Practical Machine Learning - Course Project

Luis Ballesteros

2021/08/08

Summary

From the activity data of 6 people collected in the following link http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset) we are going to create a model that allows us to discern what type of activity they are doing.

Se ha seleccionado el modelo “Random Forest” para realizar el análisis.

Data import and cleaning

We use the read.csv function to import the training and test data sets.

We set the seed value for reproducibility to 2701.

```r
# Establecemos los valores de Nan
data <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                     na.strings=c("NA","#DIV/0!",""))
dim_data <- dim(data)
# convert "classe" to factor
data$classe <- factor(data$classe)
misc_cols <- c(1,3:7)
data <- data %>%
    select(-all_of(misc_cols))

# str(data)

# remove variables with Nearly Zero Variance
variable_near_zero_variance <- nearZeroVar(data)
data <- data[, -variable_near_zero_variance]
#str(data)
#summary(data)
dim_data_nzv <- dim(data)

na_95 <- sapply(data, function(x) mean(is.na(x))) > 0.95
data <- data[, na_95==FALSE]

dim_data_na_95 <- dim(data)
# import test case
test_cases <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
                     na.strings=c("NA","#DIV/0!",""))
dim_test_cases <- dim(test_cases)
```

Data are composed of 19622 records and 160 variables.

We eliminate variables 1, 3, 4, 5, 6, 7 which are only descriptor variables. If we kept these variables the final fit by Random Forest would be perfect in both training and test data. We also eliminate variables that have a variance close to zero. The variables are reduced from 160 to 119.

We eliminated variables with more than 95% of Na values. The variables are reduced to 54.

```
str(data)
```

```
## 'data.frame':    19622 obs. of  54 variables:
##  $ user_name           : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y         : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

The objective of this analysis is to classify the data into five different types based on the variable "class". For this reason, we converted the variable "class" into a categorical variable in order to change the type of adjustment from regression to classification.

Prediction Model Building

Data split

We split the data into training (70%) and test (30%).

```
# create training and test partition
inTrain <- createDataPartition(data$classe, p=0.7, list=FALSE)
training = data[inTrain,]
# dim(training)
testing = data[-inTrain,]
# dim(testing)
```

Random Forests

For the Random Forest analysis, the randomForest library has been selected and for its explanation the randomForestExplainer library has been selected.

A cross-validation with a value of 3 has been performed through the traincontrol function.

```
# model fit
control_RF <- trainControl(method="cv", number=3, verboseIter=FALSE)
#modfit_RF <- train(classe ~ ., data=training, method="rf",
#                          trControl=control_RF)
modfit_RF <- randomForest(classe ~ .,
                          data=training,
                          trControl=control_RF,
                          localImp = TRUE)
# modfit_RF$finalModel
modfit_RF
```

