Final Techinical Documentation of Team Bigger Data

```r
library(data.table)
library(ggplot2)
library(dplyr)
library(knitr)
require(bit64)
library(randomForest)
library(stringr)
library(lubridate)
```

```r
#Open and pre-process original data set
df = fread('https://raw.githubusercontent.com/cszys888/BEGGER-DATA---Team-1/master/CloudFactory_DataSet
colnames(df)[4] = "keytype"
colnames(df)[5] = "mousemove"
colnames(df)[6] = "mouseclick"
colnames(df)[7] = "duration"
df$keytype[is.na(df$keytype)] = 0
df$mousemove[!is.na(df$mousemove)] = "Yes"
df$mousemove[is.na(df$mousemove)] = "No"
df$mouseclick[!is.na(df$mouseclick)] = "Yes"
df$mouseclick[is.na(df$mouseclick)] = "No"

#Open and process additional info of worker
#Reformat the education and gender info
worker_profile = fread('https://raw.githubusercontent.com/cszys888/BEGGER-DATA---Team-1/master/receipt_
academic_degree = worker_profile$academic_degree
master_sign = str_detect(academic_degree, "Master")
bachelor_sign = str_detect(academic_degree, "Bachelor")
higher_sign = str_detect(academic_degree, "Higher")
secondary_sign = str_detect(academic_degree, "Secondary")
na_sign = !str_detect(academic_degree, " ")
worker_profile$academic_degree[master_sign] = "master"
worker_profile$academic_degree[(!master_sign)&(bachelor_sign)] = "bachelor"
worker_profile$academic_degree[(!master_sign)&(!bachelor_sign)&(higher_sign)] = "highersecondary"
worker_profile$academic_degree[(!master_sign)&(!bachelor_sign)&(!higher_sign)&(secondary_sign)] = "secon
worker_profile$academic_degree[na_sign] = "na"
worker_profile$academic_degree = factor(worker_profile$academic_degree,
                                        levels = c("master", "bachelor",
                                                   "highersecondary", "secondary",
                                                   "na"))
worker_profile$gender = factor(worker_profile$gender, levels = c("Male", "Female"))
str(worker_profile)
```

