Propensity of email engagement and prosensity to churn models

Giulia Chiaretti (800928), Federica Fiorentini (807124), Alberto Monaco (803669

Questo progetto consiste nell'applicazione di algoritmi di Machine Learning nei seguenti due modelli di business: *Propensity of email engagement Propensity to churn*

In generale, l'obiettivo dei propensity modelling Ã Â¨ quello di analizzare il comportamento dei clienti selenzionando quelli che, con un'alta probabilitÃ Â , potrebbero commettere una certa azione nel futuro.

Il primo modello, propensity of email engagement, va ad indagare quanto una campagna marketing basata sull'invio di email riesca a "raggiungere" i clienti, e quindi ad essere efficace.

Il propensity to churn model, invece, ha l'obiettivo di prevedere quali consumatori cesseranno di essere clienti dell'azienda, i cosiddetti churner.

Entrambi i modelli consistono in problemi di classificazione binaria e, per ognuno di essi, sono stati sviluppati diversi algoritmi di Machine Learning (ed in particolare di Supervised Learning) a seguito di un'approfondita analisi dei dati a disposizione.

I diversi algoritmi sono stati tunati al fine di scegliere il parametro migliore per ognuno di essi e, infine, sono stati confrontati per valutarne la performance.

PROPENSITY OF EMAIL ENGAGEMENT MODEL

Business question

Il fulcro dei propensity of email engagement models Ã Â¨ prevedere se un cliente risponderÃ Â prontamente ad una specifica azione di marketing. Lo studio va, perciÃ Â², ad indagare se le email inviate dalla compagnia riescano a raggiungere e coinvolgere i clienti in maniera mirata. L'obiettivo finale Ã Â¨ quello di verificare l'efficacia della campagna marketing effettuata per capire se Ã Â¨ uno strumento valido di customer engagement o se deve subire modifiche e miglioramenti.

Per svolgere questo tipo di problema Ã Â¨ stato impostato un modello di classificazione in cui la variabile target si presenta come un attributo binario che indica se il cliente ha aperto o meno l'email in una finestra temporale pari a due giorni dall'invio della stesa.

DATA CLEANING AND PREPARATION

Di seguito viene riportata una consistente fase di preprocessing in cui vengono create le seguenti variabili esplicative:

- ID_EVENT_S: id del feedback
- NUM_SEND_PREV: variabile numerica che indica il numero di mail precedentemente inviate al cliente;
- NUM_OPEN_PREV: variabile numerica che indica il numero di email aperte dal cliente in passato;
- NUM_CLICK_PREV: variabile numerica che indica il numero di email clickate dal cliente in passato;
- NUM_FAIL_PREV: variabile numerica che indica il numero di email che non sono state aperte dal cliente in passato, volontariamente o a causa di errori;
- OPEN_RATE_PREV: variabile numerica che indica la percentuale di email aperte dal cliente in passato sul totale delle mail ricevute;
- CLICK_RATE_PREV: variabile numerica che indica la percentuale di email clickate dal cliente in passato sul totale delle mail ricevute;

- W_SEND_PREV: variabile booleana che indica se in passato il cliente ha ricevuto altre email dello stesso tipo;
- W_FAIL_PREV: variabile booleana che indica se in passato il cliente ha rimbalzato o non ha ricevuto a causa di errori altre email dello stesso tipo;
- SEND_WEEKDAY: variabile categorica che indica il giorno della settimana in cui Ã Â¨ stata inviata l'email;
- ID_NEG: id dello store di riferimento;
- TYP_CLI_FID: variabile booleana che indica che se l'account del cliente Ã Â¨ quello principale o meno;
- COD_FID: variabile categorica che indica il tipo del programma fedeltÃ Â del cliente (standard, premium, etc.);
- STATUS_FID: variabile booleana che indica se l'account Ã Â¨ attivo o meno;
- NUM_FIDs: variabile numerica che indica il numero di fidelity programs del cliente;
- AGE_FID: variabile numerica che indica da quanti giorni Ã Â¨ attivo il programma fedeltÃ Â del cliente;
- W_PHONE: variabile booleana che indica se il cliente ha inserito o meno il numero di telefonoM
- TYP_CLI_ACCOUNT: variabile categorica che indica il tipo di account del cliente;
- TYP_JOB: variabile categorica che indica il lavoro del cliente;
- EMAIL_PROVIDER_CLEAN: variabile categorica che indica l'email provider del cliente (clean Ã Â¨ dovuto al fatto che gli email providers che avevano una frequenza minore sono stati raggruppati in "others");
- PRV: variabile categorica che indica la provincia di residenza del cliente;
- REGION: variabile categorica che indica la regione di residenza del cliente;
- FLAG_PRIVACY_1: variabile booleana che indica se il cliente ha dato il consenso alla privacy;
- FLAG_PRIVACY_2: variabile booleana che indica se il cliente ha dato il consenso al profiling;
- FLAG_PRIVACY_MKT variabile booleana che indica se il cliente ha dato il consenso al direct marketing.

```
#### LIBRARIES ####
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(magrittr)
library(ggplot2)
library(forcats)
library(grid)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

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

```
data_path <- "Laboratorio/"
```

```
### clients fidelity subscriptions
df_1_cli_fid <- read.csv2(paste0(data_path, "raw_1_cli_fid.csv"), na.strings = c("NA", ""))

# clients accounts details
df_2_cli_account <- read.csv2(paste0(data_path, "raw_2_cli_account.csv"), na.strings = c("NA", ""))

### clients addresses ###
df_3_cli_address <- read.csv2(paste0(data_path,"raw_3_cli_address.csv"), na.strings = c(""), stringsAsFa

### clients privacy ###
df_4_cli_privacy <- read.csv2(paste0(data_path,"raw_4_cli_privacy.csv"), na.strings = c("NA", ""))

### email campaign characterization ###
df_5_camp_cat <- read.csv2(paste0(data_path,"raw_5_camp_cat.csv") , na.strings = c("NA", ""))

### email event ###
df_6_camp_event <- read.csv2(paste0(data_path,"raw_6_camp_event.csv") , na.strings = c("NA", ""))

df_7_tic <- read.csv2(paste0(data_path,"raw_7_tic.csv") , na.strings = c("NA", ""))
```

df_1_cli_fid

DATA CLEANING

```
str(df_1_cli_fid)
```

```
## 'data.frame':    370135 obs. of  7 variables:
##  $ ID_CLI     : int  500 16647 835335 9557 767877 743090 768948 813156 766232 773214 ...
##  $ ID_FID     : int  814583 781106 816369 746573 741522 776971 742716 791681 739769 752897 ...
##  $ ID_NEG     : int  32 44 28 9 41 2 31 45 4 5 ...
##  $ TYP_CLI_FID: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ COD_FID    : Factor w/ 4 levels "PREMIUM","PREMIUM BIZ",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ STATUS_FID : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ DT_ACTIVE  : Factor w/ 494 levels "2018-01-01","2018-01-02",..: 419 398 419 376 372 395 373 404 37
```

```
summary(df_1_cli_fid)
```

```
##      ID_CLI           ID_FID           ID_NEG      TYP_CLI_FID
##  Min.   :     1   Min.   :     3   Min.   : 1.0   Min.   :0.0000
##  1st Qu.:230659   1st Qu.:229067   1st Qu.: 6.0   1st Qu.:1.0000
##  Median :462034   Median :458969   Median :23.0   Median :1.0000
##  Mean   :462486   Mean   :459425   Mean   :22.1   Mean   :0.9848
##  3rd Qu.:693200   3rd Qu.:688435   3rd Qu.:36.0   3rd Qu.:1.0000
##  Max.   :934919   Max.   :928121   Max.   :49.0   Max.   :1.0000
##
##          COD_FID         STATUS_FID        DT_ACTIVE
##  PREMIUM     : 44029   Min.   :0.00   2018-11-23:  3024
##  PREMIUM BIZ :  6715   1st Qu.:1.00   2018-04-07:  1457
##  STANDARD    :290170   Median :1.00   2018-11-22:  1439
##  STANDARD BIZ: 29221   Mean   :0.99   2018-03-11:  1438
##                        3rd Qu.:1.00   2018-04-14:  1403
##                        Max.   :1.00   2018-04-28:  1403
##                                       (Other)   :359971
```

```
## cleaning dataset 1##
df_1_cli_fid_clean <- df_1_cli_fid


## formatting la data e le numerical categories as factor ##
df_1_cli_fid_clean <- df_1_cli_fid_clean %>%
  mutate(DT_ACTIVE = as.Date(DT_ACTIVE)) %>%
  mutate(ID_NEG = as.factor(ID_NEG)) %>%
  mutate(TYP_CLI_FID = as.factor(TYP_CLI_FID)) %>%
  mutate(STATUS_FID = as.factor(STATUS_FID))

## (consistency control) number of fid per client ##
#per ogni cliente ho il numero di fidelity program e di date active che Ã Â¨ la data in cui lo ha attiva
#ad esempio ci sono clienti che in una sola data hanno attivato 3 programmi fedeltÃ Â .
num_fid_x_cli <- df_1_cli_fid_clean %>%
  group_by(ID_CLI) %>%
  summarize(NUM_FIDs =  n_distinct(ID_FID), NUM_DATEs = n_distinct(DT_ACTIVE))
#impostiamo quindi una tabella riassuntiva:
dist_num_fid_x_cli <- num_fid_x_cli %>%
  group_by(NUM_FIDs, NUM_DATEs) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI))
```

```
#closer look sui clienti con piÃ¹ di un programma fedeltÃ Â
num_fid_x_cli %>% filter(NUM_DATEs == 3)
```

```
## # A tibble: 5 x 3
##   ID_CLI NUM_FIDs NUM_DATEs
##    <int>    <int>     <int>
## 1   7533        3         3
## 2 223203        3         3
## 3 621814        3         3
## 4 648813        3         3
## 5 662651        3         3
```

```
df_1_cli_fid %>% filter(ID_CLI == 621814)
```

```
##   ID_CLI ID_FID ID_NEG TYP_CLI_FID  COD_FID STATUS_FID  DT_ACTIVE
## 1 621814 578123      1           1 STANDARD          0 2018-10-13
## 2 621814 646483     18           1 STANDARD          0 2018-11-13
## 3 621814 661124     18           1 STANDARD          1 2018-11-20
```

```
## keep both first fid and last fid ##
# first --> registration date
# last --> features
df_1_cli_fid_first <- df_1_cli_fid_clean %>%
  group_by(ID_CLI) %>%
  filter(DT_ACTIVE == min(DT_ACTIVE)) %>%
  arrange(ID_FID) %>%
  filter(row_number() == 1) %>%
  ungroup() %>%
  as.data.frame()
```

```
df_1_cli_fid_last <- df_1_cli_fid_clean %>%
  group_by(ID_CLI) %>%
  filter(DT_ACTIVE == max(DT_ACTIVE)) %>%
  arrange(desc(ID_FID)) %>%
```

```
  filter(row_number() == 1) %>%
  ungroup() %>%
  as.data.frame()
```

```
df_1_cli_fid_clean <- df_1_cli_fid_last %>%
  left_join(df_1_cli_fid_first %>%
              select(c(ID_CLI, FIRST_ID_NEG = ID_NEG, FIRST_DT_ACTIVE = DT_ACTIVE))
            , by = 'ID_CLI') %>%
  left_join(num_fid_x_cli %>%
              select(c(ID_CLI, NUM_FIDs)) %>%
              mutate(NUM_FIDs = as.factor(NUM_FIDs))
            , by = 'ID_CLI')
```

```
## lets review ##
str(df_1_cli_fid_clean)
```

```
## 'data.frame':    369472 obs. of  10 variables:
##  $ ID_CLI         : int  199060 613049 648813 914880 342639 816890 898295 178553 918274 111479 ...
##  $ ID_FID         : int  928121 928118 928116 928115 928112 928110 928106 928101 928100 928099 ...
##  $ ID_NEG         : Factor w/ 49 levels "1","2","3","4",..: 19 21 6 43 46 24 34 8 17 31 ...
##  $ TYP_CLI_FID    : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ COD_FID        : Factor w/ 4 levels "PREMIUM","PREMIUM BIZ",..: 2 1 3 3 2 3 3 3 4 4 ...
##  $ STATUS_FID     : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 1 2 ...
##  $ DT_ACTIVE      : Date, format: "2019-05-11" "2019-05-11" ...
##  $ FIRST_ID_NEG   : Factor w/ 49 levels "1","2","3","4",..: 19 21 1 43 26 24 34 10 17 31 ...
##  $ FIRST_DT_ACTIVE: Date, format: "2018-02-18" "2019-03-30" ...
##  $ NUM_FIDs       : Factor w/ 4 levels "1","2","3","4": 2 2 3 2 2 2 2 2 2 2 ...
```

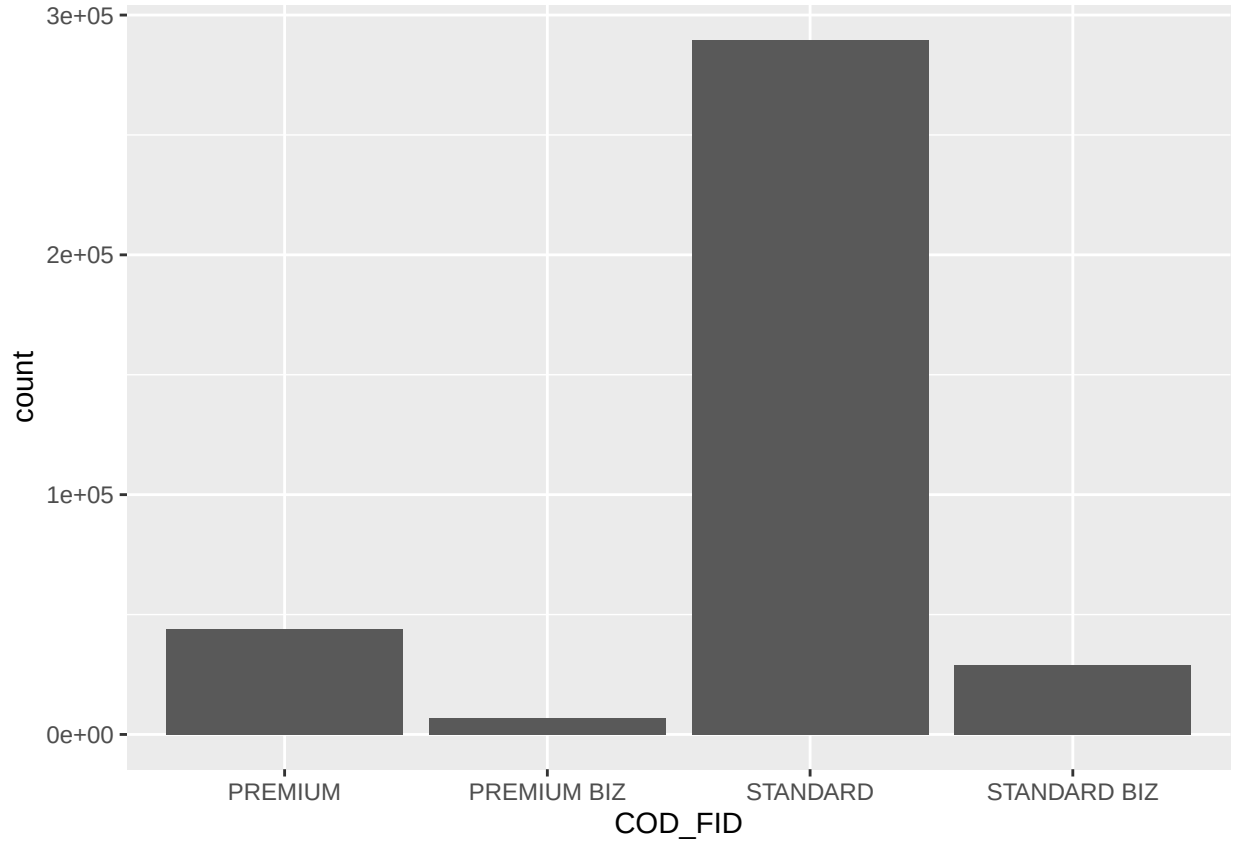
```
summary(df_1_cli_fid_clean)
```

```
##      ID_CLI           ID_FID           ID_NEG       TYP_CLI_FID
##  Min.   :     1   Min.   :     3   1      : 58174   0:  5607
##  1st Qu.:230783   1st Qu.:228931   38     : 11976   1:363865
##  Median :462063   Median :458832   34     : 11558
##  Mean   :462541   Mean   :459335   37     :  9948
##  3rd Qu.:693197   3rd Qu.:688330   33     :  9745
##  Max.   :934919   Max.   :928121   6      :  9677
##                                    (Other):258394
##          COD_FID       STATUS_FID   DT_ACTIVE           FIRST_ID_NEG
##  PREMIUM     : 43878   0:  3059   Min.   :2018-01-01   1      : 58244
##  PREMIUM BIZ :  6690   1:366413   1st Qu.:2018-04-15   38     : 11977
##  STANDARD    :289756              Median :2018-08-10   34     : 11553
##  STANDARD BIZ: 29148              Mean   :2018-08-14   37     :  9951
##                                   3rd Qu.:2018-11-30   33     :  9741
##                                   Max.   :2019-05-11   6      :  9673
##                                                        (Other):258333
##  FIRST_DT_ACTIVE      NUM_FIDs
##  Min.   :2018-01-01   1:368833
##  1st Qu.:2018-04-15   2:   617
##  Median :2018-08-09   3:    20
##  Mean   :2018-08-14   4:     2
##  3rd Qu.:2018-11-30
##  Max.   :2019-04-30
##
```

```
## explore distributions ##

# COD_FID
df_1_cli_fid_clean %>%
  group_by(COD_FID) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 4 x 3
##   COD_FID      TOT_CLIs PERCENT
##   <fct>           <int>   <dbl>
## 1 STANDARD       289756  0.784
## 2 PREMIUM         43878  0.119
## 3 STANDARD BIZ    29148  0.0789
## 4 PREMIUM BIZ      6690  0.0181
```

```
ggplot(df_1_cli_fid_clean, aes(x=COD_FID)) + geom_bar()
```

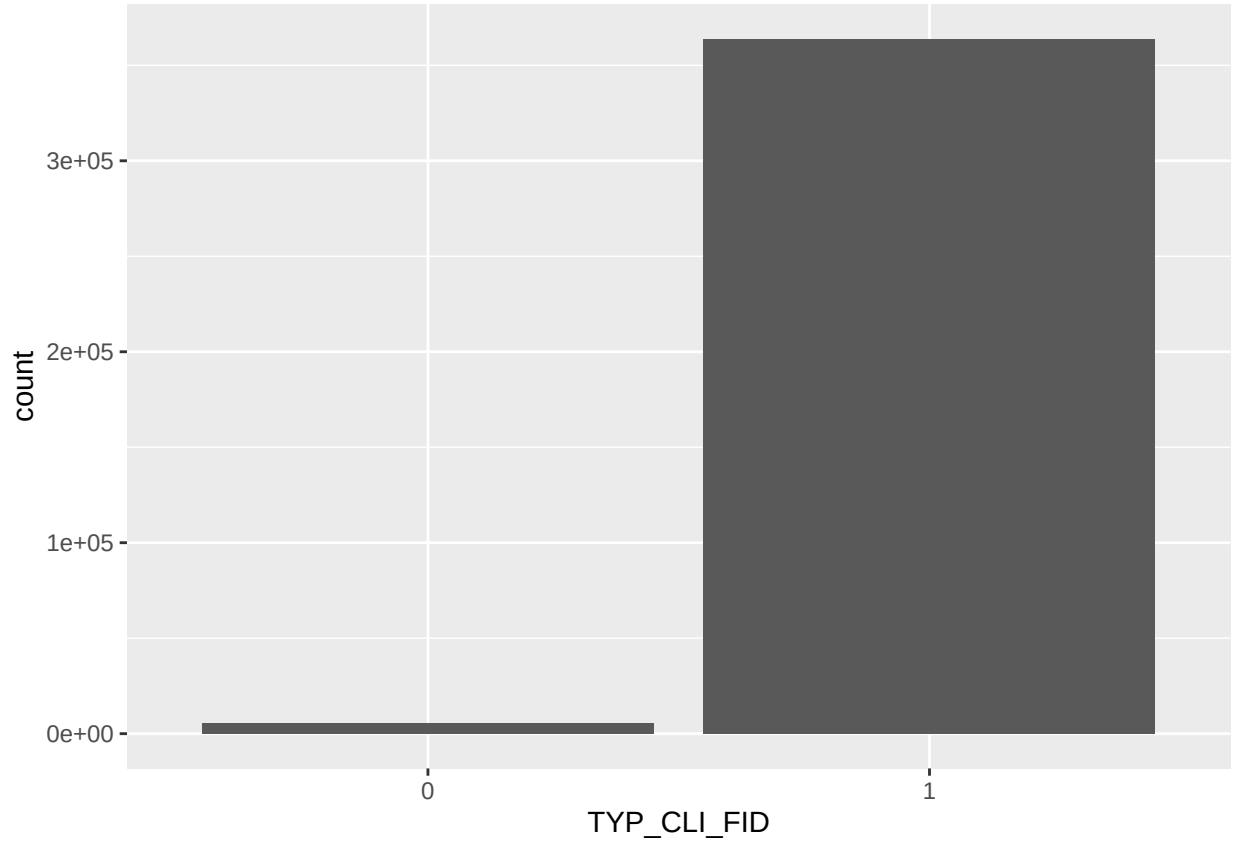



```
# TYP_CLI_FID
df_1_cli_fid_clean %>%
  group_by(TYP_CLI_FID) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 2 x 3
```

```
##   TYP_CLI_FID TOT_CLIs PERCENT
##   <fct>          <int>   <dbl>
## 1 1             363865  0.985
## 2 0               5607  0.0152
```

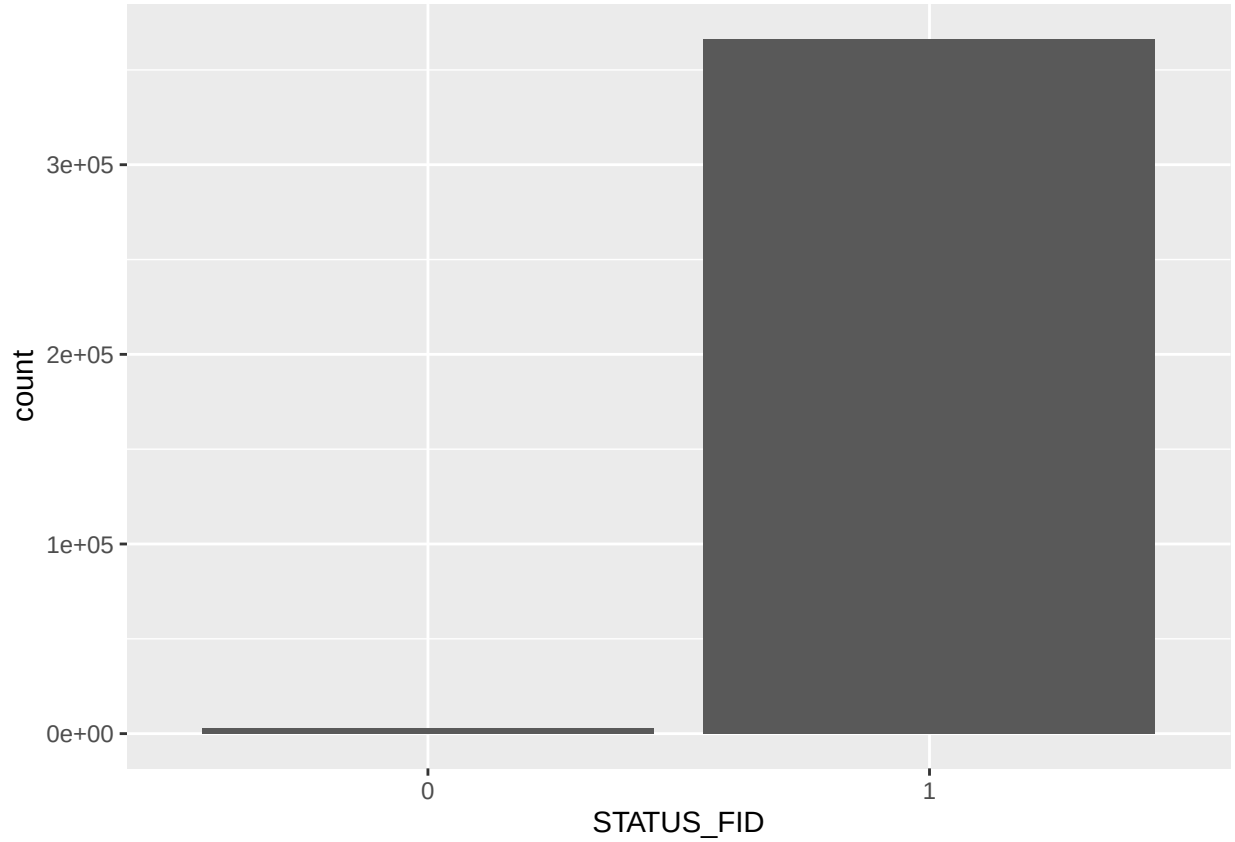
```
ggplot(df_1_cli_fid_clean, aes(x=TYP_CLI_FID)) + geom_bar()
```




```
# STATUS_FID
df_1_cli_fid_clean %>%
  group_by(STATUS_FID) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 2 x 3
##   STATUS_FID TOT_CLIs PERCENT
##   <fct>         <int>   <dbl>
## 1 1            366413 0.992
## 2 0              3059 0.00828
```

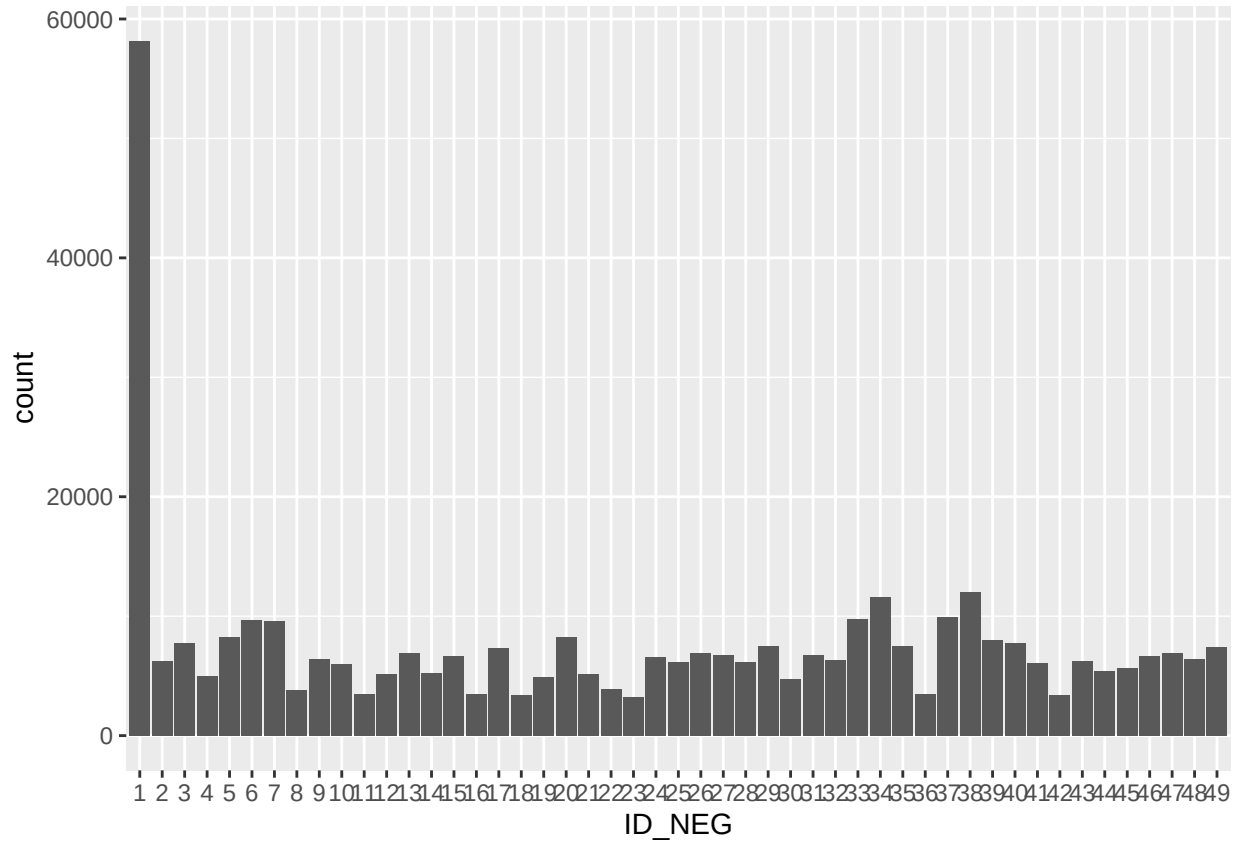
```
ggplot(df_1_cli_fid_clean, aes(x=STATUS_FID)) + geom_bar()
```




```
# ID_NEG
df_1_cli_fid_clean %>%
  group_by(ID_NEG) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs))  %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 49 x 3
##    ID_NEG TOT_CLIs PERCENT
##    <fct>     <int>   <dbl>
##  1 1         58174  0.157
##  2 38        11976  0.0324
##  3 34        11558  0.0313
##  4 37         9948  0.0269
##  5 33         9745  0.0264
##  6 6          9677  0.0262
##  7 7          9556  0.0259
##  8 5          8281  0.0224
##  9 20         8251  0.0223
## 10 39         8016  0.0217
## # ... with 39 more rows
```

```
ggplot(df_1_cli_fid_clean, aes(x=ID_NEG)) + geom_bar()
```



df_2_cli_account

