Jin Kweon - 3032235207 - lab 11

Jin Kweon

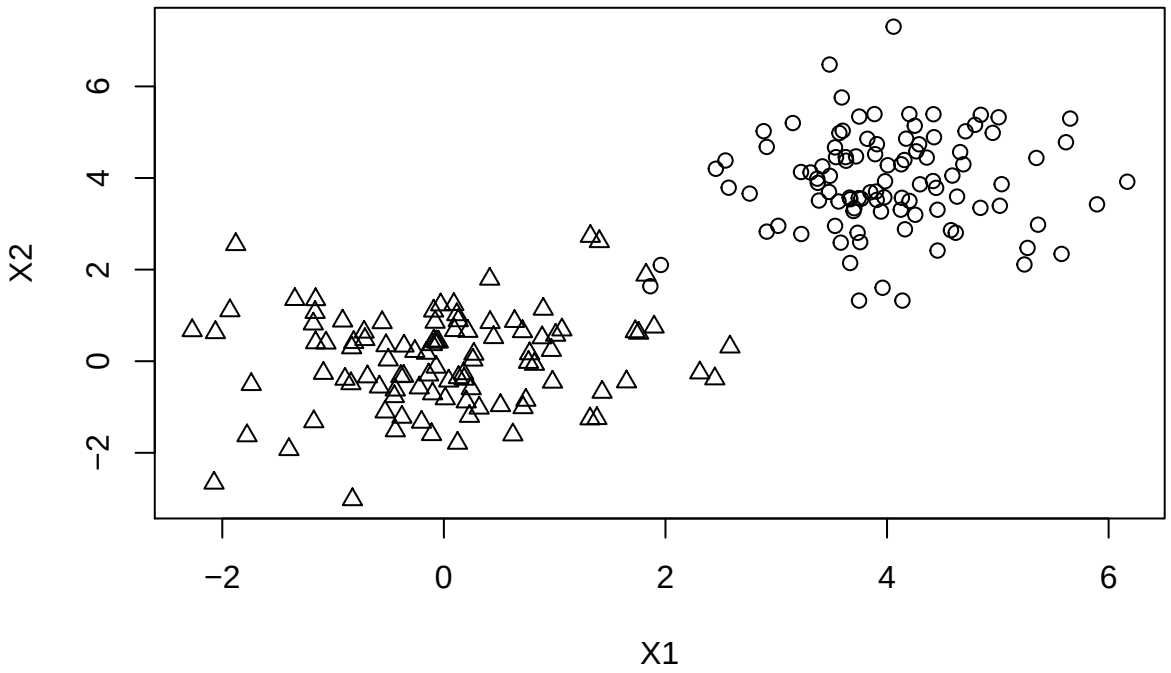
11/10/2017

Introduction

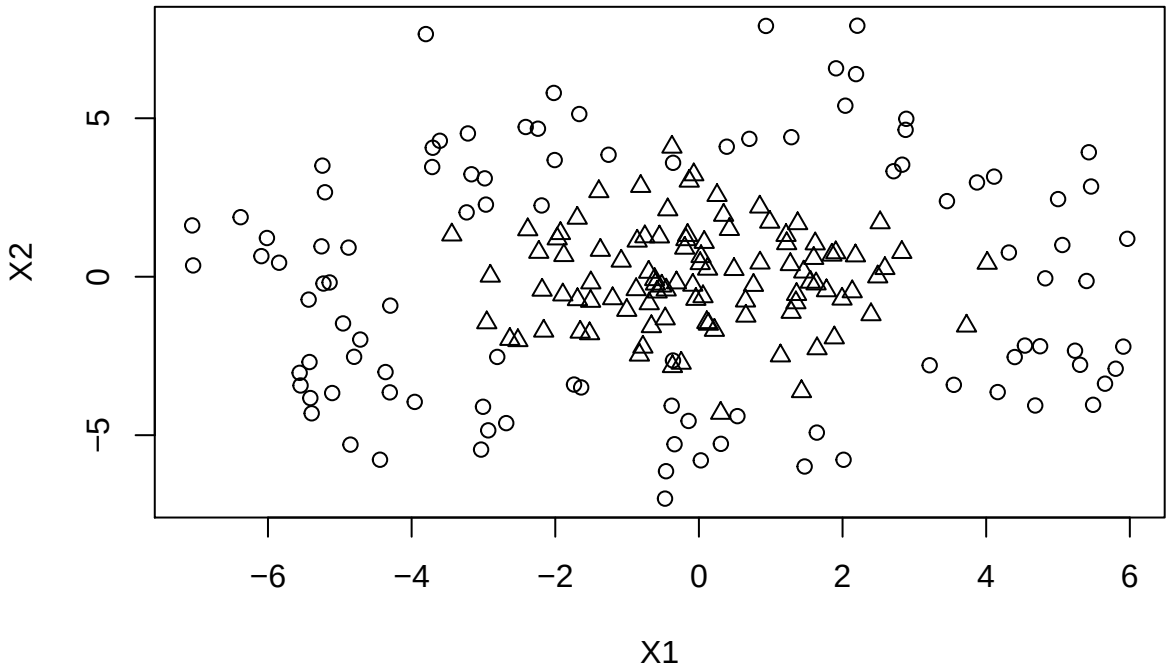
Q. pchs[0] shows numeric(0), but how does pchs[y] works fine?? Is it because y is a factor? ==> yes!!!

Q. why did you use with function, instead of using df1$X1 something like this??

```
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])) #first hundread is mean == 0...
```




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



Comment:

First pic: They are well-separated as the means of the first 100s and second 100s are quite different.

Second pic: cosine and sine noises are too overlapped as the domain is [0, 2π] are the same, and remember their range is [-1, 1] and thus uniform[1, 2] and uniform[5, 6] will be quite different. That is why the first hundred gets into the smaller circle, and the second hundred goes to the bigger circle. And, the r will be a proxy for radius.

Support vector classifier

https://www.youtube.com/watch?v=1NxnPkZM9bc

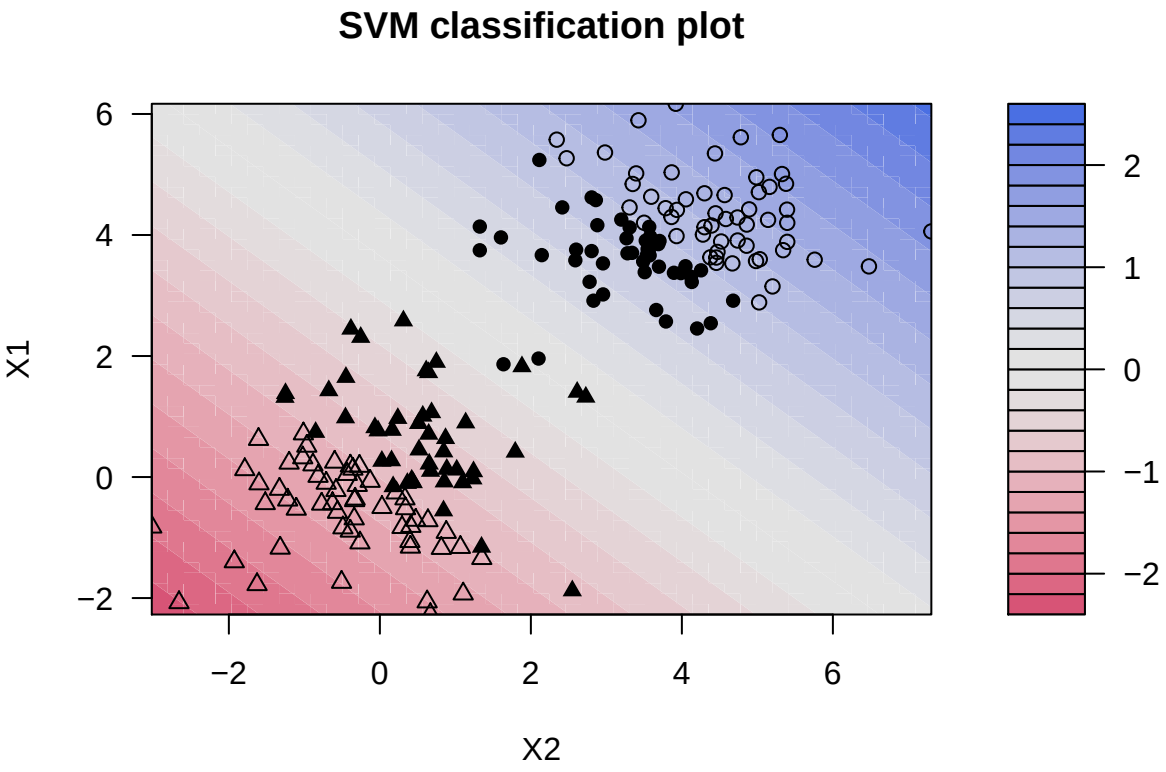
Q. What's the black & empty dots stand for?

Q. The instruction says that support vector classifier is robust, thus can be used for any dataset (when it is not linear), but why this ksvm function not really work for circled data set: df2??

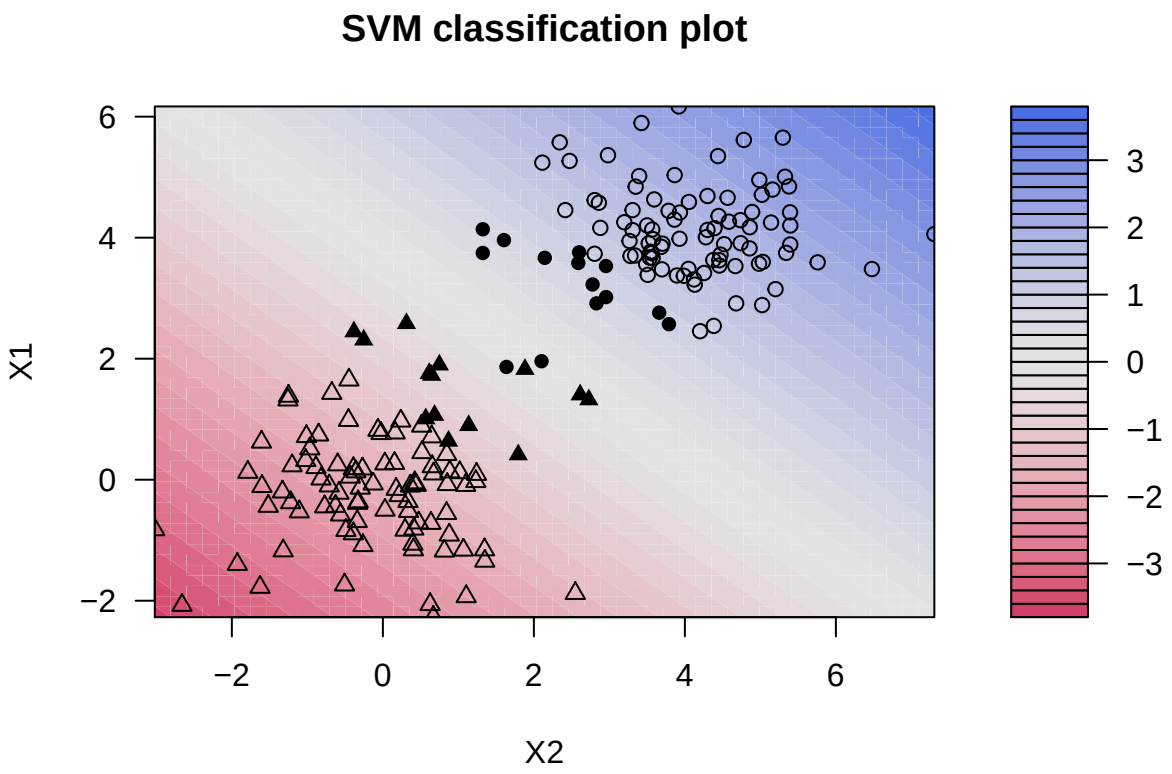
Q. Why is it called "vanilla dot"??

```
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)
  }
}
```

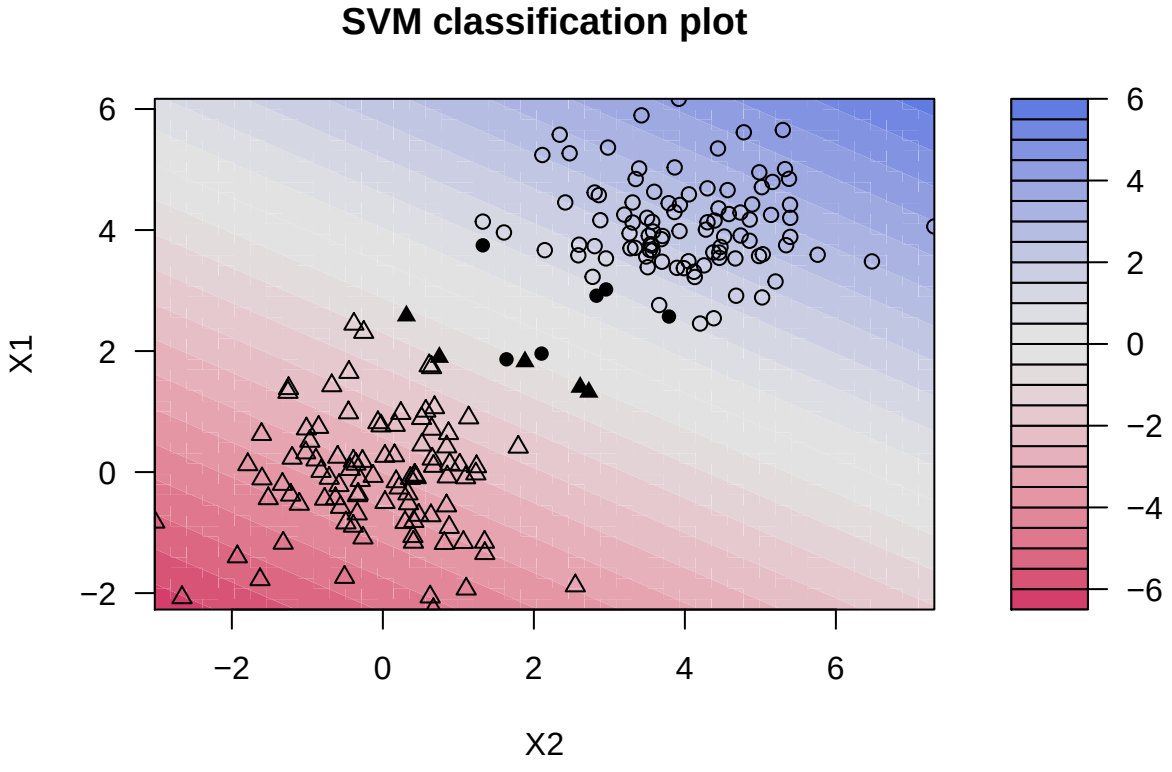
```
##  Setting default kernel parameters
```



