Prediction Assignment - Practical Machine Learning

Howard Tsang

April 25, 2017

Background

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, I 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. The goal of my model is to predict which ways they performed the exercise (“classes”) using accelerometers data [1].

Data

The sensor data consist data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. Our outcome variable is “classe”, a factor variable with 5 levels. For this data set, “participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions”.

Class A: exactly according to the specification Class B: throwing the elbows to the front Class C: lifting the dumbbell only halfway Class D: lowering the dumbbell only halfway Class E: throwing the hips to the front

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

Objectives

Prediction evaluations will be based on maximizing the accuracy and minimizing the out-of-sample error. All available variables (other than “classe”) will be used for prediction. Model bases on random forest algorithms will be fitted.

Loading Data & Libraries

Loading required libraries for analysis

```
set.seed(1636)
suppressMessages(library(dplyr))
suppressMessages(library(tidyr))
suppressMessages(library(caret))
suppressMessages(library(ggplot2))
suppressMessages(library(randomForest))
```

Loading data

```
train_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

if (!file.exists("train_data.csv")){
  download.file(train_url, destfile="train_data.csv")}
if (!file.exists("test_data.csv")){
  download.file(test_url, destfile="test_data.csv")}
```

Explore & Clean the Data

Data Preprocessing 1 - Replace missing values & excel division error strings “#DIV/0!” with “NA”.

Excel data cells have error strings “#DIV/0!” or blank value (missing data) are replaced with “NA”. I Data structure is explored as below.

```
train_data <- read.csv("train_data.csv", na.strings=c("NA","#DIV/0!",""), header=TRUE)
test_data <- read.csv("test_data.csv", na.strings=c("NA","#DIV/0!",""), header=TRUE)
str(train_data)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2
 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1
323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390
 484323 484434 ...
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9
 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ 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 -9
4.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##  $ kurtosis_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##  $ kurtosis_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

Data Preprocessing 2 - Cleaning variables with too many NAs (i.e. variables have “NA” higher than 97% threshold):

Obviously, there are lot of variables with “NA” value. We have to check if these variables have too many missing values that may not provide meaningful information for model fitting. Percentage of missing values of each variables in training and testing data sets are calculated.

```
NA_Train <- sapply(train_data, function(df) {sum(is.na(df)==TRUE)/length(df)})
NA_Test <- sapply(test_data, function(df) {sum(is.na(df)==TRUE)/length(df)})
table(NA_Train > .97)
```

```
##
## FALSE  TRUE
##    60   100
```

```
table(NA_Test > .97)
```

```
##
## FALSE  TRUE
##    60   100
```

The result indicates that 100 variables in both data sets have more that 97% missing values. Therefore we decide to remove these 100 variables from both the training and testing data set.

```
colnames1 <- names(which(NA_Train < 0.97))
train_datav2 <- train_data[, colnames1]
colnames2 <- names(which(NA_Test < 0.97))
test_datav2 <- test_data[, colnames2]
```

Double check if there are any “NA” missing value in training and testing data set.

```
sum(is.na(train_datav2) == TRUE)
```

```
## [1] 0
```

```
sum(is.na(test_datav2) == TRUE)
```

```
## [1] 0
```

Data Preprocessing 3 - Removing non-motion measurement data

By observation, column 1 to column 7 (column 1 : X, column 2: user name, column 3: raw_timestamp_part_1, column 4: raw_timestamp_part_2, column 5: cvtd_timestamp, column 6: new_window, and column 7: num_windoware) are metadata variables that do not provide motion related information and may create noise in the prediction.

```
train_datav3 <- train_datav2[,-c(1:7)]
test_datav3 <- test_datav2[,-c(1:7)]
```

Data Preprocessing 4 - Remove ZeroVariance & NearZeroVariance variables

We remove zero or near zero covariates predictors from both training and testing data sets. Zero or near zero covariates predictors are constant and almost constant across samples and do not contribute in model fitting.

