Practical Machine Learning Assignment

Kristine Loh

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, we will use data recorded from accelerometers on the belt, forearm, arm, and dumbbell 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 Set

```
download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
              destfile = "./pml-training.csv", method = "curl")
```

```
training <- read.csv("./pml-training.csv", na.strings=c("NA","#DIV/0!",""))
```

```
download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
              destfile = "./pml-testing.csv", method = "curl")
```

```
testing <- read.csv("./pml-testing.csv", na.strings=c("NA","#DIV/0!",""))
```

Look at data

```
str(training, list.len=10)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 4844
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##   [list output truncated]
```

```
table(training$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```r
prop.table(table(training$user_name, training$classe), 1)
```

```
##
##                   A         B         C         D         E
##   adelmo   0.2993320 0.1993834 0.1927030 0.1323227 0.1762590
##   carlitos 0.2679949 0.2217224 0.1584190 0.1561697 0.1956941
##   charles  0.2542421 0.2106900 0.1524321 0.1815611 0.2010747
##   eurico   0.2817590 0.1928339 0.1592834 0.1895765 0.1765472
##   jeremy   0.3459730 0.1437390 0.1916520 0.1534392 0.1651969
##   pedro    0.2452107 0.1934866 0.1911877 0.1796935 0.1904215
```

```r
prop.table(table(training$classe))
```

```
##
##         A         B         C         D         E
## 0.2843747 0.1935073 0.1743961 0.1638977 0.1838243
```

Cleaning the Data

Remove columns 1 to 6 as they are for information:

```r
training <- training[, 7:160]
```

```r
testing  <- testing[, 7:160]
```

Remove NA

```r
clean_data  <- apply(!is.na(training), 2, sum) > 19621
```

```r
training <- training[, clean_data]
```

```r
testing  <- testing[, clean_data]
```

Subsample 60% of the set for training purposes, while the 40% remainder will be used for testing

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

```
set.seed(12345)
inTrain <- createDataPartition(y=training$classe, p=0.60, list=FALSE)
trainset1  <- training[inTrain,]
trainset2  <- training[-inTrain,]
dim(trainset1)
```

```
## [1] 11776    54
```

```
dim(trainset2)
```

```
## [1] 7846   54
```

Identify the ???zero covariates???" from trainset1 and remove these ???zero covariates???" from both trainset1 and trainset2

```
nzv_cols <- nearZeroVar(trainset1)
if(length(nzv_cols) > 0) {
  trainset1 <- trainset1[, -nzv_cols]
  trainset2 <- trainset2[, -nzv_cols]
}
dim(trainset1)
```

```
## [1] 11776    54
```

```
dim(trainset2)
```

```
## [1] 7846   54
```

Data Manipulation

Building Decision Tree Model

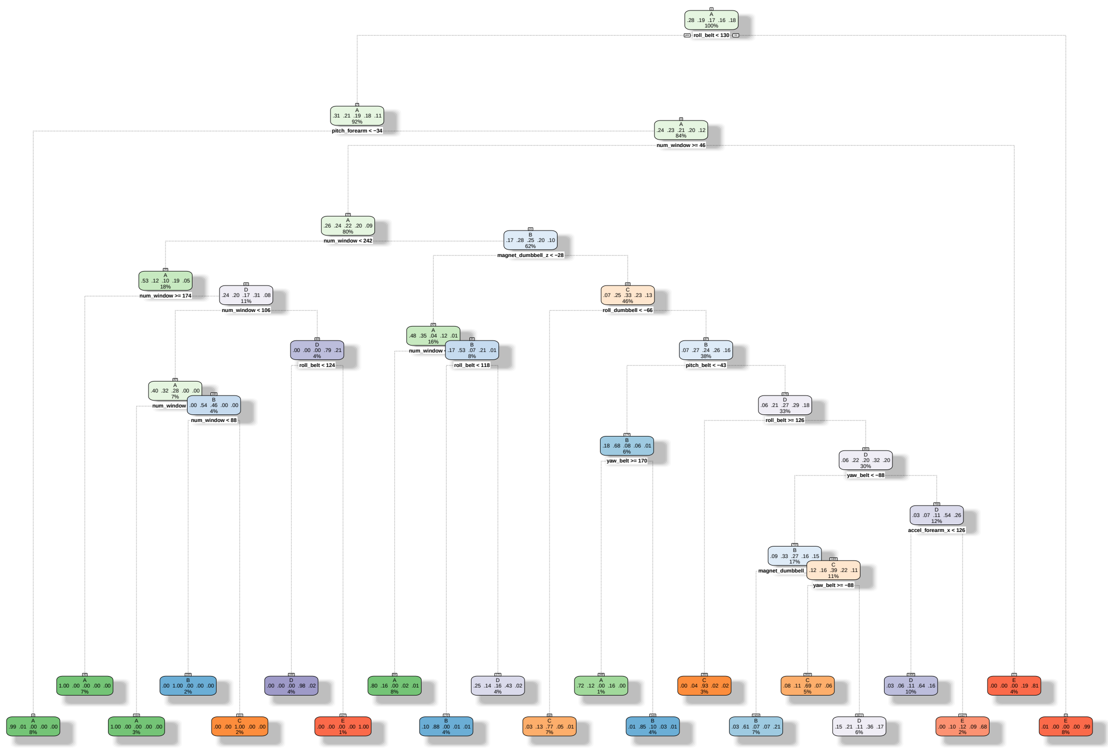
```
library(rpart)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.3 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(caret)
```

```
modFitDT <- rpart(classe ~ ., data = trainset1, method="class")
fancyRpartPlot(modFitDT)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2016−Apr−15 11:47:08 kristineloh

###Predicting with Decision Tree

