# Practical ML - Assignment

*Andrea Modini*

*23 settembre 2016*

**Problem description**

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).

## Data Sources

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

## Load Libraries

Load required libraries

```
library(caret)
library(mice)
library(reshape2)
library(VIM)
library(randomForest)
library(gbm)
```

## Load Data and Split Data

Load Data and split training set in two groups 90%-10%, the first one for training and the second for testing models

```
training=read.csv("pml-training.csv",na.strings=c("NA","#DIV/0!","","NULL"))
testing=read.csv("pml-testing.csv",na.strings=c("NA","#DIV/0!","","NULL"))

training_old=training

set.seed(1234)
training_flag = createDataPartition(training$classe, p=0.9, list=FALSE)
training = training[training_flag, ]
training_validate = training[-training_flag, ]
```

```
dim(training)
```

```
## [1] 17662   160
```

```
dim(training_validate)
```

```
## [1] 1768  160
```

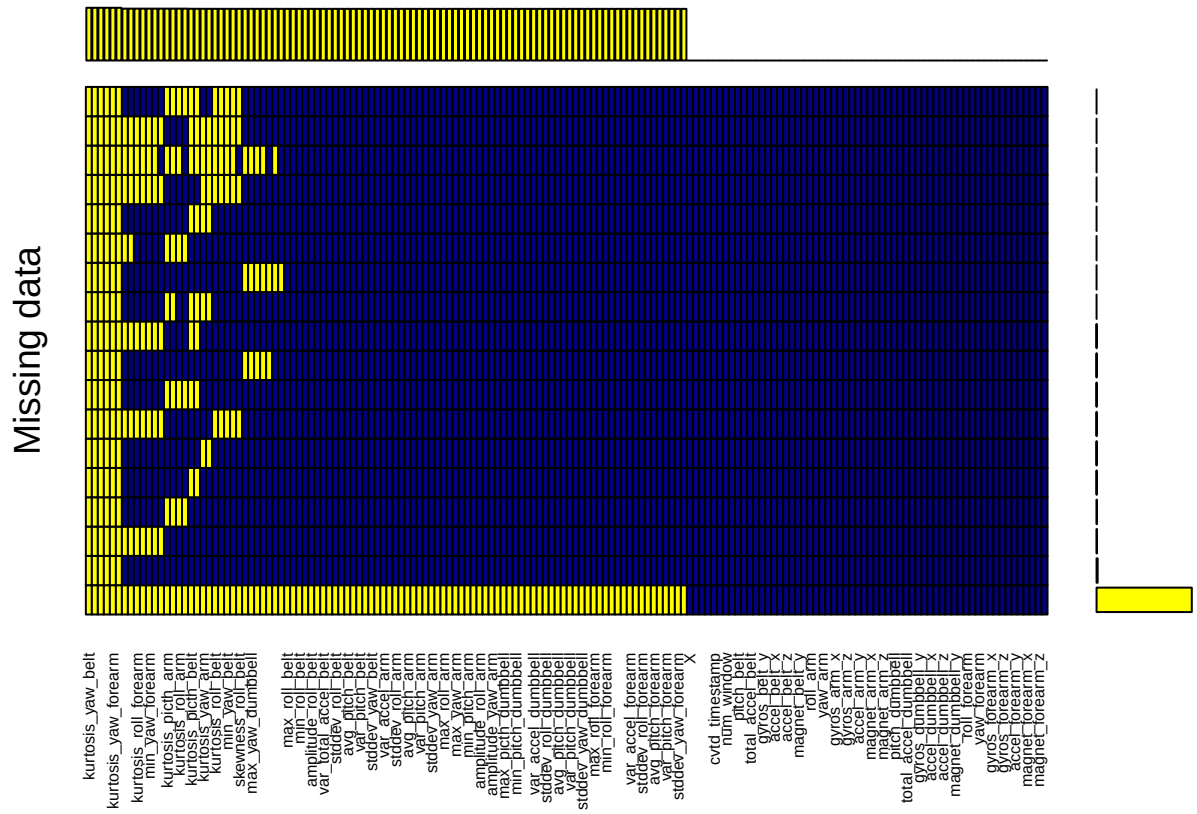
```
dim(testing)
```

```
## [1]  20 160
```

