Practical Machine Learning Project

Irene Teo

21 March, 2015

Executive Summary

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.

The aim of this report was to use data from accelerometers placed on the belt, forearm, arm, and dumbell of six participants to predict the manner in which they did the exercise.

Data Loading and Preprocessing

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.2
```

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

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

```
## Warning: package 'rpart.plot' was built under R version 3.1.2
```

```
library(RColorBrewer)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.1.2
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

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

```
set.seed(12345)
```

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# load data to memory
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

Partition the training data set into two data sets: 60% for myTraining, 40% for myTesting.

```
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]; myTesting <- training[-inTrain, ]
dim(myTraining); dim(myTesting)
```

```
## [1] 11776   160
```

```
## [1] 7846  160
```

Data Cleaning

View possible NearZeroVariance Variables

```
myDataNZV <- nearZeroVar(myTraining, saveMetrics=TRUE)
```

Create subset of data without NZV variables.

```
myNZVvars <- names(myTraining) %in% c("new_window", "kurtosis_roll_belt", "kurtosis_pict
h_belt", "kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1", "skewness_ya
w_belt", "max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_arm", "stddev_r
oll_arm", "var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_arm", "avg_yaw
_arm", "stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm", "kurtosis_picth_arm", "kurt
osis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm", "skewness_yaw_arm", "max_roll_
arm", "min_roll_arm", "min_pitch_arm", "amplitude_roll_arm", "amplitude_pitch_arm", "kur
tosis_roll_dumbbell", "kurtosis_picth_dumbbell", "kurtosis_yaw_dumbbell", "skewness_roll
_dumbbell", "skewness_pitch_dumbbell", "skewness_yaw_dumbbell", "max_yaw_dumbbell", "min
_yaw_dumbbell", "amplitude_yaw_dumbbell", "kurtosis_roll_forearm", "kurtosis_picth_forea
rm", "kurtosis_yaw_forearm", "skewness_roll_forearm", "skewness_pitch_forearm", "skewnes
s_yaw_forearm", "max_roll_forearm", "max_yaw_forearm", "min_roll_forearm", "min_yaw_fore
arm", "amplitude_roll_forearm", "amplitude_yaw_forearm", "avg_roll_forearm", "stddev_rol
l_forearm", "var_roll_forearm", "avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_
forearm", "avg_yaw_forearm", "stddev_yaw_forearm", "var_yaw_forearm")
myTraining <- myTraining[!myNZVvars]
dim(myTraining)
```

```
## [1] 11776   100
```

Remove ID to avoid interference with ML algorithms

```
myTraining <- myTraining[c(-1)]
```

Clean variables with too many NA

```
trainingV3 <- myTraining #creating another subset to iterate in loop
for(i in 1:length(myTraining)) { #for every column in the training dataset
        if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .6 ) { #if n?? NAs > 60
% of total observations
        for(j in 1:length(trainingV3)) {
            if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) ==1)  { #if t
he columns are the same:
                trainingV3 <- trainingV3[ , -j] #Remove that column
            }
        }
    }
}

dim(trainingV3)
```

```
## [1] 11776    58
```

```
myTraining <- trainingV3
rm(trainingV3)
```

Repeat cleaning process for ‘myTesting’ and ‘testing’ data sets.

```
clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58]) #already with classe column removed
myTesting <- myTesting[clean1]
testing <- testing[clean2]

dim(myTesting)
```

```
## [1] 7846   58
```

```
dim(testing)
```

