Course project

A. W. Rosen

12 July 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. 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).

Setting the Working Directory, cleaning the Global Enviroment and loading packages

```r
setwd("/Users/Andreas/Desktop/Data Science/practical machine learning")
rm(list=ls())
library(caret);library(rpart);library(rpart.plot);library(rattle);library(randomForest)
```

```
## Warning: package 'caret' was built under R version 3.2.5

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'rpart.plot' was built under R version 3.2.5

## 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.

## 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
```

Getting and cleaning the data

```
pml.training<-read.csv("pml-training.csv", sep = ",", na.strings = c("", "NA"))
pml.testing<-read.csv("pml-testing.csv", sep = ",", na.strings = c("", "NA"))
```

Remove the first 7 variables since they are either non-numerical or related to a time-series

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

Remove the columens with NAs

```
mostlyNA <- sapply(training, function(x) mean(is.na(x))) > 0.95
training <- training[, mostlyNA==F]
```

Divide the training set into two as cross validation, I choose to split it in 75% for training and 25% for testing.

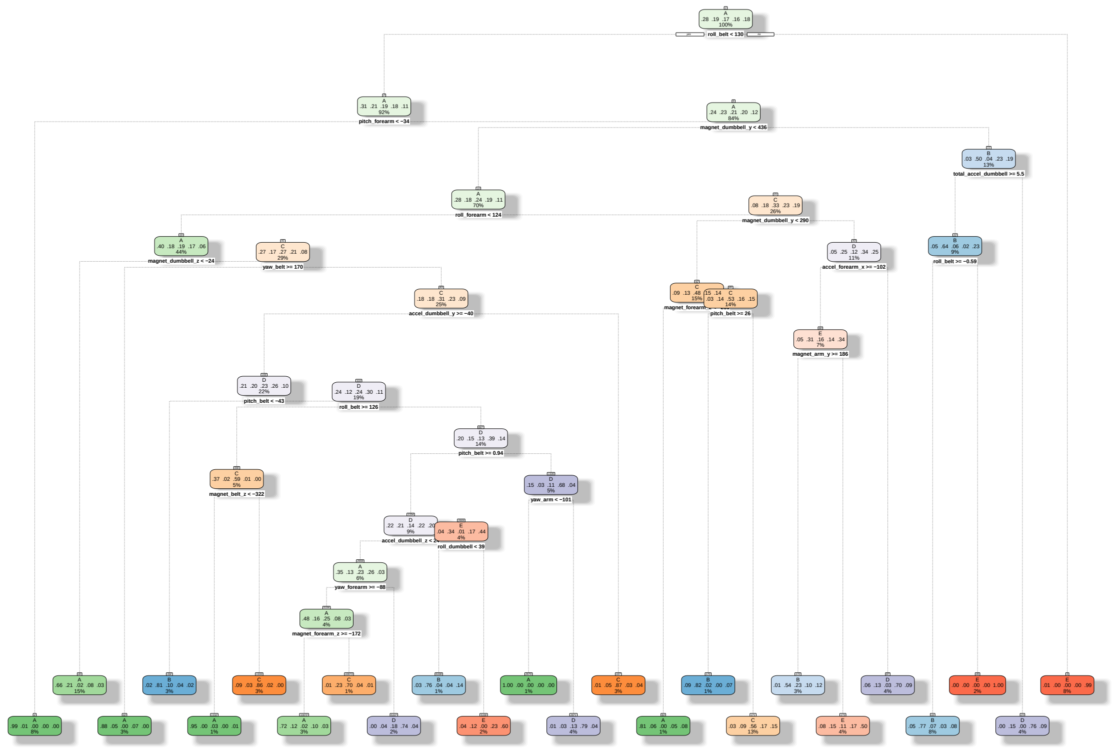
```
set.seed(1337)
inTrain<-createDataPartition(y=training$class, p=0.75,list=F)
training<-training[inTrain,]
testing<-training[-inTrain,]
```

Building the model

We start off with creating a model using a desciontree with rpart with the `training` dataset and using it with the `testing` dataset, which is not the final testing set:

```
modFit1 <- rpart(classe ~ ., data=training, method="class")
fancyRpartPlot(modFit1)
```

