Prediction Project

Samuel

5 janvier 2018

Machine Learning Final Project

Introduction

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).

Data

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.

Aim of the project

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.

```
##
## 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
```

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

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
## Loading required package: rpart
```

```
## 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
```

Data preparation

Importing the data

```
setwd("D:/Cours/Coursera/ML")

#Dowlanding training and testing sets
url1 <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
trainingDL <- basename(url1);
if (!file.exists(trainingDL)) {
  download.file(url1, trainingDL, method='curl')
}

url2 <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'
testingDL <- basename(url2);
if (!file.exists(testingDL)) {
  download.file(url1, testingDL, method='curl')
}
```

```
#Importing the data
training <- read.csv(trainingDL)
testing <- read.csv(testingDL)
```

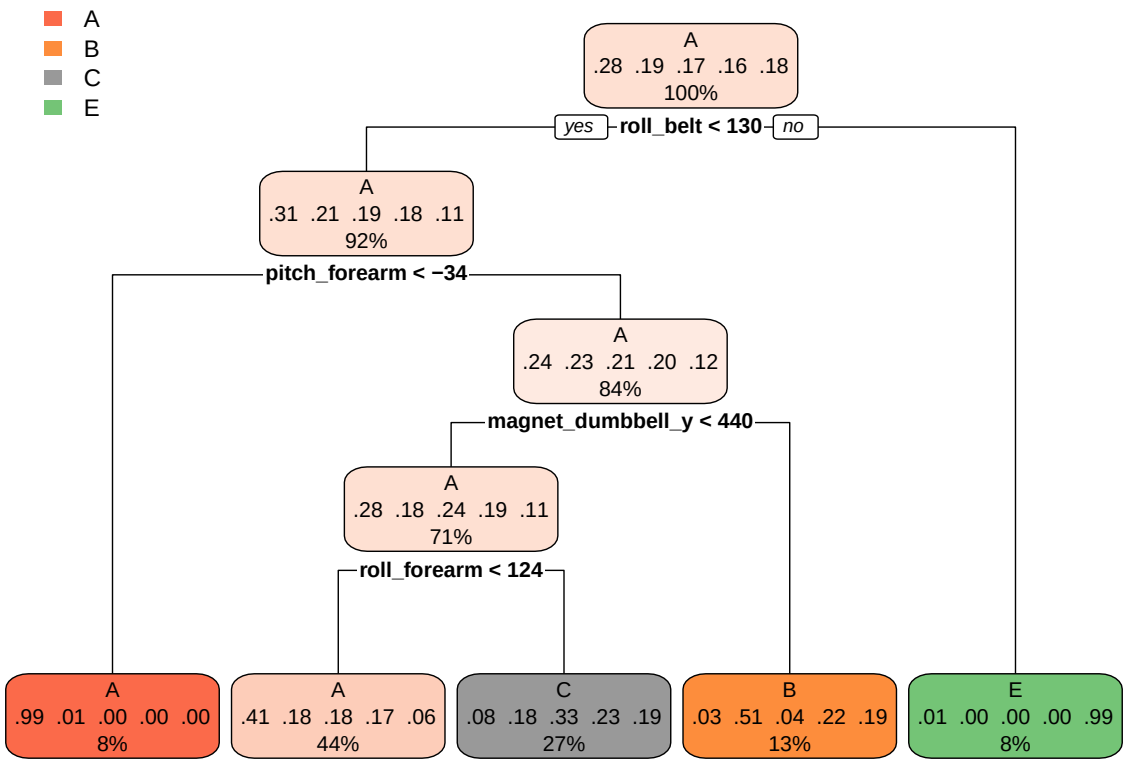
Cleaning the data

```
#A few columns have only ~400 values out of 19622 filled for each window, and a few of them basically a

emptyColsTrain2 <- training[which((summarise_all(training, funs(sum(grepl("DIV", .)))) > 100))]

trainingV1 <- training[, !(names(training) %in% names(emptyColsTrain2))]
testingV1 <- testing[, !(names(training) %in% names(emptyColsTrain2))]
```

```r
#when window is "yes", more measures, so I keep them in a separate dataset to see if any impact
#and keep another data sets with reduced column numbers
trainingW = trainingV1 %>% filter(new_window == "yes")
trainingNW = trainingV1 %>% filter(new_window == "no")

testingW = testingV1 %>% filter(new_window == "yes")
testingNW = testingV1 %>% filter(new_window == "no")

