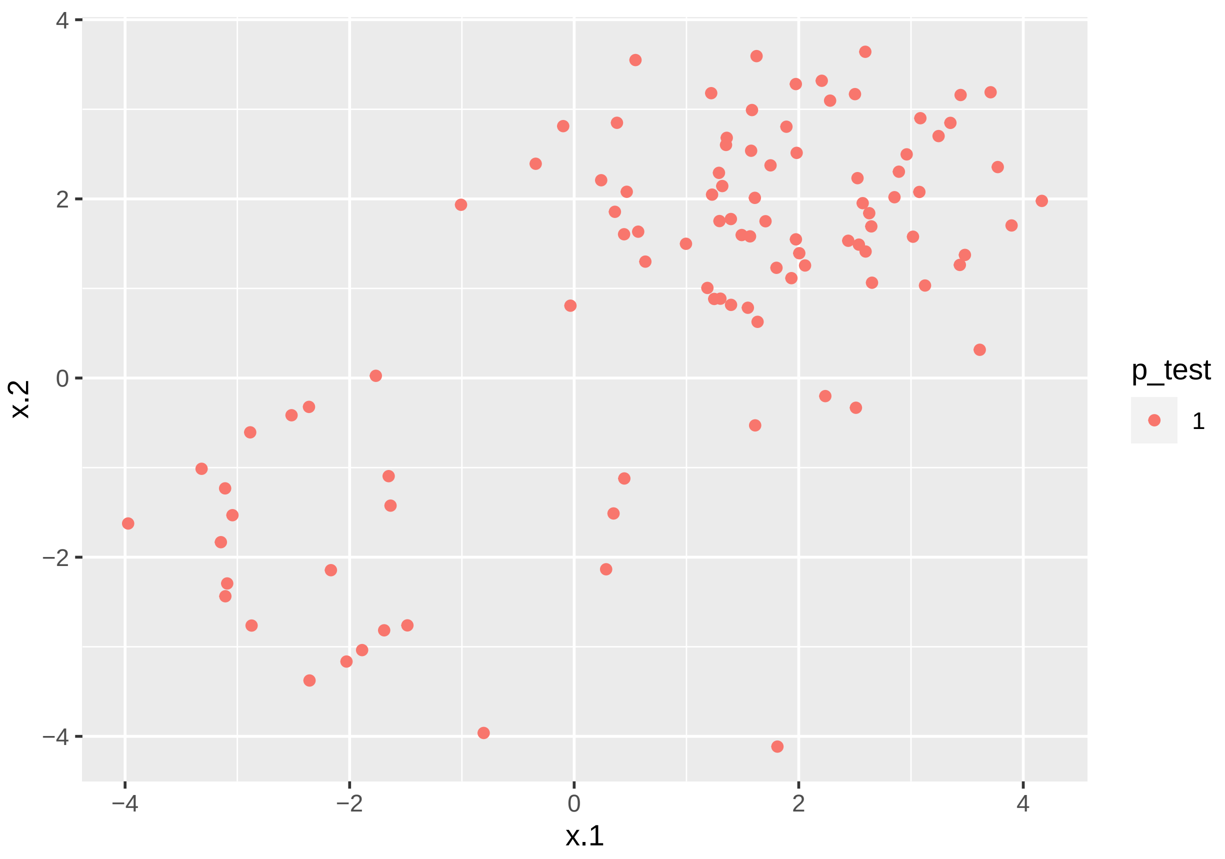
```
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost
##  0.001
##
## - best performance: 0.25
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03  0.25  0.1154701
## 2 1e-02  0.25  0.1154701
## 3 1e-01  0.25  0.1154701
## 4 1e+00  0.25  0.1154701
## 5 1e+01  0.25  0.1154701
## 6 1e+02  0.25  0.1154701
```

```
p_train <- predict(svmfit_1, train[,col], type="class")

ggplot(data=train, aes(x=x.1, y=x.2, color=p_train))+ geom_point()
```



```
library("ggplot2")
test = read.csv("SVM_test.csv")
col<-c("x.1", "x.2", "y")
test = test[,col]
```

```
test[,3] <- sapply(test[,3], as.factor)
p_test <- predict(svmfit_1, test[,col], type="class")
ggplot(data=test, aes(x=x.1, y=x.2, color=p_test))+ geom_point()
```