```
nzv_train <- nearZeroVar(train_datav3, saveMetrics=TRUE)
train_datav4 <- train_datav3[, nzv_train$nzv==FALSE]
Clean_training<-train_datav4
```

```
nzv_test <- nearZeroVar(test_datav3, saveMetrics=TRUE)
test_datav4 <- test_datav3[, nzv_test$nzv==FALSE]
Clean_testing<-test_datav4
```

Summary of Data Preprocessing

After 4 stages of data preprocessing, the final training and testing data sets have 53 variables.

```
Train_Data<-c("Preprocess 1","Preprocess 2","Preprocess 3", "Preprocess 4")
Number_of_TrainVar<-c(dim(train_data)[2],dim(train_datav2)[2],dim(train_datav3)[2],dim(train_
datav4)[2])
Test_Data<-c("Preprocess 1","Preprocess 2","Preprocess 3","Preprocess 4")
Number_of_TestVar<-c(dim(test_data)[2],dim(test_datav2)[2],dim(test_datav3)[2],dim(test_datav
4)[2])

data.frame(Train_Data,Number_of_TrainVar)
```

```
##     Train_Data Number_of_TrainVar
## 1 Preprocess 1                160
## 2 Preprocess 2                 60
## 3 Preprocess 3                 53
## 4 Preprocess 4                 53
```

```
data.frame(Test_Data,Number_of_TestVar)
```

```
##      Test_Data Number_of_TestVar
## 1 Preprocess 1               160
## 2 Preprocess 2                60
## 3 Preprocess 3                53
## 4 Preprocess 4                53
```

