Project of Practical Machine Learning

Introdution

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).

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

1. Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).
2. You should also apply your machine learning algorithm to the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

Results

```r
library(caret)
```

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

Getting data. Take the data of the work directory.

```r
getwd()
```

```
## [1] "C:/Users/Itahisa/Documents/Practical Machine Learning"
```

```r
Training <- read.csv("~/Practical Machine Learning/pml-training.csv", header=TRUE)
View(Training)

Testing <- read.csv("~/Practical Machine Learning/pml-testing.csv", header=TRUE)
View(Testing)
```

Now, we have to partion the set into two data sets.

```r
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
```

We cannot continue without cleaning the data.

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

myNZVvars <- names(myTraining) %in% c("new_window", "kurtosis_roll_belt", "kurtosis_picth_belt",
"kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1", "skewness_yaw_belt",
"max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_arm", "stddev_roll_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",
"kurtosis_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",
"kurtosis_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_forearm", "kurtosis_yaw_forearm",
"skewness_roll_forearm", "skewness_pitch_forearm", "skewness_yaw_forearm", "max_roll_forearm",
"max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm", "amplitude_roll_forearm",
"amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_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
```

```r
# Remove the first column of Dataset with the ID variable

myTraining <- myTraining[c(-1)]

# There're to many NAs, so we have to remove it.

trainingV3 <- myTraining
for(i in 1:length(myTraining)) {
        if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .6 ) {
        for(j in 1:length(trainingV3)) {
            if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) ==1)  {
                trainingV3 <- trainingV3[ , -j] } } } }

dim(trainingV3)
```

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

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

# Repeat the cleaning for the Testing data sets

clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58])
myTesting <- myTesting[clean1]
Testing <- Testing[clean2]

dim(myTesting)
```

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

```r
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])
        }
    }
}

Testing <- rbind(myTraining[2, -58] , Testing)
Testing <- Testing[-1,]
```

Prediction Model 1

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

```
## Rattle: A free graphical interface for data mining with R.
## Versión 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

```r
Model1 <- rpart(classe ~ ., data = myTraining, method="class")

Prediction1 <- predict(Model1, myTesting, type = "class")

confusionMatrix(Prediction1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2148   61    8    2    0
##          B   62 1246   78   54    0
##          C   22  203 1260  215   62
##          D    0    8   15  815   92
##          E    0    0    7  200 1288
```

```
##
## Overall Statistics
##
##                Accuracy : 0.8612
##                  95% CI : (0.8534, 0.8688)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8244
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9624   0.8208   0.9211   0.6337   0.8932
## Specificity            0.9874   0.9693   0.9225   0.9825   0.9677
## Pos Pred Value         0.9680   0.8653   0.7151   0.8763   0.8615
## Neg Pred Value         0.9851   0.9575   0.9822   0.9319   0.9758
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2738   0.1588   0.1606   0.1039   0.1642
## Detection Prevalence   0.2828   0.1835   0.2246   0.1185   0.1905
## Balanced Accuracy      0.9749   0.8951   0.9218   0.8081   0.9304
```

Prediction Model 2

```
library(randomForest)
```

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

```
Model2 <- randomForest(classe ~. , data = myTraining)

Prediction2 <- predict(Model2, myTesting, type = "class")

confusionMatrix(Prediction2, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    2    0    0    0
##          B    0 1516    2    0    0
##          C    0    0 1366    8    0
##          D    0    0    0 1276    3
##          E    0    0    0    2 1439
##
## Overall Statistics
##
##                Accuracy : 0.9978
##                  95% CI : (0.9965, 0.9987)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9973
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9987   0.9985   0.9922   0.9979
## Specificity            0.9996   0.9997   0.9988   0.9995   0.9997
## Pos Pred Value         0.9991   0.9987   0.9942   0.9977   0.9986
## Neg Pred Value         1.0000   0.9997   0.9997   0.9985   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1932   0.1741   0.1626   0.1834
## Detection Prevalence   0.2847   0.1935   0.1751   0.1630   0.1837
## Balanced Accuracy      0.9998   0.9992   0.9987   0.9959   0.9988
```

Random Forests make better results than the first one.

The End