Machine-Learning-Project

Ahmed Sherif

September 25, 2018

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 Processing

This section will download the data from the web if it is not already exist in the working directory

```
destfile <- "./pml-training.csv"
fileURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

# if the data file is not exist then download it
if(!file.exists(destfile)){
   download.file(fileURL, destfile=destfile, method="auto")
}

destfile <- "./pml-testing.csv"
fileURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# if the data file is not exist then download it
if(!file.exists(destfile)){
   download.file(fileURL, destfile=destfile, method="auto")
}

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


dim(training)
```

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

```
dim(testing)
```

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

This section will remove the first seven columns from the data sets beacause they are irrelevant to the experiment

```
set.seed(33833)

# remove first 7 columns, they don't affect the analysis
training <- training[, -c(1:7)]
testing <- testing[, -c(1:7)]
```

This section fills the missing values in the datasets, the missing value will be replaced by the mean of the column in case the column type is not a factor, otherwise it will be replaced by the most frequent level of the factor

```
fill_missing_values <- function(data){
  # for each column
  for(i in 1:ncol(data)){

      if(!is.factor(data[,i])){

        # if the column is not a factor then
        # replace NA values with the mean of the column or 0 if the column contains only NA values
        mean <- mean(data[,i], na.rm=TRUE)
        if(!is.na(mean))
        {
          data[is.na(data[,i]),i] <- mean
        }
        else
        {
          data[is.na(data[,i]),i] <- 0
        }

      }
      else
      {
        # if the column is a factor then
        # replace NA values with the most frequest Level in the column
        tt <- table(data[,i])
        data[is.na(data[,i]),i] <-  names(tt[which.max(tt)])
      }

  }
  return(data)
}

# replace NA with mean of columns
training <- fill_missing_values(training)
testing <- fill_missing_values(testing)
```

This section splits the Training data into two sets for validation purposes

```r
library(caret)
```

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

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

```r
# split the training data for validation
inTrain = createDataPartition(training$classe, p = 3/4)[[1]]
myTraining = training[ inTrain,]
myValidation = training[-inTrain,]
```

This section creates a model of type "Random Forest" starting with 500 trees

```r
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

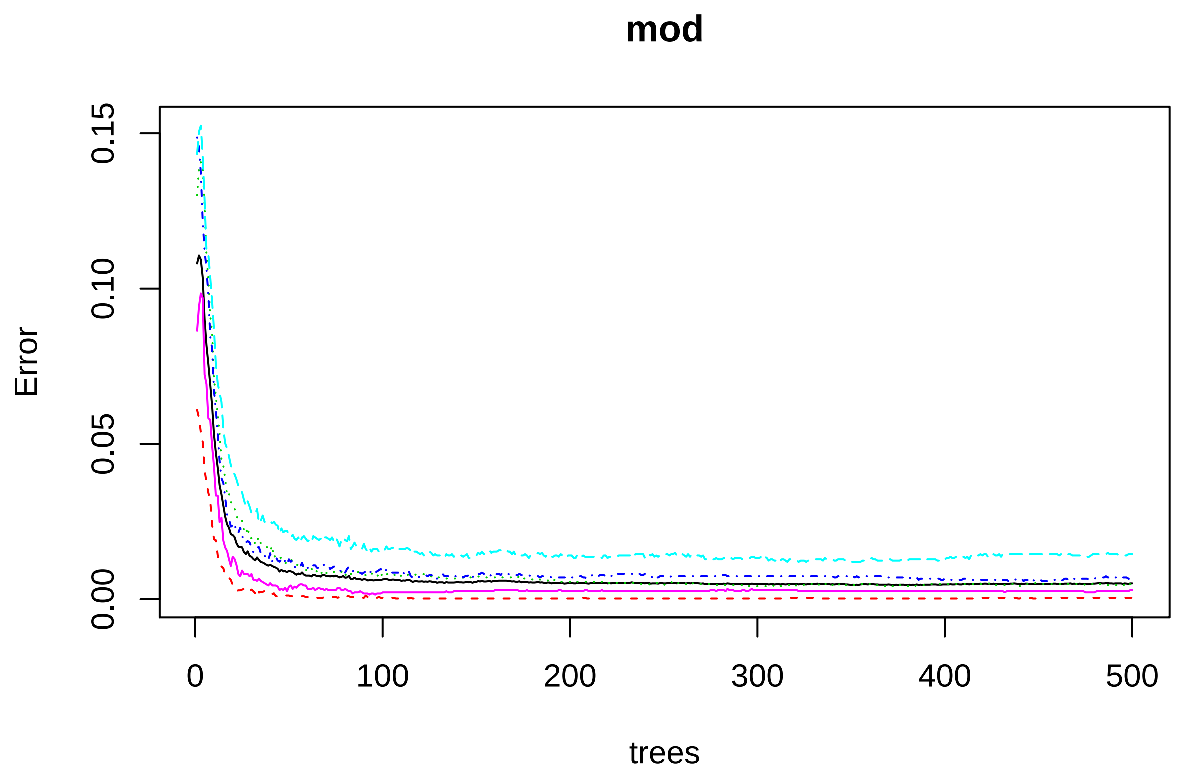
```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
# fit a random forest mod to our training set
mod <- randomForest(classe ~ ., data = myTraining, na.action = na.exclude, ntree=500)
pred <- predict(mod, myValidation)