```
## Classes 'data.table' and 'data.frame':   93 obs. of  5 variables:
##  $ worker_id      : chr  "501767d1cb022453170004dc" "503b89aacb02245d320001a4" "50531712cb022432f300
##  $ onbarded_date  : chr  "2012-07-31 05:06:25 UTC" "2012-08-27 14:52:26 UTC" "2012-09-14 11:37:54 UT
##  $ birth_year     : int  1993 1987 1980 1990 1990 1991 1992 1986 1989 1995 ...
##  $ gender         : Factor w/ 2 levels "Male","Female": 1 1 1 1 1 2 1 1 1 2 ...
##  $ academic_degree: Factor w/ 5 levels "master","bachelor",..: 3 5 3 2 3 2 1 5 2 3 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
worker_profile = worker_profile %>%
  slice(-(90:93))
present_time = Sys.time()
present_time.poslt = as.POSIXlt(present_time, tz = "America/New_York")
onboardtime = worker_profile$onbarded_date
onboardtime.posix = as.POSIXct(onboardtime, format = "%Y-%m-%d %H:%M:%S")
onboardtime.poslt = as.POSIXlt(onboardtime, tz = "America/New_York")
onboard_duration = (present_time.poslt$year - onboardtime.poslt$year) *12 +
  (present_time.poslt$mon - onboardtime.poslt$mon)
worker_profile$onboard_duration = onboard_duration
worker_profile$age = 2017 - worker_profile$birth_year
worker_profile = worker_profile %>%
  select(-onbarded_date, -birth_year)

#join worker_profile with df
dt1 = inner_join(df, worker_profile)
```

```
## Joining, by = "worker_id"
```

```r
#calculate the average duration for one task
duration_stat = dt1 %>%
  group_by(task_id) %>%
  summarise(duration = duration[1])
summary(duration_stat)
```

```
##    task_id             duration
##  Length:4967        Min.   :  1.986
##  Class :character   1st Qu.: 13.043
##  Mode  :character   Median : 19.678
##                     Mean   : 28.473
##                     3rd Qu.: 32.234
##                     Max.   :413.602
```

```r
#calculate the relative time of each operation
dt1 = dt1 %>%
  group_by(task_id) %>%
  mutate(rela_time = (timestamp - timestamp[1])/1000)
```

Full Time Model

```r
#Data pre-processing to generage input variables for random forest model
dt2 = dt1 %>%
  group_by(task_id) %>%
  summarise(duration = duration[1],
            total_op = sum(mousemove == "Yes") + sum(mouseclick == "Yes") + sum(keytype != 0),
            count_mousemove = sum(mousemove == "Yes")/total_op,
            count_mouseclick = sum(mouseclick == "Yes")/total_op,
            key1 = sum(keytype == 1)/total_op,
            key2 = sum(keytype == 2)/total_op,
            key3 = sum(keytype == 3)/total_op,
            key4 = sum(keytype == 4)/total_op,
            key5 = sum(keytype == 5)/total_op,
            key6 = sum(keytype == 6)/total_op,
            key7 = sum(keytype == 7)/total_op,
            key8 = sum(keytype == 8)/total_op,
            key9 = sum(keytype == 9)/total_op,
            key10 = sum(keytype == 10)/total_op,
            key11 = sum(keytype == 11)/total_op,
            key12 = sum(keytype == 12)/total_op,
            accuracy = accuracy[1],
            worker_id = worker_id[1],
            gender = gender[1],
            academic_degree = academic_degree[1],
            onboard_duration = onboard_duration[1],
            age = age[1])%>%
  select(-task_id, -worker_id, -total_op) %>%
  mutate(accuracy = (accuracy == 1))
dt2$accuracy = as.factor(dt2$accuracy)

#------------------------------------------------------------------------------------------

#randomforest model

#divide data into training and testing
set.seed(2000)
index = sample(1:nrow(dt2), round(0.5*nrow(dt2)))
train = dt2[index,]
test = dt2[-index,]

#build model on training data
n = names(dt2)
f = as.formula(paste("accuracy~", paste(n[!n %in% "accuracy"], collapse = "+")))
rf2 = randomForest(data = train, f, importance = TRUE)
predict_train = predict(rf2)
train_table2 = table(train$accuracy, predict_train)
kable(train_table2)
```

	FALSE	TRUE
FALSE	823	373
TRUE	316	972

```
train_accurate2 = sum(diag(train_table2))/nrow(train);train_accurate2
```

```
## [1] 0.7226248
```

```
#build model on testing data
predict_test = predict(rf2, newdata = test, type = "response")
