Practical Machine Learning Assignment - week 3

Francois Ragnet

Sunday, September 27, 2015

Summary

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, we used 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. We built a Machine Learning model to try and predict the classe of outcome, then tested on a few validation sets. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data Processing and Analysis

Loading and preprocessing the data

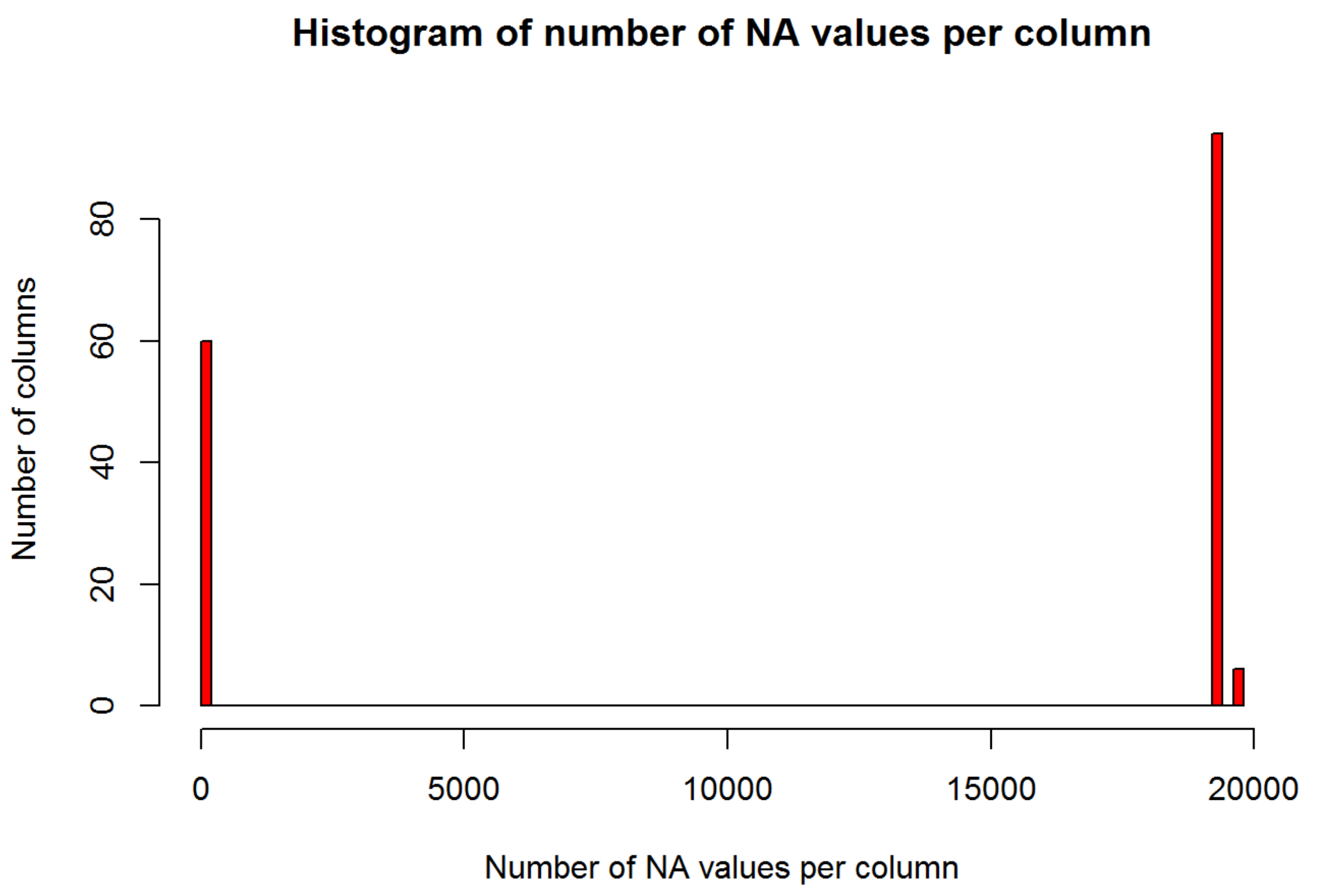
We will download the main file from https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) and the much smaller validation file from https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv). We will be using the first dataset for our model building (split into training and testing), then the second dataset as evaluation (rather than using cross-validation)

```
# setwd("C:\\Local\\My local Documents\\Training\\Data Analytics\\Practical machine learning\\Project")
# download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", dest="pml-training.csv")
# download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", dest="pml-testing.csv")
#pmlMainFULL<-read.csv("pml-training.csv", na.strings=c("NA", "#DIV/0!", ""), stringsAsFactors=FALSE)
pmlMainFULL<-read.csv("pml-training.csv", na.strings=c("NA", "#DIV/0!", ""), stringsAsFactors=TRUE)
pmlValidation<-read.csv("pml-testing.csv", na.strings=c("NA", "#DIV/0!", ""), , stringsAsFactors=FALSE)
```