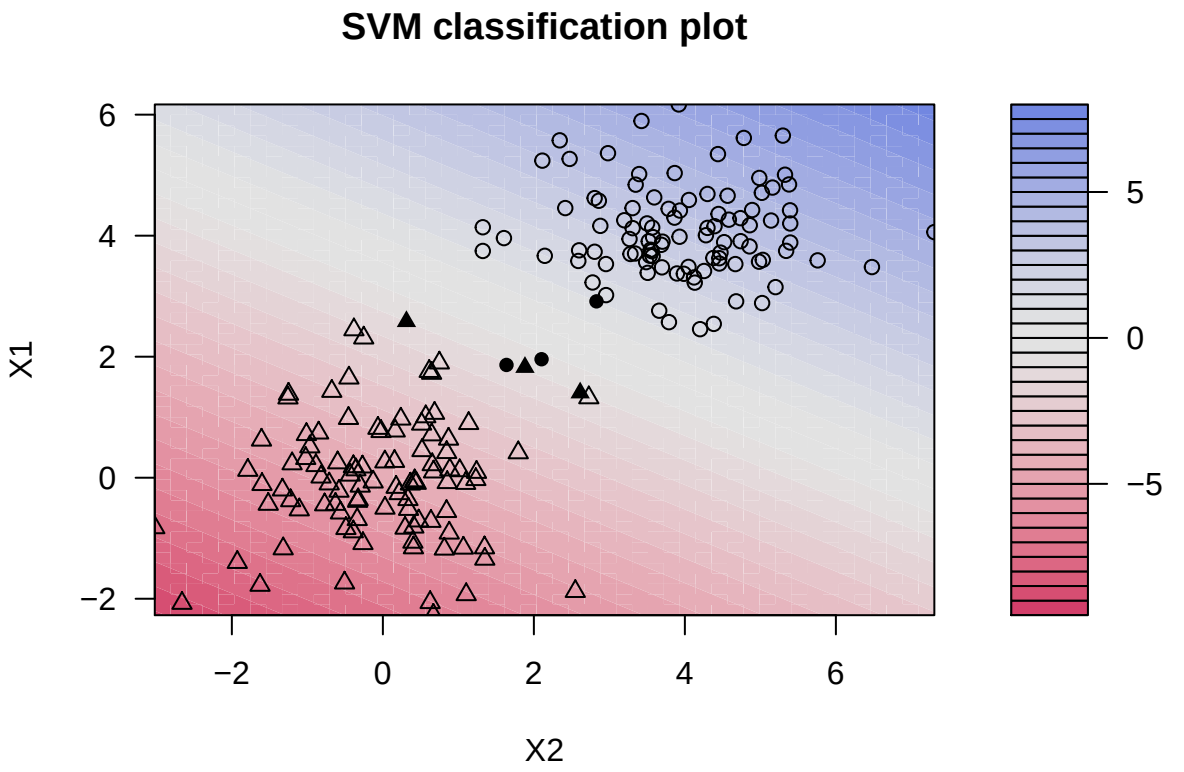

```
##  Setting default kernel parameters
```



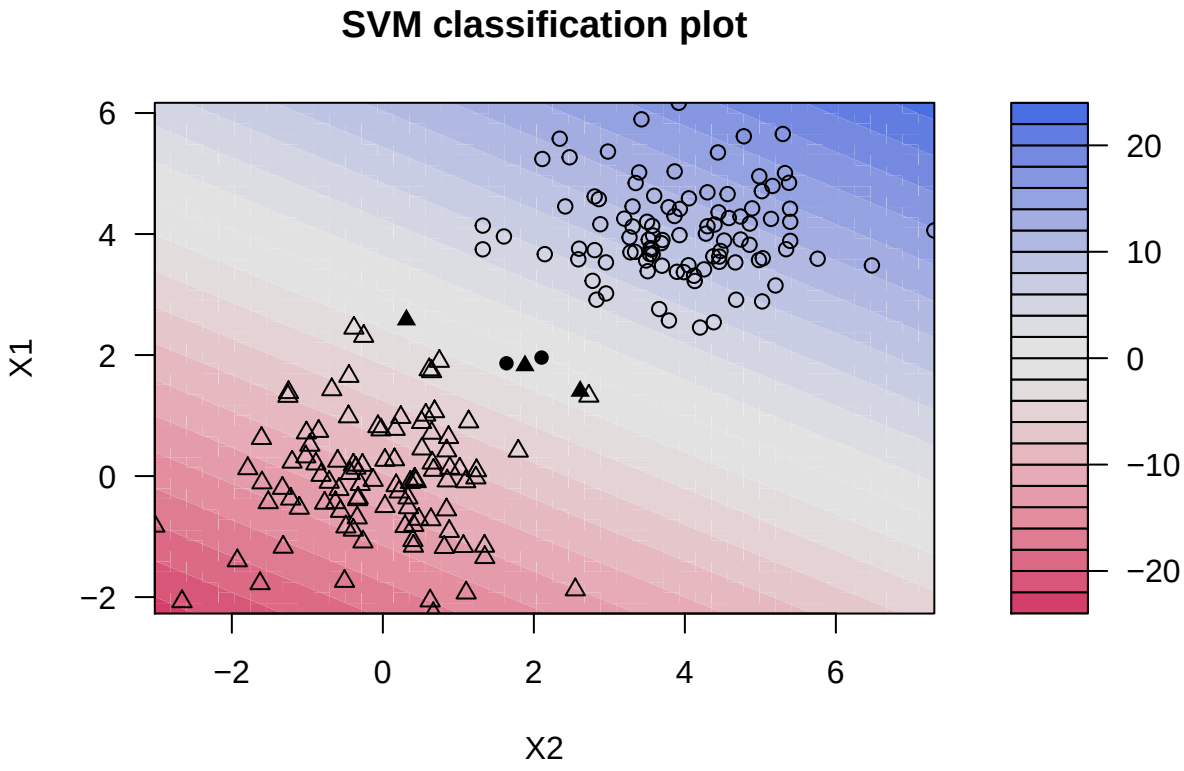

```
##  Setting default kernel parameters
```



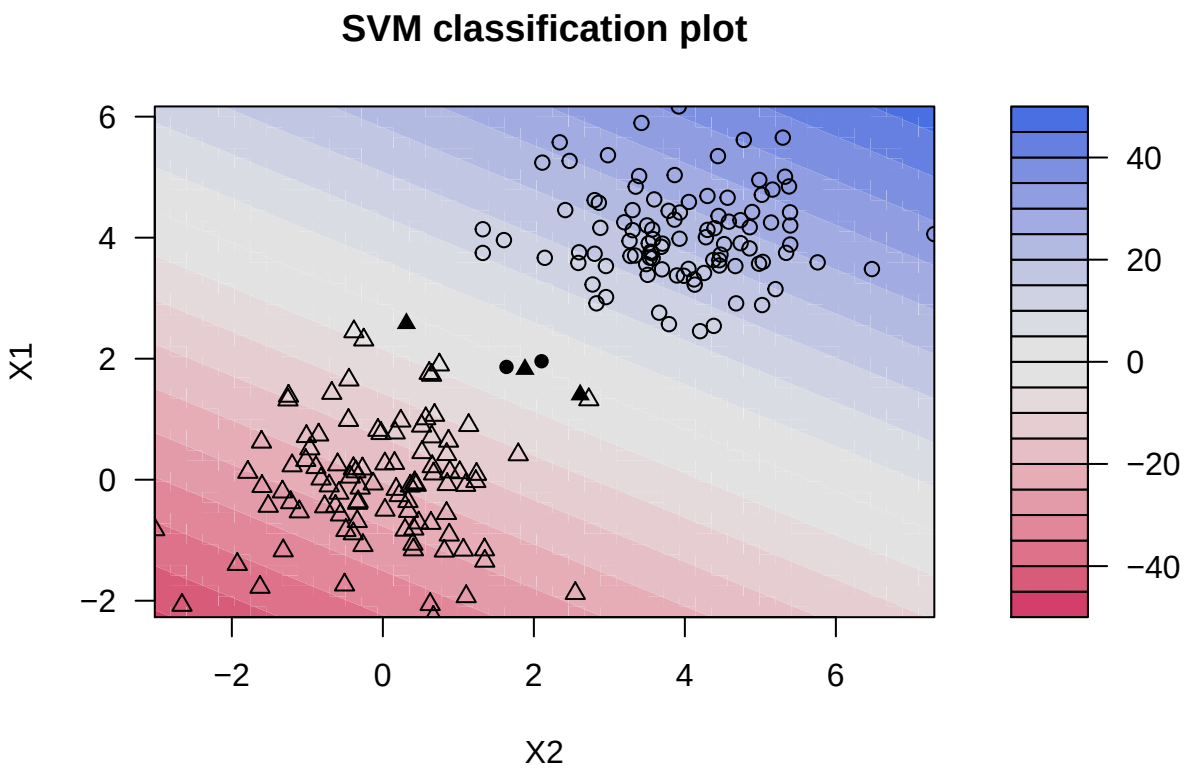

```
##  Setting default kernel parameters
```



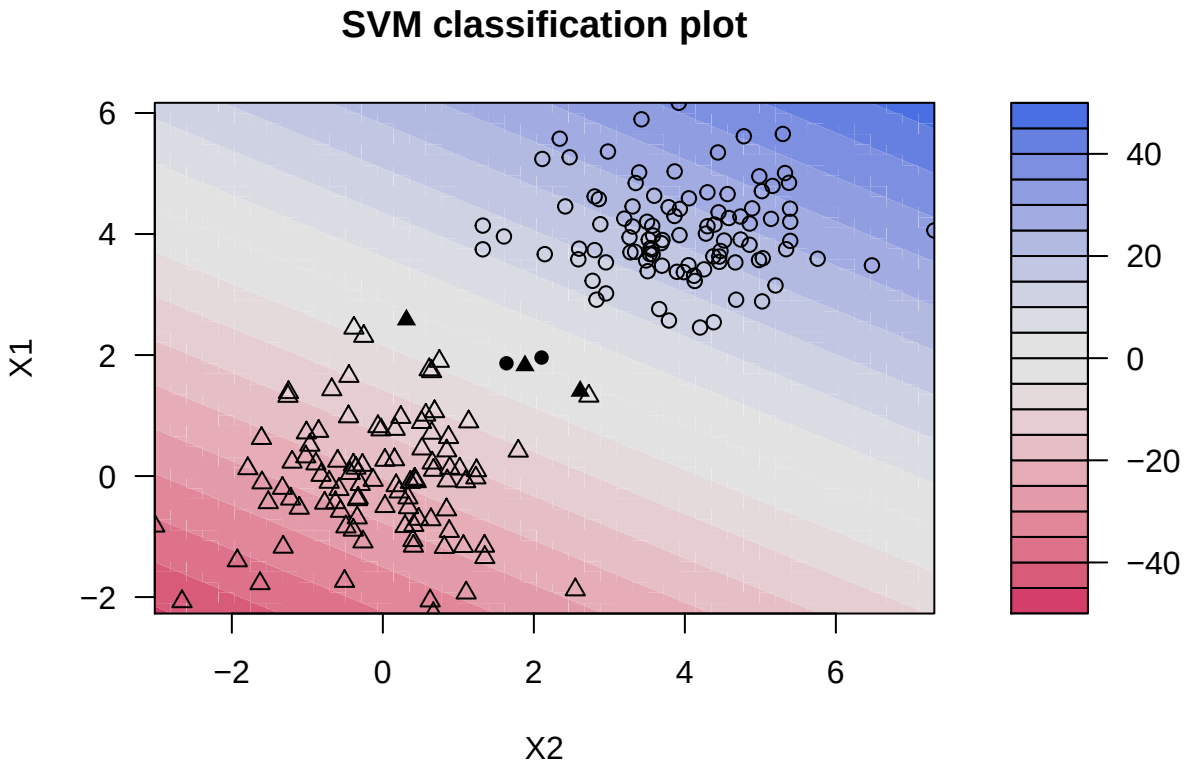

```
##  Setting default kernel parameters
```



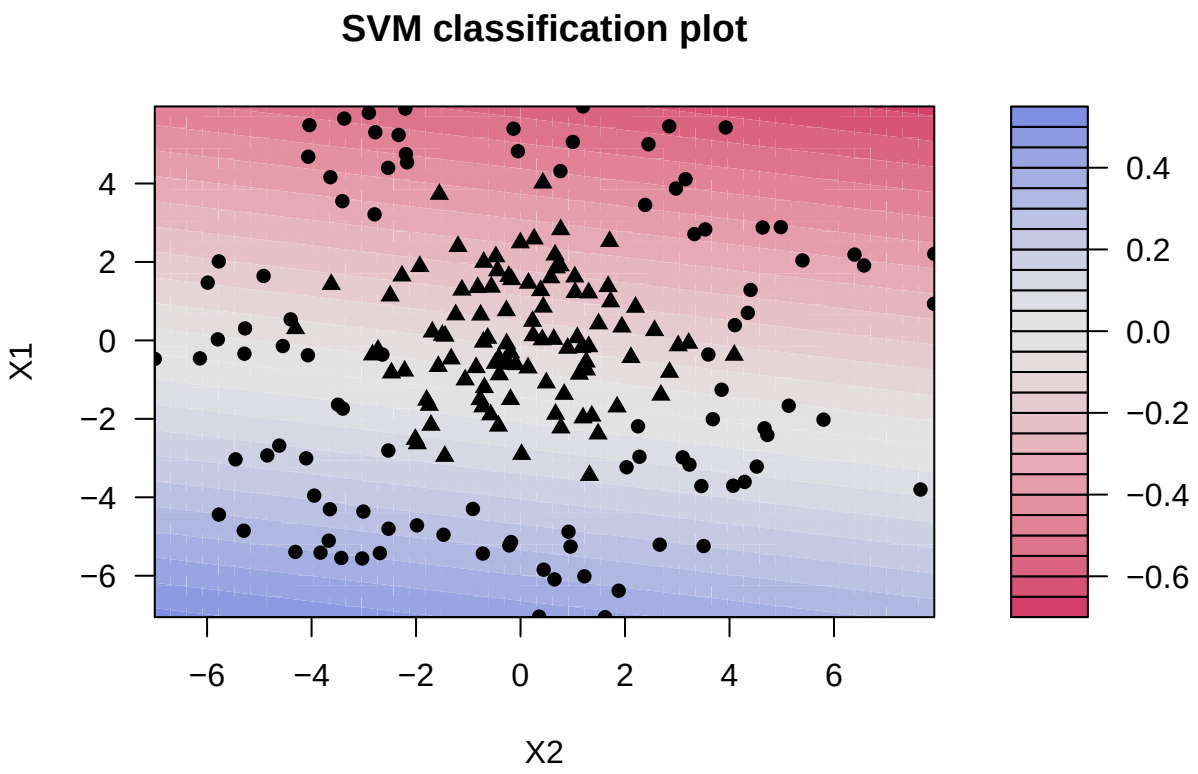

```
##  Setting default kernel parameters
```



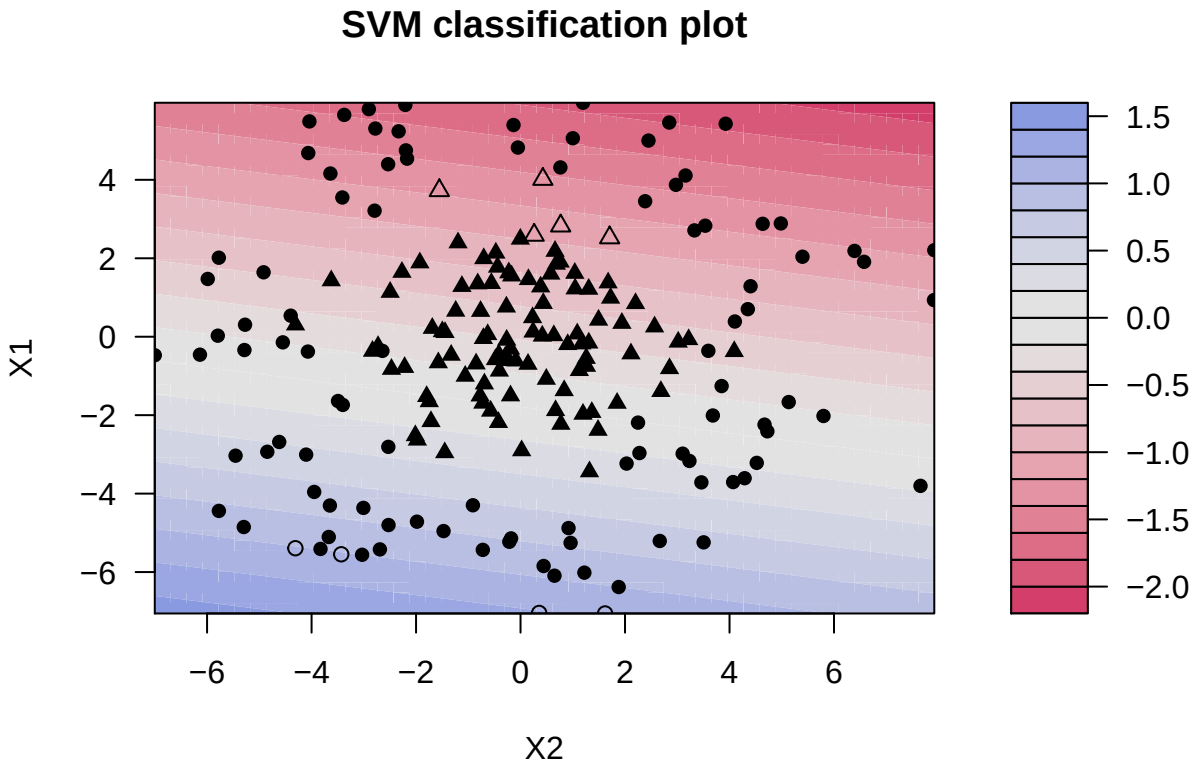

```
##  Setting default kernel parameters
```



