With the BMuCaret Shiny app you can perform Caret-models comparison and find the best one, which achieves the best performance for a given data set.

But isn't it true that “All models are wrong”?

By conducting comparison of predictive models (e.g. by using my tool BMuCaret) the goal of this experience is not to find a model, which is exactly true, but to find the best one which describes better the problem we are studying

In general, when building statistical models, we must not forget that the aim is to understand something about the real world. Or predict, choose an action, make a decision, summarize evidence, and so on, but always about the real world, not an abstract mathematical world: our models are not the reality—a point well made by George Box in his oft-cited remark that “ all models are wrong, but some are useful”.

**Usability of predictive models**

To make the selected predictive model interpretable and therefore useful for user I think the most important thing to do first is extracting the information necessary – such variable importance, variable effects, the relationship between variables and the model output …- to examine and understand the learned model.

In the context of machine learning interpretability is defined

as the ability to explain or to present in understandable terms to a human.  
… we argue that interpretability can assist in qualitatively ascertaining whether other desiderata—such as fairness, privacy, reliability, robustness, causality, usability, and trust—are met.

If you understand how your model produces predictions:

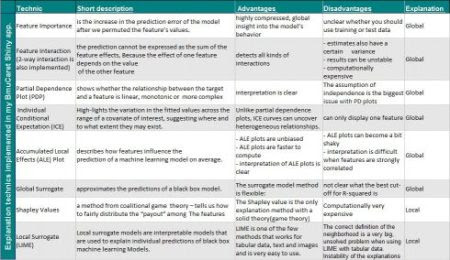
1. you will be better equipped to improve it
2. you can explain your model to user
3. your model will then assist user to accomplish tasks-e.g. produce customer-level models that describe the likelihood that a customer will take a particular action/develop an action plan to reduce customer churn

**New function in BMuCaret App.: post-hoc model explanation**

New functionalities have been implemented in my BMuCaret shiny App., which will assist you in an easy way (just some klicks) to conduct a post-hoch explanation of predictive models (candidate models: eXtreme Gradient Boosting, glmnet, LDA, Neural Network, Random Forest).

**Implemmented technics for conducting post hoch explanation**

The implemented tools/technics in BMuCaret for analyzing machine learning model on Dataset-Level (Global explanation) and on Prediction Level (local explanation) are listed in the table below.

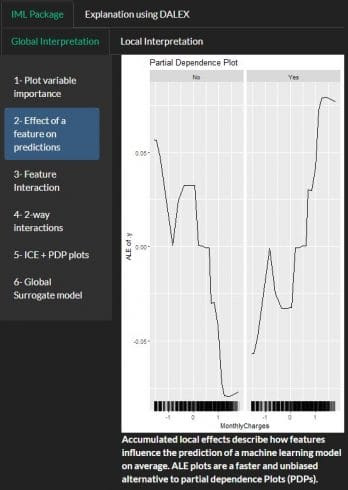


**Some Examples**

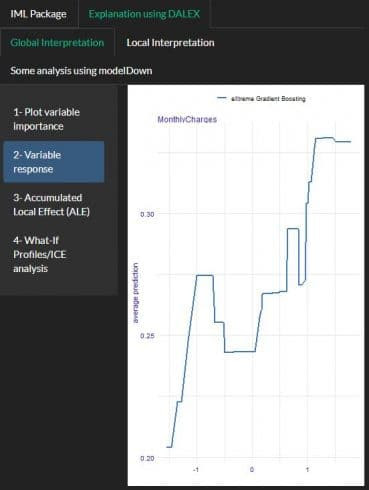
**PDP using IML and DALEX:**

A partial dependence plot can show whether the relationship between the target and a feature is linear, monotonic or more complex

**Using IML package**

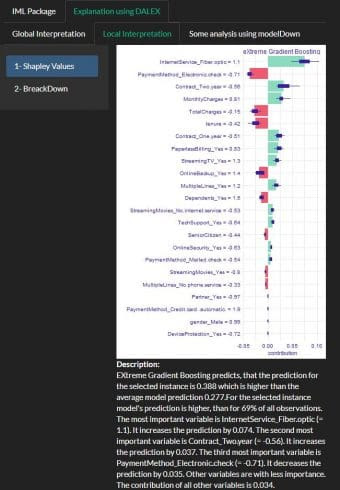


**Using DALEX package**



**Example Sharpley values using DALEX**

A prediction can be explained by assuming that each feature value of the instance is a “player” in a game where the prediction is the payout. Shapley values – a method from coalitional game theory – tells us how to fairly distribute the “payout” among the features.  
The SHAP values plot shows also the positive and negative relationships of the variables with the target variable (Churn).  
I did use the function describe{iBreakDown} to generate natural language explanations (Description at the bottom of the plot).

