Final Assignment_PracticalML

Piyush

2022-11-02

Introduction

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://web.archive.org/web/20161224072740/http://groupware.les.inf.puc-rio.br/har

The training and test data for this project are available in this two url's:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Data Loading and Processing

```
set.seed(123)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(rpart)
library(knitr)
library(lattice)
library(ggplot2)
library(randomForest)
```

```
## randomForest 4.7-1.1

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

##
## Attaching package: 'randomForest'

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

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

```r
library(rattle)
```

```
## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

##
## Attaching package: 'rattle'

## The following object is masked from 'package:randomForest':
##
##     importance
```

```r
library(rpart.plot)
library(RColorBrewer)
library(cowplot)

totalset<-read.csv("~/R files/pml-training.csv",header = TRUE)
testset<-read.csv("~/R files/pml-testing.csv",header = TRUE)
```

Cleaning and Exploratory Data Analysis

```r
#Convert empty values with NA
totalset[totalset == ""] <- NA
testset[testset == ""] <- NA

#Now we will be removing those columns where most values are NA
totalset<-totalset[, which(colMeans(!is.na(totalset)) > 0.5)]
testset<-testset[, which(colMeans(!is.na(testset)) > 0.5)]

#Now we will remove initial ID and identification variables from the set

totalset <- totalset[ , -(1:6)]
testset  <- testset [ , -(1:6)]
```

Train-Validation Split

```r
datapartition<-createDataPartition(totalset$classe,p=0.75,list = FALSE)
trainingset<-totalset[datapartition,]
cvset<-totalset[-datapartition,]
```

Model Testing

Here, we will try different models and will select the best one based on the its accuracy on Validation set.

Decision Tree

```r
#First we will try simple decison tree

DT_Model<-rpart(classe ~ .,data=trainingset,method="class")
DT_outofsample<-predict(DT_Model,trainingset,type = 'class')
confusionMatrix(table(DT_outofsample, trainingset$classe))
```

```
## Confusion Matrix and Statistics
##
##
## DT_outofsample    A    B    C    D    E
##              A 3714  371  102  111   29
##              B  128 1724  120   72   69
```

```
##              C   76  244 2083  419  102
##              D  121  241   56 1423  190
##              E  146  268  206  387 2316
##
## Overall Statistics
##
##                Accuracy : 0.765
##                  95% CI : (0.7581, 0.7719)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7025
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8875   0.6053   0.8115  0.58997   0.8559
## Specificity            0.9418   0.9672   0.9308  0.95059   0.9162
## Pos Pred Value         0.8583   0.8159   0.7124  0.70064   0.6970
## Neg Pred Value         0.9547   0.9108   0.9590  0.92205   0.9658
## Prevalence             0.2843   0.1935   0.1744  0.16388   0.1839
## Detection Rate         0.2523   0.1171   0.1415  0.09668   0.1574
## Detection Prevalence   0.2940   0.1436   0.1987  0.13799   0.2258
## Balanced Accuracy      0.9146   0.7863   0.8711  0.77028   0.8860
```

```
DT_predict<-predict(DT_Model,cvset,type = 'class')
confusionMatrix(table(DT_predict, cvset$classe))
```

```
## Confusion Matrix and Statistics
##
##
## DT_predict    A    B    C    D    E
##          A 1213  128   37   40    9
##          B   55  575   27   24   16
##          C   29   77  688  161   46
##          D   49   81   16  437   63
##          E   49   88   87  142  767
##
## Overall Statistics
##
##                Accuracy : 0.7504
##                  95% CI : (0.738, 0.7625)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6841
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.8695   0.6059   0.8047  0.54353   0.8513
## Specificity            0.9390   0.9692   0.9227  0.94902   0.9086
## Pos Pred Value         0.8500   0.8250   0.6873  0.67647   0.6770
## Neg Pred Value         0.9477   0.9111   0.9572  0.91381   0.9645
## Prevalence             0.2845   0.1935   0.1743  0.16395   0.1837
## Detection Rate         0.2473   0.1173   0.1403  0.08911   0.1564
## Detection Prevalence   0.2910   0.1421   0.2041  0.13173   0.2310
## Balanced Accuracy      0.9043   0.7875   0.8637  0.74628   0.8799
```

So we are getting around 76.5% accuracy on the trainingset and 75% after testing on Validation set from this simple decision tree.

Now we will try to form the Random forest model

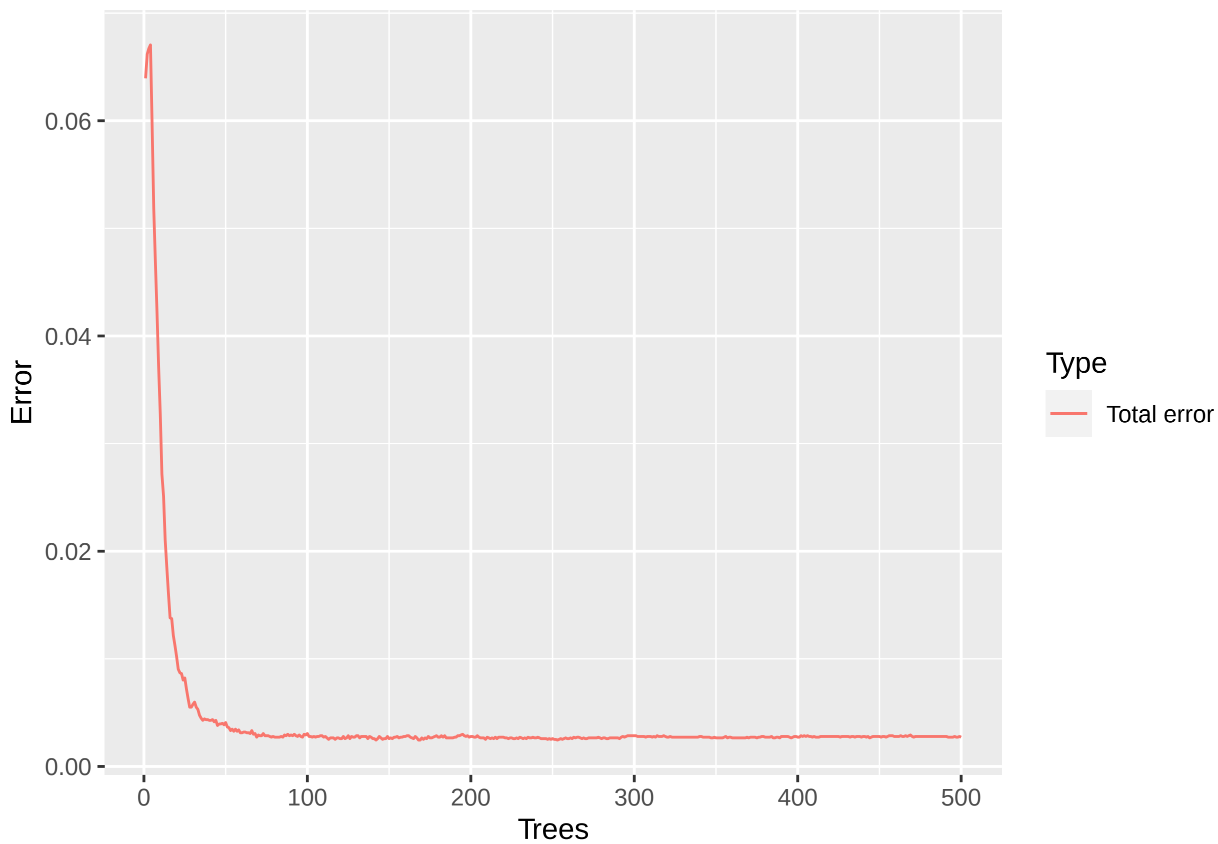
```
set.seed(1967)
rf_Model<-randomForest(as.factor(classe) ~ ., data = trainingset,proximity=TRUE)
rf_Model
```

