Practical Machine Learning

Umar Ayoub

2020-02-25

```
# Execution Summary
# First loading in the data and performing exploritory analysis
# I load the required package/data for data cleaning and model fitting,
# In the Model sectiton,
# I fit the data using first using decision trees then using
# random forest with cross validation.
# Finally, in the Prediction section, I use the model to predict the test
set.

# Loading in the data
train <- read.csv(file = file.choose())
test <- read.csv(file = file.choose(), header = T)


# Exploratory analysis
summary(train)
```

```
##        X            user_name    raw_timestamp_part_1 raw_timestamp_part_2
##  Min.   :    1   adelmo  :3892   Min.   :1.322e+09    Min.   :   294
##  1st Qu.: 4906   carlitos:3112   1st Qu.:1.323e+09    1st Qu.:252912
##  Median : 9812   charles :3536   Median :1.323e+09    Median :496380
##  Mean   : 9812   eurico  :3070   Mean   :1.323e+09    Mean   :500656
##  3rd Qu.:14717   jeremy  :3402   3rd Qu.:1.323e+09    3rd Qu.:751891
##  Max.   :19622   pedro   :2610   Max.   :1.323e+09    Max.   :998801
##
##           cvtd_timestamp  new_window    num_window      roll_belt
##  28/11/2011 14:14: 1498   no :19216   Min.   :  1.0   Min.   :-28.90
##  05/12/2011 11:24: 1497   yes:  406   1st Qu.:222.0   1st Qu.:  1.10
##  30/11/2011 17:11: 1440               Median :424.0   Median :113.00
##  05/12/2011 11:25: 1425               Mean   :430.6   Mean   : 64.41
##  02/12/2011 14:57: 1380               3rd Qu.:644.0   3rd Qu.:123.00
##  02/12/2011 13:34: 1375               Max.   :864.0   Max.   :162.00
##  (Other)         :11007
##    pitch_belt          yaw_belt       total_accel_belt kurtosis_roll_belt
##  Min.   :-55.8000   Min.   :-180.00   Min.   : 0.00              :19216
##  1st Qu.:  1.7600   1st Qu.: -88.30   1st Qu.: 3.00    #DIV/0!   :   10
##  Median :  5.2800   Median : -13.00   Median :17.00    -1.908453:    2
##  Mean   :  0.3053   Mean   : -11.21   Mean   :11.31    -0.016850:    1
##  3rd Qu.: 14.9000   3rd Qu.:  12.90   3rd Qu.:18.00    -0.021024:    1
##  Max.   : 60.3000   Max.   : 179.00   Max.   :29.00    -0.025513:    1
##  (Other)  :  391
## . . . .
```

```r
dim(train)
```

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

```r
str(train)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2
2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231
1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232
...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328
304277 368296 440390 484323 484434 ...
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9
9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1
1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1
1 1 1 1 1 1 1 1 ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1
1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1
1 1 1 1 1 1 1 1 ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...

....
```

```r
# removing variables that are almost always NA
indCol <- which(colSums(is.na(train) | train=="")>0.9*dim(train)[1])
indCol
```

```
##       kurtosis_roll_belt      kurtosis_picth_belt        kurtosis_yaw_belt
##                       12                       13                       14
##       skewness_roll_belt     skewness_roll_belt.1        skewness_yaw_belt
##                       15                       16                       17
##            max_roll_belt           max_picth_belt             max_yaw_belt
```

```
##                      18                      19                      20
```

……

```
trainc <- train[,-indCol]

# The first seven columns have information on the people who took the test,
# Removing those 7 columns
trainc <- trainc[,-c(1:7)]

dim(trainc)
```

```
## [1] 19622    53
```

```
# Exploratory analysis
summary(trainc)
```

```
##    roll_belt        pitch_belt          yaw_belt       total_accel_belt
##  Min.   :-28.90   Min.   :-55.8000   Min.   :-180.00   Min.   : 0.00
##  1st Qu.:  1.10   1st Qu.:  1.7600   1st Qu.: -88.30   1st Qu.: 3.00
##  Median :113.00   Median :  5.2800   Median : -13.00   Median :17.00
##  Mean   : 64.41   Mean   :  0.3053   Mean   : -11.21   Mean   :11.31
##  3rd Qu.:123.00   3rd Qu.: 14.9000   3rd Qu.:  12.90   3rd Qu.:18.00
##  Max.   :162.00   Max.   : 60.3000   Max.   : 179.00   Max.   :29.00
##   gyros_belt_x        gyros_belt_y       gyros_belt_z      accel_belt_x
##  Min.   :-1.040000   Min.   :-0.64000   Min.   :-1.4600   Min.   :-120.000
##  1st Qu.:-0.030000   1st Qu.: 0.00000   1st Qu.:-0.2000   1st Qu.: -21.000
##  Median : 0.030000   Median : 0.02000   Median :-0.1000   Median : -15.000
##  Mean   :-0.005592   Mean   : 0.03959   Mean   :-0.1305   Mean   :  -5.595
##  3rd Qu.: 0.110000   3rd Qu.: 0.11000   3rd Qu.:-0.0200   3rd Qu.:  -5.000
##  Max.   : 2.220000   Max.   : 0.64000   Max.   : 1.6200   Max.   :  85.000
```

