# Practical Machine Learning Prediction Assignment

*Phil Sartor*

*12/18/2019*

## The goal of our project is to predict the manner in which 6 participants did certain exercises.

## Data is taken from accelerometers on the belt, forearm, arm, and dumbell.

## Participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

### Set working directory and load required libraries

```
knitr::opts_chunk$set(echo = TRUE)
library(data.table)
library(caret)
```

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

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

```
library(knitr)
library(xtable)
library(rpart)
library(randomForest)
```

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

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

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

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

```
library(rpart.plot)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

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

```r
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
rm(list = ls())
set.seed(54321)
setwd("/Users/psartor/Desktop/Data Management & Analytics/Practical Machine Learning/Week4")
```

### Load and explore the data

```r
# Load and read the training and test data into R using, read.csv function
training  <- read.csv("pml-training.csv", na.strings = c("NA","#DIV/0!",""))
testing <- read.csv("pml-testing.csv", na.strings = c("NA","DIV/0!",""))
dim(training)
```

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

```r
dim(testing)
```

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

```r
str(training)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 4844
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ 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            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ 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        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

**The training data set contains 19622 observations and 160 variables, while the testing data set contains 20 observations and 160 variables.**

**The "classe" variable in the training set is the outcome to predict.**

**There are 5 levels of classe "A", "B", "C", "D", "E".**

## Clean the data set

**As we can see there are many "NAs" in the dataset.**

**We removed any features that contained NA values.**

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
```

**Remove columns that do not contribute much to the accelerometer measurements.**

```
classe <- training$classe
trainRemove <- grepl("^X|timestamp|window", names(training))
training <- training[, !trainRemove]
trainCleaned <- training[, sapply(training, is.numeric)]
trainCleaned$classe <- classe
testRemove <- grepl("^X|timestamp|window", names(testing))
testing <- testing[, !testRemove]
testCleaned <- testing[, sapply(testing, is.numeric)]
dim(trainCleaned)
```

```
## [1] 19622    53
```

```
dim(testCleaned)
```

```
## [1] 20 53
```

**Look for near zero variance data as it can be advantageous to remove the variable from the model.**

```
nzv <- nearZeroVar(trainCleaned, saveMetrics = TRUE)
nzv
```

```
##                       freqRatio percentUnique zeroVar   nzv
## roll_belt              1.101904     6.7781062   FALSE FALSE
## pitch_belt             1.036082     9.3772296   FALSE FALSE
## yaw_belt               1.058480     9.9734991   FALSE FALSE
## total_accel_belt       1.063160     0.1477933   FALSE FALSE
## gyros_belt_x           1.058651     0.7134849   FALSE FALSE
## gyros_belt_y           1.144000     0.3516461   FALSE FALSE
## gyros_belt_z           1.066214     0.8612782   FALSE FALSE
## accel_belt_x           1.055412     0.8357966   FALSE FALSE
## accel_belt_y           1.113725     0.7287738   FALSE FALSE
## accel_belt_z           1.078767     1.5237998   FALSE FALSE
## magnet_belt_x          1.090141     1.6664968   FALSE FALSE
## magnet_belt_y          1.099688     1.5187035   FALSE FALSE
## magnet_belt_z          1.006369     2.3290184   FALSE FALSE
## roll_arm              52.338462    13.5256345   FALSE FALSE
## pitch_arm             87.256410    15.7323412   FALSE FALSE
## yaw_arm               33.029126    14.6570176   FALSE FALSE
## total_accel_arm        1.024526     0.3363572   FALSE FALSE
## gyros_arm_x            1.015504     3.2769341   FALSE FALSE
## gyros_arm_y            1.454369     1.9162165   FALSE FALSE
## gyros_arm_z            1.110687     1.2638875   FALSE FALSE
## accel_arm_x            1.017341     3.9598410   FALSE FALSE
## accel_arm_y            1.140187     2.7367241   FALSE FALSE
## accel_arm_z            1.128000     4.0362858   FALSE FALSE
## magnet_arm_x           1.000000     6.8239731   FALSE FALSE
## magnet_arm_y           1.056818     4.4439914   FALSE FALSE
## magnet_arm_z           1.036364     6.4468454   FALSE FALSE
## roll_dumbbell          1.022388    84.2065029   FALSE FALSE
## pitch_dumbbell         2.277372    81.7449801   FALSE FALSE
## yaw_dumbbell           1.132231    83.4828254   FALSE FALSE
## total_accel_dumbbell   1.072634     0.2191418   FALSE FALSE
## gyros_dumbbell_x       1.003268     1.2282132   FALSE FALSE
## gyros_dumbbell_y       1.264957     1.4167771   FALSE FALSE
## gyros_dumbbell_z       1.060100     1.0498420   FALSE FALSE
## accel_dumbbell_x       1.018018     2.1659362   FALSE FALSE
## accel_dumbbell_y       1.053061     2.3748853   FALSE FALSE
## accel_dumbbell_z       1.133333     2.0894914   FALSE FALSE
## magnet_dumbbell_x      1.098266     5.7486495   FALSE FALSE
## magnet_dumbbell_y      1.197740     4.3012945   FALSE FALSE
## magnet_dumbbell_z      1.020833     3.4451126   FALSE FALSE
## roll_forearm          11.589286    11.0895933   FALSE FALSE
## pitch_forearm         65.983051    14.8557741   FALSE FALSE
## yaw_forearm           15.322835    10.1467740   FALSE FALSE
## total_accel_forearm    1.128928     0.3567424   FALSE FALSE
## gyros_forearm_x        1.059273     1.5187035   FALSE FALSE
## gyros_forearm_y        1.036554     3.7763735   FALSE FALSE
## gyros_forearm_z        1.122917     1.5645704   FALSE FALSE
## accel_forearm_x        1.126437     4.0464784   FALSE FALSE
```