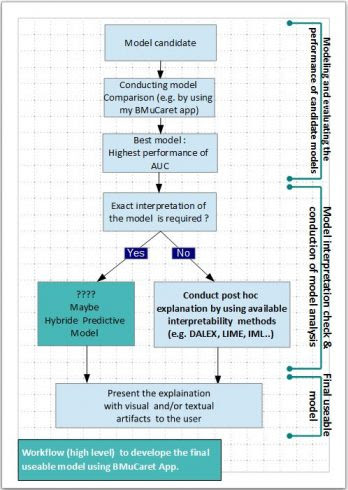


**BMuCaret Code**

The code for BMuCaret application consists of 4 Modules:

1. Module1: Load needed packages and the data set in scope
2. Module2: pre-Processing the data using recipe
3. Module3: Construct model grid and define shared settings
4. Module4: The code for the Shiny application. This module has been updated with post hoch explanation functionalities.

The following diagram is giving a high-level overview of the workflow for developing the final best and usable predictive Caret models.



**Conclusion and outlook**

With the updated BMuCaret Shiny application, you can not only perform modeling and find the best Caret model for your predictive problem but also generate explanations for the predictions of your trained black-box classifier (Caret models).  
All the implemented explanation technics are useful tools to provide insight into the internal decision-making process of the algorithm used, however, they do not exactly characterize the decision-making process and

they cannot have perfect fidelity with respect to the original model. If the explanation was completely faithful to what the original model computes, the explanation would equal the original model, and one would not need the original model in the first place, only the explanation. (In other words, this is a case where the original model would be interpretable.) This leads to the danger that any explanation  
method for a black-box model can be an inaccurate representation of the original model in parts of the feature space

In working in high stakes decisions field one should be very careful when applying post hoc methods with unknown descriptive accuracy.

Hybrid Predictive Model could be an alternative solution:

A hybrid predictive model is not a pure interpretable model. It utilizes the partial predictive power of black-box models to preserve the predictive accuracy. It is also not an explainer model which only observes but does not participate in the decision process. Here, a hybrid predictive model uses an interpretable model and a black-box model jointly to make a prediction .

This novel framework for learning a Hybrid Predictive Model is promising and it will be very likely the subject of my next blog post.

**And here is the code:**

#################################### Module1: Load needed packages and the data set in scope ########

library(shiny)

library(shinyBS)

library(shinythemes)

library(modelgrid)

library(magrittr)

library(caret)

library(e1071)

library(foreach)

library(recipes)

library(purrr)

library(dplyr)

library(pROC)

library(BradleyTerry2)

library(corrplot)

library(DataExplorer)

library(randomForest)

library(glmnet)

library(xgboost)

library(readr)

library(ggplot2)

library(mlbench)

library(e1071)

library(DALEX)

library(pdp)

library(ICEbox)

library(iml)

library(ingredients)

library(iBreakDown)

####################

# load data

set.seed(9650)

Telco\_customer\_3 <- read.csv("D:/Quellen/Telecom customer/WA\_Fn-UseC\_-Telco-Customer-Churn.csv")

data\_set <- Telco\_customer\_3

data\_set <- data\_set%>%dplyr::select(-"customerID")

data\_set$Churn <- as.factor(data\_set$Churn)

######################## Module2: pre-Processing the data using recipe

set.seed(9650)

Attri\_rec\_2 <-

recipe(Churn ~ ., data = data\_set) %>% # Formula.

step\_dummy(all\_predictors())%>% # convert nominal data into one or more numeric.

step\_knnimpute(all\_predictors())%>% # impute missing data using nearest neighbors.

step\_zv(all\_predictors()) %>% # remove variables that are highly sparse and unbalanced.

step\_corr(all\_predictors()) %>% # remove variables that have large absolute correlations with other variables.

step\_center(all\_predictors()) %>% # normalize numeric data to have a mean of zero.

step\_scale(all\_predictors()) %>% # normalize numeric data to have a standard deviation of one.

