Classe machine learning

1. Data overview and preprocessing

```
library(caret)
```

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

```
data=read.csv("pml-training.csv")
data.test=read.csv("pml-testing.csv")
```

The training data consists of 19622 observations of 160 variables. I have managed to find the description of the overall data gathering and feature extraction process, as well as the actual paper presenting the results. However, what I couldn’t find is the actuall codebook explaining what each of these variables means. Just looking at the data, I have managed to find out the following: 1. The data consists of some sort of “time windows” sampled from persons performing excercise, and the number of time window is marked as num_window. 2. For each window, there are several observations of the variables (several rows). One observation for each num_window, marked as “new_window”, contains overall characteristics of that window like mean(roll_belt), that are written in separate columns. All the other observations for that num_window only have NA’s in those columns 3. One num_window can only correspond to one specific “classe”, so one num_window probably means one excercise repetition, or a time span within one exercise repetition. That means that the data from different rows for one num_window can be used together to predict “classe”. Test data set, however, doesnt contain any “new_window” observations; it also contains only one observation for each one num_window, which means, we will have to build a model that predicts classe for the observation based only on the data from that observation.

The summaries like “mean(roll_belt)” become useless in this case, because test data set only has NA values for it. So it makes sense to simply delete them from the model, which is done by the following preprocess function:

The first seven rows were also deleted. It can be argued that keeping extra information, like “subject name”, can help model to achieve better results (at least on train data). However, since our ultimate goal is to be able to identify accuracy of exercise completion in completely new people, who haven’t performed anything for our train set, it makes sense to delete those variables from the model.

2 Model building

We now run a standard “Random Forest” algorhytm to build our prediction model. Note that we dont’s use any data slicing (further dividing train data into train and test subsets) for our cross-validation, because Random Forest algorhytm makes that intrinsically - each new tree takes bootstrapped data out of the set, and uses data, that was not used in the training of the tree, as an error rate estimation the so-called “out of bag” (oob) error estimation, which is the same as cross-validation out-of-sample error estimation.

```
fit=train(classe~.,method="rf",data=tr)
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

Let’s now look at what have we got:

```
fit
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 19622, 19622, 19622, 19622, 19622, 19622, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9926026  0.9906439  0.001152448  0.001454773
##   27    0.9928026  0.9908970  0.001147335  0.001448633
##   52    0.9853086  0.9814188  0.003207897  0.004050940
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

From the “oob” out-of-sample error estimation that was made intrinsically in random forest algorhytm, I think that my out-of-sample error rate will be no more than 2%. Let’s use our model to predict “classe” for our test sample, and go home!

```
te2=preprocess(data.test)
pre=predict(fit,newdata = te2)
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```