Practical Machine Learning Assignment

Charlotta Holm

December 11th, 2016

We start by loading the packages we need and reading in the data:

```
setwd("~/Data Science/PML/Week 4")
library(caret)
library(dplyr)
library(randomForest)
```

```
train <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
test <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
```

Exploring the data we see that the train and test data have the same variables except for the last one, that is "classe" in the train set and "problem_id" in the test data.

The variabel we are trying to predict is the classe variable. A indicates that the Unilateral Dumbbell Biceps Curl was performed exactly according to the specification, B that the elbows were thrown to the front, C that the dumbbell was lifted only halfway, D that the dumbbell was lowered only halfway E that the hips were thrown to the front.

In the training data there are the following amounts of different types of excetions:

```
summary(train$classe)
```

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

Preprocessing the data and selecting predictors

We'll examine the variables further too decide which to include in the model.

Many of the 160 variables are partially filled with NA's and/or empty. It seems that the variables with only partial information contain data only for rows with aggregated values. For the prediction we would want to use only raw data from the different sensors. Hence, we decide to exclude the aggregated data. But as the data set is very large, we decide to create a model using only the total acceleration values for the different sensors (arm, belt, forearm, dumbbell) and thus reduce the amount of variables further even though this is a trade off for accuracy of the model.

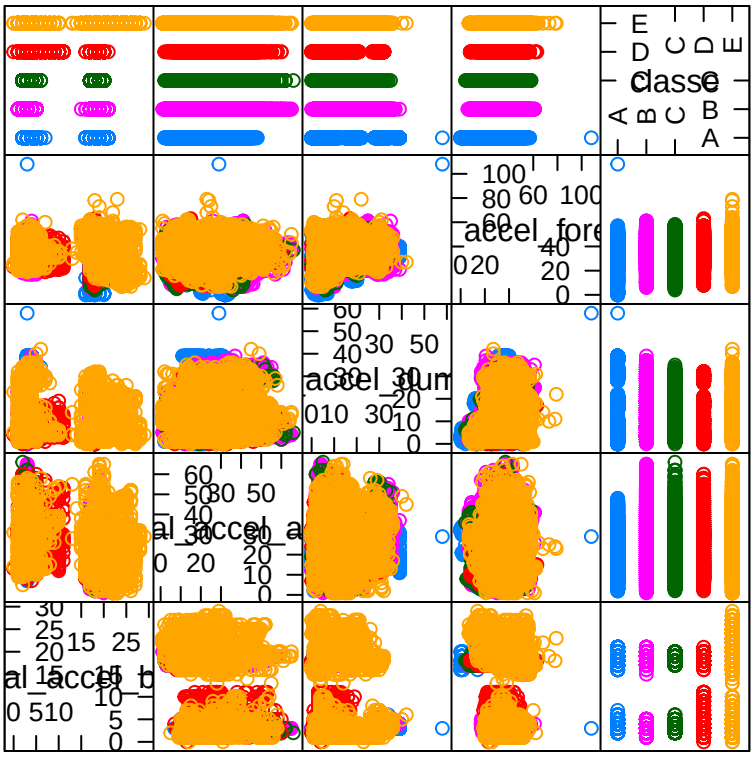
```
var_incl <- c(grep("^total", names(train)),
              grep("^classe", names(train)))


train2 <- train[, c(var_incl)]
```

Now we get a data frame of 5 variables, one of which is the variable "classe" that we want to predict.

Next we want to check the variables to see if any of them show any correlation to the variable we want to predict or any outliers etc. We'll do this by plotting a feature plot of the four predictor variables against the classe variable:

```
featurePlot(train2, train2$classe, plot="pairs")
```



Scatter Plot Matrix

The graphs don't show any particular correlations between the predictors and the variable classe, hence, we will continue with all variables.

Next we'll check for near zero values that could indicate variables that should be eliminated:

```
nz <- nearZeroVar(train2[, -53], saveMetric = TRUE)
```

The near zero -analysis shows us that the amount of values that are near zero are 0, hence we keep all variables!

Applying a model to the training set:

We will use the random forest method which is an ensemble method, i.e. using multiple algorithms, hence no prior cross validation is needed. The cross validation is accounted for by the fact that the random forest averages many decision trees that are trained on different parts of the data.

NB! My model of choice would have been to use "caret::train"" and "rf", and based many of my assumptions in this report on this, but alas, as the model appeared to be extremely slow, I had to change to randomForest::randomForest for the submission.

```
##modFit <- train(classe ~ ., data=train2, method="rf",prox=TRUE) ## Don't use, ran for seven hours bef
modFit <- randomForest(classe~., data=train2, ntree=101)
modFit
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = train2, ntree = 101)
##                Type of random forest: classification
```

```
##                       Number of trees: 101
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 30.57%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4468  281  402  362   67   0.1992832
## B  457 2269  483  273  315   0.4024230
## C  407  311 2226  335  143   0.3495032
## D  430  208  434 1981  163   0.3840174
## E  146  381  195  205 2680   0.2570003
```

Estimate of out of sample error

The out-of-sample-error is the error of the model when applied to the test set. For random forest models this is called the out-of-bag (OOB) error. The estimate for the error can, for random forets models, be seen directly in the confusion matrix. Thus we can see that our random forest model run above has an OOB error rate of 30.7% which is quite big. This is probably due to the small amount of predictors and a better model could probably be found. Had I not struggled for too many hours with the train::caret version of the randomforest model, I would probably had had time to select a better model.

Predicting on the test set and checking whether the prediction was accurate:

We apply the model to the test set to see how it predicts (which cases it predicts to have been performed how):

```
pred <- predict(modFit, test)
table(pred, test$problem_id)
```

```
##
## pred 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
##    A 1 1 0 1 1 0 0 0 1  1  0  0  0  1  1  0  1  0  0  0
##    B 0 0 0 0 0 0 0 1 0  0  1  0  1  0  0  0  0  1  0  1
##    C 0 0 1 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  1  0
##    D 0 0 0 0 0 0 1 0 0  0  0  0  0  0  0  0  0  0  0  0
##    E 0 0 0 0 0 1 0 0 0  0  0  1  0  0  0  1  0  0  0  0
```