Practical Machine Learning Project

Biswajit Chowdhury

24/06/2019

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.2     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(rpart)
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:dplyr':
##
##     combine

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

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

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

download training and test data

```
traindata <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")


testdata<- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
```

inspect the data

```
dim(traindata)
```

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

```
dim(testdata)
```

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

Split the data into training and test set

```
set.seed(123)
training.samples <- traindata$classe %>%
  createDataPartition(p = 0.7, list = FALSE)
train.data  <- traindata[training.samples, ]
test.data <- traindata[-training.samples, ]

dim(train.data)
```

```
## [1] 13737   160
```

```
dim(test.data)
```

```
## [1] 5885  160
```

tidy the dataset for further analysis

```
# remove the variables that contains missing values.
train.data <- train.data[ , colSums(is.na(train.data)) == 0] # selecting only columns that do not have
test.data <- test.data[ , colSums(is.na(test.data)) == 0]

train.data <- train.data[, -nearZeroVar(train.data)] # removing columns with near zero variance
test.data  <- test.data [, -nearZeroVar(test.data )]

train.data <- train.data[ , -c(1:5)] # removing variables for row number, username, and timestamp
test.data  <- test.data [ , -c(1:5)]

dim(train.data)
```

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

```
dim(test.data)
```

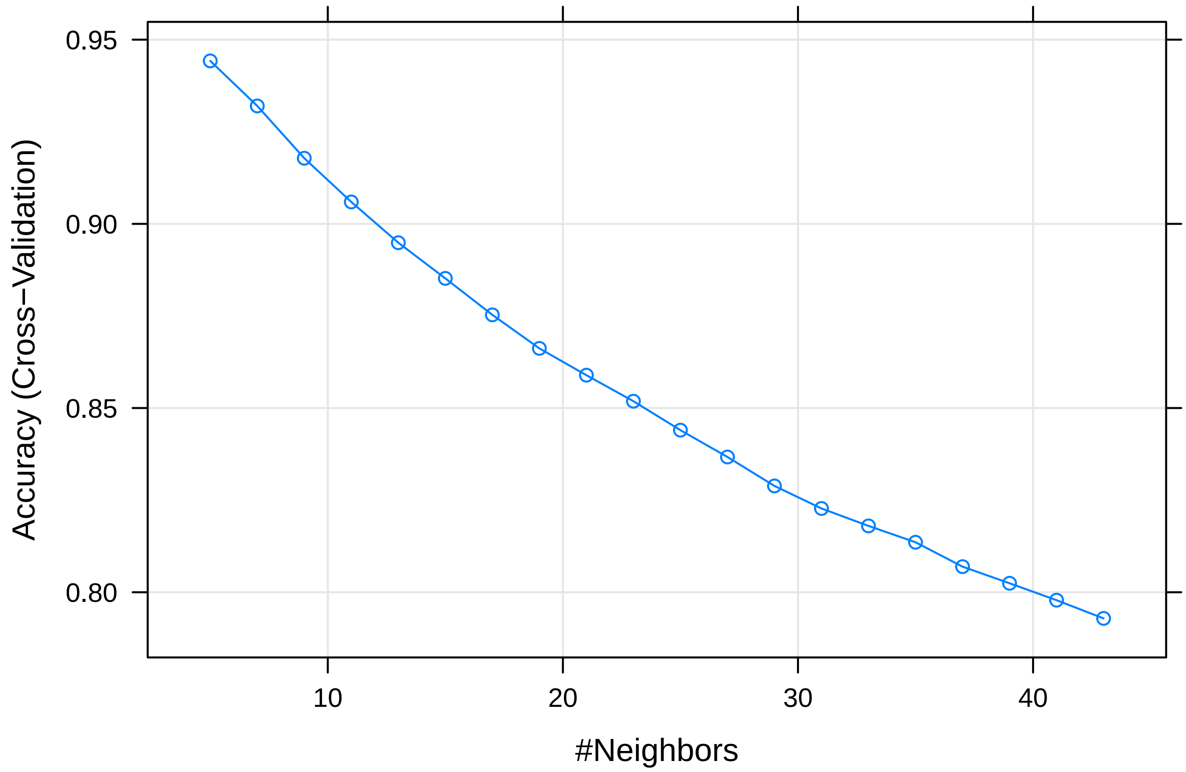
```
## [1] 5885   54
```

We randomly select three algorith to determine the best model

KNN Algorithm

```
model1 <- train(classe~., data=train.data, method="knn", trControl=trainControl("cv", number = 3), preP

#plot model accuracy vs different values of K
plot(model1)
```




```r
# print the best tunning parameter K that maximize model accuracy
model1$ bestTune
```

```
##   k
## 1 5
```

```r
# make prediction on the test data
predicted.classes<- model1 %>% predict (test.data)
head(predicted.classes)
```

```
## [1] A A A A A A
## Levels: A B C D E
```

```r
# compute model accuracy rate
mean(predicted.classes==test.data$classe)
```

```
## [1] 0.9626168
```

