HAR Weight Lifting

Frederic Bevia

28 juin 2016

Executive Summary

In this project of prÃ©dictive machine learning, using R, we have tried to build a classifier for qualitative activity recognition of weight lifting exercises, based on the “Weight Lifting Exercises Dataset” from the Human Activity Recognition site . To do so, we have evaluated six classifications algorithms including Random Forest and Support Vector Machine, using K-fold crossvalidation and features selection. After having choose the model presenting the best accuracy, we have predicted with that model the outcome of the twenty tests cases in the test dataset, as required.

The Problem

Today, using such new devices like smartphone and fitness bands, one thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do. On an expériment, described in the paper <ref http://groupware.les.inf.puc-rio.br /har#ixzz4CrzTgWBK (http://groupware.les.inf.puc-rio.br/har#ixzz4CrzTgWBK)>, six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: - Class A: exactly according to the specification
- Class B: throwing the elbows to the front - Class C: lifting the dumbbell only halfway
- Class D: lowering the dumbbell only halfway - Class E: throwing the hips to the front

(This is the “classe” variable in the training set).

Various data were collected from accelerometers on the belt, forearm, arm, and dumbell of the 6 participants.

The goal here, is to predict the manner in which they did the exercise, using the classe variable as outcome in the training dataset.

Note: All the code is in the annex of these document

Load and Prepare Data

Since the datasets are in the csv format, first of all, we open then in a spreadsheet (Excel or Calc) to have a peek on the datas. So we can note that there is a lot of missing datas, NA strings and even “#DIV/0!” strings. To correct that we will substitute all these strings by “NA” at the loading.

So after having loaded all the libraries required ..

we load the datasets:

```
# Load and prep the datas
# change for the working directory
# setwd("/media/fred/Donnees/Donnees/Coursera/Machine Learning/devoir")

# load the dataset

PmlTrain <- read.csv("pml-training.csv", stringsAsFactors=FALSE, na.strings=c("NA","#DIV/0!",""))
PmlTest <- read.csv("pml-testing.csv", stringsAsFactors=FALSE,na.strings=c("NA","#DIV/0!",""))

# View(PmlTrain)
# summary(PmlTrain)
# head(PmlTrain)

#View(PmlTest )
#summary(PmlTest)
#head(PmlTest)
```

Dimensions of the training dataset:

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

Quite Big nNumber of samples : 19622 !

Dimensions of the testing dataset:

Basic Exploratory Data Analysis

Tidying Data

Before everything else, we have to to treat the NA case, because there is lot of columns full of NA.

Frequency tables of the NAs :

Table continues below

0	19216	19217	19218	19220	19221	19225	19226	19227	19248
60	67	1	1	1	4	1	4	2	2

19293	19294	19296	19299	19300	19301	19622
1	1	2	1	4	2	6

Number of NA in the training dataset:

```
## [1] 1925102
```

Table continues below

0	19216	19217	19218	19220	19221	19225	19226	19227	19248
60	67	1	1	1	4	1	4	2	2

19293	19294	19296	19299	19300	19301	19622
1	1	2	1	4	2	6

Number of NA in the testing dataset:

```
## [1] 2000
```

0	20
60	100

So we get rid of the attributes whose columns are full of NA

```
PmlTrain<-PmlTrain[,colSums(is.na(PmlTrain)) < natrainmin]
PmlTest<-PmlTest[,colSums(is.na(PmlTest)) < natestmin]
```

We have now 60 variables for the training dataset, and 60 variables for the training dataset, But if we compare the attributes of the two sets, by intersecting and diff them, we can see that there is two differents attributes:

```
## [1] "classe"
```

```
## [1] "problem_id"
```

The two are in the last column. By the way , we can also see that the seven first attribute are unnecessary. so again, we get rid of them, and of course the last attribute of the testing set.

```
PmlTrain<-PmlTrain[,8:60]


PmlTest<-PmlTest[,8:59]
```

Now the dimensions of the training set are: 19622, 53

summary :

```
##    roll_belt        pitch_belt          yaw_belt       total_accel_belt
##  Min.   :-28.90   Min.   :-55.8000   Min.   :-180.00   Min.   : 0.00
##  1st Qu.:  1.10   1st Qu.:  1.7600   1st Qu.: -88.30   1st Qu.: 3.00
##  Median :113.00   Median :  5.2800   Median : -13.00   Median :17.00
##  Mean   : 64.41   Mean   :  0.3053   Mean   : -11.21   Mean   :11.31
##  3rd Qu.:123.00   3rd Qu.: 14.9000   3rd Qu.:  12.90   3rd Qu.:18.00
##  Max.   :162.00   Max.   : 60.3000   Max.   : 179.00   Max.   :29.00
##   gyros_belt_x        gyros_belt_y       gyros_belt_z
##  Min.   :-1.040000   Min.   :-0.64000   Min.   :-1.4600
##  1st Qu.:-0.030000   1st Qu.: 0.00000   1st Qu.:-0.2000
##  Median : 0.030000   Median : 0.02000   Median :-0.1000
##  Mean   :-0.005592   Mean   : 0.03959   Mean   :-0.1305
##  3rd Qu.: 0.110000   3rd Qu.: 0.11000   3rd Qu.:-0.0200
##  Max.   : 2.220000   Max.   : 0.64000   Max.   : 1.6200
##   accel_belt_x       accel_belt_y     accel_belt_z     magnet_belt_x
##  Min.   :-120.000   Min.   :-69.00   Min.   :-275.00   Min.   :-52.0
##  1st Qu.: -21.000   1st Qu.:  3.00   1st Qu.:-162.00   1st Qu.:  9.0
##  Median : -15.000   Median : 35.00   Median :-152.00   Median : 35.0
##  Mean   :  -5.595   Mean   : 30.15   Mean   : -72.59   Mean   : 55.6
##  3rd Qu.:  -5.000   3rd Qu.: 61.00   3rd Qu.:  27.00   3rd Qu.: 59.0
##  Max.   :  85.000   Max.   :164.00   Max.   : 105.00   Max.   :485.0
##  magnet_belt_y   magnet_belt_z       roll_arm         pitch_arm
##  Min.   :354.0   Min.   :-623.0   Min.   :-180.00   Min.   :-88.800
##  1st Qu.:581.0   1st Qu.:-375.0   1st Qu.: -31.77   1st Qu.:-25.900
##  Median :601.0   Median :-320.0   Median :   0.00   Median :  0.000
##  Mean   :593.7   Mean   :-345.5   Mean   :  17.83   Mean   : -4.612
##  3rd Qu.:610.0   3rd Qu.:-306.0   3rd Qu.:  77.30   3rd Qu.: 11.200
##  Max.   :673.0   Max.   : 293.0   Max.   : 180.00   Max.   : 88.500
##     yaw_arm          total_accel_arm  gyros_arm_x        gyros_arm_y
##  Min.   :-180.0000   Min.   : 1.00   Min.   :-6.37000   Min.   :-3.4400
##  1st Qu.: -43.1000   1st Qu.:17.00   1st Qu.:-1.33000   1st Qu.:-0.8000
##  Median :   0.0000   Median :27.00   Median : 0.08000   Median :-0.2400
##  Mean   :  -0.6188   Mean   :25.51   Mean   : 0.04277   Mean   :-0.2571
##  3rd Qu.:  45.8750   3rd Qu.:33.00   3rd Qu.: 1.57000   3rd Qu.: 0.1400
##  Max.   : 180.0000   Max.   :66.00   Max.   : 4.87000   Max.   : 2.8400
##   gyros_arm_z       accel_arm_x       accel_arm_y      accel_arm_z
```

