Practical Machine Learning Project

Loc Nguyen

11/7/2021

About this project

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. 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.

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://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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Approach

Two models will be tested using decision tree and random forest. The model with the highest accuracy will be chosen as our final model. We will divide the train data in training data and cross-validation data. Here we are using validation set approach. (train data (75% of the original Training data set) and validation data (25%)) The expected out-of-sample error will correspond to the quantity: 1-accuracy in the cross-validation data. Accuracy is the proportion of correct classified observation over the total sample

Loading libraries

Loading necessary libraries for the report

```
library(ggplot2); library(caret); library(randomForest); library(rpart); library(rpart.plot)
```

Loading data and pre-processing

Load a data set

```
training <- read.csv("pml-training.csv", na.strings = c("NA","#DIV/0!", ""))
testing <- read.csv("pml-testing.csv", na.strings = c("NA","#DIV/0!", ""))
```

You can investigate data by str function. The training dataset has 19622 observations and 160 variables, and the testing data set contains 20 observations. We will build model based on training set

```
str(testing)
str(training)
```

Cleaning data

We will delete columns which include all missing value

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

Then we delete columns which are irrelevant, and have little predicting power, follow me these are columns from 1 to 7

```
trainset <-training[, -c(1:7)]
testset <- testing[, -c(1:7)]
```

Then we change variable "classe" in training set to factor. It will help us later when we predict our test data using our predictive model.

```
trainset$classe <- as.factor(trainset$classe)
```

Data splitting

Partition the data so that 75% of the training dataset into training and the remaining 25% to validation set compute the out-of-sample errors.

```
set.seed(1712)
inTrain <- createDataPartition(y=trainset$classe, p=0.75, list=FALSE)
train <- trainset[inTrain, ]
valid <- trainset[-inTrain, ]
table(train$classe)
```

```
##
##    A    B    C    D    E
## 4185 2848 2567 2412 2706
```

So, the variable "classe" contains 5 levels: A, B, C, D and E. Level A is the most frequent while level D is the least frequent.

Building prediction algorithms

We will investigate the classification trees and random forests to predict the outcome.

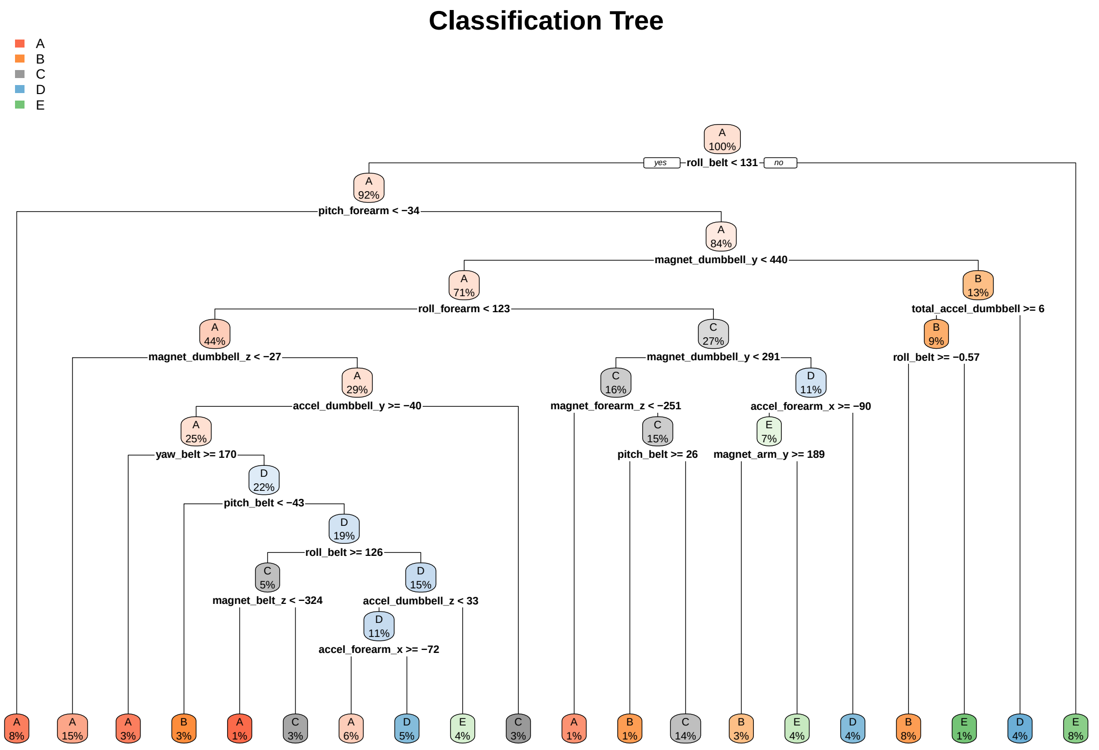
1. Classification trees

```
model1 <- rpart(classe ~ ., data=train, method="class")
prediction1 <- predict(model1, valid, type = "class")
confusionMatrix(prediction1, valid$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1273  195   59  129   50
##          B   40  526   85   41   53
##          C   29   64  617  121  104
##          D   36   73   73  423   29
##          E   17   91   21   90  665
##
## Overall Statistics
##
##                Accuracy : 0.7145
##                  95% CI : (0.7017, 0.7271)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6359
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9125   0.5543   0.7216  0.52612   0.7381
## Specificity            0.8766   0.9446   0.9215  0.94854   0.9453
## Pos Pred Value         0.7462   0.7060   0.6599  0.66719   0.7523
## Neg Pred Value         0.9619   0.8983   0.9400  0.91077   0.9413
## Prevalence             0.2845   0.1935   0.1743  0.16395   0.1837
## Detection Rate         0.2596   0.1073   0.1258  0.08626   0.1356
## Detection Prevalence   0.3479   0.1519   0.1907  0.12928   0.1803
## Balanced Accuracy      0.8946   0.7494   0.8215  0.73733   0.8417
```

If we plot the classification trees

```
rpart.plot(model1, main="Classification Tree", extra=100)
```



From the confusion matrix, the accuracy rate is 0.7445

2. Random Forest

```
model2 <- randomForest(classe ~. , data=train, method="class")
prediction2 <- predict(model2, valid, type = "class")
confusionMatrix(prediction2, valid$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    1    0    0    0
##          B    2  948    7    0    0
##          C    0    0  845   11    0
##          D    0    0    3  791    1
##          E    0    0    0    2  900
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.992, 0.9964)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.993
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9989   0.9883   0.9838   0.9989
## Specificity            0.9997   0.9977   0.9973   0.9990   0.9995
## Pos Pred Value         0.9993   0.9906   0.9871   0.9950   0.9978
## Neg Pred Value         0.9994   0.9997   0.9975   0.9968   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1933   0.1723   0.1613   0.1835
## Detection Prevalence   0.2843   0.1951   0.1746   0.1621   0.1839
## Balanced Accuracy      0.9991   0.9983   0.9928   0.9914   0.9992
```

Decision on which Prediction Model to Use

Random Forest algorithm performed better than Decision Trees. Accuracy for Random Forest model was 0.995 compared to Decision Tree model with 0.739 The expected out-of-sample error of RF algorithm is 0.005. Although, This leads to high accuracy, although RD is difficult to interpret and computationally cost.

Prediction on Testing Set

Here is the final outcome based on the Prediction Model 2 (Random Forest) applied for the testing set.

```
predict(model2, testset)
```

```
##  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
```