test_table2 = table(test$accuracy, predict_test)
kable(test_table2)
```

	FALSE	TRUE
FALSE	783	382
TRUE	314	1004

```r
test_accurate2 = sum(diag(test_table2))/nrow(test);test_accurate2
```

```
## [1] 0.7196939
```

```r
#output variable importance as csv files
varimportance = rf2$importance
write.csv(varimportance, file = "varimportance.csv")

#-------------------------------------------------------------------------------------------

#Build up Business metrics on full model
#Calculate Information Gain for training data
matrix_train2=train_table2/sum(train_table2)
Entro_Condi = -sum(matrix_train2[1,])*log2(sum(matrix_train2[1,])) - sum(matrix_train2[2,])*log2(sum(ma
Entro_Class = -sum(matrix_train2[,1])*log2(sum(matrix_train2[,1])) - sum(matrix_train2[,2])*log2(sum(ma
Entro_Matri = -sum(matrix_train2[1,1])*log2(sum(matrix_train2[1,1])) - sum(matrix_train2[1,2])*log2(sum
PIG_2_train = (Entro_Condi + Entro_Class - Entro_Matri)/Entro_Condi;PIG_2_train
```

```
## [1] 0.1473064
```

```r
#Calculate Information Gain for testing data
matrix_test2=test_table2/sum(test_table2)
Entro_Condi = -sum(matrix_test2[1,])*log2(sum(matrix_test2[1,])) - sum(matrix_test2[2,])*log2(sum(matri
Entro_Class = -sum(matrix_test2[,1])*log2(sum(matrix_test2[,1])) - sum(matrix_test2[,2])*log2(sum(matri
Entro_Matri = -sum(matrix_test2[1,1])*log2(sum(matrix_test2[1,1])) - sum(matrix_test2[1,2])*log2(sum(ma
PIG_2_test = (Entro_Condi + Entro_Class - Entro_Matri)/Entro_Condi;PIG_2_test
```

```
## [1] 0.1418801
```

```r
#Final accuracy and PIG of full model
result_full = data.frame(train_accuracy = train_accurate2,
                         train_PIG = PIG_2_train,
                         test_accuracy = test_accurate2,
                         test_PIG = PIG_2_test,
                         train_e = train_table2[1,1],
                         train_f = train_table2[1,2],
                         train_g = train_table2[2,1],
                         train_h = train_table2[2,2],
                         test_e = test_table2[1,1],
                         test_f = test_table2[1,2],
                         test_g = test_table2[2,1],
                         test_h = test_table2[2,2])

#------------------------------------------------------------------------------------

#Calculate the break-even point and the highest cost-saving value of our model, given a specific value
p = 0.08 #just an assumption, can be updated in the future
M = 30 #just an assumption, can be updated in the future
N = 1 #just an assumption, can be updated in the future
e = test_table2[1,1]/sum(test_table2)
f = test_table2[1,2]/sum(test_table2)
g = test_table2[2,1]/sum(test_table2)
h = test_table2[2,2]/sum(test_table2)
a = e+f
b = g+h
c = e+g
```

```
d = f+h
A1 = 1+((1-p)*a+a*b)
A2 = a*p

B1 = 2+(a*(1-p^2)+a*b^2)
B2 = a*p^2

C1 = d*A1+c*B1
C2 = e*p^2+f*p

CostA = A1*N+A2*M;CostA
```

```
## [1] 2.806763
```

```
CostB = B1*N+B2*M;CostB
```

```
## [1] 2.688471
```

```
CostC = C1*N+C2*M;CostC
```

```
## [1] 2.515918
```

```
#break-even point in test data
breakeven_M1 = (A1-C1)/(C2-A2)
print(paste0("The first break-even point, which is the starting point of the viable scale of our model,
```

```
## [1] "The first break-even point, which is the starting point of the viable scale of our model, is 17
```

```
breakeven_M2 = (B1-C1)/(C2-B2);breakeven_M2
```

```
## [1] 45.23904
```

```
print(paste0("The first break-even point, which is the ending point of the viable scale of our model, i
```

```
## [1] "The first break-even point, which is the ending point of the viable scale of our model, is 45.2
```

```
#highest cost-saving
highest_mn = (A1 - B1)/(B2 - A2)
print(paste0("The value of M/N which leads to the highest cost saving is ", highest_mn))
```

```
## [1] "The value of M/N which leads to the highest cost saving is 26.574452414112"
```

```
highest_saving = (A1+A2*highest_mn) - (C1+C2*highest_mn)
print(paste0("The highest cost-saving is ", highest_saving))
```

```
## [1] "The highest cost-saving is 0.211340581198765"
```

