Data in Public and Social Services - 4th practical exercise class

22nd April 2024

by Davide Chicco
davide.chicco@unimib.it

Practical exercise class lecturer: Vasco Coelho v.coelho@campus.unimib.it

The goals of this practical exercise class are the following:

A. Apply the concepts on clustering seen during the last class lecture on an EHRs dataset

R setup main commands

0) Put the following commands as header of your script:

```
setwd(".") #we use the current folder of the script as working directory
options(stringsAsFactors = FALSE) # we set the input strings not to be considered
set.seed(10) # we set a seed to be able to replicate our tests
options(repos = list(CRAN="http://cran.rstudio.com/")) # we set the URL where to download the packages
```

1) load the pacman library for an easier installation and loading of the libraries:

```
# install.packages("pacman", dependencies = TRUE)
library("pacman")
```

Load and/or install other packages we need

```
p_load("dlookr", "dplyr", "ggplot2", "pastecs", "tableone", "umap",
"textshape", "factoextra", "ggdendro", "fpc", "cluster", "ggdendro",
"clusterSim", "parameters", "randomForest", "metrica", "shapr")
```

Application of supervised machine learning methods a dataset of electronic health records of patients with diabetes type 1 from Japan

2.1) Takashi 2019 diabetes type 1 dataset: we download the preprocessed version, that is the output of the first practical exercise class

Takashi2019_diabetes_type1_dataset_preprocessed.csv file to download

Takashi Y, Ishizu M, Mori H, Miyashita K, Sakamoto F, Katakami N, et al. (2019) “Circulating osteocalcin as a bone-derived hormone is inversely correlated with body fat in patients with type 1 diabetes”. PLOS ONE 14(5): e0216416. https://doi.org/10.1371/journal.pone.0216416

Cerono G, Chicco D. 2024, “Ensemble machine learning reveals key features for diabetes duration from electronic health records”. PeerJ Computer Science 10:e1896 https://doi.org/10.7717/peerj-cs.1896

A. Load the dataset

```
fileName <- "Takashi2019_diabetes_type1_dataset_preprocessed.csv"
patients_data <- read.csv(fileName, header = TRUE, sep =",")
```

B. Quantitative description

```
# We want to generate the descriptive statistics of all the features involved

patients_data %>% dim()
patients_data %>% summary()
patients_data %>% str()
patients_data %>% colnames() %>% sort()
```

C. Let’s prepare the dataset for Random Forests with the target insulin_regimen_binary and then apply this method

```
# We need to put the target variable on the right end of the dataframe

targetName <- "insulin_regimen_binary"
patients_data <- patients_data %>% dplyr::relocate(targetName, .after =
last_col())

target_index <- ncol(patients_data)
patients_data[,target_index] <-
as.factor(patients_data[,target_index])

# We randomly shuffle the rows
patients_data <- patients_data[sample(nrow(patients_data)),]

# We randomly select 80% of the rows for training set and use the remaining part for
the test set
```

```
training_set_perc <- 80
training_set_first_index <- 1 # NEW
training_set_last_index <-
round(nrow(patients_data)*training_set_perc/100)

# the test set is the last 20% of the whole dataset
test_set_first_index <- training_set_last_index + 1
test_set_last_index <- nrow(patients_data)

cat("[Creating the training set and test set for the values]\n")
patients_data_train <-
patients_data[training_set_first_index:training_set_last_index,]
patients_data_test <-
patients_data[test_set_first_index:test_set_last_index,]

# We create the formula for predicting the target from the other features
allFeaturesFormula <-
as.formula(paste(as.factor(colnames(patients_data)[target_index]), '.',
sep=' ~ ' ))

rf_new <- randomForest(allFeaturesFormula, data=patients_data_train,
type="classification", importance=FALSE, proximity=TRUE)

cat("\n[Applying the trained random forest classifier on the test
set]\n")
patients_data_test_predictions <- as.numeric(predict(rf_new,
patients_data_test, type="response")) - 1

confusion_matrix(obs = patients_data_test[,target_index], pred =
patients_data_test_predictions)

these_metrics <- metrics_summary(obs =
patients_data_test[,target_index], pred =
patients_data_test_predictions, type="classification")

# we can compute the MCC only if the predictions are not all 0s or all 1s

if(var(patients_data_test_predictions)!=0) {
      cat("MCC = ", these_metrics[which(these_metrics ==
      "mcc"),]$"Score", "\n", sep="")
      cat("TPR = ", these_metrics[which(these_metrics ==
      "recall"),]$"Score", "\n", sep="")
      cat("TNR = ", these_metrics[which(these_metrics ==
      "specificity"),]$"Score", "\n", sep="")
      cat("PPV = ", these_metrics[which(these_metrics ==
      "precision"),]$"Score", "\n", sep="")
      cat("NPV = ", these_metrics[which(these_metrics ==
      "npv"),]$"Score", "\n", sep="")
```

```
}
```