## Data Cleaning

Check Missing Data distribution

```
missTable=as.data.frame(md.pattern(training))


mice_plot=aggr(training,col=c("navyblue","yellow"),
               numbers=TRUE,
               sortVars=TRUE,
               labels=names(training),
               cex.axis=.5,
               gap=1,
               ylab=c("Missing data", "Pattern"),
               combined=TRUE)
```




```
##
##  Variables sorted by number of missings:
##                  Variable Count
##         kurtosis_yaw_belt 17662
##         skewness_yaw_belt 17662
##     kurtosis_yaw_dumbbell 17662
##     skewness_yaw_dumbbell 17662
##      kurtosis_yaw_forearm 17662
##      skewness_yaw_forearm 17662
##    kurtosis_picth_forearm 17376
##    skewness_pitch_forearm 17376
##     kurtosis_roll_forearm 17375
##           max_yaw_forearm 17375
##           min_yaw_forearm 17375
##     amplitude_yaw_forearm 17375
##     skewness_roll_forearm 17374
##        kurtosis_picth_arm 17371
##        skewness_pitch_arm 17371
##         kurtosis_roll_arm 17369
##         skewness_roll_arm 17368
##       kurtosis_picth_belt 17330
##      skewness_roll_belt.1 17330
##          kurtosis_yaw_arm 17315
##          skewness_yaw_arm 17315
##        kurtosis_roll_belt 17313
##              max_yaw_belt 17313
##              min_yaw_belt 17313
##        amplitude_yaw_belt 17313
```

```
##        skewness_roll_belt 17312
##    kurtosis_roll_dumbbell 17309
##          max_yaw_dumbbell 17309
##          min_yaw_dumbbell 17309
##    amplitude_yaw_dumbbell 17309
##    skewness_roll_dumbbell 17308
##   kurtosis_picth_dumbbell 17306
##   skewness_pitch_dumbbell 17305
##             max_roll_belt 17304
##            max_picth_belt 17304
##             min_roll_belt 17304
##            min_pitch_belt 17304
##       amplitude_roll_belt 17304
##      amplitude_pitch_belt 17304
##      var_total_accel_belt 17304
##             avg_roll_belt 17304
##          stddev_roll_belt 17304
##             var_roll_belt 17304
##            avg_pitch_belt 17304
##         stddev_pitch_belt 17304
##            var_pitch_belt 17304
##              avg_yaw_belt 17304
##           stddev_yaw_belt 17304
##              var_yaw_belt 17304
##             var_accel_arm 17304
##              avg_roll_arm 17304
##           stddev_roll_arm 17304
##              var_roll_arm 17304
##             avg_pitch_arm 17304
##          stddev_pitch_arm 17304
##             var_pitch_arm 17304
##               avg_yaw_arm 17304
##            stddev_yaw_arm 17304
##               var_yaw_arm 17304
##              max_roll_arm 17304
##             max_picth_arm 17304
##               max_yaw_arm 17304
##              min_roll_arm 17304
##             min_pitch_arm 17304
##               min_yaw_arm 17304
##       amplitude_roll_arm 17304
##      amplitude_pitch_arm 17304
##        amplitude_yaw_arm 17304
##        max_roll_dumbbell 17304
##       max_picth_dumbbell 17304
##        min_roll_dumbbell 17304
##       min_pitch_dumbbell 17304
##   amplitude_roll_dumbbell 17304
##  amplitude_pitch_dumbbell 17304
##        var_accel_dumbbell 17304
##         avg_roll_dumbbell 17304
##      stddev_roll_dumbbell 17304
##         var_roll_dumbbell 17304
##        avg_pitch_dumbbell 17304
```

