Practical Machine Learning

Wei

7 June 2017

Background and Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit 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, we will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participant They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The five ways are 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). Only Class A corresponds to correct performance. The goal of this project is to predict the manner in which they did the exercise, i.e., Class A to E. More information is available from the website here: link (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here: link
The test data are available here: link
The data for this project come from this source: link. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Goal

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.

Getting and loading the data

```
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
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(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(randomForest)
library(repmis)
library(knitr)
```

```
#import the data from the URLs
trainurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- source_data(trainurl, na.strings = c("NA", "#DIV/0!", ""), header = TRUE)
```

```
## Downloading data from: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
```

```
## SHA-1 hash of the downloaded data file is:
## c7d3bd5499ad11292935ace3d69d92b2915be894
```

```
testing <- source_data(testurl, na.strings = c("NA", "#DIV/0!", ""), header = TRUE)
```

```
## Downloading data from: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
```

```
## SHA-1 hash of the downloaded data file is:
## 7d3270a3e28102e16420acbce902a5f05286dee1
```

Data Cleaning

Delete columns that contain missing values and remove 1st - 7th columns as it is not useful for prediction models

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
trainData <- training[, -c(1:7)]
testData <- testing[, -c(1:7)]
```

Split trainData to "Training Set" - 70% and "Validation Set" - 30%

```
set.seed(1114)
inTrain <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
train <- trainData[inTrain,]
valid <- trainData[-inTrain,]
```

Prediction Models

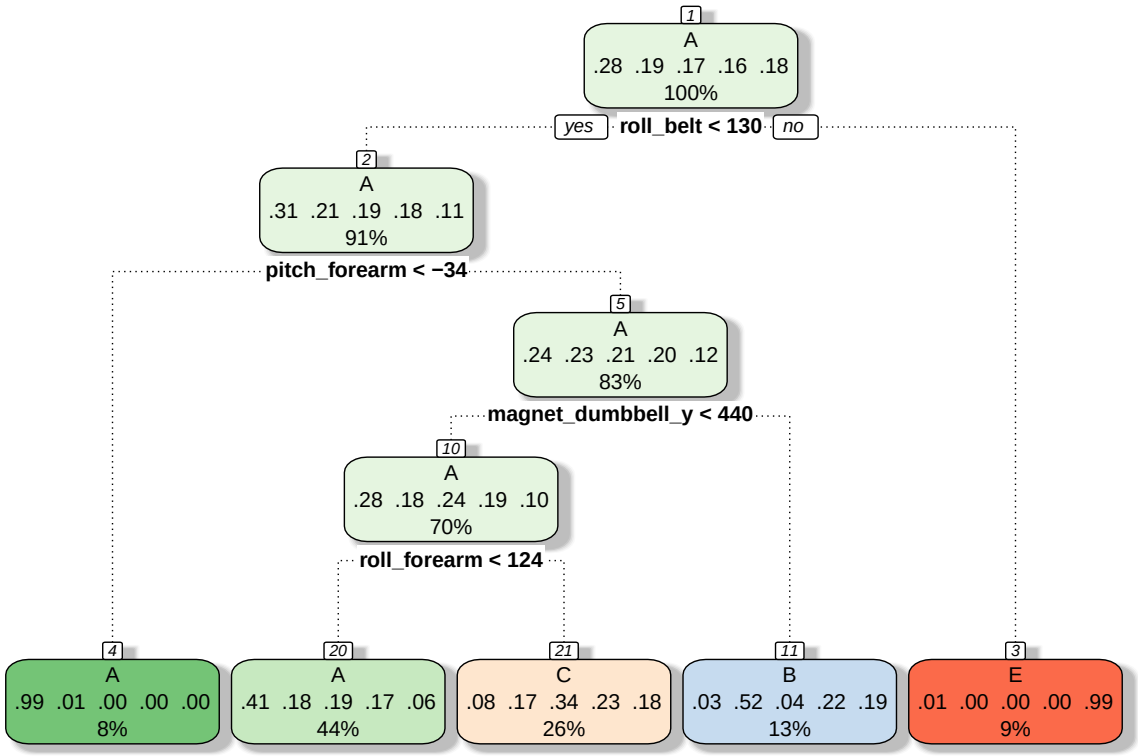
We will use classification trees and random forests to predict the outcome

Classification trees

