Practical Machine Learning - Course Project

Brian O'Donnell

September 25, 2016

Synopsis

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

Goal

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. All of the other variables will be considered to make the prediction.

```
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
library(AppliedPredictiveModeling)
```

```
set.seed(333)
```

Data Loading

```
url_training <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
url_testing <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"

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

Data Partitioning

```
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ];
myTesting <- training[-inTrain, ]
```

Data Cleansing & Preparation

Identification and removal of near zero variance predictors

```
nzv <- nearZeroVar(myTraining, saveMetrics=TRUE)
nzv <- nearZeroVar(myTesting, saveMetrics=TRUE)
myTrainingX <- myTraining[,nzv$nzv==FALSE]
myTestingX <- myTesting[,nzv$nzv==FALSE]
```

Remove the first column

```
myTrainingX <- myTrainingX[,-1]
myTestingX <- myTestingX[,-1]
```

Clean up 'NA' in the data

```
myTrainingX <- myTrainingX[, names(myTrainingX)[sapply(myTrainingX, function
(x)
        ! (any(is.na(x) | x == "")))]]
myTestingX <- myTestingX[, names(myTestingX)[sapply(myTestingX, function (x)
        ! (any(is.na(x) | x == "")))]]
```

Prediction models

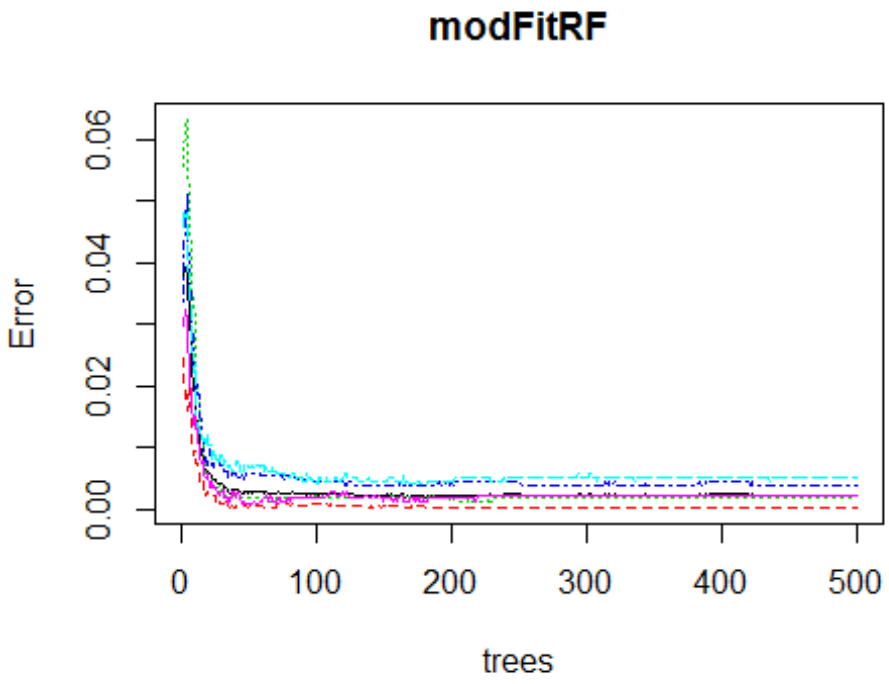
Build prediction models using Random Forests

```
modFitRF <- randomForest(classe ~ ., data=myTrainingX)
predictionB1 <- predict(modFitRF, myTesting, type = "class")
```