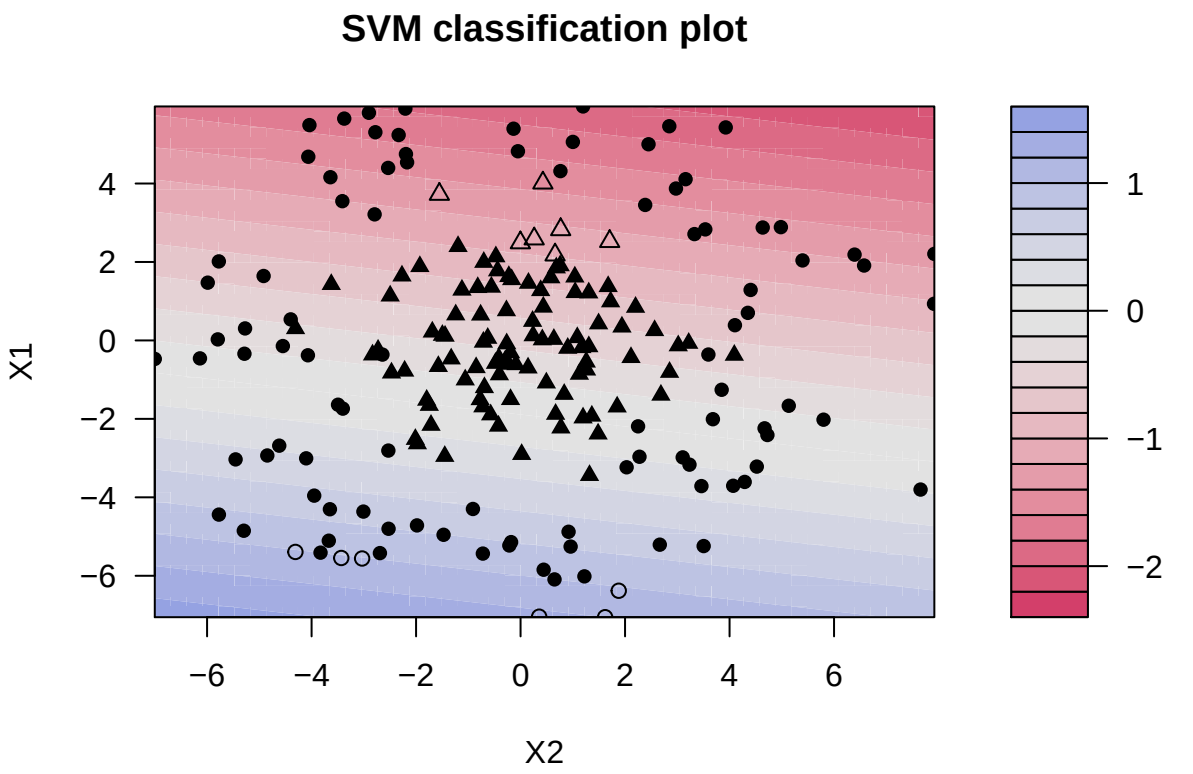

```
##  Setting default kernel parameters
```



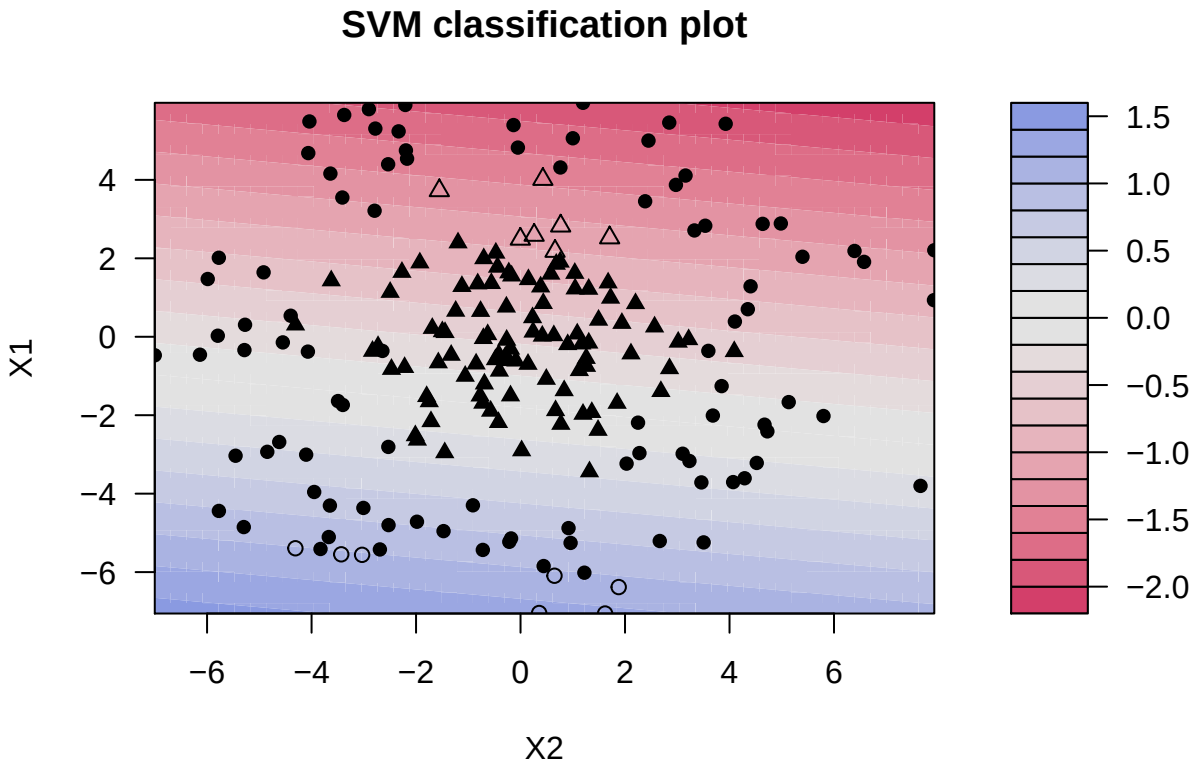

```
##  Setting default kernel parameters
```



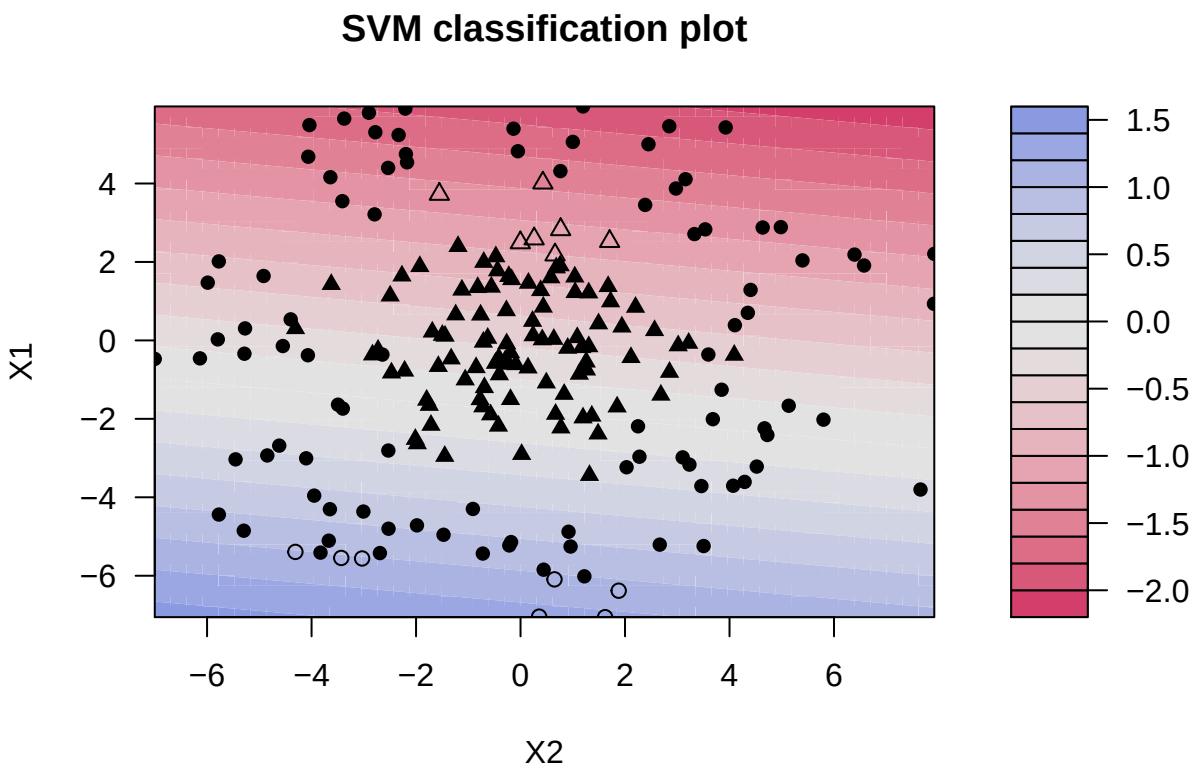

```
##  Setting default kernel parameters
```



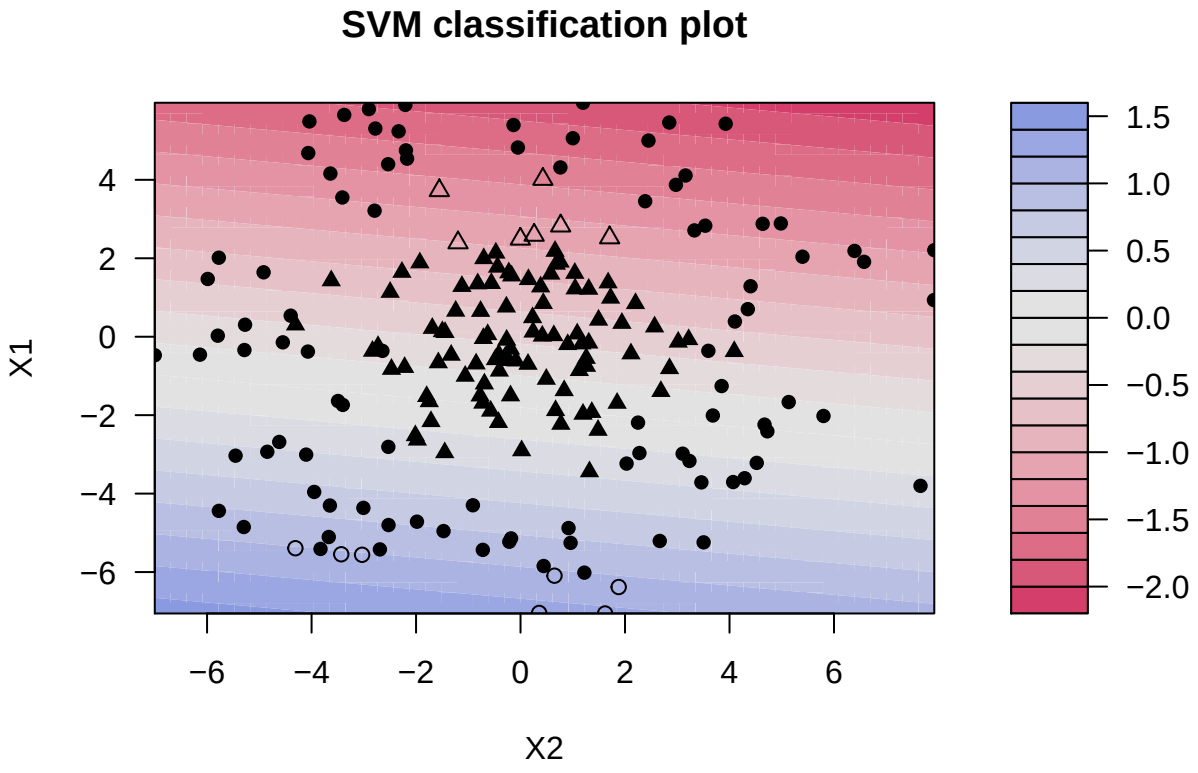

```
##  Setting default kernel parameters
```



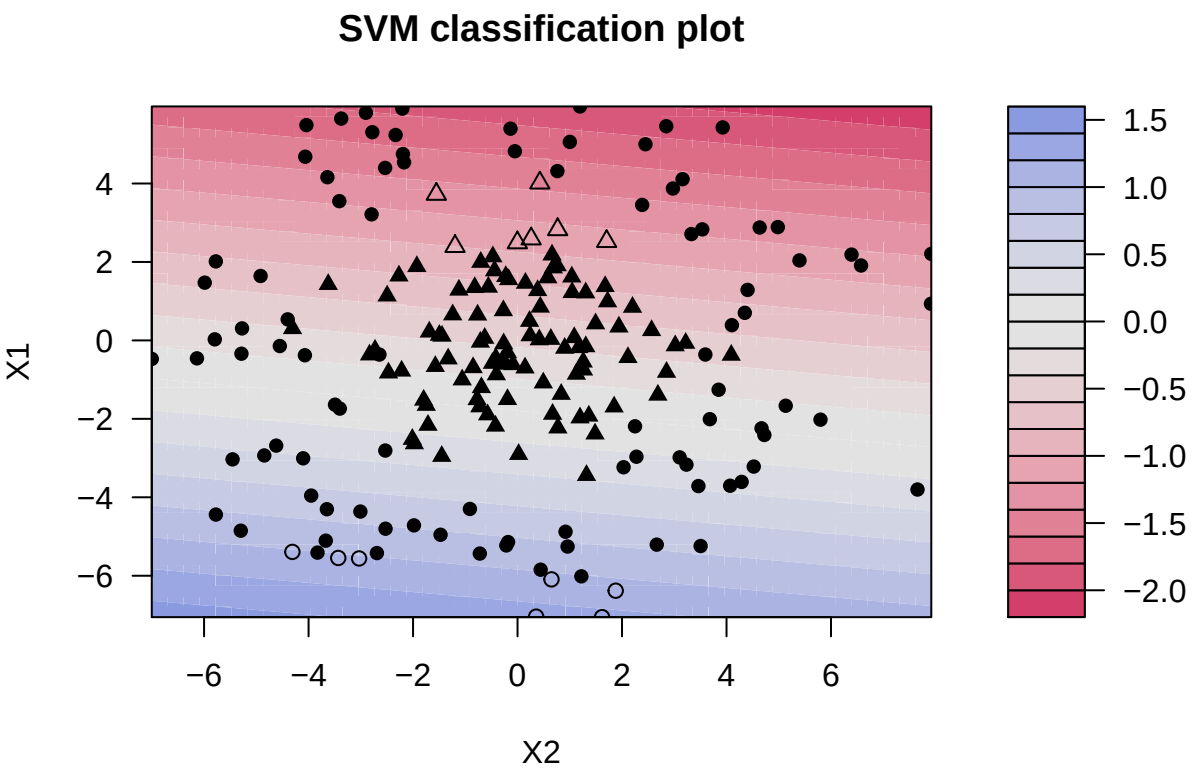

```
##  Setting default kernel parameters
```




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




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



Comment:

The first 7 is for df1, and the last 7 is for df2....

For df1 graphs, I can definitely see the black dots are decreasing.. I guess it is because when the C constraint goes up, they separate the groups better...??

