Practical Machine Learning - Assignment

Chenny

03/30/2016

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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data Processing and Analysis

The training and testing datasets used in the analysis may be found as follows:

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

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

```
library(caret)
```

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

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

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

```
## Rattle: A free graphical interface for data mining with R.
## XXXX 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## 键入'rattle()'去轻摇、晃动、翻滚你的数据。
```

```
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
```

```
library(knitr)
```

First, loading dataset:

```
training <- read.csv(file="pml-training.csv",na.strings=c("NA", ""),header=TRU
E)
dim(training)
```

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

```
testing <- read.csv(file="pml-testing.csv",na.strings=c("NA", ""),header=TRUE)
dim(testing)
```

```
## [1]  20 160
```

Train dataset

```
inTrain <- createDataPartition(training$classe, p=0.75, list=FALSE)
SubTraining <- training[inTrain, ]
SubTesting <- training[-inTrain, ]
dim(SubTraining)
```

```
## [1] 14718   160
```

```
dim(SubTesting)
```

```
## [1] 4904  160
```

Remove null data

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

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

Romve first column

```
SubTraining <- SubTraining[c(-1)]
dim(SubTraining)
```

```
## [1] 14718   121
```

Remove variables that has more than 60% NA.

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

Transform dataset

```
matchCol<- colnames(SubTraining)
matchCol2 <- colnames(SubTraining[ , -58])
SubTesting <- SubTesting [matchCol]
testing <- testing [matchCol2]
dim(SubTesting)
```

```
## [1] 4904   58
```

```
dim(testing)
```

