Practical Machine Learing Project

Junyang Liu

12/4/2016

Project Introduction

Background

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

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.

Goal

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Getting and Cleaning Data

Here are the packages needed for this project

```r
library(caret)
```

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

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

```r
library(rpart)
library(rpart.plot)
library(rattle)
```

```
## Warning: Failed to load RGtk2 dynamic library, attempting to install it.

## Please install GTK+ from http://r.research.att.com/libs/GTK_2.24.17-X11.pkg

## If the package still does not load, please ensure that GTK+ is installed and that it is on your PATH

## IN ANY CASE, RESTART R BEFORE TRYING TO LOAD THE PACKAGE AGAIN

## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(randomForest)
```

```
## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

Load Data from URL

```r
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

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

Partition

```r
Train <- createDataPartition(training$classe, p=0.6, list=FALSE)
myTrain <- training[Train, ]
myTest <- training[-Train, ]
```

Here are the dimentions of the training set after partition

```
dim(myTrain); dim(myTest)
```

```
## [1] 11776   160
```

```
## [1] 7846  160
```

Cleaning data

I used a `nearZeroVar` Function to get rid of the columns that has very low variance or has no variance. If the predictors have very low variance or have no variance, this predictor is not going to be very useful in prediction.

```
nzv <- nearZeroVar(myTrain, saveMetrics=TRUE)
myTrain <- myTrain[,nzv$nzv==FALSE]

nzv<- nearZeroVar(myTest,saveMetrics=TRUE)
myTest <- myTest[,nzv$nzv==FALSE]
```

ALso, I dropped the first column `X`, which is just a reference key of this data set.it not useful in our analysis and prediction.

```
myTrain <- myTrain[c(-1)]
```

now we need to clear our all the variables that has at least 70% of the data showing as `NA`. we also need to drop `classe` in our real test set. because that is the column we are tring to predict.

```
keep<-which(apply(is.na(myTrain[,1:ncol(myTrain)]),2,sum)/nrow(myTrain) <= 0.7)

myTrain<- myTrain[,names(keep)]
myTest<- myTest[,names(keep)]
testing <- testing[names(keep)[-58]]
```

Check data dimentions to make sure we did it right

```
dim(myTest)
```

```
## [1] 7846   58
```

```
dim(testing)
```

```
## [1] 20 57
```

Convert the class of test data into the same class as training data

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

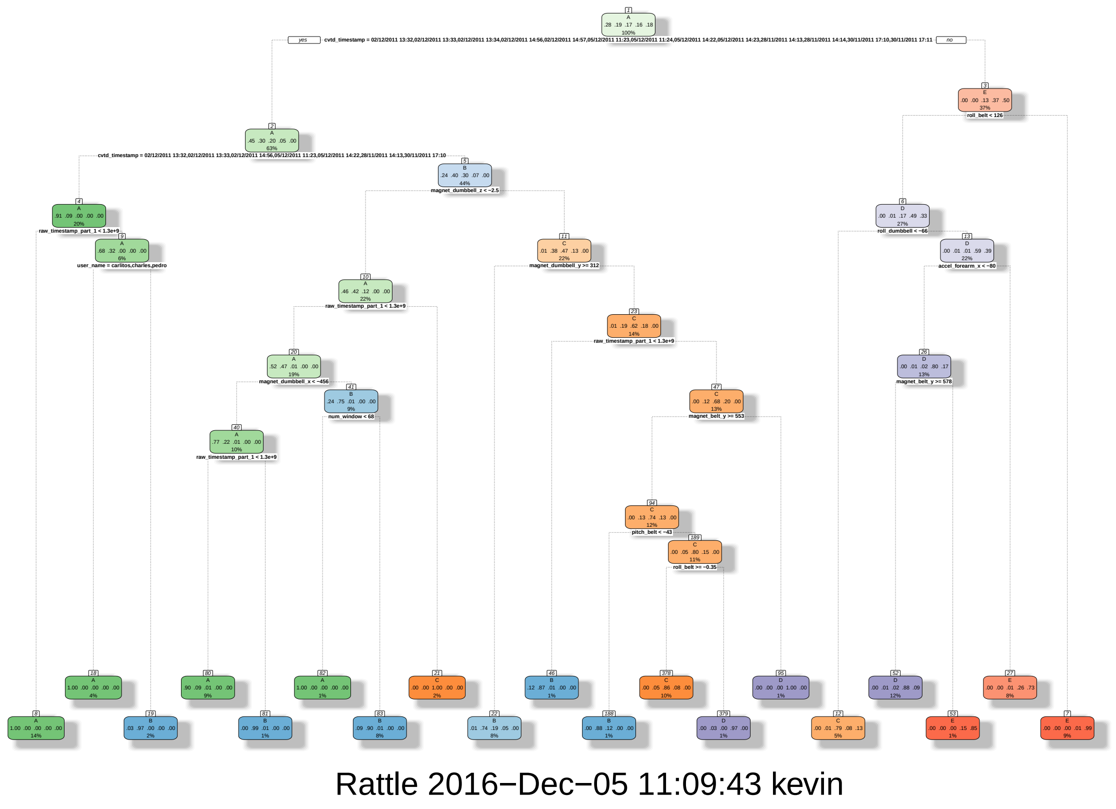
Verify our tranformation is working

```r
testing <- rbind(myTrain[2, -58] , testing)
testing <- testing[-1,]
```

