Introdução à ciência dos dados - Lista 3

Leonardo Lima - 14334311 Leonardo Makoto - 7180679

2023-05-22

Questão 1

Para o conjunto de dados Iris, use somente o comprimento de pétalas (X_1) e comprimento de sépalas (X_2) como preditores e a variável resposta Y = espécie (Setosa, Versicolor, Virginica). Construa uma árvore para classificação. Escreva com detalhes as regiões no plano e faça o gráfico da árvore e das regiões, usando o pacote **tree**. Obtenha a taxa de erro de classificação.

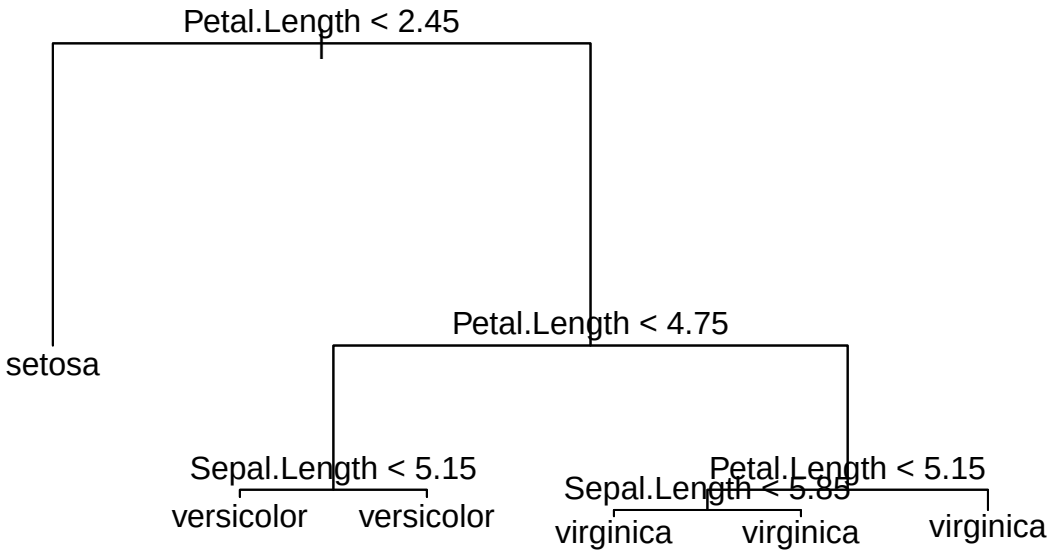
```
library(tree)
library(tidyverse)

# carregando a base de dados
data("iris")

# vamos criar criar a árvore de classificação

mod.iris <- tree(Species ~ Petal.Length + Sepal.Length, data = iris)

# vejamos os resultados da regressão
summary(mod.iris)
```

```
##
## Classification tree:
## tree(formula = Species ~ Petal.Length + Sepal.Length, data = iris)
## Number of terminal nodes:  6
## Residual mean deviance:  0.1818 = 26.17 / 144
## Misclassification error rate: 0.04667 = 7 / 150
```

De acordo com os resultados do modelo, a taxa de erro de classificação para o teste é aproximadamente 4,7%.

Para visualizar a árvore, podemos fazer o seguinte:

```
# criando a estrutura da árvore
plot(mod.iris)

# adicionaodo texto para a estrutura
text(mod.iris, pretty = 0)
```



Assim como é possível observar no gráfico, a árvore possui nós demais, que podem ser removidos (note que para todos os valores de Sepal.Lenght em que $2,45 < Petal.Lenght < 4,75$, a classificação prevista é versicolor, independente do valor de Sepal.Lenght).

Vamos realizar o processo de poda da árvore através do método de validação cruzada para avaliar se há melhora:

```
set.seed(123)
# vamos determinar que a taxa de erro de classificação servirá como guia para
# escolher o melhor parâmetro do modelo através da opção FUN

cv.mod.iris <- cv.tree(mod.iris, FUN = prune.misclass)

cv.mod.iris
```

```
## $size
## [1] 6 3 2 1
##
## $dev
## [1]   9   9  92 117
##
## $k
## [1] -Inf    0   43   50
##
## $method
## [1] "misclass"
##
```

