HumanActivityRecognition

1. Summary

This project analyze using devices to recognize human activity and to predict the manner people did exercise. This document shows these stages: getting and cleaning data, exploratory analysis and machine learning models. In Getting and cleaning the information, I've created three subsets: "train_train", "train_test" from pml-training and "test_test" from pml-test. I've cleaned those databases eliminating NA's columns. I've done exploratory data analysis with "train_train" subset and run the machine learning models. The other subsets were to test the model and predict new outcomes. The results with tree prediction models show an accuracy of 49%.

2. Getting and cleaning data

2.1 Getting the data

Firstly, I've got the information from the web

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

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

Then, I've partioned "traininghr" in two subsets: train_train and train_test

```
colnames_train <- colnames(traininghr)
colnames_test <- colnames(testinghr)

inTrain <- createDataPartition(y=traininghr$classe, p=0.7, list=FALSE)
train_train<- traininghr[inTrain, ]
train_test<- traininghr[-inTrain, ]
```

2.2 Cleaning the data

To run the machine learning models, I had to eliminate some columns that had many any NA's. Firstly, I configured those columns that have NA's

```
contarNAs <- function(x) {
        as.vector(apply(x, 2, function(x) length(which(!is.na(x)))))
}

columnaNA <- contarNAs(train_train)
drops <- c()
for (colNA in 1:length(columnaNA)) {
        if (columnaNA[colNA] < nrow(train_train)) {
                drops <- c(drops, colnames_train[colNA])
        }
}
```

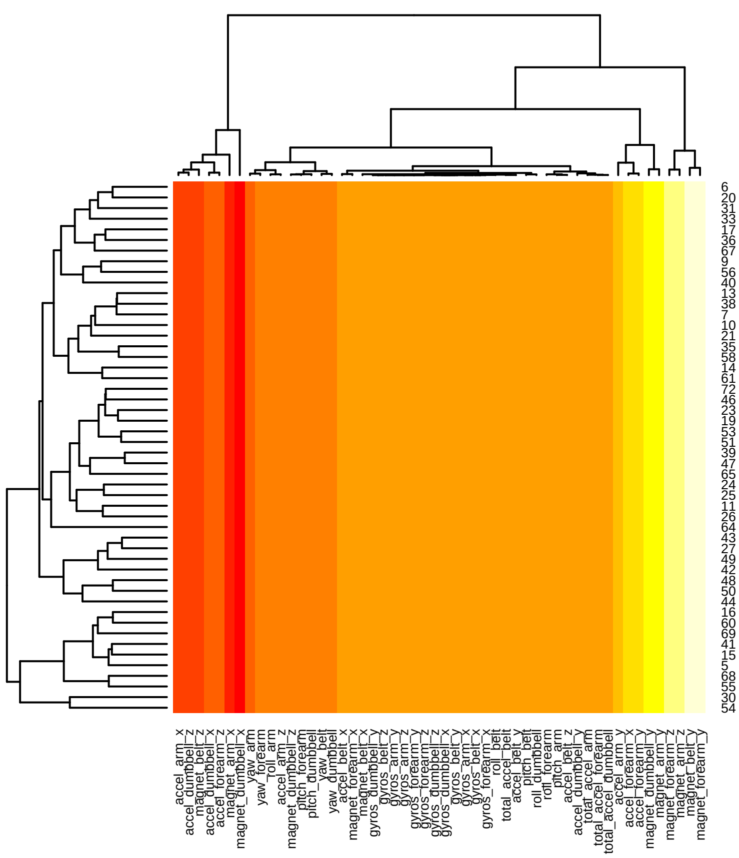
Then, I reduced the columns in the three subsets:

```
train_train<- train_train[,!(names(train_train) %in% drops)]
train_train<- train_train[,8:length(colnames(train_train))]

train_test<- train_test[,!(names(train_test) %in% drops)]
train_test<- train_test[,8:length(colnames(train_test))]

test_test<-testinghr
test_test<- test_test[,!(names(test_test) %in% drops)]
test_test<- test_test[,8:length(colnames(test_test))]
```