Prediction using Decision Tree

Fit decision tree model

```r
fit1 <- rpart(classe ~ ., data=myTrain, method="class")
fancyRpartPlot(fit1)
```



Applying Fit model to prediction

```r
prediction1 <- predict(fit1, myTest, type = "class")
result1 <- confusionMatrix(prediction1, myTest$classe)
result1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2151   64    8    2    0
##          B   78 1392  163   30    0
##          C    3   51 1176  107   61
##          D    0   11    6  935   74
##          E    0    0   15  212 1307
##
## Overall Statistics
##
##                Accuracy : 0.8872
##                  95% CI : (0.88, 0.8941)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8572
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9637   0.9170   0.8596   0.7271   0.9064
## Specificity            0.9868   0.9572   0.9657   0.9861   0.9646
## Pos Pred Value         0.9667   0.8370   0.8412   0.9113   0.8520
## Neg Pred Value         0.9856   0.9796   0.9702   0.9485   0.9786
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2742   0.1774   0.1499   0.1192   0.1666
## Detection Prevalence   0.2836   0.2120   0.1782   0.1308   0.1955
## Balanced Accuracy      0.9753   0.9371   0.9127   0.8566   0.9355
```

Prediction using Random Forest

Fit Random Forest model

```
fit2 <- randomForest(classe ~ ., data=myTrain)
```

Applying Fit model to prediction

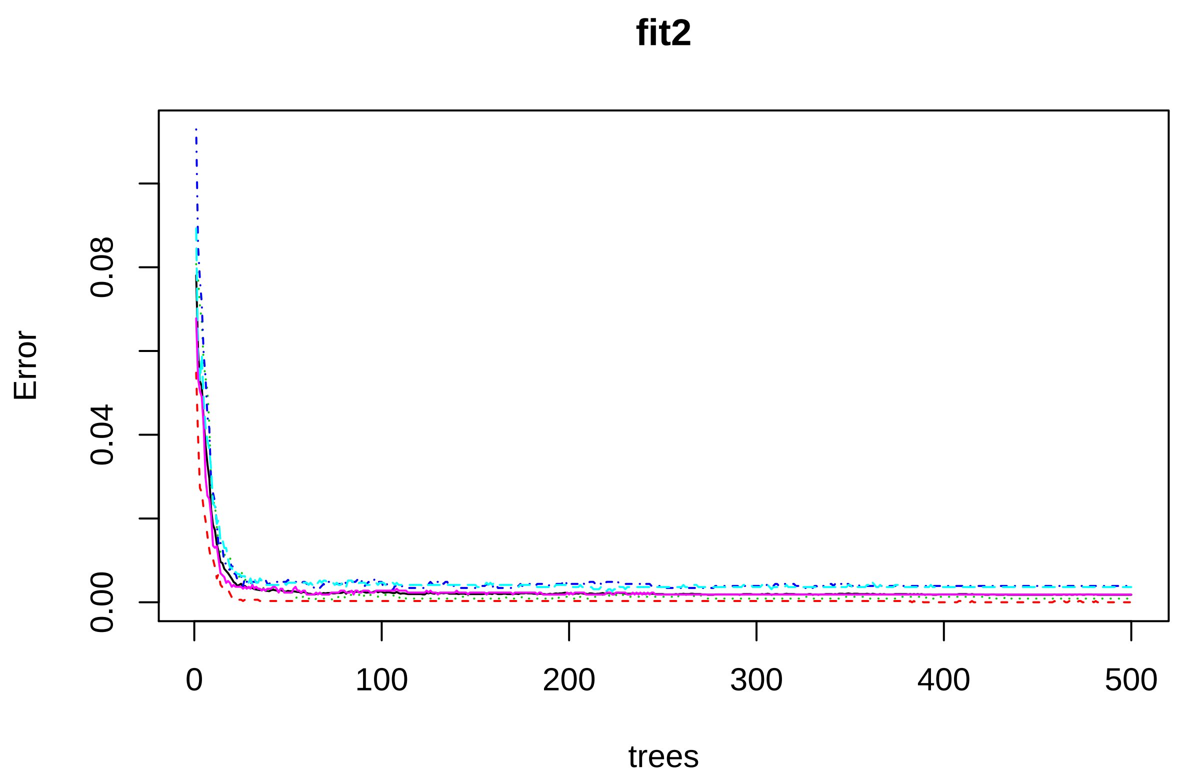
```
