# Prediction Assignment Writeup

Saja

### Introduction

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 types 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

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

The test data are available here:

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

The data for this project come from this source: http://web.archive.org/web/20161224072740/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.

### Loading the data

First let's load the data and setup our workspace.

```r
library(tidyverse)
library(caret)
library(lattice)
library(kernlab)
library(rattle)
library(corrplot)
set.seed(12321)
```

```r
train_data <- read_csv("./data/pml-training.csv")
test_data <- read_csv("./data/pml-testing.csv")
```

```r
dim(train_data)
```

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

```
# names(train_data)
sum(is.na(train_data)) / (dim(train_data)[1] * dim(train_data)[2])
```

```
## [1] 0.6120725
```

More than 60% of our data consists of missing values. We'll drop the columns which more than 95% of missing values in them.

```
dropCols <- colSums(is.na(train_data)) > 0.95 * dim(train_data)[1]
train_data <- train_data[, -c(1:dim(train_data)[2]) * dropCols]
```

## Train and Cross-Validation Data

Now let's seperat the output from input. And also split our data into two chunks, one for training and one for cross validation.

```
# lets save the output y in a seperate dataframe
out_vector <- c("...1", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp",
y <- train_data %>%
  select(out_vector)

y$classe <- as.factor(train_data$classe)

train_data <- train_data %>%
  select(-out_vector)

index <- createDataPartition(y$classe, p = 0.7, list = F)
cv_data <- train_data[-index, ]
cv_y <- y[-index, ]
train_data <- train_data[index, ]
y <- y[index, ]
```

## Training

Now I'm going to test different kinds of modelling for the data.

### LDA

```
fitModel_lda <- train(
  x = train_data,
  y = y$classe,
  preProcess = c("center", "scale"),
  method = "lda"
)

lda_predict <- predict(fitModel_lda, cv_data)

cfmtrx_lda <- confusionMatrix(cv_y$classe, lda_predict)
cfmtrx_lda$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   7.055225e-01   6.272846e-01   6.936922e-01   7.171487e-01   2.970263e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   2.704038e-66
```

### SVM

```r
fitModel_svm <- train(
  x = train_data,
  y = y$classe,
  method = "svmLinear",
  preProcess = c("center", "scale"),
  trControl = trainControl(method = "cv", number = 3, verboseIter = F),
  tuneLength = 5,
  verbose = F
)

svm_predict <- predict(fitModel_svm, cv_data)

cfmtrx_svm <- confusionMatrix(cv_y$classe, svm_predict)
cfmtrx_svm$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   7.833475e-01   7.246084e-01   7.725981e-01   7.938153e-01   3.213254e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   2.975472e-47
```

### Random Forrest

```r
fitModel_rf <- train(
  x = train_data,
  y = y$classe,
  method = "rf",
  preProcess = c("center", "scale"),
  trControl = trainControl(method = "cv", number = 3, verboseIter = F),
  tuneLength = 5,
  verbose = F
)

rf_predict <- predict(fitModel_rf, cv_data)

cfmtrx_rf <- confusionMatrix(cv_y$classe, rf_predict)
cfmtrx_rf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9935429      0.9918322      0.9911478      0.9954266      0.2847918
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

## GBM

```
fitModel_gbm <- train(
  x = train_data,
  y = y$classe,
  method = "gbm",
  preProcess = c("center", "scale"),
  trControl = trainControl(method = "cv", number = 3, verboseIter = F),
  tuneLength = 1,
  verbose = F
)

gbm_predict <- predict(fitModel_gbm, cv_data)

cfmtrx_gbm <- confusionMatrix(cv_y$classe, gbm_predict)
cfmtrx_gbm$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   7.519116e-01   6.854026e-01   7.406687e-01   7.629043e-01   3.090909e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   2.473552e-71
```

Thne we can create a mix predictor based on previous predictions.

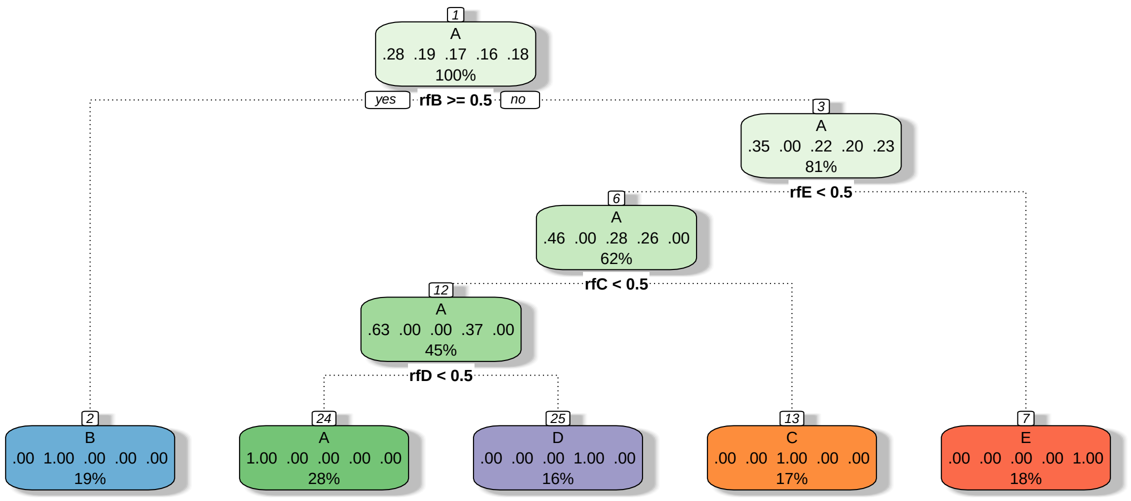
## Mixed

```
mixed_pred_data <- data.frame(
  lda = predict(fitModel_lda),
  svm = predict(fitModel_svm),
  rf = predict(fitModel_rf),
  gbm = predict(fitModel_gbm),
  classe = y$classe
)

fitModel_mix <- train(
  classe ~ .,
  data = mixed_pred_data,
  method = "rpart",
  tuneLength = 7,
)

mixed_cv_data <- data.frame(
  lda = lda_predict,
  svm = svm_predict,
  rf = rf_predict,
  gbm = gbm_predict,
  classe = cv_y$classe
)

mixed_predict <- predict(fitModel_mix, mixed_cv_data)
```

```
cfmtrx_mix <- confusionMatrix(mixed_cv_data$classe, mixed_predict)
cfmtrx_mix$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9935429      0.9918322      0.9911478      0.9954266      0.2847918
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
fancyRpartPlot(fitModel_mix$finalModel)
```



Rattle 2022−Jun−29 09:37:17 saja

By this diagram we can conclude that the fix model is exactly behaving like Random Forrest prediction. So I'll pick that model to predict my test set.

### Predict Test set

We can now use the predictor on the test data.

```
test_pred <- predict(fitModel_rf, test_data)

print(test_pred)
```

```
##  [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
```