```
cmrf <- confusionMatrix(predictionB1, myTesting$classe)
cmrf
```

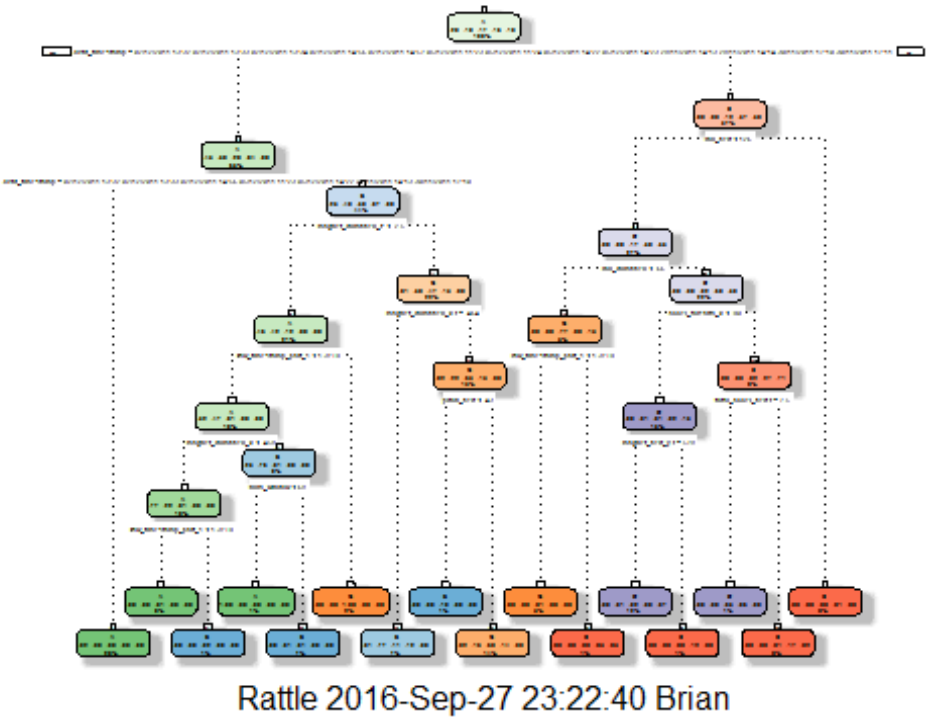
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    2    0    0    0
##          B    0 1516    4    0    0
##          C    0    0 1360    4    0
##          D    0    0    4 1282    3
##          E    0    0    0    0 1439
##
## Overall Statistics
##
##                Accuracy : 0.9978
##                  95% CI : (0.9965, 0.9987)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9973
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9987   0.9942   0.9969   0.9979
## Specificity            0.9996   0.9994   0.9994   0.9989   1.0000
## Pos Pred Value         0.9991   0.9974   0.9971   0.9946   1.0000
## Neg Pred Value         1.0000   0.9997   0.9988   0.9994   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1932   0.1733   0.1634   0.1834
## Detection Prevalence   0.2847   0.1937   0.1738   0.1643   0.1834
## Balanced Accuracy      0.9998   0.9990   0.9968   0.9979   0.9990
```

```
plot(modFitRF)
```