```
str(df_2_cli_account)
```

```
## 'data.frame':    369472 obs. of  6 variables:
##  $ ID_CLI         : int  600125 729642 304639 292497 589492 638815 658338 883322 508481 871293 ...
##  $ EMAIL_PROVIDER : Factor w/ 20511 levels "126.com","163.com",..: 203 100 20501 203 100 259 203 203
##  $ W_PHONE        : int  NA NA 1 1 NA 1 1 1 1 1 ...
##  $ ID_ADDRESS     : int  584621 714144 284176 272563 573304 622947 642716 856157 488177 847360 ...
##  $ TYP_CLI_ACCOUNT: int  4 4 4 4 2 2 4 4 4 4 ...
##  $ TYP_JOB        : Factor w/ 14 levels "Altro","Artigiano",..: NA NA NA NA NA NA NA NA NA NA ...
```

```
summary(df_2_cli_account)
```

```
##      ID_CLI          EMAIL_PROVIDER      W_PHONE        ID_ADDRESS
##  Min.   :     1   gmail.com :151508   Min.   :1       Min.   :     1
##  1st Qu.:230783   libero.it : 57782   1st Qu.:1       1st Qu.:227903
##  Median :462063   hotmail.it: 28698   Median :1       Median :456720
##  Mean   :462541   alice.it  : 18127   Mean   :1       Mean   :457283
##  3rd Qu.:693197   yahoo.it  : 16538   3rd Qu.:1       3rd Qu.:686533
##  Max.   :934919   (Other)   : 90930   Max.   :1       Max.   :900091
##                   NA's      :  5889   NA's   :27305
##  TYP_CLI_ACCOUNT                  TYP_JOB
##  Min.   :2.000   Libero professionista:  3970
##  1st Qu.:4.000   Impiegato/a          :  1560
##  Median :4.000   Altro                :   784
##  Mean   :3.806   Pensionato/a         :   641
```

```
##  3rd Qu.:4.000   Operaio/a            :   482
##  Max.   :4.000   (Other)              :  1225
##                  NA's                 :360810
```

```
## cleaning##
df_2_cli_account_clean <- df_2_cli_account

## formatting boolean as factor e numerical categories as factor  ##
df_2_cli_account_clean <- df_2_cli_account_clean %>%
  mutate(W_PHONE = as.factor(W_PHONE))%>%
  mutate(TYP_CLI_ACCOUNT = as.factor(TYP_CLI_ACCOUNT))

## correct NA in categories ##
# we make use of the package forcats
library(forcats)

df_2_cli_account_clean <- df_2_cli_account_clean %>%
  mutate(W_PHONE = fct_explicit_na(W_PHONE, "0")) %>%
  mutate(EMAIL_PROVIDER = fct_explicit_na(EMAIL_PROVIDER, "(missing)")) %>%
  mutate(TYP_JOB = fct_explicit_na(TYP_JOB, "(missing)"))
```

```
## explore distributions ##
# COD_FID
df_2_cli_account_clean %>%
  group_by(EMAIL_PROVIDER) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 20,512 x 3
##    EMAIL_PROVIDER TOT_CLIs PERCENT
##    <fct>             <int>   <dbl>
##  1 gmail.com        151508  0.410
##  2 libero.it         57782  0.156
##  3 hotmail.it        28698  0.0777
##  4 alice.it          18127  0.0491
##  5 yahoo.it          16538  0.0448
##  6 hotmail.com       10076  0.0273
##  7 virgilio.it        9161  0.0248
##  8 tiscali.it         8733  0.0236
##  9 live.it            7936  0.0215
## 10 (missing)          5889  0.0159
## # ... with 20,502 more rows
```

```
df_2_cli_account_clean %>%
  summarize(TOT_EMAIL_PROVIDER = n_distinct(EMAIL_PROVIDER))
```

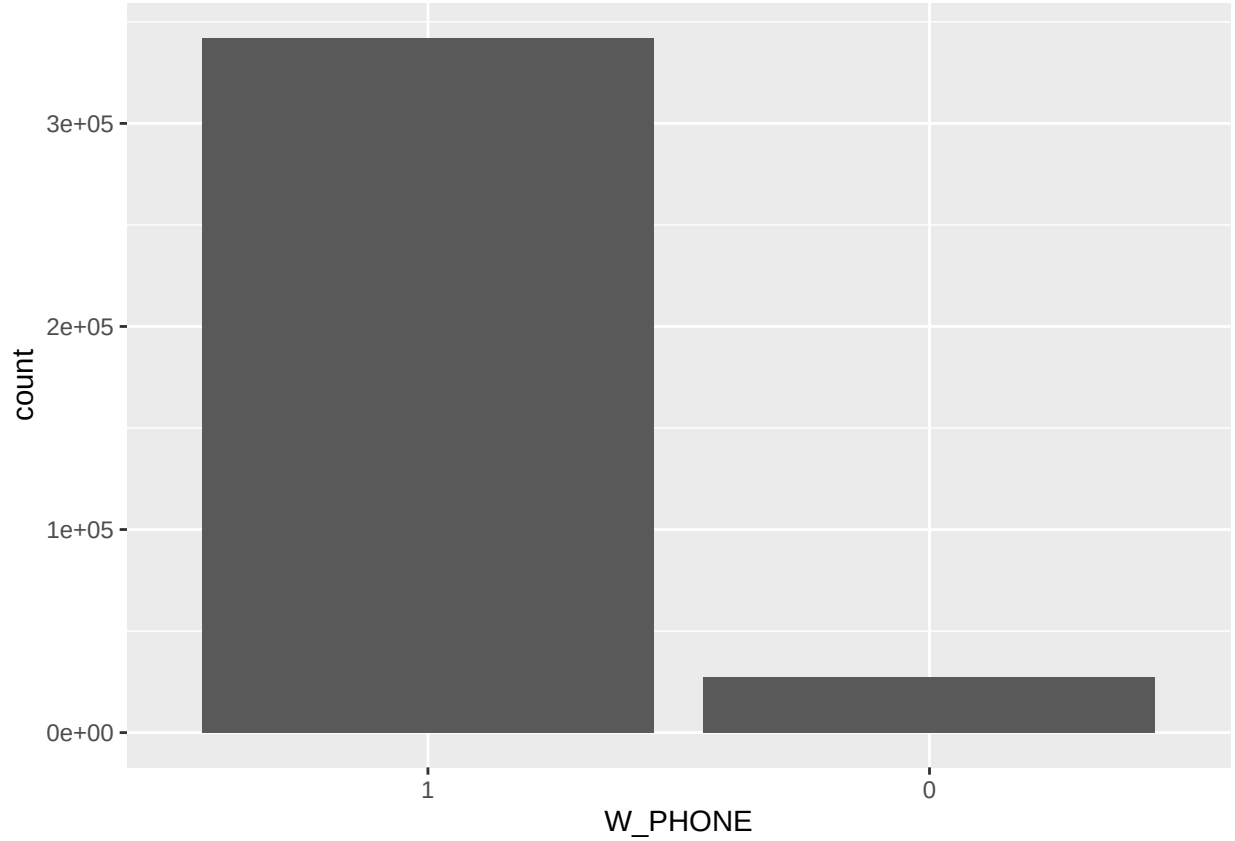
```
##   TOT_EMAIL_PROVIDER
## 1              20512
```

```
# too many different values for EMAIL_PROVIDER to be an useful category

# W_PHONE
df_2_cli_account_clean %>%
  group_by(W_PHONE) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
```

```
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 2 x 3
##   W_PHONE TOT_CLIs PERCENT
##   <fct>      <int>   <dbl>
## 1 1         342167  0.926
## 2 0          27305  0.0739
```

```
ggplot(df_2_cli_account_clean, aes(x=W_PHONE)) + geom_bar()
```




```
# TYP_JOB
df_2_cli_account_clean %>%
  group_by(TYP_JOB) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs)) %>%
  arrange(desc(PERCENT))
```

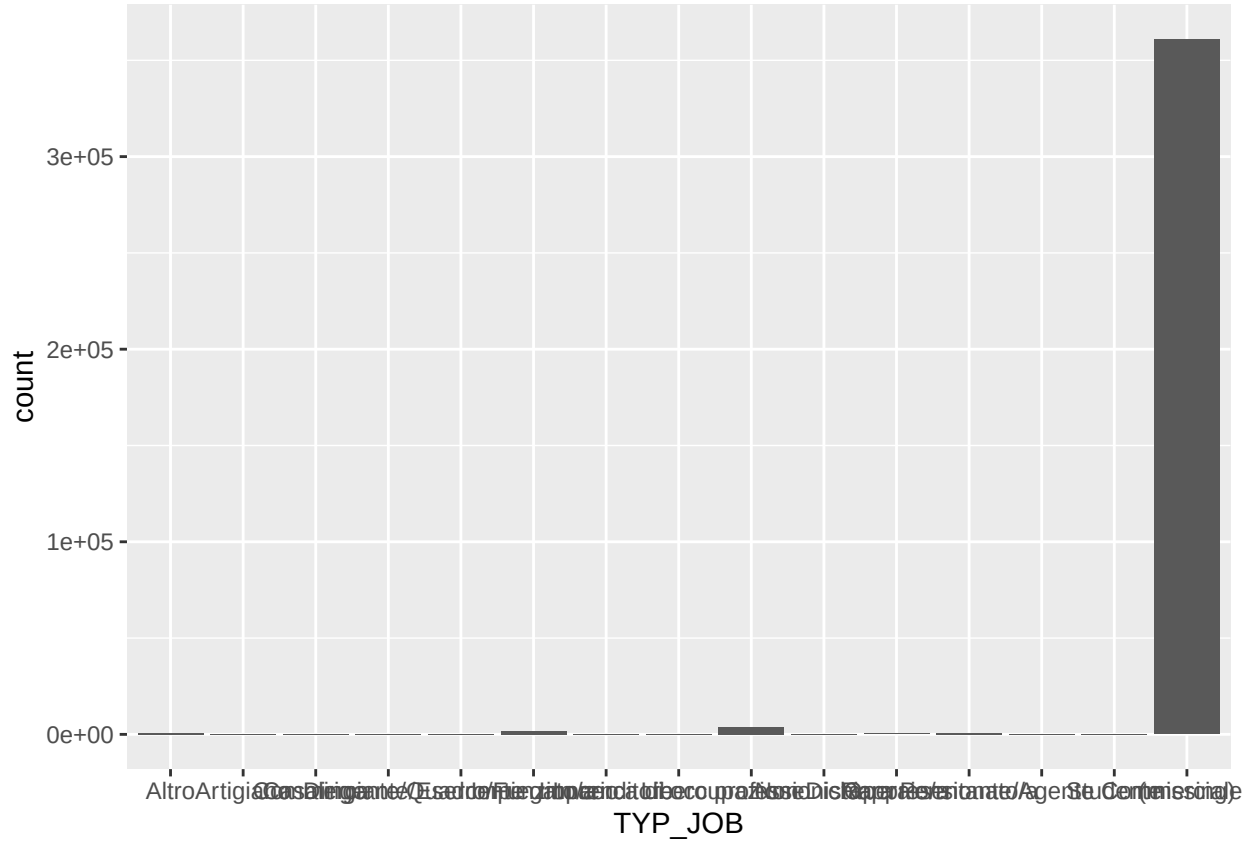
```
## # A tibble: 15 x 3
##    TYP_JOB                          TOT_CLIs   PERCENT
##    <fct>                               <int>     <dbl>
##  1 (missing)                          360810 0.977
##  2 Libero professionista                3970 0.0107
##  3 Impiegato/a                          1560 0.00422
##  4 Altro                                 784 0.00212
##  5 Pensionato/a                          641 0.00173
##  6 Operaio/a                             482 0.00130
```

```
##  7 Dirigente/Quadro/Funzionario           354 0.000958
##  8 Non Dichiara                           218 0.000590
##  9 Casalinga                              191 0.000517
## 10 Artigiano                              131 0.000355
## 11 Imprenditore                           108 0.000292
## 12 In cerca di occupazione                 79 0.000214
## 13 Commerciante/Esercente                  71 0.000192
## 14 Studente                                54 0.000146
## 15 Rappresentante/Agente Commerciale       19 0.0000514
```

```
df_2_cli_account_clean %>%
  summarize(TOT_TYP_JOB = n_distinct(TYP_JOB))
```

```
##   TOT_TYP_JOB
## 1          15
```

```
ggplot(df_2_cli_account_clean, aes(x=TYP_JOB)) + geom_bar()
```




```
## lets review ##
str(df_2_cli_account_clean)
```

```
## 'data.frame':    369472 obs. of  6 variables:
##  $ ID_CLI         : int  600125 729642 304639 292497 589492 638815 658338 883322 508481 871293 ...
##  $ EMAIL_PROVIDER : Factor w/ 20512 levels "126.com","163.com",..: 203 100 20501 203 100 259 203 203
##  $ W_PHONE        : Factor w/ 2 levels "1","0": 2 2 1 1 2 1 1 1 1 1 ...
##  $ ID_ADDRESS     : int  584621 714144 284176 272563 573304 622947 642716 856157 488177 847360 ...
##  $ TYP_CLI_ACCOUNT: Factor w/ 2 levels "2","4": 2 2 2 2 1 1 2 2 2 2 ...
##  $ TYP_JOB        : Factor w/ 15 levels "Altro","Artigiano",..: 15 15 15 15 15 15 15 15 15 15 ...
```

```r
summary(df_2_cli_account_clean)
```

```
##      ID_CLI           EMAIL_PROVIDER   W_PHONE      ID_ADDRESS
##  Min.   :     1   gmail.com  :151508   1:342167   Min.   :     1
##  1st Qu.:230783   libero.it  : 57782   0: 27305   1st Qu.:227903
##  Median :462063   hotmail.it : 28698              Median :456720
##  Mean   :462541   alice.it   : 18127              Mean   :457283
##  3rd Qu.:693197   yahoo.it   : 16538              3rd Qu.:686533
##  Max.   :934919   hotmail.com: 10076              Max.   :900091
##                   (Other)    : 86743
##  TYP_CLI_ACCOUNT                 TYP_JOB
##  2: 35816        (missing)            :360810
##  4:333656        Libero professionista:  3970
##                  Impiegato/a          :  1560
##                  Altro                :   784
##                  Pensionato/a         :   641
##                  Operaio/a            :   482
##                  (Other)              :  1225
```

```r
#too many missing values for EMAIL_PROVIDER to be an useful category
#keep the most frequent values and (missing) while changing the remaing into "OTHER"
freq_email_providers <- df_2_cli_account_clean %>%
  group_by(EMAIL_PROVIDER) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs)) %>%
  arrange(desc(PERCENT)) %>%
  mutate(PERCENT_COVERED = cumsum(TOT_CLIs)/sum(TOT_CLIs))

head(freq_email_providers, 20)
```

```
## # A tibble: 20 x 4
##    EMAIL_PROVIDER TOT_CLIs PERCENT PERCENT_COVERED
##    <fct>             <int>   <dbl>           <dbl>
##  1 gmail.com        151508 0.410            0.410
##  2 libero.it         57782 0.156            0.566
##  3 hotmail.it        28698 0.0777           0.644
##  4 alice.it          18127 0.0491           0.693
##  5 yahoo.it          16538 0.0448           0.738
##  6 hotmail.com       10076 0.0273           0.765
##  7 virgilio.it        9161 0.0248           0.790
##  8 tiscali.it         8733 0.0236           0.814
##  9 live.it            7936 0.0215           0.835
## 10 (missing)          5889 0.0159           0.851
## 11 icloud.com         3735 0.0101           0.861
## 12 yahoo.com          3259 0.00882          0.870
## 13 gmail.it           2266 0.00613          0.876
## 14 tin.it             2183 0.00591          0.882
## 15 outlook.it         2039 0.00552          0.888
## 16 fastwebnet.it      1749 0.00473          0.892
## 17 inwind.it          1514 0.00410          0.896
## 18 email.it           1103 0.00299          0.899
## 19 me.com             1034 0.00280          0.902
## 20 live.com            837 0.00227          0.904
```

```r
clean_email_providers <- freq_email_providers %>%
  mutate(EMAIL_PROVIDER = as.character(EMAIL_PROVIDER)) %>%
  mutate(AUX = if_else(PERCENT_COVERED < 0.85 | (PERCENT_COVERED > 0.85 & lag(PERCENT_COVERED) < 0.85),
  mutate(EMAIL_PROVIDER_CLEAN = if_else(AUX | EMAIL_PROVIDER == "(missing)", EMAIL_PROVIDER, "others"))

head(clean_email_providers, 20)
```

```
## # A tibble: 20 x 6
##    EMAIL_PROVIDER TOT_CLIs PERCENT PERCENT_COVERED   AUX EMAIL_PROVIDER_CL~
##    <chr>             <int>   <dbl>           <dbl> <dbl> <chr>
##  1 gmail.com        151508 0.410             0.410     1 gmail.com
##  2 libero.it         57782 0.156             0.566     1 libero.it
##  3 hotmail.it        28698 0.0777            0.644     1 hotmail.it
##  4 alice.it          18127 0.0491            0.693     1 alice.it
##  5 yahoo.it          16538 0.0448            0.738     1 yahoo.it
##  6 hotmail.com       10076 0.0273            0.765     1 hotmail.com
##  7 virgilio.it        9161 0.0248            0.790     1 virgilio.it
##  8 tiscali.it         8733 0.0236            0.814     1 tiscali.it
##  9 live.it            7936 0.0215            0.835     1 live.it
## 10 (missing)          5889 0.0159            0.851     1 (missing)
## 11 icloud.com         3735 0.0101            0.861     0 others
## 12 yahoo.com          3259 0.00882           0.870     0 others
## 13 gmail.it           2266 0.00613           0.876     0 others
## 14 tin.it             2183 0.00591           0.882     0 others
## 15 outlook.it         2039 0.00552           0.888     0 others
## 16 fastwebnet.it      1749 0.00473           0.892     0 others
## 17 inwind.it          1514 0.00410           0.896     0 others
## 18 email.it           1103 0.00299           0.899     0 others
## 19 me.com             1034 0.00280           0.902     0 others
## 20 live.com            837 0.00227           0.904     0 others
```

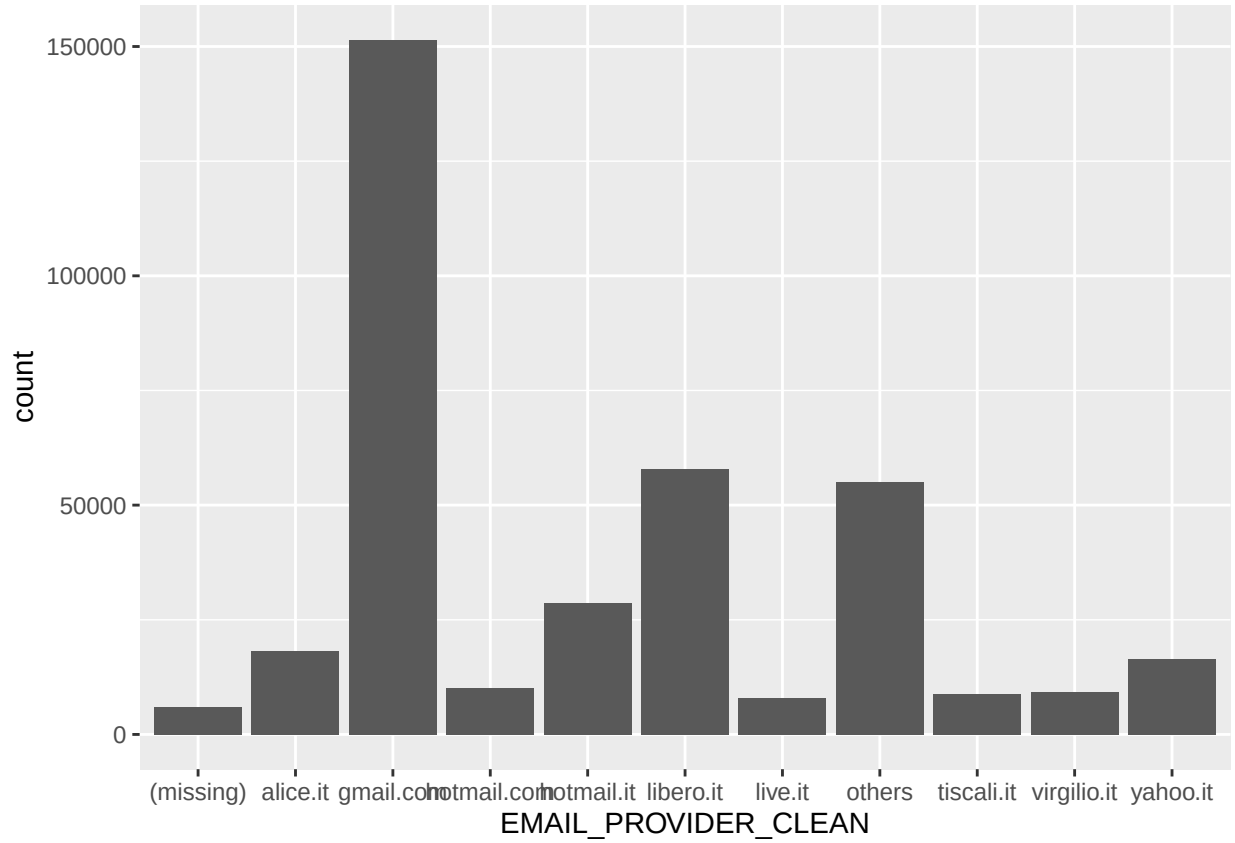
```r
df_2_cli_account_clean <- df_2_cli_account_clean %>%
  mutate(EMAIL_PROVIDER = as.character(EMAIL_PROVIDER)) %>%
  left_join(clean_email_providers %>%
              select(EMAIL_PROVIDER, EMAIL_PROVIDER_CLEAN)
            , by = "EMAIL_PROVIDER") %>%
  select(-EMAIL_PROVIDER) %>%
  mutate(EMAIL_PROVIDER_CLEAN = as.factor(EMAIL_PROVIDER_CLEAN))
```

```r
## explore distributions ##
# EMAIL_PROVIDER_CLEAN
df_2_cli_account_clean %>%
  group_by(EMAIL_PROVIDER_CLEAN) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 11 x 3
##    EMAIL_PROVIDER_CLEAN TOT_CLIs PERCENT
##    <fct>                   <int>   <dbl>
##  1 gmail.com              151508  0.410
##  2 libero.it               57782  0.156
##  3 others                  55024  0.149
##  4 hotmail.it              28698  0.0777
##  5 alice.it                18127  0.0491
```

```
##  6 yahoo.it                16538  0.0448
##  7 hotmail.com             10076  0.0273
##  8 virgilio.it              9161  0.0248
##  9 tiscali.it               8733  0.0236
## 10 live.it                  7936  0.0215
## 11 (missing)                5889  0.0159
```

```
ggplot(df_2_cli_account_clean, aes(x=EMAIL_PROVIDER_CLEAN)) + geom_bar()
```




```
## lets review ##
str(df_2_cli_account_clean)
```

```
## 'data.frame':    369472 obs. of  6 variables:
##  $ ID_CLI              : int  600125 729642 304639 292497 589492 638815 658338 883322 508481 871293
##  $ W_PHONE             : Factor w/ 2 levels "1","0": 2 2 1 1 2 1 1 1 1 1 ...
##  $ ID_ADDRESS          : int  584621 714144 284176 272563 573304 622947 642716 856157 488177 847360
##  $ TYP_CLI_ACCOUNT     : Factor w/ 2 levels "2","4": 2 2 2 2 1 1 2 2 2 2 ...
##  $ TYP_JOB             : Factor w/ 15 levels "Altro","Artigiano",..: 15 15 15 15 15 15 15 15 15 15 .
##  $ EMAIL_PROVIDER_CLEAN: Factor w/ 11 levels "(missing)","alice.it",..: 6 3 11 6 3 8 6 6 6 5 ...
```

```
summary(df_2_cli_account_clean)
```

```
##      ID_CLI       W_PHONE      ID_ADDRESS     TYP_CLI_ACCOUNT
##  Min.   :     1   1:342167   Min.   :     1   2: 35816
##  1st Qu.:230783   0: 27305   1st Qu.:227903   4:333656
##  Median :462063              Median :456720
##  Mean   :462541              Mean   :457283
##  3rd Qu.:693197              3rd Qu.:686533
```

```
##  Max.   :934919              Max.   :900091
##
##                   TYP_JOB       EMAIL_PROVIDER_CLEAN
##  (missing)            :360810   gmail.com :151508
##  Libero professionista:  3970   libero.it : 57782
##  Impiegato/a          :  1560   others    : 55024
##  Altro                :   784   hotmail.it: 28698
##  Pensionato/a         :   641   alice.it  : 18127
##  Operaio/a            :   482   yahoo.it  : 16538
##  (Other)              :  1225   (Other)   : 41795
```

df_3_cli_address

```
str(df_3_cli_address)
```

```
## 'data.frame':    1211332 obs. of  4 variables:
##  $ ID_ADDRESS: int  1337 1344 1347 1347 1347 1347 1347 1347 1347 1347 ...
##  $ CAP       : chr  "20083" "20024" "20090" "20090" ...
##  $ PRV       : chr  "MI" "MI" "MI" "MI" ...
##  $ REGION    : chr  "LOMBARDIA" "LOMBARDIA" "LOMBARDIA" "LOMBARDIA" ...
```

```
summary(df_3_cli_address)
```

```
##    ID_ADDRESS         CAP                PRV                REGION
##  Min.   :     1   Length:1211332     Length:1211332     Length:1211332
##  1st Qu.:221063   Class :character   Class :character   Class :character
##  Median :437083   Mode  :character   Mode  :character   Mode  :character
##  Mean   :443391
##  3rd Qu.:664931
##  Max.   :900090
```

```
## cleaning ##
df_3_cli_address_clean <- df_3_cli_address

## convert PRV e REGION into factors ##
df_3_cli_address_clean <- df_3_cli_address_clean %>%
  mutate(PRV = as.factor(PRV)) %>%
  mutate(REGION = as.factor(REGION)) %>%
  distinct()
```

```
# closer look on df_3_cli_address
df_3_cli_address_clean %>%
  group_by(w_CAP = !is.na(CAP), w_PRV = !is.na(PRV), w_REGION = !is.na(REGION)) %>%
  summarize(TOT_ADDs = n_distinct(ID_ADDRESS))
```

```
## # A tibble: 4 x 4
## # Groups:   w_CAP, w_PRV [3]
##   w_CAP w_PRV w_REGION TOT_ADDs
##   <lgl> <lgl> <lgl>       <int>
## 1 FALSE FALSE FALSE         121
## 2 TRUE  FALSE FALSE       23148
## 3 TRUE  TRUE  FALSE         595
## 4 TRUE  TRUE  TRUE       337466
```

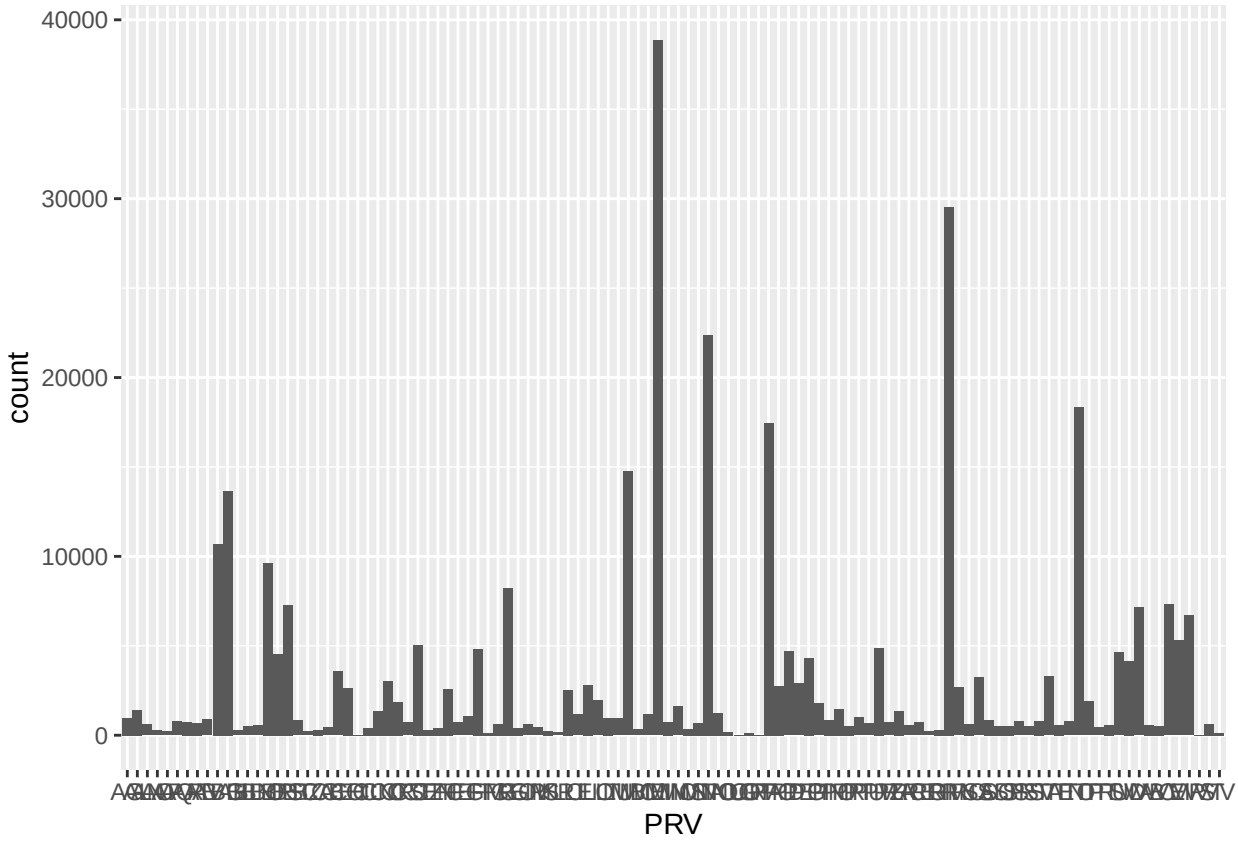
```
# drop the record without CAP - PRV - REGION
df_3_cli_address_clean <- df_3_cli_address_clean %>%
```

