Practical Machine Learning - Course Project

Coursera Data Science Specialization

Jean Dos Santos

Table of Contents

Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks.

One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

More information is available through this link (see the section on the Weight Lifting Exercise Dataset).

Objective

The goal of this project is to predict the manner in which the exercise was done among one of the five classes. This is the "classe" variable in the training set.

Import Data

Use the read_csv function from readr to import the csv files.

```
# rm(list = ls())
library(readr)

# Import training and testing set
training <- read_csv(file = "pml-training.csv", na = c("NA", "#DIV/0!", ""),
progress = FALSE)
testing <- read_csv(file = "pml-testing.csv", na = c("NA", "#DIV/0!", ""),
progress = FALSE)
```

Process data

Remove unnecessary variables (User identification, time stamps and case numbers).

```
library(tidyverse)
# Remove unnecessary variables
training <- training %>%
  select(-X1, -user_name, -raw_timestamp_part_1, -raw_timestamp_part_2, -
cvtd_timestamp, -new_window, -num_window)

testing <- testing %>%
  select(-X1, -user_name, -raw_timestamp_part_1, -raw_timestamp_part_2, -
cvtd_timestamp, -new_window, -num_window)

# Convert all but classe variable to numeric variables
training <- data.frame(classe = training$classe, apply(training[,
1:ncol(training)-1], MARGIN = 2, FUN = as.numeric))
testing <- data.frame(Problem_id = testing$problem_id, apply(testing[,
1:ncol(testing)-1], MARGIN = 2, FUN = as.numeric))
```

Print summary statistics of training set.

```
training %>%
  select(-classe) %>%
  gather(key = Parameter, value = Value) %>%
  group_by(Parameter) %>%
```

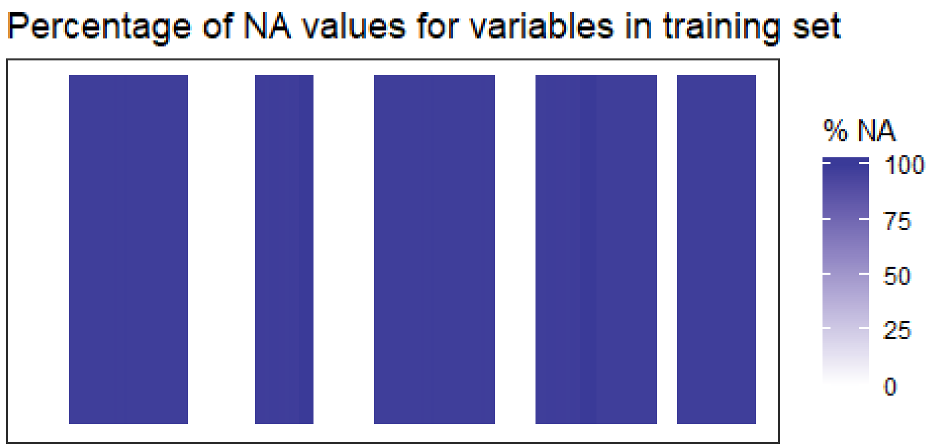
```
  summarise(Mean = round(mean(Value, na.rm = TRUE), 2),
            SD = round(sd(Value, na.rm = TRUE), 2),
            Min = round(min(Value, na.rm = TRUE), 2),
            Max = round(max(Value, na.rm = TRUE), 2),
            `% NA` = round(sum(is.na(Value))/n()*100, 2)
            )
```

```
# A tibble: 152 x 6
   Parameter          Mean    SD   Min   Max `% NA`
   <chr>             <dbl> <dbl> <dbl> <dbl>  <dbl>
 1 accel_arm_x      -60.2  182.   -404   437      0
 2 accel_arm_y       32.6  110.   -318   308      0
 3 accel_arm_z      -71.2  135.   -636   292      0
 4 accel_belt_x      -5.59  29.6  -120    85      0
 5 accel_belt_y      30.2   28.6   -69   164      0
 6 accel_belt_z     -72.6  100.   -275   105      0
 7 accel_dumbbell_x -28.6   67.3  -419   235      0
 8 accel_dumbbell_y  52.6   80.8  -189   315      0
 9 accel_dumbbell_z -38.3  109.   -334   318      0
10 accel_forearm_x  -61.6  181.   -498   477      0
# ... with 142 more rows
```

Based on the table above the values of the parameters included have different means and standard deviations. There are also several variables that have moslty `NA` values.