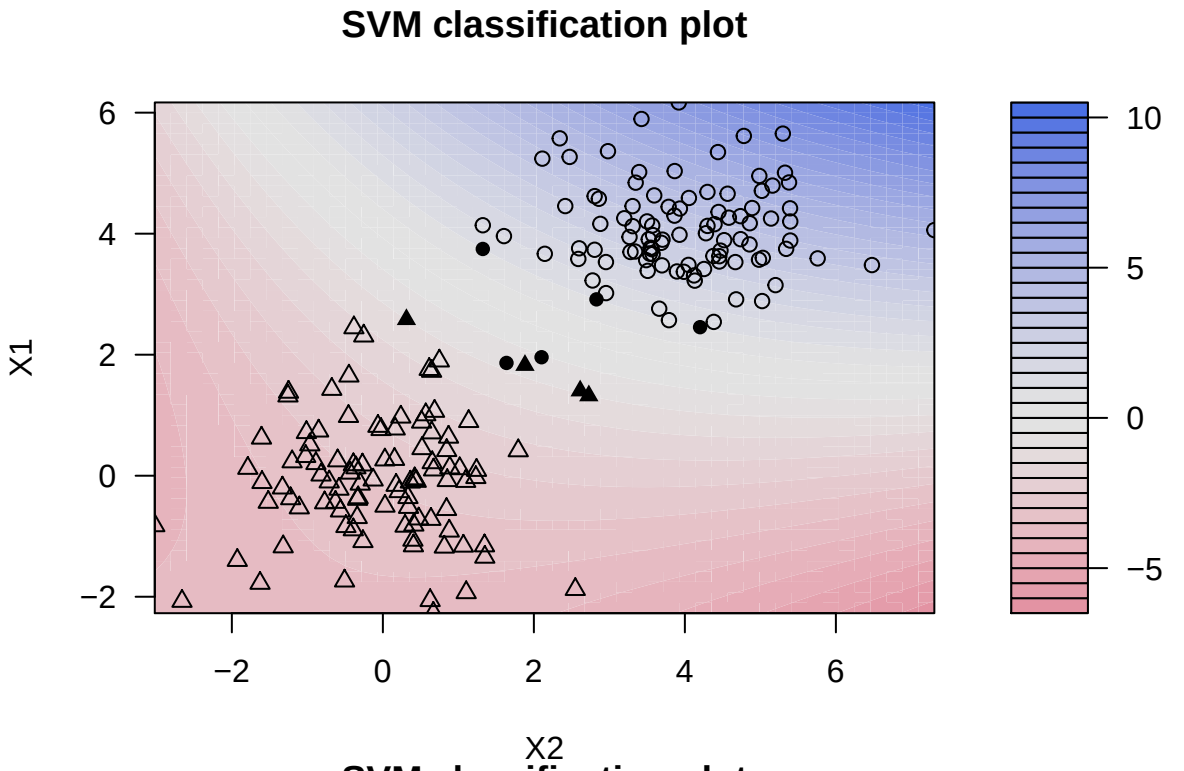
For df2 graphs, it doesn't seem to be well-separated, as they are not linearly drawn data sets...

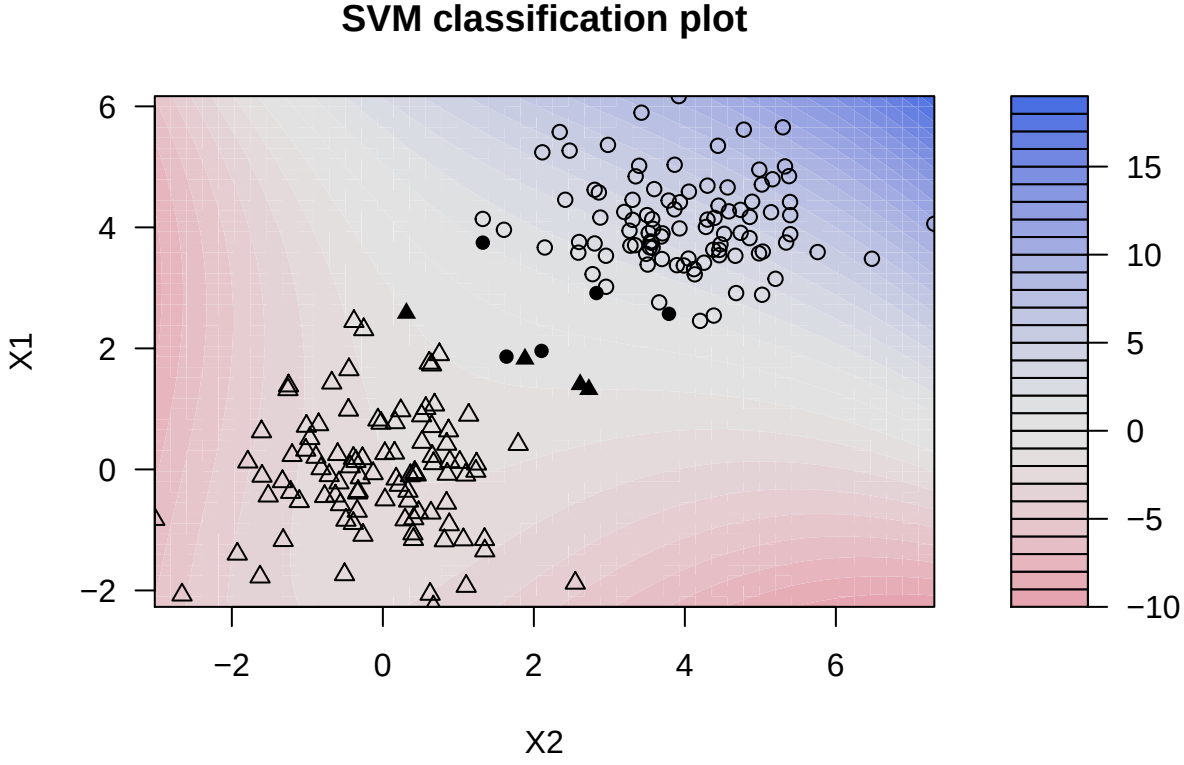
Support vector machine (SVM)

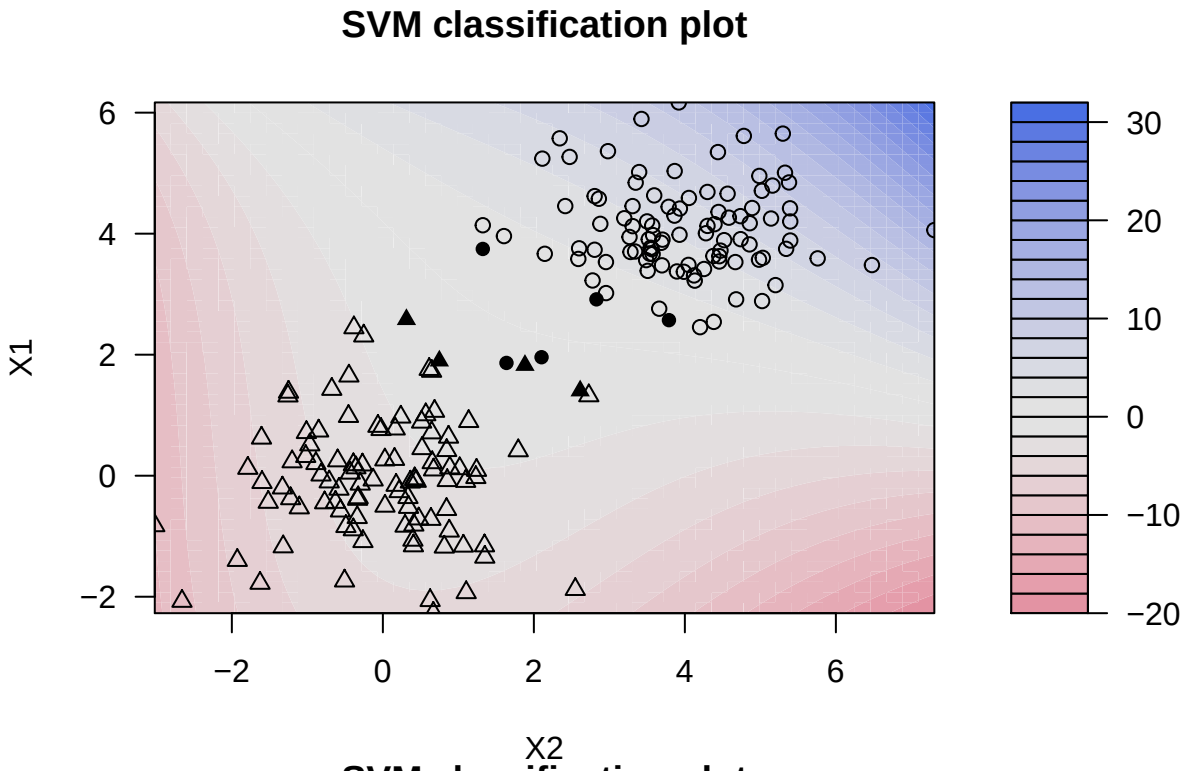
Q. What's the difference between support vector classifier and support vector machine?

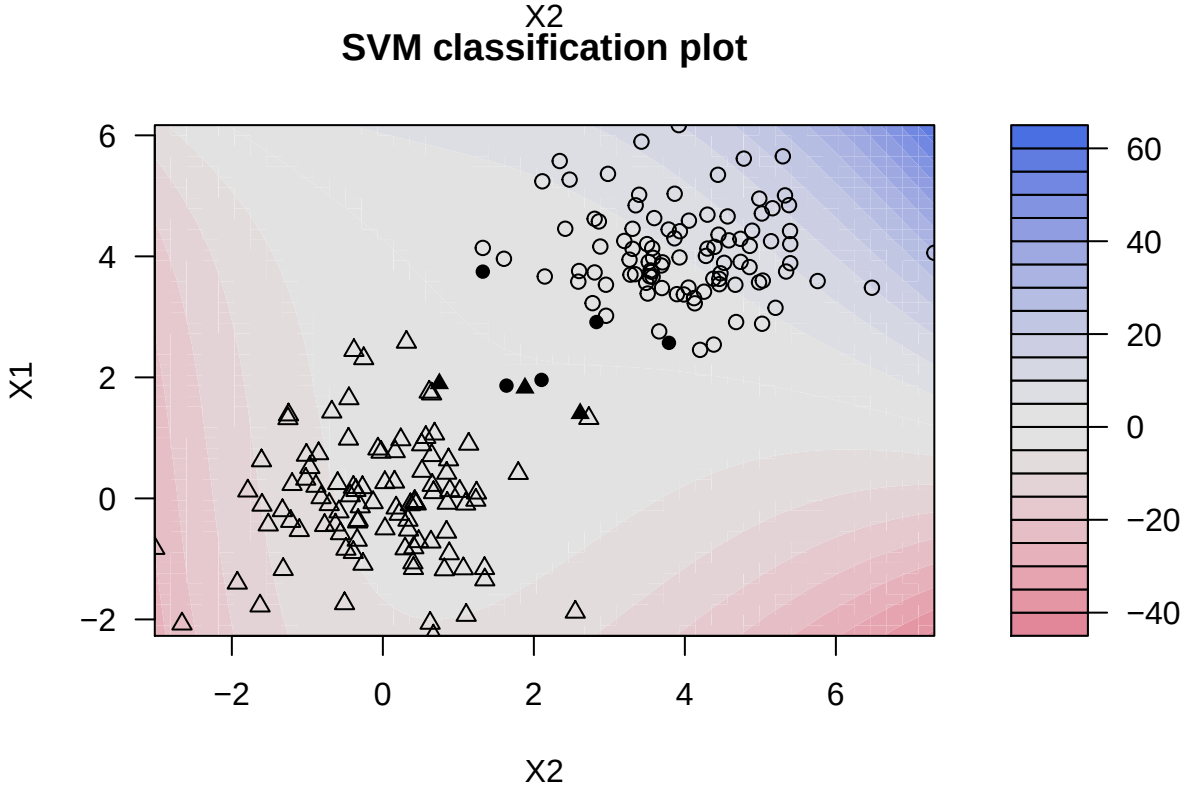
Q. Can you check whether my explanation is correct?

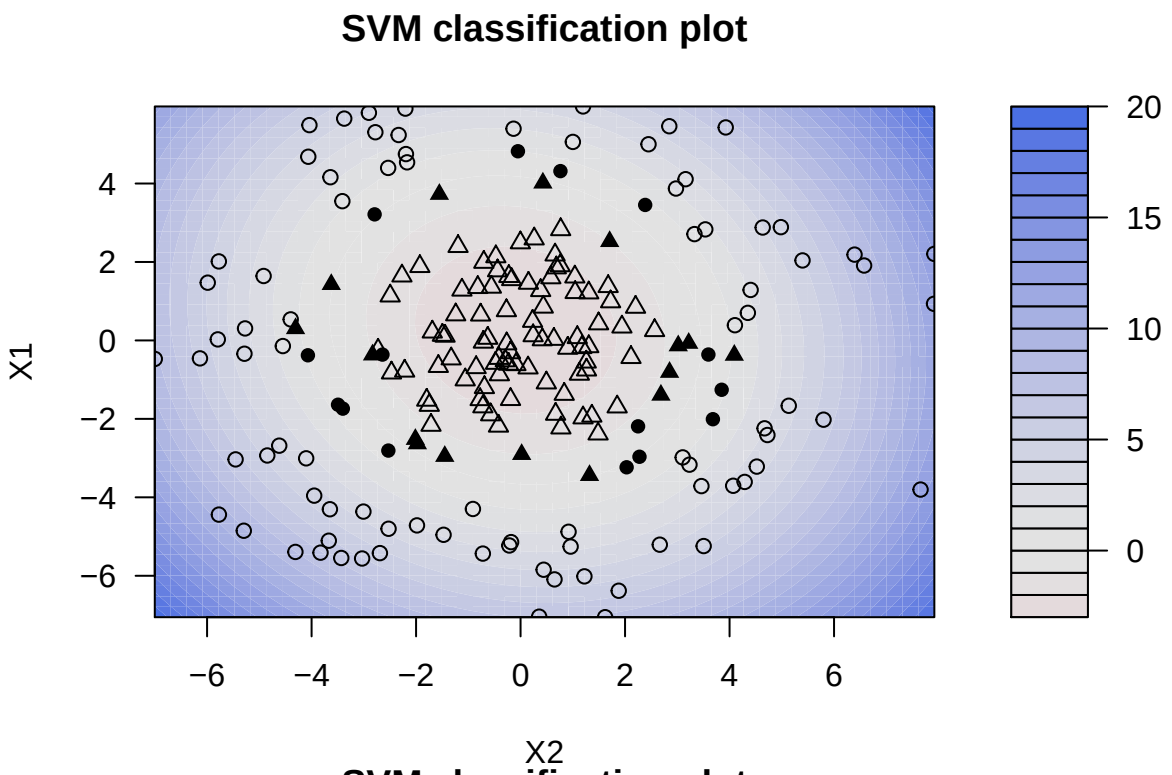
```
deg_vector <- 2:5
gam_vector <- c(0.01, 0.1, 1, 10, 100, 1000, 10000)
dataset_list <- list(df1, df2)

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)
  }
}
```