```
  filter(!is.na(CAP) & !is.na(PRV) & !is.na(REGION))
```

```
## explore distributions ##
# PRV
df_3_cli_address_clean %>%
  group_by(PRV) %>%
  summarize(TOT_ADDs = n_distinct(ID_ADDRESS)) %>%
  mutate(PERCENT = TOT_ADDs/sum(TOT_ADDs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 110 x 3
##    PRV   TOT_ADDs PERCENT
##    <fct>    <int>   <dbl>
##  1 MI       38850  0.115
##  2 RM       29529  0.0875
##  3 NA       22374  0.0663
##  4 TO       18322  0.0543
##  5 PA       17448  0.0517
##  6 MB       14751  0.0437
##  7 BG       13659  0.0405
##  8 BA       10698  0.0317
##  9 BO        9634  0.0285
## 10 GE        8234  0.0244
## # ... with 100 more rows
```

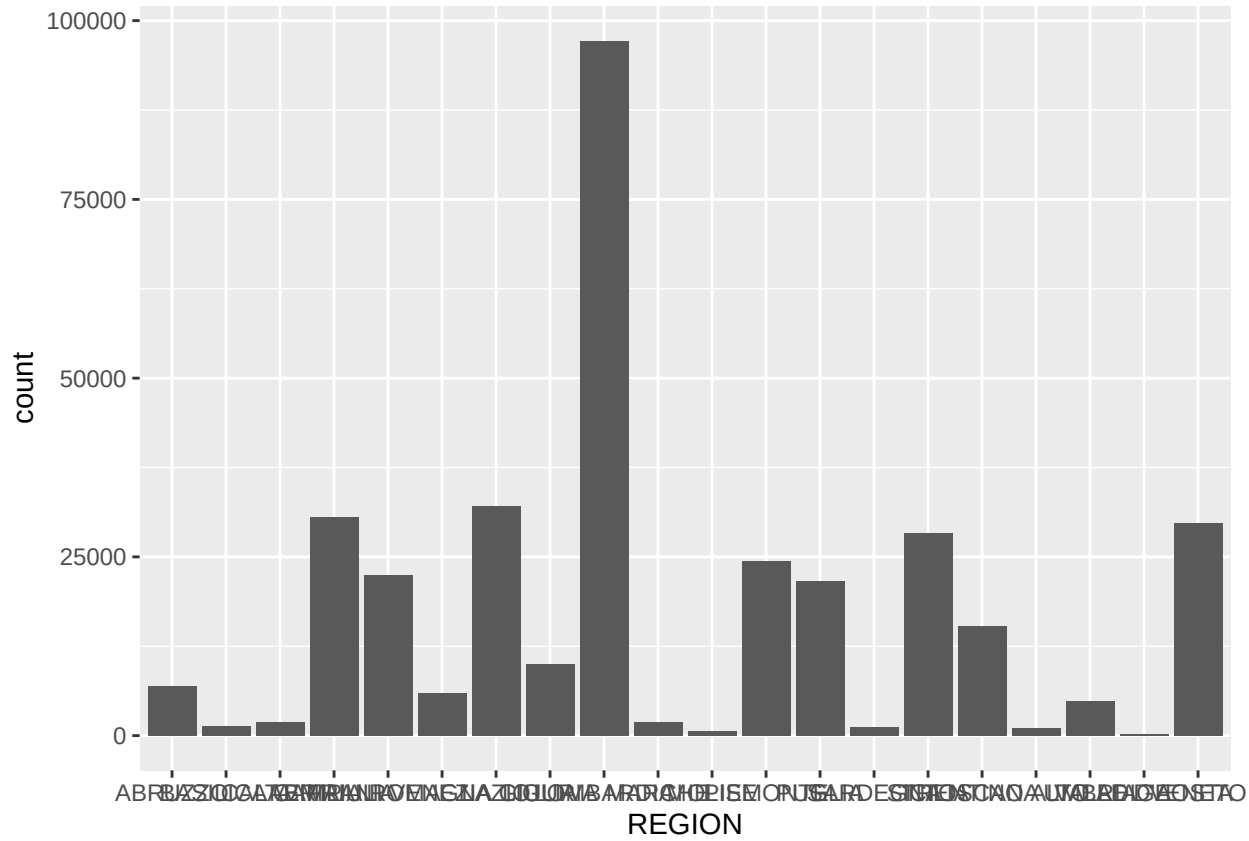
```
ggplot(df_3_cli_address_clean, aes(x=PRV)) + geom_bar()
```



```r
# REGION
df_3_cli_address_clean %>%
  group_by(REGION) %>%
  summarize(TOT_ADDs = n_distinct(ID_ADDRESS)) %>%
  mutate(PERCENT = TOT_ADDs/sum(TOT_ADDs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 20 x 3
##    REGION                TOT_ADDs  PERCENT
##    <fct>                    <int>    <dbl>
##  1 LOMBARDIA                97181 0.288
##  2 LAZIO                    32058 0.0950
##  3 CAMPANIA                 30570 0.0906
##  4 VENETO                   29696 0.0880
##  5 SICILIA                  28329 0.0839
##  6 PIEMONTE                 24377 0.0722
##  7 EMILIA ROMAGNA           22515 0.0667
##  8 PUGLIA                   21582 0.0640
##  9 TOSCANA                  15369 0.0455
## 10 LIGURIA                   9994 0.0296
## 11 ABRUZZO                   6856 0.0203
## 12 FRIULI VENEZIA GIULIA     5969 0.0177
## 13 UMBRIA                    4771 0.0141
## 14 MARCHE                    1931 0.00572
## 15 CALABRIA                  1837 0.00544
## 16 BASILICATA                1382 0.00410
## 17 SARDEGNA                  1124 0.00333
## 18 TRENTINO ALTO ADIGE        982 0.00291
## 19 MOLISE                     668 0.00198
## 20 VALLE D'AOSTA              277 0.000821
```

```r
ggplot(df_3_cli_address_clean, aes(x=REGION)) + geom_bar()
```




```
## lets review ##
str(df_3_cli_address_clean)
```

```
## 'data.frame':    337468 obs. of  4 variables:
##  $ ID_ADDRESS: int  1337 1344 1347 1352 1353 1355 1361 1379 1384 1387 ...
##  $ CAP       : chr  "20083" "20024" "20090" "20123" ...
##  $ PRV       : Factor w/ 241 levels "-",".","06061",..: 113 113 113 113 113 113 123 113 113 113 ...
##  $ REGION    : Factor w/ 20 levels "ABRUZZO","BASILICATA",..: 9 9 9 9 9 9 12 9 9 9 ...
```

```
summary(df_3_cli_address_clean)
```

```
##    ID_ADDRESS         CAP                 PRV              REGION
##  Min.   :     1   Length:337468      MI     : 38850   LOMBARDIA:97181
##  1st Qu.:219149   Class :character   RM     : 29529   LAZIO    :32058
##  Median :441844   Mode  :character   NA     : 22374   CAMPANIA :30570
##  Mean   :444313                      TO     : 18322   VENETO   :29696
##  3rd Qu.:667976                      PA     : 17448   SICILIA  :28329
##  Max.   :900090                      MB     : 14751   PIEMONTE :24377
##                                      (Other):196194   (Other)  :95257
```

df_4_cli_privacy

```
str(df_4_cli_privacy)
```

```
## 'data.frame':    369472 obs. of  4 variables:
##  $ ID_CLI         : int  4691 3434 3533 9866 5799 4660 8441 502 13290 6448 ...
##  $ FLAG_PRIVACY_1 : int  1 0 1 1 1 0 0 1 1 1 ...
```

```
##  $ FLAG_PRIVACY_2 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ FLAG_DIRECT_MKT: int  1 0 1 1 1 0 0 1 1 1 ...
```

```
summary(df_4_cli_privacy)
```

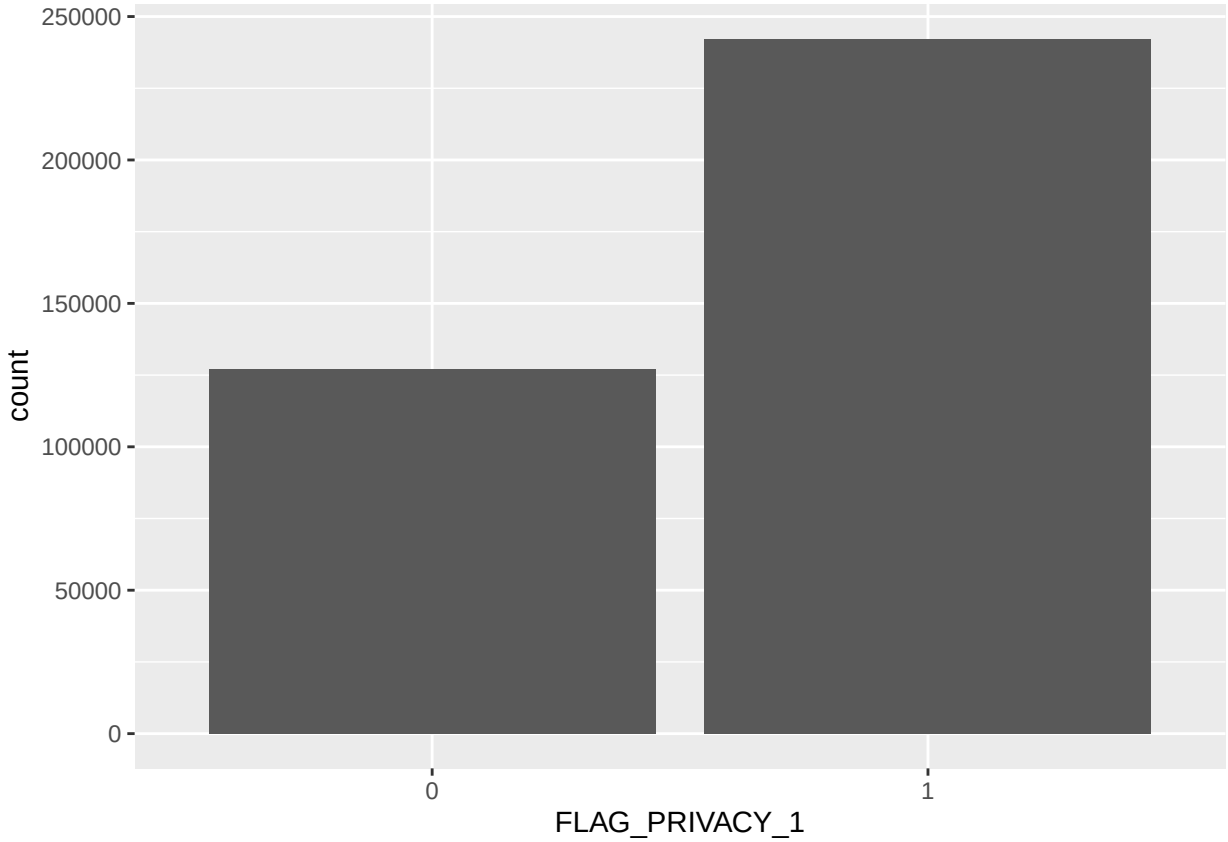
```
##      ID_CLI       FLAG_PRIVACY_1   FLAG_PRIVACY_2   FLAG_DIRECT_MKT
##  Min.   :     1   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:230783   1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:0.0000
##  Median :462063   Median :1.0000   Median :1.0000   Median :1.0000
##  Mean   :462541   Mean   :0.6557   Mean   :0.9356   Mean   :0.6707
##  3rd Qu.:693197   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :934919   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```
## cleaning ##
df_4_cli_privacy_clean <- df_4_cli_privacy

# formatting boolean into facotr
df_4_cli_privacy_clean <- df_4_cli_privacy_clean %>%
  mutate(FLAG_PRIVACY_1 = as.factor(FLAG_PRIVACY_1)) %>%
  mutate(FLAG_PRIVACY_2 = as.factor(FLAG_PRIVACY_2)) %>%
  mutate(FLAG_DIRECT_MKT = as.factor(FLAG_DIRECT_MKT))
```

```
## explore distributions ##
# FLAG_PRIVACY_1
df_4_cli_privacy_clean %>%
  group_by(FLAG_PRIVACY_1) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 2 x 3
##   FLAG_PRIVACY_1 TOT_CLIs PERCENT
##   <fct>             <int>   <dbl>
## 1 1                242251   0.656
## 2 0                127221   0.344
```

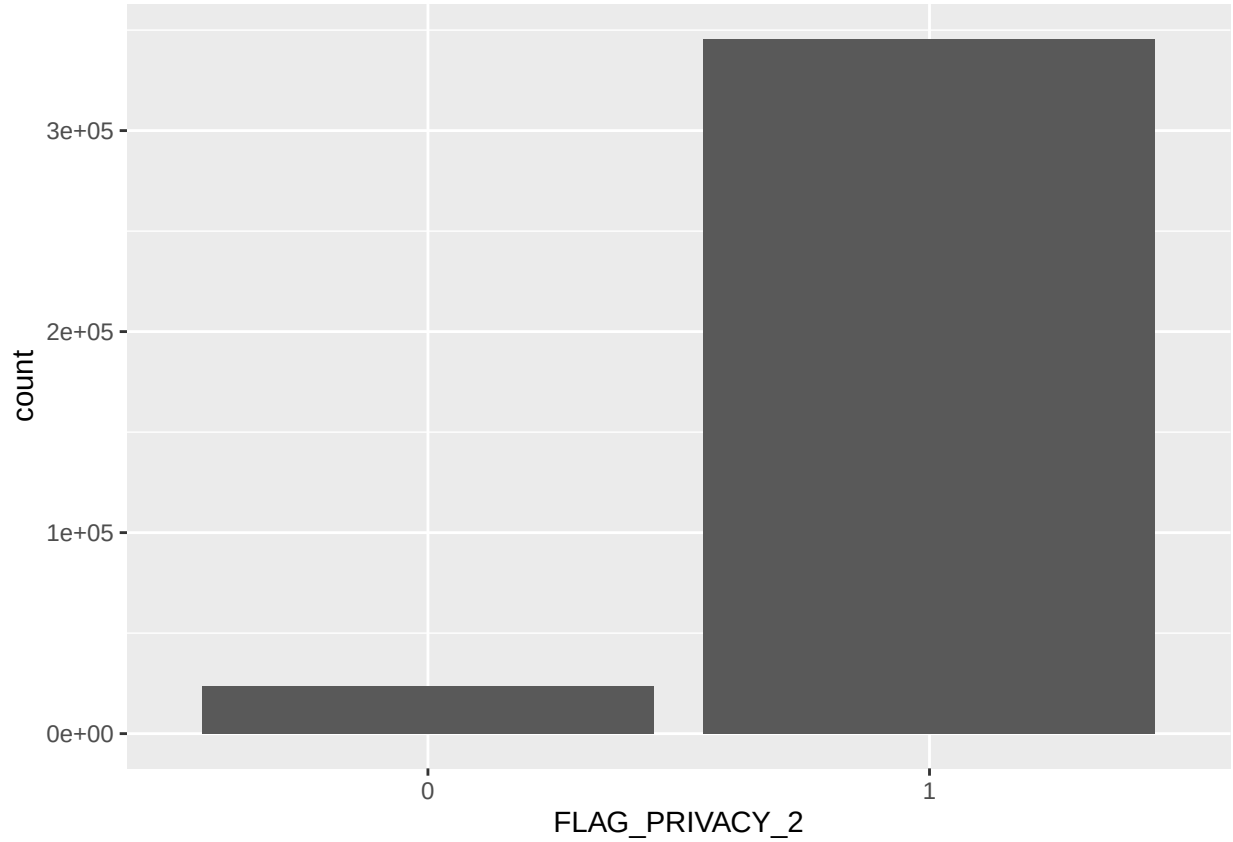
```
ggplot(df_4_cli_privacy_clean, aes(x=FLAG_PRIVACY_1)) + geom_bar()
```




```
# FLAG_PRIVACY_2
df_4_cli_privacy_clean %>%
  group_by(FLAG_PRIVACY_2) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 2 x 3
##   FLAG_PRIVACY_2 TOT_CLIs PERCENT
##   <fct>             <int>   <dbl>
## 1 1                345682  0.936
## 2 0                 23790  0.0644
```

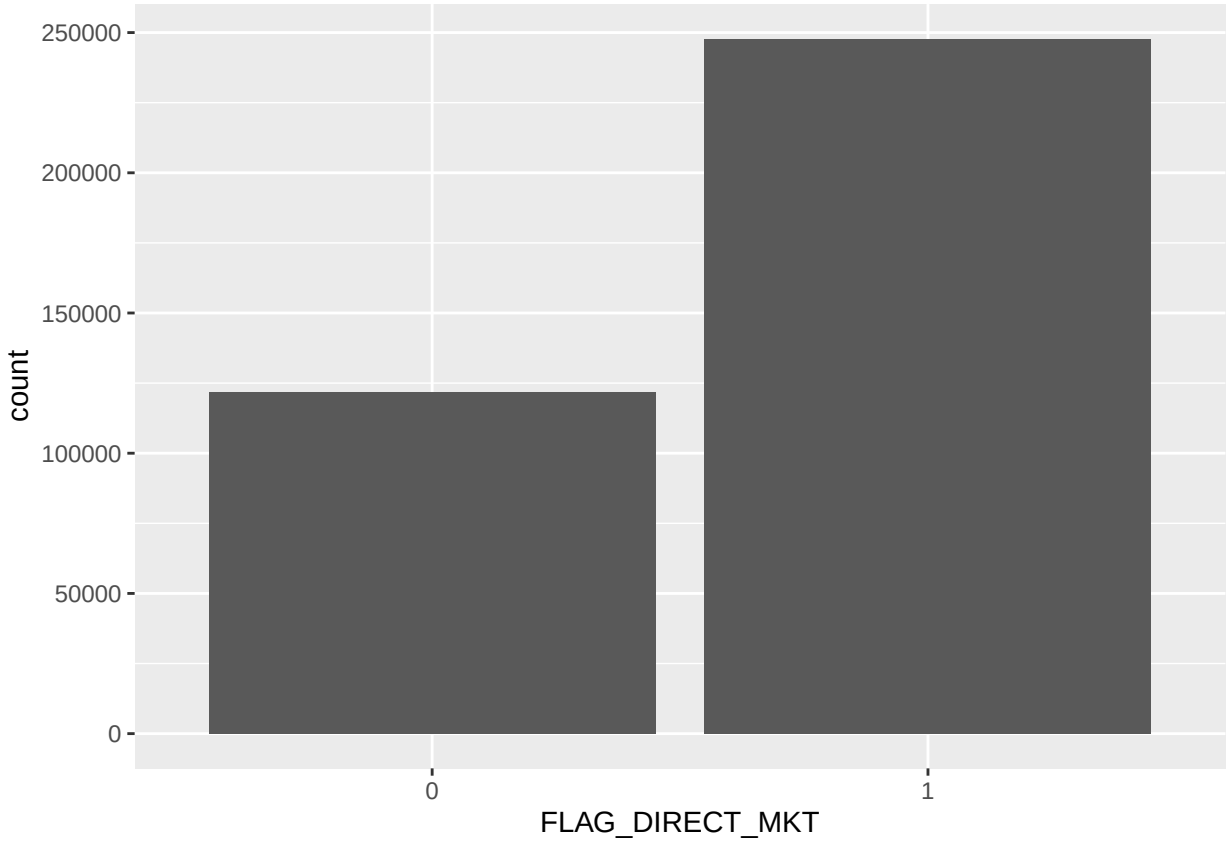
```
ggplot(df_4_cli_privacy_clean, aes(x=FLAG_PRIVACY_2)) + geom_bar()
```




```
# FLAG_DIRECT_MKT
df_4_cli_privacy_clean %>%
  group_by(FLAG_DIRECT_MKT) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 2 x 3
##   FLAG_DIRECT_MKT TOT_CLIs PERCENT
##   <fct>              <int>   <dbl>
## 1 1                247790   0.671
## 2 0                121682   0.329
```

```
ggplot(df_4_cli_privacy_clean, aes(x=FLAG_DIRECT_MKT)) + geom_bar()
```




```
df_4_cli_privacy_clean %>%
  group_by(FLAG_PRIVACY_1, FLAG_PRIVACY_2, FLAG_DIRECT_MKT) %>%
  summarize(TOT_CLIs = n_distinct(ID_CLI)) %>%
  mutate(PERCENT = TOT_CLIs/sum(TOT_CLIs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 8 x 5
## # Groups:   FLAG_PRIVACY_1, FLAG_PRIVACY_2 [4]
##   FLAG_PRIVACY_1 FLAG_PRIVACY_2 FLAG_DIRECT_MKT TOT_CLIs PERCENT
##   <fct>          <fct>          <fct>              <int>   <dbl>
## 1 1              0              1                  17042  0.983
## 2 1              1              1                 212323  0.944
## 3 0              1              0                 103527  0.857
## 4 0              0              0                   5269  0.816
## 5 0              0              1                   1187  0.184
## 6 0              1              1                  17238  0.143
## 7 1              1              0                  12594  0.0560
## 8 1              0              0                    292  0.0168
```

```
## lets review ##
str(df_4_cli_privacy_clean)
```

```
## 'data.frame':    369472 obs. of  4 variables:
##  $ ID_CLI         : int  4691 3434 3533 9866 5799 4660 8441 502 13290 6448 ...
##  $ FLAG_PRIVACY_1 : Factor w/ 2 levels "0","1": 2 1 2 2 2 1 1 2 2 2 ...
##  $ FLAG_PRIVACY_2 : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ FLAG_DIRECT_MKT: Factor w/ 2 levels "0","1": 2 1 2 2 2 1 1 2 2 2 ...
```

```
summary(df_4_cli_privacy_clean)
```

```
##      ID_CLI       FLAG_PRIVACY_1 FLAG_PRIVACY_2 FLAG_DIRECT_MKT
##  Min.   :     1   0:127221       0: 23790       0:121682
##  1st Qu.:230783   1:242251       1:345682       1:247790
##  Median :462063
##  Mean   :462541
##  3rd Qu.:693197
##  Max.   :934919
```

df_5_camp_cat

```
str(df_5_camp_cat)
```

```
## 'data.frame':    848 obs. of  3 variables:
##  $ ID_CAMP     : int  757 759 760 761 762 763 764 765 767 769 ...
##  $ TYP_CAMP    : Factor w/ 5 levels "LOCAL","NATIONAL",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ CHANNEL_CAMP: Factor w/ 1 level "EMAIL": 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(df_5_camp_cat)
```

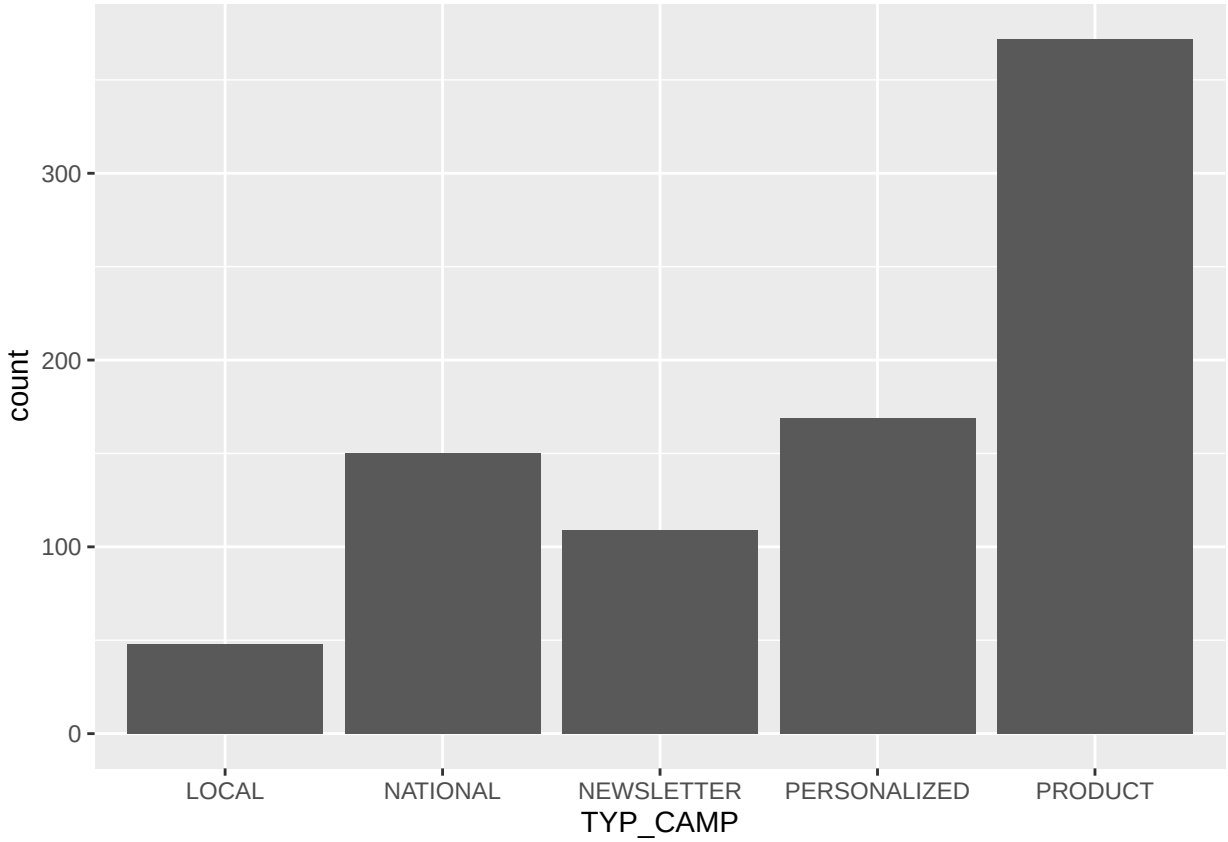
```
##     ID_CAMP               TYP_CAMP   CHANNEL_CAMP
##  Min.   :   5.0   LOCAL       : 48   EMAIL:848
##  1st Qu.: 327.8   NATIONAL    :150
##  Median : 561.5   NEWSLETTER  :109
##  Mean   : 559.6   PERSONALIZED:169
##  3rd Qu.: 812.2   PRODUCT     :372
##  Max.   :1052.0
```

```
## cleaning ##
df_5_camp_cat_clean <- df_5_camp_cat

# the field CHANNEL_CAMP has one value <-- is not relevant
df_5_camp_cat_clean <- df_5_camp_cat_clean %>%
  select(-CHANNEL_CAMP)
```

```
## explore distributions ##
df_5_camp_cat_clean %>%
  group_by(TYP_CAMP) %>%
  summarize(TOT_CAMPs = n_distinct(ID_CAMP)) %>%
  mutate(PERCENT = TOT_CAMPs/sum(TOT_CAMPs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 5 x 3
##   TYP_CAMP     TOT_CAMPs PERCENT
##   <fct>            <int>   <dbl>
## 1 PRODUCT            372  0.439
## 2 PERSONALIZED       169  0.199
## 3 NATIONAL           150  0.177
## 4 NEWSLETTER         109  0.129
## 5 LOCAL               48  0.0566
```

```
ggplot(df_5_camp_cat_clean, aes(x=TYP_CAMP)) + geom_bar()
```



df_6_camp_event

```
str(df_6_camp_event)
```

```
## 'data.frame':    2060360 obs. of  6 variables:
##  $ ID_EVENT   : int  11548588 11548640 11548572 11548515 11548609 11548497 11548441 11548490 1154862
##  $ ID_CLI     : int  411925 313259 327299 627427 265258 693938 610832 515395 59640 515395 ...
##  $ ID_CAMP    : int  948 949 941 923 950 955 946 829 951 951 ...
##  $ ID_DELIVERY: int  8996 8997 8817 8263 8998 9115 8970 8876 9032 9032 ...
##  $ TYP_EVENT  : Factor w/ 5 levels "B","C","E","S",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ EVENT_DATE : Factor w/ 403276 levels "2019-01-01T00:00:46",..: 20 38 17 12 25 8 1 5 30 6 ...
```

