Practical Machine Learning Project

Maribel

27/11/2020

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 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 (see the section on the Weight Lifting Exercise Dataset)

About the Data Set

This human activity recognition research has traditionally focused on discriminating between different activities, i.e. to predict "which" activity was performed at a specific point in time (like with the Daily Living Activities dataset above). The approach we propose for the Weight Lifting Exercises dataset is to investigate "how (well)" an activity was performed by the wearer. The "how (well)" investigation has only received little attention so far, even though it potentially provides useful information for a large variety of applications,such as sports training.

```
library(knitr)
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
library(e1071)
library(randomForest)
library(corrplot)
library(gbm)
library(RColorBrewer)



Train <-  read.csv("pml-training.csv")
Test  <- read.csv("pml-testing.csv")
```

Partioning the training set into two

```
set.seed(69420)
inTrain  <- createDataPartition(Train$classe, p=0.7, list=FALSE)
TrainD <- Train[inTrain, ]
```

```
TestD  <- Train[-inTrain, ]

dim(TrainD)
```

```
## [1] 13737   160
```

```
str(TrainD)
```

```
## 'data.frame':    13737 obs. of  160 variables:
##  $ X                       : int  3 4 5 7 9 10 11 13 14 15 ...
##  $ user_name               : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084232 1323084232 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  820366 120339 196328 368296 484323 484434 500302 560359 576390 604
##  $ cvtd_timestamp          : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "05/12/20
##  $ new_window              : chr  "no" "no" "no" "no" ...
##  $ num_window              : int  11 12 12 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.42 1.48 1.48 1.42 1.43 1.45 1.45 1.42 1.42 1.45 ...
##  $ pitch_belt              : num  8.07 8.05 8.07 8.09 8.16 8.17 8.18 8.2 8.21 8.2 ...
##  $ 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      : 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 0.02 0.02 0.02 0.03 0.03 0.02 0.02 0 ...
##  $ gyros_belt_y            : num  0 0 0.02 0 0 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.03 -0.02 -0.02 -0.02 0 -0.02 0 -0.02 0 ...
##  $ accel_belt_x            : int  -20 -22 -21 -22 -20 -21 -21 -22 -22 -21 ...
##  $ accel_belt_y            : int  5 3 2 3 2 4 2 4 4 2 ...
##  $ accel_belt_z            : int  23 21 24 21 24 22 23 21 21 22 ...
##  $ magnet_belt_x           : int  -2 -6 -6 -4 1 -3 -5 -3 -8 -1 ...
##  $ magnet_belt_y           : int  600 604 600 599 602 609 596 606 598 597 ...
```

```
##  $ magnet_belt_z          : int  -305 -310 -302 -311 -312 -308 -317 -309 -310 -310 ...
##  $ roll_arm               : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -129 ...
##  $ pitch_arm              : num  22.5 22.1 22.1 21.9 21.7 21.6 21.5 21.4 21.4 21.4 ...
##  $ 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  0.02 0.02 0 0 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y            : num  -0.02 -0.03 -0.03 -0.03 -0.03 -0.03 -0.03 -0.02 0 0 ...
##  $ gyros_arm_z            : num  -0.02 0.02 0 0 -0.02 -0.02 0 -0.02 -0.03 -0.03 ...
##  $ accel_arm_x            : int  -289 -289 -289 -289 -288 -288 -290 -287 -288 -289 ...
##  $ accel_arm_y            : int  110 111 111 111 109 110 110 111 111 111 ...
##  $ accel_arm_z            : int  -126 -123 -123 -125 -122 -124 -123 -124 -124 -124 ...
##  $ magnet_arm_x           : int  -368 -372 -374 -373 -369 -376 -366 -372 -371 -374 ...
##  $ magnet_arm_y           : int  344 344 337 336 341 334 339 338 331 342 ...
##  $ magnet_arm_z           : int  513 512 506 509 518 516 509 509 523 510 ...
##  $ 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  12.9 13.4 13.4 13.1 13.2 ...
##  $ pitch_dumbbell         : num  -70.3 -70.4 -70.4 -70.2 -70.4 ...
##  $ yaw_dumbbell           : num  -85.1 -84.9 -84.9 -85.1 -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]
```