```
highest_percent_saving = highest_saving/(A1+A2*highest_mn)
print(paste0("The highest percentage cost-saving is ", highest_percent_saving))
```

```
## [1] "The highest percentage cost-saving is 0.0789118836292593"
```

Looping 50 Different Time-Trigger Models

```
#initiate a dataframe to store the result for 50 different time triggers
result_trigger = data.frame(time_trigger = integer(),
                            train_accuracy = double(),
                            train_PIG = double(),
                            test_accuracy = double(),
                            test_PIG = double(),
                            train_e = integer(),
                            train_f = integer(),
                            train_g = integer(),
                            train_h = integer(),
                            test_e = integer(),
                            test_f = integer(),
                            test_g = integer(),
                            test_h = integer())

for (i in 1:50) {
  #select the actions which are before the time trigger in each task, and pre-process the data using th
  dt3 = dt1 %>%
    mutate(keep = ifelse(rela_time <= i, 1, 0)) %>%
    filter(keep == 1) %>%
    select(-keep) %>%
    group_by(task_id) %>%
    summarise(duration = duration[1],
              duration_to_now = max(rela_time),
              total_op = sum(mousemove == "Yes") + sum(mouseclick == "Yes") + sum(keytype != 0),
              count_mousemove = sum(mousemove == "Yes")/total_op,
              count_mouseclick = sum(mouseclick == "Yes")/total_op,
              key1 = sum(keytype == 1)/total_op,
              key2 = sum(keytype == 2)/total_op,
              key3 = sum(keytype == 3)/total_op,
              key4 = sum(keytype == 4)/total_op,
              key5 = sum(keytype == 5)/total_op,
              key6 = sum(keytype == 6)/total_op,
              key7 = sum(keytype == 7)/total_op,
              key8 = sum(keytype == 8)/total_op,
              key9 = sum(keytype == 9)/total_op,
              key10 = sum(keytype == 10)/total_op,
              key11 = sum(keytype == 11)/total_op,
              key12 = sum(keytype == 12)/total_op,
              accuracy = accuracy[1],
              worker_id = worker_id[1],
              gender = gender[1],
              academic_degree = academic_degree[1],
              onboard_duration = onboard_duration[1],
              age = age[1])%>%
    select(-task_id, -worker_id, -total_op) %>%
    mutate(accuracy = (accuracy == 1))
  dt3$accuracy = as.factor(dt3$accuracy)

  #-----------------------------------------------------------------

  #randomforest model
  #divide data into training and testing
  set.seed(2000)
  index3 = sample(1:nrow(dt3), round(0.5*nrow(dt3)))
  train3 = dt3[index3,]
  test3 = dt3[-index3,]
  #build model on training data
  n3 = names(dt3)
  f3 = as.formula(paste("accuracy~", paste(n3[!n3 %in% "accuracy"], collapse = "+")))
  rf3 = randomForest(data = train3,
                       f3, importance = TRUE)
  predict_train3 = predict(rf3)
  train_table3 = table(train3$accuracy, predict_train3)
  kable(train_table3)
  train_accurate3 = sum(diag(train_table3))/nrow(train3);train_accurate3

  #build model on testing data
  predict_test3 = predict(rf3, newdata = test3, type = "response")
  test_table3 = table(test3$accuracy, predict_test3)
  kable(test_table3)
  test_accurate3 = sum(diag(test_table3))/nrow(test);test_accurate3

  #---------------------------------------------------------------------------

  #Calculate Information Gain for training data
  matrix_train3=train_table3/sum(train_table3)
  Entro_Condi = -sum(matrix_train3[1,])*log2(sum(matrix_train3[1,])) -   sum(matrix_train3[2,])*log2(su
  Entro_Class = -sum(matrix_train3[,1])*log2(sum(matrix_train3[,1])) -   sum(matrix_train3[,2])*log2(su
  Entro_Matri = -sum(matrix_train3[1,1])*log2(sum(matrix_train3[1,1])) -   sum(matrix_train3[1,2])*log2
  PIG_3_train = (Entro_Condi + Entro_Class-Entro_Matri)/Entro_Condi;PIG_3_train
