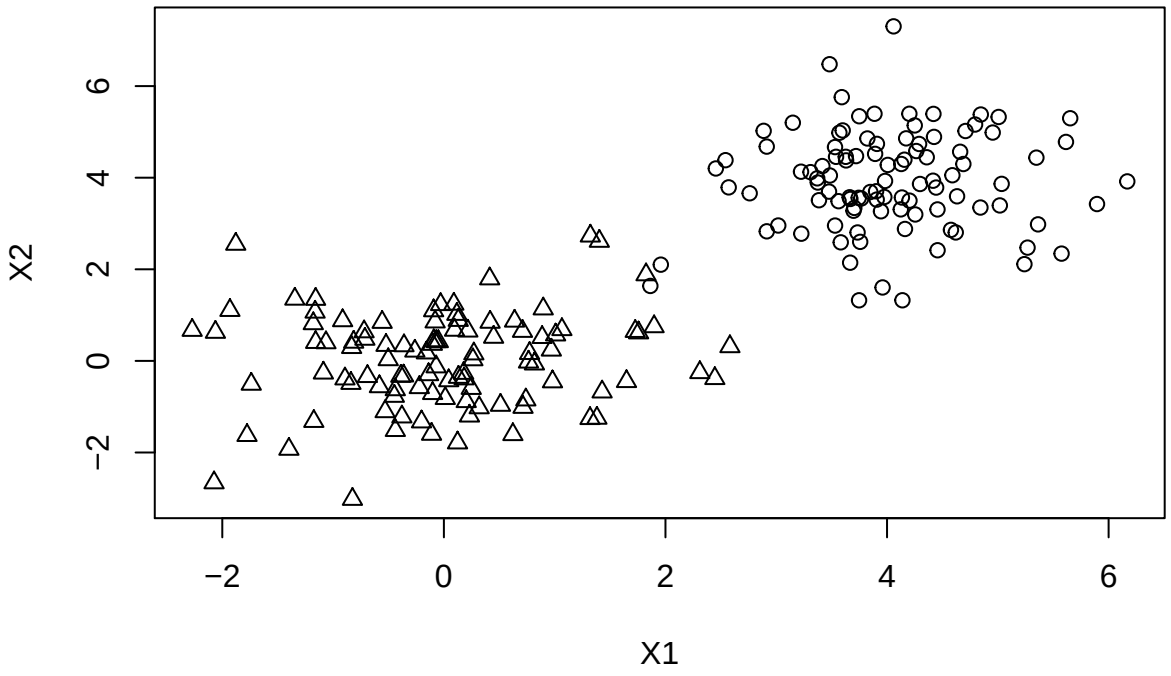
154Lab11

Jiyoon Clover Jeong

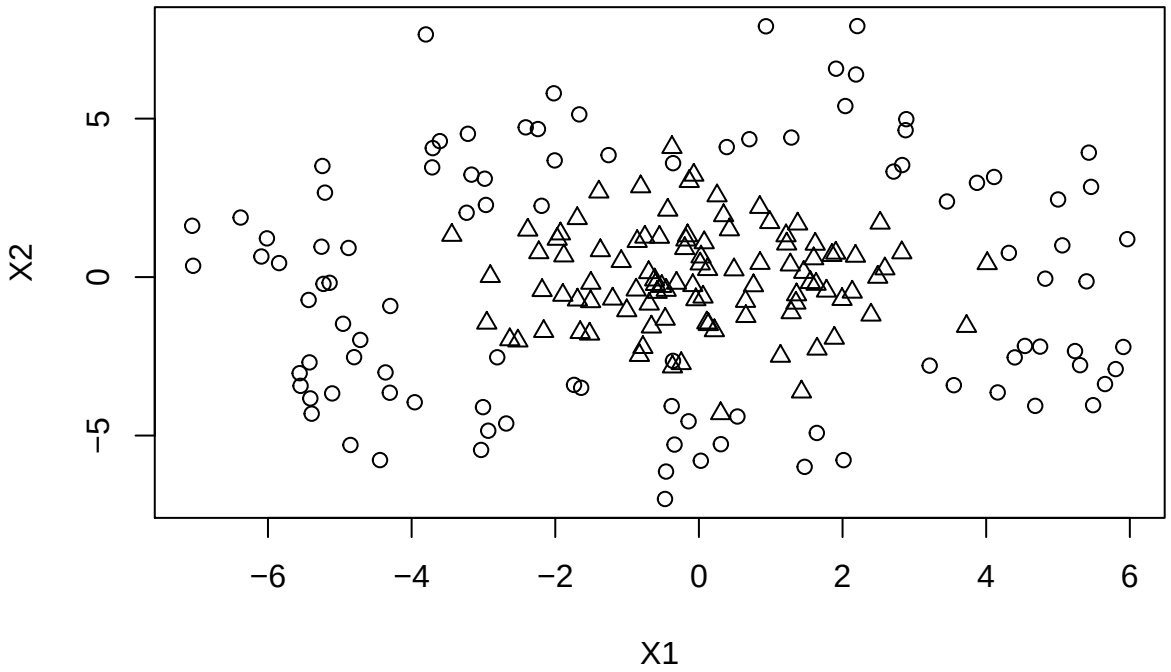
11/10/2017

Introduction

```
set.seed(100)


X1 <- c(rnorm(100), rnorm(100, mean = 4))
X2 <- c(rnorm(100), rnorm(100, mean = 4))
y <- factor(c(rep(0,100), rep(1,100)))
df1 <- data.frame(X1, X2, y)

set.seed(200)


r <- c(runif(100, 1, 2), runif(100, 5, 6))
theta <- runif(200, 0, 2 * pi)
X1 <- r * cos(theta) + rnorm(200)
X2 <- r * sin(theta) + rnorm(200)
y <- factor(c(rep(0,100), rep(1,100)))
df2 <- data.frame(X1, X2, y)

pchs <- c(2,1)
with(df1, plot(X1, X2, pch = pchs[y]))
```



```
pchs <- c(2,1)
with(df2, plot(X1, X2, pch = pchs[y]))
```



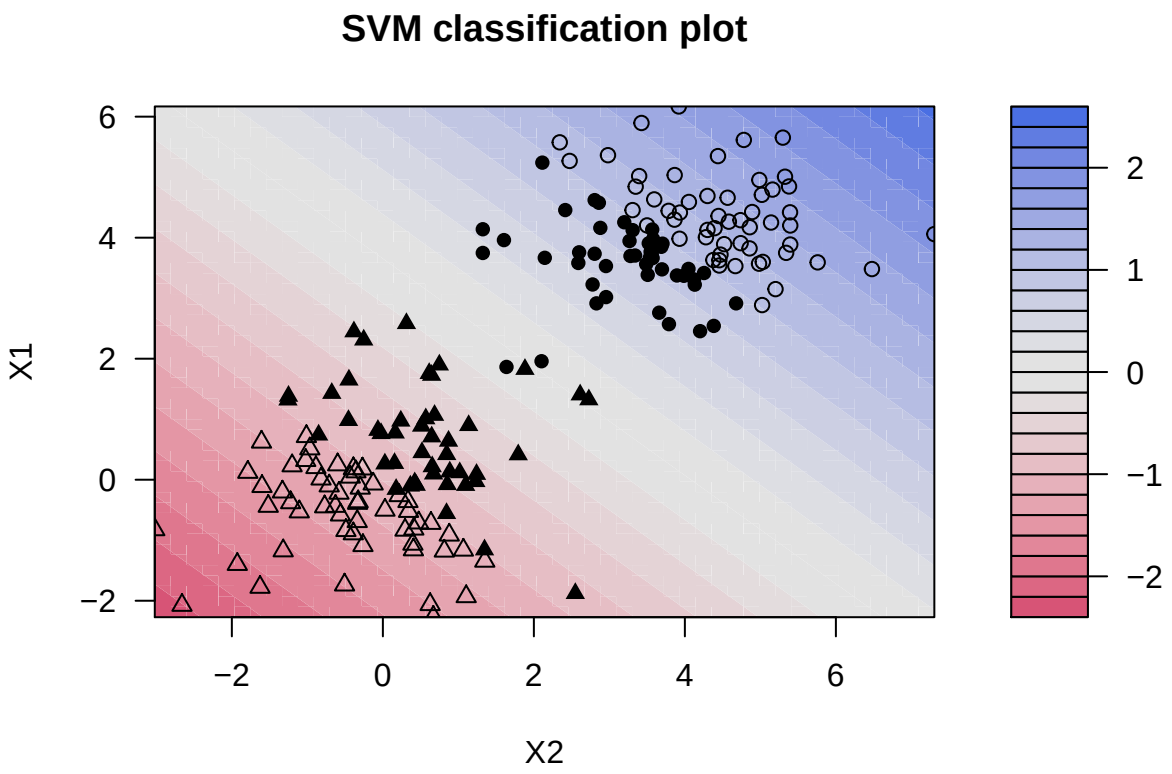
first plot : since we defined the datasets(X1, X2) to be separated by rnorm having different mean, the response variables seems like to be well separated.

second plot : We defined X1 to be a vector of r * cos(θ) + noise and X2 to be a vector of r * sin(θ) + noise and r for former is around 1~2 and and latter is around 5~6. Therefore, the first 100 points draws circle with radius 1~2 and last 100 points draws circle with radius 5~6.

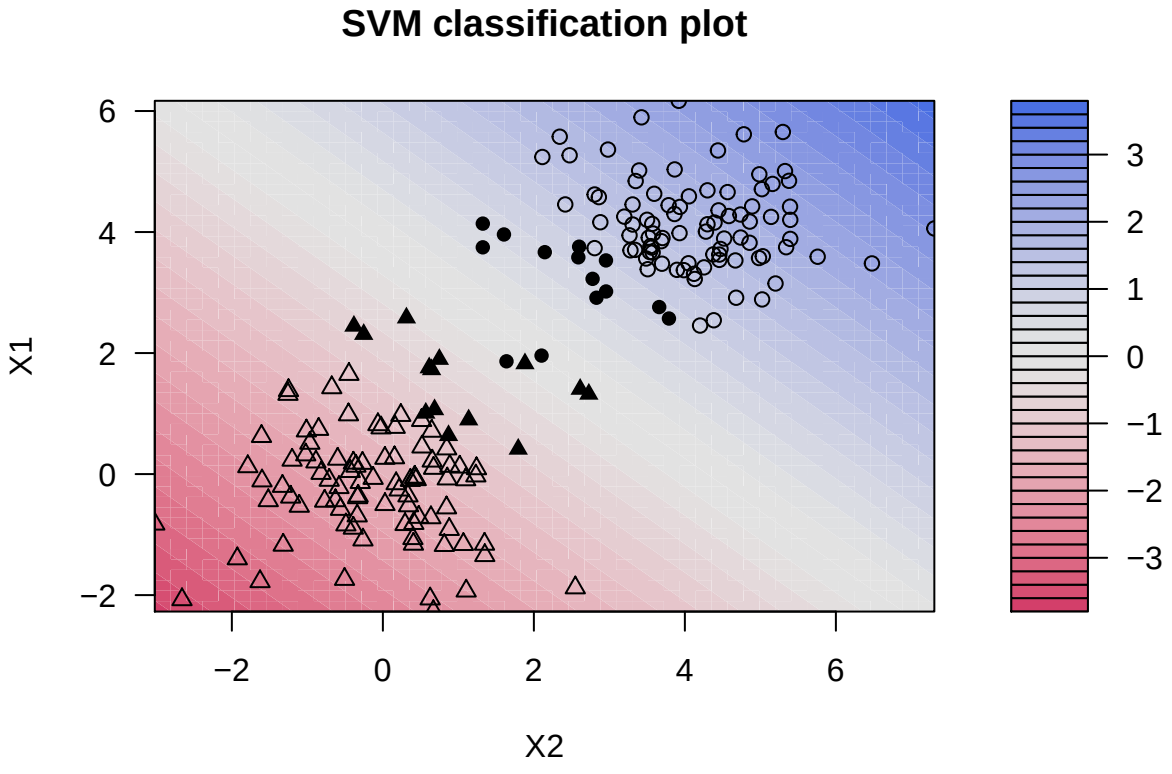
Support Vector Classifier

```
C_vector <- c(0.01, 0.1, 1, 10, 100, 1000, 10000)
dataset_list <- list(df1, df2)
for (df in dataset_list) {
  for (C in C_vector) {
    fit <- ksvm(y ~ X1 + X2, data = df, kernel = "vanilladot", C = C)
    plot(fit, data=df)

  }
}
```

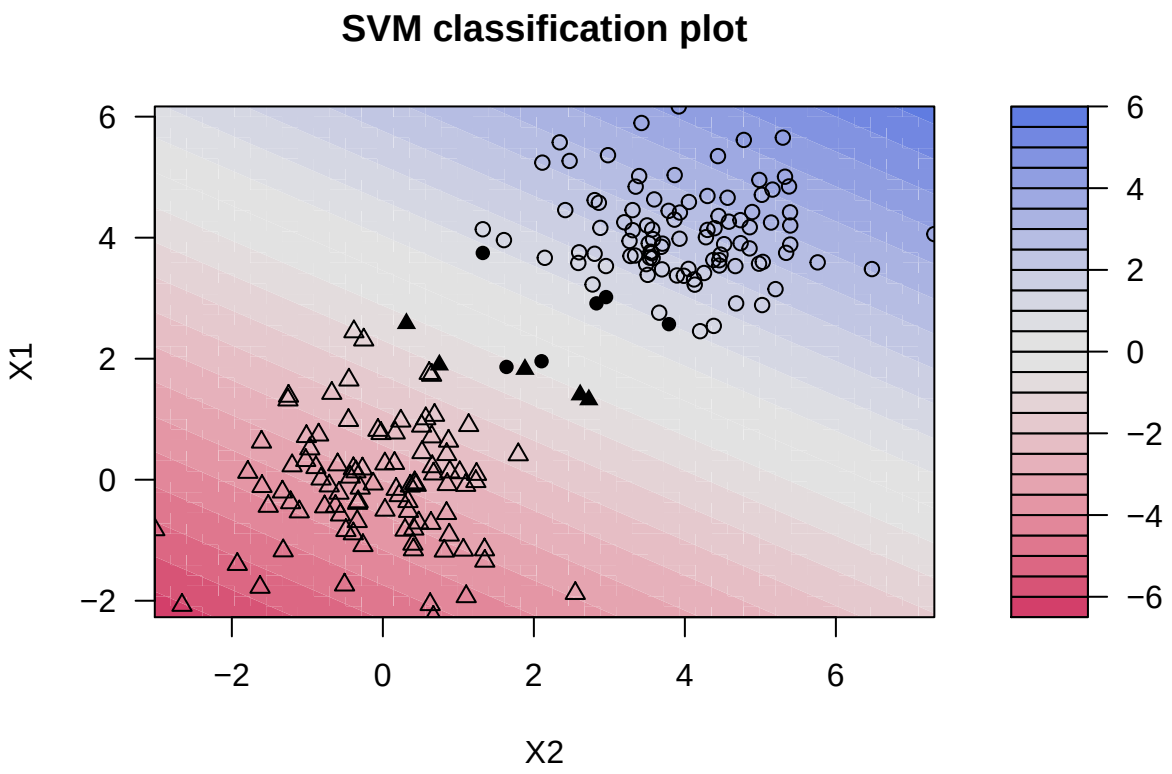
```
##  Setting default kernel parameters
```