D. Using the held-out approach, repeat the execution of the binary classification 1,000 times, by using randomly sampled data instances every time. Save all the results of the MCC into a vector. In the end, print the average MCC and its standard deviation

```
execution_number <- 1000
mcc_list <- c()

cat("Number of executions = ", execution_number, "\n", sep="")
for(exe_i in 1:execution_number)
{
  # We randomly shuffle the rows
  patients_data <- patients_data[sample(nrow(patients_data)),]

  # We randomly select 80% of the rows for training set and use the remaining part for
the test set

        training_set_perc <- 80
        training_set_first_index <- 1
        training_set_last_index <-
        round(nrow(patients_data)*training_set_perc/100)

        # the test set is the last 20% of the whole dataset
        test_set_first_index <- training_set_last_index + 1
        test_set_last_index <- nrow(patients_data)

        cat("[Creating the training set and test set for the
      values]\n")
          patients_data_train <-
      patients_data[training_set_first_index:training_set_last_index,]
          patients_data_test <-
      patients_data[test_set_first_index:test_set_last_index,]

        # We create the formula for predicting the target from the other features
        allFeaturesFormula <-
      as.formula(paste(as.factor(colnames(patients_data)[target_index])
      , '.', sep=' ~ ' ))

            rf_new <- randomForest(allFeaturesFormula,
            data=patients_data_train, type="classification",
            importance=FALSE, proximity=TRUE)

            cat("\n[Applying the trained random forest classifier on
            the test set]\n")
```

```
            patients_data_test_predictions <-
            as.numeric(predict(rf_new, patients_data_test,
            type="response")) - 1

            confusion_matrix(obs = patients_data_test[,target_index],
            pred = patients_data_test_predictions) %>% print()

          if(var(patients_data_test_predictions)!=0) {

            classification_results <- metrics_summary(obs =
            patients_data_test[,target_index], pred =
            patients_data_test_predictions, type="classification")

            cat("TPR = ",
            classification_results[which(classification_results ==
            "recall"),]$"Score", "\n", sep="")
            cat("TNR = ",
            classification_results[which(classification_results ==
            "specificity"),]$"Score", "\n", sep="")
            cat("PPV = ",
            classification_results[which(classification_results ==
            "precision"),]$"Score", "\n", sep="")
            cat("NPV = ",
            classification_results[which(classification_results ==
            "npv"),]$"Score", "\n", sep="")

            mcc_list[exe_i] <-
            classification_results[which(classification_results ==
            "mcc"),]$"Score"
                } else mcc_list[exe_i] <- NA
}

mcc_list %>% na.omit() %>% stat.desc()
```

E. Feature ranking: we use recursive feature elimination based on the MCC to assess the most predictive variables

```
patients_data_original <- patients_data

mcc_list <- c()
feature_list <- c()

for(this_feature in 1:(ncol(patients_data)-1))
{
    patients_data <- patients_data_original

    cat("We remove the ", colnames(patients_data)[this_feature], " [",
this_feature, "] column in the dataset\n", sep="")
    patients_data[, this_feature] <- NULL
```

```
    target_index <- ncol(patients_data)
    patients_data[,target_index] <-
as.factor(patients_data[,target_index])

    # We randomly shuffle the rows
    patients_data <- patients_data[sample(nrow(patients_data)),]

    # We randomly select 80% of the rows for training set and use the
remaining part for the test set

    training_set_perc <- 80
    training_set_first_index <- 1 # NEW
    training_set_last_index <-
round(nrow(patients_data)*training_set_perc/100)

    # the test set is the last 20% of the whole dataset
    test_set_first_index <- training_set_last_index + 1
    test_set_last_index <- nrow(patients_data)

    cat("[Creating the training set and test set for the values]\n")
    patients_data_train <-
patients_data[training_set_first_index:training_set_last_index,]
    patients_data_test <-
patients_data[test_set_first_index:test_set_last_index,]

    # We create the formula for predicting the target from the other
features
    allFeaturesFormula <-
as.formula(paste(as.factor(colnames(patients_data)[target_index]), '.',
sep=' ~ ' ))

    rf_new <- randomForest(allFeaturesFormula,
data=patients_data_train, type="classification", importance=FALSE,
proximity=TRUE)

    cat("\n[Applying the trained random forest classifier on the test
set]\n")
    patients_data_test_predictions <- as.numeric(predict(rf_new,
patients_data_test, type="response")) - 1

    confusion_matrix(obs = patients_data_test[,target_index], pred =
patients_data_test_predictions)

    classification_results <- metrics_summary(obs =
patients_data_test[,target_index], pred =
patients_data_test_predictions, type="classification")

  # we can compute the MCC only if the predictions are not all 0s or all 1s
```

```
   if(var(patients_data_test_predictions)!=0) {

            classification_results <- metrics_summary(obs =
patients_data_test[,target_index], pred =
patients_data_test_predictions, type="classification")

            cat("TPR = ",
classification_results[which(classification_results ==
"recall"),]$"Score", "\n", sep="")
            cat("TNR = ",
classification_results[which(classification_results ==
"specificity"),]$"Score", "\n", sep="")
            cat("PPV = ",
classification_results[which(classification_results ==
"precision"),]$"Score", "\n", sep="")
            cat("NPV = ",
classification_results[which(classification_results ==
"npv"),]$"Score", "\n", sep="")

            mcc_list[this_feature] <-
classification_results[which(classification_results == "mcc"),]$"Score"
        } else mcc_list[this_feature] <- NA

}

feature_importance_by_MCC <- data.frame(feature =
colnames(patients_data_original[,-target_index]), MCC_drop = mcc_list)

feature_importance_by_MCC[order(-feature_importance_by_MCC$MCC_drop),]
```

