Machine Learning Project

Brian Canute

Sunday, September 06, 2015

Abstract

A machine learning strategy will be applied to a data set of human movement data predicting an expert's judgement of the quality of a gymnastic skill. The predictive model will then be applied to a second test data set to see how well it performs on data set it has never seen before. Various measures of outcome accuracy and parameter variance will be used along the way to assist in tuning the model.

Machine Learning is a realtively young, technology that has emerged from the traditional area of statistics aided greatly by the recent computer revolution. Canonical concepts and vocabulary are still being defined but it is generally agreed that a machine learning model succeeds through its ability to predict rather than explain. The reputation of the area has also been extended via a series of Kaggle-style competitions, characterised by very creative, statistically minded computer scientists stretching their computers to the limit to cross validate their predictions against an unknown test set, in exchange for some rather spectacular prize money. Stripped of the big bucks, that is essentially what this course assignment is about.

This paper will therefore treat the machine metrics as a black box; 159 columns of mainly dynamic spatial data that predicts column 160.

The outcome variable in column 160 relates to six young health participants who were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl at five different qualitative levels of human movement expertise:
(Class A): exactly according to the specification,
(Class B): throwing the elbows to the front,
(Class C): lifting the dumbbell only halfway,
(Class D): lowering the dumbbell only halfway, and
(Class E): throwing the hips to the front.

More details can be found at: http://groupware.les.inf.puc-rio.br/har#ixzz3ktRql3d3

```
knitr::opts_chunk$set(warning=FALSE,message=FALSE)
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(e1071)
library(randomForest)
library(plyr)
set.seed(1)
setwd("C:/Users/Brian/Documents/Machine Learning/")
testing <- read.csv(file="pml-testing.csv",na.strings=c("NA","#DIV/0!",""))
training <- read.csv(file="pml-training.csv",na.strings=c("NA","#DIV/0!",""))
```

Tidying the Data

Columns with missing data were excluded as were variables that showed insufficient variance to realistically contribute to the predictive power of the model.

```r
#check that all the column names match
colnames(testing)[160] <- "classe"
all.equal(names(training[,-160]),names(testing[,-160]))
```

```
## [1] TRUE
```

```r
#check the testing set for na columns
col_na_sum = apply(testing,2,function(x) {sum(is.na(x))})
unique(col_na_sum) # very neat; na's either in every row or in none
```

```
## [1]  0 20
```

```r
testing = testing[,which(col_na_sum == 0)]# simply trim out the na columns
training = training[,which(col_na_sum == 0)]# from both datasets
#trim out some time oriented columns that are not useful in this context
testing <- testing[,-c(1:7)]
training <- training[,-c(1:7)]
#check for variances that are too low for useful prediction and trim them out.
nzv <- nearZeroVar(training, saveMetrics=TRUE)
training <- training[,nzv$nzv==FALSE]
nzv <- nearZeroVar(testing, saveMetrics=TRUE)
testing <- testing[,nzv$nzv==FALSE]
dim(testing)
```

```
## [1] 20 53
```

```r
dim(training)
```

```
## [1] 19622    53
```

