Drinkers Report

Damiano Pincolini

2024-02-24

Table of contents

0.1 PROJECT INTRODUCTION 1
0.2 EDA 6
0.3 MODELING 12

0.1 PROJECT INTRODUCTION

0.1.1 PROJECT GOAL

The aim is to understand if it is possibile to define a relation between a target value (drinker status: "N", "Y") and some predictors both qualitative (i.e. smoker status) and quantitative (blood pressure, body index mass, cholesterol, etc).

0.1.2 LOADING LIBRARIES

The libraries that will be mainly used during this project will be: tidyverse, Datasmart, tidymodels and shapviz.

```
pacman::p_load(readr,
               tidyverse,
               tidymodels,
               DataExplorer,
               SmartEDA,
               ggcorrplot,
               corrplot,
               moments,
               skimr,
               DescTools,
```

```
              stats,
              vip,
              datawizard,
              GGally,
              caret,
              MASS,
              kknn,
              rpart.plot,
              healthyR.ai,
              embed,
              cowplot,
              factoextra,
              LiblineaR,
              shapviz)
```

0.1.3 LOADING DATA

The dataset is taken from this Kaggle's page: https://www.kaggle.com/datasets/sooyoungher/smoking-drinking-dataset. After downloading it on the laptop hard disk, it has been imported into R as a data frame called "DataOrigin".

0.1.4 DATA CHECK

0.1.4.1 Dataset structure and datatype analysis

```
dim(DataOrigin)
```

```
[1] 991346     24
```

```
str(DataOrigin)
```

```
spc_tbl_ [991,346 x 24] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ sex             : chr [1:991346] "Male" "Male" "Male" "Male" ...
 $ age             : num [1:991346] 35 30 40 50 50 50 45 35 55 40 ...
 $ height          : num [1:991346] 170 180 165 175 165 165 150 175 170 175 ...
 $ weight          : num [1:991346] 75 80 75 80 60 55 55 65 75 75 ...
 $ waistline       : num [1:991346] 90 89 91 91 80 75 69 84.2 84 82 ...
 $ sight_left      : num [1:991346] 1 0.9 1.2 1.5 1 1.2 0.5 1.2 1.2 1.5 ...
 $ sight_right     : num [1:991346] 1 1.2 1.5 1.2 1.2 1.5 0.4 1 0.9 1.5 ...
 $ hear_left       : num [1:991346] 1 1 1 1 1 1 1 1 1 1 ...
 $ hear_right      : num [1:991346] 1 1 1 1 1 1 1 1 1 1 ...
```

```
$ SBP             : num [1:991346] 120 130 120 145 138 142 101 132 145 132 ...
$ DBP             : num [1:991346] 80 82 70 87 82 92 58 80 85 105 ...
$ BLDS            : num [1:991346] 99 106 98 95 101 99 89 94 104 100 ...
$ tot_chole       : num [1:991346] 193 228 136 201 199 218 196 185 217 195 ...
$ HDL_chole       : num [1:991346] 48 55 41 76 61 77 66 58 56 60 ...
$ LDL_chole       : num [1:991346] 126 148 74 104 117 95 115 107 141 118 ...
$ triglyceride    : num [1:991346] 92 121 104 106 104 232 75 101 100 83 ...
$ hemoglobin      : num [1:991346] 17.1 15.8 15.8 17.6 13.8 13.8 12.3 14.4 15.1 13.9 ...
$ urine_protein   : num [1:991346] 1 1 1 1 1 3 1 1 1 1 ...
$ serum_creatinine: num [1:991346] 1 0.9 0.9 1.1 0.8 0.8 0.8 0.8 0.8 0.9 ...
$ SGOT_AST        : num [1:991346] 21 20 47 29 19 29 19 18 32 21 ...
$ SGOT_ALT        : num [1:991346] 35 36 32 34 12 40 12 18 23 38 ...
$ gamma_GTP       : num [1:991346] 40 27 68 18 25 37 12 35 26 16 ...
$ SMK_stat_type_cd: num [1:991346] 1 3 1 1 1 3 1 3 1 2 ...
$ DRK_YN          : chr [1:991346] "Y" "N" "N" "N" ...
- attr(*, "spec")=
 .. cols(
 ..   sex = col_character(),
 ..   age = col_double(),
 ..   height = col_double(),
 ..   weight = col_double(),
 ..   waistline = col_double(),
 ..   sight_left = col_double(),
 ..   sight_right = col_double(),
 ..   hear_left = col_double(),
 ..   hear_right = col_double(),
 ..   SBP = col_double(),
 ..   DBP = col_double(),
 ..   BLDS = col_double(),
 ..   tot_chole = col_double(),
 ..   HDL_chole = col_double(),
 ..   LDL_chole = col_double(),
 ..   triglyceride = col_double(),
 ..   hemoglobin = col_double(),
 ..   urine_protein = col_double(),
 ..   serum_creatinine = col_double(),
 ..   SGOT_AST = col_double(),
 ..   SGOT_ALT = col_double(),
 ..   gamma_GTP = col_double(),
 ..   SMK_stat_type_cd = col_double(),
 ..   DRK_YN = col_character()
 .. )
- attr(*, "problems")=<externalptr>
```

