PML Course Project

Alvin Alon

December 2018

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

# Libraries————————————-

# ———————————————-

library(caret)  
library(rattle)

# Data Load————————————-

# ———————————————-

#Training Dataset  
#-----------------  
TrainData <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),header=TRUE)  
dim(TrainData)

## [1] 19622 160

#Testing Dataset  
#-----------------  
TestData <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),header=TRUE)  
dim(TestData)

## [1] 20 160

str(TrainData)

## '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 1323084232 1323084232 1323084232 1323084232 ...  
## $ raw\_timestamp\_part\_2 : int 788290 808298 820366 120339 196328 304277 368296 440390 484323 484434 ...  
## $ 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 : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ kurtosis\_picth\_belt : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ kurtosis\_yaw\_belt : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_roll\_belt : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_roll\_belt.1 : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_yaw\_belt : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...  
## $ 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 : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ 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 : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ 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 : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ 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 : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ kurtosis\_picth\_arm : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ kurtosis\_yaw\_arm : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_roll\_arm : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_pitch\_arm : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_yaw\_arm : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ 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 : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ kurtosis\_picth\_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ kurtosis\_yaw\_dumbbell : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_roll\_dumbbell : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_pitch\_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_yaw\_dumbbell : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...  
## $ 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 : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ 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 : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ amplitude\_roll\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...  
## [list output truncated]

# Observations———————————-

# ———————————————-

The training data set is made of 19622 observations on 160 columns. We can notice that many columns have NA values or blank values on almost every observation. So we will remove them, because they will not produce any information. The first seven columns give information about the people who did the test, and also timestamps. We will not take them in our model.

# Data Cleaning———————————

# ———————————————-

# Removing at least 90% of NA or blank values on the training dataset  
# -------------------------------------------------------------------  
  
# Train Dataset  
indColToRemove <- which(colSums(is.na(TrainData) |TrainData=="")>0.9\*dim(TrainData)[1])   
TrainDataClean <- TrainData[,-indColToRemove]  
TrainDataClean <- TrainDataClean[,-c(1:7)]  
dim(TrainDataClean)

## [1] 19622 53

# Test Dataset  
indColToRemove <- which(colSums(is.na(TestData) |TestData=="")>0.9\*dim(TestData)[1])   
TestDataClean <- TestData[,-indColToRemove]  
TestDataClean <- TestDataClean[,-1]  
dim(TestDataClean)

## [1] 20 59

str(TestDataClean)