Build prediction models using decision trees

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



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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2141  201    8    3    0
##          B   68 1131   88   59    0
##          C   23  175 1253  205    5
##          D    0   11   11  883   96
##          E    0    0    8  136 1341
##
## Overall Statistics
##
##                Accuracy : 0.8602
##                  95% CI : (0.8523, 0.8678)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8228
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9592   0.7451   0.9159   0.6866   0.9300
## Specificity            0.9622   0.9660   0.9370   0.9820   0.9775
## Pos Pred Value         0.9099   0.8403   0.7544   0.8821   0.9030
## Neg Pred Value         0.9834   0.9405   0.9814   0.9411   0.9841
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2729   0.1441   0.1597   0.1125   0.1709
## Detection Prevalence   0.2999   0.1716   0.2117   0.1276   0.1893
## Balanced Accuracy      0.9607   0.8555   0.9265   0.8343   0.9537
```

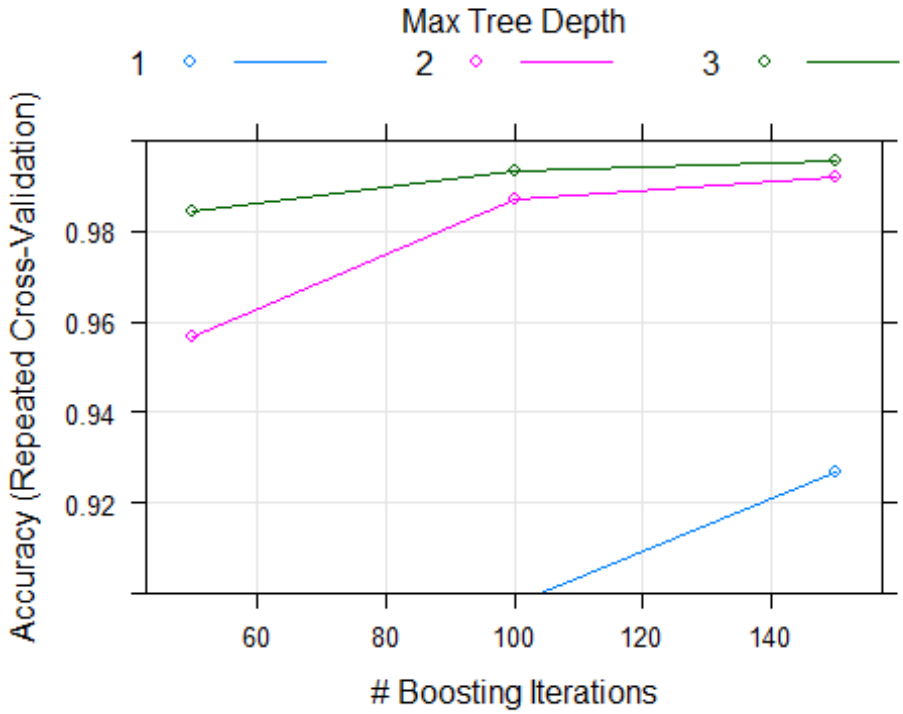
Build prediction models using Generalized Boosted Regression

```
fitControl <- trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 1)

modFitBR <- train(classe ~ ., data=myTrainingX, method = "gbm",
                  trControl = fitControl,
                  verbose = FALSE)
gbmFinMod1 <- modFitBR$finalModel
gbmPredTest <- predict(modFitBR, newdata=myTrainingX)
gbmAccuracyTest <- confusionMatrix(gbmPredTest, myTrainingX$classe)
gbmAccuracyTest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    1    0    0    0
##          B    0 2275    1    0    0
##          C    0    2 2049    0    0
##          D    0    1    4 1929    1
##          E    0    0    0    1 2164
##
## Overall Statistics
##
##                Accuracy : 0.9991
##                  95% CI : (0.9983, 0.9995)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9988
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9982   0.9976   0.9995   0.9995
## Specificity            0.9999   0.9999   0.9998   0.9994   0.9999
## Pos Pred Value         0.9997   0.9996   0.9990   0.9969   0.9995
## Neg Pred Value         1.0000   0.9996   0.9995   0.9999   0.9999
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1740   0.1638   0.1838
## Detection Prevalence   0.2844   0.1933   0.1742   0.1643   0.1838
## Balanced Accuracy      0.9999   0.9991   0.9987   0.9994   0.9997
```

```
plot(modFitBR, ylim=c(0.9, 1))
```



Conclusion

Overall accuracies:

```
cmrf$overall[1]

##  Accuracy
## 0.9978333

cmtree$overall[1]

##  Accuracy
## 0.8601835

gbmAccuracyTest$overall[1]

##  Accuracy
## 0.9990659
```

The Generalized Boosted Regression model provides the best accuracy of the three prediction models although the Random Forests model is a very close second. Using Generalized Boosted Regression model on the test data:

```
bestModelPredictions <- predict(modFitBR, newdata=testing)
bestModelPredictions
```

```
##  [1] 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
```

Generate output per the project requirement

```
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(bestModelPredictions)
```