```
summary(df_6_camp_event)
```

```
##     ID_EVENT            ID_CLI          ID_CAMP        ID_DELIVERY
##  Min.   :11548441   Min.   :     1   Min.   : 148.0   Min.   : 7680
##  1st Qu.:13526812   1st Qu.:208129   1st Qu.: 970.0   1st Qu.:10066
##  Median :14867410   Median :398084   Median : 991.0   Median :10585
##  Mean   :14757544   Mean   :413793   Mean   : 928.9   Mean   :10534
##  3rd Qu.:16284762   3rd Qu.:618440   3rd Qu.:1024.0   3rd Qu.:11151
##  Max.   :17650340   Max.   :931973   Max.   :1048.0   Max.   :11509
##
##  TYP_EVENT                 EVENT_DATE
##  B:  31732   2019-04-04T17:52:18: 132336
##  C:  51678   2019-02-13T13:08:03: 125227
##  E:     65   2019-02-28T13:07:32: 125099
```

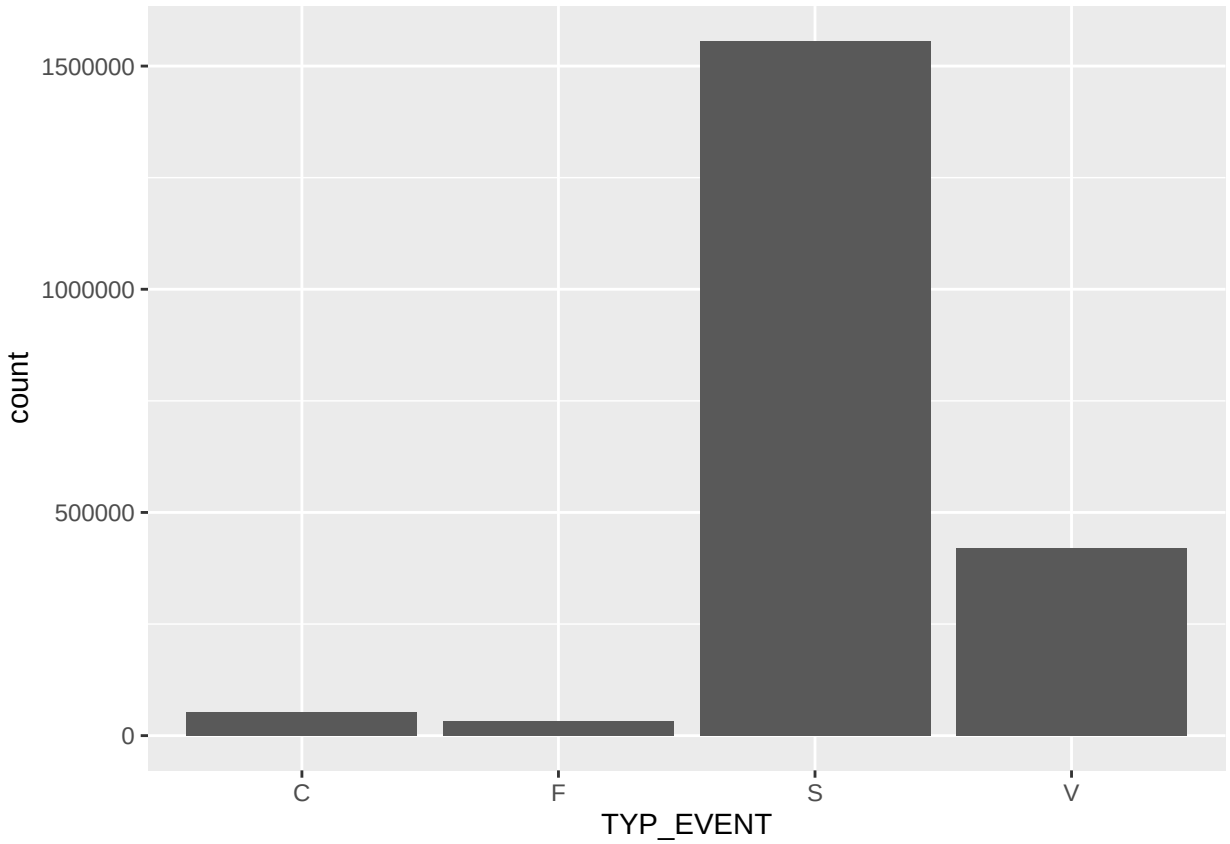
```
##  S:1556646   2019-01-08T13:08:27: 115029
##  V: 420239   2019-01-31T13:07:32: 113831
##              2019-04-11T13:06:45: 111317
##              (Other)            :1337521
```

```
## cleaning ##
df_6_camp_event_clean <- df_6_camp_event

# despite the field EVENT_TIME is datetime, we just need the corresponding dates
df_6_camp_event_clean <- df_6_camp_event_clean %>%
  mutate(EVENT_DATE = as.Date(EVENT_DATE, format="%Y-%m-%dT%H:%M:%S"))

# for the purpose of the analysis we are delivering here it would make no difference distinguish "ERROR
# lets combine them into a common category "FAILURE" with "F" as EVENT_CODE before changing the field t
df_6_camp_event_clean <- df_6_camp_event_clean %>%
  mutate(TYP_EVENT = as.factor(if_else(TYP_EVENT == "E" | TYP_EVENT == "B", "F", as.character(TYP_EVENT)
```

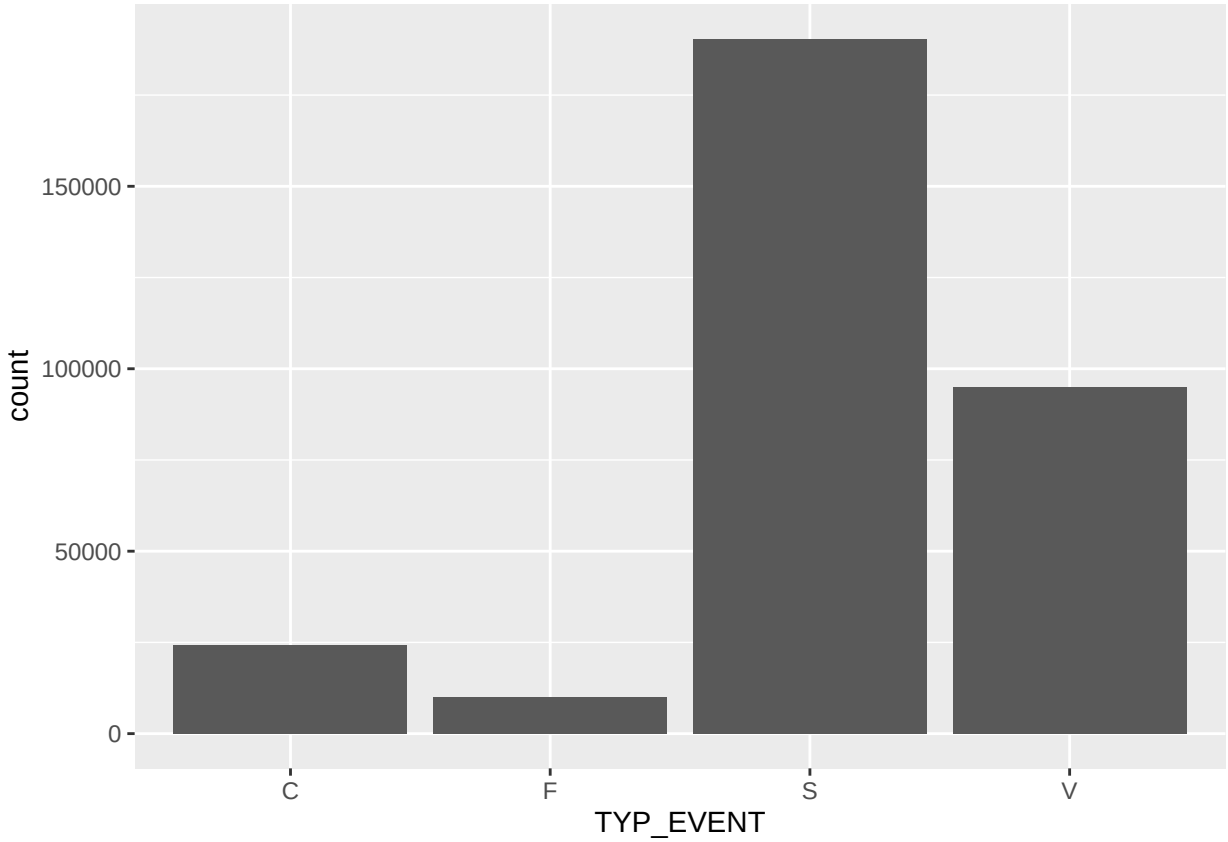
```
## explore distributions ##
# type event
df_6_camp_event_clean %>%
  group_by(TYP_EVENT) %>%
  summarize(TOT_EVENTs = n_distinct(ID_EVENT), TOT_CLIs = n_distinct(ID_CLI), TOT_CAMPs = n_distinct(ID_
  mutate(PERCENT_EVENT = TOT_EVENTs/sum(TOT_EVENTs), PERCENT_CLI = TOT_CLIs/sum(TOT_CLIs), PERCENT_CAMP
  arrange(desc(PERCENT_EVENT), desc(PERCENT_EVENT), desc(PERCENT_CAMP))
```

```
## # A tibble: 4 x 7
##   TYP_EVENT TOT_EVENTs TOT_CLIs TOT_CAMPs PERCENT_EVENT PERCENT_CLI
##   <fct>          <int>    <int>     <int>         <dbl>       <dbl>
## 1 S            1556646   190427       102        0.756       0.596
## 2 V             420239    94832       148        0.204       0.297
## 3 C              51678    24204       123        0.0251      0.0758
## 4 F              31797    10058       103        0.0154      0.0315
## # ... with 1 more variable: PERCENT_CAMP <dbl>
```

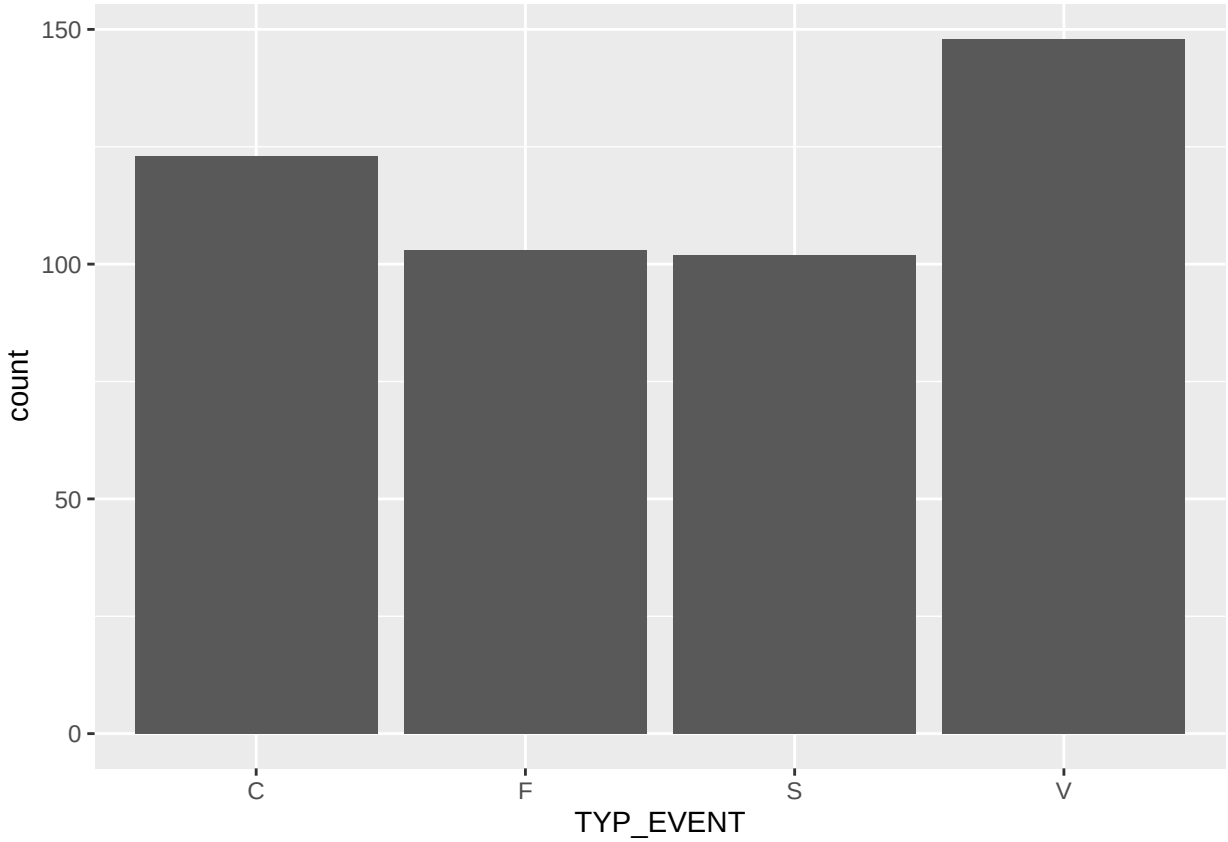
```
ggplot(df_6_camp_event_clean %>% select(TYP_EVENT, ID_EVENT) %>% distinct(), aes(x=TYP_EVENT)) + geom_ba
```




```
ggplot(df_6_camp_event_clean %>% select(TYP_EVENT, ID_CLI) %>% distinct(), aes(x=TYP_EVENT)) + geom_bar
```




```
ggplot(df_6_camp_event_clean %>% select(TYP_EVENT, ID_CAMP) %>% distinct(), aes(x=TYP_EVENT)) + geom_ba
```




```
# min - max dates
df_6_camp_event_clean %>% summarize(MIN_DATE = min(EVENT_DATE), MAX_DATE = max(EVENT_DATE))
```

```
##     MIN_DATE   MAX_DATE
## 1 2019-01-01 2019-04-30
```

```
#### DATA PREPARATION ####
# the aim is to create what we need for our model.

# first we explore the distribution

df_6_camp_event_clean_w_type <- df_6_camp_event_clean %>%
  left_join(df_5_camp_cat_clean
            , by = "ID_CAMP")
```

```
# send
df_sents <- df_6_camp_event_clean_w_type %>%
  filter(TYP_EVENT == "S") %>%
  select(-TYP_EVENT) %>%
  select(ID_EVENT_S = ID_EVENT, ID_CLI, ID_CAMP, TYP_CAMP, ID_DELIVERY, SEND_DATE = EVENT_DATE)
```

```
# open
df_opens <- df_6_camp_event_clean_w_type %>%
  filter(TYP_EVENT == "V") %>%
  select(-TYP_EVENT) %>%
  select(ID_EVENT_O = ID_EVENT, ID_CLI, ID_CAMP, TYP_CAMP, ID_DELIVERY, OPEN_DATE = EVENT_DATE) %>%
  group_by(ID_CLI, ID_CAMP, ID_DELIVERY) %>%
  filter(OPEN_DATE == min(OPEN_DATE)) %>%
```

```
  filter(row_number() == 1) %>%
  ungroup()
```

```
# click
df_clicks <- df_6_camp_event_clean_w_type %>%
  filter(TYP_EVENT == "C") %>%
  select(-TYP_EVENT) %>%
  select(ID_EVENT_C = ID_EVENT, ID_CLI, ID_CAMP, TYP_CAMP, ID_DELIVERY, CLICK_DATE = EVENT_DATE) %>%
  group_by(ID_CLI, ID_CAMP, ID_DELIVERY) %>%
  filter(CLICK_DATE == min(CLICK_DATE)) %>%
  filter(row_number() == 1) %>%
  ungroup()
```

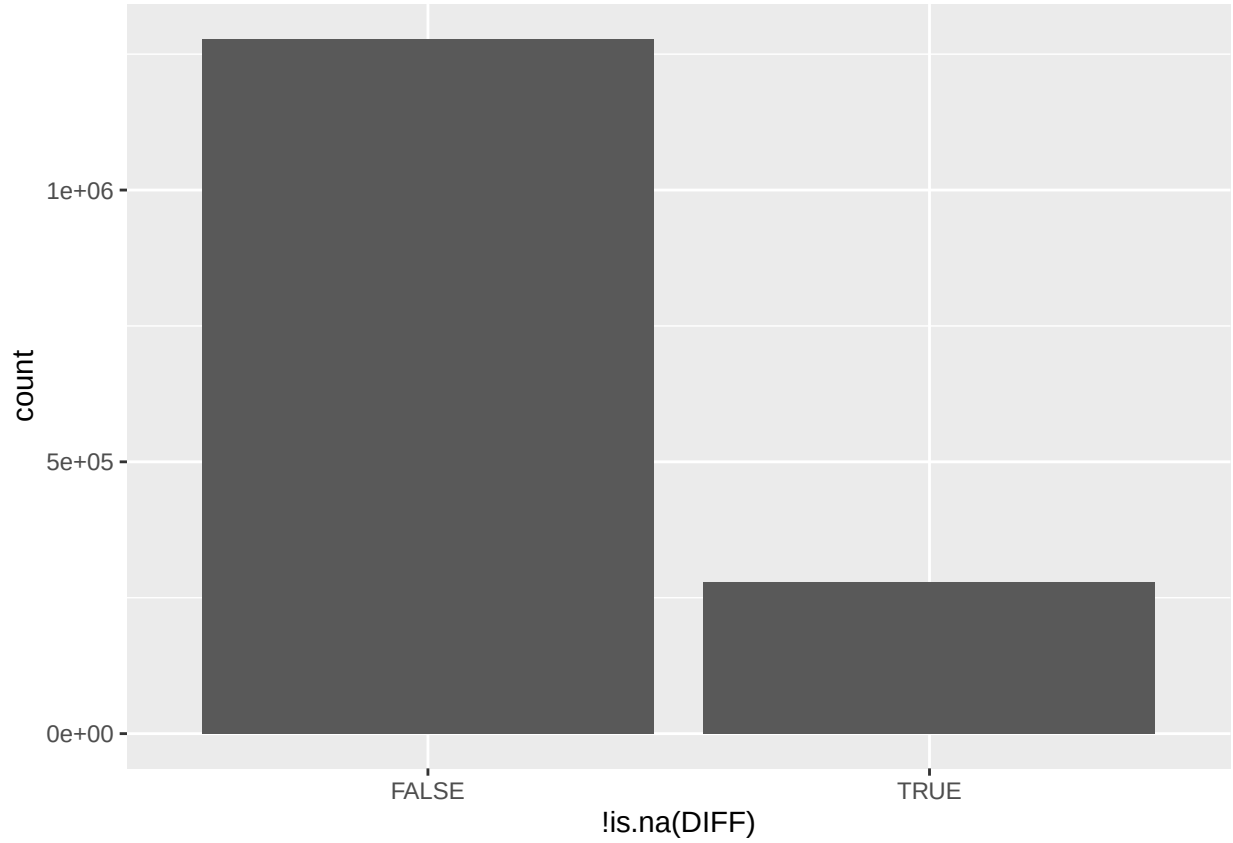
```
# failure
df_fails <- df_6_camp_event_clean_w_type %>%
  filter(TYP_EVENT == "F") %>%
  select(-TYP_EVENT) %>%
  select(ID_EVENT_F = ID_EVENT, ID_CLI, ID_CAMP, TYP_CAMP, ID_DELIVERY, FAIL_DATE = EVENT_DATE) %>%
  group_by(ID_CLI, ID_CAMP, ID_DELIVERY) %>%
  filter(FAIL_DATE == min(FAIL_DATE)) %>%
  filter(row_number() == 1) %>%
  ungroup()
```

```
# attach send to open
df_sents_w_open <- df_sents %>%
  left_join(df_opens
            , by = c("ID_CLI", "ID_CAMP", "ID_DELIVERY", "TYP_CAMP")
            ) %>%
  filter(is.na(OPEN_DATE) | SEND_DATE <= OPEN_DATE) %>%
  mutate(DIFF = as.integer(OPEN_DATE - SEND_DATE))
```

```
# number of sents without opens
df_sents_w_open %>%
  group_by(w_open = !is.na(DIFF)) %>%
  summarize(TOT_SENTs = n_distinct(ID_EVENT_S)) %>%
  mutate(PERCENT = TOT_SENTs/sum(TOT_SENTs)) %>%
  arrange(desc(PERCENT))
```

```
## # A tibble: 2 x 3
##   w_open TOT_SENTs PERCENT
##   <lgl>      <int>   <dbl>
## 1 FALSE    1278264   0.821
## 2 TRUE      278382   0.179
```

```
ggplot(df_sents_w_open, aes(x=!is.na(DIFF))) + geom_bar()
```

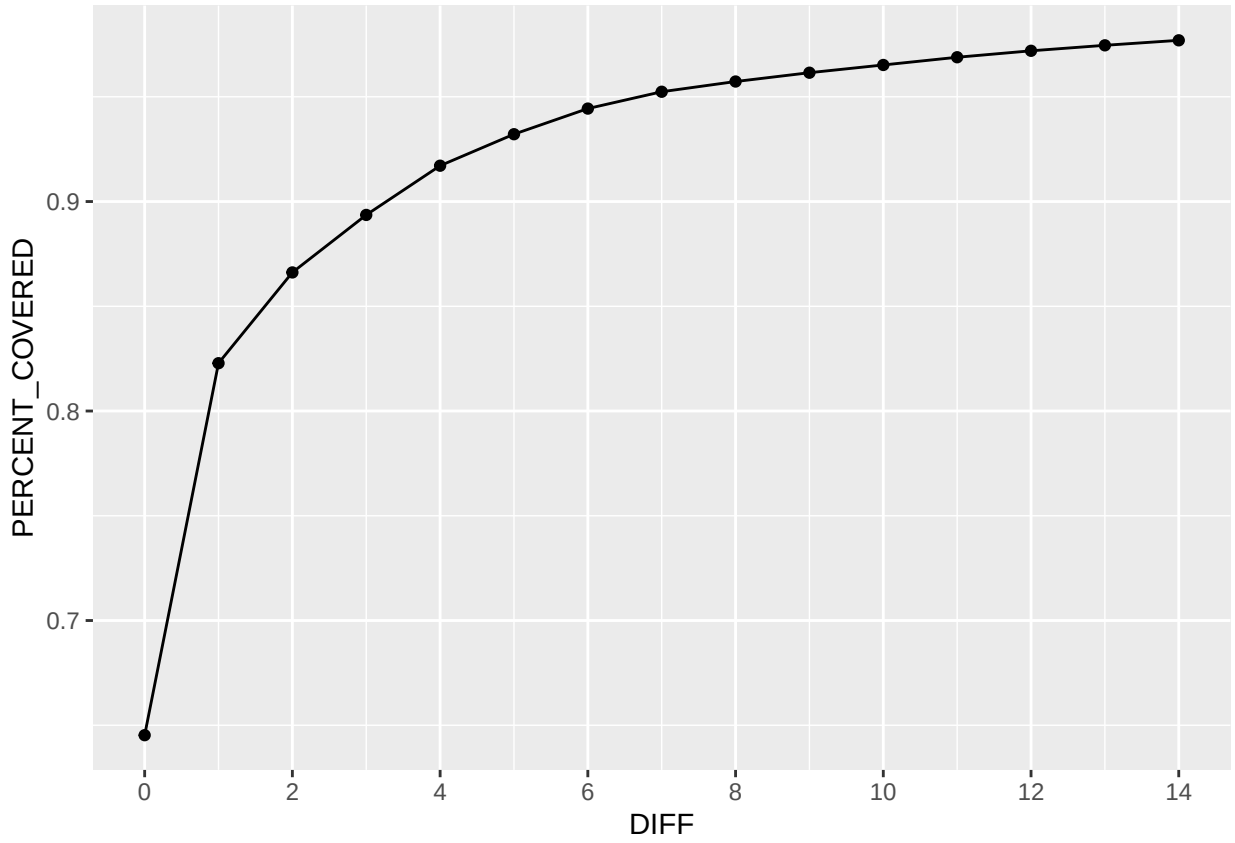



```
# distribution days opens
df_sents_w_open %>% filter(!is.na(DIFF)) %>%
  group_by(DIFF) %>%
  summarize(TOT_EVENTs = n_distinct(ID_EVENT_S)) %>%
  arrange(DIFF) %>%
  mutate(PERCENT_COVERED = cumsum(TOT_EVENTs)/sum(TOT_EVENTs))
```

```
## # A tibble: 60 x 3
##     DIFF TOT_EVENTs PERCENT_COVERED
##    <int>      <int>           <dbl>
##  1     0     179630           0.645
##  2     1      49431           0.823
##  3     2      12062           0.866
##  4     3       7641           0.894
##  5     4       6542           0.917
##  6     5       4185           0.932
##  7     6       3407           0.944
##  8     7       2236           0.952
##  9     8       1359           0.957
## 10     9       1171           0.961
## # ... with 50 more rows
```

```
ggplot(df_sents_w_open %>% filter(!is.na(DIFF)) %>%
         group_by(DIFF) %>%
         summarize(TOT_EVENTs = n_distinct(ID_EVENT_S)) %>%
         arrange(DIFF) %>%
         mutate(PERCENT_COVERED = cumsum(TOT_EVENTs)/sum(TOT_EVENTs)) %>%
```

```
        filter(DIFF <= 14)
      , aes(y=PERCENT_COVERED, x=DIFF)) + geom_line() + geom_point() + scale_x_continuous(breaks=seq(0
```




```
# we can choose as window function 2 day
window_days <- 2

### construction of the datamart ###

# our target variable will be if a send event is open within the timespan of the window days

target_event <- df_sents_w_open %>%
  mutate(TARGET = as.factor(if_else(!is.na(DIFF) & DIFF <= window_days, "1", "0"))) %>%
  select(ID_EVENT_S, ID_CLI, ID_CAMP, ID_DELIVERY, SEND_DATE, TARGET)
```

```
# some relavant variable we want to include are:
# - average open rate (within 14 days) of the communications received by the client in the 30 days befo
# - average click-through (within 14 days) rate of the communications received by the client in the 30

# in order to have comparable situation we are considering:
# - targeted sent made after the 2019-02-01 and window_days before 2019-04-30
# - targeted sent to clients registered by at least 30 days

rate_window <- 14
prev_window <- 30

dt_start <- as.Date("2019-02-01")
```

```
dt_end <- as.Date("2019-04-30") - window_days

relevant_event <- df_sents %>%
  left_join(df_opens
            , by = c("ID_CLI", "ID_CAMP", "ID_DELIVERY", "TYP_CAMP")
  ) %>%
  filter(is.na(OPEN_DATE) | SEND_DATE <= OPEN_DATE) %>%
  left_join(df_clicks
            , by = c("ID_CLI", "ID_CAMP", "ID_DELIVERY", "TYP_CAMP")
  ) %>%
  filter(is.na(CLICK_DATE) | SEND_DATE <= CLICK_DATE) %>%
  left_join(df_fails
            , by = c("ID_CLI", "ID_CAMP", "ID_DELIVERY", "TYP_CAMP")
  ) %>%
  filter(is.na(FAIL_DATE) | SEND_DATE <= FAIL_DATE) %>%
  mutate(DIFF_OPEN = as.integer(OPEN_DATE - SEND_DATE)) %>%
  mutate(DIFF_CLICK = as.integer(CLICK_DATE - SEND_DATE)) %>%
  filter(is.na(DIFF_OPEN) | DIFF_OPEN < rate_window) %>%
  filter(is.na(DIFF_CLICK) | DIFF_CLICK < rate_window)

names(relevant_event) <- sapply(names(relevant_event), paste0, "_PREV")

target_event_w_prev <- target_event %>% filter(SEND_DATE >= dt_start & SEND_DATE <= dt_end) %>%
  left_join(relevant_event
            , by = c("ID_CLI" = "ID_CLI_PREV")
            ) %>%
  filter(is.na(SEND_DATE_PREV) | (SEND_DATE_PREV < SEND_DATE & SEND_DATE <= SEND_DATE_PREV + prev_windo
  mutate(OPENED = if_else(OPEN_DATE_PREV <= SEND_DATE & SEND_DATE <= OPEN_DATE_PREV + prev_window, 1, 0
  mutate(CLICKED = if_else(CLICK_DATE_PREV <= SEND_DATE & SEND_DATE <= CLICK_DATE_PREV + prev_window, 1
  mutate(FAILED = if_else(!is.na(ID_EVENT_F_PREV), 1, 0)) %>%
  group_by(ID_EVENT_S, ID_CLI, ID_CAMP, ID_DELIVERY, SEND_DATE,  TARGET) %>%
  summarize(NUM_SEND_PREV = n_distinct(ID_EVENT_S_PREV, na.rm = T)
            , NUM_OPEN_PREV = sum(OPENED, na.rm = T)
            , NUM_CLICK_PREV = sum(CLICKED, na.rm = T)
            , NUM_FAIL_PREV = sum(FAILED, na.rm = T)
            ) %>%
  ungroup() %>%
  mutate(OPEN_RATE_PREV = NUM_OPEN_PREV/NUM_SEND_PREV) %>%
  mutate(CLICK_RATE_PREV = NUM_CLICK_PREV/NUM_OPEN_PREV) %>%
  mutate(W_SEND_PREV = as.factor(NUM_SEND_PREV > 0)) %>%
  mutate(W_FAIL_PREV = as.factor(NUM_FAIL_PREV > 0)) %>%
  mutate(SEND_WEEKDAY = as.factor(weekdays(SEND_DATE))) %>%
  mutate(OPEN_RATE_PREV = if_else(is.na(OPEN_RATE_PREV), 0, OPEN_RATE_PREV)) %>%
  mutate(CLICK_RATE_PREV = if_else(is.na(CLICK_RATE_PREV), 0, CLICK_RATE_PREV))
```

```
# add client data

df_master <- target_event_w_prev %>%
  left_join(df_1_cli_fid_clean %>%
              select(ID_CLI, ID_NEG, TYP_CLI_FID, COD_FID, STATUS_FID, FIRST_DT_ACTIVE, NUM_FIDs)
            , by = "ID_CLI") %>%
  filter(FIRST_DT_ACTIVE <= SEND_DATE) %>%
  # filter(FIRST_DT_ACTIVE <= SEND_DATE - 30) %>%
  mutate(AGE_FID = as.integer(SEND_DATE - FIRST_DT_ACTIVE)) %>%
```

```
  left_join(df_2_cli_account_clean
            , by = "ID_CLI") %>%
  left_join(df_3_cli_address_clean %>%
              select(ID_ADDRESS, PRV, REGION)
            , by = "ID_ADDRESS") %>%
  left_join(df_4_cli_privacy_clean
            , by = "ID_CLI") %>%
  mutate(PRV = fct_explicit_na(PRV)) %>%
  mutate(REGION = fct_explicit_na(REGION)) %>%
  select(-ID_ADDRESS, -ID_CLI, -ID_CAMP, -ID_DELIVERY, -SEND_DATE, -FIRST_DT_ACTIVE)
```

```
# check there are not duplicates
df_master %>%
  group_by(ID_EVENT_S) %>%
  summarize(num = n()) %>%
  group_by(num) %>%
  count()
```

```
## # A tibble: 1 x 2
## # Groups:   num [1]
##     num       n
##   <int>   <int>
## 1     1 1111503
```

```
df_master <- df_master %>%
  mutate(NUM_FIDs=as.integer(NUM_FIDs))
```

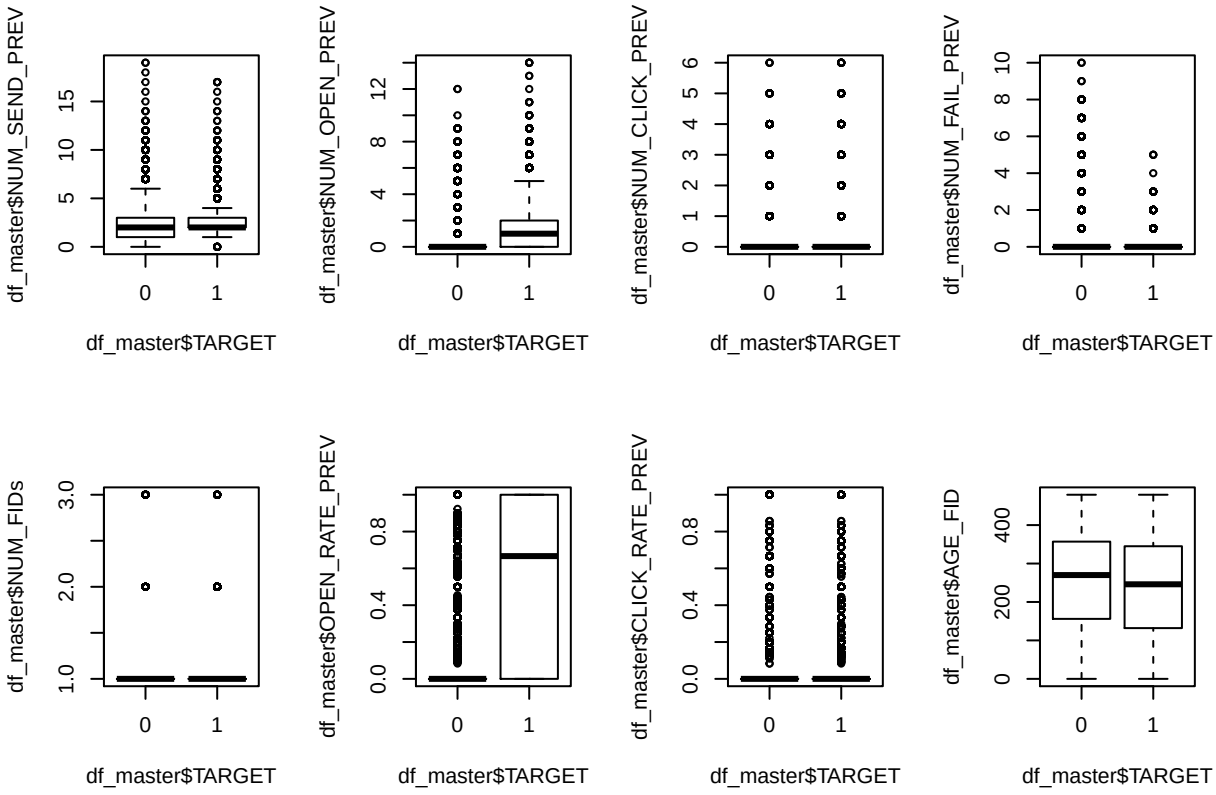
DATA EXPLORATION AND PREPROCESSING

Il processo di preparazione del dato ha permesso di creare il dataset su cui basare l'analisi, dato da 1111503 osservazioni e 26 variabiI.

A questo punto Ã Â¨ stata effettuata una fase esplorativa per comprendere a pieno il loro significato in relazione alla variabile target e avere una prima idea su quali sono gli attributi piÃ Â¹ utili a discriminare tra le due classi di clienti.

Si effettuano dei boxplot per vedere la diversa distribuzione delle variabili numeriche rispetto alle due classi della variabile target.