prep(training = data\_set, retain = TRUE) # train the data recipe

data\_rec\_2 <- as.data.frame(juice(Attri\_rec\_2))

# add the response variable

data\_rec\_2$Churn <- data\_set$Churn

# str(data\_rec\_2)

# Create a Validation Dataset (training data 70% and validation data 30% of the original data)

set.seed(9650)

validation\_index <- createDataPartition(data\_rec\_2$Churn,times= 1, p= 0.70, list=FALSE)

# select 30% of the data for validation

data\_validation <- data\_rec\_2[-validation\_index,]

# use the remaining 70% of data to train the models

data\_train <- data\_rec\_2[validation\_index, ]

# For traing the models

x\_train <- data\_train%>%select(-Churn) # Predictors

y\_train <- data\_train[["Churn"]] # Response

# for validation/test

x\_validation <- data\_validation%>%select(-Churn)

y\_validation <- data\_validation[["Churn"]]

## Vaiable list

variable\_list <- colnames(x\_validation)

############## Module 3 Construct model grid and define shared settings.####################

set.seed(9650)

mg <-

model\_grid() %>%

share\_settings(

y = y\_train,

x = x\_train,

metric = "ROC",

trControl = trainControl(

method = "adaptive\_cv",

number = 10, repeats = 5,

adaptive = list(min =3, alpha =0.05, method = "BT", complete = FALSE),

search = "random",

summaryFunction = twoClassSummary,

classProbs = TRUE))

purrr::map\_chr(mg$shared\_settings, class)

## add models to train

mg\_final <- mg %>%

add\_model(model\_name = "LDA",

method = "lda",

custom\_control = list(method = "repeatedcv",

number = 10,

repeats =5))%>%

add\_model(model\_name = "eXtreme Gradient Boosting",method = "xgbDART")%>%

add\_model(model\_name = "Neural Network", method = "nnet")%>%

add\_model(model\_name = "glmnet", method = "glmnet")%>%

add\_model(model\_name = "Random Forest", method = "rf")

#%>% add\_model(model\_name = "gbm",

# method="gbm", custom\_control = list(verbose = FALSE))

list\_model <- c(names(mg\_final$models))

################################# Module4: Shiny application #########################

ui <- fluidPage(theme = shinytheme("darkly"),

## Application title

titlePanel(wellPanel("Model Explanation with BMuCaret Shiny Application",br(),"BMuCaret2")),

navbarPage("Workflow ===>",

tabPanel("Exploratory Data Analysis",

tabsetPanel(type = "tabs",

tabPanel("Explore the data object",

navlistPanel(

tabPanel("1- Structure of the data object", verbatimTextOutput("str\_ucture")),

tabPanel("2- Missing values ?", plotOutput("missing\_value")),

tabPanel("3- Correlation analysis", plotOutput("coor\_plot", height = 700)))),

tabPanel("After Data Cleaning (using recipe method)",

navlistPanel(

tabPanel("1- Structure of the data object after Processing",

verbatimTextOutput("str\_ucture\_after")),

tabPanel("2- Missing values ?", plotOutput("missing\_value\_after") ),

tabPanel("3- Correlation analysis after", plotOutput("coor\_plot\_after", height = 600)) )))),

tabPanel("Fitting & Validation & Statistics",

sidebarLayout(

sidebarPanel(

wellPanel(selectInput(inputId = "abd",

label = "Model choose : ",

choices = c("none", list\_model),

selected = "none", width = '200px'))),

mainPanel(

tabsetPanel(type= "tabs",

## output model training and summary

tabPanel("Model Training & Summary",

navlistPanel(

tabPanel("1- Show info model grid ",verbatimTextOutput("info\_modelgrid")),

tabPanel("2- Performance statistics of the model grid (dotplot) ",

plotOutput("dotplot", width = 600, height = 600)),

tabPanel("3- Extract Performance of the model grid ", verbatimTextOutput(outputId = "summary")),

tabPanel("4- Show the AUC & Accuracy of individual models (on data training)",

verbatimTextOutput("Accurac\_AUC"), htmlOutput("best\_model\_train")),

tabPanel("5- Show Statistics of individual model", verbatimTextOutput(outputId = "Indiv\_Analysis")

