Practical Machine Learning: Prediction Assignment

Predicting exercise performance from wearables data

By Aoibhinn Reddington

Executive Summary

Data from wearables (such as FitBit, Nike FuelBand, and Jawbone Up) were analyzed to predict how the subjects performed on exercise tasks. Two different machine learning algorithms were compared. The random forest algorithm showed a much higher accuracy (99%) than the decision tree (69%), and was used for the final prediction.

Assignment

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/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.

What you should submit

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, you will use data from accelerometers on the belt, forearm, arm, and dumbell

of 6 participants. The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Methods

We make the prediction with two different algorithms: Random forests and decision tree. We assess their performances with cross validation. This means that we divide the training data set into a training set and a validation set. This is used to develop the model. Then, finally, the model can be assessed by testing it on the original testing set.

Our data is structured in such a way that each entry belongs to one of 5 classes. The output of our prediction should be the correct class. These classes are unordered, because they correspond to common mistakes made during the exercises. Therefore, out of sample error is a good measure of accuracy to assess model performance.

Results

Loading libraries and raw data

To ensure reproducibility, the libraries required and the preprocessing steps are listed here. Also, the seed is set to 23232.

```
library(caret); library(randomForest); library(rpart); library(rpart.plot);library(lattice); library(ggplot2); library(rattle)
```

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

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

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

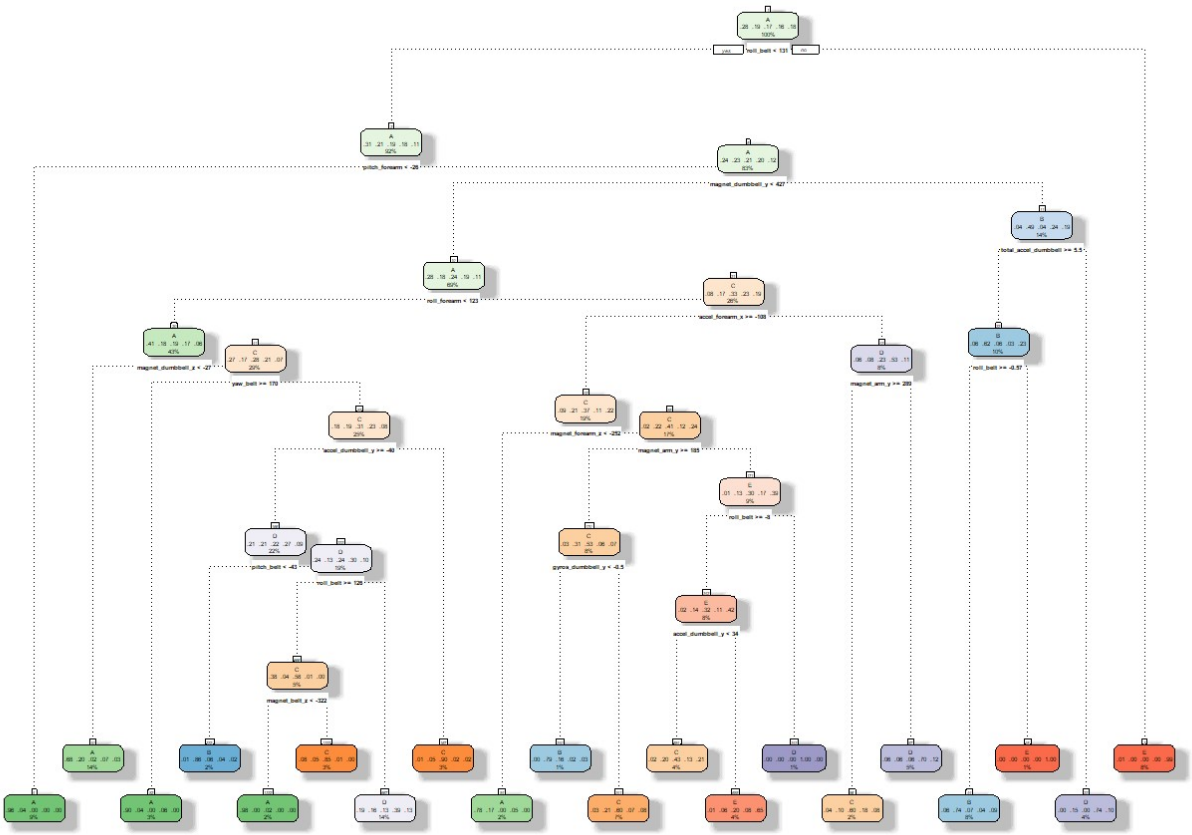
```
URL_train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
URL_test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

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

set.seed(23232)
```