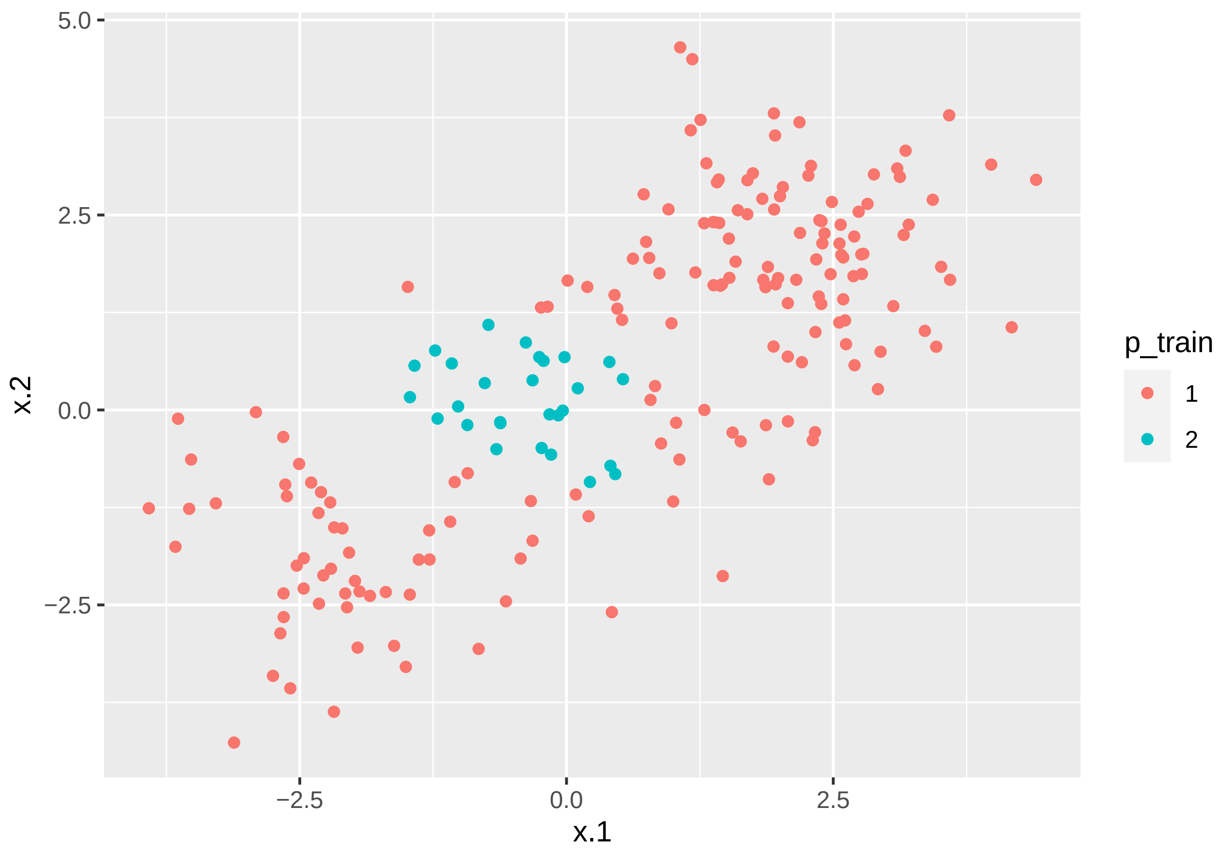
###c

```
library("ggplot2")
svmfit <- svm(y ~., data = train, kernel = "radial", cost = .1, gamma = .1, scale = FALSE)

tuned <- tune(svm, y ~., data = train, kernel = "radial",
              ranges = list(cost=c(0.001,0.01,.1,1,10,100), gamma=c(0.001,0.01,.1,1,10,100)))
summary(tuned)
```