```
par(mfrow=c(2,4))
plot(df_master$NUM_SEND_PREV ~df_master$TARGET)
plot(df_master$NUM_OPEN_PREV ~df_master$TARGET)
plot(df_master$NUM_CLICK_PREV ~df_master$TARGET)
plot(df_master$NUM_FAIL_PREV ~df_master$TARGET)
plot(df_master$NUM_FIDs ~df_master$TARGET)
plot(df_master$OPEN_RATE_PREV ~df_master$TARGET)
plot(df_master$CLICK_RATE_PREV ~df_master$TARGET)
plot(df_master$AGE_FID ~df_master$TARGET)
```




```
par(mfrow=c(1,1))
```

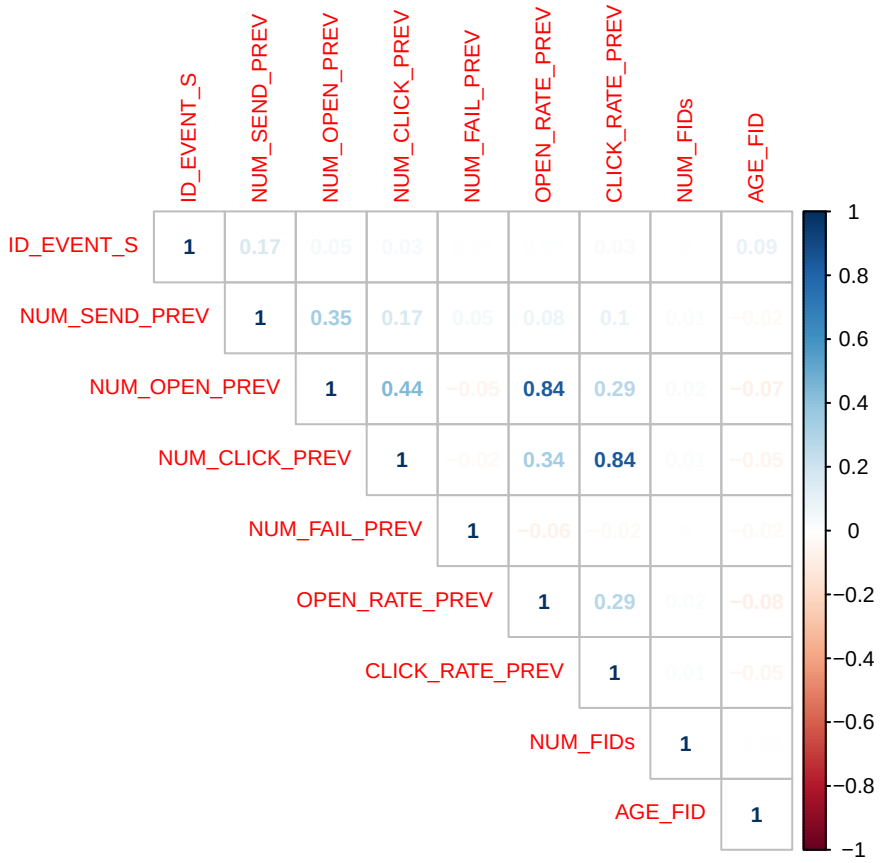
Da questi boxplot non si evincono particolari differenze tra le distribuzioni delle esplicative numeriche differenziate in base alla classe di appartenenza. L'unica eccezione Ã Â¨ data dall'esplicativa "OPEN_RATE_PREV". E' evidente, infatti, che i clienti che aprono la mail in questione hanno una percentuale di mail precedentemente aperte molto superiore a quella dell'altra classe di consumatori.

Si procede con l'analisi della correlazione tra le variabili numeriche per evitare la presenza di multicollinearitÃ Â all'interno dei modelli.

```
var_df_master_num <- which(sapply(df_master ,is.numeric))

library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corrplot(cor(df_master[var_df_master_num]),type = "upper",method='number',tl.cex = .7,cl.cex = .7,numbe
```



Dal corrplot si evince che le due coppie di variabili numeriche piÃ Â¹ correlate tra di loro sono rispettivamente NUM_OPEN_PREV-OPEN_RATE_PREV e NUM_CLICK_PREV-CLICK_RATE_PREV.

Dato il significato delle variabili, quanto emerso risulta sensato, infatti l'una Ã Â¨ "derivata" dall'altra.

Per evitare multicollinearitÃ Â nei modelli, quindi, si procede con l'eliminazione di due di queste quattro variabili, "NUM_OPEN_PREV" e "NUM_CLICK_PREV".

Inoltre, vengono eliminate anche W_SEND_PREV e W_FAIL_PREV perchÃ Â¨ rispettivamente "incluse" nelle variabili "NUM_SEND_PREV" e "NUM_FAIL_PREV" (ad esempio, tutte le volte in cui W_SEND_PREV Ã Â¨ uguale a FALSE, la variabile NUM_SAND_PREV Ã Â¨ pari a 0).

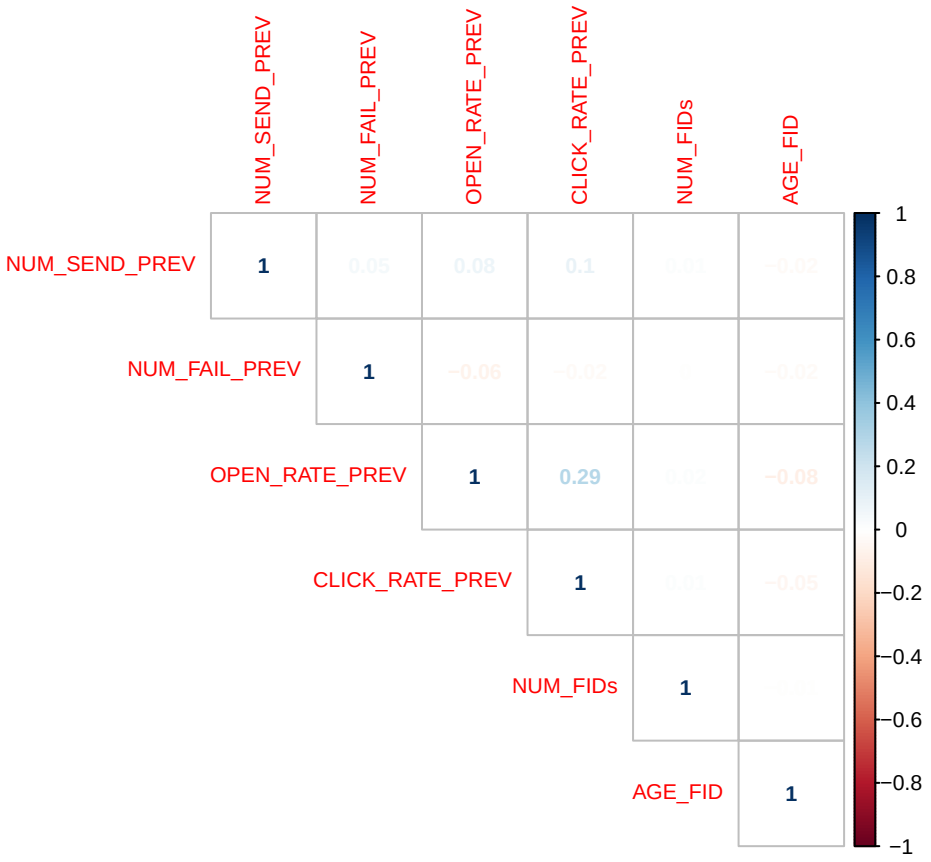
Infine, viene eliminata anche TYP_CLI_ACCOUNT perchÃ Â¨ di dubbio significato e si prenderÃ Â in considerazione soltanto la Regione, dato che la variabile PRV presenta diversi livelli e quindi un peso computazionale piuttosto elevato.

Si procede con l'eliminazione dal dataset delle variabili elencate.

```
df_sub <- df_master %>%
  select(-ID_EVENT_S,-NUM_OPEN_PREV,-NUM_CLICK_PREV,-W_SEND_PREV,-W_FAIL_PREV,-TYP_CLI_ACCOUNT,-PRV)

#assegnazione dei livelli delle variabili booleane poichÃ Â¨ le categorie 0,1 devono essere attribuite s

levels(df_sub$TARGET) <- c("not_opened", "opened")
levels(df_sub$TYP_CLI_FID) <- c("not_main", "main_account")
levels(df_sub$STATUS_FID) <- c("not_active", "active_account")
levels(df_sub$W_PHONE) <- c("with_phone", "Without_phone")
levels(df_sub$FLAG_PRIVACY_1) <- c("not_flag1", "flag1")
levels(df_sub$FLAG_PRIVACY_2) <- c("not_flag2", "flag2")
```

```
levels(df_sub$FLAG_DIRECT_MKT) <- c("not_flagMKT", "flagMKT")
```

```
var_num <- which(sapply(df_sub,is.numeric)) #subset variabili numeriche

library(corrplot)

corrplot(cor(df_sub[var_num]),type = "upper",method='number',tl.cex = .7,cl.cex = .7,number.cex = 0.7)
```



Dal corrplot si evince che, a seguito dell'eliminazione delle variabili suddette, sono state eliminate correlazione statisticamente significative.

Per quanto riguarda le variabili categoriche, si procede con una feature selection effettuata tramite il test *Chi-Quadro*, utile a verificare l'ipotesi di indipendenza degli attributi con la variabile target.

```
#subset variabili categoriche
data_factor=as.data.frame(df_sub[,which(sapply(df_sub,is.factor))])

library(knitr)
chi_test=function(dataset) {
  matrice=matrix(NA, ncol=3,nrow=dim(dataset)[2]-1)
for (i in 1:dim(dataset)[2]-1) {
  matrice[i,1]=colnames(dataset)[i+1]
  matrice[i,2]=chisq.test(table(dataset[,1],dataset[,i+1]))$p.value
  matrice[i,3]=chisq.test(table(dataset[,1],dataset[,i+1]))$statistic
}
  colnames(matrice)=c("variabile categorica","p-value","statistica test")
  return(kable(matrice))
```

```
}

chi_test(data_factor)
```

variabile categorica	p-value	statistica test
SEND_WEEKDAY	1.99670607018055e-198	933.646777286993
ID_NEG	0	7794.67162780957
TYP_CLI_FID	0.000149045121077235	14.3842324472113
COD_FID	0	1649.65050926219
STATUS_FID	0.00494835390162324	7.89821742059218
W_PHONE	4.46649301387633e-20	84.202306371497
TYP_JOB	0	12312.3319400686
EMAIL_PROVIDER_CLEAN	0	3250.43369040928
REGION	0	5391.6813500473
FLAG_PRIVACY_1	7.64401071006121e-98	440.694386202241
FLAG_PRIVACY_2	0.00925218640415786	6.77348749976283
FLAG_DIRECT_MKT	0	2140.61683291339

Dai risultati del chi-quadro test, si evince che tutte le variabili categoriche sono dipendenti dalla variabile target poichÃ Â¨ tutti i p-value prossimi allo zero permettono di rifiutare l'ipotesi nulla di indipendenza.

Terminata la fase di feature selection, si analizza la distribuzione delle osservazioni all'interno della variabile target.

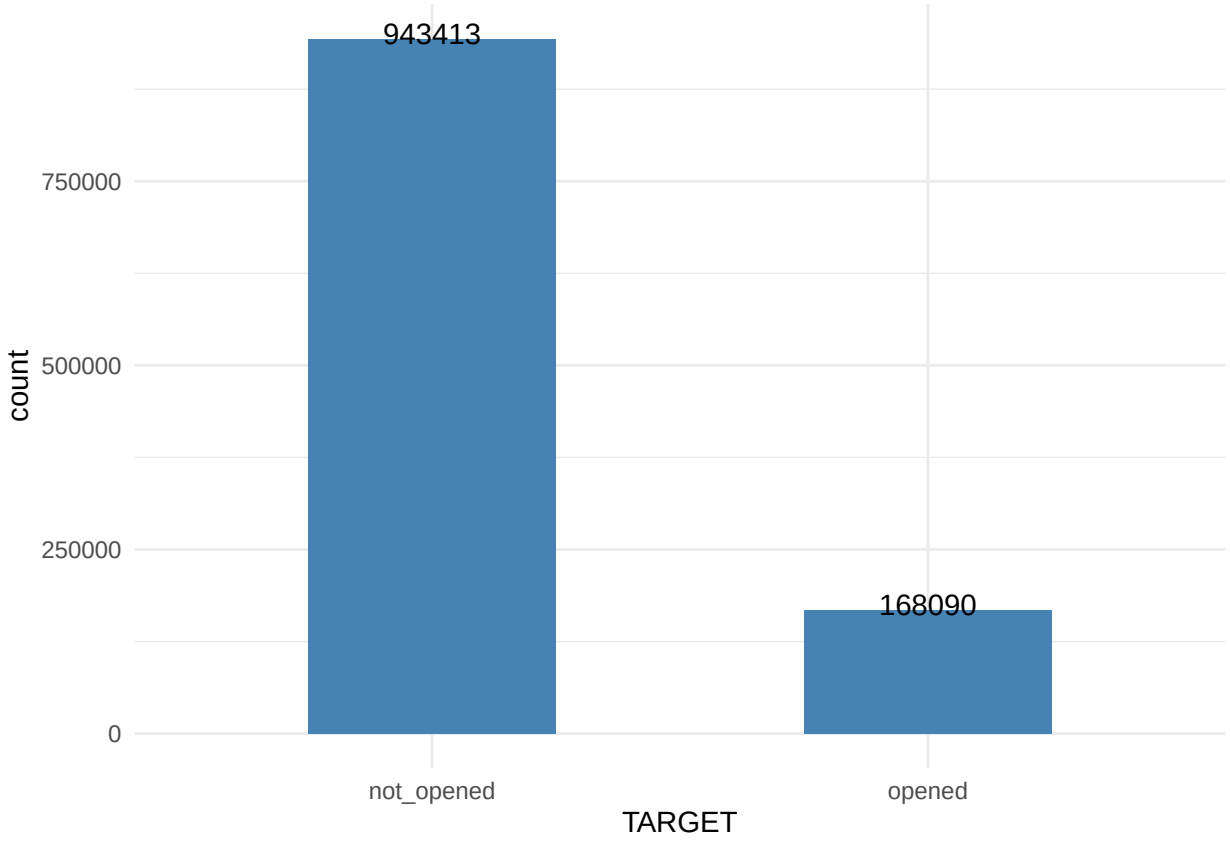
```
table(df_sub$TARGET)
```

```
##
## not_opened     opened
##     943413     168090
```

```
unbalanced_class=ggplot(df_sub, aes(x=TARGET))+
geom_bar(width=0.5,fill="steelblue")+
stat_count(binwidth=1, geom="text", aes(label=..count..), vjust=0.25) +
theme_minimal()
```

```
## Warning: Ignoring unknown parameters: binwidth
```

```
options(repr.plot.width=8,repr.plot.height=3)
unbalanced_class
```



E' evidente che si Ã Â¨ in presenza di un dataset caratterizzato da classi sbilanciate. Infatti la classe positiva, ovvero quella di interesse, Ã Â¨ pari solamente a circa 1/6 delle osservazioni totali.

Portare avanti l'analisi con un dataset dalle classi sbilanciate, i modelli potrebbero risultare distorti e aventi strutturali problemi nella stima dei coefficienti. In questi casi, il problema puÃ Â² essere risolto facendo ricorso a tecniche di bilanciamento delle due classi (oversample, undersample, etc…).

Dal momento che il numero di osservazioni appare molto elevato, effettuare un oversample richiederebbe uno sforzo computazionale troppo elevato per stimare i modelli di Machine Learning. Si Ã Â¨ deciso, a tal proposito, di effettuare un *undersample*, tecnica grazie alla quale si adotta una eliminazione randomica di alcune osservazioni appartenenti alla classe negativa in modo tale che si possano formare due classi bilanciate.

```
library(ROSE)
```

```
## Loaded ROSE 0.0-3
```

```
df_sub_under<- ovun.sample(TARGET ~ ., data = df_sub, method = "under", N=336180)$data
table(df_sub_under$TARGET)
```

```
##
## not_opened     opened
##     168090     168090
```

Come si evince dalla tabella delle frequenze della variabile target, le classi risultano bilanciate, si puÃ Â² procedere quindi con la fase di data modelling.

Data Modelling

Terminata il pre-processing, si passa alla fase di Data Modeling.

Si Ã Â¨ scelto di sviluppare i seguenti modelli: . Decision Tree; . Neural Network (NNET); . Logistic regression.

Per ottenere una validazione dei classificatori utilizzati Ã Â¨ stato utilizzato un approccio basato sulla cross validation, il cosiddetto K-Folds Cross Validation. Questa tecnica statistica suddivide il dataset in k partizioni di eguale numerositÃ Â e assicura che tutti i record vengano utilizzati almeno una volta sia nel training set che nel test set. Il numero di folds utilizzato Ã Â¨ pari a k=3.

I parametri caratteristici di ogni modello, non sono stati scelti a priori ma viene azionato un sistema di tuning al fine di identificare il parametro migliore, che permette cioÃ Â¨ di stimare il modello migliore.

Infine, sia per effettuare il tuning che per confrontare i modelli, Ã Â¨ stata utilizzata la metrica denominata ROC.

L'apprendimento (learning) dei modelli avviene sul train set, pari al 67% del dataset iniziale, su cui Ã Â¨ stato effettuato l'undersampling per risolvere il problema delle classi sbilanciate.

```
data=df_sub_under

#train e test set (1/3,2/3)
smp_size <- floor(0.67 * nrow(data))

set.seed(12345)
train_ind <- sample(seq_len(nrow(data)), size = smp_size)

train <- data[train_ind, ]
test <- data[-train_ind, ]
```

Decision Tree

Il decision tree fa parte di modelli cosiddetti "euristici"" che non sempre forniscono risultati ottimali ma sono in grado di ottenere approssimazioni ragionevoli senza richiedere sforzi computazionali eccessivi o ipotesi restrittive sui dati di partenza. Permette, inoltre, di calcolare l'importanza delle variabili rispetto alla risposta e, per questo motivo, il decision tree viene anche utilizzato come strumento di "feature selection" per selezionare un numero inferiore di variabili con cui stimare la regressione logistica e la neural network in quanto entrambi i modelli vengono penalizzatati eccessivamente dall'inserimento di troppe variabili.

```
set.seed(12345)
library(caret)
```

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

```
metric <- "ROC"
Ctrl <- trainControl(method = "cv" , number=3, classProbs = TRUE,
summaryFunction = twoClassSummary)
rpartTune <- train(TARGET ~ ., data = train, method = "rpart",tuneLength = 5, trControl = Ctrl, metric=m
rpartTune
```

```
## CART
##
## 225240 samples
##     18 predictor
##      2 classes: 'not_opened', 'opened'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 150160, 150160, 150160
## Resampling results across tuning parameters:
##
```

```
##    cp            ROC        Sens       Spec
##    0.0001687899  0.7982986  0.8427589  0.7223938
##    0.0001821154  0.7982984  0.8429455  0.7221897
##    0.0002665103  0.7983122  0.8407068  0.7242931
##    0.0004886022  0.7926360  0.8387701  0.7258285
##    0.5632162465  0.6867462  0.5637848  0.8097077
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0002665103.
```

Il complexity parameter del decision tree Ã Â¨ stato tunato e quello ottimo risulta pari a 0,00009178251.

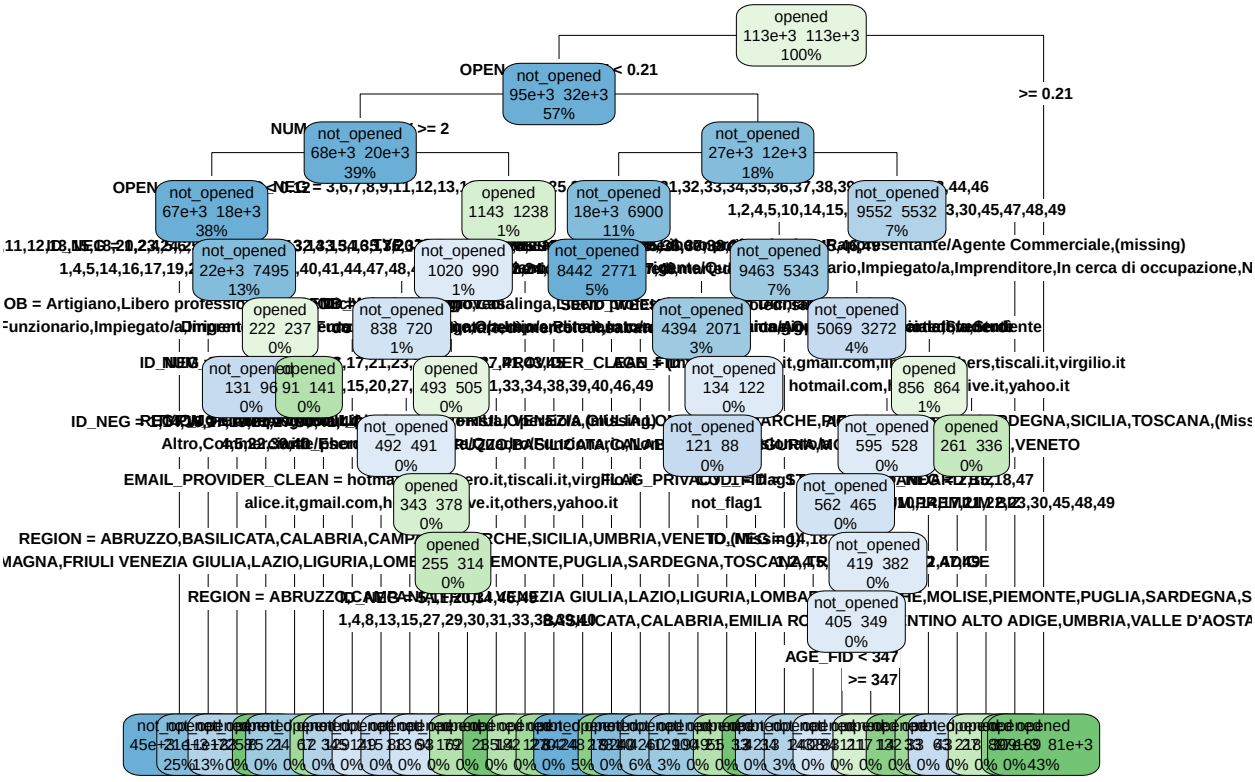
```
# performance evaluation
library(pander)
pander(getTrainPerf(rpartTune))
```

TrainROC	TrainSens	TrainSpec	method
0.7983	0.8407	0.7243	rpart

A questo punto si ristima il decision tree impostando il valore ottimale del CP.

```
set.seed(12345)
Ctrl_save <- trainControl(method = "cv" , number=3, summaryFunction = twoClassSummary,
classProbs = TRUE, savePredictions = TRUE)

rpartTuneMy <- train(TARGET ~ ., data = train, method = "rpart",
tuneGrid=data.frame(cp=9.178251e-05),
trControl = Ctrl_save, metric=metric)
```

```
set.seed(12345)
library(rpart)
library(rpart.plot)
mytree <- rpart(TARGET ~ ., data = train, method = "class", cp = 9.178251e-05)
rpart.plot(mytree, type = 4, extra = 101, cex = 0.5)
```



Si analizza l'importanza delle variabili inserite nel modello:

```
Vimportance <- varImp(rpartTuneMy)
print(Vimportance)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 117)
##
##                                 Overall
## OPEN_RATE_PREV                100.00000
## CLICK_RATE_PREV                13.74039
## TYP_JOB(missing)                4.26258
## NUM_FAIL_PREV                   2.87440
## AGE_FID                         2.06380
## NUM_SEND_PREV                   1.21045
## FLAG_PRIVACY_1flag1             0.83137
## SEND_WEEKDAYmercoledÃ¬          0.43466
## FLAG_DIRECT_MKTflagMKT          0.29921
## SEND_WEEKDAYgiovedÃ¬            0.26894
## TYP_JOBImpiegato/a              0.18111
## EMAIL_PROVIDER_CLEANhotmail.it  0.15163
## TYP_JOBLibero professionista    0.08466
## EMAIL_PROVIDER_CLEANlibero.it   0.06282
## W_PHONEWithout_phone            0.05767
## COD_FIDSTANDARD                 0.05340
## ID_NEG7                         0.05216
```

```
## ID_NEG2                          0.04789
## SEND_WEEKDAYmartedÃ¬              0.03326
## ID_NEG3                          0.02650
```

A seguito della stima del decision tree, si stila un ranking delle variabili in base alla loro importanza rispetto alla variabile target. Per i modelli successivi, Ã Â¨ stato deciso di utilizzare solamente le 10 variabili con maggior importanza.

Di seguito, quindi, prima di procedere con la stima degli altri algoritmi, si effettua un subset del dataset iniziale con le variabili scelte.

```
d_sub_net <- train %>%
  select(TARGET,OPEN_RATE_PREV,CLICK_RATE_PREV,TYP_JOB,NUM_SEND_PREV,AGE_FID,EMAIL_PROVIDER_CLEAN, SEND_
```

Neural Network

E' stata sviluppata una rete neurale, modello che permette di approssimare qualsiasi funzione tramite l'interconnessione di neuroni artificiali, creando un percettrone. Lo svantaggio della rete neurale Ã Â¨ la difficile interpretazione dei risultati finali.

