Machine Learning Project - Prediction Assignment

Zubeir Siddiqui

June 22, 2017

Project Overview

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, 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.

The classe variable contains 5 different ways barbell lifts were performed correctly and incorrectly:

Class A: exactly according to the specification Class B: throwing the elbows to the front
Class C: lifting the dumbbell only halfway Class D: lowering the dumbbell only halfway
Class E: throwing the hips to the front

Objective

The goal of this project is to predict the manner in which people performed barbell lifts. This is the classe variable in the training set. Use any of the other variables to predict with. It should create a report describing how model is built, how used cross validation, what the expected out of sample error is, and why made the choices. It will also use developed prediction model to predict 20 different test cases.

Getting and loading the data

Adding libraries

```
library(caret)

## Warning: package 'caret' was built under R version 3.3.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.3.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.3

library(rpart)

## Warning: package 'rpart' was built under R version 3.3.3

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.3.3

library(RColorBrewer)
library(rattle)

## Warning: package 'rattle' was built under R version 3.3.3

## 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.

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.3.3

## 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(RCurl)

## Loading required package: bitops
```

Load Data

Download the training data

```
if(!file.exists("train_data.csv")){
binData <-
getBinaryURL("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv", ssl.verifypeer=0L, followlocation=1L)
destFileHandle <- file("train_data.csv", open="wb")
writeBin(binData,destFileHandle)
close(destFileHandle)
}
```

Download the testing data

```
if(!file.exists("test_data.csv")){
binData <-
getBinaryURL("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv", ssl.verifypeer=0L, followlocation=1L)
destFileHandle <- file("test_data.csv", open="wb")
writeBin(binData,destFileHandle)
close(destFileHandle)
}
```

Loading data into R and replacing missing values with NA

```
# Read the training file and replace missing values
train_data <- read.csv("train_data.csv", na.strings=c("NA","#DIV/0!",""),
header=TRUE)

# Read the testing file and replace missing values
test_data <- read.csv("test_data.csv", na.strings=c("NA","#DIV/0!",""),
header=TRUE)

# Look summary of Training data classe variable
summary(train_data$classe)

##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

Partition the data for Cross-validation

Split train data into 60% for training and 40% for testing based on classe variable

```
inTrain <- createDataPartition(y=train_data$classe, p = 0.60, list=FALSE)
training <- train_data[inTrain,]
testing <- train_data[-inTrain,]

dim(training); dim(testing)

## [1] 11776   160

## [1] 7846  160
```

Data Processing

Cleaning the data

Drop the first 7 variables because these are made up of metadata.

```
training <- training[,-c(1:7)]
```

Remove NearZeroVariance variables

```
# training data
nzv <- nearZeroVar(training, saveMetrics=TRUE)
training <- training[, nzv$nzv==FALSE]
# testing data
nzv <- nearZeroVar(testing, saveMetrics=TRUE)
testing <- testing[, nzv$nzv==FALSE]
```

Clean variables with more than 60% NA in training data

```
training_clean <- training
for(i in 1:length(training)) {
  if( sum( is.na( training[, i] ) ) /nrow(training) >= .6) {
    for(j in 1:length(training_clean)) {
      if( length( grep(names(training[i]), names(training_clean)[j]) ) == 1)
{
        training_clean <- training_clean[ , -j]
      }
    }
  }
}

# Set the new cleaned up dataset
training <- training_clean
```

Transform the test_data dataset to match with columns of trianing data

```
# Get the column names in the training dataset
columns <- colnames(training)
# Drop the classe variable
columns2 <- colnames(training[, -53])
# Subset the test data on the variables that are in the training data set
test_data <- test_data[columns2]
dim(test_data)

## [1] 20 52
```

Cross-Validation

Prediction with Random Forest

```
set.seed(12345)
modFit <- randomForest(classe ~ ., data=training)
```

```r
prediction <- predict(modFit, testing)
Conf_Mat <- confusionMatrix(prediction, testing$classe)
print(Conf_Mat)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    8    0    0    0
##          B    0 1506   13    0    0
##          C    0    4 1355   22    1
##          D    0    0    0 1262    3
##          E    0    0    0    2 1438
##
## Overall Statistics
##
##                Accuracy : 0.9932
##                  95% CI : (0.9912, 0.9949)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9915
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9921   0.9905   0.9813   0.9972
## Specificity            0.9986   0.9979   0.9958   0.9995   0.9997
## Pos Pred Value         0.9964   0.9914   0.9805   0.9976   0.9986
## Neg Pred Value         1.0000   0.9981   0.9980   0.9964   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1919   0.1727   0.1608   0.1833
## Detection Prevalence   0.2855   0.1936   0.1761   0.1612   0.1835
## Balanced Accuracy      0.9993   0.9950   0.9932   0.9904   0.9985
```

```r
model_overall_accuracy <- round(Conf_Mat$overall['Accuracy'] * 100, 2)
sample_error <- round(1 - Conf_Mat$overall['Accuracy'],2)
model_overall_accuracy
```