Remove NearZeroVariance variables and NA

```
TrainD <- TrainD[, -c(1:7)]
TestD <- TestD[, -c(1:7)]
NZV <- nearZeroVar(TrainD)
TrainD <- TrainD[, -NZV]
TestD  <- TestD[, -NZV]
allNA    <- sapply(TrainD, function(x) mean(is.na(x))) > 0.95
TrainD <- TrainD[, allNA==FALSE]
TestD  <-TestD[, allNA==FALSE]
dim(TrainD)
```

```
## [1] 13737    53
```

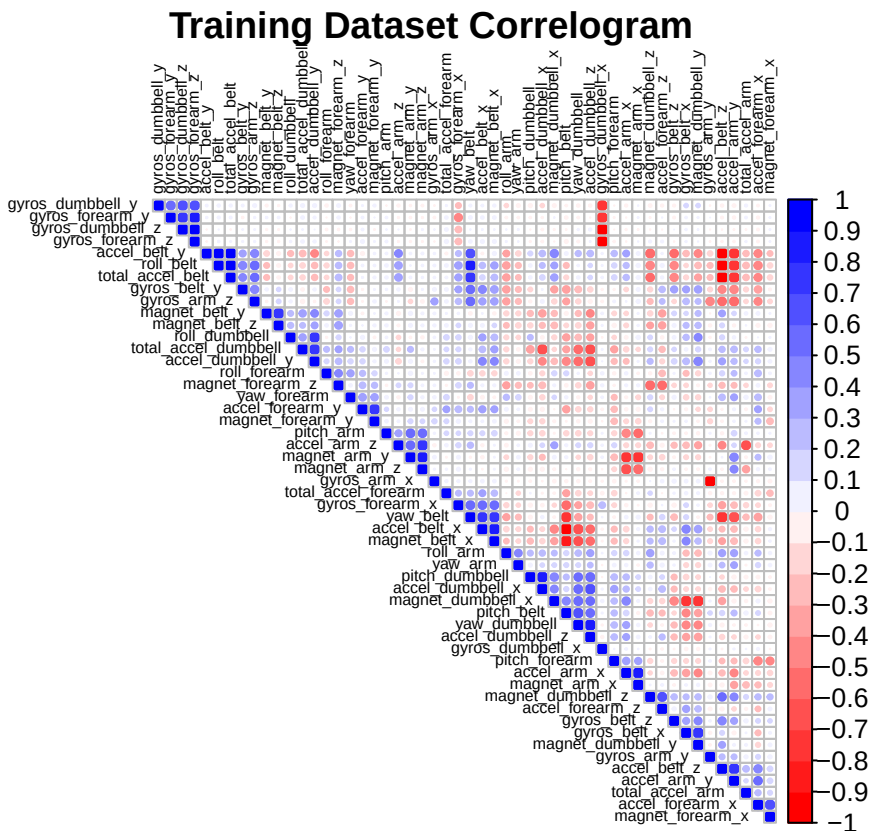
After cleaning, the new training data set has only 53 columns.

Exploratoy Data Analysis

we perform a correlation analysis to find out if multicollinearity is a problem

```
col<- colorRampPalette(c("red", "white", "blue"))(20)


corMatrix <- cor(TrainD[, -53])
corrplot(corMatrix, order = "hclust", method = "circle", type = "upper", tl.cex = 0.5,tl.col="black",col
```



The different tones show the correlation blue closer to 1, red closer to -1

Building the Predictive Model

```
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRF01 <- train(classe ~ ., data=TrainD, method="rf", trControl=controlRF)
modFitRF01
```