At a first sight, we can observe what follows: 1. “sex” and “DRK_YN” columns which are character data type. 2. All the other variables are numeric. 3. “hear_left”, “hear_right” are dicotomic numeric features (1=normal, 2=abnormal). They should be represented by a factor. 4. “urine_protein” is a numeric ordinal variable to be transformed into a factor (1=-, 2=+/-, 3=+1, 4=+2, 5=+3, 6=+4). 5. “SMK_stat_type” is a numeric ordinal variable to be transformed into a factor (1=never, 2=used to smoke but quit, 3=still smoke). 6. “DRK_YN” is a dicotomic character features (Y, N) that should be turned into a factor. This is supposed to be the target variable. 7. There are visible differences in features’ scales.

```
skim(DataOrigin)
```

Table 1: Data summary

Name	DataOrigin
Number of rows	991346
Number of columns	24
_____	
Column type frequency:	
character	2
numeric	22
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
sex	0	1	4	6	0	2	0
DRK_YN	0	1	1	1	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	47.61	14.18	20.0	35.0	45.0	60.0	85.0	
height	0	1	162.24	9.28	130.0	155.0	160.0	170.0	190.0	
weight	0	1	63.28	12.51	25.0	55.0	60.0	70.0	140.0	
waistline	0	1	81.23	11.85	8.0	74.1	81.0	87.8	999.0	
sight_left	0	1	0.98	0.61	0.1	0.7	1.0	1.2	9.9	
sight_right	0	1	0.98	0.60	0.1	0.7	1.0	1.2	9.9	
hear_left	0	1	1.03	0.17	1.0	1.0	1.0	1.0	2.0	
hear_right	0	1	1.03	0.17	1.0	1.0	1.0	1.0	2.0	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
SBP	0	1	122.43	14.54	67.0	112.0	120.0	131.0	273.0	
DBP	0	1	76.05	9.89	32.0	70.0	76.0	82.0	185.0	
BLDS	0	1	100.42	24.18	25.0	88.0	96.0	105.0	852.0	
tot_chole	0	1	195.56	38.66	30.0	169.0	193.0	219.0	2344.0	
HDL_chole	0	1	56.94	17.24	1.0	46.0	55.0	66.0	8110.0	
LDL_chole	0	1	113.04	35.84	1.0	89.0	111.0	135.0	5119.0	
triglyceride	0	1	132.14	102.20	1.0	73.0	106.0	159.0	9490.0	
hemoglobin	0	1	14.23	1.58	1.0	13.2	14.3	15.4	25.0	
urine_protein	0	1	1.09	0.44	1.0	1.0	1.0	1.0	6.0	
serum_creatinine	0	1	0.86	0.48	0.1	0.7	0.8	1.0	98.0	
SGOT_AST	0	1	25.99	23.49	1.0	19.0	23.0	28.0	9999.0	
SGOT_ALT	0	1	25.76	26.31	1.0	15.0	20.0	29.0	7210.0	
gamma_GTP	0	1	37.14	50.42	1.0	16.0	23.0	39.0	999.0	
SMK_stat_type_cd	0	1	1.61	0.82	1.0	1.0	1.0	2.0	3.0	

No missing value.