```
##     stddev_pitch_dumbbell 17304
##        var_pitch_dumbbell 17304
##          avg_yaw_dumbbell 17304
##       stddev_yaw_dumbbell 17304
##          var_yaw_dumbbell 17304
##          max_roll_forearm 17304
##         max_picth_forearm 17304
##          min_roll_forearm 17304
##         min_pitch_forearm 17304
##    amplitude_roll_forearm 17304
##   amplitude_pitch_forearm 17304
##         var_accel_forearm 17304
##          avg_roll_forearm 17304
##       stddev_roll_forearm 17304
##          var_roll_forearm 17304
##         avg_pitch_forearm 17304
##      stddev_pitch_forearm 17304
##         var_pitch_forearm 17304
##           avg_yaw_forearm 17304
##        stddev_yaw_forearm 17304
##           var_yaw_forearm 17304
##                         X     0
##                 user_name     0
##      raw_timestamp_part_1     0
##      raw_timestamp_part_2     0
##            cvtd_timestamp     0
##                new_window     0
##                num_window     0
##                 roll_belt     0
##                pitch_belt     0
##                  yaw_belt     0
##          total_accel_belt     0
##              gyros_belt_x     0
##              gyros_belt_y     0
##              gyros_belt_z     0
##              accel_belt_x     0
##              accel_belt_y     0
##              accel_belt_z     0
##             magnet_belt_x     0
##             magnet_belt_y     0
##             magnet_belt_z     0
##                  roll_arm     0
##                 pitch_arm     0
##                   yaw_arm     0
##           total_accel_arm     0
##               gyros_arm_x     0
##               gyros_arm_y     0
##               gyros_arm_z     0
##               accel_arm_x     0
##               accel_arm_y     0
##               accel_arm_z     0
##              magnet_arm_x     0
##              magnet_arm_y     0
##              magnet_arm_z     0
```

```
##              roll_dumbbell     0
##             pitch_dumbbell     0
##               yaw_dumbbell     0
##      total_accel_dumbbell     0
##          gyros_dumbbell_x     0
##          gyros_dumbbell_y     0
##          gyros_dumbbell_z     0
##          accel_dumbbell_x     0
##          accel_dumbbell_y     0
##          accel_dumbbell_z     0
##         magnet_dumbbell_x     0
##         magnet_dumbbell_y     0
##         magnet_dumbbell_z     0
##              roll_forearm     0
##             pitch_forearm     0
##               yaw_forearm     0
##       total_accel_forearm     0
##           gyros_forearm_x     0
##           gyros_forearm_y     0
##           gyros_forearm_z     0
##           accel_forearm_x     0
##           accel_forearm_y     0
##           accel_forearm_z     0
##          magnet_forearm_x     0
##          magnet_forearm_y     0
##          magnet_forearm_z     0
##                    classe     0
```

