Practical Machine Learning: Prediction Assignment Wrapup

Tom Mat

2023-01-28

Summary

This document is the assignment of the Practical Machine Learning course. Using the several machine learning techniques, I created the prediction model and applied them to the test data. I evaluated each methods and finally I reccomend the best model I tried to. You can see my trial at the following website.

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 (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data provided by the course

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)

Set up libraries

```
suppressMessages(library(dplyr))
suppressMessages(library(tidyr))
suppressMessages(library(ggplot2))
suppressMessages(library(caret))
suppressMessages(library(gt))
suppressMessages(library(rpart))
suppressMessages(library(rattle))
```

Uptaking the data

I downloaded the two dataset and set them to training_pre and testing dataset.

```
training_pre <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")
```

Cleaning training_pre dataset

I dropped the useless variables.

```
# drop useless variables. First 7 variables will obviously not contribute the creation of models
training_pre <- training_pre[,-(1:7)]
# drop the variables which contains a lot of NA.
LotOfNA <- sapply(training_pre, function(x) mean(is.na(x))) > 0.95
training_pre <- training_pre[,LotOfNA==FALSE]
# drop the variables with near zero variance
NearZero <- nearZeroVar(training_pre)
training_pre <- training_pre[,-NearZero]
```

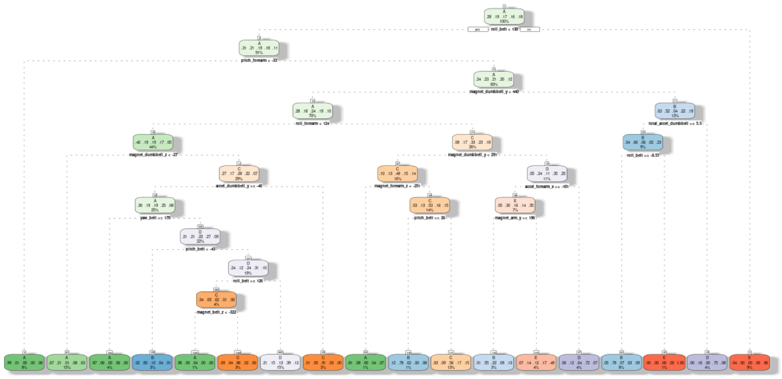
Creating the training and validation dataset

I divided the training_pre dataset into training and validation dataset. 70 % of training_pre goes into the training dataset.

```
set.seed(333)
inTrain <- createDataPartition(y=training_pre$classe, p=0.7, list=FALSE)
training <- training_pre[inTrain,]
validation <- training_pre[-inTrain,]
```

Building model by using rpart

```
model_rpart <- rpart(classe ~ ., data=training, cp=0.01, maxdepth=10)
fancyRpartPlot(model_rpart)
```



Rattle 2023- 1月-29 10:52:09 tommat

Predcition by rpart

```
predict_rpart <- predict(model_rpart, newdata=validation, type="class")
conf_matrix_rpart <- confusionMatrix(predict_rpart, as.factor(validation$classe))
conf_matrix_rpart
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1340  195   12   82   29
##          B   39  609  102   38   66
##          C   40   98  772  132  105
##          D  206  202  116  663  177
##          E   49   35   24   49  705
##
## Overall Statistics
##
##                Accuracy : 0.6948
##                  95% CI : (0.6829, 0.7066)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6151
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8005   0.5347   0.7524   0.6878   0.6516
## Specificity            0.9245   0.9484   0.9228   0.8575   0.9673
## Pos Pred Value         0.8082   0.7131   0.6731   0.4861   0.8179
## Neg Pred Value         0.9210   0.8947   0.9464   0.9334   0.9249
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2277   0.1035   0.1312   0.1127   0.1198
## Detection Prevalence   0.2817   0.1451   0.1949   0.2318   0.1465
## Balanced Accuracy      0.8625   0.7415   0.8376   0.7727   0.8094
```

Building model by using random forest

```
model_rf <- train(classe ~ ., data=training, method="rf",
                  trControl=trainControl(method="cv",number=3))
```

Predicting by using random forest

```
predict_rf <- predict(model_rf, newdata=validation)
conf_matrix_rf <- confusionMatrix(predict_rf, as.factor(validation$classe))
conf_matrix_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    3    0    0    0
##          B    3 1134    9    0    0
##          C    0    2 1016   24    0
##          D    0    0    1  940   11
##          E    0    0    0    0 1071
##
## Overall Statistics
##
##                Accuracy : 0.991
##                  95% CI : (0.9882, 0.9932)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9886
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9956   0.9903   0.9751   0.9898
## Specificity            0.9993   0.9975   0.9946   0.9976   1.0000
## Pos Pred Value         0.9982   0.9895   0.9750   0.9874   1.0000
## Neg Pred Value         0.9993   0.9989   0.9979   0.9951   0.9977
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1927   0.1726   0.1597   0.1820
## Detection Prevalence   0.2845   0.1947   0.1771   0.1618   0.1820
## Balanced Accuracy      0.9987   0.9965   0.9925   0.9863   0.9949
```

Comparing rpart model and random forest model

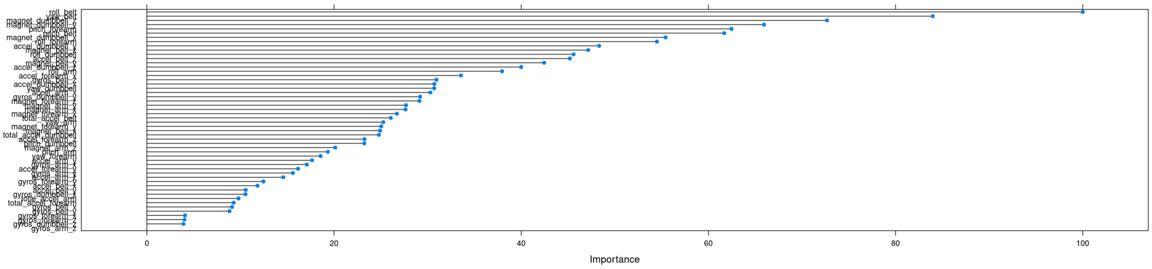
As shown above, random forest model is much better than rpart model. The accuracy of the rpart model is 0.6948, while the accuracy of the random forest model is 0.9907. The sensitivity of the rpart model is low at Class B, D and E. As for positive prediction value of the rpart model is low especially at Class D. On the other hand, the random forest model is excellent at any accuracy statistics including sensitivity, specificity, positive prediction values and negative prediction values. Therefore, I chose to use the random forest model. Next I examine which factors are most important in the random forest model.

Important variables in the random forest model

```
print(varImp(model_rf))
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                   Overall
## roll_belt          100.00
## yaw_belt            83.97
## magnet_dumbbell_z   72.68
## magnet_dumbbell_y   65.94
## pitch_forearm       62.47
## pitch_belt          61.70
## magnet_dumbbell_x   55.43
## roll_forearm        54.50
## accel_dumbbell_y    48.33
## magnet_belt_z       47.16
## roll_dumbbell       45.59
## accel_belt_z        45.19
## magnet_belt_y       42.45
## accel_dumbbell_z    39.99
## roll_arm            37.96
## accel_forearm_x     33.56
## gyros_belt_z        30.94
## accel_dumbbell_x    30.72
## yaw_dumbbell        30.70
## accel_arm_x         30.29
```

```
plot(varImp(model_rf))
```



Predictin the classification of test data.

```
predict_test <- predict(model_rf, newdata=testing)
predict_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
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.