Practical Machine Learning Course Project

Practical Machine Learning Course Project

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).

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.

First, lets load our libraries and load our data

```
library(caret)
```

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

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

```
library(ggplot2)
library(rpart)
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## 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

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

```
training <- read.csv('/Users/daniel/Desktop/Practical Machine Learning Course Project/training data.csv
testing <- read.csv('/Users/daniel/Desktop/Practical Machine Learning Course Project/test data.csv')
```

Cleaning Data

Lets Keep the columns that have no NAs and remove the rest

```
training <- training[,colSums(is.na(training))==0]
```

Lets also remove the first seven columns, as they just are for identification purposes

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

Finally, lets remove variables that have a majority of values near zero

```
nz <- nearZeroVar(training)
training <- training[, -nz]
```

Now we can start training our prediction models, lets split the data into training and test parts

```
trainingset <- createDataPartition(training$classe, p=.7, list = FALSE)
train <- training[trainingset, ]
test <- training[-trainingset, ]

##Confirming number of rows for correct partitioning, and number of columns for same column count
c(nrow(train), ncol(train))
```

```
## [1] 13737    53
```

```
c(nrow(test), ncol(test))
```

```
## [1] 5885   53
```

Decision Trees

Our first model will be with Decision Trees

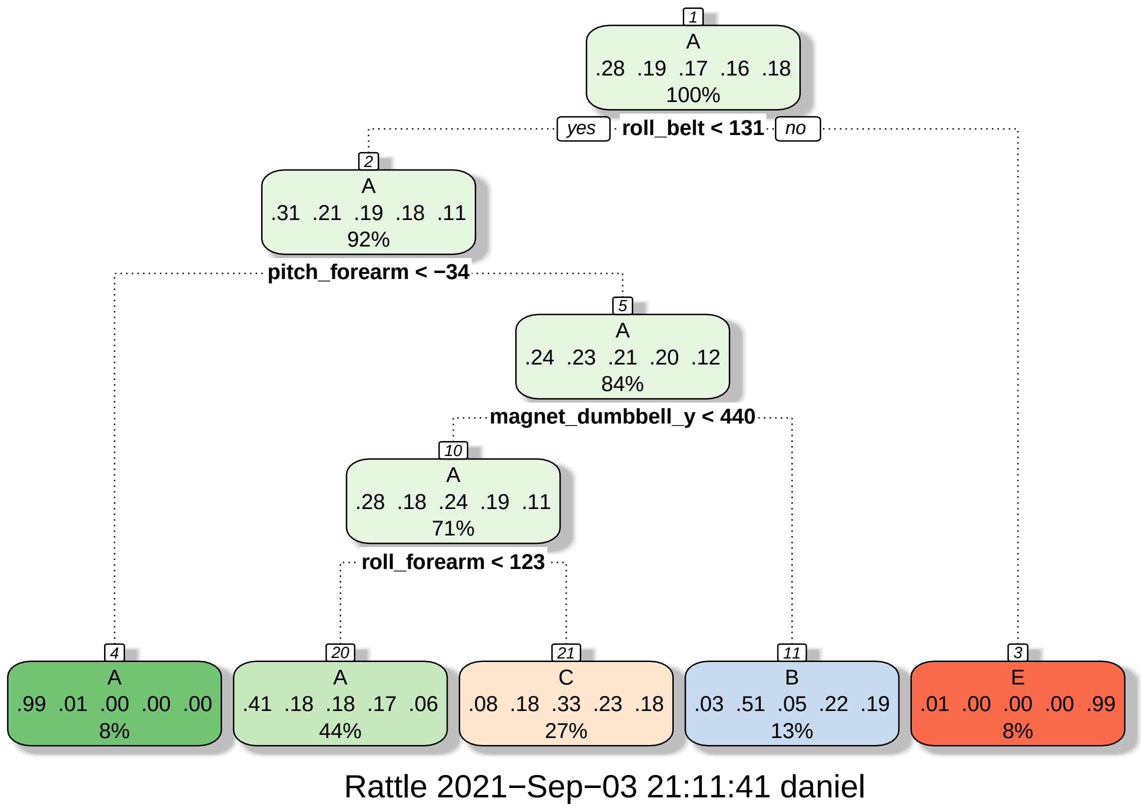
```
set.seed(15000)
modFit1 <- train(classe~., data=train, method="rpart")