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



Rattle 2016−Jul−15 15:48:13 Andreas

```r
predictions1 <- predict(modFit1, testing, type = "class")
confusionMatrix(predictions1,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 999 144  14  74  23
##          B  20 414  52  25  53
##          C  22  72 513  83  79
##          D   7  38  37 366  31
##          E  19  33  23  51 484
##
## Overall Statistics
##
##                Accuracy : 0.7552
##                  95% CI : (0.7409, 0.769)
##     No Information Rate : 0.2903
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6877
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9363   0.5906   0.8028  0.61102   0.7224
## Specificity            0.9023   0.9496   0.9157  0.96328   0.9581
## Pos Pred Value         0.7967   0.7340   0.6671  0.76409   0.7934
## Neg Pred Value         0.9719   0.9078   0.9567  0.92712   0.9393
## Prevalence             0.2903   0.1907   0.1738  0.16295   0.1823
## Detection Rate         0.2718   0.1126   0.1396  0.09956   0.1317
## Detection Prevalence   0.3411   0.1534   0.2092  0.13030   0.1659
## Balanced Accuracy      0.9193   0.7701   0.8593  0.78715   0.8402
```

The accuracy is only at 0.7552, which we should be able to improve with other techniques.

Applying a random forrest model to the `training` dataset and using it on the `testing` data set:

```
modFit2 <- randomForest(training$classe ~ .,data=training)
predictions2 <- predict(modFit2, testing, type = "class")
confusionMatrix(predictions2, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1067    0    0    0    0
##          B    0  701    0    0    0
##          C    0    0  639    0    0
##          D    0    0    0  599    0
##          E    0    0    0    0  670
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.999, 1)
##     No Information Rate : 0.2903
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2903   0.1907   0.1738   0.1629   0.1823
## Detection Rate         0.2903   0.1907   0.1738   0.1629   0.1823
## Detection Prevalence   0.2903   0.1907   0.1738   0.1629   0.1823
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

This results in an empressiv accurarcy of 1(!) which mean that the entire dataset was predicted correctly.

Applying a generalized boosted regression model to the `training`dataset and using it on the `testing` data set:

```r
modFit3 <- train(training$classe ~ ., method="gbm",data=training,verbose=FALSE)
```

```
## Loading required package: gbm

## Loading required package: survival

## Warning: package 'survival' was built under R version 3.2.5

##
## 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

## Loading required package: plyr

## Warning: package 'plyr' was built under R version 3.2.5
```

```r
predictions3 <- predict(modFit3, testing)
confusionMatrix(predictions3, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1055   15    0    0    1
##          B    8  671   13    1    7
##          C    3   14  619   17    7
##          D    0    1    7  577    5
##          E    1    0    0    4  650
##
## Overall Statistics
##
##                Accuracy : 0.9717
##                  95% CI : (0.9658, 0.9768)
##     No Information Rate : 0.2903
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9642
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9888   0.9572   0.9687   0.9633   0.9701
## Specificity            0.9939   0.9903   0.9865   0.9958   0.9983
## Pos Pred Value         0.9851   0.9586   0.9379   0.9780   0.9924
## Neg Pred Value         0.9954   0.9899   0.9934   0.9929   0.9934
## Prevalence             0.2903   0.1907   0.1738   0.1629   0.1823
## Detection Rate         0.2870   0.1825   0.1684   0.1570   0.1768
## Detection Prevalence   0.2913   0.1904   0.1795   0.1605   0.1782
## Balanced Accuracy      0.9913   0.9737   0.9776   0.9795   0.9842
```

Here we get an accuracy of 0.9747 which is also good, but not as impressive as the random forrest model.

Due to the "perfect" accuracy from the `modFit2` with the random forrest model, I choose not to try any model ensambling, since I'm afraid that by including the generalized boosted regression model, might just lead to greater overfitting with little to no benefit in terms of accuracy, compared to the random forrest model alone. Since the accuracy is 1 in my final model, I would expect some overfitting, which might lead to a some "out of sample error". However in the next paragraph where we try to apply the model to the final testset `pml.training` all the 20 predictions turn out right, so the "out of sample error" doesn't appear to bad, even tough it's a small test set.

Predict the 20 cases for the quiz in `pml.training`

```r
predictions4 <- predict(modFit2, pml.testing, type = "class")
predictions4
```

```
##  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
```