# accuracy
confusionMatrix(pred, myValidation$classe)$overall[1]
```

```
##  Accuracy
## 0.9959217
```

```r
# plot the mod
plot(mod)
```



Based on the previous plot, we found that we can reduce the number of trees while having the same magnitude of error, so we will fit a gain with 100 trees

```r
library(randomForest)

# reduce the number of trees
mod <- randomForest(classe ~ ., data = myTraining, na.action = na.exclude, ntree=100)
pred <- predict(mod, myValidation)

# accuracy
pred_accuracy <- confusionMatrix(pred, myValidation$classe)$overall[1]
pred_accuracy
```

```
##  Accuracy
## 0.9949021
```

In this section, we will remove the features that have variance near zero and we will fit a model again, then we will compare the accuracies

```r
library(caret)

# remove the columns with variance near to zero
near.zero.var.cols <- names(training)[nearZeroVar(training)]
```

```r
length(near.zero.var.cols)
```

```
## [1] 100
```

We found that the number of near-zero-variance columns is 100, we will fit again without these columns

```r
library(randomForest)
library(caret)

# remove the  near-zero-variance columns
training <- training [,!(colnames(training) %in% near.zero.var.cols)]
testing <- testing [,!(colnames(testing) %in% near.zero.var.cols)]

# split the training data
inTrain = createDataPartition(training$classe, p = 3/4)[[1]]
myTraining = training[ inTrain,]
myValidation = training[-inTrain,]

# fit a random forest mod to our training set
mod2 <- randomForest(classe ~ ., data = myTraining, na.action = na.exclude, ntree=100)
pred2 <- predict(mod2, myValidation)

# accuracy
confusionMatrix(pred2, myValidation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    6    0    0    0
##          B    0  939    4    0    0
##          C    0    4  851    9    0
##          D    0    0    0  795    5
##          E    0    0    0    0  896
##
## Overall Statistics
##
##                Accuracy : 0.9943
##                  95% CI : (0.9918, 0.9962)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9928
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9895   0.9953   0.9888   0.9945
## Specificity            0.9983   0.9990   0.9968   0.9988   1.0000
## Pos Pred Value         0.9957   0.9958   0.9850   0.9938   1.0000
## Neg Pred Value         1.0000   0.9975   0.9990   0.9978   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1915   0.1735   0.1621   0.1827
## Detection Prevalence   0.2857   0.1923   0.1762   0.1631   0.1827
## Balanced Accuracy      0.9991   0.9942   0.9961   0.9938   0.9972
```

```
pred2_accuracy <- confusionMatrix(pred2, myValidation$classe)$overall[1]
pred2_accuracy
```

```
##  Accuracy
## 0.9942904
```

Conclusion

The accuracy while we use all the columns is 0.9949021, and the accuracy when we remove the near-zero-variance columns is 0.9942904, We can find the accuracy is not affected greatly after removing the near-zero-variance columns, so we will choose that last mod to predict the Testing data, because this mod improves the performance

```
pred_testing <- predict(mod2, testing)
pred_testing
```

```
##  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
```