#Getting the mostly empty columns out
emptyCols <- trainingNW[,which(summarise_all(trainingNW, funs(sum(grepl("^$", .)))) > 10)]
trainingNW <- trainingNW[, !(names(trainingNW) %in% names(emptyCols))]

NACols <- trainingNW[,which(summarise_all(trainingNW, funs(sum(is.na(.)))) > 100)]
trainingNW <- trainingNW[, !(names(trainingNW) %in% names(NACols))]
testingNW <- testingNW[, (names(testingNW) %in% names(trainingNW))]

#Transforming the time data into hour of the day and day of the week, then deleting the others timestam
trainingNW2 <- trainingNW %>% mutate(cvtd_timestamp = as.POSIXct((cvtd_timestamp), format = "%d/%m/%Y %
trainingNW2 <- trainingNW2 %>%
    separate(cvtd_timestamp, c("weekday","hour"), sep = " ")
trainingNW3 <- trainingNW2 %>%
    mutate(weekday = as.factor(weekdays(as.Date(weekday, "%Y-%m-%d"))), hour = as.factor(substr(trainin
trainingNW3 <- trainingNW3 %>% select(-c(1:4,7,8))
trainingNW3 <- trainingNW3 %>% mutate_at(c(3:54), funs(as.numeric(.)))
#Converting the day and hour column into dummies
dmyTrain <- dummyVars( ~ weekday + hour, data = trainingNW3)
dummyTrain <- data.frame(predict(dmyTrain, newdata = trainingNW3))
finalTrain <- cbind(dummyTrain, trainingNW3) %>% select(-c(weekday, hour))

#Same for test data
testingNW2 <- testingNW %>% mutate(cvtd_timestamp = as.POSIXct((cvtd_timestamp), format = "%d/%m/%Y %H:
testingNW2 <- testingNW2 %>%
    separate(cvtd_timestamp, c("weekday","hour"), sep = " ")
testingNW3 <- testingNW2 %>%
    mutate(weekday = as.factor(weekdays(as.Date(weekday, "%Y-%m-%d"))), hour = as.factor(substr(testing
testingNW3 <- testingNW3 %>% select(-c(1:4,7,8))
testingNW3 <- testingNW3 %>% mutate_at(c(3:54), funs(as.numeric(.)))

dmyTest <- dummyVars( ~ weekday + hour, data = testingNW3)
dummyTest <- data.frame(predict(dmyTest, newdata = testingNW3))
finalTest <- cbind(dummyTest, testingNW3) %>% select(-c(weekday, hour))
```

Prediction

```r
#Let's first start with a simple tree model
set.seed(123)

modT <- train(classe ~., data = finalTrain, method = "rpart")

predicT <- predict(modT, newdata = finalTest)

rpart.plot(modT$finalModel)
```




```
confusionMatrix(predicT, finalTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4981 1551 1550 1419  516
##          B   80 1259  108  556  476
##          C  397  908 1694 1172  947
##          D    0    0    0    0    0
##          E   13    0    0    0 1589
##
## Overall Statistics
##
##                Accuracy : 0.4956
##                  95% CI : (0.4885, 0.5027)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3405
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9104  0.33862  0.50537   0.0000  0.45040
```

