Practical Machine Learning Course Project

Di Yang

December 18, 2015

Introduction

Background

It is possible to collect large amounts of data about personal exercise activity quickly and inexpensively. This data is reguarly quantified for how much an activity is done, but rarely quantified for how well the activity is done. This report uses data from accelerometers on the belt, forearm, arm, and dumb bell of six participants, who were asked to perform barbell lifts correctly and incorrectly in five different ways.

Data

The training and test data sets for this project were provided from http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) as follows:

Training: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

Testing: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Goal

The goal of this report is to predict the manner in which the participants completed the exercise. This is the “classe” variable in the training set. Other variables can be used for prediction as well. This report will describe how the model was built, how cross validation was used, what the expected out of sample error is, and the reasoning for the choices made. This prediction model will also be used to predict 20 test cases.

Setup

Getting the data

Set up the libraries

```
library(caret)
library(rattle)
library(RCurl)
library(rpart)
library(randomForest)
```

Load the data

```
set.seed(12345)
urlTrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urlTest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

train <- read.csv(url(urlTrain), na.strings=c("NA","#DIV/0!",""))
test <- read.csv(url(urlTest), na.strings=c("NA","#DIV/0!",""))
```

Partition the train data set into two

```
inTrain <- createDataPartition(train$classe, p=0.6, list=FALSE)
myTrain <- train[inTrain, ]
myTest <- train[-inTrain, ]
dim(myTrain); dim(myTest)
```

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

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

Cleaning the data

Transformations for training data set

Remove the NearZeroVariance variables

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

Remove the first column in myTrain data set to avoid interference with machine learning algorithms

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

Clean variables with more than 60 percent N/A

```
train3 <- myTrain
for(i in 1:length(myTrain)) {
    if( sum( is.na( myTrain[, i] ) ) /nrow(myTrain) >= .7) {
        for(j in 1:length(train3)) {
            if( length( grep(names(myTrain[i]), names(train3)[j]) ) == 1)  {
                train3 <- train3[ , -j]
            }
        }
    }
}
```

Set back to the original variable name

```
myTrain <- train3
rm(train3)
```

Transformations for test data set

```
clean1 <- colnames(myTrain)
clean2 <- colnames(myTrain[, -58])    # remove the classe column
myTest <- myTest[clean1]                # allow only variables in myTest that are
also in myTrain
test <- test[clean2]                # allow only variables in test that are also i
n myTrain

dim(myTest)     # check the number of observations
```

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

```
dim(test)       # check the number of observations
```

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

Coerce the data into the same type to ensure proper functioning of decision trees and RandomForest algorithm with the test data set

