PMLproject

Jie Zhang

December 6, 2016

Summary

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, our 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)

```
library(caret)
library(rattle)
library(randomForest)
library(rpart)
```

Getting and Cleaning Data

Get the training and testing data from the given urls and read in them.

```
##read in training and testing data sets
training <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.c
sv", na.strings = c("NA","#DIV/0!",""))
testing <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.cs
v", na.strings=c("NA","#DIV/0!",""))
```

The origianl training set has 19622 observations of 160 variables.

```
intrain <- createDataPartition(training$classe, p = 0.7, list = FALSE)
subtrain <- training[intrain, ]
subtest <- training[-intrain, ]

##Remove NearZeroVariance
nzv <- nearZeroVar(subtrain)
subtrain <- subtrain[, -nzv]
subtest <- subtest[, -nzv]

##Remove the variables with more than 95% "NA" in the training set.
AllNA    <- sapply(subtrain, function(x) mean(is.na(x))) > 0.95
subtrain <- subtrain[, AllNA==FALSE]
subtest  <- subtest[, AllNA==FALSE]

## Remove variables from 1 to 5
subtrain <- subtrain[, -(1:5)]
subtest <- subtest[, - (1:5)]

dim(subtrain)
```

```
## [1] 13737    54
```

```
dim(subtest)
```

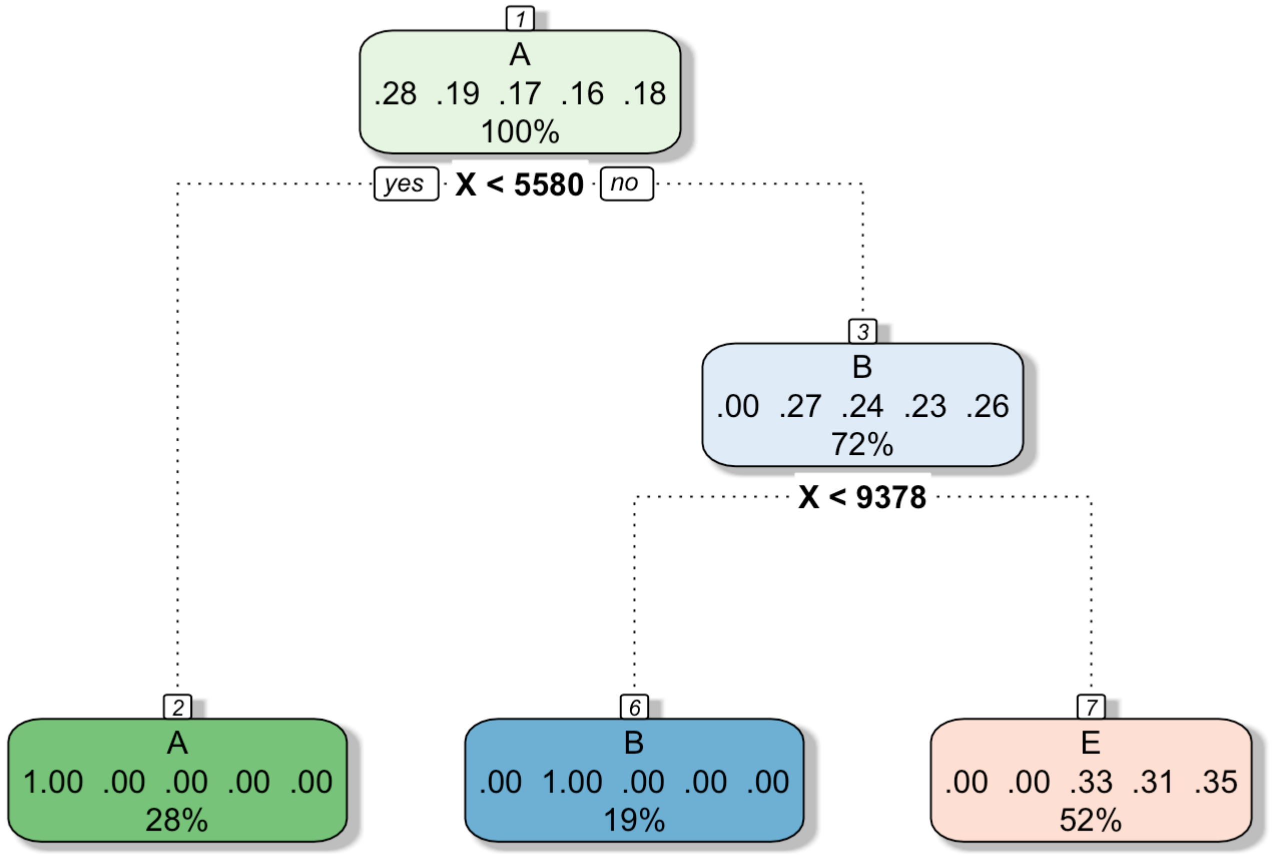
```
## [1] 5885   54
```

Model Building

Before build any model, we will first devide the training data set into `subtrain` and `subtest` using the `classe` variable. By doing this, we can use the `subtest` data set as the cross validation data. We will train the model on the `subtrain` data set and predict the result on the `subtest` data set, using both random forest and boosting. Therefore, we can see the accuries of the two models, and the

First of all, we will fit a tree based model and see how does the model works.

```
seet.seed = 724
rpartfit <- train(classe ~., data = subtrain, method = "rpart",
                  trControl = trainControl(method = "cv", number = 5))
fancyRpartPlot(rpartfit$finalModel)
```



