Prac_Mach_Learn_2

N3P2

6 October 2015

Introduction

Using health tracking 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, we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise.

Data Preprocessing

```
setwd("C:/Users/Qingyuan/Documents/8_Prac_Mach_Learn")
```

```
library(caret)
```

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

```
library(rpart)
library(rpart.plot)
library(randomForest)
```

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

```
library(corrplot)
```

Data download

```
training.file   <- 'pml-training.csv'
test.cases.file <- 'pml-test.csv'
training.url    <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
test.cases.url  <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'

download.file(training.url, training.file)
download.file(test.cases.url,test.cases.file)
```

Data read

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

```
trainRaw <- read.csv("pml-training.csv")
testRaw <- read.csv("pml-test.csv")
dim(trainRaw)
```

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

```
dim(testRaw)
```

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

The *training data set* contains **19622 observations and 160 variables**, while the *testing data set* contains **20 observations and 160 variables**.

Data cleaning

Cleaning Data

First all blank(‘ “” ’), ‘#DIV/0’ and ‘NA’ values are converted to ‘NA’. Any Columns containing ‘NA’ are removed from both downloaded data sets.

```
training.df   <-read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!", ""))
test.cases.df <-read.csv("pml-test.csv" , na.strings=c("NA", "#DIV/0!", ""))
training.df<-training.df[,colSums(is.na(training.df)) == 0]
test.cases.df <-test.cases.df[,colSums(is.na(test.cases.df)) == 0]
```

The first 7 columns are not related to calculations thus removed from data sets. 1. *user_name* 2. *raw_timestamp_part_1* 3. *raw_timestamp_part_2* 4. *cvtd_timestamp* 5. *new_window* 6. *num_window*

```
training.df   <-training.df[,-c(1:7)]
test.cases.df <-test.cases.df[,-c(1:7)]
```

Now, the cleaned training data set contains **19622 observations and 53 variables**, while the testing data set contains **20 observations and 53 variables**. The “classe” variable is still in the cleaned training set.

Slice the data

Then, we can split the cleaned training set into a pure training data set (70%) and a validation data set (30%). We will use the validation data set to conduct cross validation in future steps.

```
set.seed(22519) # For reproducibile purpose

inTrain <- createDataPartition(training.df$classe, p = 0.75, list = F)
trainData <- training.df[inTrain, ]
testData  <- training.df[-inTrain, ]
```

Data Modeling

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.

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

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11776, 11775, 11775, 11773, 11773
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD   Kappa SD
##    2    0.9914389  0.9891698  0.0008472351  0.001072009
##   27    0.9912350  0.9889124  0.0015647255  0.001979460
##   52    0.9857317  0.9819498  0.0021088807  0.002668672
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Then, we estimate the performance of the model on the validation data set.

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    0    0    0    1
##          B    4  940    5    0    0
##          C    0    3  850    2    0
##          D    0    0   15  789    0
##          E    0    0    0    4  897
##
## Overall Statistics
##
##                Accuracy : 0.9931
##                  95% CI : (0.9903, 0.9952)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9912
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9971   0.9968   0.9770   0.9925   0.9989
## Specificity            0.9997   0.9977   0.9988   0.9963   0.9990
## Pos Pred Value         0.9993   0.9905   0.9942   0.9813   0.9956
## Neg Pred Value         0.9989   0.9992   0.9951   0.9985   0.9998
## Prevalence             0.2851   0.1923   0.1774   0.1621   0.1831
## Detection Rate         0.2843   0.1917   0.1733   0.1609   0.1829
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9984   0.9973   0.9879   0.9944   0.9989
```

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

```
##  Accuracy     Kappa
## 0.9930669 0.9912299
```

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