Creating the training and test set; Preparing the data

A quick analysis of the data shows that many columns are near-empty, or even totally empty.

```
hist(colSums(is.na(pmlMainFULL)), main="Histogram of number of NA values per column", b
reaks = 100, xlab = "Number of NA values per column", ylab = "Number of columns", co
l="red")
```



```
sum((colSums(is.na(pmlMainFULL))>19000)==TRUE)
```

```
## [1] 100
```

```
sum((colSums(is.na(pmlMainFULL))==0)==TRUE)
```

```
## [1] 60
```

60 columns have no NA values, while the other 100 rows are empty or almost (over 19000 NA values, out of 19622). With only a few NA values, we could have tried to impute, but that will most likely not work here since there is so much missing data. Let’s remove these last columns from our main dataset, and apply the same pre-processing to our Validation set.

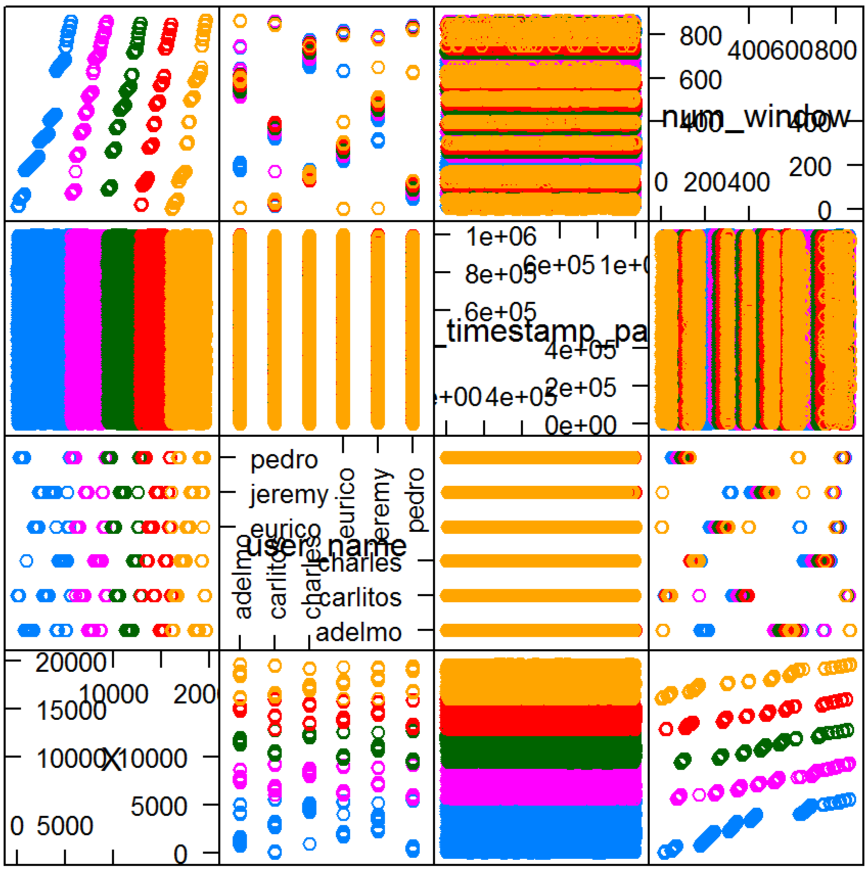
```
pmlMain<-pmlMainFULL[,colSums(is.na(pmlMainFULL))==0]
pmlValidation<-pmlValidation[,colSums(is.na(pmlValidation)) == 0]
```

