Practical Machine Learning Project

Oye Akinnikawe

January 23, 2017

Introduction

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 training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

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.

Data Processing

import packages and data

```r
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
library(randomForest)
library(RColorBrewer)

trainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(trainUrl), na.strings = c("NA", "DIV/0!", ""))
testing <- read.csv(url(testUrl), na.strings = c("NA", "DIV/0!", ""))
```

partition training into a training set and a cross validation set

```r
set.seed(45213)
inTrain <- createDataPartition(training$classe, p = 0.6, list = FALSE)
```

```
myTraining <- training[inTrain,]
myTesting <- training[-inTrain,]
```

Data Cleaning

Remove near zero variance variables

```
nzv <- nearZeroVar(myTraining)

myTraining <- myTraining[, -nzv]
myTesting <- myTesting[, -nzv]
```

Create a function to remove variables that are mostly NA

```
mostlyNA <- sapply(myTraining, function(x) mean(is.na(x))) > 0.95

myTraining <- myTraining[, mostlyNA==F]
myTesting <- myTesting[, mostlyNA==F]
```

Remove variable that won't influence prediction

```
myTraining <- myTraining[, -(1:5)]
myTesting <- myTesting[, -(1:5)]

clean <- colnames(myTraining[, -54]) # remove column "classe"
testing <- testing[clean]
```

Predictions

Three types of predictions will be used: Decision Trees, Random Forest, Generalized Boosted Regression

Prediction with decision trees

To save computing time, a 5-fold cross validation is used.

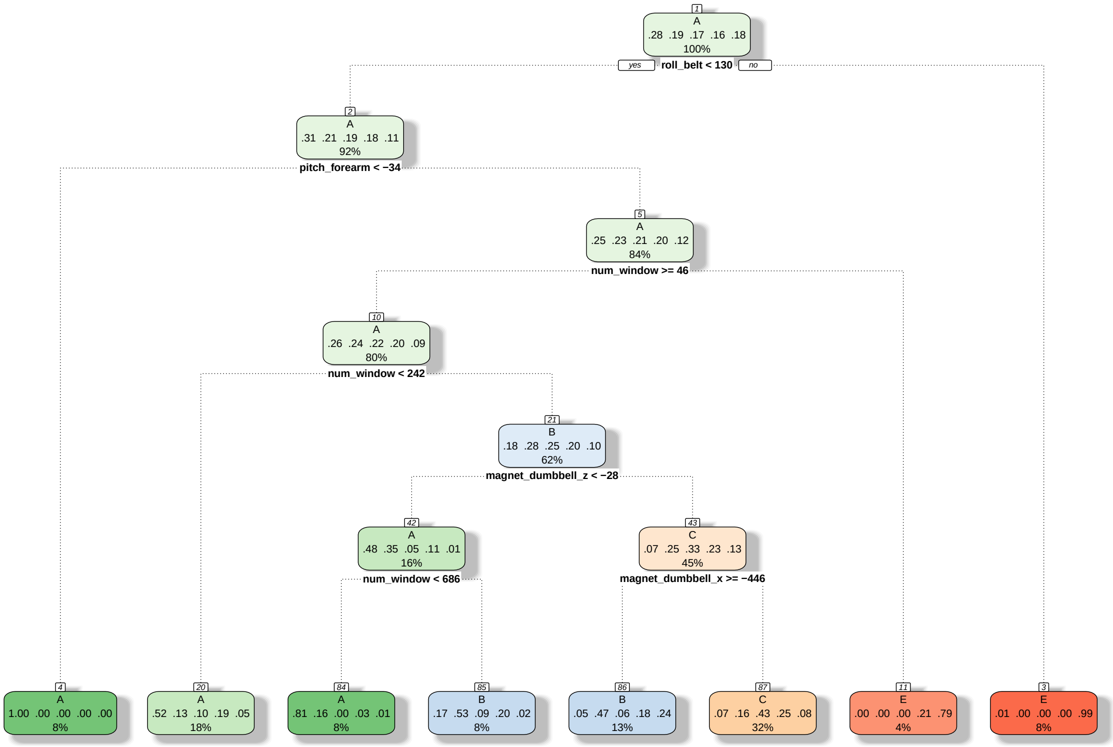
```
set.seed(45213)
control <- trainControl(method = "cv", number = 5)

rpart_fit <- train(classe ~ ., data = myTraining, method = "rpart",
                   trControl = control)

print(rpart_fit, digits = 4)
```

```
## CART
##
## 11776 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9420, 9421, 9422, 9421, 9420
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.02468  0.6301    0.52931
##   0.04212  0.5590    0.43215
##   0.11580  0.3325    0.07338
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.02468.
```

```
fancyRpartPlot(rpart_fit$finalModel)
```



Rattle 2017−Jan−23 10:46:09 Rock

predict outcomes using the cross validation set

```
rpart_predict <- predict(rpart_fit, myTesting)
```

