Practical Machine Learning Course Project (revised version).

Martin HEIN (m#)

06 March 2016

Contents

---

1 INTRODUCTION

1.1 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 (see the section on the *Weight Lifting Exercise Dataset*).

1.2 UNDERLYING DATA SET

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv.

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv.

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

1.3 EXPECTED SUBMISSION

The goal of this project is to predict the manner in which they did the exercise. This is the ***`classe`*** variable in the ***training set***. A corresponding report has to be created, describing how the model was built, how cross validation has been applied, the expected out of sample error, and why the choices has been done the way they are. As a final step the prediction model will be facilitated to predict 20 different test cases.

2 SETUP

2.1 PREPARING THE ENVIRONMENT

```
## load libraries
library(data.table)
library(lubridate)
library(ggplot2)
library(caret)
library(gbm)
library(parallel)
library(doParallel)
library(plyr)
```

2.2 GETTING THE DATA SETS

Let us start with retrieving and loading the underlying data set.

```
urlTrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urlTest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

datTrain <- fread(urlTrain, na.strings=c("NA","#DIV/0!",""))
datTest <- fread(urlTest, na.strings=c("NA","#DIV/0!",""))

datTStamp <- Sys.time()
```

```r
dimTbl <- rbind(dim(datTrain), dim(datTest))
colnames(dimTbl) <- c("observations", "variables")
rownames(dimTbl) <- c("training", "testing")
dimTbl
```

```
##          observations variables
## training        19622       160
## testing            20       160
```

3 PREPARING THE DATA SETS

3.1 REMOVING VARIABLES NOT REQUIRED

First variable contains only the observation number, so we can dispose of this variable.

```r
rmVar <- c("V1", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
           "cvtd_timestamp", "new_window", "num_window")
datTrain <- datTrain[, -rmVar, with=FALSE]
datTest <- datTest[, -rmVar, with=FALSE]
```

Let us also remove any variable containing NA values as much as ***at least 60%*** of all the observations of that particular variable.

```r
DTnaVar <- function(dt) {
    cs <- colSums(sapply(dt, is.na))
    dt[, (cs / nrow(dt) < 0.6), with=FALSE]
} # DTnaVar

datTrain <- DTnaVar(datTrain)
datTest <- DTnaVar(datTest)
```

3.2 REMOVING NEAR ZERO VARIANCE VARIABLES

```r
DTnearZeroVar <- function(dt) {
    nzvDT <- nearZeroVar(dt, saveMetrics=TRUE)
    dt[, nzvDT$nzv == FALSE, with=FALSE]
} # DTnearZeroVar

datTrain <- DTnearZeroVar(datTrain)
datTest <- DTnearZeroVar(datTest)
```

3.3 HARMONISE VARIABLES IN DATA SETS

Next let us focus on only those variables present across all data sets.

```
clsTrain <- datTrain[, classe]
datTrain <- datTrain[, colnames(datTrain) %in% colnames(datTest), with=FALSE]
datTrain[, classe := clsTrain]

datTest <- datTest[, colnames(datTest) %in% colnames(datTest), with=FALSE]
```

3.4 SETTING VARIABLE CLASSES IN DATA SETS

```
datTrain[, classe := as.factor(classe)]
```

3.5 PARTITIONING THE DATA SET

As a final step, we will split the ***training data set*** into a ***(new) training set*** and a ***(testing) validation set***, against which we well test the model to build.

```
inTrain <- createDataPartition(y=datTrain$classe, p=0.6, list=FALSE)
datTrainNew <- datTrain[inTrain, ]
datValid <- datTrain[-inTrain, ]
```

```
dimTbl <- rbind(dim(datTrain), dim(datTrainNew), dim(datValid), dim(datTest))
colnames(dimTbl) <- c("observations", "variables")
rownames(dimTbl) <- c("(orignal) training set", "new training set", "(new testing) validation set", "(o
dimTbl
```

```
##                                observations variables
## (orignal) training set                19622        53
## new training set                      11776        53
## (new testing) validation set           7846        53
## (original) testing set                   20        53
```

4 BUILDING A MODEL

4.1 CHOSING A MODELLING APPROACH

Having the data sets thus prepared, we are now ready to build our model.

As our training data set comprises of 53 variables, which might be of a possibly weak nature, our model will be **pursuing a boosting approach**.

4.2 BUiLDING THE MODEL WITH CROSS-VALIDATION

There are various ways of cross-validating a prediction model, one of these would be ***k-fold cross-validation***, which will be facilitated in our case, using ***10 folds***.

```r
set.seed(20160306)

## setup parallel processing
clustFit <- makeCluster(detectCores() - 1)
registerDoParallel(clustFit)

## define training control
ctrlTrain <- trainControl(method="cv", number=10,classProbs=TRUE,
                          savePredictions=TRUE, allowParallel=TRUE)

## fit the model
#modFit <- train(classe ~ ., method="gbm", data=datTrainNew, trControl=ctrlTrain, verbose=FALSE)
modFit <- readRDS(file.path(".", "modFit.RDS"))

## stop parallel processing
stopCluster(clustFit)
```

```r
## summarize results
print(modFit)
```

```
## Stochastic Gradient Boosting
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 10597, 10598, 10598, 10598, 10600, 10600, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa      Accuracy SD
##   1                   50      0.7504234  0.6833775  0.015542902
##   1                  100      0.8219275  0.7744852  0.007621118
##   1                  150      0.8564058  0.8182044  0.006908124
##   2                   50      0.8574236  0.8192785  0.008136478
##   2                  100      0.9070169  0.8823315  0.006866445
##   2                  150      0.9319813  0.9139333  0.005972797
##   3                   50      0.8992902  0.8725049  0.008921923
##   3                  100      0.9414069  0.9258717  0.004169981
##   3                  150      0.9593242  0.9485454  0.004808839
##   Kappa SD
##   0.019825841
##   0.009610841
##   0.008754421
##   0.010296022
##   0.008685263
##   0.007549295
##   0.011324854
##   0.005273169
##   0.006079727
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
```

```
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

In order to save computing power and time, the model will get save upon completion, so that it can be read from file.

```
saveRDS(modFit, file.path(".", "modFit.RDS"))
```

4.3 EVALUATING THE MODEL

Next we will evaluate the accuracy of our model by applying it to the validation data set.

```
predFitValid <- predict(modFit, newdata=datValid)
accFitValid <- confusionMatrix(predFitValid, datValid[, classe])$overall[1]
```

This model applied to the validation data set will render an accurary of 97.08%.

We also will investigate the imporance of the various variable and the significance in contributing to the model.

```
varImp(modFit)
```

```
## gbm variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                   Overall
## roll_belt         100.000
## pitch_forearm      50.934
## yaw_belt           39.927
## magnet_dumbbell_z  33.348
## magnet_dumbbell_y  26.123
## roll_forearm       25.152
## magnet_belt_z      21.814
## pitch_belt         18.518
## gyros_belt_z       15.898
## accel_forearm_x    14.744
## roll_dumbbell      14.059
## magnet_forearm_z   10.147
## gyros_dumbbell_y    9.496
## accel_dumbbell_y    9.269
## accel_forearm_z     8.580
## accel_dumbbell_x    7.426
## yaw_arm             6.808
## magnet_arm_z        5.528
## magnet_belt_y       5.268
## accel_dumbbell_z    4.991
```

5 PREDICTING TEST CASES

As a final step, we will predict some test cases based upon our prediction model.

```r
(predFitTest <- predict(modFit, newdata=datTest))
```

```
##  [1] 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
```

```r
predTest <- cbind(datTest, predFitTest)
```