0.1.4.2 Conversion to factor

I force character data type to factor with the command below.

```
Data <- DataOrigin|>
  mutate(sex=as_factor((sex)),
         drkStatus=as_factor((DRK_YN)),
         hear_left=as_factor((hear_left)),
         hear_right=as_factor(hear_right),
         urine_protein=as_factor(urine_protein),
         smk_status=as_factor(SMK_stat_type_cd),
         .keep="unused")
```

0.1.5 DATA PARTITIONING

In order to speed up the CPU pace and avoid crashing it, I want to reduce on a random base the original data set and work, from this point, only on the new "reduced" object starting from data partitioning.

```
set.seed(666, sample.kind = "Rounding")

Index<- sample(1:nrow(Data),
                    size=100000,
```

```
                    replace=FALSE)

DataReduced <- Data[Index, ]

DataSplit <- initial_split(DataReduced,
                           prop=0.8,
                           strata=drkStatus)

trainData <- training(DataSplit)

testData <- testing(DataSplit)
```

0.2 EDA

0.2.1 Target variable distribution

```
prop.table(table(Data$drkStatus))
```

```
        Y         N
0.4998134 0.5001866
```

```
prop.table(table(trainData$drkStatus))
```

```
        Y         N
0.4985312 0.5014688
```

```
prop.table(table(testData$drkStatus))
```

```
        Y         N
0.4985251 0.5014749
```

0.2.2 Feature variables distribution and variability (noise detection)

0.2.2.1 Numerical features

```
DataSmartEDA <- ExpNumStat(trainData,
                           by="A",
                           gp=NULL,
                           Qnt=c(0.25,0.75),
                           MesofShape=2,
                           Outlier=TRUE,
                           round=2)
```

```
DataSmartEDA
```

```
              Vname Group    TN nNeg nZero  nPos NegInf PosInf NA_Value
1               age   All 79999    0     0 79999      0      0        0
9              BLDS   All 79999    0     0 79999      0      0        0
8               DBP   All 79999    0     0 79999      0      0        0
18        gamma_GTP   All 79999    0     0 79999      0      0        0
11        HDL_chole   All 79999    0     0 79999      0      0        0
2            height   All 79999    0     0 79999      0      0        0
14       hemoglobin   All 79999    0     0 79999      0      0        0
12        LDL_chole   All 79999    0     0 79999      0      0        0
7               SBP   All 79999    0     0 79999      0      0        0
15 serum_creatinine   All 79999    0     0 79999      0      0        0
17         SGOT_ALT   All 79999    0     0 79999      0      0        0
16         SGOT_AST   All 79999    0     0 79999      0      0        0
5        sight_left   All 79999    0     0 79999      0      0        0
6       sight_right   All 79999    0     0 79999      0      0        0
10        tot_chole   All 79999    0     0 79999      0      0        0
13     triglyceride   All 79999    0     0 79999      0      0        0
4         waistline   All 79999    0     0 79999      0      0        0
3            weight   All 79999    0     0 79999      0      0        0
   Per_of_Missing        sum   min    max   mean median     SD   CV  IQR
1               0  3813730.0  20.0   85.0  47.67   50.0  14.22 0.30 25.0
9               0  8037556.0  33.0  769.0 100.47   96.0  24.51 0.24 17.0
8               0  6080231.0  34.0  150.0  76.00   76.0   9.87 0.13 12.0
18              0  2941203.0   1.0  999.0  36.77   23.0  49.35 1.34 23.0
11              0  4562319.0   1.0 8110.0  57.03   55.0  32.29 0.57 20.0
2               0 12976735.0 130.0  190.0 162.21  160.0   9.31 0.06 15.0
14              0  1137771.5   4.1   22.0  14.22   14.2   1.59 0.11  2.2
12              0  9045010.0   1.0 5119.0 113.06  111.0  39.37 0.35 46.0
7               0  9793196.0  73.0  240.0 122.42  120.0  14.55 0.12 19.0
15              0    68706.9   0.1   35.0   0.86    0.8   0.31 0.36  0.3
17              0  2055293.0   1.0  671.0  25.69   20.0  21.46 0.84 14.0
16              0  2074566.0   1.0  750.0  25.93   23.0  16.29 0.63  9.0
```

```
5               0    78416.4   0.1    9.9   0.98    1.0   0.60 0.61  0.5
6               0    78529.9   0.1    9.9   0.98    1.0   0.62 0.63  0.5
10              0 15641766.0  60.0 1736.0 195.52  193.0  39.03 0.20 50.0
13              0 10552756.0   1.0 9490.0 131.91  106.0 106.67 0.81 85.0
4               0  6491807.5  30.0  999.0  81.15   81.0  11.55 0.14 13.2
3               0  5056675.0  30.0  130.0  63.21   60.0  12.50 0.20 15.0
   Skewness Kurtosis   25%   75% LB.25% UB.75% nOutliers
1      0.15    -0.59  35.0  60.0  -2.50  97.50         0
9      4.81    45.14  88.0 105.0  62.50 130.50      5180
8      0.39     0.85  70.0  82.0  52.00 100.00      1049
18     7.75    98.67  16.0  39.0 -18.50  73.50      7478
11   194.19 48380.39  46.0  66.0  16.00  96.00      1085
2     -0.02    -0.54 155.0 170.0 132.50 192.50         4
14    -0.38     0.69  13.2  15.4   9.90  18.70       778
12    27.24  3314.26  89.0 135.0  20.00 204.00       804
7      0.49     1.06 112.0 131.0  83.50 159.50      1293
15    28.23  2259.46   0.7   1.0   0.25   1.45       673
17     6.73   101.53  15.0  29.0  -6.00  50.00      5945
16    12.72   354.62  19.0  28.0   5.50  41.50      5534
5     10.05   147.83   0.7   1.2  -0.05   1.95       903
6      9.97   141.35   0.7   1.2  -0.05   1.95       894
10     2.23    66.58 169.0 219.0  94.00 294.00       916
13    14.02   858.23  73.0 158.0 -54.50 285.50      4481
4     25.16  1990.82  74.0  87.2  54.20 107.00       451
3      0.58     0.36  55.0  70.0  32.50  92.50      1538
```

