The Final Poject for Practical Machine Learning Course

Title: “To predict the manner in the exercise”

by Yuan Ren

Goal—

The goal of your project is to predict the manner in which persons did the exercise. This is the “classe” variable in the training set. We may use any of the other variables to predict with.

Introduction and Summary—

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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data—

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv) The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Loading R packages—

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(repmis)
```

Data Loading and processing —

The data was already downloaded into my local dictory

```
setwd("C:/Users/21722/Desktop/temp16/Coursera/Practical Machine Learning/Project")
training <- read.csv('./pml-training.csv', na.strings = c("NA", ""), header=T)
testing <- read.csv('./pml-testing.csv', na.strings = c("NA", ""), header=T)

training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
```

The training data will be split into two data sets – training data set (60%) and a validation data set (40%). The validation data set will be used to conduct cross validation.

```
set.seed(12500)
InTraining <- createDataPartition(training$classe, p = 0.6, list = FALSE)
Training_F <- training[InTraining, ]
dim(Training_F)
```

```
## [1] 11776    60
```

```
Validation_F <- training[-InTraining, ]
dim(Validation_F)
```

```
## [1] 7846   60
```

Remove the data with almost all NA values and only keep the data items used in modeling

```
Training_F <- Training_F[, -c(1:7)]
Validation_F <- Validation_F[c(8:60)]
```

Model building

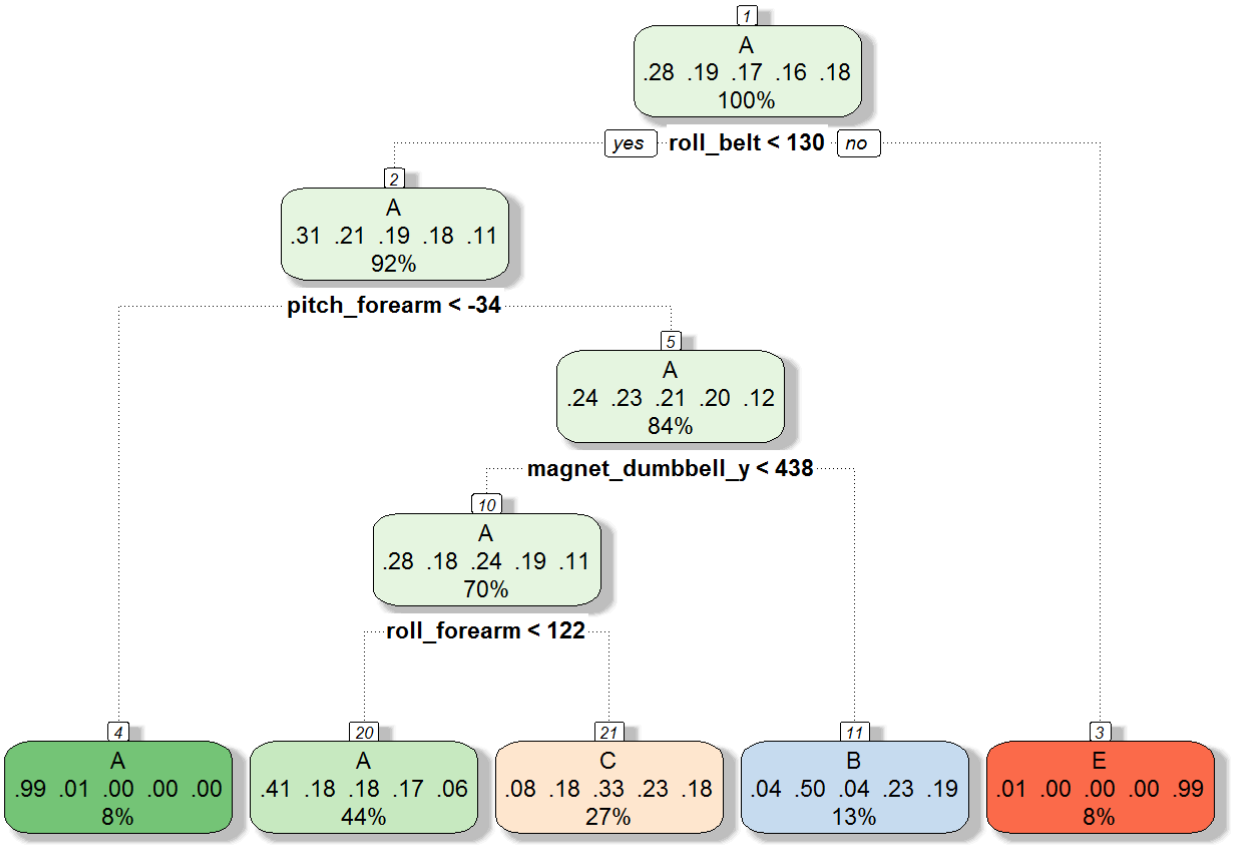
We will use three algorithms in the modeling, Decision Trees with CART, Stochatics Gradient Boosting, and Radom Forest Desision Trees.

1 Decision Trees with CART

```
FitControl <- trainControl(method = "cv", number = 5)
Model_rpart <- train(classe ~ ., data = Training_F, method = "rpart",
                     trControl = FitControl)
print(Model_rpart, digits = 5)
```

