~~~
---
title: "Practical Machine Learning - Course Final Project"
author: "Alberto A. Caeiro Jr"
date: "February 16, 2015"
output: html_document
---

# Executive Summary & 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.

The goal of this 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.

# About the Data
The data for this project are available here:
. Training dataset :
"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
. Testing dataset:
"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
The data for this project come from this source:
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. So let's have a look on the dataset and on the classe
variable.

# Data Processing
## Obtaining and Cleaning the data
```{r, message=FALSE}
# Enabling Multi Core for modeling processing
library(doMC)
  registerDoMC(cores = 2)

#Loading used libraries
library(caret);library(klaR);  library(rpart)
library(randomForest); library(gbm)

#setting the seed for reproducible computation
set.seed(12345)

#setting the working directory folder
setwd("~/Developer/Data Science Specialization/Practical Machine
~~~

~~~
Learning/Project")

# loading both testing and training dataset (considering both files were
already downloaded)
trainFile <- "./pml-training.csv"
training <- read.csv(file=trainFile, header=TRUE, sep=",",
na.strings=c("NA","#DIV/0!",""))
testFile <- "./pml-testing.csv"
testing <- read.csv(file=testFile, header=TRUE, sep=",",
na.strings=c("NA","#DIV/0!",""))

# Summary for the training predictors and outcome
str(training)
```
As one can see in the "str(training)", the dataset has a lot of NAs.
Let's first clean the data basically removing the NAs, the IDs and the
zero variability columns.
```{r}
# Starting the Cleaning Process
nzvCol <- nearZeroVar(training)
training <- training[,-nzvCol]

# Since we have lots of variables, remove any with NA's or have empty
strings, and the one's that are not predictors variables
filterData <- function(idf) {
    idx.keep <- !sapply(idf, function(x) any(is.na(x)))
    idf <- idf[, idx.keep]
    idx.keep <- !sapply(idf, function(x) any(x==""))
    idf <- idf[, idx.keep]

    # Remove the columns that aren't the predictor variables
    col.rm <- c("X", "user_name", "raw_timestamp_part_1",
"raw_timestamp_part_2",
                "cvtd_timestamp", "new_window", "num_window")
    idx.rm <- which(colnames(idf) %in% col.rm)
    idf <- idf[, -idx.rm]
    return(idf)
}

training <- filterData(training)
finalTrainingDS <- training
dim(finalTrainingDS)

# Now let's perform the same cleaning process to the testing dataset as
well
nzvCol <- nearZeroVar(testing)
testing <- testing[,-nzvCol]
testing <- filterData(testing)
finalTestingDS <- testing
dim(finalTestingDS)
```

## Data Partitioning
Now we'll partition the data into training and testing datasets.
```{r}
inTrain <- createDataPartition(y=finalTrainingDS$classe, p=0.85,
list=FALSE)
~~~

~~~~
newTraining <- finalTrainingDS[inTrain, ]
newTesting <- finalTrainingDS[-inTrain, ]
dim(newTraining); dim(newTesting)
```

## Model Selection
We'll run some simulations with different machine learning algoritms.
We'll use Random Forest, Decision Trees, Naive Bayes, Linear Discrimant
Analysis and Generalized Boosted Regression Model. Besides this we'll
check if using Principal Component Analysis also generates a good basis
for prediction.

. Note: from the referenced paper we know the AdaBoost algo yields 99,9%
accuracy. For this work we'll consider "good" anything higher than 98%.

## Training Control Parameters
```{r}
#Some fitting params - Repeated 5 Cross Validations
fitControl <- trainControl(method="repeatedcv", number=5, repeats=1,
verboseIter=FALSE)
```

## Predicting models with PCA pre-processing
```{r, cache=TRUE, warning=FALSE, message=FALSE, echo=FALSE }
prePro <- preProcess(newTraining[,-53], method="pca", tresh=0.99)
newTrainingPCA <- predict(prePro,newTraining[,-53]) ; newTestingPCA <-
predict(prePro,newTesting[,-53])

# Decision Trees
modelFitTreePCA <- train(newTraining$classe ~ . , method="rpart",
trControl=fitControl, data=newTrainingPCA)
cm_tree_pca <- confusionMatrix(newTesting$classe,
predict(modelFitTreePCA, newdata=newTestingPCA))

