Lab 14

Daniel Tshiani

2025-06-18

```
library(e1071)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
load("../data/Auto-3.rda")
attach(Auto)
```

```
## The following object is masked from package:lubridate:
##
##     origin
##
## The following object is masked from package:ggplot2:
##
##     mpg
```

a

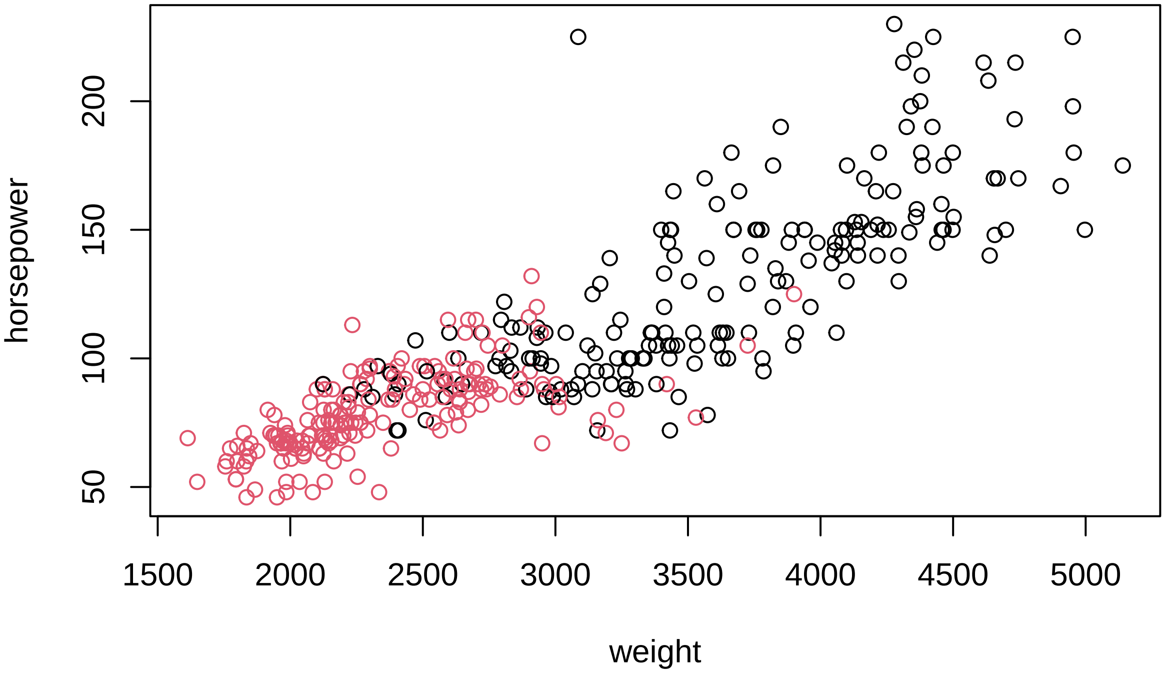
```
ECO = ifelse( mpg > 22.75, "Economy", "Consuming")
Auto$ECO <- as.factor(ECO)
rm(ECO)
attach(Auto)
```

```
## The following objects are masked from Auto (pos = 3):
##
##     acceleration, cylinders, displacement, horsepower, mpg, name,
##     origin, weight, year

## The following object is masked from package:lubridate:
##
##     origin

## The following object is masked from package:ggplot2:
##
##     mpg
```

```
svm <- svm(ECO ~., data = Auto)
svm
```

```
##
## Call:
## svm(formula = ECO ~ ., data = Auto)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  174
```

```
plot(weight, horsepower, col = as.numeric(Auto$ECO))
```



