Block III

Sara Dovalo and Javier Muñoz Flores

21/3/2022

Contents

Introduction

The content of this section is, mainly, a further description of the dataset used as well as the exposition of the problem to solve.

The dataset selected contains several patient records of different medical measurements in order to predict wether a person is more likely to suffer a heart disease, i.e. a failure . The data has been retrieved from the public *kaggle* repository and it is the product of the combination of five different datasets from different regions of EEUU. The final dataset contains in total 918 instances and 12 attributes, which 7 of them are categorical and the five remaining are numerical:

- `Age`(*quantitative*): age of the patient in years
- `Sex`(*qualitative*): sex of the patient [M: Male, F: Female]
- `ChestPainType`(*qualitative*): Angina type, i.e. chest pain, frequently caused when the heart muscle is not able to get enough oxygen-rich blood [*TA*: Typical Angina, *ATA*: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
- `RestingBP`(*quantitative*): resting blood pressure [mm Hg]. A normal level is less than 180 mm Hg.
- `Cholesterol`(*quantitative*): serum cholesterol [mm/dl]
- `FastingBS`(*qualitative*): fasting blood sugar [1: if FastingBS $>$ 120 mg/dl, 0: otherwise]
- `RestingECG`(*qualitative*): resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of $>$ 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
- `MaxHR`(*quantitative*): maximum heart rate achieved [Numeric value between 60 and 202]
- `ExerciseAngina`: exercise-induced angina [Y: Yes, N: No]
- `Oldpeak`(*quantitative*): oldpeak = ST [Numeric value measured in depression]
- `ST_Slope`(*qualitative*): the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
- `HeartDisease`(*qualitative*): class variable [1: heart disease, 0: Normal]

Clearly, it is a binary classification problem since the target variable has two levels.

Preprocessing

First of all, it is suitable to visualize the data and to identify if there are missing values which could add noise and disrupt the performance of the future models created.

```
# Count number of zeros (missing values)
heart.data %>% filter(Cholesterol == 0) %>% summarize(count = n())
```

```
##   count
## 1   172
```

We notice that the variable `Cholesterol` contains 172 values equal to zero. They must be considered as missing values as a person cannot that such a low level of cholesterol in blood. We decide to eliminate those observations that have a zero in that variable.

```
# Eliminate rows which contain 0 in variable Cholesterol
heart.data = heart.data %>%
filter(Cholesterol != 0)
# Number of missing values
heart.data %>% filter(Cholesterol == 0) %>% summarize(count = n())
```

```
##   count
## 1     0
```

Furthermore, since the dataset includes several categorical attributes, it is necessary to transform them into *factors*.

```
# Categorical variables as factors
heart.data = heart.data %>%
       mutate_each_(funs(factor(.)),c(2,3,6,7,9,11,12))
str(heart.data)
```

