Rappi Challenge

Diego Fernando Armas Texta

24/2/2021

Rappi Pay Challenge

funciones que ocuparemos

```
'%!in%' <- function(x,y)!('%in%'(x,y))
right = function(text, num_char) {
  substr(text, nchar(text) - (num_char-1), nchar(text))
}
```

quitamos la notación científica

```
options(scipen = 999)
```

Creamos un directorio para nuestros outputs

```
mkdirs("outputs")
```

```
## [1] FALSE
```

Cargamos nuestra data con las clases necesarias

```
data = fread("raw_data\\BIQuiz_022021.csv",colClasses = c('ID' = 'character',
                                                          'UPDATE' = 'POSIXct',
                                                           'CP' ='character'))
```

Carguemos nuestra data y separemos el dia del mes, en nombre del dia, mes y año

```
data = data %>%
       mutate(., "INTEREST_RATE" = INTEREST_RATE/100,
                 "fecha_measure" = as.Date(data$UPDATE),
                 "hora" = lubridate::hour(UPDATE),
                 "num_dia" = lubridate::day(UPDATE),
                 "num_dia_sem" = lubridate::wday(UPDATE, week_start = 1),
                 "nom_dia"= gsub("[[:punct:]]","",lubridate::wday(UPDATE, label = T)),
                 "mes_nombre" = lubridate::month(UPDATE, label = T),
                 "mes" = lubridate::month(UPDATE),
                 "trimestre" = lubridate::quarter(UPDATE),
                 "anio" = lubridate::year(UPDATE))
```

Veamos cuantos usuarios unicos tenemos en nuestro archivo

```
uniqueN(data$ID)
```

```
## [1] 3341
```

Sacamos la fecha min y max que tenemos

```
stDte = min(data$fecha_measure)
edDte =  max(data$fecha_measure)
```

Traemos valores mayores al día de hoy; dejemos fuera del análisis estos datos para evitar cualquier sesgo en el futuro

```
data = data %>%
       filter(., fecha_measure <= today())
```

Veamos si después de este filtro tuvimos una cantidad significativa de pérdida de usuarios

```
uniqueN(data$ID)
```

```
## [1] 3335
```

La distribución de los estatus está un poco dispareja, tenemos que está más cargada hacia los estatus vacíos

```
tbl
```

```
##           conteo Porcentaje
##             4765      37.57
## APPROVED    1501      11.83
## DELIVERED   1043       8.22
## REJECTED     791       6.24
## RESPONSE    2292      18.07
## RISK        2292      18.07
## <NA>           0       0.00
```

En la hoja de los ejercicios se nos menciona que el proceso empieza cuando un cliente responde una comunicación; viendo la tabla anterior vemos que todo los que respondieron a la comunicación (2,292) son los mismos que entraron al modelo de riesgo (2,292) , y de estos 1,501 fueron aprobados y 791 fueron rechazados (notemos que se siguen sumando 2,292) de los cuales 1,043 ya se les fue enviado tu tarjeta. la incógnita aún es que está pasando con nuestro estatus vacío

Veamos como se ve la distribución de nuestro estatus de vacío a lo largo de los días

```
status_vacio
```

```
## # A tibble: 5 x 2
##   UPDATE              conteo
##   <dttm>               <int>
## 1 2019-11-11 00:00:00   1043
## 2 2019-11-12 15:28:06      1
## 3 2019-11-12 17:24:27      1
## 4 2019-11-12 21:23:17      1
## 5 2019-11-13 06:38:53      1
```

Existe que existe una gran cantidad de registros vacíos en el primer día de nuestra información, sin pérdida de generalidad (spg) y basándonos en la hoja de los ejercicios supongamos que el día 11 de noviembre haya sido el lanzamiento de la tarjeta; existe la posibilidad de que se haya tenido un error en sistemas, para verificar esto veamos como los registros que vienen vacíos en la columna de estatus se comportan a lo largo de las otras columnas

```
nas_dias
```

```
## # A tibble: 5 x 8
##   UPDATE              nas_interest_rate nas_amount nas_cat nas_txn nas_cp
##   <dttm>                          <int>      <int>   <int>   <int>  <int>
## 1 2019-11-11 00:00:00              1043       1043    1043    1043      0
## 2 2019-11-12 15:28:06                 1          1       1       0      0
## 3 2019-11-12 17:24:27                 1          1       1       0      0
## 4 2019-11-12 21:23:17                 1          1       1       0      0
## 5 2019-11-13 06:38:53                 1          1       1       0      0
## # ... with 2 more variables: nas_delivery_score <int>, total_nas <int>
```

```
sum(nas_dias$nas_txn)
```

```
## [1] 1043
```

De esta tabla pasada la fecha del 11 de noviembre de 2019 es la única que contiene valores “NA’s” en la columna de txn; es decir podemos decir que de nuestra base con registros vacíos 3,722 registros corresponden a txns (4765-1043) hemos encontrado un problema en el etiquetado de los datos, en los siguientes pasos haremos el re-etiquetado.Por otro lado pareciera que los 1,043 registros no tienen sentido pero si recordamos nuestro universo y le restamos el número de personas que respondieron obtenemos 1,043 (3,335 - 2,292) es decir estas 1043 personas no respondieron a nuestro acercamiento/comunicación!!! de igual manera hagamos un nuevo etiquetado.

```
data = data %>%
        mutate(., STATUS = ifelse( STATUS == "" & is.na(data$TXN),"NO_RESPONSE",
                                   ifelse(STATUS == "" & !is.na(data$TXN),"TXN",STATUS)))
```