```
##
## Call:
##  randomForest(formula = as.factor(classe) ~ ., data = trainingset,      proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.27%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4184    0    0    0    1 0.0002389486
## B    4 2841    3    0    0 0.0024578652
## C    0    6 2559    2    0 0.0031164784
## D    0    0   19 2393    0 0.0078772803
## E    0    0    0    5 2701 0.0018477458
```

Now we will plot the error data vs the number of trees

```
oob.error.data<-data.frame(Trees=rep(1:nrow(rf_Model$err.rate),times=1),Type=rep(c("Total error"),each=

ggplot(data=oob.error.data,aes(x=Trees,y=Error))+geom_line(aes(color=Type))
```



So as we can see that after total no. of trees crossed 100, there is not any significant change in total error so we can set optimal number of trees at 100.

Now we can experiment with different number of splits to be considered at each node.

```
rftrial<-vector(length = 12)

for (i in 4:15){
  tempmodel<-randomForest(as.factor(classe) ~ ., data = trainingset,mtry=i,ntree=100)
  rftrial[i]<-tempmodel$err.rate[nrow(tempmodel$err.rate),1]
}

rftrial
```

```
##  [1] 0.000000000 0.000000000 0.000000000 0.004756081 0.004008697 0.003261313
##  [7] 0.003465145 0.002649817 0.002717761 0.002242152 0.002445985 0.002785705
## [13] 0.002785705 0.002921593 0.002378040
```

So we can note that keeping no. of splits at 10 will be our best solution So finally, we will select this combination of ntree=100 and mtry=10 and will test this model on our cvset.

Random Forest Final Model

```r
rf_finalModel<-randomForest(as.factor(classe) ~ ., data = trainingset,mtry=10,ntree=100,proximity=TRUE)

predict_trainingdata<-predict(rf_finalModel, newdata = trainingset)
confusionMatrix(table(predict_trainingdata, trainingset$classe))
```

```
## Confusion Matrix and Statistics
##
##
## predict_trainingdata    A    B    C    D    E
##                    A 4185    0    0    0    0
##                    B    0 2848    0    0    0
##                    C    0    0 2567    0    0
##                    D    0    0    0 2412    0
##                    E    0    0    0    0 2706
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```r
predict_RF <- predict(rf_finalModel, newdata = cvset)
confusionMatrix(table(predict_RF, cvset$classe))
```

```
## Confusion Matrix and Statistics
##
##
## predict_RF    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    5    0    0
##          C    0    0  850    1    0
##          D    0    0    0  803    2
##          E    0    0    0    0  899
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9984
##                  95% CI : (0.9968, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9942   0.9988   0.9978
## Specificity            1.0000   0.9987   0.9998   0.9995   1.0000
## Pos Pred Value         1.0000   0.9948   0.9988   0.9975   1.0000
## Neg Pred Value         1.0000   1.0000   0.9988   0.9998   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1733   0.1637   0.1833
## Detection Prevalence   0.2845   0.1945   0.1735   0.1642   0.1833
## Balanced Accuracy      1.0000   0.9994   0.9970   0.9991   0.9989
```

So we are getting 100% accuracy on the trainingdata and 99.9% accuracy with this model on our CVset which is really good.So, we will finalise this as our final model.

Prediction on Testing Set

Now, we will use this model to get the prediction on our test data.

```
test_predict<-predict(rf_finalModel, newdata = testset[,-54])
test_predict
```

```
##  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
```