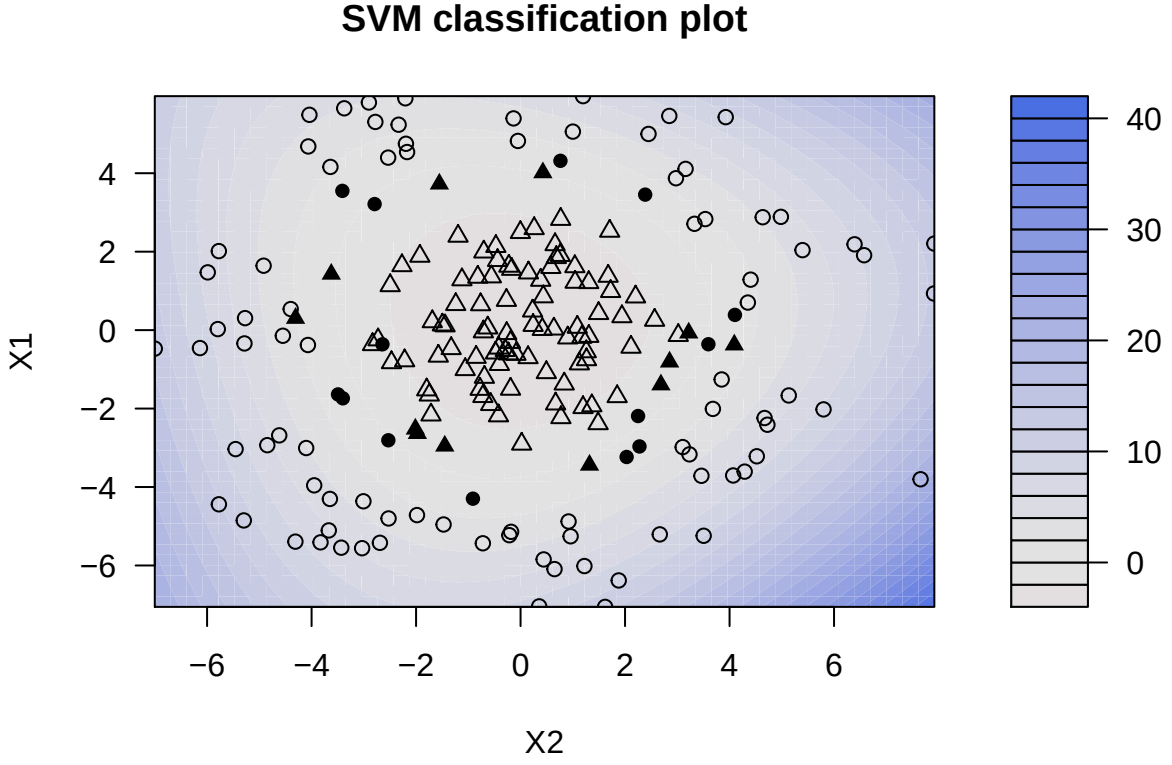

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
10
5
0
−5

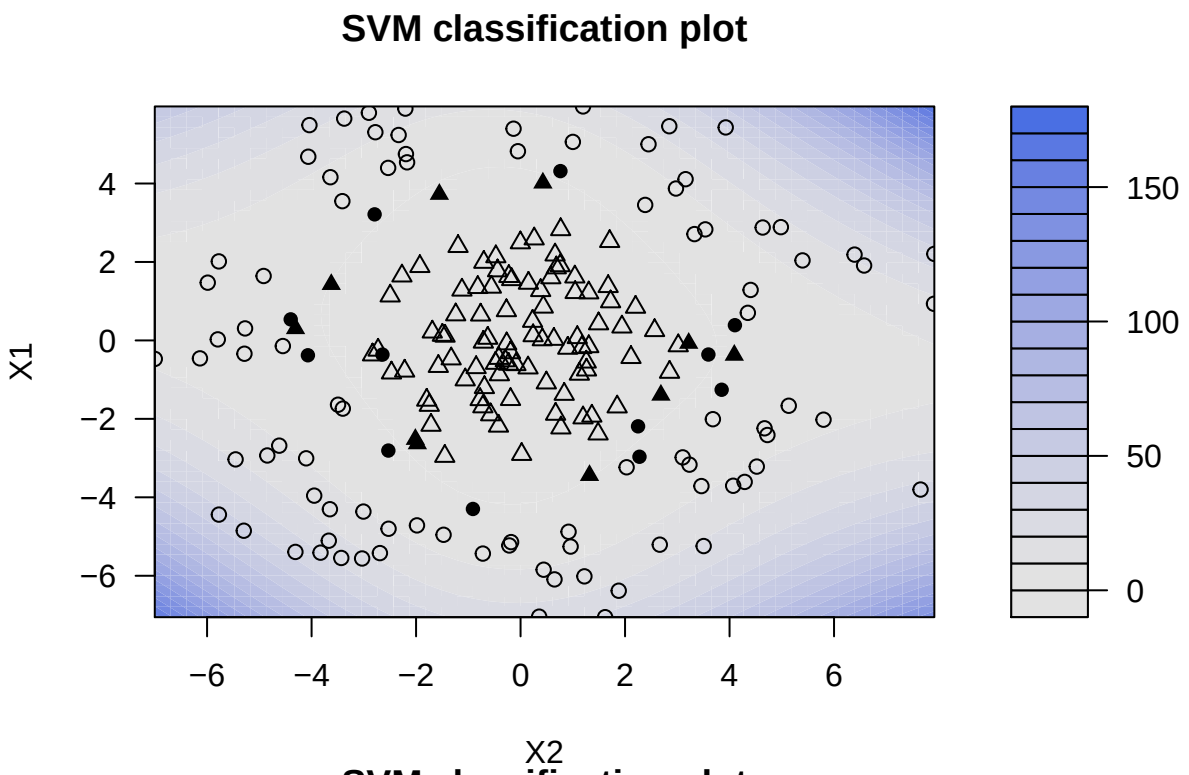

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
15
10
5
0
−5
−10

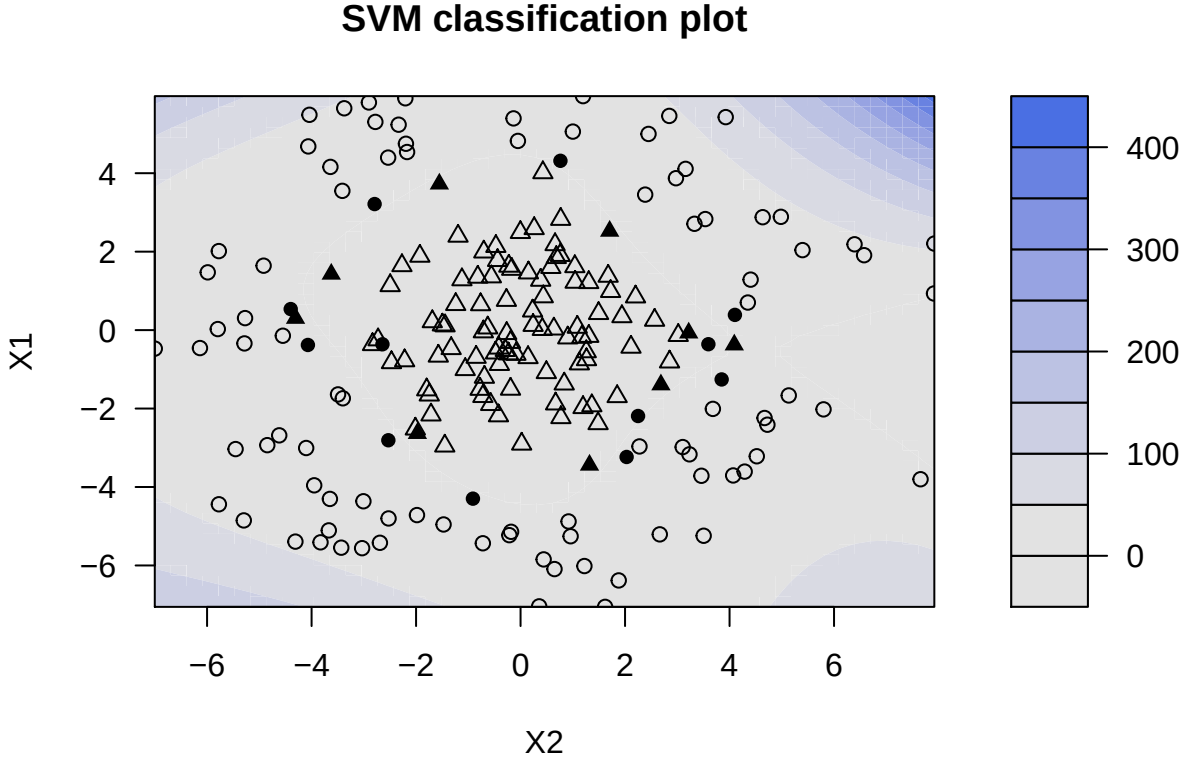

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
30
20
10
0
−10
−20


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

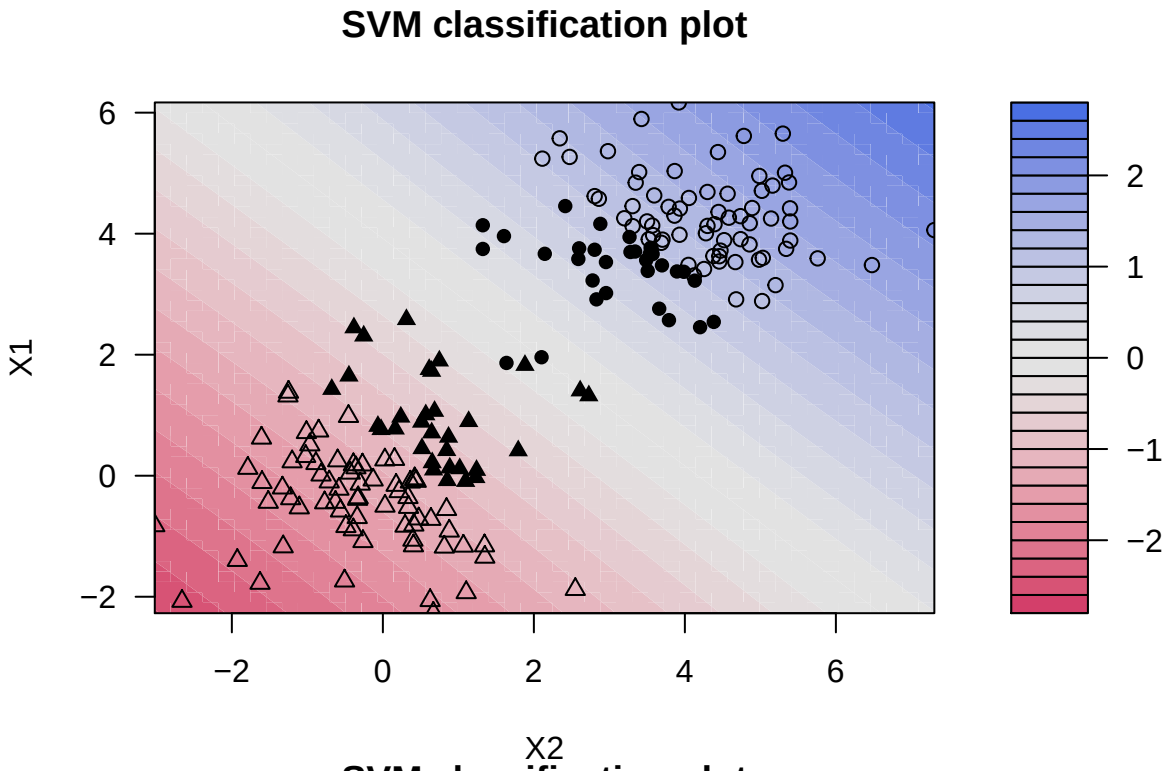

SVM classification plot
X1
X2
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
20
15
10
5
0

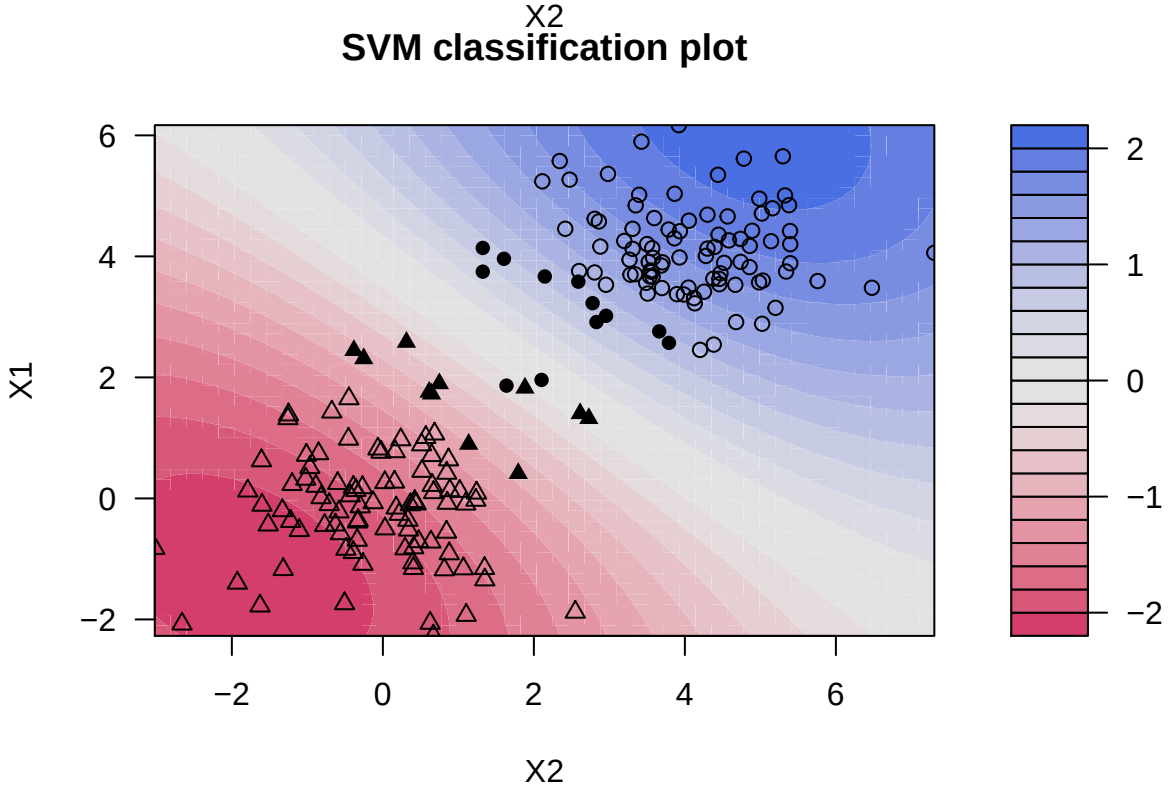

SVM classification plot
X1
X2
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
40
30
20
10
0

