Practical Machine Learning Assignment

Colby Smithmeyer

May 27, 2019

Executive Summary

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: 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). (Taken from http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har#weight_lifting_exercises)

This report will detail the analysis performed to predict, based on a number of variables, the method (A, B, C, D, or E) used to perform the exercise. Specifically this analysis will use Classfication Tree, Random Forest, and Gradient Boosting Models for prediction.

Overall the Random Forest model yieled the most accurated result on the training data set. It had an accuracy of 99.3%, compared to the Classification Tree (50%), and the Gradient Boosting Model (96.8%).

Reading and Cleaning the Data

Here we simply read in the testing and training data sets.

```
## Loading required package: lattice

## Loading required package: ggplot2

## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

## 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

## corrplot 0.84 loaded
```

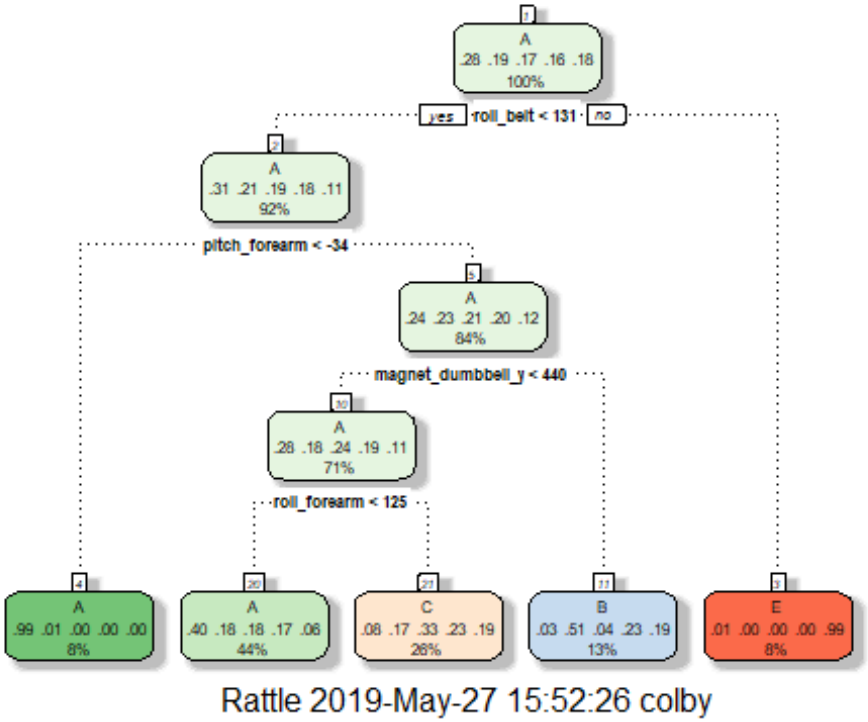
Now we must clean the data sets by removing the columns with blank entries or NA entries of atleast 70%. Additionally we will partition the training set into a training and test set for cross validation of the models.

```
training<-training[,-
(which(colSums(is.na(training)|training=="")/nrow(training)>=.7))]
training<-training[,-c(1:7)]

#Now we perform the same for the test data set
testing<-testing[,-
(which(colSums(is.na(testing)|testing=="")/nrow(testing)>=.7))]
testing<-testing[,-c(1:7)]

#Now we will partition the training set
set.seed(1434)
inTrain1 <- createDataPartition(training$classe, p=0.75, list=FALSE)
train1 <- training[inTrain1,]
test1 <- training[-inTrain1,]
```

Classificaion Trees

Now we will use classification tree method to create a prediction model.

```
trControl <- trainControl(method="cv", number=5)
model_CT <- train(classe~., data=train1, method="rpart", trControl=trControl)

fancyRpartPlot(model_CT$finalModel)
```




```
trainpred <- predict(model_CT,newdata=test1)

confMatCT <- confusionMatrix(test1$classe,trainpred)

confMatCT$table

##           Reference
## Prediction    A    B    C    D    E
##          A 1272   15  105    0    3
##          B  384  315  250    0    0
##          C  388   28  439    0    0
##          D  367  128  309    0    0
##          E  118  126  248    0  409

confMatCT$overall[1]

##  Accuracy
## 0.4965334
```

As you can see the accuracy is on 50%. This will not be a very useful model for predicting.

