# PML

Shengbai Zhang

8/9/2021

# Synopsis

“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.” - Coursera

The aim of this project is to predict the manner in which they did the exercise (“classe” variable in the training set). This report describes the building and testing of the different models. The best model will then be used to predict 20 different test cases.

**Data used in this project were provided by Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers’ Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6.**

# R libraries

```
library("caret")
```

```
## Warning: package 'caret' was built under R version 4.0.5
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library("rattle")
```

```
## Warning: package 'rattle' was built under R version 4.0.5
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library("randomForest")
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
##
##     importance
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library("corrplot")
```

```
## corrplot 0.90 loaded
```

```
library("gbm")
```

```
## Warning: package 'gbm' was built under R version 4.0.5
## Loaded gbm 2.1.8
```

# Preparation of the data

## Loading

```
#Training set
trainData <- read.csv("C:/Users/celin/Desktop/MachineLearning/pml-training.csv", head
er=T)
#Testing set
testData <- read.csv("C:/Users/celin/Desktop/MachineLearning/pml-testing.csv", header
=T)
#Dimensions of training set
dim(trainData)
```

```
## [1] 19622   160
```

```
#Dimensions of testing set
dim(testData)
```

```
## [1]  20 160
```

The training set is composed of 19622 observations and 160 variables whereas the testing set contains 20 observations.

## Cleaning

We remove variables that contains missing values (NA or ""):

```
#Training set
trainData <- trainData[,colSums(is.na(trainData)) == 0]
trainData <- trainData[,colSums(trainData == "") == 0]
#Testing set
testData <- testData[,colSums(is.na(testData)) == 0]
testData <- testData[,colSums(testData == "") == 0]
#New dimensions of training set
dim(trainData)
```

```
## [1] 19622    60
```

```
#New dimensions of testing set
dim(testData)
```

```
## [1] 20 60
```

We also remove the first column (X = ID) as this will not be informative

```
trainData$X <- NULL
testData$X <- NULL
```

# Preparation

We split the training data such as 70% will be used as training set and 30% will be used as test set: it will be used to compute the out-of-samples errors. The loaded testing set (pml-testing.csv) is renamed as validTestData and will be used later on for the 20 cases.

```
#Rename the loaded testing set testData -> validTestData & remove the testData
validTestData <- testData
rm(testData)

set.seed(42)
partitionTrainData <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
trainData <- trainData[partitionTrainData, ]
testData <- trainData[-partitionTrainData, ]
#Dimension of the training set (obtained after partitioning)
dim(trainData)
```

```
## [1] 13737    59
```

```
#Dimensions of the testing set (obtained after partitioning)
dim(testData)
```

```
## [1] 4129   59
```

Remove the variables having a near zero variance before performing the training & testing steps

```
nzv <- nearZeroVar(trainData)
trainData <- trainData[, -nzv]
testData  <- testData[, -nzv]
#Dimensions of the training set ready
dim(trainData)
```

