Human Activity Recognition

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 goal of this project is to predict the manner in which they did the exercise. The data was collected 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.

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. (see the section on the Weight Lifting Exercise Dataset).

Exploratory analysis

The data consists of

a) a training set of 19622 observations with 160 variables

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

b) a testing set of 20 observations with 160 variables

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

The outcome we want to predict is the variable “classe”, with possible values A/B/C/D/E. Please refer to the summary in Output 1.2 and 1.3

Please refer to Output 1.4 for the structure of the training data. All the data has been initially loaded as characters but most of the variables in the training data set are numeric in nature. Also a lot of variables are sparsely populated - see some examples in Output 1.5

Models

Predictive models were created as listed below. Please refer to Output 2.1 for the relevant r code.

1. The following variables are not directly related to the accelerometer readings. They were not included in the model.

   user_name

   cvtd_timestamp

   classe

   X

   raw_timestamp_part_1

   raw_timestamp_part_2

   new_window

2. From the remaining variables, all the sparsely populated variables (that have less than 50% of data) were also excluded from the model.

3. The training data was split into true training (15%) and cross-validation data sets (85%)

4. The following two methods were applied to the true training data set

   a) random forests

   b) gradient boosing

5. The two models were then used to predict the outcomes of the observations in the cross-validation data set as well as the testing data set.

6. The accuracy was measured by comparing the predicted values with the actual values in the cross-validation data set.

7. These steps were repeated a few times to get the average out of sample error Output 2.2

7. These steps were repeted a few times to get the average out of sample error Output 2.2

Conclusions & Out of sample error

The accuracy and out of sample error for the two models considered were

Random forests - Accuracy = 97.6% ; Out of sample error = 2.4%

Gradient boosing - Accuracy = 96.7% ; Out of sample error = 3.3%

The prediction accuracy of the 20 test observations was 100%

Appendix

Output 1.1 - Load the libraries and files

```
# load the libraries
library(caret)
```

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

```
library(randomForest)
```

```
## randomForest 4.6-7
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(gbm)
```

```
## Loading required package: survival
## Loading required package: splines
## 
## Attaching package: 'survival'
## 
## The following object is masked from 'package:caret':
## 
##     cluster
## 
## Loading required package: parallel
## Loaded gbm 2.1
```

```
library(survival)
library(splines)
library(plyr)

# set training sample percentage
trSamplePct = 0.15

set.seed(233)

# load training and test data files

training = read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
    colClasses = "character")
testing = read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
    colClasses = "character")
```

Output 1.2 - Summary of the variable “classe”

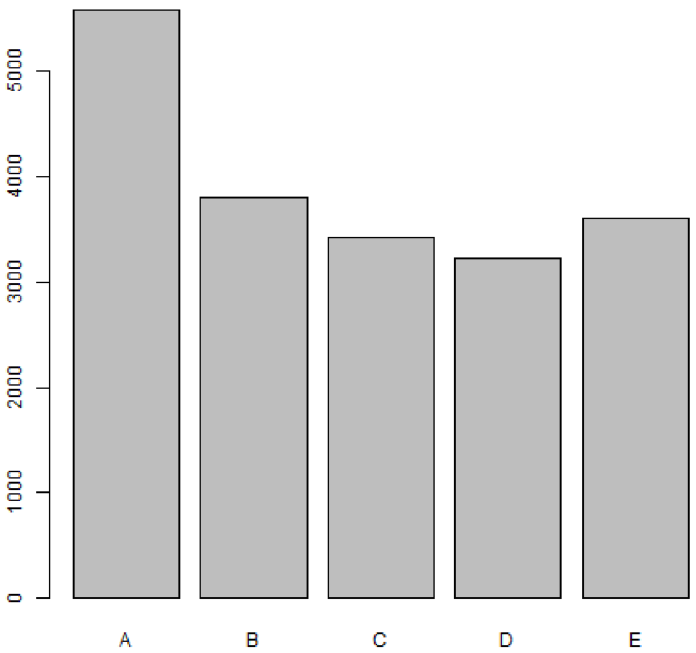
```
summary(as.factor(training$classe))
```