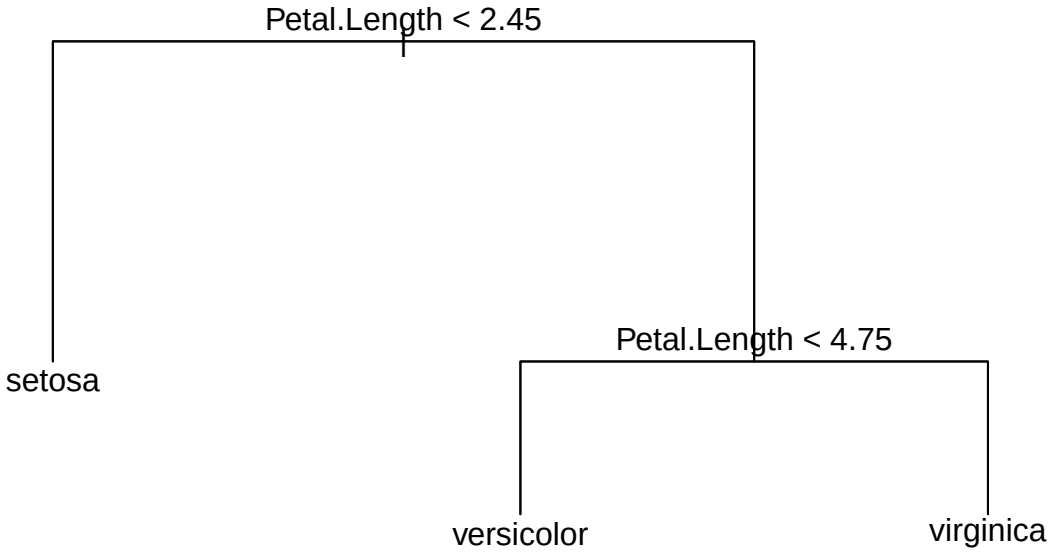
```
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

Os resultados mostram que as árvores com 3 e 6 nós terminais são as que possuem a menor quantidade de erros (9). Dessa maneira, vamos selecionar a mais simples (3 nós) e criar a nova classificação com essa limitação.

```
# nova classificação
poda.iris <- prune.misclass(mod.iris, best = 3)

# vejamos a taxa de erro e o gráfico dessa nova árvore
summary(poda.iris)
```

```
##
## Classification tree:
## snip.tree(tree = mod.iris, nodes = 6:7)
## Variables actually used in tree construction:
## [1] "Petal.Length"
## Number of terminal nodes:  3
## Residual mean deviance:  0.3231 = 47.5 / 147
## Misclassification error rate: 0.04667 = 7 / 150
```

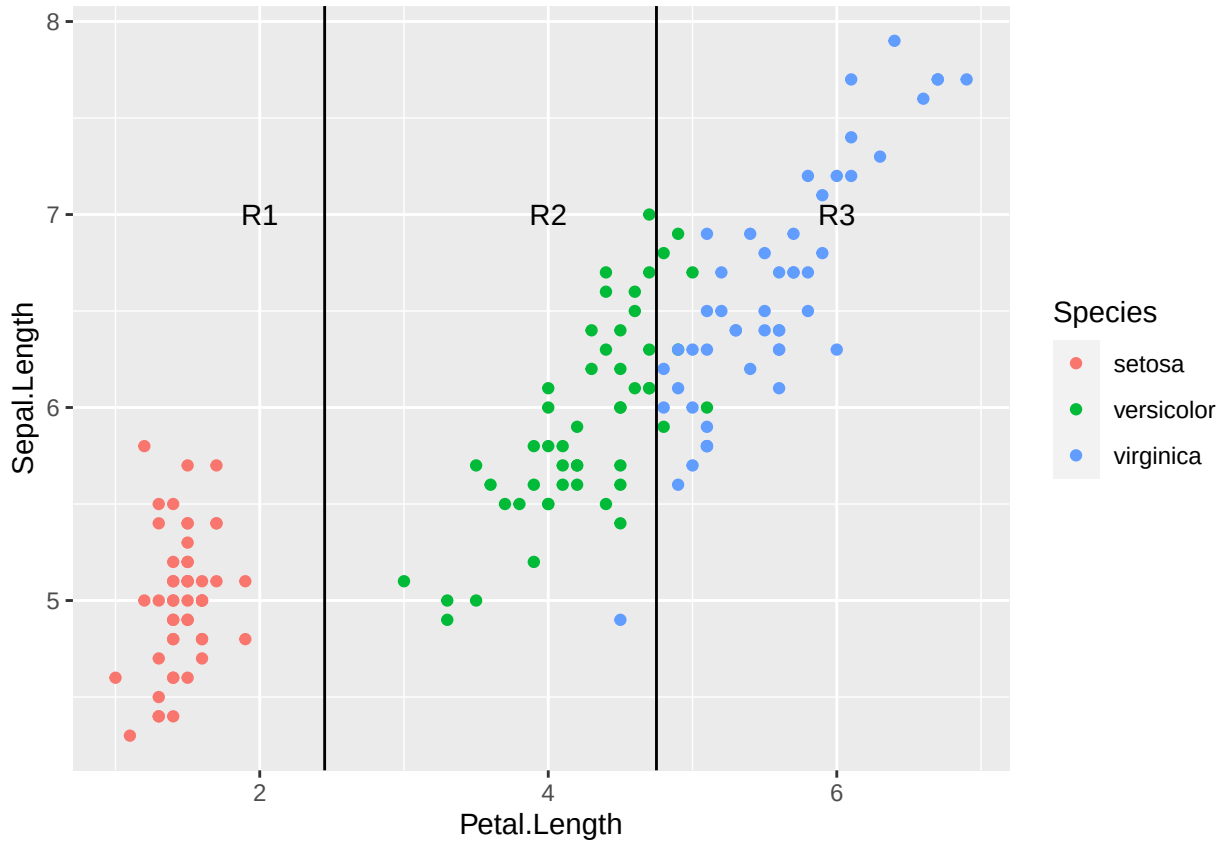
```
# gráfico
plot(poda.iris)
text(poda.iris, pretty = 0)
```



Comparativamente ao caso anterior, a taxa de erros dessa nova árvore é idêntica (4,7%), mas há um ganho significativo em termos de simplificação do critério de seleção.

Vejamos o gráfico das regiões de classificação:

```
# vamos criar um gráfico com as regiões de classificação dessa árvore
ggplot(data = iris, aes(x = Petal.Length, y = Sepal.Length, color = Species)) +
  geom_point() +
  geom_vline(xintercept = c(2.45, 4.75)) +
  annotate("text", x = 2, y = 7, label = "R1") +
  annotate("text", x = 4, y = 7, label = "R2") +
  annotate("text", x = 6, y = 7, label = "R3")
```



Portanto, há 3 regiões de classificação para esta árvore:

$$R_1 = \{X : X_1 < 2,45\}, \text{ o valor previsto para Species é Setosa}$$
$$R_2 = \{X : 2,45 \leq X_1 < 4,75\}, \text{ o valor previsto para Species é Versicolor}$$
$$R_3 = \{X : X_1 \geq 4,75\}, \text{ o valor previsto para Species é Virginica}$$

Questão 2

Considere o conjunto de dados **rehabcardio**, sendo preditores $X_1 = HDL$, $X_2 = LDL$, $X_3 = Trigl$, $X_4 = Glicose$ e $X_5 = Peso$ e resposta $Y =$ Diabetes (presente $= 1$, ausente $= 0$). Utilize um subconjunto

em que as amostras têm todas as medidas completas. Construa árvores usando bagging e floresta aleatória. Usando a taxa de erro de classificação, escolha o melhor classificador.

Questão 3

Considere as variáveis Altura e Idade da Tabela 12.1 do Capítulo 12 (Análise de Agrupamentos):

item a

Obtenha os agrupamentos usando o método hierárquico, até um ponto que você considere adequado, usando a distância Euclidiana e o método do centróide. Obtenha o dendrograma correspondente.

---

item b

Refaça o item (a) usando a distância L1 (Manhattan).

---

item c

Use o algoritmo K-médias, com $K = 3$ para obter os grupos para os mesmos dados. Comente o resultado. Qual é o centróide de cada grupo?

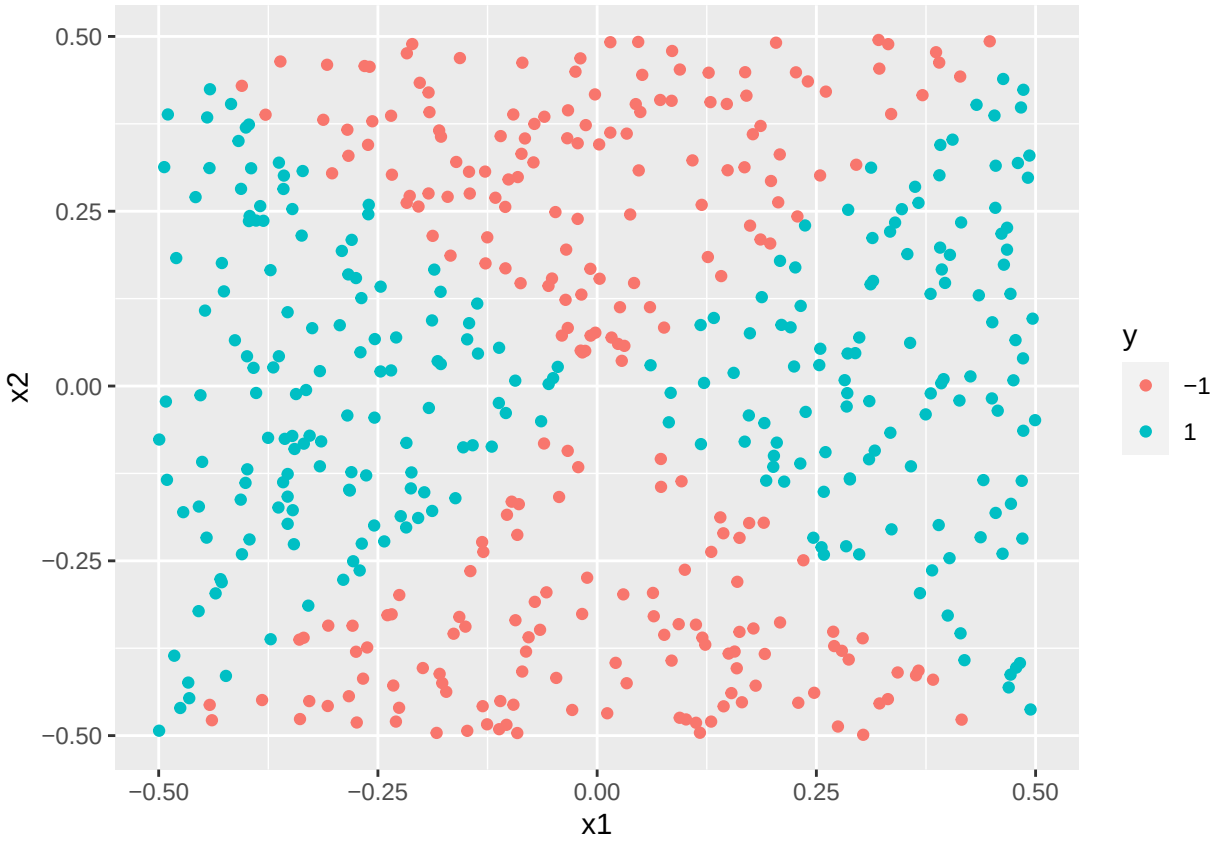
Questão 4