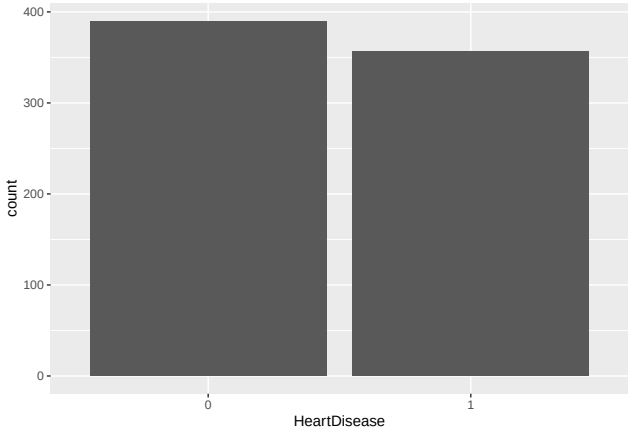
```
## 'data.frame':    746 obs. of  12 variables:
##  $ Age           : int  40 49 37 48 54 39 45 54 37 48 ...
##  $ Sex           : Factor w/ 2 levels "F","M": 2 1 2 1 2 2 1 2 2 1 ...
##  $ ChestPainType : Factor w/ 4 levels "ASY","ATA","NAP",..: 2 3 2 1 3 3 2 2 1 2 ...
##  $ RestingBP     : int  140 160 130 138 150 120 130 110 140 120 ...
##  $ Cholesterol   : int  289 180 283 214 195 339 237 208 207 284 ...
##  $ FastingBS     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ RestingECG    : Factor w/ 3 levels "LVH","Normal",..: 2 2 3 2 2 2 2 2 2 2 ...
##  $ MaxHR         : int  172 156 98 108 122 170 170 142 130 120 ...
##  $ ExerciseAngina: Factor w/ 2 levels "N","Y": 1 1 1 2 1 1 1 1 2 1 ...
##  $ Oldpeak       : num  0 1 0 1.5 0 0 0 0 1.5 0 ...
##  $ ST_Slope      : Factor w/ 3 levels "Down","Flat",..: 3 2 3 2 3 3 3 3 2 3 ...
##  $ HeartDisease  : Factor w/ 2 levels "0","1": 1 2 1 2 1 1 1 1 2 1 ...
```

To finish this section, we check if the classes of the response `HeartDisease` are balanced or not. We visualize it through a barplot to see it more clearly.

```
# Count the observation of each class
heart.data %>%
count(HeartDisease)
```

```
##   HeartDisease   n
## 1            0 390
## 2            1 356
```

```
# Barplot
ggplot(heart.data, aes(HeartDisease)) + geom_bar()
```



The classes are balanced, it will be not needed to over/undersampling the sample.

Modelling with H2O package

We follow the same procedure that we did in classroom for using `h2o.automl()`, i.e. fitting, benchmarking, predicting and explaining, in that order.

Fitting

Before starting with the selection of the models, the *local H2O cluster* is initialized in order to leverage the parallelization in the virtual machine which the package provides and to use H2O functions.

The first step is to convert the data into a `h2o`object and then, to identify the names of the predictors as well as the name of the response.

Then, it is important to mention that there will not be splitting into train and test, since we will use cross-validation metrics on the leaderboard model. Thus, we have to specify the number of folds (in our case will be `nfolds = 8`) needed to the cross-validation process. However, we do not have to indicate the predictors names, instead only the`dataFrame`and the response variable. In addition, we do not exclude any algorithm in `h2o.automl()` method, but we limit the time for fitting the models (`max_runtime_secs = 30` and `max_runtime_secs_per_model = 5`)

```
# Initialize h2o
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         19 hours 31 minutes
##     H2O cluster timezone:       Europe/Madrid
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.36.0.3
##     H2O cluster version age:    1 month and 6 days
##     H2O cluster name:           H2O_started_from_R_saradovalo_thv417
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.55 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     R Version:                  R version 4.0.2 (2020-06-22)
```

```
# Send data to local H2O cluster
data.h <- as.h2o(heart.data)
```

```
##   |                                                                              |
```

```
# Data summary
h2o.describe(data.h)
```

```
##             Label Type Missing Zeros PosInf NegInf  Min   Max        Mean
```

```
## 1             Age  int       0     0      0      0 28.0  77.0  52.8820375
## 2             Sex enum       0   182      0      0  0.0   1.0   0.7560322
## 3   ChestPainType enum       0   370      0      0  0.0   3.0          NA
## 4       RestingBP  int       0     0      0      0 92.0 200.0 133.0227882
## 5     Cholesterol  int       0     0      0      0 85.0 603.0 244.6353887
## 6       FastingBS enum       0   621      0      0  0.0   1.0   0.1675603
## 7      RestingECG enum       0   176      0      0  0.0   2.0          NA
## 8           MaxHR  int       0     0      0      0 69.0 202.0 140.2265416
## 9  ExerciseAngina enum       0   459      0      0  0.0   1.0   0.3847185
## 10        Oldpeak real       0   317      0      0 -0.1   6.2   0.9016086
## 11       ST_Slope enum       0    43      0      0  0.0   2.0          NA
## 12   HeartDisease enum       0   390      0      0  0.0   1.0   0.4772118
##         Sigma Cardinality
## 1   9.5058879          NA
## 2   0.4297617           2
## 3          NA           4
## 4  17.2827498          NA
## 5  59.1535237          NA
## 6   0.3737260           2
## 7          NA           3
## 8  24.5241072          NA
## 9   0.4868551           2
## 10  1.0728611          NA
## 11         NA           3
## 12  0.4998155           2
```

```
# Identify the names of the response and predictors
resp_h <- "HeartDisease"
pred_h<- setdiff(names(data.h), resp_h)
# Call h2o.automl()
model_h <- h2o.automl(y = resp_h, training_frame = data.h, max_runtime_secs = 30,max_runtime_secs_per_m
seed = 1, verbosity = NULL)
```

```
##   |                                                                              |
```