```
## Accuracy
##    99.32
```

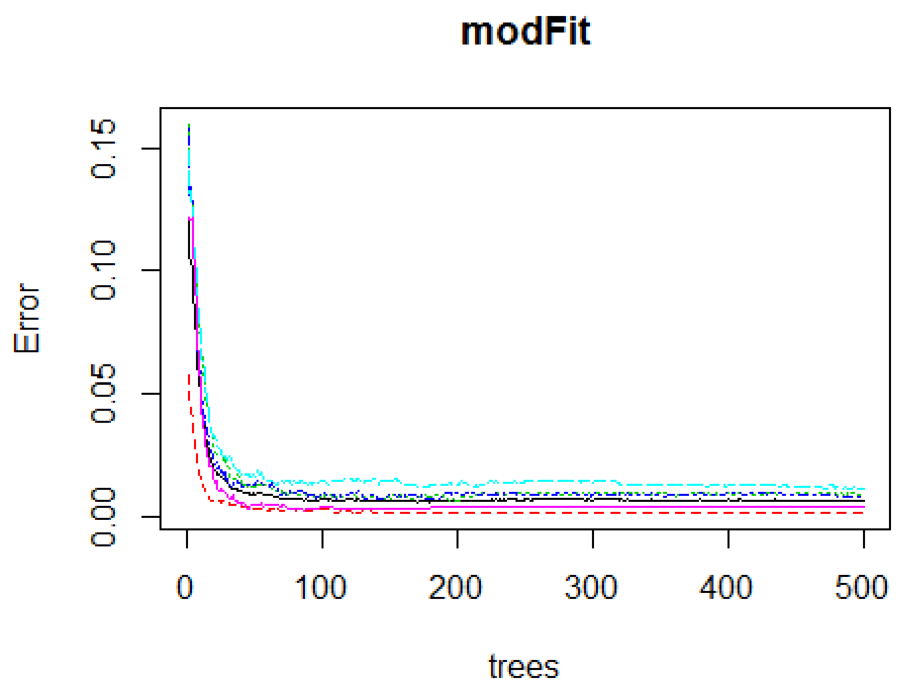
```r
sample_error
```

```
## Accuracy
##     0.01
```

The model is over 99% accurate on the testing data partitioned from the training data. The expected out of sample error is roughly 0.01%.

Plot Random Forest Model

```r
plot(modFit)
```



In the above figure, error rates of the model are plotted over 500 trees. The error rate is less than 0.03 for all 5 classe.

Prediction with a Decision Tree

```r
set.seed(12345)
modFit2 <- rpart(classe ~ ., data=training, method="class")
prediction2 <- predict(modFit2, testing, type="class")
Conf_Mat2 <- confusionMatrix(prediction2, testing$classe)
print(Conf_Mat2)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2034  282   46   84   52
##          B   97  912  150  142  284
##          C   54  189 1088  175  137
##          D   32  109   84  820   91
##          E   15   26    0   65  878
##
## Overall Statistics
##
##                Accuracy : 0.7306
##                  95% CI : (0.7206, 0.7404)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6577
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9113   0.6008   0.7953   0.6376   0.6089
## Specificity            0.9173   0.8936   0.9143   0.9518   0.9834
## Pos Pred Value         0.8143   0.5754   0.6622   0.7218   0.8923
## Neg Pred Value         0.9630   0.9032   0.9549   0.9306   0.9178
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2592   0.1162   0.1387   0.1045   0.1119
## Detection Prevalence   0.3184   0.2020   0.2094   0.1448   0.1254
## Balanced Accuracy      0.9143   0.7472   0.8548   0.7947   0.7962
```

```
model_overall_accuracy <- round(Conf_Mat2$overall['Accuracy'] * 100, 2)
sample_error <- round(1 - Conf_Mat2$overall['Accuracy'],2)
model_overall_accuracy
```

```
## Accuracy
##    73.06
```

```
sample_error
```

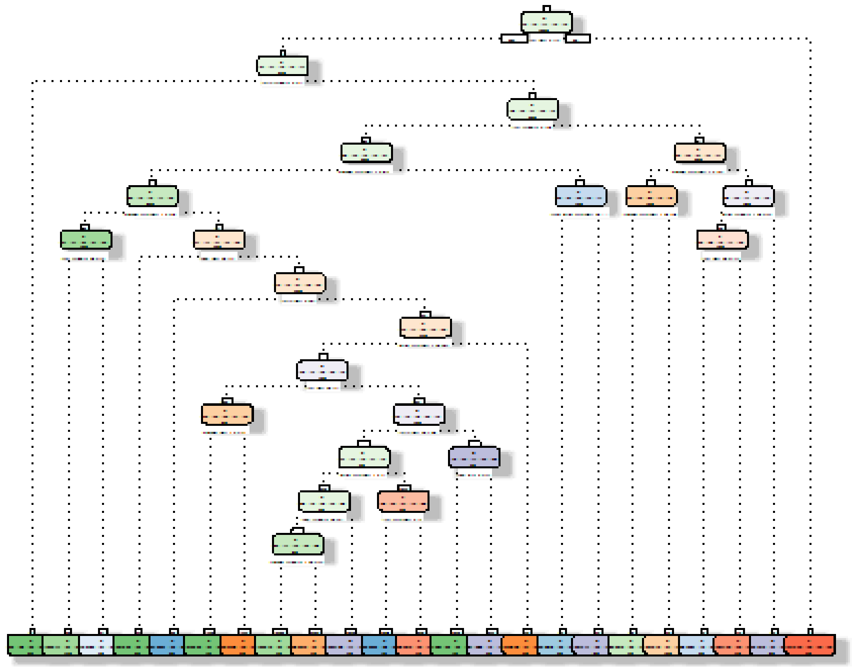
```
## Accuracy
##     0.27
```

The model is ~75% accurate on the testing data partitioned from the training data. The expected out of sample error is roughly ~0.25%.

Plot the decision tree model

```
fancyRpartPlot(modFit2)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2017-Jun-23 07:26:30 zubeir

Prediction on the Test Data

The Random Forest model gave an accuracy of over 99%, which is much higher than the ~75% accuracy from the Decision Tree. So we will use the Random Forest model to make the predictions on the test data to predict the way 20 participates performed the exercise.

```
final_prediction <- predict(modFit, test_data, type="class")
print(final_prediction)

##  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
```

Conclusion

For this data, the Random Forest proved to be a more accurate way to predict the manner in which the exercise was done.