# Linear Discriminant Analysis
modelFitLDAPCA <- train(newTraining$classe ~ . , method="lda",
trControl=fitControl, data=newTrainingPCA)
cm_lda_pca <- confusionMatrix(newTesting$classe, predict(modelFitLDAPCA,
newdata=newTestingPCA))

# Naive Bayes
modelFitNBPCA <- train(newTraining$classe ~ . , method="nb",
trControl=fitControl, data=newTrainingPCA)
cm_nb_pca <- confusionMatrix(newTesting$classe, predict(modelFitNBPCA,
newdata=newTestingPCA))

# Random Forest
modelFitRFPCA <- train(newTraining$classe ~ . , method="rf",
data=newTrainingPCA)
cm_rf_pca <- confusionMatrix(newTesting$classe, predict(modelFitRFPCA,
newdata=newTestingPCA))

# Generalized Boosted Regression Modeling
modelFitGBMPCA <- train(newTraining$classe ~ . , method="gbm",
trControl=fitControl, data=newTrainingPCA)
cm_gbm_pca <- confusionMatrix(newTesting$classe, predict(modelFitGBMPCA,
newdata=newTestingPCA))
~~~~

~~~
```

## Predicting Models without PCA
```{r, cache=TRUE, warning=FALSE, message=FALSE, echo=FALSE }
# Decision Trees
modelFitTree <- train(classe ~ . , method="rpart", data=newTraining,
trControl=fitControl)
cm_tree <- confusionMatrix(newTesting$classe, predict(modelFitTree,
newdata=newTesting))

# Linear Discriminant Analysis
modelFitLDA <- train(classe ~ ., method="lda", data=newTraining,
trControl=fitControl)
cm_lda <- confusionMatrix(newTesting$classe, predict(modelFitLDA,
newdata=newTesting))

#Naive Bayes
modelFitNB <- train(classe ~ ., method="nb", data=newTraining,
trControl=fitControl)
cm_nb <- confusionMatrix(newTesting$classe, predict(modelFitNB,
newdata=newTesting))

#Random Forest
modelFitRF <- train(classe ~ . , method="rf", data=newTraining)
cm_rf <- confusionMatrix(newTesting$classe, predict(modelFitRF,
newdata=newTesting))

#Generalized Boosted Regression Modeling
modelFitGBM <- train(classe ~ ., method="gbm", data=newTraining,
trControl=fitControl)
cm_gbm <- confusionMatrix(newTesting$classe, predict(modelFitGBM,
newdata=newTesting))
```

For the sake of comparison, we can see the overall indicadors for each
prediction model in the table below:
```{r}
# Analyzing the results
resultingDataTable <- rbind(cm_tree$overall, cm_tree_pca$overall,
cm_lda$overall, cm_lda_pca$overall,
                       cm_nb$overall, cm_nb_pca$overall, cm_rf$overall,
cm_rf_pca$overall,
                       cm_gbm$overall, cm_gbm_pca$overall)
rownames(resultingDataTable) <- c("Tree", "Tree w/ PCA", "LDA", "LDA w/
PCA",
                             "Naive Baeyes", "Naive Bayes w/ PCA",
"Random Forest", "Random Forest w/ PCA",
                             "GBM", "GBM w/ PCA")

resultingDataTable
```

## Out-Of-Sample Error
Our out-of-sample error can be found using the (1 - Testing Accurary), as
evaluated below (for Random Forest Algo).
```{r}
oos_error <- 1 - cm_rf$overall[1]
~~~

~~~~
print(paste("Out of Error Sample is: ", oos_error * 100, "%"))
```

# Conclusion
From the resulting table we can see the Random Forest algorithm yields a
better result, and that using Principal Component Analysis, with a
threshold of 99% of the variance would descrease of accuracy in about
(aprox) 2% points.
Besides this, comparing our results with the original paper results, we
can see we have reach a very good prediction accuracy using Random Forest
algorithm.

# Appendix
## Variable Importance
Just for sake of curiosity, lets check each variable importance, in the
final model (random forest)
```{r}
varImp(modelFitRF)
```

# Generating files to submitt
Now we'll use the original testing dataset and the best model for
predicting the values.
```{r}
# This uses the code supplied by the class instructions
answers <- predict(modelFitRF, newdata=testing)
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)

    }
}
write_files(answers)
```
~~~~