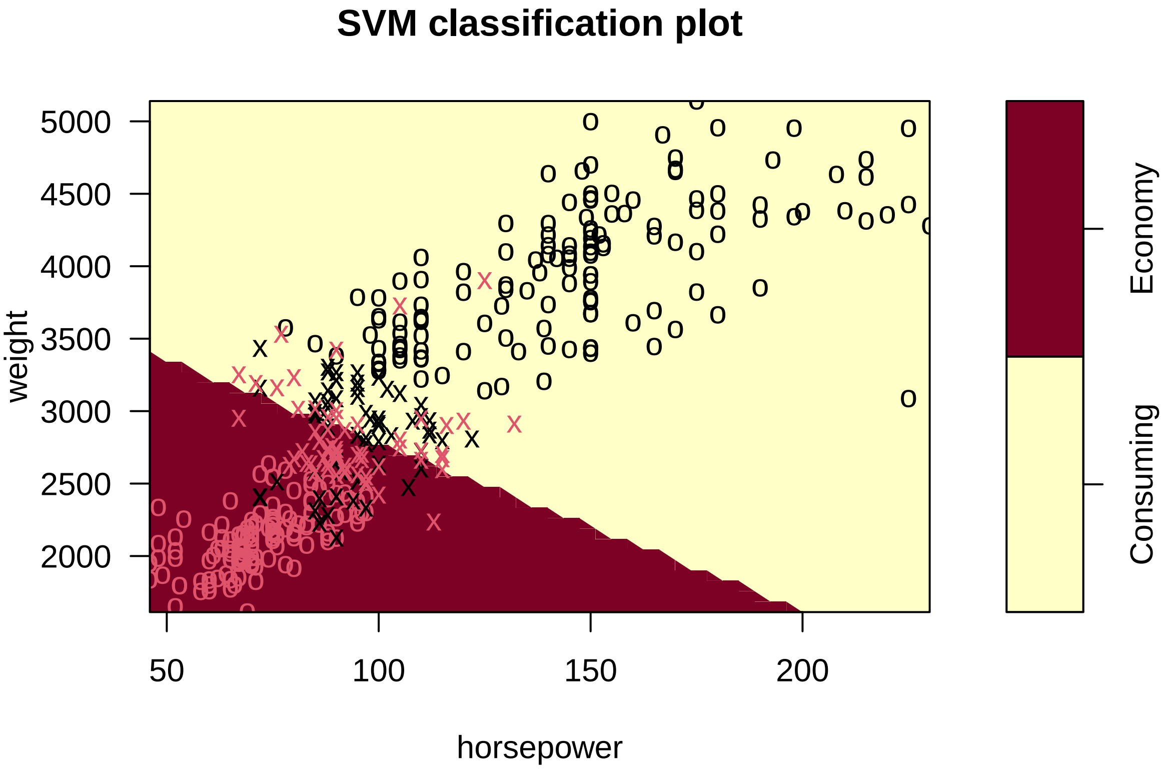
in the visualization we can see that the 2 classes overlap, so SVM is not a good option here.

b

```
d = data.frame(ECO, weight, horsepower)
svm <- svm(ECO ~., data = d, kernel = "linear")
summary(svm)
```

```
##
## Call:
## svm(formula = ECO ~ ., data = d, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  120
##
##  ( 60 60 )
##
##
## Number of Classes:  2
##
## Levels:
##  Consuming Economy
```