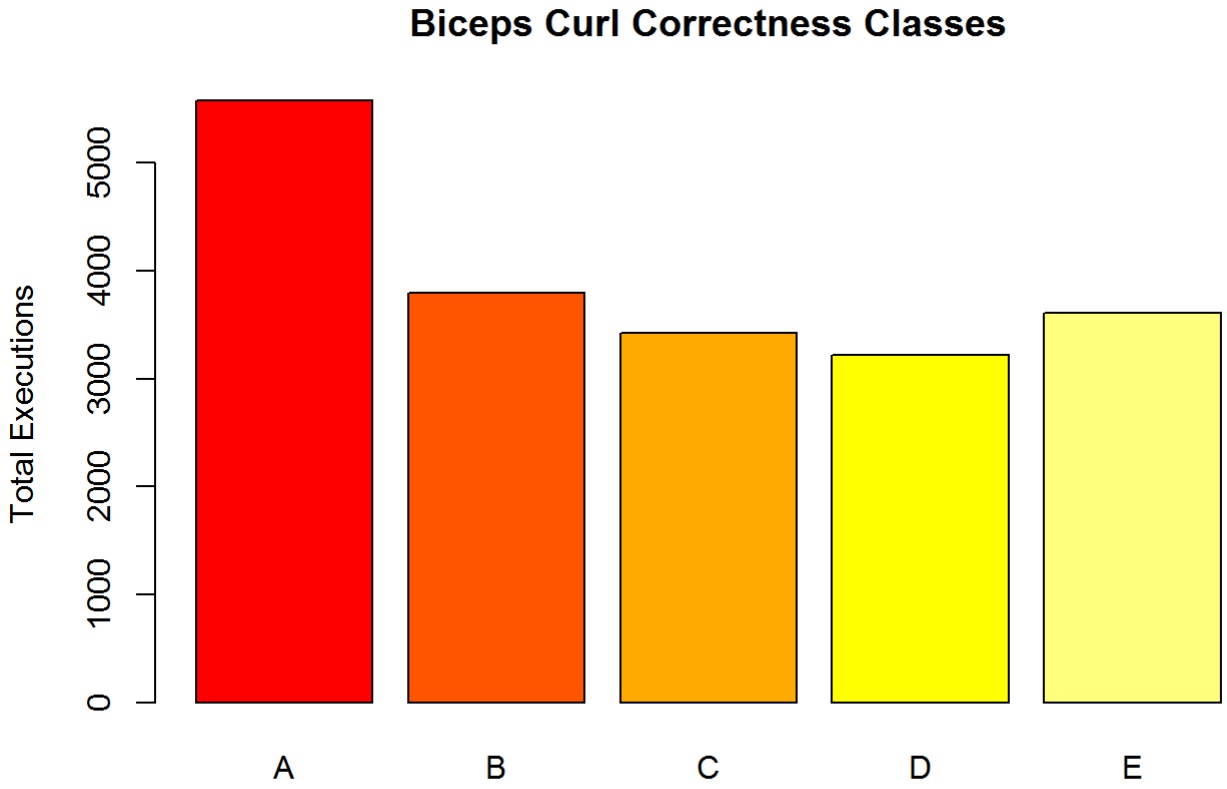
```
##  Min.   :-2.3300   Min.   :-404.00   Min.   :-318.0   Min.   :-636.00
##  1st Qu.:-0.0700   1st Qu.:-242.00   1st Qu.: -54.0   1st Qu.:-143.00
##  Median : 0.2300   Median : -44.00   Median :  14.0   Median : -47.00
##  Mean   : 0.2695   Mean   : -60.24   Mean   :  32.6   Mean   : -71.25
##  3rd Qu.: 0.7200   3rd Qu.:  84.00   3rd Qu.: 139.0   3rd Qu.:  23.00
##  Max.   : 3.0200   Max.   : 437.00   Max.   : 308.0   Max.   : 292.00
##   magnet_arm_x     magnet_arm_y     magnet_arm_z    roll_dumbbell
##  Min.   :-584.0   Min.   :-392.0   Min.   :-597.0   Min.   :-153.71
##  1st Qu.:-300.0   1st Qu.:  -9.0   1st Qu.: 131.2   1st Qu.: -18.49
##  Median : 289.0   Median : 202.0   Median : 444.0   Median :  48.17
##  Mean   : 191.7   Mean   : 156.6   Mean   : 306.5   Mean   :  23.84
##  3rd Qu.: 637.0   3rd Qu.: 323.0   3rd Qu.: 545.0   3rd Qu.:  67.61
##  Max.   : 782.0   Max.   : 583.0   Max.   : 694.0   Max.   : 153.55
##  pitch_dumbbell     yaw_dumbbell      total_accel_dumbbell
##  Min.   :-149.59   Min.   :-150.871   Min.   : 0.00
##  1st Qu.: -40.89   1st Qu.: -77.644   1st Qu.: 4.00
##  Median : -20.96   Median :  -3.324   Median :10.00
##  Mean   : -10.78   Mean   :   1.674   Mean   :13.72
##  3rd Qu.:  17.50   3rd Qu.:  79.643   3rd Qu.:19.00
##  Max.   : 149.40   Max.   : 154.952   Max.   :58.00
##  gyros_dumbbell_x    gyros_dumbbell_y   gyros_dumbbell_z
##  Min.   :-204.0000   Min.   :-2.10000   Min.   : -2.380
##  1st Qu.:  -0.0300   1st Qu.:-0.14000   1st Qu.: -0.310
##  Median :   0.1300   Median : 0.03000   Median : -0.130
##  Mean   :   0.1611   Mean   : 0.04606   Mean   : -0.129
##  3rd Qu.:   0.3500   3rd Qu.: 0.21000   3rd Qu.:  0.030
##  Max.   :   2.2200   Max.   :52.00000   Max.   :317.000
##  accel_dumbbell_x  accel_dumbbell_y  accel_dumbbell_z  magnet_dumbbell_x
##  Min.   :-419.00   Min.   :-189.00   Min.   :-334.00   Min.   :-643.0
##  1st Qu.: -50.00   1st Qu.:  -8.00   1st Qu.:-142.00   1st Qu.:-535.0
##  Median :  -8.00   Median :  41.50   Median :  -1.00   Median :-479.0
##  Mean   : -28.62   Mean   :  52.63   Mean   : -38.32   Mean   :-328.5
##  3rd Qu.:  11.00   3rd Qu.: 111.00   3rd Qu.:  38.00   3rd Qu.:-304.0
##  Max.   : 235.00   Max.   : 315.00   Max.   : 318.00   Max.   : 592.0
##  magnet_dumbbell_y magnet_dumbbell_z  roll_forearm       pitch_forearm
##  Min.   :-3600     Min.   :-262.00   Min.   :-180.0000   Min.   :-72.50
```

