# Classification models

Luis Nicolás Luarte Rodríguez

- Load Boston database
- Generate binary variable out of criminality

```
library(MASS)
library(dplyr)
## load database
df <- Boston

## transform crim to binary
df$crim <- ifelse(df$crim >= mean(df$crim), 1, 0)
print(head(df))
```

```
  crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat medv
1    0 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98 24.0
2    0  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14 21.6
3    0  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03 34.7
4    0  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94 33.4
5    0  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33 36.2
6    0  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21 28.7
```

- Generate train and test partition (0.7, 0.3)

```
## option 1 self-made
df_partition <- function(data, train_size)
{
  set.seed(101)
  # get indices
  sample <- sample.int(n = nrow(data),
                       size = floor(train_size*nrow(data)),
                       replace = F)
  # generate train data
  train <- data[sample, ]
  # and test data
  test  <- data[-sample, ]
  # list of train and test
  return(list(train, test))
                                        # return it
}

## example
fun_list <- df_partition(df, 0.7)
df_train <- fun_list[[1]]
df_test <- fun_list[[2]]

## option 2 using caret
library(caret)
## LGOCV <- leave group out
## p <- train percentage
## number <- iterations
## same as 1 train/test split
train_control <- trainControl(method = "LGOCV",
                              p = 0.7,
                              number = 1,
                              savePredictions = TRUE)

## the first function will be used since it’s easier to deal with
## native R objects
```

- Train following models
- KNN
- SVM
- Decision tree
- Neural net

```
##########################
## K-nearest-neighbours ##
##########################
```

```
## knn parameter is number neighbours 'k'
library(class)
knn_mdl <- knn(as.factor(crim) ~ .,
               data = df,
               k = 3)
## confusion matrix
tbl <- table(knn_mdl, df_test$crim)
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tbl)

## optimize with grid search
grid <- seq(from = 1, to = 200, by = 1)
k_opt <- sapply(grid, function(x) accuracy(table(knn(df_train,
                             df_test,
                             cl = as.factor(df_train$crim),
                             k = x), df_test$crim)))

## using caret
knn_caret <- train(as.factor(crim) ~ .,
                   data = df,
                   method = "knn",
                   trControl = train_control,
                   tuneGrid = expand.grid(k = 1:200))
## we can get the optimal parameter back a train using that
knn_mdl_opt <- knn(df_train,
               df_test,
               cl = as.factor(df_train$crim),
               k = knn_caret$bestTune$k)
## confusion matrix optimal model
tbl_2 <- table(knn_mdl, df_test$crim)
accuracy(tbl_2)

############################
## Support vector machine ##
############################
## here on out I'll be using caret because is way much faster
## assumming a linear kernel there's only one parameter to optimize
## the cost o C
## parallel so it goes faster
library(doParallel)
registerDoParallel(cores=8)
svm_caret <- train(as.factor(crim) ~ .,
                   data = df, method = "svmLinear",
                   tuneGrid = expand.grid(C = 1:100),
                   preProcess = c("scale"),
                   metric = "Accuracy")

## we do the same as before
library(e1017)
svm_mdl_opt <- svm(as.factor(crim) ~ .,
                   data = df_train,
                   kernel.type="linear",
                   cost = svm_caret$bestTune$C)
svm_pred <- predict(svm_mdl_opt, newdata = df_test)
## confusion matrix optimal model
tbl_3 <- table(svm_pred, df_test$crim)
accuracy(tbl_3)

###################
## Decision tree ##
###################
## trained in all data
library(tree)
tree_boston <- tree(as.factor(crim) ~ .,
                 data = df)
par(mfrow = c(1,3))
plot(tree_boston)
