Practical Machine Learning

Joan Braithwaite

April 2nd, 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.

The goal of this project, is to predict the manner in which they did the exercise using data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. Participants 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. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

```
rm(list = ls())
if (!file.exists("pml-training.csv")) {
  download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = "pml-
}
if (!file.exists("pml-testing.csv")) {
  download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "pml-te
}
Test.data <- read.csv("pml-testing.csv", sep = ",", na.strings = c("", "NA"))
Train.data <- read.csv("pml-training.csv", sep = ",", na.strings = c("", "NA"))
```

Prepare the data

```
# Remove columns with NAs.
features <- names(Test.data[,colSums(is.na(Test.data)) == 0])[8:59]

# Only use features used in Test.data cases.
Train.data <- Train.data[,c(features,"classe")]
Test.data <- Test.data[,c(features,"problem_id")]
```

Building & Identifying the best model

The goal of this project is to predict the manner in which they did the exercise. So my process is to:

1) create a training and test set of data.

2) try a few different models to see which one would provide the most accurate prediction.

3) cross-validation was used on each model (using a separate set of data) to check for overfitting.

4) use the one with the highest accurary to predict the values from the test data to be submitted for the class as part of this project.

Create training and test data to verify model

```
library(caret)
```

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

```
set.seed(3433)

library(AppliedPredictiveModeling)

inTrain = createDataPartition(Train.data$classe, p = 0.75, list = F)
training = Train.data[inTrain,]
testing = Train.data[-inTrain,]
```

Prediction with Decision Trees

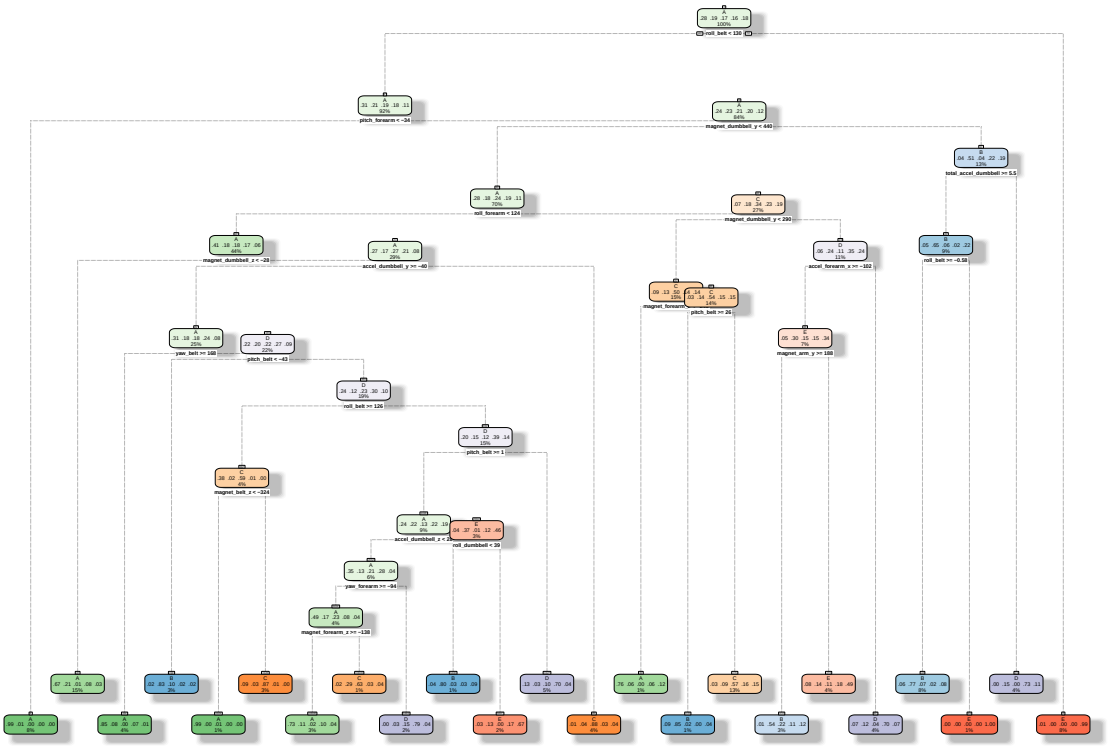
```
library(rpart)
library(rattle)
```

```
## Loading required package: RGtk2
## Rattle: A free graphical interface for data mining with R.
## Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
set.seed(12345)

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

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



Rattle 2016−Apr−02 21:45:26 Joan

