Task on Modeling

Ignacio Almodóvar & Alejandra Estrada

03/22/2022

Preprocess

First of all, we load the data.

```
data=read.csv2("drug200.csv",sep = ",",header = TRUE)
```

Then we search for missing values.

```
missingValues=function(data){
  count=0
  a=cbind(lapply(lapply(data, is.na), sum))
  for(i in 1:ncol(data)){
    if(a[i]!=0){
      cat("There are", a[i], "missing values in column ", i,"\n" )
      count=count+1
    }
  }
  if(count==0){
    cat("There are no missing values in this dataset")
  }
}
missingValues(data)
```

```
## There are no missing values in this dataset
```

As we can see, the summary says that all variables except Age are factors. However, analyzing the data the variable "Na_to_k" looks like a numeric variable so we must change it.

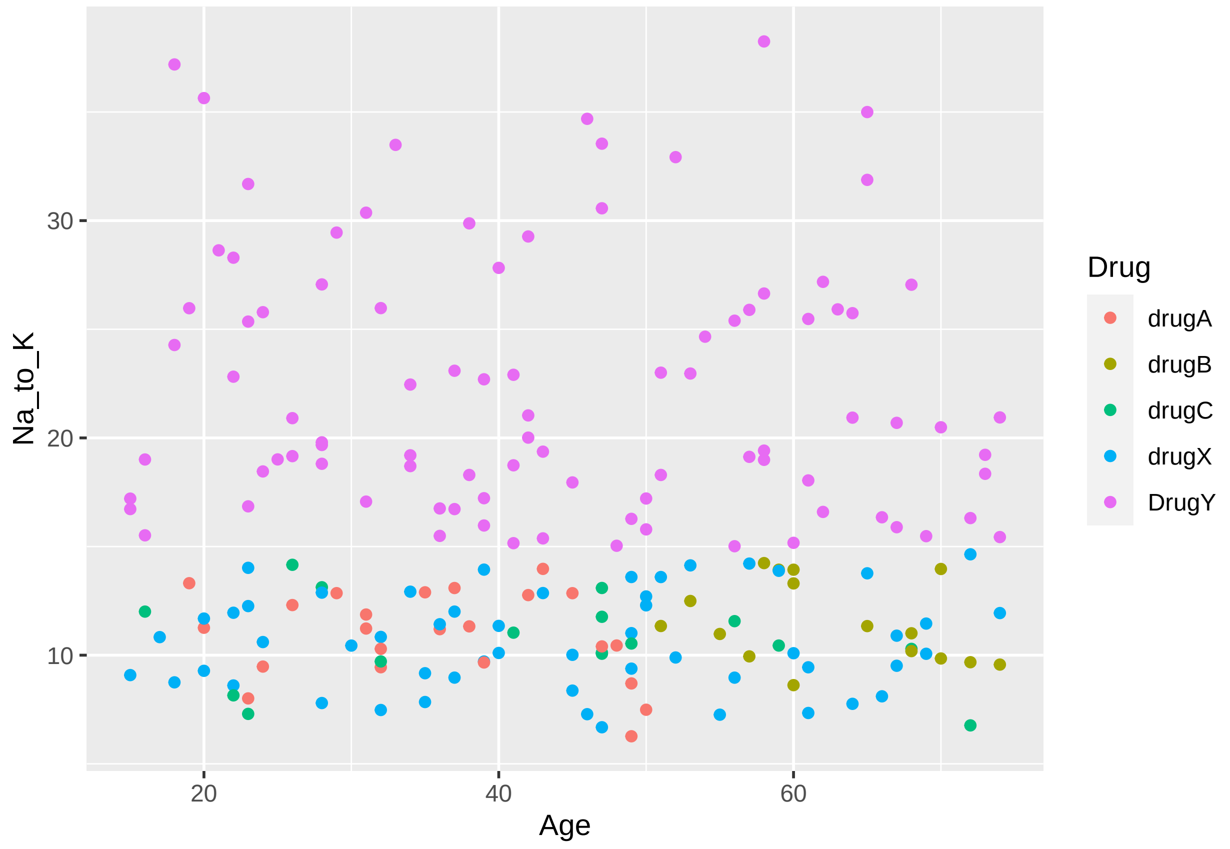
```
data[,5]  %<>% as.numeric()
summary(data)
```

```
##       Age            Sex                 BP            Cholesterol
##  Min.   :15.00   Length:200         Length:200         Length:200
##  1st Qu.:31.00   Class :character   Class :character   Class :character
##  Median :45.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   :44.31
##  3rd Qu.:58.00
##  Max.   :74.00
##     Na_to_K           Drug
##  Min.   : 6.269   Length:200
##  1st Qu.:10.445   Class :character
##  Median :13.937   Mode  :character
##  Mean   :16.084
##  3rd Qu.:19.380
##  Max.   :38.247
```

H2O

We now plot the continuous variables to see if we can find any group evidences for the type of drug

```
ggplot(data,aes(Age,Na_to_K,col=Drug)) + geom_point()
```



Beforehand there is not clear evidence for the differentiation in groups given the age and the Na_t_k. However, as can be seen, for the DrugY there is a clear bandwidth for Na_to_k being higher than 15.