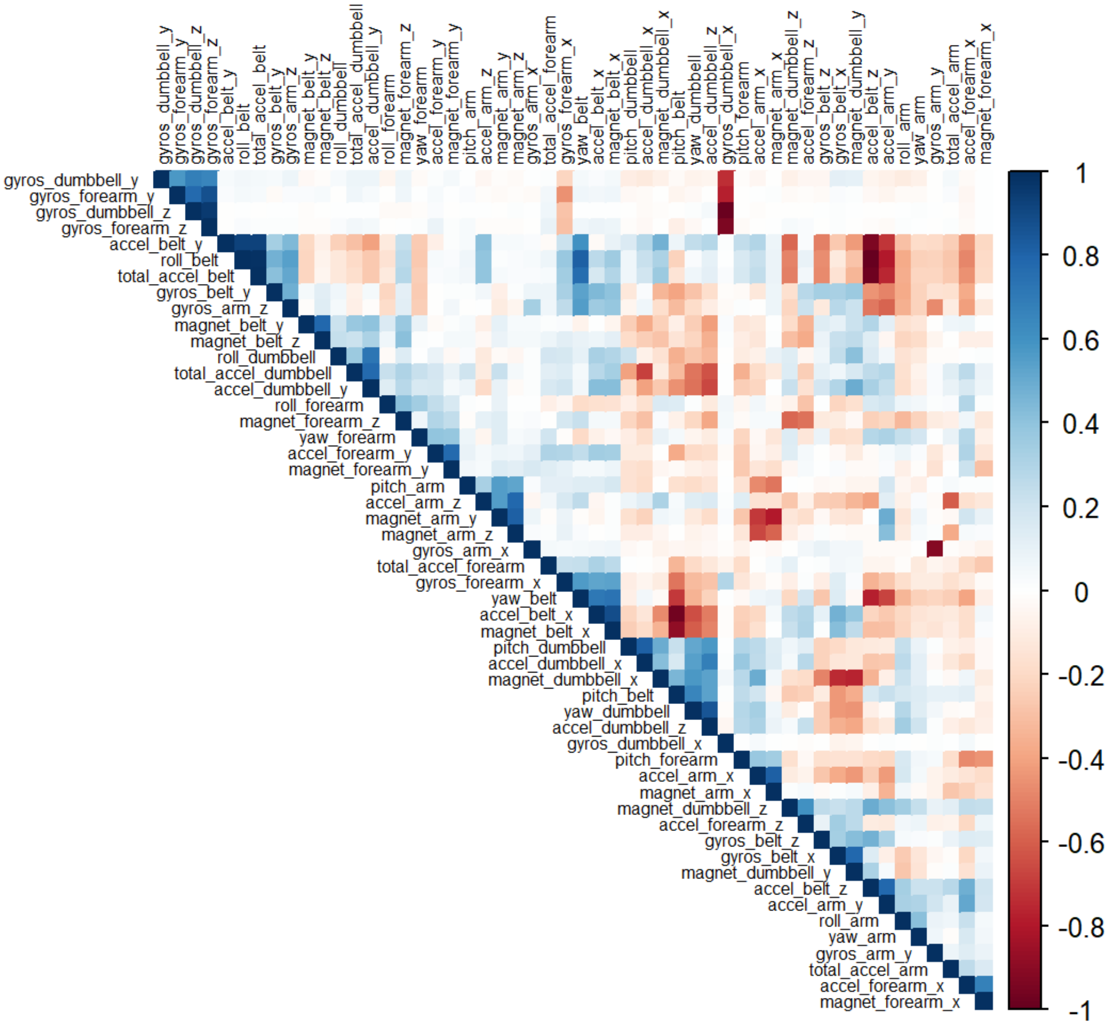
```
## [1] 13737    58
```

```
#Dimensions of the testing set ready
dim(testData)
```

```
## [1] 4129   58
```

After all these QC steps, we now have 58 variables remaining. The correlation between numerical variables can be observed in the following corrplot:

```
#Remove non numerical variables
corMtx <- cor(trainData[,-c(1,2,3,4,5,58)])
#Plot
corrplot(corMtx, method = "color", type = "upper", order = "hclust", tl.cex = 0.5,tl.
col="black")
```



We can observe that several variables are highly correlated (dark red = strong positive correlation, dark blue = strong negative correlation).