```
## accel_forearm_y       1.059406     5.1116094   FALSE FALSE
## accel_forearm_z       1.006250     2.9558659   FALSE FALSE
## magnet_forearm_x      1.012346     7.7667924   FALSE FALSE
## magnet_forearm_y      1.246914     9.5403119   FALSE FALSE
## magnet_forearm_z      1.000000     8.5771073   FALSE FALSE
## classe                1.469581     0.0254816   FALSE FALSE
```

## Partitioning the training set into two datasets

**Next, we can split the cleaned training set into a pure training data set (70%) and a validation data set (30%).**

**The training set is used to train or build the model.**

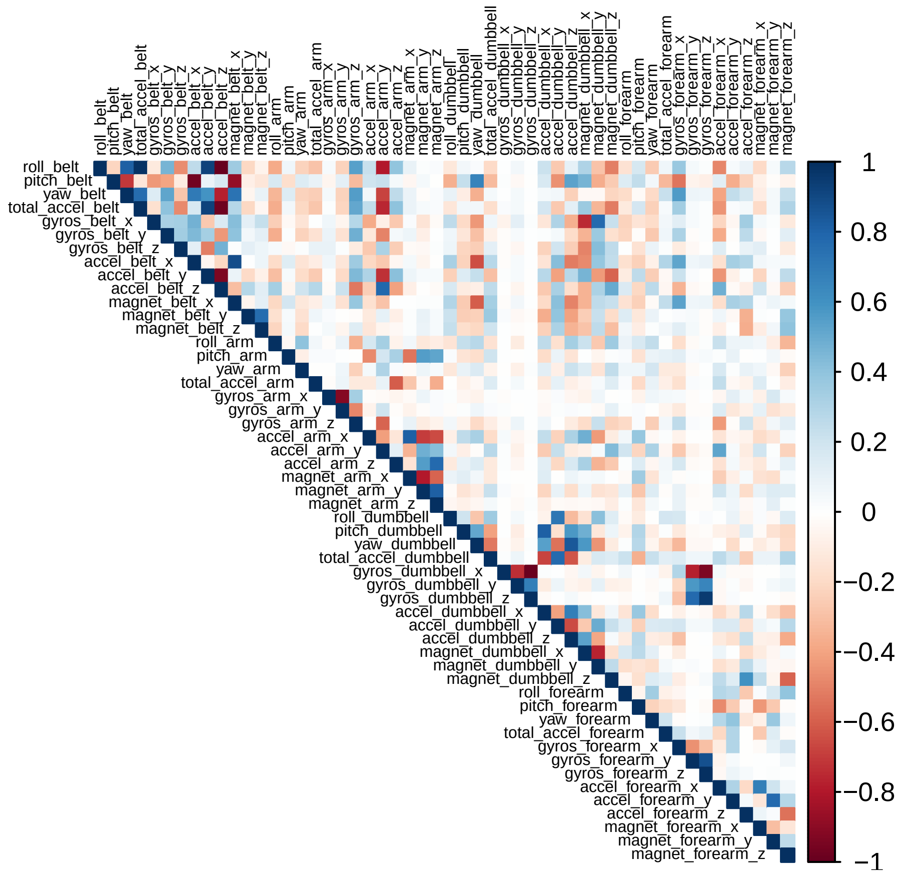
**The testing set (or validation set) is used to test or validate the model by estimating the prediction error.**