  #Calculate Information Gain for testing data
  matrix_test3=test_table3/sum(test_table3)
  Entro_Condi = -sum(matrix_test3[1,])*log2(sum(matrix_test3[1,])) - sum(matrix_test3[2,])*log2(sum(mat
  Entro_Class = -sum(matrix_test3[,1])*log2(sum(matrix_test3[,1])) - sum(matrix_test3[,2])*log2(sum(mat
  Entro_Matri = -sum(matrix_test3[1,1])*log2(sum(matrix_test3[1,1])) -   sum(matrix_test3[1,2])*log2(su
  PIG_3_test = (Entro_Condi + Entro_Class - Entro_Matri)/Entro_Condi;PIG_3_test

  #------------------------------------------------------------------------------------------

  #store result for time trigger i
  result_tmp = data.frame(time_trigger = i,
                          train_accuracy = train_accurate3,
                          train_PIG = PIG_3_train,
                          test_accuracy = test_accurate3,
                          test_PIG = PIG_3_test,
                          train_e = train_table3[1,1],
                          train_f = train_table3[1,2],
                          train_g = train_table3[2,1],
                          train_h = train_table3[2,2],
                          test_e = test_table3[1,1],
                          test_f = test_table3[1,2],
                          test_g = test_table3[2,1],
                          test_h = test_table3[2,2])
  result_trigger[i,] = result_tmp
  print(paste0("The ", i, "iteration of the loop is finished"))
}
```

```
## [1] "The 1iteration of the loop is finished"
## [1] "The 2iteration of the loop is finished"
## [1] "The 3iteration of the loop is finished"
## [1] "The 4iteration of the loop is finished"
## [1] "The 5iteration of the loop is finished"
## [1] "The 6iteration of the loop is finished"
## [1] "The 7iteration of the loop is finished"
## [1] "The 8iteration of the loop is finished"
## [1] "The 9iteration of the loop is finished"
## [1] "The 10iteration of the loop is finished"
## [1] "The 11iteration of the loop is finished"
## [1] "The 12iteration of the loop is finished"
## [1] "The 13iteration of the loop is finished"
## [1] "The 14iteration of the loop is finished"
## [1] "The 15iteration of the loop is finished"
## [1] "The 16iteration of the loop is finished"
## [1] "The 17iteration of the loop is finished"
## [1] "The 18iteration of the loop is finished"
## [1] "The 19iteration of the loop is finished"
## [1] "The 20iteration of the loop is finished"
## [1] "The 21iteration of the loop is finished"
## [1] "The 22iteration of the loop is finished"
## [1] "The 23iteration of the loop is finished"
## [1] "The 24iteration of the loop is finished"
## [1] "The 25iteration of the loop is finished"
## [1] "The 26iteration of the loop is finished"
## [1] "The 27iteration of the loop is finished"
## [1] "The 28iteration of the loop is finished"
## [1] "The 29iteration of the loop is finished"
## [1] "The 30iteration of the loop is finished"
## [1] "The 31iteration of the loop is finished"
## [1] "The 32iteration of the loop is finished"
## [1] "The 33iteration of the loop is finished"
## [1] "The 34iteration of the loop is finished"
## [1] "The 35iteration of the loop is finished"
## [1] "The 36iteration of the loop is finished"
## [1] "The 37iteration of the loop is finished"
## [1] "The 38iteration of the loop is finished"
## [1] "The 39iteration of the loop is finished"
## [1] "The 40iteration of the loop is finished"
## [1] "The 41iteration of the loop is finished"
## [1] "The 42iteration of the loop is finished"
## [1] "The 43iteration of the loop is finished"
## [1] "The 44iteration of the loop is finished"
## [1] "The 45iteration of the loop is finished"
## [1] "The 46iteration of the loop is finished"
## [1] "The 47iteration of the loop is finished"
## [1] "The 48iteration of the loop is finished"
## [1] "The 49iteration of the loop is finished"
## [1] "The 50iteration of the loop is finished"
```

```r
#Output the result of full model and 50 different time-trigger model as csv files
write.csv(result_full, file = "result_full.csv")
write.csv(result_trigger, file = "result_trigger.csv")
```