project

Felix Garcia A.

23/6/2020

Final project on Machine Learning

Overview

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).

About 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. ### What you should submit

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.

Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-). You should also apply your machine learning algorithm to the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

Cleaning and prepearing the data

Reproduceability

My random number was seed was set at 1234 for all code to reproduce the results below. Also different packages were downloaded and installed, such as caret and randomForest. These should also be installed in order to reproduce the results below.

Model building

The variable “classe”,is a factor variable with 5 levels to answer the performance set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions:

- exactly according to the specification (Class A)
- throwing the elbows to the front (Class B)
- lifting the dumbbell only halfway (Class C)
- lowering the dumbbell only halfway (Class D)
- throwing the hips to the front (Class E)?

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes." Prediction evaluations will be based on maximizing the accuracy and minimizing the out-of-sample error. All other available variables after cleaning will be used for prediction. Two models will be tested using decision tree and random forest algorithms. The model with the highest accuracy will be chosen as our final model.

Cross-validation

Our training data was set randomly without replacement into 2 subsamples: subTraining data (p = 80% of the original Training data set) and subTesting data (20%). Our models will be fitted on the subTraining data set, and tested on the subTesting data. Once the most accurate model is choosen, it will be tested on the original Testing data set as needed.

Expected out-of-sample error

The expected out-of-sample error will correspond to the quantity: 1-accuracy in the cross-validation data. Accuracy is the proportion of correct classified observation over the total sample in the subTesting data set. Expected accuracy is the expected accuracy in the out-of-sample data set. Thus, the expected value of the out-of-sample error will correspond to the expected number of missclassified observations/total observations in the Test data set, which is the quantity: 1-accuracy found from the cross-validation data set.

Our outcome variable “classe” is an unordered factor variable. Thus, we can choose our error type as 1-accuracy. We have a large sample size in the Training data set. This allow us to divide our Training sample into subTraining and subTesting to allow cross-validation. Features with all missing values will be discarded as well as features that are irrelevant. All other features will be kept as relevant variables. Decision tree and random forest algorithms are known for their ability of detecting the features that are important for classification.

Preparing the data and packages needed

First we need to install,call the packages and seed them to make our report reproducible.

```
library(caret)
```

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

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

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

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Versión 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

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

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

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

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

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
set.seed(1234)
```

Loading the data

We set our data url as following:

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
```

The testing data also was set as following:

```
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

So our next step is load the data.

```
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

Partioning the training set into two

Partioning Training data set into two data sets, 80% for myTraining, 20% for myTesting:

```
inTrain <- createDataPartition(y=training$classe, p=0.8, list=FALSE)
myTraining <- training[inTrain, ]; myTesting <- training[-inTrain, ]
dim(myTraining); dim(myTesting)
```

```
## [1] 15699   160
```

```
## [1] 3923  160
```

Cleaning the data

- Transformation 1: Cleaning NearZeroVariance Variables Run this code to view possible NZV Variables:

```
myDataNZV <- nearZeroVar(myTraining, saveMetrics=TRUE)
myNZVvars <- names(myTraining) %in% c("new_window", "kurtosis_roll_belt", "kurtosis_picth_bel
t",
"kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1", "skewness_yaw_belt",
"max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_arm", "stddev_roll_arm",
"var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_arm", "avg_yaw_arm",
"stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm", "kurtosis_picth_arm",
"kurtosis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm", "skewness_yaw_arm",
"max_roll_arm", "min_roll_arm", "min_pitch_arm", "amplitude_roll_arm", "amplitude_pitch_arm",
"kurtosis_roll_dumbbell", "kurtosis_picth_dumbbell", "kurtosis_yaw_dumbbell", "skewness_roll_
dumbbell",
"skewness_pitch_dumbbell", "skewness_yaw_dumbbell", "max_yaw_dumbbell", "min_yaw_dumbbell",
"amplitude_yaw_dumbbell", "kurtosis_roll_forearm", "kurtosis_picth_forearm", "kurtosis_yaw_fo
rearm",
"skewness_roll_forearm", "skewness_pitch_forearm", "skewness_yaw_forearm", "max_roll_forearm"
,
"max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm", "amplitude_roll_forearm",
"amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_forearm", "var_roll_forearm",
"avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_forearm", "avg_yaw_forearm",
"stddev_yaw_forearm", "var_yaw_forearm")
myTraining <- myTraining[!myNZVvars]

dim(myTraining)
```

```
## [1] 15699   100
```

