Practical Machine Learning Project

Predicting Exercise Manners

by **Yong Wu**

Introduction

This project aims to predict exercise manners based on data collected from wearable devices, such as Jawbone Up, Nike FuelBand, and Fitbit. We will be using measurements from accelerometers on the belt, forearm, arm and dumbell of six participants to predict the quality of their exercise. For more information, please visit(http://groupware.les.inf.puc-rio.br/har).

Data and Data Preprocessing

The data used for this exercise composes two parts: a training data set and a test data set. The training data set is available at here (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv); while the test data set is available at here (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv).

We start with loading necessary libraries required for the analyis.

```r
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(corrplot)
```

Subsequently, we download the data from the sources provided.

```r
setInternet2(use = TRUE) # Add this to make the https works on Windows,
                         # we also suppress the warnings
                         # to the next statement to filter some 'noise'.
train_URL <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_URL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

train_file <- "./data/pml-training.csv"
test_file  <- "./data/pml-testing.csv"

if (!file.exists("./data"))
{
  dir.create("./data")
}

if (!file.exists(train_file))
{
  suppressWarnings(download.file(train_URL, destfile = train_file))
}

if (!file.exists(test_file))
{
  suppressWarnings(download.file(test_URL, destfile = test_file))
}
```

Now we are ready to read the data into R and observe the structure of the two data frames.

```
train_raw <- read.csv("./data/pml-training.csv")
test_raw <- read.csv("./data/pml-testing.csv")
dim(train_raw)
```

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

```
dim(test_raw)
```

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

The training data set contains 19,622 observations and 160 variables, while the testing data set contains 20 observations and 160 variables. The `classe` variable in the training set is the outcome we want to predict.

```
sum(complete.cases(train_raw))
```

```
## [1] 406
```

A glance of both data sets indicates that there are some variables contain missing values, and there are some variables that are not very meaningful for our predication. We clean up the data (for both the train and test sets so we can apply the model built on the train data set to the test set later) so that we can concentrate on the most meaningful variables.

We clean the data in a few steps:

1. Columns that contain `NA` missing values are removed at first.
2. Columns that are not very meaningful, such as those with time stamp, are removed.
3. Columns that contain no numeric values are removed.

```
train_raw <- train_raw[, colSums(is.na(train_raw)) == 0]
test_raw <- test_raw[, colSums(is.na(test_raw)) == 0]
classe <- train_raw$classe
train_remove <- grepl("^X|timestamp|window", names(train_raw))
train_raw <- train_raw[, !train_remove]
train_clean_data <- train_raw[, sapply(train_raw, is.numeric)]
train_clean_data$classe <- classe
test_remove <- grepl("^X|timestamp|window", names(test_raw))
test_raw <- test_raw[, !test_remove]
test_clean_data <- test_raw[, sapply(test_raw, is.numeric)]
dim(train_clean_data)
```

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

```
dim(test_clean_data)
```

```
## [1] 20 53
```

We end up with 19,622 observations for the cleaned training data set, and 20 for the test data set. Both sets contain 53 variables. Care has been taken to make sure that the `classe` variable remains in the cleaned training sets.

Model Building and Cross Validation

We will not split the cleaned training set into a pure training data set (70%) and a validation data set (30%) for cross validation purpose, which will be conducted later.

```
set.seed(12321) # For reproducibile purpose
inTrain <- createDataPartition(train_clean_data$classe, p=0.70, list=F)
train_data <- train_clean_data[inTrain, ]
test_data <- train_clean_data[-inTrain, ]
```

In order to build the prediction model, we utilize the R package `randomForest` as it can automatically select key variables while maintain a high level of robustness. We will use 5-fold cross validation as well.

```
controlRf <- trainControl(method="cv", 5)
modelRf <- train(classe ~ ., data = train_data, method="rf", trControl = controlRf, ntree = 250)
modelRf
```

```
## 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, 10990, 10988, 10990, 10990
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9894440  0.9866445  0.003387060  0.004288084
##   27    0.9906089  0.9881195  0.003351225  0.004240477
##   52    0.9844211  0.9802894  0.005222552  0.006607347
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Now we test the prediction model on the validation data set that we partitioned earlier.

