Machine Learning - Course Project

Hoang Anh Tuan

Sunday, September 27, 2015

Introduction

This article is a report for the course project of Practical Machine Learning course on Coursera.

The purpose of this course project is to:

- Build a prediction model to predict “classe” variable within Weight Lifting Exercise Dataset
- Report cross validation results
- Report the expected out of sample error
- Apply the prediction model to predict 20 different test cases

Weight Lifting Exercise Dataset

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).

Preprocessing

Data loading

```
#install.packages("randomForest")
#install.packages("rpart")
setwd("D:/Projects/training/DataScience Specilization Certificate/08.MachineLearning/CourseProject")
training <- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!"))
validation <- read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!"))
varNames <- names(training)
commonVars <- "X|user_name|raw_timestamp_part_1|raw_timestamp_part_2|cvtd_timestamp|new_window|classe"
```

According to observation, common variables include:

- X: measurement order
- user_name: User Names (there are 6 users)
- raw_timestamp_part_1: timestamp part 1
- raw_timestamp_part_2: timestamp part 2
- cvtd_timestamp: Formatted Datetime

- new_window: Mark the last measurement of an activity, when this variable is set to “yes”, summarized varialbes (mentioned below) will be included
- num_window: Number of Windows
- classe: Kind of exercise

The remaining variables are measurements; There are 2 types of measurements:

- Instance measurements

```
commonVars <- grep("X|user_name|raw_timestamp_part_1|raw_timestamp_part_2|cvtd_timestamp|new_window|num_
measurementNames <- varNames[-commonVars]
sumVars <- grep("amplitude_.*|avg_.*|kurtosis_.*|max_.*|min_.*|skewness_.*|stddev_.*|var_.*", measuremen
sort(measurementNames[-sumVars])
```

```
##  [1] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
##  [4] "accel_belt_x"         "accel_belt_y"         "accel_belt_z"
##  [7] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [10] "accel_forearm_x"      "accel_forearm_y"      "accel_forearm_z"
## [13] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [16] "gyros_belt_x"         "gyros_belt_y"         "gyros_belt_z"
## [19] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [22] "gyros_forearm_x"      "gyros_forearm_y"      "gyros_forearm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [28] "magnet_belt_x"        "magnet_belt_y"        "magnet_belt_z"
## [31] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [34] "magnet_forearm_x"     "magnet_forearm_y"     "magnet_forearm_z"
## [37] "pitch_arm"            "pitch_belt"           "pitch_dumbbell"
## [40] "pitch_forearm"        "roll_arm"             "roll_belt"
## [43] "roll_dumbbell"        "roll_forearm"         "total_accel_arm"
## [46] "total_accel_belt"     "total_accel_dumbbell" "total_accel_forearm"
## [49] "yaw_arm"              "yaw_belt"             "yaw_dumbbell"
## [52] "yaw_forearm"
```