Data Partitioning

For the purpose of cross-validation re-sampling methods, we split the cleaned training data set into training set (70%) and validating set (30%). We shall fit model with training set and validating the fitted model with validating set.

```
inTrain <- createDataPartition (Clean_training$classe, p=0.7, list=FALSE)
training <- Clean_training [inTrain ,]
validating <- Clean_training [- inTrain,]
```

Model fitting

In this study we attempt to predict the quality of exercise (“classes”) using accelerometers motion data. This is a classification problem and I choose to model it by random forest method. The reasons are: (1) even after data cleaning, we have sufficient amount of variables for random forest to run. Random forest scale well to large n. (2) Random forest able to automatically select important variables and only variables used in defining splits are in the model.

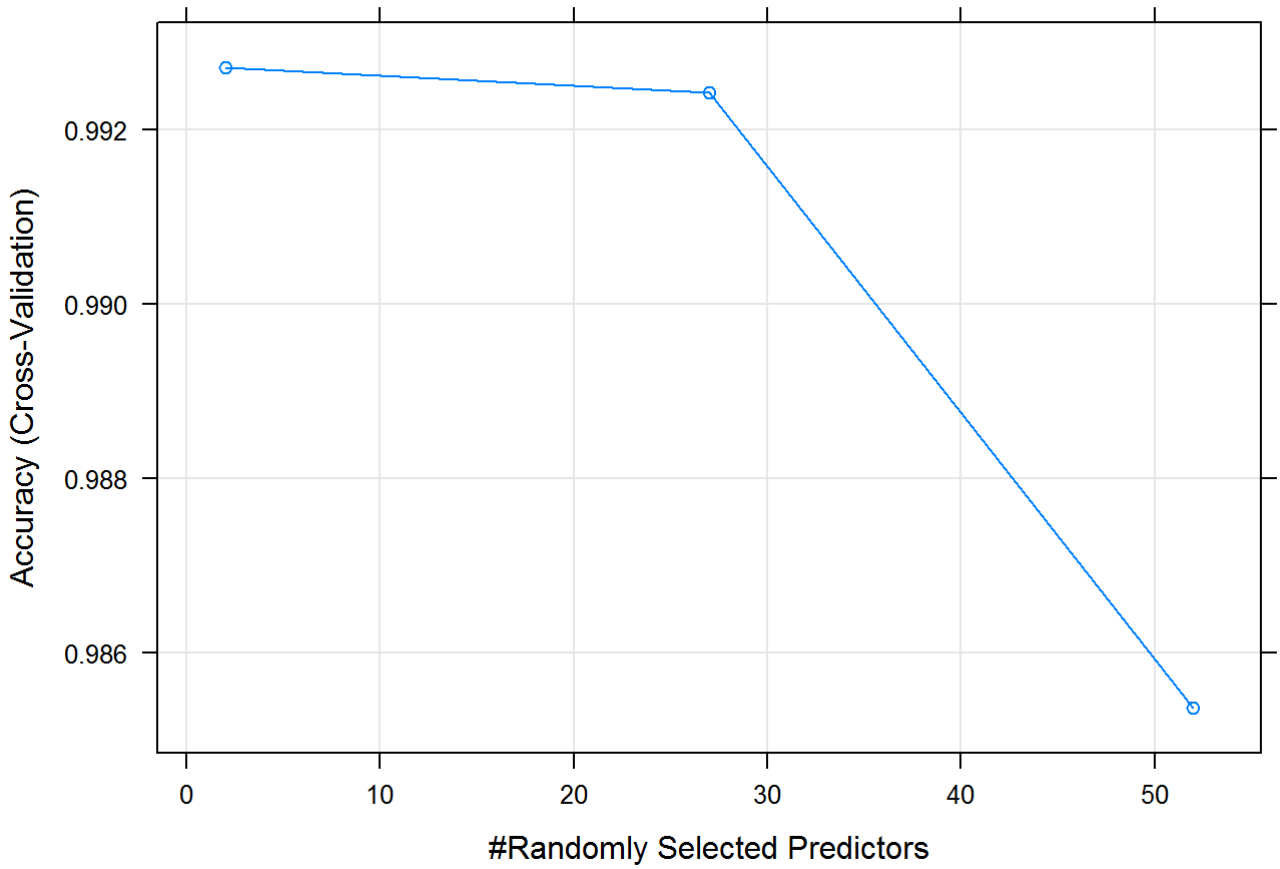
I run random forest with 10 folds cross validation, a common choice for the number of folds. In terms of the errors vs number of trees, as shows in following analysis, limits the number of trees to 200 is sufficient.