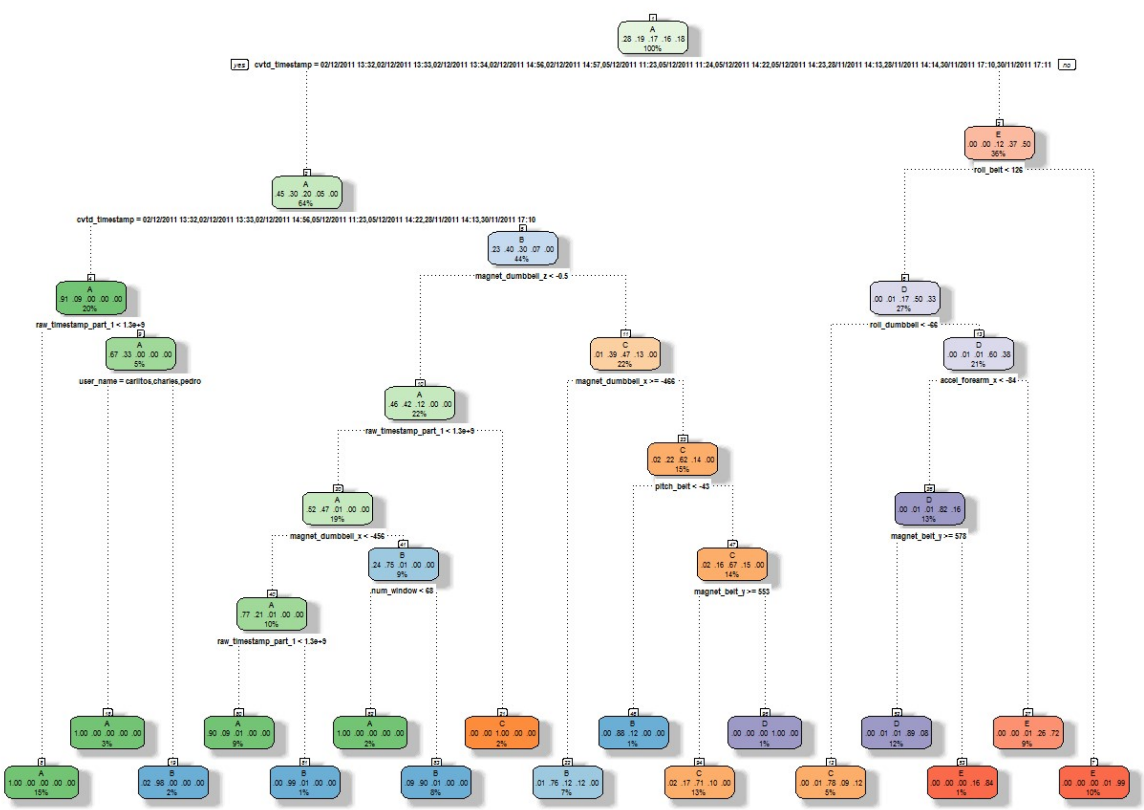
```
## [1] 20 57
```

Coerce the data into the same type

```
for (i in 1:length(testing) ) {
    for(j in 1:length(SubTraining)) {
        if( length( grep(names(SubTraining[i]), names(testing)[j]) ) == 1)  {
            class(testing[j]) <- class(SubTraining[i])
        }
    }
}
testing <- rbind(SubTraining[2, -58] , testing)
testing <- testing[-1,]
```

Predict with decision trees

```
set.seed(2016)
modFitDT <- rpart(classe ~ ., data=SubTraining, method="class")
fancyRpartPlot(modFitDT)
```