```
## Specificity            0.6336  0.92128  0.78417   1.0000  0.99917
## Pos Pred Value         0.4973  0.50787  0.33099      NaN  0.99189
## Neg Pred Value         0.9467  0.85308  0.88239   0.8362  0.88992
## Prevalence             0.2847  0.19348  0.17444   0.1638  0.18360
## Detection Rate         0.2592  0.06552  0.08816   0.0000  0.08269
## Detection Prevalence   0.5213  0.12901  0.26634   0.0000  0.08337
## Balanced Accuracy      0.7720  0.62995  0.64477   0.5000  0.72478
```

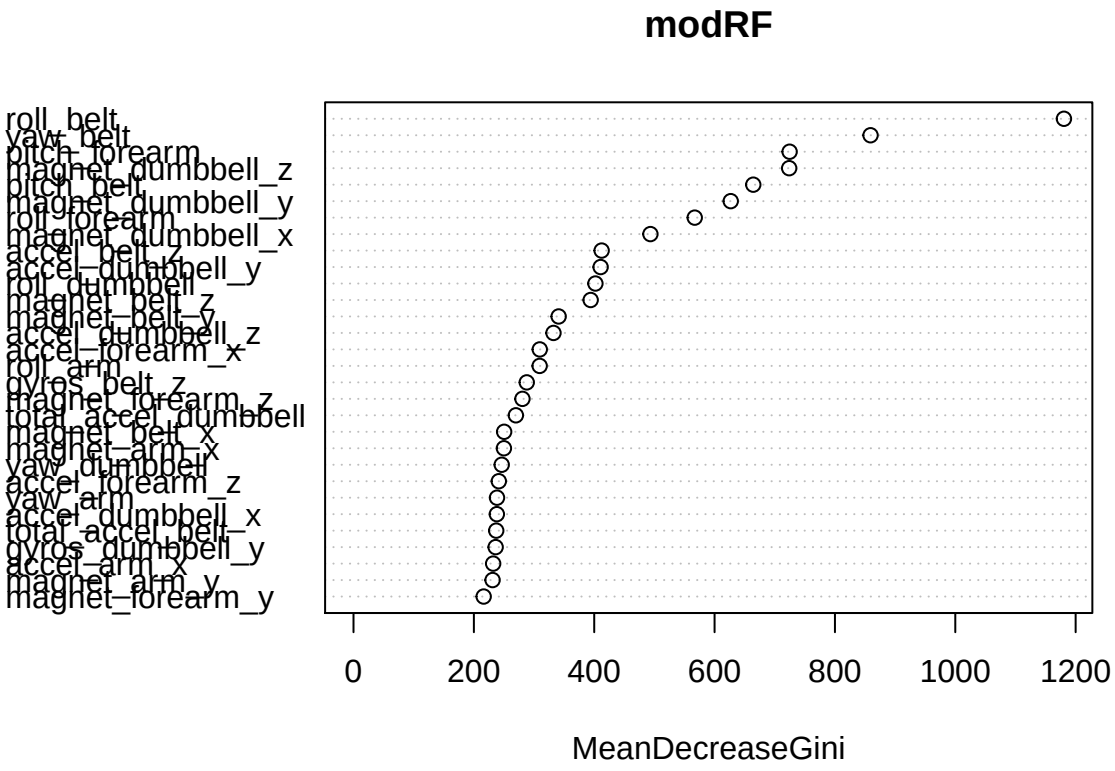
Only about 50% accuracy, with way too many As predicted - very specific for the other classes.

```
#Random forest, I am no using caret package because after 30 minutes, I had still no results
set.seed(123)


modRF <- randomForest(classe ~., data = finalTrain)

predicTree <- predict(modRF, newdata = finalTest)

