Practical Machine Learning Final Project

Shun-Wen Chang

December 16, 2015

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, our goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise.

Load Required Packges and Set Seeds

```
library(caret)
```

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

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

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.0.5 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(3314)
```

Getting Data

```
train <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),na.string
test <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),na.strings=
```

Divide the Training Data Set into Two Parts

60% for myTrain and 40% for myTest

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

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

```
dim(myTest)
```

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

Clean the Training Data

Process #1 : Remove near-zero-variance variables

```
myNZV <- nearZeroVar(myTrain, saveMetrics=TRUE)
NZVnames <- subset(myNZV, nzv==TRUE)
myNZVvars <- names(myTrain) %in% row.names(NZVnames)
myTrain <- myTrain[!myNZVvars]
dim(myTrain)
```

```
## [1] 11776   129
```

Process #2 : Remove the first two columns because they are user names and IDs

```
myTrain <- myTrain[c(-1,-2)]
```

Process #3 : Remove variables with over 50% NAs

```
train3 <- myTrain
for(i in 1:length(myTrain)) {
        if( sum( is.na( myTrain[, i] ) ) /nrow(myTrain) >= .5 ) {
        for(j in 1:length(train3)) {
            if( length( grep(names(myTrain[i]), names(train3)[j]) ) ==1)  {
                train3 <- train3[ , -j]
            }
        }
    }
}

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

```
## [1] 11776    57
```

Repeat the same cleaning procedure on myTest set

```
proc1 <- colnames(myTrain)
proc2 <- colnames(myTrain[,-57])

myTest <- myTest[proc1]
test <- test[proc2]

dim(test)
```

```
## [1] 20 56
```

Transofrm Data Types

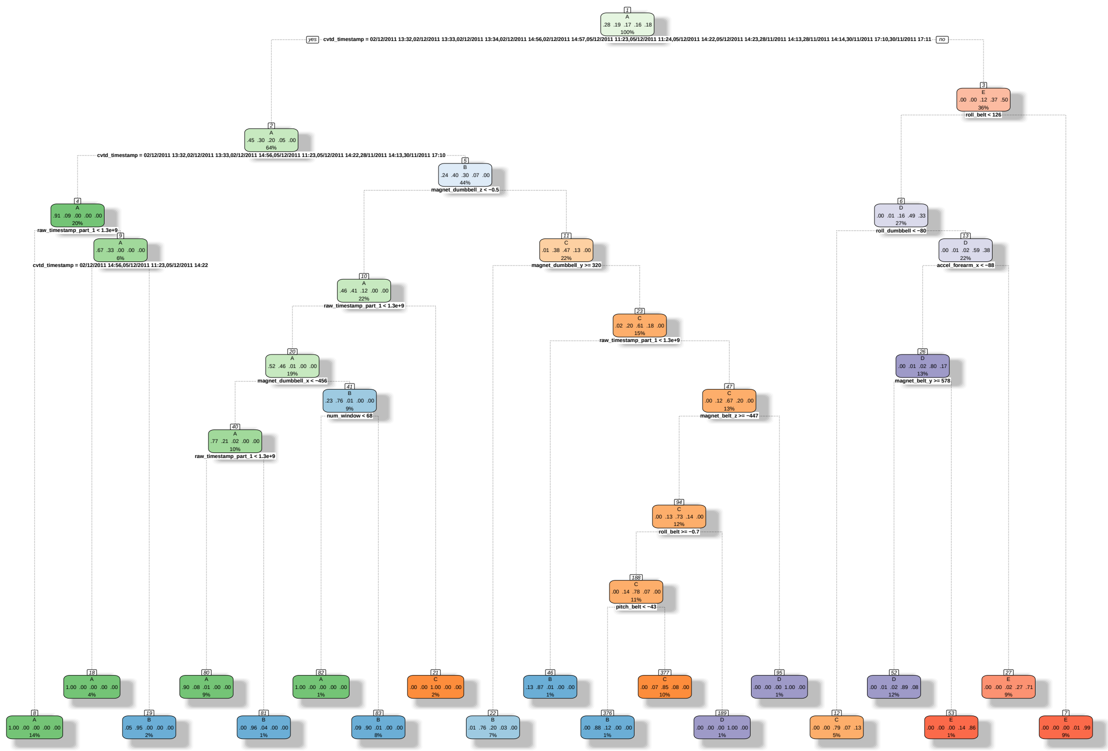
Because the data type in training data is different from that in the test data set, we have to coerce the data into the same type.

```
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])
        }
    }
}

test <- rbind(myTrain[2,-57], test)
test <- test[-1,]
```

Using Decision Tress for Prediction

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



Rattle 2015−Dec−17 22:18:46 EstherChang

```r
predictDT <- predict(modelDT, myTest, type = "class")
confusionMatrix(predictDT, myTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2159   69    6    2    0
##          B   72 1389  135   27    0
##          C    1   53 1187  104   43
##          D    0    7   23  936   79
##          E    0    0   17  217 1320
##
## Overall Statistics
##
##                Accuracy : 0.891
##                  95% CI : (0.8839, 0.8978)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.862
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9673   0.9150   0.8677   0.7278   0.9154
```

```
## Specificity            0.9863   0.9630   0.9690   0.9834   0.9635
## Pos Pred Value         0.9656   0.8558   0.8552   0.8957   0.8494
## Neg Pred Value         0.9870   0.9793   0.9720   0.9485   0.9806
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2752   0.1770   0.1513   0.1193   0.1682
## Detection Prevalence   0.2850   0.2069   0.1769   0.1332   0.1981
## Balanced Accuracy      0.9768   0.9390   0.9183   0.8556   0.9394
```

Using Random Forest for Prediction

```
modelRF <- randomForest(classe ~ ., data = myTrain)
predictRF <- predict(modelRF, myTest, type = "class")
confusionMatrix(predictRF, myTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1517    3    0    0
##          C    0    0 1365    1    0
##          D    0    0    0 1285    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9994
##                  95% CI : (0.9985, 0.9998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9992
##  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   1.0000
## Specificity            0.9998   0.9995   0.9998   1.0000   1.0000
## Pos Pred Value         0.9996   0.9980   0.9993   1.0000   1.0000
## Neg Pred Value         1.0000   0.9998   0.9995   0.9998   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1740   0.1638   0.1838
## Detection Prevalence   0.2846   0.1937   0.1741   0.1638   0.1838
## Balanced Accuracy      0.9999   0.9994   0.9988   0.9996   1.0000
```

As you shall see, Random Forests prediction has higher accuracy, 0.9994 compared to 0.891 from Decision Trees.

Generating Files for Submission

We use Random Forests for prediction since it gives better results!

```
predictRF_test <- predict(modelRF, test, type ="class")

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)
    }
}


write_files(predictRF_test)
```