```
tunegrid <- expand.grid(size=c(1:3), decay = c(0.0002, 0.0003, 0.00001, 0.0001))
nnetFit_defgridDR1 <- train(d_sub_net[-1], d_sub_net$TARGET,
                            method = "nnet",
                            preProcess = 'pca',
                            #Nel tunare la NNET viene effettuato il pre-processing delle variabili tram
                            metric=metric,
                            trControl=Ctrl, tuneGrid=tunegrid,
                            trace = FALSE,
                            maxit = 10)
pander(getTrainPerf(nnetFit_defgridDR1))
```

TrainROC	TrainSens	TrainSpec	method
0.8208	0.874	0.6779	nnet

```
pander(nnetFit_defgridDR1$bestTune) #best parameter
```

	size	decay
9	3	1e-05

```
tunegrid <- expand.grid(size=c(1:3), decay = c(0.0002, 0.0003, 0.00001, 0.0001))
nnetFit_defgridDR2 <- train(d_sub_net[-1], d_sub_net$TARGET,
                            method = "nnet",
                            preProcess = c('center', "scale"),
                            #Nel tunare la NNET viene effettuata la standardizzazione delle variabili
                            metric=metric,
                            trControl=Ctrl, tuneGrid=tunegrid,
                            trace = FALSE,
                            maxit = 10)
pander(getTrainPerf(nnetFit_defgridDR2))
```

TrainROC	TrainSens	TrainSpec	method
0.819	0.8556	0.7057	nnet

```
pander(nnetFit_defgridDR2$bestTune) #best parameter
```

	size	decay
9	3	1e-05

```
tunegrid <- expand.grid(size=c(1:3), decay = c(0.0002, 0.0003, 0.00001, 0.0001))
nnetFit_defgridDR3 <- train(d_sub_net[-1], d_sub_net$TARGET,
                            method = "nnet",
                            preProcess = c('range'),
                            #Nel tunare la NNET viene effettuata la normalizzazione delle variabili
                            metric=metric,
                            trControl=Ctrl, tuneGrid=tunegrid,
                            trace = FALSE,
                            maxit = 10)
pander(getTrainPerf(nnetFit_defgridDR3))
```

TrainROC	TrainSens	TrainSpec	method
0.8182	0.8523	0.7089	nnet

```
pander(nnetFit_defgridDR3$bestTune) #best parameter
```

	size	decay
12	3	3e-04

Si utilizzano quindi i parametri e il metodo di preprocess migliori emersi in precedenza per la rete neurale

```
set.seed(12345)
tunegrid <- expand.grid(size=2, decay =0.0003)
nnetFit_finale <- train(d_sub_net[-1], d_sub_net$TARGET,
method = "nnet",
preProcess = 'pca',
metric=metric,
trControl=Ctrl_save, tuneGrid=tunegrid,
trace = FALSE,
maxit = 10)
```

Logistic Regression

La regressione logistica, al contrario, fa parte dei modelli di regressione, modelli facilmente comprensibili in quanto Ã Â¨ possibile misurare quantitativamente l'effetto delle diverse variabili sulla risposta a partire dai coefficienti assegnati dal modello.

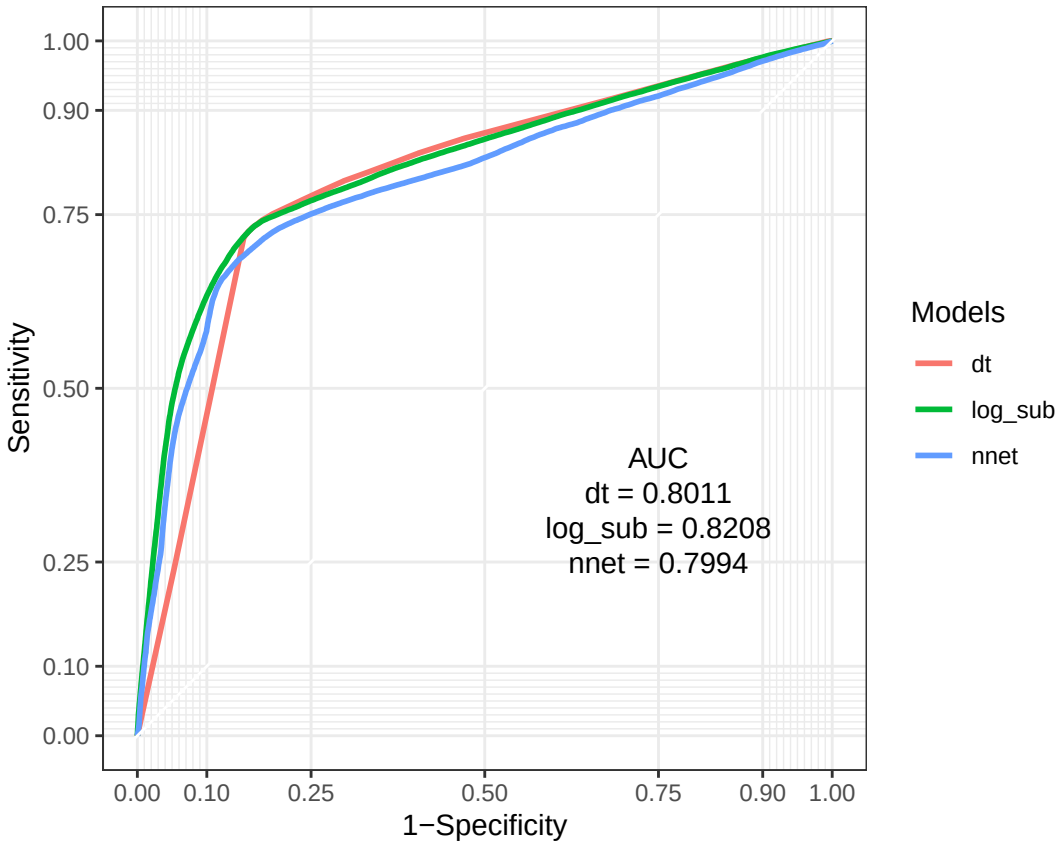
```
set.seed(12345)
logistic_sub <- train(TARGET~., data=d_sub_net, trControl=Ctrl_save, metric=metric,
method="glm",family=binomial())
pander(getTrainPerf(logistic_sub))
```

TrainROC	TrainSens	TrainSpec	method
0.8209	0.864	0.6963	glm

MODELS EVALUATION

Per confrontare i tre algoritmi sviluppati Ã Â¨ stata utilizzata la ROC Curve e l'AUC, ovvero l'area sottostante la curva. L'obiettivo Ã Â¨ quello di scegliere il modello migliore, con un AUC piÃ Â¹ alto per utilizzarlo nella fase di data driven action.

```
roc_values <- cbind(as.data.frame(logistic_sub$pred$obs), as.data.frame(logistic_sub$pred$opened))
nnet <- as.data.frame(nnetFit_finale$pred$opened)
dt <- as.data.frame(rpartTuneMy$pred$opened)

roc_values <- cbind(roc_values, nnet, dt)
names(roc_values) <- c("obs","log_sub","nnet", "dt")
library(plotROC)
longtest <- melt_roc(roc_values, "obs", c("log_sub","nnet", "dt"))
longtest$D <- ifelse(longtest$D=="opened",1,0)
names(longtest)[3] <- "Models"

g <- ggplot(longtest, aes(m=M, d=D, color=Models)) +
geom_roc(n.cuts=0) +
coord_equal() +
style_roc(xlab="1-Specificity", ylab="Sensitivity")

g + annotate("text", x=0.75, y=0.4, label="AUC") +
annotate("text", x=0.75, y=0.35, label=paste("dt =", round(calc_auc(g)$AUC[1], 4))) +
annotate("text", x=0.75, y=0.30, label=paste("log_sub =", round(calc_auc(g)$AUC[2], 4))) +
annotate("text", x=0.75, y=0.25, label=paste("nnet =", round(calc_auc(g)$AUC[3], 4)))
```



```
test_model<- function(model, y, len = NULL, search = "grid") {
  test_pred <- predict(model,test)
  prec <- precision(data = test_pred, reference = test$TARGET)
  Fmeas <- F_meas(data = test_pred, reference = test$TARGET)
  rec <- recall(data = test_pred, reference = test$TARGET)

  return (c(prec, rec, Fmeas))
}
```

```
performance_value= as.data.frame(test_model(rpartTuneMy))
performance_value$log_sub= test_model(logistic_sub)
performance_value$nnet= test_model(nnetFit_finale)
colnames(performance_value)[1]= 'dt'
rownames(performance_value)= c('Precision','Recall','F1-measure')
kable(performance_value)
```

	dt	log_sub	nnet
Precision	0.7556984	0.7416970	0.7260530
Recall	0.8383402	0.8619336	0.8820150
F1-measure	0.7948770	0.7973078	0.7964708

Dalla rappresentazione della ROC curve e dal calcolo dell'AUC, si evince che il modello migliore Ã Â¨ rappresentato dalla regressione logistica, avente un'AUC pari a 0.8213.

Essendo in presenza di un dataset avente classi sbilanciate, l'accuracy, ovvero la capacitÃ Â di effettuare previsioni corrette su nuovi records, non Ã Â¨ significativa per misurare la "bontÃ Â " del modello poichÃ Â¨, in questo caso, si basa sulla cosiddetta ZeroR rule, secondo il quale il classificatore considera solamente i records del dataset appartenenti alla classe piÃ Â¹ frequente. Le misure piÃ Â¹ adatte per analizzare la capacitÃ Â predittiva del modello, quindi, risultano essere la precision e la recall. In particolare, diventa opportuno considerare le due metriche simultaneamente utilizzando la F1 Measure che, tramite la media armonica, riassume precision e recall.

Come si evince dalla suddetta tabella, la regressione logistica rimane il modello migliore anche dal punto di vista della metrica F1-Measure.

Si utilizza, quindi, la regressione logistica per effettuare un'azione di data driven, con l'obiettivo di verificare l'efficacia del modello stimato e analizzare qualche esempio pratico.

DATA DRIVEN ACTION

Per concludere, si cerca una risposta alla domanda di business iniziale effettuando dei test con un individuo esempio.

L'obiettivo di questi test Ã Â¨ mostrare concretamente e confermare l'importanza delle variabili cosÃ Â¬ come Ã Â¨ emersa grazie al modello logistico, in modo tale da capire in che modo migliorare la campagna di marketing effetuata tramite mail per far si che le email vengano aperte entro due giorni da piÃ Â¹ clienti possibili.

Di seguito Ã Â¨ riportato un esempio in cui possibile osservare come, al cambiare del valore di una sola delle esplicative risultate piÃ Â¹ importanti, cambia la classe di appartenenza dell'individuo in questione.

```
summary(logistic_sub)
```

```
##
## Call:
## NULL
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0314  -0.7639   0.1842   0.6890   1.8542
##
## Coefficients:
##                                          Estimate Std. Error z value
## (Intercept)                              5.907e-01  2.034e-01   2.905
## OPEN_RATE_PREV                           3.575e+00  1.645e-02 217.333
## CLICK_RATE_PREV                          2.108e-01  3.072e-02   6.861
## TYP_JOBArtigiano                        -1.829e-02  2.697e-01  -0.068
## TYP_JOBCasalinga                         3.651e-01  2.163e-01   1.688
## `TYP_JOBCommerciante/Esercente`          5.714e-02  2.973e-01   0.192
## `TYP_JOBDirigente/Quadro/Funzionario`    5.320e-01  1.710e-01   3.111
## `TYP_JOBImpiegato/a`                     3.150e-01  1.213e-01   2.597
## TYP_JOBImprenditore                      5.626e-01  2.479e-01   2.269
## `TYP_JOBIn cerca di occupazione`         1.727e-01  2.927e-01   0.590
## `TYP_JOBLibero professionista`          -2.674e-01  1.149e-01  -2.327
## `TYP_JOBNon Dichiara`                    3.961e-01  2.077e-01   1.907
## `TYP_JOBOperaio/a`                       2.808e-01  1.490e-01   1.884
## `TYP_JOBPensionato/a`                    4.537e-01  1.546e-01   2.935
## `TYP_JOBRappresentante/Agente Commerciale` 4.056e-01  6.008e-01   0.675
## TYP_JOBStudente                          3.513e-01  3.750e-01   0.937
## `TYP_JOB(missing)`                      -4.933e-01  1.059e-01  -4.657
## NUM_SEND_PREV                            4.342e-03  4.141e-03   1.048
## AGE_FID                                 -3.736e-04  4.322e-05  -8.645
## EMAIL_PROVIDER_CLEANalice.it             3.853e-01  1.650e-01   2.334
## EMAIL_PROVIDER_CLEANgmail.com            3.604e-01  1.637e-01   2.202
## EMAIL_PROVIDER_CLEANhotmail.com          4.457e-01  1.663e-01   2.680
## EMAIL_PROVIDER_CLEANhotmail.it           4.695e-01  1.645e-01   2.854
## EMAIL_PROVIDER_CLEANlibero.it            2.124e-01  1.640e-01   1.295
## EMAIL_PROVIDER_CLEANlive.it              3.411e-01  1.671e-01   2.041
## EMAIL_PROVIDER_CLEANothers               3.567e-01  1.641e-01   2.173
## EMAIL_PROVIDER_CLEANtiscali.it           3.057e-01  1.668e-01   1.833
## EMAIL_PROVIDER_CLEANvirgilio.it          3.302e-01  1.667e-01   1.980
## EMAIL_PROVIDER_CLEANyahoo.it             4.894e-01  1.651e-01   2.964
## SEND_WEEKDAYgiovedÃ¬                     -2.631e-01  2.122e-02 -12.398
## SEND_WEEKDAYlunedÃ¬                       3.214e-02  4.842e-02   0.664
## SEND_WEEKDAYmartedÃ¬                      7.718e-03  2.552e-02   0.302
## SEND_WEEKDAYmercoledÃ¬                   -2.499e-01  2.425e-02 -10.303
## SEND_WEEKDAYsabato                      -2.082e-01  4.999e-02  -4.166
## SEND_WEEKDAYvenerdÃ¬                     -1.994e-01  3.577e-02  -5.576
## FLAG_DIRECT_MKTflagMKT                  -9.714e-01  7.435e-02 -13.065
## FLAG_PRIVACY_1flag1                     -2.857e-01  2.198e-02 -12.998
##                                         Pr(>|z|)
## (Intercept)                              0.00367 **
## OPEN_RATE_PREV                           < 2e-16 ***
## CLICK_RATE_PREV                         6.83e-12 ***
## TYP_JOBArtigiano                         0.94592
## TYP_JOBCasalinga                         0.09141 .
## `TYP_JOBCommerciante/Esercente`          0.84761
## `TYP_JOBDirigente/Quadro/Funzionario`    0.00187 **
## `TYP_JOBImpiegato/a`                     0.00939 **
## TYP_JOBImprenditore                      0.02326 *
```

```
## `TYP_JOBIn cerca di occupazione`            0.55520
## `TYP_JOBLibero professionista`              0.01994 *
## `TYP_JOBNon Dichiara`                       0.05646 .
## `TYP_JOBOperaio/a`                          0.05950 .
## `TYP_JOBPensionato/a`                       0.00333 **
## `TYP_JOBRappresentante/Agente Commerciale`  0.49964
## TYP_JOBStudente                             0.34890
## `TYP_JOB(missing)`                         3.21e-06 ***
## NUM_SEND_PREV                               0.29442
## AGE_FID                                     < 2e-16 ***
## EMAIL_PROVIDER_CLEANalice.it                0.01958 *
## EMAIL_PROVIDER_CLEANgmail.com               0.02767 *
## EMAIL_PROVIDER_CLEANhotmail.com             0.00736 **
## EMAIL_PROVIDER_CLEANhotmail.it              0.00431 **
## EMAIL_PROVIDER_CLEANlibero.it               0.19534
## EMAIL_PROVIDER_CLEANlive.it                 0.04129 *
## EMAIL_PROVIDER_CLEANothers                  0.02975 *
## EMAIL_PROVIDER_CLEANtiscali.it              0.06687 .
## EMAIL_PROVIDER_CLEANvirgilio.it             0.04768 *
## EMAIL_PROVIDER_CLEANyahoo.it                0.00304 **
## SEND_WEEKDAYgiovedÃ¬                         < 2e-16 ***
## SEND_WEEKDAYlunedÃ¬                          0.50681
## SEND_WEEKDAYmartedÃ¬                         0.76229
## SEND_WEEKDAYmercoledÃ¬                       < 2e-16 ***
## SEND_WEEKDAYsabato                         3.10e-05 ***
## SEND_WEEKDAYvenerdÃ¬                         2.47e-08 ***
## FLAG_DIRECT_MKTflagMKT                      < 2e-16 ***
## FLAG_PRIVACY_1flag1                         < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 312249  on 225239  degrees of freedom
## Residual deviance: 225083  on 225203  degrees of freedom
## AIC: 225157
##
## Number of Fisher Scoring iterations: 5
```

Il coefficiente delle variabili stimato dal modello logistico esprime la variazione del logit della risposta media. Quindi, calcolando l'esponenziale del coefficiente si ottiene l'odds ratio, ovvero la ragione di scommessa. Se l'odds ratio Ã Â¨ pari a 1 significa che non c'Ã Â¨ associazione tra l'esplicativa e la risposta, se invece questo Ã Â¨ maggiore di uno c'Ã Â¨ associazione positiva. Se infine risulta minore di 1, l'associazione Ã Â¨ negativa.

Dall'analisi del summary del modello logistico stimato si evince che le variabili che risultano piÃ Â¹ significative sono le seguenti:

- OPEN_RATE_PREV (odds ratio = exp(3.561) = 35.19838). Calcolando l'odds ratio si evince che c'Ã Â¨ correlazione positiva tra la risposta e questa esplicativa. Questo vuol dire che aumentando di un'unitÃ Â il valore di OPEN_RATE_PREV la ragione di scommessa che la risposta diventi positiva aumenta del 35%.
- CLICK_RATE_PREV (odds ratio = exp(2.386e-01) = 1.269471). Anche per questa esplicativa l'associazione con la risposta Ã Â¨ positiva quindi puÃ Â² essere effettuato un ragionamento analogo alla precedente.

- AGE_FID (odds ratio = exp(-4.153e-04) = 0.9995848). Per questa variabile l'associazione con la risposta Ã Â¨ negativa. Questo vuol dire che aumentando di un giorno "l'etÃ Â " del programma fedeltÃ Â , la probabilitÃ Â che il cliente passi dalla classe negativa a quella positiva scende dello 0,04%.
- SEND_WEEKDAY
- FLAG_DIRECT_MKTflagMKT
- FLAG_PRIVACY_1flag1

Si utilizza a questo scopo un cliente selezionato randomicamente dal test set. In questo caso, il modello di regressione logistica predice correttamente il fatto che il cliente appartiene alla classe negativa "not-opened" di coloro che non aprono la mail entro due giorni dalla data di invio.

```
#test not opened

#test$ID <- seq.int(nrow(test))

user_test1=test[89,]

user_test10<-user_test9<-user_test8<-user_test7<-user_test6<-user_test5<-user_test4<-user_test3<-user_t
user_test2$AGE_FID<-150
user_test3$OPEN_RATE_PREV <- 0.73
user_test4$CLICK_RATE_PREV<-0.5
user_test5$SEND_WEEKDAY<-'lunedÃ¬'
user_test6$SEND_WEEKDAY<-'martedÃ¬'
user_test7$SEND_WEEKDAY<-'mercoledÃ¬'
user_test8$SEND_WEEKDAY<-'venerdÃ¬'
user_test9$SEND_WEEKDAY<-'sabato'
user_test10$SEND_WEEKDAY<-'domenica'

predict(logistic_sub,user_test1)
```

```
## [1] not_opened
## Levels: not_opened opened
```

```
predict(logistic_sub,user_test2) #age fid diminuito
```

```
## [1] not_opened
## Levels: not_opened opened
```

```
predict(logistic_sub,user_test3) #open rate aumentato di 0.4
```

```
## [1] opened
## Levels: not_opened opened
```

```
predict(logistic_sub,user_test4) #click rate aumentato di 0.5
```

```
## [1] not_opened
## Levels: not_opened opened
```

```
predict(logistic_sub,user_test5) #lun
```

```
## [1] not_opened
## Levels: not_opened opened
```

```
predict(logistic_sub,user_test6) #mart
```

```
## [1] not_opened
```

```
## Levels: not_opened opened
```

```
predict(logistic_sub,user_test7) #merc
```

```
## [1] not_opened
## Levels: not_opened opened
```

```
predict(logistic_sub,user_test8) #ven
```

```
## [1] not_opened
## Levels: not_opened opened
```

```
predict(logistic_sub,user_test9) #sab
```

```
## [1] not_opened
## Levels: not_opened opened
```

```
predict(logistic_sub,user_test10) #dom
```

```
## [1] not_opened
## Levels: not_opened opened
```

```
predict(logistic_sub,user_test1, type="prob")
```

```
##     not_opened    opened
## 259  0.7083407 0.2916593
```

```
predict(logistic_sub,user_test2, type="prob")
```

```
##     not_opened    opened
## 259  0.6930645 0.3069355
```

```
predict(logistic_sub,user_test3, type="prob")
```

```
##     not_opened    opened
## 259  0.1515809 0.8484191
```

```
predict(logistic_sub,user_test4, type="prob")
```

```
##     not_opened    opened
## 259  0.6861016 0.3138984
```

```
predict(logistic_sub,user_test5, type="prob")
```

```
##     not_opened    opened
## 259  0.7083407 0.2916593
```

```
predict(logistic_sub,user_test6, type="prob")
```

```
##     not_opened    opened
## 259  0.7133602 0.2866398
```

```
predict(logistic_sub,user_test7, type="prob")
```

```
##     not_opened    opened
## 259  0.7630298 0.2369702
```

```
predict(logistic_sub,user_test8, type="prob")
```

```
##     not_opened    opened
## 259  0.7537852 0.2462148
```

```
predict(logistic_sub,user_test9, type="prob")
```

```
##     not_opened    opened
## 259  0.7554181 0.2445819
```

```
predict(logistic_sub,user_test10, type="prob")
```

```
##     not_opened    opened
## 259  0.7149359 0.2850641
```

Sul cliente test per cui veniva predetta correttamente l'appartenenza alla classe negativa "not-opened" sono stati effettuati i seguenti cambiamenti, sempre lasciando invariate le altre variabili: - da una AGE_FID pari a 418 + stata impostata una AGE_FID pari a 150 e, con una probabilitÃ del 51% il cliente aprirÃ la email; - da un OPER_RATE pari a 0.333 viene aumentato di 0.5 e, con una probabilitÃ del 79%, il cliente aprirÃ la mail; - da un CLICK_RATE pari a zero viene aumentato di 0.5 e, con una probabilitÃ del 52%, il cliente aprirÃ la mail; - il giorno di invio della mail del cliente era il giovedÃ , testando il modello su tutti gli altri giorno il cliente apre la mail, tranne nel caso del mercoledÃ Â¬.

Tramite questo test Ã Â¨ stato possibile avere una conferma empirica dell'importanza delle precedenti variabili nel discriminare se un cliente apre o meno l'email. Infatti, anche cambiando una sola delle precedenti, a paritÃ di tutte le altre, la classe di appartenenza cambia.

E' stato quindi possibile trarre le seguenti conclusioni:

se per limiti di budget la campagna marketing deve essere limitata ad un gruppo ristretto di clienti, allora sarebbe opportuno inviare mail mirate agli utenti che hanno un click e open rate alto e che hanno attivato da non molti giorni il fidelity program. Si Ã Â¨ visto, infatti, che questa tipologia di clienti apre con molto proabilitÃ la mail e continuerebbe a farlo. Se invece l'obiettivo della campagna marketing Ã Â¨ aumentare il numero di clienti che aprono la mail, allora sarebbe opportuno concentrarsi su quei consumatori che hanno attivato il fidelity programme da piÃ Â¹ tempo e con un click e open rate basso. Questo potrebbe essere effettuato tramite l'invio di sconti personalizzati in modo da invogliare il cliente ad aprire la mail e tornare ad essere "attivo".

Per quanto riguarda i giorni della settimana, si potrebbe pensare di non inviare le mail tutte nello stesso giorno ma differenziando in base alle caratteristiche del cliente. Ad esempio, l'individuo utilizzato come test, non apre le mail inviate nei giorni centrali della settimana.

MIGLIORAMENTI E SVILUPPI FUTURI

Il dataset in questione Ã Â¨ risultato molto pesante e questo ha penalizzato e impedito l'utilizzo di algoritmi che risultavano computazionalmente troppo dispendiosi.

Una possibile soluzione potrebbe essere quella di effettuare un'analisi di clustering dei clienti in modo tale da suddividere il dataset in piÃ Â¹ subset per effettuare algoritmi piÃ Â¹ sofisticati e migliorare e affinare i risultati.

PROPENSITY TO CHURN MODEL

Business question Il secondo modello di business sviluppato Ã Â¨ il *propensity to churn model* che ha l'obiettivo di assegnare a ciascun cliente la sua probabilitÃ di abbandono, in modo da implementare specifiche azioni di marketing correttive finalizzate a trattenere i clienti con piÃ Â¹ alto valore.

Lo studio, tramite l'applicazione di algoritmi di machine learning, indaga i possibili comportamenti che portano il cliente ad abbandonare la compagnia per valutare se effettuare possibili campagne di marketing per "trattenere" i clienti.

Per svolgere questo tipo di problema Ã Â¨ stato impostato un modello di classificazione in cui la variabile target consiste in un attributo binario che indica se il cliente Ã Â¨ un churner o meno.

Per “etichettare” il cliente come churner, Ã Â¨ stato necessario definire un time span dopo il quale, secondo le analisi, il consumatore ha un’alta di probabilitÃ Â di non effettuare piÃ Â¹ acquisti.

DATA PREPARATION

Di seguito viene riportata una prima fase di preprocessing dei dati grezzi e, successivamente, una fase di data transformation per definire la variabile target. Inoltre, verranno modificate alcune variabili esistenti e saranno create delle nuove feature.

Set up e Data imput

```
library(dplyr)
library(magrittr)
library(ggplot2)

data_path <- "Laboratorio/"
df_7_tic <- read.csv2(paste0(data_path,"raw_7_tic.csv") , na.strings = c("NA", ""))
```

Preprocessing

```
df_7_tic <- df_7_tic %>%
  dplyr::mutate(DATE = as.Date(DATETIME)) %>%
  dplyr::select(-DATETIME)
```

```
df_7_tic <- df_7_tic %>%
  mutate(ID_CLI = as.factor(ID_CLI))
```

Numero di clienti diversi

```
num_client <- df_7_tic %>%
  dplyr::summarise(client = n_distinct(ID_CLI))
```

Spesa giornaliera per ogni cliente:

```
spend_per_day <- df_7_tic %>%
  dplyr::mutate(ID_CLI = as.factor(ID_CLI)) %>%
  dplyr::group_by(ID_CLI, ID_SCONTRINO, DATE) %>%
  dplyr::summarise(SPEND = sum(IMPORTO_LORDO)) %>%
  dplyr::ungroup() %>%
  dplyr::filter(SPEND > 0)
```

Viene calcolato per ogni cliente il tempo intercorso tra gli acquisti

```
time_between <- spend_per_day %>%
  dplyr::arrange(ID_CLI, DATE) %>%
  dplyr::group_by(ID_CLI) %>%
  dplyr::mutate(dt = as.numeric(DATE - lag(DATE), unit=  'days')) %>%
  dplyr::ungroup() %>%
  na.omit()
```

Numero di transazioni effettuate dal cliente

```
Ntrans <- spend_per_day %>%
  dplyr::group_by(ID_CLI) %>%
  dplyr::summarise(N = n()) %>%
  dplyr::filter(N > 1) %>%
  dplyr::arrange(N)
```

Numero di clienti che hanno effettuato quel determinato numero di transazioni

```
Ntrans2 <- Ntrans %>%
  dplyr::group_by(N) %>%
  dplyr::mutate(n_clienti = n_distinct(ID_CLI)) %>%
  dplyr::filter(row_number(N) == 1) %>%
  dplyr::arrange(N) %>%
  dplyr::select(-ID_CLI)
```

Giunti a questo punto, viene calcolato il percentile di ordine 9 degli intervalli di tempo che intercorrono tra le gli acquisti di ogni cliente (dt). Il valore cosÃ Â¬ trovato, espresso in giorni, indica che 9 volte su 10 il cliente effettua un acquisto entro tot giorni. Quindi un cliente viene considerato churner se, rispetto al 30 aprile 2019 (ovvero l'ultimo giorno in cui si ha la disponibilitÃ Â dei dati) i giorni per cui egli non ha effettuato un acquisto sono maggiori rispetto al suo percentile.

Per definire se un cliente Ã Â¨ churner o no sono stati considerati soltanto i clienti ritenuti non occasionali, ovvero coloro che hanno effettuato un numero di transazioni pari almeno a 5.

Nel calcolo del percentile viene utilizzato un approccio non parametrico. Nello specifico verrÃ Â presa in considerazione una distribuzione di tipo ECDF (Empirical Cumulative Distribution Function), in grado di approssimare il quantile della distribuzione degli acquisti del cliente.

```
get_quantile <- function(x,a = 0.9){
  if(a>1|a<0){
    print('Check your quantile')
  }
  X <- sort(x)
  e_cdf <- 1:length(X) / length(X)
  aprx = approx(e_cdf, X, xout = c(0.9))
  return(aprx$y)
}

percentiles <- time_between %>%
  dplyr::inner_join(Ntrans) %>%
  dplyr::filter(N>4) %>%
  dplyr::group_by(ID_CLI) %>%
  dplyr::summarise(percentile.90= get_quantile(dt, 0.9)) %>%
  dplyr::arrange(percentile.90)
```

```
## Joining, by = "ID_CLI"
```

Si arrotonda il percentile e si fissa un giorno come soglia di tolleranza.

```
percentiles <- percentiles %>%
  dplyr::mutate(percentile_round = round(percentile.90) + 1)
```

Si fissa quindi, per ogni cliente, la data in cui Ã Â¨ stato effettuato l'ultimo acquisto

```
last_purchase <- time_between %>%
  dplyr::group_by(ID_CLI) %>%
  dplyr::filter(DATE == max(DATE)) %>%
  dplyr::distinct(ID_CLI, .keep_all = T)
```

Si ordina il dataset contente le informazioni di interesse sui clienti, in base alla data.

```
df_7_tic <- df_7_tic %>%
  dplyr::arrange(DATE)
```

Viene identificata come last_day l'ultima data per cui sono disponibili le rilevazioni sul dataset e come days_no_purc i giorni intercorsi tra l'ultimo acquisto e questa data.

```
last_purchase <- last_purchase %>%
  dplyr::mutate(last_day = as.Date("2019-04-30")) %>%
  dplyr::mutate(days_no_purc = last_day - DATE)
```

Si uniscono quindi i dataset in cui Ã Â¨ stato costruito il percentile ed il dataset in cui sono presenti le variabili utili alla definizione dei clienti churner (in particolare days_no_churn).

```
last_purc_churn <- dplyr::left_join(last_purchase, percentiles, by = "ID_CLI")
```

Si costruisce, in via definitiva, la variabile dipendente cosÃ Â¬ come Ã Â¨ stata definita nei passi precedenti. Il valore della variabile Ã Â¨ pari ad 1 se il cliente Ã Â¨ churner, 0 altrimenti.

Un cliente sarÃ Â churner se il numero di giorni percorsi dal suo ultimo acquisto sono maggiori di quanto Ã Â¨ solito fare 9 volte su 10. Ad esempio se un cliente non ha acquistato per 70 giorni, ma 9 volte su 10 acquista entro 50 giorni, sarÃ Â considerato churner.

