Practical Machine Learning Project

David Black

April 2018

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.

Step 1. Load the data

```
library("rattle")
```

```
## Warning: package 'rattle' was built under R version 3.4.4
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library("randomForest")
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

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

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

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

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

```
library("caret")
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.4.3
```

```
##
## Attaching package: 'ggplot2'
```

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

```
training=read.csv("pml-training.csv")
testing=read.csv("pml-testing.csv")
```

Step 2. Clean the data

Remove columns that are not needed and that have NA's. Data started with 160 variables and narrowed down to 53.

```
set.seed(1234)
CleanTrain=training[,-c(1:7)]
CleanTest=testing[,-c(1:7)]

CleanTrain=CleanTrain[,colSums(is.na(CleanTrain))==0]
CleanTrain=CleanTrain[,sapply(CleanTrain[,-c(86)],is.numeric)]

CleanTest=CleanTest[,colSums(is.na(CleanTest))==0]
CleanTest=CleanTest[,sapply(CleanTest[,-c(86)],is.numeric)]
```

Step 3. Partition the data

Partition the data into a training set (70%) and testing set (30%)

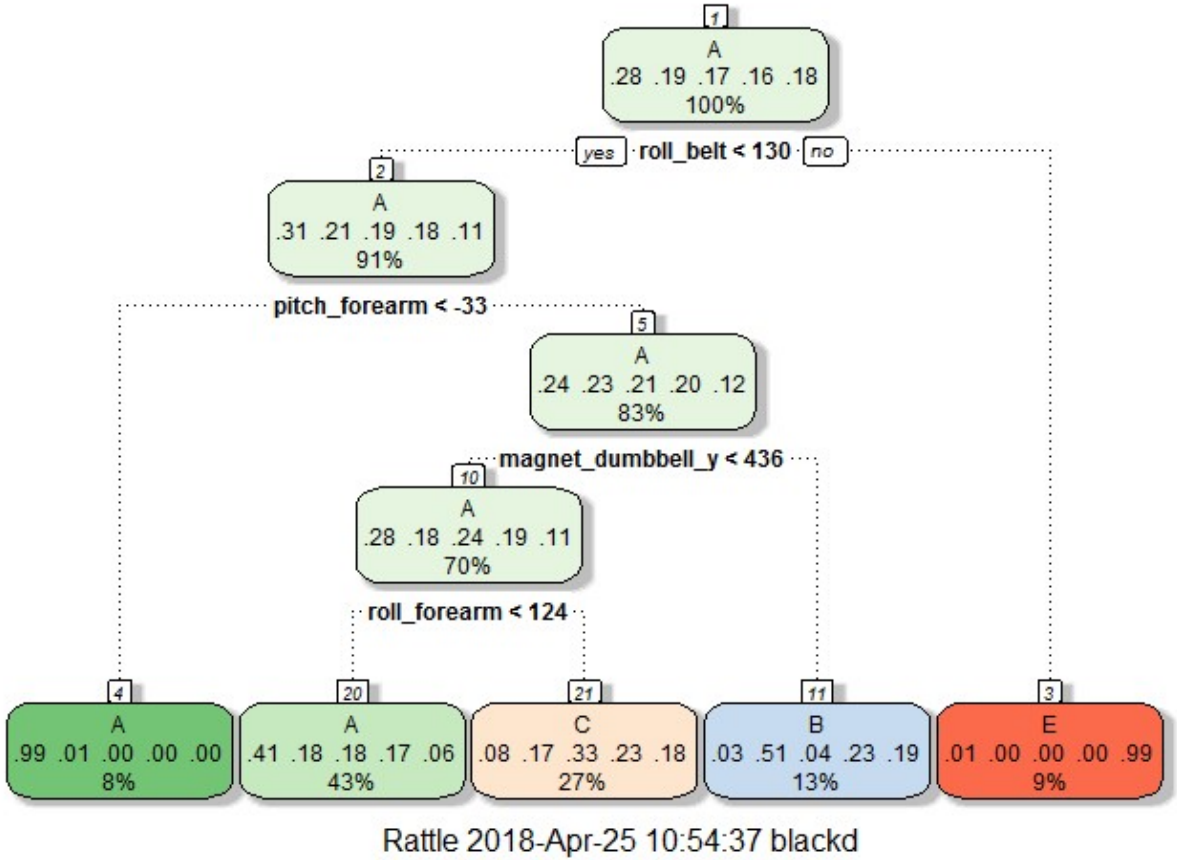
```
inTrain=createDataPartition(CleanTrain$classe, p=.70,list=FALSE)
trainData=CleanTrain[inTrain, ]
testData=CleanTrain[-inTrain, ]
```

Step 4. Building prediction models

Three prediction methods will be evaluated with the cross-validation data set as best fit according to the accuracy. The methods will be: Classification Tree, Random Forest, and Boosting modeling.

a.) Predict with Classification Tree

