title: "Practical Machine Learning Coursera Peer Assessment" output: html_document: highlight: pygments keep_md: yes theme: united pdf_document: highlight: Aditya Tandon

date: "oct-8-2022"

Summary

This report uses machine learning algorithms to predict the manner in which users of exercise devices exercise.

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: (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

Set the work environment and knitr options

```{r setoptions} rm(list=ls(all=TRUE)) #start with empty workspace startTime <- Sys.time()

library(knitr) opts_chunk$set(echo = TRUE, cache= TRUE, results = 'hold')

```

### Load libraries and Set Seed

Load all libraries used, and setting seed for reproducibility. *Results Hidden, Warnings FALSE and Mes

```{r library_calls, message=FALSE, warning=FALSE, results='hide'}

library(ElemStatLearn)
library(caret)
library(rpart)
library(randomForest)
library(RCurl)
set.seed(2014)

```

Load and prepare the data and clean up the data

```{r Ass_Dir Hide, echo=FALSE} data_dir <- "C:/Users/User/Desktop/MachineLearning/Practical-Machine-Learning-Peer-
```

Assessment-1";

pathAnswers <- "C:/Users/User/Desktop/MachineLearning/Practical-Machine-Learning-Peer-Assessment-1/"

Load and prepare the data

```{r load_prep_call}

trainingLink <- getURL("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
pml_CSV  <- read.csv(text = trainingLink, header=TRUE, sep=",", na.strings=c("NA",""))

pml_CSV <- pml_CSV[,-1] # Remove the first column that represents a ID Row

```

Data Sets Partitions Definitions

Create data partitions of training and validating data sets.

```{r dataPart}

inTrain = createDataPartition(pml_CSV$classe, p=0.60, list=FALSE) training = pml_CSV[inTrain,] validating = pml_CSV[-inTrain,]

# number of rows and columns of data in the training set

dim(training)

# number of rows and columns of data in the validating set

dim(validating)
```

~~~
 ## Data Exploration and Cleaning

Since we choose a random forest model and we have a data set with too many columns, first we check if

```{r CkNA, echo=TRUE, results='asis'}

# Number of cols with less than 60% of data
sum((colSums(!is.na(training[,-ncol(training)])) < 0.6*nrow(training)))

# apply our definition of remove columns that most doesn't have data, before its apply to the model.

Keep <- c((colSums(!is.na(training[,-ncol(training)])) >= 0.6*nrow(training)))
training   <-  training[,Keep]
validating <- validating[,Keep]

# number of rows and columns of data in the final training set

dim(training)

# number of rows and columns of data in the final validating set

dim(validating)
~~~

Modeling

In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, during the execution. So, we proceed with the training the model (Random Forest) with the training data set.

```{r rf_apply}

model <- randomForest(classe~.,data=training) print(model)

```
### Model Evaluate
And proceed with the verification of variable importance measures as produced by random Forest:

```{r CkImportVar}

importance(model)
```

Now we evaluate our model results through confusion Matrix.

```{r confMx}

confusionMatrix(predict(model,newdata=validating[,-ncol(validating)]),validating$classe)
```

~~~
And confirmed the accuracy at validating data set by calculate it with the formula:

```{r CAccur}

accuracy <-c(as.numeric(predict(model,newdata=validating[,-ncol(validating)])==validating$classe))

accuracy <-sum(accuracy)*100/nrow(validating)
~~~

Model Accuracy as tested over Validation set = `r round(accuracy,1)`%.

Model Test

Finally, we proceed with predicting the new values in the testing csv provided, first we apply the same data cleaning operations on it and coerce all columns of testing data set for the same class of previous data set.

Getting Testing Dataset

```{r GetTestData}

testingLink <- getURL("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv") pml_CSV <- read.csv(text = testingLink, header=TRUE, sep=",", na.strings=c("NA",""))

pml_CSV <- pml_CSV[,-1] # Remove the first column that represents a ID Row pml_CSV <- pml_CSV[ , Keep] # Keep the same columns of testing dataset pml_CSV <- pml_CSV[,-ncol(pml_CSV)] # Remove the problem ID

# Apply the Same Transformations and Coerce Testing Dataset

# Coerce testing dataset to same class and strucuture of training dataset

testing <- rbind(training[100, -59] , pml_CSV)

# Apply the ID Row to row.names and 100 for dummy row from testing dataset

row.names(testing) <- c(100, 1:20)

```
#### Predicting with testing dataset

```{r PredictingTestingResults}

predictions <- predict(model,newdata=testing[-1,])
print(predictions)
```

The following function to create the files to answers the Prediction Assignment Submission:

```{r WriteResults}

pml_write_files = function(x){ n = length(x) for(i in 1:n){ filename = paste0(pathAnswers,"answers/problem_id_",i,".txt") write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE) } }

pml_write_files(predictions)

# get the time

```
```{r cache=FALSE}
endTime <- Sys.time()
```

The analysis was completed on `r format(Sys.time(), "%a %b %d %X %Y")` in `r round(difftime(endTime, startTime, units = c( "secs")),0)` seconds.