```r
# Simulando os dados
set.seed(123)

# criando as preditoras
x1 <- runif(500)-0.5
x2 <- runif(500)-0.5

# criando a variável y
y <- 1* (x1^2 - x2^2 > 0)

# substituindo o 0 por -1
y <- ifelse(y == 0, -1, y)

# transformando em base de dados
base <- data.frame(x1,x2,y = as.factor(y))
```

Item a

Faça um gráfico das observações, com símbolos (ou cores) de acordo com cada classe.

```
# vamos criar o gráfico das observações
ggplot(aes(x = x1, y = x2, colour = y), data = base) +
  geom_point()
```




```
rm(x1,x2,y)
```

item b

Separe os dados em conjunto de treinamento e de teste. Obtenha o classificador de margem máxima, tendo X1 e X2 como preditores. Obtenha as previsões para o conjunto de teste e a acurácia do classificador.

```
# carregando o pacote para realizar o sum
set.seed(123)

# vamos criar a amostra de treino com 80% das observações da base
```

```r
treino <- sample(500, 500*0.8, replace = FALSE)

library(e1071)
```

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

```r
# vamos calibrar o modelo de margem máxima:

svm_unif <- svm(y ~ x1 + x2, data = base[treino,],
                type = "C-classification", kernel = "linear")

# vejamos os resultados

summary(svm_unif)
```

```
##
## Call:
## svm(formula = y ~ x1 + x2, data = base[treino, ], type = "C-classification",
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  390
##
##  ( 197 193 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