Random Forest Model

```r
set.seed(123)
model2<- train(classe~., data=train.data, method="rf", trControl=trainControl("cv", number=3), importanc
# Best tuning parameter
model2$bestTune
```

```
##   mtry
## 2   27
```

```r
#final model
model2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.24%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    7 2648    2    1    0 0.0037622272
## C    0    5 2391    0    0 0.0020868114
## D    0    0   11 2241    0 0.0048845471
## E    0    1    0    4 2520 0.0019801980
```

```r
# importance of each variable
importance(model2$finalModel)
```

```
##                              A        B         C         D         E
## num_window           55.266480 79.59999 84.651336 56.857220 66.548431
## roll_belt            60.161272 72.69532 63.099240 62.154163 61.831912
## pitch_belt           27.374778 53.65065 41.827146 36.636683 35.291092
## yaw_belt             35.517424 45.59135 48.029272 50.750954 36.367088
## total_accel_belt      9.249371 12.33755  9.348368 11.278583 10.834864
## gyros_belt_x         17.647787 12.30895 15.362293  8.832274 14.869552
## gyros_belt_y          6.096310 11.73882 12.793943 10.975397 15.484897
## gyros_belt_z         17.637518 22.27540 21.621044 16.971328 21.312511
## accel_belt_x         10.006321 10.03746 13.435500  9.057926  8.922109
## accel_belt_y          8.052497 13.32921 10.333990 15.160252  8.981393
## accel_belt_z         16.859516 21.83227 16.843481 16.936472 14.309763
## magnet_belt_x        11.395896 22.28171 21.694518 17.128448 21.277430
## magnet_belt_y        14.629111 22.46903 22.730713 18.448112 18.154864
## magnet_belt_z        14.493483 21.32397 18.346063 25.193365 20.163539
## roll_arm             15.084114 23.23581 16.105334 18.131868 14.534749
## pitch_arm            15.066466 20.02968 15.945071 17.983344 19.913348
## yaw_arm              22.672727 19.95073 21.348894 20.401556 14.359084
## total_accel_arm       9.396583 16.33282 12.263605 14.051004 13.610184
## gyros_arm_x          14.255690 16.14648 16.802937 15.344697 16.288664
## gyros_arm_y          15.837080 19.18898 16.511248 19.500031 16.037683
```

```
## gyros_arm_z           7.164529 12.16819 10.082337  7.488734  6.888830
## accel_arm_x           8.844325 13.56719 15.400917 17.437555 12.466608
## accel_arm_y          10.287244 14.65579  9.422579 12.239481  9.511545
## accel_arm_z           7.528002 13.33434 14.211333 16.045498 13.037553
## magnet_arm_x         10.339097 11.39079 12.545810 12.973826 11.214163
## magnet_arm_y          9.186540 15.22830 14.467481 20.525207 12.250204
## magnet_arm_z         12.661153 19.69772 15.207292 16.334117 14.301821
## roll_dumbbell        19.248383 27.14607 23.391753 24.870097 28.040968
## pitch_dumbbell       11.319059 18.77260 11.108364 12.852955 12.736648
## yaw_dumbbell         12.398042 21.25066 16.635706 15.824826 15.883915
## total_accel_dumbbell 14.780154 23.33686 17.425815 23.110551 23.144112
## gyros_dumbbell_x     11.542165 18.79682 16.691418 13.514375 12.990255
## gyros_dumbbell_y     30.466619 23.59353 22.526246 19.851398 18.897647
## gyros_dumbbell_z     15.402974 15.09281 11.677276 12.774110 10.470505
## accel_dumbbell_x     11.926293 18.75276 13.547562 15.501623 14.439876
## accel_dumbbell_y     24.782288 22.84485 30.448372 24.519195 25.572067
## accel_dumbbell_z     15.740956 24.90308 22.014175 24.995396 25.620319
## magnet_dumbbell_x    20.487403 22.09581 23.399257 23.195730 19.791978
## magnet_dumbbell_y    49.237269 49.35456 48.757011 45.436888 40.678354
## magnet_dumbbell_z    56.994512 44.38419 57.111856 44.929128 43.059400
## roll_forearm         31.711701 25.61385 26.486067 23.337805 22.714888
## pitch_forearm        46.353486 53.14263 73.063831 48.353332 43.666188
## yaw_forearm          12.696703 16.45021 14.106031 17.997435 17.223135
## total_accel_forearm  14.110726 17.15863 12.688147  8.999978 11.645803
## gyros_forearm_x       6.059137 10.84721 12.924674 12.176016  9.915962
## gyros_forearm_y      11.391011 16.85877 16.452114 16.021043 12.406219
## gyros_forearm_z      10.124477 16.62014 17.469105 10.510322 11.098456
## accel_forearm_x      16.432888 27.47805 22.536088 32.159366 30.275345
## accel_forearm_y      12.475880 14.31931 19.793584 12.084266 14.797695
## accel_forearm_z      16.695558 21.99477 15.352512 18.002827 17.526898
## magnet_forearm_x      8.988383 17.19981 13.937994 11.640919 17.062290
## magnet_forearm_y     16.339390 18.32642 15.232784 16.985674 17.375210
## magnet_forearm_z     