Let’s see a summary of our variables, then pair-plot the first variables, which are not sensor variables.

```
summary(pmlMain[,1:10])
```

```
##        X            user_name    raw_timestamp_part_1 raw_timestamp_part_2
##  Min.   :    1   adelmo  :3892   Min.   :1.322e+09    Min.   :   294
##  1st Qu.: 4906   carlitos:3112   1st Qu.:1.323e+09    1st Qu.:252912
##  Median : 9812   charles :3536   Median :1.323e+09    Median :496380
##  Mean   : 9812   eurico  :3070   Mean   :1.323e+09    Mean   :500656
##  3rd Qu.:14717   jeremy  :3402   3rd Qu.:1.323e+09    3rd Qu.:751891
##  Max.   :19622   pedro   :2610   Max.   :1.323e+09    Max.   :998801
##
##           cvtd_timestamp  new_window    num_window      roll_belt
##  28/11/2011 14:14: 1498   no :19216   Min.   :  1.0   Min.   :-28.90
##  05/12/2011 11:24: 1497   yes:  406   1st Qu.:222.0   1st Qu.:  1.10
##  30/11/2011 17:11: 1440               Median :424.0   Median :113.00
##  05/12/2011 11:25: 1425               Mean   :430.6   Mean   : 64.41
##  02/12/2011 14:57: 1380               3rd Qu.:644.0   3rd Qu.:123.00
##  02/12/2011 13:34: 1375               Max.   :864.0   Max.   :162.00
##  (Other)         :11007
##    pitch_belt          yaw_belt
##  Min.   :-55.8000   Min.   :-180.00
##  1st Qu.:  1.7600   1st Qu.: -88.30
##  Median :  5.2800   Median : -13.00
##  Mean   :  0.3053   Mean   : -11.21
##  3rd Qu.: 14.9000   3rd Qu.:  12.90
##  Max.   : 60.3000   Max.   : 179.00
##
```

```
#Before we go into heavy algorithms, let's enable parallel processing
library(doParallel)
registerDoParallel(cores=3)
#Now plot the first variables
library(caret); library(kernlab)
featurePlot(x=pmlMain[,c("X","user_name","raw_timestamp_part_2","num_window")],y=pmlMai
n$classe,plot="pairs")
```



Scatter Plot Matrix

These look like strange - X in has almost perfect correlation with classe, our outcome, and some of the time stamp variables appear to be the same. X is most likely an index after ordering by classe, or and/or experiments were ordered by time and by “classe”.

Let’s not use those variables in dataset and focus our training solely on sensor data.

```
pmlMain<-pmlMain[,-c(1:7)]
pmlValidation<-pmlValidation[,-c(1:7)]
```

Preparing the dataset

First, let’s create a training set and a testing set, using 75% of samples for training.

```
set.seed(7081971)
inTrain<-createDataPartition(y=pmlMain$classe, p=0.75, list = FALSE)
trainingPML<-pmlMain[inTrain,]
testingPML<-pmlMain[-inTrain,]
```

Building Machine Learning Models.

In this part we’ll try a few categorization models that can produce a factor output, plot the importance of the variables, then build a few models and estimate the Out-of-Sample Error rate.

Let’s first set our code to use multiple cores - if not, training certain models can be extremely long.

```
library(doParallel)
registerDoParallel(cores=3)
```

Gradient Boosted Model

Let’s start with a boosted tree model. A simple model could have been built using:

```
#Build our basic (simple validation) model
#modFitGBM<-train(classe~.,method="gbm",data=trainingPML)
```