,"Examine the relationship between the estimates of performance

and the tuning parameters", br(),

plotOutput(outputId= "tuning\_parameter")),

tabPanel("6- Show single model's Accuracy (on data training)",

verbatimTextOutput(outputId = "accuracy")))),

## output model validation and statistics

tabPanel("Model Validation & Statistics",

navlistPanel(

tabPanel("1-Show the AUC & Accuracy of individual models (on validation data)",

verbatimTextOutput(outputId = "AUC\_of\_individ"), htmlOutput("best\_model\_vali")),

tabPanel("2- Show single model's Accuracy/data validation",

verbatimTextOutput("accuracy\_vali"))))

)))),

########################################################### Module post hoch explanation ################

tabPanel("Model Explanation",

sidebarLayout(

sidebarPanel(

wellPanel(selectInput(inputId = "perf",

label = "Model choose : ",

choices = c("none", list\_model),

selected = "none", width = '200px')),

wellPanel(selectInput(inputId = "Variab\_le",

label = "Variable choose : ",

choices = c("none", variable\_list),

selected = "none", width = '200px')),

wellPanel(numericInput(inputId="row\_index",

label= "Insert the row number",

value=1,

min=1, max=20, width = '200px'))),

mainPanel(

tabsetPanel(type = "tabs",

#### IML Packge#######

tabPanel("IML Package",

tabsetPanel(type = "tabs",

tabPanel("Global Interpretation",

navlistPanel(

tabPanel("1- Plot variable importance", plotOutput("variable\_imp", height = "600px"),

verbatimTextOutput("condition")),

tabPanel("2- Effect of a feature on predictions",

plotOutput("PD\_P", height = "600px"), htmlOutput("des")),

tabPanel("3- Feature Interaction", plotOutput("inter\_action", height = "600px")),

tabPanel("4- 2-way interactions", plotOutput("two\_way\_inter", height = "600px"),

verbatimTextOutput("condition2")),

tabPanel("5- ICE + PDP plots", plotOutput("IC\_E")),

tabPanel("6- Global Surrogate model",

plotOutput("surrogate\_model", height = "600px")))),

tabPanel("Local Interpretation",

navlistPanel(

tabPanel("1- Shapley Values", plotOutput(outputId = "shapley\_Values", height = "600px")),

tabPanel("2- LIME", plotOutput(outputId = "single\_plot\_lime", height = "600px")))))),

#### DALEX Packge#######

tabPanel("DALEX Package",

tabsetPanel(type = "tabs",

tabPanel("Global Interpretation ",

navlistPanel(

tabPanel("1- Plot variable importance",

plotOutput("variable\_imp\_da", height = "600px"),

verbatimTextOutput("condition\_da")),

tabPanel("2- Variable response", plotOutput("Vari\_resp", height = "600px")),

tabPanel("3- Accumulated Local Effect (ALE)", plotOutput("ALE\_da"),

verbatimTextOutput("condition2\_da")),

tabPanel("4- What-If Profiles/ICE analysis ",

plotOutput("what\_if", height = "600px")))),

tabPanel("Local Interpretation ",

navlistPanel(

tabPanel("1- Shapley Values", plotOutput(outputId = "shapley\_Values\_da", height = "600px"),

htmlOutput("descri\_ption\_sharp")),

tabPanel("2- BreackDown", plotOutput(outputId = "single\_plot\_brd\_da",

height = "600px"),htmlOutput("descri\_ption")))),

tabPanel("Some analysis using modelDown ", br(), br(),

actionButton("do", "Click here to generate a website with HTML

to get summary: auditor, drifter, model\_performance,

variable\_importance, and variable\_response", width = '90px' ))))

))))),

############ Contact #############

br(), br(),br(),

h5("BMuCaret App. built with", img(src =

"<https://d33wubrfki0l68.cloudfront.net/8f68bc526580587d60714f9279fbe9d86b25362a/79958/assets/img/shiny.png>",

height= "30px"),

"by",

br(),

"Author: Abderrahim Lyoubi-Idrissi",

br())

)

################################# Define server logic #########################

server <- function(input, output, session) {

set.seed(9650)

# before processing the data

# data structure

output$str\_ucture <- renderPrint({

str(data\_set)

})

## Missing values ?

output$missing\_value <- renderPlot({

plot\_missing(data\_set)

})

# plot to check the results of step\_corr(all\_predictors())

output$coor\_plot <- renderPlot({

plot\_correlation(data\_set)

})

## after processing the data

# data structure

output$str\_ucture\_after <- renderPrint({

str(data\_rec\_2)

