lab10.R

sathvik

2021-04-08

```
# 171EC146 Sathvik S Prabhu

## Chapter 10
sms_results<-read.csv("/home/sathvik/EC8/ML/Lab/Lab7/sms_results.csv")
table(sms_results$actual_type, sms_results$predict_type)
```

```
##
##          ham spam
##   ham   1202    5
##   spam    29  154
```

```
library(gmodels)
CrossTable(sms_results$actual_type, sms_results$predict_type)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1390
##
##
##                          | sms_results$predict_type
## sms_results$actual_type |       ham |      spam | Row Total |
## ------------------------|-----------|-----------|-----------|
##                     ham |      1202 |         5 |      1207 |
##                         |    16.565 |   128.248 |           |
##                         |     0.996 |     0.004 |     0.868 |
##                         |     0.976 |     0.031 |           |
##                         |     0.865 |     0.004 |           |
## ------------------------|-----------|-----------|-----------|
##                    spam |        29 |       154 |       183 |
##                         |   109.256 |   845.876 |           |
##                         |     0.158 |     0.842 |     0.132 |
##                         |     0.024 |     0.969 |           |
##                         |     0.021 |     0.111 |           |
```

```
## -------------------------|-----------|-----------|-----------|
##              Column Total |      1231 |       159 |      1390 |
##                           |     0.886 |     0.114 |           |
## -------------------------|-----------|-----------|-----------|
##
##
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
sms_results$predict_type<-as.factor(sms_results$predict_type)
sms_results$actual_type<-as.factor(sms_results$actual_type)
confusionMatrix(sms_results$predict_type,sms_results$actual_type,positive = "spam")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  ham spam
##       ham  1202   29
##       spam    5  154
##
##                Accuracy : 0.9755
##                  95% CI : (0.966, 0.983)
##     No Information Rate : 0.8683
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8867
##
##  Mcnemar's Test P-Value : 7.998e-05
##
##             Sensitivity : 0.8415
##             Specificity : 0.9959
##          Pos Pred Value : 0.9686
##          Neg Pred Value : 0.9764
##              Prevalence : 0.1317
##          Detection Rate : 0.1108
##    Detection Prevalence : 0.1144
##       Balanced Accuracy : 0.9187
##
##        'Positive' Class : spam
##
```

```
sensitivity(sms_results$predict_type, sms_results$actual_type,
            positive = "spam")
```

```
## [1] 0.8415301
```

```
specificity(sms_results$predict_type, sms_results$actual_type,
            negative = "ham")
```

```
## [1] 0.9958575
```

```
posPredValue(sms_results$predict_type, sms_results$actual_type,
             positive = "spam")