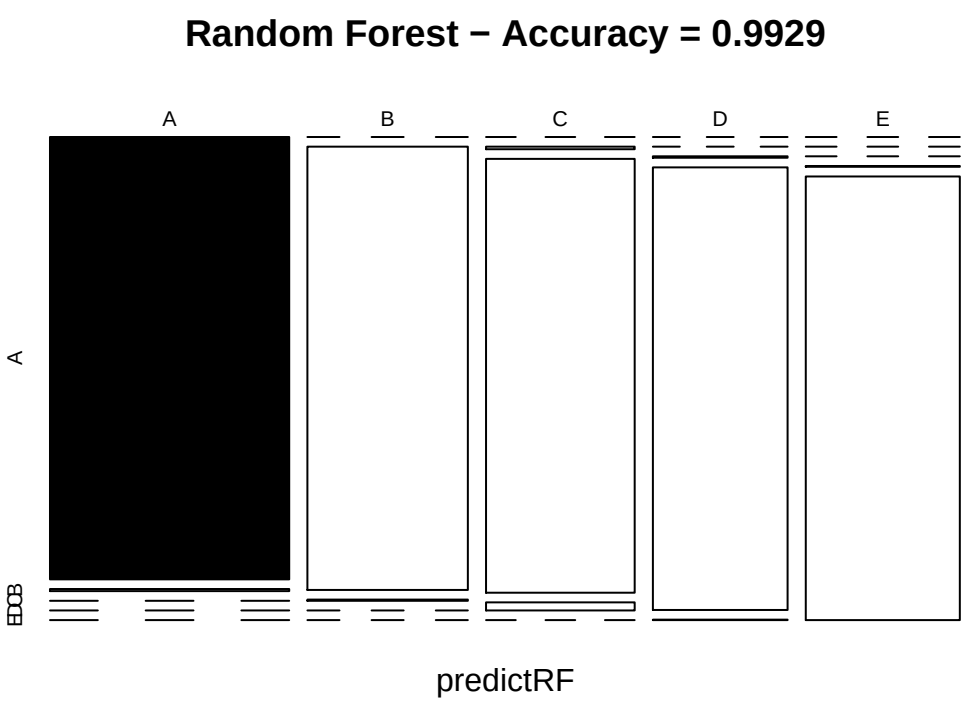
```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9158, 9159, 9157
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9884983  0.9854486
##   27    0.9880615  0.9848962
##   52    0.9845672  0.9804746
##
```

```
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
predictRF <- predict(modFitRF01, newdata=TestD)
confMatRF <- confusionMatrix(table(predictRF, TestD$classe))
confMatRF
```

```
## Confusion Matrix and Statistics
##
##
## predictRF    A    B    C    D    E
##         A 1674    8    0    0    0
##         B    0 1125    3    0    0
##         C    0    6 1020   19    0
##         D    0    0    3  943    1
##         E    0    0    0    2 1081
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9904, 0.9949)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9877   0.9942   0.9782   0.9991
## Specificity            0.9981   0.9994   0.9949   0.9992   0.9996
## Pos Pred Value         0.9952   0.9973   0.9761   0.9958   0.9982
## Neg Pred Value         1.0000   0.9971   0.9988   0.9957   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1912   0.1733   0.1602   0.1837
## Detection Prevalence   0.2858   0.1917   0.1776   0.1609   0.1840
## Balanced Accuracy      0.9991   0.9935   0.9945   0.9887   0.9993
```

```
plot(confMatRF$table, col = confMatRF$byClass, main = paste("Random Forest - Accuracy =", round(confMatl
```