Benchmarking

In this section, we have to explore the leaderboard of models in order to carry out a first comparison.

```
# Leaderboard
lead_h <- model_h@leaderboard
names(lead_h)[5] <- "mpce" # Rename mean_per_class_error to shorten output
print(lead_h[, -6], n = nrow(lead_h)) # Exclude final column to fit the table in one page
```

```
##                                                model_id       auc   logloss
## 1  StackedEnsemble_BestOfFamily_3_AutoML_8_20220323_115340 0.9309349 0.3351113
## 2     StackedEnsemble_AllModels_2_AutoML_8_20220323_115340 0.9302651 0.3363152
## 3     StackedEnsemble_AllModels_1_AutoML_8_20220323_115340 0.9291595 0.3337144
## 4                           GBM_1_AutoML_8_20220323_115340 0.9289182 0.3419149
## 5  StackedEnsemble_BestOfFamily_2_AutoML_8_20220323_115340 0.9285941 0.3383963
## 6                           GBM_5_AutoML_8_20220323_115340 0.9283708 0.3411928
## 7                           GBM_2_AutoML_8_20220323_115340 0.9283420 0.3366764
```

```
## 8                           GBM_3_AutoML_8_20220323_115340 0.9281727 0.3353939
## 9  StackedEnsemble_BestOfFamily_1_AutoML_8_20220323_115340 0.9271175 0.3404670
## 10                      XGBoost_2_AutoML_8_20220323_115340 0.9270815 0.3430975
## 11                          DRF_1_AutoML_8_20220323_115340 0.9270059 0.5059250
## 12                          XRT_1_AutoML_8_20220323_115340 0.9268258 0.3601909
## 13                          GBM_4_AutoML_8_20220323_115340 0.9265557 0.3416805
## 14                      XGBoost_1_AutoML_8_20220323_115340 0.9260480 0.3426434
## 15                          GLM_1_AutoML_8_20220323_115340 0.9259399 0.3459442
## 16                      XGBoost_3_AutoML_8_20220323_115340 0.9225475 0.3626312
## 17         XGBoost_grid_1_AutoML_8_20220323_115340_model_1 0.9210494 0.3751298
## 18                DeepLearning_1_AutoML_8_20220323_115340 0.9130906 0.3836811
##        aucpr      mpce        mse
## 1  0.9119205 0.1306612 0.10143079
## 2  0.8985378 0.1270599 0.10007393
## 3  0.9007273 0.1272400 0.09991896
## 4  0.9069595 0.1268727 0.10349420
## 5  0.8940059 0.1231489 0.10147999
## 6  0.9033091 0.1332253 0.10407953
## 7  0.8987484 0.1309637 0.10135075
## 8  0.8989884 0.1293143 0.10064549
## 9  0.9042399 0.1295016 0.10292541
## 10 0.9044969 0.1241285 0.10278212
## 11 0.8964508 0.1338375 0.10443263
## 12 0.9131320 0.1400605 0.10955955
## 13 0.8962562 0.1309061 0.10246846
## 14 0.9007739 0.1313310 0.10344801
## 15 0.9070012 0.1382311 0.10631715
## 16 0.8850892 0.1398732 0.10639814
## 17 0.8871566 0.1456137 0.11313975
## 18 0.8878903 0.1407952 0.11186889
##
## [18 rows x 6 columns]
```