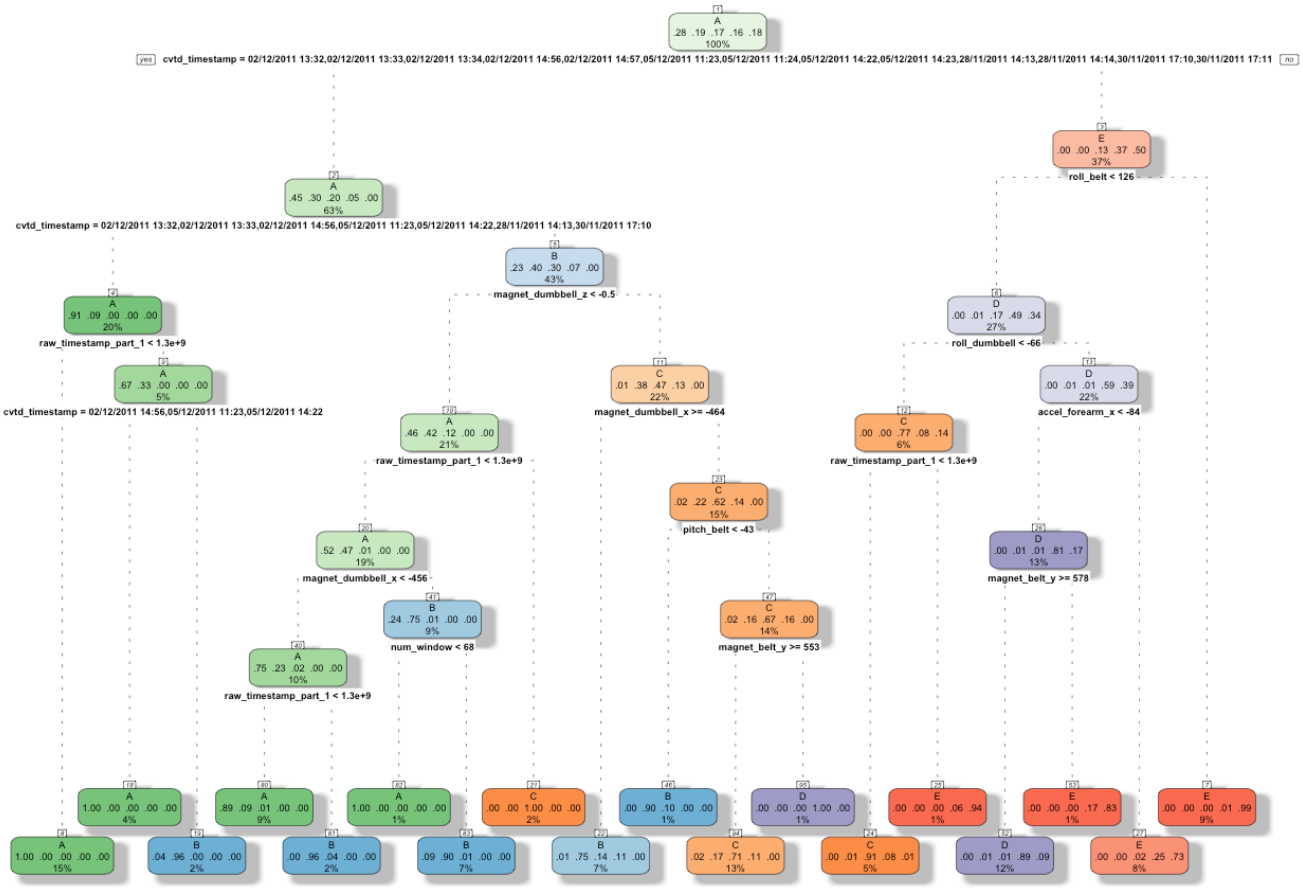
```
for (i in 1:length(test) ) {
    for(j in 1:length(myTrain)) {
        if( length( grep(names(myTrain[i]), names(test)[j]) ) == 1)  {
            class(test[j]) <- class(myTrain[i])
        }
    }
}
```

To get the same class between test and myTrain

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

Prediction with Decision Trees

```
set.seed(12345)
modFitA1 <- rpart(classe ~ ., data=myTrain, method="class")
fancyRpartPlot(modFitA1)
```

