Machine Learning Assignment

Geons

March 2016

Synopsis

Background

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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Objective

In this project data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants is used to evaluate the correctness of exercise execution. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The analysis aims to accurately predict the manner in which the participants did the exercise.

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Data Processing

Environment setup

```
# I like to clean up before I start / tabula rasa (optional)
rm(list=ls()) # this deletes all your env variables ... be very carefull!

# Always make code visible
echo = TRUE

# Turn off scientific notations for numbers
options(scipen = 1)
options(digits=3)

# Load needed libraries - can be done later as well
library(caret)
```

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

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

```
library(corrplot)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
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:dplyr':
##
##     combine
```

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

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

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
# check the workdir and reset if needed
getwd()
```

```
## [1] "D:/005 Ouns/Statistics/Coursera/PML"
```

```
setwd("d:/005 Ouns/Statistics/Coursera/PML")
```

Data Loading

We load the data into R

```
# Specify URL's
workURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# Load the data
work_DF <- read.csv(file=workURL, na.strings = c("NA", ""), stringsAsFactors=FALSE)
test_DF <- read.csv(file=testURL, na.strings = c("NA", ""), stringsAsFactors=FALSE)
```

Pre-Processing

Data exploration indicates that significant amounts of data are missing, so it is wise to choose to eliminate the variables with missing data from the dataframes. Furthermore the “X” variable and “user_name” variable (used exclusively for tracking purposes) are removed along with the time stamp and window variables (which show no relation to the classe variable. The dependent (“classe”) variable is changed to a factor variable.

```
# removing variables with missing values
work_DF <- work_DF[colSums(is.na(work_DF)) == 0]
test_DF <- test_DF[colSums(is.na(test_DF)) == 0]

# remove username, X, time stamp and window variables
work_DF <- select(work_DF, -contains("timestamp"), -ends_with("window"), -starts_with("user"), -X)
test_DF <- select(test_DF, -contains("timestamp"), -ends_with("window"), -starts_with("user"), -X)

# set "classe" to a factor variable
work_DF$classe <- as.factor(work_DF$classe)
```

The dataset is now split into a 60% training and 40% testing dataset.

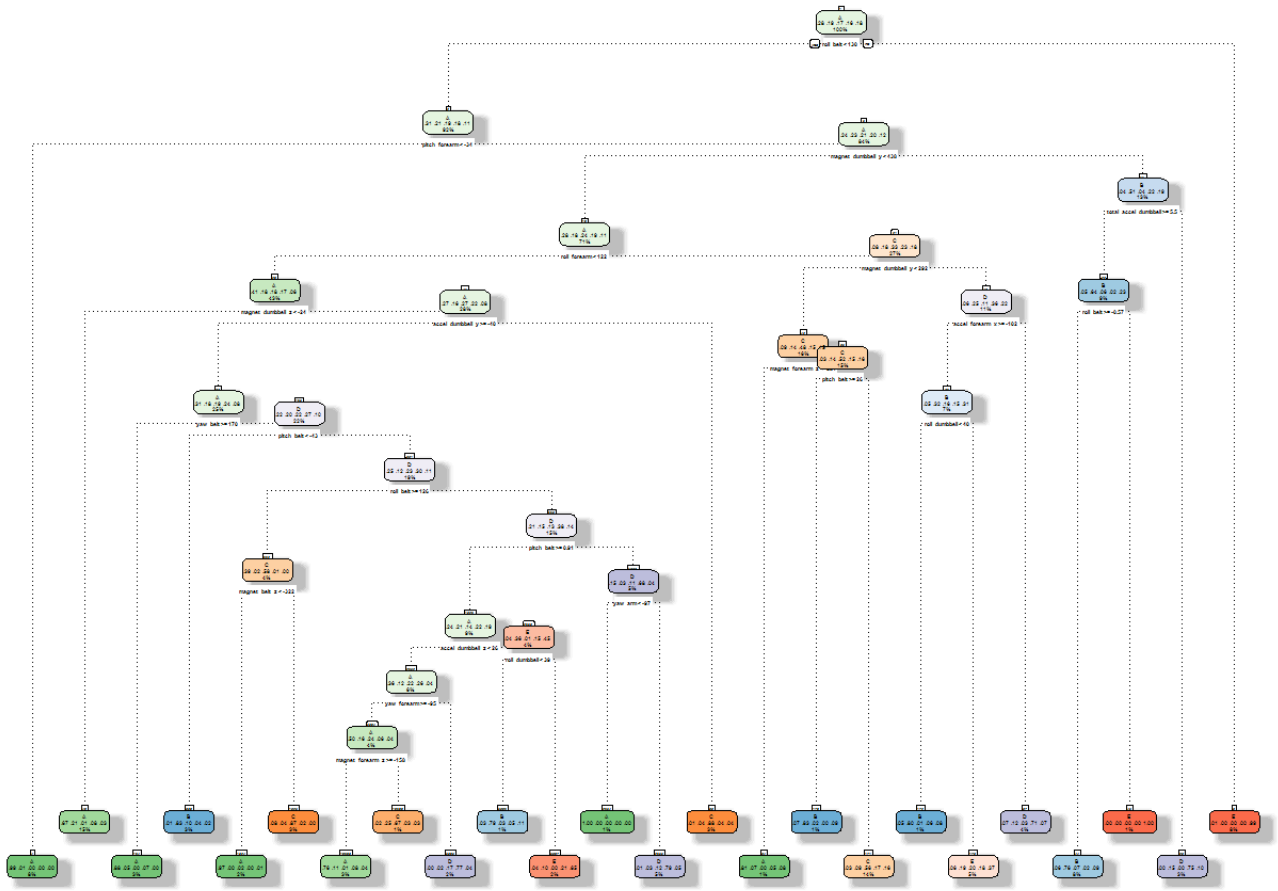
```
seed <- as.numeric(as.Date("2016-03-12"))
set.seed(seed)
InternalTrain <- createDataPartition(work_DF$classe, p=0.6, list = FALSE)
subTrain <- work_DF[InternalTrain,]
subProbe <- work_DF[-InternalTrain,]
```

Building the Decision Tree Model

Using Decision Tree, we shouldn’t expect the accuracy to be high. In fact, anything around 80% would be acceptable.

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

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



Rattle 2016-mrt-14 01:07:18 Family

Predicting with the Decision Tree Model