## 'data.frame': 20 obs. of 59 variables:  
## $ user\_name : Factor w/ 6 levels "adelmo","carlitos",..: 6 5 5 1 4 5 5 5 2 3 ...  
## $ raw\_timestamp\_part\_1: int 1323095002 1322673067 1322673075 1322832789 1322489635 1322673149 1322673128 1322673076 1323084240 1322837822 ...  
## $ raw\_timestamp\_part\_2: int 868349 778725 342967 560311 814776 510661 766645 54671 916313 384285 ...  
## $ cvtd\_timestamp : Factor w/ 11 levels "02/12/2011 13:33",..: 5 10 10 1 6 11 11 10 3 2 ...  
## $ new\_window : Factor w/ 1 level "no": 1 1 1 1 1 1 1 1 1 1 ...  
## $ num\_window : int 74 431 439 194 235 504 485 440 323 664 ...  
## $ roll\_belt : num 123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.93 114 ...  
## $ pitch\_belt : num 27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.72 22.4 ...  
## $ yaw\_belt : num -4.75 -88.9 -88.5 162 -88.6 -87.7 -87.3 -88.5 -93.7 -13.1 ...  
## $ total\_accel\_belt : int 20 4 5 17 3 4 4 4 4 18 ...  
## $ gyros\_belt\_x : num -0.5 -0.06 0.05 0.11 0.03 0.1 -0.06 -0.18 0.1 0.14 ...  
## $ gyros\_belt\_y : num -0.02 -0.02 0.02 0.11 0.02 0.05 0 -0.02 0 0.11 ...  
## $ gyros\_belt\_z : num -0.46 -0.07 0.03 -0.16 0 -0.13 0 -0.03 -0.02 -0.16 ...  
## $ accel\_belt\_x : int -38 -13 1 46 -8 -11 -14 -10 -15 -25 ...  
## $ accel\_belt\_y : int 69 11 -1 45 4 -16 2 -2 1 63 ...  
## $ accel\_belt\_z : int -179 39 49 -156 27 38 35 42 32 -158 ...  
## $ magnet\_belt\_x : int -13 43 29 169 33 31 50 39 -6 10 ...  
## $ magnet\_belt\_y : int 581 636 631 608 566 638 622 635 600 601 ...  
## $ magnet\_belt\_z : int -382 -309 -312 -304 -418 -291 -315 -305 -302 -330 ...  
## $ roll\_arm : num 40.7 0 0 -109 76.1 0 0 0 -137 -82.4 ...  
## $ pitch\_arm : num -27.8 0 0 55 2.76 0 0 0 11.2 -63.8 ...  
## $ yaw\_arm : num 178 0 0 -142 102 0 0 0 -167 -75.3 ...  
## $ total\_accel\_arm : int 10 38 44 25 29 14 15 22 34 32 ...  
## $ gyros\_arm\_x : num -1.65 -1.17 2.1 0.22 -1.96 0.02 2.36 -3.71 0.03 0.26 ...  
## $ gyros\_arm\_y : num 0.48 0.85 -1.36 -0.51 0.79 0.05 -1.01 1.85 -0.02 -0.5 ...  
## $ gyros\_arm\_z : num -0.18 -0.43 1.13 0.92 -0.54 -0.07 0.89 -0.69 -0.02 0.79 ...  
## $ accel\_arm\_x : int 16 -290 -341 -238 -197 -26 99 -98 -287 -301 ...  
## $ accel\_arm\_y : int 38 215 245 -57 200 130 79 175 111 -42 ...  
## $ accel\_arm\_z : int 93 -90 -87 6 -30 -19 -67 -78 -122 -80 ...  
## $ magnet\_arm\_x : int -326 -325 -264 -173 -170 396 702 535 -367 -420 ...  
## $ magnet\_arm\_y : int 385 447 474 257 275 176 15 215 335 294 ...  
## $ magnet\_arm\_z : int 481 434 413 633 617 516 217 385 520 493 ...  
## $ roll\_dumbbell : num -17.7 54.5 57.1 43.1 -101.4 ...  
## $ pitch\_dumbbell : num 25 -53.7 -51.4 -30 -53.4 ...  
## $ yaw\_dumbbell : num 126.2 -75.5 -75.2 -103.3 -14.2 ...  
## $ total\_accel\_dumbbell: int 9 31 29 18 4 29 29 29 3 2 ...  
## $ gyros\_dumbbell\_x : num 0.64 0.34 0.39 0.1 0.29 -0.59 0.34 0.37 0.03 0.42 ...  
## $ gyros\_dumbbell\_y : num 0.06 0.05 0.14 -0.02 -0.47 0.8 0.16 0.14 -0.21 0.51 ...  
## $ gyros\_dumbbell\_z : num -0.61 -0.71 -0.34 0.05 -0.46 1.1 -0.23 -0.39 -0.21 -0.03 ...  
## $ accel\_dumbbell\_x : int 21 -153 -141 -51 -18 -138 -145 -140 0 -7 ...  
## $ accel\_dumbbell\_y : int -15 155 155 72 -30 166 150 159 25 -20 ...  
## $ accel\_dumbbell\_z : int 81 -205 -196 -148 -5 -186 -190 -191 9 7 ...  
## $ magnet\_dumbbell\_x : int 523 -502 -506 -576 -424 -543 -484 -515 -519 -531 ...  
## $ magnet\_dumbbell\_y : int -528 388 349 238 252 262 354 350 348 321 ...  
## $ magnet\_dumbbell\_z : int -56 -36 41 53 312 96 97 53 -32 -164 ...  
## $ roll\_forearm : num 141 109 131 0 -176 150 155 -161 15.5 13.2 ...  
## $ pitch\_forearm : num 49.3 -17.6 -32.6 0 -2.16 1.46 34.5 43.6 -63.5 19.4 ...  
## $ yaw\_forearm : num 156 106 93 0 -47.9 89.7 152 -89.5 -139 -105 ...  
## $ total\_accel\_forearm : int 33 39 34 43 24 43 32 47 36 24 ...  
## $ gyros\_forearm\_x : num 0.74 1.12 0.18 1.38 -0.75 -0.88 -0.53 0.63 0.03 0.02 ...  
## $ gyros\_forearm\_y : num -3.34 -2.78 -0.79 0.69 3.1 4.26 1.8 -0.74 0.02 0.13 ...  
## $ gyros\_forearm\_z : num -0.59 -0.18 0.28 1.8 0.8 1.35 0.75 0.49 -0.02 -0.07 ...  
## $ accel\_forearm\_x : int -110 212 154 -92 131 230 -192 -151 195 -212 ...  
## $ accel\_forearm\_y : int 267 297 271 406 -93 322 170 -331 204 98 ...  
## $ accel\_forearm\_z : int -149 -118 -129 -39 172 -144 -175 -282 -217 -7 ...  
## $ magnet\_forearm\_x : int -714 -237 -51 -233 375 -300 -678 -109 0 -403 ...  
## $ magnet\_forearm\_y : int 419 791 698 783 -787 800 284 -619 652 723 ...  
## $ magnet\_forearm\_z : int 617 873 783 521 91 884 585 -32 469 512 ...  
## $ problem\_id : int 1 2 3 4 5 6 7 8 9 10 ...

