Prediction Assignment Writeup - Practical Machine Learning

nhatmn

Sunday, May 24, 2015

# 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>.

# Load Load neccessary library

library("caret")  
library("tree")  
library("rattle")  
library("randomForest")  
library("rpart")  
library("rpart.plot")

# Read data from "pml-training.csv" and "pml-testing.csv".

# Load data  
trainingOrg = read.csv("pml-training.csv", na.strings=c("", "NA", "NULL"))  
testingOrg = read.csv("pml-testing.csv", na.strings=c("", "NA", "NULL"))  
  
# Now see dimension of "pml-training.csv" and "pml-testing.csv".  
dim(trainingOrg)

## [1] 19622 160

dim(testingOrg)

## [1] 20 160

# Remove variables that have too many NA values

training.dena <- trainingOrg[ , colSums(is.na(trainingOrg)) == 0]  
# Now see dimension of training.dena  
dim(training.dena)

## [1] 19622 60

# Remove unrelevant variables.  
remove = c('X', 'user\_name', 'raw\_timestamp\_part\_1', 'raw\_timestamp\_part\_2', 'cvtd\_timestamp', 'new\_window', 'num\_window')  
training.dere <- training.dena[, -which(names(training.dena) %in% remove)]  
dim(training.dere)

## [1] 19622 53

# Check the variables that have extremely low variance  
zeroVar= nearZeroVar(training.dere[sapply(training.dere, is.numeric)], saveMetrics = TRUE)  
training.nonzerovar = training.dere[, zeroVar[, 'nzv']==0]  
dim(training.nonzerovar)

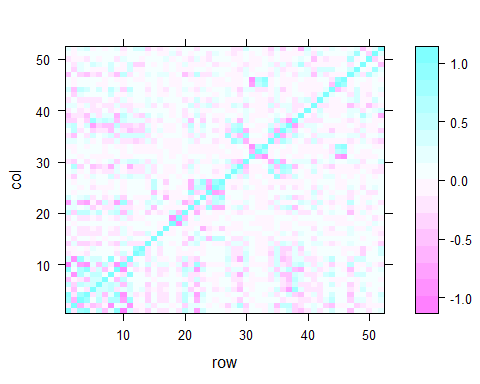
## [1] 19622 53

# Remove highly correlated variables 90%  
corrMatrix <- cor(na.omit(training.nonzerovar[sapply(training.nonzerovar, is.numeric)]))  
dim(corrMatrix)

## [1] 52 52

corrDF <- expand.grid(row = 1:52, col = 1:52)  
corrDF$correlation <- as.vector(corrMatrix)

levelplot(correlation ~ row+ col, corrDF)



# Remove high correlation variables.  
removecor = findCorrelation(corrMatrix, cutoff = .90, verbose = TRUE)

training.decor = training.nonzerovar[, -removecor]  
dim(training.decor)

## [1] 19622 46

# Split data to training and testing for cross validation.

inTrain <- createDataPartition(y=training.decor$classe, p=0.7, list=FALSE)  
training <- training.decor[inTrain,]; testing <- training.decor[-inTrain,]  
dim(training)

## [1] 13737 46

dim(testing)