```

```
## [1] 0.9685535
```

```
library(vcd)
```

```
## Loading required package: grid
```

```
Kappa(table(sms_results$actual_type, sms_results$predict_type))
```

```
##             value     ASE     z Pr(>|z|)
## Unweighted 0.8867 0.01909 46.45        0
## Weighted   0.8867 0.01909 46.45        0
```

```
library(irr)
```

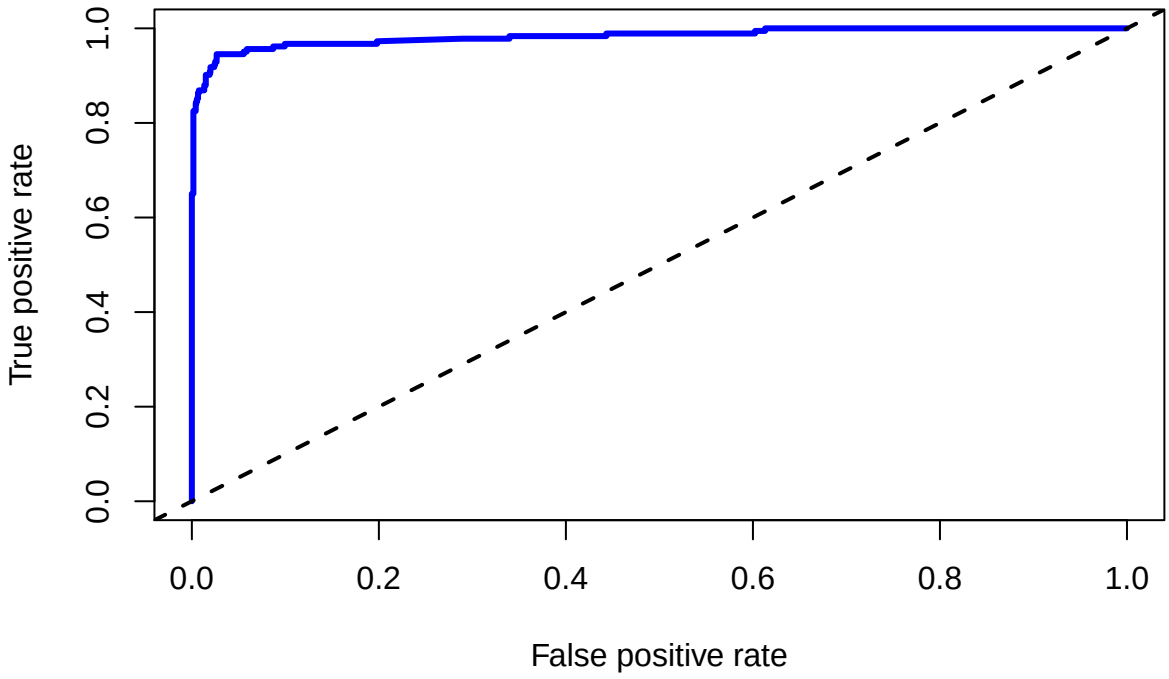
```
## Loading required package: lpSolve
```

```
kappa2(sms_results[1:2])
```

```
##  Cohen's Kappa for 2 Raters (Weights: unweighted)
##
##  Subjects = 1390
##    Raters = 2
##     Kappa = 0.887
##
##         z = 33.2
##   p-value = 0
```

```
# Visualizing performance tradeoffs
library(ROCR)
pred <- prediction(predictions = sms_results$prob_spam,labels = sms_results$actual_type)

perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf, main = "ROC curve for SMS spam filter",
     col = "blue", lwd = 3)
abline(a = 0, b = 1, lwd = 2, lty = 2)
```




```
perf.auc <- performance(pred, measure = "auc")
str(perf.auc)
```

```
## Formal class 'performance' [package "ROCR"] with 6 slots
##   ..@ x.name      : chr "None"
##   ..@ y.name      : chr "Area under the ROC curve"
##   ..@ alpha.name  : chr "none"
##   ..@ x.values    : list()
##   ..@ y.values    :List of 1
##   .. ..$ : num 0.983
##   ..@ alpha.values: list()
```

```
unlist(perf.auc@y.values)
```

```
## [1] 0.9829999
```

```
# Estimating future performance
credit <- read.csv("/home/sathvik/EC8/ML/Lab/Lab5/credit.csv", stringsAsFactors= T)
str(credit)
```

```
## 'data.frame':    1000 obs. of  21 variables:
##  $ checking_balance    : Factor w/ 4 levels "< 0 DM","> 200 DM",..: 1 3 4 1 1 4 4 3 4 3 ...
##  $ months_loan_duration: int  6 48 12 42 24 36 24 36 12 30 ...
##  $ credit_history      : Factor w/ 5 levels "critical","delayed",..: 1 5 1 5 2 5 5 5 5 1 ...
##  $ purpose             : Factor w/ 10 levels "business","car (new)",..: 8 8 5 6 2 5 6 3 8 2 ...
##  $ amount              : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
##  $ savings_balance     : Factor w/ 5 levels "< 100 DM","> 1000 DM",..: 5 1 1 1 1 5 4 1 2 1 ...
##  $ employment_length   : Factor w/ 5 levels "> 7 yrs","0 - 1 yrs",..: 1 3 4 4 3 3 1 3 4 5 ...
##  $ installment_rate    : int  4 2 2 2 3 2 3 2 2 4 ...
##  $ personal_status     : Factor w/ 4 levels "divorced male",..: 4 2 4 4 4 4 4 4 1 3 ...
##  $ other_debtors       : Factor w/ 3 levels "co-applicant",..: 3 3 3 2 3 3 3 3 3 3 ...
```

```
##  $ residence_history   : int  4 2 3 4 4 4 4 2 4 2 ...
##  $ property            : Factor w/ 4 levels "building society savings",..: 3 3 3 1 4 4 1 2 3 2 ...
##  $ age                 : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ installment_plan    : Factor w/ 3 levels "bank","none",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ housing             : Factor w/ 3 levels "for free","own",..: 2 2 2 1 1 1 2 3 2 2 ...