Rattle 2016-Dec-09 10:26:45 jiezhang

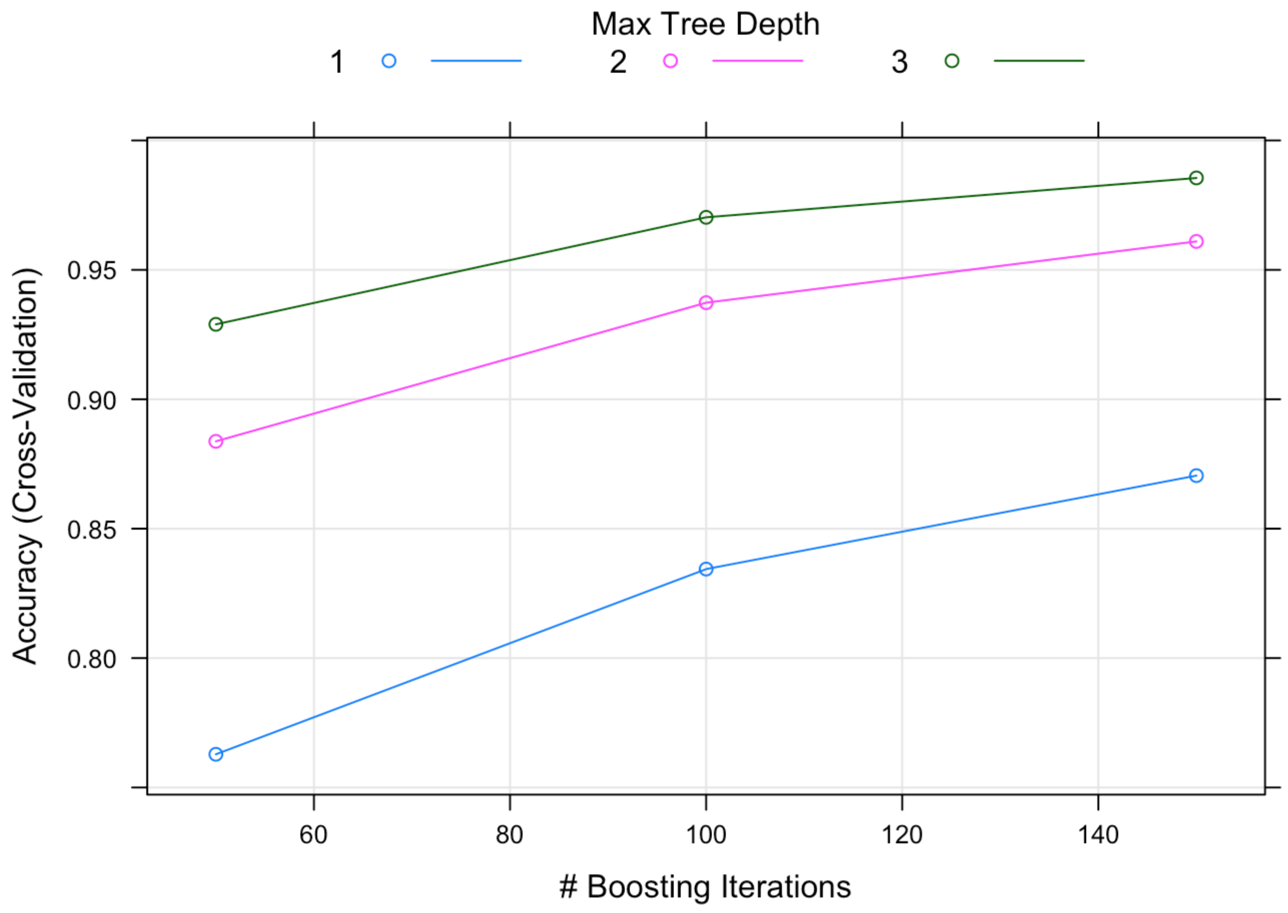
```
rpartpred <- predict(rpartfit, subtest)
## check accuracy
confusionMatrix(rpartpred, subtest$classe)$overall[1]
```

```
##  Accuracy
## 0.6618522
```

The accuracy of the tree based model came out to be 0.66, which is pretty low. Thus, we will train the model using random forest and boosting, and see how the two model works.

```
##Train the subtrain data using Random Forest
set.seed(724)
RFfit <- train(classe ~., method = "rf", data = subtrain,
               trControl = trainControl(method = "cv", number = 3, verboseIter = FALSE))
## Predcit
RFpred <- predict(RFfit, subtest)

##Train the subtrain data using Boosting
gbmfit <- train(classe ~., method = "gbm", data = subtrain,
            trControl = trainControl(method = "cv", repeats = 1, number = 5))
gbmpred <- predict(gbmfit, subtest)
gbmpred
plot(gbmfit)
```



```
##Extract accuracies for random forest and boosting
confusionMatrix(RFpred, subtest$classe)$overall[1]
```

```
##  Accuracy
## 0.9986406
```

```
confusionMatrix(gbmpred, subtest$classe)$overall[1]
```

```
## Accuracy
## 0.986576
```

Both of the model predit pretty well with a verh high accuracy value. The random forest prediction accuracy is 0.9986, so the out of sample error is 0.0014. And the boosting prediction accuracy is just a little lower, 0.9865 leave the sample error to 0.0135.

Prediction on the testing data set

Because the random forest model has a lightly higher prediction accuracy, we will use the random forest model to predic on the testing data.

```
Testpred <- predict(RFfit, testing)
Testpred
```

```
##  [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
```