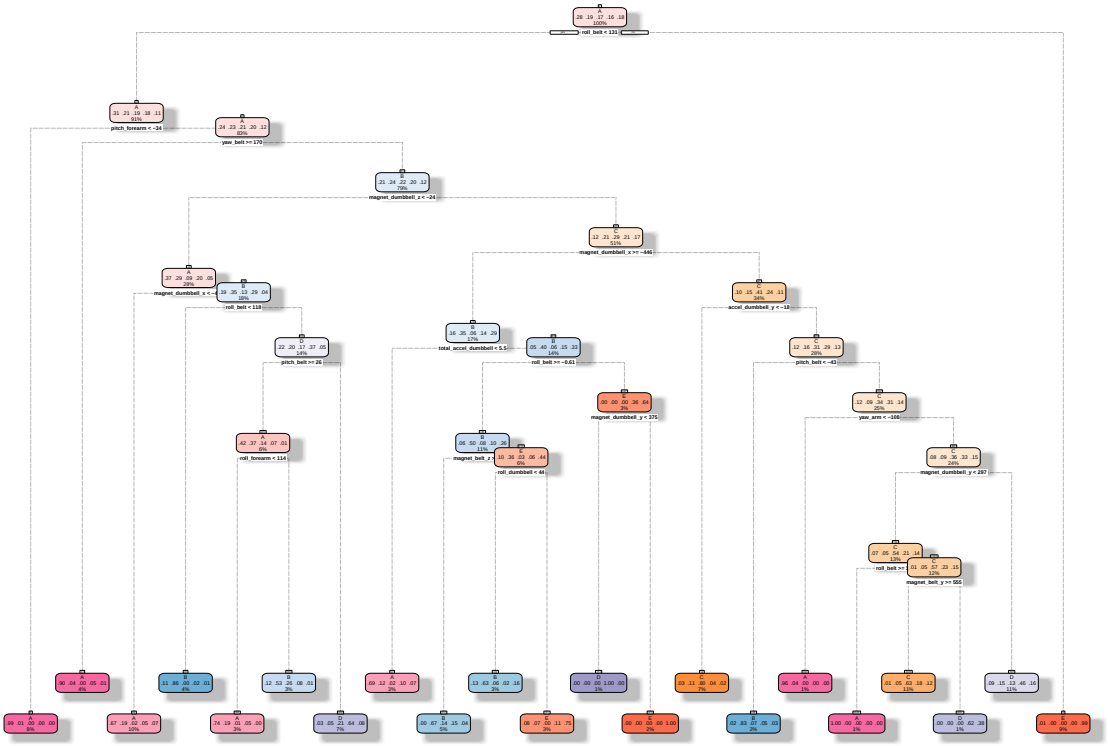
Prediction with Decision Trees

I use 3 types of models in this section

Decision tree

```
modFitDT025 <- rpart(classe ~ ., data=TrainD, method="class")
fancyRpartPlot(modFitDT025,palettes="RdPu")
```

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



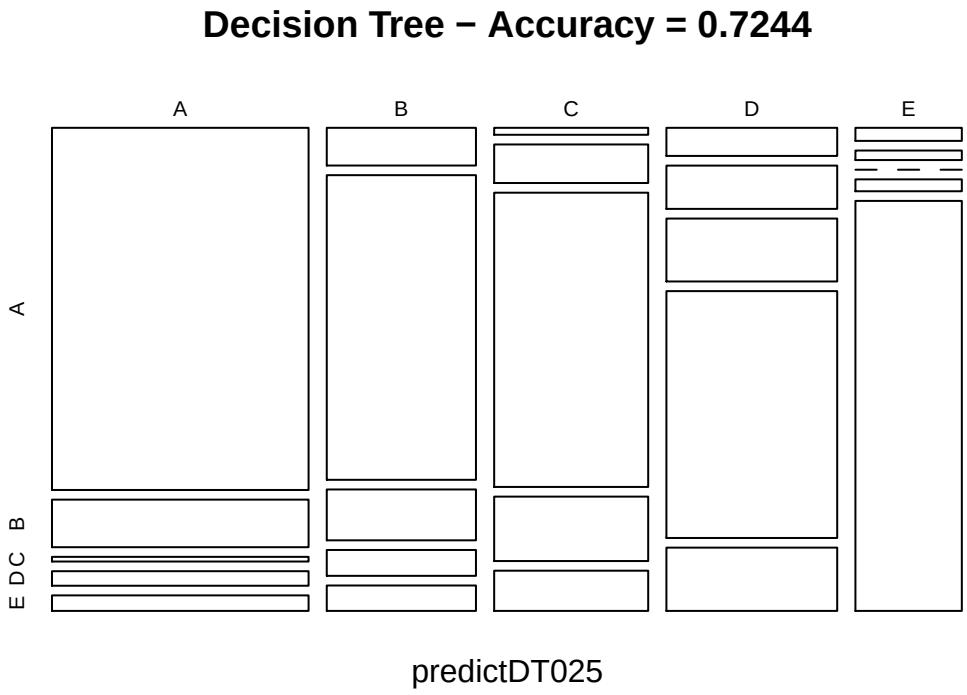
Random Forest

```
predictDT025 <- predict(modFitDT025, newdata=TestD, type="class")
confMat_Tree <- confusionMatrix(table(predictDT025, TestD$classe))
confMat_Tree
```

```
## Confusion Matrix and Statistics
##
##
## predictDT025    A    B    C    D    E
##            A 1470  193   19   59   63
##            B   89  719  120   61   60
##            C   17   94  717  157   98
##            D   76  117  170  667  171
##            E   22   16    0   20  690
##
## Overall Statistics
##
##                Accuracy : 0.7244
##                  95% CI : (0.7128, 0.7358)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.6509
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8781   0.6313   0.6988   0.6919   0.6377
## Specificity            0.9207   0.9305   0.9247   0.8915   0.9879
## Pos Pred Value         0.8149   0.6854   0.6620   0.5554   0.9225
## Neg Pred Value         0.9500   0.9132   0.9357   0.9366   0.9237
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2498   0.1222   0.1218   0.1133   0.1172
## Detection Prevalence   0.3065   0.1782   0.1840   0.2041   0.1271
## Balanced Accuracy      0.8994   0.7809   0.8118   0.7917   0.8128
```

```
plot(confMat_Tree$table, col = confMat_Tree$byClass, main = paste("Decision Tree - Accuracy =", round(c
```