})

# plot to check the missing values

output$missing\_value\_after <- renderPlot({

plot\_missing(data\_rec\_2)

})

# plot to check the correlation

output$coor\_plot\_after <- renderPlot({

plot\_correlation(data\_rec\_2)

})

set.seed(9650)

## Infos about the componets of the constructed model grid

output$info\_modelgrid <- renderPrint({

mg\_final$models

})

## train all models

set.seed(9650)

mg\_final\_tra <- reactive({train(mg\_final)})

## plot to compare

output$dotplot <- renderPlot({

mg\_final\_tra()$model\_fits %>%

caret::resamples(.) %>%

lattice::dotplot(.)

})

## Show the overall summary

set.seed(9650)

output$summary <- renderPrint({

mg\_final\_tra()$model\_fits %>%

caret::resamples(.) %>%

summary(.)

})

## computing the auc & the accuracy (on train data)

set.seed(9650)

output$Accurac\_AUC <- renderPrint({

AUC1 <- mg\_final\_tra()$model\_fits%>% predict(., newdata = x\_train, type ="prob")%>%

map(~roc(y\_train, .x[,2]))

auc\_value\_train <- map\_dbl(AUC1, ~(.x)$auc)

auc\_value\_df\_train\_df <- as.data.frame(auc\_value\_train)

accuarcy\_all\_train <- predict(mg\_final\_tra()$model\_fits, newdata = x\_train)%>%

map( ~confusionMatrix(.x, y\_train))%>%

map\_dfr(~ cbind(as.data.frame(t(.x$overall)),as.data.frame(t(.x$byClass))), .id = "Modelname")%>%

select(Modelname, Accuracy, Kappa)

accuracy\_auc\_train <- bind\_cols(accuarcy\_all\_train, auc\_value\_df\_train\_df)

accuracy\_auc\_train

})

## computing the auc and the Accuracy

set.seed(9650)

output$best\_model\_train <- renderUI({

AUC1 <- mg\_final\_tra()$model\_fits%>% predict(., newdata = x\_train, type ="prob")%>%

map(~roc(y\_train, .x[,2]))

auc\_value\_train <- map\_dbl(AUC1, ~(.x)$auc)

auc\_value\_df\_train\_df <- as.data.frame(auc\_value\_train)

accuarcy\_all\_train <- predict(mg\_final\_tra()$model\_fits, newdata = x\_train)%>%

map( ~confusionMatrix(.x, y\_train))%>%

map\_dfr(~ cbind(as.data.frame(t(.x$overall)),as.data.frame(t(.x$byClass))), .id = "Modelname")%>%

select(Modelname, Accuracy, Kappa)

accuracy\_auc\_train <- bind\_cols(accuarcy\_all\_train, auc\_value\_df\_train\_df)

max\_auc\_train <- filter(accuracy\_auc\_train, auc\_value\_train == max(accuracy\_auc\_train$auc\_value\_train))%>%

select(Modelname)

max\_Accuracy\_train <- filter(accuracy\_auc\_train, Accuracy == max(accuracy\_auc\_train$Accuracy))%>%

select(Modelname)

HTML(paste("Results", br(),

"1- The model", strong(max\_auc\_train), "has the highest AUC value.", br(),

"2- The model",strong(max\_Accuracy\_train), "has the highest Accuracy" ))

})

# Show the summary of individual model

output$Indiv\_Analysis <- renderPrint({if(input$abd == "none"){print("Please select a model")}

mg\_final\_tra()$model\_fits[[input$abd]]

})

# Plot the individual model

output$tuning\_parameter <- renderPlot({

ggplot( mg\_final\_tra()$model\_fits[[input$abd]])

})

# ConfusionMatrix on training data

output$accuracy <- renderPrint({if(input$abd == "none"){print("Please select a model")}

else{ confusionMatrix( predict(mg\_final\_tra()$model\_fits[[input$abd]], x\_train), y\_train)}

})

