Prediction Assignment Writeup

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://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

Analysis

First, we load some useful packages using:

```{r warning=FALSE, error=FALSE}
library(rattle)
library(caret)
library(rpart)
library(rpart.plot)
library(corrplot)
library(randomForest)
library(RColorBrewer)
```

Getting Data

First of all, we need to set current working directory.

```{r warning=FALSE, error=FALSE}

setwd("~/GitHub/Practical-Machine-Learning-Johns-Hopkins-Bloomberg-School-of-Public-Health-Coursera/Project")

```

The following code fragment downloads the dataset to the `data` folder in the current working directory.

```{r warning=FALSE, error=FALSE}

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

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

trainFile <- "./data/pml-training.csv"

testFile  <- "./data/pml-testing.csv"

if (!file.exists("./data")) {

  dir.create("./data")

}

if (!file.exists(trainFile)) {

  download.file(trainUrl, destfile = trainFile, method = "curl")

}

if (!file.exists(testFile)) {

  download.file(testUrl, destfile = testFile, method = "curl")

}

rm(trainUrl)

rm(testUrl)

```

Reading Data

After downloading the data from the data source, we can read the two csv files into two data frames.

```{r warning=FALSE, error=FALSE}

trainRaw <- read.csv(trainFile)

testRaw <- read.csv(testFile)
```

dim(trainRaw)

dim(testRaw)

rm(trainFile)

rm(testFile)

```

The training data set contains `r dim(trainRaw)[1]` observations and `r dim(trainRaw)[2]` variables, while the testing data set contains `r dim(testRaw)[1]` observations and `r dim(testRaw)[2]` variables. The `classe` variable in the training set is the outcome to predict.

## Cleaning Data

Now we will clean the data, to get rid of observations with missing values, NA and unwanted values.

1. Cleaning the <b>Near Zero Variance</b> Variables.

```{r warning=FALSE, error=FALSE}

NZV <- nearZeroVar(trainRaw, saveMetrics = TRUE)

head(NZV, 20)

training01 <- trainRaw[, !NZV$nzv]

testing01 <- testRaw[, !NZV$nzv]

dim(training01)

dim(testing01)

rm(trainRaw)

rm(testRaw)

rm(NZV)

```

```
[1] 19622   100
[1]  20 100
```

2. Removing not required vbalues.

```{r warning=FALSE, error=FALSE}

regex <- grepl("^X|timestamp|user_name", names(training01))

training <- training01[, !regex]

testing <- testing01[, !regex]

rm(regex)

rm(training01)

rm(testing01)

dim(training)

dim(testing)

```
[1] 19622   95
[1] 20 95
```

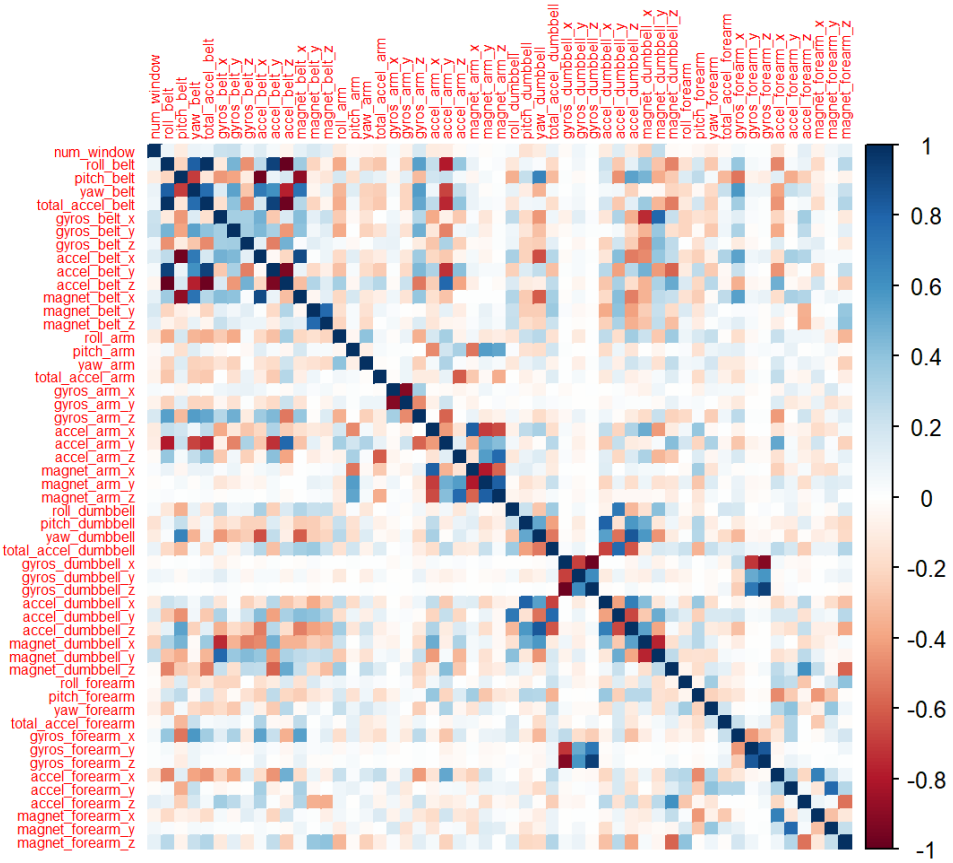
3. Removing NA values`.

```{r warning=FALSE, error=FALSE}
cond <- (colSums(is.na(training)) == 0)

training <- training[, cond]

testing <- testing[, cond]

rm(cond)
```

Now, the cleaned training data set contains `r dim(training)[1]` observations and `r dim(training)[2]` variables, while the testing data set contains `r dim(testing)[1]` observations and `r dim(testing)[2]` variables.