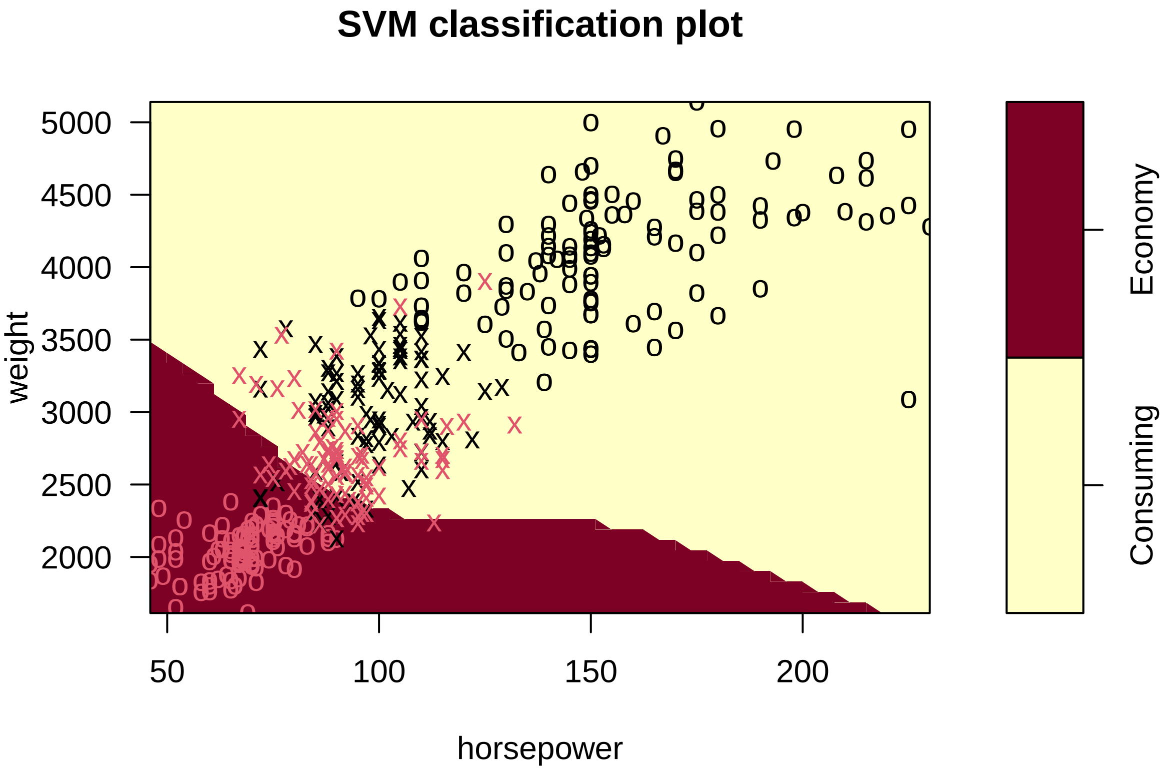
```
plot(svm, data = Auto[, c(4, 5, 10)])
```



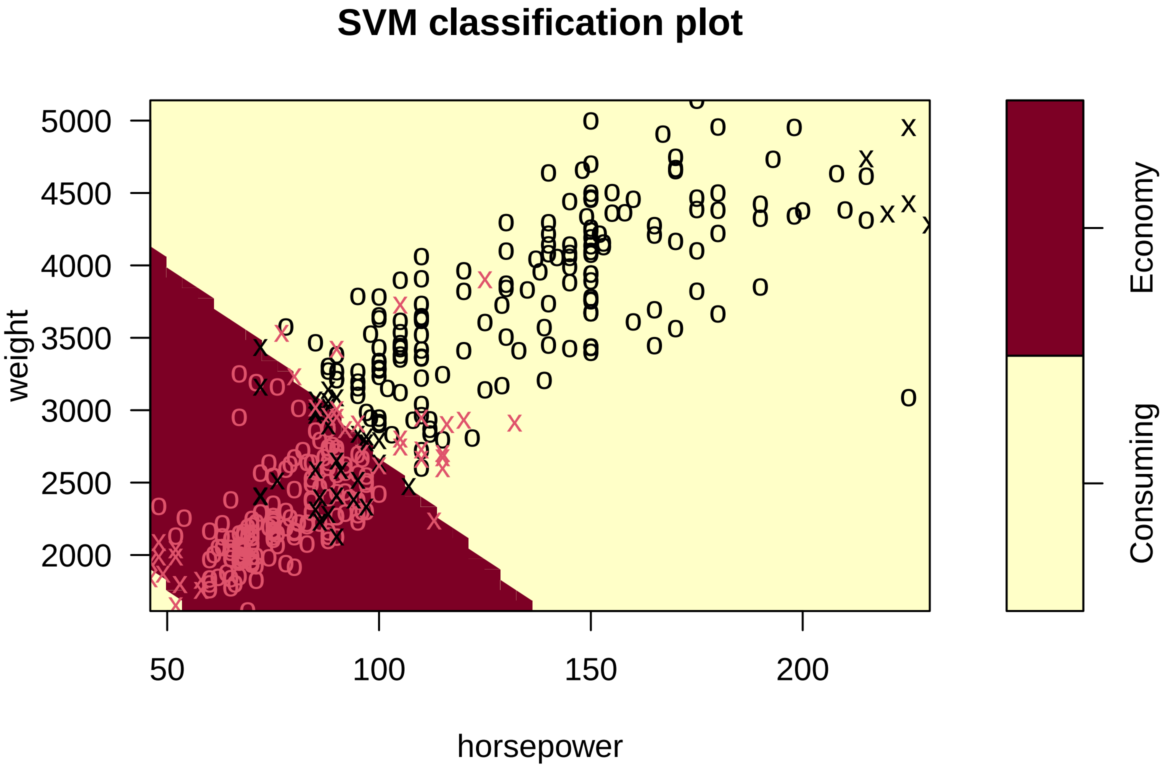
the hyperplane does a better job in seperating the different class ECO but there is still some overlap.