Cross-validation training

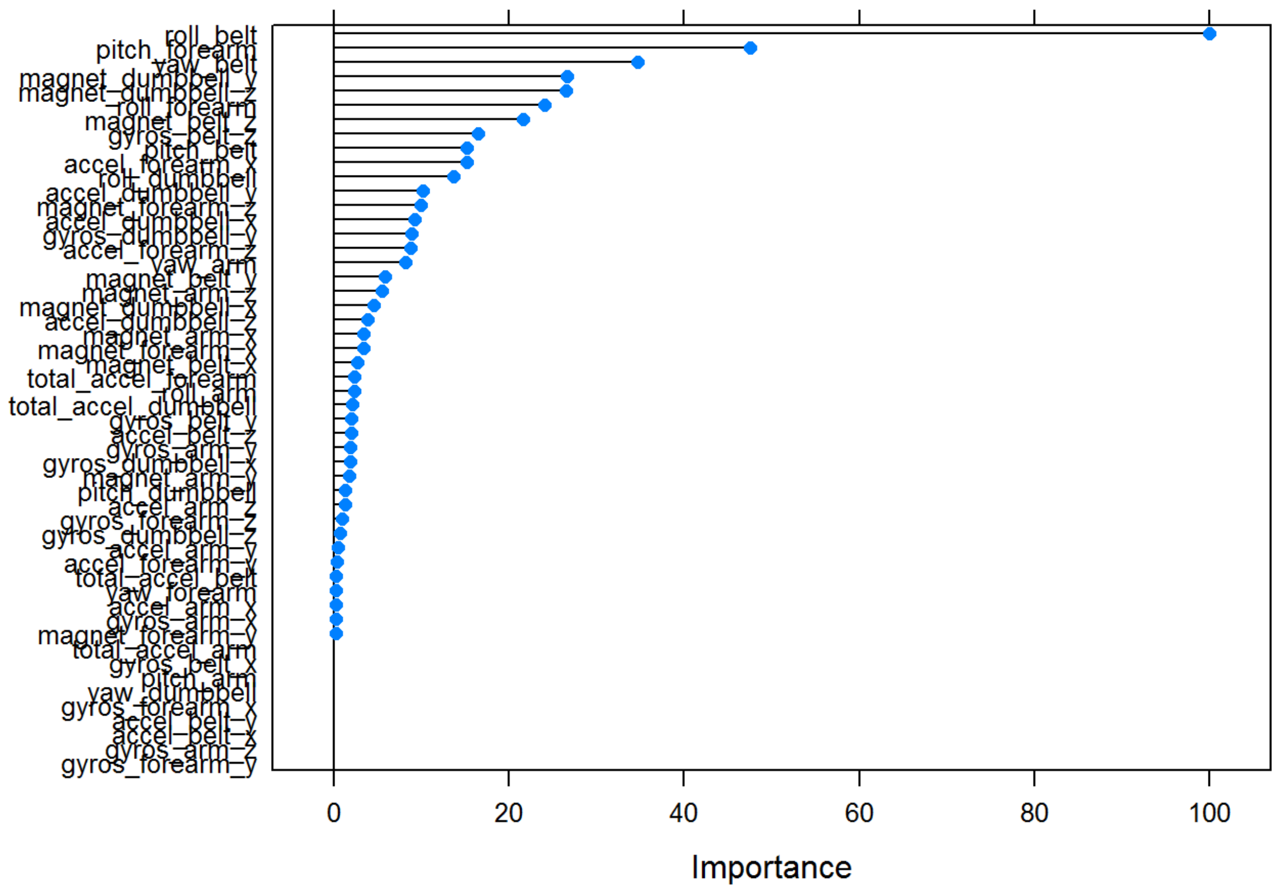
Contrarily to what I said in the submission on the Coursera side, I was able to introduce cross-validation in time for the deadline We will use 5-fold cross validation to train our model:

```
fitControl <- trainControl(method = "repeatedcv", number = 5, repeats = 5)
```

```
#Build our 5-fold cross validation model
modFitGBM<-train(classe~.,method="gbm",trControl = fitControl, verbose = FALSE, data=tr
ainingPML)
```

Let’s look at variable importance.

```
#Plot variable importance
plot(varImp(modFitGBM))
```



roll_belt is the most important predictor, followed by *pitch_forearm*, *yaw_belt*, then others