print(modFit1$finalModel)
```

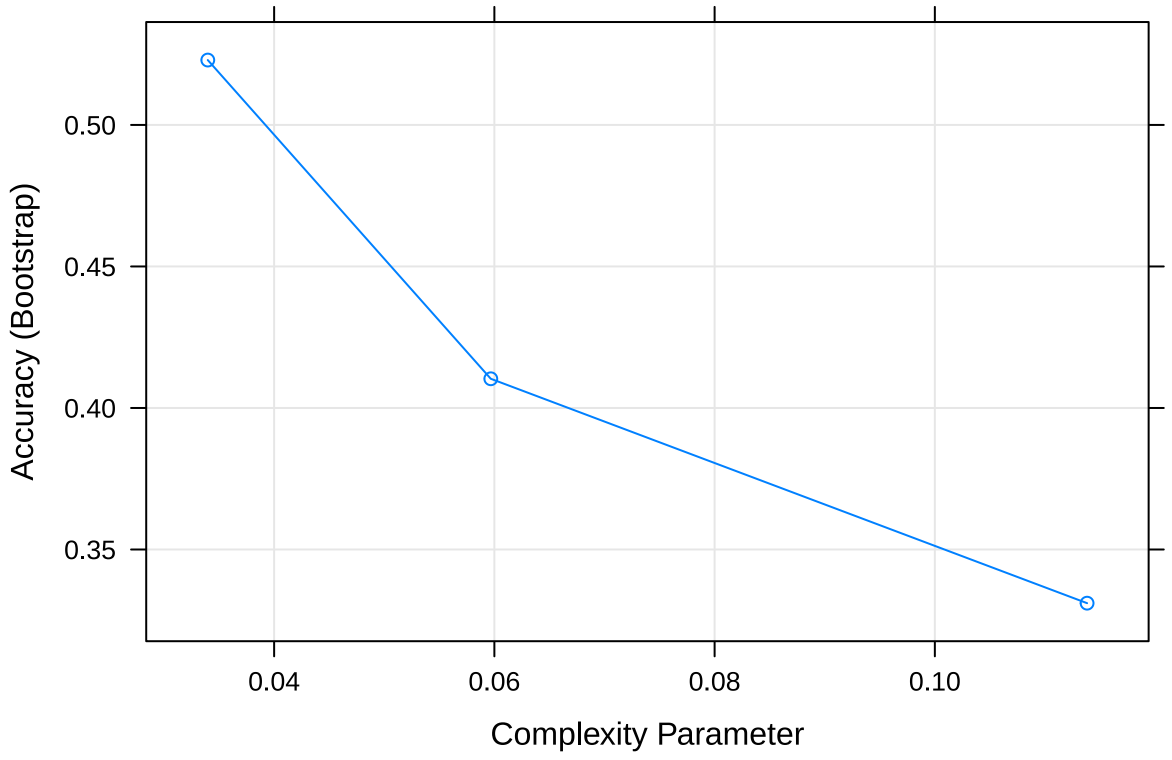
```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 12594 8700 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1094    6 A (0.99 0.0055 0 0 0) *
##      5) pitch_forearm>=-33.95 11500 8694 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 439.5 9730 6983 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 122.5 6036 3579 A (0.41 0.18 0.18 0.17 0.061) *
##         21) roll_forearm>=122.5 3694 2484 C (0.079 0.18 0.33 0.23 0.18) *
##       11) magnet_dumbbell_y>=439.5 1770  871 B (0.033 0.51 0.046 0.22 0.19) *
##    3) roll_belt>=130.5 1143   12 E (0.01 0 0 0 0.99) *
```

Plotting a Dendogram and the results, we get

```
fancyRpartPlot(modFit1$finalModel)
```



```
plot(modFit1)
```



We have a model, lets compare results to the test data

```
predict1 <- predict(modFit1, newdata=test)
confusionMatrix(predict1, as.factor(test$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  459  482  430  155
##          B   22  387   26  183  141
##          C  120  293  518  351  286
##          D    0    0    0    0    0
##          E    2    0    0    0  500
##
## Overall Statistics
##
##                Accuracy : 0.4987
##                  95% CI : (0.4859, 0.5116)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3449
```