- Transformation 2: Killing first column of Dataset - ID Removing first ID variable so that it does not interfer with ML Algorithms:

```
myTraining <- myTraining[c(-1)]
```

- Transformation 3: Cleaning Variables with too many NAs. For Variables that have more than a 60% threshold of NA's I'm going to leave them out:

```
trainingV3 <- myTraining #creating another subset to iterate in loop
for(i in 1:length(myTraining)) { #for every column in the training dataset
        if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .6 ) { #if n?? NAs > 60% of
 total observations
        for(j in 1:length(trainingV3)) {
            if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) ==1)  { #if the co
lumns are the same:
                trainingV3 <- trainingV3[ , -j] #Remove that column
            }
        }
    }
}
#To check the new N?? of observations
dim(trainingV3)
```

```
## [1] 15699    58
```

Setting back to our set:

```
myTraining <- trainingV3
rm(trainingV3)
```

Now let us do the exact same 3 transformations for myTesting and testing data sets.

```
clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58]) #already with classe column removed
myTesting <- myTesting[clean1]
testing <- testing[clean2]

#To check the new N?? of observations
dim(myTesting);dim(testing)
```

```
## [1] 3923   58
```

```
## [1] 20 57
```

Coercing the data into the same type.

```
for (i in 1:length(testing) ) {
        for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) ==1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}
#And to make sure Coertion really worked, simple smart ass technique:
testing <- rbind(myTraining[2, -58] , testing) #note row 2 does not mean anything, this will
 be removed right.. now:
testing <- testing[-1,]
```

Model 1: ML algorithms from Decision Tree

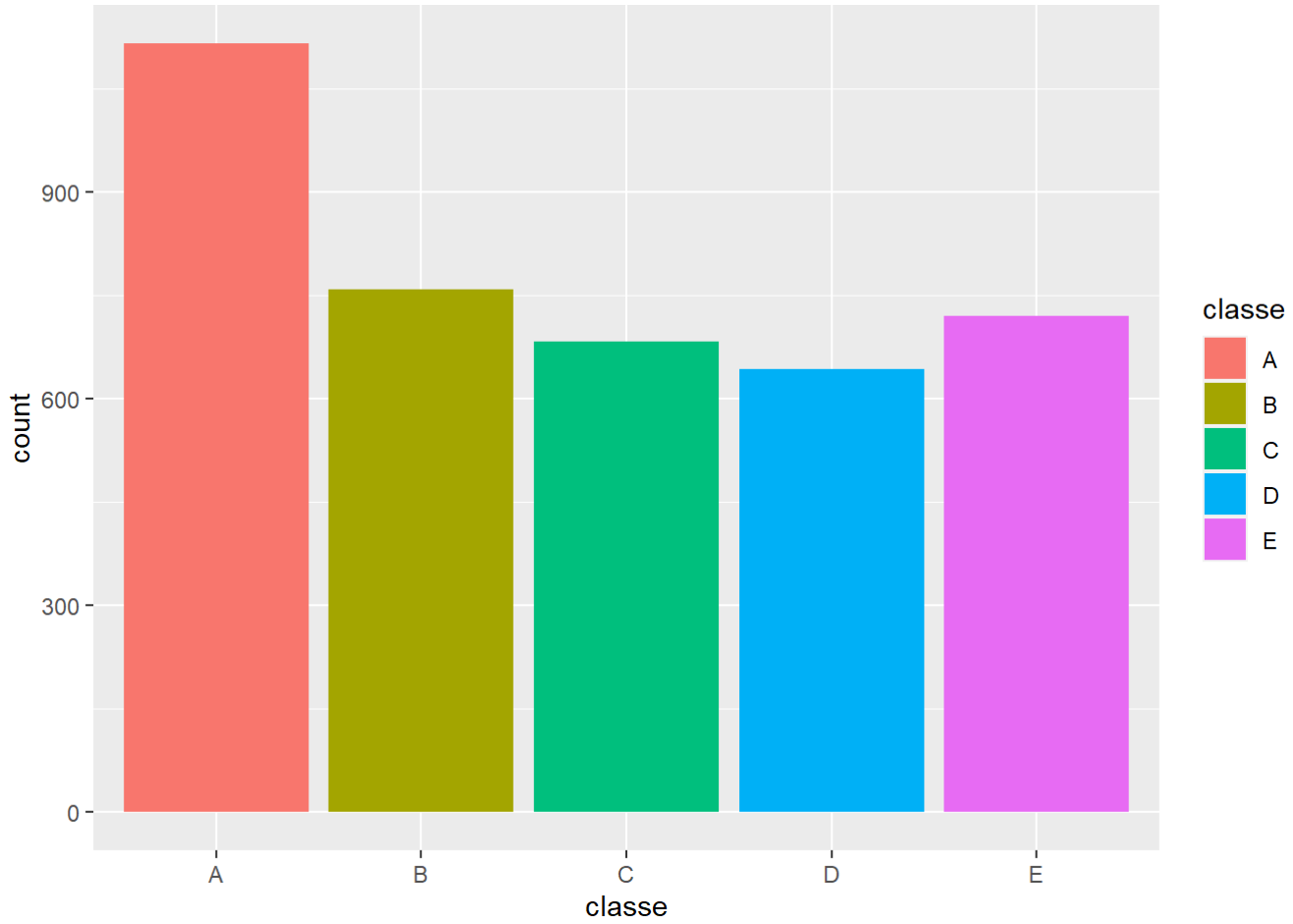
Exploratory analysis

```
head(myTraining)
```