We can test if our predictive models are able to predict the classe by just including variables that have no or almost no `NA`.

```
NA_DF <- data.frame(Variable = names(training), `Percentage NA` =
colMeans(is.na(training))*100, row.names = NULL)

NA_DF %>%
  ggplot(mapping = aes(x = Variable, y = 1, fill = Percentage.NA)) +
    geom_tile() +
    # coord_flip() +
    scale_fill_gradient2() +
    theme_bw() +
    labs(title = "Percentage of NA values for variables in training set",
fill = "% NA") +
    theme(panel.grid = element_blank(), aspect.ratio = 0.5, axis.text.x =
element_blank(), axis.text.y = element_blank(), axis.ticks = element_blank(),
axis.title.y = element_blank(), axis.ticks.y = element_blank(), axis.title.x
= element_blank(), axis.ticks.x = element_blank())
```



Based on the plot above there are several variables that have mostly or exclusively NA values.

We will remove variables with mostly NA values:

```
# Remove columns with exclusively NA values
training <- training[, !apply(X = is.na(training), MARGIN = 2, FUN = all)]

# Remove variables with more than 95% of rows are NA
training <- training[, colMeans(is.na(training)) < 0.95]

# Remove the same columns in the test set
testing <- testing[, names(training)[-1]]
```

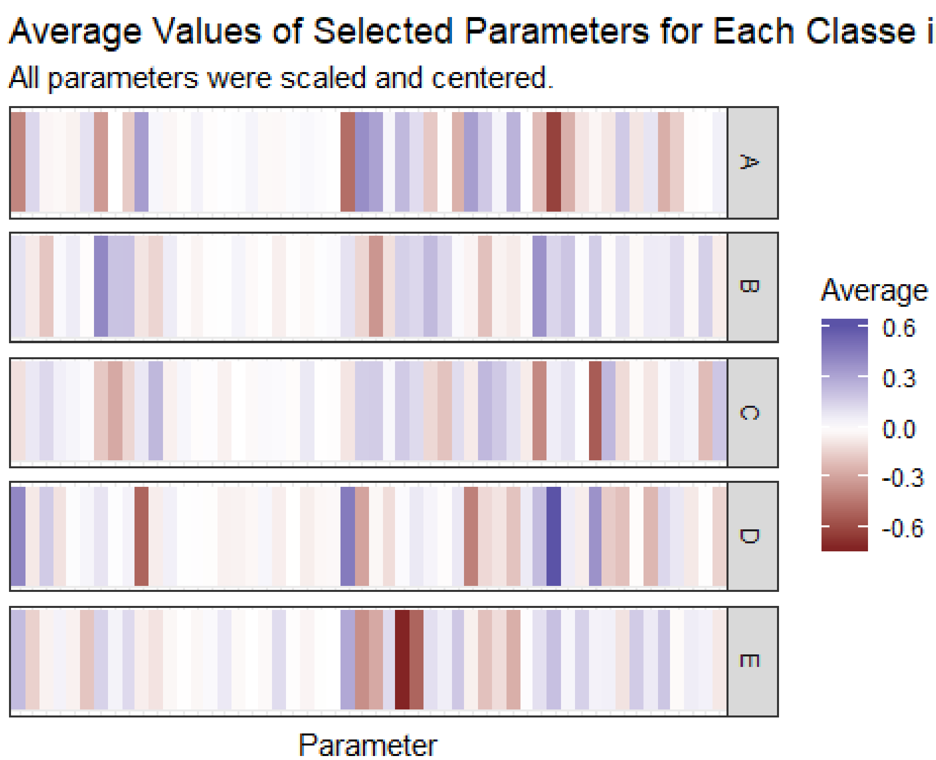
Based on the plot above there are several features that have no predictive value since they are populated mostly by NA values.