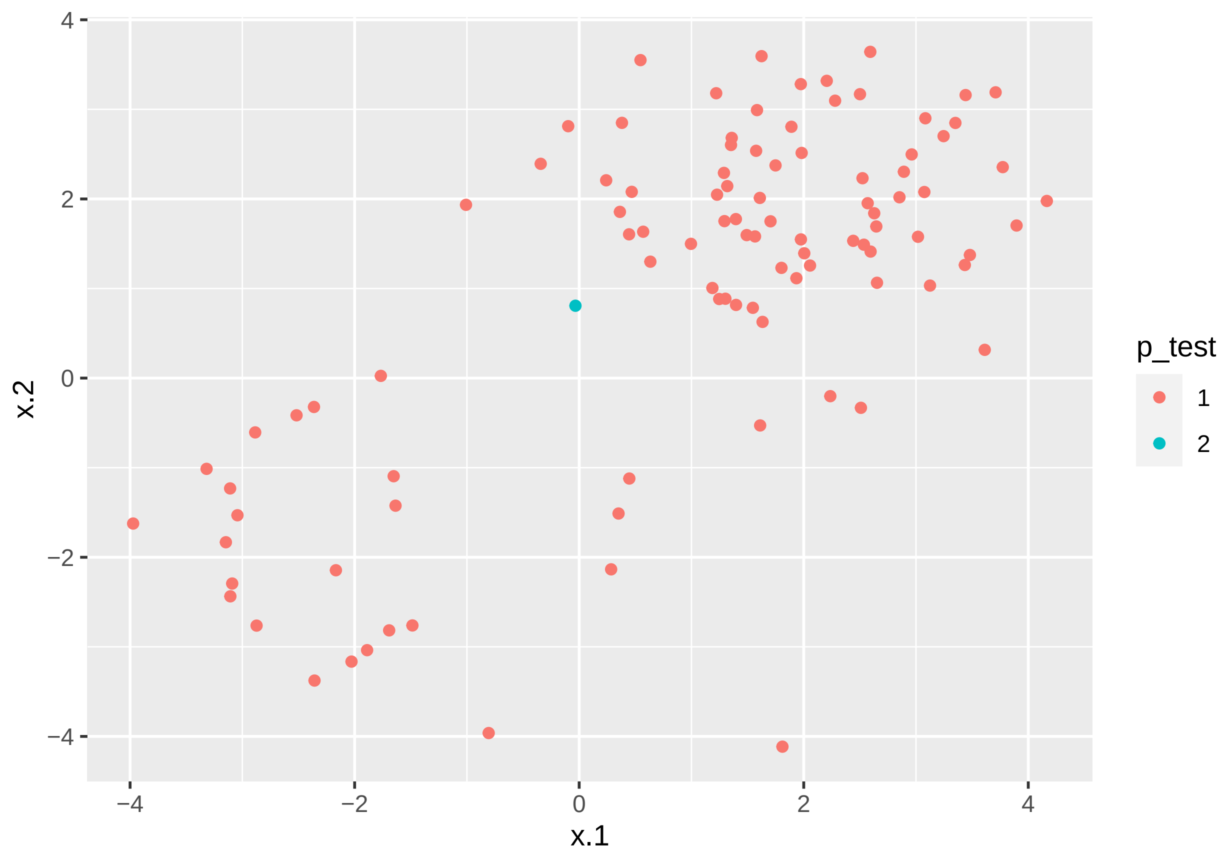
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##   100   0.1
##
## - best performance: 0.105
##
## - Detailed performance results:
```

```
##     cost gamma error dispersion
## 1  1e-03 1e-03 0.250 0.11303883
## 2  1e-02 1e-03 0.250 0.11303883
## 3  1e-01 1e-03 0.250 0.11303883
## 4  1e+00 1e-03 0.250 0.11303883
## 5  1e+01 1e-03 0.250 0.11303883
## 6  1e+02 1e-03 0.250 0.11303883
## 7  1e-03 1e-02 0.250 0.11303883
## 8  1e-02 1e-02 0.250 0.11303883
## 9  1e-01 1e-02 0.250 0.11303883
## 10 1e+00 1e-02 0.250 0.11303883
## 11 1e+01 1e-02 0.250 0.11303883
## 12 1e+02 1e-02 0.115 0.07090682
## 13 1e-03 1e-01 0.250 0.11303883
## 14 1e-02 1e-01 0.250 0.11303883
## 15 1e-01 1e-01 0.250 0.11303883
## 16 1e+00 1e-01 0.150 0.10540926
## 17 1e+01 1e-01 0.110 0.06146363
## 18 1e+02 1e-01 0.105 0.06433420
## 19 1e-03 1e+00 0.250 0.11303883
## 20 1e-02 1e+00 0.250 0.11303883
## 21 1e-01 1e+00 0.120 0.05374838
## 22 1e+00 1e+00 0.115 0.05797509
## 23 1e+01 1e+00 0.120 0.06749486
## 24 1e+02 1e+00 0.150 0.08498366
## 25 1e-03 1e+01 0.250 0.11303883
## 26 1e-02 1e+01 0.250 0.11303883
## 27 1e-01 1e+01 0.250 0.11303883
## 28 1e+00 1e+01 0.125 0.07168604
## 29 1e+01 1e+01 0.140 0.09944289
## 30 1e+02 1e+01 0.190 0.10219806
## 31 1e-03 1e+02 0.250 0.11303883
## 32 1e-02 1e+02 0.250 0.11303883
## 33 1e-01 1e+02 0.250 0.11303883
## 34 1e+00 1e+02 0.230 0.10055402
## 35 1e+01 1e+02 0.205 0.09559754
## 36 1e+02 1e+02 0.205 0.08316650
```