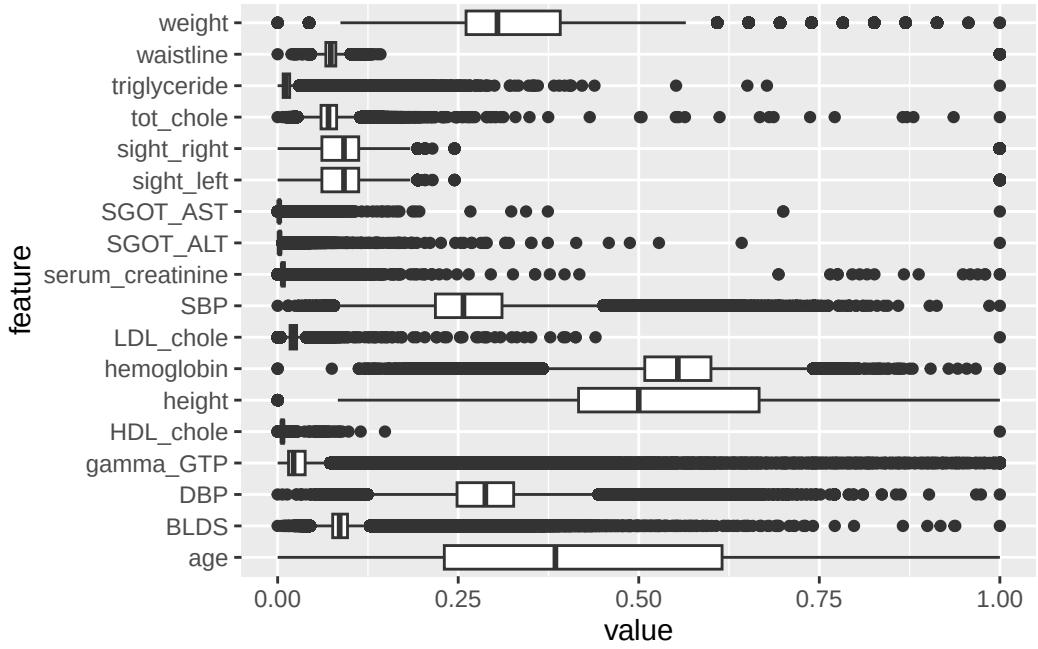
SGOT_ALT and SGOT_AST have a CV around 100% (102% and 90% respectively), gamma_GTP' CV measures 1.36 several variables have a very high skewness and kurtosis and quite a high number of outliers. Generally speaking, based on these statistics, it seems there's quite a clear need for feature transformation.

Come calcola gli outlier???

visualizzo "a mano" gli outlier. Non so se s'è qualche funzione in SmartEDA dal grafico dei boxplot mi sembra emerga che molti outlier sono davvero fuori scala. Servirebbe un approfondimento per capire se sono errori di registrazione o anomalie "accettabili" in quanto plausibili (patologie?). Io, non potendo valutare, prendo per buoni i dati

```
Data|>
  dplyr::select(where(is.numeric))|>
  rescale(to=c(0,1))|> # pacchetto datawizard
  as_tibble()|>
  pivot_longer(cols=everything(),
               names_to="feature",
```

```
                values_to="value")|>
  ggplot(aes(x = feature, y = value))+
  geom_boxplot()+
  coord_flip()
```