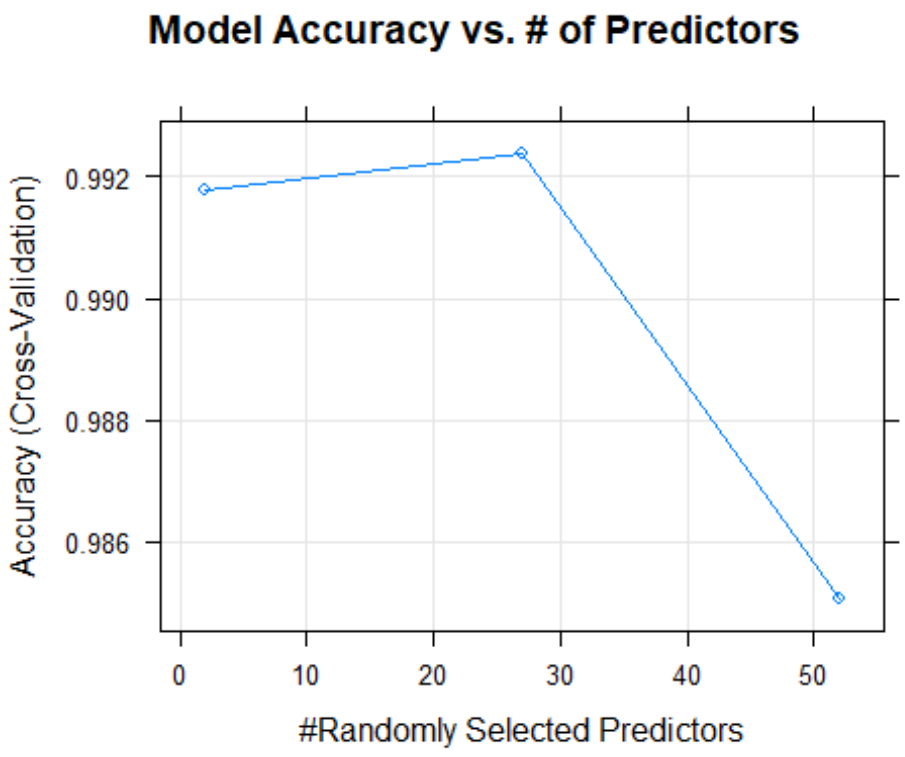
Random Forest

Now we will use the random forest method to create a prediction model.

```
model_RF <- train(classe~., data=train1, method="rf", trControl=trControl,
verbose=FALSE)
print(model_RF)
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11774, 11773, 11776, 11774
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9917791  0.9896001
##   27    0.9923907  0.9903741
##   52    0.9850526  0.9810904
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
plot(model_RF,main="Model Accuracy vs. # of Predictors")
```




```
#As you can see the best model was mtry 2 with an accuracy of 99.3%
```

```r
trainpred <- predict(model_RF,newdata=test1)

confMatRF <- confusionMatrix(test1$classe,trainpred)

# display confusion matrix and model accuracy
confMatRF$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    9  937    3    0    0
##          C    0    7  845    3    0
##          D    0    0   14  788    2
##          E    0    0    2    3  896
```

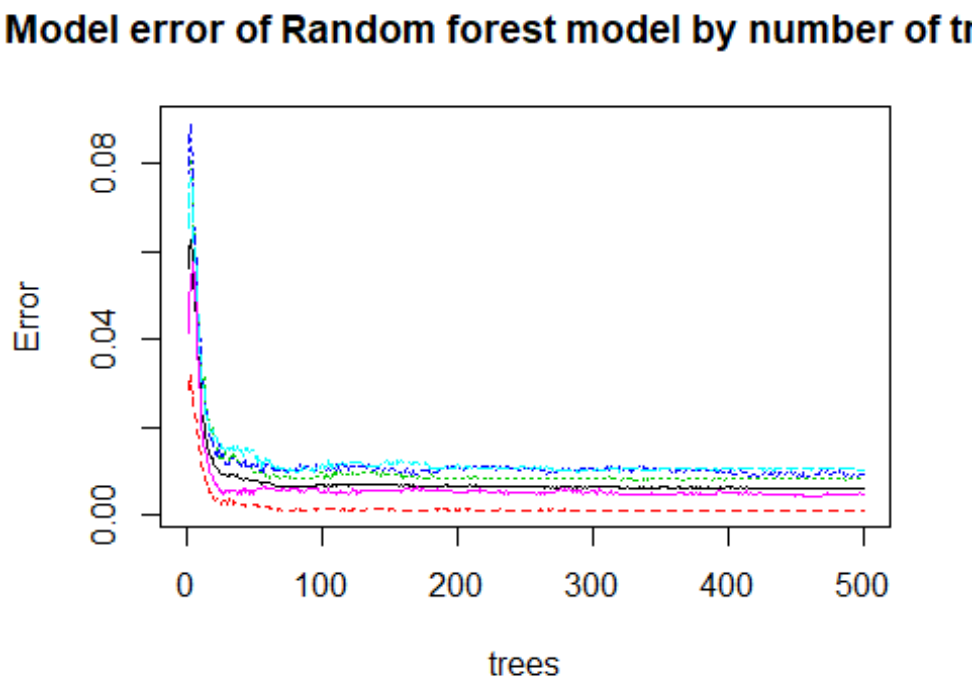
```r
confMatRF$overall[1]
```

```
##  Accuracy
## 0.9912316
```

```r
#Display the variable used in the final model
names(model_RF$finalModel)
```

```
##  [1] "call"            "type"            "predicted"
##  [4] "err.rate"        "confusion"       "votes"
##  [7] "oob.times"       "classes"         "importance"
## [10] "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"
## [16] "y"               "test"            "inbag"
## [19] "xNames"          "problemType"     "tuneValue"
## [22] "obsLevels"       "param"
```

```r
plot(model_RF$finalModel,main="Model error of Random forest model by number
of trees")
```




```
varImp(model_RF)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                       Overall
## roll_belt             100.000
## pitch_forearm          61.412
## yaw_belt               55.149
## pitch_belt             43.991
## magnet_dumbbell_z      43.397
## magnet_dumbbell_y      41.256
## roll_forearm           41.179
## accel_dumbbell_y       21.628
## roll_dumbbell          17.091
## magnet_dumbbell_x      16.410
## accel_forearm_x        15.654
## magnet_belt_z          15.454
## accel_belt_z           13.788
## total_accel_dumbbell   13.702
## magnet_forearm_z       13.248
## magnet_belt_y          12.502
## accel_dumbbell_z       12.081
## gyros_belt_z           11.859
## yaw_arm                10.498
## magnet_belt_x           9.786
```

