Course #8 Project

David Stanley

November 10, 2019

Introduction:

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, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and predict the manner in which they did the exercise. This is the "classe" variable in the training set described by the following classes A through E: 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).

As requested, the data was made available by the following website: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

and related publication.

Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6.

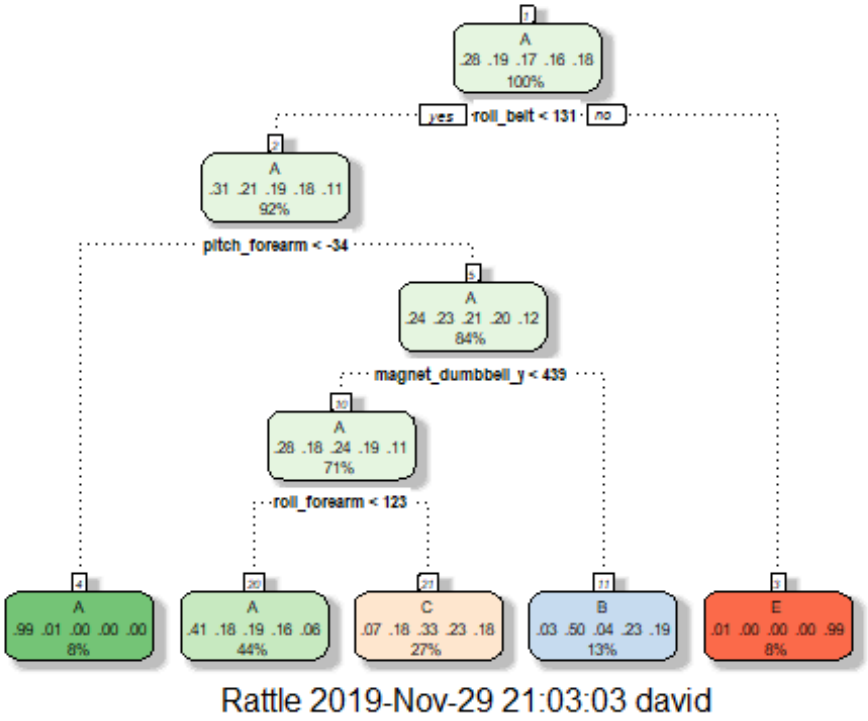
Loading and cleaning the data:

```
#download files to local computer and set up working directory for loading in the data:

setwd("D:/JHU Data Science Program/Course 8 Practical Machine Learning/Course Project")
Data_Train=read.csv("pml-training.csv")
Data_Test=read.csv("pml-testing.csv")

#load in necessary packages related to the machine learning models:

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rattle)
```

```
## 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.

# check for missing values and NA values and based on this information, clean
the data set for better manipulation
# criteria for clean data in this exercise was to remove columns having
majority (90%) NA or blank values

remove_info_train=which(colSums(is.na(Data_Train)|
Data_Train=="")>0.9*dim(Data_Train)[1])
Clean_Data_Train=Data_Train[,-remove_info_train]

# remove first 7 columns because they just contain info on the people who did
the test, timestamps, etc which isn't relevant to the study

Clean_Data_Train= Clean_Data_Train[,-c(1:7)]

#Same procedure for cleaning the Test Data Set

remove_info_test=which(colSums(is.na(Data_Test)|
Data_Test=="")>0.9*dim(Data_Test)[1])
Clean_Data_Test=Data_Test[,-remove_info_test]
Clean_Data_Test= Clean_Data_Test[,-c(1:7)]
```

Applying 3 types of modeling from class: Classification Tree, Gradient Boosting, and Random Forest:

```
# Set up the partitions for the Training and Testing Data sets to be used in
the three models:

set.seed(3344)
inTrain=createDataPartition(Clean_Data_Train$classe,p=0.75,list=FALSE)
Training=Clean_Data_Train[inTrain,]
Testing=Clean_Data_Train[-inTrain,]

# test 3 models to see which is the most accurate  (classification tree,
random forest, gradient boosting method)
# cross validation with 5 folds will be used
validation=trainControl(method="cv", number=5)

#Classification Tree:
C_Tree=train(classe ~., data=Training, method="rpart",trControl=validation)
fancyRpartPlot(C_Tree$finalModel)
```




```
predict_train=predict(C_Tree,newdata=Testing)
Matrix_CT=confusionMatrix(Testing$classe,predict_train)
Matrix_CT$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1254   22  116    0    3
##          B  394  322  233    0    0
##          C  386   23  446    0    0
##          D  372  134  298    0    0
##          E  128  122  239    0  412
```

```
Matrix_CT$overall[1]
```

```
##  Accuracy
## 0.4963295
```

The Accuracy of this model was about 50%, let's now see what results we can achieve with ther other two models.