```
predictRf <- predict(modelRf, test_data)
confusionMatrix(test_data$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    0    1    0    1
##          B    5 1131    3    0    0
##          C    0    2 1020    4    0
##          D    0    0    9  954    1
##          E    0    0    1    3 1078
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9927, 0.9966)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9982   0.9865   0.9927   0.9981
## Specificity            0.9995   0.9983   0.9988   0.9980   0.9992
## Pos Pred Value         0.9988   0.9930   0.9942   0.9896   0.9963
## Neg Pred Value         0.9988   0.9996   0.9971   0.9986   0.9996
## Prevalence             0.2850   0.1925   0.1757   0.1633   0.1835
## Detection Rate         0.2841   0.1922   0.1733   0.1621   0.1832
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9983   0.9983   0.9926   0.9953   0.9987
```

```
accuracy <- postResample(predictRf, test_data$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9949023 0.9935518
```

```
oose <- 1 - as.numeric(confusionMatrix(test_data$classe, predictRf)$overall[1])
oose
```

```
## [1] 0.005097706
```

We can observe that the estimated accuracy of the model is 99.49% and the estimated out-of-sample error is 0.51%.

Prediction

The application to 20 test instance is rather straigtforward:

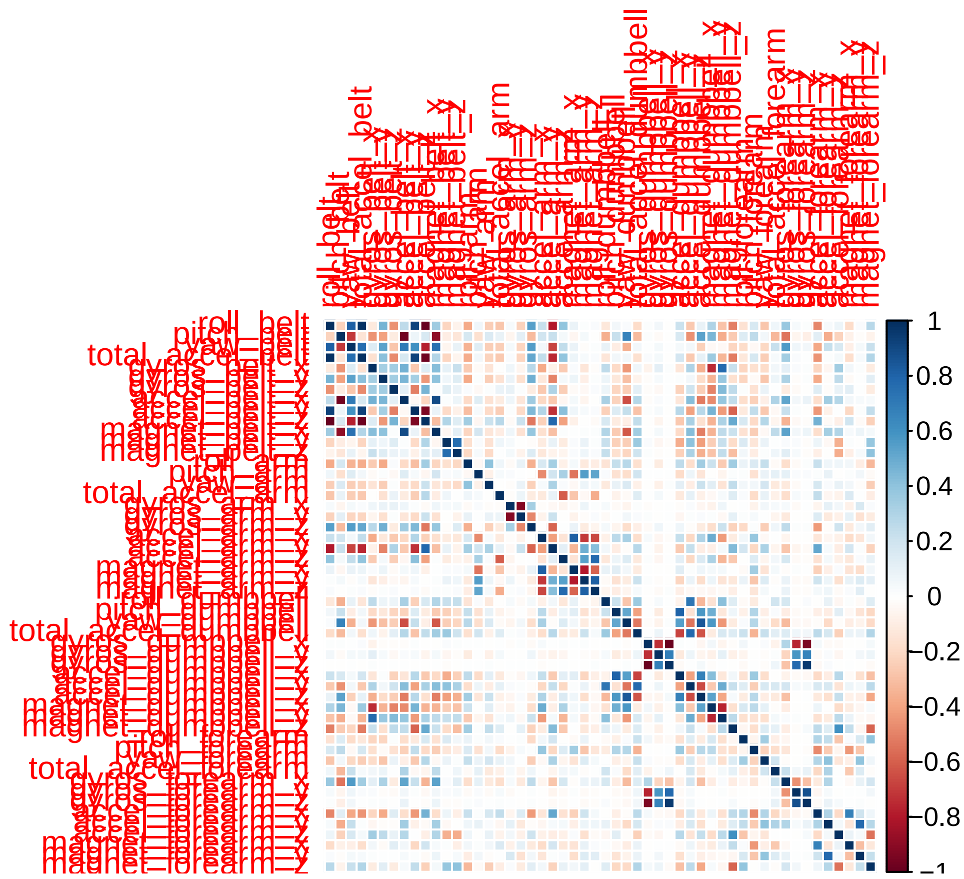
```
result <- predict(modelRf, test_clean_data[, -length(names(test_clean_data))])
result
```