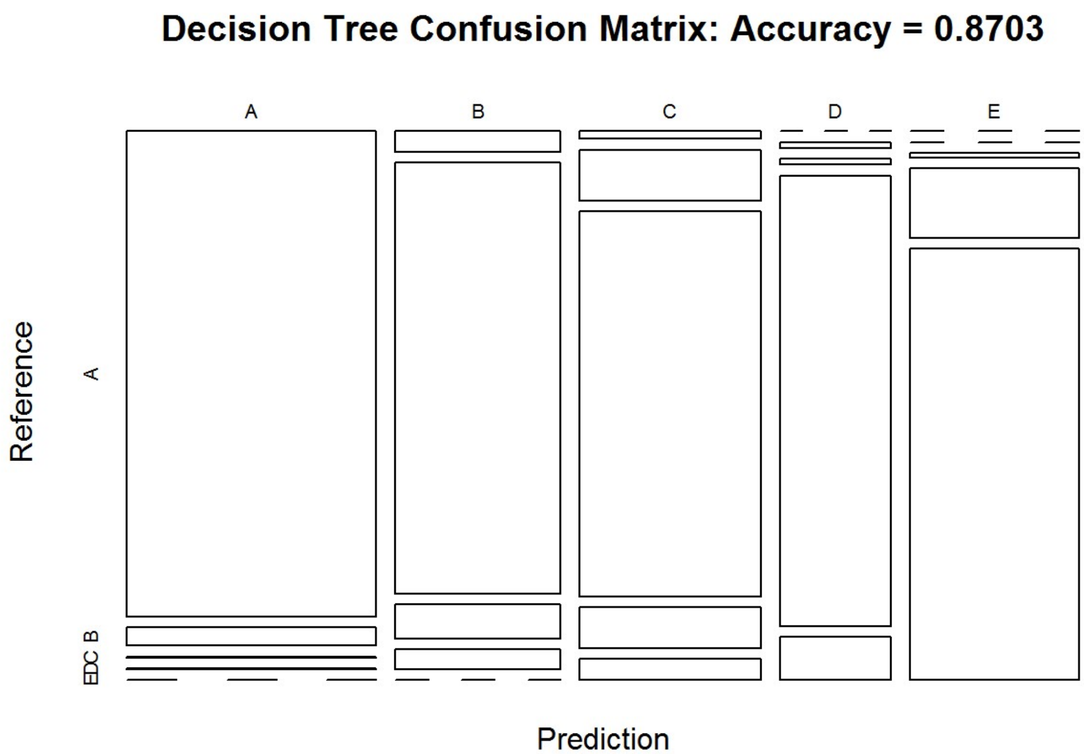


Rattle 2016-4月-02 22:46:52 XCY

```
predictDT <- predict(modFitDT, SubTesting, type = "class")
TreeDT <- confusionMatrix(predictDT, SubTesting$classe)
TreeDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1342   50    2    2    0
##          B   38  790   62   36    0
##          C   15  102  776   82   42
##          D    0    7    7  554   53
##          E    0    0    8  130  806
##
## Overall Statistics
##
##                Accuracy : 0.8703
##                  95% CI : (0.8606, 0.8796)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8359
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9620   0.8325   0.9076   0.6891   0.8946
## Specificity            0.9846   0.9656   0.9405   0.9837   0.9655
## Pos Pred Value         0.9613   0.8531   0.7630   0.8921   0.8538
## Neg Pred Value         0.9849   0.9600   0.9797   0.9416   0.9760
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2737   0.1611   0.1582   0.1130   0.1644
## Detection Prevalence   0.2847   0.1888   0.2074   0.1266   0.1925
## Balanced Accuracy      0.9733   0.8990   0.9240   0.8364   0.9300
```