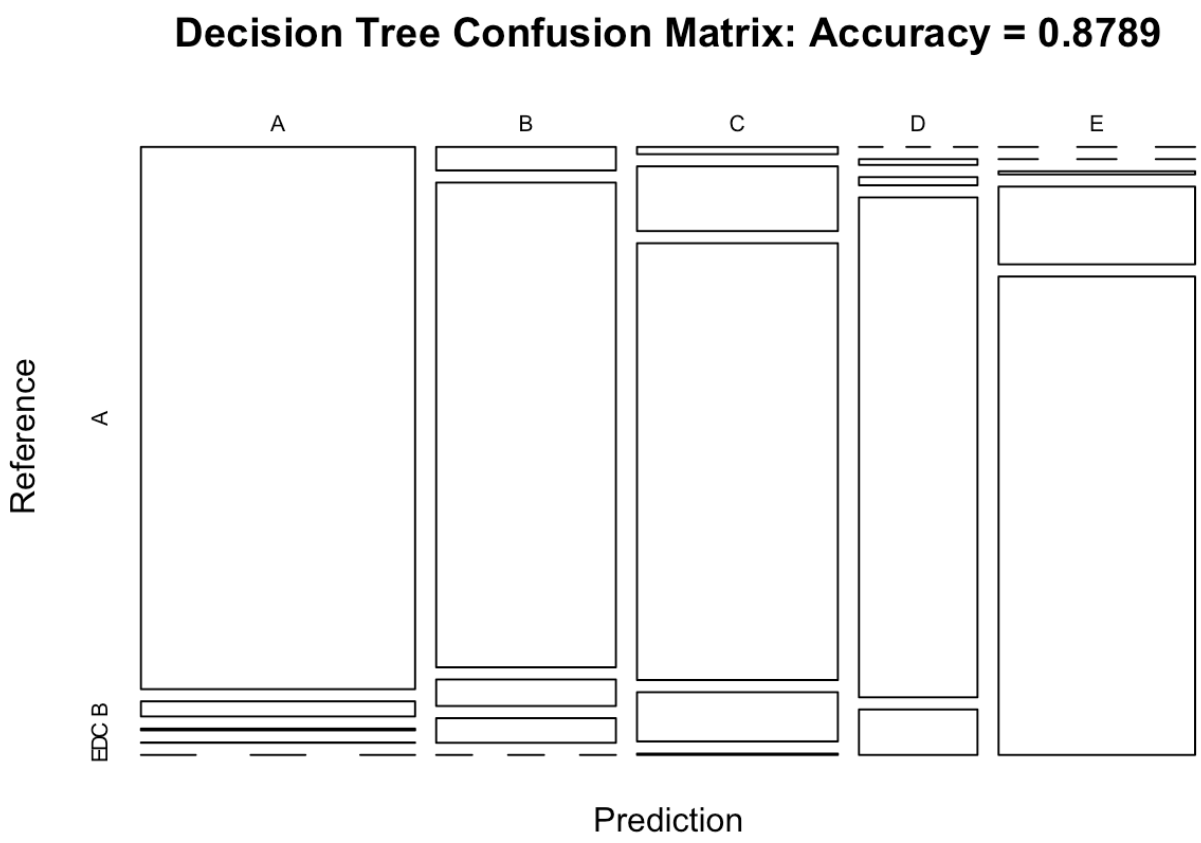


Rattle 2015-Dec-18 13:02:36 diyaaang

```
predictionsA1 <- predict(modFitA1, myTest, type = "class")
cmtree <- confusionMatrix(predictionsA1, myTest$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2150   60    7    1    0
##          B   61 1260   69   64    0
##          C   21  188 1269  143    4
##          D    0   10   14  857   78
##          E    0    0    9  221 1360
##
## Overall Statistics
##
##                Accuracy : 0.8789
##                  95% CI : (0.8715, 0.8861)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8468
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9633   0.8300   0.9276   0.6664   0.9431
## Specificity            0.9879   0.9693   0.9450   0.9845   0.9641
## Pos Pred Value         0.9693   0.8666   0.7809   0.8936   0.8553
## Neg Pred Value         0.9854   0.9596   0.9841   0.9377   0.9869
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2740   0.1606   0.1617   0.1092   0.1733
## Detection Prevalence   0.2827   0.1853   0.2071   0.1222   0.2027
## Balanced Accuracy      0.9756   0.8997   0.9363   0.8254   0.9536
```

```
plot(cmtree$table, col = cmtree$byClass, main = paste("Decision Tree Confusion Mat
rix: Accuracy =", round(cmtree$overall['Accuracy'], 4)))
```

