**Recap**

This is a continuation on the explanation of machine learning model predictions. Specifically, random forest models. Today, we will explore external packages which aid in explaining random forest predictions.

**External packages**

There are external a few packages which offer to calculate variable importance for random forest models apart from the conventional measurements found within the random forest package.

**rfVarImpOOB**

The first package is rfVarImpOOB which was recently released on CRAN. The vignette is unfortunately too brief in helping me understand the functions and the mechanism of this novel scoring of feature importance.

**VSURF**

The next package is VSURF

**randomForestExplainer**

The above journal article which used VSURF also used another package to select important features from a random forest. This other package is the randomForestExplainer. We will elaborate and demonstrate this package for the rest of the post.

**Dataset**

Similar to the previous posts, the [Cleveland heart dataset will be used as well as principles of tidymodels](https://notast.netlify.com/post/explaining-predictions-random-forest-post-hoc-analysis-randomforestexplainer-package/).

#library

library(tidyverse)

library(tidymodels)

#import

heart<-read\_csv("<https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data>", col\_names = F)

# Renaming var

colnames(heart)<- c("age", "sex", "rest\_cp", "rest\_bp",

"chol", "fast\_bloodsugar","rest\_ecg","ex\_maxHR","ex\_cp",

"ex\_STdepression\_dur", "ex\_STpeak","coloured\_vessels", "thalassemia","heart\_disease")

#elaborating cat var

##simple ifelse conversion

heart<-heart %>% mutate(sex= ifelse(sex=="1", "male", "female"),fast\_bloodsugar= ifelse(fast\_bloodsugar=="1", ">120", "<120"), ex\_cp=ifelse(ex\_cp=="1", "yes", "no"),

heart\_disease=ifelse(heart\_disease=="0", "no", "yes"))

## complex ifelse conversion using `case\_when`

heart<-heart %>% mutate(

rest\_cp=case\_when(rest\_cp== "1" ~ "typical",rest\_cp=="2" ~ "atypical", rest\_cp== "3" ~ "non-CP pain",rest\_cp== "4" ~ "asymptomatic"), rest\_ecg=case\_when(rest\_ecg=="0" ~ "normal",rest\_ecg=="1" ~ "ST-T abnorm",rest\_ecg=="2" ~ "LV hyperthrophy"), ex\_STpeak=case\_when(ex\_STpeak=="1" ~ "up/norm", ex\_STpeak== "2" ~ "flat",ex\_STpeak== "3" ~ "down"), thalassemia=case\_when(thalassemia=="3.0" ~ "norm",

thalassemia== "6.0" ~ "fixed", thalassemia== "7.0" ~ "reversable"))

# convert missing value "?" into NA

heart<-heart%>% mutate\_if(is.character, funs(replace(., .=="?", NA)))

# convert char into factors

heart<-heart %>% mutate\_if(is.character, as.factor)

#train/test set

set.seed(4595)

data\_split <- initial\_split(heart, prop=0.75, strata = "heart\_disease")

heart\_train <- training(data\_split)

heart\_test <- testing(data\_split)

Though random forest model itself doesn’t need explicit one hot encoding, some of randomForestExplainer functions need dummy variables. Thus we will create them in our recipe.

# create recipe object

heart\_recipe<-recipe(heart\_disease ~., data= heart\_train) %>%

step\_knnimpute(all\_predictors()) %>%

step\_dummy(all\_nominal(), -heart\_disease)

# process the traing set/ prepare recipe(non-cv)

heart\_prep <-heart\_recipe %>% prep(training = heart\_train, retain = TRUE)

set.seed(69)

rf\_model<-rand\_forest(trees = 2000, mtry = 4, mode = "classification") %>% set\_engine("randomForest",

importance=T, localImp = T, ) %>% fit(heart\_disease ~ ., data = juice(heart\_prep))

**randomForestExplainer (variable importance)**

**Measuring variable importance**

The randomForestExplainer generates more feature importance score than the randomForest package. These scores generated by the randomForestExplainer::measure\_importance function can be categorized into 3 types of feature importance score.

1. Measures of importance based on the structure of the forest which is not examine in the randomForest package.

1i) Mean\_minimal\_depth: Minimal depth for a variable in a tree equals to the depth of the node which splits on that variable and is the closest to the root of the tree. If it is low then a lot of observations are divided into groups on the basis of this variable.

1ii) no\_of\_nodes: Usually, number of trees and number of nodes are the same if trees are shallow. The number of nodes contain similar information as the number of trees and p-values. Thus, no\_of\_tress and p\_value are omitted in plot\_importance\_ggpairs graph (which will be described later).

