Practical Machine Learning Assignment

Wenjing Liu

November 7, 2018

1. Introduction

1.1. 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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

1.2. Data

The training data for this project are available here:

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

The test data are available here:

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

1.3. Task

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.

2. Data Analysis

I went though the following steps to finish the task:

1. Upload training data and store it in variable **training**, testing data in **testing**;

2. Randomly divide training data into 2 parts: 75% as sub training data and store it in variable **train**, 25% as sub testing data and store it in **test**;
3. Sample training data 3 times more and each time choose 25% as sub testing data;
4. Run different models and test the accuracy with 4 sub testing data sets:

- a. Random Forest with Principal Component Analysis
- b. Random Forest without PCA
- c. Classification and Regression Tree with PCA
- d. Classification (default rpart())
- e. Classification (rpart method="class")
- f. Boosted Tree with PCA (Without PCA it would be extremely slow.)

5. Compare the accuracy and time to choose a model:

- Accuracy (high to low): b > a > f > e > d > c
- Speed (high to low): c > d > e > a > b > f

6. Random Forest has the most accuracy, and the speed is moderate. So I chose it to apply on the testing data.

2.1 Data Uploading and Cleansing

```
set.seed(55555)
library(caret)
library(AppliedPredictiveModeling)
library(randomForest) ## for randomForest()
library(rattle) ## for fancyRpartPlot()
library(rpart) ## for rpart()
library(rpart.plot) ## for rpart.plot()

## Upload data
training = read.csv("D:/R/data/pml-training.csv",
header = TRUE,
stringsAsFactors=FALSE,
na.strings=c("NA", "N/A"))
dim(training) ## 19622 rows, 160 columns
```

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

```
summary(training[,c(2:7, 160)]) ## There are 6 users, 5 classes.
```

```
##   user_name         raw_timestamp_part_1 raw_timestamp_part_2
##  Length:19622       Min.   :1.322e+09    Min.   :   294      
##  Class :character   1st Qu.:1.323e+09    1st Qu.:252912      
##  Mode  :character   Median :1.323e+09    Median :496380      
##                     Mean   :1.323e+09    Mean   :500656      
##                     3rd Qu.:1.323e+09    3rd Qu.:751891      
##                     Max.   :1.323e+09    Max.   :998801      
##  cvtd_timestamp      new_window          num_window      classe         
##  Length:19622       Length:19622       Min.   :  1.0   Length:19622      
##  Class :character   Class :character   1st Qu.:222.0   Class :character  
##  Mode  :character   Mode  :character   Median :424.0   Mode  :character  
##                                        Mean   :430.6                     
##                                        3rd Qu.:644.0                     
##                                        Max.   :864.0
```

```
str(training) ## e.g. column "kurtosis_roll_belt", type chr, contains "NA"
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1
323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390
484323 484434 ...
##  $ cvtd_timestamp          : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23"
"05/12/2011 11:23" ...
##  $ new_window              : chr  "no" "no" "no" "no" ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -9
4.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : chr  "" "" "" "" ...
##  $ kurtosis_picth_belt     : chr  "" "" "" "" ...
##  $ kurtosis_yaw_belt       : chr  "" "" "" "" ...
##  $ skewness_roll_belt      : chr  "" "" "" "" ...
##  $ skewness_roll_belt.1    : chr  "" "" "" "" ...
##  $ skewness_yaw_belt       : chr  "" "" "" "" ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : chr  "" "" "" "" ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : chr  "" "" "" "" ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : chr  "" "" "" "" ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0
...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03
...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm       : chr  "" "" "" "" ...
##  $ kurtosis_picth_arm      : chr  "" "" "" "" ...
##  $ kurtosis_yaw_arm        : chr  "" "" "" "" ...
##  $ skewness_roll_arm       : chr  "" "" "" "" ...
##  $ skewness_pitch_arm      : chr  "" "" "" "" ...
##  $ skewness_yaw_arm        : chr  "" "" "" "" ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell  : chr  "" "" "" "" ...
##  $ kurtosis_picth_dumbbell : chr  "" "" "" "" ...
##  $ kurtosis_yaw_dumbbell   : chr  "" "" "" "" ...
##  $ skewness_roll_dumbbell  : chr  "" "" "" "" ...
##  $ skewness_pitch_dumbbell : chr  "" "" "" "" ...
##  $ skewness_yaw_dumbbell   : chr  "" "" "" "" ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : chr  "" "" "" "" ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : chr  "" "" "" "" ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