########################### Model Validation #########################

# Extract the auc values

output$AUC\_of\_individ <- renderPrint({

AUC2 <- mg\_final\_tra()$model\_fits%>% predict(., newdata = x\_validation, type ="prob")%>%

map(~roc(y\_validation, .x[,2]))

auc\_value\_vali <- map\_dbl(AUC2, ~(.x)$auc)

auc\_value\_df\_vali\_df <- as.data.frame(auc\_value\_vali)

cf\_vali <- predict(mg\_final\_tra()$model\_fits, newdata = x\_validation)%>%

map( ~confusionMatrix(.x, y\_validation))

cf\_vali%>%

map\_dfr(~ cbind(as.data.frame(t(.x$overall)),as.data.frame(t(.x$byClass))), .id = "Modelname")%>%

select(Modelname, Accuracy, Kappa)

accuarcy\_all\_vali <- predict(mg\_final\_tra()$model\_fits, newdata = x\_validation)%>%

map( ~confusionMatrix(.x, y\_validation))%>%

map\_dfr(~ cbind(as.data.frame(t(.x$overall)),as.data.frame(t(.x$byClass))), .id = "Modelname")%>%

select(Modelname, Accuracy, Kappa)

accuracy\_auc\_vali <- bind\_cols(accuarcy\_all\_vali,auc\_value\_df\_vali\_df)

print(accuracy\_auc\_vali)

})

output$best\_model\_vali <- renderUI({

AUC2 <- mg\_final\_tra()$model\_fits%>%

predict(., newdata = x\_validation, type ="prob")%>%

map(~roc(y\_validation, .x[,2]))

auc\_value\_vali <- map\_dbl(AUC2, ~(.x)$auc)

auc\_value\_df\_vali\_df <- as.data.frame(auc\_value\_vali)

cf\_vali <- predict(mg\_final\_tra()$model\_fits, newdata = x\_validation)%>%

map( ~confusionMatrix(.x, y\_validation))

cf\_vali%>%

map\_dfr(~ cbind(as.data.frame(t(.x$overall)),as.data.frame(t(.x$byClass))), .id = "Modelname")%>%

select(Modelname, Accuracy, Kappa)

accuarcy\_all\_vali <- predict(mg\_final\_tra()$model\_fits, newdata = x\_validation)%>%

map( ~confusionMatrix(.x, y\_validation))%>%

map\_dfr(~ cbind(as.data.frame(t(.x$overall)),as.data.frame(t(.x$byClass))), .id = "Modelname")%>%

select(Modelname, Accuracy, Kappa)

accuracy\_auc\_vali <- bind\_cols(accuarcy\_all\_vali,auc\_value\_df\_vali\_df)

max\_auc\_vali <- filter(accuracy\_auc\_vali, auc\_value\_vali == max(accuracy\_auc\_vali$auc\_value\_vali))%>%

select(Modelname)

max\_Accuracy\_vali <- filter(accuracy\_auc\_vali, Accuracy == max(accuracy\_auc\_vali$Accuracy))%>%

select(Modelname)

HTML(paste("Results", br(),

"1- The model", strong(max\_auc\_vali), "has the highest AUC value.", br(),

"2- The model",strong(max\_Accuracy\_vali), "has the highest Accuracy."))

})

## confusion matrix on data validation

output$accuracy\_vali<- renderPrint({if(input$abd == "none"){print("Please select a model")}

else{

confusionMatrix(predict(mg\_final\_tra()$model\_fits[[input$abd]], x\_validation), y\_validation)}

})

############################## Model explaination #######################################

############## Using IML Package ###################################

## define the IML predictor

mod2 <- reactive({ Predictor$new(mg\_final\_tra()$model\_fits[[input$perf]],

data = x\_validation, y= y\_validation, type = "prob")})

## plot the variable importance

output$condition <- renderPrint({

if(input$perf == "none"){print("Please select a model")}

else{

output$variable\_imp <- renderPlot({

vari\_imp <- FeatureImp$new(mod2(), loss = "ce")

plot(vari\_imp)})

}})

## Visualise the features interaction

output$inter\_action <- renderPlot({

plot(Interaction$new(mod2()))

})

output$condition2 <- renderPrint({

if(input$Variab\_le == "none"){print("Please select a variable")}

else{

output$two\_way\_inter <- renderPlot({

plot(Interaction$new(mod2(), feature = input$Variab\_le)) })}

})

## Feature effects

output$PD\_P <- renderPlot({

ale <- FeatureEffect$new(mod2(), feature = input$Variab\_le)

plot(ale) +

ggtitle("Partial Dependence Plot")

})

output$des <- renderUI({

br()

HTML(paste(

strong("Accumulated local effects describe how features influence the prediction

of a machine learning model on average. ALE plots

are a faster and unbiased alternative to partial dependence Plots (PDPs).")))