Vejamos agora as previsões para o conjunto de teste

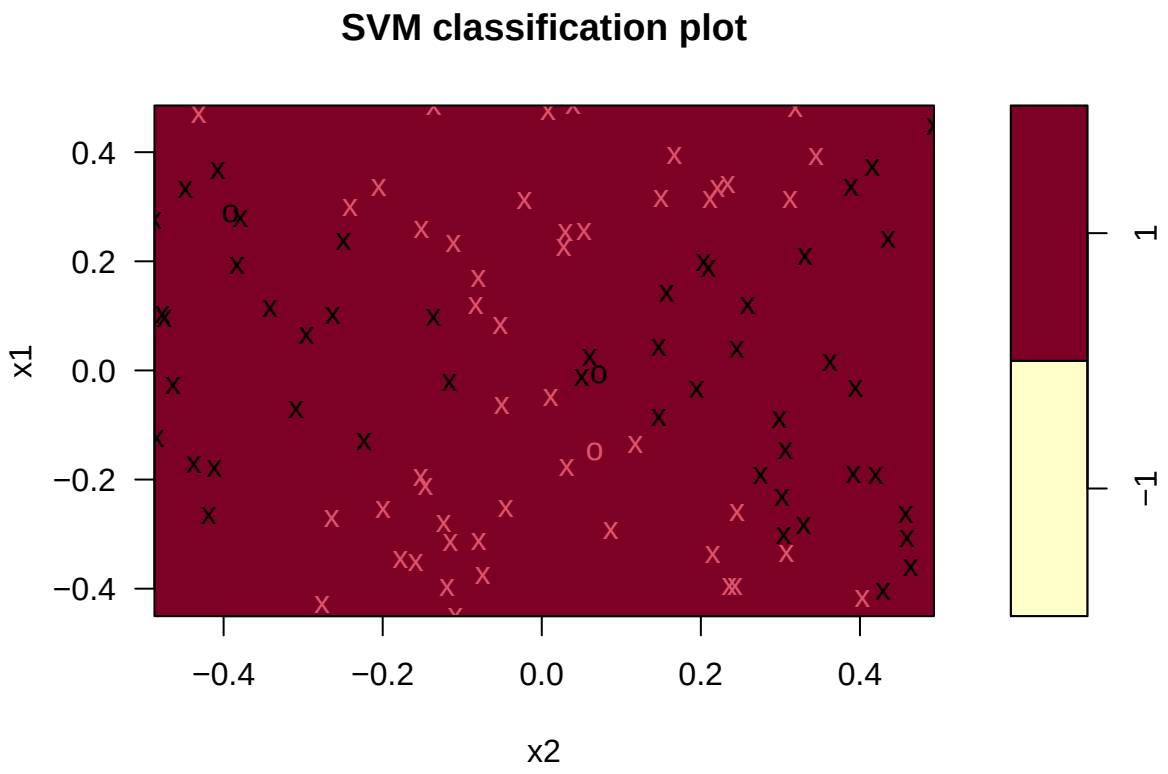
```r
# previsões para o conjunto de teste
svm_pred <- predict(svm_unif, base)[-treino]

# vejamos a acurácia do classificador através da matriz de confusão:
table(svm_pred,
      base$y[-treino])
```

```
##
## svm_pred -1  1
##       -1  0  0
##       1  51 49
```

```r
# a acurácia é: 49/100 = 49%
```

Como é possível perceber, o algorítmo classifica todos como 1. Vejamos o gráfico

```r
# gráfico
plot(svm_unif, base[-treino,])
```



Os vetores de suporte são os x e a região em vermelho é a que classifica em 1. Como podemos ver, praticamente todas as observações são vetores de suporte e toda a região designa para 1, indicando que o modelo não é adequado. Os acertos são fruto do acaso.

item c

Obtenha o classificador de margem flexível, tendo X1 e X2 com preditores. Obtenha as previsões para o conjunto de teste e a taxa de erros de classificação.

```r
# vamos encontrar o parâmetro gamma e custo para fazer usar a margem flexível:
set.seed(123)

tune <- tune.svm(y ~ x1 + x2, data = base[treino,],
    cost = 2^(2:5), gamma = 2^(-2:2))
```

```r
# vendo os melhores parâmetros
summary(tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##      4   16
##
## - best performance: 0.0075
##
## - Detailed performance results:
##    gamma cost  error dispersion
## 1   0.25    4 0.0275 0.03216710
## 2   0.50    4 0.0225 0.02993047
## 3   1.00    4 0.0225 0.02188988
## 4   2.00    4 0.0225 0.02188988
## 5   4.00    4 0.0275 0.01419116
## 6   0.25    8 0.0200 0.03073181
## 7   0.50    8 0.0275 0.03216710
## 8   1.00    8 0.0150 0.01748015
## 9   2.00    8 0.0225 0.02188988
## 10  4.00    8 0.0175 0.02371708
## 11  0.25   16 0.0175 0.02371708
## 12  0.50   16 0.0125 0.01767767
## 13  1.00   16 0.0200 0.01972027
## 14  2.00   16 0.0150 0.01290994
## 15  4.00   16 0.0075 0.01687371
## 16  0.25   32 0.0100 0.01748015
## 17  0.50   32 0.0125 0.01767767
## 18  1.00   32 0.0150 0.01290994
## 19  2.00   32 0.0125 0.01767767
## 20  4.00   32 0.0125 0.01767767
```

Os melhores parâmetros são: custo = 16 e gamma = 4. Como pretendemos usar kernel linear, gamma é necessariamente 0,5. Vamos inserir no modelo de margem flexível o valor do custo encontrado:

```r
# Modelo de margem flexível:
set.seed(123)

svm_unif_flex <- svm(y ~ x1 + x2, data = base[treino,],
                     kernel = "linear", cost = 16)

# vejamos os resultados

summary(svm_unif_flex)
```

```
##
## Call:
```

```
## svm(formula = y ~ x1 + x2, data = base[treino, ], kernel = "linear",
##     cost = 16)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  16
##
## Number of Support Vectors:  391
##
##  ( 198 193 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

Vejamos a acurácia do classificador:

```
# previsões para o conjunto de teste
svm_pred_flex <- predict(svm_unif_flex, base)[-treino]