20.105275 25.06644 22.711862 20.562494 20.146346
##                      MeanDecreaseAccuracy MeanDecreaseGini
## num_window                       81.44577       1957.92636
## roll_belt                        87.83625       1237.50903
## pitch_belt                       53.14051        547.60510
## yaw_belt                         68.86152        648.30203
## total_accel_belt                 13.30173         61.57631
## gyros_belt_x                     23.39244         34.76118
## gyros_belt_y                     17.13795         37.31834
## gyros_belt_z                     33.54491        104.06100
## accel_belt_x                     17.74593         30.70451
## accel_belt_y                     18.54157         50.32245
## accel_belt_z                     21.93269        197.37436
## magnet_belt_x                    31.32560        117.79260
## magnet_belt_y                    23.39946        177.75876
## magnet_belt_z                    27.48354        154.89183
## roll_arm                         24.15136        121.16787
## pitch_arm                        25.22761         70.29612
## yaw_arm                          32.51774         83.24880
## total_accel_arm                  21.98983         30.99967
## gyros_arm_x                      22.99208         35.33624
## gyros_arm_y                      28.35079         45.05826
```

```
## gyros_arm_z                  17.83690         14.14624
## accel_arm_x                  15.21790         87.85684
## accel_arm_y                  17.54267         47.87178
## accel_arm_z                  14.76127         37.33687
## magnet_arm_x                 12.01453        105.19659
## magnet_arm_y                 17.56675         84.55643
## magnet_arm_z                 25.01968         51.26905
## roll_dumbbell                28.36899        247.77885
## pitch_dumbbell               18.66316         57.12056
## yaw_dumbbell                 22.03432        103.99530
## total_accel_dumbbell         27.47324        182.69981
## gyros_dumbbell_x             25.63532         40.54663
## gyros_dumbbell_y             42.24174         87.35634
## gyros_dumbbell_z             24.95218         25.55399
## accel_dumbbell_x             18.22852         95.43074
## accel_dumbbell_y             35.57693        252.34158
## accel_dumbbell_z             30.46945        177.96343
## magnet_dumbbell_x            25.39323        244.21568
## magnet_dumbbell_y            56.64565        544.31962
## magnet_dumbbell_z            60.37148        562.89843
## roll_forearm                 28.00611        442.52777
## pitch_forearm                68.87053        787.74375
## yaw_forearm                  25.80302         64.86745
## total_accel_forearm          18.82141         36.49106
## gyros_forearm_x              20.60649         18.59645
## gyros_forearm_y              24.39909         33.26042
## gyros_forearm_z              23.74152         23.08513
## accel_forearm_x              29.84244        210.19556
## accel_forearm_y              22.78469         39.85169
## accel_forearm_z              25.52075        112.09467
## magnet_forearm_x             18.47870         76.24430
## magnet_forearm_y             22.77849         79.80038
## magnet_forearm_z             28.79496        140.60775
```

```
# Make prediction on test data
predicted.classes<- model2 %>% predict (test.data)
head(predicted.classes)
```

```
## [1] A A A A A A
## Levels: A B C D E
```

```
# compute model accuracy rate

mean(predicted.classes==test.data$classe)
```

```
## [1] 0.9981308
```

Boosting model

```r
model3<- train(classe~., data=train.data, method="xgbTree", trControl=trainControl("cv", number=3))
# Best tuning parameter
model3$bestTune
```

```
##     nrounds max_depth eta gamma colsample_bytree min_child_weight
## 105     150         3 0.4     0              0.8                1
##     subsample
## 105      0.75
```

```r
# Make prediction on test data
predicted.classes<- model3 %>% predict (test.data)
head(predicted.classes)
```

```
## [1] A A A A A A
## Levels: A B C D E
```

```r
# compute model accuracy rate

mean(predicted.classes==test.data$classe)
```

```
## [1] 0.9996602
```

Based on three models, model2 (randomForest) has stronger accuracy rate than other two. So the final validation has been assayed using model2.

Applying the best model to the validation data

```r
results<- predict(model2, newdata=testdata)
results
```

```
##  [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
```