})

## Plot PDP and ICE

output$IC\_E <- renderPlot({

plot(FeatureEffect$new(mod2(), feature = input$Variab\_le, method = "pdp+ice"))

})

## Surrogate model

output$surrogate\_model <- renderPlot({

tree <- TreeSurrogate$new(mod2(), maxdepth = 2)

plot(tree)

})

## Shapley Values

# Explain single predictions with game theory

output$shapley\_Values <- renderPlot({

shapley <- reactive ({Shapley$new(mod2(), x.interest = x\_validation[input$row\_index, ])})

plot(shapley())

})

## LIME

# Explain single prediction with LIME

output$single\_plot\_lime <- renderPlot({

lime.explain = LocalModel$new(mod2(),x.interest = x\_validation[input$row\_index,])

lime.explain$results %>%

ggplot(aes(x = reorder(feature.value, -effect), y = effect, fill = .class)) +

facet\_wrap(~ .class, ncol = 1) +

geom\_bar(stat = "identity", alpha = 1) +

scale\_fill\_discrete() +

coord\_flip() +

labs(title = paste0("Instance row number: ", input$row\_index)) +

guides(fill = FALSE)

})

############## using DALEX Package ###################################

## definde DALEX explainer

y\_validation\_da <- factor(ifelse(y\_validation == "Yes", 1, 0))

yTest = as.numeric(as.character(y\_validation\_da))

p\_fun <- function(object, newdata){predict(object, newdata= newdata, type="prob")[,2]}

explainer\_da <- reactive({ explain(mg\_final\_tra()$model\_fits[[input$perf]], label = input$perf,

data = x\_validation, y= yTest,

predict\_function = p\_fun)

})

## plot variable importance with DALEX

output$condition\_da <- renderPrint({

if(input$perf == "none"){print("Please select a model")}

else{

output$variable\_imp\_da <- renderPlot({

vari\_imp\_da <- feature\_importance(explainer\_da(), loss\_function = loss\_root\_mean\_square)

plot(vari\_imp\_da)

})

}})

## PDP plot

output$Vari\_resp <- renderPlot({

Vari\_response <- ingredients::partial\_dependency( explainer\_da(), variables = input$Variab\_le)

plot(Vari\_response)

})

## ALE plot

output$ALE\_da <- renderPlot({

accumulated\_da <- accumulated\_dependency( explainer\_da(), variables = input$Variab\_le)

plot( accumulated\_da)

})

## What if Profiles/ICE

output$what\_if <- renderPlot({

cp\_model <- ceteris\_paribus(explainer\_da(), x\_validation[input$row\_index, ])

plot(cp\_model, variables = input$Variab\_le) +

show\_observations(cp\_model, variables = input$Variab\_le) +

ggtitle("Ceteris Paribus Profiles of the selected model")

})

## BreackDown Plot

output$single\_plot\_brd\_da <- renderPlot({

brd\_da <- break\_down(explainer\_da(),

x\_validation[input$row\_index, ], keep\_distributions = TRUE)

plot(brd\_da)

})

## Description

output$descri\_ption <- renderUI({

brd\_da <- break\_down(explainer\_da(), x\_validation[input$row\_index, ],

keep\_distributions = TRUE)

br()

HTML(paste(strong("Description:"),

br(),

describe(brd\_da,

short\_description = FALSE,

display\_values = TRUE,

display\_numbers = TRUE,

display\_distribution\_details = FALSE)))

})

## Shapley Values

# Explain single predictions with game theory

output$shapley\_Values\_da <- renderPlot({

shap\_values <- shap(explainer\_da(), x\_validation[input$row\_index, ], B=25)

plot(shap\_values)

})

output$descri\_ption\_sharp <- renderUI({

shap\_values <- shap(explainer\_da(), x\_validation[input$row\_index, ], B=25)

br()

HTML(paste(

strong("Description:"),

br(),

describe(shap\_values,

short\_description = FALSE,

display\_values = TRUE,

display\_numbers = TRUE,

display\_distribution\_details = FALSE)))

})

## Generates a website with HTML summaries for the selected predictive model

observeEvent(input$do,

{modelDown::modelDown( explainer\_da())})

output$down\_mod <- renderUI({HTML(paste("to summary: auditor, drifter, model\_performance,

variable\_importance, and variable\_response"))})

################################# Run the application #########################

}

shinyApp(ui = ui, server = server)