c

```
svm_poly <- svm(ECO ~., data = d, kernel = "polynomial")
plot(svm_poly, data = Auto[, c(4, 5, 10)])
```



```
svm_s <- svm(ECO ~., data = d, kernel = "sigmoid")
plot(svm_s, data = Auto[, c(4, 5, 10)])
```



d

```r
rm(svm)
set.seed(1234)
svm_c <- tune(svm, ECO ~ ., data = d, ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10, 100,
1000), kernel = c("linear", "polynomial", "radial", "sigmoid")))
```

```r
summary(svm_c)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost kernel
##  1000 radial
##
## - best performance: 0.1096154
##
## - Detailed performance results:
##     cost     kernel     error dispersion
## 1  1e-03     linear 0.2322436 0.12569409
## 2  1e-02     linear 0.1376923 0.06926822
## 3  1e-01     linear 0.1121154 0.03803233
## 4  1e+00     linear 0.1247436 0.03961149
## 5  1e+01     linear 0.1272436 0.04980718
## 6  1e+02     linear 0.1246795 0.05040856
## 7  1e+03     linear 0.1246795 0.05040856
## 8  1e-03 polynomial 0.3775641 0.10307077
## 9  1e-02 polynomial 0.2653846 0.10583116
## 10 1e-01 polynomial 0.1757692 0.10640674
## 11 1e+00 polynomial 0.1732051 0.07054312
## 12 1e+01 polynomial 0.1580128 0.04527823
## 13 1e+02 polynomial 0.1529487 0.04599053
## 14 1e+03 polynomial 0.1554487 0.04656674
## 15 1e-03     radial 0.5562821 0.04500063
## 16 1e-02     radial 0.1350641 0.05626604
## 17 1e-01     radial 0.1171795 0.03375365
## 18 1e+00     radial 0.1221795 0.04682189
## 19 1e+01     radial 0.1146795 0.03818090
## 20 1e+02     radial 0.1146795 0.03818090
## 21 1e+03     radial 0.1096154 0.03800777
## 22 1e-03    sigmoid 0.5562821 0.04500063
## 23 1e-02    sigmoid 0.1530128 0.07426316
## 24 1e-01    sigmoid 0.1171795 0.04649761
## 25 1e+00    sigmoid 0.1144872 0.04416435
## 26 1e+01    sigmoid 0.1451282 0.04376974
## 27 1e+02    sigmoid 0.1553205 0.03908102
## 28 1e+03    sigmoid 0.1578846 0.03980793
```

e

```
1 - svm_c$performances[svm_c$performances$error == min(svm_c$performances$error),]$error
```

