Project_JHU

Yujie Liu

July 20, 2017

Project Introduction

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. 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.

Analysis

```r
library(rpart)
library(caret)
library(rattle)
library(tidyr)
library(dplyr)
library(randomForest)
```

1. Load and Clean the Data

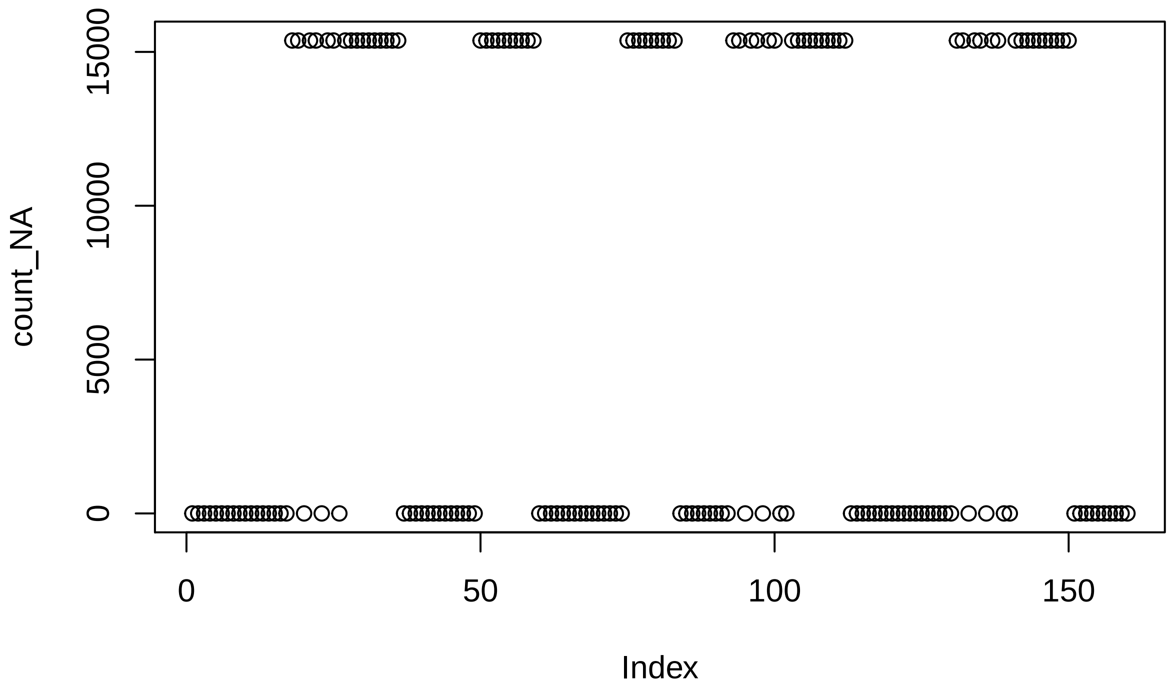
In the data file "pml-testing.csv", there is no response. So my job is to make the prediction. Thus, I break the data file "pml-training.csv" into training data (80%) and testing data(80%).

```r
data = read.csv('pml-training.csv')
pred = read.csv('pml-testing.csv')

set.seed(1)
inTrain = createDataPartition(data$classe, p = 0.8, list = FALSE)

# training and testing set
training = data[inTrain,]
testing = data[-inTrain,]


# Check the summary of the data
# str(training)
#
# summary(training)
```

The data has 160 variables, which is too large for direct training. I need to check the data and see whether I can reduce the dimension of the training data.

First, check the NA values in the data. I found some features have too many NAs. We need to delete those.

```r
# count NAs of each column
count_NA = apply(training, 2, function(x) sum(is.na(x)))
plot(count_NA)
```



```r
summary(count_NA)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       0    6436   15370   15370
```

```
# find the column which has too many NAs
ind = count_NA > 1000

# delete the columns with too many NAs
training1 = training[,!ind]
testing1 = testing[,!ind]
pred1 = pred[,!ind]
```

Then I found that some columns are unique to each observation. That feature does not provide new information. I need to delete them as well.

```
# delete the following columns:
# X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp


training2 = select(training1, -(X:new_window))
testing2 = select(testing1, -(X:new_window))
pred2 = select(pred1, -(X:new_window))
```

Some columns have unreasonable values and need to be deleted.

```
# find the columns with the name containing "amplitude"
# delete the columns with yaw_forearm, yaw_dumbbell, yaw_belt
ind_amp = sapply(names(training2), function(x) grepl("amplitude",x) | grepl("yaw_forearm",x) | grepl("y


training3 = training2[,!ind_amp]
testing3 = testing2[,!ind_amp]
pred3 = pred2[,!ind_amp]
```

Some columns are factors. The levels of the factors are similar to a float number. After checking on that, too many NAs appear after we force it to be numeric. Instead of deleting the rows, I choose to delete the columns.