Show prediction result

```
conf_rpart <- confusionMatrix(rpart_predict, myTesting$classe)
conf_rpart
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1902  283  164  283   75
##          B  148  826  112  329  280
##          C  177  409 1092  620  215
##          D    0    0    0    0    0
##          E    5    0    0   54  872
## 
## Overall Statistics
## 
##                Accuracy : 0.598
##                  95% CI : (0.5871, 0.6089)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.4862
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8522   0.5441   0.7982   0.0000   0.6047
## Specificity            0.8566   0.8627   0.7806   1.0000   0.9908
## Pos Pred Value         0.7026   0.4873   0.4345      NaN   0.9366
## Neg Pred Value         0.9358   0.8875   0.9482   0.8361   0.9176
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2424   0.1053   0.1392   0.0000   0.1111
## Detection Prevalence   0.3450   0.2160   0.3203   0.0000   0.1187
## Balanced Accuracy      0.8544   0.7034   0.7894   0.5000   0.7978
```

```
rpart_accuracy <- conf_rpart$overall[1]
rpart_accuracy
```

```
##  Accuracy 
## 0.5980117
```

Prediction with Random Forest

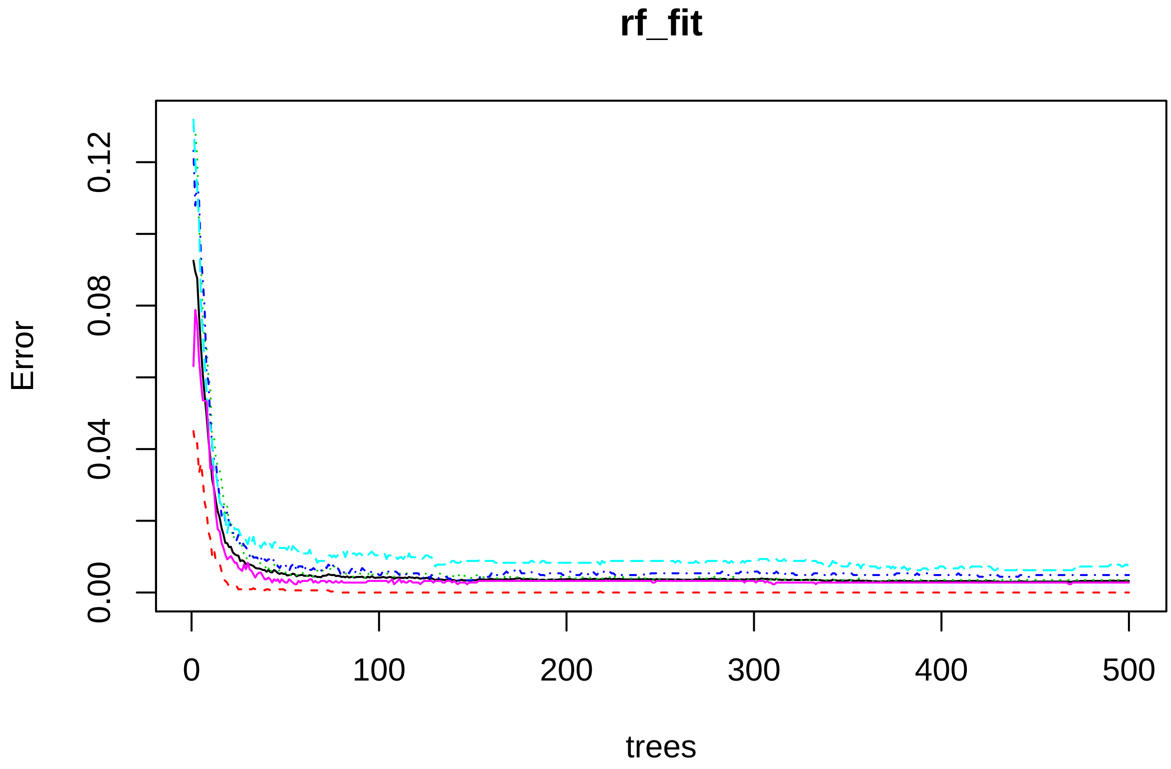
```
set.seed(45213)
rf_fit <- randomForest(classe ~ ., data = myTraining)
rf_predict <- predict(rf_fit, myTesting, type = "class")
conf_rf <- confusionMatrix(rf_predict, myTesting$classe)
conf_rf
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1514    1    0    0
##          C    0    3 1367   10    0
##          D    0    0    0 1274    5
##          E    0    0    0    2 1437
## 
## Overall Statistics
## 
##                Accuracy : 0.9972
##                  95% CI : (0.9958, 0.9982)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9965
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9993   0.9907   0.9965
## Specificity            0.9998   0.9998   0.9980   0.9992   0.9997
## Pos Pred Value         0.9996   0.9993   0.9906   0.9961   0.9986
## Neg Pred Value         1.0000   0.9994   0.9998   0.9982   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1930   0.1742   0.1624   0.1832
## Detection Prevalence   0.2846   0.1931   0.1759   0.1630   0.1834
## Balanced Accuracy      0.9999   0.9986   0.9986   0.9950   0.9981
```

```
rf_accuracy <-conf_rf$overall[1]
rf_accuracy
```

```
## Accuracy
## 0.997196
```

```
plot(rf_fit)
```