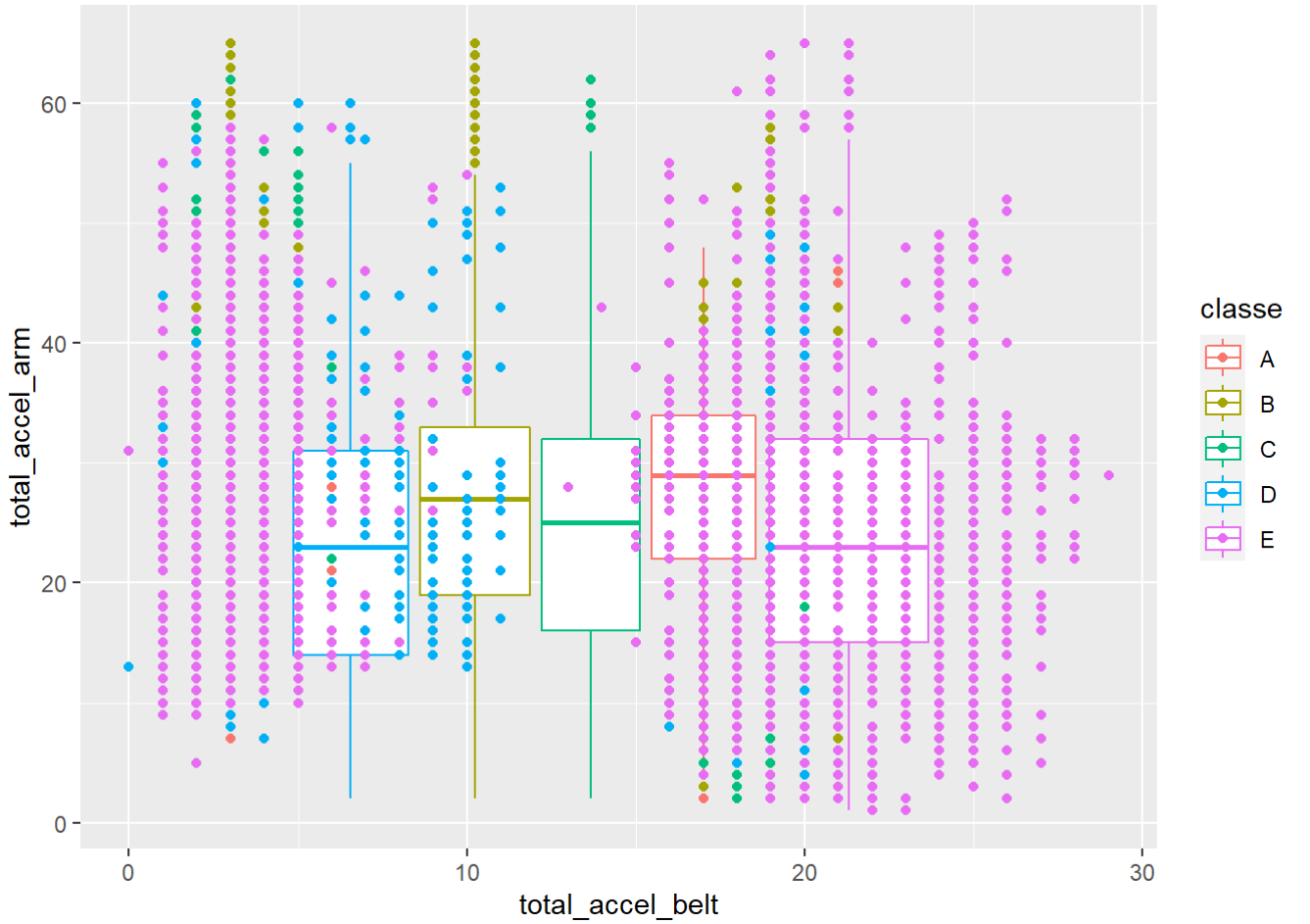
```
##   user_name raw_timestamp_part_1 raw_timestamp_part_2   cvtd_timestamp
## 2  carlitos           1323084231               808298 05/12/2011 11:23
## 3  carlitos           1323084231               820366 05/12/2011 11:23
## 4  carlitos           1323084232               120339 05/12/2011 11:23
## 5  carlitos           1323084232               196328 05/12/2011 11:23
## 7  carlitos           1323084232               368296 05/12/2011 11:23
## 9  carlitos           1323084232               484323 05/12/2011 11:23
##   num_window roll_belt pitch_belt yaw_belt total_accel_belt gyros_belt_x
## 2         11      1.41       8.07    -94.4                3         0.02
## 3         11      1.42       8.07    -94.4                3         0.00
## 4         12      1.48       8.05    -94.4                3         0.02
## 5         12      1.48       8.07    -94.4                3         0.02
## 7         12      1.42       8.09    -94.4                3         0.02
## 9         12      1.43       8.16    -94.4                3         0.02
##   gyros_belt_y gyros_belt_z accel_belt_x accel_belt_y accel_belt_z
## 2         0.00        -0.02          -22            4           22
## 3         0.00        -0.02          -20            5           23
## 4         0.00        -0.03          -22            3           21
## 5         0.02        -0.02          -21            2           24
## 7         0.00        -0.02          -22            3           21
## 9         0.00        -0.02          -20            2           24
##   magnet_belt_x magnet_belt_y magnet_belt_z roll_arm pitch_arm yaw_arm
## 2            -7           608          -311     -128      22.5    -161
## 3            -2           600          -305     -128      22.5    -161
## 4            -6           604          -310     -128      22.1    -161
## 5            -6           600          -302     -128      22.1    -161
## 7            -4           599          -311     -128      21.9    -161
## 9             1           602          -312     -128      21.7    -161
##   total_accel_arm gyros_arm_x gyros_arm_y gyros_arm_z accel_arm_x accel_arm_y
## 2              34        0.02       -0.02       -0.02        -290         110
## 3              34        0.02       -0.02       -0.02        -289         110
## 4              34        0.02       -0.03        0.02        -289         111
## 5              34        0.00       -0.03        0.00        -289         111
## 7              34        0.00       -0.03        0.00        -289         111
## 9              34        0.02       -0.03       -0.02        -288         109
##   accel_arm_z magnet_arm_x magnet_arm_y magnet_arm_z roll_dumbbell
## 2        -125         -369          337          513      13.13074
## 3        -126         -368          344          513      12.85075
## 4        -123         -372          344          512      13.43120
## 5        -123         -374          337          506      13.37872
## 7        -125         -373          336          509      13.12695
## 9        -122         -369          341          518      