- Summarized measurements: is included in data set only when new_window = “yes”

```
sort(measurementNames[sumVars])
```

```
##   [1] "amplitude_pitch_arm"      "amplitude_pitch_belt"
##   [3] "amplitude_pitch_dumbbell" "amplitude_pitch_forearm"
##   [5] "amplitude_roll_arm"       "amplitude_roll_belt"
##   [7] "amplitude_roll_dumbbell"  "amplitude_roll_forearm"
##   [9] "amplitude_yaw_arm"        "amplitude_yaw_belt"
##  [11] "amplitude_yaw_dumbbell"   "amplitude_yaw_forearm"
##  [13] "avg_pitch_arm"            "avg_pitch_belt"
##  [15] "avg_pitch_dumbbell"       "avg_pitch_forearm"
##  [17] "avg_roll_arm"             "avg_roll_belt"
##  [19] "avg_roll_dumbbell"        "avg_roll_forearm"
##  [21] "avg_yaw_arm"              "avg_yaw_belt"
##  [23] "avg_yaw_dumbbell"         "avg_yaw_forearm"
##  [25] "kurtosis_picth_arm"       "kurtosis_picth_belt"
##  [27] "kurtosis_picth_dumbbell"  "kurtosis_picth_forearm"
##  [29] "kurtosis_roll_arm"        "kurtosis_roll_belt"
##  [31] "kurtosis_roll_dumbbell"   "kurtosis_roll_forearm"
##  [33] "kurtosis_yaw_arm"         "kurtosis_yaw_belt"
##  [35] "kurtosis_yaw_dumbbell"    "kurtosis_yaw_forearm"
##  [37] "max_picth_arm"            "max_picth_belt"
##  [39] "max_picth_dumbbell"       "max_picth_forearm"
##  [41] "max_roll_arm"             "max_roll_belt"
##  [43] "max_roll_dumbbell"        "max_roll_forearm"
##  [45] "max_yaw_arm"              "max_yaw_belt"
##  [47] "max_yaw_dumbbell"         "max_yaw_forearm"
##  [49] "min_pitch_arm"            "min_pitch_belt"
##  [51] "min_pitch_dumbbell"       "min_pitch_forearm"
##  [53] "min_roll_arm"             "min_roll_belt"
##  [55] "min_roll_dumbbell"        "min_roll_forearm"
##  [57] "min_yaw_arm"              "min_yaw_belt"
##  [59] "min_yaw_dumbbell"         "min_yaw_forearm"
##  [61] "skewness_pitch_arm"       "skewness_pitch_dumbbell"
##  [63] "skewness_pitch_forearm"   "skewness_roll_arm"
##  [65] "skewness_roll_belt"       "skewness_roll_belt.1"
##  [67] "skewness_roll_dumbbell"   "skewness_roll_forearm"
##  [69] "skewness_yaw_arm"         "skewness_yaw_belt"
##  [71] "skewness_yaw_dumbbell"    "skewness_yaw_forearm"
##  [73] "stddev_pitch_arm"         "stddev_pitch_belt"
##  [75] "stddev_pitch_dumbbell"    "stddev_pitch_forearm"
##  [77] "stddev_roll_arm"          "stddev_roll_belt"
##  [79] "stddev_roll_dumbbell"     "stddev_roll_forearm"
##  [81] "stddev_yaw_arm"           "stddev_yaw_belt"
##  [83] "stddev_yaw_dumbbell"      "stddev_yaw_forearm"
##  [85] "var_accel_arm"            "var_accel_dumbbell"
##  [87] "var_accel_forearm"        "var_pitch_arm"
##  [89] "var_pitch_belt"           "var_pitch_dumbbell"
##  [91] "var_pitch_forearm"        "var_roll_arm"
##  [93] "var_roll_belt"            "var_roll_dumbbell"
##  [95] "var_roll_forearm"         "var_total_accel_belt"
##  [97] "var_yaw_arm"              "var_yaw_belt"
##  [99] "var_yaw_dumbbell"         "var_yaw_forearm"
```

Because the test set does not contain summarized measurements, therefore we cannot include summarized measurements in buildig prediction model

```
training <- training[,c("classe", "user_name", measurementNames[-sumVars])]
#training <- training[,c("classe", measurementNames[-sumVars])]
#Remove any column containing at least one NA value
training <- training[,colSums(is.na(training))<1]
training$classe <- factor(training$classe)
dim(training)
```

```
## [1] 19622    54
```

Build prediction model

Split training data set into training & testing sets

```
library(caret)
```

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

```
set.seed(543)
inTrain <- createDataPartition(y=training$classe, p=0.8, list=FALSE)
testing <- training[-inTrain,]
dim(testing)
```

```
## [1] 3923   54
```

```
training <- training[inTrain,]
dim(training)
```

```
## [1] 15699    54
```