Split the Training Set

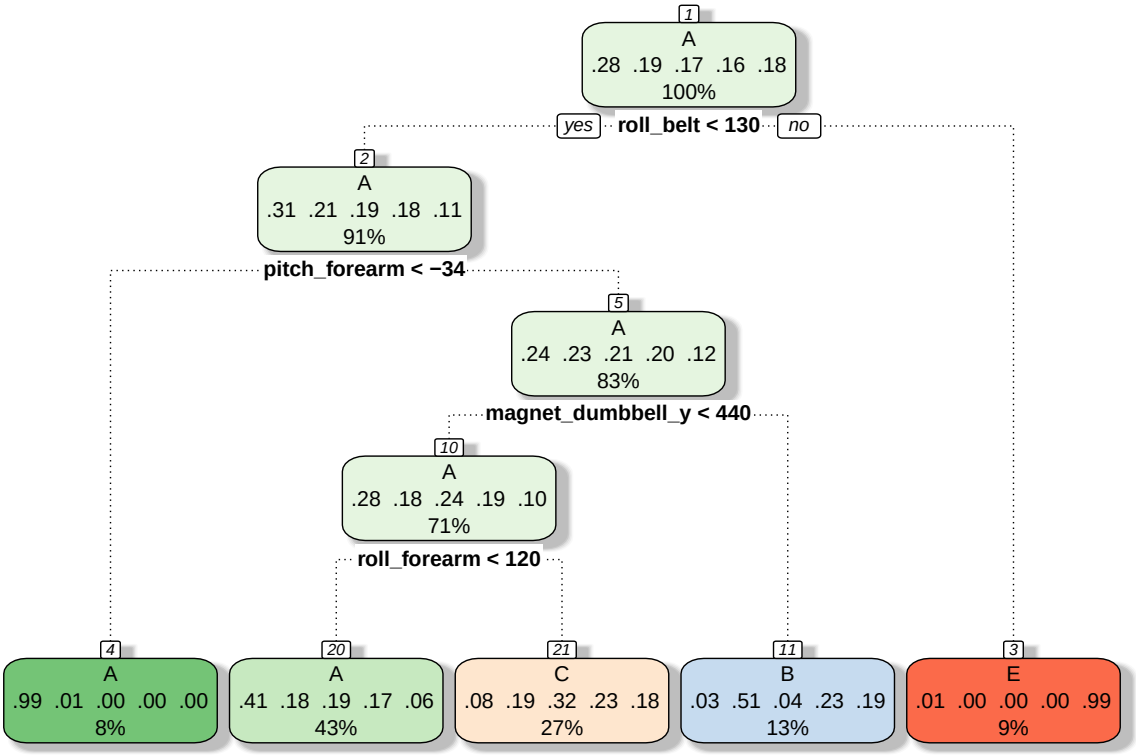
The conventional approach is to tune the model using a large training data set, cross validate the model using a smaller validation data set and finally, to submit the best model to the test data set, which in this case contains 20 records with hidden outcomes. The test outcomes produced by our selected model will be submitted to a machine to test the accuracy of our model. Our validation data was created by excising 30% from the training data.

```r
in.training <- createDataPartition(training$classe, p=0.70, list=F)
learning <- training[in.training, ]
validation <- training[-in.training, ]
```

Cart(rPart) Tree Model

The first model attempted involves a traditional tree structure traditionally used to predict categorical outcomes, like our Classe variable.

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa  Accuracy SD  Kappa SD
##   0.0366  0.515     0.367  0.0222       0.0354
##   0.0595  0.434     0.237  0.0655       0.1082
##   0.1179  0.346     0.091  0.0369       0.0580
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0366.
```



Rattle 2015−Sep−25 19:18:58 Brian

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1510   27  132    0    5
##          B  463  389  287    0    0
##          C  462   36  528    0    0
##          D  427  163  374    0    0
##          E  179  141  299    0  463
```

```
## Accuracy
## 0.491079
```

The cross validation accuracy of this model when applied to the validation set is quite disappointing. Fortunately, there have been a series of recent enhancements to the tree model.