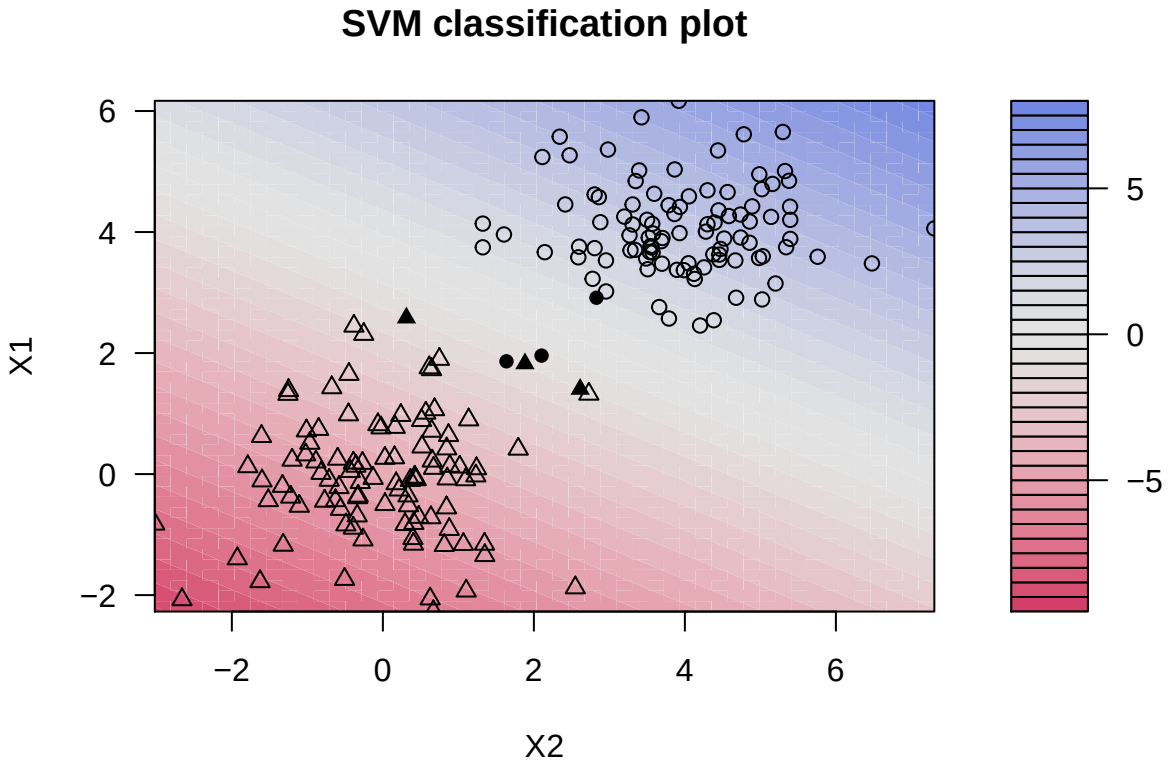

```
##  Setting default kernel parameters
```



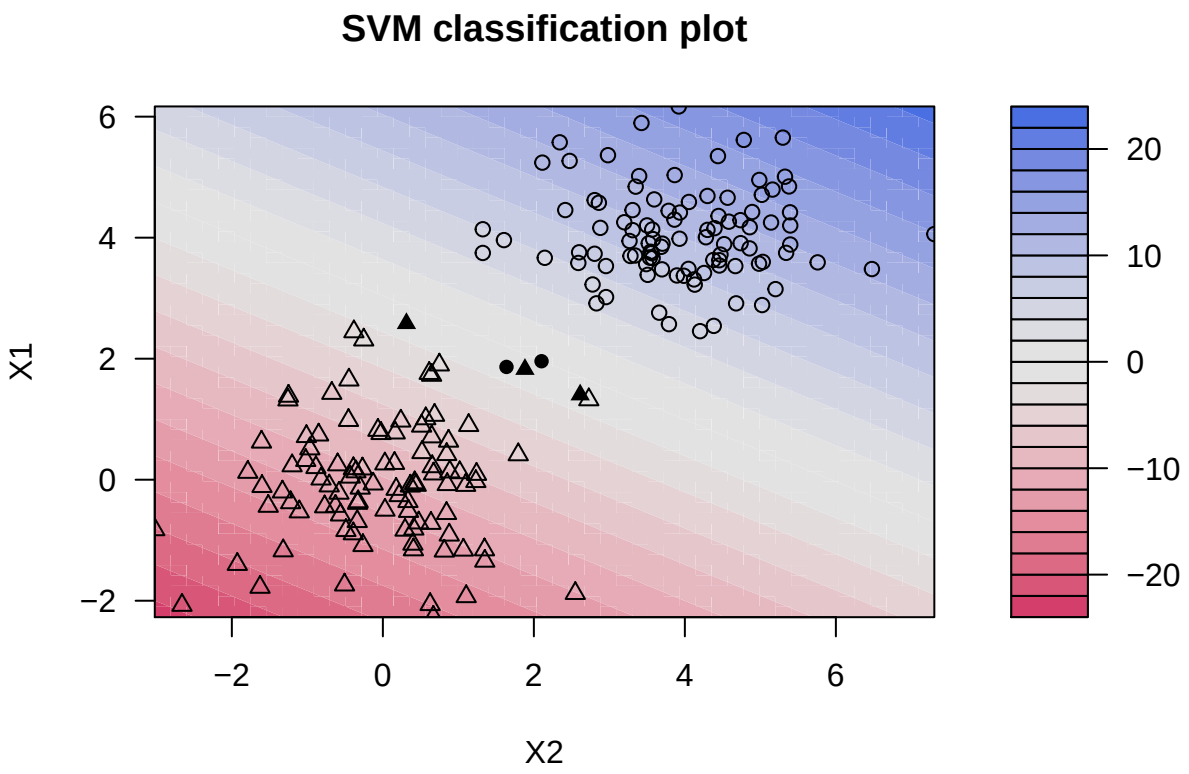

```
##  Setting default kernel parameters
```



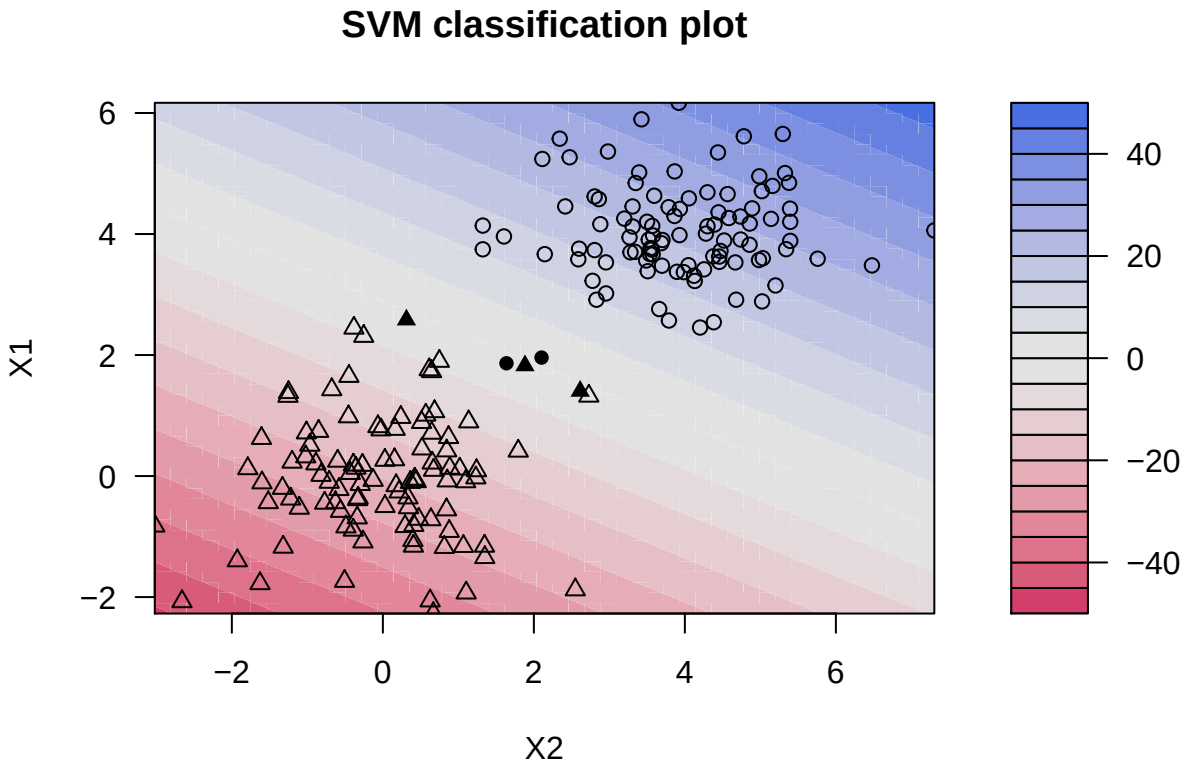

```
##  Setting default kernel parameters
```



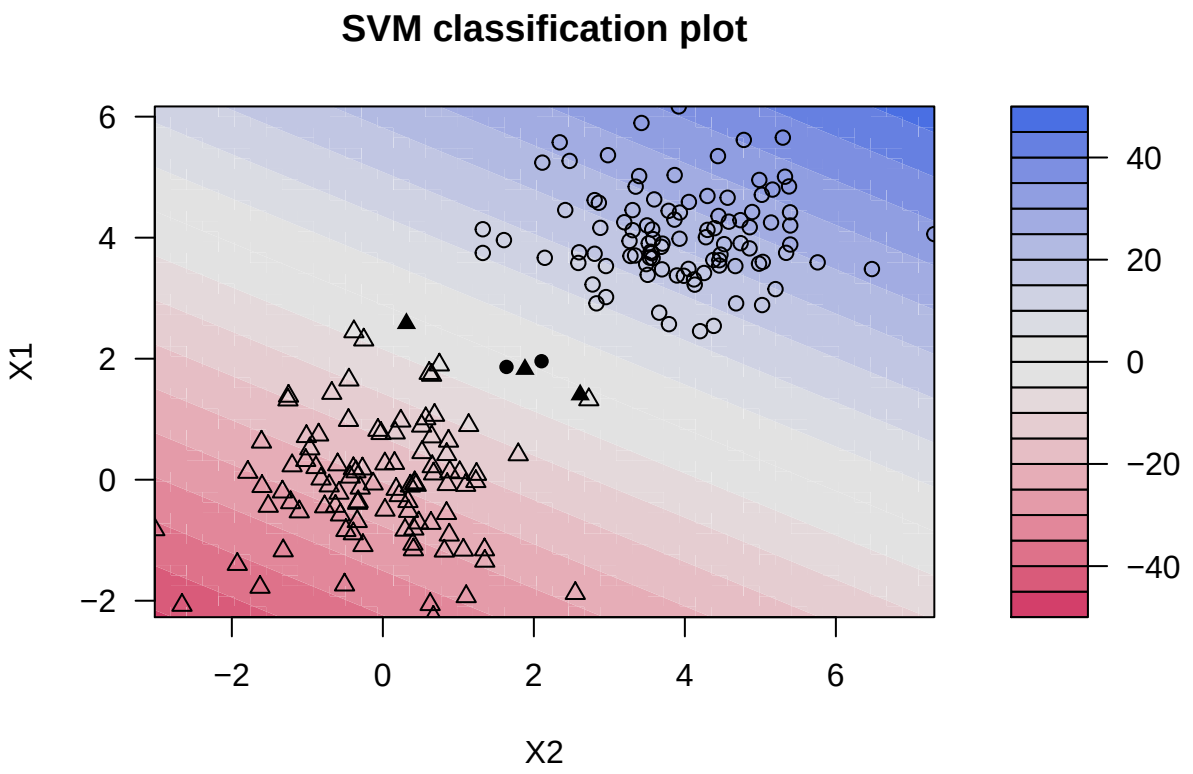

```
##  Setting default kernel parameters
```



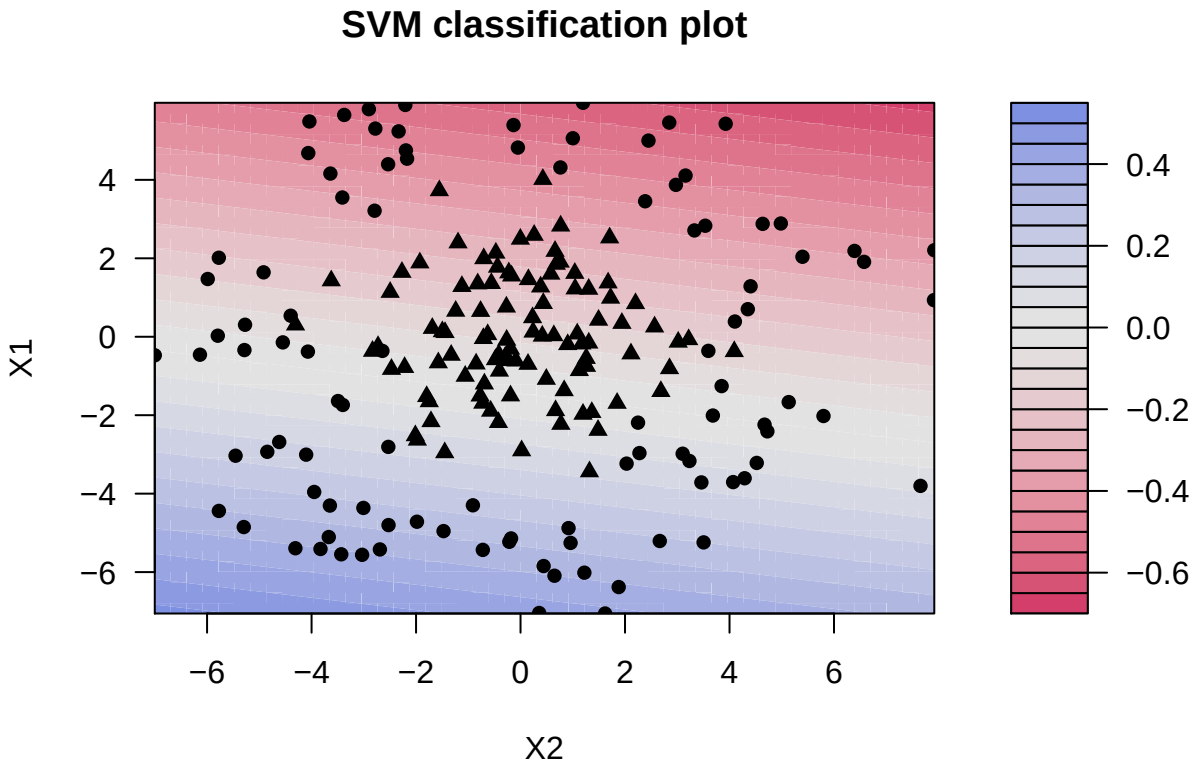

```
##  Setting default kernel parameters
```