....

```
str(trainc)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43
1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16
8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -
94.4 -94.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03
...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -
0.02 -0.02 0 ...
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : int  22 22 23 21 24 21 21 21 24 22 ...
```

```
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : int  599 608 600 604 600 603 599 603 602 609 ...
…….
```

```
dim(trainc)
```

```
## [1] 19622    53
```

```
# performing the same function on test set
indCol <- which(colSums(is.na(test) | test=="") > 0.9*dim(test)[1])
testc <- test[, -indCol]
testc <- testc[, -1]

###
dim(testc)
```

```
## [1] 20 59
```

```
str(testc)
```

```
## 'data.frame':    20 obs. of  59 variables:
##  $ user_name           : Factor w/ 6 levels "adelmo","carlitos",..: 6 5 5
1 4 5 5 5 2 3 ...
##  $ raw_timestamp_part_1: int  1323095002 1322673067 1322673075 1322832789
1322489635 1322673149 1322673128 1322673076 1323084240 1322837822 ...
##  $ raw_timestamp_part_2: int  868349 778725 342967 560311 814776 510661
766645 54671 916313 384285 ...
##  $ cvtd_timestamp      : Factor w/ 11 levels "02/12/2011 13:33",..: 5 10
10 1 6 11 11 10 3 2 ...
##  $ new_window          : Factor w/ 1 level "no": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window          : int  74 431 439 194 235 504 485 440 323 664 ...
##  $ roll_belt           : num  123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.93
114 ...
##  $ pitch_belt          : num  27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.72
22.4 ...
##  $ yaw_belt            : num  -4.75 -88.9 -88.5 162 -88.6 -87.7 -87.3 -
88.5 -93.7 -13.1 ...
##  $ total_accel_belt    : int  20 4 5 17 3 4 4 4 4 18 ...
##  $ gyros_belt_x        : num  -0.5 -0.06 0.05 0.11 0.03 0.1 -0.06 -0.18
0.1 0.14 ...
##  $ gyros_belt_y        : num  -0.02 -0.02 0.02 0.11 0.02 0.05 0 -0.02 0
0.11 ...
##  $ gyros_belt_z        : num  -0.46 -0.07 0.03 -0.16 0 -0.13 0 -0.03 -0.02
-0.16 ...
…….
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.2
```

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

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

```
# Spliting the data into training and test set
ntrian <- createDataPartition(trainc$classe, p = 0.75, list = F)
traindata <- trainc[ntrian, ]
testdata <- trainc[-ntrian, ]
```

```
dim(traindata)
```

```
## [1] 14718    53
```

```
                                  # MODEL SELECTION
# Prediction with classification trees
# Creating the model
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
library(rpart)
library(rattle)
```