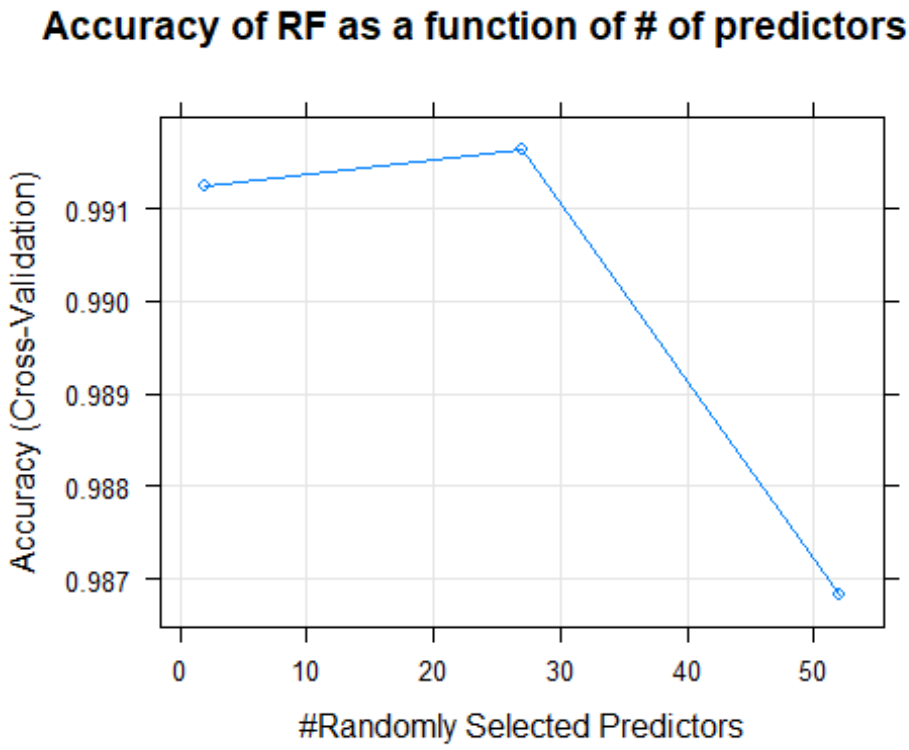
Applying the Random Forests model:

```
RF_model=train(classe
~.,data=Training,method="rf",trControl=validation,verbose=FALSE)
print(RF_model)
```

```
## 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: 11776, 11774, 11775, 11774, 11773
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9912352  0.9889113
##   27    0.9916430  0.9894277
##   52    0.9868191  0.9833250
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

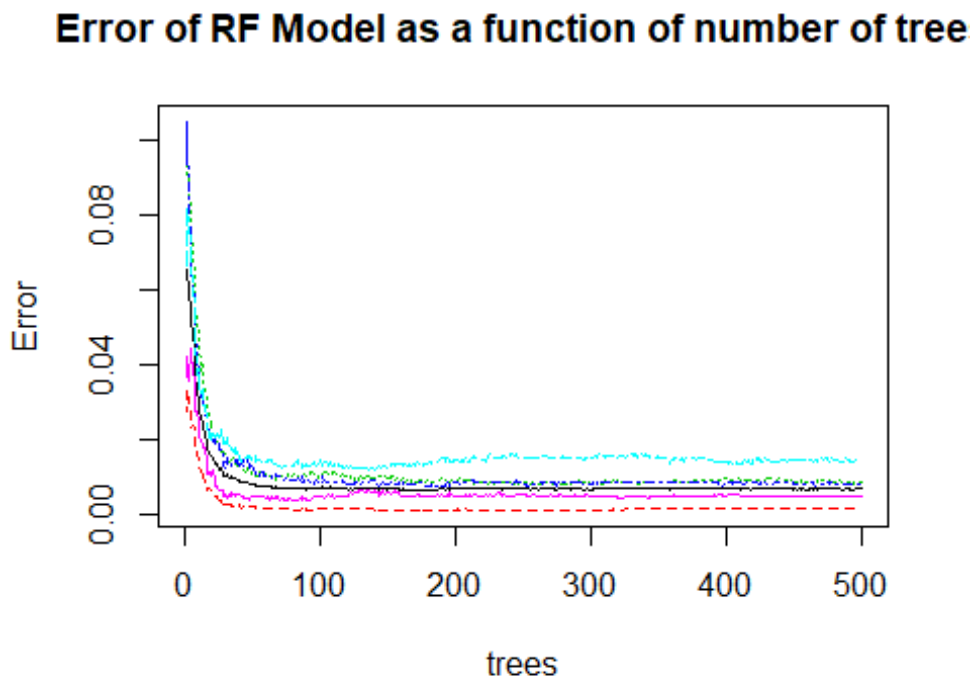
```
plot(RF_model,main="Accuracy of RF as a function of # of predictors")
```




