````
---
title: "Human Activity Recognition"
author: "Francisco J. Valverde-Albacete"
date: "23 de enero de 2015"
output: html_document
---

# Executive Summary

We studied classification trees (CT), bagged CT, boosted CT, random
forests, and linear discriminant analysis on a task of multiclass weight
lifting classification.

The models were all fitted by 5-fold cross validation and some of them
had their own inherent resampling features. All other parameters were set
to the default provided by the caret package (v. 6.0-41).

The bagged/boosted models were signiticatively superior to the non-
bagged/boosted ones. The differences between the best two are not
significant. We choose a bagged tree model for predictions.

# Development

## Prelude: data loading and transformation

```{r message=FALSE}
library(plyr);library(dplyr); library(ggplot2); library(caret)
# Load the data
training <- read.csv("pml-training.csv",
                     header=TRUE,
                     na.strings=c("NA","", "#DIV/0!")
                     )
testing <- read.csv("pml-testing.csv",
                     header=TRUE,
                     na.strings=c("NA","", "#DIV/0!")
                     )
dim(training)
dim(testing)
```

There is a number of problems with these data:

1. Some columns should not be used for prediction, since they are
arbitrary, like observation numbers, timestamps, user names

```{r}
deletedColumns <- c("X", "user_name", "new_window", names(training)
[grep("timestamp", names(training))])
```

2. There is an enormous proportion of NA values. They are differently
encoded as: "NA", "#DIV/0!" or simply absent ("") in the original data.
We delete all of those columns which are all NAs, or have zero variance.

Note that we also observe *some data* in the test set, only to discard
it! That is,  the retained information in the test set has still not been
seen in model development.
````

```{r}
print(sprintf("The proportion of NA in training values is rather high:
%0.2f",
    sum(is.na(training))/(prod(dim(training)))))
print(sprintf("but not so for testing: %0.2f",
             sum(is.na(testing))/(prod(dim(training)))))
#Some columns are completely filled with NAs
deletedColumns <- union(
    union(deletedColumns,
          names(which(unlist(lapply(testing,
function(x)all(is.na(x)))))),
    union(
        names(which(unlist(lapply(training, function(x)all(is.na(x)))))),
        names(which(unlist(lapply(training,
                           function(x)is.numeric(x) & var(x,
na.rm=TRUE)==0.0))))
        )
    )

```

3. There may be data-collection artifacts: combinations of predictors which imply the classe or variables with zero variance

<!--
```{r}
#table(training$classe,training$user_name)
#There should be no correlation between user_name and classe: might as well ignore it, to prevent learning for these particular users.
# Note: this is NOT the setting used in the original paper.
deletedColumns <- union(deletedColumns, "user_name")
```
-->

We retain only those columns not marked for deletion

```{r}
preTraining <- dplyr::select(training, -one_of(deletedColumns))
preTesting <- dplyr::select(testing, -one_of(deletedColumns))
dim(preTraining)
dim(preTesting)
```

Preliminary analysis and biases

1. The training set is slightly unbalanced towards class A (well-done exercises.)

```{r}
# Have a look at the distribution of the classes in the training set.
summary(preTraining$classe)
```

2. Since the data set is still to big for the more demanding techniques,

we subsample:

```{r}
selected <- createDataPartition(y=preTraining$classe, p=0.3, list=FALSE)
preTraining <- preTraining[selected, ]
dim(preTraining)
```

3. We tried some preprocessing in earlier versions, but this provided no advantages, so we have avoided it in the final version.

<!--
The preprocessing has to be common: extracted from training, then applied to both training and testing. Since all the columns left are numeric, no problem is bound to appear in the following centering, scaling and imputation.

CAVEAT: this does not seem to provide any advantage.

```{r}
last <- which(names(preTraining)=="classe")
preObj <- preProcess(preTraining[,-last],
                     method=c("center", "scale", "knnImpute"))
preTraining <- predict(preObj, preTraining[,-last]) %>%
    mutate(classe=preTraining$classe)
preTesting <- predict(preObj, preTesting)
#These warnings are OK, since the preprocessing in the testing data is *blind*.
```
-->

Exploratory Analysis: Model construction

In prediction exploratory analysis takes the form of fitting several different models to the task and then choosing one based in the performance on the training set.