```
##
```

```
## Call:
##  randomForest(formula = classe ~ ., data = training, trControl = control_RF,      localImp = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.55%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    2    0    0    0 0.0005120328
## B    9 2644    5    0    0 0.0052671181
## C    0   14 2380    2    0 0.0066777963
## D    0    0   35 2214    3 0.0168738899
## E    0    0    2    3 2520 0.0019801980
```

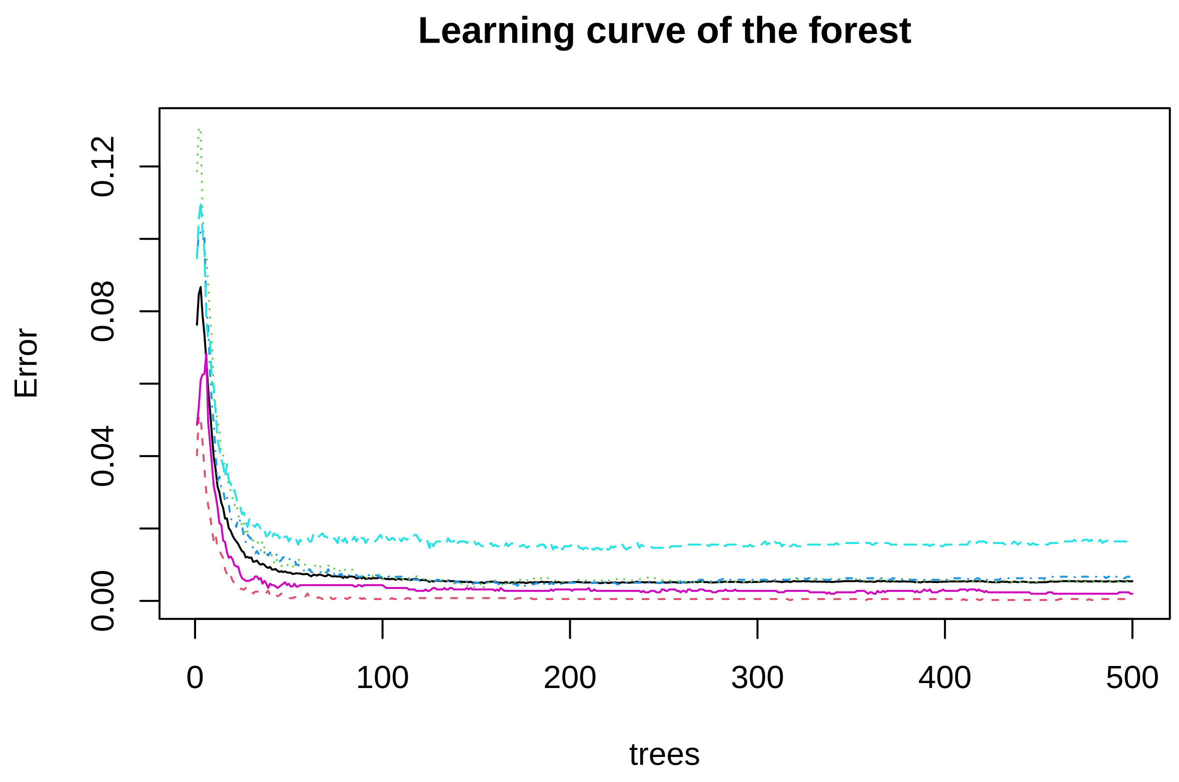
The Out-of-bag OOB error estimate rate is 0.55%, which is very good, i.e. 99.45 % accuracy. If we look at the Confusion Matrix, we can see that classification error is very low. This shows that our RF model is performing well in classifying the train set.

```
# prediction on Test dataset

predict_RF <- predict(modfit_RF, newdata=testing)
# plot(predict_RF)
confu_matrix_rf <- confusionMatrix(predict_RF, testing$classe)
confu_matrix_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    4    0    0    0
##          B    0 1134    7    0    0
##          C    0    1 1019   16    1
##          D    0    0    0  946    1
##          E    1    0    0    2 1080
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9921, 0.9961)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9956   0.9932   0.9813   0.9982
## Specificity            0.9991   0.9985   0.9963   0.9998   0.9994
## Pos Pred Value         0.9976   0.9939   0.9826   0.9989   0.9972
## Neg Pred Value          0.9998   0.9989   0.9986   0.9964   0.9996
## Prevalence              0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate          0.2843   0.1927   0.1732   0.1607   0.1835
## Detection Prevalence    0.2850   0.1939   0.1762   0.1609   0.1840
## Balanced Accuracy       0.9992   0.9971   0.9947   0.9906   0.9988
```