```
ctrl <- trainControl(method = "cv", number=10)

rf_fit <- train(classe ~ .,
             data = training,
             method = "rf",
             trControl = ctrl,
             allowParallel=TRUE,
             ntree=200)

print(rf_fit)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12364, 12363, 12364, 12362, 12362, 12363, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9927207  0.9907919
##   27    0.9924293  0.9904231
##   52    0.9853666  0.9814877
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

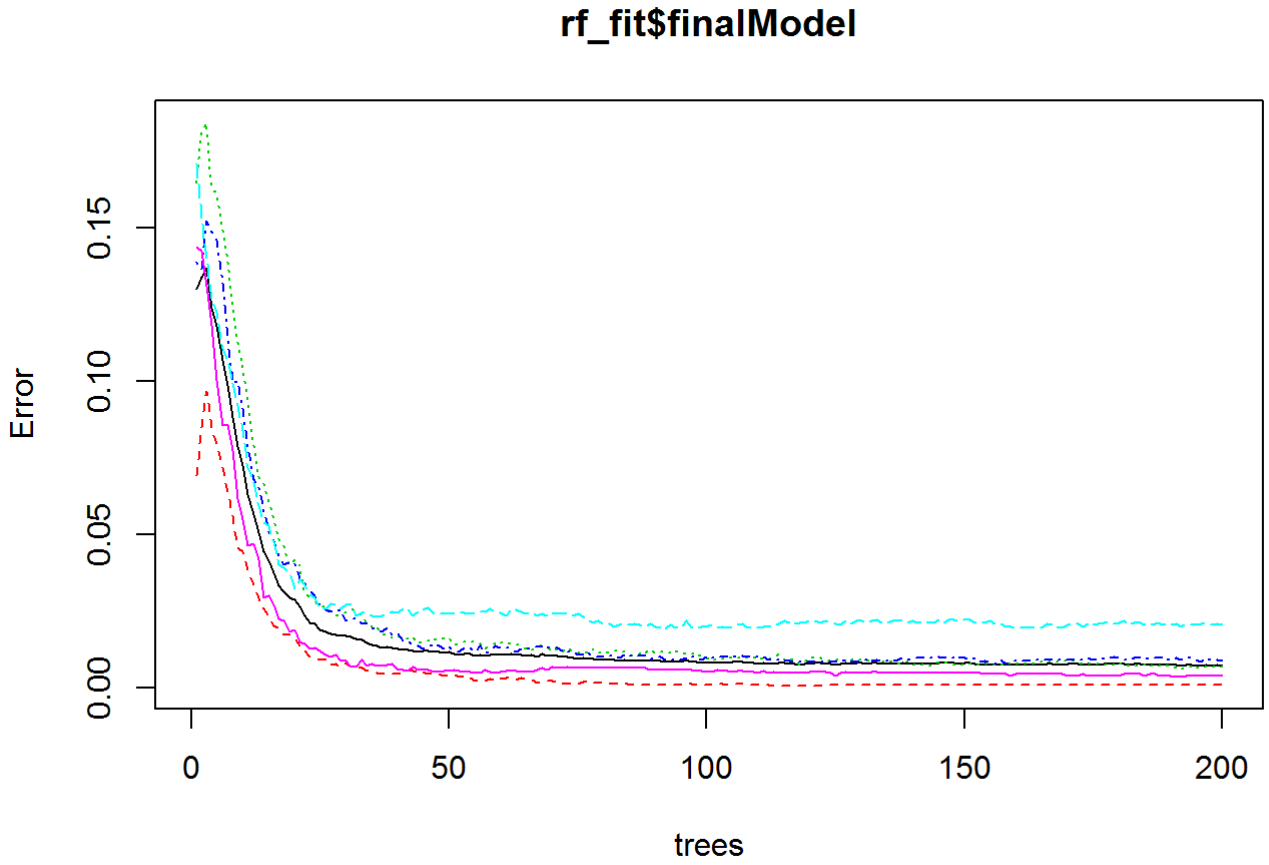
In term of accuracy vs number of predictors be considered for each split of tree (mtry), as showed in above analysis, the most accurate value for mtry was 2 with an accuracy of 99.27%.

```
plot(rf_fit)
```