The leader has been a *StackedEnsemble* model with an error

Prediction

We select the leader of the models and predict a few rows.

```
h2o.predict(object = model_h@leader, newdata = data.h[1:8, ])
```

```
##   |                                                                              |
```

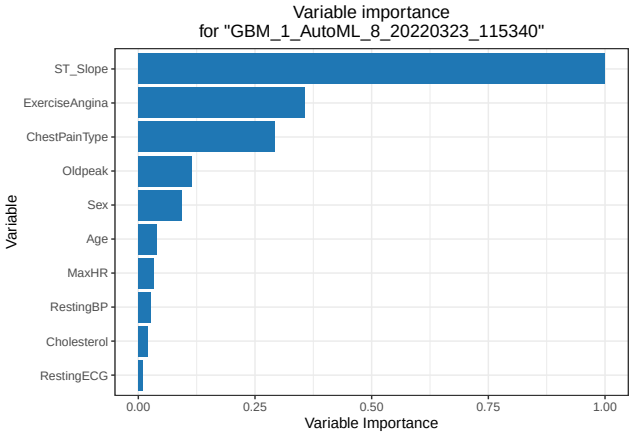
```
##   predict        p0         p1
## 1       0 0.9721761 0.02782389
## 2       1 0.5781783 0.42182172
## 3       0 0.9880889 0.01191107
## 4       1 0.1754888 0.82451121
## 5       0 0.9640488 0.03595125
## 6       0 0.9513570 0.04864298
##
## [8 rows x 3 columns]
```