```
##  1st Qu.:  231     1st Qu.: -45.00   1st Qu.:  -0.7375   1st Qu.:  0.00
##  Median :  311     Median :  13.00   Median :  21.7000   Median :  9.24
##  Mean   :  221     Mean   :  46.05   Mean   :  33.8265   Mean   : 10.71
##  3rd Qu.:  390     3rd Qu.:  95.00   3rd Qu.: 140.0000   3rd Qu.: 28.40
##  Max.   :  633     Max.   : 452.00   Max.   : 180.0000   Max.   : 89.80
##   yaw_forearm      total_accel_forearm gyros_forearm_x
##  Min.   :-180.00   Min.   :  0.00      Min.   :-22.000
##  1st Qu.: -68.60   1st Qu.: 29.00      1st Qu.: -0.220
##  Median :   0.00   Median : 36.00      Median :  0.050
##  Mean   :  19.21   Mean   : 34.72      Mean   :  0.158
##  3rd Qu.: 110.00   3rd Qu.: 41.00      3rd Qu.:  0.560
##  Max.   : 180.00   Max.   :108.00      Max.   :  3.970
##  gyros_forearm_y     gyros_forearm_z    accel_forearm_x   accel_forearm_y
##  Min.   : -7.02000   Min.   : -8.0900   Min.   :-498.00   Min.   :-632.0
##  1st Qu.: -1.46000   1st Qu.: -0.1800   1st Qu.:-178.00   1st Qu.:  57.0
##  Median :  0.03000   Median :  0.0800   Median : -57.00   Median : 201.0
##  Mean   :  0.07517   Mean   :  0.1512   Mean   : -61.65   Mean   : 163.7
##  3rd Qu.:  1.62000   3rd Qu.:  0.4900   3rd Qu.:  76.00   3rd Qu.: 312.0
##  Max.   :311.00000   Max.   :231.0000   Max.   : 477.00   Max.   : 923.0
##  accel_forearm_z   magnet_forearm_x  magnet_forearm_y magnet_forearm_z
##  Min.   :-446.00   Min.   :-1280.0   Min.   :-896.0   Min.   :-973.0
##  1st Qu.:-182.00   1st Qu.: -616.0   1st Qu.:   2.0   1st Qu.: 191.0
##  Median : -39.00   Median : -378.0   Median : 591.0   Median : 511.0
##  Mean   : -55.29   Mean   : -312.6   Mean   : 380.1   Mean   : 393.6
##  3rd Qu.:  26.00   3rd Qu.:  -73.0   3rd Qu.: 737.0   3rd Qu.: 653.0
##  Max.   : 291.00   Max.   :  672.0   Max.   :1480.0   Max.   :1090.0
##  classe
##  A:5580
##  B:3797
##  C:3422
##  D:3216
##  E:3607
##
```