```
control <- trainControl(method = "cv", number = 5)
fit_rpart <- train(classe ~ ., data = train, method = "rpart",
                   trControl = control)
print(fit_rpart, digits = 4)
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10990, 10991, 10989, 10988
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.03804  0.5204    0.3739
##   0.06151  0.4197    0.2134
##   0.11738  0.3333    0.0746
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03804.
```

```
fancyRpartPlot(fit_rpart$finalModel)
```



Rattle 2017−Jun−10 15:57:40 Wei

```
# predict outcomes using validation set
predict_rpart <- predict(fit_rpart, valid)
(conf_rpart <- confusionMatrix(valid$classe, predict_rpart))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1515   31  127    0    1
##          B  473  362  304    0    0
##          C  477   40  509    0    0
##          D  435  172  357    0    0
##          E  184  138  296    0  464
##
## Overall Statistics
##
##                Accuracy : 0.4843
##                  95% CI : (0.4714, 0.4971)
##     No Information Rate : 0.524
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3257
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4912  0.48721  0.31952       NA  0.99785
## Specificity            0.9432  0.84889  0.87954   0.8362  0.88598
## Pos Pred Value         0.9050  0.31782  0.49610       NA  0.42884
## Neg Pred Value         0.6274  0.91972  0.77691       NA  0.99979
## Prevalence             0.5240  0.12625  0.27069   0.0000  0.07901
## Detection Rate         0.2574  0.06151  0.08649   0.0000  0.07884
## Detection Prevalence   0.2845  0.19354  0.17434   0.1638  0.18386
## Balanced Accuracy      0.7172  0.66805  0.59953       NA  0.94191
```

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

```
##  Accuracy
## 0.4842821
```

From the confusion matrix, the accuracy rate is 0.5, and so the out-of-sample error rate is 0.5. Using classification tree does not predict the outcome classe very well.

Random forests

```
fit_rf <- train(classe ~ ., data = train, method = "rf",
                  trControl = control)
print(fit_rf, digits = 4)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10990, 10991, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9908    0.9883
##   27    0.9905    0.9880
##   52    0.9800    0.9747
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
# predict outcomes using validation set
predict_rf <- predict(fit_rf, valid)
# Show prediction result
(conf_rf <- confusionMatrix(valid$classe, predict_rf))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    7 1131    1    0    0
##          C    0   10 1015    1    0
```

```
##          D    0    0   17  945    2
##          E    0    0    0    2 1080
##
## Overall Statistics
##
##                Accuracy : 0.993
##                  95% CI : (0.9906, 0.995)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9912
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9904   0.9826   0.9968   0.9982
## Specificity            0.9998   0.9983   0.9977   0.9962   0.9996
## Pos Pred Value         0.9994   0.9930   0.9893   0.9803   0.9982
## Neg Pred Value         0.9983   0.9977   0.9963   0.9994   0.9996
## Prevalence             0.2855   0.1941   0.1755   0.1611   0.1839
## Detection Rate         0.2843   0.1922   0.1725   0.1606   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9978   0.9943   0.9902   0.9965   0.9989
```

```
(accuracy_rf <- conf_rf$overall[1])
```

```
##  Accuracy
## 0.9930331
```

For this dataset, random forest method is way better than classification tree method. The accuracy rate is 0.991, and so the out-of-sample error rate is 0.009. This may be due to the fact that many predictors are highly correlated. Random forests chooses a subset of predictors at each split and decorrelate the trees. This leads to high accuracy, although this algorithm is sometimes difficult to interpret and computationally inefficient.

Predict Test Set

```
(predict(fit_rf, testData))
```

```
##  [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
```