```
## Clean covariants
df <- training[,-c(1:7, 160)] ## drop the first 7 and the last columns
df <- suppressWarnings(data.frame(apply(df, 2, as.numeric))) ## convert all columns to numeri
c
df[is.na(df)] <- 0; sum(is.na(df)) ## replace <NA> with 0
```

```
## [1] 0
```

```
df <- df[vapply(df, function(x) length(unique(x)) > 1, logical(1L))] ## remove same value col
umns
dim(df) ## 19622 rows, 143 columns
```

```
## [1] 19622   143
```

```
training <- data.frame(classe=training$classe, df); rm(df)
training$classe <- as.factor(training$classe)
```

2.2 Data Partition

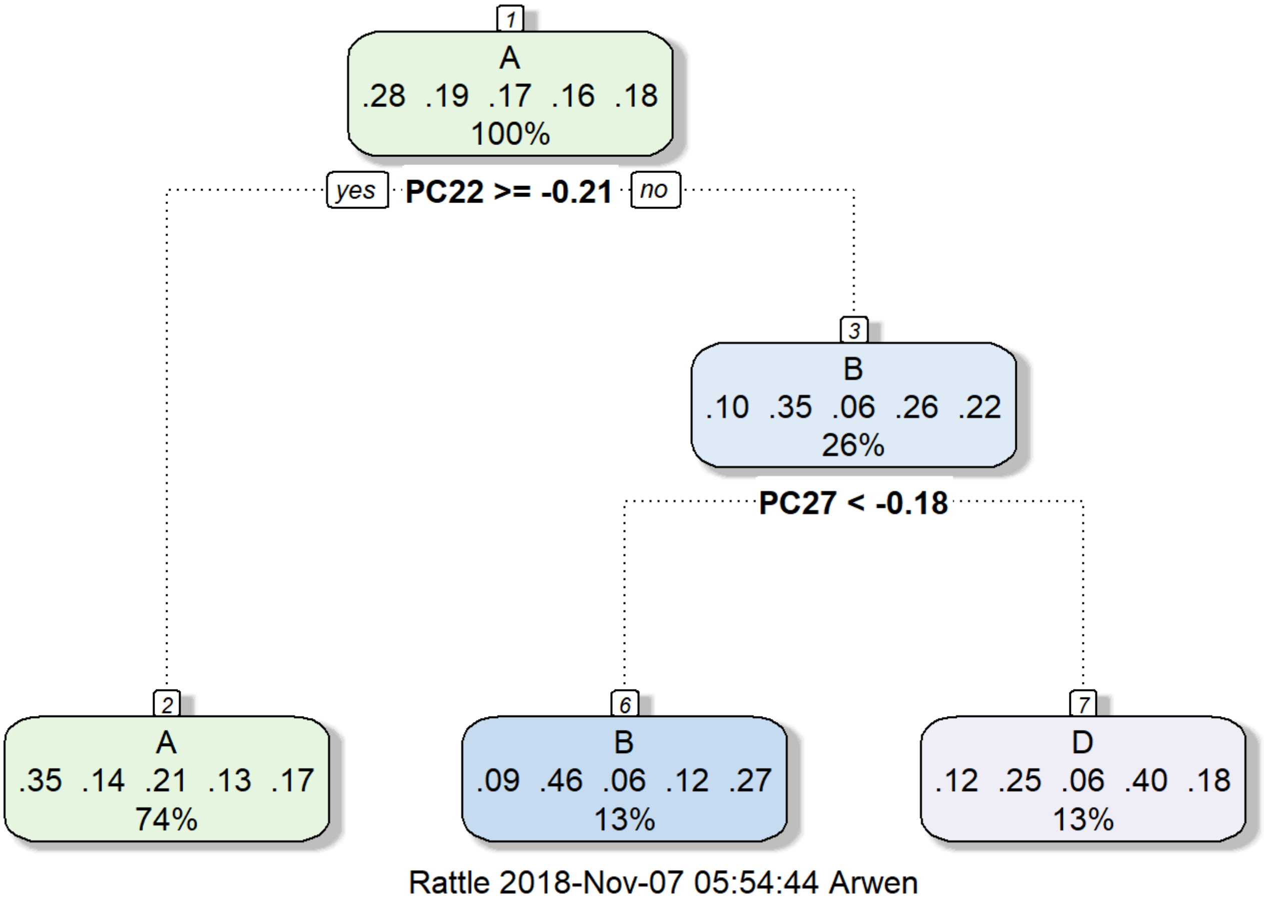
```
## Data partition
trainIndex = createDataPartition(training$classe, p = 3/4, list=FALSE)
train = training[trainIndex,]
library(rlist)
testIndices<- list(-trainIndex)
for (i in 2:4) {
testIndices <- list.append(testIndices, createDataPartition(training$classe, p = 1/4, list=FA
LSE))
}
```

2.3 Train Models

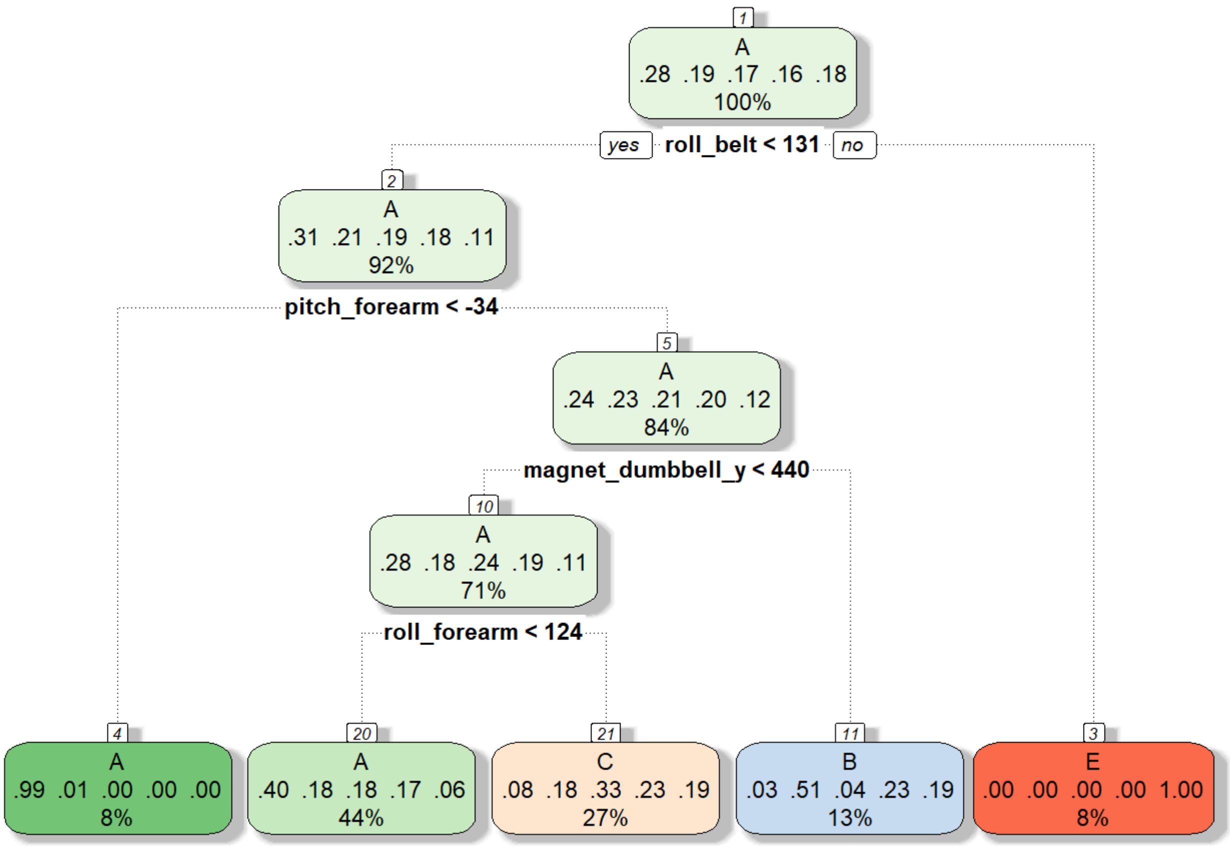
```
## Principal Component Analysis
preProc <- preProcess(train[,-1], method=c("center", "scale", "pca"), thresh=0.8); preProc
```