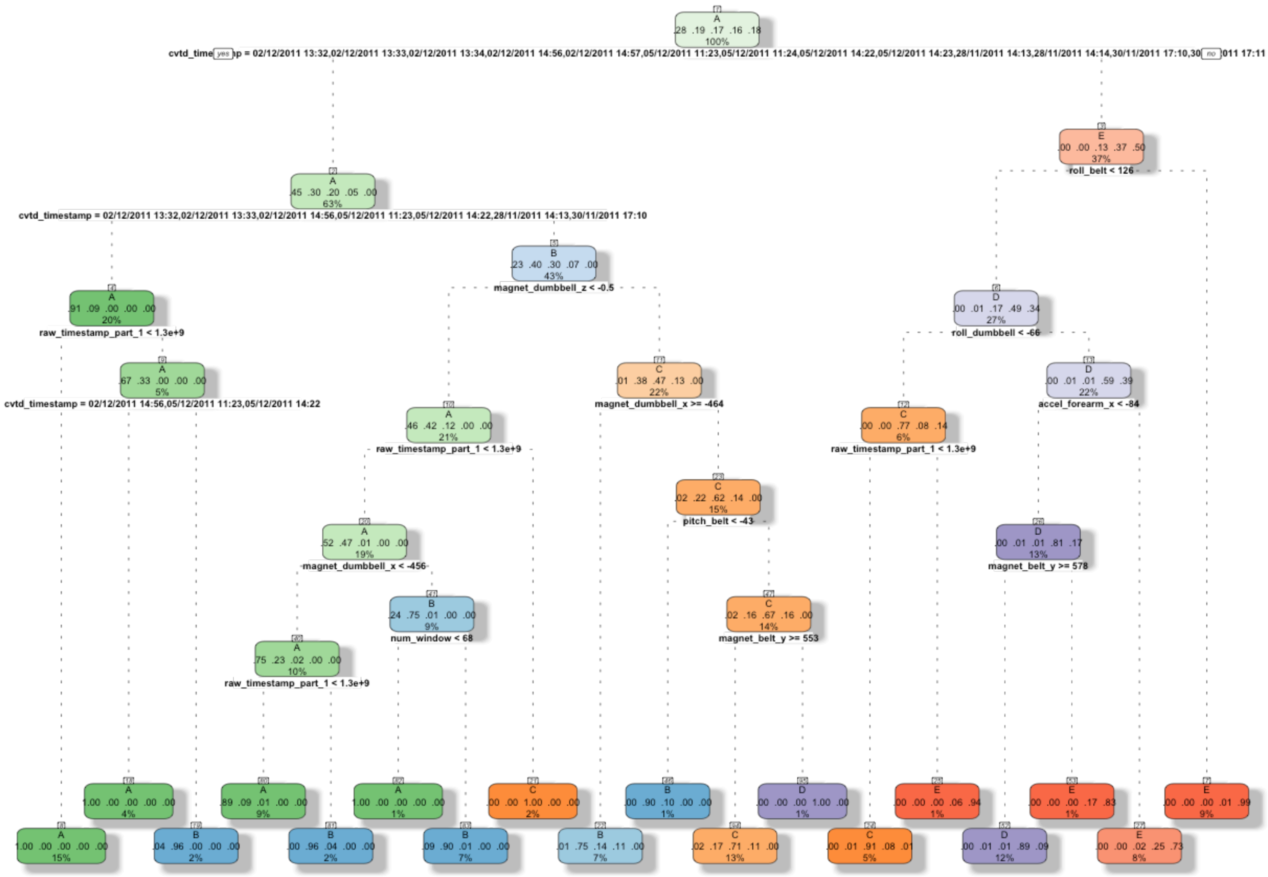
```
## [1] 20 57
```

Coerce data into same type

```
for (i in 1:length(testing) ) {
        for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) ==1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}
#And to make sure Coertion really worked, simple smart ass technique:
testing <- rbind(myTraining[2, -58] , testing) #note row 2 does not mean anything, this
will be removed right.. now:
testing <- testing[-1,]
```

Prediction using ML algorithms, Decision Tree

```
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modFitA1)
```



Rattle 2015-Mar-21 16:36:46 ireneteo

```
predictionsA1 <- predict(modFitA1, myTesting, type = "class")
confusionMatrix(predictionsA1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2150   60    7    1    0
##          B   61 1260   69   64    0
##          C   21  188 1269  143    4
##          D    0   10   14  857   78
##          E    0    0    9  221 1360
##
## Overall Statistics
##
##                Accuracy : 0.8789
##                  95% CI : (0.8715, 0.8861)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8468
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9633   0.8300   0.9276   0.6664   0.9431
## Specificity            0.9879   0.9693   0.9450   0.9845   0.9641
## Pos Pred Value         0.9693   0.8666   0.7809   0.8936   0.8553
## Neg Pred Value         0.9854   0.9596   0.9841   0.9377   0.9869
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2740   0.1606   0.1617   0.1092   0.1733
## Detection Prevalence   0.2827   0.1853   0.2071   0.1222   0.2027
## Balanced Accuracy      0.9756   0.8997   0.9363   0.8254   0.9536
```

Prediction using ML algorithms, Random Forest

```
modFitB1 <- randomForest(classe ~. , data=myTraining)
predictionsB1 <- predict(modFitB1, myTesting, type = "class")
confusionMatrix(predictionsB1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    2    0    0    0
##          B    1 1516    2    0    0
##          C    0    0 1366    3    0
##          D    0    0    0 1282    2
##          E    0    0    0    1 1440
##
## Overall Statistics
##
##                Accuracy : 0.9986
##                  95% CI : (0.9975, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9982
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9987   0.9985   0.9969   0.9986
## Specificity            0.9996   0.9995   0.9995   0.9997   0.9998
## Pos Pred Value         0.9991   0.9980   0.9978   0.9984   0.9993
## Neg Pred Value         0.9998   0.9997   0.9997   0.9994   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1741   0.1634   0.1835
## Detection Prevalence   0.2846   0.1936   0.1745   0.1637   0.1837
## Balanced Accuracy      0.9996   0.9991   0.9990   0.9983   0.9992
```

Random Forest produced better results

File Generation for Submission of Assignment

We use the following formula for Random Forests, which produced a better prediction in in-sample:

```r
predictionsB2 <- predict(modFitB1, testing, type = "class")

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(predictionsB2)
```