13.15463
##   pitch_dumbbell yaw_dumbbell total_accel_dumbbell gyros_dumbbell_x
## 2      -70.63751    -84.71065                  37                0
## 3      -70.27812    -85.14078                  37                0
## 4      -70.39379    -84.87363                  37                0
## 5      -70.42856    -84.85306                  37                0
## 7      -70.24757    -85.09961                  37                0
## 9      -70.42520    -84.91563                  37                0
##   gyros_dumbbell_y gyros_dumbbell_z accel_dumbbell_x accel_dumbbell_y
## 2            -0.02             0.00             -233               47
## 3            -0.02             0.00             -232               46
## 4            -0.02            -0.02             -232               48
## 5            -0.02             0.00             -233               48
## 7            -0.02             0.00             -232               47
## 9            -0.02             0.00             -232               47
##   accel_dumbbell_z magnet_dumbbell_x magnet_dumbbell_y magnet_dumbbell_z
```

```
## 2              -269              -555               296              -64
## 3              -270              -561               298              -63
## 4              -269              -552               303              -60
## 5              -270              -554               292              -68
## 7              -270              -551               295              -70
## 9              -269              -549               292              -65
##   roll_forearm pitch_forearm yaw_forearm total_accel_forearm gyros_forearm_x
## 2         28.3         -63.9        -153                  36            0.02
## 3         28.3         -63.9        -152                  36            0.03
## 4         28.1         -63.9        -152                  36            0.02
## 5         28.0         -63.9        -152                  36            0.02
## 7         27.9         -63.9        -152                  36            0.02
## 9         27.7         -63.8        -152                  36            0.03
##   gyros_forearm_y gyros_forearm_z accel_forearm_x accel_forearm_y
## 2            0.00           -0.02             192             203
## 3           -0.02            0.00             196             204
## 4           -0.02            0.00             189             206
## 5            0.00           -0.02             189             206
## 7            0.00           -0.02             195             205
## 9            0.00           -0.02             193             204
##   accel_forearm_z magnet_forearm_x magnet_forearm_y magnet_forearm_z classe
## 2            -216              -18              661              473      A
## 3            -213              -18              658              469      A
## 4            -214              -16              658              469      A
## 5            -214              -17              655              473      A
## 7            -215              -18              659              470      A
## 9            -214              -16              653              476      A
```