0.2.2.2 Categorical features

```
DataSmartEda_cat <- ExpCatStat(trainData,
                               Target="drkStatus",
                               result="stat",
                               clim=10,
                               nlim=5,
                               bins=10,
                               Pclass="Y",
                               plot=FALSE,
                               top=20,
                               Round=2)|>
  rename(PredPower='Predictive Power')|>
  dplyr::filter(PredPower=="Highly Predictive")
```

```
Warning in FUN(X[[i]], ...): NA introdotti per coercizione
```

```
DataSmartEda_cat
```

```
    Variable    Target Unique Chi-squared p-value df IV Value Cramers V
1        sex drkStatus      2   10329.654       0 NA     0.55       0.36
2 smk_status drkStatus      3   10350.076       0 NA     0.56       0.36
3        age drkStatus      9    6847.839       0 NA     0.35       0.29
4     height drkStatus      7   11263.846       0 NA     0.69       0.38
5     weight drkStatus      8    5879.255       0 NA     0.30       0.27
6 hemoglobin drkStatus     10    7836.990       0 NA     0.42       0.31
7  gamma_GTP drkStatus     10    7096.551       0 NA     0.39       0.30
  Degree of Association         PredPower
1                Strong Highly Predictive
2                Strong Highly Predictive
3              Moderate Highly Predictive
4                Strong Highly Predictive
5              Moderate Highly Predictive
6                Strong Highly Predictive
7                Strong Highly Predictive
```

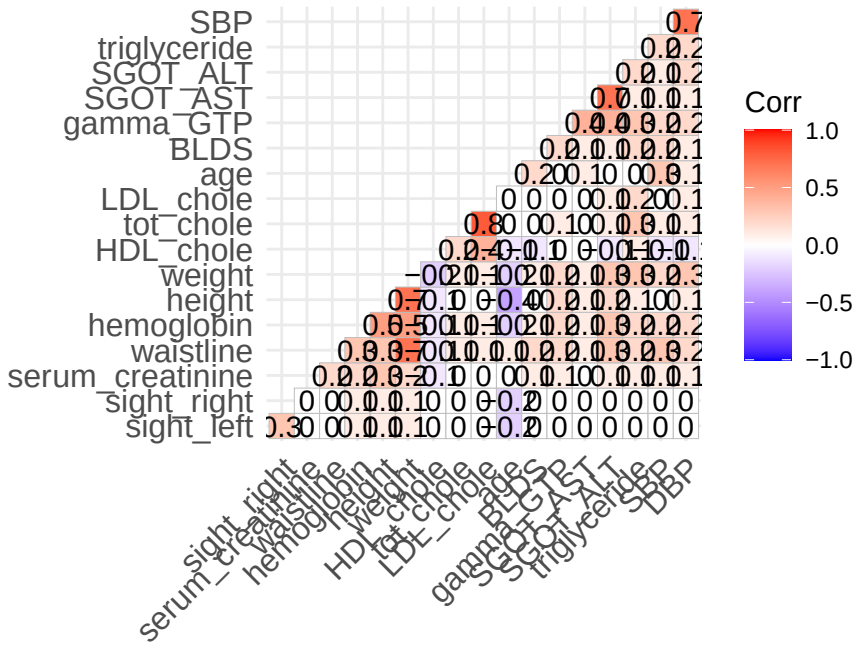
To sum up categoriacal analysis: 1. We can see six predictors that seem to have a highly predictive power. 2. Compared to decision tree's most important variable, five out of six are the same. 3. Serum_creatinine is considered amongst the top 6 by decision tree, while in Information Value's ranking it gets a "somewhat predictive"/"moderate" predictive power. 4. On the other hand, gamma_GTP is more considered by Information value while reaches the eight place only in decision tree's important variables ranking. 5. Amongst these six feature only height and weight seems pretty much correlated (0.7). Very, very curiously, height is more collerated than weight to drinking habit (the target value). 6. I will "merge" these two feature into one (body mass index) in order to use the information of both with only one predictor.

Categorical variables analysis leads towards following points: - 1. According to Cramer's v index no strong association between the target feature (drkStatus) and predictors is detected. - 2. On the other hand Information Value seems to show some strong relations between some indipendent features and the outcome.

0.2.3 Correlation between numerical predictors

```
corMatr <- round(cor(trainData[,-c(1,8,9,18,23,24)], use="complete.obs"), 1)

ggcorrplot(corMatr,
           hc.order = TRUE,
           type = "lower",
           lab = TRUE)
```



Only a few numerical features seem to be significantly mutually correlated: - tot_chole and LDL_chole (0.9), - height and weight (0.7), - SBP and DBP (0.7). There could be some room for feature selection.