Correlation Matrix of Columns in the Training Data set.

```{r warning=FALSE, error=FALSE}
corrplot(cor(training[, -length(names(training))]), method = "color", tl.cex = 0.5)
```



Partitioning Training dataset into training (70%) and validation (30%)

The validation dataset will be used to conduct cross validation later.

```{r warning=FALSE, error=FALSE}

inTrain <- createDataPartition(training$classe, p = 0.70, list = FALSE)

validation <- training[-inTrain, ]

training <- training[inTrain, ]

rm(inTrain)

```

The Dataset now consists of `r dim(training)[2]` variables with the observations divided as following:

1. Training Data: `r dim(training)[1]` observations.

2. Validation Data: `r dim(validation)[1]` observations.

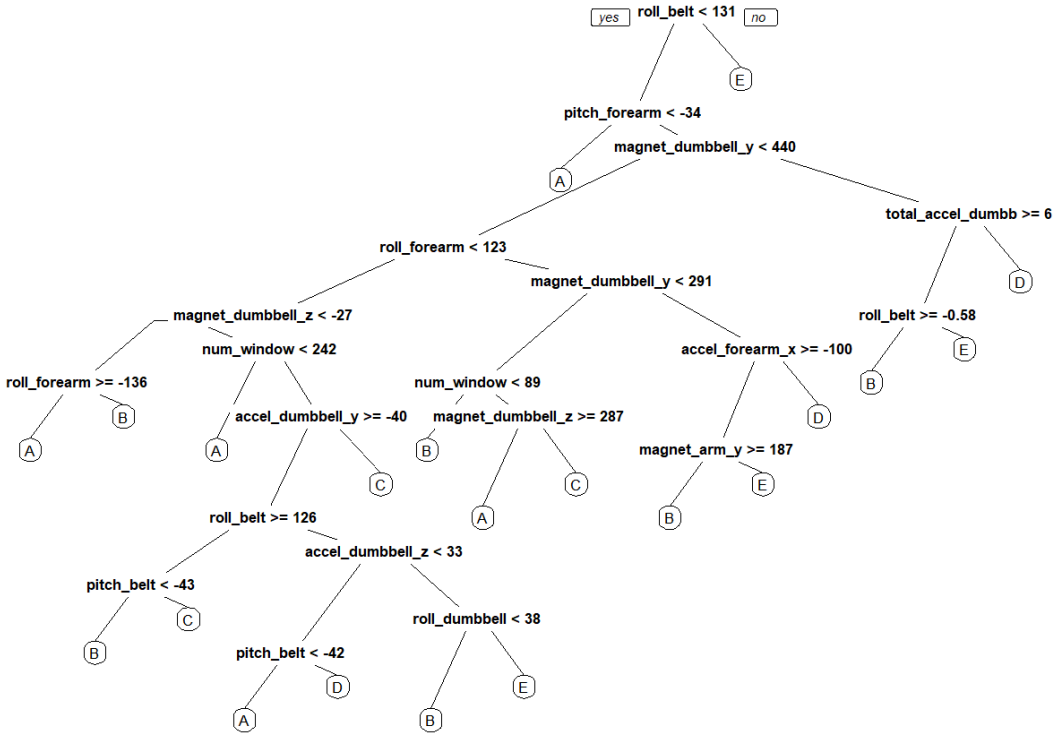
3. Testing Data: `r dim(testing)[1]` observations.

Data Modelling

Decision Tree

We fit a predictive model for activity recognition using Decision Tree algorithm.

```{r warning=FALSE, error=FALSE}

modelTree <- rpart(classe ~ ., data = training, method = "class")

prp(modelTree)

```



Random Forest

We fit a predictive model for activity recognition using Random Forest algorithm because it automatically selects important variables and is robust to correlated covariates & outliers in general.

We will use 5-fold cross validation when applying the algorithm.

```{r warning=FALSE, error=FALSE}

modelRF <- train(classe ~ ., data = training, method = "rf", trControl = trainControl(method = "cv", 5), ntree = 250)

modelRF

```

```
Random Forest

13737 samples
   53 predictor
    5 classes: 'A', 'B', 'C', 'D', 'E'
```

```
No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 10991, 10990, 10989, 10989, 10989
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.9935939  0.9918964
  27    0.9975248  0.9968692
  53    0.9953408  0.9941063

Accuracy *** used to select the optimal model using the largest value.
The final value used for the ***** was mtry = 27.
```

The Estimated Accuracy of the Random Forest Model is `r accuracy[1]*100`% and the Estimated Out-of-Sample Error is `r ose*100`%.

Random Forests yielded better Results, as expected!

Predicting The Manner of Exercise for Test Data Set

Now, we apply the Random Forest model to the original testing data set downloaded from the data source. We remove the problem_id column first.

```{r warning=FALSE, error=FALSE}

rm(accuracy)

rm(ose)

predict(modelRF, testing[, -length(names(testing))])

```