3. Exploratory Analysis

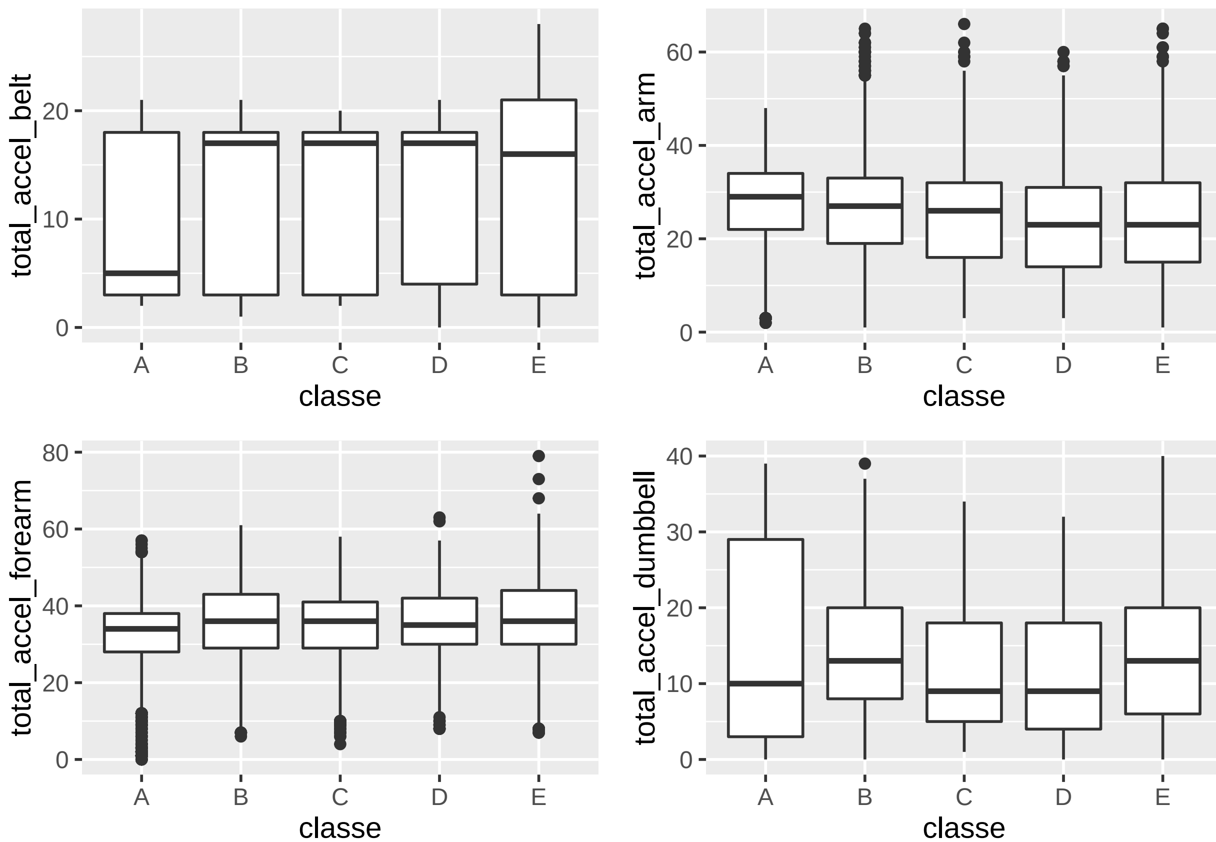
In this dataset and after reducing some NA´s variables, there are more than fifty possile variables. In that sense, I´ve decided to explore trouhg a heat map if there is any pattern or association.

```
ccc<-as.matrix(train_train[c(1:50), c(1:52)])
heatmap(ccc, cexCol=0.5, cexRow = 0.5)
```



After checking the variables there was not any strong relationship. Even so, there were more accel variables on the left and magnet variables on the right. Then, I've chosen the variable total acceleration to observe which classe has more acceleration.

```r
bp1<-qplot(classe, total_accel_belt, data=train_train, geom="boxplot")
bp2<-qplot(classe, total_accel_arm, data=train_train, geom="boxplot")
bp3<-qplot(classe, total_accel_forearm, data=train_train, geom="boxplot")
bp4<-qplot(classe, total_accel_dumbbell, data=train_train, geom="boxplot")
grid.arrange(bp1, bp2, bp3, bp4, ncol=2)
```



Again, there is no a strong distinction between classe. Even so, it seems that classe A has less acceleration.

4. Machine learning models

I've tried to use "lm method"" but R pointed that it was not the best election. Therefore, I've changed to decision tree models trough "rpart" method.

```r
tree1<-train(classe~., data=train_train, method="rpart", trControl=trainControl(method = "cv", number =
```

I've used cross validation (10 folds). We can see the results in those commands:

```r
print(tree1)
```

