Prediction Assignment Writeup

Mooc Coursera

27 Feb, 2016

Executive Summary

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.

Data

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Preprocessing Step

```
library(caret)
```

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

```
library(Hmisc)
```

```
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
```

```
##
## Loading required package: Formula
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

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

Variables with no missing values will be good candidate for Predictor

```
isAnyMissing <- sapply(testingWeb, function (x) any(is.na(x) | x == ""))
isPredictor <- !isAnyMissing & grepl("belt|[^(fore)]arm|dumbbell|forearm", names(isAnyMissing))
predictorCandidates <- names(isAnyMissing)[isPredictor]
predictorCandidates
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"
```

Subsetting the primary dataset to include only the predictor candidates and the outcome variable,

```
varincluded <- c("classe", predictorCandidates)
DataTraining <- training[ ,varincluded]
dim(DataTraining)
```

```
## [1] 19622    53
```

```r
names(DataTraining)
```

```
##  [1] "classe"               "roll_belt"            "pitch_belt"
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [37] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [43] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [46] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [49] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [52] "magnet_forearm_y"     "magnet_forearm_z"
```

Create the Data Partition of 60 percent training set and 30 percent testing set

```r
inTrain <- createDataPartition(y = DataTraining$classe,p = 0.6,list = FALSE)
training <- DataTraining[inTrain,]
testing <- DataTraining[-inTrain,]
dim(training)
```

```
## [1] 11776    53
```

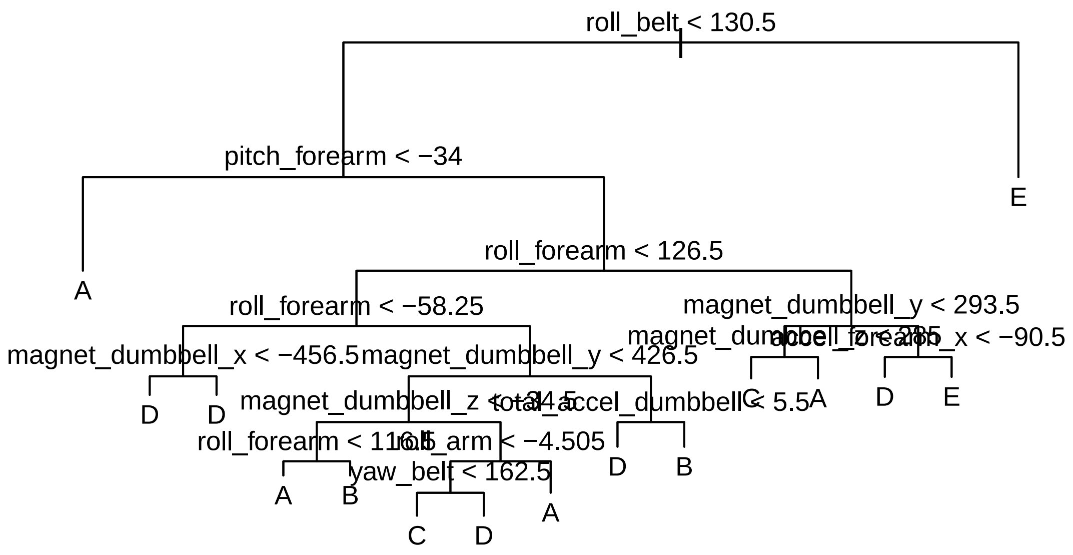
```r
dim(testing)
```

```
## [1] 7846   53
```

```r
library(tree)
set.seed(333)
tree.training=tree(classe~.,data=training)
summary(tree.training)
```

```
##
## Classification tree:
## tree(formula = classe ~ ., data = training)
## Variables actually used in tree construction:
##  [1] "roll_belt"            "pitch_forearm"        "roll_forearm"
##  [4] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
##  [7] "roll_arm"             "yaw_belt"             "total_accel_dumbbell"
## [10] "accel_forearm_x"
## Number of terminal nodes:  15
## Residual mean deviance:  1.816 = 21360 / 11760
## Misclassification error rate: 0.3699 = 4356 / 11776
```

```r
plot(tree.training)
text(tree.training,pretty=0, cex =.8)
```



# Cross Validation and predicting new Values We are going to predict the new values and check the performance using cross Validation

