Practical Machine Learning Project

I 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). The question is: Can we predict the appropriate activity quality (class A-E), based on recorded information ?

II study

Data cleansing

First, read the data and the description, examin it, do some cleaning.

```
usePackage <- function(p) {
  if (!is.element(p, installed.packages()[,1]))
    install.packages(p, dep = TRUE)
  require(p, character.only = TRUE)
}
usePackage("RCurl")
training <- read.csv(file = "pml-training.csv",head=TRUE,sep=",",na.strings=c("#DIV/0!"))
testing <- read.csv(file = "pml-testing.csv",head=TRUE,sep=",",na.strings=c("#DIV/0!"))
summary(training)
head(training)
names(training)
dim(training)
```

classe function of 159 predictors 19622 rows

Some columns contain essentialy Na. Some lines are empy Some lines contains DIV/0, replaced by Na. Do the cleaning, nb of usefull predictor reduced.

```
# remove near zero variance covariates
usePackage <- function(p) {
  if (!is.element(p, installed.packages()[,1]))
    install.packages(p, dep = TRUE)
  require(p, character.only = TRUE)
}
# suppress 8 first columns, used for data management
training <- training[, -seq(from = 1, to = 8, by = 1)]
testing <- testing[, -seq(from = 1, to = 8, by = 1)]

options(repos = list(CRAN = "http://cran.at.r-project.org/"))
usePackage("caret")
```

```
nzv <- nearZeroVar(training, saveMetrics = T)
training <- training[, !nzv$nzv]
testing <-testing[, !nzv$nzv]
dim(training)
```

```
## [1] 19622    76
```

```
# remove variables with more than 90% missing values
nav <- sapply(colnames(training), function(x) if(sum(is.na(training[, x])) > 0.9*nrow(training)){return
training <- training[, !nav]
trainingsetindex <- createDataPartition(y = training$classe, p = 0.7, list = FALSE)
trainingset<- training[trainingsetindex, ]

validatingset <- training[-trainingsetindex, ]
```

Separate avalable data in a training set (70%) and a validation set (30%). the validation set will be used to evaluate out of sample error.

II Feature selection and model construction

Referenced paper identifies folowing features: The 12 features selected through this procedure were: (1) Sensor on the Belt: discretization of the module of acceleration vector, variance of pitch, and variance of roll;
(2) Sensor on the leftthigh: module of acceleration vector, discretization, and variance of pitch;
(3) Sensor on the right ankle: variance of pitch, and variance of roll;
(4) Sensor on the right arm: discretization of the module of acceleration vector;
From all sensors: average acceleration and standard deviation of acceleration.

To optimize the computation time, take advantage of the parallel computing. The code is run in a multi-core machine, so we allow it to use up to the total number of cores - 1. (suppressed, not working)

Apply random forests algorithm, with 15 fold cross validation:

Random forests are a good candidate when number of predictors is large, and influential predictors are unknown. choice could be reconsidered if accuracy is low.

```
usePackage <- function(p) {
  if (!is.element(p, installed.packages()[,1]))
    install.packages(p, dep = TRUE)
  require(p, character.only = TRUE)
}
usePackage("caret")
usePackage("randomForest")
usePackage("class")
usePackage("e1071")
set.seed(33800)
rfFitfunc <- train(classe ~ ., method = "rf", data = trainingset, preProcess=c("center", "scale"), impo
```

```
load("rfFitfunc.RData")

imp <- varImp(rfFitfunc)$importance
```

```
imp$max <- apply(imp, 1, max)
imp <- imp[order(imp$max, decreasing = T), ]
```

Overall accuracy:

Final model confusion matrix, showing good and false predictions:

```
# final model
confusion<-rfFitfunc$finalModel
confusion
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = ..1)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 26
##
##         OOB estimate of  error rate: 0.72%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3900    3    1    0    2    0.001536
## B   20 2630    8    0    0    0.010534
## C    0   15 2372    9    0    0.010017
## D    0    0   28 2220    4    0.014210
## E    0    0    3    6 2516    0.003564
```

The random forests algorithm generated a very accurate model with **accuracy close to 1**. as we see from the plots.(Error 0.75%)
We keep this approch to continue the study.