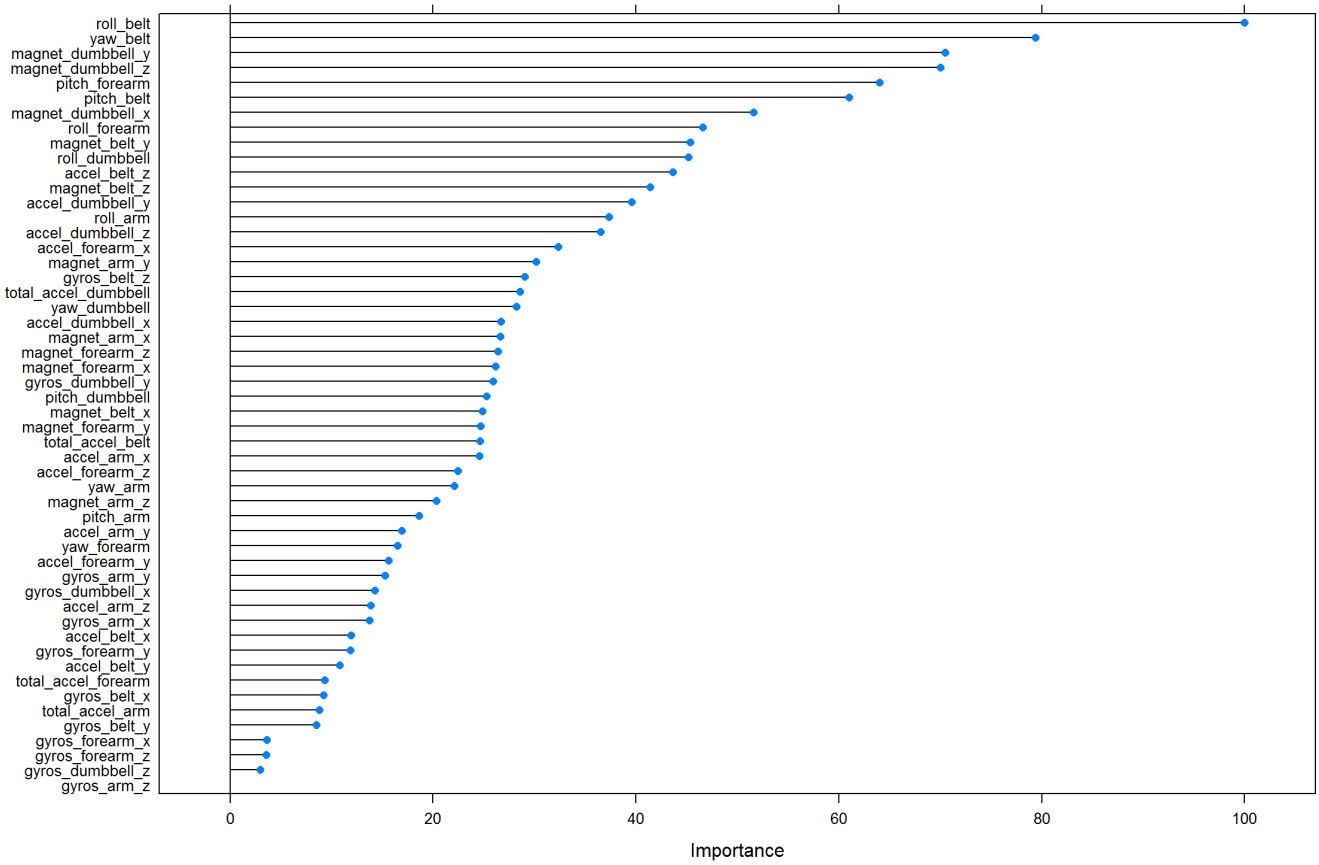
Considering the number of trees vs error, as showed in graph below, 200 is more than sufficient, even if we limit the number of trees to 50 will be sufficient to achieve our goal.

```
plot(rf_fit$finalModel)
```



Because each tree in a random forest uses a different set of variables, it is possible to keep track of which variables seem to be the most consistently influential. This is captured in the notion of importance. Here, we see that “roll_belt” and “yaw_belt” seem to be influential.

```
print(plot(varImp(rf_fit)))
```