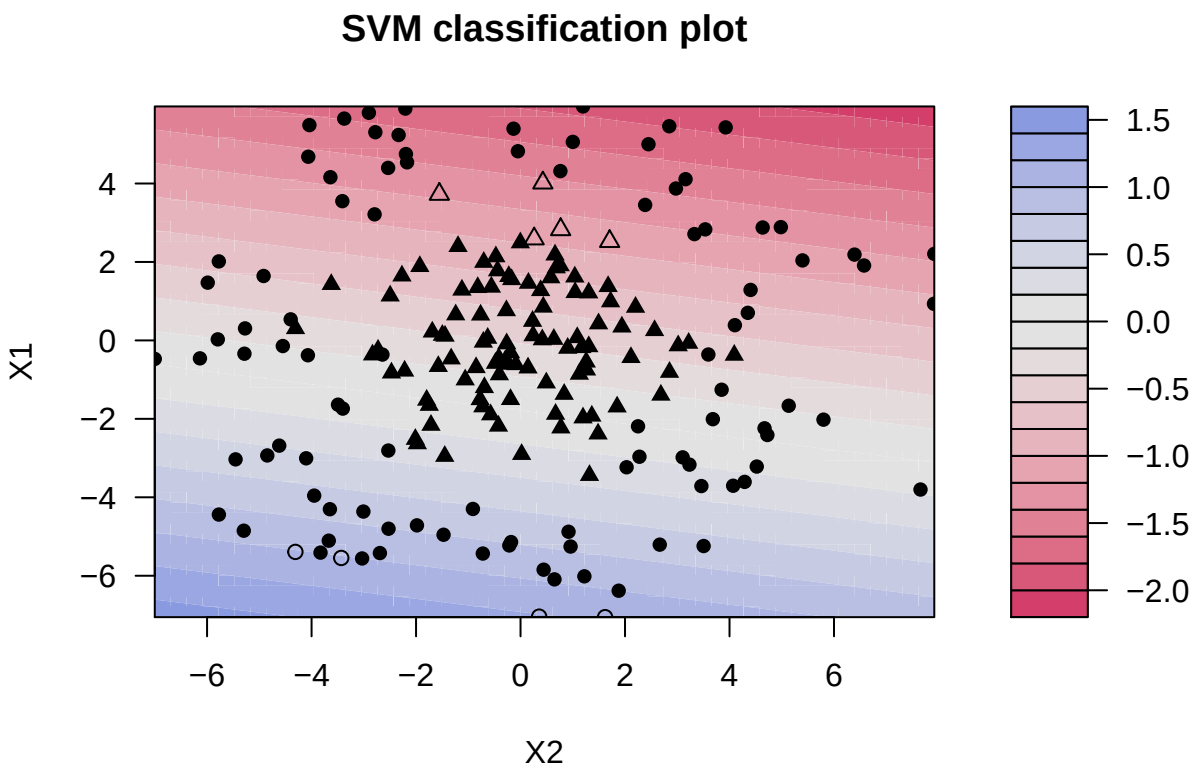

```
##  Setting default kernel parameters
```



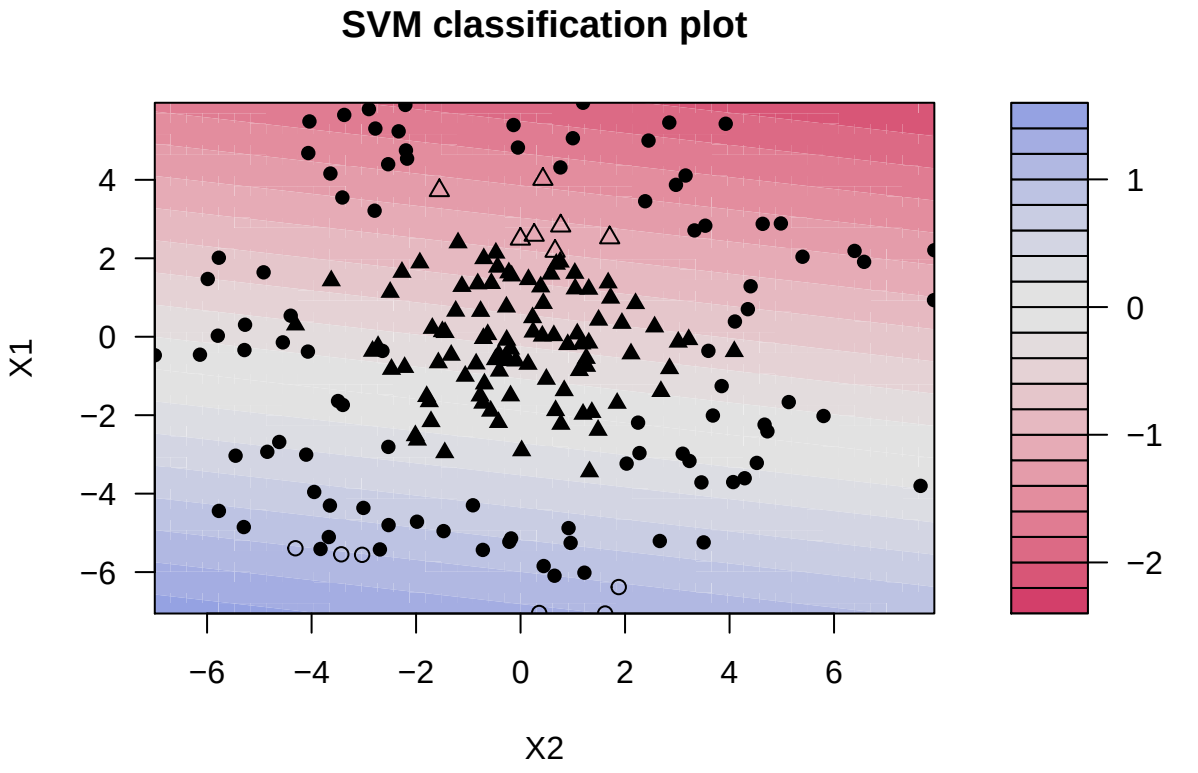

```
##  Setting default kernel parameters
```



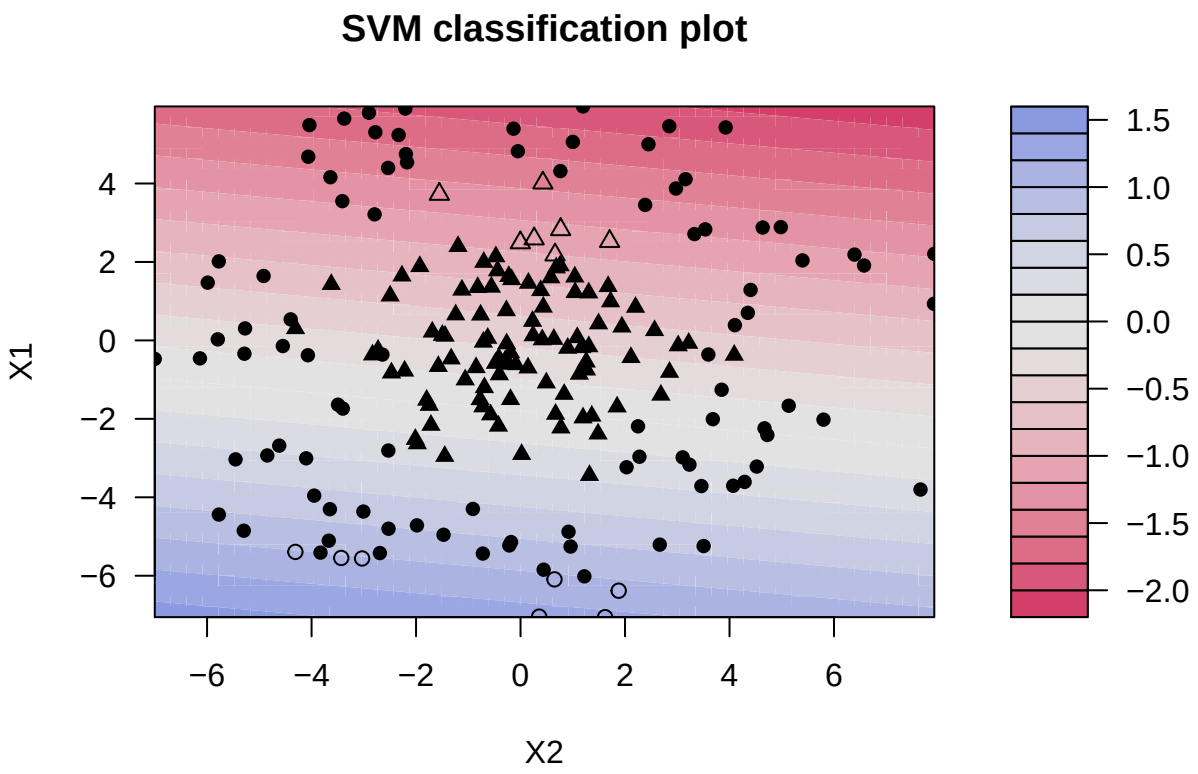

```
##  Setting default kernel parameters
```



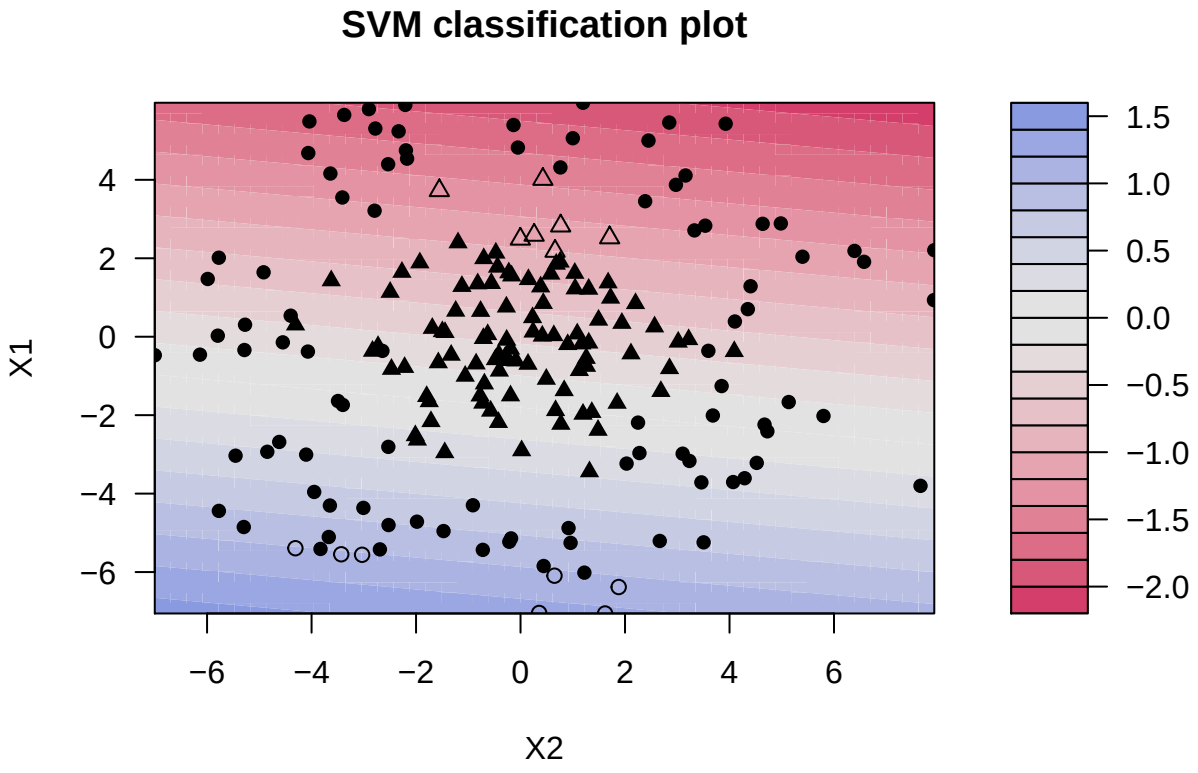

```
##  Setting default kernel parameters
```



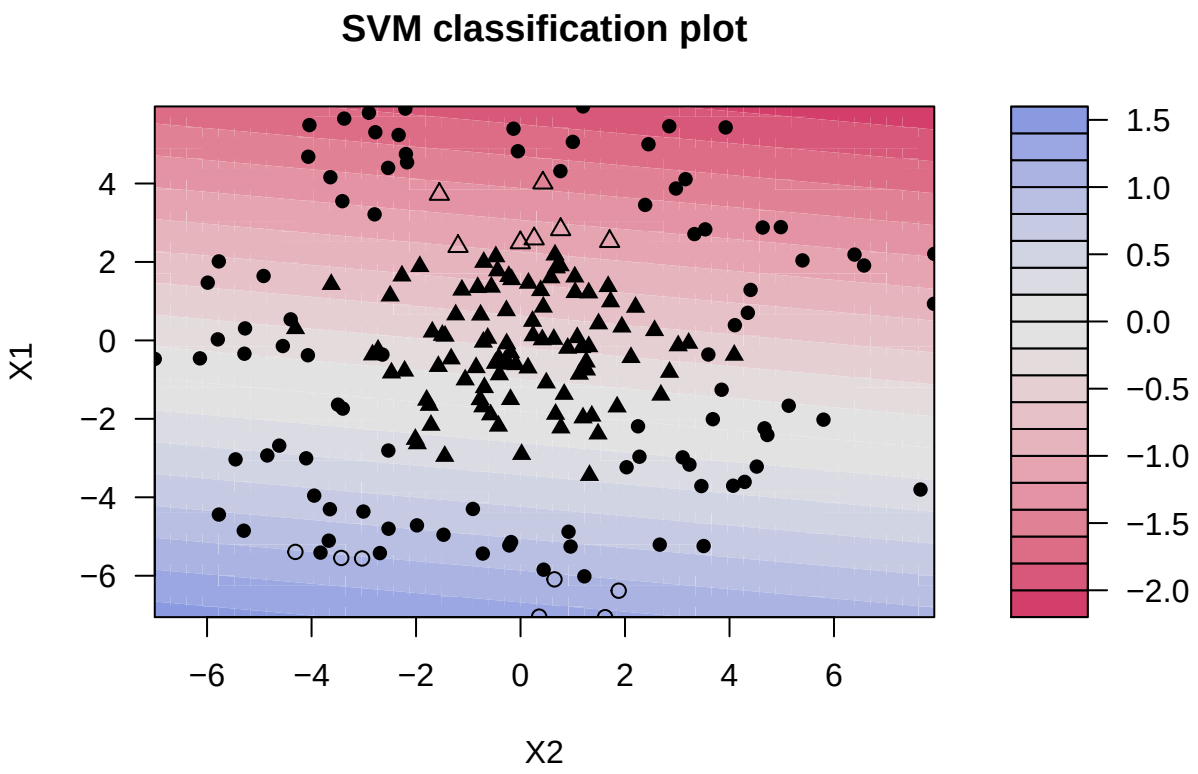

```
##  Setting default kernel parameters
```