```
##  [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
```

Appendix: Figures

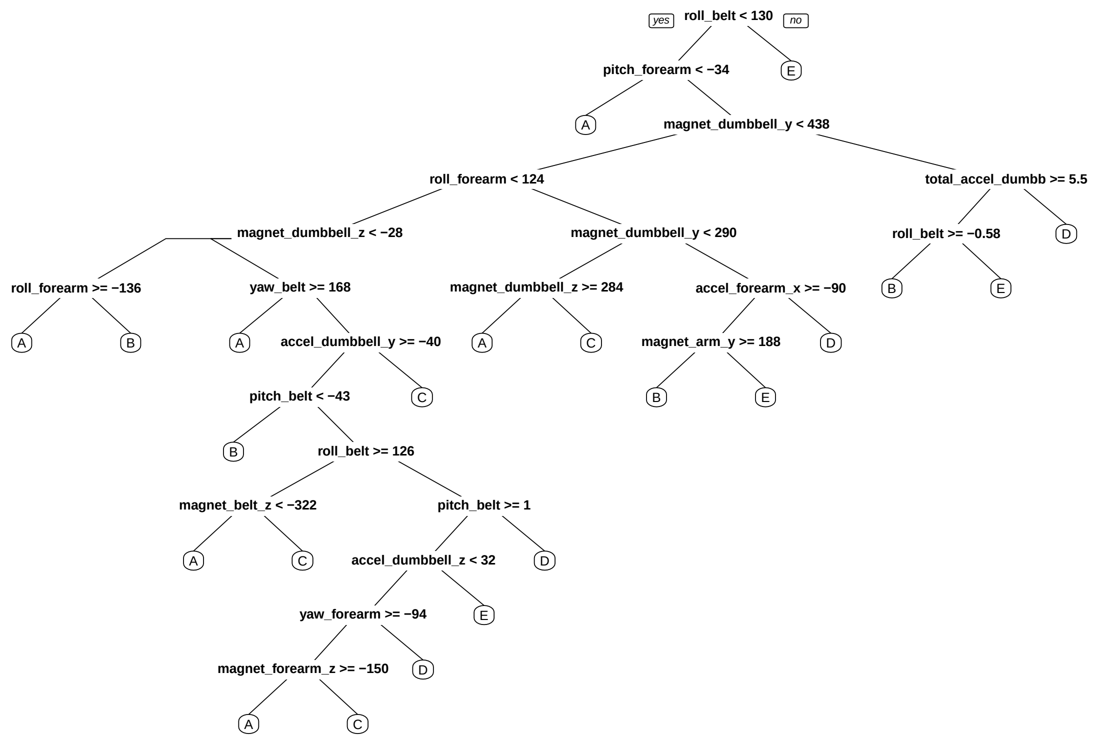
Correlation Matrix Visualization

```r
corrPlot <- cor(train_data[, -length(names(train_data))])
corrplot(corrPlot, method="color")
```



Decision Tree Visualization

```r
treeModel <- rpart(classe ~ ., data=train_data, method="class")
prp(treeModel) # fast plot
```


yes
roll_belt < 130
no
pitch_forearm < −34
E
A
magnet_dumbbell_y < 438
roll_forearm < 124
total_accel_dumbb >= 5.5
magnet_dumbbell_z < −28
magnet_dumbbell_y < 290
roll_belt >= −0.58
D
roll_forearm >= −136
yaw_belt >= 168
magnet_dumbbell_z >= 284
accel_forearm_x >= −90
B
E
A
B
A
accel_dumbbell_y >= −40
A
C
magnet_arm_y >= 188
D
pitch_belt < −43
C
B
E
B
roll_belt >= 126
magnet_belt_z < −322
pitch_belt >= 1
A
C
accel_dumbbell_z < 32
D
yaw_forearm >= −94
E
magnet_forearm_z >= −150
D
A
C