##  $ existing_credits    : int  2 1 1 1 2 1 1 1 1 2 ...
##  $ default             : int  1 2 1 1 2 1 1 1 1 2 ...
##  $ dependents          : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ telephone           : Factor w/ 2 levels "none","yes": 2 1 1 1 1 2 1 2 1 1 ...
##  $ foreign_worker      : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ job                 : Factor w/ 4 levels "mangement self-employed",..: 2 2 4 2 2 4 2 1 4 1 ...
```

```
library(caret)
library(C50)
library(irr)

# Holdout method
random_ids <- order(runif(1000))
credit_train <- credit[random_ids[1:500],]
credit_validate <- credit[random_ids[501:750], ]
credit_test <- credit[random_ids[751:1000], ]

in_train <- createDataPartition(credit$default, p = 0.75,
                                list = FALSE)
credit_train <- credit[in_train, ]
head(credit_train[,1:5])
```

```
##   checking_balance months_loan_duration credit_history   purpose amount
## 1           < 0 DM                    6       critical  radio/tv   1169
## 2       1 - 200 DM                   48         repaid  radio/tv   5951
## 3          unknown                   12       critical education   2096
## 4           < 0 DM                   42         repaid furniture   7882
## 5           < 0 DM                   24        delayed car (new)   4870
## 7          unknown                   24         repaid furniture   2835
```

```
credit_test <- credit[-in_train, ]
head(credit_test[,1:5])
```

```
##    checking_balance months_loan_duration credit_history    purpose amount
## 6           unknown                   36         repaid  education   9055
## 13      1 - 200 DM                   12         repaid   radio/tv   1567
## 16          < 0 DM                   24         repaid   radio/tv   1282
## 23          < 0 DM                   10       critical  car (new)   2241
## 44          < 0 DM                   30       critical car (used)   6187
## 48          < 0 DM                    6         repaid car (used)   1352
```

```
# Cross Validation
folds <- createFolds(credit$default, k = 10)
str(folds)
```

```
## List of 10
##  $ Fold01: int [1:100] 12 17 22 39 45 62 69 73 90 94 ...
##  $ Fold02: int [1:100] 7 10 19 33 37 56 59 63 75 102 ...
##  $ Fold03: int [1:100] 6 28 29 32 52 53 67 81 85 96 ...
##  $ Fold04: int [1:100] 8 18 38 40 44 49 50 65 68 70 ...
##  $ Fold05: int [1:100] 3 5 11 47 55 58 72 79 82 103 ...
```

```
##  $ Fold06: int [1:100] 15 31 34 74 77 88 91 97 101 117 ...
##  $ Fold07: int [1:100] 14 20 23 26 30 35 46 48 66 83 ...
##  $ Fold08: int [1:100] 1 9 13 25 61 64 80 89 92 104 ...
##  $ Fold09: int [1:100] 2 16 21 24 36 41 42 43 60 71 ...
##  $ Fold10: int [1:100] 4 27 51 54 57 76 87 95 98 105 ...
```

```
credit01_train <- credit[folds$Fold01, ]
credit01_test <- credit[-folds$Fold01, ]

set.seed(123)
folds <- createFolds(credit$default, k = 10)
str(folds)
```

