Assignment: Practical Machine Learning

Li Xin 25/Mar/2016

```
knitr::opts_chunk$set(echo=TRUE)
```

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, 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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Summary

In this project you creare a predictor to predict like how they did in the exercise; this is the "classe" variable in the training set.

The Random Forest algorithm perform well in this case and we can generate a model with more than 99% accuracy.

20 different test cases are provided for a final check in the model realized.

Import and Process Data

Import the necessary libraries and set the working directory and the seed for reproducibility.

```
library(rpart)
library(rpart.plot)
library(caret)
```

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

```
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(corrplot)
setwd("~/Practical_Machine_Learning")
set.seed(135246)
```

Both training and testing datasets are downloaded in the directory.

```
training <- read.csv(file="pml-training.csv",header=TRUE)
dim(training)
```

```
## [1] 19622   160
```

```
testing <- read.csv(file="pml-testing.csv",header=TRUE)
dim(testing)
```

```
## [1]  20 160
```

Training dataset contains 19622 observations and 160 variables; Testing dataset contains 20 observations and 160 variables. variable "classe" in the training set is the outcome to predict.

Remove columns full of missing values and columns which are not in numeric (but we will save class that is not numeric).

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
classe <- training$classe
training <- training[, sapply(training, is.numeric)]
training$classe <- classe
testing <- testing[, sapply(testing, is.numeric)]
```

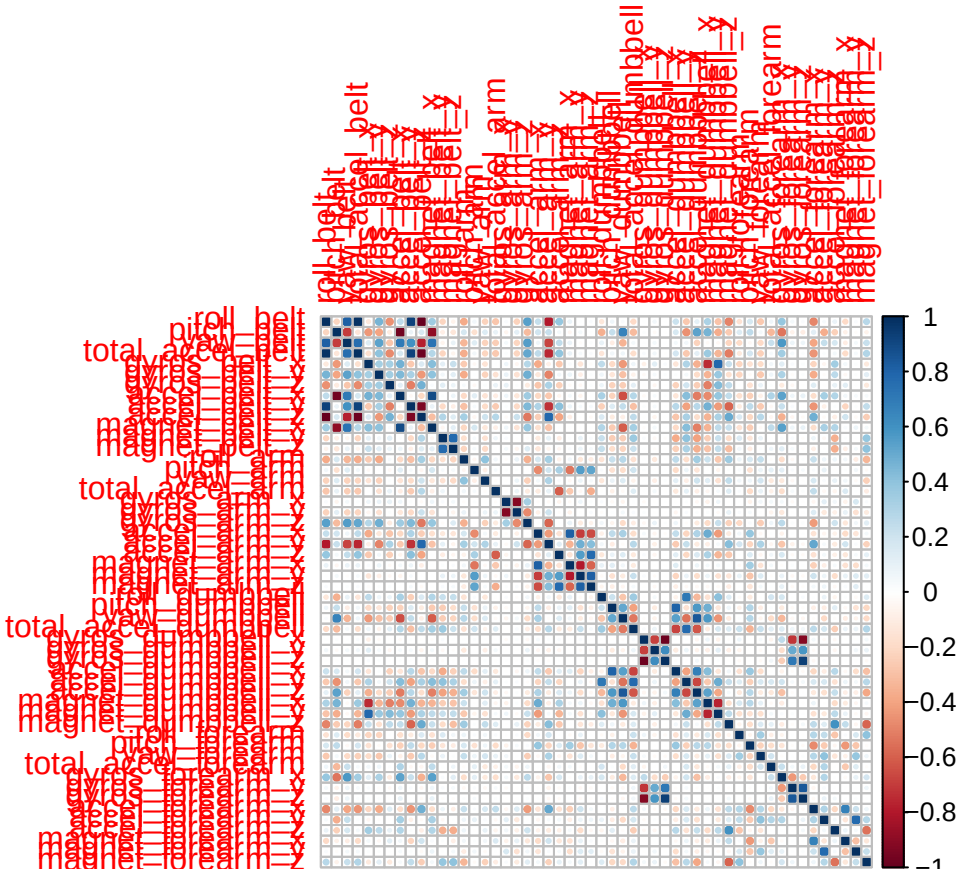
Remove variables - row counter, timestamps and windows.