For those that are not meta-models (e.g. do not include bagging or boosting in their algorithm) we use 10-fold cross-validation to estimate the oob error:

```{r}
numMethods <- 6
n <- 1
methodsNames <- replicate(numMethods,"")
accuracies <- replicate(numMethods,0.0)
accSd <- replicate(numMethods,0.0)
# We are using repeated cross-validation
trControl <- trainControl(
    method="cv", #repeatedc #too costly
    number=5, # 5-fold cv
#    classProbs=TRUE # Provide class probabilities
    )
```

Classification Trees

1. CART trees: plain, bagged and boosted.

```{r}
# set seed to the same, always
set.seed(1235)
library(rpart)
# Use CART for starters... It's too complicated to start doing feature
selection.
CARTfit <- train(classe ~ ., data=preTraining,
                  method="rpart",
                  trControl=trControl
                  )
#CARTfit
methodsNames[n] <- "ctree"
accuracies[n] <- CARTfit$results[1,2]
accSd[n] <- CARTfit$results[1,4]
n <- n + 1
#Bagged trees
library(ipred)
set.seed(1235)
BaggedCARTfit <- train(classe ~ ., data=preTraining,
                       method="treebag",
                       trControl=trControl
                       )
#BaggedCARTfit
methodsNames[n] <- "Bagged ctree"
accuracies[n] <- BaggedCARTfit$results[1,2]
accSd[n] <- BaggedCARTfit$results[1,4]
n <- n + 1
#Boosted CART trees
set.seed(1235)
library(survival); library(splines); library(gbm)
BoostedCARTfit <-
    train(classe ~ ., data=preTraining,
          method="gbm",
          trControl=trControl,
          verbose=FALSE #  for gbm
          )
methodsNames[n] <- "Boosted ctree"
accuracies[n] <- BoostedCARTfit$results[9,4]
accSd[n] <- BoostedCARTfit$results[9,6]
n <- n + 1
```

We also try trees with bagging under the guise of a random forest:

```{r}
library(randomForest)
set.seed(1235)
rfFit <- train(classe ~ .,
               data=preTraining,
               method="rf",
               trControl=trControl, # Same one since it is only cv.
               prox=TRUE, allowParallel=TRUE,
               verbose=TRUE
```

```
               )
methodsNames[n] <- "rf"
accuracies[n] <- rfFit$results[1,2]
accSd[n] <- rfFit$results[1,4]
n <- n + 1
```


Model-based prediction

For comparison we usa a plain linear discriminant analysis.

```{r}
# Linear discriminant analysis
library(MASS)
set.seed(1235)
ldaFit <- train(classe ~ .,
                data=preTraining,
                method="lda",
                trControl=trControl # only cv.
                )
#ldaFit
methodsNames[n] <- "lda"
accuracies[n] <- ldaFit$results[1,2]
accSd[n] <- ldaFit$results[1,4]
#n <- n + 1
# Naive Bayes
# library(klaR)
# set.seed(1235)
# nbFit <- train(classe ~ .,
#                 data=preTraining,
#                 method="nb",
#                 trControl=trControl # only cv.
#                 )
# #nbFit
# methodsNames[n] <- "nb"
# accuracies[n] <- nbFit$results[2,3]
# accSd[n] <- nbFit$results[2,5]
#n <- n + 1
```


Final exploration

We plot the accuracies vs the methods:

```{r}
results <- data.frame(methodsNames,acc=round(100*accuracies,2),
accSd=round(100*accSd,2))
results <- results[1:n, ]
results$methodsNames <- with(results, reorder(methodsNames,acc))
ggplot(results, aes(x=methodsNames, y=acc)) + ylim(0,100.0)  +
    geom_bar(stat="identity", fill="white", colour="black") +
    geom_text(aes(label=acc), vjust=3.5) +
    geom_errorbar(aes(ymin=acc-accSd, ymax=acc+accSd), width=.2)
```

Prediction

In predictive analysis, the last step is predicting and reporting the predictions. Note that there can be no assessment of these predictions!

```{r}
# We choose the highest accuracy, although this is not authoritative,
given the variances.
best <- which.max(results$acc)  # Decide on the best method to predic
print(sprintf("The best accuracy is attained by %s",
results$methodsNames[best]))
modelFit <- BaggedCARTfit
modelFit
modelFit$finalModel
predictions <- predict(modelFit, preTesting)  # predictions for
submission
predictions
# write the solutions
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(predictions)
```