```
plot(modfit_RF, main = "Learning curve of the forest") #, xlim= c(0,100))
```

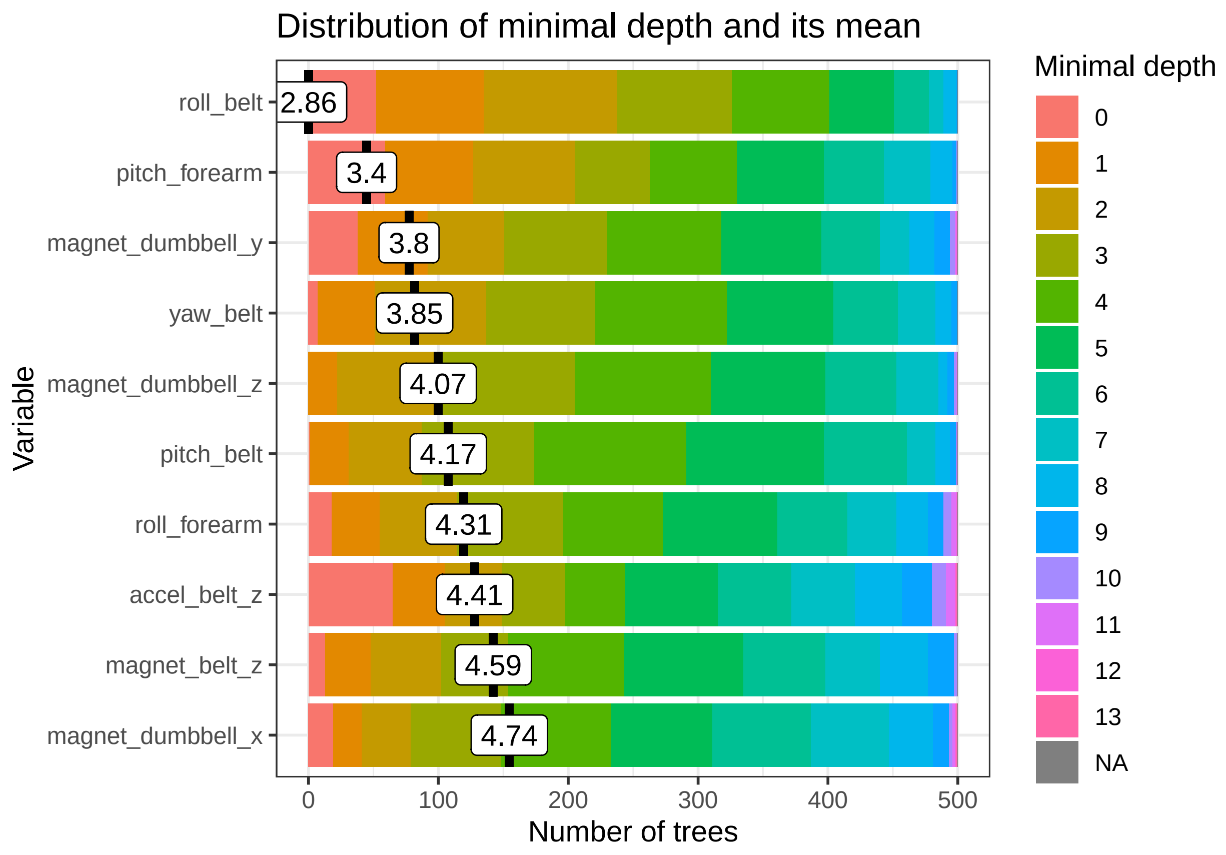


Distribution of minimal depth

The plot below shows the distribution of minimal depth among the trees of forest analysis. Note that:

- the mean of the distribution is marked by a vertical bar with a value label on it (the scale for it is different than for the rest of the plot),
- the scale of the X axis goes from zero to the maximum number of trees in which any variable was used for splitting.