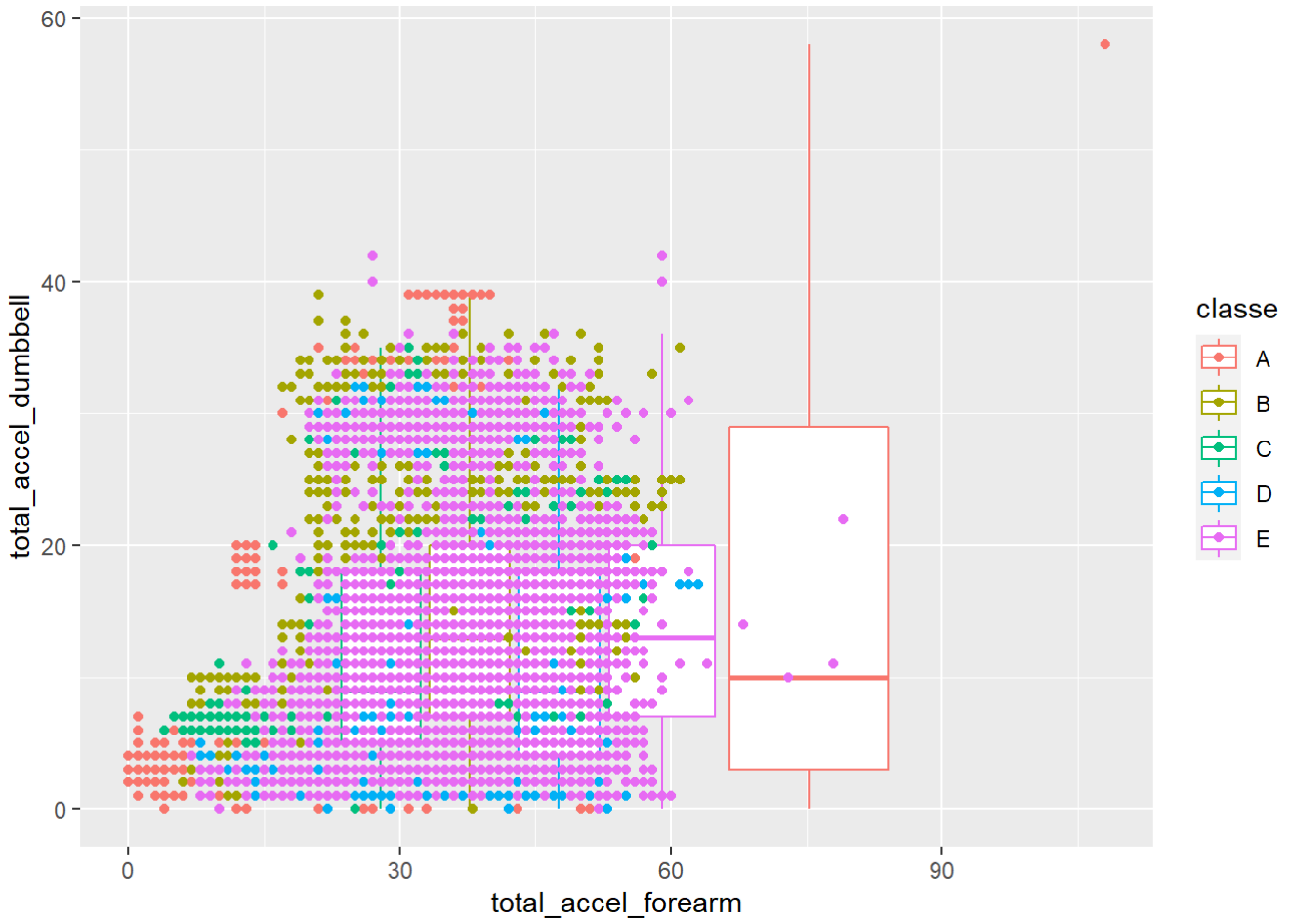
```
myTesting %>% ggplot(aes(classe, fill= classe))+ geom_bar()
```



```
myTraining%>% ggplot(aes(total_accel_belt,total_accel_arm, color= classe)) + geom_boxplot()+g
eom_point()
```