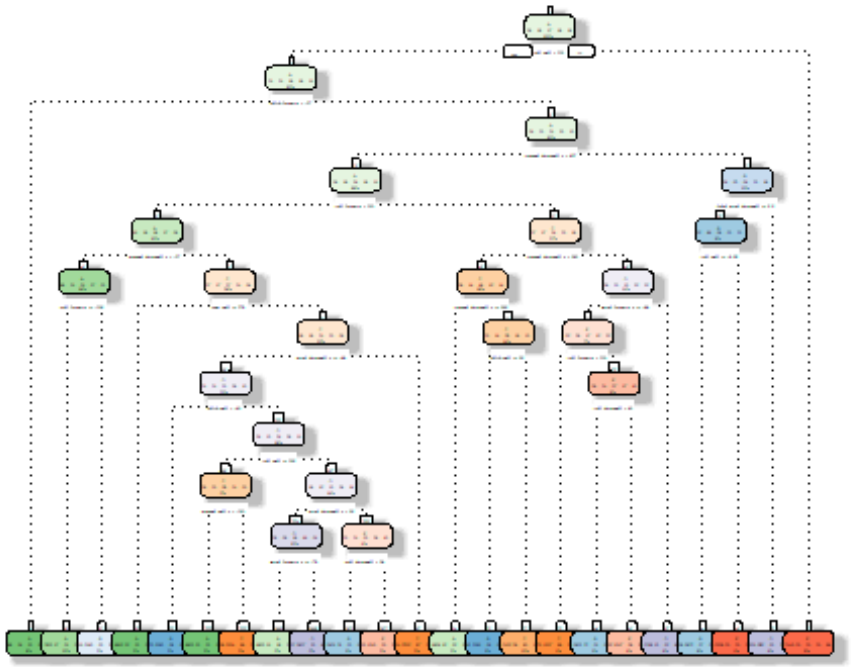
```
## Warning: package 'rattle' was built under R version 3.6.2
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
set.seed(0311)
dtmdl1 <- rpart(classe ~ ., data=traindata, method="class")
```

```
# ploting the tree as dendogra
fancyRpartPlot(dtmdl1)
```

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



Rattle 2020-Feb-25 14:07:08 Umar

```
# # Testing our model on development data set
predtreemdl1 <- predict(dtmdl1, testdata, type = "class")
conftree <- confusionMatrix(predtreemdl1, testdata$classe)
conftree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1269  188   61  119   48
##          B   48  582   53   52   63
##          C   32   62  665  111   84
##          D   29   73   55  433   45
##          E   17   44   21   89  661
##
## Overall Statistics
##
##                Accuracy : 0.7361
##                  95% CI : (0.7236, 0.7484)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6636
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9097   0.6133   0.7778   0.5386   0.7336
## Specificity            0.8814   0.9454   0.9286   0.9507   0.9573
## Pos Pred Value         0.7531   0.7293   0.6971   0.6819   0.7945
## Neg Pred Value         0.9609   0.9106   0.9519   0.9131   0.9411
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2588   0.1187   0.1356   0.0883   0.1348
## Detection Prevalence   0.3436   0.1627   0.1945   0.1295   0.1697
## Balanced Accuracy      0.8956   0.7793   0.8532   0.7446   0.8455

# This gives us an accuracy of just 74% which is not good enough

                              # Random Forest
library(doSNOW)

## Loading required package: foreach

## Loading required package: iterators

## Loading required package: snow

# By default Rstudio provides 1 core
# we need more than one core for training the Model
cl <- makeCluster(4, type = "SOCK")

# have to register < R does not have auto register for dosnow
registerDoSNOW(cl)

# using 5 fold cross validation to selet best tuning paramets
fitControl <- trainControl(method="cv", number=5, verboseIter=F)

# Training the model
set.seed(1103)
mdl <- train(classe ~ ., data=traindata, method="rf",
             trControl=fitControl)


stopCluster(cl)

# Checking the final model
mdl$finalModel

##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.77%
```