Out of Sample Error:

```
# out of sample error calculation
predictionvalidateset <- as.character(predict(rfFitfunc, newdata=validatingset))
outOfSampleError <- sum(predictionvalidateset != as.character(validatingset$classe)) * 100 / nrow(valida
```

The Out of Sample error calculated on the test set is 0.24 %

Prepare predictions for testing data

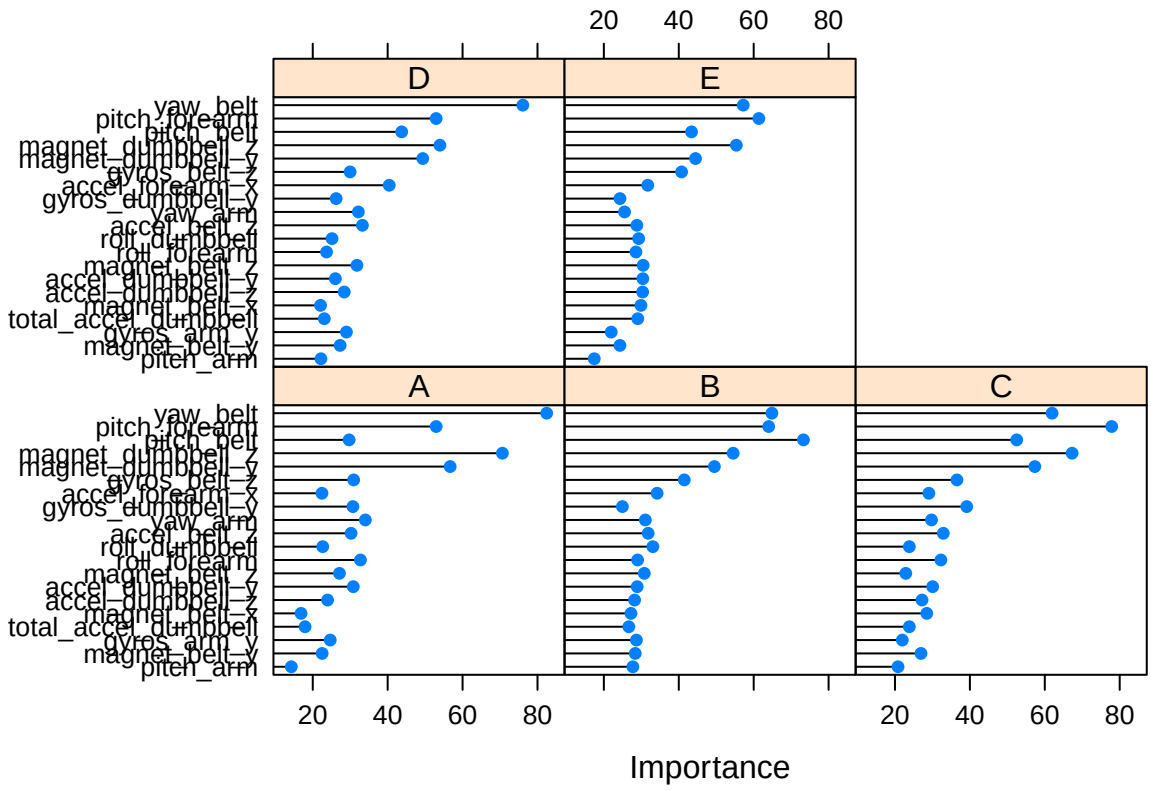
```
#predictions:
prediction
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```

III Figures

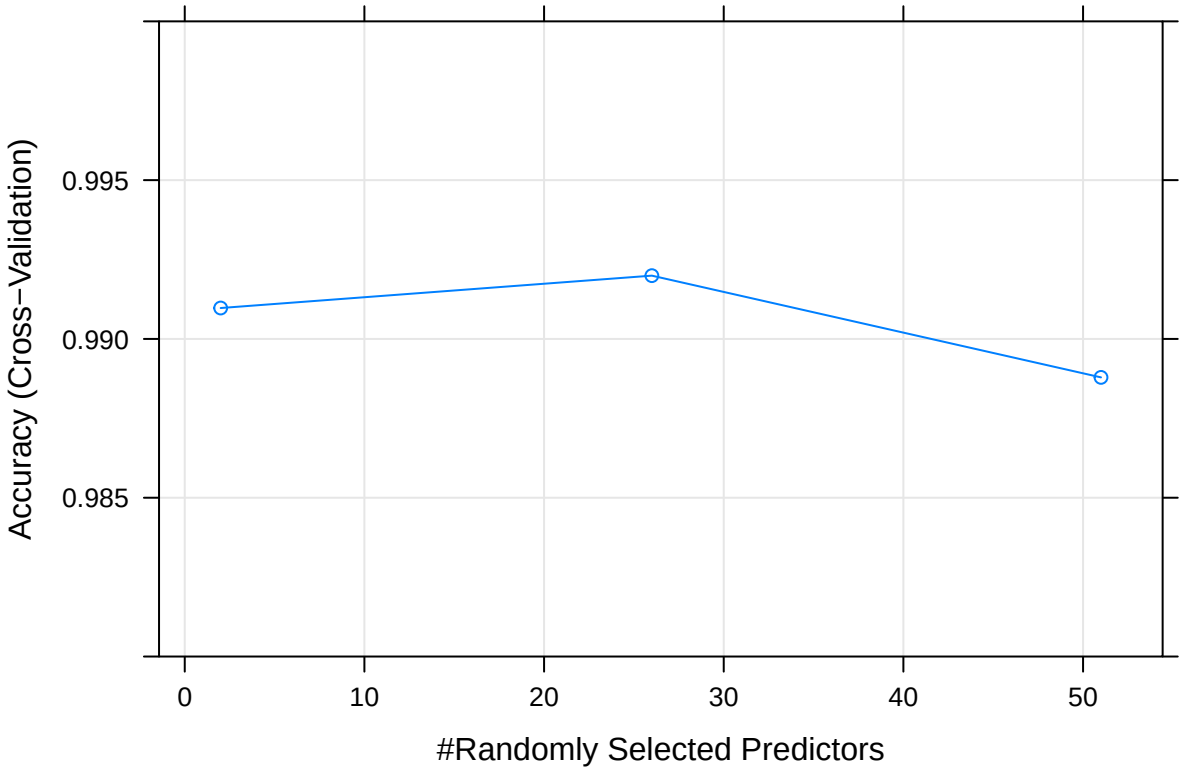
The next figure shows the importance measures for the top 20 attributes, in decreasing order of importance for each class.