Explanation

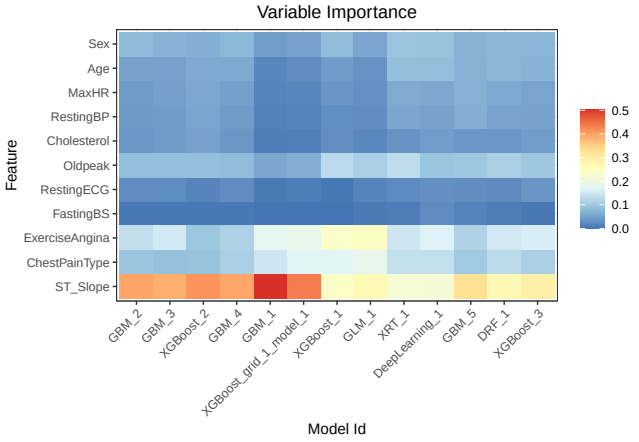
```
exp <- h2o.explain(object = model_h, newdata = data.h)
exp
```

```
##
##
## Leaderboard
## ===========
##
## > Leaderboard shows models with their metrics. When provided with H2OAutoML object, the leaderboard s
##
##
## |  | model_id | auc | logloss | aucpr | mean_per_class_error | rmse | mse | training_time_ms | predi
## |:---:|:---:|:---:|:---:|:---:|:---:|:---:|:---:|:---:|:---:|:---:|
## | **1** |StackedEnsemble_BestOfFamily_3_AutoML_8_20220323_115340 | 0.930934889080956 | 0.33511134879
## | **2** |StackedEnsemble_AllModels_2_AutoML_8_20220323_115340 | 0.930265053298761 | 0.33631522501332
## | **3** |StackedEnsemble_AllModels_1_AutoML_8_20220323_115340 | 0.929159464131374 | 0.33371439782068
## | **4** |GBM_1_AutoML_8_20220323_115340 | 0.928918179199078 | 0.341914907368858 | 0.906959474290582
## | **5** |StackedEnsemble_BestOfFamily_2_AutoML_8_20220323_115340 | 0.928594065110919 | 0.33839629327
## | **6** |GBM_5_AutoML_8_20220323_115340 | 0.928370786516854 | 0.341192794293264 | 0.903309148947157
## | **7** |GBM_2_AutoML_8_20220323_115340 | 0.928341976375684 | 0.33667635791848 | 0.898748420196508 |
## | **8** |GBM_3_AutoML_8_20220323_115340 | 0.928172716796312 | 0.335393930304343 | 0.898988422057164
## | **9** |StackedEnsemble_BestOfFamily_1_AutoML_8_20220323_115340 | 0.927117545375972 | 0.34046695423
## | **10** |XGBoost_2_AutoML_8_20220323_115340 | 0.92708153269951 | 0.343097483513869 | 0.904496899050
## | **11** |DRF_1_AutoML_8_20220323_115340 | 0.92700590607894 | 0.505924977634171 | 0.896450822614937
## | **12** |XRT_1_AutoML_8_20220323_115340 | 0.926825842696629 | 0.360190911690676 | 0.913131963879078
## | **13** |GBM_4_AutoML_8_20220323_115340 | 0.926555747623163 | 0.341680487314038 | 0.896256180736215
## | **14** |XGBoost_1_AutoML_8_20220323_115340 | 0.926047968885047 | 0.342643380745472 | 0.90077385047
## | **15** |GLM_1_AutoML_8_20220323_115340 | 0.925939930855661 | 0.34594420032564 | 0.907001227276824
## | **16** |XGBoost_3_AutoML_8_20220323_115340 | 0.92254753673293 | 0.36263117766026 | 0.8850892351447
## | **17** |XGBoost_grid_1_AutoML_8_20220323_115340_model_1 | 0.921049409392106 | 0.375129757917747 |
## | **18** |DeepLearning_1_AutoML_8_20220323_115340 | 0.913090607893979 | 0.383681064434899 | 0.887890
##
##
## Confusion Matrix
## ================
##
## > Confusion matrix shows a predicted class vs an actual class.
##
##
##
## StackedEnsemble_BestOfFamily_3_AutoML_8_20220323_115340
## --------------------------------------------------------
##
## |  | 0 | 1 | Error | Rate
## |:---:|:---:|:---:|:---:|:---:|
## | **0** |360 | 30 | 0.0769230769230769 |  =30/390 |
## | **1** |21 | 335 | 0.0589887640449438 |  =21/356 |
## | **Totals** |381 | 365 | 0.0683646112600536 |  =51/746 |
##
##
## Variable Importance
```

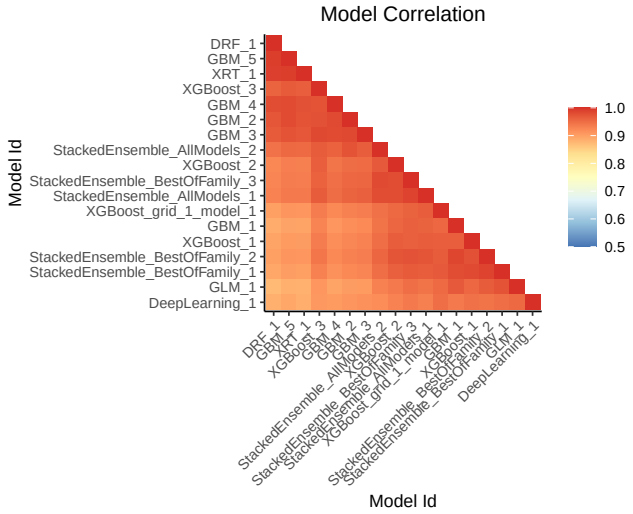
```
## ====================
##
## > The variable importance plot shows the relative importance of the most important variables in the r
```