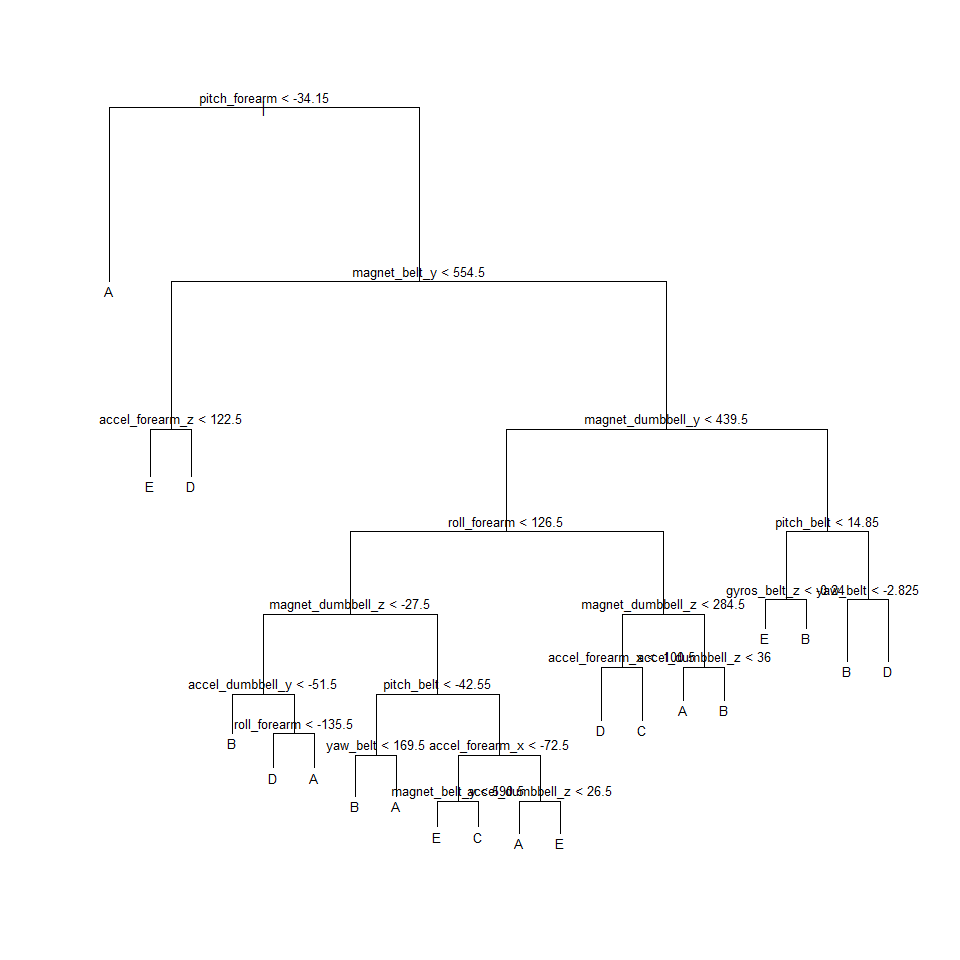
## [1] 5885 46

# Fit a tree to these data, and summarize and plot it

set.seed(2125) #Birthday date of me and my girlfriend :)  
tree.training=tree(classe~., data=training)  
summary(tree.training)

##   
## Classification tree:  
## tree(formula = classe ~ ., data = training)  
## Variables actually used in tree construction:  
## [1] "pitch\_forearm" "magnet\_belt\_y" "accel\_forearm\_z"   
## [4] "magnet\_dumbbell\_y" "roll\_forearm" "magnet\_dumbbell\_z"  
## [7] "accel\_dumbbell\_y" "pitch\_belt" "yaw\_belt"   
## [10] "accel\_forearm\_x" "accel\_dumbbell\_z" "gyros\_belt\_z"   
## Number of terminal nodes: 20   
## Residual mean deviance: 1.677 = 23000 / 13720   
## Misclassification error rate: 0.3319 = 4559 / 13737