text(tree_boston, pretty = 0)
## using train test
tree_boston_cv = tree(as.factor(crim) ~ .,
                      data = df_train)
plot(tree_boston_cv)
text(tree_boston_cv, pretty=0)
## check accuracy
tree_pred = predict(tree_boston_cv, newdata = df_train, type="class")
tbl_4 <- with(df_train, table(tree_pred, crim))
```

```
accuracy(tbl_4)

## prune the tree
tree_boston_prune <- cv.tree(tree_boston_cv, FUN = prune.misclass)
plot(tree_boston_prune)

####################
## neural network ##
####################
library(nnet)
nnet_boston <- train(as.factor(crim) ~ .,
                     data = df,
                     method = "nnet",
                     trControl = train_control,
                     tuneGrid = expand.grid(size = 1:20, decay = 0.1))
```

```
Error in knn(as.factor(crim) ~ ., data = df, k = 3) :
  unused argument (data = df)
[1] 98.02632
[1] 98.02632
Error in library(e1017) : there is no package called ‘e1017’
[1] 98.02632
[1] 99.15254
# weights:  16
initial  value 226.196132
iter  10 value 201.052318
iter  20 value 72.846098
iter  30 value 43.211325
# weights:  31
initial  value 292.368582
iter  40 value 37.945742
iter  10 value 201.029879
iter  20 value 54.747725
iter  50 value 29.668717
iter  60 value 28.102580
iter  30 value 43.887130
iter  70 value 21.512648
# weights:  46
```

```
initial  value 309.287302
iter  80 value 19.677944
iter  90 value 19.619891
iter  40 value 29.438416
iter 100 value 19.608291
final  value 19.608291
stopped after 100 iterations
iter  10 value 173.198654
# weights:  61
initial  value 214.048752
iter  50 value 20.226776
iter  20 value 157.993503
# weights: iter  10 value 39.557842
 76
initial  value 255.688362
iter  60 value 12.007125
iter  30 value 69.971068
iter  20 value 33.132042
iter  40 value 18.964819
iter  70 value 9.268370
iter  10 value 182.757707
# weights:  136
initial  value 214.028284
iter  50 value 17.174742
iter  80 value 7.809193
iter  30 value 31.198290
iter  90 value 7.671440
iter  20 value 40.145347
iter 100 value 7.653837
final  value 7.653837
stopped after 100 iterations
# weights:  91iter  60 value 14.524319

initial  value 340.440439
iter  40 value 30.921707
iter  10 value 172.297840
iter  30 value 32.763593
iter  70 value 13.387735
iter  10 value 40.655366
iter  50 value 28.857423
iter  80 value 11.049919
iter  40 value 31.875453
iter  20 value 21.516621
iter  20 value 135.271112
# weights:  151
iter  90 value 9.801624
initial  value 559.517776
iter  60 value 20.806553
iter  50 value 28.671111
# weights:  121
initial  value 229.795568
iter  70 value 9.624488
iter 100 value 9.455691
final  value 9.455691
stopped after 100 iterations
iter  30 value 12.653412
iter  60 value 19.803904
iter  10 value 188.049895
iter  10 value 55.884273
iter  30 value 38.661605
iter  80 value 8.587769
iter  40 value 10.842930
iter  70 value 17.658193
iter  20 value 52.969410
# weights:  iter  20 value 41.172773
iter  90 value 8.103390
iter  40 value 31.245365
iter  80 value 12.201893
iter  30 value 37.512875
# weights:  166
initial  value 298.937291
iter  50 value 10.159459
iter 100 value 6.892904
final  value 6.892904
stopped after 100 iterations
106
```

```
iter  90 value 8.984671
initial  value 275.413411
iter  30 value 34.438604
iter  50 value 29.245895
iter  10 value 72.501052
iter  60 value 9.264950
iter 100 value 8.273510
final  value 8.273510
stopped after 100 iterations
iter  40 value 35.790249
iter  10 value 50.579123
# weights:  181
initial  value 272.266464
iter  60 value 26.492098
iter  20 value 40.808950
iter  70 value 7.648837
iter  40 value 32.098814
iter  50 value 30.341618
# weights:  196
initial  value 241.582623
iter  70 value 15.221633
iter  80 value 7.430490
iter  10 value 72.235171
iter  30 value 30.459756
iter  50 value 26.898181
iter  60 value 27.346294
iter  80 value 12.539122
iter  90 value 7.113488
iter  10 value 117.076454
iter  20 value 35.800538
iter  20 value 47.142967
iter  40 value 28.080017
iter 100 value 6.136970
final  value 6.136970
stopped after 100 iterations
iter  60 value 20.460130
iter  90 value 9.236030