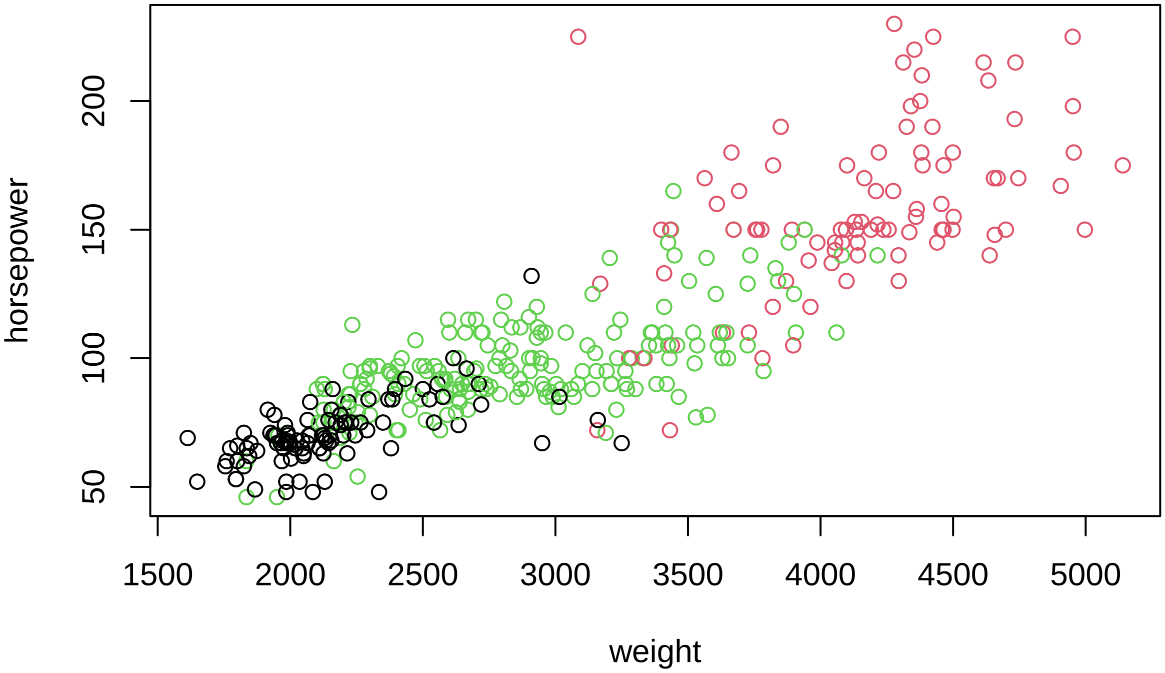
```
## [1] 0.8903846
```

f

```
Auto$ECO = ifelse( mpg < 17, "Low Consuming",
                   ifelse( mpg < 29, "Mid Consuming", "Economy"))
Auto$ECO <- as.factor(Auto$ECO)
attach(Auto)
```

```
## The following objects are masked from Auto (pos = 3):
##
##     acceleration, cylinders, displacement, ECO, horsepower, mpg, name,
##     origin, weight, year

## The following objects are masked from Auto (pos = 4):
##
##     acceleration, cylinders, displacement, horsepower, mpg, name,
##     origin, weight, year

## The following object is masked from package:lubridate:
##
##     origin

## The following object is masked from package:ggplot2:
##
##     mpg
```