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



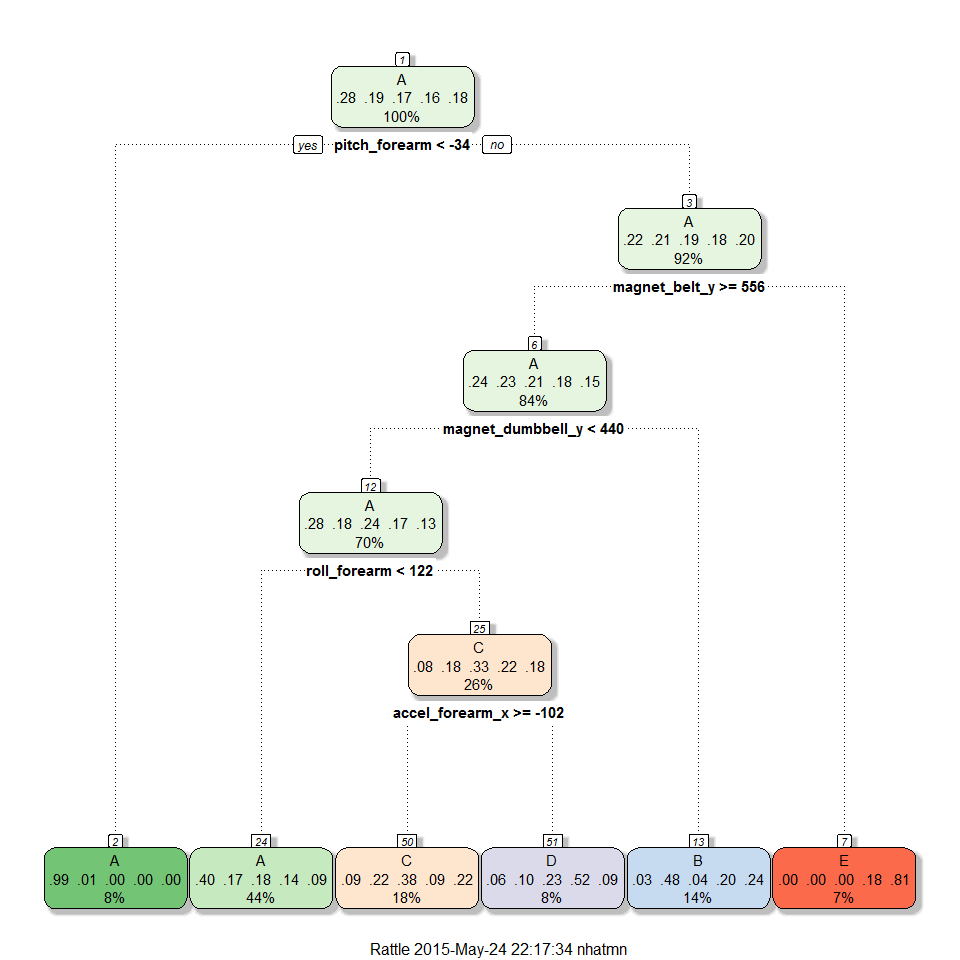
# Running rpart for the form Caret

modFit <- train(classe ~ ., method="rpart", data=training)  
print(modFit$finalModel)

## n= 13737   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)   
## 2) pitch\_forearm< -33.95 1114 8 A (0.99 0.0072 0 0 0) \*  
## 3) pitch\_forearm>=-33.95 12623 9823 A (0.22 0.21 0.19 0.18 0.2)   
## 6) magnet\_belt\_y>=555.5 11604 8807 A (0.24 0.23 0.21 0.18 0.15)   
## 12) magnet\_dumbbell\_y< 439.5 9642 6906 A (0.28 0.18 0.24 0.17 0.13)   
## 24) roll\_forearm< 121.5 6051 3605 A (0.4 0.17 0.18 0.14 0.095) \*  
## 25) roll\_forearm>=121.5 3591 2398 C (0.081 0.18 0.33 0.22 0.18)   
## 50) accel\_forearm\_x>=-101.5 2494 1557 C (0.091 0.22 0.38 0.09 0.22) \*  
## 51) accel\_forearm\_x< -101.5 1097 525 D (0.057 0.099 0.23 0.52 0.088) \*  
## 13) magnet\_dumbbell\_y>=439.5 1962 1024 B (0.031 0.48 0.042 0.2 0.24) \*  
## 7) magnet\_belt\_y< 555.5 1019 193 E (0.0029 0.002 0.00098 0.18 0.81) \*

# Prettier plots

fancyRpartPlot(modFit$finalModel)

 The result from 'caret' 'rpart' is close to 'tree'.

# Check the performance of the tree on the testing data by cross validation.

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

## [1] 0.6530161

tree.pred=predict(modFit, testing)  
predMatrix = with(testing,table(tree.pred, classe))  
sum(diag(predMatrix))/sum(as.vector(predMatrix))

## [1] 0.4951572

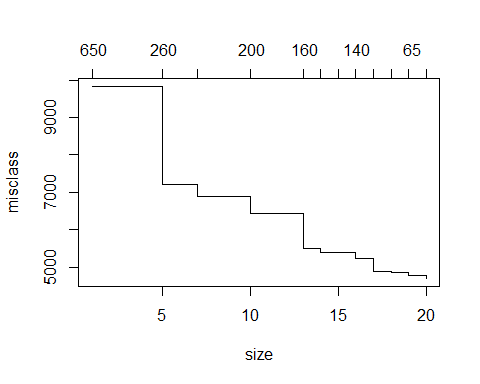
The result from 'caret' is much lower than the result from 'tree'.