We are now going to fitt a classification model using h2o package.

```
table(data$Drug)
h2o.init()
data_h2o=as.h2o(data)
resp_data="Drug"
pred_data=setdiff(names(data_h2o), resp_data)

setdiff(names(data_h2o), resp_data)
```

```
data_h2o[, resp_data] <- as.factor(data_h2o[, resp_data])

splits = h2o.splitFrame(data = data_h2o, ratios = 0.8, seed = 42)
train = splits[[1]]
test = splits[[2]]

# Run AutoML
aml_mul = h2o.automl(x = pred_data, y = resp_data, training_frame = train, leaderboard_frame = test,
                      include_algos = c("GLM", "XGBoost", "DeepLearning","DRF","GBM","StackedEnsemble")
```

If we do leaderboard we obtain the next results:

```
lb_mul <- h2o.get_leaderboard(aml_mul)
head(lb_mul)
```

```
##                                                  model_id mean_per_class_error
## 1    DeepLearning_grid_1_AutoML_1_20220324_12141_model_1            0.4825397
## 2            GBM_grid_1_AutoML_1_20220324_12141_model_56            0.5111111
## 3 StackedEnsemble_BestOfFamily_4_AutoML_1_20220324_12141            0.5158730
## 4   DeepLearning_grid_1_AutoML_1_20220324_12141_model_12            0.5285714
## 5    DeepLearning_grid_1_AutoML_1_20220324_12141_model_4            0.5333333
## 6            GBM_grid_1_AutoML_1_20220324_12141_model_39            0.5333333
##     logloss      rmse       mse
## 1 1.6372224 0.5839158 0.3409576
## 2 1.0631995 0.5985999 0.3583218
## 3 1.2275490 0.5479127 0.3002083
## 4 0.8657503 0.5753320 0.3310069
## 5 0.8648328 0.5440349 0.2959740
## 6 1.4979195 0.6284823 0.3949900
```

The classification is not really good. The `mean_per_class_error` of the best model of type `"DeepLearning"`, based on fully-connected multilayer artificial neural network, is 0.4015873. As a comparison, the mean-per-class error by "weighted guessing" is:

```
probs <- table(as.matrix(data$Drug))
probs <- probs / sum(probs)
(mean(1 - probs))
```

```
## [1] 0.8
```

And th probability of correct classification by pure chance is:

```
(sum(probs^2))
```

```
## [1] 0.30595
```

We do prediction in the test dataset to see that the prediction is not very good.

```
pred_mul <- h2o.predict(object = aml_mul, newdata = test)
```

```
##   |                                                                               |
h2o.head(pred_mul)
```

```
##   predict        DrugY         drugA         drugB         drugC         drugX
## 1   drugC 5.806396e-13  5.251812e-02  2.892658e-01  3.781932e-01  2.800229e-01
## 2   drugC 2.193186e-10  5.549356e-02  3.145458e-01  3.420578e-01  2.879028e-01
## 3   DrugY 1.000000e+00 1.569431e-200 1.687176e-164 3.466728e-137 3.504394e-107
## 4   DrugY 9.989732e-01  4.664399e-09  1.191727e-07  1.583093e-09  1.026640e-03
## 5   drugX 4.101541e-17  4.644084e-06  1.650876e-01  1.102436e-02  8.238834e-01
## 6   drugB 5.237041e-15  2.195299e-02  3.760536e-01  3.664320e-01  2.355614e-01
```

And we check the accuracy of the label assignments with the real labels

```
labels_mul <- as.matrix(pred_mul$predict)
table(labels_mul, as.matrix(test$Drug))
```

```
## 
## labels_mul drugA drugB drugC drugX DrugY
##      drugA     1     0     0     0     0
##      drugB     0     3     0     2     0
##      drugC     3     0     1     3     0
##      drugX     3     3     1     4     0
##      DrugY     0     0     0     0    14
```

As we expected after observing `h2o.head(pred_mul)` the only label that has been assigned correctly is that of the variable `DrugY`. The rest labels give us very unfavorable results.

So we can conclude that none of these models is very useful since the classification is poor.