# vejamos a acurácia do classificador através da matriz de confusão:
table(svm_pred_flex,
      base$y[-treino])
```

```
##
## svm_pred_flex -1  1
##            -1  0  0
##            1  51 49
```

```
# a acurácia é: 49/100 = 49%
```

Novamente, o classificador atribui todos ao valor 1. A taxa de erros de classificação será 1 - acurácia. Ou seja, 51%.

item d

Obtenha o classificador de margem não linear, usando um kernel apropriado. Calcule a taxa de erros de classificação.

Façamos a estimação usando o kernel do tipo polinômio:

```
# vamos encontrar o grau do polinômio e o parâmetro custo para usar o classificador de margem não linea
set.seed(123)
tune2 <- tune.svm(y ~ x1 + x2, data = base[treino,],
    cost = 2^(2:8),
```

```
    kernel = "polynomial", degree = 1:4)

# vendo os melhores parâmetros
summary(tune2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  degree cost
##       2   16
##
## - best performance: 0.01
##
## - Detailed performance results:
##    degree cost  error dispersion
## 1       1    4 0.4900 0.03374743
## 2       2    4 0.0150 0.02415229
## 3       3    4 0.4475 0.06503204
## 4       4    4 0.0800 0.04684490
## 5       1    8 0.4900 0.03374743
## 6       2    8 0.0150 0.02415229
## 7       3    8 0.4450 0.06749486
## 8       4    8 0.0700 0.04048319
## 9       1   16 0.4900 0.03374743
## 10      2   16 0.0100 0.01748015
## 11      3   16 0.4475 0.06816035
## 12      4   16 0.0500 0.04082483
## 13      1   32 0.4900 0.03374743
## 14      2   32 0.0150 0.02415229
## 15      3   32 0.4475 0.06816035
## 16      4   32 0.0400 0.01748015
## 17      1   64 0.4900 0.03374743
## 18      2   64 0.0150 0.02415229
## 19      3   64 0.4475 0.06816035
## 20      4   64 0.0450 0.02838231
## 21      1  128 0.4900 0.03374743
## 22      2  128 0.0125 0.01767767
## 23      3  128 0.4475 0.06816035
## 24      4  128 0.0425 0.02058182
## 25      1  256 0.4900 0.03374743
## 26      2  256 0.0125 0.01767767
## 27      3  256 0.4450 0.06952218
## 28      4  256 0.0250 0.01178511
```

Os melhores parâmetros são: custo = 16 e grau do polinômio= 2. Lembrando que o parâmetro gamma não é utilizado para o kernel polinomial, apenas para o radial.

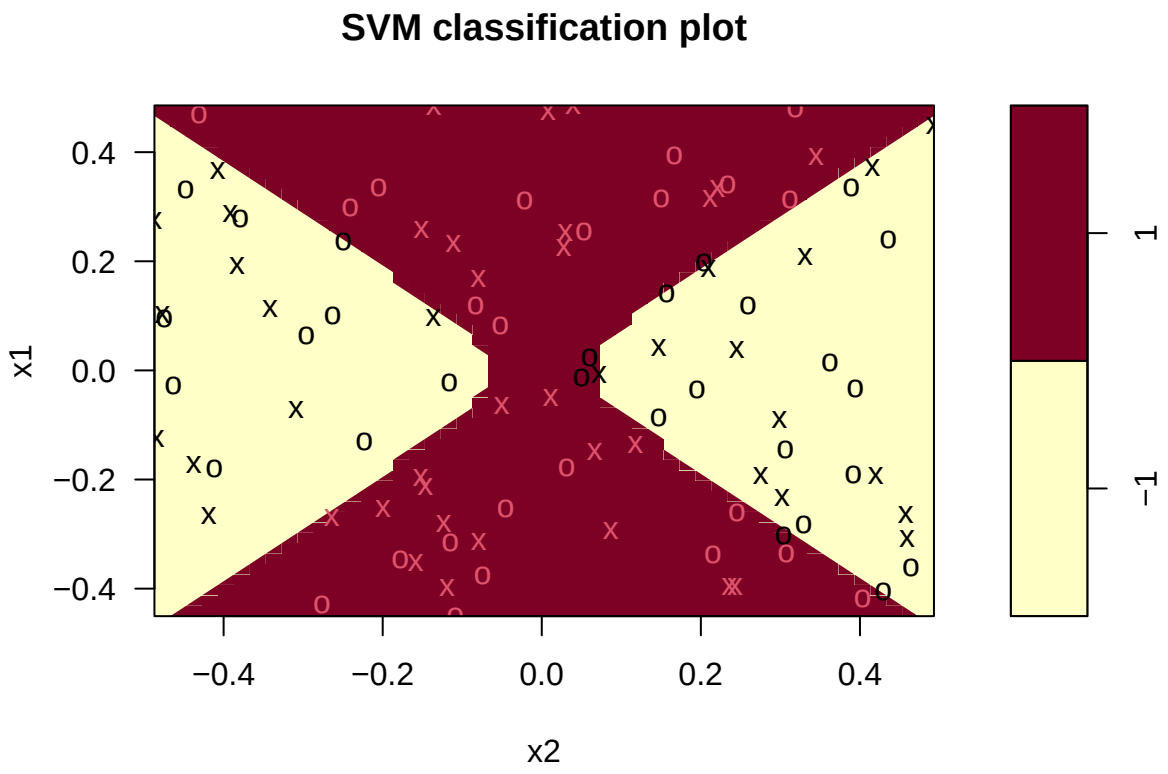
```
set.seed(123)