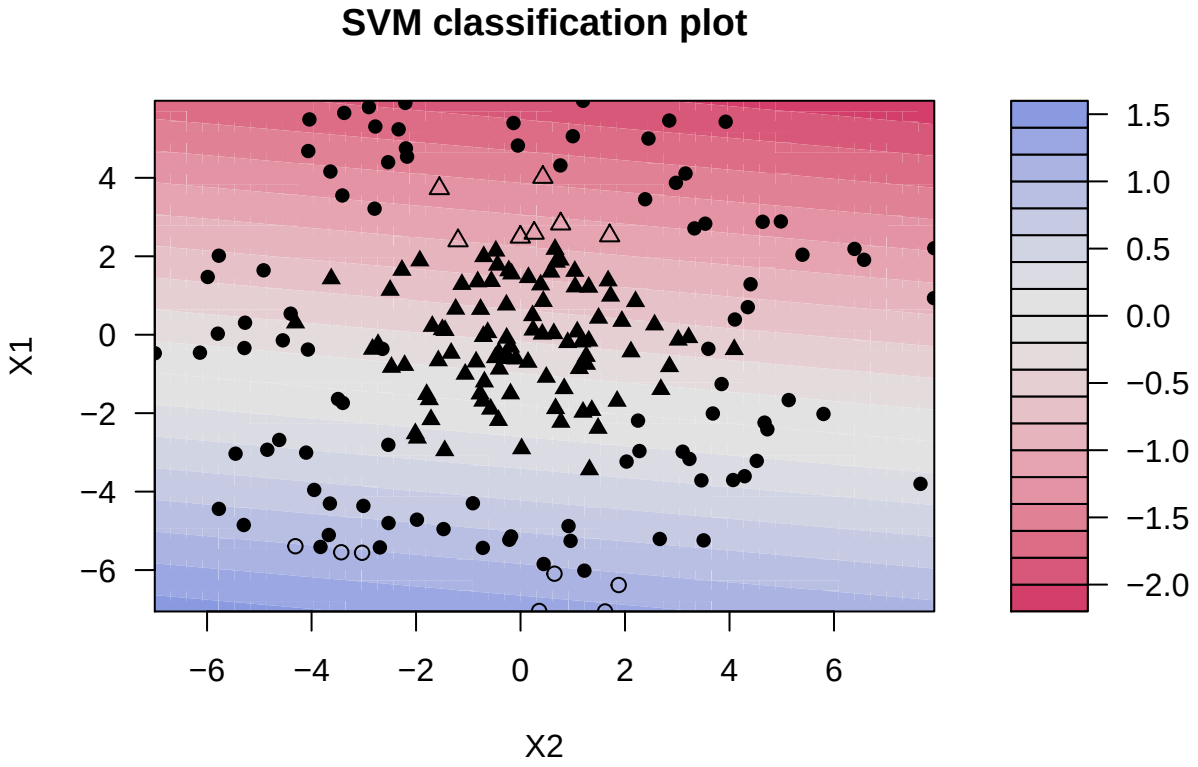

```
##  Setting default kernel parameters
```




```
##  Setting default kernel parameters
```




```
##  Setting default kernel parameters
```



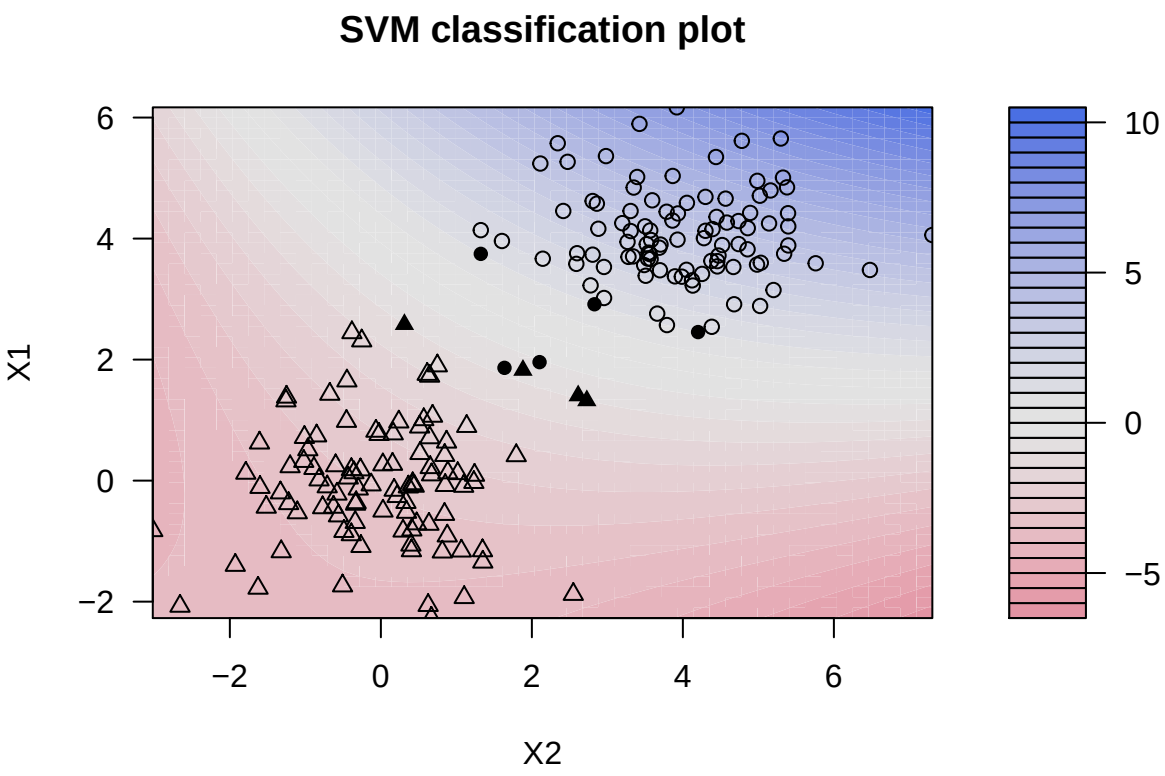
DF1 : As we increase C, we can see clear trends that the black dots are disappearing from the edges and when C = 10000, onlh the dots in the middle of the plots remained

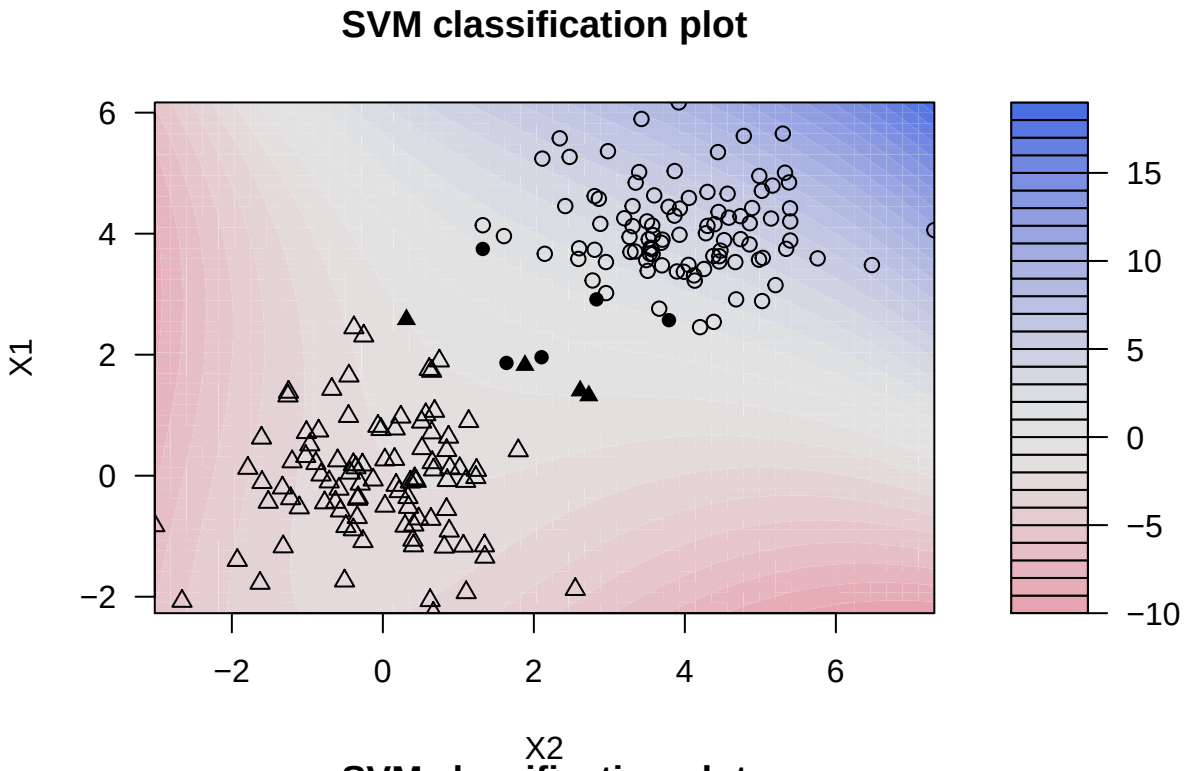
DF2 : As we increase C, we do not see clear trends of movement of the black dots. We can see that the black

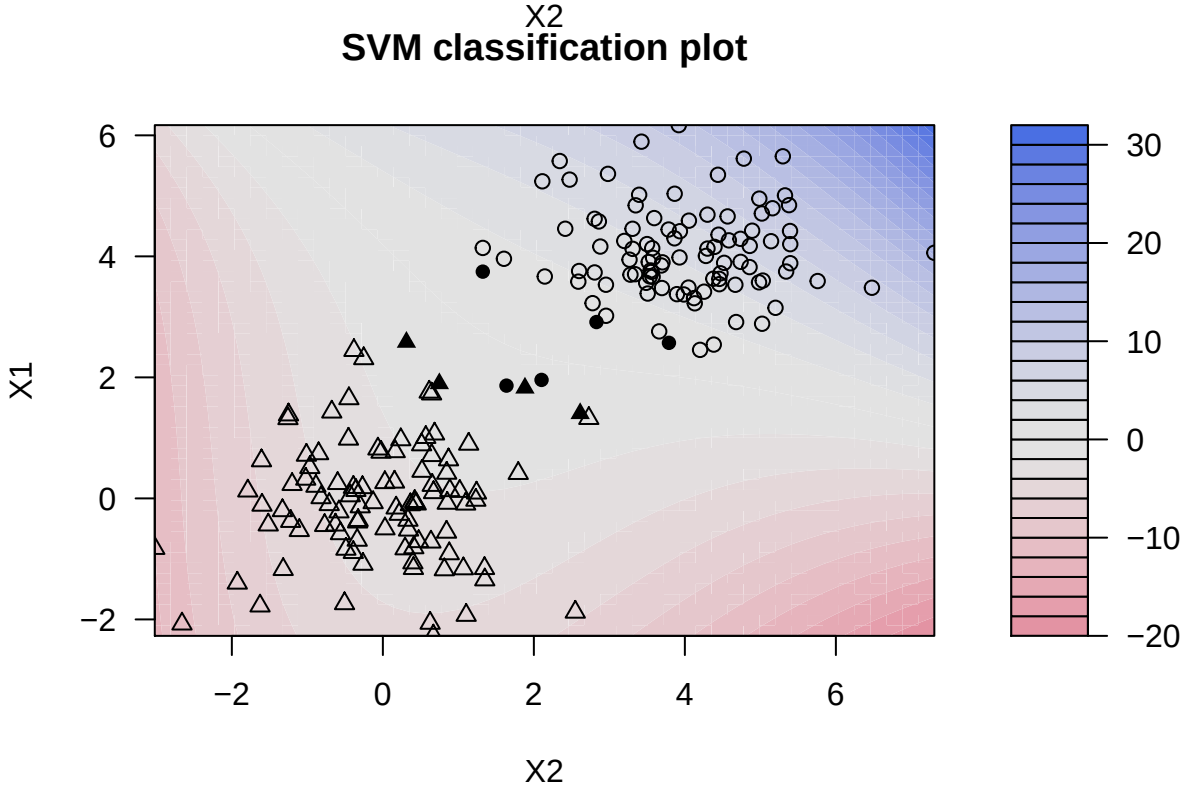
dots when C = 0.1 is equal to when C = 10000 unlike the results in DF1.

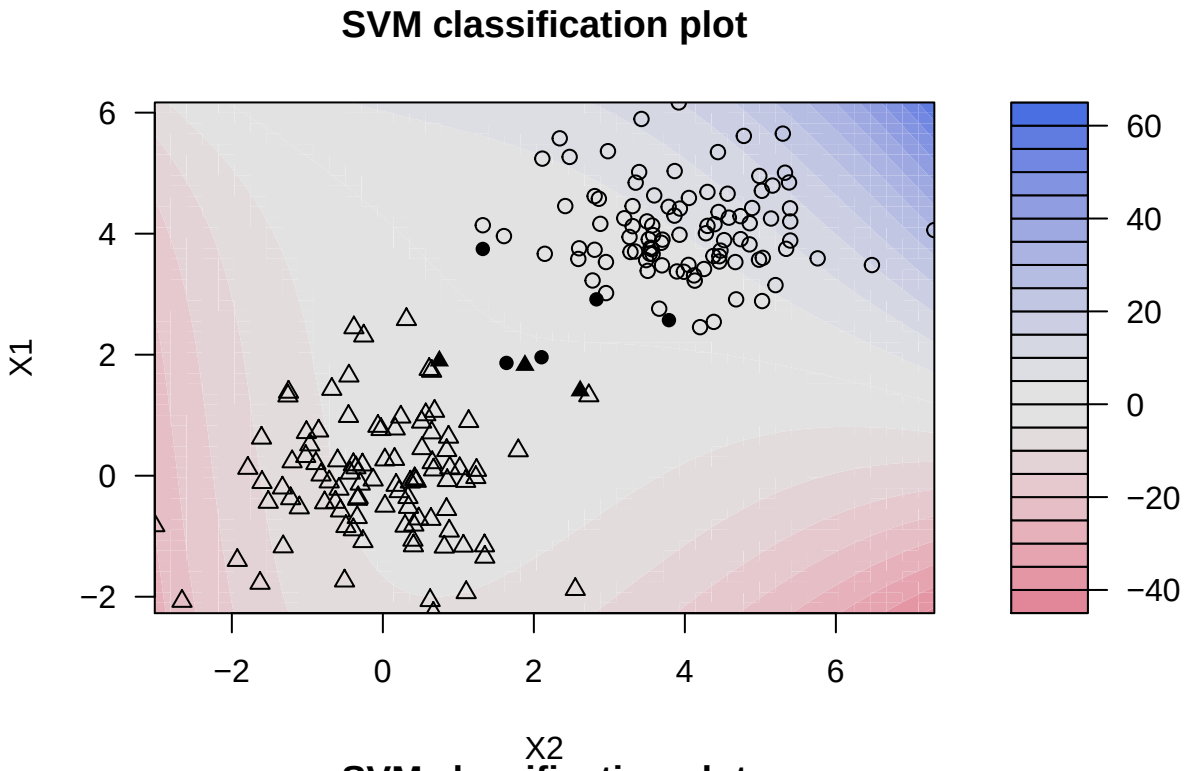
Support Vector Machine (SVM)

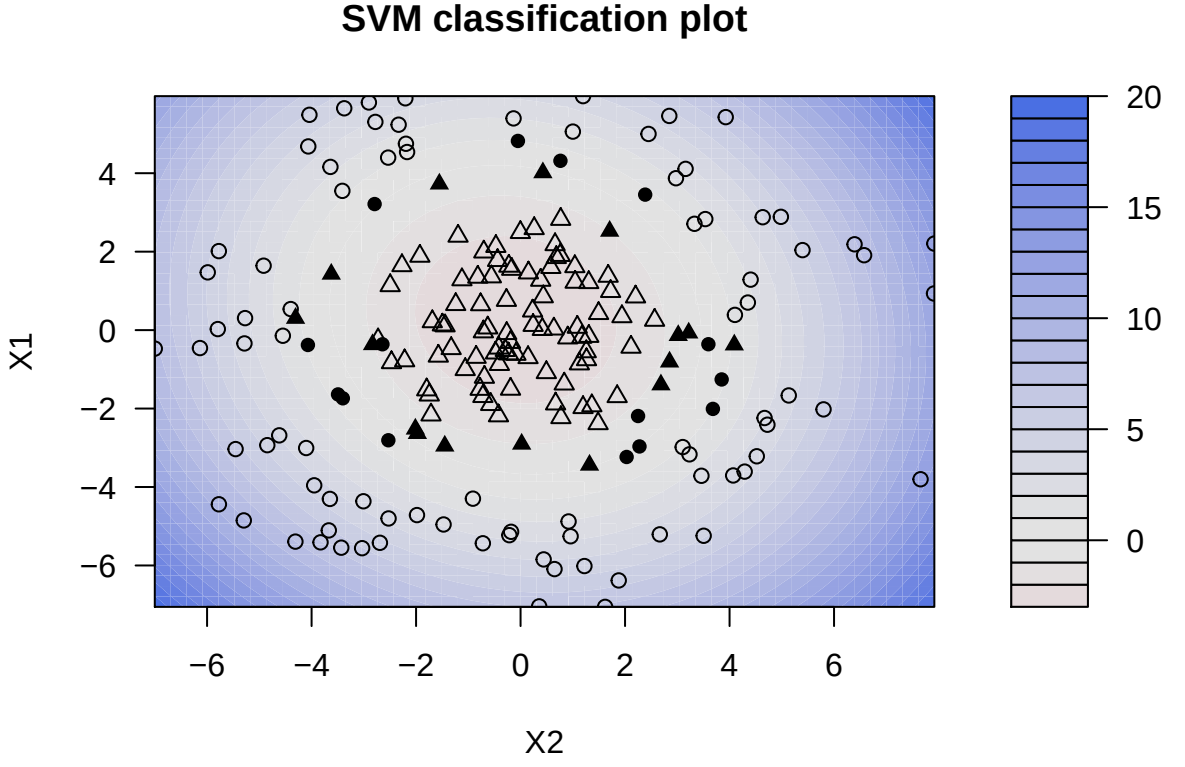
```
deg_vector <- 2:5
gam_vector <- c(0.01, 0.1, 1, 10, 100, 1000, 10000)



for (df in dataset_list) {
  for (deg in deg_vector) {
    fit <- ksvm(y~X1+X2, data = df, kernel = "polydot",
                kpar=list(degree=deg))
    plot(fit, data=df)
  }
}
```