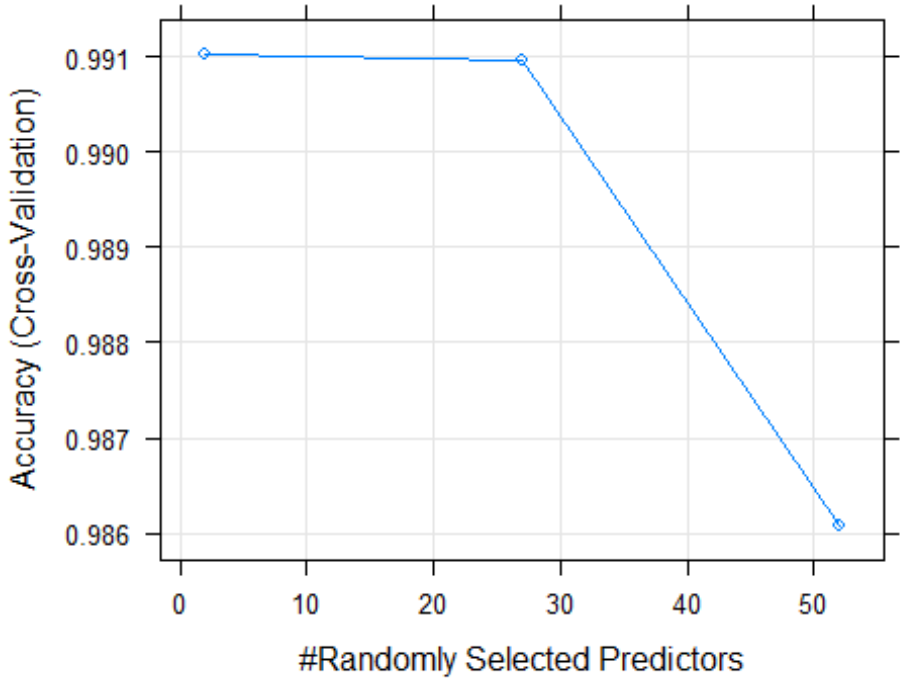
```
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4182    3    0    0    0 0.0007168459
## B   18 2820   10    0    0 0.0098314607
## C    0   24 2539    4    0 0.0109076743
## D    0    0   43 2365    4 0.0194859038
## E    0    0    1    6 2699 0.0025868441
```

```r
# Testing our model on development data set
predrf1 <- predict(mdl, newdata=testdata)
conf <- confusionMatrix(predrf1, testdata$classe)
conf
```

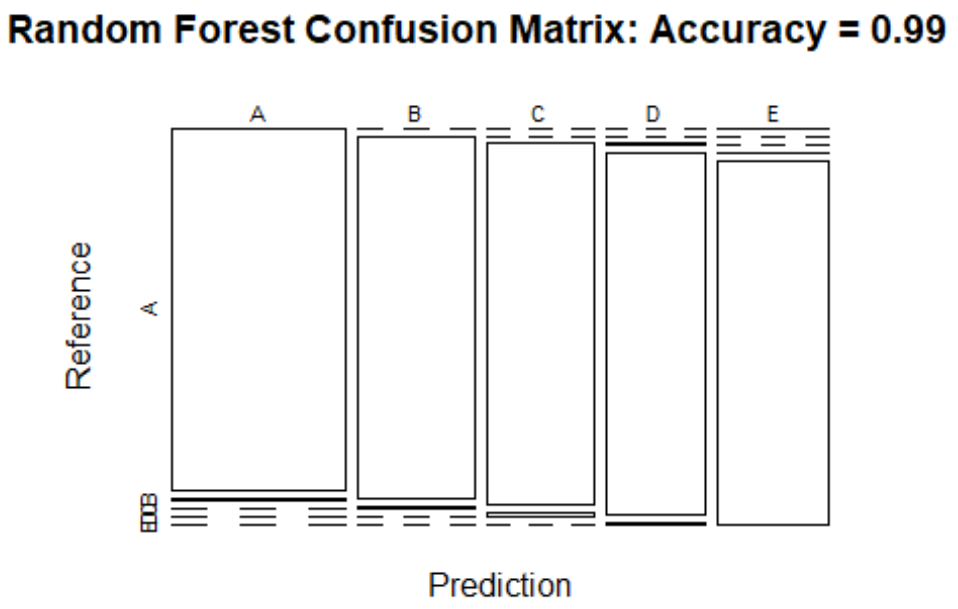
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394   11    0    0    0
##          B    0  938    3    0    0
##          C    0    0  851   11    0
##          D    0    0    1  792    2
##          E    1    0    0    1  899
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9913, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9884   0.9953   0.9851   0.9978
## Specificity            0.9969   0.9992   0.9973   0.9993   0.9995
## Pos Pred Value         0.9922   0.9968   0.9872   0.9962   0.9978
## Neg Pred Value         0.9997   0.9972   0.9990   0.9971   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1913   0.1735   0.1615   0.1833
## Detection Prevalence   0.2865   0.1919   0.1758   0.1621   0.1837
## Balanced Accuracy      0.9981   0.9938   0.9963   0.9922   0.9986
```

```r
# The accuracy rate using the random forest is very high
# therefore the out-of-sample-error is negligible.
# But it might be due to overfitting.
```

```r
# Ploting the model for errors
plot(mdl)
```




```r
plot(conf$table, col = conf$byClass,
     main = paste("Random Forest Confusion Matrix: Accuracy =",
                  round(conf$overall['Accuracy'], 4)))
```




```
# Predicting on our test data set
Final <- predict(mdl, newdata=test)

Final

##  [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
```