```
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140  0.33977  0.50487   0.0000  0.46211
## Specificity            0.6376  0.92162  0.78391   1.0000  0.99958
## Pos Pred Value         0.5007  0.50988  0.33036      NaN  0.99602
## Neg Pred Value         0.9491  0.85330  0.88233   0.8362  0.89188
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2600  0.06576  0.08802   0.0000  0.08496
## Detection Prevalence   0.5193  0.12897  0.26644   0.0000  0.08530
## Balanced Accuracy      0.7758  0.63069  0.64439   0.5000  0.73085
```

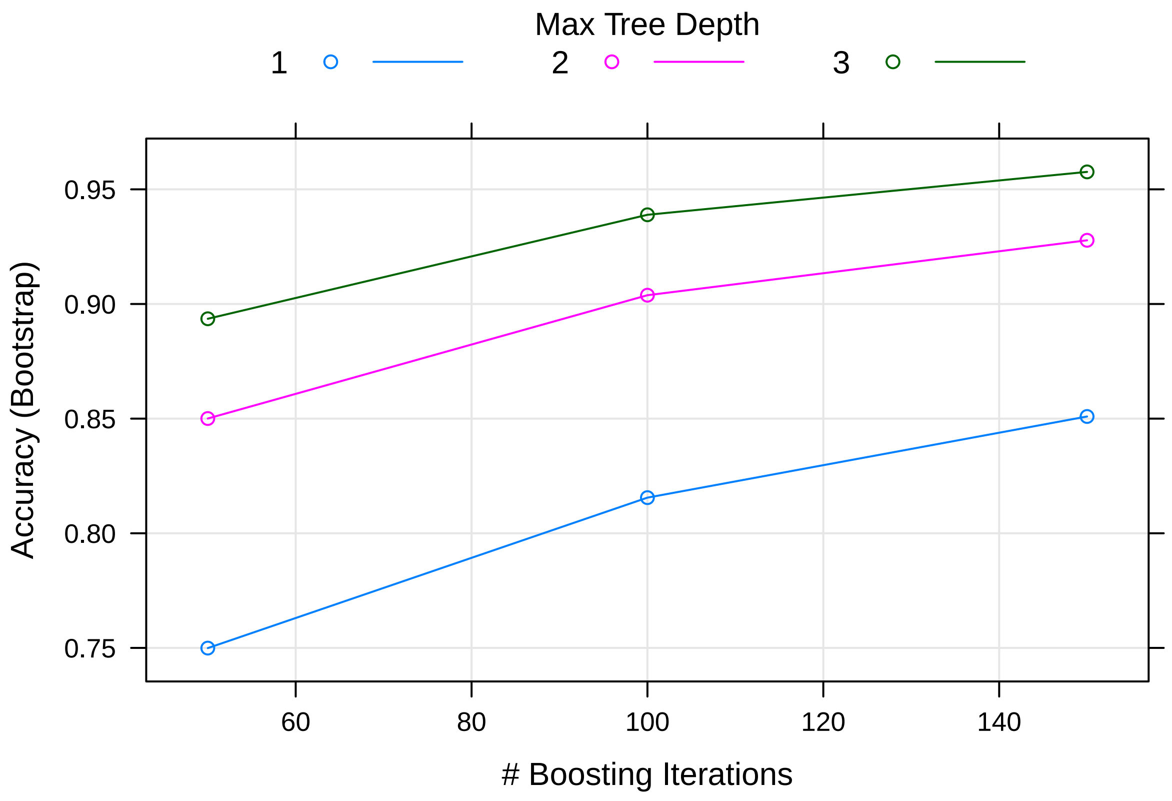
Unfortunately, results show the Accuracy as only .4802 which is low. Time to try a different algorithm

Boosting

```
set.seed(15000)
modFit2 <- train(classe~., data=train, method="gbm")
```

Plotting our accuracy, we get

```
plot(modFit2)
```



lets compare results to the test dataset

```
##Comparing results to the test data for Boosting
predict2 <- predict(modFit2, newdata=test)
confusionMatrix(predict2, as.factor(test$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1641   37    0    0    1
##          B   25 1077   46    5   10
##          C    6   24  971   31   10
##          D    1    1    8  919   15
##          E    1    0    1    9 1046
##
## Overall Statistics
##
##                Accuracy : 0.9607
##                  95% CI : (0.9555, 0.9656)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9503
##
##  Mcnemar's Test P-Value : 1.517e-07
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9803   0.9456   0.9464   0.9533   0.9667
## Specificity            0.9910   0.9819   0.9854   0.9949   0.9977
## Pos Pred Value         0.9774   0.9261   0.9319   0.9735   0.9896
## Neg Pred Value         0.9922   0.9869   0.9886   0.9909   0.9925
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2788   0.1830   0.1650   0.1562   0.1777
## Detection Prevalence   0.2853   0.1976   0.1771   0.1604   0.1796
## Balanced Accuracy      0.9856   0.9637   0.9659   0.9741   0.9822
```

Results show the Accuracy is much higher at .963, so we will use this algorithm for our testing dataset.

Results

From our two models, we can see that Boosting has much higher accuracy than Random Forests. The results are: Boosting has .963 accuracy and .037 out of sample. Decision Tree has .4802 accuracy and .5198 out of sample.

Prediction

```
predict(modFit2, newdata=testing)
```

```
##  [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
```