```r
svmfit_2 <- svm(y ~., data = train, kernel = "radial", cost = 0.1, gamma = 1, scale = FALSE)
p_train <- predict(svmfit_2, train[,col], type="class")
p_test <- predict(svmfit_2, test[,col], type="class")
ggplot(data=train, aes(x=x.1, y=x.2, color=p_train))+ geom_point()
```

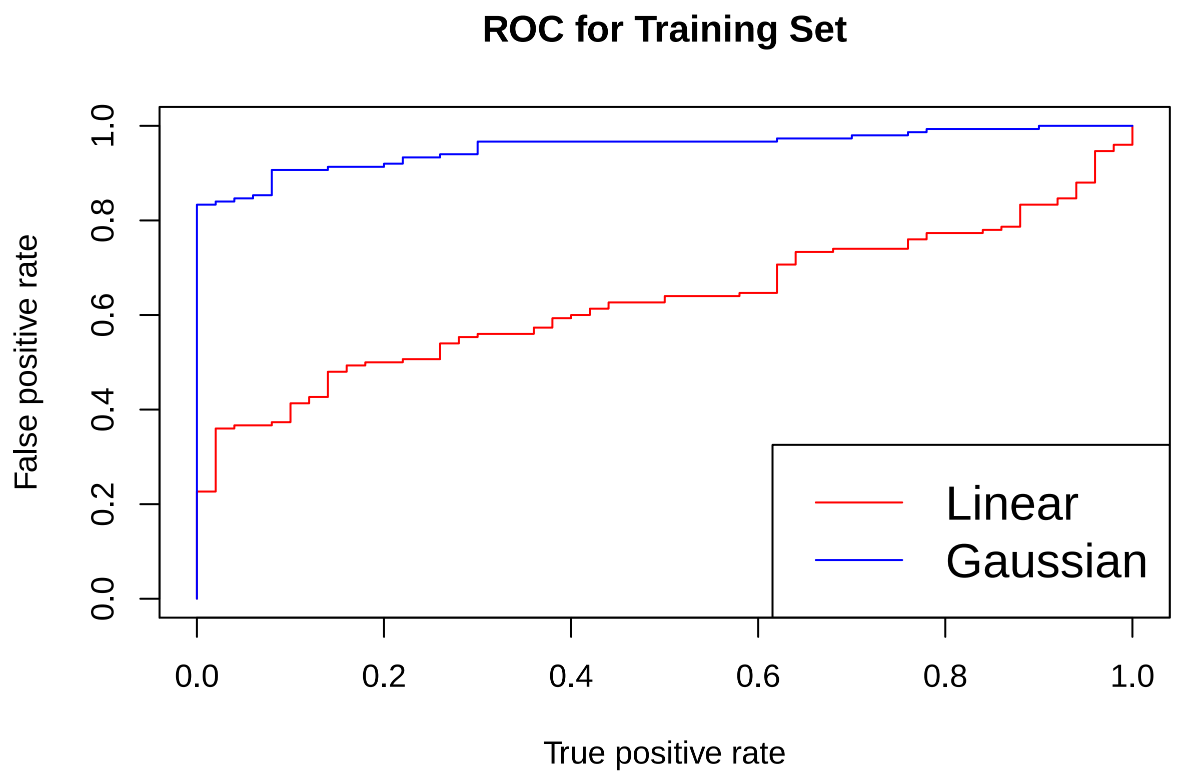


