Practical Machine Learning Project

Matías

21 de octubre de 2015

Loading the Data

```
library(caret)
```

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

```
set.seed(4433)
traindata <- read.csv("pml-training.csv", header = T, sep = ",")
```

Cleaning Data

Removing variables with near zero varibilty

```
dim(traindata)
```

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

```
nzv <- nearZeroVar(traindata)
filtereddata <- traindata[, -nzv]

dim(filtereddata)
```

```
## [1] 19622   100
```

Removing variables with more than 14000 NA's

```
c<-c()
for (i in  1:100) {
       if(sum(is.na(filtereddata[, i]))>14000)
          c <- c(c,i)
       }
new_data <- filtereddata[,-c]
```

Removing id, name and time stamp; this variables are useless

```
new_data <- new_data[, -c(1,2)]
new_data <- new_data[,-3]
```

Creating train and test partitions.

I create a training and a testing data set from the pml-training file.

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

There are some highly correlated variables, but since I'm going to use cart and random forest I choose to leave them.

```
cor_mat <- cor(subset(new_data, select = -classe))
highlycorrelated <- findCorrelation(cor_mat, cutoff = 0.75)
names(new_data)[highlycorrelated]
```

```
##  [1] "accel_belt_z"      "roll_belt"         "accel_belt_y"
##  [4] "accel_arm_y"       "total_accel_belt"  "accel_dumbbell_z"
##  [7] "accel_belt_x"      "magnet_belt_x"     "magnet_dumbbell_x"
## [10] "accel_dumbbell_y"  "magnet_dumbbell_y" "magnet_dumbbell_z"
## [13] "accel_arm_x"       "accel_dumbbell_x"  "accel_arm_z"
## [16] "magnet_arm_y"      "magnet_belt_y"     "accel_forearm_y"
## [19] "gyros_arm_y"       "gyros_forearm_z"   "gyros_dumbbell_x"
```

```
highlycorrelated
```

```
##  [1] 13  4 12 25  7 39 11 14 40 38 41 42 24 37 26 28 15 51 22 49 34
```

Training the model.

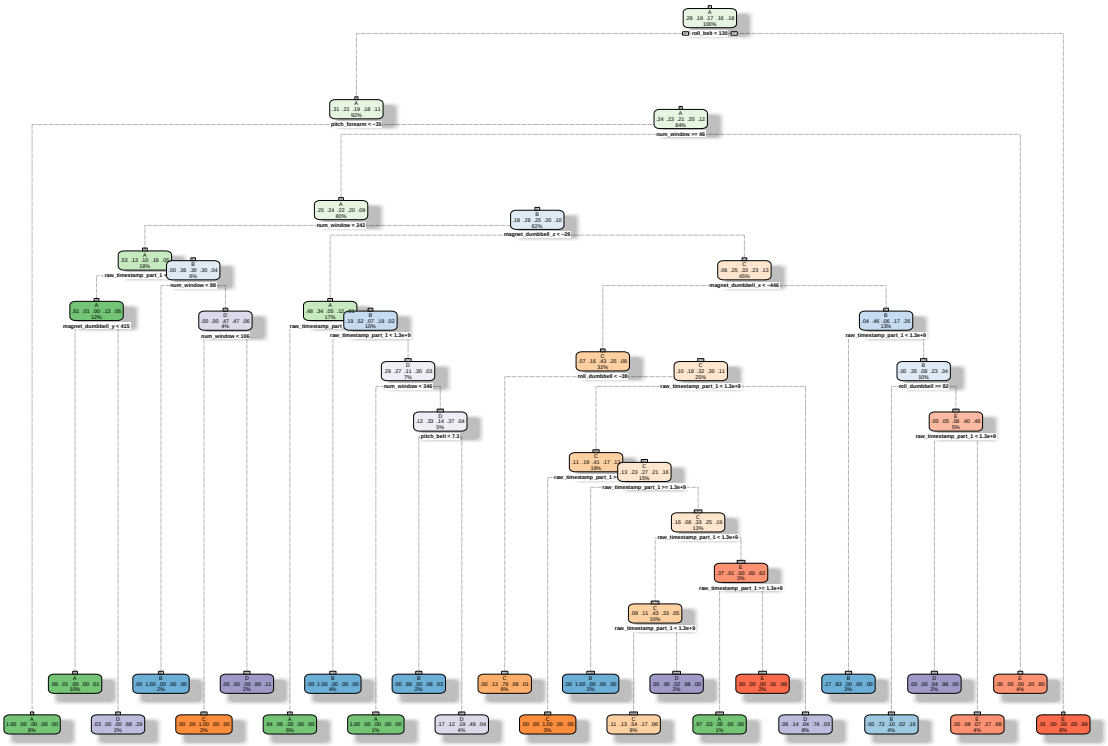
```
library(rpart)
modelFit1 <- rpart(classe ~ ., data=training, method="class")
```

```
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Versión 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

```
fancyRpartPlot(modelFit1)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



#Evaluating the model.

```
predictions1 <- predict(modelFit1, testing, type = "class")
confusionMatrix(predictions1, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1267   24    1    0    2
##          B   30  740   18   15   32
##          C   40   96  772   86   23
##          D   55   64   45  631   64
##          E    3   25   19   72  780
##
## Overall Statistics
##
##                Accuracy : 0.8544
##                  95% CI : (0.8442, 0.8642)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.8165
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9082   0.7798   0.9029   0.7848   0.8657
## Specificity            0.9923   0.9760   0.9395   0.9444   0.9703
## Pos Pred Value         0.9791   0.8862   0.7591   0.7346   0.8676
## Neg Pred Value         0.9645   0.9486   0.9786   0.9572   0.9698
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2584   0.1509   0.1574   0.1287   0.1591
## Detection Prevalence   0.2639   0.1703   0.2074   0.1752   0.1833
## Balanced Accuracy      0.9503   0.8779   0.9212   0.8646   0.9180
```

Doing the confusion Matrix we get the accuracy of the model, more than 85%, pretty high, but we will try with the random forest algorithm to see if we get better results

Training the new model.

```
library(randomForest)
```

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

```
modelFit2 <- randomForest(classe ~. , data=training)
print(modelFit2)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.14%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4184    1    0    0    0 0.0002389486
## B    3 2845    0    0    0 0.0010533708
## C    0    5 2561    1    0 0.0023373588
## D    0    0    7 2404    1 0.0033167496
## E    0    0    0    2 2704 0.0007390983
```

Evaluating the new model.

```r
predictions2 <- predict(modelFit2, testing, type = "class")
confusionMatrix(predictions2, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    0    0    0
##          C    0    0  855    2    0
##          D    0    0    0  802    3
##          E    0    0    0    0  898
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.9976, 0.9997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##  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   0.9975   0.9967
## Specificity            1.0000   1.0000   0.9995   0.9993   1.0000
## Pos Pred Value         1.0000   1.0000   0.9977   0.9963   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   0.9995   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1743   0.1635   0.1831
## Detection Prevalence   0.2845   0.1935   0.1748   0.1642   0.1831
## Balanced Accuracy      1.0000   1.0000   0.9998   0.9984   0.9983
```

We can see that applying random forest we get better accuracy and less missclassifications.