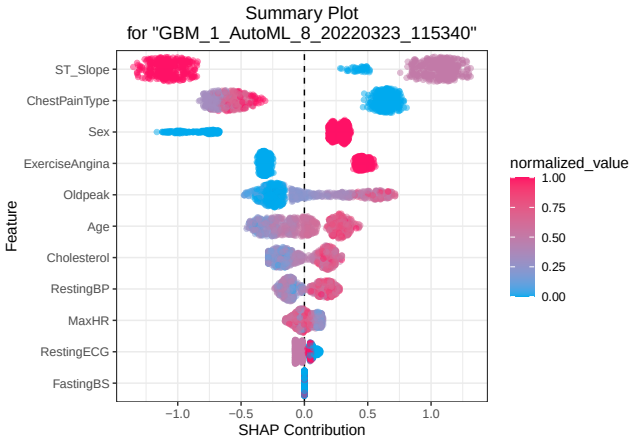

```
##
##
## Variable Importance Heatmap
## ============================
##
## > Variable importance heatmap shows variable importance across multiple models. Some models in H2O r
```



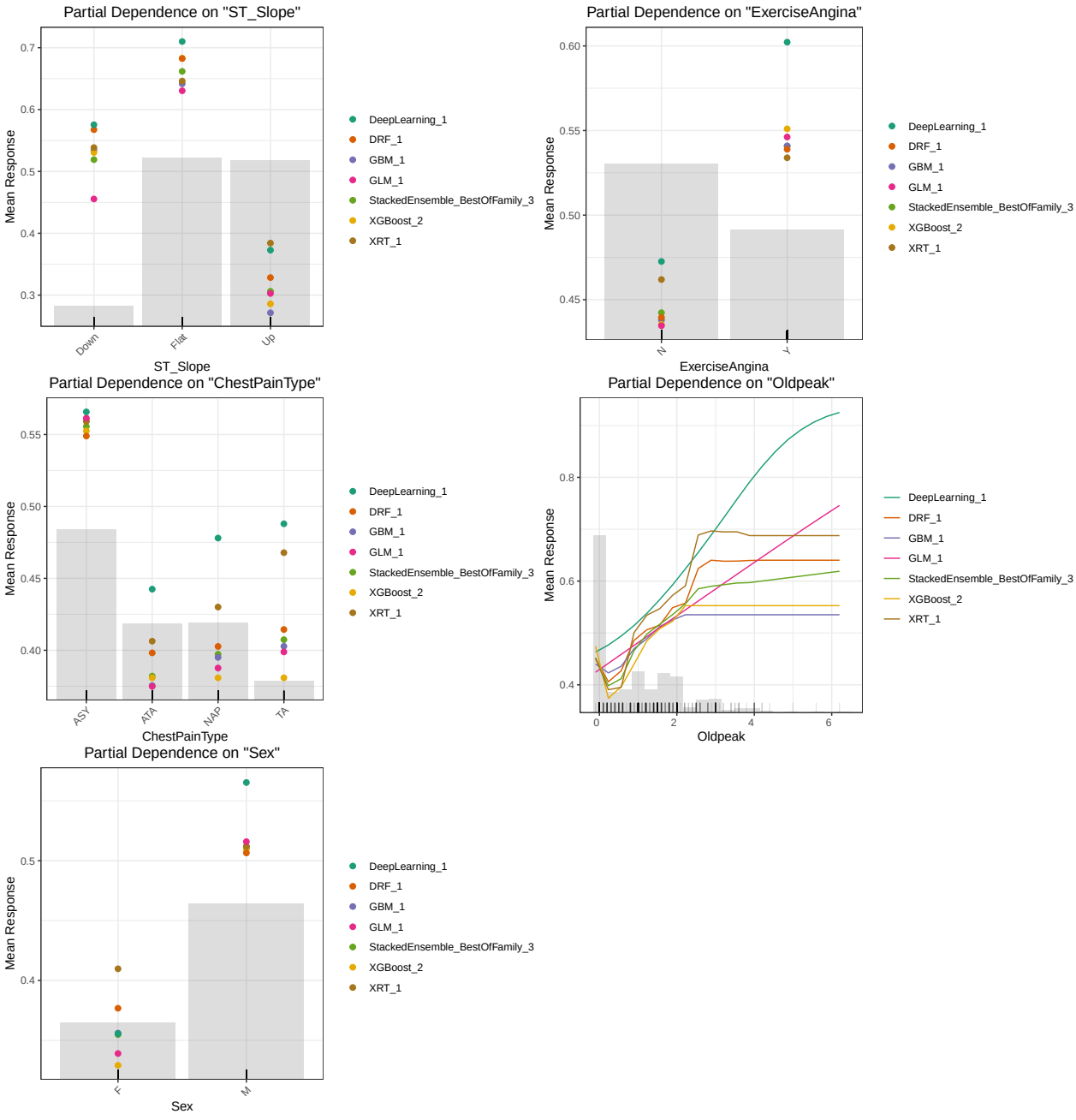

```
##
##
## Model Correlation
## ==================
##
## > This plot shows the correlation between the predictions of the models. For classification, frequen
```




```
## Interpretable models: GLM_1_AutoML_8_20220323_115340
##
##
## SHAP Summary
## ============
##
## > SHAP summary plot shows the contribution of the features for each instance (row of data). The sum
```




```
##
##
## Partial Dependence Plots
## ========================
##
## > Partial dependence plot (PDP) gives a graphical depiction of the marginal effect of a variable on
```