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

Output 1.3 - Barplot of the variable “classe”

Output 1.3 - Barplot of the variable "classe"

```
barplot(summary(as.factor(training$classe)))
```



Output 1.4 - Structure of the training data set

```
str(training)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                        : chr  "1" "2" "3" "4" ...
##  $ user_name                : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1     : chr  "1323084231" "1323084231" "1323084231" "1323084232" ...
##  $ raw_timestamp_part_2     : chr  "788290" "808298" "820366" "120339" ...
##  $ cvtd_timestamp           : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23"
"05/12/2011 11:23" ...
##  $ new_window               : chr  "no" "no" "no" "no" ...
##  $ num_window               : chr  "11" "11" "11" "12" ...
##  $ roll_belt                : chr  "1.41" "1.41" "1.42" "1.48" ...
##  $ pitch_belt               : chr  "8.07" "8.07" "8.07" "8.05" ...
##  $ yaw_belt                 : chr  "-94.4" "-94.4" "-94.4" "-94.4" ...
##  $ total_accel_belt         : chr  "3" "3" "3" "3" ...
##  $ kurtosis_roll_belt       : chr  "" "" "" "" ...
##  $ kurtosis_picth_belt      : chr  "" "" "" "" ...
##  $ kurtosis_yaw_belt        : chr  "" "" "" "" ...
##  $ skewness_roll_belt       : chr  "" "" "" "" ...
##  $ skewness_roll_belt.1     : chr  "" "" "" "" ...
##  $ skewness_yaw_belt        : chr  "" "" "" "" ...
##  $ max_roll_belt            : chr  NA NA NA NA ...
##  $ max_picth_belt           : chr  NA NA NA NA ...
##  $ max_yaw_belt             : chr  "" "" "" "" ...
##  $ min_roll_belt            : chr  NA NA NA NA ...
##  $ min_pitch_belt           : chr  NA NA NA NA ...
##  $ min_yaw_belt             : chr  "" "" "" "" ...
##  $ amplitude_roll_belt      : chr  NA NA NA NA ...
##  $ amplitude_pitch_belt     : chr  NA NA NA NA ...
##  $ amplitude_yaw_belt       : chr  "" "" "" "" ...
##  $ var_total_accel_belt     : chr  NA NA NA NA ...
##  $ avg_roll_belt            : chr  NA NA NA NA ...
##  $ stddev_roll_belt         : chr  NA NA NA NA ...
##  $ var_roll_belt            : chr  NA NA NA NA ...
##  $ avg_pitch_belt           : chr  NA NA NA NA ...
##  $ stddev_pitch_belt        : chr  NA NA NA NA ...
##  $ var_pitch_belt           : chr  NA NA NA NA ...
##  $ avg_yaw_belt             : chr  NA NA NA NA ...
##  $ stddev_yaw_belt          : chr  NA NA NA NA ...
##  $ var_yaw_belt             : chr  NA NA NA NA
##  $ var_yaw_belt            : chr  NA NA NA NA ...
##  $ gyros_belt_x            : chr  "0" "0.02" "0" "0.02" ...
##  $ gyros_belt_y            : chr  "0" "0" "0" "0" ...
##  $ gyros_belt_z            : chr  "-0.02" "-0.02" "-0.02" "-0.03" ...
##  $ accel_belt_x            : chr  "-21" "-22" "-20" "-22" ...
##  $ accel_belt_y            : chr  "4" "4" "5" "3" ...
##  $ accel_belt_z            : chr  "22" "22" "23" "21" ...
##  $ magnet_belt_x           : chr  "-3" "-7" "-2" "-6" ...
##  $ magnet_belt_y           : chr  "599" "608" "600" "604" ...
##  $ magnet_belt_z           : chr  "-313" "-311" "-305" "-310" ...
##  $ roll_arm                : chr  "-128" "-128" "-128" "-128" ...