```
## List of 10
##  $ Fold01: int [1:100] 19 30 35 41 50 56 60 68 69 98 ...
##  $ Fold02: int [1:100] 15 31 45 57 61 83 120 126 133 135 ...
##  $ Fold03: int [1:100] 2 16 20 33 36 38 46 47 64 72 ...
##  $ Fold04: int [1:100] 11 12 13 37 39 51 52 65 76 80 ...
##  $ Fold05: int [1:100] 1 5 14 23 34 42 43 49 59 62 ...
##  $ Fold06: int [1:100] 4 25 27 29 53 71 74 75 78 88 ...
##  $ Fold07: int [1:100] 44 54 66 77 97 108 111 113 125 142 ...
##  $ Fold08: int [1:100] 6 7 8 17 21 58 73 79 85 94 ...
##  $ Fold09: int [1:100] 3 9 10 18 22 26 28 48 63 67 ...
##  $ Fold10: int [1:100] 24 32 40 55 70 100 104 109 122 123 ...
```

```
cv_results <- lapply(folds, function(x) {
  credit_train <- credit[x, ]
  credit_test <- credit[-x, ]
  credit_train$default<-as.factor(credit_train$default)
  credit_model <- C5.0(default ~ ., data = credit_train)
  credit_pred <- predict(credit_model, credit_test)
  credit_actual <- credit_test$default
  kappa <- kappa2(data.frame(credit_actual, credit_pred))$value
  return(kappa)
})

# kappa statistics
str(cv_results)
```

```
## List of 10
##  $ Fold01: num 0.127
##  $ Fold02: num 0.0595
##  $ Fold03: num 0.138
##  $ Fold04: num 0.242
##  $ Fold05: num 0.111
##  $ Fold06: num 0.138
##  $ Fold07: num 0.0678
##  $ Fold08: num 0.228
##  $ Fold09: num 0.0811
##  $ Fold10: num 0.19
```

```
mean(unlist(cv_results))
```

```
## [1] 0.1382527
```

```
## Chapter 11
```

```r
# Creating a simple tuned model
credit <- read.csv("/home/sathvik/EC8/ML/Lab/Lab5/credit.csv", stringsAsFactors= T)
set.seed(300)
credit$default<-as.factor(credit$default)
m <- train(default ~ ., data = credit, method = "C5.0")
m
```

```
## C5.0
##
## 1000 samples
##   20 predictor
##    2 classes: '1', '2'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
## Resampling results across tuning parameters:
##
##   model  winnow  trials  Accuracy   Kappa
##   rules  FALSE    1      0.7027014  0.2920803
##   rules  FALSE   10      0.7243269  0.3453188
##   rules  FALSE   20      0.7294474  0.3499065
##   rules   TRUE    1      0.6923914  0.2744217
##   rules   TRUE   10      0.7263089  0.3420603
##   rules   TRUE   20      0.7360215  0.3558315
##   tree   FALSE    1      0.6951701  0.2677459
##   tree   FALSE   10      0.7372294  0.3271180
##   tree   FALSE   20      0.7395288  0.3333668
##   tree    TRUE    1      0.6946925  0.2680391
##   tree    TRUE   10      0.7364909  0.3303587
##   tree    TRUE   20      0.7420638  0.3427882
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were trials = 20, model = tree and winnow
##  = TRUE.
```

```r
p <- predict(m, credit)
table(p, credit$default)
```

```
##
## p     1   2
##   1 676  79
##   2  24 221
```

```r
head(predict(m, credit))
```

```
## [1] 1 2 1 1 2 1
## Levels: 1 2
```

```r
head(predict(m, credit, type = "prob"))
```

```
##           1          2
## 1 0.8720819 0.12791809
## 2 0.3284062 0.67159380
## 3 1.0000000 0.00000000
## 4 0.7563177 0.24368229
```

```
## 5 0.4531722 0.54682783
## 6 0.9085110 0.09148904
```

```
# Customizing the tuning process
# . The oneSE function
# chooses the simplest candidate within one standard error of the best performance,
# and tolerance uses the simplest candidate within a user-specified percentage.
ctrl <- trainControl(method = "cv", number = 10,
                     selectionFunction = "oneSE")
grid <- expand.grid(.model = "tree",
                    .trials = c(1, 5, 10, 15, 20, 25, 30, 35),
                    .winnow = "FALSE")
grid
```

```
##   .model .trials .winnow
## 1   tree       1   FALSE
## 2   tree       5   FALSE
## 3   tree      10   FALSE
## 4   tree      15   FALSE
## 5   tree      20   FALSE
## 6   tree      25   FALSE
## 7   tree      30   FALSE
## 8   tree      35   FALSE
```

```
set.seed(300)
m <- train(default ~ ., data = credit, method = "C5.0",metric = "Kappa",
           trControl = ctrl,tuneGrid = grid)
```

```
## Warning in Ops.factor(x$winnow): '!' not meaningful for factors
```

```
m
```

```
## C5.0
##
## 1000 samples
##   20 predictor
##    2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...
## Resampling results across tuning parameters:
##
##   trials  Accuracy  Kappa
##    1      0.708     0.2824182
##    5      0.740     0.3604766
##   10      0.751     0.3604338
##   15      0.752     0.3716050
##   20      0.752     0.3655537
##   25      0.755     0.3775273
##   30      0.756     0.3761998
##   35      0.758     0.3818372
##
## Tuning parameter 'model' was held constant at a value of tree
## Tuning
##  parameter 'winnow' was held constant at a value of FALSE
## Kappa was used to select the optimal model using  the one SE rule.
```