```r
tree.pred <- predict(tree.training,testing,type = "class")
predMatrix <- with(testing,table(tree.pred,classe))
sum(diag(predMatrix))/sum(as.vector(predMatrix))
```

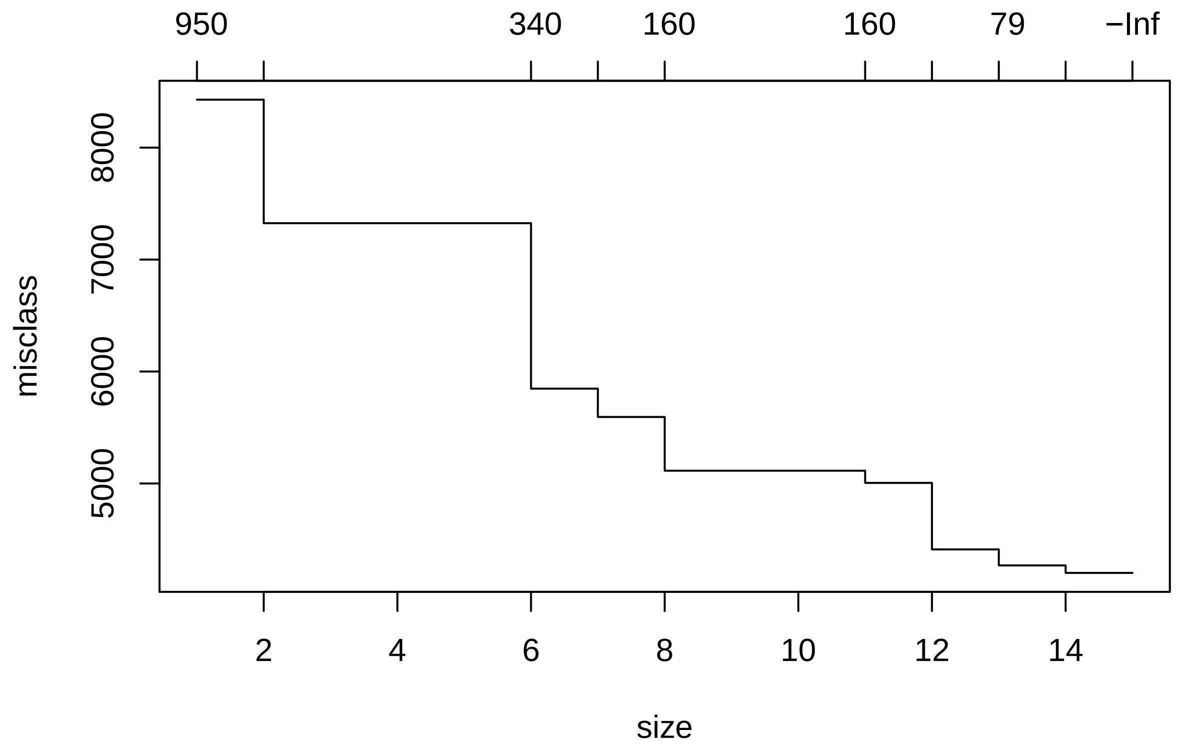
```
## [1] 0.6215906
```

Implementing the Prunning technique

```r
cv.training=cv.tree(tree.training,FUN=prune.misclass)
cv.training
```

```
## $size
##  [1] 15 14 13 12 11  8  7  6  2  1
##
## $dev
##  [1] 4201 4201 4268 4411 5005 5114 5594 5847 7325 8428
##
## $k
##  [1]     -Inf   0.0000  79.0000 120.0000 159.0000 164.3333 266.0000
##  [8] 335.0000 418.2500 947.0000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cv.training)
```



This shows that when the size of the tree goes down,the deviance goes up.It means that 16 is a good size Suppose we prune it at the size of nodes equal 14

```
prune.training=prune.misclass(tree.training,best=14)
```

Now let's evaluate this pruned tree on the test data

```
tree.pred=predict(prune.training,testing,type="class")
predMatrix = with(testing,table(tree.pred,classe))
sum(diag(predMatrix))/sum(as.vector(predMatrix)) # error rate
```

