##Practical Machine Learning Project

author: “MAlexey”
date: 22.03.14 <<<<<<< HEAD output: html_document:

keep_md: true

Course Project Assignment

“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). ”

Data Sources

The training and test data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The original source of the data is: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Project Objectives

The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases. 1. Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-). 2. You should also apply your machine learning algorithm to

the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

Reproducibility

Libraries Needed

```
library(abind)
library(arm)
library(caret)
library(kernlab)
library(klaR)
library(rattle)
library(randomForest)
library(rpart)
```

Seed for pseudo-random generator

```
set.seed(1234)
```

Data Importing into R

Setting train and testing datasets' urls:

```
urlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-train
urlTest <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testin
```

File retrieval and reading:

```
csvTrain <- "pml-training.csv"

if (file.exists(csvTrain)) {
        train <- read.csv(csvTrain, na.strings=c("NA","#DIV/0!",""))
} else {
        download.file(urlTrain,csvTrain)
        train <- read.csv(csvTrain, na.strings=c("NA","#DIV/0!",""))
        }

csvTest <-  "pml-testing.csv"

if (file.exists(csvTest)) {
        test <- read.csv(csvTest, na.strings=c("NA","#DIV/0!",""))
} else {
        download.file(urlTest,csvTest)
        test <- read.csv(csvTest, na.strings=c("NA","#DIV/0!",""))
}
```

Training-Set cleaning and pre-processing

To ease the computation and due to the low informativity loss, the dataset is cleaned from the variables with an high share of NAs and from the ones characterized by low variance.

```
nearzero <- nearZeroVar(train, saveMetrics = TRUE)
train <- train[, !nearzero$nzv]
```

Variables with more than 50% missing values are removed

```
toberem <- sapply(colnames(train), function(x) if(sum(is.na(train[, x])) 
}else{
return(FALSE)
}
)
train <- train[, !toberem]
```

Variables related with data acquisition (like: id, timestamps, individuals' names, etc.) are not suitable to be used in prediction and are removed

```
train <- train[, -(1:6)]
```

Correlation analysis:

```
Hcorr <- caret::findCorrelation(cor(train[, -53]), cutoff=0.8)
names(train)[Hcorr]
```

```
##  [1] "accel_belt_z"     "roll_belt"        "accel_belt_y"
##  [4] "accel_dumbbell_z" "accel_belt_x"     "pitch_belt"
##  [7] "accel_arm_x"      "accel_dumbbell_x" "magnet_arm_y"
## [10] "gyros_arm_y"      "gyros_forearm_z"  "gyros_dumbbell_x"
```

Many variables are highly correlated. PCA will be used in the pre-processing. After the data cleaning the variables selected to specify the model are:

```
names(train)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbe
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

Model Specification and Cross Validation

In order to avoid overfitting and to reduce out of sample errors, TrainControl is used to perform 7-fold cross validation.

```
tc <- trainControl(method = "cv", number = 7, verboseIter=FALSE , preProc
```

Six models are estimated: Random forest, Support Vector Machine (both radial and linear), a Neural net, a Bayes Generalized linear model and a Logit Boosted model.

```
rf <- train(classe ~ ., data = train, method = "rf", trControl= tc)
svmr <- train(classe ~ ., data = train, method = "svmRadial", trControl= 
NN <- train(classe ~ ., data = train, method = "nnet", trControl= tc, ver
```

```
## Loading required package: nnet
```

```
svml <- train(classe ~ ., data = train, method = "svmLinear", trControl= 
bayesglm <- train(classe ~ ., data = train, method = "bayesglm", trContro
logitboost <- train(classe ~ ., data = train, method = "LogitBoost", trCo
```

```
## Loading required package: caTools
```

Accuracy comparision

```
model <- c("Random Forest", "SVM (radial)","LogitBoost","SVM (linear)","N
Accuracy <- c(max(rf$results$Accuracy),
        max(svmr$results$Accuracy),
        max(logitboost$results$Accuracy),
        max(svml$results$Accuracy),
        max(NN$results$Accuracy),
        max(bayesglm$results$Accuracy))

Kappa <- c(max(rf$results$Kappa),
        max(svmr$results$Kappa),
        max(logitboost$results$Kappa),
        max(svml$results$Kappa),
        max(NN$results$Kappa),
        max(bayesglm$results$Kappa))

performance <- cbind(model,Accuracy,Kappa)
```

```
knitr::kable(performance)
```

model	Accuracy	Kappa
Random Forest	0.995107700518884	0.993811309402383
SVM (radial)	0.936449294114219	0.919466962977943
LogitBoost	0.90037516095944	0.873309348702899
SVM (linear)	0.786616098229509	0.728704679568245
Neural Net	0.429203084526822	0.280747746232619
Bayes GLM	0.400774771503264	0.233871448540575

Random forest and SVM(radial) provide the best results and will provide the predictions for the submission. Even if the Out of sample error cannot be estimated exactly, the in-sample error obtained through cross-validation is calculated over different test sets and should provide a better estimate of out-of sample error with respect to the case of no cross-validation.

Prediction of “classe” variable for the test set

```
rfPred <- predict(rf, test)
svmrPred <- predict(svmr, test)
```

Checking if the models give same predictions

```
prediction <- data.frame(cbind(rfPred, svmrPred))
prediction$same <- with(prediction, rfPred == svmrPred)
colnames(prediction) <- c("Random Forest", "SVM (radial)", "Same Predicti
```

```
knitr::kable(prediction)
```

Random Forest	SVM (radial)	Same Prediction
2	2	TRUE
1	1	TRUE
2	2	TRUE
1	1	TRUE
1	1	TRUE
5	5	TRUE
4	4	TRUE
2	2	TRUE
1	1	TRUE
1	1	TRUE
2	2	TRUE
3	3	TRUE
2	2	TRUE
1	1	TRUE
5	5	TRUE
5	5	TRUE
1	1	TRUE
2	2	TRUE
2	2	TRUE
2	2	TRUE

Generation of the files to be submitted is made through the provided 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=FALS
        }
}

pml_write_files(rfPred)
pml_write_files(svmrPred)
```

###Conclusions The random forest model provides an outstanding accuracy and, accordingly, the predictions for the test set were correct in 100% of the cases.