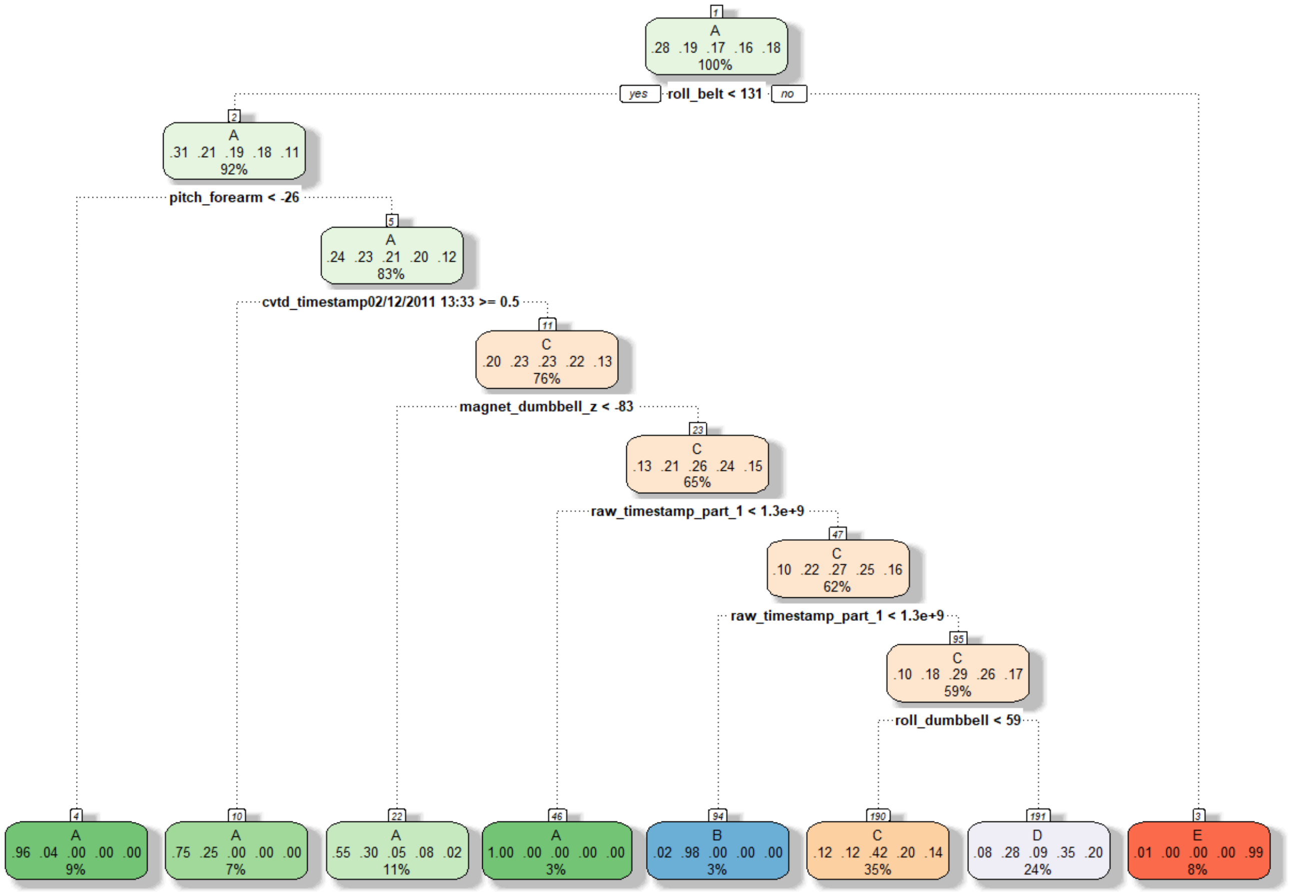
# Models

We will build the model for predictions using three different algorithms studied during the course: (1) decision tree, (2) random forest and (3) generalized boosted model.

# Decision tree

## Building

```
set.seed(1234)
#3-fold cross validation
ctrlTreeModel <- trainControl(method="cv", number=3)
decisionTreeModel <- train(classe ~ ., data=trainData, method="rpart", trControl=ctrl
TreeModel)
#Visualisation of the constructed decision tree
fancyRpartPlot(decisionTreeModel$finalModel)
```