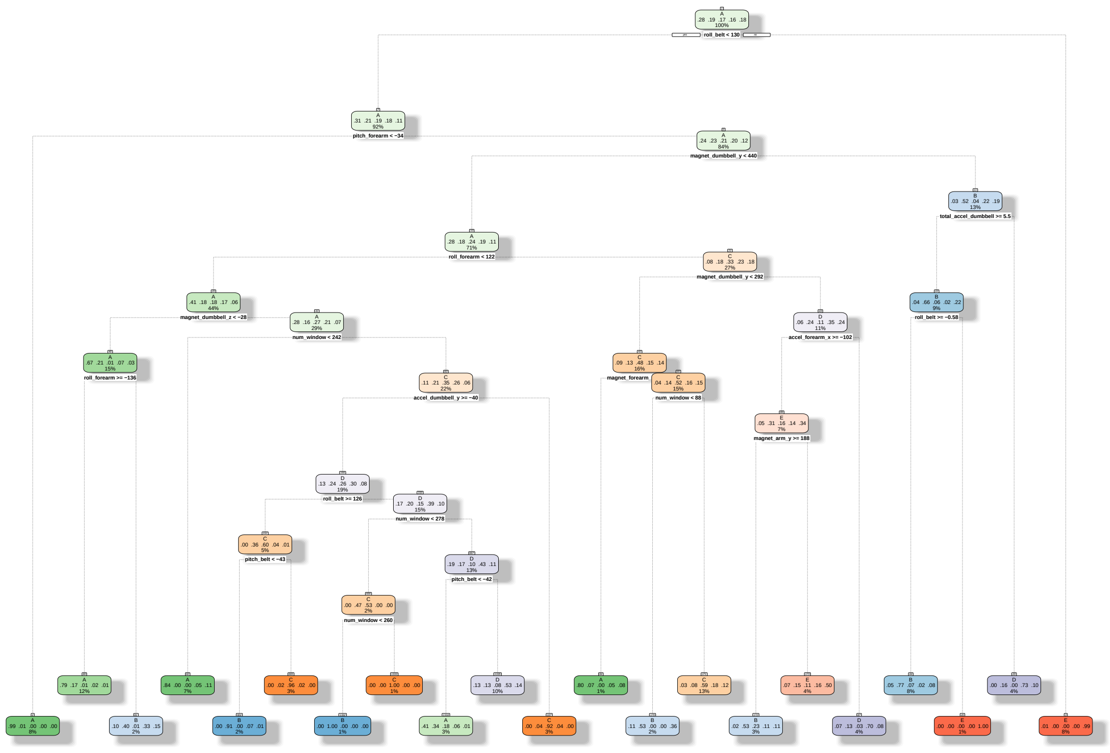
```
# find the factors of all columns
# index of factor
ind_fac = sapply(training3, function(x) is.factor(x))

ind_fac[length(ind_fac)] = FALSE # neglect the last column in factor

# sum(ind_fac)

# convert the factor to numeric
tmp = sapply(training3[,ind_fac], function(x) as.numeric(levels(x))[x])
# Observation ------------------------------------------------------------
# We find NAs after the forcing the column to be factors
# ----------------------------------------------------------------------

# if deleting all the factor columns
training4 = training3[,!ind_fac]
testing4 = testing3[,!ind_fac]
pred4 = pred3[,!ind_fac]
pred4 = pred4[,-ncol(pred4)]
```