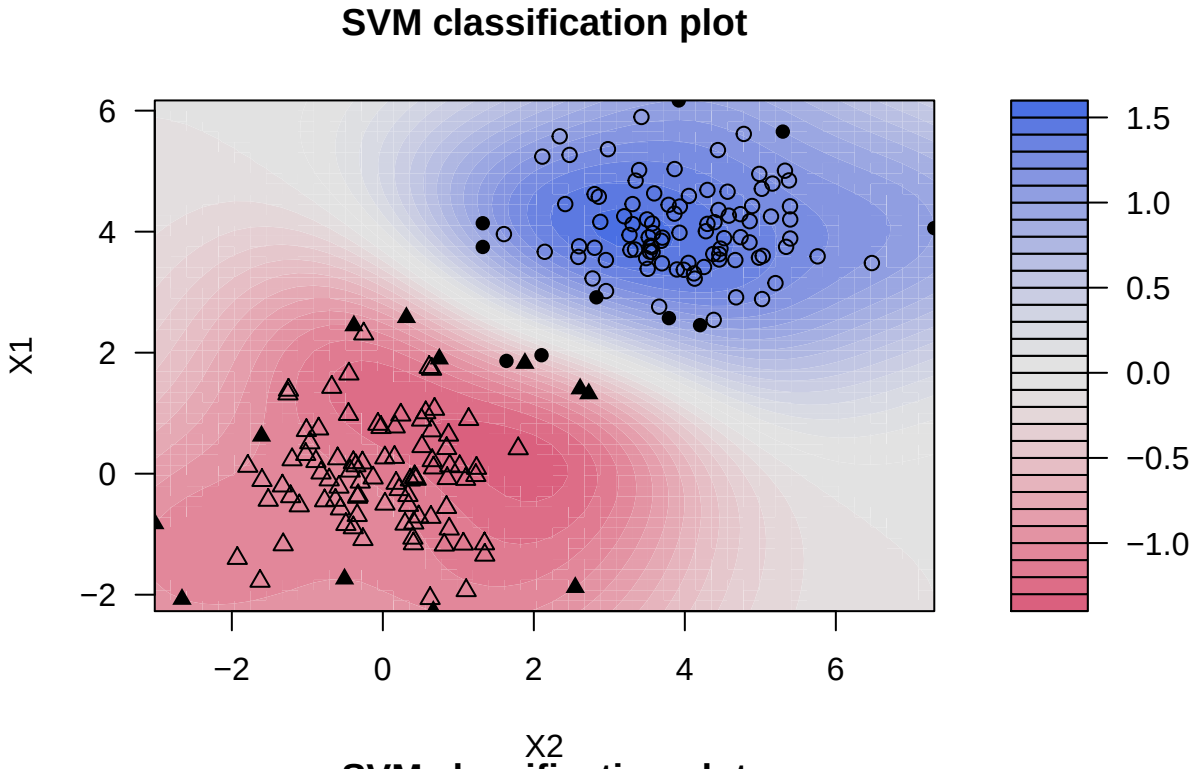


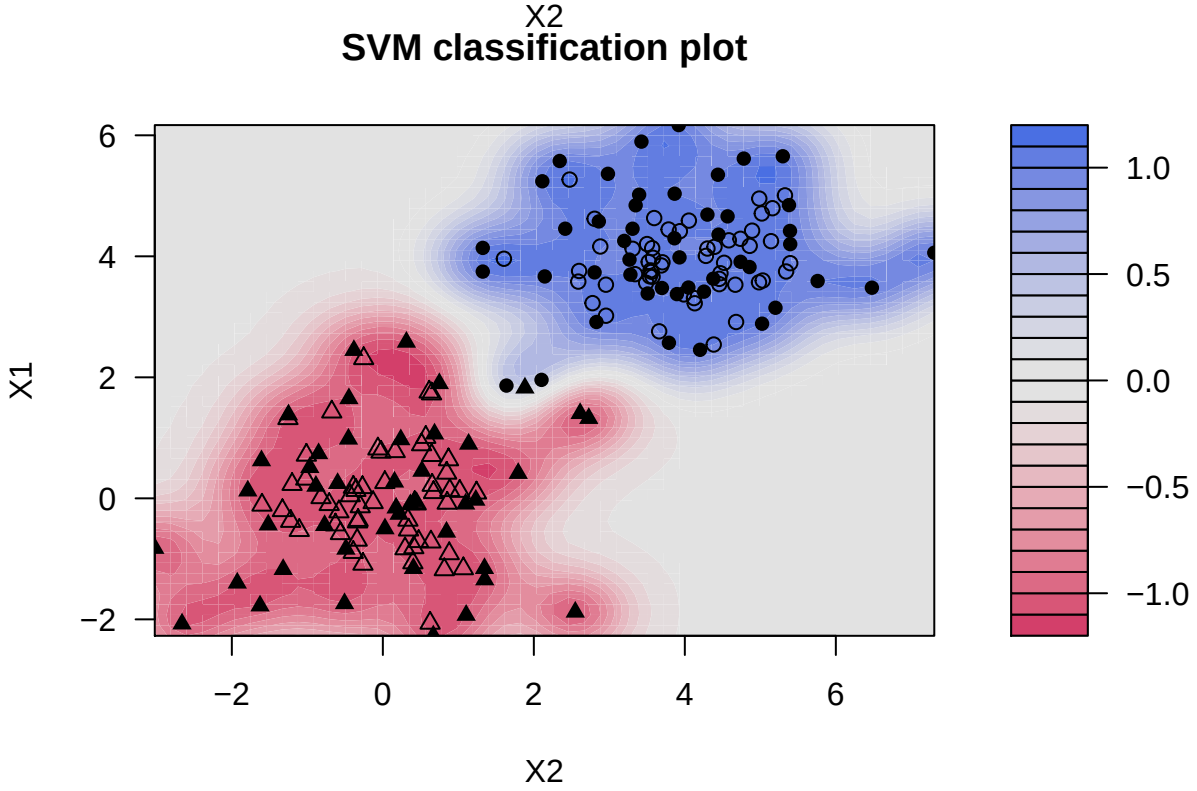


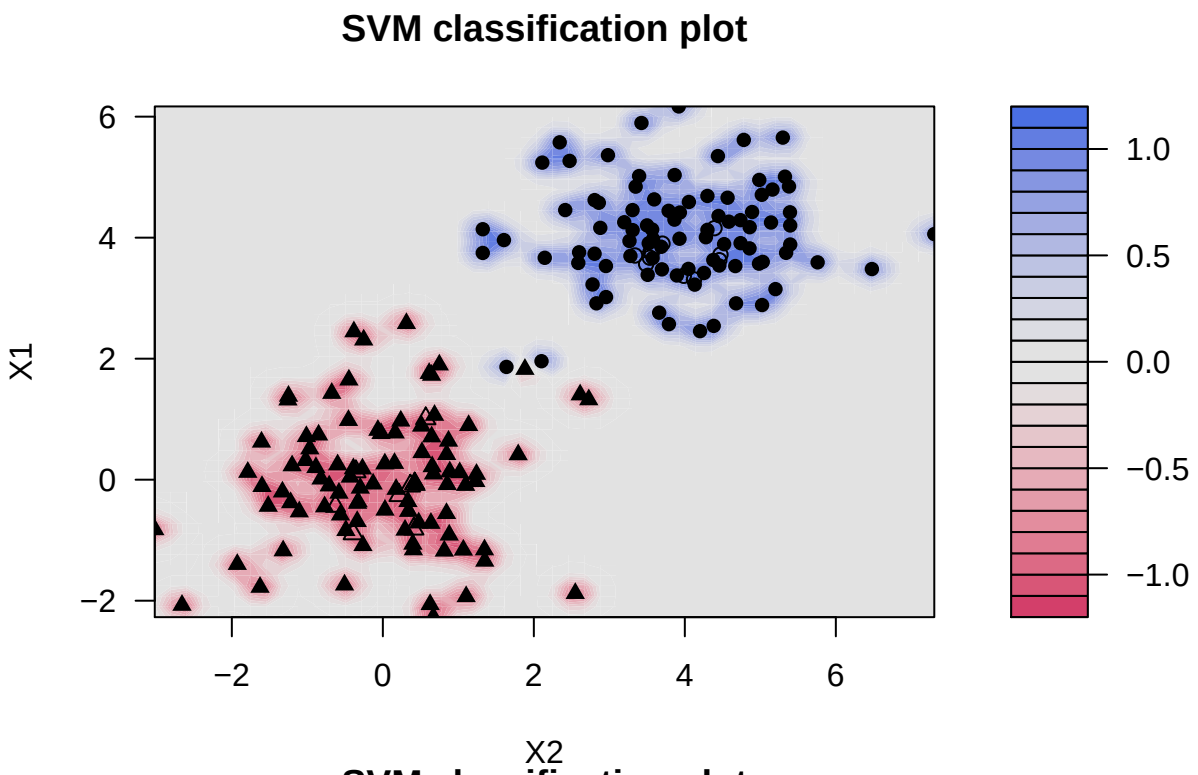

```
for (df in dataset_list) {
  for (gam in gam_vector) {
    fit <- ksvm(y~X1+X2, data = df, kernel = "rbfdot", kpar=list(sigma=gam))
    plot(fit, data=df)
  }
}
```