Prediction Tree

Model training

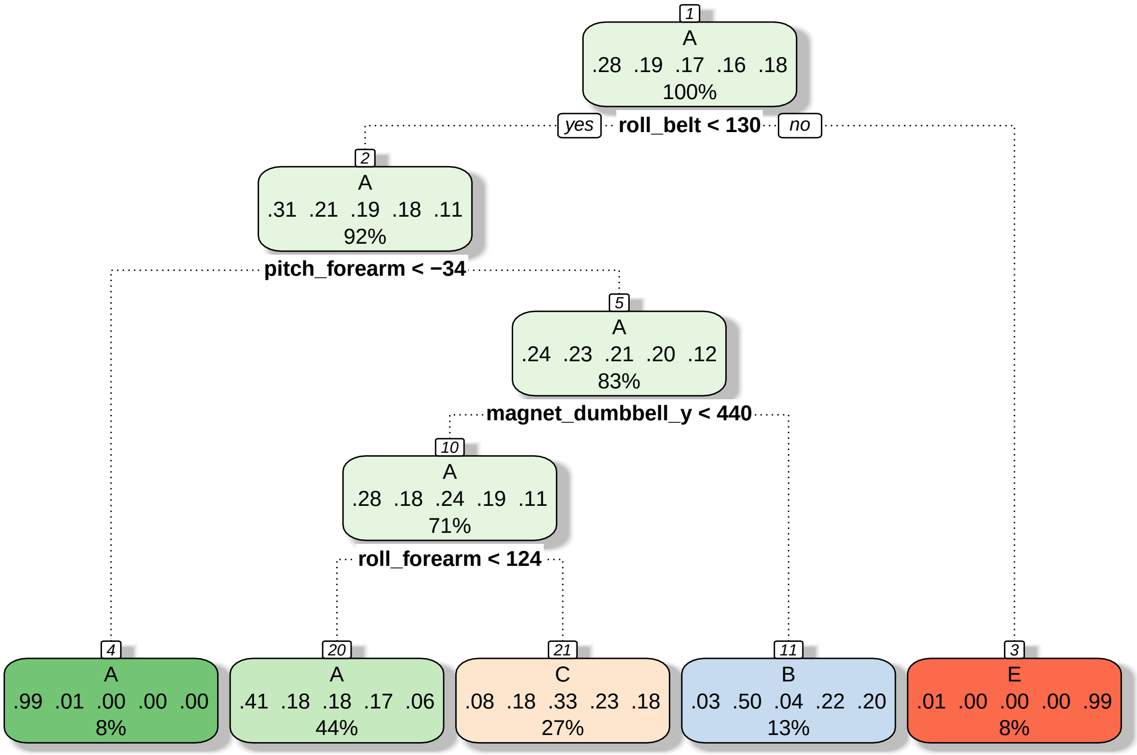
```
treeModel <- train(classe ~ ., data=training, method="rpart")
```

```
## Loading required package: rpart
```

```
library(rattle)
```

```
## Loading required package: RGtk2
## Rattle: A free graphical interface for data mining with R.
## Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
fancyRpartPlot(treeModel$finalModel)
```



Rattle 2015−Sep−28 06:23:12 tuanhoang

Model Validation

```
trainingPredict <- predict(treeModel, training)
print(confusionMatrix(trainingPredict, training$classe))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 4059 1272 1254 1150  419
##          B   68 1011   86  453  396
##          C  324  755 1398  970  765
##          D    0    0    0    0    0
##          E   13    0    0    0 1306
## 
## Overall Statistics
## 
##                Accuracy : 0.4952
##                  95% CI : (0.4873, 0.503)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.3403
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
```

```
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9093   0.3328  0.51059   0.0000  0.45253
## Specificity            0.6355   0.9208  0.78289   1.0000  0.99899
## Pos Pred Value         0.4978   0.5020  0.33191      NaN  0.99014
## Neg Pred Value         0.9463   0.8519  0.88335   0.8361  0.89013
## Prevalence             0.2843   0.1935  0.17441   0.1639  0.18383
## Detection Rate         0.2586   0.0644  0.08905   0.0000  0.08319
## Detection Prevalence   0.5194   0.1283  0.26830   0.0000  0.08402
## Balanced Accuracy      0.7724   0.6268  0.64674   0.5000  0.72576
```