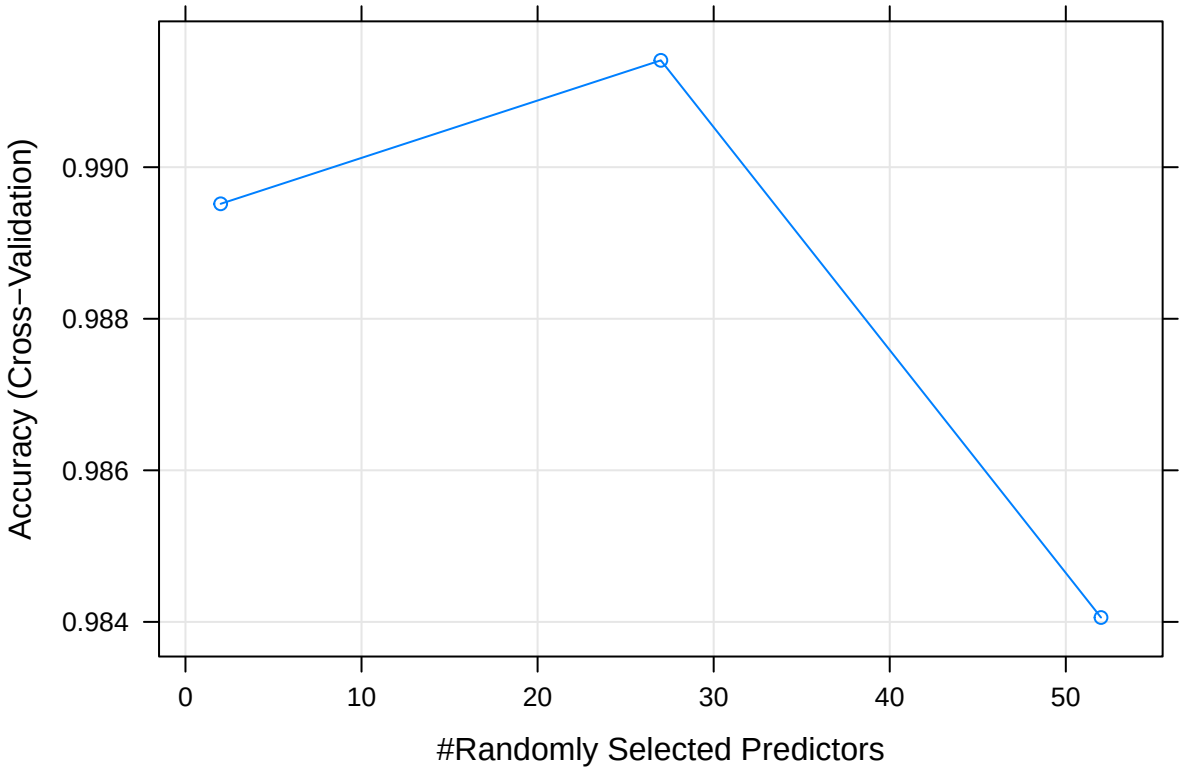
Random Forest Model

The Random Forest is considered by Developers such as Hastie and Tibshirani to represent a good value in terms of accuracy, simplicity and computing overhead.

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10990, 10989, 10989, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.990     0.987  0.00273      0.00345
##   27    0.991     0.989  0.00142      0.00180
##   52    0.984     0.980  0.00137      0.00173
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    0    2    0    1
##          B    7 1128    3    1    0
##          C    0    4 1019    3    0
##          D    0    0    5  957    2
##          E    0    1    2    1 1078
```

