Human Activity Recognition

Dongjun_Cho

7/11/2020

Background

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://web.archive.org/web/20161224072740/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

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Summary

The goal of our project is to predict the manner in which they did the exercise.

Libraries

```
library(lattice)
library(caret)
```

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

```
library(randomForest)
```

```
## randomForest 4.6-14
```

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

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

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

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

##
## Attaching package: 'rattle'

## The following object is masked from 'package:randomForest':
##
##     importance
```

Getting and cleaning the data

```
train <- read.csv("C:/Users/dongj/Desktop/R_data_Desk/Practical_machine_learning/pml-training.csv", na.s

test <- read.csv("C:/Users/dongj/Desktop/R_data_Desk/Practical_machine_learning/pml-testing.csv", na.str

dim(train)
```

```
## [1] 19622   160
```

```
dim(test)
```

```
## [1]  20 160
```

Remove missing values

In the summary, We can find there are missing values in our dataset.

```
train <- train[,colSums(is.na(train))==0]
test <- test[,colSums(is.na(test))==0]
```

Remove unrelated data columns

columns 1 through columns 7 are the data that are not related to this model (user_name, raw_timestamp_part_1, raw_timestamp_part_,2 cvtd_timestamp, new_window, and num_window)

```
train <- train[,-c(1:7)]
test <- test[,-c(1:7)]
```

Cross Validation

Using the training data set, we split the data set into training and test sets

```
inTrain <- createDataPartition(train$classe, p=0.75, list = FALSE)
training <- train[inTrain,]
testing <- train[-inTrain,]
```

Decision Tree

```
train_fit <- train(classe ~ .,method = "rpart", data = training)
print(train_fit$finalModel)
```

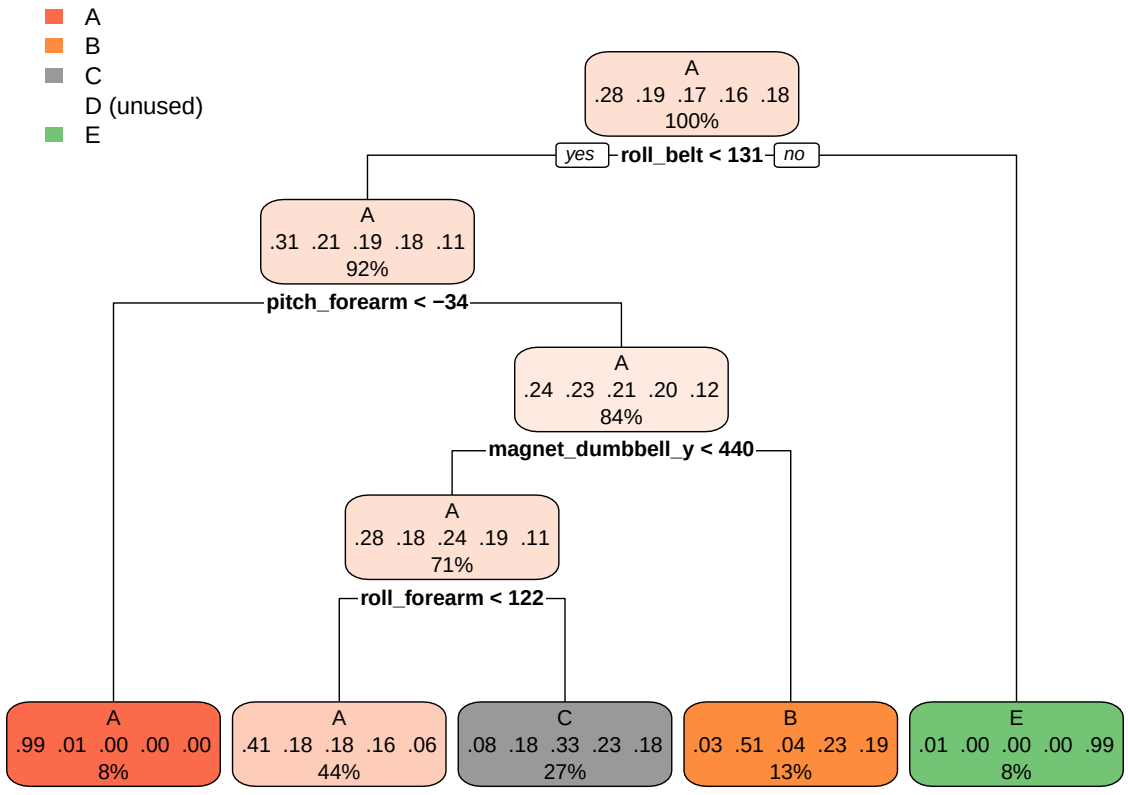
```
## n= 14718
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 14718 10533 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 13479  9307 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -34.15 1168     8 A (0.99 0.0068 0 0 0) *
##      5) pitch_forearm>=-34.15 12311  9299 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 439.5 10420  7472 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 121.5 6420  3785 A (0.41 0.18 0.18 0.16 0.061) *
##         21) roll_forearm>=121.5 4000  2696 C (0.078 0.18 0.33 0.23 0.18) *
##       11) magnet_dumbbell_y>=439.5 1891   930 B (0.034 0.51 0.041 0.23 0.19) *
##    3) roll_belt>=130.5 1239    13 E (0.01 0 0 0 0.99) *
```