```
last_purc_churn <- last_purc_churn %>%
  na.omit() %>%
  dplyr::mutate(churn = ifelse(days_no_purc > percentile_round, 1, 0))
```

Giunti a questo punto last_purc_churn Ã Â¨ il dataset in cui Ã Â¨ stata definita la variabile target.

Si concentra ora l'analisi sul dataset iniziale df_7_tic, per riuscire ad ottenere alcune indicazioni e caratteristiche dei clienti, che potrebbero tornare utili nella fase di Modelling. In particolare sono create 3 variabili:

1) attitudine_sconto: tendenza del cliente ad acquistare prodotti per cui Ã Â¨ attivo uno sconto. Questa variabile Ã Â¨ data dalla percentuale del rapporto tra gli articoli acquistati in saldo durante tutto l'anno sul totale degli articoli acquistati.

2) perc_risparmio: percentuale di risparmio. Questa variabile Ã Â¨ data dalla percentuale del rapporto tra il totale degli euro risparmiati sul totale della spesa effettuata durante tutto l'anno.

3) perc_refound: percentuale di reso. Questa variabile Ã Â¨ data dalla percentuale del rapporto tra il totale degli articoli restituiti e sul totale degli articoli acquistati (e restituiti).

```
control_direction <- df_7_tic %>%
  dplyr::mutate(ID_CLI = as.factor(ID_CLI)) %>%
  dplyr::group_by(ID_CLI) %>%
  dplyr::summarise(perc_refound = (sum(DIREZIONE == -1) / length(DIREZIONE)) * 100, perc_risparmio = (su
  ungroup()
```

Numero di clienti churner e non churner

```
table(last_purc_churn$churn)
```

```
##
##     0     1
## 29180 29211
```

Dalla tabella di frequenza della variabile target, si evince che le due classi identificate appaiono equamente bilanciate.

Si crea un file csv del dataset costruito fin qui, in modo tale da poterlo successivamente leggere, senza passare nuovamente tutte le fasi precedenti.

```
write.csv(last_purc_churn, paste0(data_path, "churner.csv"))
```

Si legge il file prima creato

```
df_churn <- read.csv(paste0(data_path, "churner.csv"))
```

Si calcola il totale della spesa e la media della spesa di ogni cliente

```
total_spend <- spend_per_day %>%
  group_by(ID_CLI) %>%
  summarise(total_spend = sum(SPEND), mean_spend = mean(SPEND))
```

Modifica del tipo di ID_CLI, utile per dopo.

```
df_churn <- df_churn %>%
  mutate(ID_CLI = as.factor(ID_CLI))
```

Fusione tra il dataset letto e quello contenente i dati sulla spesa (spesa totale e media per cliente).

```
df_churn <- inner_join(df_churn, total_spend) %>%
  dplyr::select(-SPEND)
```

```
## Joining, by = "ID_CLI"
```

```
## Warning: Column `ID_CLI` joining factors with different levels, coercing to
## character vector
```

Data integration

Inizia ora la fase di integrazione dei vari dataset a disposizione. L'obiettivo e integrare tutte le caratteristiche che potrebbero tornare utili per identificare e poi prevedere un cliente churner. Le variabile giudicate ridondanti o non utili all'analisi, saranno eliminate durante ogni join.

Fusione con il primo dataset con le informazioni sul cliente (df_1_cli_fid_clean)

```
df_1_cli_fid_clean <- df_1_cli_fid_clean %>%
  dplyr::mutate(ID_CLI = as.factor(ID_CLI))

df_churn_cli_1 <- inner_join(df_churn, df_1_cli_fid_clean, by = "ID_CLI") %>%
  dplyr::select(-c(ID_FID, FIRST_ID_NEG, FIRST_DT_ACTIVE)) %>%
  dplyr::mutate(date_last_pur = DATE) %>%
  dplyr::select(-DATE)
```

```
## Warning: Column `ID_CLI` joining character vector and factor, coercing into
## character vector
```

Fusione con il dataset 2, dopo aver fuso il dataset 2 con le informazioni contenute nel 3.

```
df_2_cli_account_clean <- df_2_cli_account_clean %>%
  mutate(ID_CLI = as.factor(ID_CLI))

df_2_cli_account_clean <- inner_join(df_2_cli_account_clean, df_3_cli_address_clean, by = "ID_ADDRESS")


df_churn_cli_2 <- inner_join(df_churn_cli_1, df_2_cli_account_clean, by= "ID_CLI") %>%
  select(-c(ID_ADDRESS, EMAIL_PROVIDER_CLEAN))
```

```
## Warning: Column `ID_CLI` joining character vector and factor, coercing into
## character vector
```

Fusione con il dataset 4.

```
df_4_cli_privacy_clean <- df_4_cli_privacy_clean %>%
  mutate(ID_CLI = as.factor(ID_CLI))

df_churn_cli_4 <- inner_join(df_churn_cli_2, df_4_cli_privacy_clean, by = "ID_CLI")
```

```
## Warning: Column `ID_CLI` joining character vector and factor, coercing into
```

```
## character vector
```

Eliminazione della variabile avente gli indici (ridondante)

```
df_churn_cli_4 <- df_churn_cli_4 %>%
  select(-X)
```

Modifica di alcune variabili

```
df_churn_cli_4 <- df_churn_cli_4 %>%
  mutate(W_PHONE = as.factor(W_PHONE))

df_churn_cli_4 <- df_churn_cli_4 %>%
  mutate(TYP_CLI_ACCOUNT = as.factor(TYP_CLI_ACCOUNT))

df_churn_cli_4 <- df_churn_cli_4 %>%
  mutate(W_PHONE = fct_explicit_na(W_PHONE, "0")) %>%
  mutate(TYP_JOB = fct_explicit_na(TYP_JOB, "(missing)"))
```

Modifica di alcune variabili ed eliminazione di altre. Inoltre fusione del dataset ottenuto da tutte le join eseguite in precedenza, con il dataset avente le 3 variabili create ex novo. (perc_refound, perc_risparmio, attitudine_sconto)

```
data_churn_all <- df_churn_cli_4

data_churn_all <- data_churn_all %>%
  mutate(ID_CLI = as.factor(ID_CLI))

data_churn_all <- data_churn_all %>%
  mutate(churn = as.factor(churn)) %>%
  mutate(CAP = as.factor(CAP)) %>%
  select(-ID_SCONTRINO,-last_day,-percentile.90,-percentile_round,-total_spend,-date_last_pur)

data_churn_all <- left_join(data_churn_all, control_direction, by= "ID_CLI")
```

```
## Warning: Column `ID_CLI` joining factors with different levels, coercing to
## character vector
```

Di nuovo, come prima, viene scritto il file per evitare di partire ogni volta dal principio.

```
write.csv(data_churn_all, paste0(data_path, "data_churn_all.csv"))
```

```
data_churn_all <- read.csv(paste0(data_path, "data_churn_all.csv"))
```

Si passa quindi alla fase di data modeling.

MODELING

SETUP

```
library(caret)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-18
```

```
library(e1071)
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(arm)
```

```
## Loading required package: lme4
## Registered S3 methods overwritten by 'lme4':
##   method                          from
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car
##
## arm (Version 1.10-1, built: 2018-4-12)
## Working directory is C:/Users/GiuliaChiaretti(Stag/Desktop/Digital_Marketing_DataScience_Project
##
## Attaching package: 'arm'
## The following object is masked from 'package:car':
##
##     logit
## The following object is masked from 'package:corrplot':
##
##     corrplot
```

```
library(MCMCpack)
```

```
## Loading required package: coda
##
## Attaching package: 'coda'
## The following object is masked from 'package:arm':
##
##     traceplot
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
## ## Copyright (C) 2003-2019 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
## ##
## ## Support provided by the U.S. National Science Foundation
```

```
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following object is masked from 'package:glmnet':
##
##     auc
```

```
## The following object is masked from 'package:plotROC':
##
##     ggroc
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(plotROC)
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:gridExtra':
##
##     combine
```

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

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

```
data_path <- "Laboratorio/"
```

READ DATA

```
data_churn_all <- read.csv(paste0(data_path, "data_churn_all.csv"))

data_churn_all <- data_churn_all %>%
  dplyr::mutate(NUM_FIDs = as.factor(NUM_FIDs)
         , W_PHONE = as.factor(W_PHONE)
         , TYP_CLI_ACCOUNT = as.factor(TYP_CLI_ACCOUNT)
         , churn = as.factor(churn)
         , TYP_CLI_FID = as.factor(TYP_CLI_FID)
         , STATUS_FID = as.factor(STATUS_FID)
         , FLAG_PRIVACY_1 = as.factor(FLAG_PRIVACY_1)
         , FLAG_PRIVACY_2 = as.factor(FLAG_PRIVACY_2)
         , FLAG_DIRECT_MKT = as.factor(FLAG_DIRECT_MKT)) %>%
  dplyr::select(-c(X))
```

Letti i dati precedentemente processati ed effettuati i dovuti accorgimenti, si passa alla fase di data modelling. L'obiettivo Ã Â¨ scoprire ed evidenziare le caratteristiche principali che rendono portano un cliente a diventare Churner, per poi riuscire a prevedere e contrastare un eventuale Churn da parte di altri consumatori. Abbiamo quindi una variabile target: Churn (1 se il cliente Ã Â¨ Churner, 0 altrimenti) e diverse variabili esplicative. Si tratta di un problema di Supervised learning. Si decide, quindi, come prima ipotesi, di stimare un modello di regressione logistica attraverso la funzione glm. Dal dataset letto in partenza vengono rimosse le variabili considerate ridondanti, non utili allo scopo o nelle quali Ã Â¨ presente una quantitÃ Â molto elevata di missing. Training e test set vengono separati attribuendo rispettivamente il 75% delle osservazioni al primo e il restante 25% al secondo.

LOGISTIC REGRESSION

```
data_logistic <- data_churn_all %>%
  dplyr::select(-c(PRV, CAP, REGION, ID_CLI, ID_NEG, TYP_JOB)) %>%
  dplyr::mutate(DT_ACTIVE = as.Date(DT_ACTIVE))

set.seed(12345)
training_logistic <- data_logistic$churn %>%
  createDataPartition(p = 0.75, list = F)

train_logistic <- data_logistic[training_logistic,]
test_logistic <- data_logistic[-training_logistic,]

mod_logistic <- glm(churn ~., family = "binomial", data = train_logistic)
```

```
summary(mod_logistic)
```

```
##
## Call:
## glm(formula = churn ~ ., family = "binomial", data = train_logistic)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5009  -0.3596   0.0000   0.1257   3.5099
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.605e+02  1.195e+02  -1.343    0.179
## dt                  -1.886e-02  5.101e-04 -36.977  < 2e-16 ***
## days_no_purc         8.502e-02  9.405e-04  90.397  < 2e-16 ***
## mean_spend          -5.181e-04  5.565e-05  -9.310  < 2e-16 ***
## TYP_CLI_FID1         1.654e-01  1.978e-01   0.836    0.403
## COD_FIDPREMIUM BIZ   9.335e+00  1.195e+02   0.078    0.938
## COD_FIDSTANDARD     -4.835e-01  4.465e-02 -10.828  < 2e-16 ***
## COD_FIDSTANDARD BIZ  9.155e+00  1.195e+02   0.077    0.939
## STATUS_FID1         -5.941e-01  6.144e-01  -0.967    0.334
## DT_ACTIVE            8.356e-03  1.789e-04  46.705  < 2e-16 ***
## NUM_FIDs2           -5.321e-01  3.009e-01  -1.768    0.077 .
## W_PHONE1            -6.460e-02  5.918e-02  -1.092    0.275
## TYP_CLI_ACCOUNT4     9.295e+00  1.195e+02   0.078    0.938
## FLAG_PRIVACY_11      7.833e-02  6.611e-02   1.185    0.236
## FLAG_PRIVACY_21     -2.091e-03  6.167e-02  -0.034    0.973
## FLAG_DIRECT_MKT1    -4.704e-02  6.810e-02  -0.691    0.490
## perc_refound         1.626e-02  2.559e-03   6.353 2.11e-10 ***
## perc_risparmio      -3.806e-03  3.269e-03  -1.164    0.244
```

```
## attitudine_sconto    1.045e-02  9.839e-04  10.620  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58928  on 42507  degrees of freedom
## Residual deviance: 20650  on 42489  degrees of freedom
## AIC: 20688
##
## Number of Fisher Scoring iterations: 9
```

Previsioni sul test set

```
probabilities_logistic <- predict(mod_logistic, test_logistic, type="response")
predicted_logistic = ifelse(probabilities_logistic > 0.5, 1, 0)
observed_classes <- test_logistic$churn
mean(predicted_logistic == observed_classes)
```

```
## [1] 0.8986448
```

Vengono quindi indagati ulteriori caratteristiche del modello utilizzato. In particolare si analizza la variance inflation function per indagare eventuale collinearitÃ delle feature utilizzate nel modello stimato.

Ulteriori osservazioni

```
vif(mod_logistic)
```

```
##                          GVIF Df GVIF^(1/(2*Df))
## dt                1.276649e+00  1        1.129889
## days_no_purc      1.790783e+00  1        1.338201
## mean_spend        1.215934e+00  1        1.102694
## TYP_CLI_FID       1.009312e+00  1        1.004645
## COD_FID           6.438998e+06  3       13.639644
## STATUS_FID        1.000467e+00  1        1.000234
## DT_ACTIVE         1.559510e+00  1        1.248803
## NUM_FIDs          1.004399e+00  1        1.002197
## W_PHONE           1.036588e+00  1        1.018130
## TYP_CLI_ACCOUNT   4.815605e+06  1     2194.448750
## FLAG_PRIVACY_1    2.918331e+00  1        1.708312
## FLAG_PRIVACY_2    1.153689e+00  1        1.074099
## FLAG_DIRECT_MKT   2.980663e+00  1        1.726460
## perc_refound      1.088202e+00  1        1.043169
## perc_risparmio    1.300643e+00  1        1.140457
## attitudine_sconto 1.689546e+00  1        1.299825
```

Come possiamo vedere dalla funzione VIF, la quale restituisce il grado di multicollinearitÃ delle variabili inserite nel modello, le feature TYP_CLI_ACCOUNT e COD_FID sono collineari. Detto ciÃ², risulta utile considerare soltanto una delle due presenti. Viene eliminata per tale motivo la prima delle due e si ristima il modello.

VIFj = 1/1 - Rj

NUOVO MODELLO DI REGRESSIONE LOGISTICA

```
data_logistic <- data_churn_all %>%
  dplyr::select(-c(PRV, CAP, REGION, ID_CLI, ID_NEG, TYP_JOB, TYP_CLI_ACCOUNT)) %>%
  dplyr::mutate(DT_ACTIVE = as.Date(DT_ACTIVE))
```

```r
set.seed(12345)
training_logistic <- data_logistic$churn %>%
  createDataPartition(p = 0.75, list = F)

train_logistic <- data_logistic[training_logistic,]
test_logistic <- data_logistic[-training_logistic,]

mod_logistic <- glm(churn ~., family = "binomial", data = train_logistic)
```

```r
summary(mod_logistic)
```

```
##
## Call:
## glm(formula = churn ~ ., family = "binomial", data = train_logistic)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5013  -0.3595   0.0000   0.1258   3.5100
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.512e+02  3.273e+00 -46.206  < 2e-16 ***
## dt                  -1.886e-02  5.101e-04 -36.980  < 2e-16 ***
## days_no_purc         8.502e-02  9.404e-04  90.410  < 2e-16 ***
## mean_spend          -5.181e-04  5.565e-05  -9.311  < 2e-16 ***
## TYP_CLI_FID1         1.655e-01  1.978e-01   0.836   0.4029
## COD_FIDPREMIUM BIZ   4.080e-02  8.410e-02   0.485   0.6276
## COD_FIDSTANDARD     -4.836e-01  4.465e-02 -10.830  < 2e-16 ***
## COD_FIDSTANDARD BIZ -1.399e-01  8.147e-02  -1.718   0.0858 .
## STATUS_FID1         -5.939e-01  6.144e-01  -0.967   0.3337
## DT_ACTIVE            8.356e-03  1.789e-04  46.714  < 2e-16 ***
## NUM_FIDs2           -5.322e-01  3.009e-01  -1.769   0.0769 .
## W_PHONE1            -6.453e-02  5.918e-02  -1.090   0.2755
## FLAG_PRIVACY_11      7.826e-02  6.611e-02   1.184   0.2365
## FLAG_PRIVACY_21     -2.038e-03  6.167e-02  -0.033   0.9736
## FLAG_DIRECT_MKT1    -4.713e-02  6.810e-02  -0.692   0.4889
## perc_refound         1.626e-02  2.559e-03   6.355 2.08e-10 ***
## perc_risparmio      -3.811e-03  3.269e-03  -1.166   0.2437
## attitudine_sconto    1.045e-02  9.839e-04  10.617  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58928  on 42507  degrees of freedom
## Residual deviance: 20651  on 42490  degrees of freedom
## AIC: 20687
##
## Number of Fisher Scoring iterations: 7
```

NUOVE PREVISIONI SUL TEST SET

```r
probabilities_logistic <- predict(mod_logistic, test_logistic, type="response")
predicted_logistic = ifelse(probabilities_logistic > 0.5, 1, 0)
observed_classes <- test_logistic$churn
```

```
mean(predicted_logistic == observed_classes)
```

```
## [1] 0.898786
```

VARIANCE INFLATION FUNCTION

```
vif(mod_logistic)
```

```
##                       GVIF Df GVIF^(1/(2*Df))
## dt                1.276612  1        1.129873
## days_no_purc      1.790637  1        1.338147
## mean_spend        1.215932  1        1.102693
## TYP_CLI_FID       1.009312  1        1.004645
## COD_FID           1.474989  3        1.066919
## STATUS_FID        1.000467  1        1.000234
## DT_ACTIVE         1.559477  1        1.248790
## NUM_FIDs          1.004399  1        1.002197
## W_PHONE           1.036587  1        1.018129
## FLAG_PRIVACY_1    2.918718  1        1.708426
## FLAG_PRIVACY_2    1.153692  1        1.074100
## FLAG_DIRECT_MKT   2.981092  1        1.726584
## perc_refound      1.088192  1        1.043165
## perc_risparmio    1.300679  1        1.140473
## attitudine_sconto 1.689533  1        1.299820
```

Come mostrato dai risultati, la media delle volte in cui i risultati delle previsioni risultano uguali ai valori osservati della variabile target resta uguale (pari circa al 90%). Inoltre viene risolta la collinearitÃ delle variabili esplicative, come visto dalla funzione VIF.

Si decide a questo punto di indagare la possibilitÃ di rimozione di alcune variabili dal modello. Nello specifico viene applicata la logica di selezione stepwise in entrambe le direzioni possibili (forward and backword). Il risultato proveniente da questa tecnica riesce ad indicare il modello per cui risulta minimo l'Akaike Information Criterion, metodo di valutazione in grado di fornire la qualitÃ di stima di un modello, considerando congiuntamente la bontÃ di adattamento e la complessitÃ dello stesso.

```
step.model <- stepAIC(mod_logistic, direction = "both",
                      trace = FALSE)
step.model
```

```
##
## Call:  glm(formula = churn ~ dt + days_no_purc + mean_spend + COD_FID +
##     DT_ACTIVE + NUM_FIDs + perc_refound + attitudine_sconto,
##     family = "binomial", data = train_logistic)
##
## Coefficients:
##         (Intercept)                   dt         days_no_purc
##           -1.518e+02           -1.888e-02              8.504e-02
##           mean_spend   COD_FIDPREMIUM BIZ      COD_FIDSTANDARD
##           -5.160e-04            3.002e-02             -4.892e-01
## COD_FIDSTANDARD BIZ            DT_ACTIVE              NUM_FIDs2
##           -1.489e-01            8.360e-03             -5.227e-01
##         perc_refound    attitudine_sconto
##            1.642e-02            9.977e-03
##
## Degrees of Freedom: 42507 Total (i.e. Null);  42497 Residual
## Null Deviance:       58930
## Residual Deviance: 20660     AIC: 20680
```

Si procede quindi nel definire il modello di regressione logistica facendo affidamento su quanto emerso dalla stepwise model selection

STEPWISE MODEL

```
logistic_step <- glm(formula = churn ~ dt + days_no_purc + mean_spend + COD_FID + DT_ACTIVE + FLAG_PRIV
```

```
summary(logistic_step)
```

```
##
## Call:
## glm(formula = churn ~ dt + days_no_purc + mean_spend + COD_FID +
##     DT_ACTIVE + FLAG_PRIVACY_1 + perc_refound + attitudine_sconto,
##     family = "binomial", data = train_logistic)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5061  -0.3591   0.0000   0.1262   3.5173
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.517e+02  3.134e+00 -48.397  < 2e-16 ***
## dt                  -1.887e-02  5.098e-04 -37.014  < 2e-16 ***
## days_no_purc         8.502e-02  9.396e-04  90.484  < 2e-16 ***
## mean_spend          -5.156e-04  5.530e-05  -9.323  < 2e-16 ***
## COD_FIDPREMIUM BIZ   4.492e-02  8.257e-02   0.544   0.5864
## COD_FIDSTANDARD     -4.854e-01  4.345e-02 -11.172  < 2e-16 ***
## COD_FIDSTANDARD BIZ -1.329e-01  7.998e-02  -1.662   0.0965 .
## DT_ACTIVE            8.353e-03  1.752e-04  47.689  < 2e-16 ***
## FLAG_PRIVACY_11      4.163e-02  3.984e-02   1.045   0.2960
## perc_refound         1.635e-02  2.556e-03   6.398 1.58e-10 ***
## attitudine_sconto    9.962e-03  8.894e-04  11.200  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58928  on 42507  degrees of freedom
## Residual deviance: 20658  on 42497  degrees of freedom
## AIC: 20680
##
## Number of Fisher Scoring iterations: 7
```

Com'Ã Â¨ possibile notare dai risultati, il livello di AIC Ã Â¨ sceso. Inoltre tutte le variabili inserite nella stima, presentano un coefficiente significativo.

NUOVE PREVISIONI SUL TEST SET

```
probabilities_logistic_step <- predict(logistic_step, test_logistic, type="response")
predicted_logistic_step = ifelse(probabilities_logistic > 0.5, 1, 0)
observed_classes <- test_logistic$churn
mean(predicted_logistic_step == observed_classes)
```

```
## [1] 0.898786
```

I risultati rimangono ottimi e viene ridotta notevolmente la complessitÃ Â del modello utilizzato.

Per testare in modo consistente la bontÃ Â delle previsioni del modello, vengono calcolate tre metriche che

indagano la capacitÃ dell'algoritmo di assegnare in modo corretto una osservazione come churner o non churner quando realmente questa appartiene a le due categorie: F1 measure, Precision and Recall:

```
caret::F_meas(data = as.factor(predicted_logistic_step), reference = test_logistic$churn)
```

```
## [1] 0.9010762
```

```
caret::precision(data = as.factor(predicted_logistic_step), reference = test_logistic$churn)
```

```
## [1] 0.8794775
```

```
caret::recall(data = as.factor(predicted_logistic_step), reference = test_logistic$churn)
```

```
## [1] 0.9237624
```

Le tre metriche calcolate danno ottimi risultati.

Si decide ora di passare ad un tipo di algoritmo diverso. Si procede aggiungendo una penalitÃ per controllare le proprietÃ dei coefficienti di regressione, andando oltre a quanto la mera funzione di verosimiglianza permette.

Viene infatti cercata l'ottimizzazione della verosimiglianza e della penalitÃ piuttosto che tentare di massimizzare solamente la prima.

Si parla in questo caso di Penalized logistic regression. Questa impone una penalitÃ al modello logistico quando questo presenta troppe variabili. Questo metodo, chiamato anche regolarizzazione, permette di portare a zero il valore assoluto dei coefficienti delle variabili meno costributive per il modello stimato.

I modelli di regressione penalizzata piÃ¹ comuni sono: - Ridge Regression - Lasso Regression - Elastic Net Regression

Si decide di applicare la regressione con penalitÃ di tipo LASSO, la quale permette di forzare il valore assoluto dei coefficienti meno significativi a 0.

REGRESSIONE LASSO

```
data_lasso <- data_logistic
set.seed(12345)
training <- data_lasso$churn %>%
  createDataPartition(p = 0.75, list = F)

train_lasso <- data_lasso[training,]
test_lasso <- data_lasso[-training,]
```

```
x <- model.matrix(churn~., train_lasso)[,-1]
y <- train_lasso$churn
```

```
set.seed(12345)
cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial")
model_lasso <- glmnet(x, y, alpha = 1, family = "binomial",
                lambda = cv.lasso$lambda.min)
coef(model_lasso)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                                s0
## (Intercept)         -1.491763e+02
## dt                  -1.849445e-02
## days_no_purc         8.379360e-02
## mean_spend          -4.878645e-04
## TYP_CLI_FID1         1.087767e-01
## COD_FIDPREMIUM BIZ   3.147183e-02
```

```
## COD_FIDSTANDARD     -4.684892e-01
## COD_FIDSTANDARD BIZ -1.143329e-01
## STATUS_FID1         -3.828704e-01
## DT_ACTIVE            8.231905e-03
## NUM_FIDs2           -4.516960e-01
## W_PHONE1            -5.189208e-02
## FLAG_PRIVACY_11      3.238197e-02
## FLAG_PRIVACY_21      .
## FLAG_DIRECT_MKT1     .
## perc_refound         1.575869e-02
## perc_risparmio      -2.558115e-03
## attitudine_sconto    1.001983e-02
```

L'unica variabile completamente priva di valore aggiunto per spiegare la variabile target d'interesse, risulta essere FLAG_DIRECT_MKT

PREVISIONI LASSO

```
x.test <- model.matrix(churn ~., test_lasso)[,-1]
probabilities <- model_lasso %>% predict(newx = x.test)
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)
observed.classes <- test_lasso$churn
mean(predicted.classes == observed.classes)
```

```
## [1] 0.8904574
```

Si cerca ora di ottimizzare i parametri riguardanti la specificazione del modello, in particolare, viene indagato il tipo di parametro lambda che permette di ottenere il modello piÃ Â¹ accurato:

```
cv.lasso$lambda.min
```

```
## [1] 0.0003145739
```

```
coef(cv.lasso, cv.lasso$lambda.min)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)         -1.492024e+02
## dt                  -1.849976e-02
## days_no_purc         8.381355e-02
## mean_spend          -4.880367e-04
## TYP_CLI_FID1         1.087409e-01
## COD_FIDPREMIUM BIZ   3.146551e-02
## COD_FIDSTANDARD     -4.685261e-01
## COD_FIDSTANDARD BIZ -1.142629e-01
## STATUS_FID1         -3.829899e-01
## DT_ACTIVE            8.233340e-03
## NUM_FIDs2           -4.518267e-01
## W_PHONE1            -5.190066e-02
## FLAG_PRIVACY_11      3.238961e-02
## FLAG_PRIVACY_21      .
## FLAG_DIRECT_MKT1     .
## perc_refound         1.576064e-02
## perc_risparmio      -2.557011e-03
## attitudine_sconto    1.002219e-02
```

Ed anche quello che dÃ Â prova del modello piÃ Â¹ semplice

```
cv.lasso$lambda.1se
```

```
## [1] 0.002933727
```

```
coef(cv.lasso, cv.lasso$lambda.1se)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                                 1
## (Intercept)         -1.333876e+02
## dt                  -1.600330e-02
## days_no_purc         7.550194e-02
## mean_spend          -1.945255e-04
## TYP_CLI_FID1         .
## COD_FIDPREMIUM BIZ   .
## COD_FIDSTANDARD     -3.493880e-01
## COD_FIDSTANDARD BIZ  .
## STATUS_FID1          .
## DT_ACTIVE            7.337651e-03
## NUM_FIDs2            .
## W_PHONE1             .
## FLAG_PRIVACY_11      .
## FLAG_PRIVACY_21      .
## FLAG_DIRECT_MKT1     .
## perc_refound         1.184904e-02
## perc_risparmio       .
## attitudine_sconto    7.360997e-03
```

Quest'ultimo parametro pare essere troppo restringente; vengono infatti posti pari a zero anche i coefficienti delle variabili considerate nella stima del modello di regressione logistica iniziale. Si stima, ad ogni modo, il modello di regressione logistica penalizzata considerando entrambi i lambda menzionati in precedenza.

Con Lambda min

```
model_lasso <- glmnet(x, y, alpha = 1, family = "binomial",
                      lambda = cv.lasso$lambda.min)
x.test <- model.matrix(churn ~., test_lasso)[,-1]
probabilities <- model_lasso %>% predict(newx = x.test)
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)
observed.classes <- test_lasso$churn
mean(predicted.classes == observed.classes)
```

```
## [1] 0.8904574
```

Con lambda 1se

```
model_lasso_1se <- glmnet(x, y, alpha = 1, family = "binomial",
                      lambda = cv.lasso$lambda.1se)
x.test <- model.matrix(churn ~., test_lasso)[,-1]
probabilities <- model_lasso_1se %>% predict(newx = x.test)
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)
observed.classes <- test_lasso$churn
mean(predicted.classes == observed.classes)
```

```
## [1] 0.8875635
```

I risultati sono molto interessanti. Il lambda 1se permette di semplificare molto la stima del modello e, pur eliminando diverse variabili esplicative, restituisce un'accuratezza delle previsioni quasi identica a quella

ottenuta con Lambda min. Si potrebbe optare quindi per tenere in considerazione l'ultimo degli algoritmi sviluppati. Risultati molto simili, ma semplicitÃ maggiore.

Si prova ora a considerare un tipo di regressione logistica basata sulla statistica Bayesiana.

