Machine Learning Project

Santiago Armida

24 de octubre de 2015

Introduction

We have information from activity levels obtained through the use of devices such as Jawbone Up, Nike FuelBand, and Fitbit. These are used by a group of people who take measurments of themselves, regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. . The data will be used to analyze if the exercise was done correctly. This is set in the classe variable. This is all in the Weigth lifting Exercise Dataset, the reference is given at the end. #Description "Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five 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) and throwing the hips to the front (Class E)." #Data Although there is a lot of different measures, many of them are not informed at all, and some others are informed partially.

```
#   install.packages("caret")
#   install.packages("e1071")
#   install.packages("randomForest")
#   install.packages("inTrees")
#   install.packages("klaR")   model lda
#   install.packages("MASS")   model nb
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3

## Loading required package: lattice
## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.1.3
```

```
## read data  pml-training.csv in same directory
 setwd("~/Documents/Data Science 8 ML")
 pmldata <- read.csv("pml-training.csv",na.strings=c("","NA","#DIV/0!"))
# environment take advantage of processor
set.seed(12345)
library(inTrees)
library(doMC)
```

```
## Loading required package: foreach
## Loading required package: iterators
## Loading required package: parallel
```

```
registerDoMC(cores=3)
```

Cleaning the data

Although there is a lot of different measures, many of them are not informed at all, and some others are informed partially. I will set a seed to get consiten runs. In order to build a proper model that can pedict correctly, we need to clean the data, and just use what is well informed in order to do it in a more correct way. For that reason the columns with nulls will be eliminated. The information concerning the number of row, name of subject, and the timestamp data is informative, so to reduce the dataset they will be taken out. Columns 1 to 7. With this done, then we will try to detect the correlation of the variables. If it is too high, we need to eliminate the variables, as we will not get too much effect if we have many correlated variables, so if correlation is higher than 85% they will be eliminated.

```
    countna <- apply(pmldata, 2, function(x) sum(is.na(x)))
    colsOK <- names(which(countna/dim(pmldata[1])< 0.1, arr.ind=T))
    trainclean <- pmldata[,colsOK]
    # Data not necessary for analysis
    trainclean <- trainclean[,-(1:7)]
    # data correlated
    train1 <- cor(subset(trainclean, select = -classe))
    corrvars  <- findCorrelation(train1, cutoff=0.85)
    trainclean <- trainclean[, -corrvars]
    dim(trainclean)
```

```
## [1] 19622    45
```

Now that the data is good, 45 columns, we will divide our data, as we want a training part and then part of it as test data to validate our model, so we will use the other part of the data. There is additional test data, but that is a file that will use to get the values being searched in the second part of the project.

```
intrain  <- createDataPartition(y=trainclean$classe,p=0.7,list=FALSE)
train <- trainclean[intrain,]
test <- trainclean[-intrain,]
    dim(train);dim(test)
```

```
## [1] 13737    45
```

```
## [1] 5885   45
```

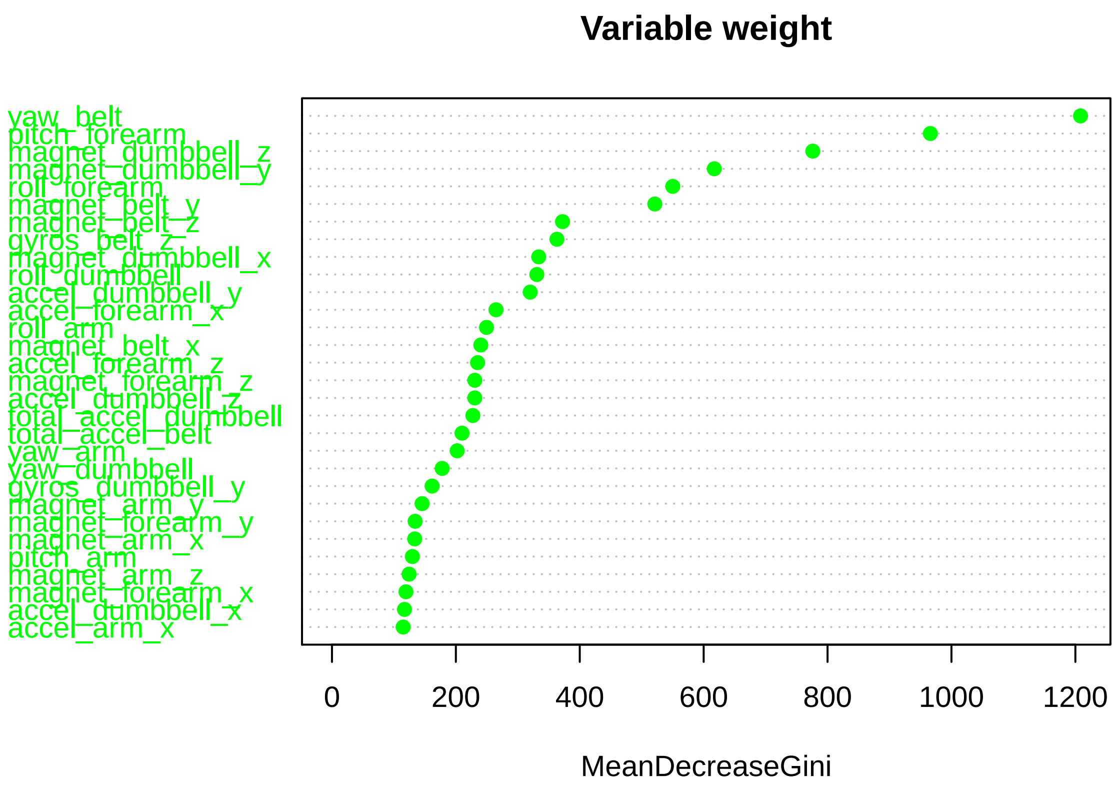
With this done, and looking at the data, we see it is a combination of info, therefore we will use one of the models that can process this combination correctly. For this we will use the Random Forest algorithm.