# Modelo de margem não linear:
```

```r
svm_unif_n_linear <- svm(y ~ x1 + x2, data = base[treino,],
                     kernel = "polynomial",type = "C-classification",
                     degree = 2, cost = 16, scale = F)

# vejamos os resultados
summary(svm_unif_n_linear)
```

```
##
## Call:
## svm(formula = y ~ x1 + x2, data = base[treino, ], kernel = "polynomial",
##     type = "C-classification", degree = 2, cost = 16, scale = F)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  16
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  196
##
##  ( 98 98 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

Os resultados mostram que há 196 vetores de suporte. Vamos plotar o gráfico com o modelo vs observações de teste para ver a performance:

```r
plot(svm_unif_n_linear, base[-treino,] )
```



Pelo gráfico, podemos observar que o método é mais adequado para a classificação. Vejamos os resultados de acurácia e taxa de erros:

```
# previsões para o conjunto de teste
svm_pred_n_linear <- predict(svm_unif_n_linear, base)[-treino]

# vejamos a acurácia do classificador através da matriz de confusão:
table(svm_pred_n_linear,
      base$y[-treino])
```

```
##
## svm_pred_n_linear -1  1
##                -1 43  0
##                 1  8 49
```

A acurácia é: 49+43/100 = 92%. Já a taxa de erros é 1 - acurácia: 8%.

Vamos fazer os mesmos testes usando kernel radial:

```
set.seed(123)

tune3 <- tune.svm(y ~ x1 + x2, data = base[treino,],
    cost = 2^(2:8),
    kernel = "radial", gamma = 2^(1:6))

# vendo os melhores parâmetros
summary(tune3)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##     16    4
##
## - best performance: 0.0075
##
## - Detailed performance results:
##    gamma cost  error dispersion
## 1      2    4 0.0225 0.02188988
## 2      4    4 0.0275 0.01419116
## 3      8    4 0.0175 0.02371708
## 4     16    4 0.0075 0.01687371
## 5     32    4 0.0175 0.01687371
## 6     64    4 0.0250 0.02041241
## 7      2    8 0.0225 0.02188988
## 8      4    8 0.0175 0.02371708
## 9      8    8 0.0100 0.01748015
## 10    16    8 0.0100 0.01748015
## 11    32    8 0.0200 0.01972027
## 12    64    8 0.0250 0.02041241
## 13     2   16 0.0150 0.01290994
## 14     4   16 0.0075 0.01687371
## 15     8   16 0.0125 0.01767767
## 16    16   16 0.0100 0.01748015
## 17    32   16 0.0200 0.01972027
## 18    64   16 0.0250 0.02041241
## 19     2   32 0.0125 0.01767767
## 20     4   32 0.0125 0.01767767
## 21     8   32 0.0125 0.01767767
## 22    16   32 0.0100 0.01748015
## 23    32   32 0.0200 0.01972027
## 24    64   32 0.0250 0.02041241
## 25     2   64 0.0100 0.01290994
## 26     4   64 0.0150 0.01748015
## 27     8   64 0.0125 0.01767767
## 28    16   64 0.0100 0.01748015
## 29    32   64 0.0200 0.01972027
## 30    64   64 0.0250 0.02041241
## 31     2  128 0.0150 0.02108185
## 32     4  128 0.0150 0.01748015
## 33     8  128 0.0125 0.01767767
## 34    16  128 0.0100 0.01748015
## 35    32  128 0.0200 0.01972027
## 36    64  128 0.0250 0.02041241
## 37     2  256 0.0175 0.02058182
## 38     4  256 0.0150 0.01748015
## 39     8  256 0.0125 0.01767767
## 40    16  256 0.0100 0.01748015
## 41    32  256 0.0200 0.01972027
```