iter  70 value 18.871383
iter  20 value 48.273706
iter  50 value 27.767165
iter  30 value 32.966870
iter  70 value 13.744073
iter 100 value 7.819880
final  value 7.819880
stopped after 100 iterations
iter  80 value 16.419615
# weights:  211
initial  value 368.251083
iter  80 value 11.023044
iter  90 value 8.512111
iter  60 value 23.608656
iter  30 value 45.136249
# weights:  256
initial  value 262.384382
iter  40 value 29.517882
iter  10 value 42.773477
iter  30 value 43.857493
iter 100 value 7.495080
final  value 7.495080
stopped after 100 iterations
iter  90 value 9.437385
iter  70 value 12.118256
iter  50 value 22.475089
iter  40 value 39.667434
iter  20 value 37.181665
iter  10 value 93.785797
# weights:  241
initial  value 448.753982
iter 100 value 7.292606
final  value 7.292606
stopped after 100 iterations
iter  80 value 9.026718
iter  60 value 17.068397
iter  40 value 32.806673
iter  50 value 14.929911
iter  10 value 46.681667
```

```
# weights:  271
iter  20 value 36.340653
initial  value 436.949875
iter  70 value 15.173936
iter  90 value 7.278977
iter  30 value 31.227938
iter  50 value 25.997305
iter  60 value 12.625639
iter  20 value 36.741133
iter  80 value 14.120632
iter 100 value 6.715546
final  value 6.715546
stopped after 100 iterations
iter  10 value 61.990227
iter  40 value 29.345213
iter  30 value 23.504202
iter  70 value 8.570259
iter  60 value 23.335348
iter  90 value 12.027490
iter  30 value 33.055696
# weights:  286
initial  value 406.574475
iter  20 value 41.257323
iter  50 value 17.985714
iter  80 value 7.571126
iter 100 value 6.969828
final  value 6.969828
stopped after 100 iterations
iter  40 value 20.083305
iter  70 value 17.229861
iter  40 value 29.815509
iter  10 value 114.777632
iter  60 value 15.383874
# weights:  301
iter  90 value 7.110509
initial  value 260.956503
iter  80 value 14.952952
iter  30 value 32.329012
iter  50 value 16.132398
iter  90 value 13.747189
iter 100 value 5.512198
final  value 5.512198
stopped after 100 iterations
iter  20 value 91.334228
iter  50 value 28.989031
iter  70 value 14.691812
iter  10 value 42.799434
iter  40 value 19.973625
iter 100 value 13.374171
iter  60 value 11.478912
final  value 13.374171
stopped after 100 iterations
iter  30 value 68.231096
iter  60 value 23.404093
iter  80 value 14.243021
iter  20 value 38.966559
iter  50 value 11.724646
# weights:  226
initial  value 218.478341
iter  70 value 9.548567
iter  40 value 52.879250
iter  70 value 12.594034
iter  90 value 12.789243
iter  30 value 36.965256
iter  60 value 7.561389
iter  10 value 45.918771
iter 100 value 10.899136
final  value 10.899136
stopped after 100 iterations
iter  80 value 11.116552
iter  50 value 30.944902
iter  80 value 8.834259
iter  20 value 34.546945
iter  70 value 7.128295
iter  40 value 32.822016
iter  60 value 27.615291
iter  90 value 9.976340
iter  90 value 8.300747
iter  30 value 30.854265
iter  80 value 6.386393
iter 100 value 6.169060
final  value 6.169060
stopped after 100 iterations
iter  70 value 17.429110
iter  50 value 31.872403
iter 100 value 9.360403
final  value 9.360403
stopped after 100 iterations
iter  40 value 27.200476
iter  90 value 5.314960
iter  80 value 14.887858
iter  50 value 14.883211
iter  60 value 26.635680
iter  60 value 11.916040
iter  90 value 11.444747
iter 100 value 5.232587
final  value 5.232587
stopped after 100 iterations
iter  70 value 9.975334
iter  70 value 23.239447
iter  80 value 8.028598
iter 100 value 8.502123
final  value 8.502123
stopped after 100 iterations
iter  90 value 6.773246
iter  80 value 20.511832
iter 100 value 6.634517
final  value 6.634517
stopped after 100 iterations
iter  90 value 16.150784
iter 100 value 12.085513
final  value 12.085513
stopped after 100 iterations
# weights:  121
```

```
initial  value 344.534530
iter  10 value 71.092929
iter  20 value 63.838360
iter  30 value 56.438977
iter  40 value 40.868469
iter  50 value 35.911126
iter  60 value 31.406140
iter  70 value 26.444076
iter  80 value 21.994367
iter  90 value 20.213973
iter 100 value 19.302098
final  value 19.302098
stopped after 100 iterations
```