Exploratory Data Analysis

We will scale and centre the values of all parameters and plot their values.

```
library(tidyverse)
options(scipen = 99, digits = 3)
```

```r
scaled_training <- data.frame(classe = training$classe, apply(training[, -1],
MARGIN = 2, FUN = scale, center = TRUE, scale = TRUE))

scaled_training %>%
  gather(key = Parameter, value = Value, -classe) %>%
  group_by(Parameter = factor(Parameter), classe) %>%
  summarise(Average = mean(Value, na.rm = TRUE) # ,
            # SD = round(sd(Value, na.rm = TRUE), 2),
            # Min = min(Value, na.rm = TRUE),
            # Max = max(Value, na.rm = TRUE)
            ) %>%
  ggplot(mapping = aes(x = Parameter, y = 1, fill = Average)) +
    geom_tile() +
    facet_grid(classe~.) +
    scale_fill_gradient2() +
    labs(title = "Average Values of Selected Parameters for Each Classe in
the Training Set", subtitle = "All parameters were scaled and centered.") +
    theme_bw() +
    theme(axis.text.x = element_blank(), axis.text.y = element_blank(),
axis.ticks = element_blank(), axis.title.y = element_blank(), axis.ticks.y =
element_blank(), axis.ticks.x = element_blank())
```



Based on the plot above there seems to be significant differences between the various classes in one or more variables.

Modelling

Model training using k-fold cross-validation with 5 folds repeated 3 times. Accuracy will be used to select the optimal model.

In order to reduce computing time, we will use parallel processing using the `parallel` package.

```
N_folds <- 5 # number of folds
N_repetitions <- 3 # number of partitions to create

# # install.packages("doParallel")
library(caret); library(parallel); library(doParallel)
cluster <- makeCluster(detectCores() - 2) # number of cores, convention to
leave 1 core for OS
registerDoParallel(cluster) # register the parallel processing
set.seed(1) # set seed for reproducibility

# Training Options
control_options <- trainControl(method = "cv", # resampling method
                                          number = N_folds, # number of folds
                                          repeats = N_repetitions, # number
of repetitions
                                          # search = "grid",
                                          allowParallel = TRUE # allow
parallel processing
                                )

# Train random forest model
rf_model <- train(classe ~.,
                  method = "rf", # use random forests
                  data = training,
                  # preProcess = c("knnImpute", "center", "scale"),
                  na.action = "na.omit",
                  trControl = control_options)

stopCluster(cluster) # shut down the cluster
registerDoSEQ() #  force R to return to single threaded processing

# Save model in disk (optional)
saveRDS(object = rf_model, file = "rf_model.RSD")
```

Model Evaluation

Model Statistics

We can evaluate the obtained model by printing summary statistics and the confusion matrix. Since we repeated the process 3 times we obtain standard deviations for both the accuracy and Kappa.

```
rf_model$results

  mtry Accuracy Kappa AccuracySD KappaSD
1    2    0.995 0.993    0.00147 0.00186
2   27    0.994 0.992    0.00162 0.00205
3   52    0.989 0.986    0.00207 0.00262
```

Based on the table above the optimal model was obtained with `mtry=2` (i.e. 2 predictors are randomly selected at each node). The obtained accuracy was 0.995 and a Kappa value of 0.993.