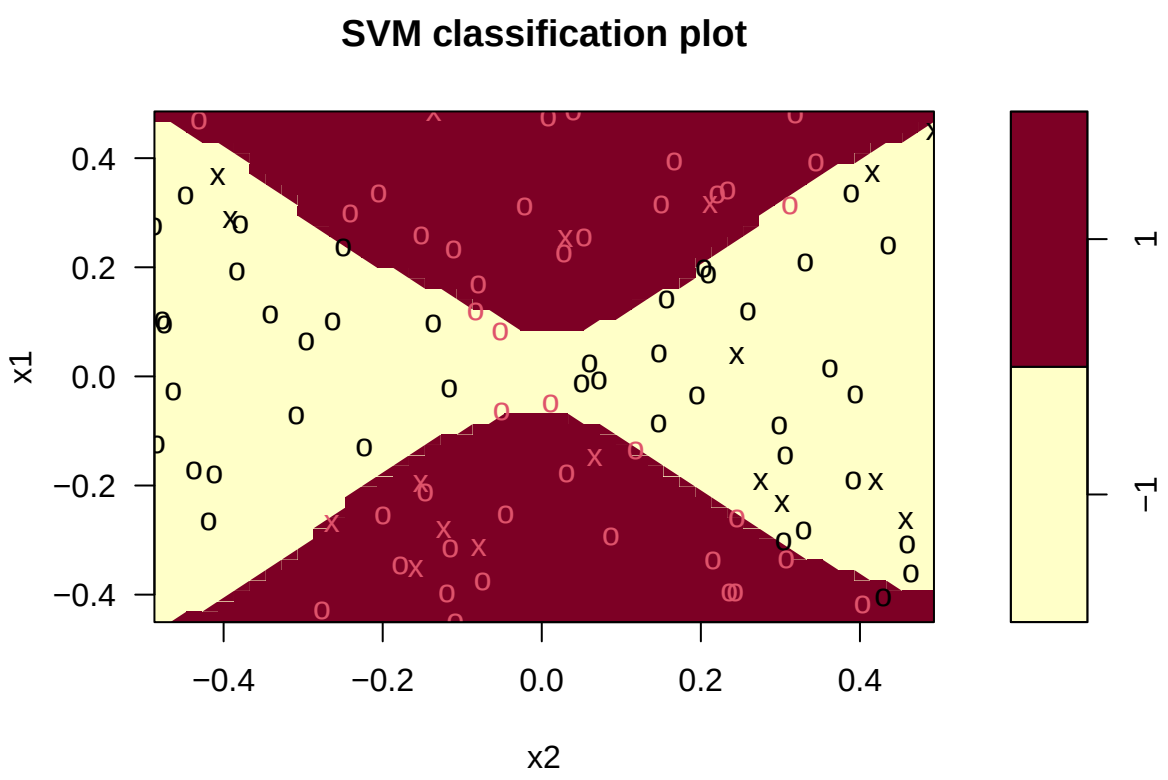
```
## 42    64  256 0.0250 0.02041241
```

```
# Modelo de margem não linear:

svm_unif_radial <- svm(y ~ x1 + x2, data = base[treino,],
                       kernel = "radial",type = "C-classification",
                       gamma = 16, cost = 4, scale = F)

# vejamos os resultados

summary(svm_unif_radial)
```

```
##
## Call:
## svm(formula = y ~ x1 + x2, data = base[treino, ], kernel = "radial",
##     type = "C-classification", gamma = 16, cost = 4, scale = F)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  4
##
## Number of Support Vectors:  76
##
##  ( 39 37 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

Vamos ver como fica o gráfico do radial:

```
plot(svm_unif_radial, base[-treino,] )
```



A figura mostra uma classificação menos adequada do que no caso do polinômio - pontos vermelhos sobre a área amarela são mais frequentes. Vamos ver a taxa de erros do modelo:

```
# previsões para o conjunto de teste
svm_pred_radial <- predict(svm_unif_radial, base)[-treino]

# vejamos a acurácia do classificador através da matriz de confusão:
table(svm_pred_radial,
      base$y[-treino])
```

```
##
## svm_pred_radial -1  1
##              -1 47  4
##              1   4 45
```

Nesse caso, a taxa de erro e acurácia de ambos os modelos são iguais. Olhando a acurácia balanceada, o modelo polinomial apresenta acurácia de 92,1%, enquanto o radial de aprox. 92%. Sendo assim, iremos usar o modelo polinomial como referência para o item e.

item e

Compare os dois classificadores.

Assim como apresentado nos itens anteriores, a tentativa de estimar um modelo linear em um cenário cujas classes apresentem uma fronteira de decisão não linear leva a resultados consideravelmente ruins. No caso em questão, os modelos lineares previam que todas as observações pertenciam apenas a uma classe, y = 1, independentemente se a margem linear considerada era flexível ou não.

Como consequência da aplicação do modelo inadequado, a taxa de erros dos dois modelos lineares era próxima de 50%, i.e., resultados próximos a probabilidade a priori de cada classe.

Ao estimar o modelo de margem não linear, adequado para o padrão dos dados, a acurácia aumentou consideravelmente e a taxa de erros caiu para aproximadamente 8%. Isso é reflexo da melhor adequação do modelo ao formato dos dados, que agora passa a ter as duas classificações na previsão.