As you can see the accuracy of the final model is 99.3%. This will be a useful model in predicting the classe variable.

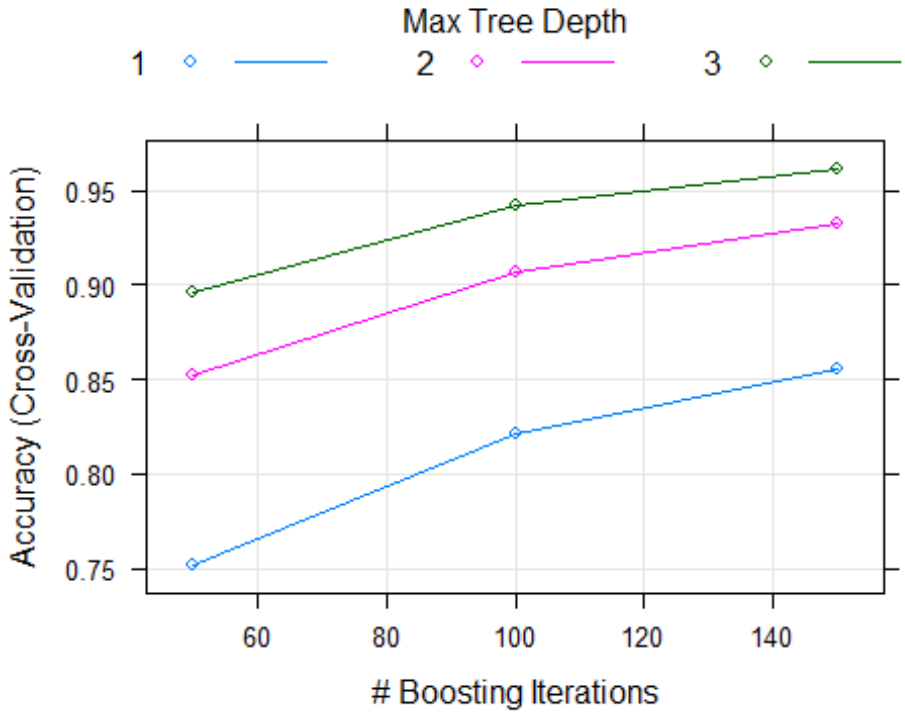
Gradient Boosting Method

Finally we will use the gradient boosting method to build the final model.

```
model_GBM <- train(classe~., data=train1, method="gbm", trControl=trControl,
verbose=FALSE)
print(model_GBM)
```

```
## Stochastic Gradient Boosting
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11775, 11773, 11774, 11775
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7515310  0.6850385
##   1                  100      0.8215801  0.7742323
##   1                  150      0.8557556  0.8174883
##   2                   50      0.8518832  0.8123566
##   2                  100      0.9069177  0.8822315
##   2                  150      0.9320566  0.9140233
##   3                   50      0.8957063  0.8679557
##   3                  100      0.9421119  0.9267472
##   3                  150      0.9616118  0.9514281
##
## 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.
```

```
# As you can see the best model yields an accuracy of 96%.
plot(model_GBM)
```




```
trainpred <- predict(model_GBM,newdata=test1)

confMatGBM <- confusionMatrix(test1$classe,trainpred)
confMatGBM$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1367   16    7    3    2
##          B   38  880   29    0    2
##          C    0   20  826    7    2
##          D    0    4   27  765    8
##          E    0   10   10   15  866
```

```
confMatGBM$overall[1]
```

```
## Accuracy
## 0.959217
```

As you can see this model has an accuracy of 96.8%, slightly worse than the random forest method.

Conclusion

We will use each model to predict the final test set.

```
FinalTestPred_CT <- predict(model_CT,newdata=testing)
FinalTestPred_RF <- predict(model_RF,newdata=testing)
```

```
FinalTestPred_GBM <- predict(model_GBM,newdata=testing)

FinalTestPred_CT

##  [1] C A C A A C C A A A C C C A C A A A A C
## Levels: A B C D E

FinalTestPred_RF

##  [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

FinalTestPred_GBM

##  [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
```

Since the Random Forest model is the most accurate we would want to use its prediction, but it is interesting to note that the Random Forest and Gradient Boosting Method yielded the same prediction. Given the low accuracy of the Classification Tree method we should not use it to predict. If we had the classe variables of the test set we could test for the out of sample error of each model.