0.2.4 Principal Component Analysis (factoextra approach)

0.2.4.1 Run and explore the PCA

```
dsPca2 <- trainData|>
  dplyr::select(where(is.numeric))|>
  prcomp(scale=TRUE)

summary(dsPca2)
```

```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7
Standard deviation     1.9170 1.4584 1.4109 1.27541 1.10753 1.01832 0.96567
Proportion of Variance 0.2041 0.1182 0.1106 0.09037 0.06815 0.05761 0.05181
Cumulative Proportion  0.2041 0.3223 0.4329 0.52327 0.59142 0.64903 0.70083
                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
Standard deviation     0.94929 0.88607 0.85389 0.82180 0.76756 0.75266 0.62535
Proportion of Variance 0.05006 0.04362 0.04051 0.03752 0.03273 0.03147 0.02173
Cumulative Proportion  0.75090 0.79451 0.83502 0.87254 0.90527 0.93674 0.95847
```

```
                          PC15    PC16    PC17    PC18
Standard deviation     0.4894 0.48019 0.41398 0.32573
Proportion of Variance 0.0133 0.01281 0.00952 0.00589
Cumulative Proportion  0.9718 0.98458 0.99411 1.00000
```

According to variance explained by Principal Components, it seems there is little room for a strong feature reduction via PCA. We need to get till the ninth PC to explain 80% of variance. We should expect that models that require few predictors (i.e. SVM) could not perform very well.

0.3 MODELING

In this project, I want to try a gradient boosting model, namely XGBoost. According to the tidymodels approach, I will define a recipe that defines the target variable and its relation with other features, a preprocessing recipe, a model (already spotted in the XGBoost) with hyperparameters to tune (with cross validation); once we have found the optimal values of hyperparameters, we fit the model and then compute evaluation metrics. Finally, the model will be explained with the SHAP approach.

0.3.1 Preprocessing recipe

Preprocessing steps that I want to take are the following: - winsorization of numerical features in order to handle outliers, - target encoding of categorical features in order to manage numerical features only.

```
xgbTuneRecipe <- trainData|>
  recipe(drkStatus ~.)|>
  step_mutate(winsHeight=DescTools::Winsorize(height),
              winsWeight=DescTools::Winsorize(weight),
              winsWaistline=DescTools::Winsorize(waistline),
              winsSight_left=DescTools::Winsorize(sight_left),
              winsSight_right=DescTools::Winsorize(sight_right),
              winsSBP=DescTools::Winsorize(SBP),
              winsDBP=DescTools::Winsorize(DBP),
              winsBLDS=DescTools::Winsorize(BLDS),
              winsTot_chole=DescTools::Winsorize(tot_chole),
              winsHDL_chole=DescTools::Winsorize(HDL_chole),
              winsLDL_chole=DescTools::Winsorize(LDL_chole),
              winsTriglyceride=DescTools::Winsorize(triglyceride),
              winsHemoglobin=DescTools::Winsorize(hemoglobin),
              winsSerum_creatinine=DescTools::Winsorize(serum_creatinine),
```

```
              winsSGOT_AST=DescTools::Winsorize(SGOT_AST),
              winsSGOT_ALT=DescTools::Winsorize(SGOT_ALT),
              winsGamma_GTP=DescTools::Winsorize(gamma_GTP))|>
  step_rm(c("height",
            "weight",
            "waistline",
            "sight_left",
            "sight_right",
            "SBP",
            "DBP",
            "BLDS",
            "tot_chole",
            "HDL_chole",
            "LDL_chole",
            "triglyceride",
            "hemoglobin",
            "serum_creatinine",
            "SGOT_AST",
            "SGOT_ALT",
            "gamma_GTP"))|>
  step_lencode_mixed(all_nominal_predictors(),
                     outcome=vars(drkStatus))
```

After setting the recipe, I create the preprocessed matrix (mandotory for XGBoost model) of predictors both to check recipe feedback and to use for shapviz once the final model will be fitted. The preprocessing steps are applied to the training data.

```
xgbTunePrep <- xgbTuneRecipe|>
  prep(training=trainData)|>
  bake(new_data=NULL)|>
  dplyr::select(-drkStatus)|>
  as.matrix()
```