There is a big group of variables with a huge number of NAs, lets remove that variables. Then remove unrelevant variables and variables with a variance close to zero.

```
#Clean 1 exclude NA >90%
training_app=training

for(i in 1:length(training)) {
        if (sum(is.na( training[, i] ))/nrow(training) >= .9 ) {
        training_app=training_app[,-i]
        }
    }

training=training_app

#Clean 2 exclude unrelevant vars
exclude.var = c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2', 'cvtd_timestamp', 'new
training = training[, -which(names(training) %in% exclude.var)]


#Check Zero var
lowVar= nearZeroVar(training[sapply(training, is.numeric)], saveMetrics = TRUE)
toremove=names(training[sapply(training, is.numeric)][,lowVar[,4]==TRUE])
training = training[,-which(names(training) %in% toremove)]
```

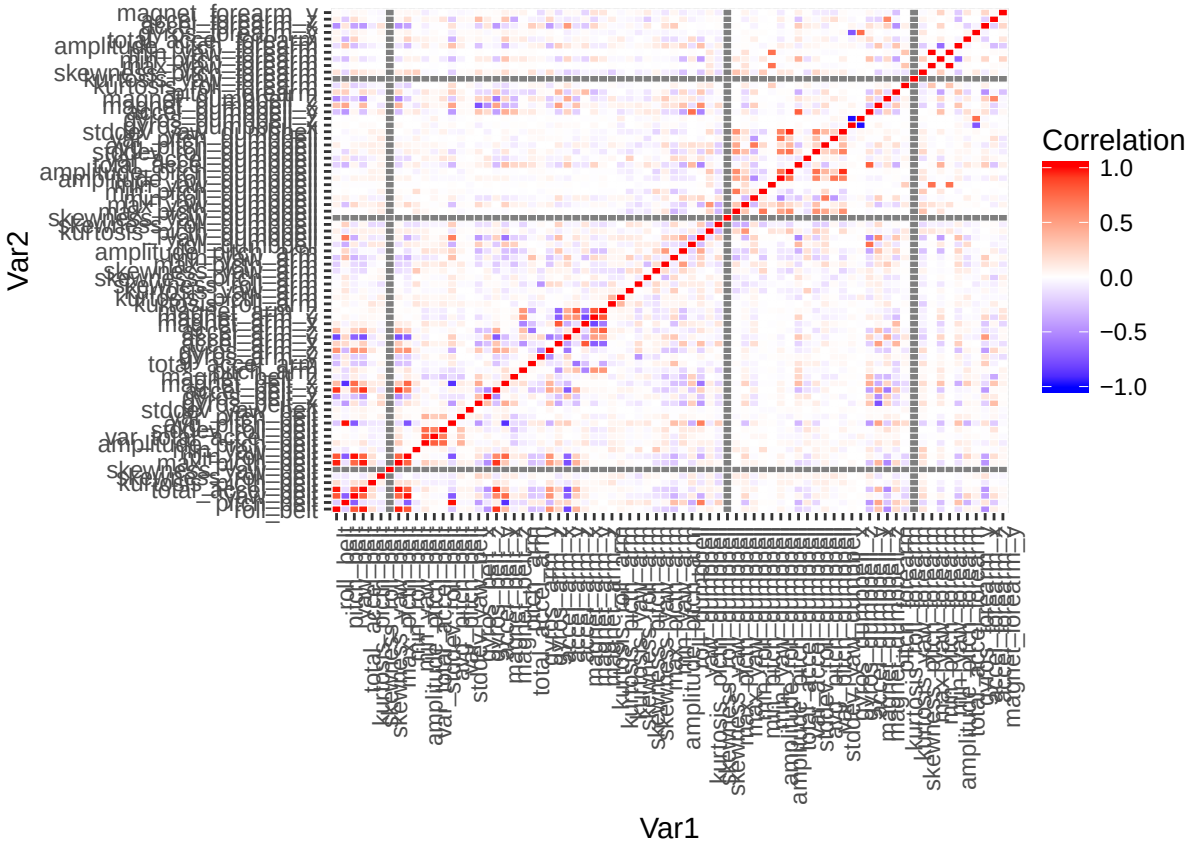
Lets fill remeaning NAs with multiple imputation.

```
#Preprocess, fill NA
set.seed(1981)
imputed=complete(mice(training))
training=imputed

for (k in 1:nrow(training)) {
  v=training[k,]
  v[which(v=="Inf")]=0
  v[which(v=="-Inf")]=0
  v[is.na(v)]=0
  v[which(v=="NaN")]=0
  training[k,]=v
}

#save.image("WorkingSpace_PracticalML_1.RData")
```

## Correlation Analysis

Check correlation between remeaning variables

```
cormat= round(cor(na.omit(training[sapply(training, is.numeric)])),2)
melted_cormat=melt(cormat)

ggplot(data=melted_cormat,aes(x=Var1,y=Var2,fill=value)) + geom_tile(color="white") + scale_fill_gradie
```



Correlation map is acceptable for machine learning algorithm. There are only small spots with high absolute correlation Lets build a GBM and a RF model, tuning parameters with 3-fold cross-validation.

```
Control = trainControl(method='repeatedcv', number = 3, repeats=1, verboseIter=TRUE)

tGrid_gbm = expand.grid(.interaction.depth = seq(2,8,1),.n.trees=seq(50,550,50),.shrinkage=c(0.01,0.05,
Gbm = train(classe ~ ., data=training, trControl=Control, tuneGrid = tGrid_gbm,  method='gbm')


#tGrid_rf <- expand.grid(.mtry =seq(4,14,1)) #mtry=14
tGrid_rf <- expand.grid(.mtry =seq(10,20,1)) #mtry=19
Rf = train(classe ~ ., data=training, trControl=Control, tuneGrid = tGrid_rf, method='rf', ntree=1000)

#save.image("WorkingSpace_PracticalML_2.RData")
```

