Data Science Course 8, Week 4

Dimitri Stucki

0) Preamble

Load the necessary packages

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

## Warning: replacing previous import by 'plyr::ddply' when loading 'caret'

## Warning: replacing previous import by 'tibble::as_tibble' when loading
## 'broom'

## Warning: replacing previous import by 'tibble::tibble' when loading 'broom'

## Warning: replacing previous import by 'rlang::!!' when loading 'recipes'

## Warning: replacing previous import by 'rlang::expr' when loading 'recipes'

## Warning: replacing previous import by 'rlang::f_lhs' when loading 'recipes'

## Warning: replacing previous import by 'rlang::f_rhs' when loading 'recipes'

## Warning: replacing previous import by 'rlang::invoke' when loading
## 'recipes'

## Warning: replacing previous import by 'rlang::is_empty' when loading
## 'recipes'

## Warning: replacing previous import by 'rlang::lang' when loading 'recipes'

## Warning: replacing previous import by 'rlang::na_dbl' when loading
## 'recipes'

## Warning: replacing previous import by 'rlang::names2' when loading
## 'recipes'

## Warning: replacing previous import by 'rlang::quos' when loading 'recipes'

## Warning: replacing previous import by 'rlang::sym' when loading 'recipes'

## Warning: replacing previous import by 'rlang::syms' when loading 'recipes'
```

```
library(knitr)
```

1) Introduction

The aim of this project is to predict whether 6 participants did a training exercise correctly or not. This is coded in the "classe" variable in the training set. Any of the other variables may be used to predict the outcome. Furthermore, the established model will be used to predict 20 different test cases.

Following is a report describing how the model was built, how cross validation was used to asses the error, what the expected out of sample error is, and why the choices were made.

2) Training

The approach to this analysis will be to try and maximize speed. The computation times on an old laptop are too long. A lot of variables will be removed, possibly at the expense of accuracy.

2.1) Reading and cleaning

The data was downloaded from the Coursera website, originally provided by http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

Links to the data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

```
## Read the training data
training <- read.csv("pml-training.csv")
## What do we have?
names(training)
```

```
##   [1] "X"                        "user_name"
##   [3] "raw_timestamp_part_1"     "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"           "new_window"
##   [7] "num_window"               "roll_belt"
##   [9] "pitch_belt"               "yaw_belt"
##  [11] "total_accel_belt"         "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"      "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"       "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"        "max_roll_belt"
##  [19] "max_picth_belt"           "max_yaw_belt"
##  [21] "min_roll_belt"            "min_pitch_belt"
##  [23] "min_yaw_belt"             "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"     "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"     "avg_roll_belt"
##  [29] "stddev_roll_belt"         "var_roll_belt"
##  [31] "avg_pitch_belt"           "stddev_pitch_belt"
##  [33] "var_pitch_belt"           "avg_yaw_belt"
##  [35] "stddev_yaw_belt"          "var_yaw_belt"
##  [37] "gyros_belt_x"             "gyros_belt_y"
##  [39] "gyros_belt_z"             "accel_belt_x"
##  [41] "accel_belt_y"             "accel_belt_z"
```

```
##  [43] "magnet_belt_x"            "magnet_belt_y"
##  [45] "magnet_belt_z"            "roll_arm"
##  [47] "pitch_arm"                "yaw_arm"
##  [49] "total_accel_arm"          "var_accel_arm"
##  [51] "avg_roll_arm"             "stddev_roll_arm"
##  [53] "var_roll_arm"             "avg_pitch_arm"
##  [55] "stddev_pitch_arm"         "var_pitch_arm"
##  [57] "avg_yaw_arm"              "stddev_yaw_arm"
##  [59] "var_yaw_arm"              "gyros_arm_x"
##  [61] "gyros_arm_y"              "gyros_arm_z"
##  [63] "accel_arm_x"              "accel_arm_y"
##  [65] "accel_arm_z"              "magnet_arm_x"
##  [67] "magnet_arm_y"             "magnet_arm_z"
##  [69] "kurtosis_roll_arm"        "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"         "skewness_roll_arm"
##  [73] "skewness_pitch_arm"       "skewness_yaw_arm"
##  [75] "max_roll_arm"             "max_picth_arm"
##  [77] "max_yaw_arm"              "min_roll_arm"
##  [79] "min_pitch_arm"            "min_yaw_arm"
##  [81] "amplitude_roll_arm"       "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"        "roll_dumbbell"
##  [85] "pitch_dumbbell"           "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
## [101] "amplitude_yaw_dumbbell"   "total_accel_dumbbell"
## [103] "var_accel_dumbbell"       "avg_roll_dumbbell"
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
```

```
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

Wow, this is a lot of variables.
Let's do some data exploration first.

Start with the response *classe* (i.e. the way in which the exercises were performed).

```
## How are the observations distributed?
table(training$classe,useNA="always")
```

```
##
##    A    B    C    D    E <NA>
## 5580 3797 3422 3216 3607    0
```

-> No missing values and a more or less balanced distribution.

Maybe there are some variables with an excessive amount of NAs. I could get rid of these.

```
## What proportions of NAs are there?
table(apply(training,2,FUN=function(x){sum(is.na(x))/length(x)}))
```

```
##
##                 0 0.979308938946081
##                93                67
```

-> Indeed, it's either no NAs or almost all NAs. Interesting.