1iii) no\_of\_trees

1iv) p\_value: Number of nodes in which predictor X was used for splitting exceeds the theoretical number of successes if they were random, following the binomial distribution given.

1v) times\_a\_root

1. Measures of importance based on the decrease in predictive accuracy post variable perturbation

2i) accuracy\_decrease: Use for classification problems. Same value as randomForest::importance(rf\_model$fit, type=2)

2ii) mse\_increase: Use for regression problems.

1. Measure of importance based on loss function

3i) gini\_decrease Use for classifcation cases. Same value as randomForest::importance(rf\_model$fit, type=1).

3ii) node\_purity\_increase: Use for regression cases. Measured by the decrease in sum of squares

library(randomForestExplainer)

impt\_frame<-measure\_importance(rf\_model$fit)

impt\_frame %>% head()

## variable mean\_min\_depth no\_of\_nodes accuracy\_decrease

## 1 age 3.076865 8635 0.004269917

## 2 chol 3.377786 8872 -0.003120879

## 3 coloured\_vessels\_X1.0 4.969157 3202 0.009133193

## 4 coloured\_vessels\_X2.0 4.155141 2600 0.010876652

## 5 coloured\_vessels\_X3.0 5.476794 1756 0.006136837

## 6 ex\_cp\_yes 3.791160 3033 0.009524295

## gini\_decrease no\_of\_trees times\_a\_root p\_value

## 1 8.686010 1979 54 0

## 2 7.997923 1983 3 0

## 3 2.814661 1749 13 1

## 4 3.769974 1675 82 1

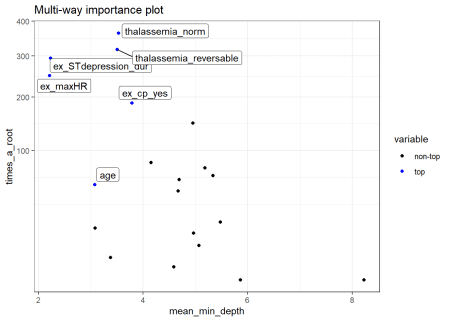
## 5 2.044707 1280 20 1

## 6 5.557652 1766 187 1

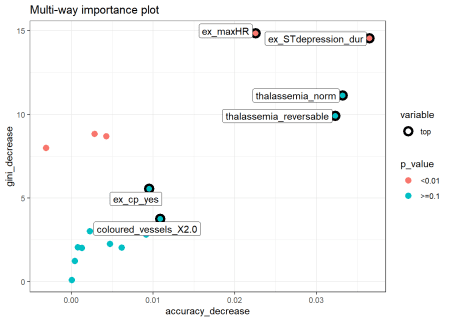
**Ploting**

plot\_multi\_way\_importance plots variable importance scores from importance\_frame. By default, the function plots scores measuring importance based on the structure of the forest, mean\_min\_depth against times\_a\_root.

plot\_multi\_way\_importance(impt\_frame, no\_of\_labels = 6)

  
plot\_multi\_way\_importance can plot up to 3 feature importance scores using any of the three kinds of importance measurements.

plot\_multi\_way\_importance(impt\_frame, x\_measure = "accuracy\_decrease", y\_measure = "gini\_decrease", size\_measure = "p\_value", no\_of\_labels = 6)



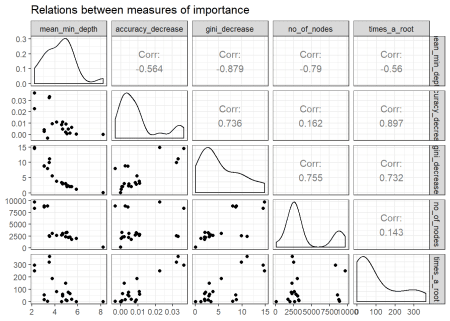
**Comments**

* Comparing the top 6 variables from both plots, only 4 variables thalaseemia\_norm, thalassemia\_reversable, ex\_maxHR and ex\_STdepression\_dur appear in both plots. These variables are more likely to be essential in the random forest’s prediction.
* thalaseemia\_norm and thalassemia\_reversable are identified as one of the six variables with good gini\_decrease and accuracy\_decrease but they are insignificant as p\_value >=.1.