```
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

## Graphical display to visualise the correlation matrix

**In the following graph, positive correlations are displayed in blue and negative correlations in red. Colour intensity is proportional to the correlation coefficients.**

```
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, tl.cex = 0.5, tl.col = rgb(0, 0, 0), method="color", type = "upper")
```




# Prediction model building

**We will use Random Forest, Decision Trees, and the Generalized Boosted Regression Model.**

**From this, we will determine the alogorithim that provides the best out-of-sample accuracy.**

# Prediction with Random Forest

```
controlRF <- trainControl(method = "cv", number = 4, verbose = FALSE)
modFitRandForest <- train(classe ~., data = trainData, method = "rf",
                 preProcess = c("center", "scale"),
                 trControl = controlRF)
modFitRandForest
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (52), scaled (52)
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 10303, 10303, 10303, 10302
```

```
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9906097  0.9881208
##   27    0.9895177  0.9867405
##   52    0.9826748  0.9780817
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

### Cross Validation on our testing data

```
predR <- predict(modFitRandForest, newdata = testData)
RF <- confusionMatrix(predR, testData$classe)
RF$overall["Accuracy"]
```

```
##  Accuracy
## 0.9916737
```

## Prediction with Decision Tree

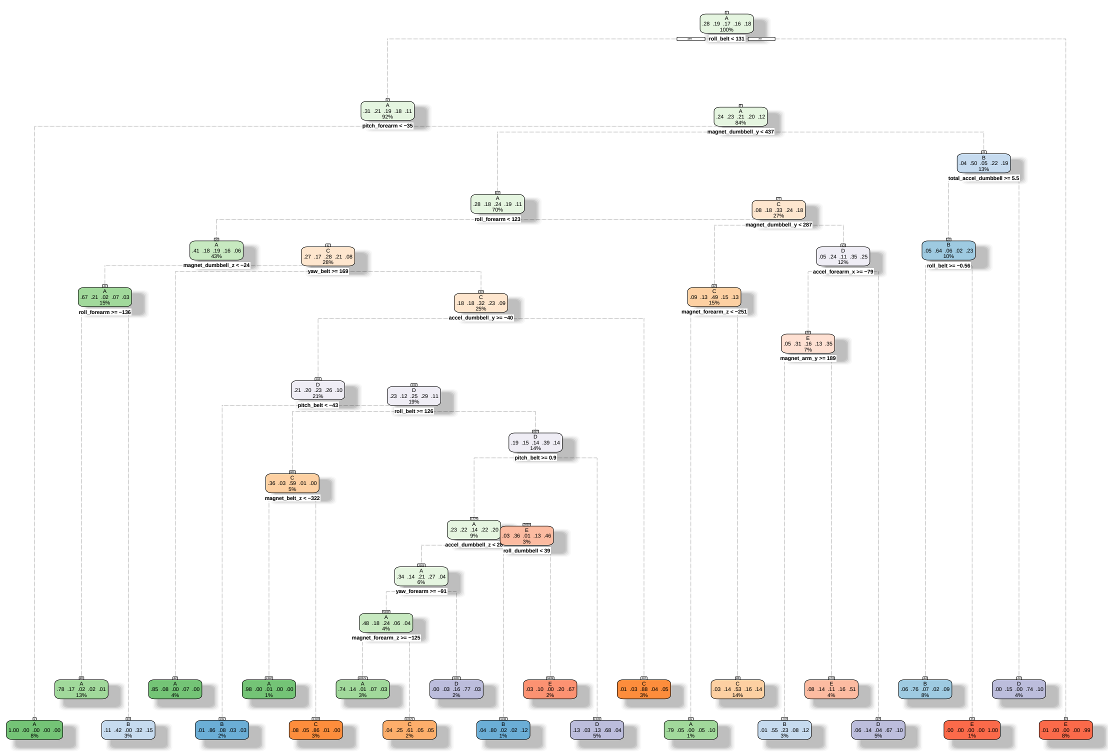
```
modFitDecTree <- rpart(classe ~., data = trainData, method = "class")
modFitDecTree
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##     1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##       2) roll_belt< 130.5 12587 8694 A (0.31 0.21 0.19 0.18 0.11)