Prediction Accuracy and Error rate

We can now predict on the testing dataset to estimate out-of-sample accuracy and error rate of the model.

```
predictions<-predict(modFitGBM, newdata=testingPML)
#Estimate Prediction accuracy
confusionMatrix(predictions, testingPML$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1370   29    0    0    2
##          B   19  903   39    2   13
##          C    5   15  802   29    9
##          D    0    2   11  770   10
##          E    1    0    3    3  867
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9608          
##                  95% CI : (0.955, 0.9661)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9505          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9821   0.9515   0.9380   0.9577   0.9623
## Specificity            0.9912   0.9815   0.9857   0.9944   0.9983
## Pos Pred Value         0.9779   0.9252   0.9326   0.9710   0.9920
## Neg Pred Value         0.9929   0.9883   0.9869   0.9917   0.9916
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2794   0.1841   0.1635   0.1570   0.1768
## Detection Prevalence   0.2857   0.1990   0.1754   0.1617   0.1782
## Balanced Accuracy      0.9866   0.9665   0.9618   0.9761   0.9803
```

Results look good. The **out-of-sample error rate is around 4%**, with the confidence intervals shown above.

Evaluation on the validation dataset

The objective of this exercise was also to predict the results on 20 new values.

```
predictions<-predict(modFitGBM, newdata=pmlValidation)
```

Let’s create one file per answer using the following function:

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
```

we’ll write our predictions in the results subdirectory/

```
dir.create("results")
setwd("results")
pml_write_files(predictions)
```

RPART

We then tried a second training type - a classification tree, with RPART.

```
modFitRPART<-train(classe~.,method="rpart",data=trainingPML)
```

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

```
#plot(varImp(modFitRPART))
predictions<-predict(modFitRPART, newdata=testingPML)
confusionMatrix(predictions, testingPML$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1268  384  389  351  137
##          B   18  331   35  156  120
##          C  104  234  431  297  238
##          D    0    0    0    0    0
##          E    5    0    0    0  406
##
## Overall Statistics
##
##                Accuracy : 0.4967
##                  95% CI : (0.4826, 0.5108)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3426
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9090   0.3488  0.50409   0.0000  0.45061
## Specificity            0.6406   0.9168  0.78439   1.0000  0.99875
## Pos Pred Value         0.5014   0.5015  0.33052      NaN  0.98783
## Neg Pred Value         0.9465   0.8544  0.88222   0.8361  0.88983
## Prevalence             0.2845   0.1935  0.17435   0.1639  0.18373
## Detection Rate         0.2586   0.0675  0.08789   0.0000  0.08279
## Detection Prevalence   0.5157   0.1346  0.26591   0.0000  0.08381
## Balanced Accuracy      0.7748   0.6328  0.64424   0.5000  0.72468
```

This model is not working well at all, as indicated by the resulting Accuracy (around 50%!).

Further investigations

Here are a few other models we were planning to run, but were not able to because of time:

GBM PCA

Principal Component Analysis with Gradient Boosted Model

```
#Estimate Error rates
# modFitGBMPCA<-train(classe~.,method="gbm",preProcess="pca",data=trainingPML)
# predictions<-predict(modFitGBMPCA, newdata=testingPML)
# confusionMatrix(predictions, testingPML$classe)
```

Random Forest

Random Forest

```
#Estimate Error rates
# modFitRF<-train(classe~.,method="rf",data=trainingPML)
# plot(varImp(modFitRF))
# predictions<-predict(modFitRF, newdata=testingPML)
# confusionMatrix(predictions, testingPML$classe)
```

Treebag

Bagging option 1

```
# modFitTreeBag<-train(classe~.,method="treebag",data=trainingPML)
# plot(varImp(modFitTreeBag))
# predictions<-predict(modFitTreeBag, newdata=testingPML)
# confusionMatrix(predictions, testingPML$classe)
```

BagFDA

Bagging option 2

```
#Estimate Error rates
# modFitBagFDA<-train(classe~.,method="bagFDA",data=trainingPML)
# plot(varImp(modFitBagFDA))
# predictions<-predict(modFitBagFDA, newdata=testingPML)
# confusionMatrix(predictions, testingPML$classe)
```