```
#Plot of important variables
plot(varImp(rfFitfunc, scale=FALSE), top=20)
```



The next plot shows accuracy function of the predictors

```
plot(rfFitfunc, ylim = c(0.98, 1))
```

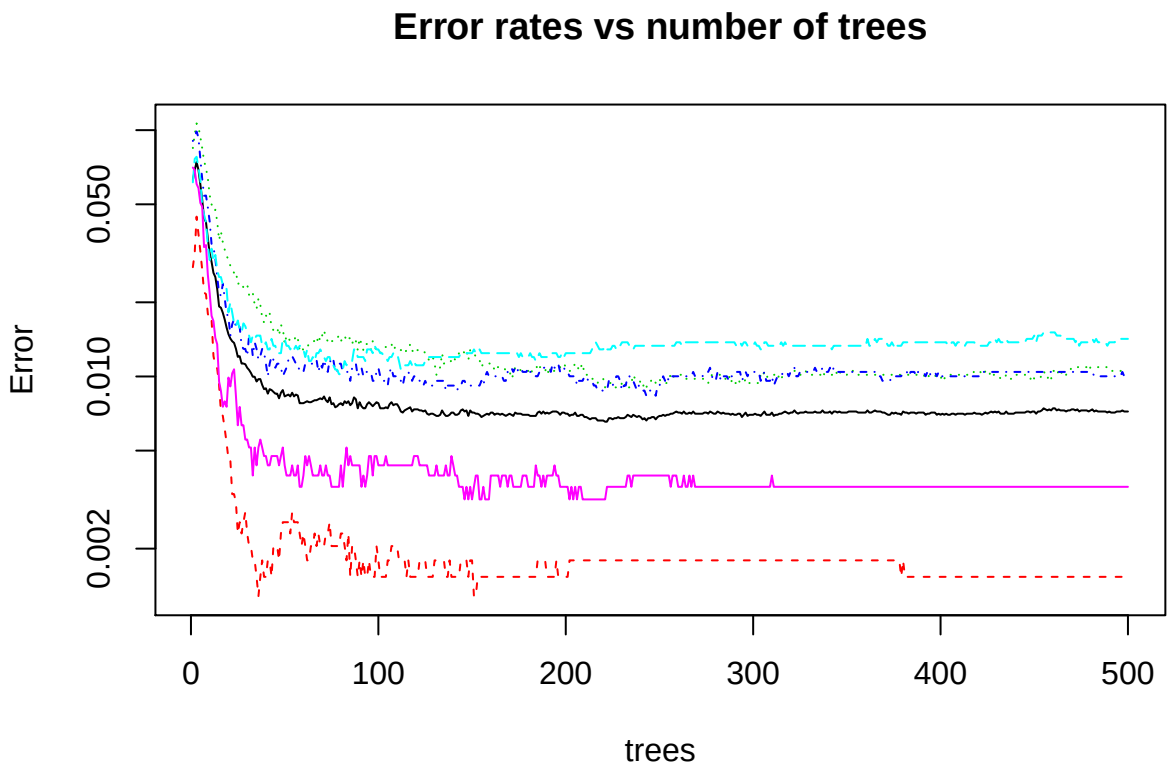



```
rfFitfunc
```

```
## Random Forest
##
## 13737 samples
##    51 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered, scaled
## Resampling: Cross-Validated (10 fold)
##
## Summary of sample sizes: 12364, 12362, 12364, 12363, 12364, 12363, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    1         1      0.002        0.003
##   26    1         1      0.002        0.003
##   51    1         1      0.002        0.003
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 26.
```

The next plot shows the error rates vs number of trees for each class. As the number of trees increases the error rates decrease. The number of trees used in the analysis is 500.

```r
#Plot error rates
plot(rfFitfunc$finalModel, log="y", main="Error rates vs number of trees")
```



IV Prepare results for online automated evaluation

Write prediction files

```r
# Prepare results for online automated evaluation
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)
  }
}
pml_write_files(prediction)
```