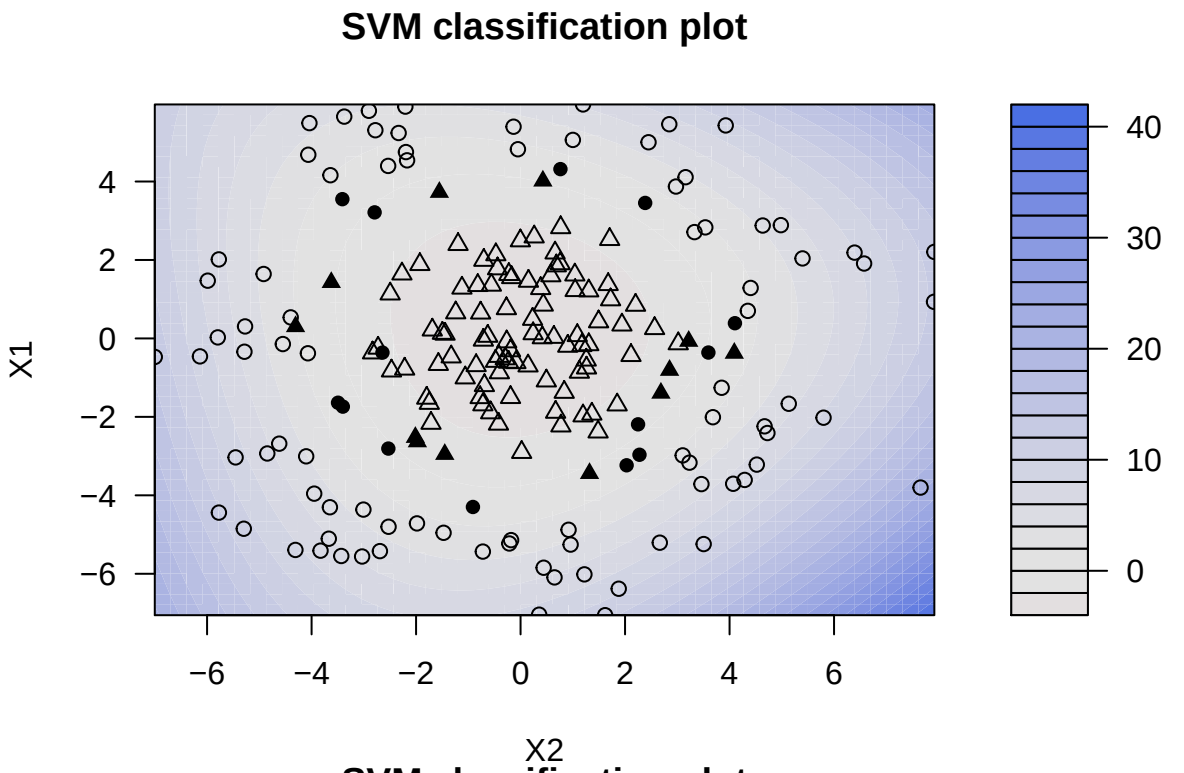


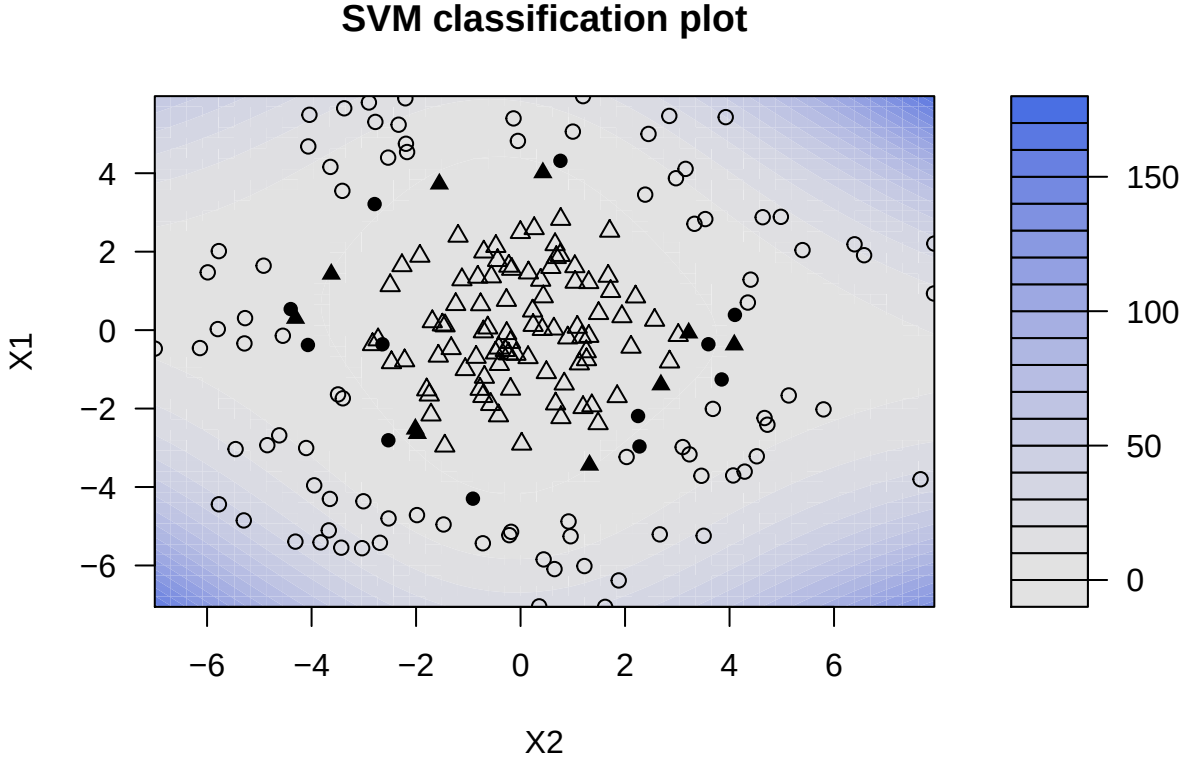

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
15
10
5
0
−5
−10

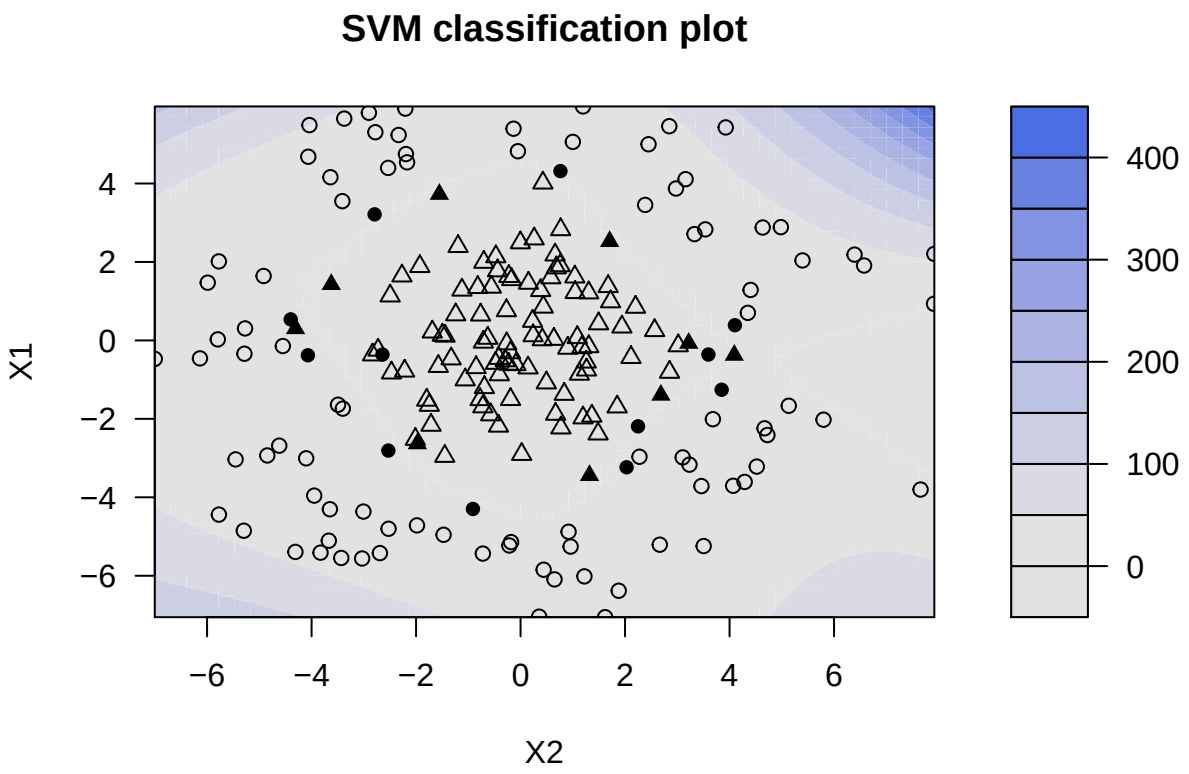

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
30
20
10
0
−10
−20


SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
60
40
20
0
−20
−40

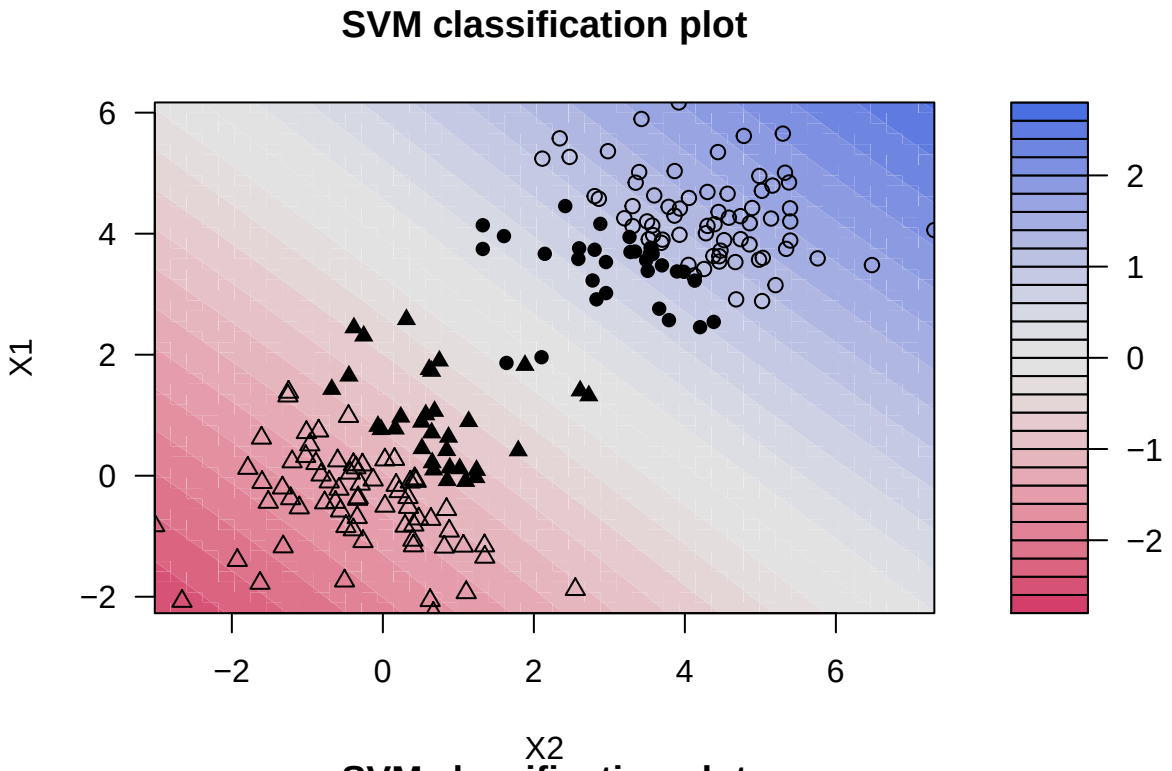

SVM classification plot
X1
X2
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
20
15
10
5
0

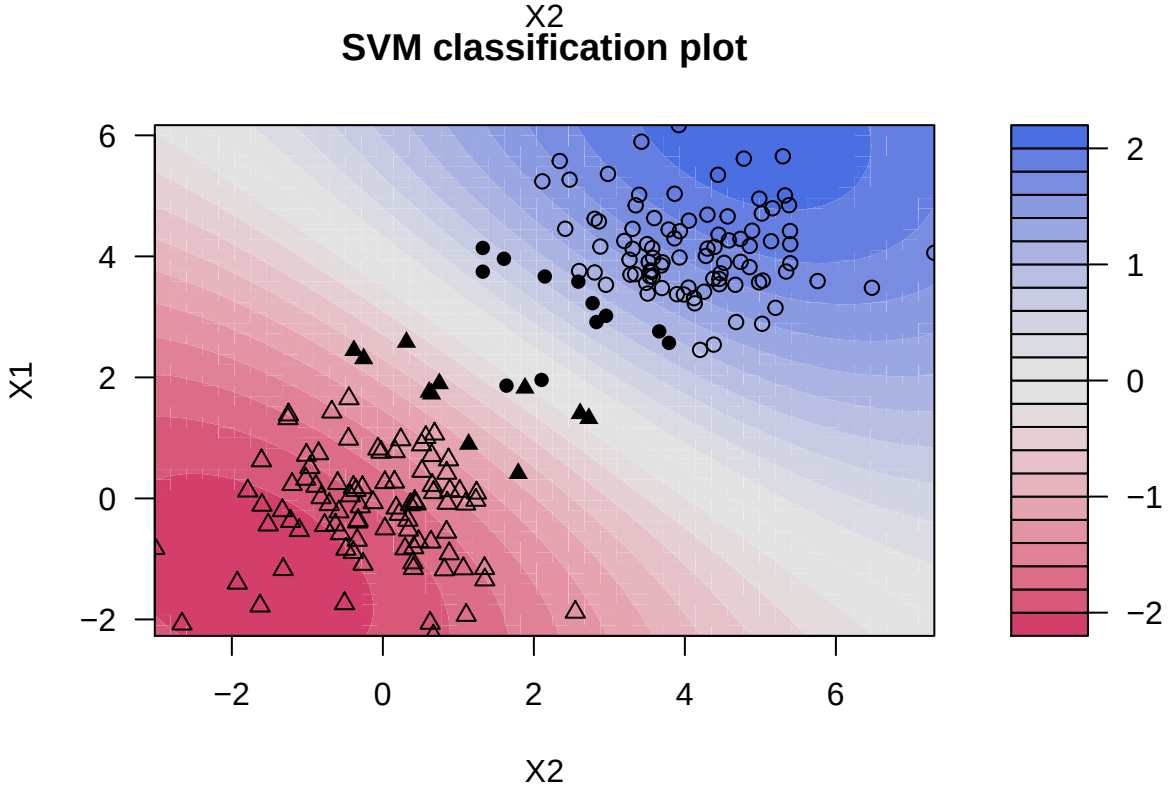

SVM classification plot
X1
X2
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
40
30
20
10
0