To end, we will explain the leader model compare with all AutoML models.

```
ex_mul <- h2o.explain(object = aml_mul, newdata = test)
```

The explainers clearly point to `Na_to_K` being the most relevant predictor. The partial dependence plots are not so useful in this case, we can only conclude that the partial dependence with highest mean response is the partial dependence on "Na_to_K" with target = "DrugY".

Finally, we close the h2o cluster:

```
h2o.shutdown(prompt = FALSE)
```

Tidymodels

```
boot_data <- bootstraps(data, times = 10)
analysis(boot_data$splits[[1]] )%>% head()
```

```
##     Age Sex     BP Cholesterol Na_to_K  Drug
## 103  28   F    LOW        HIGH  13.127 drugC
## 131  70   F NORMAL        HIGH  20.489 DrugY
## 156  49   M    LOW        HIGH  10.537 drugC
## 52   67   M NORMAL      NORMAL  10.898 drugX
## 19   23   M    LOW        HIGH   7.298 drugC
## 110  23   M NORMAL        HIGH  16.850 DrugY
```

Parsnip

```r
library(tidymodels)

# Create an initial split stratifying by the response
set.seed(42)
data_split <- initial_split(data, prop = 0.75)
ames_train <- training(data_split)
ames_test <- testing(data_split)

ames_train$Drug %<>% as.factor()

mnr_spec <- multinom_reg(penalty = 0.1) %>%
  set_engine("nnet")
mnr_spec
```

```
## Multinomial Regression Model Specification (classification)
##
## Main Arguments:
##   penalty = 0.1
##
## Computational engine: nnet
```

```r
mnr_fit <- mnr_spec %>%
  fit(Drug ~ ., data = ames_train)
mnr_fit
```

```
## parsnip model object
##
## Fit time:  0ms
## Call:
## nnet::multinom(formula = Drug ~ ., data = data, decay = ~0.1,
##     trace = FALSE)
##
## Coefficients:
##       (Intercept)          Age        SexM     BPLOW  BPNORMAL
## drugB  -2.2722165  0.152748559 -1.20844566 -1.946016 -1.744401
## drugC   1.6520972  0.004857262 -1.13182828  5.085658 -0.285746
## drugX   0.4929378  0.023586655 -0.89276664  4.297885  5.907929
## DrugY  -8.0458019 -0.004643564 -0.07080169  1.089960  1.436271
##       CholesterolNORMAL    Na_to_K
## drugB          0.017088 -0.4091351
## drugC         -2.716356 -0.3381201
## drugX          1.990205 -0.3703790
## DrugY          0.194722  0.6001168
##
## Residual Deviance: 107.2972
## AIC: 163.2972
```

```r
test_results <- bind_cols(
  dplyr::select(ames_test, "Drug"),
  predict(mnr_fit, ames_test),
  predict(mnr_fit, ames_test, type = "prob")
)

table(test_results$Drug, test_results$.pred_class)
```

```
##
##          drugA drugB drugC drugX DrugY
##   drugA     4     1     0     0     0
##   drugB     0     2     0     0     0
##   drugC     0     0     7     0     0
##   drugX     0     0     0    12     0
##   DrugY     0     0     0     0    24
```

```
mean(test_results$Drug == test_results$.pred_class, na.rm = TRUE)
```

```
## [1] 0.98
```

Discrim

```
library(discrim)
```

```
## Warning: package 'discrim' was built under R version 4.1.2

##
## Attaching package: 'discrim'

## The following object is masked from 'package:dials':
##
##     smoothness
```

```
# Fit a Naive Bayes model (which is actually a kernel discriminant analysis done by combining univariat

summary(data)
```

```
##       Age            Sex                  BP             Cholesterol
##  Min.   :15.00   Length:200         Length:200         Length:200
##  1st Qu.:31.00   Class :character   Class :character   Class :character
##  Median :45.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   :44.31
##  3rd Qu.:58.00
##  Max.   :74.00
##     Na_to_K           Drug
##  Min.   : 6.269   Length:200
##  1st Qu.:10.445   Class :character
##  Median :13.937   Mode  :character
##  Mean   :16.084
##  3rd Qu.:19.380
##  Max.   :38.247
```

```r
data$Drug %<>% as.factor()
nb_mod <- naive_Bayes() %>%
  set_engine("naivebayes") %>%
  fit(Drug ~ ., data = data)
```

```
## Warning: naive_bayes(): Feature BP - zero probabilities are present. Consider
## Laplace smoothing.
```

```
## Warning: naive_bayes(): Feature Cholesterol - zero probabilities are present.
## Consider Laplace smoothing.
```