```
## [1] 0.6215906
```

Thus result with test data only differs by small amount which implies pruning did not hurt with respect to misclassification errors.we used less predictors to get almost similar results.By prunning we got a shallower tree.

Now single tree is not quite efficient ,so we are going to use Bagging to improve the efficiency.we are going to use the popular Random Forest Algorithm.

Prediction using Random Forest

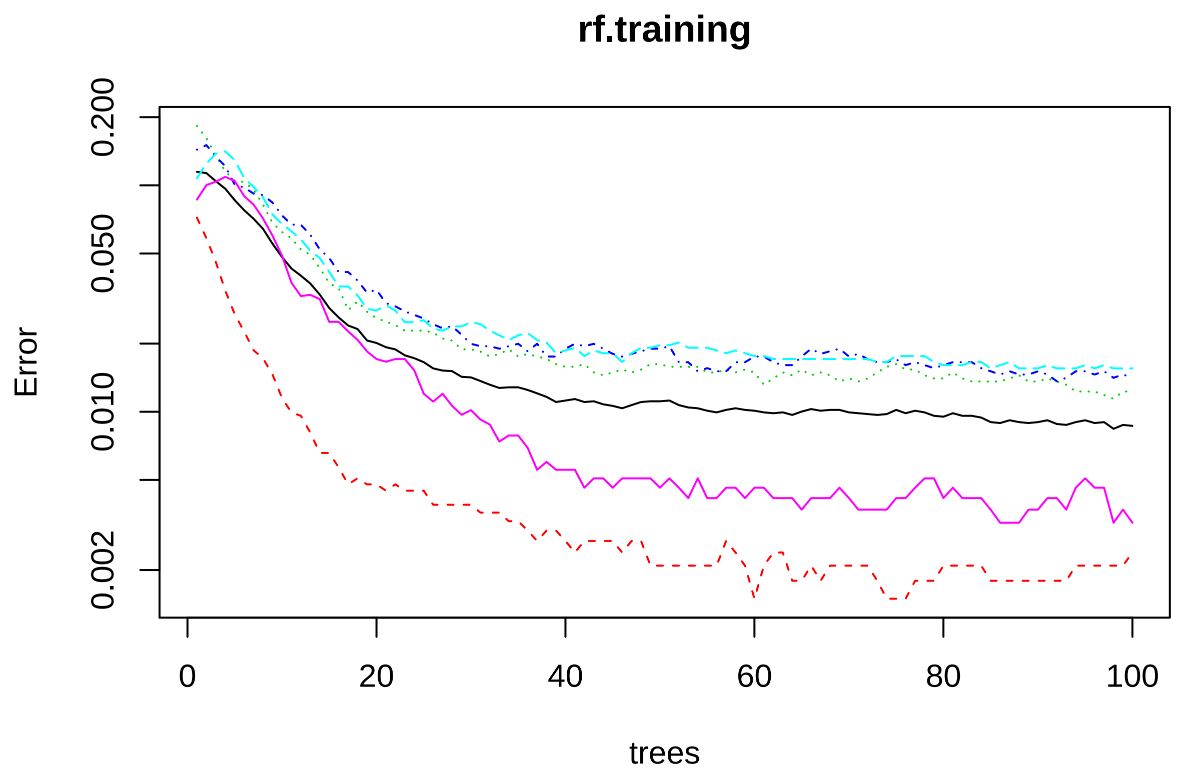
```r
require(randomForest)
```

```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:Hmisc':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
rf.training=randomForest(classe~.,data=training,ntree=100, importance=TRUE)
rf.training
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, ntree = 100,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.87%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3340    6    1    0    1 0.002389486
## B   19 2251    8    0    1 0.012286090
## C    0   20 2025    9    0 0.014118793
## D    0    1   27 1900    2 0.015544041
## E    0    0    3    4 2158 0.003233256
```

```r
plot(rf.training,log="y")
```



Our random forest algorithm shows an OOB estimate of error rate: 0.82% which is quite good

Out of Sample errors:

Now let's evaluate the tree on testing Data

```
tree.pred=predict(rf.training,testing,type="class")
predMatrix = with(testing,table(tree.pred,classe))
sum(diag(predMatrix))/sum(as.vector(predMatrix)) # error rate
```

```
## [1] 0.9942646
```

The result 0.99 tells us that we are getting very good estimate.Now let's print the prediction answers for the Last quiz # Inference We can predict the test data from the web:

```
quizAnswers <- predict(rf.training, testingWeb)
quizAnswers
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```