Well, let's lose the 67 NA variables.

```
## Remove
training.noNA <- training[,(apply(training,2,
                                  FUN=function(x){sum(is.na(x))/length(x)})==0)]
## How many variables left?
ncol(training.noNA)
```

```
## [1] 93
```

-> 93 variables left.

Some variables are coded as factors and have a lot of missing entries as well. Removing them could clearly speed up the process. I may include them if the cross validation results in a very low accuracy.

Remove all the ones that are coded as factors, except for user name and classe.

```
## Remove
training.noNA.noFac <- training.noNA[,!names(training.noNA) %in%
                                       names(Filter(is.factor,training.noNA))
                                     [-c(1,37)]]
## How many left?
ncol(training.noNA.noFac)
```

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

-> 53 variables left.

I could also exclude variables that strongly correlate, as they would anyway predict the same thing, and may even introduce a bias (collinearity).

Check the correlations for the numeric variables and remove those with a high correlation.

```
## Create correlation matrix
CorMat <- cor(apply(training.noNA.noFac,2,FUN=as.numeric))
```

```
## Warning in apply(training.noNA.noFac, 2, FUN = as.numeric): NAs introduced
## by coercion

## Warning in apply(training.noNA.noFac, 2, FUN = as.numeric): NAs introduced
## by coercion
```

```
## Are there any correlations larger than 0.8 (or smaller -0.8)
any(abs(CorMat)>0.8)
```

```
## [1] TRUE
```

```
## Convert the matrix to easily identify the correlating variables
CorDf <- as.data.frame(as.table(CorMat))
CorVar <- CorDf[which(abs(CorDf$Freq)>0.8 & CorDf$Var1!=CorDf$Var2),]
## Remove
training.noNA.noFac.noCor <- training.noNA.noFac[, !(names(training.noNA.noFac)
    %in% CorVar$Var2)]
## What's left?
names(training.noNA.noFac.noCor)
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "num_window"           "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "magnet_belt_y"
## [10] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [13] "yaw_arm"              "total_accel_arm"      "gyros_arm_z"
## [16] "accel_arm_y"          "accel_arm_z"          "roll_dumbbell"
## [19] "total_accel_dumbbell" "gyros_dumbbell_y"     "accel_dumbbell_y"
## [22] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [25] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [28] "total_accel_forearm"  "gyros_forearm_x"      "accel_forearm_x"
## [31] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [34] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

-> Maybe a harsh decision, but removing everything with a correlation larger than 0.8 seems acceptable to gain some computational speed.

Also remove X. This is just the row number repeated.

```
## Remove
training.noNA.noFac.noCor <- training.noNA.noFac.noCor[,-1]
## How many are left?
ncol(training.noNA.noFac.noCor)
```

```
## [1] 35
```

-> Down to 35 variables.

Seems okay now. Let's go to model building.

2.3) Build/Train the model

I started out with random forest to maximize speed. But this had a very low accuracy (~0.5). So I arbitrarily switched to gbm (I remembered this from the exercises).

```
## Set the seed for repeatability
set.seed(687)

## Fit a general boosting machine model
modFit.GBM <-
    train(classe~.,method="gbm",
          data=training.noNA.noFac.noCor)

## Predict classe and calculate the confusion matrix to assess accuracy
predicts <- predict(modFit.GBM.CV,training.noNA.noFac.noCor)
confusionMatrix(predicts,training.noNA.noFac.noCor$classe)

## Show the final model
modFit.GBM$finalModel
```

Due to the very long computation time of gbm only the cross validated results are shown (below).

2.4) Cross validate the model

I'll use arbitrarily five folds for cross validation. Let's see how this performs.

```
## Set the seed for repeatability
set.seed(687)

## Fit a general boosting machine model with cross validation
modFit.GBM.CV <-
    train(classe~.,method="gbm",
          data=training.noNA.noFac.noCor,
          trControl=trainControl(method="cv", number=5, savePredictions = TRUE),
          verbose=FALSE)

## Predict classe and calculate the confusion matrix to assess accuracy
predicts.CV <- predict(modFit.GBM.CV,training.noNA.noFac.noCor)
confusionMatrix(predicts.CV,training.noNA.noFac.noCor$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5579    1    0    0    0
##          B    1 3791    3    0    1
```

```
##          C    0    5 3410    4    1
##          D    0    0    9 3206    0
##          E    0    0    0    6 3605
##
## Overall Statistics
##
##                Accuracy : 0.9984
##                  95% CI : (0.9978, 0.9989)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.998
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9998   0.9984   0.9965   0.9969   0.9994
## Specificity            0.9999   0.9997   0.9994   0.9995   0.9996
## Pos Pred Value         0.9998   0.9987   0.9971   0.9972   0.9983
## Neg Pred Value         0.9999   0.9996   0.9993   0.9994   0.9999
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1738   0.1634   0.1837
## Detection Prevalence   0.2844   0.1935   0.1743   0.1638   0.1840
## Balanced Accuracy      0.9999   0.9991   0.9979   0.9982   0.9995
```

-> Wow, that is some accuracy (0.9989)

3) Test the model

2.1) Reading and cleaning

```
## Read the testing data
testing <- read.csv("pml-testing.csv")
```

2.2) Test/Predict the model

```
## Predict the values for the test cases and show them
predicts.test <- predict(modFit.GBM.CV,testing)
predicts.test
```

```
##  [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
```

-> The predictions were all accurate in the quiz, so I guess the out of sample error is 0%?