prediction2 <- predict(fit2, myTest, type = "class")
result2 <- confusionMatrix(prediction2, myTest$classe)
result2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1517    2    0    0
##          C    0    0 1365    1    0
##          D    0    0    1 1285    4
##          E    0    0    0    0 1438
##
## Overall Statistics
##
##                Accuracy : 0.9989
##                  95% CI : (0.9978, 0.9995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                  Kappa : 0.9985
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   0.9978   0.9992   0.9972
## Specificity            0.9998   0.9997   0.9998   0.9992   1.0000
## Pos Pred Value         0.9996   0.9987   0.9993   0.9961   1.0000
## Neg Pred Value         1.0000   0.9998   0.9995   0.9998   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1740   0.1638   0.1833
## Detection Prevalence   0.2846   0.1936   0.1741   0.1644   0.1833
## Balanced Accuracy      0.9999   0.9995   0.9988   0.9992   0.9986
```

Graph of random forest fit

```r
plot(fit2)
```



Prediction using Generalized Boosting

Fit Generalized Boosting model

```r
tControl <- trainControl(method = "repeatedcv",number = 5,repeats = 1)
fit3raw <- train(classe ~ ., data=myTrain, method = "gbm",trControl = tControl,verbose = FALSE)
```

```
## Loading required package: gbm
## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.1

## Loading required package: plyr
```

```
fit3<- fit3raw$finalModel
```