confusionMatrix(predicTree, finalTest$classe)
```

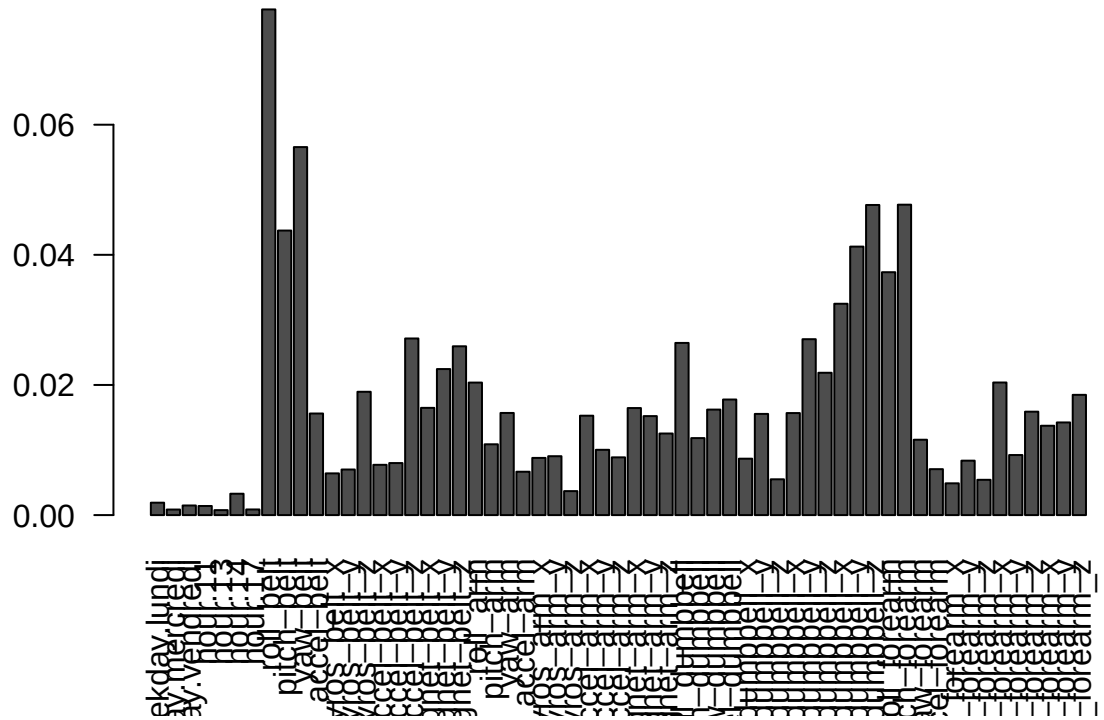
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5471    0    0    0    0
##          B    0 3718    0    0    0
##          C    0    0 3352    0    0
##          D    0    0    0 3147    0
##          E    0    0    0    0 3528
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2847   0.1935   0.1744   0.1638   0.1836
## Detection Rate         0.2847   0.1935   0.1744   0.1638   0.1836
## Detection Prevalence   0.2847   0.1935   0.1744   0.1638   0.1836
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

We obtain a (way too) amazing accuracy of 100%. There is a huge risk of overfitting, however random forest is still one of the best algorithm and we can hope than the accuracy would "only" be about 95% on another dataset, especially since my accuracy isn't really amazing with other classifiers.

```r
#Exploring the variable importance
varImpPlot(modRF,type=2)
```




```r
barplot(t(importance(modRF)/sum(importance(modRF))), las=2)
```



We can see that the weekday and hour variables that I created have very little impact, and I will take them in my final model.

```
#trying out an lda classifier
set.seed(123)
modLDA <- train(classe ~., data = finalTrain, method = "lda", trControl=trainControl(method='cv', numbe
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
predicTree2 <- predict(modLDA, newdata = finalTest)

confusionMatrix(predicTree2, finalTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4612  571  316  200  104
##          B  124 2400  319  112  473
##          C  353  447 2238  360  247
##          D  380  128  419 2390  312
##          E    2  172   60   85 2392
##
## Overall Statistics
##
##                Accuracy : 0.7302
##                  95% CI : (0.7239, 0.7365)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6583
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8430   0.6455   0.6677   0.7595   0.6780
## Specificity            0.9134   0.9337   0.9113   0.9229   0.9797
## Pos Pred Value         0.7948   0.7001   0.6140   0.6586   0.8823
## Neg Pred Value         0.9360   0.9165   0.9285   0.9514   0.9312
## Prevalence             0.2847   0.1935   0.1744   0.1638   0.1836
## Detection Rate         0.2400   0.1249   0.1165   0.1244   0.1245
## Detection Prevalence   0.3020   0.1784   0.1897   0.1889   0.1411
## Balanced Accuracy      0.8782   0.7896   0.7895   0.8412   0.8288
```

73% accuracy is better than the rpart classifier, but still nowhere near as good as the random forest.

Final Model

Considering the difference in accuracy, I keep the random forest model with the risk of overfitting. I retrain the model while taking out some of the useless variables to simplify the model, with no effect on the accuracy.

```
set.seed(123)

FT <- finalTrain %>% dplyr::select(-c(1:7))
ft <- finalTest %>% dplyr::select(-c(1:7))

modRF2 <- randomForest(classe ~., data = FT)

predicTreeV2 <- predict(modRF2, newdata = ft)

confusionMatrix(predicTreeV2, ft$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5471    0    0    0    0
##          B    0 3718    0    0    0
##          C    0    0 3352    0    0
##          D    0    0    0 3147    0
##          E    0    0    0    0 3528
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2847   0.1935   0.1744   0.1638   0.1836
## Detection Rate         0.2847   0.1935   0.1744   0.1638   0.1836
## Detection Prevalence   0.2847   0.1935   0.1744   0.1638   0.1836
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```