```
predict_fit <- predict(train_fit, testing)
confusionMatrix(table(predict_fit, testing$classe))
```

```
## Confusion Matrix and Statistics
##
##
## predict_fit    A    B    C    D    E
##           A 1270  392  398  370  132
##           B   17  325   31  135  130
##           C  107  232  426  299  234
##           D    0    0    0    0    0
##           E    1    0    0    0  405
##
## Overall Statistics
##
##                Accuracy : 0.4947
##                  95% CI : (0.4806, 0.5088)
##     No Information Rate : 0.2845
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3393
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9104  0.34247  0.49825   0.0000  0.44950
## Specificity            0.6318  0.92086  0.78464   1.0000  0.99975
## Pos Pred Value         0.4957  0.50940  0.32820      NaN  0.99754
## Neg Pred Value         0.9466  0.85373  0.88103   0.8361  0.88973
## Prevalence             0.2845  0.19352  0.17435   0.1639  0.18373
## Detection Rate         0.2590  0.06627  0.08687   0.0000  0.08259
## Detection Prevalence   0.5224  0.13010  0.26468   0.0000  0.08279
## Balanced Accuracy      0.7711  0.63166  0.64144   0.5000  0.72463
```

```
rpart.plot(train_fit$finalModel)
```



Random Forest Model

```r
train_rf <- train(classe ~ .,method = "rf", data = training, ntree= 50)
print(train_rf)
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9873239  0.9839658
##   27    0.9886901  0.9856954
##   52    0.9807505  0.9756518
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```r
predict_rf <- predict(train_rf, testing)
confusionMatrix(table(predict_rf, testing$classe))
```

```
## Confusion Matrix and Statistics
##
##
## predict_rf    A    B    C    D    E
##          A 1394    9    0    0    0
##          B    1  935    7    0    0
##          C    0    5  846    7    1
##          D    0    0    2  795    1
##          E    0    0    0    2  899
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9901, 0.995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9852   0.9895   0.9888   0.9978
## Specificity            0.9974   0.9980   0.9968   0.9993   0.9995
## Pos Pred Value         0.9936   0.9915   0.9849   0.9962   0.9978
```

```
## Neg Pred Value         0.9997   0.9965   0.9978   0.9978   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1907   0.1725   0.1621   0.1833
## Detection Prevalence   0.2861   0.1923   0.1752   0.1627   0.1837
## Balanced Accuracy      0.9984   0.9916   0.9931   0.9940   0.9986
```

Accuracy for Decision tree gives 49.84% compared to the Random forest model which is 99.23% accuracy. The formula for expected out-of-sample error is 1 - accuracy for prediction. The expected out-of-sample error for the decision tree is 50.14%, but the Random forest model gives only 0.77%.

Since the Random forest model prediction gives higher accuracy than decision tree prediction, we choose a random forest model prediction.

Random Forest Prediction Model

```
predict(train_rf, test)
```

```
##  [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
```