Next, we create training, testing, and validation sets with the caret package. We put 70% of the data into the training subset and the other 30% in the test subset.

```
# Turn the classe variable into a factor
training$classe<-as.factor(training$classe)

# Removing columns with >= 90% NaNs
training <- training[,(colSums(is.na(training))<0.9*nrow(training))]

# Removing irrelevant variables
training <-training[,-c(1:7)]

# Creating train, test and validation sets
inTrain <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
training2 <- training[inTrain,]
testing2 <- training[-inTrain,]
```

The first model: Decision tree

```
model1 <- rpart(classe ~ ., data=training2, method="class")

# plot the decision tree
fancyRpartPlot(model1)
```



Rattle 2019-nov-20 16:17:47 areddington

Take a look at performance:

```
prediction1 <- predict(model1, testing2, type = "class")

confusionMatrix(prediction1, testing2$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1368  248   12   82   24
##          B   46  538   69   23   58
##          C   45  165  742  111  101
##          D  208  179  153  724  186
##          E    7    9   50   24  713
##
## Overall Statistics
##
##                Accuracy : 0.6941
##                  95% CI : (0.6822, 0.7059)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6139
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8172  0.47234   0.7232   0.7510   0.6590
## Specificity            0.9131  0.95870   0.9132   0.8525   0.9813
## Pos Pred Value         0.7889  0.73297   0.6375   0.4993   0.8879
## Neg Pred Value         0.9263  0.88332   0.9398   0.9459   0.9274
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2325  0.09142   0.1261   0.1230   0.1212
## Detection Prevalence   0.2946  0.12472   0.1978   0.2464   0.1364
## Balanced Accuracy      0.8651  0.71552   0.8182   0.8018   0.8201
```

The second model: Random Forest

```
model2 <- randomForest(classe ~. , data=training2, method="class")
prediction2 <- predict(model2, testing2, type = "class")
confusionMatrix(prediction2, testing2$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    9    0    0    0
##          B    0 1127   11    0    0
##          C    0    3 1015   11    0
##          D    0    0    0  953    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9942
##                  95% CI : (0.9919, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9927
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9895   0.9893   0.9886   1.0000
## Specificity            0.9979   0.9977   0.9971   1.0000   1.0000
## Pos Pred Value         0.9947   0.9903   0.9864   1.0000   1.0000
## Neg Pred Value         1.0000   0.9975   0.9977   0.9978   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1915   0.1725   0.1619   0.1839
## Detection Prevalence   0.2860   0.1934   0.1749   0.1619   0.1839
## Balanced Accuracy      0.9989   0.9936   0.9932   0.9943   1.0000
```

Conclusion

We see that the random forests method achieves a much higher accuracy: 99.42% versus the 69.41% that was achieved by the decision tree method.

The expected out of sample error is 1 - 99.42% = 0.58%

We therefore make our final prediction with the random forest model:

```
prediction <- data.frame(predict(model2,testing))

prediction
```

```
##    predict.model2..testing.
## 1                         B
## 2                         A
## 3                         B
## 4                         A
## 5                         A
## 6                         E
## 7                         D
## 8                         B
## 9                         A
## 10                        A
## 11                        B
## 12                        C
## 13                        B
## 14                        A
## 15                        E
## 16                        E
## 17                        A
## 18                        B
## 19                        B
## 20                        B
```