Propongamos una breve segmentación para los clientes a los que ya se les aprobó su tarjeta, esta segmentación será sobre el comportamiento de del uso de la tarjeta, para esto chequemos a nivel cliente cuantas txns hacen y el monto total

```
segmentacion = data %>%
               filter(., STATUS == "TXN")  %>%
               group_by(ID) %>%
               dplyr::summarise(., "txns" = n(),
                                "monto_total" = sum(TXN,na.rm = T),
                                "segmentacion" = case_when(monto_total <= 800 ~ "4.-Bronce",
                                                           monto_total <= 6043 & monto_total > 800 ~
"3.-Plata",
                                                           monto_total <= 15000 & monto_total > 6043
~ "2.-Oro",
                                                           monto_total > 15000 ~ "1.-Diamante"))
```

```
table(segmentacion$segmentacion,segmentacion$txns)
```

```
##
##                 1   2   3   4   5
##   1.-Diamante   6  14  25  27  35
##   2.-Oro        9  22  31  47  61
##   3.-Plata     52 113 137 136 130
##   4.-Bronce   173 118  60  36  15
```

Dada nuestra segmentación parece existir una relación entre el el segmento asignado y el número de txns que realiza.

Guardaremos esta tabla para crear un modelo de datos dentro de nuestra herramienta de BI

```
fwrite(segmentacion,"outputs\\segmetacion.csv", row.names = F)
```

De igual manera guardemos nuestros datos correctamente filtrados y etiquetado

```
fwrite(data,"outputs\\data_limpia.csv", row.names = F)
```

Creamos un agrupado a nivel fecha; esto para que esta columna sea con la que podamos hacer cálculos a través del tiempo en nuestra herramienta de BI

```
fechas
```

```
## # A tibble: 292 x 4
##    fecha_measure num_intereciones monto_transaccionado monto_lineas_liberadas
##  * <date>                  <int>                <dbl>                  <int>
##  1 2019-11-11               1942                   0                  58000
##  2 2019-11-12                762                 786.                519600
##  3 2019-11-13                691               41915.               1410200
##  4 2019-11-14                711              136038.               2196300
##  5 2019-11-15                614              102816.               1990200
##  6 2019-11-16                584              234560.               1755300
##  7 2019-11-17                544              125454.               1982200
##  8 2019-11-18                465              146132.               1310500
##  9 2019-11-19                440              169361.               1032500
## 10 2019-11-20                352              189819.               1099900
## # ... with 282 more rows
```

Como queremos crear un rolling over year para nuestro dashbord necesitamos información de todos los días de nuestro primer dia hasta nuestro ultimo dia, al no tener información de todos los días seguidos hagamos una simulación de los días a través del tiempo para hacerle un left join y de esta manera conseguir fechas consecutivas.

```
head(fechas_final)
```

```
##          dia num_intereciones monto_transaccionado monto_lineas_liberadas
## 1 2019-11-11             1942               0.0000                  58000
## 2 2019-11-12              762             785.9097                 519600
## 3 2019-11-13              691            41914.7847                1410200
## 4 2019-11-14              711           136038.0490                2196300
## 5 2019-11-15              614           102816.1478                1990200
## 6 2019-11-16              584           234560.4994                1755300
##   num_dia num_dia_sem nom_dia mes mes_nombre trimestre anio
## 1      11           1     lun  11        nov         4 2019
## 2      12           2     mar  11        nov         4 2019
## 3      13           3     mié  11        nov         4 2019
## 4      14           4     jue  11        nov         4 2019
## 5      15           5     vie  11        nov         4 2019
## 6      16           6     sáb  11        nov         4 2019
```

Guardemos nuestras fechas para usarla en nuestras visualizaciones

```
fwrite(fechas_final,"outputs\\fecha_measure.csv", row.names = F)
```

Archivo para el equipo de sistemas; la finalidad de este output es que el equipo de IT nos ayude a ver porque tenemos registros con fechas que aún no llegan

```
fechas_it = fread("raw_data\\BIQuiz_022021.csv",colClasses = c('ID' = 'character',
                                                               'UPDATE' = 'POSIXct')) %>%
            filter(.,UPDATE > today())
fwrite(fechas_final,"outputs\\feedback_it.csv", row.names = F)
```

Cargamos la información de la geolocalización de los Códigos Postales

```
cps_geo = readxl::read_xlsx("external_data\\CP.xlsx") %>%
                mutate(., "CP" = as.character(right(Estado_CP,5)))
```

Filtramos el universo de aprobados

```
aprobados = data %>%
     filter(., STATUS == "APPROVED") %>%
     select(ID,AMOUNT)
```

Filtramos el universo de enviados y lo cruzamos con la información de geolocalización de los Códigos Postales

```
cp
```

```
## # A tibble: 5 x 7
##   CP    Número_de_clientes Monto_promedio Score_promedio Estado_CP       latitud
##   <chr>              <int>          <dbl>          <dbl> <chr>              <dbl>
## 1 11560                215         14285.           1.91 CIUDAD_DE_MEXI~    19.4
## 2 44100                296         14075.           1.95 JALISCO_44100      20.7
## 3 44620                102         13771.           1.95 JALISCO_44620      20.6
## 4 53100                321         14759.           2.02 MEXICO_53100       19.5
## 5 64000                109         13944.           1.95 NUEVO_LEON_640~    25.7
## # ... with 1 more variable: longitud <dbl>
```