##         4) pitch_forearm< -34.55 1107    2 A (1 0.0018 0 0 0) *
##         5) pitch_forearm>=-34.55 11480 8692 A (0.24 0.23 0.21 0.2 0.12)
##          10) magnet_dumbbell_y< 436.5 9644 6924 A (0.28 0.18 0.24 0.19 0.11)
##            20) roll_forearm< 122.5 5972 3532 A (0.41 0.18 0.19 0.16 0.061)
##              40) magnet_dumbbell_z< -24.5 2080  690 A (0.67 0.21 0.017 0.075 0.03)
##                80) roll_forearm>=-136.5 1730  377 A (0.78 0.17 0.02 0.025 0.0052) *
##                81) roll_forearm< -136.5 350  204 B (0.11 0.42 0.0029 0.32 0.15) *
##              41) magnet_dumbbell_z>=-24.5 3892 2819 C (0.27 0.17 0.28 0.21 0.078)
##                82) yaw_belt>=168.5 518   77 A (0.85 0.075 0.0019 0.068 0.0039) *
##                83) yaw_belt< 168.5 3374 2302 C (0.18 0.18 0.32 0.23 0.09)
##                 166) accel_dumbbell_y>=-40.5 2910 2141 D (0.21 0.2 0.23 0.26 0.097)
##                   332) pitch_belt< -43.15 319   44 B (0.0094 0.86 0.075 0.028 0.025) *
##                   333) pitch_belt>=-43.15 2591 1831 D (0.23 0.12 0.25 0.29 0.11)
##                     666) roll_belt>=125.5 637  260 C (0.36 0.035 0.59 0.0094 0.0016)
##                      1332) magnet_belt_z< -322.5 203    5 A (0.98 0.0049 0.015 0 0.0049) *
##                      1333) magnet_belt_z>=-322.5 434   60 C (0.076 0.048 0.86 0.014 0) *
##                     667) roll_belt< 125.5 1954 1200 D (0.19 0.15 0.14 0.39 0.14)
##                      1334) pitch_belt>=0.895 1237  956 A (0.23 0.22 0.14 0.22 0.2)
##                        2668) accel_dumbbell_z< 27.5 778  512 A (0.34 0.14 0.21 0.27 0.037)
##                          5336) yaw_forearm>=-90.9 557  291 A (0.48 0.18 0.24 0.065 0.039)
##                           10672) magnet_forearm_z>=-125.5 348   90 A (0.74 0.14 0.011 0.075 0.032) *
##                           10673) magnet_forearm_z< -125.5 209   82 C (0.038 0.25 0.61 0.048 0.053) *
```