```
predictionsDS <- predict(modFitDS, testing, type = "class")
cmtree <- confusionMatrix(predictionsDS, testing$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1267  219   10   86   38
##          B   25  530   85   28   71
##          C   38   94  684  139  107
##          D   42   64   54  491   42
##          E   23   42   22   60  643
##
## Overall Statistics
##
##                Accuracy : 0.7372
##                  95% CI : (0.7246, 0.7494)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6657
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9082   0.5585   0.8000   0.6107   0.7137
## Specificity            0.8994   0.9472   0.9066   0.9507   0.9633
## Pos Pred Value         0.7821   0.7172   0.6441   0.7085   0.8139
## Neg Pred Value         0.9610   0.8994   0.9555   0.9257   0.9373
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2584   0.1081   0.1395   0.1001   0.1311
## Detection Prevalence   0.3303   0.1507   0.2166   0.1413   0.1611
## Balanced Accuracy      0.9038   0.7528   0.8533   0.7807   0.8385
```

Prediction with Random Forests

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(12345)

modFitRF <- randomForest(classe ~ ., data=training)
predictionRF <- predict(modFitRF, testing, type = "class")
cmrf <- confusionMatrix(predictionRF, testing$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    7    0    0    0
##          B    0  942    4    0    0
##          C    0    0  849    5    0
##          D    0    0    2  798    5
##          E    0    0    0    1  896
##
## Overall Statistics
##
##                Accuracy : 0.9951
##                  95% CI : (0.9927, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9938
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9926   0.9930   0.9925   0.9945
## Specificity            0.9980   0.9990   0.9988   0.9983   0.9998
## Pos Pred Value         0.9950   0.9958   0.9941   0.9913   0.9989
```

```
## Neg Pred Value         1.0000   0.9982   0.9985   0.9985   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1921   0.1731   0.1627   0.1827
## Detection Prevalence   0.2859   0.1929   0.1741   0.1642   0.1829
## Balanced Accuracy      0.9990   0.9958   0.9959   0.9954   0.9971
```

Prediction with Generalized Boosted Regression

```
library(gbm)
```

```
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
```

```
set.seed(12345)

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

gbmFit1 <- train(classe ~ ., data=training, method = "gbm",
                 trControl = fitControl,
                 verbose = FALSE)
```

```
## Loading required package: plyr
```

```
gbmFinMod1 <- gbmFit1$finalModel

gbmPredTest <- predict(gbmFit1, newdata=testing)
gbmAccuracyTest <- confusionMatrix(gbmPredTest, testing$classe)
gbmAccuracyTest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1377   27    0    0    1
##          B   12  898   28   10    9
##          C    6   24  807   30   11
##          D    0    0   20  757   11
##          E    0    0    0    7  869
```

```
##
## Overall Statistics
##
##                Accuracy : 0.96
##                  95% CI : (0.9542, 0.9653)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9494
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9871   0.9463   0.9439   0.9415   0.9645
## Specificity            0.9920   0.9851   0.9825   0.9924   0.9983
## Pos Pred Value         0.9801   0.9383   0.9191   0.9607   0.9920
## Neg Pred Value         0.9949   0.9871   0.9881   0.9886   0.9921
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2808   0.1831   0.1646   0.1544   0.1772
## Detection Prevalence   0.2865   0.1951   0.1790   0.1607   0.1786
## Balanced Accuracy      0.9896   0.9657   0.9632   0.9670   0.9814
```

Overall Summary

Random Forests gave an accuracy in the testing dataset of 99.89%, which was more accurate than 96.00% using Generalized Boosted Regression or 73.72% using Decision Trees.

The expected out-of-sample error is 100-99.89 = 0.11%.

Use the prediction model to predict 20 different test cases

```
# Code to submit for test data
predictionRF2 <- predict(modFitRF, Test.data, type = "class")
predictionRF2
```

```
##  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
```