# Observations———————————-

# ———————————————-

After cleaning, the new training data set has only 53 columns.

# Splitting of Datasets————————-

# ———————————————-

set.seed(12345)  
inTrain1 <- createDataPartition(TrainDataClean$classe, p=0.75, list=FALSE)  
  
#Train  
Train1 <- TrainDataClean[inTrain1,]  
dim(Train1)

## [1] 14718 53

#Test  
Test1 <- TrainDataClean[-inTrain1,]  
dim(Test1)

## [1] 4904 53

# Model Training————————-

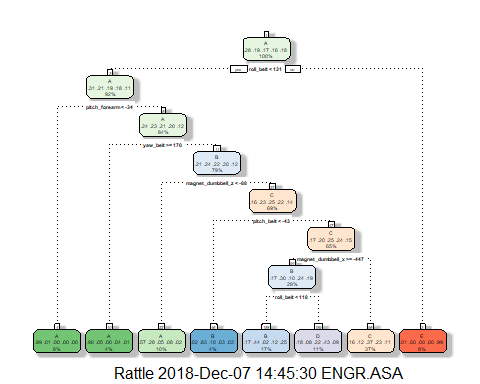
# ———————————————-

Testing of 3 different models : \* classification\_tree \* random\_forest \* gradient\_boosting\_method

In order to limit the effects of overfitting and improve the efficicency of the models, \*cross-validation technique will use a 5 folds (usually, 5 or 10 can be used, but 10 folds gives higher run times with no significant increase of the accuracy).

# Classification Tree

# Classification Tree  
# -------------------  
trControl <- trainControl(method="cv", number=5)  
model\_CT <- train(classe~., data=Train1, method="rpart", trControl=trControl)  
  
# Print Classification Tree Model  
fancyRpartPlot(model\_CT$finalModel)



# Prediction  
trainpred <- predict(model\_CT,newdata=Test1)  
  
#Confusion Matrix  
confMatCT <- confusionMatrix(Test1$classe,trainpred)  
confMatCT$table

## Reference  
## Prediction A B C D E  
## A 870 159 273 88 5  
## B 162 530 214 43 0  
## C 29 36 674 116 0  
## D 46 136 429 193 0  
## E 16 221 224 51 389

confMatCT$overall[1]

## Accuracy   
## 0.5415987

# Observations for Classification Tree (CT)

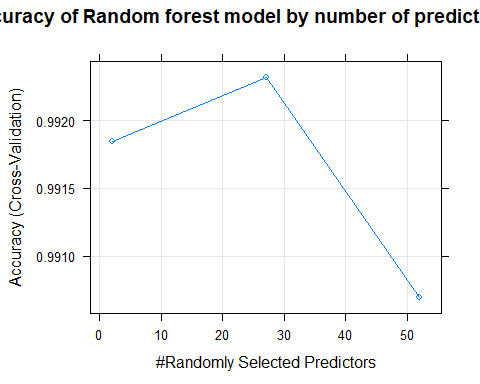
We can notice that the accuracy of this CT is very low (about 55%). This means that the outcome class will not be predicted very well by the other predictors.

