Fraud Detection
Rita P. Ribeiro

2022/2023
DCC - FCUP

2 Hands On: Data Preparation

Data Quality

Load the following packages: `tidyverse`, `tidymodels`. This last meta package includes the package `recipes` that provides a collection of tools that support data many data pre-processing steps.

```
library(tidyverse)
library(tidymodels)
```

1. Handling Missing Values. Load the `carIns.Rdata` data set about the insurance risk rating of cars based on several characteristics of each car. Detailed information on this can be found in here.

```
load("carInsurance.Rdata")
str(carIns)
```

(a) Use basic functions of `tidyr` to:

- remove all the observations that contain missing values;

```
carIns %>% drop_na()
```

- replace all the missing values of the attribute `nDoors` with the value 'four'.

```
carIns %>% replace_na(list(nDoors='four'))
```

(b) Using `recipes` package from `tidymodels`, initialize a new `recipe` for pre-processing your data, as indicated below.

```
rec <- recipe(carIns)
```

We will now define several imputation methods, add them to the original recipe, preparing to bake all at the end as follows.

```
rec %>%
    # pre-processing steps imputation methods %>%
    prep() %>%
    bake(carIns)
```

(c) Run the following instructions and be critical regarding the obtained results.

- ```
  carIns_imput <-
    rec %>%
    step_impute_mode(nDoors) %>% step_impute_mean(normLoss) %>%
    prep() %>%
    bake(carIns)

  carIns_imput %>% select(nDoors,normLoss) %>% summary()
  ```
  ```

- ```
  carIns_imput  <-
    rec %>%
    step_impute_mode(all_nominal()) %>%
    step_impute_mean(all_numeric()) %>%
    prep() %>%
    bake(carIns)

  summary(carIns_imput)
  ```

- Explore other imputation methods available in package `recipes`.

Data Transformation

2. Resorting to a `step_*()` family functions, apply the following transformations to the price attribute of the same data set. Be critical regarding the obtained results.

(a) range-based normalization

```
carIns_stand <-
  rec %>%
  step_range(price) %>%
  prep() %>%
  bake(carIns)

carIns_stand %>% select(price) %>% summary()
```

(b) z-score normalization

```
carIns_stand <-
  rec %>%
  step_normalize(price) %>%
  prep() %>%
  bake(carIns)

carIns_stand %>% select(price) %>% summary()
```

(c) cut it into 4 equal-frequency ranges

```
carIns_discr <-
  rec %>%
  step_discretize(price,num_breaks=4) %>%
  prep() %>%
  bake(carIns)


carIns_discr %>% select(price) %>% summary()
```

Sampling Data

3. With the seed 123 and the `slice_sample` function obtain the following samples on the car insurance data set.

(a) A random sample of 50% of the cases

```
set.seed(123)
carIns_sample1 <- carIns %>% slice_sample(prop=0.5)
```

(b) A stratified sample of 60% of the cases of cars, according to the fuelType attribute.
```

```
set.seed(123)
carIns_sample2 <- carIns %>% group_by(fuelType) %>% slice_sample(prop=0.5)
```

(c) Inspect the distribution of values of variable `fuelType` in each of the two samples above.

```
carIns %>% select(fuelType) %>% summary()
carIns_sample1 %>% select(fuelType) %>% summary()
carIns_sample2 %>% select(fuelType) %>% summary()
```

Dimensionality Reduction

4. Use the filter feature selection method based on *pearson correlation coefficient*.

(a) Select the numeric attributes of the car insurance data set with imputation and use the function `cor()` to obtain the *pearson correlation coefficient* between each pair of variables.

```
carIns_num <- carIns_imput %>% select(where(is.numeric))
res <- carIns_num %>% cor()
```

(b) Load the package `corrplot`. Plot the all correlation information using the function `corrplot`. Explore some of its parameters.

```
library(corrplot)
corrplot(res,type="lower",
         method="number",
         number.cex = 0.5,diag=FALSE)
corrplot.mixed(res,
               lower = "circle",
               upper = "number",
               number.cex = 0.5,
               tl.col = "black",
               tl.cex = 0.5)
```

(c) Apply a pre-processing step that removes all the variables that have a correlation value above 0.8.

```
carIns_corr <-
  rec %>%
  step_impute_mean(all_numeric()) %>%
  step_corr(all_numeric(),threshold=.8) %>%
  prep() %>%
  bake(carIns)


setdiff(carIns %>% colnames(),carIns_corr %>% colnames())

carIns_corr_num <- carIns_corr %>% select(where(is.numeric))
res_corr <- carIns_corr_num %>% cor()
corrplot.mixed(res_corr,
               lower = "circle",
               upper = "number",
               number.cex = 0.65,
               tl.col = "black",
               tl.cex = 0.65)
```

(d) Apply the function `cor.mtest()` to the previous result to calculate the p-values and confidence intervals of the correlation coefficient for each pair of variables. Plot again the correlogram, making use of this information.

```
res1 <- cor.mtest(carIns_num,conf.level=0.95)
corrplot(res,p.mat=res1$p,type="lower",
         diag=FALSE,sig.level = 0.05,insig="blank")
```

5. Load the data set USArrests, from the datasets package, containing statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973, and also the percent of the population living in urban areas.

(a) Apply the function prcomp() to obtain the principal components. Inspect how each variable is obtained by the linear combination of each component.

(b) Plot using ggbiplot (Biplot for Principal Components using ggplot2) with following commands:

```r
# install.packages("devtools")
# library(devtools)
# install_github("vqv/ggbiplot")
library(ggbiplot)
data("USArrests")
res_pca <- prcomp(USArrests, scale=TRUE,center=TRUE)
res_pca
ggbiplot(res_pca,labels=rownames(USArrests))
```

Note that:

- PC1: higher loads on Murder, Assault and Rape. It could be interpreted as general measure of crime.
- PC2: higher load on UrbanPop. Level of urbanization.
- The states that are close to each other on the plot have similar data patterns in regards to the variables in the original dataset.
- Certain states are more highly associated with certain crimes than others.