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

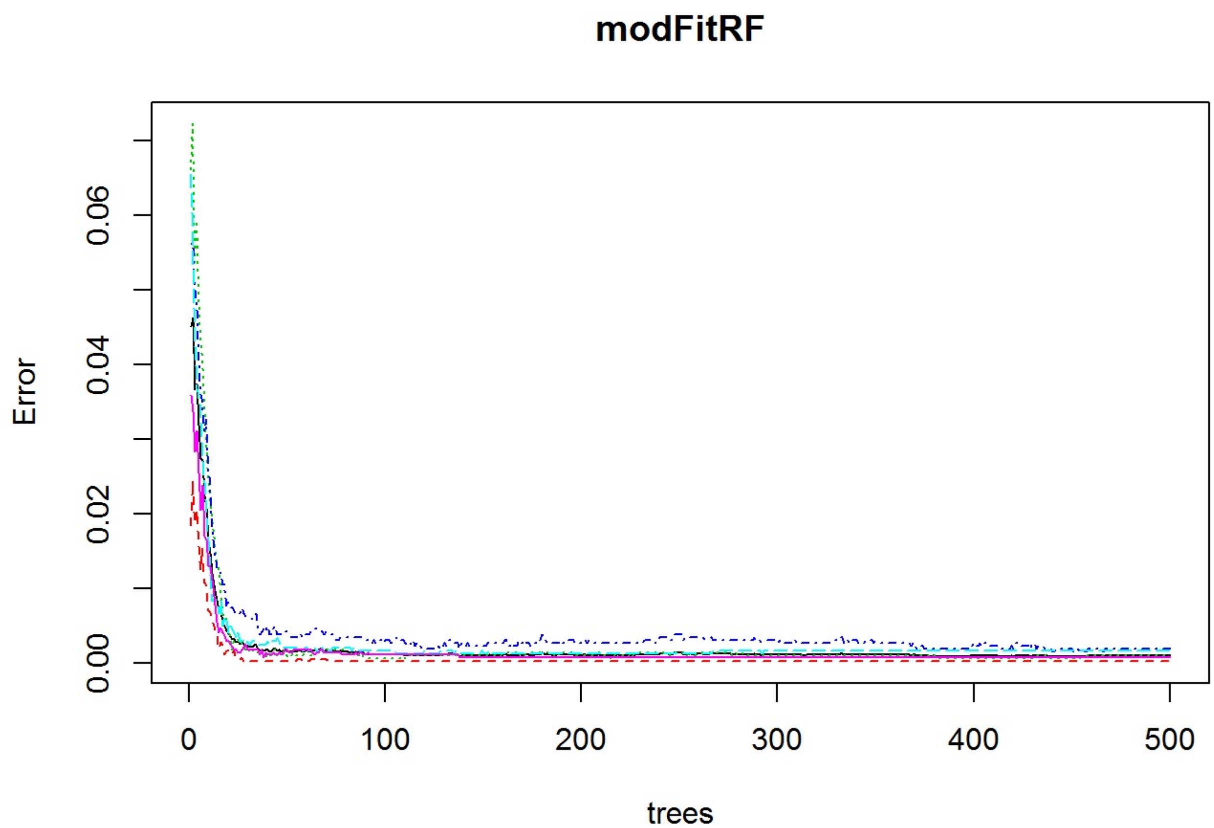


Prediction with Random Forests

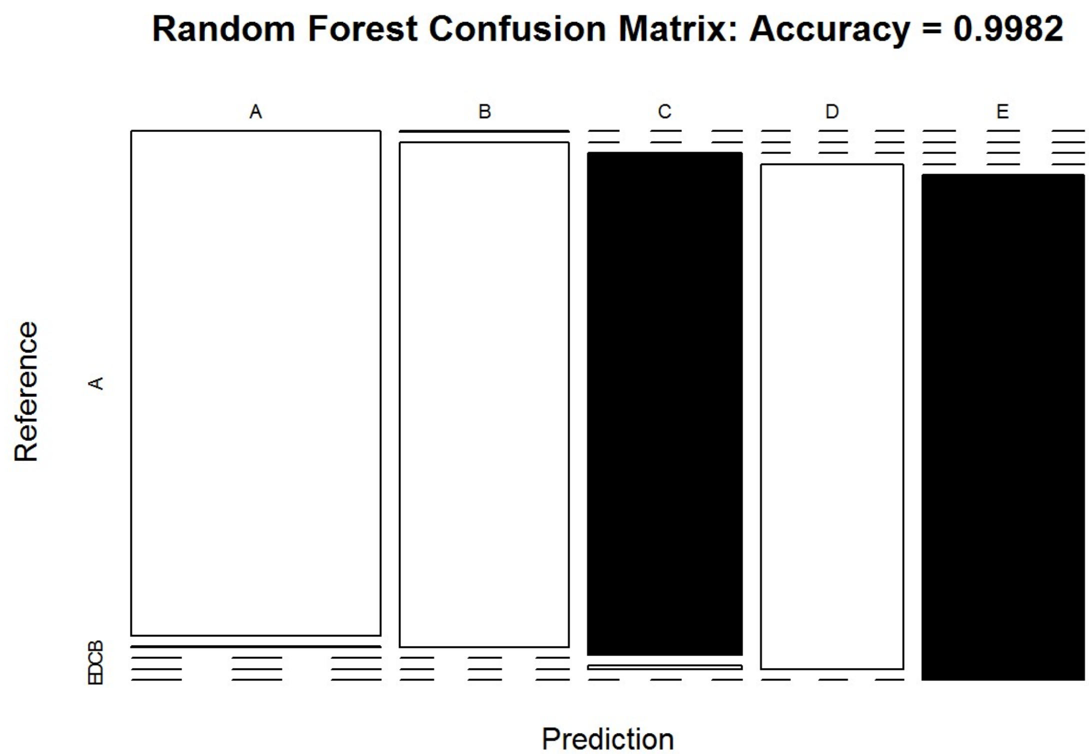
```
modFitRF <- randomForest(classe ~ ., data=SubTraining)
predictRF <- predict(modFitRF, SubTesting, type = "class")
TreeRF <- confusionMatrix(predictRF, SubTesting$classe)
TreeRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    2    0    0    0
##          B    1  947    0    0    0
##          C    0    0  855    6    0
##          D    0    0    0  798    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9982
##                  95% CI : (0.9965, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9977
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9979   1.0000   0.9925   1.0000
## Specificity            0.9994   0.9997   0.9985   1.0000   1.0000
## Pos Pred Value         0.9986   0.9989   0.9930   1.0000   1.0000
## Neg Pred Value         0.9997   0.9995   1.0000   0.9985   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1931   0.1743   0.1627   0.1837
## Detection Prevalence   0.2847   0.1933   0.1756   0.1627   0.1837
## Balanced Accuracy      0.9994   0.9988   0.9993   0.9963   1.0000
```

```
plot (modFitRF)
```




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



Predicting Results on the Test Data Random Forests gave an better Accuracy in the SubTesting dataset.

```
predictRF2 <- predict(modFitRF, testing, type="class")
predictRF2
```

```
## 22  3  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
```