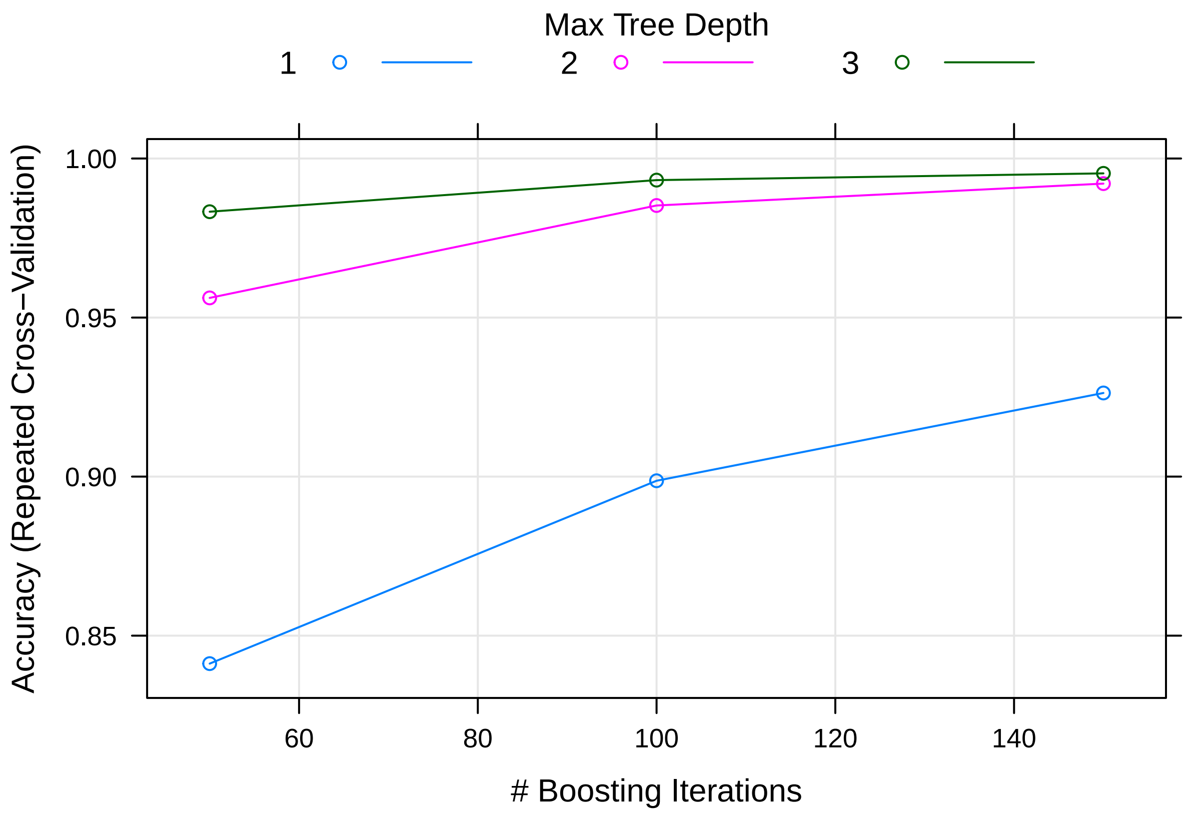
Applying Fit model to prediction

```
predition3 <- predict(fit3raw, newdata=myTest)
accuracy <- confusionMatrix(predition3, myTest$classe)
accuracy
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1514    1    0    0
##          C    0    2 1358    2    0
##          D    0    1    9 1282    6
##          E    0    0    0    2 1436
##
## Overall Statistics
##
##                Accuracy : 0.9969
##                  95% CI : (0.9955, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9961
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9927   0.9969   0.9958
## Specificity            0.9998   0.9998   0.9994   0.9976   0.9997
## Pos Pred Value         0.9996   0.9993   0.9971   0.9877   0.9986
## Neg Pred Value         1.0000   0.9994   0.9985   0.9994   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1930   0.1731   0.1634   0.1830
## Detection Prevalence   0.2846   0.1931   0.1736   0.1654   0.1833
## Balanced Accuracy      0.9999   0.9986   0.9960   0.9972   0.9978
```

Graph of Generalized Boosting

```
plot(fit3raw)
```



From above modeling fitting and prediction using our training data, we have found that random forest will provide the best prediction (have highest accuracy). Therefore, we will choose random forest as our predicting method for real test set

Predict on the real test set

```
prediction_real <- predict(fit2, testing, type = "class")
prediction_real
```

```
##  1 21  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
```

Submission

Results to a text file for submission

```r
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)
    }
}
pml_write_files(prediction_real)
```