```
filter <- grepl("^X|timestamp|window", names(training))
training <- training[, !filter]
filter <- grepl("^X|timestamp|window", names(testing))
testing <- testing[, !filter]
testing <- testing[, sapply(testing, is.numeric)]
```

The result of training dataset contains 19622 observations and 53 variables and the testing data set contains 20 observations and 53 variables.

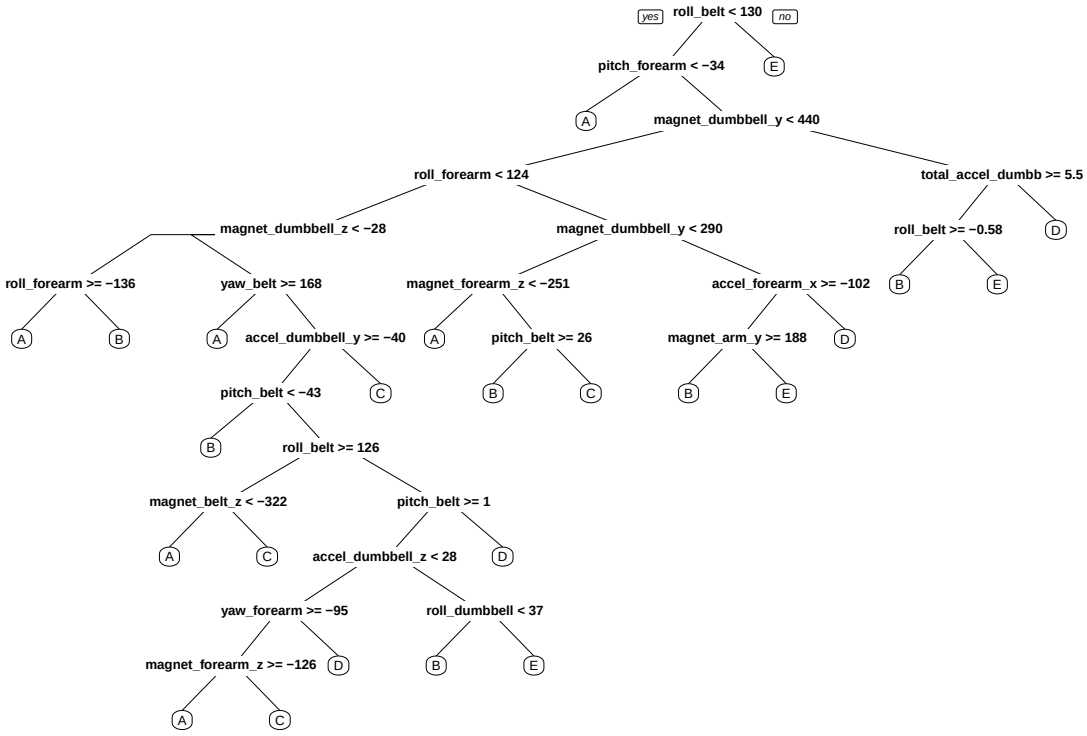
Plot the correlation between variables ("classe" is a factor and should be removed from the set.)

```
corrplot(cor(training[, -length(names(training))]))
```



The partitioning of data can be fitted in an easy model based on "classe" and the tree is printed as below.

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



Training dataset partitioning

Partition the training dataset to use 60% data for the training and remaining 40% for model testing.

```
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
splitTraining <- training[inTrain, ]
splitTesting <- training[-inTrain, ]
```

Random Forest

Following the machine learning steps, Random Forest algorithm is choosen to build the predictive model because it robust to outliers/correlated covariates and also applied using 5 folds.

```
model <- train(classe ~ ., data=splitTraining, method="rf", trControl=trainControl(method="cv", 5), ntr
model
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9420, 9420, 9423, 9421, 9420
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD   Kappa SD
##    2    0.9881963  0.9850665  0.0015424948  0.0019515413
##   27    0.9883663  0.9852818  0.0007077729  0.0008954458
##   52    0.9832701  0.9788334  0.0023464445  0.0029719500
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Then we can use the model on the test dataset (variable splitTesting) and compare the prediction with real results in a confusionMatrix.

```
predictions <- predict(model, splitTesting)
confusionMatrix(splitTesting$classe, predictions)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2227    4    1    0    0
##          B    9 1504    5    0    0
##          C    0    7 1353    8    0
##          D    0    0   22 1263    1
##          E    0    1    0    7 1434
##
## Overall Statistics
##
##                Accuracy : 0.9917
##                  95% CI : (0.9895, 0.9936)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9895
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9960   0.9921   0.9797   0.9883   0.9993
## Specificity            0.9991   0.9978   0.9977   0.9965   0.9988
## Pos Pred Value         0.9978   0.9908   0.9890   0.9821   0.9945
## Neg Pred Value         0.9984   0.9981   0.9957   0.9977   0.9998
## Prevalence             0.2850   0.1932   0.1760   0.1629   0.1829
## Detection Rate         0.2838   0.1917   0.1724   0.1610   0.1828
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9975   0.9949   0.9887   0.9924   0.9990
```

The result look good and accuracy of 99,17%

Prediction on test dataset

The final predict results on the testing dataset removing last column (problem_id).

```
result <- predict(model, testing[, -length(names(testing))])
result
```

```
##  [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
```

Write prediction results

The predictions files are saved as requested by project with the following code.

```
writeresults = function(x){
        n = length(x)
        for(i in 1:n)
                write.table(x[i],file=paste0("problem_id_",i,".txt"),quote=FALSE,row.names=FALSE,col.na
}
writeresults(result)
```