Frequencies of the classes

A **B** **C** **D** **E**

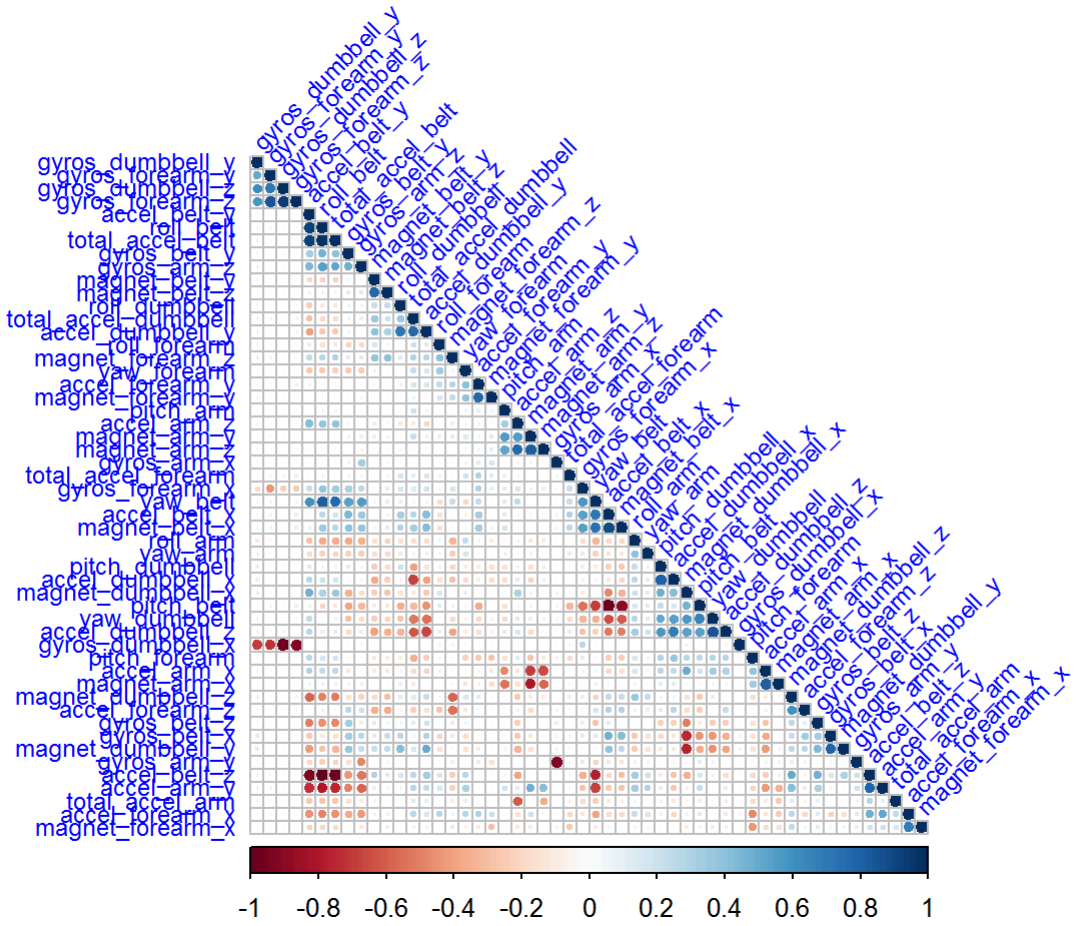
A	B	C	D	E
5580	3797	3422	3216	3607



we can see that there is a relatively consistant distribution among the classes, except for the classe A whith a higher value, which mean that the partcipant did this move,the correct biceps curl, more than the others bad moves.

features selection

In order to see if we can still lessen the number of sinificant variables lets do some feature selection, with the correlation matrix:



We can see that there is somes attributes wich are correlated. Let's compute which of them are highly correlated (ideally >0.75):

```
hCor <- findCorrelation(corMx, cutoff=0.75)



# print indexes of highly correlated attributes
print(hCor)
```

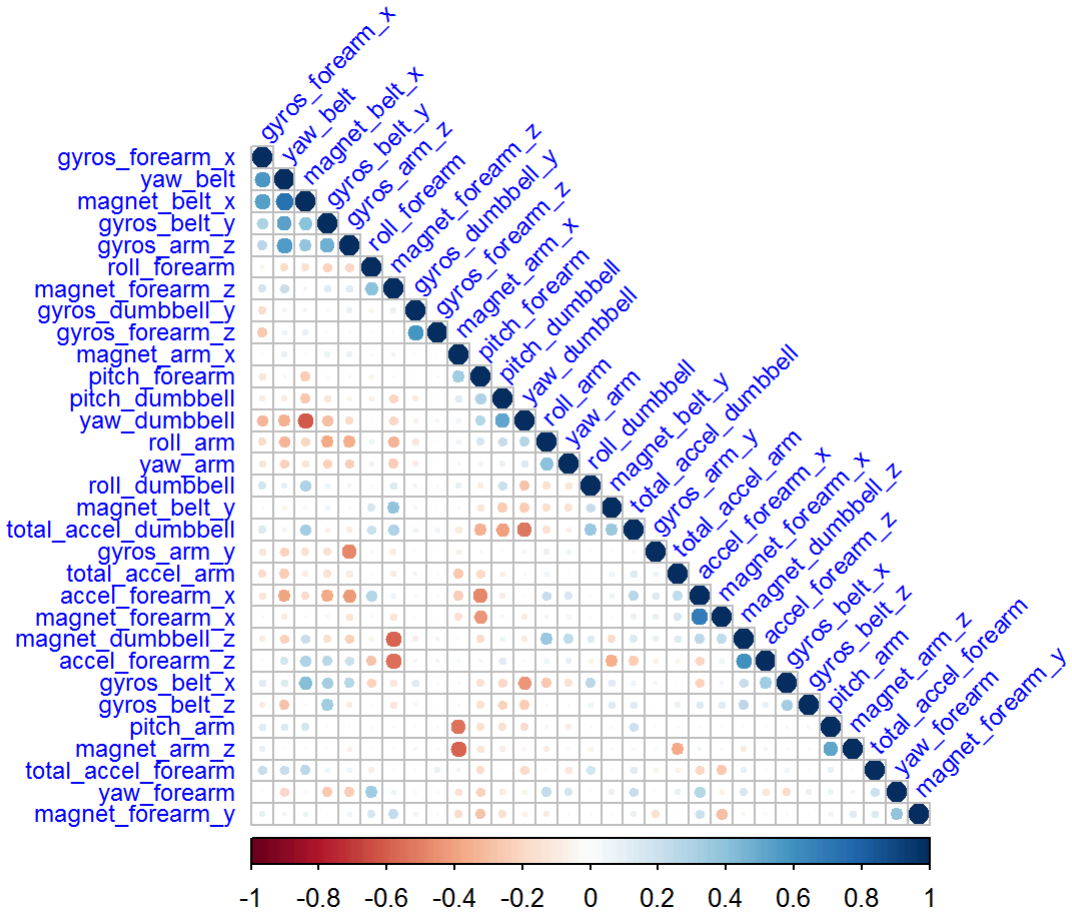
```
##  [1] 10  1  9 22  4 36  8  2 37 35 38 21 34 23 25 13 48 45 31 33 18
```

There is `r length(hCor)' attributes that are highly corelated and which can be taken out:

```
PmlTrain<-PmlTrain[,-hCor]
PmlTest <-PmlTest[,-hCor]
```

Computing again the correlation matrix,

```
## [1] 19622    32
```



we can see now that there is less correlation among the 31 attributes remaning.

Now, we can evaluate somes algorithms and build models

Building Models

Were going to create some models of the data and estimate their accuracy on unseen data.

To do so we're going to

- Set-up the test to use 5-fold cross validation.

- Build 6 different models to predict Classes of Biceps movement
- Select the best model, upon is accuracy.

In order to test different type of algorithms, i choose six well know classifiers which are representatives of differents methods:

- CART: classification and regression tree
- LVQ: Learning Vector Quantization (a special case of neural network)
- LDA: Linear discriminant analysis
- GBM: Gradient Boosted Machine
- RF: Random Forest
- SVM: Support Vector Machine

For the cross-validation, each model is tuned and evaluated using 3 repeats of 5-fold, thanks to the caret package.

