Practical ML Report

Haonan

November 17, 2016

Overview

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 report, we'll use a weight lifting exercise dataset, these data comes from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, and we'll use it to build a machine learning model to predict, under certain condition, the "goodness" of the efficiency an exercise performed.

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)

Load all the necessary packages we'll use later, and set the working directory properly

```
setwd("C:/Users/Haonan/Desktop/Study/R/Machine Learning Project")
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.2
```

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

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

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.3.2
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.3.2
```

```
## 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)
```

```
## Warning: package 'randomForest' was built under R version 3.3.2
```

```
## 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
```

Data Viewing and Cleaning

Load the two data set first, and save them in a data frame namely “combined” and “prediction”

Then take a look at the dimension of the data sets and have a glance of the combined dataset

```
dim(combined);dim(prediction)
```

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

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

```
View(combined); View(prediction)
```

There’re three characteristics which are obvious, first, only 20 obserevations in the prediction data set; second, there’re too many NAs in the dataset, third, the prediction dataset doesn’t have “classe” feature, which is to be predicted.

Let’s check the proportion of the NAs in the data set

```
sum(is.na(combined))/(dim(combined)[1] * dim(combined)[2])
```

```
## [1] 0.6131835
```

The result is incredible, thus we should remove NAs at first. Here we'll use a function called na_removal. This function will return the column index in a list which contains more than 80% NAs, and then we'll remove the returned columns. Code is as follows:

```
na_removal <- function(x){
    rmlist <- 0
    for (i in 1:dim(x)[2]){
        if (sum(is.na(x[,i]))/dim(x)[1] > 0.8){
            rmlist <- c(rmlist,i)
        }
    }
    return(rmlist)
}
rmlist <- na_removal(combined)
combined <- combined[,-rmlist]
prediction <- prediction[,-rmlist]
dim(combined);dim(prediction)
```

```
## [1] 19622    60
```

```
## [1] 20 60
```

Now the dimension has benn reduced, most of the unnecessary variables have been removed. We've finished the initial data cleaning part. And in the mean while, the first column of the data set is X, here I regard it as a useless number index thus to delete it: (If we do not do that we'll get the totally wrong outcome, trust me! I've spent a day to figure out the why we have to eliminate X)

```
combined <- combined[,-1]
prediction <- prediction[,-1]
```

Data Preprocessing

This step is to eliminate the near zero variables. These variables cannot make a difference of the outcome, thus remove them will improve the interpretability and model efficiency. We'll use the code as follows:

```
nzvlist <- nearZeroVar(combined)
combined <- combined[,-nzvlist]
prediction <- prediction[,-nzvlist]
dim(combined);dim(prediction)
```

```
## [1] 19622    58
```

```
## [1] 20 58
```

Unfortunately there's only one variable has been removed and still 59 left.

Modeling

The prediction set cannot made to be the test set because the observation is very less. Thus we'll separate the combined set into training and testing set.

```
set.seed(1)
inTrain <- createDataPartition(combined$classe, p = .7, list = F)
training <- combined[inTrain,]
testing <- combined[-inTrain,]
```

And first we establish a simple decision tree model to test its accuracy

```
modtree <- rpart(classe ~ ., data = training, method = "class")
confusionMatrix(testing$classe, predict(modtree, testing, type = "class"))$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.8694987      0.8349861      0.8606241      0.8780053      0.2837723
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

As we could see using a decision tree mod will get a 86.94% accuracy model which is really cool. But to make the report more prudent and precise we should use bagging method to build random forest to train the data again and apply a 5-folds corss validation:

```
fivefold <- trainControl(method = "cv", number = 5, repeats = 3)
modrf <- randomForest(classe ~ ., trControl = fivefold, data = training, ntree = 100)
confusionMatrix(testing$classe, predict(modrf, testing))$overall
```

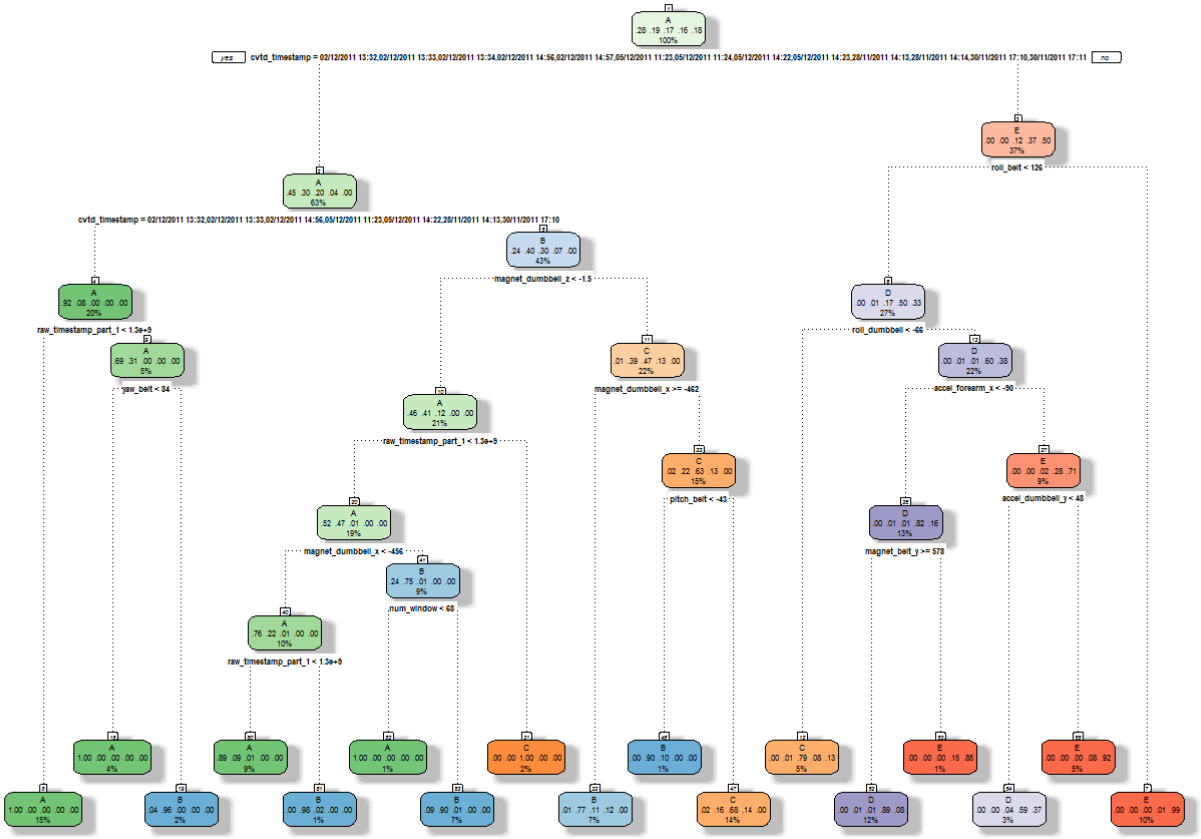
```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9988105      0.9984955      0.9975508      0.9995216      0.2844520
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

As we see the accuracy is 99.88%, which is nearly perfect. Thus we'll use the random forest model as our prediction model. Even though it takes some time to calculate the algorithm. But before the prediction, the levels of the factor variable in training set and prediction set is different thus this will cause trouble. We firstly fix this by leveling the factor variables

```
levels(prediction$user_name) <- levels(training$user_name)
levels(prediction$cvtd_timestamp) <- levels(training$cvtd_timestamp)
predict(modtree, prediction, type = "class")
```

Appendix

Plot of the decision tree:



Rattle 2016-Nov-23 13:22:11 Haonan