Cross-Validation

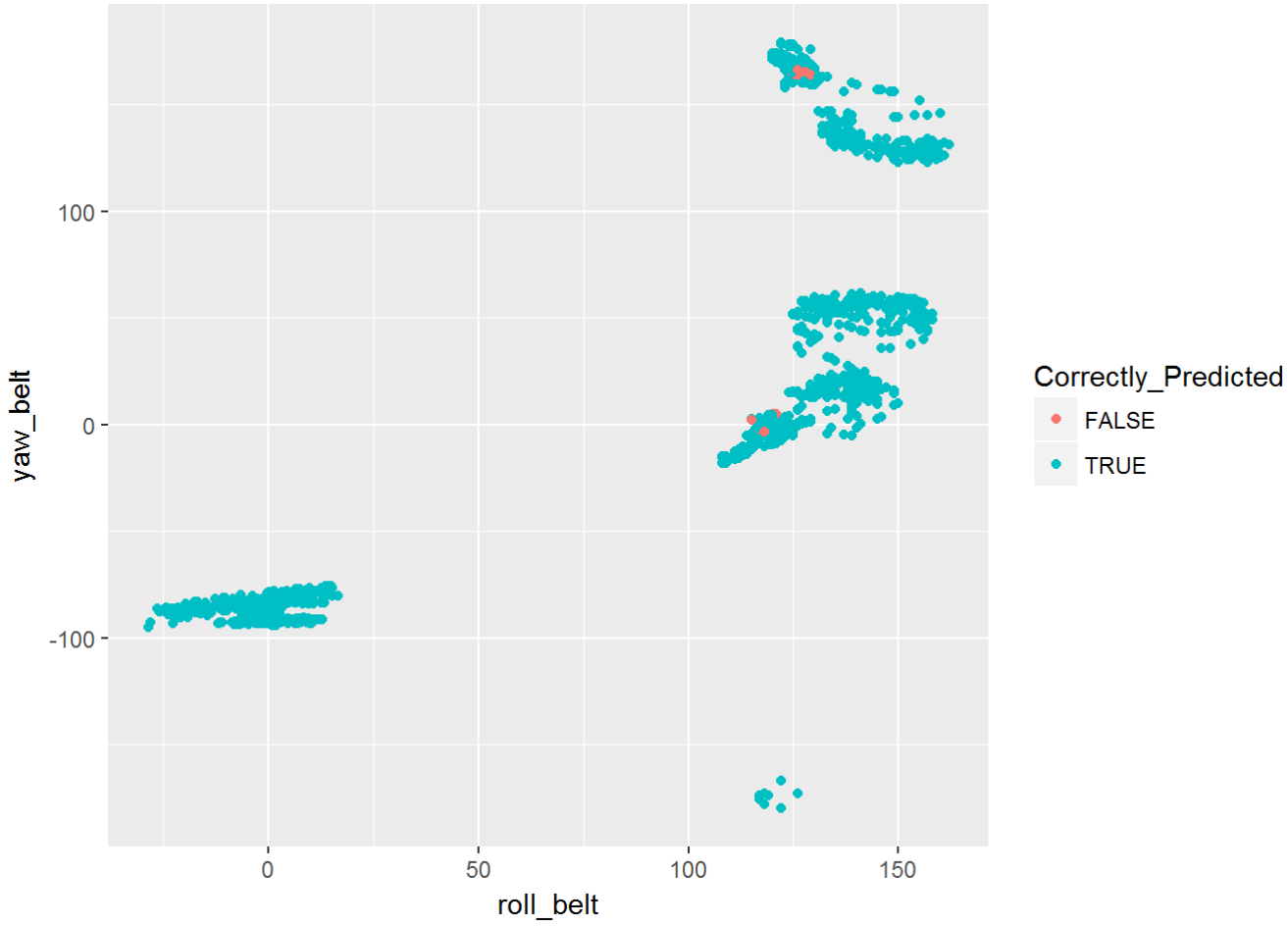
Let us evaluate the fitted model by validation data set. We compute the confusion matrix and associated statistics performance of the fitted model below.

```
pre_rf <- predict(rf_fit, newdata = validating)
confusionMatrix(data = pre_rf, validating$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    6    0    0    0
##          B    1 1133    9    0    0
##          C    0    0 1008   22    0
##          D    0    0    9  941    8
##          E    0    0    0    1 1074
##
## Overall Statistics
##
##                Accuracy : 0.9905
##                  95% CI : (0.9877, 0.9928)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.988
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9947   0.9825   0.9761   0.9926
## Specificity            0.9986   0.9979   0.9955   0.9965   0.9998
## Pos Pred Value         0.9964   0.9913   0.9786   0.9823   0.9991
## Neg Pred Value         0.9998   0.9987   0.9963   0.9953   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1925   0.1713   0.1599   0.1825
## Detection Prevalence   0.2853   0.1942   0.1750   0.1628   0.1827
## Balanced Accuracy      0.9990   0.9963   0.9890   0.9863   0.9962
```

The random forest fitted model predicted with 99.05% accuracy (with 95% confidence interval between 98.77% to 99.28%) in validation data set.

```
Correctly_Predicted <- pre_rf == validating$classe
qplot(roll_belt, yaw_belt, data=validating, col=Correctly_Predicted)
```



Prediction with testing data

Finally, we apply the testing data set to this random forest fitted model to predict “classe”. The predictions are showed below.

```
Modfit_pred_rf<-predict(rf_fit, Clean_testing)
Modfit_pred_rf
```

```
##  [1] 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
```

Reference:

[1] For more information of the data, refer to: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset). The training data for this project are available here: [https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)] The test data are available here: [https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)]