0.3.2 Modeling

First, I set the hyperparameters to tune, with cross-validation. I create a grid with 6 combination of hyperparameters to tune.

```
xgbTuneModel <- boost_tree()|>
  set_args(tree_depth = tune(),
           min_n = tune(),
           loss_reduction = tune(),
           sample_size = tune(),
```

```
            mtry = tune(),
            learn_rate = tune())|>
  set_engine("xgboost")|>
  set_mode("classification")

set.seed(123, sample.kind = "Rounding")
```

```
Warning in set.seed(123, sample.kind = "Rounding"): non-uniform 'Rounding'
sampler used
```

```
xgbTuneFold <- vfold_cv(trainData, v=10)

xgbTuneGrid <- grid_max_entropy(
  tree_depth(),
  min_n(),
  loss_reduction(),
  sample_size = sample_prop(),
  finalize(mtry(), trainData),
  learn_rate(),
  size = 6)

xgbTuneModel <- boost_tree()|>
  set_args(tree_depth = tune(),
            min_n = tune(),
            loss_reduction = tune(),
            sample_size = tune(),
            mtry = tune(),
            learn_rate = tune())|>
  set_engine("xgboost")|>
  set_mode("classification")

set.seed(123, sample.kind = "Rounding")
```

```
Warning in set.seed(123, sample.kind = "Rounding"): non-uniform 'Rounding'
sampler used
```

```
xgbTuneFold <- vfold_cv(trainData, v=10)

xgbTuneFold

#  10-fold cross-validation
# A tibble: 10 x 2
```

```
   splits               id
   <list>               <chr>
 1 <split [71999/8000]> Fold01
 2 <split [71999/8000]> Fold02
 3 <split [71999/8000]> Fold03
 4 <split [71999/8000]> Fold04
 5 <split [71999/8000]> Fold05
 6 <split [71999/8000]> Fold06
 7 <split [71999/8000]> Fold07
 8 <split [71999/8000]> Fold08
 9 <split [71999/8000]> Fold09
10 <split [72000/7999]> Fold10
```

```
xgbTuneGrid <- grid_max_entropy(
  tree_depth(),
  min_n(),
  loss_reduction(),
  sample_size = sample_prop(),
  finalize(mtry(), trainData),
  learn_rate(),
  size = 10)
```

0.3.3 Workflow creation + fitting + best result picking + model finalizing

```
xgbTuneWorkflow <- workflow()|>
  add_recipe(xgbTuneRecipe)|>
  add_model(xgbTuneModel)

xgbTuneFit <- xgbTuneWorkflow|>
  tune_grid(resamples = xgbTuneFold,
            grid = xgbTuneGrid)

xgbTuneFit|>
  collect_metrics()
```

```
# A tibble: 20 x 12
    mtry min_n tree_depth learn_rate loss_reduction sample_size .metric
   <int> <int>      <int>      <dbl>          <dbl>       <dbl> <chr>
 1    24     7         14   9.71e- 2       4.06e- 2       0.719 accuracy
 2    24     7         14   9.71e- 2       4.06e- 2       0.719 roc_auc
 3     5    31          4   3.09e- 2       1.33e-10       0.627 accuracy
 4     5    31          4   3.09e- 2       1.33e-10       0.627 roc_auc
```

```
 5     9    31           6   1.80e- 6       1.42e- 4       0.260 accuracy
 6     9    31           6   1.80e- 6       1.42e- 4       0.260 roc_auc
 7     8    33          14   3.69e- 3       2.26e- 7       0.348 accuracy
 8     8    33          14   3.69e- 3       2.26e- 7       0.348 roc_auc
 9    23    35          13   1.52e- 3       2.12e- 4       0.309 accuracy
10    23    35          13   1.52e- 3       2.12e- 4       0.309 roc_auc
11    11    27          15   6.65e- 9       4.40e- 9       0.557 accuracy
12    11    27          15   6.65e- 9       4.40e- 9       0.557 roc_auc
13     4     8           1   2.00e- 6       1.02e- 7       0.145 accuracy
14     4     8           1   2.00e- 6       1.02e- 7       0.145 roc_auc
15    12    32          10   2.30e- 5       5.84e- 2       0.755 accuracy
16    12    32          10   2.30e- 5       5.84e- 2       0.755 roc_auc
17    19    29          11   5.60e-10       2.54e- 6       0.378 accuracy
18    19    29          11   5.60e-10       2.54e- 6       0.378 roc_auc
19    22    10           5   7.87e- 5       1.31e+ 0       0.263 accuracy
20    22    10           5   7.87e- 5       1.31e+ 0       0.263 roc_auc
# i 5 more variables: .estimator <chr>, mean <dbl>, n <int>, std_err <dbl>,
#   .config <chr>
```

```
xgbTunebest <- xgbTuneFit|>
  select_best("accuracy")

xgbTuneWorkflowFinal <-
  xgbTuneWorkflow|>
  finalize_workflow(xgbTunebest)

xgbTuneFinal <- xgbTuneWorkflowFinal|>
  fit(trainData)
```