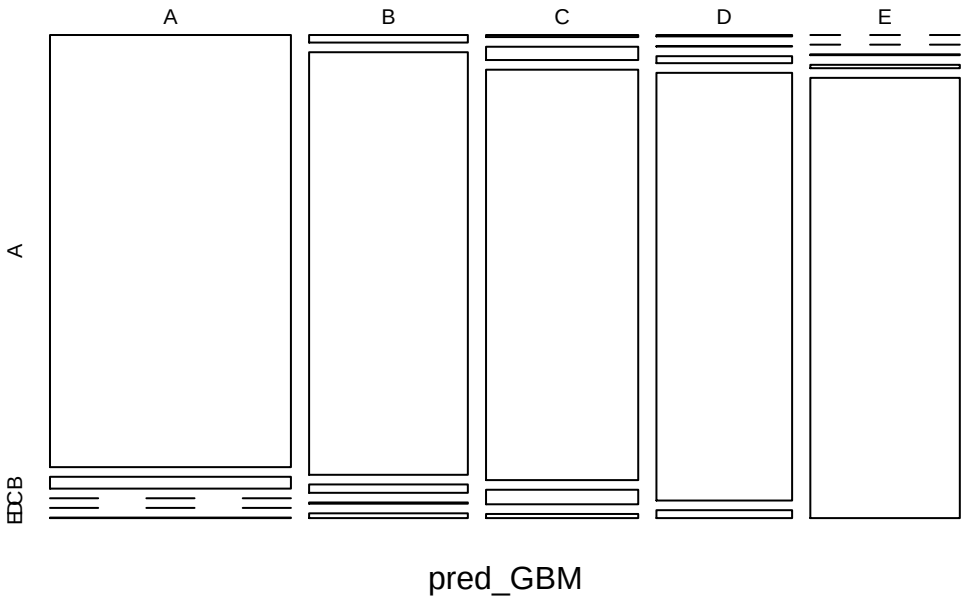

```
cont_GBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modF_GBM  <- train(classe ~ ., data=TrainD, method = "gbm", trControl = cont_GBM, verbose = FALSE)
modF_GBM$finalModel
```

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

```r
pred_GBM <- predict(modF_GBM, newdata=TestD)
conf_GBM <- confusionMatrix(table(pred_GBM, TestD$classe))
conf_GBM
```

```
## Confusion Matrix and Statistics
##
##
## pred_GBM    A    B    C    D    E
##        A 1647   45    0    0    2
##        B   19 1061   21    3   12
##        C    5   32  988   35   10
##        D    3    1   15  918   17
##        E    0    0    2    8 1041
##
## Overall Statistics
##
##                Accuracy : 0.9609
##                  95% CI : (0.9556, 0.9657)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9505
##
##  Mcnemar's Test P-Value : 9.544e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9315   0.9630   0.9523   0.9621
## Specificity            0.9888   0.9884   0.9831   0.9927   0.9979
## Pos Pred Value         0.9723   0.9507   0.9234   0.9623   0.9905
## Neg Pred Value         0.9936   0.9836   0.9921   0.9907   0.9915
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2799   0.1803   0.1679   0.1560   0.1769
## Detection Prevalence   0.2879   0.1896   0.1818   0.1621   0.1786
## Balanced Accuracy      0.9864   0.9600   0.9730   0.9725   0.9800
```

```r
plot(conf_GBM$table, col = conf_GBM$byClass,
     main = paste("GBM - Accuracy =", round(conf_GBM$overall['Accuracy'], 4)))
```



Random Forests gave an Accuracy in the Test dataset of 99.29, which was more accurate that what I got from the Decision Trees or GBM