##  $ pitch_arm               : chr  "22.5" "22.5" "22.5" "22.1" ...
##  $ yaw_arm                 : chr  "-161" "-161" "-161" "-161" ...
##  $ total_accel_arm         : chr  "34" "34" "34" "34" ...
##  $ var_accel_arm           : chr  NA NA NA NA ...
##  $ avg_roll_arm            : chr  NA NA NA NA ...
##  $ stddev_roll_arm         : chr  NA NA NA NA ...
##  $ var_roll_arm            : chr  NA NA NA NA ...
##  $ avg_pitch_arm           : chr  NA NA NA NA ...
##  $ stddev_pitch_arm        : chr  NA NA NA NA ...
##  $ var_pitch_arm           : chr  NA NA NA NA ...
##  $ avg_yaw_arm             : chr  NA NA NA NA ...
##  $ stddev_yaw_arm          : chr  NA NA NA NA ...
##  $ var_yaw_arm             : chr  NA NA NA NA ...
##  $ gyros_arm_x             : chr  "0" "0.02" "0.02" "0.02" ...
##  $ gyros_arm_y             : chr  "0" "-0.02" "-0.02" "-0.03" ...
##  $ gyros_arm_z             : chr  "-0.02" "-0.02" "-0.02" "0.02" ...
##  $ accel_arm_x             : chr  "-288" "-290" "-289" "-289" ...
##  $ accel_arm_y             : chr  "109" "110" "110" "111" ...
##  $ accel_arm_z             : chr  "-123" "-125" "-126" "-123" ...
##  $ magnet_arm_x            : chr  "-368" "-369" "-368" "-372" ...
##  $ magnet_arm_y            : chr  "337" "337" "344" "344" ...
##  $ magnet_arm_z            : chr  "516" "513" "513" "512" ...
##  $ kurtosis_roll_arm       : chr  "" "" "" "" ...
##  $ kurtosis_picth_arm      : chr  "" "" "" "" ...
##  $ kurtosis_yaw_arm        : chr  "" "" "" "" ...
##  $ skewness_roll_arm       : chr  "" "" "" "" ...
##  $ skewness_pitch_arm      : chr  "" "" "" "" ...
##  $ skewness_yaw_arm        : chr  "" "" "" "" ...
##  $ max_roll_arm            : chr  NA NA NA NA ...
##  $ max_picth_arm           : chr  NA NA NA NA ...
##  $ max_yaw_arm             : chr  NA NA NA NA ...
##  $ min_roll_arm            : chr  NA NA NA NA ...
##  $ min_pitch_arm           : chr  NA NA NA NA ...
##  $ min_yaw_arm             : chr  NA NA NA NA ...
##  $ amplitude_roll_arm      : chr  NA NA NA NA ...
##  $ amplitude_pitch_arm     : chr  NA NA NA NA ...
##  $ amplitude_yaw_arm       : chr  NA NA NA NA ...
##  $ roll_dumbbell           : chr  "13.05217456" "13.13073959" "12.85074981" "13.43119971"
...
##  $ pitch_dumbbell          : chr  "-70.49400371" "-70.63750507" "-70.27811982" "-
70.39379464" ...
##  $ yaw_dumbbell            : chr  "-84.87393888" "-84.71064711" "-85.14078134" "-
84.87362553" ...
##  $ kurtosis_roll_dumbbell  : chr  "" "" "" "" ...
##  $ kurtosis_picth_dumbbell : chr  "" "" "" "" ...
##  $ kurtosis_yaw_dumbbell   : chr  "" "" "" "" ...
##  $ skewness_roll_dumbbell  : chr  "" "" "" "" ...
##  $ skewness_pitch_dumbbell : chr  "" "" "" "" ...
##  $ skewness_yaw_dumbbell   : chr  "" "" "" "" ...
##  $ max_roll_dumbbell       : chr  NA NA NA NA ...
##  $ max_picth_dumbbell      : chr  NA NA NA NA ...
##  $ max_yaw_dumbbell        : chr  "" "" "" "" ...
##  $ min_roll_dumbbell       : chr  NA NA NA NA ...
##  $ min_pitch_dumbbell      : chr  NA NA NA NA ...
##  $ min_yaw_dumbbell        : chr  "" "" "" "" ...
##  $ amplitude_roll_dumbbell : chr  NA NA NA NA ...
##   [list output truncated]
```