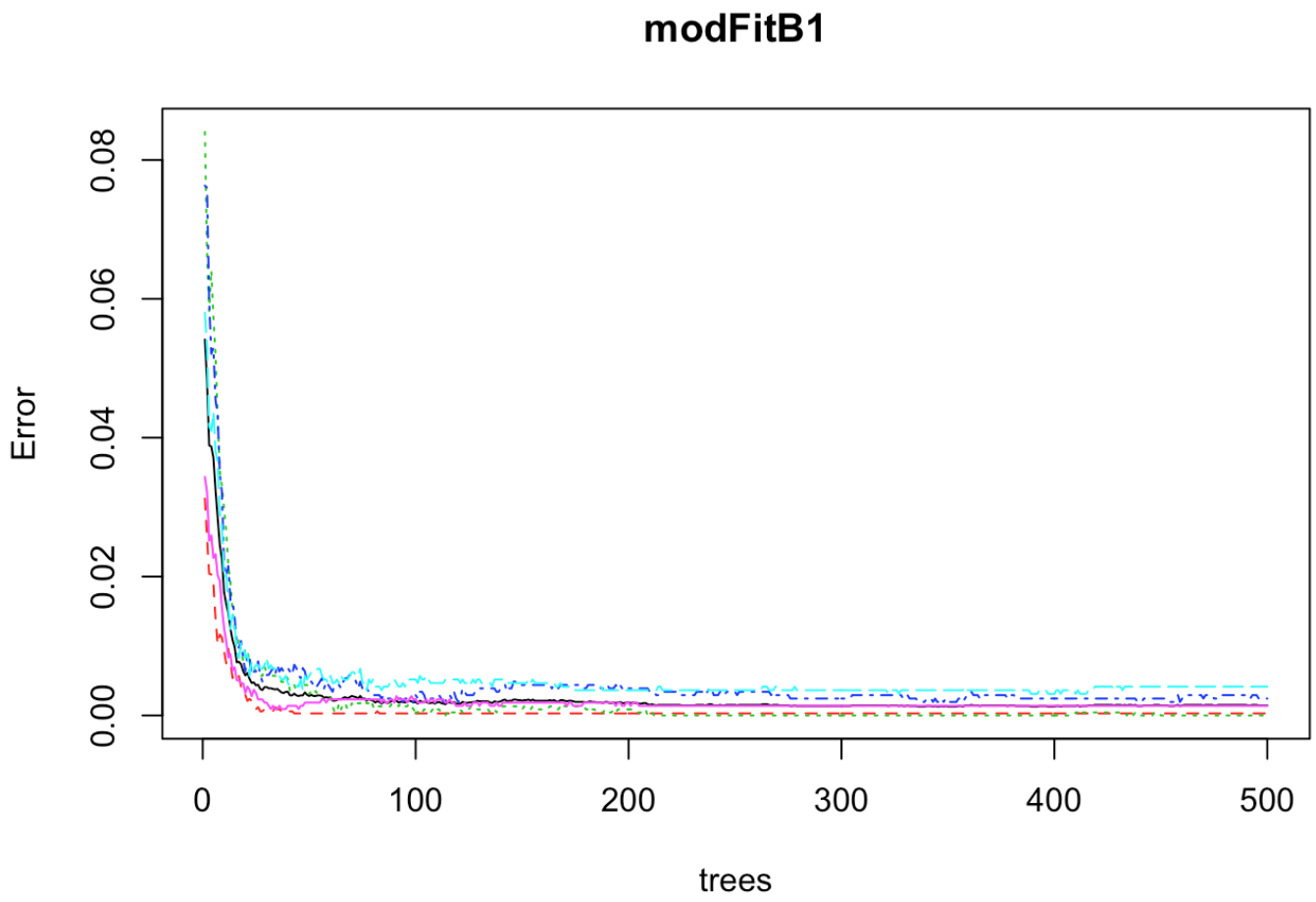


Prediction with Random Forests

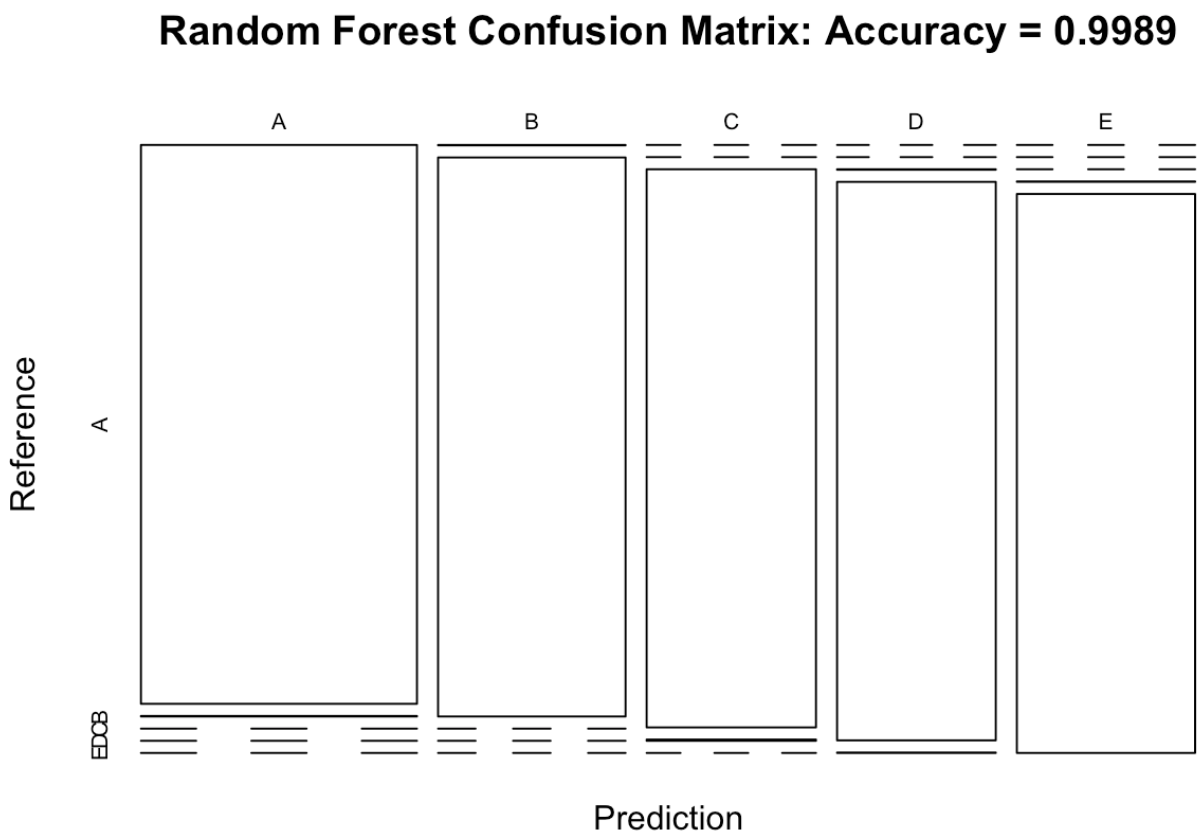
```
set.seed(12345)
modFitB1 <- randomForest(classe ~ ., data=myTrain)
predictionB1 <- predict(modFitB1, myTest, type = "class")
cmrf <- confusionMatrix(predictionB1, myTest$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    2    0    0    0
##          B    1 1516    0    0    0
##          C    0    0 1367    3    0
##          D    0    0    1 1282    1
##          E    0    0    0    1 1441
##
## 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            0.9996   0.9987   0.9993   0.9969   0.9993
## Specificity            0.9996   0.9998   0.9995   0.9997   0.9998
## Pos Pred Value         0.9991   0.9993   0.9978   0.9984   0.9993
## Neg Pred Value         0.9998   0.9997   0.9998   0.9994   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1742   0.1634   0.1837
## Detection Prevalence   0.2846   0.1933   0.1746   0.1637   0.1838
## Balanced Accuracy      0.9996   0.9993   0.9994   0.9983   0.9996
```

```
plot(modFitB1)
```




```
plot(cmrf$table, col = cmtree$byClass, main = paste("Random Forest Confusion Matri
x: Accuracy =", round(cmrf$overall['Accuracy'], 4)))
```



Predicting Results on the Test Data

Random Forests gave an Accuracy in the myTest dataset of 99.89%, which was more accurate than the Accuracy on the Decision Trees. The expected out-of-sample error is 100-99.89 = 0.11%.

```
predictionB2 <- predict(modFitB1, test, type = "class")
predictionB2
```

```
##  2 31  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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
```

```
# write the results to a .txt file to submit
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=FALS
E)
    }
}

# pml_write_files(predictionB2)
```