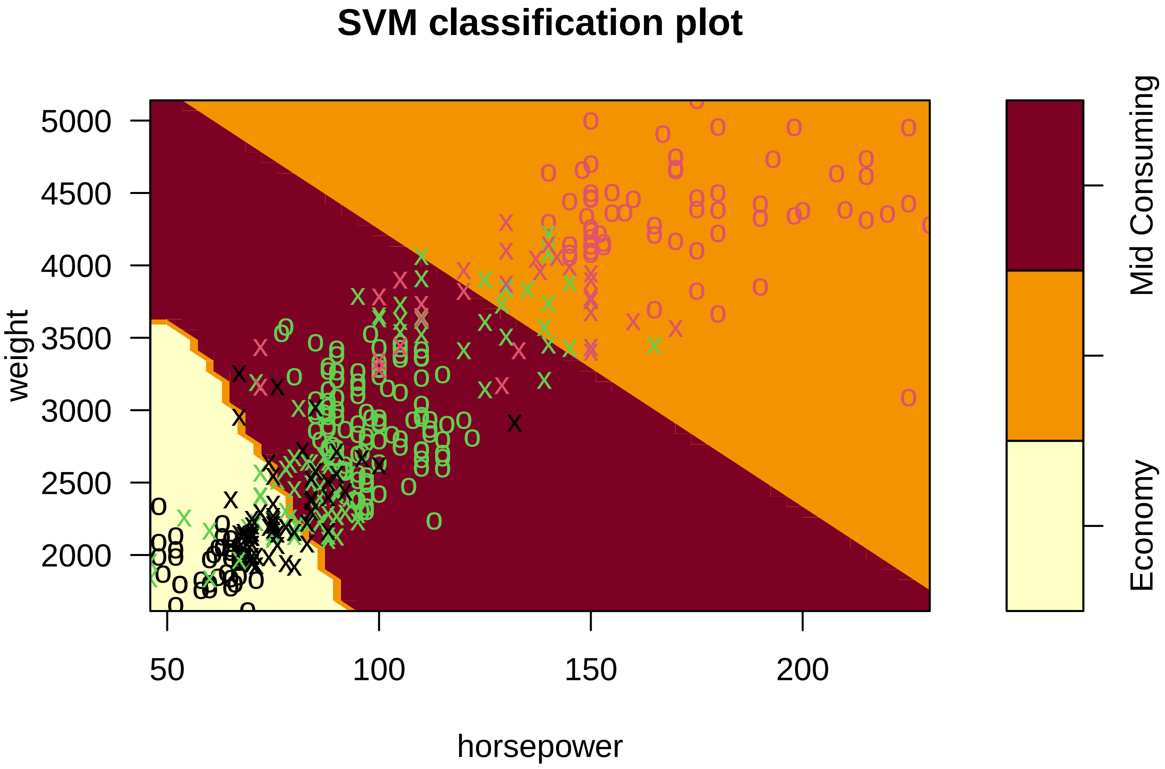
```
plot(weight, horsepower, col = as.numeric(Auto$ECO))
```



```
d = data.frame(ECO, weight, horsepower)
svm <- svm(ECO ~., data = d, kernel = "linear")
summary(svm)
```

```
##
## Call:
## svm(formula = ECO ~ ., data = d, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  178
##
##  ( 89 31 58 )
##
##
## Number of Classes:  3
##
## Levels:
##  Economy Low Consuming Mid Consuming
```

```
plot(svm, data = Auto[, c(4, 5, 10)])
```



```
rm(svm)
set.seed(1234)
svm_c <- tune(svm, ECO ~ ., data = d, ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10, 100,
1000), kernel = c("linear", "polynomial", "radial", "sigmoid")))
```

```
summary(svm_c)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost     kernel
##    10 polynomial
##
## - best performance: 0.1709615
##
## - Detailed performance results:
##     cost     kernel     error dispersion
## 1  1e-03     linear 0.4974359 0.10109344
## 2  1e-02     linear 0.3215385 0.08070714
## 3  1e-01     linear 0.1836538 0.08144993
## 4  1e+00     linear 0.1862821 0.07808723
## 5  1e+01     linear 0.1837821 0.07124333
## 6  1e+02     linear 0.1863462 0.07453318
## 7  1e+03     linear 0.1863462 0.07453318
## 8  1e-03 polynomial 0.4310897 0.08887720
## 9  1e-02 polynomial 0.3062179 0.11100671
## 10 1e-01 polynomial 0.1786538 0.05805996
## 11 1e+00 polynomial 0.1761538 0.06457433
## 12 1e+01 polynomial 0.1709615 0.06166302
## 13 1e+02 polynomial 0.1735256 0.06140481
## 14 1e+03 polynomial 0.1709615 0.05924628
## 15 1e-03     radial 0.4974359 0.10109344
## 16 1e-02     radial 0.4974359 0.10109344
## 17 1e-01     radial 0.1810897 0.07741489
## 18 1e+00     radial 0.1760256 0.07739185
## 19 1e+01     radial 0.1735897 0.07797348
## 20 1e+02     radial 0.1710256 0.07910821
## 21 1e+03     radial 0.1837179 0.08148857
## 22 1e-03    sigmoid 0.4974359 0.10109344
## 23 1e-02    sigmoid 0.4974359 0.10109344
## 24 1e-01    sigmoid 0.2780128 0.09367461
## 25 1e+00    sigmoid 0.3547436 0.09663942
## 26 1e+01    sigmoid 0.4107692 0.10271614
## 27 1e+02    sigmoid 0.4286538 0.10009768
## 28 1e+03    sigmoid 0.4112821 0.14138623
```

```
1 - svm_c$performances[svm_c$performances$error == min(svm_c$performances$error),]$error
```

```
## [1] 0.8290385 0.8290385
```