```
set.seed(seed)

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2048  300   45  136   53
##          B   63  836   72   33   85
##          C   61  157 1086  205  163
##          D   24  106   77  802   77
##          E   36  119   88  110 1064
##
## Overall Statistics
##
##                Accuracy : 0.744
##                  95% CI : (0.734, 0.753)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.674
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.918    0.551    0.794    0.624    0.738
## Specificity             0.905    0.960    0.910    0.957    0.945
## Pos Pred Value          0.793    0.768    0.650    0.738    0.751
## Neg Pred Value          0.965    0.899    0.954    0.928    0.941
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.261    0.107    0.138    0.102    0.136
## Detection Prevalence    0.329    0.139    0.213    0.138    0.181
## Balanced Accuracy       0.911    0.755    0.852    0.790    0.841
```

Building the Random Forest Model

Using random forest, the out of sample error should be small. The error will be estimated using the 40% testing sample. We should expect an error estimate of < 3%.

```
set.seed(seed)
modFitRF <- randomForest(classe ~ ., data = subTrain, ntree = 500)
```

Predicting with the Random Forest Model

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    8    0    0    0
##          B    2 1507    7    0    0
##          C    0    3 1358   19    0
##          D    0    0    3 1267    6
##          E    0    0    0    0 1436
##
## Overall Statistics
##
##                Accuracy : 0.994
##                  95% CI : (0.992, 0.995)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.992
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.993    0.993    0.985    0.996
## Specificity             0.999    0.999    0.997    0.999    1.000
## Pos Pred Value          0.996    0.994    0.984    0.993    1.000
## Neg Pred Value          1.000    0.998    0.998    0.997    0.999
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.173    0.161    0.183
## Detection Prevalence    0.285    0.193    0.176    0.163    0.183
## Balanced Accuracy       0.999    0.996    0.995    0.992    0.998
```

Predicting on the Testing Data (pml-testing.csv) using the Random Forest Prediction

```
predictionRF <- predict(modFitRF, test_DF, 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
```

Conclusion

As can be seen from the confusion matrix the Random Forest model reaches an accuracy of more than 99%. The accuracy rate of 99.4% is good and gives us an estimated out-of-sample error of 0.60%!