The result are two models, GBM and RF, tuned with 3-fold cross-validation Resulting parameters are: GBM: interaction.depth=7, n.trees=550, shrinkage=0.1, n.minobsinnode=5 RF: mtry=19, ntree=1000

Now lets adjust testing set (for model evaluation on training set)

```
training_validate= training_validate[,which(names(training_validate) %in% names(training))]

for (i in 1:length(training_validate) ) {
        for(j in 1:length(training)) {
        if( length( grep(names(training[i]), names(training_validate)[j]) ) ==1)  {
            class(training_validate[j]) <- class(training[i])
        }
    }
}

for (k in 1:nrow(training_validate)) {
  v=training_validate[k,]
  v[which(v=="Inf")]=0
  v[which(v=="-Inf")]=0
  v[is.na(v)]=0
  v[which(v=="NaN")]=0
  training_validate[k,]=v
}
```

Model Performances on a portion of training set reserved for evaluation

```
Gbm.pred= predict(Gbm, newdata=training_validate)
```

```
## Loading required package: plyr
```

```
Gbm.matrix=confusionMatrix(Gbm.pred, training_validate$classe)
Gbm.matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 513   0   0   0   2
##          B   0 316   0   0   0
##          C   2   5 311   0   6
##          D   0   0   0 290   4
##          E   0   0   0   0 319
##
## Overall Statistics
##
##                Accuracy : 0.9893
##                  95% CI : (0.9833, 0.9935)
##     No Information Rate : 0.2913
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9864
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9961   0.9844   1.0000   1.0000   0.9637
## Specificity            0.9984   1.0000   0.9911   0.9973   1.0000
## Pos Pred Value         0.9961   1.0000   0.9599   0.9864   1.0000
## Neg Pred Value         0.9984   0.9966   1.0000   1.0000   0.9917
## Prevalence             0.2913   0.1816   0.1759   0.1640   0.1872
## Detection Rate         0.2902   0.1787   0.1759   0.1640   0.1804
## Detection Prevalence   0.2913   0.1787   0.1833   0.1663   0.1804
## Balanced Accuracy      0.9973   0.9922   0.9955   0.9986   0.9819
```

```
Rf.pred= predict(Rf, newdata=training_validate)
Rf.matrix= confusionMatrix(Rf.pred, training_validate$classe)
Rf.matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 515   0   0   0   0
##          B   0 321   0   0   0
##          C   0   0 311   0   0
##          D   0   0   0 290   0
##          E   0   0   0   0 331
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9979, 1)
##     No Information Rate : 0.2913
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000    1.000   1.0000
## Specificity            1.0000   1.0000   1.0000    1.000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000    1.000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000    1.000   1.0000
## Prevalence             0.2913   0.1816   0.1759    0.164   0.1872
## Detection Rate         0.2913   0.1816   0.1759    0.164   0.1872
## Detection Prevalence   0.2913   0.1816   0.1759    0.164   0.1872
## Balanced Accuracy      1.0000   1.0000   1.0000    1.000   1.0000
```