```
## CART
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9420, 9420, 9420, 9422, 9422
## Resampling results across tuning parameters:
##
##   cp        Accuracy  Kappa
##   0.033341  0.48286   0.31399
##   0.060157  0.36472   0.12268
##   0.112720  0.33169   0.07241
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.033341.
```

```
fancyRpartPlot(Model_rpart$finalModel)
```



Rattle 2017-Jan-23 01:30:31 21722

To predict outcomes using validation set and Use confusion Matrix to test results

```
predict_rpart <- predict(Model_rpart, Validation_F)
(confusion_rpart <- confusionMatrix(Validation_F$classe, predict_rpart))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2026   25  176    0    5
##          B  617  513  388    0    0
##          C  638   42  688    0    0
##          D  560  215  511    0    0
##          E  195  189  386    0  672
##
## Overall Statistics
##
##                Accuracy : 0.4969
##                  95% CI : (0.4858, 0.5081)
##     No Information Rate : 0.5144
##     P-Value [Acc > NIR] : 0.999
##
##                   Kappa : 0.3431
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5020  0.52134  0.32015       NA  0.99261
## Specificity            0.9459  0.85354  0.88064   0.8361  0.89259
## Pos Pred Value         0.9077  0.33794  0.50292       NA  0.46602
## Neg Pred Value         0.6420  0.92557  0.77447       NA  0.99922
## Prevalence             0.5144  0.12541  0.27390   0.0000  0.08629
## Detection Rate         0.2582  0.06538  0.08769   0.0000  0.08565
## Detection Prevalence   0.2845  0.19347  0.17436   0.1639  0.18379
## Balanced Accuracy      0.7240  0.68744  0.60039       NA  0.94260
```

```
(accuracy_rpart <- confusion_rpart$overall[1])
```

```
##  Accuracy
## 0.4969411
```

The value of accuracy indicates the model doesn't work well.

2 Stochatics Gradient Boosting

```
Model_GBM <- train(classe ~ ., data = Training_F, method = "gbm",
                   trControl = FitControl)
```

```
## Loading required package: gbm
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

```
## Loading required package: plyr
```

```
print(Model_GBM, digits = 5)
```

```
## Stochastic Gradient Boosting
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9422, 9422, 9420, 9420, 9420
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa
##   1                   50      0.74907   0.68175
##   1                  100      0.82091   0.77330
##   1                  150      0.85207   0.81277
##   2                   50      0.84867   0.80829
##   2                  100      0.90472   0.87941
##   2                  150      0.93011   0.91155
##   3                   50      0.89530   0.86741
##   3                  100      0.93809   0.92166
##   3                  150      0.95635   0.94478
##
## 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 outcomes using validation set and Use confusion Matrix to test results

```
predict_GBM <- predict(Model_GBM, Validation_F)
(confusion_GMB <- confusionMatrix(Validation_F$classe, predict_GBM))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2197   26    5    2    2
##          B   43 1426   45    3    1
##          C    0   56 1293   17    2
##          D    1    1   44 1229   11
##          E    1   18   13   12 1398
##
## Overall Statistics
##
##                Accuracy : 0.9614
##                  95% CI : (0.9569, 0.9655)
##     No Information Rate : 0.2858
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9511
##  Mcnemar's Test P-Value : 8.242e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9799   0.9339   0.9236   0.9731   0.9887
## Specificity            0.9938   0.9854   0.9884   0.9913   0.9932
## Pos Pred Value         0.9843   0.9394   0.9452   0.9557   0.9695
## Neg Pred Value         0.9920   0.9840   0.9835   0.9948   0.9975
## Prevalence             0.2858   0.1946   0.1784   0.1610   0.1802
## Detection Rate         0.2800   0.1817   0.1648   0.1566   0.1782
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9868   0.9596   0.9560   0.9822   0.9909
```

3 Radom Forest Desision Trees

```
Model_RF <- train(classe ~ ., data = Training_F, method = "rf",
                  trControl = FitControl)
print(Model_RF, digits = 5)
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9420, 9420, 9422, 9421, 9421
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.98837   0.98528
##   27    0.98955   0.98679
##   52    0.98319   0.97873
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
predict_RF <- predict(Model_RF, Validation_F)
(confusion_RF <- confusionMatrix(Validation_F$classe, predict_RF))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    2    0    0    0
##          B    3 1512    2    1    0
##          C    0   19 1340    9    0
##          D    0    0   14 1271    1
##          E    0    0    1    3 1438
##
## Overall Statistics
##
##                Accuracy : 0.993
##                  95% CI : (0.9909, 0.9947)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9911
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9863   0.9875   0.9899   0.9993
## Specificity            0.9996   0.9990   0.9957   0.9977   0.9994
## Pos Pred Value         0.9991   0.9960   0.9795   0.9883   0.9972
## Neg Pred Value         0.9995   0.9967   0.9974   0.9980   0.9998
## Prevalence             0.2846   0.1954   0.1730   0.1637   0.1834
## Detection Rate         0.2842   0.1927   0.1708   0.1620   0.1833
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9992   0.9927   0.9916   0.9938   0.9993
```

Conclusion—

The analysis show both Stochatics Gradient Boosting and Radom Forest work well and RF gave the best modeling. So we will use RF for prediction in testing data. However, RF demands much higher computation.

Predicting in testing data—

```
prediction_testing <-(predict(Model_RF, testing))
prediction_testing
```

```
##  [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
```