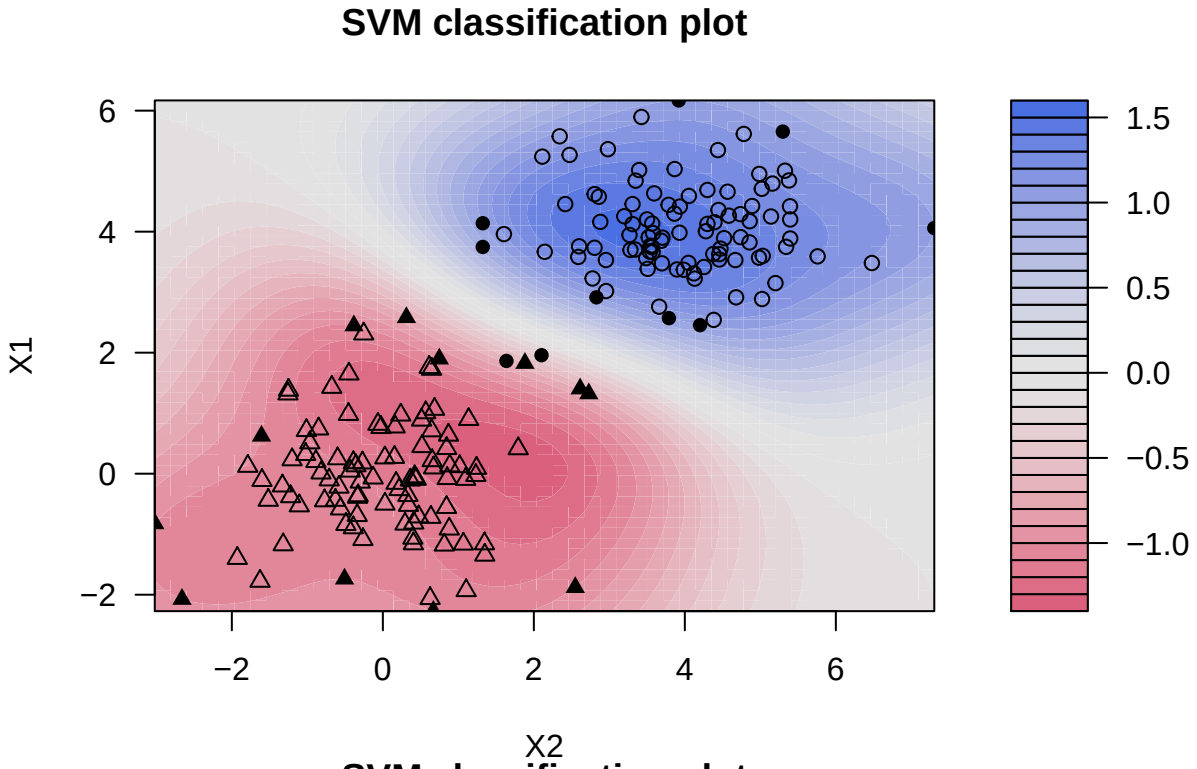

SVM classification plot
X1
X2
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
150
100
50
0

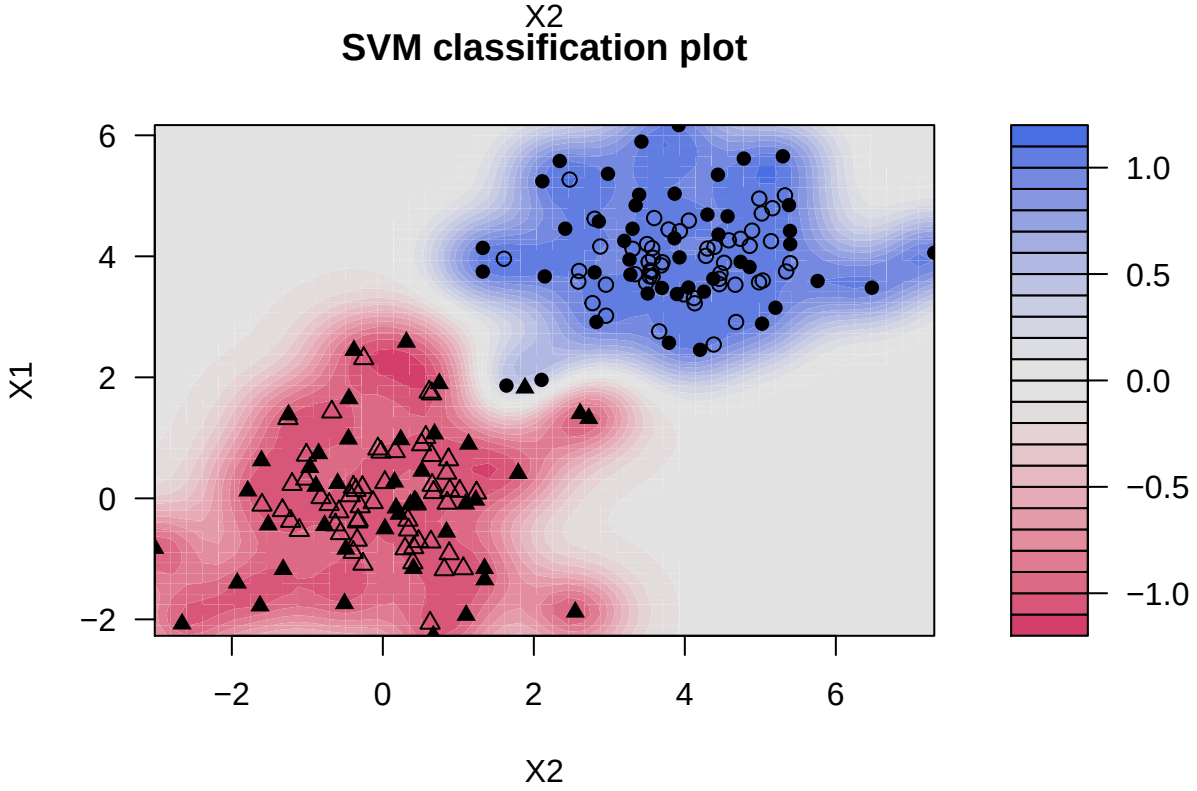


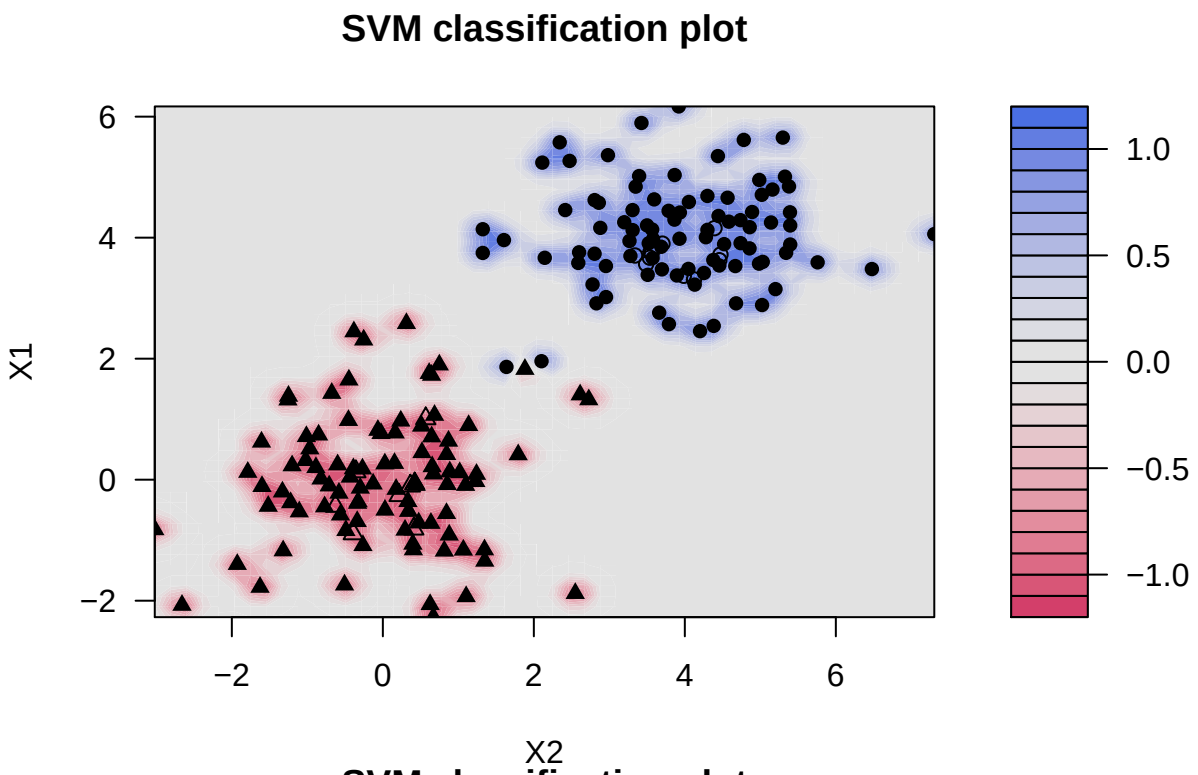

```
for (df in dataset_list) {
  for (gam in gam_vector) {
    fit <- ksvm(y~X1+X2, data = df, kernel = "rbfdot", kpar=list(sigma=gam))
    plot(fit, data=df)
  }
}
```