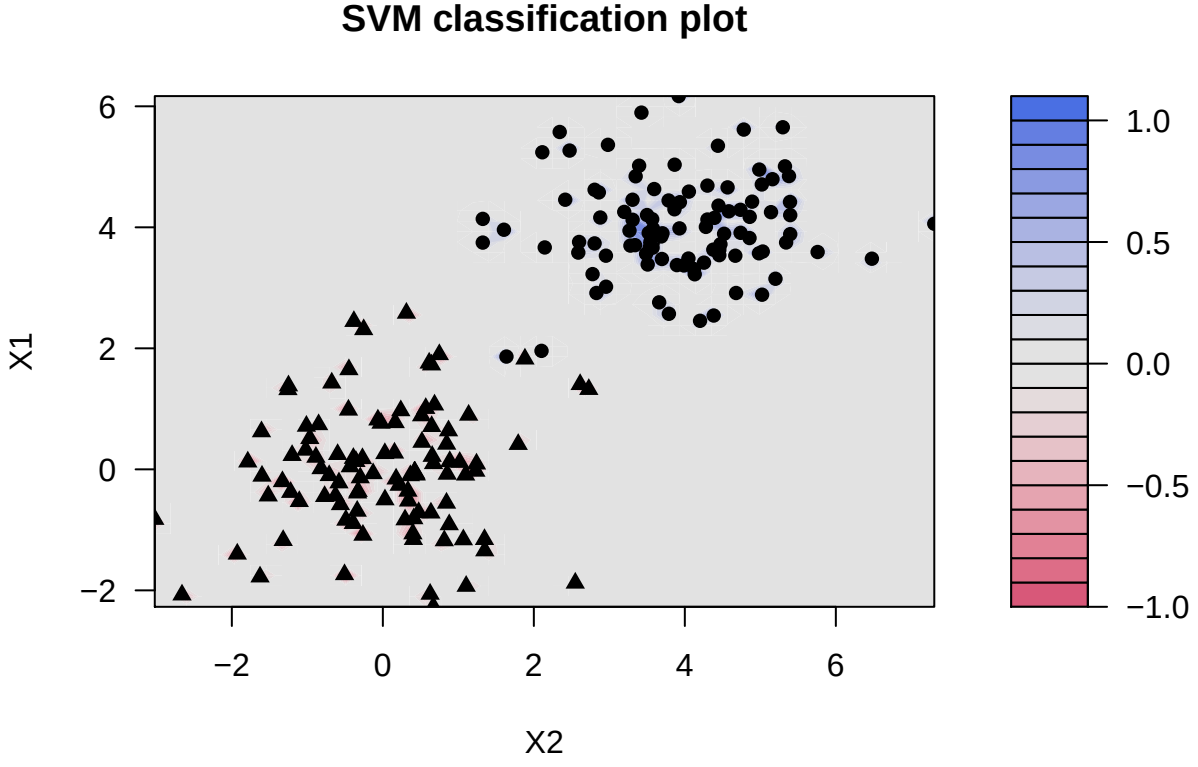

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
2
1
0
−1
−2

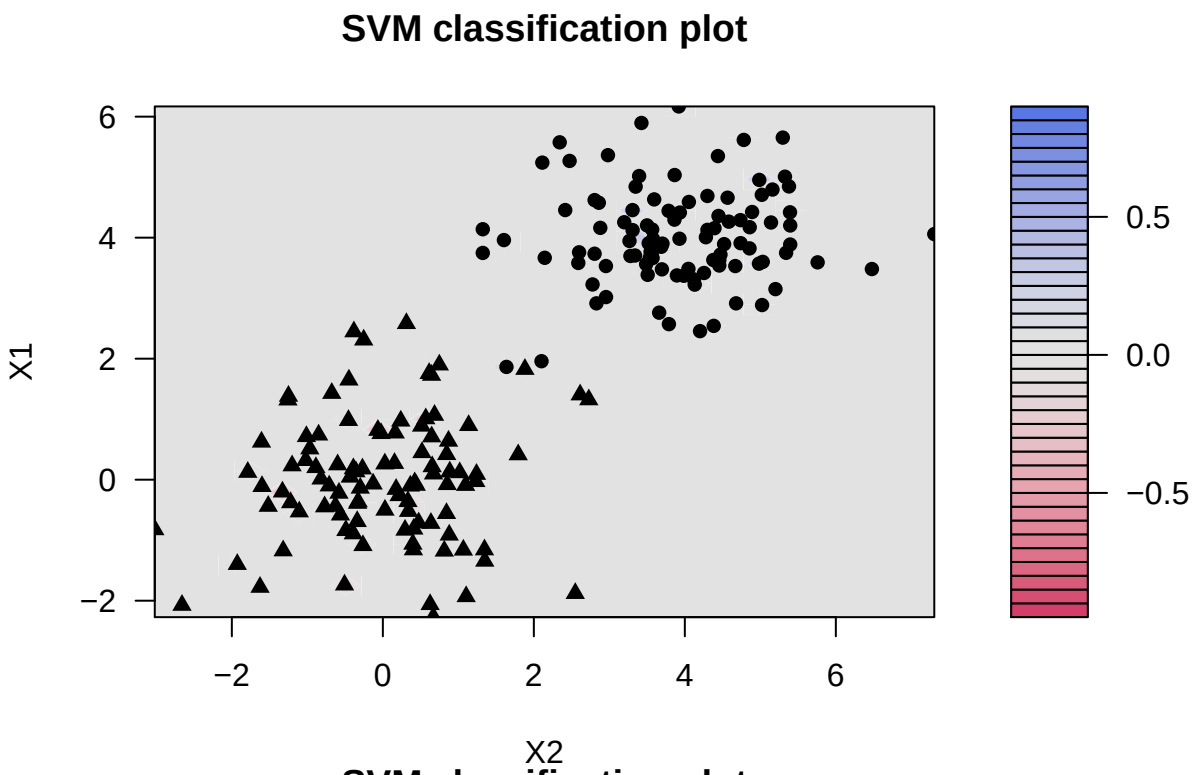

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
2
1
0
−1
−2

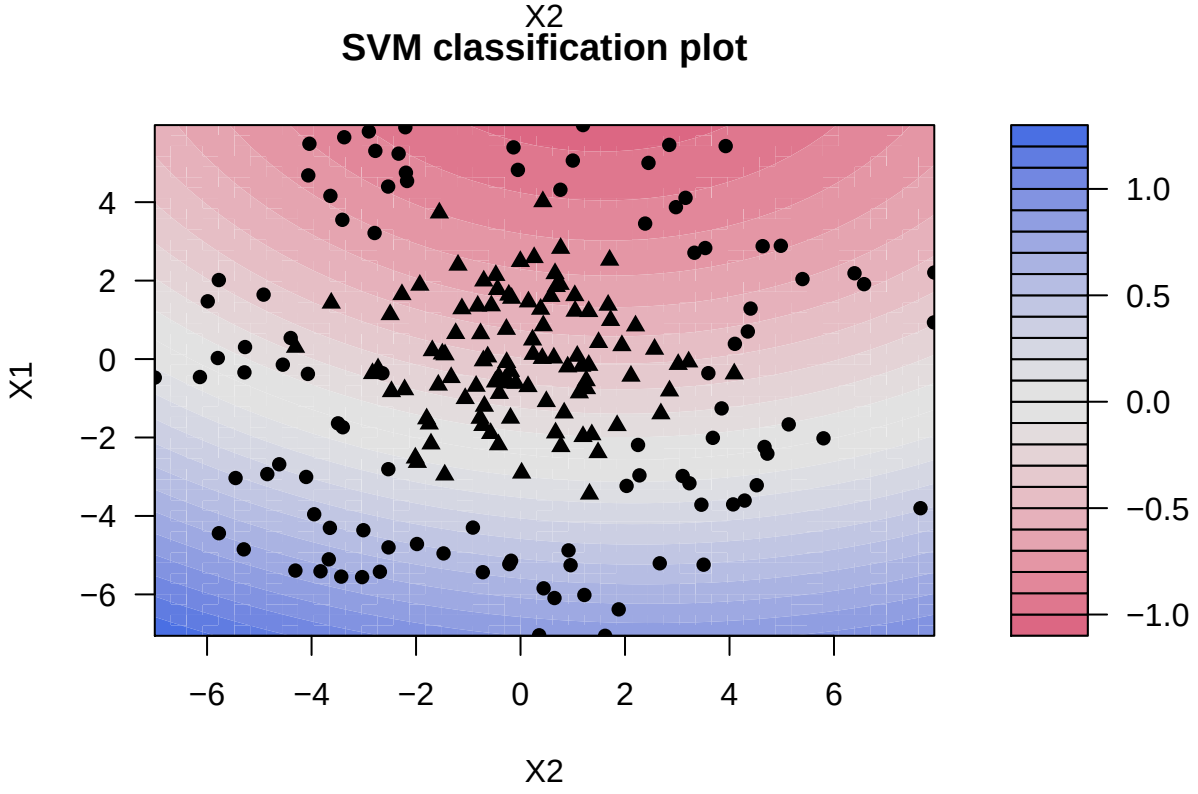

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
1.5
1.0
0.5
0.0
−0.5
−1.0

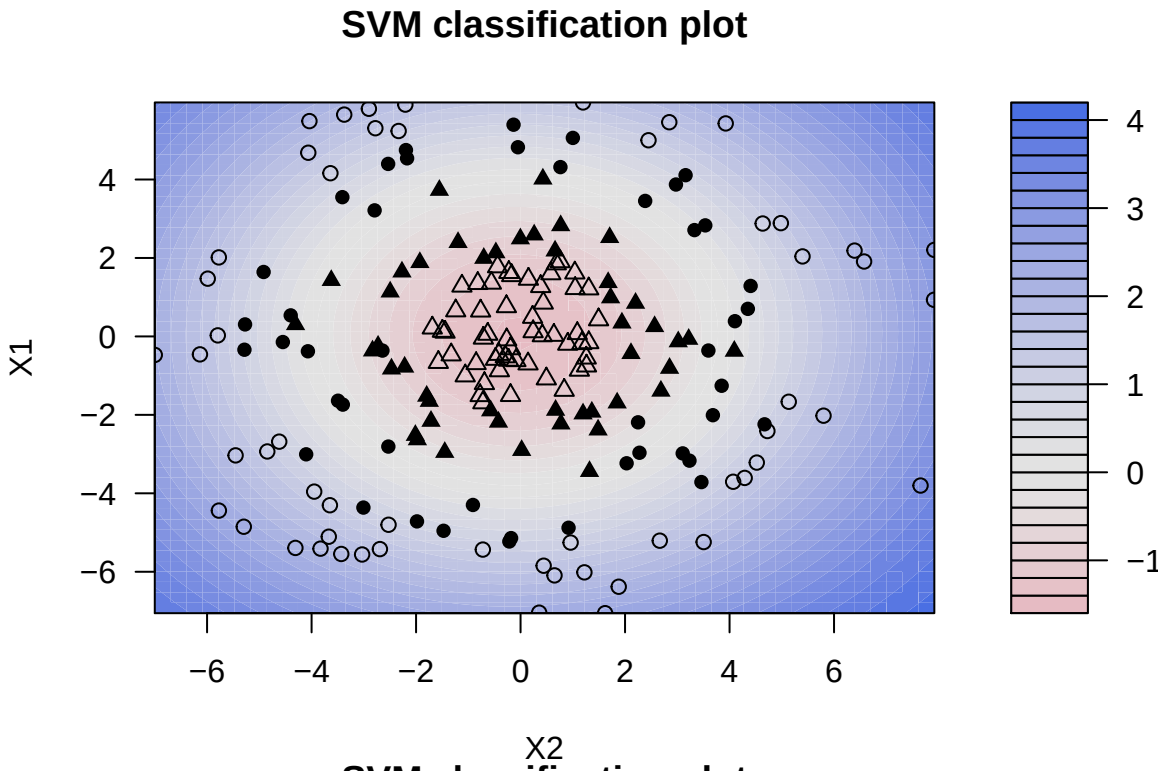

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
1.0
0.5
0.0
−0.5
−1.0

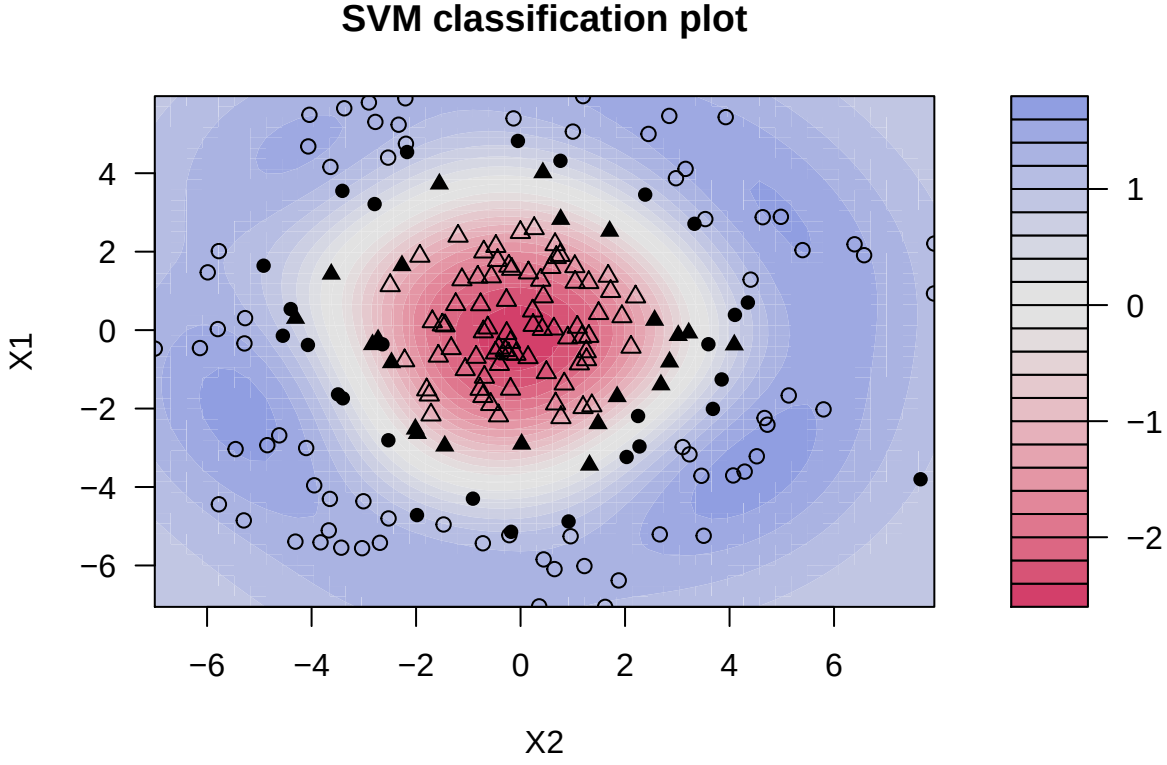

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
1.0
0.5
0.0
−0.5
−1.0

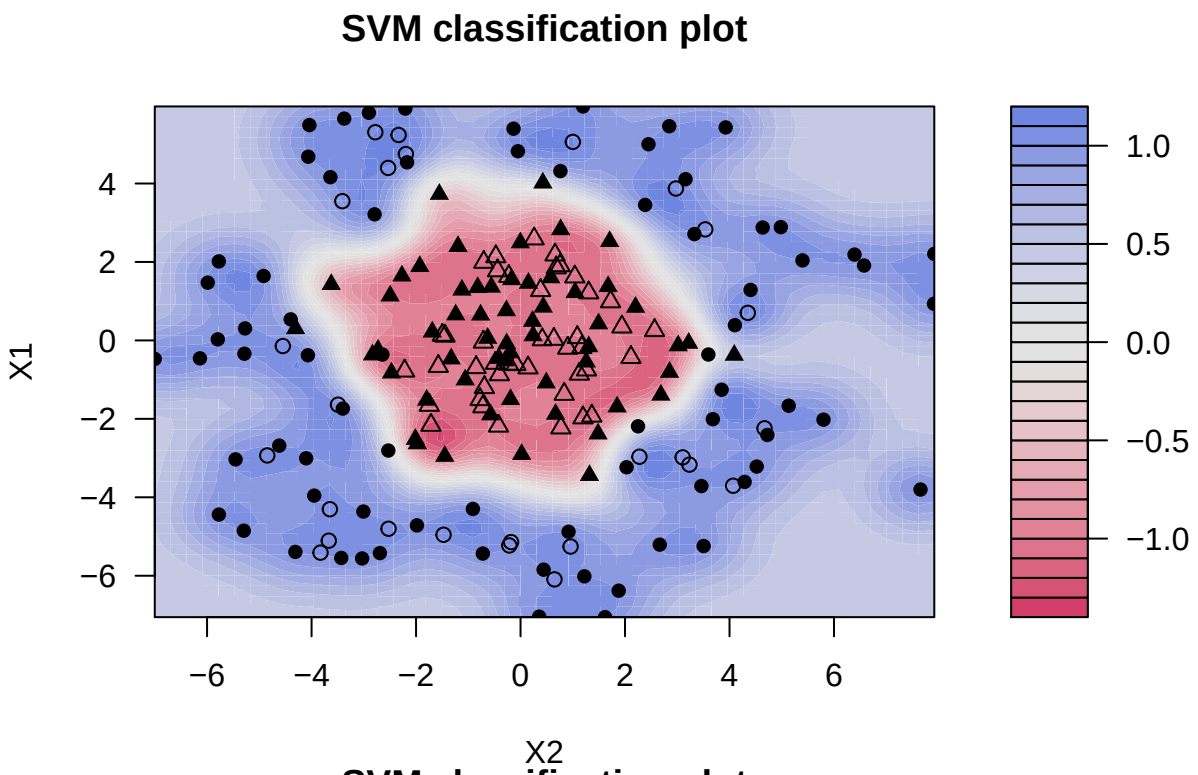

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
1.0
0.5
0.0
−0.5
−1.0

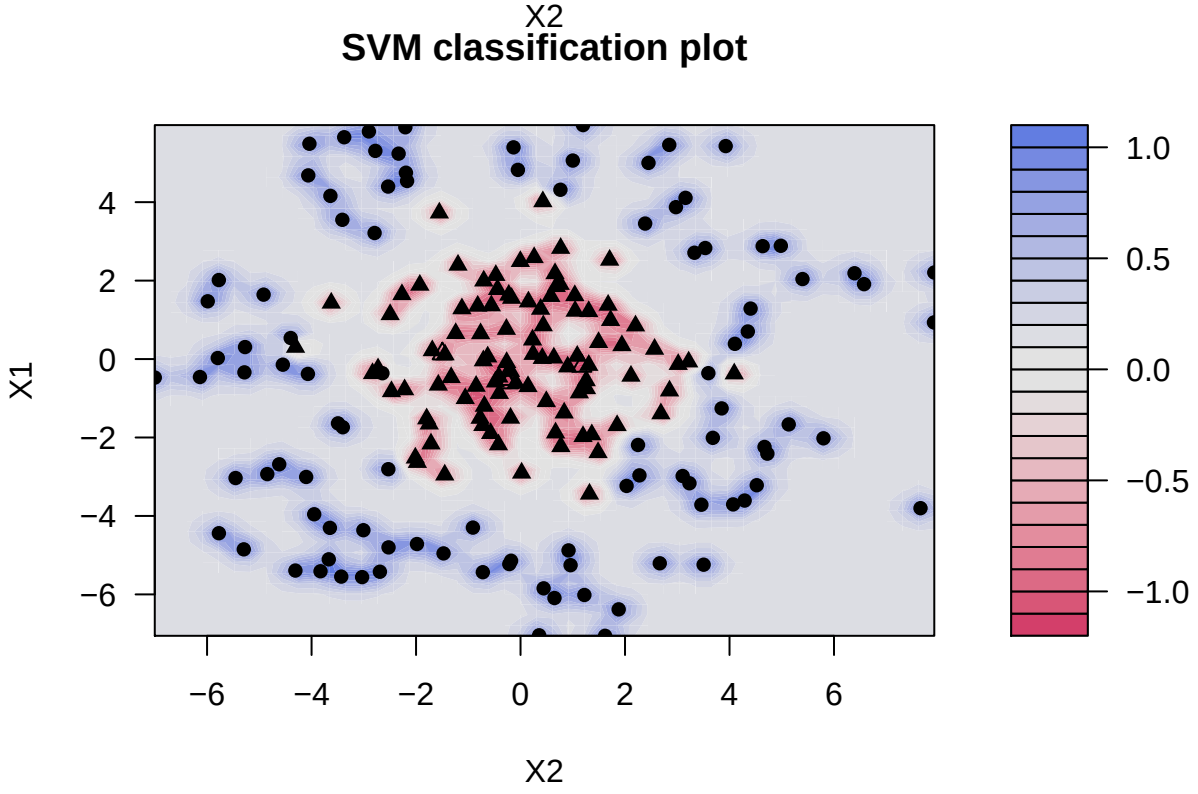

SVM classification plot
X1
X2
6
4
2
0
−2
−2
0
2
4
6
0.5
0.0
−0.5

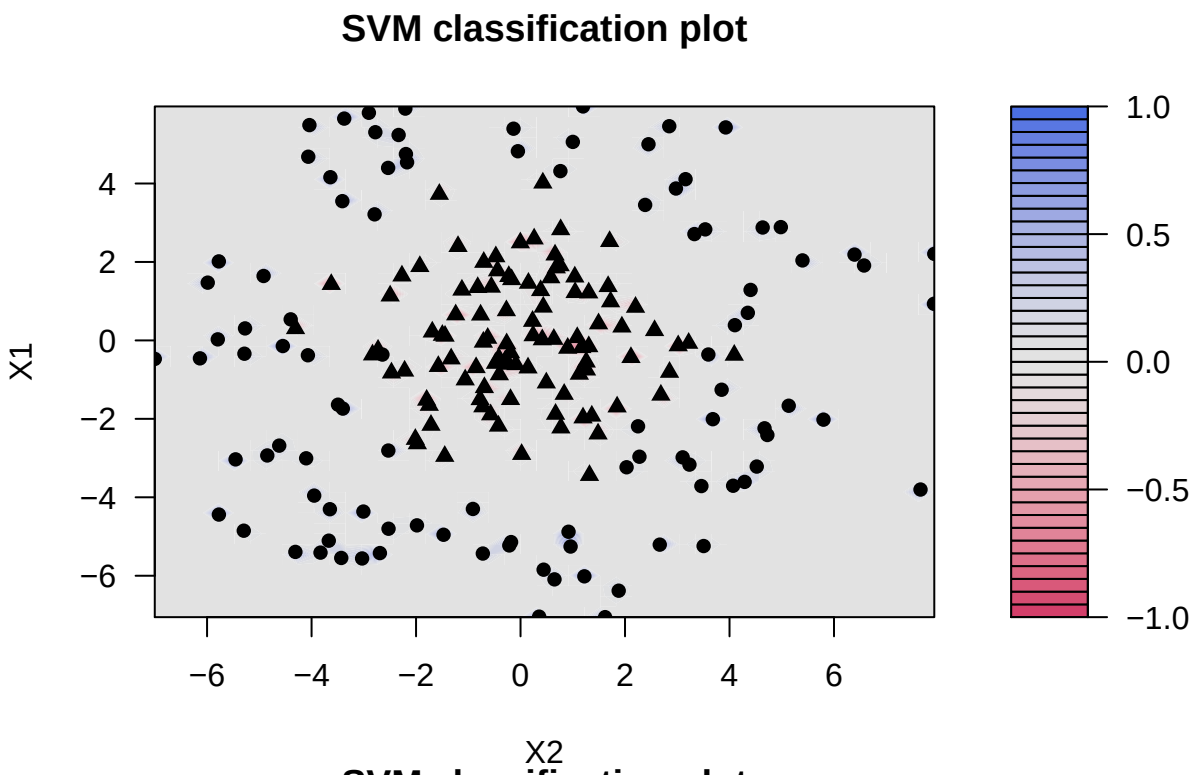

SVM classification plot
X1
X2
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
1.0
0.5
0.0
−0.5
−1.0

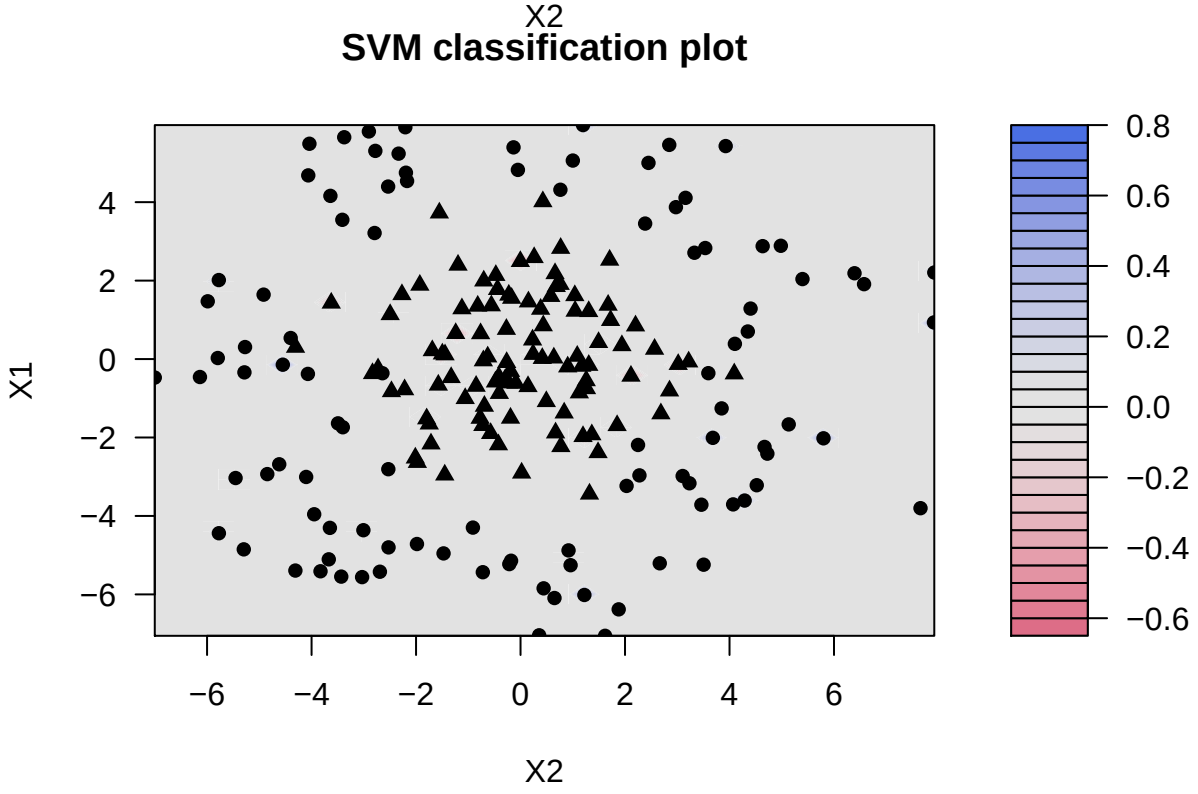

SVM classification plot
X1
X2
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
4
3
2
1
0
−1


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





Comment:

Degree:

The first 4 is for df1, and the last 4 is for df2. . . .