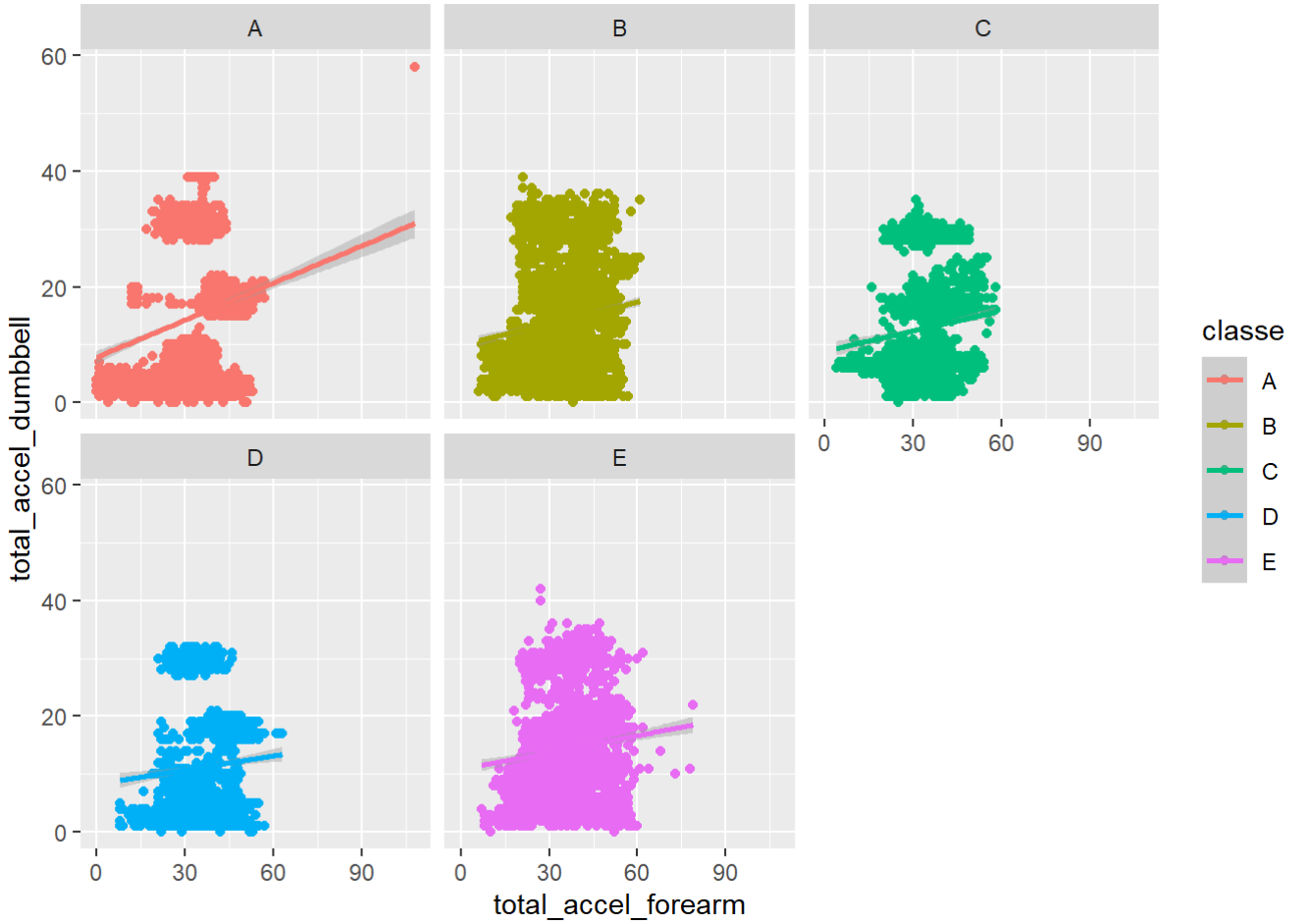



```
myTraining%>% ggplot(aes(total_accel_forearm,total_accel_dumbbell, color= classe)) + geom_box
plot()+geom_point()
```




```
myTraining%>% ggplot(aes(total_accel_forearm,total_accel_dumbbell, color= classe)) + geom_poi
nt() + geom_smooth(method = "lm") + facet_wrap(.~classe)
```

```
## `geom_smooth()` using formula 'y ~ x'
```