Note: Initialy, i would use 5 repeat and 10-fold, but on my laptop, for somes of the algorithms, it was not possible (infinite time or Rstudio out), even with the doParallel package which permit to use the for(4) cores of my PC.

```
#  Use the cores Luke !!

registerDoParallel(cores=4)
```

To insure the accuracy the model which we will select, we are partionning the training test, and putting aside a part of the training dataset. We will use these test set to cross-validate the selected model before using it to predict the classes for the initial testing set.

```
#set.seed(1960)
inTrain <- createDataPartition(PmlTrain$classe, p = 0.75,list=FALSE)
training <- PmlTrain[ inTrain,]
testing <- PmlTrain[-inTrain,]
```

Predictives Algorithms Evalution

```
# prepare training scheme
control <- trainControl(method="repeatedcv", number=5, repeats=3)



# train the CART model
set.seed(1960)
modelCart <- train(classe~., data=training , method="rpart", metric="Accuracy", trControl=control)

# train the LVQ model <- very long to execute on my PC
set.seed(1960)
modelLvq <- train(classe~., data=training , method="lvq",metric="Accuracy", trControl=control)

# train the LDA model
set.seed(1960)
modelLda <- train(classe~., data=training , method="lda",metric="Accuracy",trControl=control)

# train the GBM model
set.seed(1960)
modelGbm <- train(classe~., data=training , method="gbm",metric="Accuracy", trControl=control, verbose=FALSE)

# train the RF model
set.seed(1960)
modelRF<- train(classe~., data=training , method="rf",metric="Accuracy",trControl=control)

# train the SVM model
set.seed(1960)
modelSvm <- train(classe~., data=training , method="svmRadial",metric="Accuracy", trControl=control)
```

Results

- **values**:

  Table continues below

CART~Accuracy	CART~Kappa	LVQ~Accuracy	LVQ~Kappa	LDA~Accuracy

CART~Accuracy	CART~Kappa	LVQ~Accuracy	LVQ~Kappa	LDA~Accuracy
0.5386	0.4202	0.4716	0.3354	0.5763
0.5323	0.4125	0.5088	0.3747	0.5754
0.5804	0.4715	0.4794	0.3415	0.5943
0.5627	0.4486	0.4917	0.3556	0.5868
0.5639	0.45	0.4704	0.3353	0.5805
0.5406	0.4225	0.4815	0.3493	0.5732
0.5567	0.4405	0.4613	0.316	0.5747
0.5442	0.4267	0.4983	0.3638	0.5867
0.564	0.4517	0.4791	0.343	0.5973
0.5591	0.4437	0.4793	0.3456	0.5822
0.5542	0.4425	0.4652	0.3301	0.5745
0.5382	0.4188	0.4553	0.3101	0.5695
0.569	0.4559	0.5071	0.3784	0.5815
0.5674	0.4565	0.4781	0.3413	0.5804
0.5391	0.4201	0.4861	0.3463	0.5751

Table continues below

LDA~Kappa	GBM~Accuracy	GBM~Kappa	RF~Accuracy	RF~Kappa	SVM~Accuracy
0.4634	0.947	0.9329	0.9908	0.9884	0.9069
0.463	0.9378	0.9213	0.9905	0.988	0.8984
0.4867	0.947	0.9329	0.9878	0.9845	0.9079

LDA~Kappa	GBM~Accuracy	GBM~Kappa	RF~Accuracy	RF~Kappa	SVM~Accuracy	SVM~Kappa
0.4781	0.9521	0.9394	0.9864	0.9828	0.912	0.8819
0.4692	0.9436	0.9287	0.9922	0.9901	0.9083	0.8712
0.4603	0.9412	0.9256	0.9884	0.9854	0.9157	0.8832
0.4617	0.947	0.9329	0.9912	0.9888	0.9113	0.8885
0.4768	0.9477	0.9338	0.9884	0.9854	0.9201	0.8838
0.4903	0.948	0.9342	0.9915	0.9893	0.907	0.8932
0.4705	0.9429	0.9278	0.9918	0.9897	0.9147	0.8876
0.4616	0.9457	0.9313	0.9915	0.9893	0.9114	
0.4552	0.9399	0.9239	0.9912	0.9888	0.9103	
0.4701	0.9443	0.9295	0.9925	0.9905	0.9069	
0.4679	0.9514	0.9385	0.9905	0.988	0.9181	
0.4604	0.9412	0.9256	0.9888	0.9858	0.9049	

SVM~Kappa
0.8987
0.8821
0.8919
0.8876
0.8863
0.8821
0.8962
0.8794

- **call**: `summary.resamples(object = results)`
- **statistics**:
  - **Accuracy**:

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
CART	0.5323	0.5398	0.5567	0.554	0.5639	0.5804	0
LVQ	0.4553	0.471	0.4793	0.4809	0.4889	0.5088	0
LDA	0.5695	0.5749	0.5804	0.5806	0.5844	0.5973	0
GBM	0.9378	0.9421	0.9457	0.9451	0.9473	0.9521	0
RF	0.9864	0.9886	0.9908	0.9902	0.9915	0.9925	0
SVM	0.8984	0.9069	0.9103	0.9103	0.9134	0.9201	0

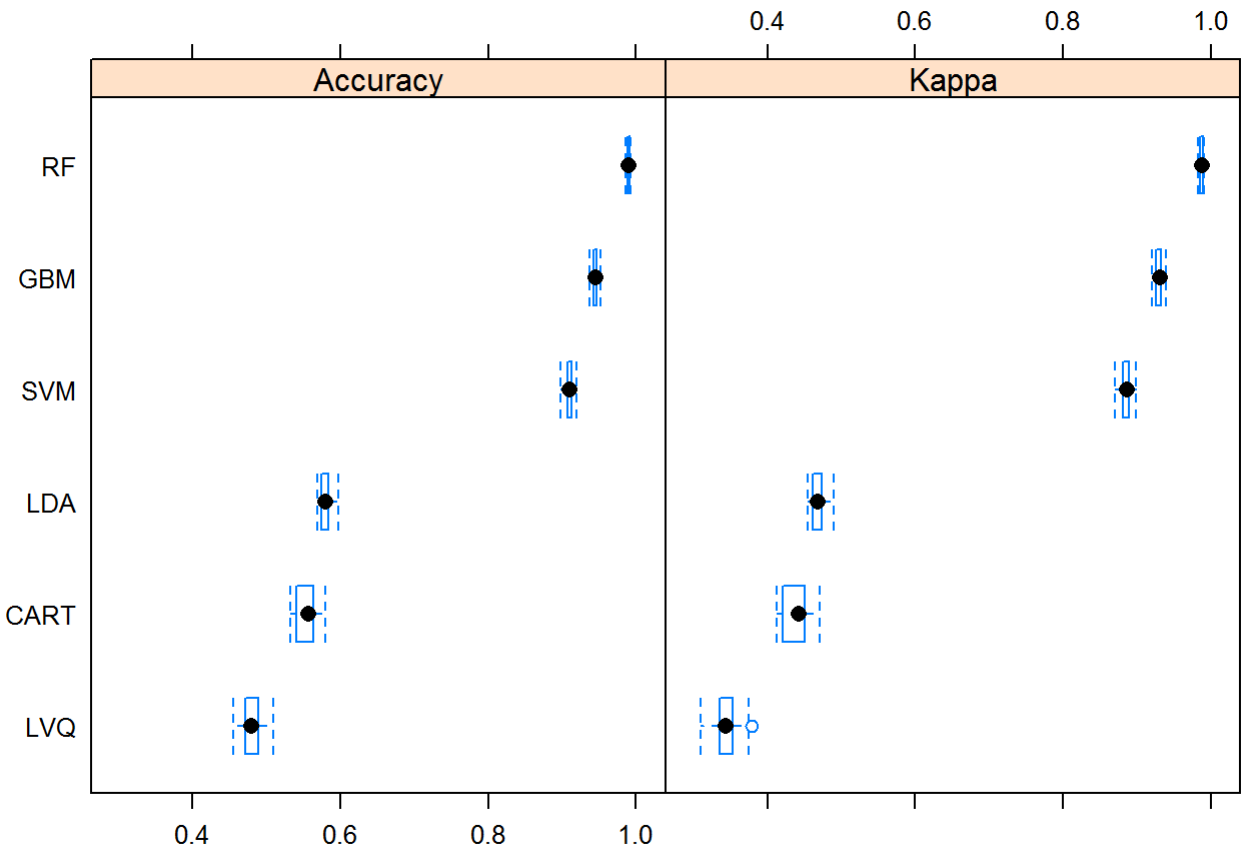
  - **Kappa**:

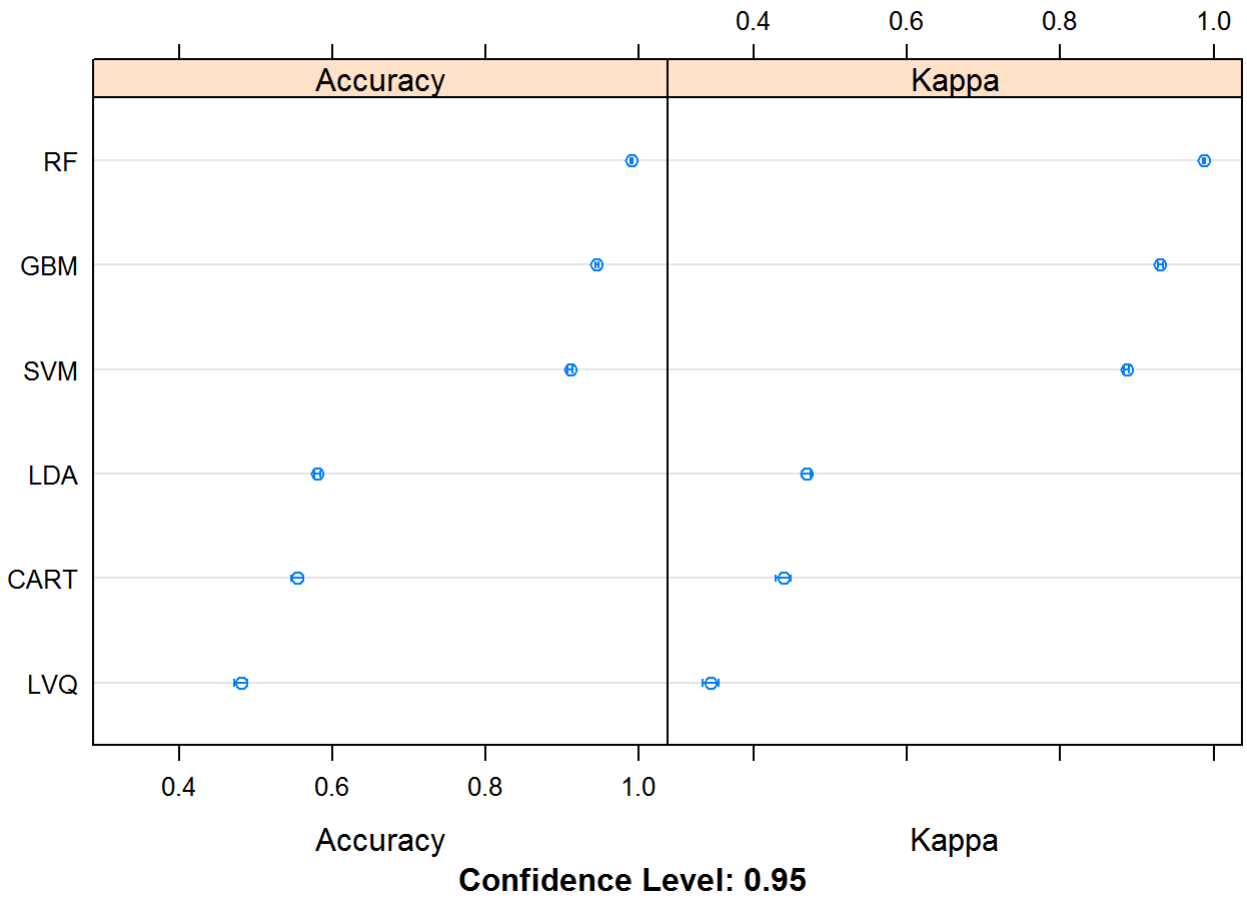
	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
CART	0.4125	0.4214	0.4425	0.4388	0.4508	0.4715	0
LVQ	0.3101	0.3353	0.343	0.3444	0.3525	0.3784	0
LDA	0.4552	0.4616	0.4679	0.469	0.4737	0.4903	0
GBM	0.9213	0.9267	0.9313	0.9306	0.9334	0.9394	0
RF	0.9828	0.9856	0.9884	0.9877	0.9893	0.9905	0
SVM	0.8712	0.8821	0.8863	0.8863	0.8902	0.8987	0

- **models**: *CART*, *LVQ*, *LDA*, *GBM*, *RF* and *SVM*
- **metrics**: *Accuracy* and *Kappa*
- **methods**:

CART	LVQ	LDA	GBM	RF	SVM
rpart	lvq	lda	gbm	rf	svmRadial


Accuracy
Kappa
RF
GBM
SVM
LDA
CART
LVQ
0.4
0.6
0.8
1.0



Both for the accuracy (o.99) and the Kappa (near 1), the model which is the best is the one produced by the **Random Forrest** Algorithm. As we can see all the more on the boxplot and the dotplot. We can also see that the tree first algorithms, cART, LVQ and LDA are not performing very well, while GBM and SVM are near the performances of the RF.

So we select these model to do the predictions .

