A General Boosting Regression Model (GMB) to Predict a Weightlifting Exercise

By: Wahsabii Neanderthal

Purpose. To predict the manner in which subjects did a barbell curl exercise.

Data.
The data used were from six subjects who participated in a dumbbell lifting exercise. The subjects used the dumbbell five different ways. The five ways, as described in the study, were "exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. The classe variable is the response variable to predict.

```
# load library used to model data
library(caret)

# Read data file into memory
filename <- "pml-training.csv"  # Assumes user has preloaded data in working directory
pmlData <- read.csv(filename, header=TRUE, na.strings=c("NA","#DIV/0!",""))
pmlData <- pmlData[,complete.cases(t(pmlData))] # eliminate incomplete columns
pmlData <- pmlData[-c(1:7)] # remove the first seven admin columns

# Show the structure of the data
str(pmlData)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ roll_belt            : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt           : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt             : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt     : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x         : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y         : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z         : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x         : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y         : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z         : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x        : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y        : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z        : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm             : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm            : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm              : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm      : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x          : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y          : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z          : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x          : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y          : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z          : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x         : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y         : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z         : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell        : num  13.1 13.1 12.9 13.4 13.4 ...
```

```
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

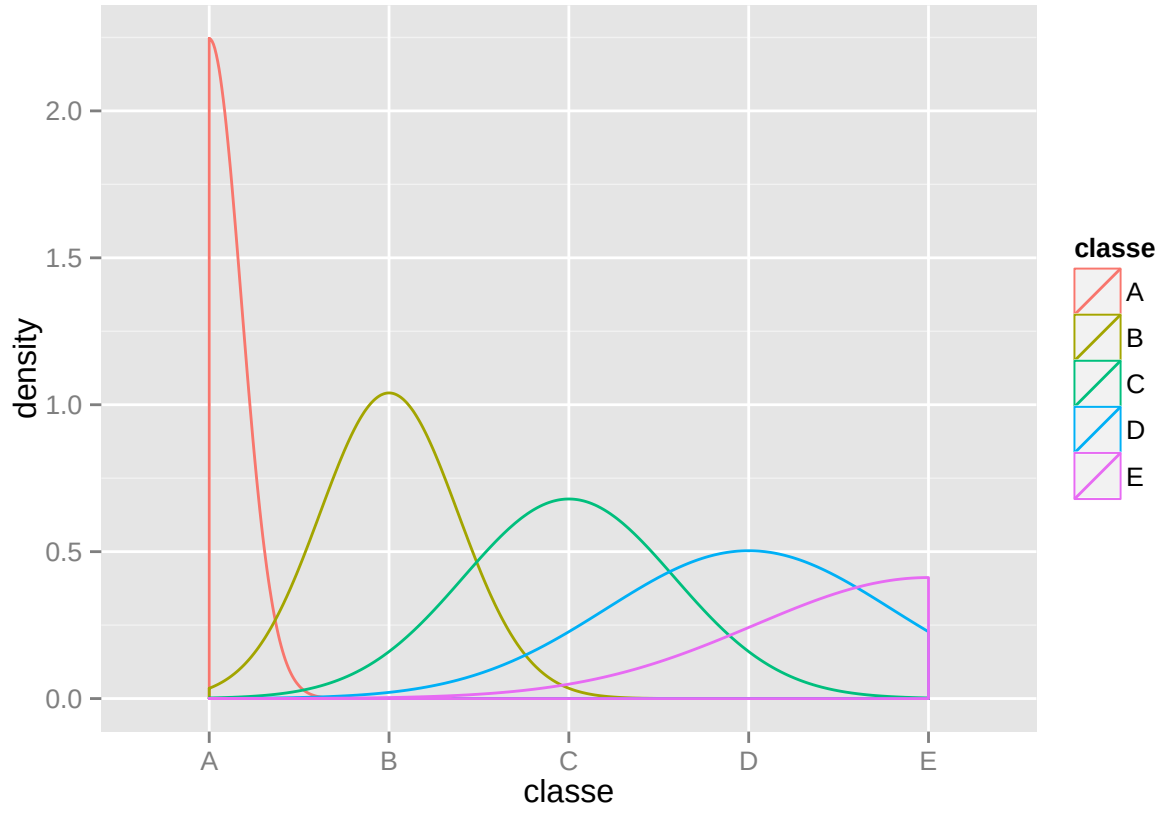
To create the model, the data was partitioned in two segments with approximately 60% of the data used to tune the model and the remaining 40% used to validate the model.

```
# Partition the data
set.seed(8675309)
inTrain <- createDataPartition(y=pmlData$classe, p=0.60, list=FALSE)
training <- pmlData[inTrain,]
testing <-  pmlData[-inTrain,]
c(dim(training), dim(testing))
```

```
## [1] 11776    53  7846    53
```

A density plot of the classe data factors are as follows:

```
# Density plot of the classe factor variable
qplot(classe, colour=classe, data=training, geom="density")
```



How the model was built.
The Generalized Boosting Regression (gmb) Model Package in r's caret package was used to model the various sensor readings recorded in pmlData above. The classe variable is the response variable and all the other sensor data were explanatory variables.

How cross validation was used.
Repeated cross validation was used to model pmlData. Three folds and three repeats were used to model the training data.

Expected out of sample error estimate.
Initial hypothesis for the expected out of error rate was 5 to 10% was acceptable.

Why model choices were made. Three folds and three repeats instead of 10 were chosen due to the excessively long computing time to build a model.

```
# WARNING: this took me 12 hours to build this model
# if the gbmFit2 model exist load else build it
if(file.exists("gbmModel.Rda")){load("gbmModel.Rda")} else {

        fitControl <- trainControl(## 3-fold CV
                method = "repeatedcv",
                number = 3,
                repeats = 3)

        gbmGrid <-  expand.grid(interaction.depth = c(1, 5, 9),
                                n.trees = (1:30)*50,
                                shrinkage = 0.1,
                                n.minobsinnode = 20)
```

```
        set.seed(8675309)
        gbmFit2 <- train(classe ~ ., data = training,
                         method = "gbm",
                         trControl = fitControl,
                         verbose = FALSE,
                         tuneGrid = gbmGrid)
}
```

The model exceeded expectations at 99% accuracy. All twenty test questions were predicted with 100% accuracy.

```
# Out of sample prediction
testPred <- predict(gbmFit2,testing)
# Accuracy
confusionMatrix(testPred,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228    4    0    0    0
##          B    3 1513    2    0    0
##          C    0    1 1365    5    0
##          D    0    0    1 1281    3
##          E    1    0    0    0 1439
##
## Overall Statistics
##
##                Accuracy : 0.9975
##                  95% CI : (0.9961, 0.9984)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9967   0.9978   0.9961   0.9979
## Specificity            0.9993   0.9992   0.9991   0.9994   0.9998
## Pos Pred Value         0.9982   0.9967   0.9956   0.9969   0.9993
## Neg Pred Value         0.9993   0.9992   0.9995   0.9992   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1928   0.1740   0.1633   0.1834
## Detection Prevalence   0.2845   0.1935   0.1747   0.1638   0.1835
## Balanced Accuracy      0.9987   0.9980   0.9984   0.9978   0.9989
```