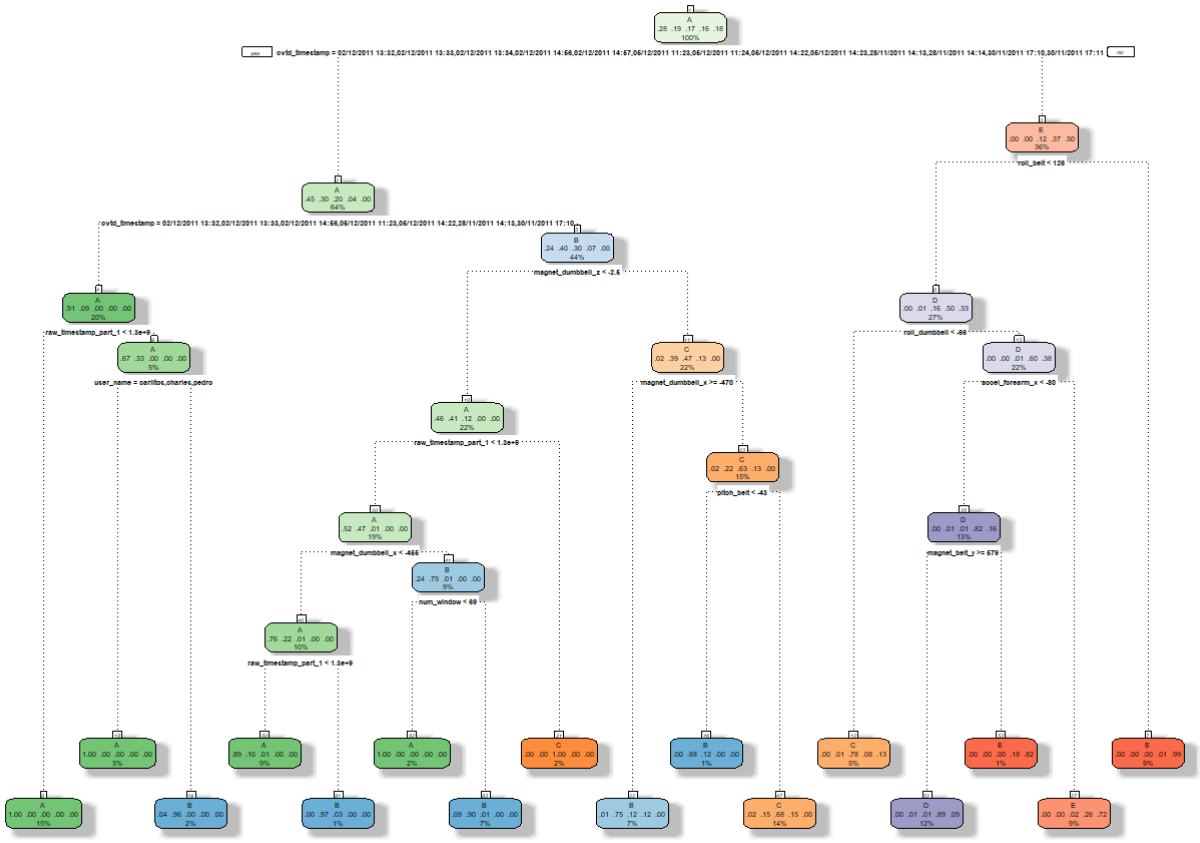


Model Fitting

```
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
```

To view the decision tree with fancy :

```
fancyRpartPlot(modFitA1)
```



Rattle 2020-Jun.-23 18:44:12 Ougoust_Brais

Predicting:

```
predictionsA1 <- predict(modFitA1, myTesting, type = "class")
```

Using confusion Matrix to test results:

```
myTesting$classe <- as.factor(myTesting$classe)
confusionMatrix(predictionsA1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1085   28    4    1    0
##          B   27  631   48   27    0
##          C    4   94  616  111   30
##          D    0    6   11  410   42
##          E    0    0    5   94  649
##
## Overall Statistics
##
##                Accuracy : 0.8644
##                  95% CI : (0.8533, 0.875)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8284
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9722   0.8314   0.9006   0.6376   0.9001
## Specificity            0.9882   0.9678   0.9262   0.9820   0.9691
## Pos Pred Value         0.9705   0.8608   0.7205   0.8742   0.8676
## Neg Pred Value         0.9889   0.9599   0.9778   0.9325   0.9773
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2766   0.1608   0.1570   0.1045   0.1654
## Detection Prevalence   0.2850   0.1868   0.2179   0.1196   0.1907
## Balanced Accuracy      0.9802   0.8996   0.9134   0.8098   0.9346
```

Model 2: Using ML algorithms for prediction: Random Forests

```
myTraining$classe <- as.factor(myTraining$classe)
modFitB1 <- randomForest(classe ~. , data=myTraining)
```

Predicting in-sample error

```
predictionsB1 <- predict(modFitB1, myTesting, type = "class")
```

Using confusion Matrix to test results:

```
confusionMatrix(predictionsB1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    0  759    2    0    0
##          C    0    0  682    3    0
##          D    0    0    0  640    0
##          E    0    0    0    0  721
##
## Overall Statistics
##
##                Accuracy : 0.9987
##                  95% CI : (0.997, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9984
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9971   0.9953   1.0000
## Specificity            1.0000   0.9994   0.9991   1.0000   1.0000
## Pos Pred Value         1.0000   0.9974   0.9956   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   0.9994   0.9991   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1935   0.1738   0.1631   0.1838
## Detection Prevalence   0.2845   0.1940   0.1746   0.1631   0.1838
## Balanced Accuracy      1.0000   0.9997   0.9981   0.9977   1.0000
```

So we can conclude that Random Forests yielded better Results.

Generating Files:

Finally, using the provided Test Set out-of-sample error. For Random Forests we use the following formula, which yielded a much better prediction in in-sample:

```
predictionsB2 <- predict(modFitB1, testing, type = "class")
```

Also que generate this function to create prediccion:

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(predictionsB2)
```