```
predict_train_RF=predict(RF_model,newdata=Testing)
Matrix_RF=confusionMatrix(Testing$classe,predict_train_RF)
Matrix_RF$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    8  938    2    1    0
```

```
##          C    0    1  854    0    0
##          D    0    0    5  799    0
##          E    0    2    3    0  896

Matrix_RF$overall[1]

##  Accuracy
## 0.9955139

names(RF_model$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"

RF_model$finalModel$classes

## [1] "A" "B" "C" "D" "E"

plot(RF_model$finalModel,main="Error of RF Model as a function of number of
trees")
```



Random forest gives us an accuracy of about 99.6% with cross validation of 5 steps. Final approach will be the gradient boosting method.

Using the gradient boosting method:

```
GBM_model=train(classe
~.,data=Training,method="gbm",trControl=validation,verbose=FALSE)
print(GBM_model)
```

```
## 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: 11774, 11773, 11775, 11775, 11775
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7564212  0.6912282
##   1                  100      0.8205605  0.7728882
##   1                  150      0.8541249  0.8154308
##   2                   50      0.8524267  0.8130470
##   2                  100      0.9070532  0.8823913
##   2                  150      0.9294750  0.9107687
##   3                   50      0.8933281  0.8649385
##   3                  100      0.9413648  0.9258038
##   3                  150      0.9600492  0.9494540
##
## 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.
```

```
predict_train_GBM=predict(GBM_model,newdata=Testing)
Matrix_GBM=confusionMatrix(Testing$classe,predict_train_GBM)
Matrix_GBM$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1377   16    1    0    1
##          B   36  898   14    0    1
##          C    0   23  828    4    0
##          D    0    4   22  769    9
##          E    2   10   16    5  868
```

```
Matrix_GBM$overall[1]
```

```
##  Accuracy
## 0.9665579
```

Best model overall was the random forest model, therefore we will apply that particular model to the test data set.

```
Predict_Test=predict(RF_model,newdata=Clean_Data_Test)
Final_Table=as.data.frame(Predict_Test)
colnames(Final_Table)="Classe"
Final_Names=as.data.frame(Data_Test[,2])
colnames(Final_Names)="Name"
Final_Table1=cbind(Final_Names,Final_Table)
Final_Table1
```

```
##        Name Classe
## 1     pedro      B
## 2    jeremy      A
## 3    jeremy      B
## 4    adelmo      A
## 5    eurico      A
## 6    jeremy      E
## 7    jeremy      D
## 8    jeremy      B
## 9  carlitos      A
## 10  charles      A
## 11 carlitos      B
## 12   jeremy      C
## 13   eurico      B
## 14   jeremy      A
## 15   jeremy      E
## 16   eurico      E
## 17    pedro      A
## 18 carlitos      B
## 19    pedro      B
## 20   eurico      B
```