```
## Created from 14718 samples and 143 variables
## 
## Pre-processing:
##   - centered (143)
##   - ignored (0)
##   - principal component signal extraction (143)
##   - scaled (143)
## 
## PCA needed 28 components to capture 80 percent of the variance
```

```
trainPC <- predict(preProc, train)

########################################
## Random Forest
########################################
## RF + PCA
modFit <- randomForest(classe~., data=trainPC, method="class")

## Cross validation
accRF <- matrix(0, nrow=4, ncol=1)
for (i in 1:4) {
test = training[testIndices[[i]],]
testPC <- predict(preProc, test)
pred <- predict(modFit, testPC)
accRF[i,1] <- confusionMatrix(test$classe, pred)$overall["Accuracy"]
}; rm(modFit, test, testPC, pred)

## RF without PCA
accRF1 <- matrix(0, nrow=4, ncol=1)
modFit <- randomForest(classe~., data=train, method="class")
for (i in 1:4) {
test = training[testIndices[[i]],]
pred <- predict(modFit, test)
accRF1[i,1] <- confusionMatrix(test$classe, pred)$overall["Accuracy"]
}; rm(modFit, test, pred)

########################################
## Classification and Regression Trees
########################################
## rpart + PCA
modFit <- train(classe~., data=trainPC, method="rpart")
fancyRpartPlot(modFit$finalModel)
```

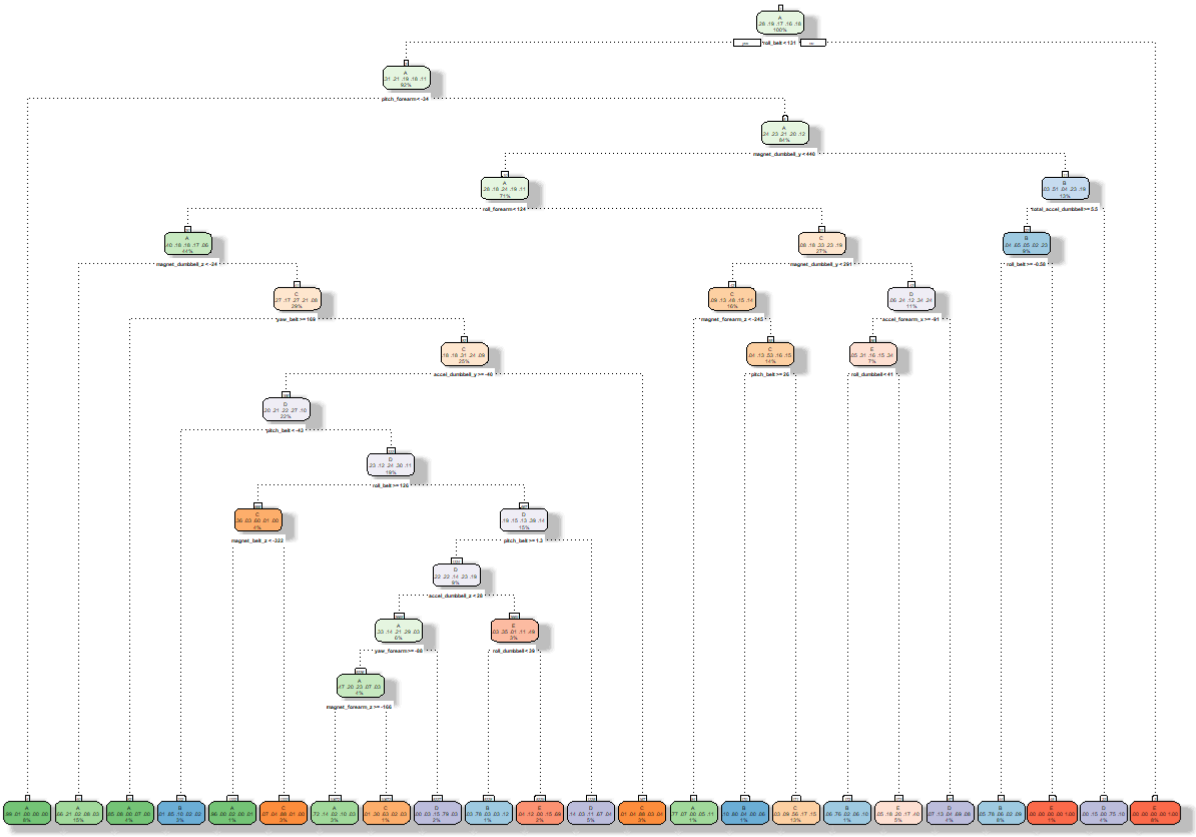