```
min_depth_frame <- min_depth_distribution(modfit_RF)
# head(min_depth_frame, n = 10)
plot_min_depth_distribution(min_depth_frame)
```



Importance measures

The following table lists the parameters of the 10 most important variables ordered by mean_min_depth.

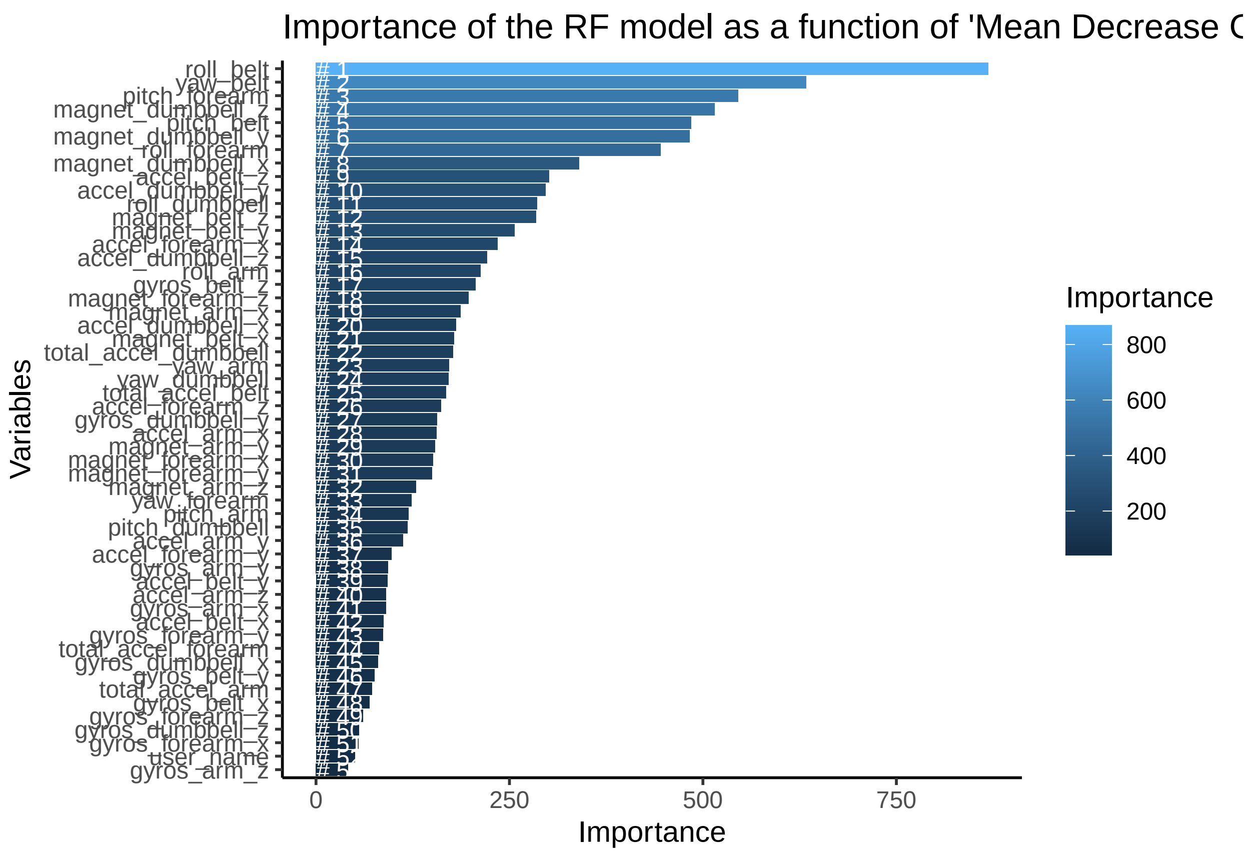
```
importance_frame <- measure_importance(modfit_RF)
knitr::kable(head(arrange(importance_frame, mean_min_depth), n = 10))
```

variable	mean_min_depth	no_of_nodes	accuracy_decrease	gini_decrease	no_of_trees	times_a_root	p_value
roll_belt	2.862	12187	0.1229790	868.8549	500	52	0.0000000
pitch_forearm	3.404	9077	0.0809053	545.3282	500	59	0.0000000
magnet_dumbbell_y	3.800	10302	0.1227885	482.8265	500	38	0.0000000
yaw_belt	3.852	14075	0.1160618	633.5841	500	7	0.0000000
magnet_dumbbell_z	4.072	11726	0.1067051	515.1004	500	0	0.0000000
pitch_belt	4.166	12495	0.0889413	485.0233	500	1	0.0000000
roll_forearm	4.310	9220	0.1303533	445.1350	500	18	0.0000000
accel_belt_z	4.412	7001	0.0437320	301.0979	500	65	0.3171061
magnet_belt_z	4.586	8038	0.0560768	284.1665	500	13	0.0000000
magnet_dumbbell_x	4.736	9503	0.0936211	340.1967	500	19	0.0000000

```
importance <- importance(modfit_RF)
varImportance <- data.frame(Variables = row.names(importance),
                            Importance =round(importance[, "MeanDecreaseGini"],2))
```

```r
rankImportance <- varImportance %>%
    mutate(Rank=paste("#",dense_rank(desc(Importance))))
```