Importance

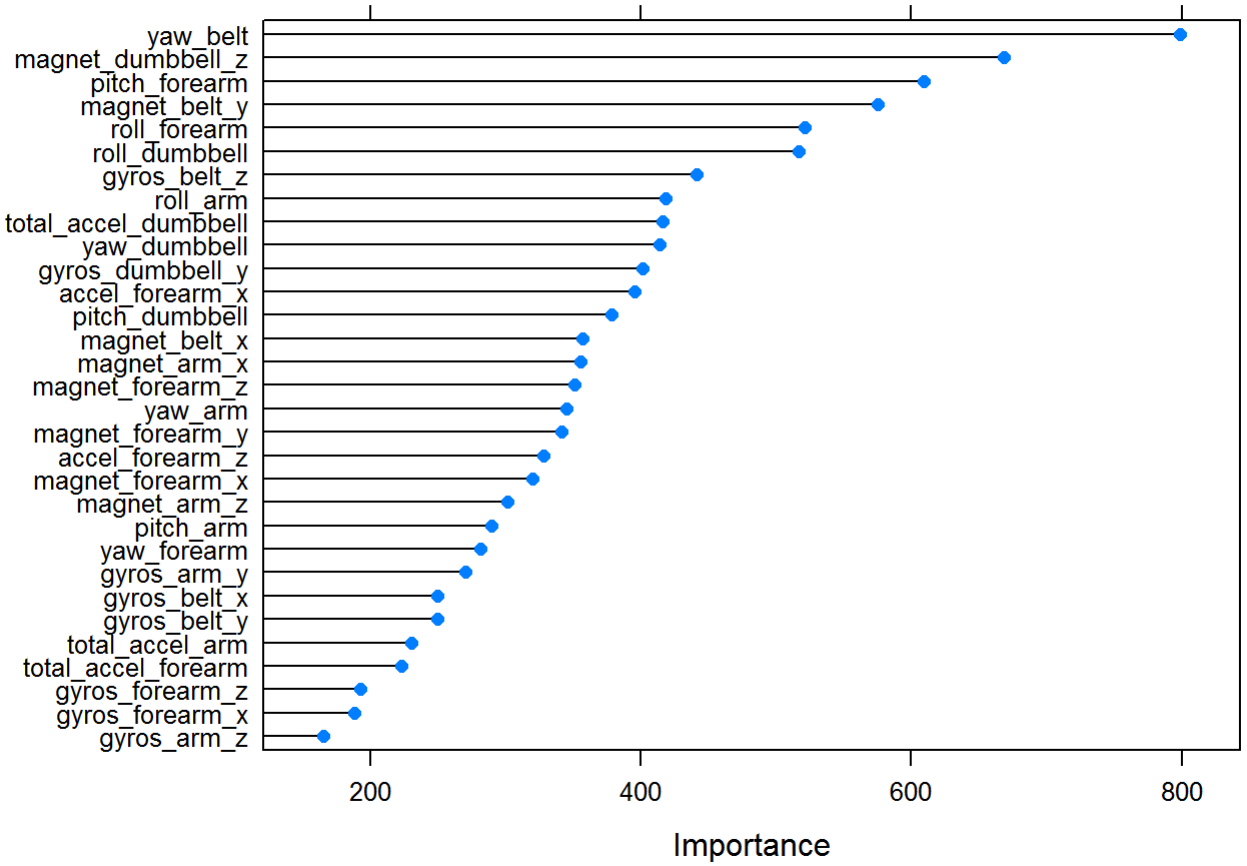
Before doing so, we can look at the importance of the variables:

- **importance**:

	Overall
yaw_belt	799.1
gyros_belt_x	250.3
gyros_belt_y	249.6
gyros_belt_z	441.4
magnet_belt_x	357.1
magnet_belt_y	575.3
roll_arm	418.9
pitch_arm	289.7
yaw_arm	345.6
total_accel_arm	230.9
gyros_arm_y	270.9
gyros_arm_z	165.4
magnet_arm_x	355.6
magnet_arm_z	301.9
roll_dumbbell	517.2
pitch_dumbbell	378.5
yaw_dumbbell	414.5
total_accel_dumbbell	416.7
gyros_dumbbell_y	401.4
magnet_dumbbell_z	668.7
roll_forearm	521.3

	Overall
pitch_forearm	609.1
yaw_forearm	281.9
total_accel_forearm	223
gyros_forearm_x	188.6
gyros_forearm_z	193
accel_forearm_x	395.8
accel_forearm_z	328.1
magnet_forearm_x	319.9
magnet_forearm_y	341.5
magnet_forearm_z	351.6

- **model**: rf
- **calledFrom**: varImp



We can see here that we could have eliminated again some variables, the 3 or 4 last one.

Cross Validation againts the validation set

Now, before applying the model to the real test set, we are testing it again the part of the training set that we have pu apart for that:

```
prediction <- predict(modelRF, testing)
confMX<-confusionMatrix(testing$classe, prediction)

print(confMX)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    5  939    5    0    0
##          C    0    6  846    3    0
##          D    0    0   19  784    1
##          E    0    0    0    3  898
##
## Overall Statistics
##
##                Accuracy : 0.9914
##                  95% CI : (0.9884, 0.9938)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9892
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9937   0.9724   0.9924   0.9989
## Specificity            1.0000   0.9975   0.9978   0.9951   0.9993
## Pos Pred Value         1.0000   0.9895   0.9895   0.9751   0.9967
## Neg Pred Value         0.9986   0.9985   0.9941   0.9985   0.9998
## Prevalence             0.2855   0.1927   0.1774   0.1611   0.1833
## Detection Rate         0.2845   0.1915   0.1725   0.1599   0.1831
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9982   0.9956   0.9851   0.9938   0.9991
```

```
pander(postResample(prediction, testing$classe))
```

Accuracy	Kappa

Accuracy	Kappa
0.9914	0.9892

We can see that the accuracy and the kappa are verigood and on the confusion matrix there is not much misclassification, curiuouly just on the second diagonal, under the first diagonal.

Testing the Model

We're applying the model to the initial test Dataset, to predict the classes of the 20 samples

```
#
predictedClasses <- predict(modelRF, PmlTest)

print(predictedClasses)
```

```
##  [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
```

Conclusions

In these project we have evaluated several algorith to produce a model with good accuracy, and even if it's the Random Forest which is the better, two others algorithms are very near in term of accuracy, the GBM and a SVM. But to be complete, we should take into account the fact that the most perfoming algorithms are very greedy in cpu and RAM, and most consuming in time, and because of that I couldn't do more than 3 repeats on 5 fold in the repeated cross-validation K-fold method.Also, we might improve the features selection.