Run the model and take the most importan vriables.

```
modelRF <- train(classe ~ ., data=train, method="rf")
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
varImpPlot(modelRF$finalModel, main="Variable weight",
           col="green", pch =19, cex=0.9)
```



```r
TOPV <- varImp(modelRF)[[1]]
weight <- order(-TOPV$Overall)
TOPNAMES <- row.names(TOPV)[weight][1:24]
traincut <- cbind(train[,TOPNAMES], train$classe)
names(traincut)[25] <- "classe"
modelRF
```

```
## Random Forest
##
## 13737 samples
##    44 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9859826  0.9822588  0.002356378  0.002983808
##   23    0.9870757  0.9836440  0.002075595  0.002623628
##   44    0.9787966  0.9731632  0.003589676  0.004552844
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 23.
```

When this is made, it takes some time, because of the iterations. The acurracy is very high in the order of

0.98. We get the coefficients, but not all of them are as important, so with the weight and result from our model we decide to take the top 24. With these we will try the prediction again with this other models. So now with our 24 variables and predictor, we will run two more models.

```
modlda <- train(classe ~ ., data=traincut, method="lda")
```

```
## Loading required package: MASS
```

```
#modnb <- train(classe ~ ., data=traincut, method="nb")
modrf <- train(classe ~ ., data=traincut, method="rf")
predlda  = predict(modlda,test); predrf <- predict(modrf,test)
table(predlda,predrf)
```

```
##        predrf
## predlda    A    B    C    D    E
##       A 1257  256  149   76   79
##       B  130  505   67   95  204
##       C  169  178  638  125  157
##       D  113  101  164  574  140
##       E   13   90   30   75  500
```

There are some warnings with the naive bayes, so we will not include this. The table shows that the predictions are not very precise. So as we have better results with the Random Forest, we will use that to predict our final results. The tree is very big, so we will just show the first lines of k1 and k2

```
modrf
```

```
## Random Forest
##
## 13737 samples
##    24 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9845883  0.9804918  0.001737040  0.002190005
##   13    0.9840463  0.9798051  0.002020572  0.002560505
##   24    0.9762268  0.9699054  0.004249996  0.005392856
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
head(getTree(modrf$finalModel, k=1, labelVar=T))
```

```
##   left daughter right daughter         split var split point status
## 1             2              3     pitch_forearm     -33.95      1
## 2             4              5 magnet_dumbbell_z     -74.50      1
## 3             6              7  total_accel_belt      21.50      1
## 4             8              9     magnet_belt_z    -301.50      1
## 5            10             11   accel_dumbbell_y     158.50      1
## 6            12             13 magnet_dumbbell_y     426.50      1
##   prediction
## 1       <NA>
## 2       <NA>
## 3       <NA>
## 4       <NA>
## 5       <NA>
## 6       <NA>
```

```
head(getTree(modrf$finalModel, k=2, labelVar=T))
```

```
##   left daughter right daughter            split var split point status
## 1             2              3    magnet_dumbbell_y    426.5000      1
## 2             4              5        roll_dumbbell    -95.2096      1
## 3             6              7     total_accel_belt     18.5000      1
## 4             8              9 total_accel_dumbbell      8.5000      1
## 5            10             11 total_accel_dumbbell     29.5000      1
## 6            12             13              yaw_arm     32.4500      1
##   prediction
## 1       <NA>
## 2       <NA>
## 3       <NA>
## 4       <NA>
## 5       <NA>
## 6       <NA>
```

```
#library(inTrees)
#treeList <- RF2List(modrf$finalModel)
#extract <- extractRules(treeList, traincut)
#ruleM <- getRuleMetric(extract, traincut, traincut$classe)
#ruleM <- pruneRule(ruleM, traincut, traincut$classe)
#ruleM <- selectRuleRRF(ruleM, traincut, traincut$classe)
#rules <- presentRules(ruleM,colnames(traincut))
#head(rules)
```

We will use these coeficienta (FINAL MODEL) to predict the results from the given TEST dataset to fill the second part of the project.

Bibliography.

Information taken from : 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.