Rattle 2021-Jul-25 11:39:37 celin

# Predictions

```
predTreeModel <- predict(decisionTreeModel, testData)
confMatTree <- confusionMatrix(predTreeModel, as.factor(testData$classe))
print(confMatTree)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   A   B   C   D   E
##          A 880 191  12  34   9
##          B   2 130   1   0   0
##          C 188 180 625 300 202
##          D  88 305  89 338 196
##          E   2   0   0   0 357
## 
## Overall Statistics
## 
##                Accuracy : 0.5643
##                  95% CI : (0.549, 0.5795)
##     No Information Rate : 0.2809
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.4535
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7586  0.16129   0.8597  0.50298  0.46728
## Specificity            0.9171  0.99910   0.7443  0.80388  0.99941
## Pos Pred Value         0.7815  0.97744   0.4181  0.33268  0.99443
## Neg Pred Value         0.9068  0.83083   0.9613  0.89271  0.89204
## Prevalence             0.2809  0.19520   0.1761  0.16275  0.18503
## Detection Rate         0.2131  0.03148   0.1514  0.08186  0.08646
## Detection Prevalence   0.2727  0.03221   0.3621  0.24606  0.08695
## Balanced Accuracy      0.8379  0.58019   0.8020  0.65343  0.73334
```

The **accuracy of the model** is poor and equal to **0.5643** and hence a **out-of-sample error** of 1-0.5643 = **0.4357**.

# Random forest

# Building

```
set.seed(1234)
#3-fold cross validation
rfCtrl <- trainControl(method="cv", number=3)
rfModel <- train(classe ~ ., data=trainData, method="rf", trControl=rfCtrl)
```

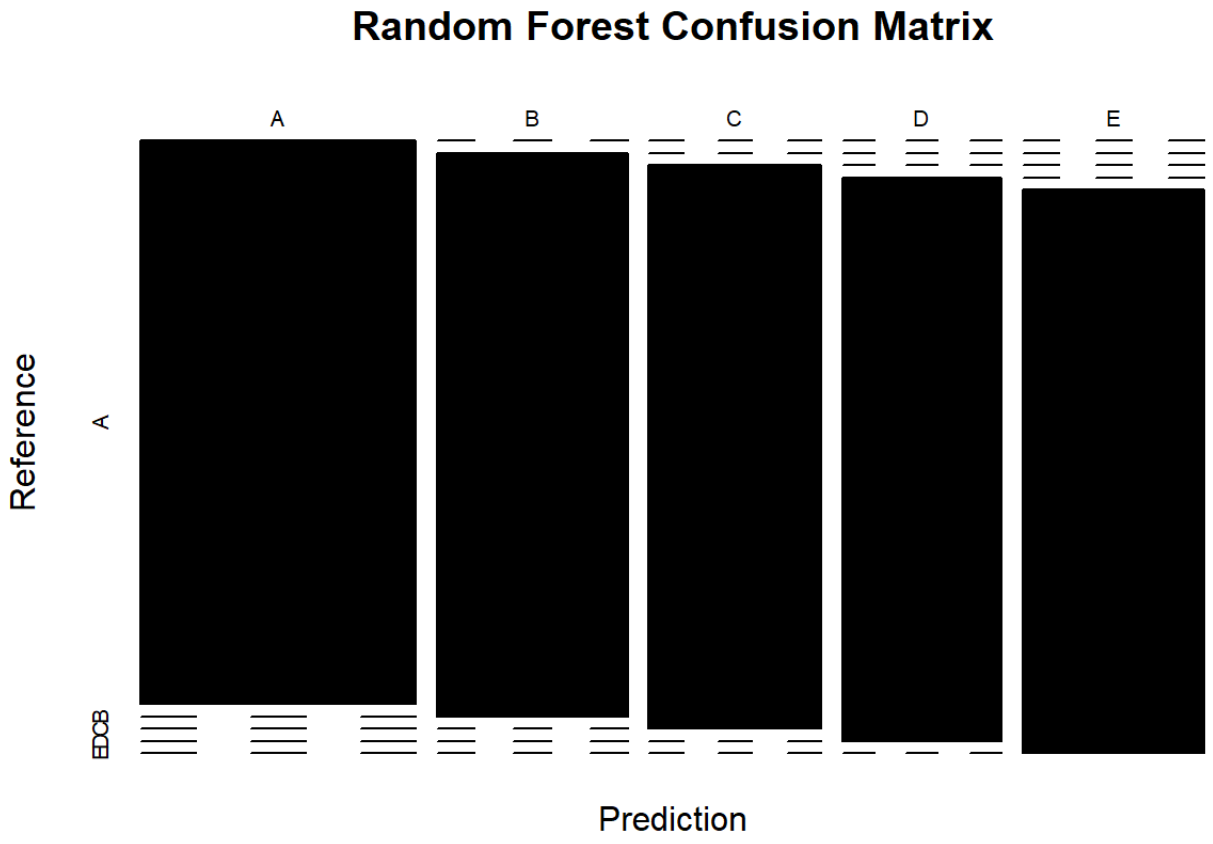
# Predictions

```
predRf <- predict(rfModel,testData)
confMatRf <- confusionMatrix(predRf, as.factor(testData$classe))
print(confMatRf)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1160    0    0    0    0
##          B    0  806    0    0    0
##          C    0    0  727    0    0
##          D    0    0    0  672    0
##          E    0    0    0    0  764
## 
## Overall Statistics
## 
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2809
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 1
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000    1.000
## Specificity            1.0000   1.0000   1.0000   1.0000    1.000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Prevalence             0.2809   0.1952   0.1761   0.1628    0.185
## Detection Rate         0.2809   0.1952   0.1761   0.1628    0.185
## Detection Prevalence   0.2809   0.1952   0.1761   0.1628    0.185
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000    1.000
```