```r
ggplot(rankImportance,aes(x=reorder(Variables,Importance),
                          y=Importance,fill=Importance))+
  geom_bar(stat="identity") +
  geom_text(aes(x = Variables, y = 0.5, label = Rank),
            hjust=0, vjust=0.55, size = 3, colour = "white") +
  labs(x = "Variables", title = "Importance of the RF model as a function of 'Mean Decrease Gini'") +
  coord_flip() +
  theme_classic()
```



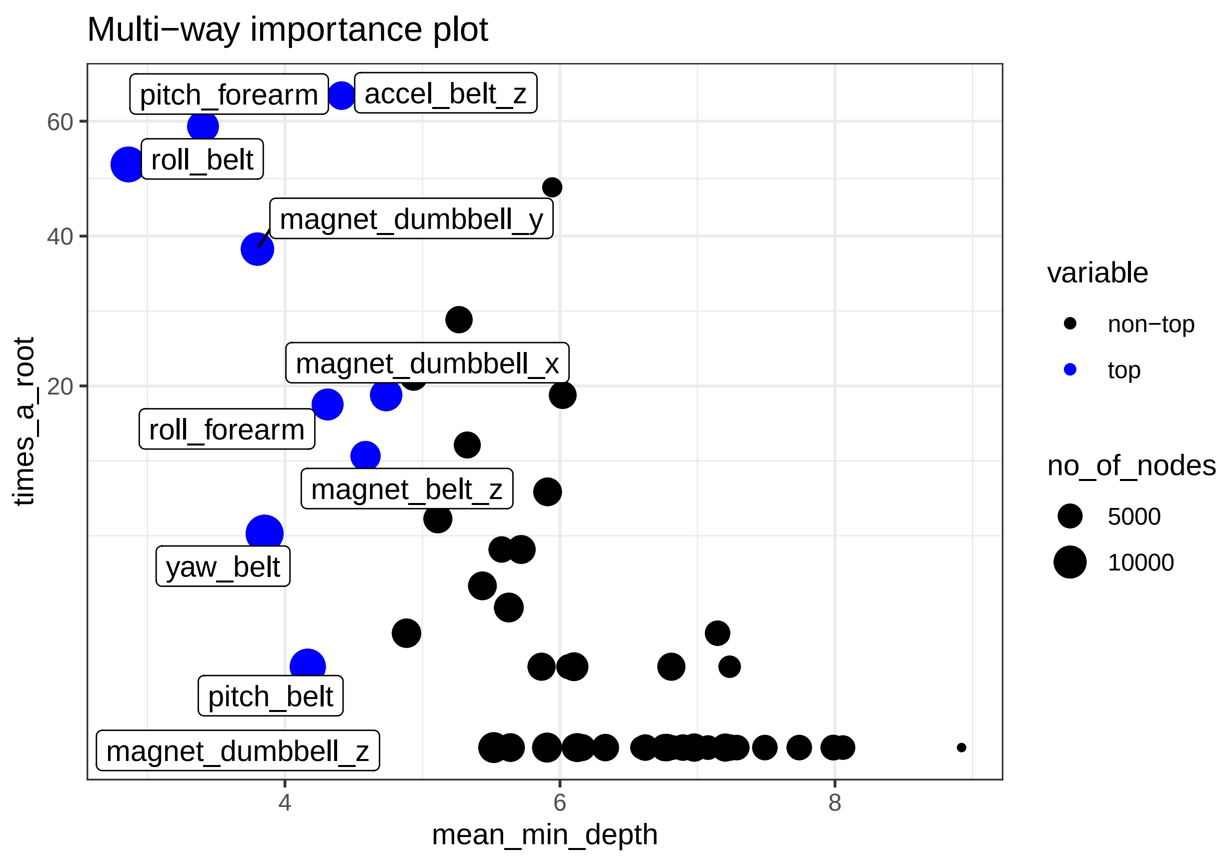
Multi-way importance plot

The multi-way importance plot shows the relation between three measures of importance and labels 10 variables which scored best when it comes to these three measures (i.e. for which the sum of the ranks for those measures is the lowest).

The first multi-way importance plot focuses on three importance measures that derive from the structure of trees in the forest:

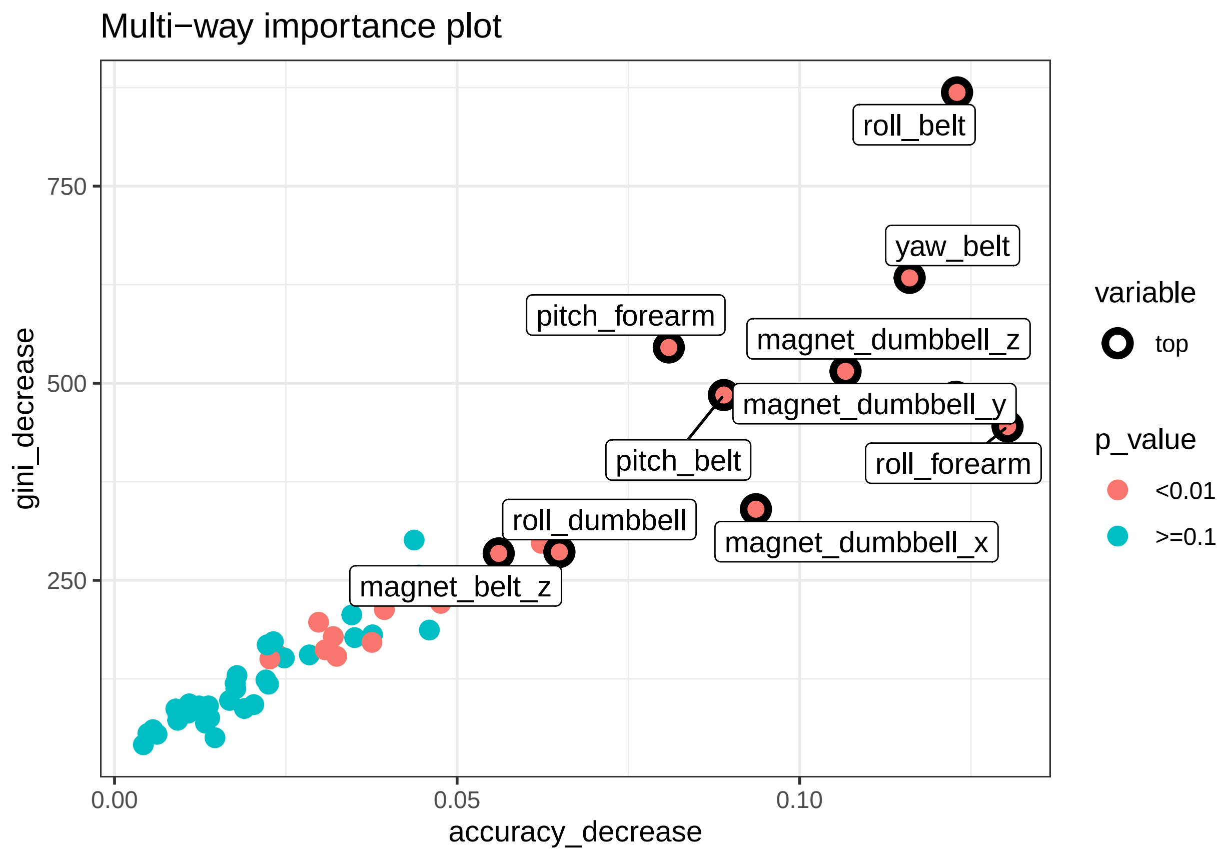
- mean depth of first split on the variable,
- number of trees in which the root is split on the variable,
- the total number of nodes in the forest that split on that variable.

```r
plot_multi_way_importance(importance_frame, size_measure = "no_of_nodes", min_no_of_trees = 30)
```



The second multi-way importance plot shows importance measures that derive from the role a variable plays in prediction: accuracy_decrease and gini_decrease with the additional information on the p-value based on a binomial distribution of the number of nodes split on the variable assuming that variables are randomly drawn to form splits (i.e. if a variable is significant it means that the variable is used for splitting more often than would be the case if the selection was random).

```r
plot_multi_way_importance(importance_frame, x_measure = "accuracy_decrease", y_measure = "gini_decrease
```



Applying the selected Model to the Test Data

The Random Forest model will be applied to predict the 20 quiz results (testing dataset) as shown below.

```
predict_test_cases <- predict(modfit_RF, newdata=test_cases)
predict_test_cases
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```