Cross validation

```
testingPredict <- predict(treeModel, testing)
print(confusionMatrix(testingPredict, testing$classe))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1021  309  333  299  105
##          B   13  275   22  115   90
##          C   81  175  329  229  201
##          D    0    0    0    0    0
##          E    1    0    0    0  325
## 
## Overall Statistics
## 
##                Accuracy : 0.4971
##                  95% CI : (0.4813, 0.5128)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.342
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9149   0.3623  0.48099   0.0000  0.45076
## Specificity            0.6274   0.9241  0.78821   1.0000  0.99969
## Pos Pred Value         0.4940   0.5340  0.32414      NaN  0.99693
## Neg Pred Value         0.9488   0.8580  0.87792   0.8361  0.88991
## Prevalence             0.2845   0.1935  0.17436   0.1639  0.18379
## Detection Rate         0.2603   0.0701  0.08386   0.0000  0.08284
## Detection Prevalence   0.5269   0.1313  0.25873   0.0000  0.08310
## Balanced Accuracy      0.7711   0.6432  0.63460   0.5000  0.72523
```

Cross validation accuracy of prediction tree model is 49.71% meaning error rate is 50.29% which is too high.

Random Forest Model

Model training

```r
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```r
#rfModel <- train(classe ~ ., data=training, method="rf")
rfModel <- randomForest(classe ~ ., data = training, importance = TRUE, ntrees = 10)
summary(rfModel)
```

```
##                 Length Class  Mode
## call                5  -none- call
## type                1  -none- character
## predicted       15699  factor numeric
## err.rate         3000  -none- numeric
## confusion          30  -none- numeric
## votes           78495  matrix numeric
## oob.times       15699  -none- numeric
## classes             5  -none- character
## importance        371  -none- numeric
## importanceSD      318  -none- numeric
## localImportance     0  -none- NULL
## proximity           0  -none- NULL
## ntree               1  -none- numeric
## mtry                1  -none- numeric
## forest             14  -none- list
## y               15699  factor numeric
## test                0  -none- NULL
## inbag               0  -none- NULL
## terms               3  terms  call
```

Model Validation

```r
trainingPredict <- predict(rfModel, training)
print(confusionMatrix(trainingPredict, training$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4464    0    0    0    0
##          B    0 3038    0    0    0
##          C    0    0 2738    0    0
##          D    0    0    0 2573    0
##          E    0    0    0    0 2886
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Cross validation

```
testingPredict <- predict(rfModel, testing)
print(confusionMatrix(testingPredict, testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    4    0    0    0
##          B    0  754    1    0    0
##          C    0    1  683    8    0
##          D    0    0    0  635    4
##          E    0    0    0    0  717
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9928, 0.9973)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9934   0.9985   0.9876   0.9945
## Specificity            0.9986   0.9997   0.9972   0.9988   1.0000
## Pos Pred Value         0.9964   0.9987   0.9870   0.9937   1.0000
## Neg Pred Value         1.0000   0.9984   0.9997   0.9976   0.9988
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1922   0.1741   0.1619   0.1828
## Detection Prevalence   0.2855   0.1925   0.1764   0.1629   0.1828
## Balanced Accuracy      0.9993   0.9965   0.9979   0.9932   0.9972
```

Cross validation accuracy of prediction tree model is 99.54% meaning error rate is 0.46% which is quite good.

Out of sample error

The highest accuracy in this report is based on prediction algorithm of Random Forest Model, its out of sample error is estimated as 0.46%.

Prediction for 20 test cases

```r
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)
  }
}
output <- predict(rfModel, validation)
pml_write_files(output)
print(output)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 
##  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
```