Modeling

First, I apply decision tree to fit the data. I obtain 73% accuracy on the cross validation set.

```
# Decision tree to fit the value ======================================================

ind_spl = sample(1:nrow(training4), nrow(training4))
fit = rpart(classe~., method = 'class', data = training4)

# printcp(fit)
# plotcp(fit)
# summary(fit)

# plot tree
# plot(fit, uniform=TRUE,
#      main="Classification Tree")

fancyRpartPlot(fit)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



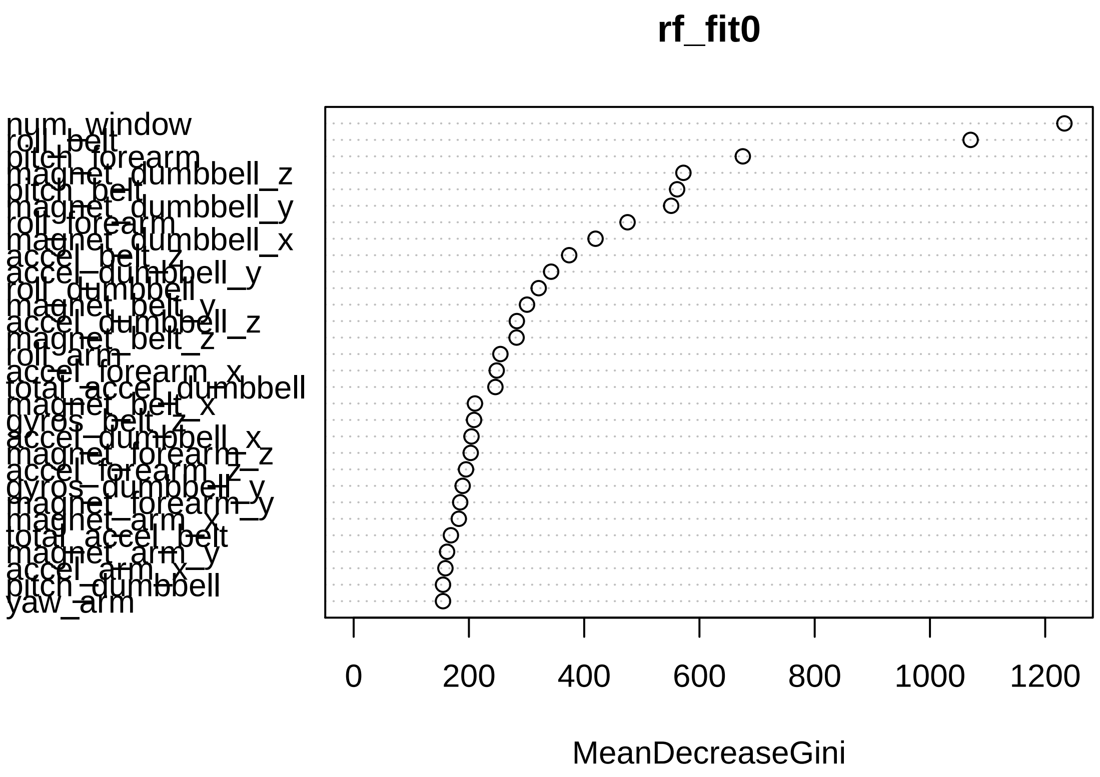
Rattle 2017−Jul−20 20:07:58 yjliu

```
# prediction on the testing data
fit_pred = predict(fit, newdata = testing4, type = 'class')
confusionMatrix(fit_pred, testing4$classe)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   A   B   C   D   E
##          A 994 122  28  54  33
##          B  42 486  45  55  80
##          C   8  47 559 102  60
##          D  53  83  37 393  95
##          E  19  21  15  39 453
##
## Overall Statistics
##
##                Accuracy : 0.7354
##                  95% CI : (0.7213, 0.7492)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6643
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8907   0.6403   0.8173   0.6112   0.6283
## Specificity            0.9156   0.9298   0.9330   0.9183   0.9706
## Pos Pred Value         0.8075   0.6864   0.7204   0.5946   0.8282
## Neg Pred Value         0.9547   0.9151   0.9603   0.9234   0.9206
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2534   0.1239   0.1425   0.1002   0.1155
## Detection Prevalence   0.3138   0.1805   0.1978   0.1685   0.1394
## Balanced Accuracy      0.9031   0.7851   0.8751   0.7647   0.7995
```

Then I used the random forest. The accuracy is 99.5%. Thus, we can use the random forest to make the prediction.

```r
# random forest =============================================================


rf_fit0 = randomForest(classe ~., data = training4, ntree = 100, mtry = 7)
# varImp(rf_fit0)
varImpPlot(rf_fit0,type=2)
```



```r
fit0_pred = predict(rf_fit0, newdata = testing4, type = 'class')
confusionMatrix(fit0_pred, testing4$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    2    0    0    0
##          B    0  756    3    0    0
##          C    0    1  681    8    0
##          D    0    0    0  635    1
##          E    0    0    0    0  720
##
## Overall Statistics
##
##                Accuracy : 0.9962
##                  95% CI : (0.9937, 0.9979)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9952
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9960   0.9956   0.9876   0.9986
## Specificity            0.9993   0.9991   0.9972   0.9997   1.0000
## Pos Pred Value         0.9982   0.9960   0.9870   0.9984   1.0000
## Neg Pred Value         1.0000   0.9991   0.9991   0.9976   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1927   0.1736   0.1619   0.1835
## Detection Prevalence   0.2850   0.1935   0.1759   0.1621   0.1835
## Balanced Accuracy      0.9996   0.9975   0.9964   0.9936   0.9993
```

```
# fit0_pred
```

Prediction

Finally, the prediction is obstained.

```
# check on the pred data set ==========================================================
pred_data = predict(rf_fit0, newdata = pred, type = 'class')
pred_data
```

```
##  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
```