Prediction with Generalized Boosted Regression (gbm)

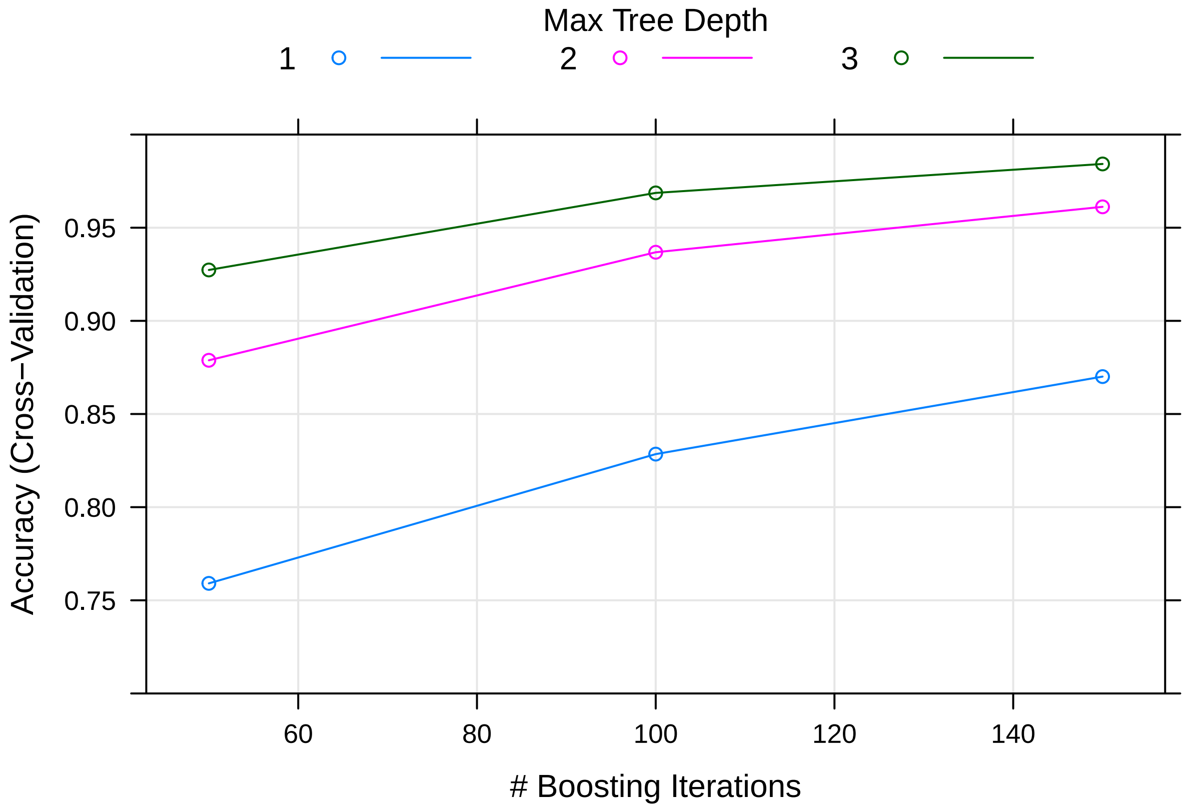
```
set.seed(45213)
gbmfit <- train(classe ~ ., data = myTraining, method = "gbm",
                trControl = control, verbose=FALSE)
gbmPredict <- predict(gbmfit, newdata = myTesting)
conf_gbm <- confusionMatrix(gbmPredict, myTesting$classe)
conf_gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230   11    0    0    0
##          B    2 1487   11   10   11
##          C    0   18 1354   20    3
##          D    0    2    2 1254   16
##          E    0    0    1    2 1412
##
## Overall Statistics
##
##                Accuracy : 0.9861
##                  95% CI : (0.9833, 0.9886)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9824
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9796   0.9898   0.9751   0.9792
## Specificity            0.9980   0.9946   0.9937   0.9970   0.9995
## Pos Pred Value         0.9951   0.9776   0.9706   0.9843   0.9979
## Neg Pred Value         0.9996   0.9951   0.9978   0.9951   0.9953
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1895   0.1726   0.1598   0.1800
## Detection Prevalence   0.2856   0.1939   0.1778   0.1624   0.1803
## Balanced Accuracy      0.9986   0.9871   0.9917   0.9860   0.9894
```

```
gbm_accuracy <- conf_gbm$overall[1]
gbm_accuracy
```

```
##  Accuracy
## 0.9861076
```

```
plot(gbmfit, ylim=c(0.7, 1))
```



Predicting Results on the Test Data

Random Forests gave an Accuracy in the cross validation (myTesting) dataset of 99.72%, which was more accurate than what was got from the Decision Trees or GBM. The expected out-of-sample error is 100-99.72 = 0.28%.

```
finalPredict <- predict(rf_fit, testing, type = "class")
finalPredict
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```