```r
accRP <- matrix(0, nrow=4, ncol=1)
for (i in 1:4) {
test = training[testIndices[[i]],]
testPC <- predict(preProc, test)
pred <- predict(modFit, testPC)
accRP[i,1] <- confusionMatrix(test$classe, pred)$overall["Accuracy"]
}; rm(modFit, test, testPC, pred)

## rpart(default)
modFit <- train(classe~., data=train, method="rpart")
fancyRpartPlot(modFit$finalModel)
```



Rattle 2018-Nov-07 05:55:08 Arwen

```
accRP1 <- matrix(0, nrow=4, ncol=1)
for (i in 1:4) {
test = training[testIndices[[i]],]
pred <- predict(modFit, test)
accRP1[i,1] <- confusionMatrix(test$classe, pred)$overall["Accuracy"]
}; rm(modFit, test, pred)

## rpart(class)
modFit <- rpart(classe~., data=train, method="class")
fancyRpartPlot(modFit)
```



Rattle 2018-Nov-07 05:55:13 Arwen

```
accRP2 <- matrix(0, nrow=4, ncol=1)
for (i in 1:4) {
test = training[testIndices[[i]],]
pred <- predict(modFit, test, type="class")
accRP2[i,1] <- confusionMatrix(test$classe, pred)$overall["Accuracy"]
}; rm(modFit, test, pred)


################################
## Boosted Trees + PCA (extremely slow)
################################
```

```
## Caution: Need to suppress output explicitly
modFit <- train(classe~., method="gbm", data=trainPC)
```

```
accGBM <- matrix(0, nrow=4, ncol=1)
for (i in 1:4) {
test = training[testIndices[[i]],]
testPC <- predict(preProc, test)
pred <- predict(modFit, testPC)
accGBM[i,1] <- confusionMatrix(test$classe, pred)$overall["Accuracy"]
}; rm(modFit, test, testPC, pred, i)
```

2.4 Compare Accuracy

```
## Compare Accuracy
acc <- data.frame(accRF, accRF1, accRP, accRP1, accRP2, accGBM)
names(acc) <- c("RF+PCA", "RF", "rpart+PCA", "rpart(default)", "rpart(class)", "gbm+PCA"); ac
c
```

```
##      RF+PCA        RF rpart+PCA rpart(default) rpart(class)   gbm+PCA
## 1 0.9524878 0.9928630 0.3621533      0.4957178    0.7349103 0.7555057
## 2 0.9902181 0.9973507 0.3641736      0.4870593    0.7413899 0.7915223
## 3 0.9881802 0.9989810 0.3664153      0.4974526    0.7491339 0.7845934
## 4 0.9904218 0.9977583 0.3664153      0.4917465    0.7428164 0.7929488
```

```
colMeans(acc)
```

```
##         RF+PCA             RF      rpart+PCA rpart(default)   rpart(class)
##      0.9803270      0.9967383      0.3647894      0.4929940      0.7420626
##        gbm+PCA
##      0.7811426
```

2.5 Predict

```
rm(training, trainIndex, testIndices, trainPC, preProc)

## Upload then clean testing data
testing = read.csv("D:/R/data/pml-testing.csv",
header = TRUE,
stringsAsFactors=FALSE,
na.strings=c("NA", "N/A"))
dim(testing) ## 20 rows, 160 columns
```

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

```
tail(names(testing)) ## The last column is "problem_id".
```

```
## [1] "accel_forearm_y"  "accel_forearm_z"  "magnet_forearm_x"
## [4] "magnet_forearm_y" "magnet_forearm_z" "problem_id"
```

```
df <- testing[,-c(1:7, 160)] ## drop the first 7 and the last columns
df <- suppressWarnings(data.frame(apply(df, 2, as.numeric))) ## convert all columns to numeri
c
df[is.na(df)] <- 0; ## replace <NA> with 0
testing <- data.frame(classe="", df); rm(df)

## Predict
modFit <- randomForest(classe~. , data=train, method="class")
pred <- predict(modFit, testing); pred
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```