```
## [1] 0.006933116
```

So, the estimated accuracy of the model is 99.42% and the estimated out-of-sample error is 0.58%.

Predicting for Test Data Set

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

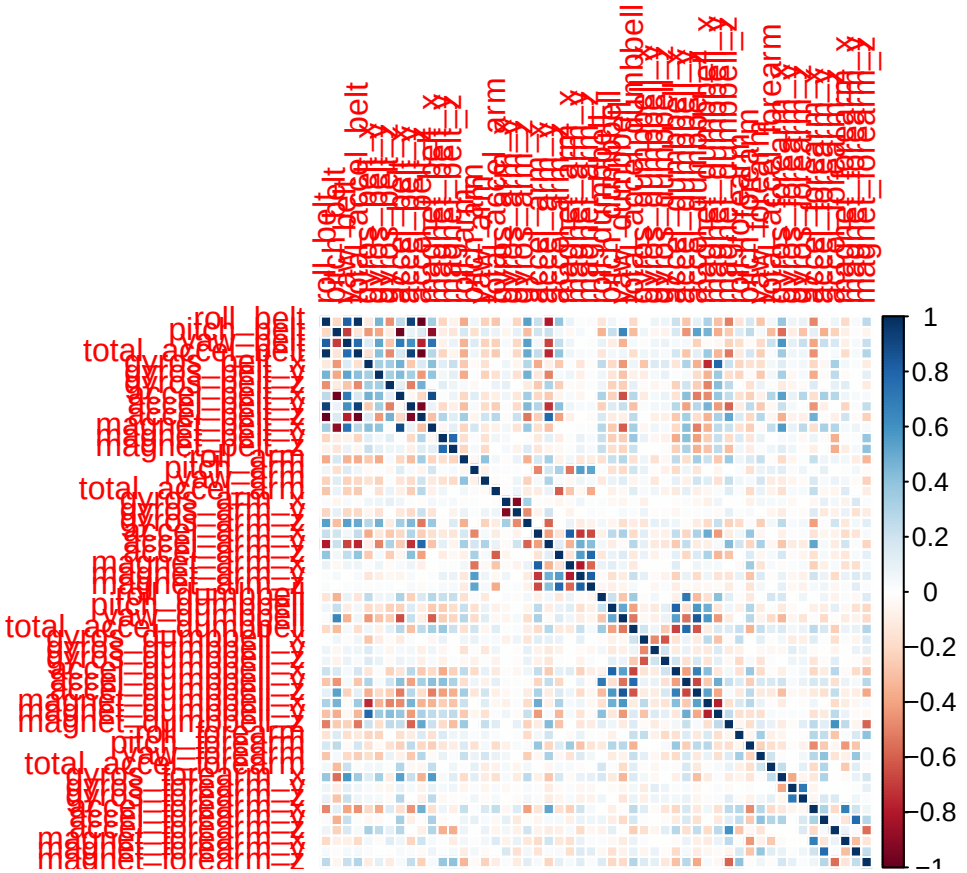
```
result <- predict(modelRf, test.cases.df[, -length(names(test.cases.df))])
result
```

```
##  [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

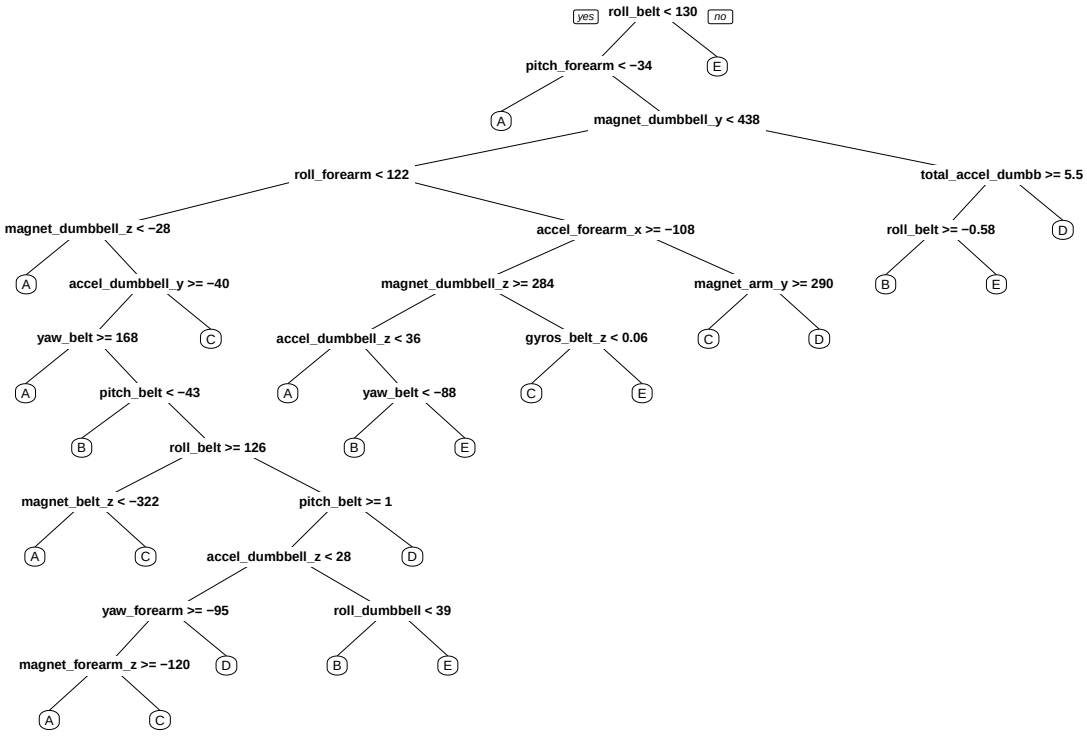
1. Correlation Matrix Visualization

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



2. Decision Tree Visualization

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



Coursera provided code for submission

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