# Random Forests

#Random Forests  
model\_RF <- train(classe~., data=Train1, method="rf", trControl=trControl, verbose=FALSE)  
print(model\_RF)

## Random Forest   
##   
## 14718 samples  
## 52 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 11776, 11775, 11773, 11774, 11774   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.9918466 0.9896855  
## 27 0.9923226 0.9902884  
## 52 0.9906918 0.9882252  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 27.

#Print Random Forests  
plot(model\_RF,main="Accuracy of Random forest model by number of predictors")



#Prediction  
trainpred <- predict(model\_RF,newdata=Test1)  
  
#Confusion Matrix  
confMatRF <- confusionMatrix(Test1$classe,trainpred)  
confMatRF$table

## Reference  
## Prediction A B C D E  
## A 1394 1 0 0 0  
## B 6 939 4 0 0  
## C 0 2 849 4 0  
## D 0 0 10 794 0  
## E 0 0 2 5 894

confMatRF$overall[1]

## Accuracy   
## 0.9930669

# Observations for Random Forests

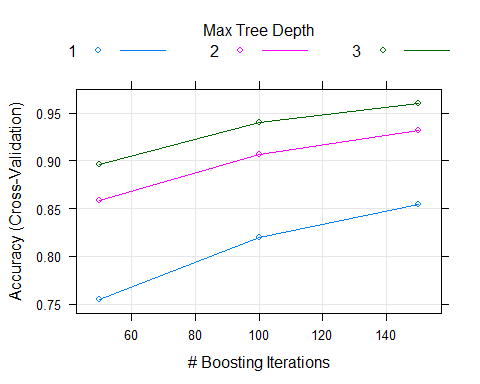
With random forest, an accuracy of 99.3% has been using cross-validation with 5 steps.

# Gradient Boosting Method

#Gradient Boosting Method  
model\_GBM <- train(classe~., data=Train1, method="gbm", trControl=trControl, verbose=FALSE)  
print(model\_GBM)

## Stochastic Gradient Boosting   
##   
## 14718 samples  
## 52 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 11775, 11775, 11774, 11773, 11775   
## Resampling results across tuning parameters:  
##   
## interaction.depth n.trees Accuracy Kappa   
## 1 50 0.7547203 0.6891712  
## 1 100 0.8194732 0.7714678  
## 1 150 0.8541930 0.8154916  
## 2 50 0.8585420 0.8207615  
## 2 100 0.9064419 0.8816012  
## 2 150 0.9317853 0.9136787  
## 3 50 0.8959101 0.8682235  
## 3 100 0.9406855 0.9249385  
## 3 150 0.9605249 0.9500546  
##   
## 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.

#Print Random Forests  
plot(model\_GBM)



#Prediction  
trainpred <- predict(model\_GBM,newdata=Test1)  
  
#Confusion Matrix  
confMatGBM <- confusionMatrix(Test1$classe,trainpred)  
confMatGBM$table

## Reference  
## Prediction A B C D E  
## A 1378 14 3 0 0  
## B 39 874 34 1 1  
## C 0 20 824 10 1  
## D 0 1 21 776 6  
## E 3 7 8 16 867

confMatGBM$overall[1]

## Accuracy   
## 0.9622757

# Observation for Gradient Boosting Method(GBM)

With GBM, an accuracy of 95.9% has been using cross-validation with 5 folds.

# Conclusion————————————

# ———————————————-

This shows that the random forest (RF) model is the best one. We will then use RF to predict the values of classe for the test data set.

FinalTestPred\_RF <- predict(model\_RF,newdata=TestDataClean)  
FinalTestPred\_RF

## [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