Writeup : Practical Machine Learning Project

Siti Salwani Yaacob

2/10/2020

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, your goal will be to 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 from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

What You Should Submit

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Analysis

Getting and Cleaning Data

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rpart)
library(rpart.plot)
#knitr::opts_chunk$set(cache=TRUE)

if (!file.exists("pml-training.csv" )){
        fileUrl = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
        download.file(fileUrl, destfile="./pml-training.csv", method = "curl")
}

if (!file.exists("pml-testing.csv" )){
        fileUrl = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
        download.file(fileUrl, destfile="./pml-testing.csv", method = "curl")
}

#Read in the data:
training_data <- read.csv("pml-training.csv", header = TRUE, sep = ",", na.strings = c("NA", ""))
testing_data <- read.csv("pml-testing.csv", header = TRUE, sep = ",", na.strings = c("NA", ""))
```

Check the data :

```
str(training_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 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
##  $ 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 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 396 levels "-0.016850","-0.021024",..: NA NA NA NA NA NA NA N
##  $ kurtosis_picth_belt     : Factor w/ 316 levels "-0.021887","-0.060755",..: NA NA NA NA NA NA NA N
##  $ kurtosis_yaw_belt       : Factor w/ 1 level "#DIV/0!": NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : Factor w/ 394 levels "-0.003095","-0.010002",..: NA NA NA NA NA NA NA N
##  $ skewness_roll_belt.1    : Factor w/ 337 levels "-0.005928","-0.005960",..: NA NA NA NA NA NA NA N
##  $ skewness_yaw_belt       : Factor w/ 1 level "#DIV/0!": NA NA NA NA 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            : Factor w/ 67 levels "-0.1","-0.2",..: 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            : Factor w/ 67 levels "-0.1","-0.2",..: 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      : Factor w/ 3 levels "#DIV/0!","0.00",..: 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       : Factor w/ 329 levels "-0.02438","-0.04190",..: NA NA NA NA NA NA NA NA I
##  $ kurtosis_picth_arm      : Factor w/ 327 levels "-0.00484","-0.01311",..: NA NA NA NA NA NA NA NA I
##  $ kurtosis_yaw_arm        : Factor w/ 394 levels "-0.01548","-0.01749",..: NA NA NA NA NA NA NA NA I
##  $ skewness_roll_arm       : Factor w/ 330 levels "-0.00051","-0.00696",..: NA NA NA NA NA NA NA NA I
##  $ skewness_pitch_arm      : Factor w/ 327 levels "-0.00184","-0.01185",..: NA NA NA NA NA NA NA NA I
##  $ skewness_yaw_arm        : Factor w/ 394 levels "-0.00311","-0.00562",..: NA NA NA NA NA NA NA NA I
##  $ 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  : Factor w/ 397 levels "-0.0035","-0.0073",..: NA NA NA NA NA NA NA NA NA
##  $ kurtosis_picth_dumbbell : Factor w/ 400 levels "-0.0163","-0.0233",..: NA NA NA NA NA NA NA NA NA
##  $ kurtosis_yaw_dumbbell   : Factor w/ 1 level "#DIV/0!": NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : Factor w/ 400 levels "-0.0082","-0.0096",..: NA NA NA NA NA NA NA NA NA
##  $ skewness_pitch_dumbbell : Factor w/ 401 levels "-0.0053","-0.0084",..: NA NA NA NA NA NA NA NA NA
##  $ skewness_yaw_dumbbell   : Factor w/ 1 level "#DIV/0!": NA NA NA NA 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        : Factor w/ 72 levels "-0.1","-0.2",..: 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        : Factor w/ 72 levels "-0.1","-0.2",..: 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]
```

From first view, it seems that there are a lot of variables containing NA's and missing values. Let's remove these from the training data;

```
na_training <- sapply(training_data, function(x) {sum(is.na(x))})
training_data <- training_data[,which(na_training == 0)]

na_testing <- sapply(testing_data, function(x) {sum(is.na(x))})
testing_data <- testing_data[, which(na_testing == 0)]

dim(training_data)
```

```
## [1] 19622    60
```

These variables have now been reduced from 160 to 60 in the dataset. Let's check which variables have near zero variance and remove them from the data:

```
nzv <- nearZeroVar(training_data, saveMetrics = TRUE)
training_data <- training_data[,nzv$nzv == "FALSE"]
training_data$classe <- as.factor(training_data$classe)

nzv <- nearZeroVar(testing_data, saveMetrics = TRUE)
testing_data <- testing_data[, nzv$nzv == "FALSE"]


dim(training_data)
```

```
## [1] 19622    59
```

Finally remove the first 6 variables, as they have nothing to do with making the predictions:

```
training_data <- training_data[,-c(1:6)]
testing_data <- testing_data[, -c(1:6)]
dim(training_data)
```

```
## [1] 19622    53
```

Cross Validation

Cross validation will be performed by subsampling our training data set randomly without replacement into 2 sub-samples: 70% of the data will be used for training the model and 30% for checking the prediction performance of the mode.