```
##                            5337) yaw_forearm< -90.9 221   50 D (0 0.032 0.16 0.77 0.032) *
##                          2669) accel_dumbbell_z>=27.5 459  246 E (0.033 0.36 0.0087 0.13 0.46)
##                            5338) roll_dumbbell< 38.61985 170   34 B (0.035 0.8 0.024 0.024 0.12) *
##                            5339) roll_dumbbell>=38.61985 289   96 E (0.031 0.1 0 0.2 0.67) *
##                        1335) pitch_belt< 0.895 717  231 D (0.13 0.025 0.13 0.68 0.042) *
##                   167) accel_dumbbell_y< -40.5 464   57 C (0.0086 0.032 0.88 0.037 0.045) *
##             21) roll_forearm>=122.5 3672 2470 C (0.076 0.18 0.33 0.24 0.18)
##               42) magnet_dumbbell_y< 286.5 2091 1063 C (0.093 0.13 0.49 0.15 0.13)
##                 84) magnet_forearm_z< -251 165   34 A (0.79 0.055 0 0.048 0.1) *
##                 85) magnet_forearm_z>=-251 1926  898 C (0.033 0.14 0.53 0.16 0.14) *
##               43) magnet_dumbbell_y>=286.5 1581 1034 D (0.054 0.24 0.11 0.35 0.25)
##                 86) accel_forearm_x>=-79.5 942  612 E (0.051 0.31 0.16 0.13 0.35)
##                  172) magnet_arm_y>=188.5 384  171 B (0.013 0.55 0.23 0.078 0.12) *
##                  173) magnet_arm_y< 188.5 558  276 E (0.077 0.14 0.11 0.16 0.51) *
##                 87) accel_forearm_x< -79.5 639  212 D (0.059 0.14 0.036 0.67 0.097) *
##           11) magnet_dumbbell_y>=436.5 1836  913 B (0.037 0.5 0.046 0.22 0.19)
##             22) total_accel_dumbbell>=5.5 1317  473 B (0.052 0.64 0.063 0.018 0.23)
##               44) roll_belt>=-0.565 1114  270 B (0.061 0.76 0.075 0.022 0.085) *
##               45) roll_belt< -0.565 203    0 E (0 0 0 0 1) *
##             23) total_accel_dumbbell< 5.5 519  133 D (0 0.15 0.0039 0.74 0.1) *
##       3) roll_belt>=130.5 1150   13 E (0.011 0 0 0 0.99) *
```

```r
fancyRpartPlot(modFitDecTree)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2019−Dec−19 12:10:14 psartor

### Cross Validation on our testing data

```
predD <- predict(modFitDecTree, testData, type = "class")
DT <-confusionMatrix(testData$classe, predD)
DT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1482   50   50   60   32
##          B  166  647  182   85   59
##          C   28   93  801   72   32
##          D   56  100  152  595   61
##          E   19   92  132   54  785
##
## Overall Statistics
##
##                Accuracy : 0.7324
##                  95% CI : (0.7209, 0.7436)
##     No Information Rate : 0.2975
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6611
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8464   0.6589   0.6082   0.6871   0.8101
## Specificity            0.9536   0.8997   0.9507   0.9265   0.9396
## Pos Pred Value         0.8853   0.5680   0.7807   0.6172   0.7255
## Neg Pred Value         0.9361   0.9294   0.8938   0.9449   0.9617
## Prevalence             0.2975   0.1669   0.2238   0.1472   0.1647
## Detection Rate         0.2518   0.1099   0.1361   0.1011   0.1334
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9000   0.7793   0.7795   0.8068   0.8748
```

```
DT$overall["Accuracy"]
```

```
##  Accuracy
## 0.7323704
```

## Prediction with Generalized Boosted Regression

```
modFitBoostRegress <- train(classe ~., data = trainData, method = "gbm",verbose =  FALSE, trControl=tra
modFitBoostRegress$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

### Cross Validation on our testing data

```
predG <- predict(modFitBoostRegress, testData)
GBM <- confusionMatrix(testData$classe, predG)
GBM$overall["Accuracy"]
```

```
##  Accuracy
## 0.9575191
```

## Applying the best model to the provided test set

### The Random Forest model yielded the best prediction in in-sample. Therefore, this model will be applied to predict the provided 20 different test cases.

```
FinalTestPred <- predict(modFitRandForest, newdata = testCleaned)
FinalTestPred
```

```
##  [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
```