```
ggplot(data=test, aes(x=x.1, y=x.2, color=p_test))+ geom_point()
```

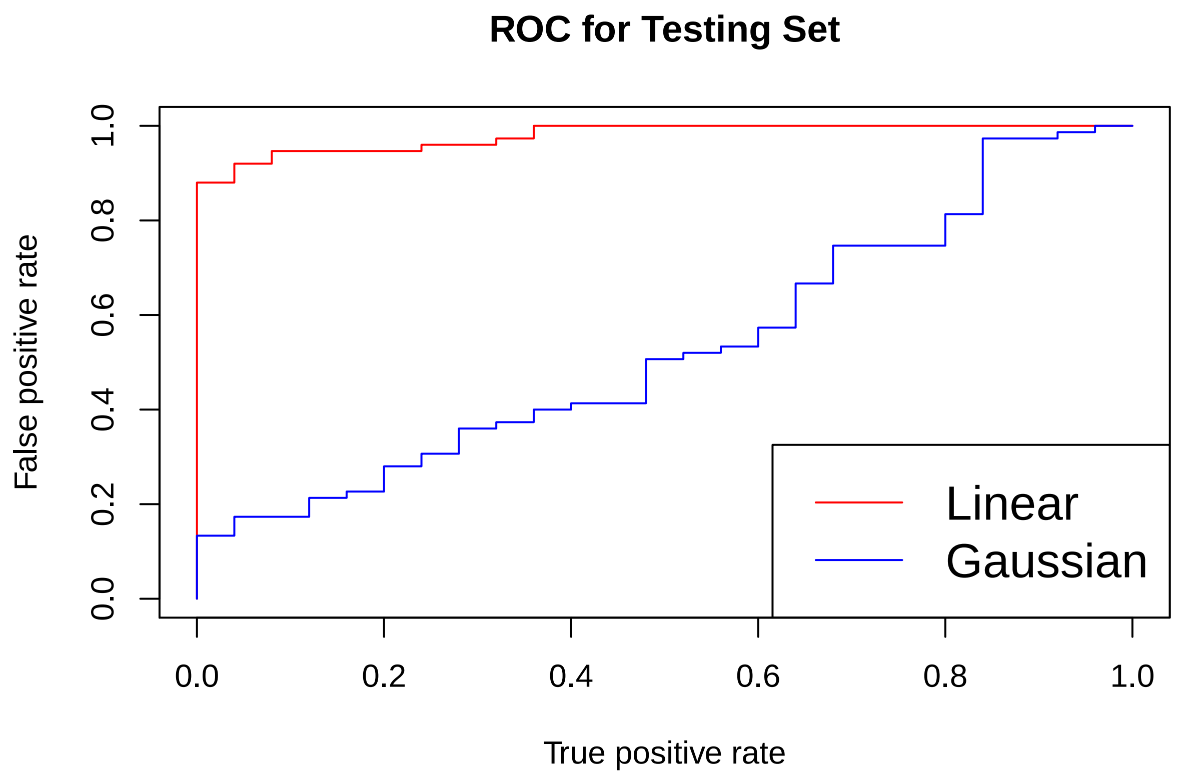


###d

```
rocplot =function (pred , truth , ...){
  predob = prediction(pred , truth )
  perf=performance(predob , "fpr", "tpr")
  plot(perf ,...)}

# Prepare the training and testing predictions for ROC curve
linear.opt=svm(y~., train, kernel ="linear", cost=0.001, decision.values =TRUE)
linear.train=attributes(predict (linear.opt, train, decision.values =TRUE))$decision.values
linear.test=attributes(predict (linear.opt, test, decision.values =TRUE))$decision.values

gaussian.opt=svm(y~., train, kernel ="radial", gamma=0.1, cost=1, decision.values =TRUE)
gaussian.train=attributes(predict (gaussian.opt, train, decision.values =TRUE))$decision.values
gaussian.test=attributes(predict (gaussian.opt, test, decision.values =TRUE))$decision.values

#install.packages("ROCR")
library(ROCR)
rocplot(linear.train, train[,"y"], main="ROC for Training Set", col="red")
rocplot(gaussian.train, train[,"y"], add=T, col="blue")
legend("bottomright", legend=c("Linear", "Gaussian"),
       col=c("red", "blue"), lty=c(1,1,1), cex=1.5)
```



```
# Plot ROC curves for testing set
rocplot(linear.test, test[,"y"], main="ROC for Testing Set", col="red")
rocplot(gaussian.test, test[,"y"], add=T, col="blue")
legend("bottomright", legend=c("Linear", "Gaussian"),
       col=c("red", "blue"), lty=c(1,1,1), cex=1.5)
```



According to the plot, the Gaussian kernel has better performance on the training set while linear kernel has better performance on the testing set.