```
set.seed(12345)
inTrain <- createDataPartition(training_data$classe, p = 0.7, list = FALSE)
training <- training_data[inTrain,]
testing <- training_data[-inTrain,]
```

Prediction Model 1 : Random Forest

The method used for building the model will be Random Forest. The reason for this is that Random Forest is very accurate among other algorithms and it runs very efficiently on large data sets. We will run the set on 5-fold cross validation. In 5-fold cross-validation, the original sample is randomly partitioned into 5 equal sized subsamples. Of the 5 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining 4 subsamples are used as training data. The cross-validation process is then repeated 5 times (the folds), with each of the 5 subsamples used exactly once as the validation data. The 5 results from the folds can then be averaged (or otherwise combined) to produce a single estimation.

```
set.seed(12345)
random_forest <- train(classe ~., method = "rf", data = training,
                 trControl = trainControl(method = "cv", number = 5),
                 prox = TRUE, allowParallel = TRUE)

random_forest
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10991, 10989, 10990, 10988
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9911194  0.9887655
##   27    0.9901000  0.9874765
##   52    0.9844943  0.9803840
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

The model will be tested on the validation data (partition of the training data) and a confusion matrix will be used to check the accuracy of the prediction on the validation data:

```
predictTesting <- predict(random_forest, testing)
confusionMatrix(testing$classe, predictTesting)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    2    0    0    0
##          B   12 1121    6    0    0
##          C    0   16 1007    3    0
##          D    0    0   23  941    0
##          E    0    0    0    2 1080
##
## Overall Statistics
##
##                Accuracy : 0.9891
##                  95% CI : (0.9861, 0.9916)
##     No Information Rate : 0.2862
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9862
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9929   0.9842   0.9720   0.9947   1.0000
## Specificity            0.9995   0.9962   0.9961   0.9953   0.9996
## Pos Pred Value         0.9988   0.9842   0.9815   0.9761   0.9982
## Neg Pred Value         0.9972   0.9962   0.9940   0.9990   1.0000
## Prevalence             0.2862   0.1935   0.1760   0.1607   0.1835
## Detection Rate         0.2841   0.1905   0.1711   0.1599   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9962   0.9902   0.9840   0.9950   0.9998
```

The accuracy from the prediction model is 98.90% and the out of sample error is 1.10%. As this is a very accurate result, we will run the Random Forest model on the test data.

Prediction Model 2 : Random Forest

The Random Forest model is now applied to the test data to predict the outcome

```
answer <- predict(random_forest, testing_data)

answer
```

```
##  [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
```

Discussion

In this analysis, 19622 observations from weight lifting exercises were used to analyse and predict correct body movement from others during the exercise. 70% of the total observations (13737 observations) were used to build a model by using Random Forest Model. The rest 30% of the observations (5885 observations) were used for Cross Validation. The statistic shows that the built model had the overall accuracy of 98% for the testing set, which is not overlapping with observations used to built the model. The sensitivity was in between 92% - 99% and the specificity was over 98% for all classes (class A - class E, total 5 classes. Class A is a data from a correct exercise while the other classes were data from exercises done in a wrong way). Overall, the model is well developed to predict the exercise classes during weight lifting. As for the limitation

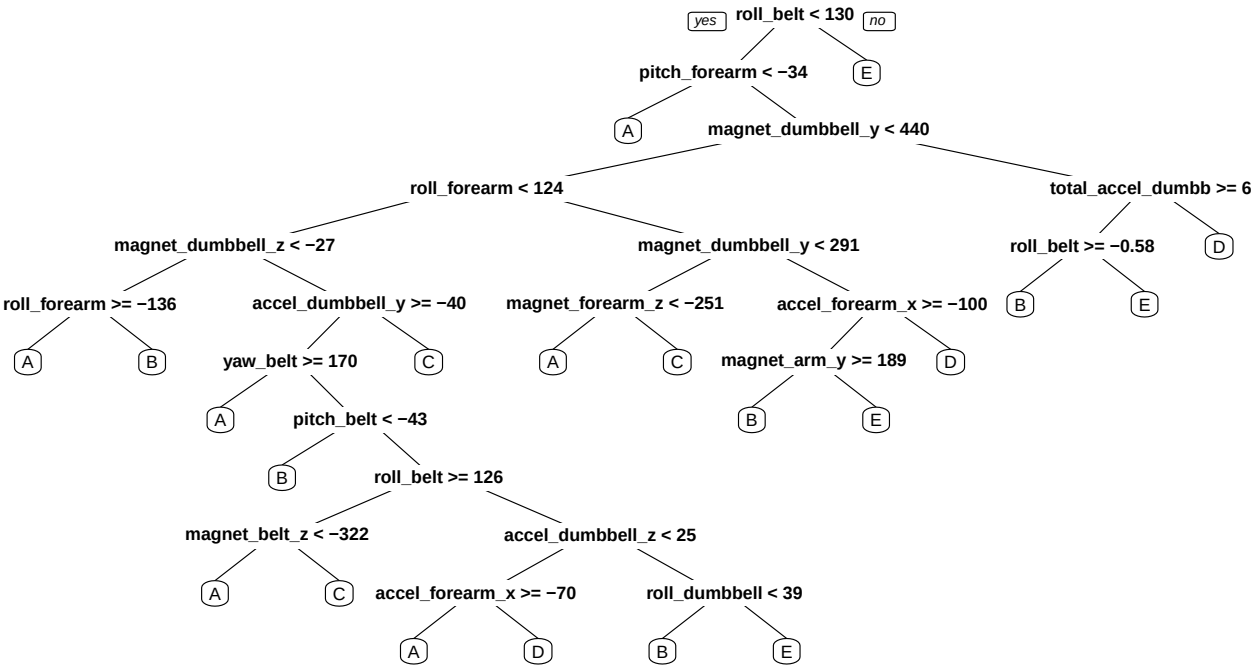
in this analysis, the observation data used in the analysis was collected from 6 young health participants in an experiment using Microsoft Kinetic. Therefore, under those condition, the model is expected to perform over 95% accuracy. However with different conditions, such as experiment with elderly people and or using different devices, the model might not perform well as shown in the analysis.