```
ModelFitDT=train(classe ~., method="rpart",data=trainData)
fancyRpartPlot(ModelFitDT$finalModel)
```




```
predictDT=predict(ModelFitDT,newdata=testData)
confusionMatrix(testData$classe,predictDT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530   35  105    0    4
##          B  486  379  274    0    0
##          C  493   31  502    0    0
##          D  452  164  348    0    0
##          E  168  145  302    0  467
##
## Overall Statistics
##
##                Accuracy : 0.489
##                  95% CI : (0.4762, 0.5019)
##     No Information Rate : 0.5317
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3311
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4890   0.5027   0.3279       NA  0.99151
## Specificity            0.9478   0.8519   0.8797   0.8362  0.88641
## Pos Pred Value         0.9140   0.3327   0.4893       NA  0.43161
## Neg Pred Value         0.6203   0.9210   0.7882       NA  0.99917
## Prevalence             0.5317   0.1281   0.2602   0.0000  0.08003
## Detection Rate         0.2600   0.0644   0.0853   0.0000  0.07935
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638  0.18386
## Balanced Accuracy      0.7184   0.6773   0.6038       NA  0.93896
```

Accuracy is at 49%, which is not a very good fit. The out-of-sample error was 100-49 = 51%

b.). Predict with random forests

```
ModelFitRF=randomForest(classe ~., data=trainData)
predictRF=predict(ModelFitRF,newdata=testData)
confusionMatrix(testData$classe,predictRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    6 1132    1    0    0
##          C    0    6 1019    1    0
##          D    0    0    5  958    1
##          E    0    0    1    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9963
##                  95% CI : (0.9943, 0.9977)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9953
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9947   0.9932   0.9979   0.9991
## Specificity            1.0000   0.9985   0.9986   0.9988   0.9996
## Pos Pred Value         1.0000   0.9939   0.9932   0.9938   0.9982
## Neg Pred Value         0.9986   0.9987   0.9986   0.9996   0.9998
## Prevalence             0.2855   0.1934   0.1743   0.1631   0.1837
## Detection Rate         0.2845   0.1924   0.1732   0.1628   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9982   0.9966   0.9959   0.9983   0.9993
```

Accuracy increases to 99.6%, much better than a classification prediction method. If the predictors are highly correlated the random forest method may be able to decouple some of that through selecting subsets of trees. The out-of sample error was 100 - 99.6 = .04%

c.). Predict with generalized boosting

```
ModelFitGBM=train(classe ~., method="gbm",data=trainData,verbose = FALSE)
predictGBM=predict(ModelFitGBM,newdata=testData)
confusionMatrix(testData$classe,predictGBM)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1651   17    3    2    1
##          B   42 1075   21    0    1
##          C    0   34  977   12    3
##          D    0    5   25  926    8
##          E    1   12   11   17 1041
##
## Overall Statistics
##
##                Accuracy : 0.9635
##                  95% CI : (0.9584, 0.9681)
##     No Information Rate : 0.2879
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9538
##  Mcnemar's Test P-Value : 1.878e-06
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9746   0.9405   0.9421   0.9676   0.9877
## Specificity            0.9945   0.9865   0.9899   0.9923   0.9915
## Pos Pred Value         0.9863   0.9438   0.9522   0.9606   0.9621
## Neg Pred Value         0.9898   0.9857   0.9877   0.9937   0.9973
## Prevalence             0.2879   0.1942   0.1762   0.1626   0.1791
## Detection Rate         0.2805   0.1827   0.1660   0.1573   0.1769
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9846   0.9635   0.9660   0.9799   0.9896
```

Though the accuracy was good at 96.4%, the random forest model was a better fit. The out-of-sample error was 100 - 96.4 = 3.6%

Step 6. Final prediction using the random forest model

```
finalPrediction=predict(ModelFitRF,testing)
finalPrediction
```

```
##  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
```