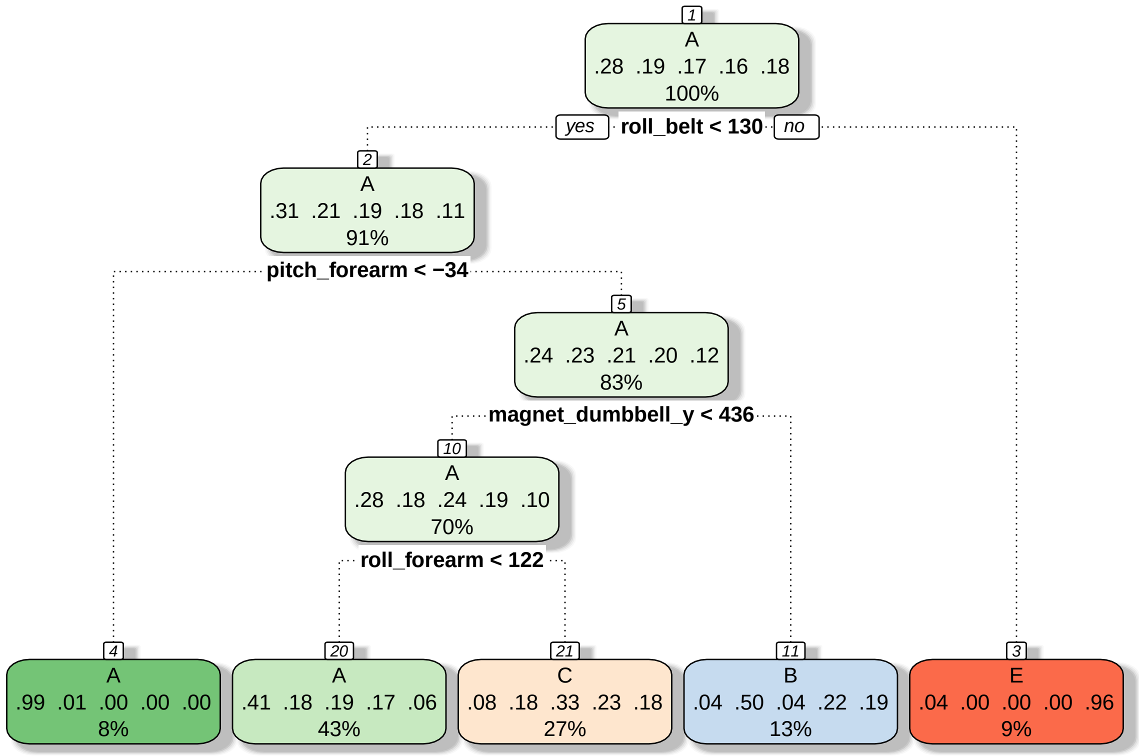
```
## CART
##
## 13737 samples
##    52 predictor
```

```
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12363, 12362, 12363, 12365, 12363, 12362, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03570339  0.5051294  0.35422869
##   0.06008205  0.4152987  0.20824261
##   0.11595972  0.3241689  0.06110194
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03570339.
```

```r
print(tree1$finalModel)
```

```
## 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< 129.5 12503 8644 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1114    6 A (0.99 0.0054 0 0 0) *
##      5) pitch_forearm>=-33.95 11389 8638 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 436.5 9573 6891 A (0.28 0.18 0.24 0.19 0.1)
##         20) roll_forearm< 121.5 5874 3486 A (0.41 0.18 0.19 0.17 0.057) *
##         21) roll_forearm>=121.5 3699 2479 C (0.079 0.18 0.33 0.23 0.18) *
##       11) magnet_dumbbell_y>=436.5 1816  901 B (0.038 0.5 0.044 0.22 0.19) *
##    3) roll_belt>=129.5 1234   47 E (0.038 0 0 0 0.96) *
```

```r
grid.newpage()
ff<-fancyRpartPlot(tree1$finalModel)
```



Rattle 2016−Sept.−09 13:36:47 ronny

```
ff
```

```
## NULL
```

Additionally I've applied a confusion matrix to see the prediction power of the model

```
prediccion1_1<-predict(tree1, train_test)
print(confusionMatrix(prediccion1_1, train_test$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1501  463  487  433  157
##          B   26  387   29  176  126
##          C  120  289  510  355  287
##          D    0    0    0    0    0
##          E   27    0    0    0  512
##
## Overall Statistics
##
##                Accuracy : 0.4945
##                  95% CI : (0.4816, 0.5073)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.3396
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8967  0.33977  0.49708   0.0000  0.47320
## Specificity            0.6343  0.92478  0.78370   1.0000  0.99438
## Pos Pred Value         0.4936  0.52016  0.32671      NaN  0.94991
## Neg Pred Value         0.9392  0.85372  0.88067   0.8362  0.89338
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2551  0.06576  0.08666   0.0000  0.08700
## Detection Prevalence   0.5167  0.12642  0.26525   0.0000  0.09159
## Balanced Accuracy      0.7655  0.63228  0.64039   0.5000  0.73379
```

The accuracy of the model is between 49-55% which is not a strong prediction model and it will require further research.

Even so, I've applied this algorithm to the test_test subset.

```
prediccion1_2<-predict(tree1, test_test)
test_test$prediccion1_2<-prediccion1_2
table(test_test$prediccion1_2)
```

```
##
##  A  B  C  D  E
## 11  0  9  0  0
```