Annex

References

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

http://www.academia.edu/7619059/Human_Activity_Recognition_using_machine_learning (http://www.academia.edu/7619059/Human_Activity_Recognition_using_machine_learning) http://link.springer.com/article/10.1007%2Fs12652-011-0068-9#page-1 (http://link.springer.com/article/10.1007%2Fs12652-011-0068-9#page-1)

http://michaelryoo.com/cvpr2011tutorial/ (http://michaelryoo.com/cvpr2011tutorial/)

http://blog.aicry.com/r-parallel-computing-in-5-minutes/ (http://blog.aicry.com/r-parallel-computing-in-5-minutes/)

https://en.wikipedia.org/wiki/Decision_tree_learning (https://en.wikipedia.org/wiki/Decision_tree_learning) https://en.wikipedia.org/wiki/Learning_vector_quantization (https://en.wikipedia.org/wiki/Learning_vector_quantization) https://en.wikipedia.org/wiki/Gradient_boosting (https://en.wikipedia.org/wiki/Gradient_boosting) https://en.wikipedia.org/wiki/Support_vector_machine (https://en.wikipedia.org/wiki/Support_vector_machine) https://en.wikipedia.org/wiki/Linear_discriminant_analysis (https://en.wikipedia.org/wiki/Linear_discriminant_analysis)

Code

```
# load the libraries
library(mlbench)
library(caret)
library(corrplot)
library(rpart)
library(class)
library(randomForest)
library(MASS)
library(gbm)
library(survival)
library(splines)
library(parallel)
library(plyr)
library(doParallel)
library(kernlab)

# Load and prep the datas
# change for the working directory
# setwd("/media/fred/Donnees/Donnees/Coursera/Machine Learning/devoir")

# load the dataset

PmlTrain <- read.csv("pml-training.csv", stringsAsFactors=FALSE, na.strings=c("NA","#DIV/0!",""))
PmlTest <- read.csv("pml-testing.csv", stringsAsFactors=FALSE,na.strings=c("NA","#DIV/0!",""))

# View(PmlTrain)
# summary(PmlTrain)
# head(PmlTrain)

#View(PmlTest )
#summary(PmlTest)
#head(PmlTest)

dim(PmlTrain)
dim(PmlTest)
```

```
# The outcome as factor
PmlTrain$classe <-as.factor(PmlTrain$classe)


# Tidying Data
# The Na case

nbnatraining <-sum(is.na(PmlTrain)) #very important Na
nbnatesting <- sum(is.na(PmlTest))

tna.testing<-table(colSums(is.na(PmlTest)))
tna.training<-table(colSums(is.na(PmlTrain)))

natrainmin<-as.numeric(min(names(tna.training)[-1]))
natestmin<-as.numeric(min(names(tna.testing)[-1]))


PmlTrain<-PmlTrain[,colSums(is.na(PmlTrain)) < natrainmin]
PmlTest<-PmlTest[,colSums(is.na(PmlTest)) < natestmin]

length(names(PmlTrain))
length(names(PmlTest))
length(intersect(names(PmlTrain),names(PmlTest)))
setdiff(names(PmlTrain),names(PmlTest))
setdiff(names(PmlTest),names(PmlTrain))

PmlTrain<-PmlTrain[,8:60]
dim(PmlTrain)

PmlTest<-PmlTest[,8:59]

print(dim(PmlTrain))

print(summary(PmlTrain))
```

```
freqclasse<-table(PmlTrain$classe)
barplot(freqclasse, main="Biceps Curl Correctness Classes", ylab= "Total Executions",beside=TRUE, col=heat.colors(
5))


#features selection

set.seed(1960)
# calculate correlation matrix
corMx<- cor(PmlTrain[,-53])
# summarize the correlation matrix
#print(corMx)
corrplot(corMx, method = "circle", type="lower", order="hclust", tl.cex = 0.75, tl.col="blue", tl.srt = 45, addrec
t = 3)
# find attributes that are highly corrected (ideally >0.75)
hCor <- findCorrelation(corMx, cutoff=0.75)
# print indexes of highly correlated attributes
print(hCor)

PmlTrain<-PmlTrain[,-hCor]
PmlTest <-PmlTest[,-hCor]

print(dim(PmlTrain))

corMx2<- cor(PmlTrain[,-32])
corrplot(corMx2, method = "circle", type="lower", order="hclust", tl.cex = 0.75, tl.col="blue", tl.srt = 45, addre
ct = 3)

#  Use the cores Luke !!

registerDoParallel(cores=4)


set.seed(1960)

# Partition in a training and an intermediary test set, the latter will serve to cross-validate the selected model
```

```
# before applying it to the test dataset

inTrain <- createDataPartition(PmlTrain$classe, p = 0.75,list=FALSE)
training <- PmlTrain[ inTrain,]
testing <- PmlTrain[-inTrain,]


# prepare training scheme
control <- trainControl(method="repeatedcv", number=5, repeats=3)



# train the CART model
set.seed(1960)
modelCart <- train(classe~., data=training , method="rpart", metric="Accuracy", trControl=control)

# train the LVQ model <- very long to execute on my PC
set.seed(1960)
modelLvq <- train(classe~., data=training , method="lvq",metric="Accuracy", trControl=control)

# train the LDA model
set.seed(1960)
modelLda <- train(classe~., data=training , method="lda",metric="Accuracy",trControl=control)

# train the GBM model
set.seed(1960)
modelGbm <- train(classe~., data=training , method="gbm",metric="Accuracy", trControl=control, verbose=FALSE)

# train the RF model
set.seed(1960)
modelRF<- train(classe~., data=training , method="rf",metric="Accuracy",trControl=control)

# train the SVM model
set.seed(1960)
modelSvm <- train(classe~., data=training , method="svmRadial",metric="Accuracy", trControl=control)
```

```
# collect resamples
results <- resamples(list( CART=modelCart,LVQ=modelLvq, LDA=modelLda, GBM=modelGbm, RF=modelRF, SVM=modelSvm))
# summarize the distributions

print(summary(results))

# boxplots of results
bwplot(results)
# dot plots of results
dotplot(results)


# estimate variable importance
importance <- varImp(modelRF, scale=FALSE)

# summarize importance
print(importance)

# plot importance

plot(importance)

#Cross Validation againts the  validation set

prediction <- predict(modelRF, testing)
confMX<-confusionMatrix(testing$classe, prediction)

accuracy<-postResample(prediction, testing$classe)

print(accuracy)

# Applying the model to the initial test Dataset
# to predict the classes of the 20 samples

predictedClasses <- predict(modelRF, PmlTest)
print(predictedClasses)
```