```
## The final values used for the model were trials = 5, model = tree and winnow
##  = FALSE.
```

```
# Bagging (Bootstrap aggregating)
library(ipred)
set.seed(300)
mybag <- bagging(default ~ ., data = credit, nbagg = 25) # 25 decision trees
credit_pred <- predict(mybag, credit)
table(credit_pred, credit$default)
```

```
##
## credit_pred   1   2
##           1 699   3
##           2   1 297
```

```
library(caret)
set.seed(300)
ctrl <- trainControl(method = "cv", number = 10)
train(default ~ ., data = credit, method = "treebag",
      trControl = ctrl)
```

```
## Bagged CART
##
## 1000 samples
##   20 predictor
##    2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.743     0.3543981
```

```
str(svmBag)
```

```
## List of 3
##  $ fit      :function (x, y, ...)
##  $ pred     :function (object, x)
##  $ aggregate:function (x, type = "class")
```

```
svmBag$fit
```

```
## function (x, y, ...)
## {
##     loadNamespace("kernlab")
##     out <- kernlab::ksvm(as.matrix(x), y, prob.model = is.factor(y),
##         ...)
##     out
## }
## <bytecode: 0x55a6b46fdf70>
## <environment: namespace:caret>
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
credit$default<-as.factor(credit$default)
bagctrl <- bagControl(fit = svmBag$fit,
                      predict = svmBag$pred,
                      aggregate = svmBag$aggregate)
set.seed(300)
# svmbag <- train(default ~ ., data = credit, "bag",
#                   trControl = ctrl, bagControl = bagctrl)
# svmbag

# Boosting

# Random Forests
# bagging with random featureselection

# Training
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(300)
credit$default<-as.factor(credit$default)
rf <- randomForest(default ~ ., data = credit)
rf
```

```
##
## Call:
##  randomForest(formula = default ~ ., data = credit)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 23.2%
## Confusion matrix:
##     1   2 class.error
## 1 648  52  0.07428571
## 2 180 120  0.60000000
```

```
# Evaluation
library(caret)
ctrl <- trainControl(method = "repeatedcv",
                     number = 10, repeats = 10)
# mtry defines how many features are randomly selected at each split.
grid_rf <- expand.grid(.mtry = c(2, 4, 8, 16))
set.seed(300)
m_rf <- train(default ~ ., data = credit, method = "rf",metric = "Kappa",
```

```
              trControl = ctrl, tuneGrid = grid_rf)

grid_c50 <- expand.grid(.model = "tree",
                        .trials = c(10, 20, 30, 40),
                        .winnow = "FALSE")
set.seed(300)
m_c50 <- train(default ~ ., data = credit, method = "C5.0",
               metric = "Kappa", trControl = ctrl,
               tuneGrid = grid_c50)
```

```
## Warning in Ops.factor(x$winnow): '!' not meaningful for factors
```

```
# Comparing RF and C50
m_rf
```

```
## Random Forest
##
## 1000 samples
##   20 predictor
##    2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.7202    0.09787729
##    4    0.7486    0.27551507
##    8    0.7550    0.32980623
##   16    0.7601    0.36418906
##
## Kappa was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 16.
```

```
m_c50
```

```
## C5.0
##
## 1000 samples
##   20 predictor
##    2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...
## Resampling results across tuning parameters:
##
##   trials  Accuracy  Kappa
##   10      0.7381    0.3314105
##   20      0.7463    0.3530474
##   30      0.7491    0.3602113
##   40      0.7553    0.3753119
##
## Tuning parameter 'model' was held constant at a value of tree
```

```
## Tuning
##  parameter 'winnow' was held constant at a value of FALSE
## Kappa was used to select the optimal model using the largest value.
## The final values used for the model were trials = 40, model = tree and winnow
##  = FALSE.
```

```
# Chapter 12

library(RCurl)
webpage <- getURL("https://www.packtpub.com/")
str(webpage)
```

```
##  chr " <!doctype html>\n<html lang=\"en\">\n<head>\n<script>\n    var BASE_URL = 'https://www.packtp
```

```
library(rjson)
sample_json <- '{"breakfast" : [ "milk", "fruit loops", "juice" ],"lunch" : [ "left over sushi" ]}'

r_object <- fromJSON(sample_json)
# #To convert from an R object to a JSON object:
json_string <- toJSON(r_object)

system.time(rnorm(1000000))
```

```
##    user  system elapsed
##   0.123   0.000   0.125
```