The first 4 does not help, since this is linearly drawn.

The last 4 does not help either, but theoretically speaking, I guess it should... ??

Gamma:

The first 7 is for df1, and the last 7 is for df2....

The first 7 shows contour diviation between two groups... But, I am not sure what they mean... Also, I don't understand the difference between black and empty dots...

The last 7 shows the similar thing!

I think this method is useful for both data sets!!!

ROC curve

Q. random classifier is 45 degree iff threshold is 0.5?? But, if it is bigger than 0.5, then the line will be less slanted, but if it is smaller than 0.5, then it will be steeper??

Q. For predictionc inside of prediction function, do we include posterior of 0 or 1???

Q. Here, lda does perform better than random classifier for around half of the times, but less than it half of the times... Then what do we say...???

Q. But how do we know whether lda will perform well on a df1 but not on a df2 without doing these tests intuitively??

```
set.seed(100)
head(df2)
```

```
##           X1         X2 y
## 1  0.1212625  0.2403716 0
## 2 -1.6940727  1.8486692 0
## 3 -0.7799670 -2.2187340 0
## 4  0.9858265  1.7233923 0
## 5  0.1319833 -1.4964557 0
## 6 -2.1611704 -1.7139424 0
```

```
train <- createDataPartition(1:nrow(df2), p = 0.7)
length(train[[1]])
```

```
## [1] 140
```

```
test <- c(1:nrow(df2))[-train[[1]]]

trainset <- df2[train[[1]],]
testset <- df2[test, ]


ldas <- lda(y~., data = trainset)
predict(ldas, testset)$class
```

```
##  [1] 0 0 0 1 0 0 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 1 1 0 1 0 1 1 1 0 1 0 1
## [36] 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 0 0 0 1 0
## Levels: 0 1
```

```r
predict(ldas, testset)$posterior
```

```
##             0         1
## 1   0.5226463 0.4773537
## 2   0.5166625 0.4833375
## 4   0.5723523 0.4276477
## 8   0.4841151 0.5158849
## 9   0.5686075 0.4313925
## 14  0.5508010 0.4491990
## 20  0.4313526 0.5686474
## 26  0.4449693 0.5550307
## 31  0.6046586 0.3953414
## 37  0.5025964 0.4974036
## 39  0.5569302 0.4430698
## 40  0.5258948 0.4741052
## 42  0.6102165 0.3897835
## 44  0.4869345 0.5130655
## 47  0.4179641 0.5820359
## 51  0.5293108 0.4706892
## 52  0.4613685 0.5386315
## 53  0.5032568 0.4967432
## 61  0.5619089 0.4380911
## 63  0.5388875 0.4611125
## 64  0.5630004 0.4369996
## 68  0.4995576 0.5004424
## 71  0.5499794 0.4500206
## 72  0.4706183 0.5293817
## 74  0.4627011 0.5372989
## 76  0.5458869 0.4541131
## 77  0.4873255 0.5126745
## 78  0.5766609 0.4233391
## 79  0.4163608 0.5836392
## 90  0.4473987 0.5526013
## 105 0.4977521 0.5022479
## 106 0.5684263 0.4315737
## 112 0.4736657 0.5263343
## 114 0.5501823 0.4498177
## 115 0.3725528 0.6274472
## 116 0.7041591 0.2958409
## 118 0.6406230 0.3593770
## 126 0.5134738 0.4865262
## 130 0.3596903 0.6403097
## 133 0.4276953 0.5723047
## 140 0.5916236 0.4083764
## 141 0.6859340 0.3140660
## 142 0.6854519 0.3145481
## 144 0.6373572 0.3626428
## 147 0.5819663 0.4180337
## 155 0.6754360 0.3245640
## 157 0.5916052 0.4083948
## 160 0.3708327 0.6291673
## 162 0.4013326 0.5986674
## 168 0.3776733 0.6223267
## 173 0.3601771 0.6398229
```

```
## 174 0.6326441 0.3673559
## 176 0.3282118 0.6717882
## 182 0.3996218 0.6003782
## 183 0.4476364 0.5523636
## 191 0.5724209 0.4275791
## 193 0.6917067 0.3082933
## 195 0.6646811 0.3353189
## 196 0.3529602 0.6470398
## 199 0.6276984 0.3723016
```

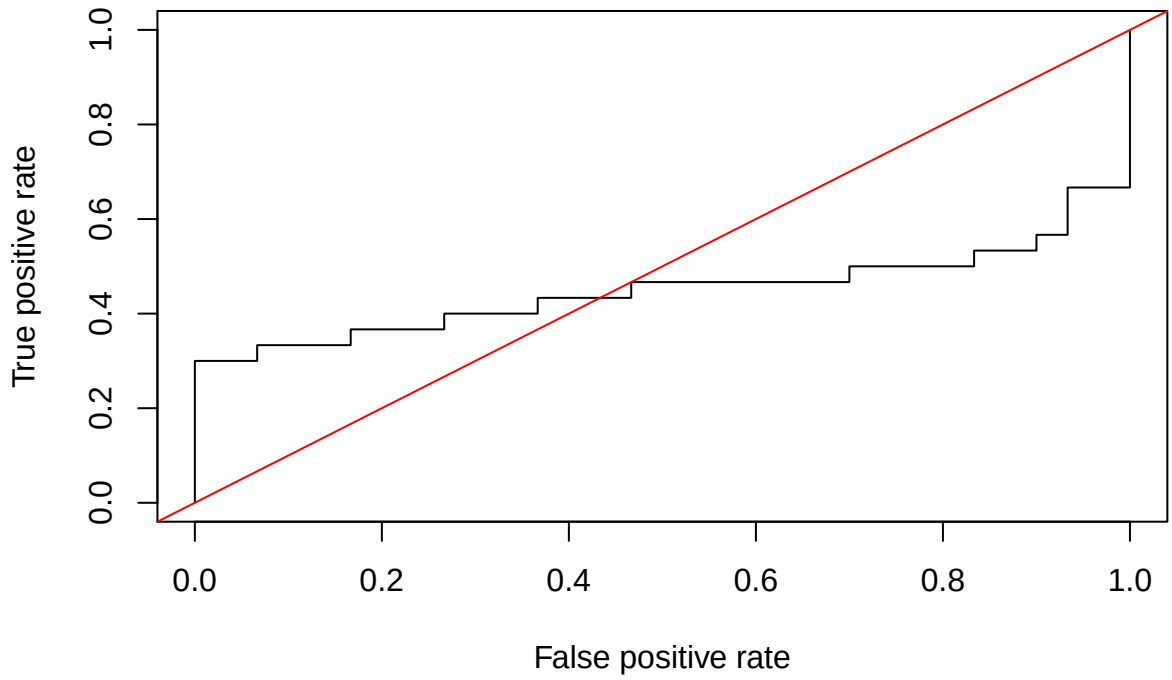
```
mean(predict(ldas, testset)$class != testset[,3])
```

```
## [1] 0.4833333
```

```
table(predict(ldas, testset)$class, testset[,3])
```

```
##
##      0  1
##   0 18 17
##   1 12 13
```

```
predict <- prediction(predict(ldas, testset)$posterior[,2], testset[,3])

roc <- performance(predict, measure = "tpr", x.measure = "fpr")
plot(roc)
abline(0,1, col = "red")
```




```
auc <- performance(predict, measure = "auc")
auc@y.values
```

```
## [[1]]
## [1] 0.4477778
```

```
#df1
```

```r
trainset2 <- df1[train[[1]],]
testset2 <- df1[test, ]

ldas2 <- lda(y~., data = trainset2)
predict(ldas2, testset2)$class
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1
## [36] 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## Levels: 0 1
```

```r
predict(ldas2, testset2)$posterior
```

```
##                0            1
## 1   1.000000e+00 5.327402e-09
## 2   1.000000e+00 3.327611e-08
## 4   9.999723e-01 2.771747e-05
## 8   9.999999e-01 6.398756e-08
## 9   1.000000e+00 3.189007e-14
## 14  9.999999e-01 1.262193e-07
## 20  9.976353e-01 2.364660e-03
## 26  1.000000e+00 7.778774e-10
## 31  9.999998e-01 1.689710e-07
## 37  1.000000e+00 3.828137e-08
## 39  9.998791e-01 1.208829e-04
## 40  9.999837e-01 1.634943e-05
## 42  6.793631e-01 3.206369e-01
## 44  1.000000e+00 5.315070e-09
## 47  1.000000e+00 3.149872e-09
## 51  1.000000e+00 4.555233e-10
## 52  1.000000e+00 1.839347e-12
## 53  9.999999e-01 5.661123e-08
## 61  1.000000e+00 3.330674e-08
## 63  1.000000e+00 1.373927e-10
## 64  9.405628e-01 5.943718e-02
## 68  1.000000e+00 7.708801e-09
## 71  9.999968e-01 3.200225e-06
## 72  1.000000e+00 1.156023e-09
## 74  9.999547e-01 4.533586e-05
## 76  1.000000e+00 3.801179e-09
## 77  1.000000e+00 1.073889e-10
## 78  9.999996e-01 3.810043e-07
## 79  9.999388e-01 6.122614e-05
## 90  9.999999e-01 1.215791e-07
## 105 1.750630e-05 9.999825e-01
## 106 9.955409e-09 1.000000e+00
## 112 2.034463e-07 9.999998e-01
## 114 1.154221e-04 9.998846e-01
## 115 1.171094e-06 9.999988e-01
## 116 8.808470e-09 1.000000e+00
## 118 7.401812e-07 9.999993e-01
## 126 1.280416e-10 1.000000e+00
## 130 1.689567e-08 1.000000e+00
## 133 5.446763e-10 1.000000e+00
## 140 1.082130e-06 9.999989e-01
```

```
## 141 8.642039e-10 1.000000e+00
## 142 8.583164e-01 1.416836e-01
## 144 2.526787e-10 1.000000e+00
## 147 2.348301e-06 9.999977e-01
## 155 2.706226e-10 1.000000e+00
## 157 6.904213e-10 1.000000e+00
## 160 2.183626e-04 9.997816e-01
## 162 4.307978e-13 1.000000e+00
## 168 9.452302e-07 9.999991e-01
## 173 5.840221e-09 1.000000e+00
## 174 8.882135e-10 1.000000e+00
## 176 4.362661e-01 5.637339e-01
## 182 2.062270e-09 1.000000e+00
## 183 2.664155e-07 9.999997e-01
## 191 6.958093e-07 9.999993e-01
## 193 1.397187e-11 1.000000e+00
## 195 2.701622e-11 1.000000e+00
## 196 1.368279e-08 1.000000e+00
## 199 5.564993e-04 9.994435e-01
```

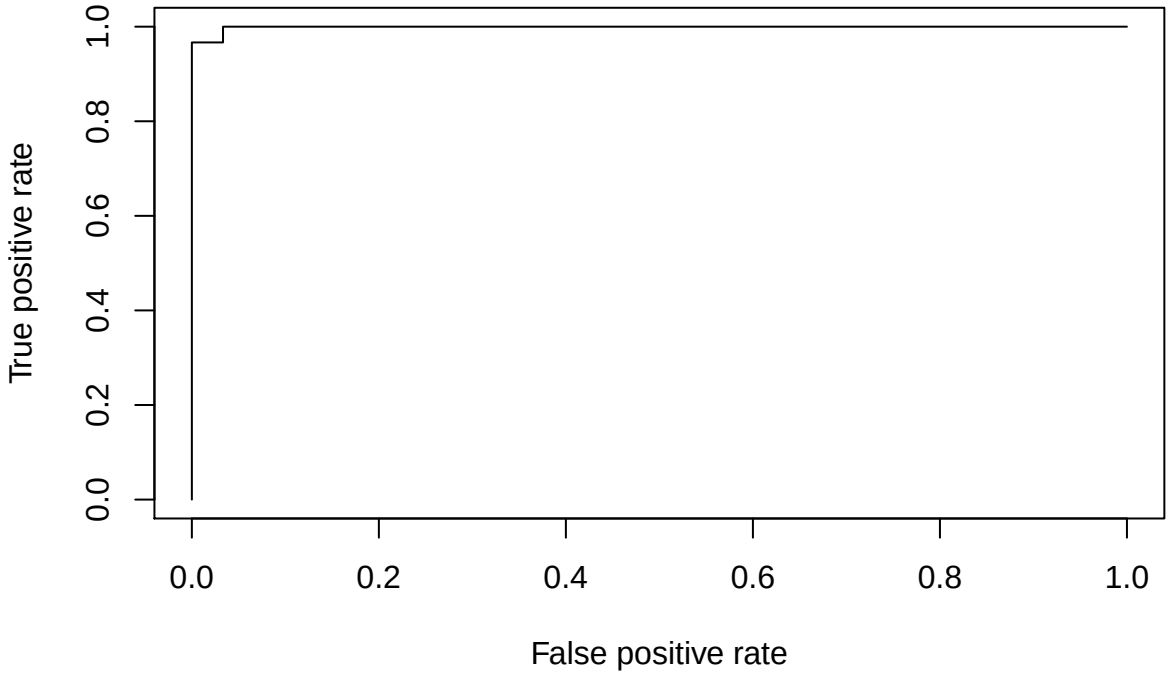
```
mean(predict(ldas2, testset2)$class != testset2[,3])
```

```
## [1] 0.01666667
```

```
table(predict(ldas2, testset2)$class, testset2[,3])
```

```
##
##      0  1
##   0 30  1
##   1  0 29
```

```
predict2 <- prediction(predict(ldas2, testset2)$posterior[,2], testset2[,3])

roc <- performance(predict2, measure = "tpr", x.measure = "fpr")
plot(roc)
```




```
auc2 <- performance(predict2, measure = "auc")
auc2@y.values
```

```
## [[1]]
## [1] 0.9988889
```