```
model_bayesian_glm <- bayesglm(churn ~., family = "binomial", data = train_logistic, prior.df = Inf, pri
summary(model_bayesian_glm)
```

```
##
## Call:
## bayesglm(formula = churn ~ ., family = "binomial", data = train_logistic,
##     prior.scale = Inf, prior.df = Inf)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5013  -0.3595   0.0000   0.1258   3.5100
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.512e+02  3.273e+00 -46.206  < 2e-16 ***
## dt                  -1.886e-02  5.101e-04 -36.980  < 2e-16 ***
## days_no_purc         8.502e-02  9.404e-04  90.410  < 2e-16 ***
## mean_spend          -5.181e-04  5.565e-05  -9.311  < 2e-16 ***
## TYP_CLI_FID1         1.655e-01  1.978e-01   0.836   0.4029
## COD_FIDPREMIUM BIZ   4.080e-02  8.410e-02   0.485   0.6276
## COD_FIDSTANDARD     -4.836e-01  4.465e-02 -10.830  < 2e-16 ***
## COD_FIDSTANDARD BIZ -1.399e-01  8.147e-02  -1.718   0.0858 .
## STATUS_FID1         -5.939e-01  6.144e-01  -0.967   0.3337
## DT_ACTIVE            8.356e-03  1.789e-04  46.714  < 2e-16 ***
## NUM_FIDs2           -5.322e-01  3.009e-01  -1.769   0.0769 .
## W_PHONE1            -6.453e-02  5.918e-02  -1.090   0.2755
## FLAG_PRIVACY_11      7.826e-02  6.611e-02   1.184   0.2365
## FLAG_PRIVACY_21     -2.038e-03  6.167e-02  -0.033   0.9736
## FLAG_DIRECT_MKT1    -4.713e-02  6.810e-02  -0.692   0.4889
## perc_refound         1.626e-02  2.559e-03   6.355 2.08e-10 ***
## perc_risparmio      -3.811e-03  3.269e-03  -1.166   0.2437
## attitudine_sconto    1.045e-02  9.839e-04  10.617  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58928  on 42507  degrees of freedom
## Residual deviance: 20651  on 42490  degrees of freedom
## AIC: 20687
##
## Number of Fisher Scoring iterations: 7
```

L'output del modello coincide con quanto emerso dalla regressione logistica tradizionale. In veritÃ questo si presenta come risultato piÃ¹ probabile. Infatti, al crescere della numerositÃ del campione, i risultati dei due modelli menzionati, dovrebbero combaciare (come avviene realmente).

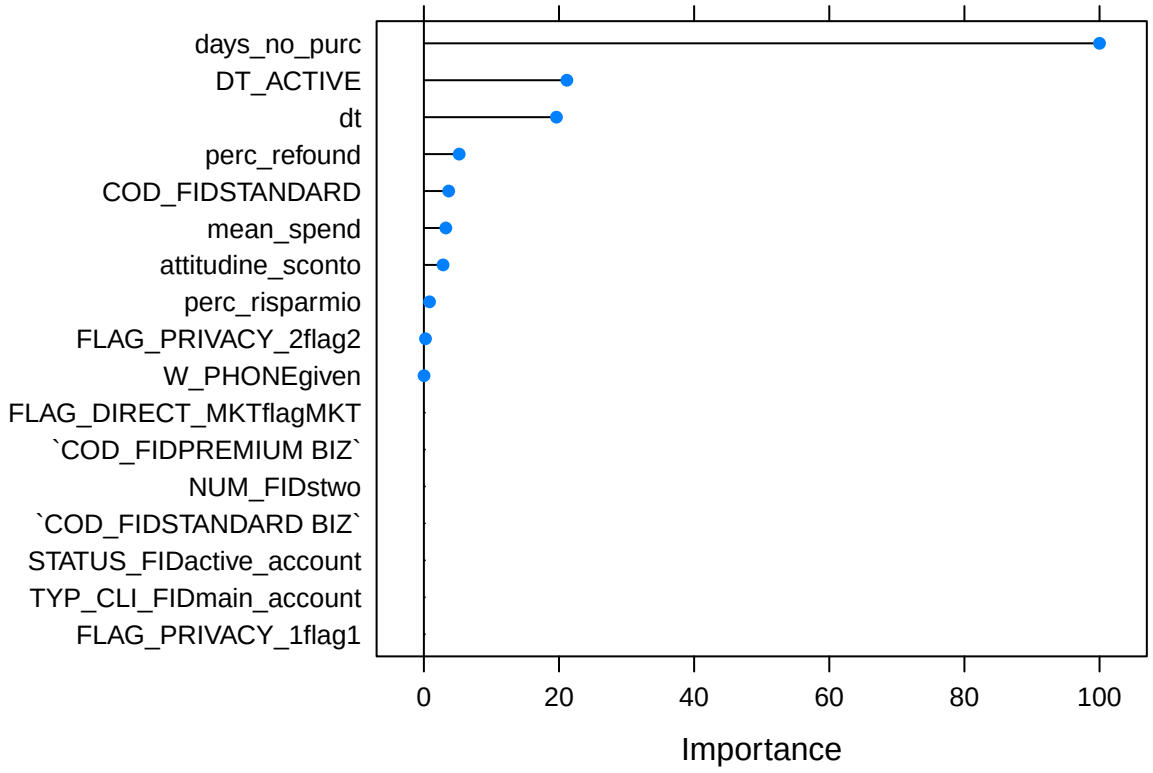
Si sceglie a questo punto di utilizzare un albero decisionale per provare ad indagare la relazione tra le variabili e la loro importanza. Il fine Ã¨ poi utilizzare i risultati emersi dall'albero per stimare una rete neurale di tipo NNET.

```r
data_decision <- data_logistic

set.seed(12345)
levels(data_decision$churn) <- c("not_churner", "churner")
levels(data_decision$TYP_CLI_FID) <- c("not_main", "main_account")
levels(data_decision$STATUS_FID) <- c("not_active", "active_account")
levels(data_decision$NUM_FIDs) <- c("one", "two")
levels(data_decision$W_PHONE) <- c("not_given", "given")
levels(data_decision$FLAG_PRIVACY_1) <- c("not_flag1", "flag1")
levels(data_decision$FLAG_PRIVACY_2) <- c("not_flag2", "flag2")
levels(data_decision$FLAG_DIRECT_MKT) <- c("not_flagMKT", "flagMKT")

set.seed(12345)
training_decision <- data_decision$churn %>%
  createDataPartition(p = 0.75, list = F)

train_decision <- data_decision[training_decision,]
test_decision <- data_decision[-training_decision,]

metric <- "ROC"
Ctrl <- trainControl(method = "cv" , number=3, classProbs = TRUE,
summaryFunction = twoClassSummary)
rpartTune <- train(churn~ ., data = train_decision, method = "rpart", tuneLength = 10, trControl = Ctrl
rpartTune
```

```
## CART
##
## 42508 samples
##    15 predictor
##     2 classes: 'not_churner', 'churner'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 28339, 28338, 28339
## Resampling results across tuning parameters:
##
##   cp            ROC        Sens       Spec
##   0.0007071136  0.9531190  0.9128836  0.8797842
##   0.0007778249  0.9529567  0.9129308  0.8792676
##   0.0009899590  0.9528949  0.9168906  0.8744306
##   0.0015085089  0.9515888  0.9054825  0.8842448
##   0.0027813133  0.9506303  0.9028426  0.8813339
##   0.0031270133  0.9484656  0.9010984  0.8784692
##   0.0052562108  0.9404556  0.9154764  0.8563988
##   0.0069454265  0.9109142  0.8961014  0.8562103
##   0.0237825862  0.8847131  0.8879461  0.8439063
##   0.6996652996  0.7311616  0.5987838  0.8635395
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0007071136.
```

Il tuning restituisce in cima un parametro cp grazie al quale risulta possibile massimizzare il risultato in termini di ROC, metrica impostata per questo algoritmo.

```
Vimportance <- varImp(rpartTune)
plot(Vimportance)
```

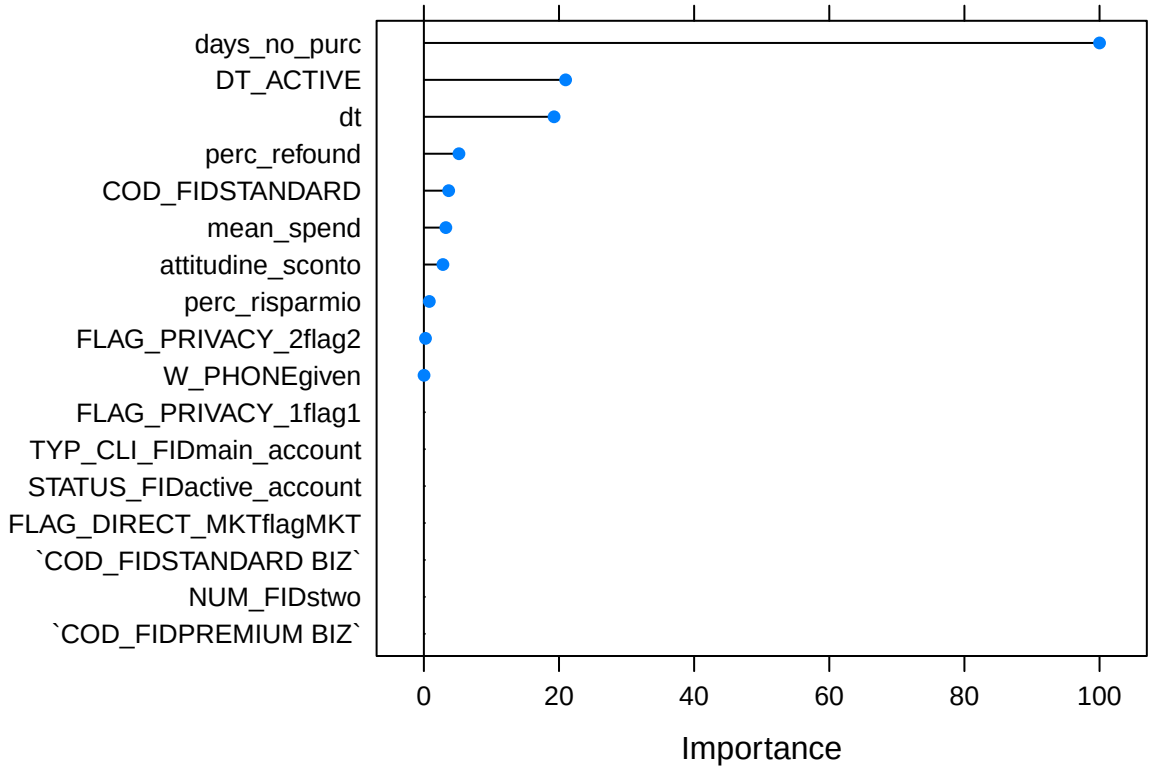


Dal grafico soprastante Ã Â¨ possibile notare il ranking delle variabili ordinate per importanza.

Si effettua di nuovo l'albero decisionale, settando, in questo caso, i parametri che dal tuning precedente risultavano essere migliori.

```
set.seed(12345)
Ctrl_save <- trainControl(method = "cv" , number=3, summaryFunction = twoClassSummary,
classProbs = TRUE, savePredictions = TRUE)
rpartTuneMy <- train(churn ~ ., data = train_decision, method = "rpart", tuneGrid=data.frame(cp=0.00092
```

```
Vimportance_optm <- varImp(rpartTuneMy)
plot(Vimportance_optm)
```



Viene verificato che l'ordine di importanza delle variabili rimane intatto. Si decide quindi di effettuare una feature selection sulla base delle indicazioni ottenute dal Decision tree.

```
selected_training <- data_decision %>%
  dplyr::select(c(churn, days_no_purc, DT_ACTIVE, dt, mean_spend, COD_FID, FLAG_PRIVACY_2, W_PHONE, per

set.seed(12345)
training_selected_training <- selected_training$churn %>%
  createDataPartition(p = 0.75, list = F)

train_selected <- selected_training[training_selected_training,]
test_selected <- selected_training[-training_selected_training,]
```

A questo punto, identificato il set di variabili migliori, si tenta di lanciare una rete neurale. Prima di ciÃ Â², tuttavia, si svolge un'analisi riguardo l'approccio migliore da utilizzare nella stima della stessa. Vengono, a tal proposito, testate una Principal Component Analysis, una Normalizzazione ed una Standardizzazione come metodi di preprocessing. Il metodo dei tre che restituirÃ Â i risultati migliori sarÃ Â quindi inserito nel preprocessing della Rete.

PCA

```
tunegrid <- expand.grid(size=c(1:5), decay = c(0.0002, 0.0003, 0.00001, 0.0001))
nnetFit_defgridDR1 <- train(churn ~ .,
                            method = "nnet",
                            preProcess = 'pca',
                            metric=metric,
                            trControl=Ctrl, tuneGrid=tunegrid,
                            trace = FALSE,
```

```
                            data = train_selected,
                            maxit = 100)
getTrainPerf(nnetFit_defgridDR1)
```

```
##    TrainROC TrainSens TrainSpec method
## 1 0.9704285 0.9183991 0.8886124   nnet
```

```
nnetFit_defgridDR1$bestTune
```

```
##    size decay
## 18    5 1e-04
```

STANDARDIZZAZIONE

```
tunegrid <- expand.grid(size=c(1:5), decay = c(0.0002, 0.0003, 0.00001, 0.0001))
nnetFit_defgridDR2 <- train(churn ~ .,
                            method = "nnet",
                            preProcess = c("center", "scale"),
                            metric=metric,
                            trControl=Ctrl, tuneGrid=tunegrid,
                            trace = FALSE,
                            data = train_selected,
                            maxit = 100)
getTrainPerf(nnetFit_defgridDR2)
```

```
##    TrainROC TrainSens TrainSpec method
## 1 0.9705407 0.9142036  0.890397   nnet
```

```
nnetFit_defgridDR2$bestTune
```

```
##    size decay
## 19    5 2e-04
```

NORMALIZZAZIONE

```
tunegrid <- expand.grid(size=c(1:5), decay = c(0.0002, 0.0003, 0.00001, 0.0001))
nnetFit_defgridDR3 <- train(churn ~ .,
                            method = "nnet",
                            preProcess = c("range"),
                            metric=metric,
                            trControl=Ctrl, tuneGrid=tunegrid,
                            trace = FALSE,
                            data = train_selected,
                            maxit = 100)
getTrainPerf(nnetFit_defgridDR3)
```

```
##   TrainROC TrainSens TrainSpec method
## 1 0.970021 0.9161363 0.8874384   nnet
```

```
nnetFit_defgridDR3$bestTune
```

```
##    size decay
## 15    4 2e-04
```

Il parametro in grado di restituire il livello maggiore di ROC, metrica utilizzata in quest'ambito, risulta essere la Standardizzazione. Si procede quindi nell'implementazione della rete neurale, settendo i parametri emersi dalla Standardizzazione.

```
set.seed(12345)
tunegrid <- expand.grid(size=5, decay = 2e-04)
nnetFit_final <- train(churn ~ .,
                            method = "nnet",
                            preProcess = c("center", "scale"),
                            metric=metric,
                            trControl=Ctrl_save, tuneGrid=tunegrid,
                            trace = FALSE,
                            data = train_selected,
                            maxit = 100)
getTrainPerf(nnetFit_final)
```

```
##    TrainROC TrainSens TrainSpec method
## 1 0.9685811 0.9117522 0.8876734   nnet
```

Anche i risultati della Rete Neurale sembrano essere ottimi.

Si analizzano ora i risultati in termini di Precision, Recall, F Measure ed AIC delle 3 regressioni utilizzate:

```
results_step_logistic <- cbind(caret::F_meas(data = as.factor(predicted_logistic_step), reference = tes
caret::recall(data = as.factor(predicted_logistic_step), reference = test_logistic$churn), AIC(logistic_

results_logistic <- cbind(caret::F_meas(data = as.factor(predicted_logistic), reference = test_logistic$
caret::recall(data = as.factor(predicted_logistic), reference = test_logistic$churn), AIC(mod_logistic)


results_lasso <- cbind(caret::F_meas(data = as.factor(predicted.classes), reference = test_lasso$churn)
caret::recall(data = as.factor(predicted.classes), reference = test_lasso$churn), NA)

results <- rbind(results_logistic, results_step_logistic, results_lasso)

row.names(results) <- c("Logistic" ,"Logistic Stepwise", "LASSO")
colnames(results) <- c("Fmeas","Precision", "Recall", "AIC")

results
```

```
##                      Fmeas Precision    Recall      AIC
## Logistic          0.9010762 0.8794775 0.9237624 20686.56
## Logistic Stepwise 0.9010762 0.8794775 0.9237624 20680.27
## LASSO             0.8943143 0.8421842 0.9533239       NA
```

La migliore tra le 3 sembra essere la logistic Stepwise, ovvero la tradizionale Regressione Logistica, su cui perÃ Â² Ã Â¨ stata svolta un metodo di selezione delle variabili di tipo Stepwise (Forward and Backword). Tra questo modello e la regressione su cui non Ã Â¨ stata effettuata la tecnica precedente, in realtÃ Â , avviene un miglioramento solamente riguardo al Criterio di informazione basato sulla Verosimiglianza, ma non sulle previsioni. Per quanto riguarda, quindi, il futuro utilizzo di un modello od un altro, le considerazioni sulla capacitÃ Â previsiva dei due modelli sono esattamente le stesse. Se perÃ Â², si dovesse prediligere un approccio basato anche sulla semplicitÃ Â , la strada da seguire Ã Â¨ quella tracciata dalla Stepwise Logistic.

A questo punto, i risultati ottenuti sono pienamente soddisfacenti. Si decide, perÃ Â² di provare a testare un modello di tipo Random Forest. Dati gli output precedenti, ci si aspetta una buona performance anche da questo algoritmo, per provare ad individuare e prevedere i possibili churner. Al fine di ottimizzare la stima del modello, viene effettuato, anche in questo caso, un tuning dei parametri. Una volta scovati i parametri migliori per ottenere i risultati ottimi, questi saranno inseriti nella stima del modello.

```
customRF <- list(type = "Classification", library = "randomForest", loop = NULL)
customRF$parameters <- data.frame(parameter = c("mtry", "ntree"),
class = rep("numeric", 2),
label = c("mtry", "ntree"))
customRF$grid <- function(x, y, len = NULL, search = "grid") {}
customRF$fit <- function(x, y, wts, param, lev, last, weights, classProbs, ...) {
randomForest(x, y, mtry = param$mtry, ntree=param$ntree, ...)
}
customRF$predict <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
predict(modelFit, newdata)
customRF$prob <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
predict(modelFit, newdata, type = "prob")
customRF$sort <- function(x) x[order(x[,1]),]
customRF$levels <- function(x) x$classes
set.seed(123)
tunegrid <- expand.grid(.mtry=c(4:9), .ntree=c(100,500))
rpartTuneMyRf <- train(churn ~ ., data = train_decision, method = customRF,
tuneGrid=tunegrid, trControl = Ctrl, metric=metric)
rpartTuneMyRf
```

```
## 42508 samples
##    15 predictor
##     2 classes: 'not_churner', 'churner'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 28339, 28339, 28338
## Resampling results across tuning parameters:
##
##   mtry  ntree  ROC        Sens       Spec
##   4     100    0.9699811  0.9183048  0.8911009
##   4     500    0.9709110  0.9190591  0.8919931
##   5     100    0.9707956  0.9180691  0.8912889
##   5     500    0.9716692  0.9183048  0.8933081
##   6     100    0.9710868  0.9177391  0.8931673
##   6     500    0.9719250  0.9190119  0.8933081
##   7     100    0.9710063  0.9170320  0.8917114
##   7     500    0.9718735  0.9192476  0.8931202
##   8     100    0.9711491  0.9182105  0.8915236
##   8     500    0.9718622  0.9192005  0.8926507
##   9     100    0.9709855  0.9168906  0.8918054
##   9     500    0.9717541  0.9193419  0.8918993
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 6 and ntree = 500.
```

I valori finali per stimare la Random Forest saranno quindi quelli espressi dal tuning dei parametri.

```
set.seed(12345)
tunegrid <- expand.grid(.mtry=8, .ntree=500)
rpartTuneMyRf_ok <- train(churn ~ ., data = train_decision, method = customRF,
tuneGrid=tunegrid, trControl = Ctrl_save, metric=metric)
rpartTuneMyRf_ok
```

```
## 42508 samples
```

```
##    15 predictor
##     2 classes: 'not_churner', 'churner'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 28338, 28339, 28339
## Resampling results:
##
##   ROC        Sens       Spec
##   0.9714853  0.9182105  0.8918058
##
## Tuning parameter 'mtry' was held constant at a value of 8
## Tuning
##  parameter 'ntree' was held constant at a value of 500
```

Stimata la Random Forest viene adottato un criterio unico per confrontare tutti i modelli utilizzati in questa analisi. Il confronto ed il giudizio viene basato sulla metrica AUC, ovvero Area under the curve. Dove la curva Ã Â¨ intesa in questo cosa come la ROC curve.

```
roc_logistic <- roc(observed_classes ,probabilities_logistic)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
roc_logistic_step <- roc(observed_classes, probabilities_logistic_step)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
roc_LASSO <- roc(observed.classes, probabilities)
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(observed.classes, probabilities): Deprecated use
## a matrix as predictor. Unexpected results may be produced, please pass a
## numeric vector.
```

```
## Setting direction: controls < cases
```

```
result_predicted_nnet <- predict(nnetFit_final, test_selected, type = "prob")
roc_nnet <- roc(test_selected$churn, result_predicted_nnet$churn)
```

```
## Setting levels: control = not_churner, case = churner
## Setting direction: controls < cases
```

```
result_predicted_rf <- predict(rpartTuneMyRf_ok, test_decision, type = "prob")
roc_rf <- roc(test_decision$churn, result_predicted_rf$churn)
```

```
## Setting levels: control = not_churner, case = churner
## Setting direction: controls < cases
```

```
data_results <- as.data.frame(rbind(roc_LASSO$auc, roc_logistic$auc, roc_logistic_step$auc, roc_nnet$auc
rownames(data_results) <- c("LASSO AUC", "Logistic AUC", "Step Logistic AUC", "NNET AUC", "Random Forest
data_results
```

```
##                          V1
## LASSO AUC         0.9642990
## Logistic AUC      0.9642255
## Step Logistic AUC 0.9642658
```

```
## NNET AUC          0.9700885
## Random Forest AUC 0.9731467
```

I risultati migliori provengono dall'ultimo dei modelli utilizzati, ovvero la Random Forest, su cui verrÃ svolto un approccio di tipo data driven.

Per concludere, cosÃ¬ come fatto per il primo obiettivo di business, si effettuano dei test con un individuo esempio. Lo scopo rimane quello di mostrare concretamente l'impotanza delle variabili. Si cerca di osservare se e come, al cambiare del valore di una sola delle esplicative risultate piÃ¹ importanti, puÃ² cambiare la classe di appartenenza dell'individuo in questione.

Si utilizza a questo scopo un cliente presente nel test set. In questo caso, per semplicitÃ verrÃ condotto l'esempio utilizzando il modello di regressione logistica step wise.

Analizziamo le variabili, in termini di influenza sul modello

```
summary(logistic_step)
```

```
##
## Call:
## glm(formula = churn ~ dt + days_no_purc + mean_spend + COD_FID +
##     DT_ACTIVE + FLAG_PRIVACY_1 + perc_refound + attitudine_sconto,
##     family = "binomial", data = train_logistic)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5061  -0.3591   0.0000   0.1262   3.5173
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.517e+02  3.134e+00 -48.397  < 2e-16 ***
## dt                  -1.887e-02  5.098e-04 -37.014  < 2e-16 ***
## days_no_purc         8.502e-02  9.396e-04  90.484  < 2e-16 ***
## mean_spend          -5.156e-04  5.530e-05  -9.323  < 2e-16 ***
## COD_FIDPREMIUM BIZ   4.492e-02  8.257e-02   0.544   0.5864
## COD_FIDSTANDARD     -4.854e-01  4.345e-02 -11.172  < 2e-16 ***
## COD_FIDSTANDARD BIZ -1.329e-01  7.998e-02  -1.662   0.0965 .
## DT_ACTIVE            8.353e-03  1.752e-04  47.689  < 2e-16 ***
## FLAG_PRIVACY_11      4.163e-02  3.984e-02   1.045   0.2960
## perc_refound         1.635e-02  2.556e-03   6.398 1.58e-10 ***
## attitudine_sconto    9.962e-03  8.894e-04  11.200  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58928  on 42507  degrees of freedom
## Residual deviance: 20658  on 42497  degrees of freedom
## AIC: 20680
##
## Number of Fisher Scoring iterations: 7
```

Modifichiamo ora il valore di alcune variabili risultate importanti, prendendo come riferimento un cliente giudicato churner.

```
user_test1 <- test_logistic[1,] #cliente churner

user_test1$mean_spend <- 300 #Cambia la spesa media
```

```
user_test1$perc_refound <- 0 #percentuale di refound pari a 0
user_test1$dt <- 150 #giorni tra gli acquisti
user_test1$days_no_purch <- 150 #giorni dall'ultimo acquisto

predict(logistic_step,user_test1[,] , type = "response")
```

```
##        10
## 0.3961912
```

Da questo breve esempio possiamo condurre alcune osservazioni: Il cliente preso come riferimento era chiaramente un churner. La sua spesa media era abbastanza bassa (61.9 euro), perÃ Â² i giorni trascorsi tra i suoi acquisti non erano elevati.

Aumentando a 300 la spesa media degli acquisti, ma aumentando il periodo di passaggio solito tra le sue spese, il cliente Ã Â¨ portato a diventare non churner. Questo risultato Ã Â¨ molto interessante e puÃ Â² scaturire degli eventuali suggerimenti. Sarebbe, ad esempio, utile portare il cliente a spendere di piÃ Â¹, aumentando gli euro spesi mediamente negli acquisti. Per attenuare questa spesa, potremmo lasciare che il cliente aspetti diverso tempo (piÃ Â¹ di quanto sia solito fare attualmente) prima di effettuare un altro acquisto; fermo restando perÃ Â², due fattori chiave: la costanza tra le compere e una nulla percentuale di refound degli articoli (da incentivare).

Si tenta un approccio simile al primo, ma diminuendo la media della spessa cosÃ Â¬ come i giorni tra gli acquisti.

```
user_test2 <- test_logistic[1,] #cliente churner

user_test2$mean_spend <- 50 #Cambia la spesa media
user_test2$perc_refound <- 0 #percentuale di refound pari a 0
user_test2$dt <- 15 #giorni tra gli acquisti
user_test2$days_no_purch <- 15 #giorni dall'ultimo acquisto

predict(logistic_step,user_test2[,] , type = "response")
```

```
##        10
## 0.9050929
```

CiÃ Â² che emerge dai dati Ã Â¨ particolarmente interessante. Prima dell'analisi, era facile credere che portare un cliente a rimanere nell'azienda fosse una questione di "frequenza" degli acquisti, e che fosse quindi necessario incentivare il cliente a comprare spesso, affinchÃ Â¨ fosse costante la sua presenza. Dallo studio effettuato e dall'approccio di tipo data driven, perÃ Â² si evince il contrario. Si nota infatti come aumentando la spesa ma permettendo al cliente di tornare ad acquistare dopo parecchio tempo e mantenendo una percentuale di refound pari a 0, egli non sarÃ Â piÃ Â¹ churner. Al contrario, emerge che un cliente con una spesa media bassa, pur avendo una frequenza agli acquisti molto alta, Ã Â¨ piÃ Â¹ portato ad abbandonare l'azienda, anche se la sua percentuale di refound Ã Â¨ pari a 0.

```
sink("sessionInfo.txt")
sessionInfo()
```

```
## R version 3.6.0 (2019-04-26)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Italian_Italy.1252  LC_CTYPE=Italian_Italy.1252
## [3] LC_MONETARY=Italian_Italy.1252 LC_NUMERIC=C
```

```
## [5] LC_TIME=Italian_Italy.1252
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] randomForest_4.6-14 pROC_1.15.0         MCMCpack_1.4-4
##  [4] coda_0.19-3         arm_1.10-1          lme4_1.1-21
##  [7] MASS_7.3-51.4       car_3.0-3           carData_3.0-2
## [10] e1071_1.7-2         glmnet_2.0-18       foreach_1.4.4
## [13] Matrix_1.2-17       plotROC_2.2.1       rpart.plot_3.0.7
## [16] rpart_4.1-15        pander_0.6.3        caret_6.0-84
## [19] lattice_0.20-38     ROSE_0.0-3          knitr_1.23
## [22] corrplot_0.84       gridExtra_2.3       forcats_0.4.0
## [25] ggplot2_3.2.0       magrittr_1.5        dplyr_0.8.1
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-139       mcmc_0.9-6         lubridate_1.7.4
##  [4] tools_3.6.0        backports_1.1.4    utf8_1.1.4
##  [7] R6_2.4.0           lazyeval_0.2.2     colorspace_1.4-1
## [10] nnet_7.3-12        withr_2.1.2        tidyselect_0.2.5
## [13] curl_3.3           compiler_3.6.0     cli_1.1.0
## [16] quantreg_5.41      SparseM_1.77       labeling_0.3
## [19] scales_1.0.0       stringr_1.4.0      digest_0.6.19
## [22] foreign_0.8-71     minqa_1.2.4        rmarkdown_1.13
## [25] rio_0.5.16         pkgconfig_2.0.2    htmltools_0.3.6
## [28] highr_0.8          rlang_0.3.4        readxl_1.3.1
## [31] generics_0.0.2     ModelMetrics_1.2.2 zip_2.0.3
## [34] Rcpp_1.0.1         munsell_0.5.0      fansi_0.4.0
## [37] abind_1.4-5        stringi_1.4.3      yaml_2.2.0
## [40] plyr_1.8.4         recipes_0.1.5      crayon_1.3.4
## [43] haven_2.1.0        splines_3.6.0      hms_0.4.2
## [46] zeallot_0.1.0      pillar_1.4.1       boot_1.3-22
## [49] reshape2_1.4.3     codetools_0.2-16   stats4_3.6.0
## [52] glue_1.3.1         evaluate_0.14      data.table_1.12.2
## [55] vctrs_0.1.0        nloptr_1.2.1       MatrixModels_0.4-1
## [58] cellranger_1.1.0   gtable_0.3.0       purrr_0.3.2
## [61] assertthat_0.2.1   xfun_0.7           gower_0.2.1
## [64] openxlsx_4.1.0.1   prodlim_2018.04.18 class_7.3-15
## [67] survival_2.44-1.1  timeDate_3043.102  tibble_2.1.3
## [70] iterators_1.0.10   lava_1.6.5         ipred_0.9-9
```

```
sink()
writeLines(capture.output(sessionInfo()), "sessionInfo.txt")
```