Credit : Michiko % P. Ohlson

Appendix

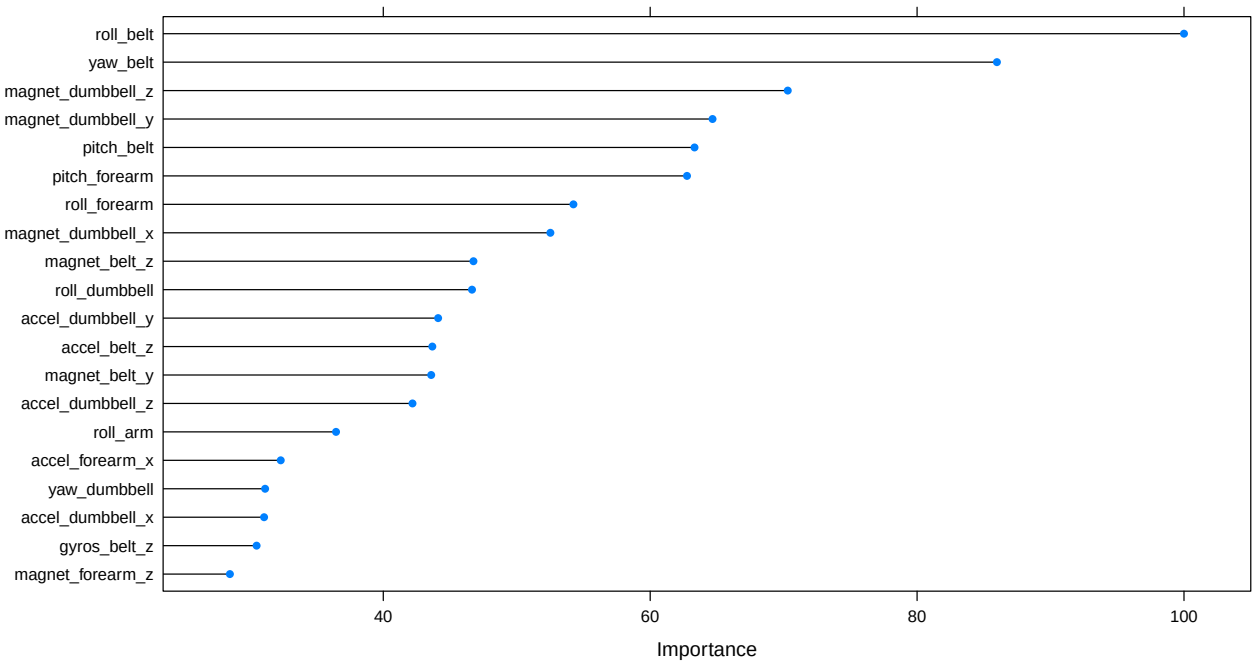
1. Random Forest Decision Tree

```
random_forest_tree <- rpart(classe ~., data = training, method = "class")
prp(random_forest_tree)
```



2. Top 20 Variables Impact on Outcome

```
plot(varImp(random_forest), top = 20)
```


roll_belt
yaw_belt
magnet_dumbbell_z
magnet_dumbbell_y
pitch_belt
pitch_forearm
roll_forearm
magnet_dumbbell_x
magnet_belt_z
roll_dumbbell
accel_dumbbell_y
accel_belt_z
magnet_belt_y
accel_dumbbell_z
roll_arm
accel_forearm_x
yaw_dumbbell
accel_dumbbell_x
gyros_belt_z
magnet_forearm_z
40
60
80
100
Importance