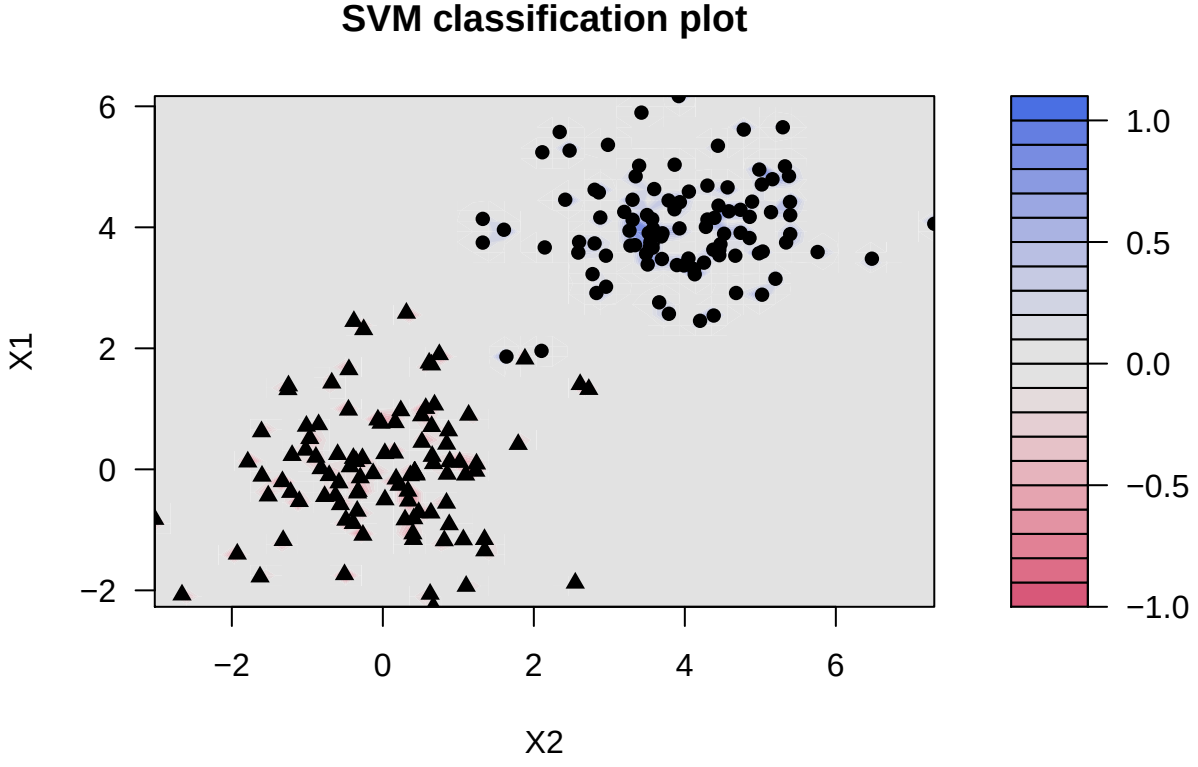

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
2
1
0
−1
−2

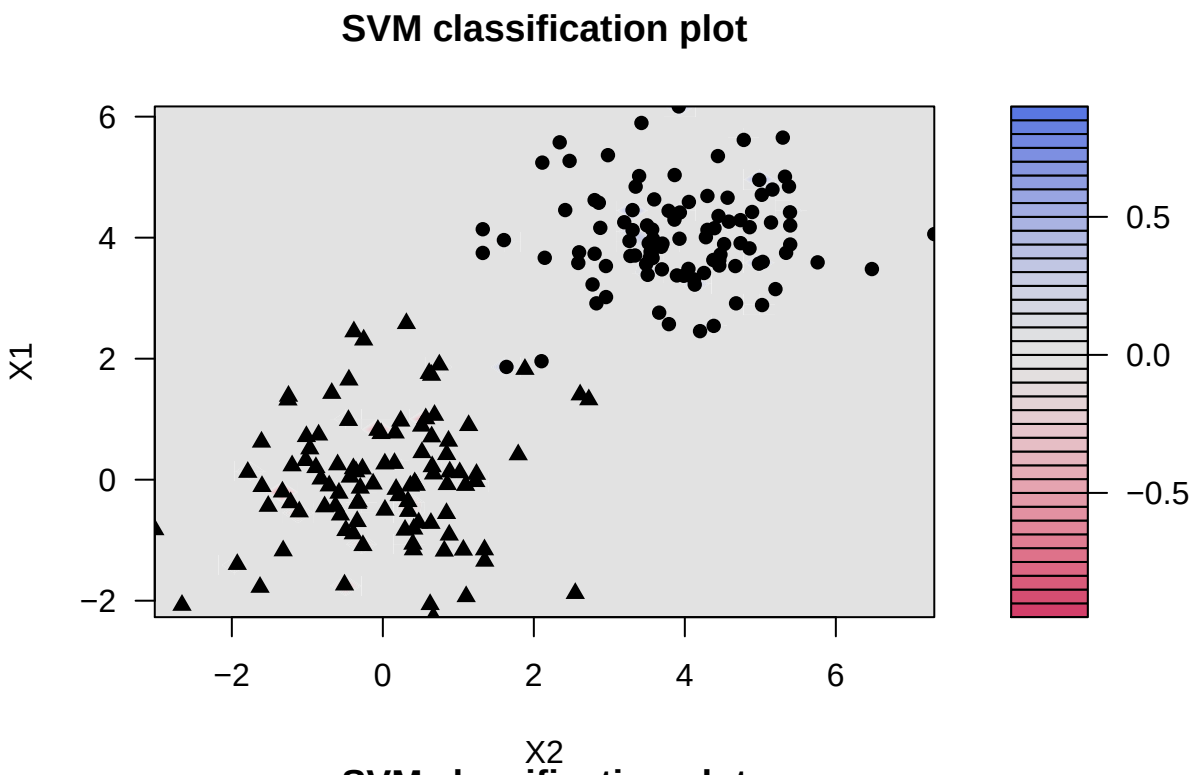

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
2
1
0
−1
−2

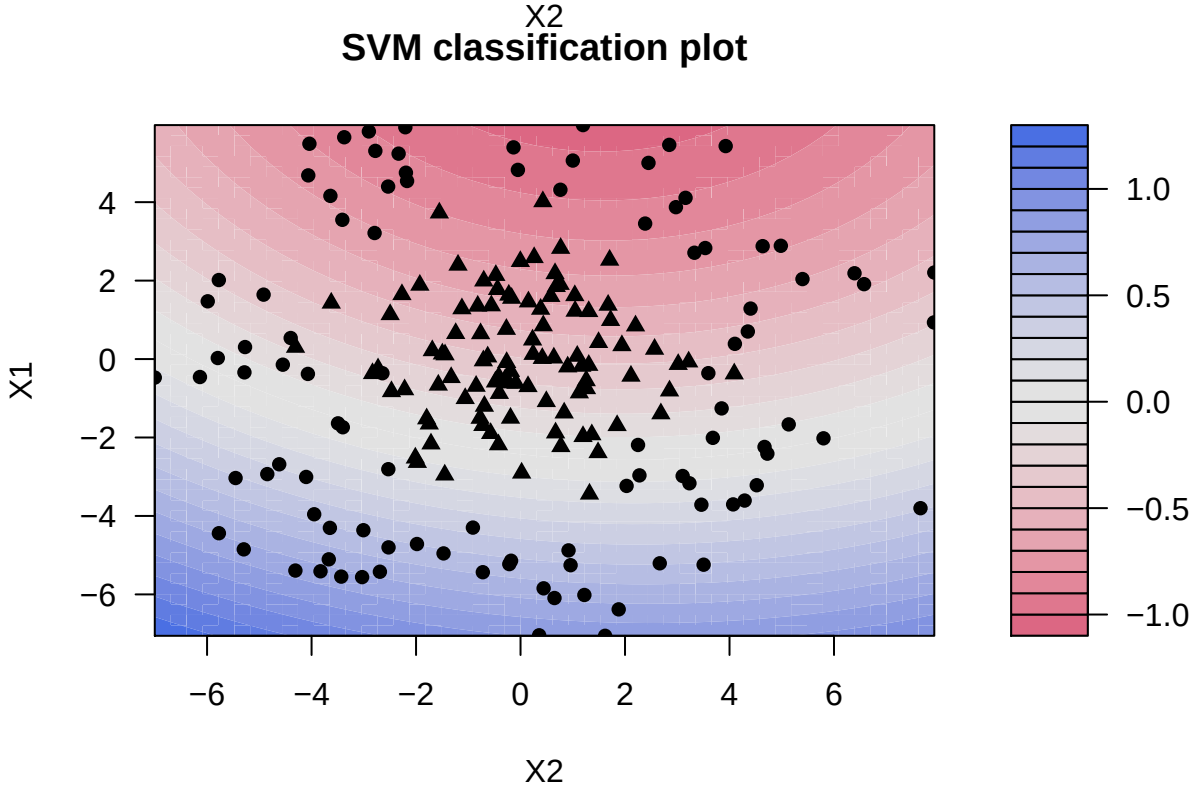

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
1.5
1.0
0.5
0.0
−0.5
−1.0

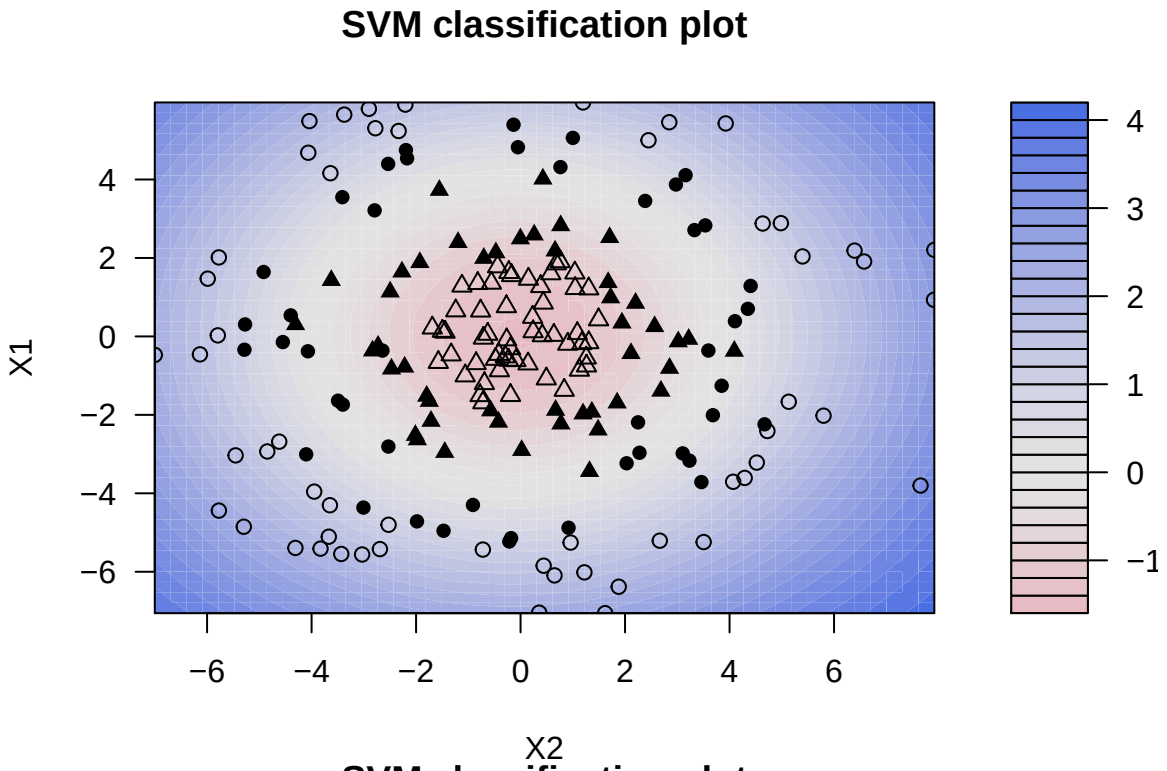

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
1.0
0.5
0.0
−0.5
−1.0

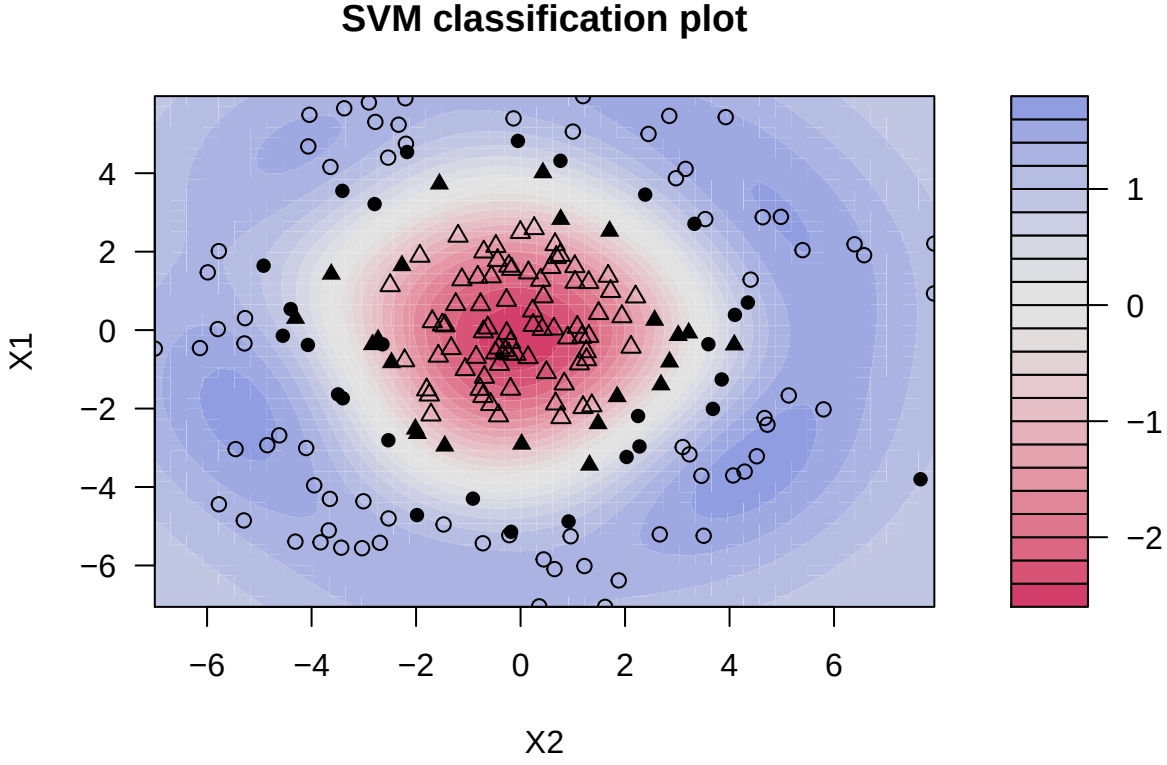

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
1.0
0.5
0.0
−0.5
−1.0

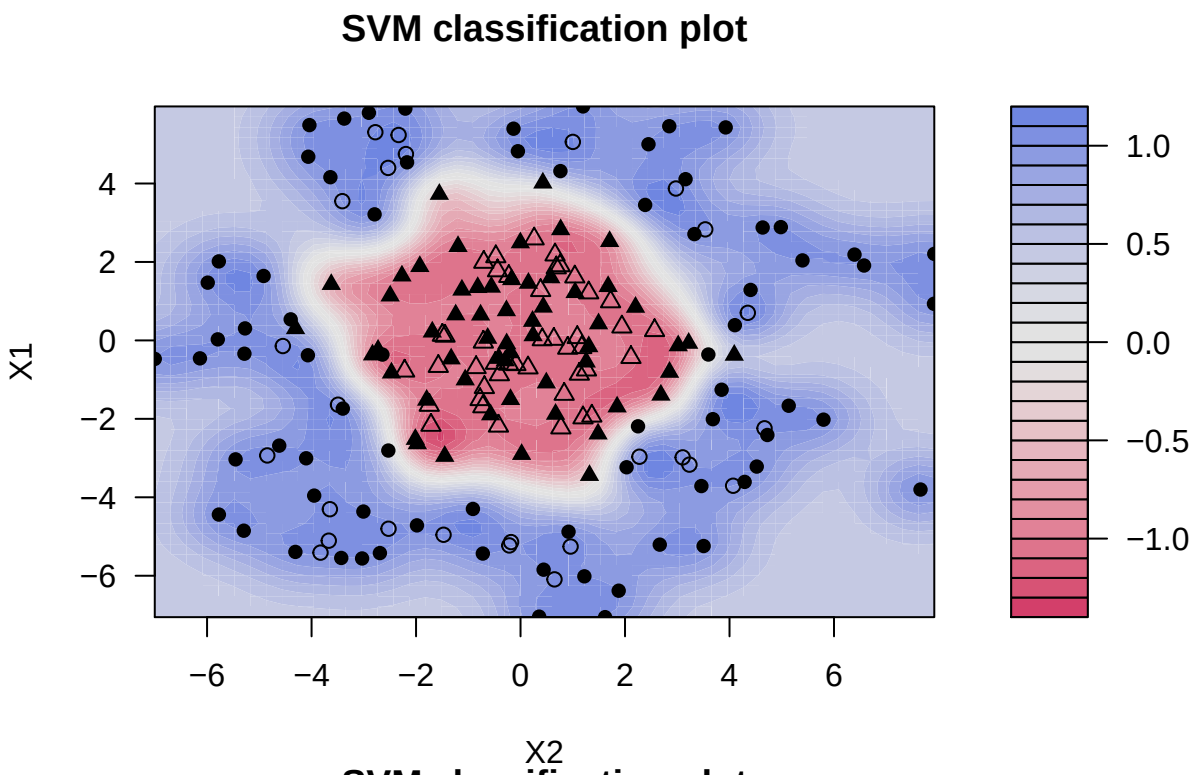

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
1.0
0.5
0.0
−0.5
−1.0

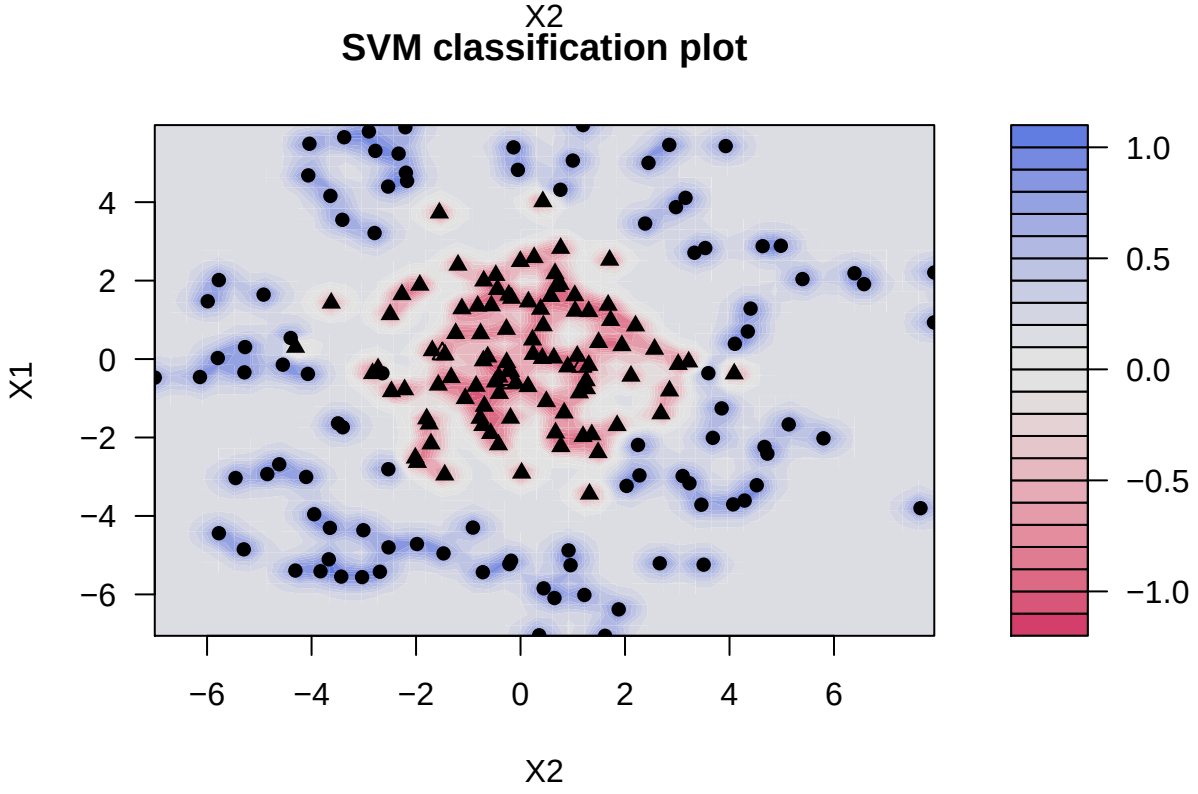

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
0.5
0.0
−0.5

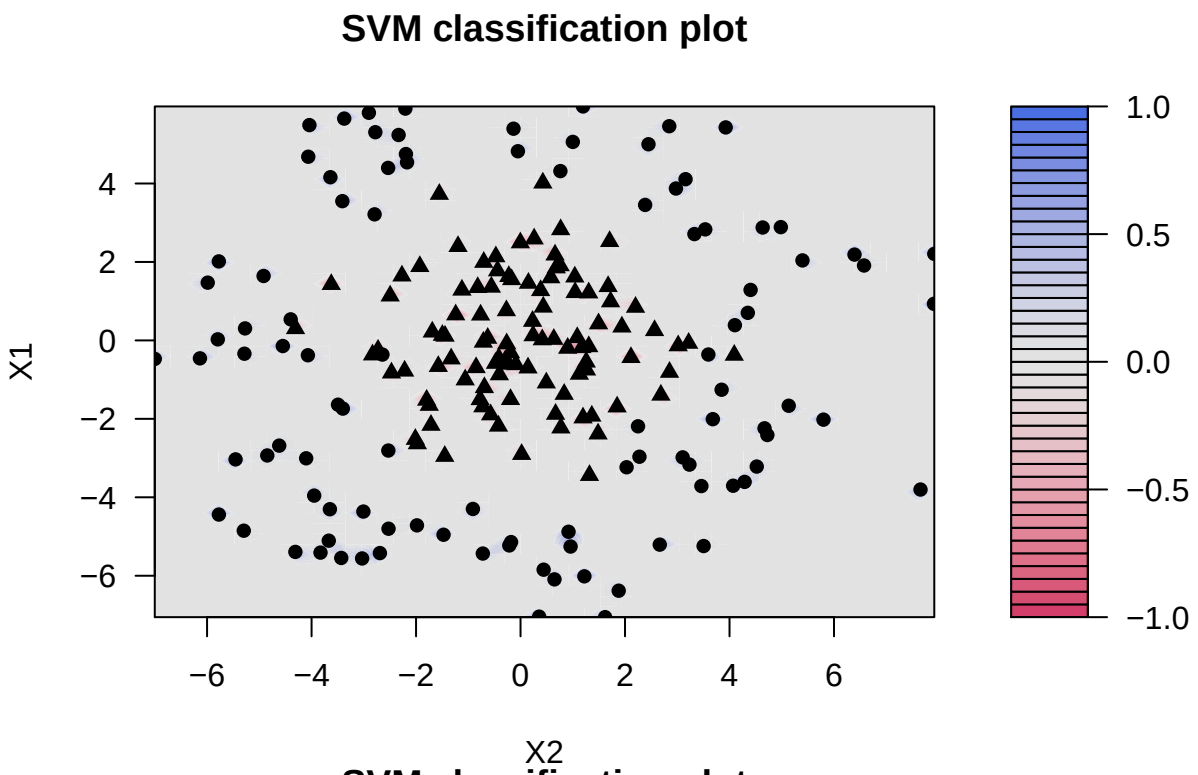

SVM classification plot
X1
X2
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
1.0
0.5
0.0
−0.5
−1.0

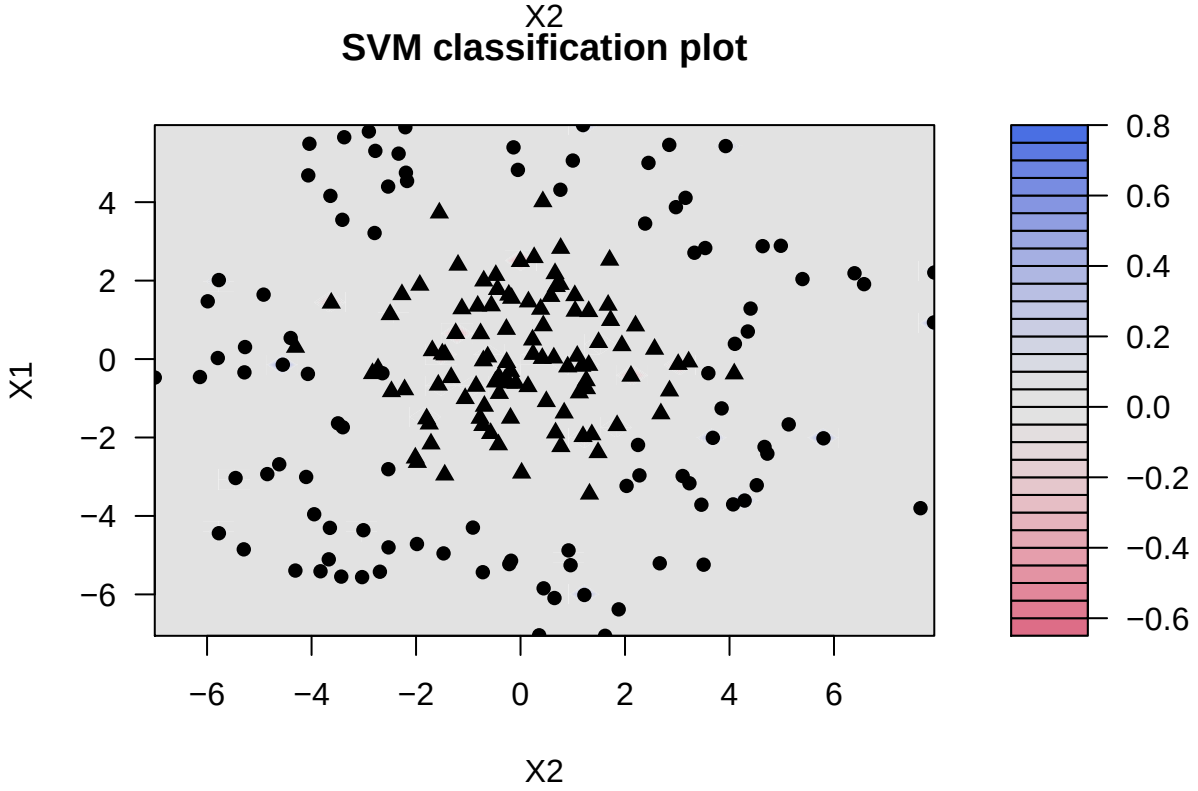

SVM classification plot
X1
X2
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
4
3
2
1
0
−1


SVM classification plot
X1
X2
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
1
0
−1
−2


SVM classification plot
X1
X2
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
1.0
0.5
0.0
−0.5
−1.0


SVM classification plot
X1
X2
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
1.0
0.5
0.0
−0.5
−1.0





The plot from polynomial kernel of degree = 2 doesn't shows clear distinction as we change degree from 2 to 5 for each dataframe.

In df1,

As we increase gamma, we can see that the area around the groups becomes more distinct. However, at some point if gamma is too large, the area which differentiated in blue and red are disappeared.

In df2,

As we increase gamma, we can see that the area around the groups becomes more distinct. However, at some point if gamma is too large, the area which differentiated in blue and red are disappeared.

In conclusion, I believe that it is significant to find the right gamma for the radial basis kernel method.

ROC curve

Train LDA using 70% of the data. Generate posterior class probabilities (for class 1 only) on the remaining 30%.

```
set.seed(100)
index <- createDataPartition(df2[,3], p = 0.7)

train <- df2[index$Resample1,]
test <- df2[-index$Resample1,]


ldafit <- lda( y ~., data = train)
ldafit
```

```
## Call:
## lda(y ~ ., data = train)
##
## Prior probabilities of groups:
##   0   1
## 0.5 0.5
##
## Group means:
##            X1          X2
## 0  0.05105363 -0.04861091
## 1 -0.93237441 -0.09481015
##
## Coefficients of linear discriminants:
##            LD1
## X1 -0.33881881
## X2  0.01122831
```

```
predicted <- predict(ldafit, test)