The **accuracy of the RF model is 1 and hence the out-of-samples errors is 0**. However, this value is most likely due to an over-fitting of the model.

```
#Plot of the model performances
plot(confMatRf$table, col = confMatRf$byClass, main = "Random Forest Confusion Matrix
")
```



# Generalized boosted model

## Building

```
set.seed(1432)
#3-fold cross validation
gbmCtrl <- trainControl(method = "cv", number = 3)
gbmModel  <- train(classe ~ ., data=trainData, method = "gbm", trControl = gbmCtrl, v
erbose = FALSE)
print(gbmModel)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    57 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9158, 9157, 9159
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.8322795  0.7870363
##   1                  100      0.8991783  0.8723415
##   1                  150      0.9287325  0.9097151
##   2                   50      0.9565411  0.9449714
##   2                  100      0.9870422  0.9836070
##   2                  150      0.9925019  0.9905155
##   3                   50      0.9831841  0.9787260
##   3                  100      0.9935210  0.9918049
##   3                  150      0.9959962  0.9949357
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

## Predictions

```
predGbm <- predict(gbmModel,testData)
confMatGbm <- confusionMatrix(predGbm, as.factor(testData$classe))
print(confMatGbm)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1160    0    0    0    0
##          B    0  805    1    0    0
##          C    0    1  724    0    0
##          D    0    0    2  671    1
##          E    0    0    0    1  763
## 
## Overall Statistics
## 
##                Accuracy : 0.9985
##                  95% CI : (0.9968, 0.9995)
##     No Information Rate : 0.2809
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9982
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9988   0.9959   0.9985   0.9987
## Specificity            1.0000   0.9997   0.9997   0.9991   0.9997
## Pos Pred Value         1.0000   0.9988   0.9986   0.9955   0.9987
## Neg Pred Value         1.0000   0.9997   0.9991   0.9997   0.9997
## Prevalence             0.2809   0.1952   0.1761   0.1628   0.1850
## Detection Rate         0.2809   0.1950   0.1753   0.1625   0.1848
## Detection Prevalence   0.2809   0.1952   0.1756   0.1632   0.1850
## Balanced Accuracy      1.0000   0.9992   0.9978   0.9988   0.9992
```

The GBM model has an **accuracy of 0.9985 and hence an out-of-samples errors of 0.0025**.

# Selection of the best model for the assignement

The random forest showed an accuracy of 1 that might be due to over-fitting but it is followed closely by GBM with an accuracy of almost 1. We will then **select the random forest model** as the best one to use for the assignment on the 20 cases.

We finally apply the RF model on the original testing set (validTestData) to get the predictions:

```
finalPredictions <- predict(rfModel,validTestData)
print(finalPredictions)
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```