Partial Dependence on "ST_Slope"
Mean Response
0.7
0.6
0.5
0.4
0.3
Down
Flat
Up
ST_Slope
DeepLearning_1
DRF_1
GBM_1
GLM_1
StackedEnsemble_BestOfFamily_3
XGBoost_2
XRT_1
Partial Dependence on "ExerciseAngina"
Mean Response
0.60
0.55
0.50
0.45
N
Y
ExerciseAngina
DeepLearning_1
DRF_1
GBM_1
GLM_1
StackedEnsemble_BestOfFamily_3
XGBoost_2
XRT_1
Partial Dependence on "ChestPainType"
Mean Response
0.55
0.50
0.45
0.40
ASY
ATA
NAP
TA
ChestPainType
DeepLearning_1
DRF_1
GBM_1
GLM_1
StackedEnsemble_BestOfFamily_3
XGBoost_2
XRT_1
Partial Dependence on "Oldpeak"
Mean Response
0.8
0.6
0.4
0
2
4
6
Oldpeak
DeepLearning_1
DRF_1
GBM_1
GLM_1
StackedEnsemble_BestOfFamily_3
XGBoost_2
XRT_1
Partial Dependence on "Sex"
Mean Response
0.5
0.4
F
M
Sex
DeepLearning_1
DRF_1
GBM_1
GLM_1
StackedEnsemble_BestOfFamily_3
XGBoost_2
XRT_1

Modelling with `tidymodels` package

`Tidymodelsis` an interface that unifies hundreds of functions from different packages, facilitating all stages of pre-processing, training, optimisation and validation of predictive models. The main packages that are part of the tidymodels ecosystem are:`brrom`, `rsample`, `parsnip`, `discrim`, `corr` y `tidypredict` among others.

This package offers so many possibilities that they can hardly be shown with a single example.

Build a model

As the variable of interest `HeartDisease` is a binary variable, a logistic regression model is chosen, which will be our basis. For simplicity we only consider 7 of the 11 explanatory variables.

Preprocess our data with recipes

Evaluation of the model with resampling

Tune model parameters

Prediction