0.3.4 Prediction and evaluation

```
xgbTunePrediction <- xgbTuneFinal|>
  predict(testData)|>
  bind_cols(testData)|>
  relocate(drkStatus, .after = .pred_class)

conf_mat(data=xgbTunePrediction,
         truth=drkStatus,
         estimate=.pred_class,
         dnn=c("Prediction", "Truth"),
         case_weights=NULL)
```

```
          Truth
Prediction    Y    N
         Y 7433 2895
         N 2538 7135
```

```
xgbTuneMetrics <- xgbTuneFinal|>
  predict(testData)|>
  bind_cols(testData)|>
  metrics(truth=drkStatus,
          estimate=.pred_class)
```

```
multiMetrics <- metric_set(accuracy, kap, f_meas, ppv, sens)
```

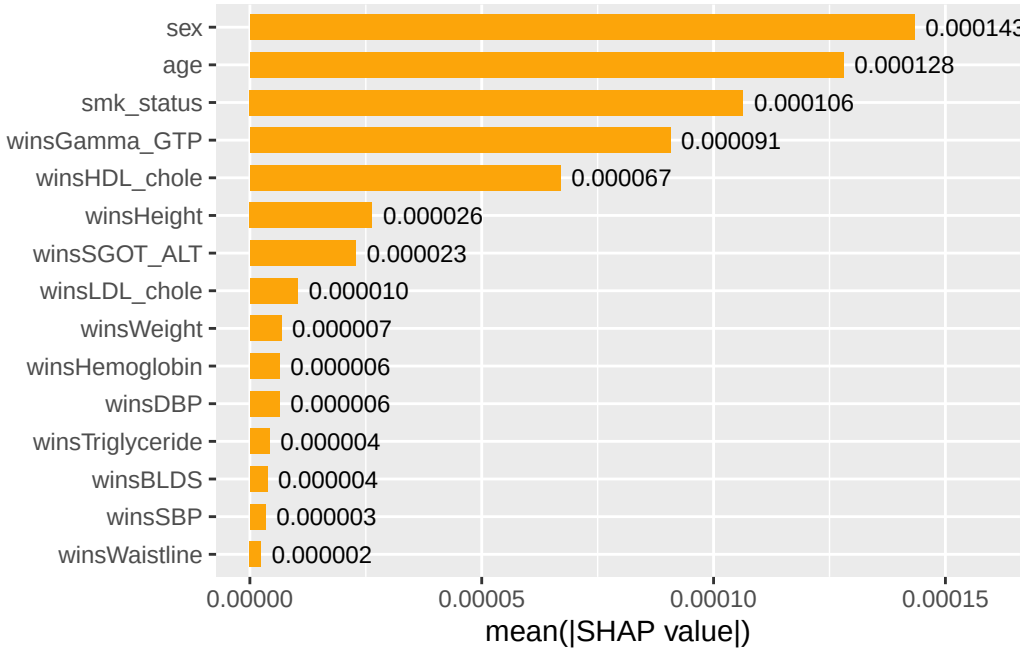
```
xgbTuneMultiMetrics <- xgbTuneFinal|>
  predict(testData)|>
  bind_cols(testData)|>
  multiMetrics(truth=drkStatus,
               estimate=.pred_class)
```

0.3.5 Model interpretation

```
xgbTuneEngine <- extract_fit_engine(xgbTuneFinal)
```

```
xgbTuneShap <- shapviz(object=xgbTuneEngine,
                       X_pred=xgbTunePrep)
```

```
sv_importance(xgbTuneShap, kind = "bar", show_numbers = TRUE)
```


sex
age
smk_status
winsGamma_GTP
winsHDL_chole
winsHeight
winsSGOT_ALT
winsLDL_chole
winsWeight
winsHemoglobin
winsDBP
winsTriglyceride
winsBLDS
winsSBP
winsWaistline
0.000143
0.000128
0.000106
0.000091
0.000067
0.000026
0.000023
0.000010
0.000007
0.000006
0.000006
0.000004
0.000004
0.000003
0.000002
0.00000
0.00005
0.00010
0.00015
mean(|SHAP value|)