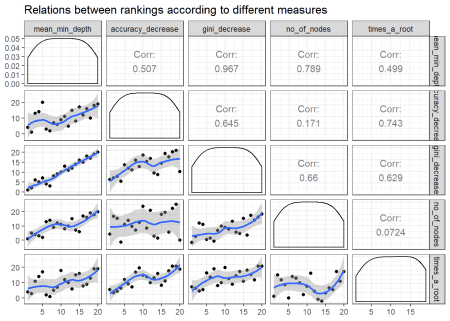
**Selecting optimal importance scores to plot**

There are many multi-way importance plots which one can create thus we need to identify which sub-set of importance scores will provide us with more helpful plots. Examining the relationship between the importance measurements with plot\_importance\_ggpairs can assist us in identifying a sub-set of scores to use. We can pick three scores that least agree with each other, points in plots which are most dispersed.

plot\_importance\_ggpairs(impt\_frame)

  
Plotting the rankings of importance measures with a LOESS curve using plot\_importance\_rankings informs us of *pairs of measures almost exactly agreeing in their rankings of variables*. *This approach might be useful as rankings are more evenly spread than corresponding importance measures. This may also more clearly show where the different measures of importance disagree or agree*.

plot\_importance\_rankings(impt\_frame)

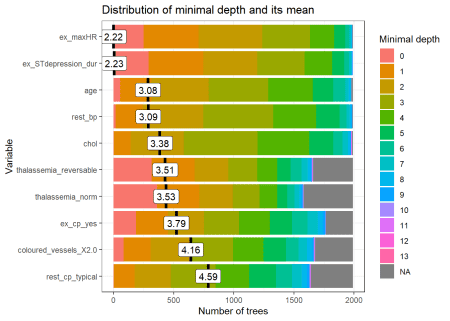
  
mean\_min\_depth and gini\_decrease agree the most thus we should avoid this combination for plot\_multi\_way\_importance.

**randomForestExplainer: variable depth**

Previously, we looked at various types of importance measures. Now we are specifically examining the mean minimal depth in detail. The distribution of the mean minimal depth allows us to appreciate the variable’s role in the random forest’s structure and prediction.  
plot\_min\_depth\_distribution plots the top ten variables according to mean minimal depth calculated using top trees.  
The mean minimal depth can be calculated in 3 different ways in plot\_min\_depth\_distribution using the mean\_sample argument. The calculation differs in the way they treat missing values that appear when a feature is not used for tree splitting. As a result, the ranking of variables may change for each calculation.

md\_frame <- min\_depth\_distribution(rf\_model$fit)

plot\_min\_depth\_distribution(md\_frame, mean\_sample = "top\_trees") # default mean\_sample arg

  
The mean minimal depth is indicated by a vertical bar with the mean value beside it. The smaller the mean minimal depth, the more important the variable is and the higher up the y-axis the variable will be.  
The rainbow gradient reveals the min and max minimal depth for each variable. The bigger the proportion of minimal depth zero (red blocks), the more frequent the variable is the root of a tree. The smaller the proportion of NA minimal depth (gray blocks), the more frequent the variable is used for splitting trees.  
The range of the x-axis is from zero to the maximum number of trees for the feature.

**randomForestExplainer : variable interaction**

We will use the important\_variables function to select the top 6 variables based on the following variable importance measurements, times\_a\_root and no\_of\_nodes.

vars<- important\_variables(impt\_frame, k = 6, measures = c("times\_a\_root", "no\_of\_nodes"))

After identifying the top 6 variables, we can examine the interactions between the variables with the min\_depth\_interactions function.

interactions\_frame <- min\_depth\_interactions(rf\_model$fit, vars)

The interaction is reflected as the mean\_min\_depth which is the mean conditional minimal depth, where a variable is taken as a root node/root\_variable and the mean minimal depth is calculated for the other variable.  
The uncond\_mean\_min\_depth represents the unconditional mean minimal depth of the variable which is the mean minimal depth of the variable without having a stipulated root variable. This value is the same as the mean value seen on the vertical bar in plot\_min\_depth\_distribution.

head(interactions\_frame)

## variable root\_variable mean\_min\_depth occurrences

## 1 age age 2.766519 838

## 2 age ex\_cp\_yes 2.678235 940

## 3 age ex\_maxHR 2.068416 1251

## 4 age ex\_STdepression\_dur 2.142738 1215

## 5 age thalassemia\_norm 2.719905 1036

## 6 age thalassemia\_reversable 2.539302 1064

## interaction uncond\_mean\_min\_depth

## 1 age:age 3.076865

## 2 ex\_cp\_yes:age 3.076865

## 3 ex\_maxHR:age 3.076865

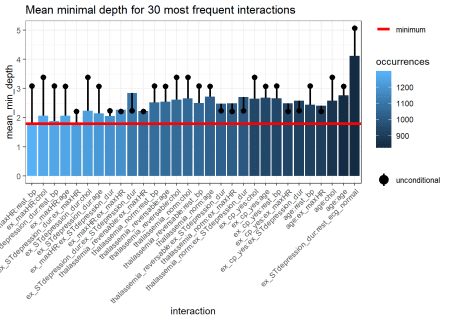
## 4 ex\_STdepression\_dur:age 3.076865

## 5 thalassemia\_norm:age 3.076865

## 6 thalassemia\_reversable:age 3.076865

We can plot the outputs from min\_depth\_interactions with plot\_min\_depth\_interactions.