# Use Cross Validation to prune the tree

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

## $size  
## [1] 20 19 18 17 16 15 14 13 10 7 5 1  
##   
## $dev  
## [1] 4696 4769 4849 4886 5226 5399 5399 5508 6423 6889 7220 9831  
##   
## $k  
## [1] -Inf 65.0000 80.0000 114.0000 140.0000 156.0000 157.0000  
## [8] 164.0000 200.6667 226.0000 260.5000 648.7500  
##   
## $method  
## [1] "misclass"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

plot(cv.training)

 Suppose that the size of nodes is 19

# Suppose that the size of nodes is 19  
prune.training=prune.misclass(tree.training, best=19)  
# 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))

## [1] 0.6499575

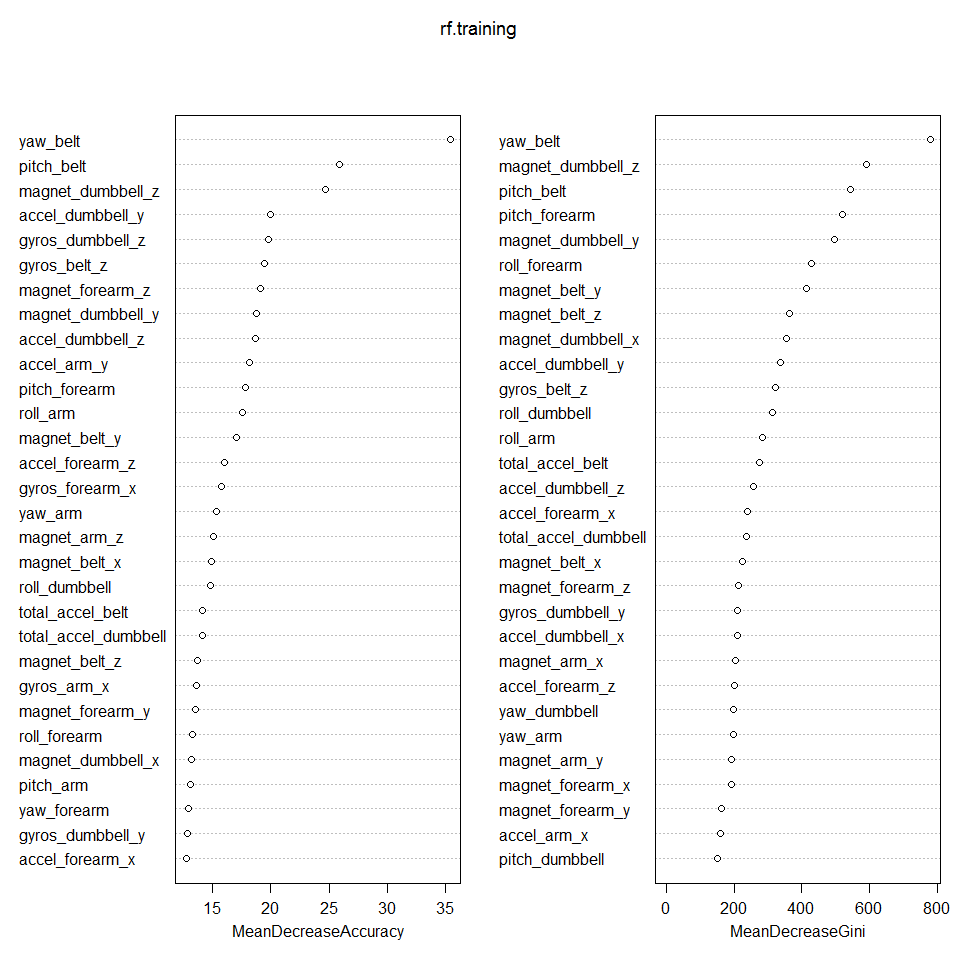
The single tree is not good enough, so we are going to use bootstrap to improve the accuracy. We are going to try random forests.

# Random Forests

set.seed(2125) #Birthday date of me and my girlfriend :)  
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: 6  
##   
## OOB estimate of error rate: 0.63%  
## Confusion matrix:  
## A B C D E class.error  
## A 3902 1 1 0 2 0.001024066  
## B 13 2635 9 0 1 0.008653123  
## C 0 16 2376 4 0 0.008347245  
## D 0 0 29 2221 2 0.013765542  
## E 0 0 2 6 2517 0.003168317

varImpPlot(rf.training,)



# Evaluate this tree on the test data.

Our Random Forest model shows OOB estimate of error rate: 0.72% for the training data. Now we will predict it for out-of sample accuracy.

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

## [1] 0.9940527

# Conclusion: Predict the testing data from the website

answers <- predict(rf.training, testingOrg)  
# See answers  
answers

## 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

Then print the result to files.

# Function to write "answers" vector to files  
pml\_write\_files = function(x){  
 n = length(x)  
 for(i in 1:n){  
 filename = paste0("problem\_id\_", i ,".txt")  
 write.table(x[i], file = filename, quote = FALSE, row.names = FALSE, col.names = FALSE)  
 }  
}  
# Call the function to write "answers" vector to files  
pml\_write\_files(answers)