F. Repeat the recursive feature elimination by arranging the ranking on the precision rather than using the MCC

G. Repeat all the binary classification by implementing the *k*-fold cross validation, with k=5. Report the final value of the MCC.

H. We saw how to perform feature ranking through recursive feature elimination (RFE), now we see how to do it with the Shapley method (additional explanation of the Shapley feature importance here)

```
fileName <- "Takashi2019_diabetes_type1_dataset_preprocessed.csv"
patients_data <- read.csv(fileName, header = TRUE, sep =",")

targetName <- "insulin_regimen_binary"
patients_data <- patients_data %>% dplyr::relocate(targetName, .after =
last_col())
```

```
patients_data_original <- patients_data

# We randomly shuffle the rows
patients_data <- patients_data[sample(nrow(patients_data)),]

target_index <- ncol(patients_data)

# We randomly select 80% of the rows for training set and use the remaining part for
the test set

training_set_perc <- 80
training_set_first_index <- 1 # NEW
training_set_last_index <-
round(nrow(patients_data)*training_set_perc/100)

# the test set is the last 20% of the whole dataset
test_set_first_index <- training_set_last_index + 1
test_set_last_index <- nrow(patients_data)

cat("[Creating the training set and test set for the values]\n")
patients_data_train <-
patients_data[training_set_first_index:training_set_last_index,]
patients_data_test <-
patients_data[test_set_first_index:test_set_last_index,]

# We create the formula for predicting the target from the other features
allFeaturesFormula <-
as.formula(paste(as.factor(colnames(patients_data)[target_index]), '.',
sep=' ~ ' ))


patients_data_train_without_target <- patients_data_train[,
-target_index]
patients_data_test_without_target <- patients_data_test[,
-target_index]

# We use logistic regression
lm_model <- lm(allFeaturesFormula, data = patients_data_train)


# Prepare the data for explanation
explainer <- shapr(patients_data_train_without_target, lm_model,
n_combinations = 10000)

# Specifying the phi_0, i.e. the expected prediction without any
features
p <- mean(patients_data_test[, target_index])
```

```
# Computing the actual Shapley values with kernelSHAP accounting for feature
dependence
explanation <- explain(
  patients_data_test_without_target,
  approach = "empirical",
  explainer = explainer,
  prediction_zero = p
)

# we plot the explanation for example for two observations (first two patients' profiles)
plot(explanation, plot_phi0 = FALSE, index_x_test = c(1:2))

dataframe_explanations <- (explanation$dt)

cat("Final feature ranking: ")
dataframe_explanations %>% colMeans() %>% sort() %>% print()
```

Application of the main data cleaning and data preparation steps to a dataset of electronic health records of patients with diabetes type 2 from Saudi Arabia

Repeat all the steps of the previous analysis [A, B, C, ..., I]
Convert the file from XLSX to CSV, first.
Use Diabetic retinopathy (DR) as target for the data unbalance phase

For one-hot encoding, use the `one_hot()` function of the `nestedcv` package:
https://search.r-project.org/CRAN/refmans/nestedcv/html/one_hot.html

AlOlaiwi LA, AlHarbi TJ, Tourkmani AM (2018) Prevalence of cardiovascular autonomic neuropathy and gastroparesis symptoms among patients with type 2 diabetes who attend a primary health care center. PLOS ONE 13(12): e0209500.
https://doi.org/10.1371/journal.pone.0209500

Cerono G, Chicco D. 2024, “Ensemble machine learning reveals key features for diabetes duration from electronic health records”. PeerJ Computer Science 10:e1896
https://doi.org/10.7717/peerj-cs.1896

2.2) pone.0209500.s001.xlsx file to download