Performances are very good. GBM Accuracy : 98% RF Accuracy : 100% Lets see most important variables for the best model

```
best_model= Rf
top <- varImp(best_model)
top$importance$Overall <- sort(top$importance$Overall, decreasing=TRUE)
best_vars= data.frame(Feature=row.names(top$importance),Importance=top$importance$Overall)
best_vars
```

```
##                    Feature   Importance
## 1                roll_belt 100.00000000
## 2               pitch_belt  67.73241896
## 3                 yaw_belt  65.36592819
## 4          total_accel_belt  59.08288354
## 5        kurtosis_picth_belt  49.74205500
## 6         skewness_roll_belt  45.88462773
## 7         skewness_yaw_belt  34.75403898
## 8            max_picth_belt  31.08688270
## 9             min_roll_belt  30.61335000
## 10            min_yaw_belt  22.85609826
## 11     amplitude_pitch_belt  21.16831561
## 12     var_total_accel_belt  20.22076592
## 13         stddev_roll_belt  20.17697639
## 14           avg_pitch_belt  19.02375464
## 15           var_pitch_belt  18.01846642
## 16         stddev_yaw_belt  17.74640304
## 17            gyros_belt_x  17.03087735
## 18            gyros_belt_z  15.55809402
## 19            accel_belt_y  14.35078031
## 20           magnet_belt_x  14.28371978
## 21           magnet_belt_z  14.12667865
## 22               pitch_arm  13.02649243
## 23          total_accel_arm  10.94771781
## 24              gyros_arm_y  10.24331883
## 25              gyros_arm_z   9.71810971
## 26              accel_arm_x   9.42389031
## 27              accel_arm_y   9.41846365
## 28              accel_arm_z   8.40466946
## 29             magnet_arm_x   6.92212433
## 30             magnet_arm_y   6.69563088
## 31             magnet_arm_z   6.65550346
## 32        kurtosis_roll_arm   6.39876797
## 33       kurtosis_picth_arm   6.28554452
```

```
## 34         kurtosis_yaw_arm   6.10238021
## 35        skewness_roll_arm   3.73078464
## 36       skewness_pitch_arm   3.53878677
## 37         skewness_yaw_arm   3.51264149
## 38            max_picth_arm   3.17561723
## 39              min_yaw_arm   2.55431664
## 40      amplitude_pitch_arm   2.44754422
## 41             roll_dumbbell   2.26794002
## 42              yaw_dumbbell   2.10161873
## 43  kurtosis_picth_dumbbell   2.02458830
## 44   skewness_roll_dumbbell   1.88735537
## 45    skewness_yaw_dumbbell   1.57826945
## 46       max_picth_dumbbell   1.46733312
## 47         max_yaw_dumbbell   1.46183775
## 48        min_roll_dumbbell   1.44852131
## 49       min_pitch_dumbbell   1.32788110
## 50         min_yaw_dumbbell   1.31455318
## 51  amplitude_roll_dumbbell   1.27367594
## 52 amplitude_pitch_dumbbell   1.26006320
## 53     total_accel_dumbbell   1.23412715
## 54       var_accel_dumbbell   1.19333923
## 55     stddev_roll_dumbbell   1.16492840
## 56       avg_pitch_dumbbell   1.16448732
## 57       var_pitch_dumbbell   1.14467457
## 58      stddev_yaw_dumbbell   1.13700479
## 59         gyros_dumbbell_x   1.04570785
## 60         gyros_dumbbell_z   1.00092239
## 61         accel_dumbbell_y   0.95529842
## 62        magnet_dumbbell_x   0.93175557
## 63        magnet_dumbbell_z   0.88829178
## 64             pitch_forearm   0.85021176
## 65    kurtosis_roll_forearm   0.83024512
## 66     kurtosis_yaw_forearm   0.82860444
## 67   skewness_pitch_forearm   0.81279096
## 68          max_yaw_forearm   0.80609107
## 69        min_pitch_forearm   0.79990544
## 70          min_yaw_forearm   0.79787634
## 71  amplitude_pitch_forearm   0.13077336
## 72      total_accel_forearm   0.06736513
## 73          gyros_forearm_y   0.06469307
## 74          accel_forearm_x   0.00000000
## 75          accel_forearm_z   0.00000000
## 76         magnet_forearm_y   0.00000000
```

Now lets adjust testing set for blind prediction

```
testing= testing[,which(names(testing) %in% names(training))]

for (i in 1:length(testing) ) {
        for(j in 1:length(training)) {
        if( length( grep(names(training[i]), names(testing)[j]) ) ==1)  {
            class(testing[j]) <- class(training[i])
        }
    }
```

```
}

for (k in 1:nrow(testing)) {
  v=testing[k,]
  v[which(v=="Inf")]=0
  v[which(v=="-Inf")]=0
  v[is.na(v)]=0
  v[which(v=="NaN")]=0
  testing[k,]=v
}

Rf.pred.test= predict(Rf, newdata=testing)
Rf.pred.test
```

```
##  [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
```