Confusion Matrix

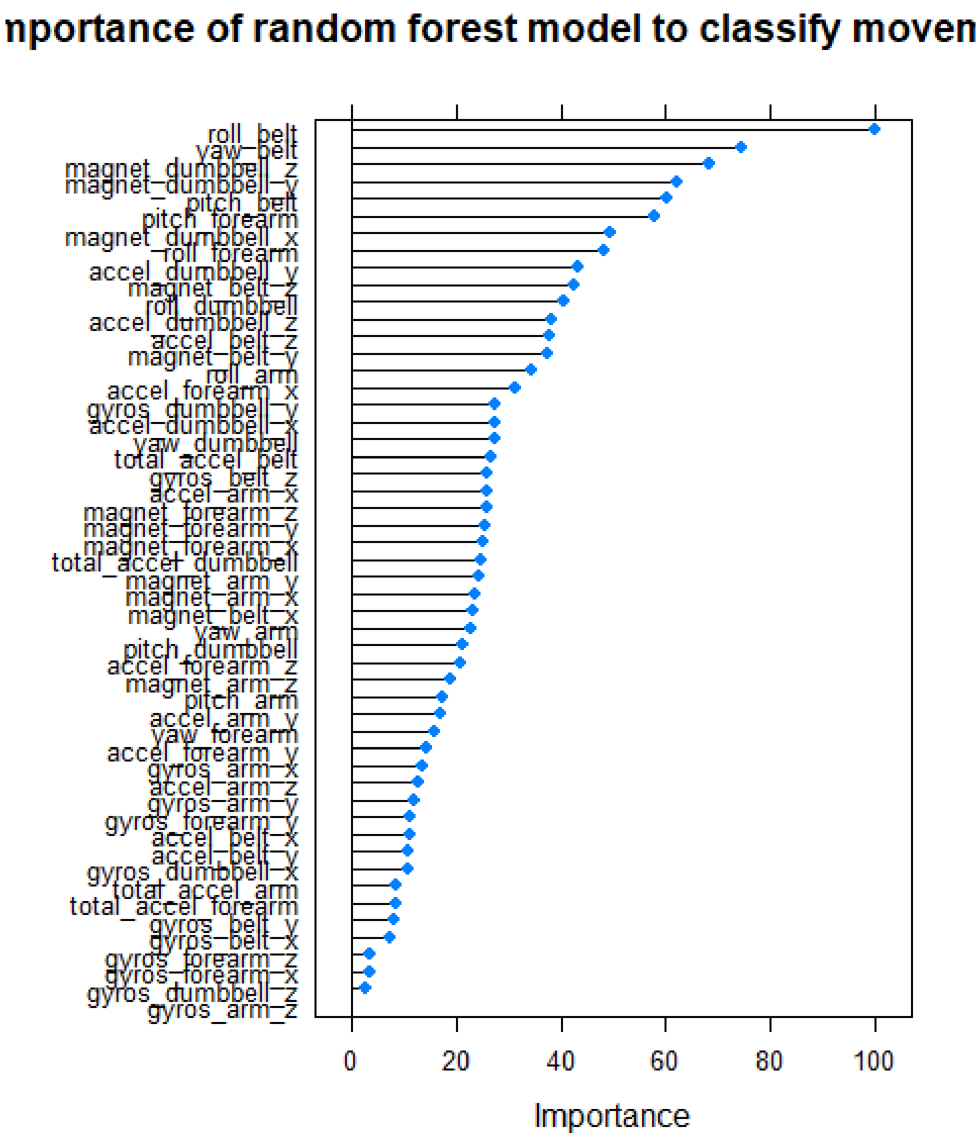
```
confusionMatrix(rf_model)

Cross-Validated (5 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction    A    B    C    D    E
         A 28.4  0.1  0.0  0.0  0.0
         B  0.0 19.3  0.1  0.0  0.0
         C  0.0  0.0 17.3  0.3  0.0
         D  0.0  0.0  0.0 16.1  0.0
         E  0.0  0.0  0.0  0.0 18.3

 Accuracy (average) : 0.9946
```

Based on the confusion matrix above, the majority of the errors occured between classe C and D, classe A and B, and classe B and C. Classe E correctly identied every single instance.

Variable Importance

We can use the `varImp` function from the `caret` package to calculate and plot variable importance

```
plot(varImp(rf_model), main = "Variable importance of random forest model to classify movement class")
```



Model Predictions

We can now use the random forest model obtained to make predictions on the testing set:

```
data.frame(ID = 1:nrow(testing), Prediction = predict(object = rf_model,
newdata = testing))
```

```
  ID Prediction
1  1          B
2  2          A
3  3          B
4  4          A
```

```
5   5           A
6   6           E
7   7           D
8   8           B
9   9           A
10 10           A
11 11           B
12 12           C
13 13           B
14 14           A
15 15           E
16 16           E
17 17           A
18 18           B
19 19           B
20 20           B
```

Reference

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. **Qualitative Activity Recognition of Weight Lifting Exercises**. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.