plot\_min\_depth\_interactions(interactions\_frame)

  
The interactions are arranged from the most frequent occurring on the left side of the plot and in lighter blue to the least frequent occurring interaction on the right side of the plot and in darker blue.  
The horizontal red line represents the minimum mean\_min\_depth.  
The black lollipop represents the uncond\_mean\_min\_depth.  
Similar to plot\_min\_depth\_distribution, the ranking of interactions in plot\_min\_depth\_interactions may change depending on the arguments for mean\_sample and uncond\_mean\_sample.

**Interactive variables and forest prediction**

We can further evaluate the variable interactions by plotting the probability of a prediction against the variables making up the interaction. For instance we plot the probability of having heart disease against resting blood pressure rest\_bp and ST depression duration during exercise test ex\_STdepression\_dur. The interaction of these two variables are the most frequent interaction as seen in plot\_min\_depth\_interactions. We plot the forest prediction against interactive variables with plot\_predict\_interaction.

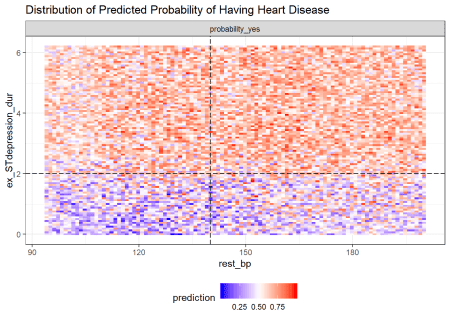
#plot\_predict\_interaction(rf\_model$fit, bake(heart\_prep, new\_data = heart\_train), "rest\_bp", "ex\_STdepression\_dur")

However, there is an error when the input supplied is a model created with parsnip. There is no error when the model is created directly from the randomForest package.

set.seed(69)

forest <- randomForest::randomForest(heart\_disease ~ ., data = bake(heart\_prep, new\_data = heart\_train), localImp = TRUE, importance=T, ntree = 2000, mtry = 4, type= "classification")

plot\_predict\_interaction(forest, bake(heart\_prep, new\_data = heart\_train), "rest\_bp", "ex\_STdepression\_dur", main = "Distribution of Predicted Probability of Having Heart Disease") + theme(legend.position="bottom") + geom\_hline(yintercept = 2, linetype="longdash") + geom\_vline(xintercept = 140, linetype="longdash")

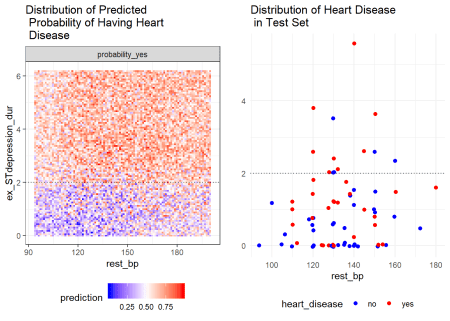
  
ST depression duration during exercise test ex\_STdepression\_dur longer than 2s results in higher predicted probability of having a heart disease. The predicted probability of having heart disease for individuals with shorter ST depression duration (<2s) increases if they have high resting blood pressure rest\_bp above 140.

As plot\_predict\_interaction is a ggplot object which you can treat it like any other ggplot. In this case, we can place it side by side with the ggplot of the distribution of heart disease in the test set.

predict\_plot<-plot\_predict\_interaction(forest, bake(heart\_prep, new\_data = heart\_train), "rest\_bp", "ex\_STdepression\_dur", main = "Distribution of Predicted\n Probability of Having Heart\n Disease") + theme(legend.position="bottom") + geom\_hline(yintercept = 2, linetype="dotted")

test\_plot<-ggplot(heart\_test, aes(rest\_bp, ex\_STdepression\_dur, colour=heart\_disease)) +geom\_jitter(size=1.8) + labs(title="Distribution of Heart Disease\n in Test Set", y="") + theme\_minimal() + scale\_color\_manual(values=c("blue", "red")) + theme(legend.position="bottom") + geom\_hline(yintercept = 2, linetype="dotted")

gridExtra::grid.arrange(predict\_plot, test\_plot, ncol=2)



**Conclusion**

In this post, we learned how random forest predictions can be explained based on various variable importance measures, variable interactions and variable depth. The post-hoc analysis was aided with the randomForestExplainer package.