Output 1.5 - Examples of sparsely populated variables in the training data set

```
summary(suppressWarnings(as.numeric(training$kurtosis_roll_belt)))
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##       -2      -1      -1       0       0      33   19226
```

```
summary(suppressWarnings(as.numeric(training$skewness_roll_belt)))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##      -6       0       0       0       0       4   19225
```

Output 2.1 - Models

```
# this function cleans up the training and testing data sets and prepares
# them for model fitting
cleanupData = function(df) {

    # create a temp copy of df
    dfTmp = df

    # remove all the variables that are not going to be used in the model
    dfTmp$user_name = NULL
    dfTmp$cvtd_timestamp = NULL
    dfTmp$classe = NULL
    dfTmp$X = NULL
    dfTmp$raw_timestamp_part_1 = NULL
    dfTmp$raw_timestamp_part_2 = NULL
    dfTmp$new_window = NULL


    # convert all the variables with numerical data to numeric data types use
    # suppressWarnings () to avoid the warning 'NAs introduced by coercion'
    dfTmp[sapply(dfTmp, is.character)] = suppressWarnings(lapply(dfTmp[sapply(dfTmp, 
        is.character)], as.numeric))

    # combine all the variables to form a clean dataframe and return
    dfClean = cbind(classe = as.factor(df$classe), dfTmp)

    return(dfClean)
}

# add missing variable classe to testing data set
testing$classe = "E"

# cleanup the training and testing data sets
trainingClean = cleanupData(training)

testingClean = cleanupData(testing)
# remove the extra columns in testing
testingClean$problem_id = NULL

# Remove all columns that have less than a threshold percentage of data
thresh = 0.5

for (i in colnames(trainingClean)) {

    if (sum(!is.na(trainingClean[i]))/nrow(trainingClean) < thresh) {
        trainingClean[i] = NULL

        # remove the same from testing data as well
        testingClean[i] = NULL
    }
}

# create separate training and cross-validation data sets from the training
# data
inTrain = createDataPartition(trainingClean$classe, p = trSamplePct)[[1]]
trTrain = trainingClean[inTrain, ]
trTest = trainingClean[-inTrain, ]

set.seed(62433)

## Model - 1 fit a model with random forest using the train data
modRf = train(trTrain$classe ~ ., method = "rf", data = trTrain)

# predict using the cross-validation data created from training data
pRf = predict(modRf, newdata = trTest)

# predict using the test data
pRfTest = predict(modRf, newdata = testingClean)

## Model - 2 fit a model with gbm using the train data
modGbm = train(trTrain$classe ~ ., method = "gbm", data = trTrain, verbose = FALSE)

# predict using the cross-validation data created from training data
pGbm = predict(modGbm, newdata = trTest)
```

```
# predict using the test data
pGbmTest = predict(modGbm, newdata = testingClean)
```

Output 2.2 - Accuracy and Out of sample error

```
# check the accuracy of the model on the cross-validation data
accuracyRf = (confusionMatrix(pRf, trTest$classe))$overall[1]

# check the accuracy of the model on the cross-validation data
accuracyGbm = (confusionMatrix(pGbm, trTest$classe))$overall[1]

# Out of sample error
print(paste0("Random forests - Out of sample error (%) = ", round((1 - accuracyRf) *
    100, 1)))
```

```
## [1] "Random forests - Out of sample error (%) = 2.4"
```

```
print(paste0("Gradient Boosting - Out of sample error (%) = ", round((1 - accuracyGbm) *
    100, 1)))
```

```
## [1] "Gradient Boosting - Out of sample error (%) = 3.3"
```