```
##  Accuracy
## 0.9945624
```



The cross validation accuracy is clearly superior to the first tree model.

Boosting Model

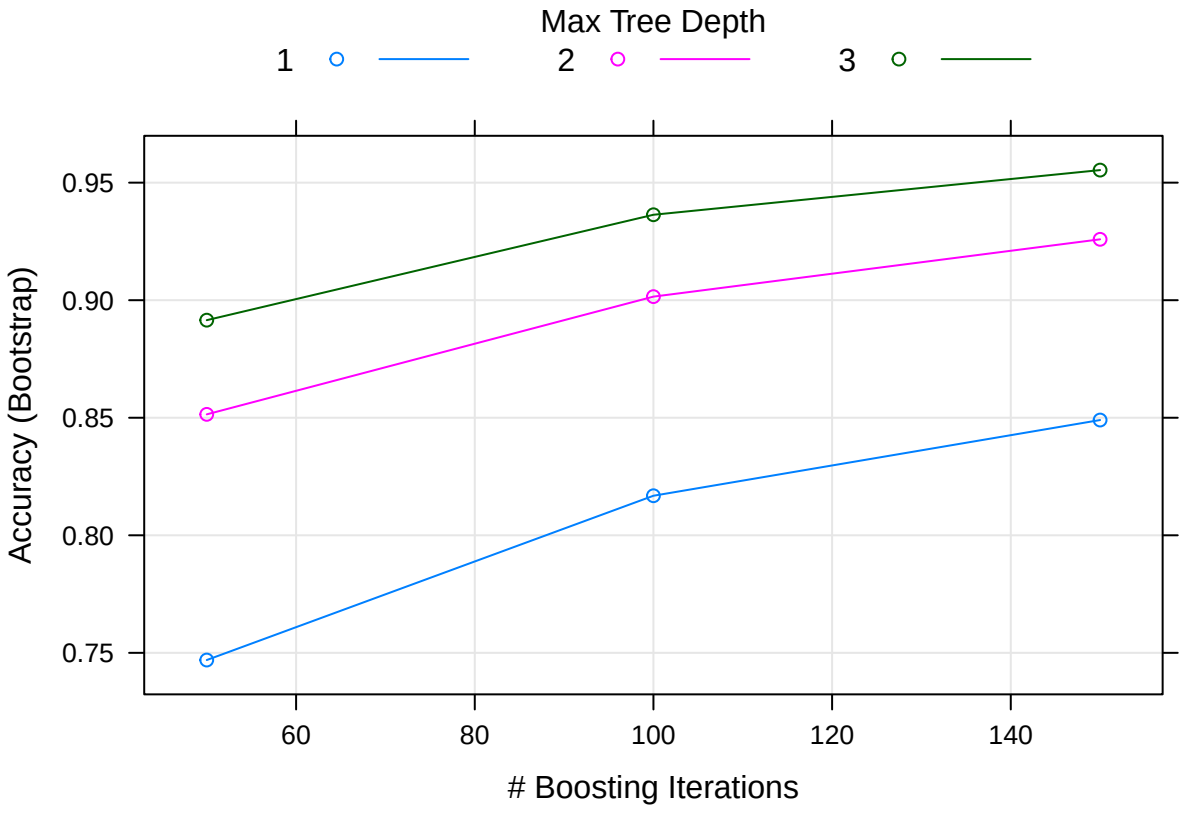
Boosting aims to improve upon the basic ensemble approach of the Random Forest. Such enhancements in accuracy are based upon the artful manipulation of several tuning parameters, the patient slow tweaking of the model for very modest gains and an enormous amount of computing power, including parallel processing. Having attempted this model, I have concluded that it is currently beyond my limited abilities. However, here is the best model I was able to produce on my home computer.

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa  Accuracy SD  Kappa SD
##   1                   50      0.747     0.679  0.00823      0.01033
##   1                  100      0.817     0.768  0.00538      0.00669
##   1                  150      0.849     0.809  0.00536      0.00671
##   2                   50      0.851     0.812  0.00568      0.00714
```

```
##   2                  100      0.902     0.875  0.00472      0.00591
##   2                  150      0.926     0.906  0.00491      0.00617
##   3                   50      0.892     0.863  0.00605      0.00760
##   3                  100      0.936     0.919  0.00474      0.00596
##   3                  150      0.955     0.943  0.00338      0.00425
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1637   22    8    4    3
##          B   39 1074   24    1    1
##          C    0   30  986    8    2
##          D    2    2   24  927    9
##          E    1   11   12    9 1049
```

```
##  Accuracy
## 0.9639762
```



As can be seen, the accuracy while impressive is not as good as the results of the much simpler Random Forest. My attempts to use some of the tuning parameters only met with frustration and a plethora of error

messages, for which documentation was in short supply. Such is the chaotic outcome when you offer hackers $1meg to produce models that subsequently prove to be of limited commercial value; just like we do not all drive F1 Racing Cars.

Submit Predictions on the Test Set

As a result, I have contented myself with the Random Forest model, which I submitted to the machine. The reults appeared to have passed muster.

```
##  [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
```