posteriors <- predicted$posterior

classes <- predicted$class



confusionMatrix(classes, test[,3])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 16 16
##          1 14 14
```

```
##
##                Accuracy : 0.5
##                  95% CI : (0.3681, 0.6319)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.5513
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : 0.8551
##
##             Sensitivity : 0.5333
##             Specificity : 0.4667
##          Pos Pred Value : 0.5000
##          Neg Pred Value : 0.5000
##              Prevalence : 0.5000
##          Detection Rate : 0.2667
##    Detection Prevalence : 0.5333
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

Plot the ROC curve. Add the 45 degree dotted line to the plot.

```
data(ROCR.simple)
head(cbind(ROCR.simple$predictions, ROCR.simple$labels), 5)
```

```
##           [,1] [,2]
## [1,] 0.6125478    1
## [2,] 0.3642710    1
## [3,] 0.4321361    0
## [4,] 0.1402911    0
## [5,] 0.3848959    0
```

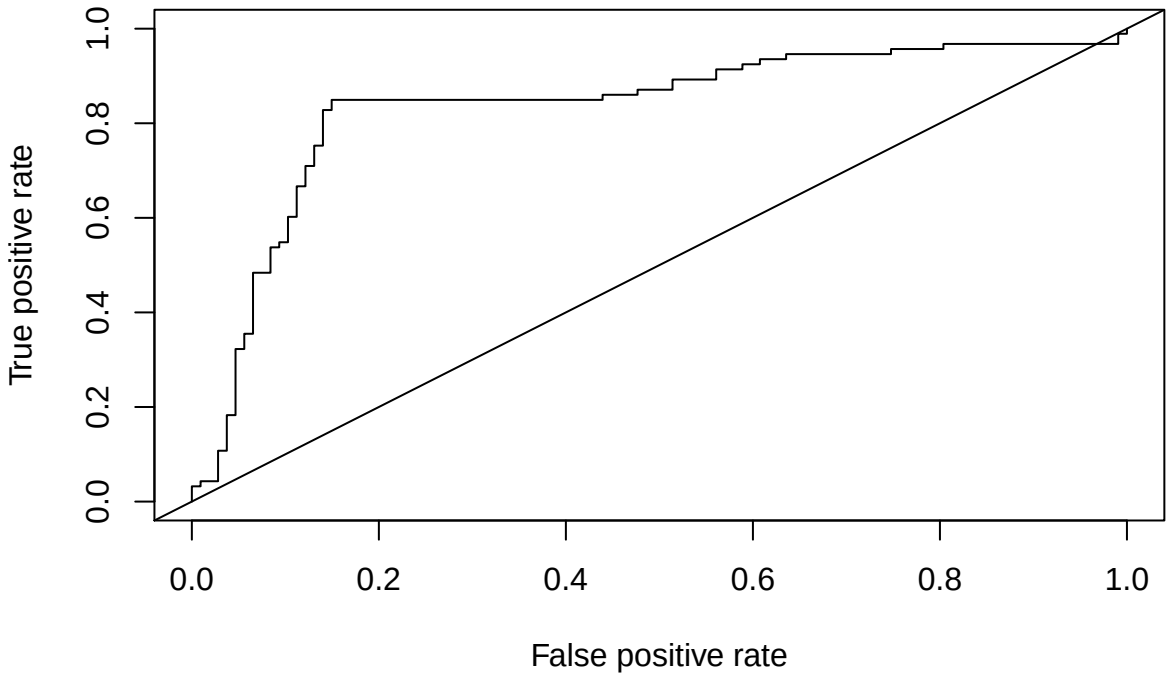
```
pred <- prediction(ROCR.simple$predictions,ROCR.simple$labels)
class(pred)
```

```
## [1] "prediction"
## attr(,"package")
## [1] "ROCR"
```

```
slotNames(pred)
```

```
##  [1] "predictions" "labels"      "cutoffs"     "fp"          "tp"
##  [6] "tn"          "fn"          "n.pos"       "n.neg"       "n.pos.pred"
## [11] "n.neg.pred"
```

```
roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
plot(roc.perf)
abline(a=0, b= 1)
```



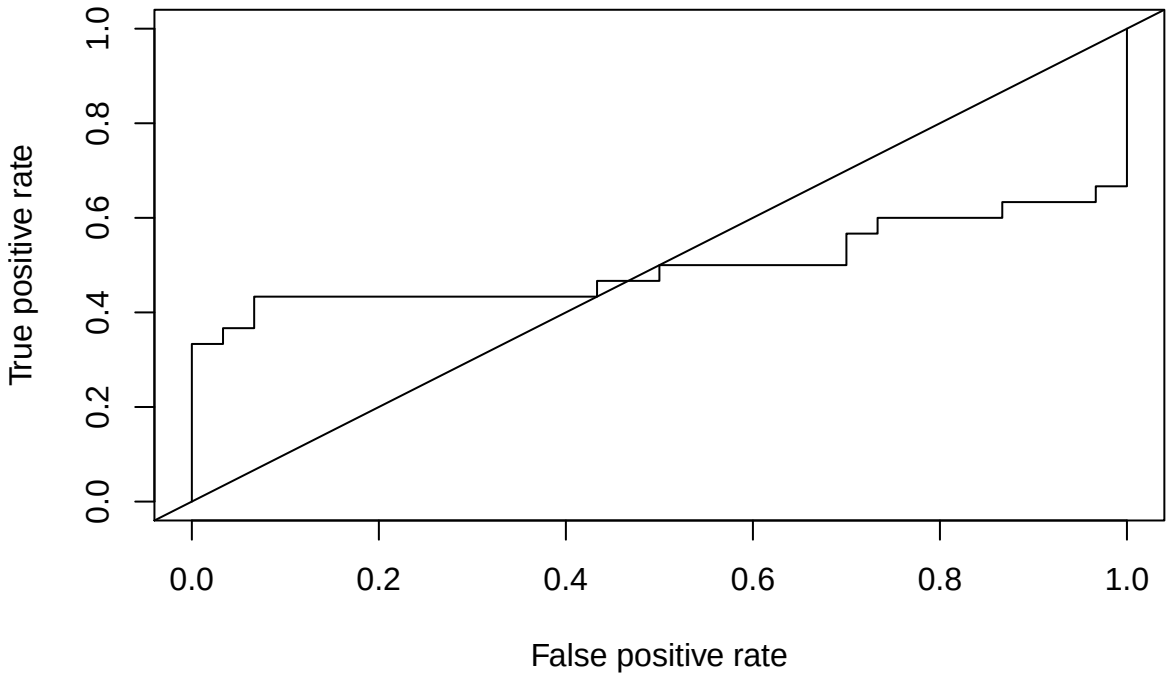

```
pred <- prediction(posteriors[,2], test[,3])
class(pred)
```

```
## [1] "prediction"
## attr(,"package")
## [1] "ROCR"
```

```
slotNames(pred)
```

```
##  [1] "predictions" "labels"      "cutoffs"     "fp"          "tp"
##  [6] "tn"          "fn"          "n.pos"       "n.neg"       "n.pos.pred"
## [11] "n.neg.pred"
```

```
roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
plot(roc.perf)
abline(a=0, b= 1)
```



Compute the area under ROC (AUC). You might find performance(. . . , measure="auc") useful

```
auc.perf = performance(pred, measure = "auc")
auc.perf@y.values
```

```
## [[1]]
## [1] 0.4977778
```

Does LDA perform much better than a random classifier for this particular dataset?

No

```
performance_metrics <- function(confusion) {

  tpr <- confusion[1,1] / (confusion[1,1] + confusion[2,1])
  tnr <- confusion[2,2] / (confusion[2,2] + confusion[1,2])
  fpr <- confusion[1,2] / (confusion[1,2] + confusion[2,2])
  fnr <- confusion[2,1] / (confusion[2,1] + confusion[1,1])
  specificity <- tnr
  sensitivity <- tpr

  return(list(tpr = tpr, tnr = tnr, fpr = fpr, fnr = fnr, specificity = specificity, sensitivity = sens

}
```