```r
set.seed(12345)

prediction <- predict(modFitDT, trainset2, type = "class")
confusionMatrix(prediction, trainset2$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1961  121    1   33    8
##          B   57 1067   43   41  130
##          C   54  152 1098   63   21
##          D  152  160  193 1061  226
##          E    8   18   33   88 1057
##
## Overall Statistics
##
##                Accuracy : 0.7958
##                  95% CI : (0.7867, 0.8047)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7427
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8786   0.7029   0.8026   0.8250   0.7330
## Specificity            0.9710   0.9572   0.9552   0.8886   0.9770
## Pos Pred Value         0.9233   0.7975   0.7911   0.5921   0.8779
## Neg Pred Value         0.9526   0.9307   0.9582   0.9628   0.9420
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2499   0.1360   0.1399   0.1352   0.1347
## Detection Prevalence   0.2707   0.1705   0.1769   0.2284   0.1535
## Balanced Accuracy      0.9248   0.8300   0.8789   0.8568   0.8550
```

The accuracy is 0.8

Building the Random Forest Model

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(12345)
modFitRF <- randomForest(classe ~ ., data = trainset1, ntree = 1000)
```

The accuracy is 0.99

Predicting with Random Forest Model

```
prediction <- predict(modFitRF, trainset2, type = "class")
confusionMatrix(prediction, trainset2$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    4    0    0    0
##          B    0 1511    4    0    0
##          C    0    3 1364   14    0
##          D    0    0    0 1271    3
##          E    0    0    0    1 1439
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9963
##                  95% CI : (0.9947, 0.9975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9953
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9954   0.9971   0.9883   0.9979
## Specificity            0.9993   0.9994   0.9974   0.9995   0.9998
## Pos Pred Value         0.9982   0.9974   0.9877   0.9976   0.9993
## Neg Pred Value         1.0000   0.9989   0.9994   0.9977   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1926   0.1738   0.1620   0.1834
## Detection Prevalence   0.2850   0.1931   0.1760   0.1624   0.1835
## Balanced Accuracy      0.9996   0.9974   0.9972   0.9939   0.9989
```

Out of sample error rate

```
missClass = function(values, predicted) {
  sum(predicted != values) / length(values)
}
OOS_errRate = missClass(trainset2$classe, prediction)
OOS_errRate
```

```
## [1] 0.003696151
```

Predicting on Testing Data

Predicting with Decision Tree

```
predictionDT <- predict(modFitDT,testing, type = "class")
predictionDT
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  D  A  A  C  D  D  A  A  D  C  B  A  D  E  A  A  B  B
## Levels: A B C D E
```

Predicting with Random Forest

```
predictionRF <- predict(modFitRF, testing, type = "class")
predictionRF
```

```
##  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
```

Submission

```
write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_",i,".txt")
    write.table(x[i], file=filename, quote=FALSE, row.names=FALSE, col.names=FALSE)
  }
}
write_files(predictionRF)
```