STAT5125 Final Project

AUTHOR
Jae Eun Lee, Haoxian Ruan

Objective

1. Data Tidying - Transform messy data from various tables into a tidy format to facilitate further analysis.

2. Data Analysis - Analyze sales distribution across product categories and evaluate the impact of promotions from both product and business perspectives.

3. User Churn Prediction Model - Develop a churn prediction model. This involves explaining the model's principles, describing the input features used, and presenting evaluation metrics.

Dataset

source: [Kaggle: Fashion Ecommerce Indonesia]

Data Description: [Transaction]

created_at: Transaction occurrence time, customer_id: Customer ID, booking_id: Booking ID, session_id: Session ID, payment_method: Payment method, payment_status: Payment status, promo_amount: Promotion discount amount, promo_code: Promotion code, shipment_fee: Shipment fee, shipment_date_limit: Shipment completion deadline, shipment_location_lat: Shipment location latitude, shipment_location_long: Shipment location longitude, total_amount: Total price, item_price: Item price

[Customer]

customer_id: Customer ID, first_name: First name, last_name: Last name, username: Username, email: Email, gender: Gender, birthdate: Birthdate, device_type: Device type, device_id: Device ID, device_version: Device version, home_location_lat: Home location latitude, home_location_long: Home location longitude, home_location: Home location, home_country: Home country, first_join_date: First join date

[Product]

id: ID, gender: Gender, masterCategory: Top-level category, subCategory: Sub-category, articleType: Product type, baseColour: Base color, season: Season, year: Year, usage: Usage, productDisplayName: Product name

[Click Stream] event_name: Event name, event_time: Event occurrence time, event_id: Event ID, traffic_source: Traffic source, product_id: Product ID, quantity: Quantity, item_price: Item price, payment_status: Payment status, search_keywords: Search keywords, promo_code: Promotion code, promo_amount: Promotion discount amount

```
library(tidymodels)
```

```
── Attaching packages ──────────────────────────────────────
tidymodels 1.2.0 ──

✔ broom        1.0.5      ✔ recipes      1.0.10
✔ dials        1.2.1      ✔ rsample      1.2.1
✔ dplyr        1.1.4      ✔ tibble       3.2.1
✔ ggplot2      3.5.0      ✔ tidyr        1.3.1
✔ infer        1.0.7      ✔ tune         1.2.0
✔ modeldata    1.3.0      ✔ workflows    1.1.4
✔ parsnip      1.2.1      ✔ workflowsets 1.1.0
✔ purrr        1.0.2      ✔ yardstick    1.3.1

── Conflicts ─────────────────────────────────────────
tidymodels_conflicts() ──
✖ purrr::discard() masks scales::discard()
✖ dplyr::filter()  masks stats::filter()
✖ dplyr::lag()     masks stats::lag()
✖ recipes::step()  masks stats::step()
• Search for functions across packages at
https://www.tidymodels.org/find/
```

```
library(dplyr)
library(lubridate)
```

```

Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ────────────────────────
```

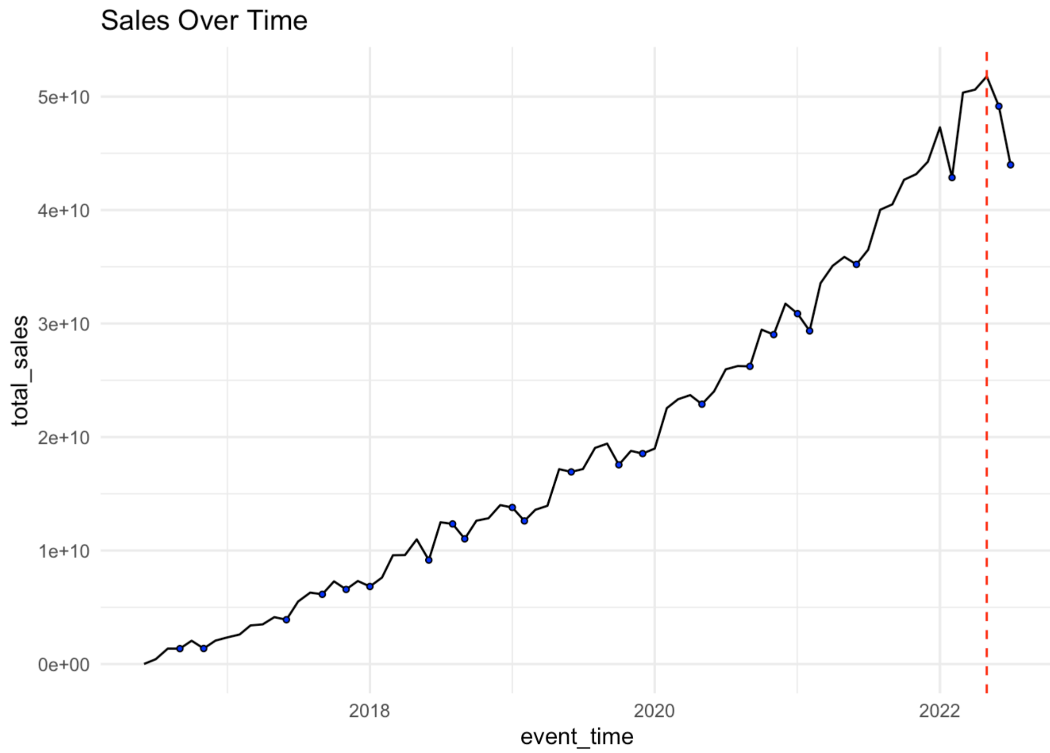
```
tidyverse 2.0.0 ——
✔ forcats 1.0.0     ✔ stringr 1.5.1
✔ readr   2.1.5

── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
✖ readr::col_factor() masks scales::col_factor()
✖ purrr::discard()    masks scales::discard()
✖ dplyr::filter()     masks stats::filter()
✖ stringr::fixed()    masks recipes::fixed()
✖ dplyr::lag()        masks stats::lag()
✖ readr::spec()       masks yardstick::spec()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>)
to force all conflicts to become errors
```

```
tidymodels_prefer()
theme_set(theme_bw())
```

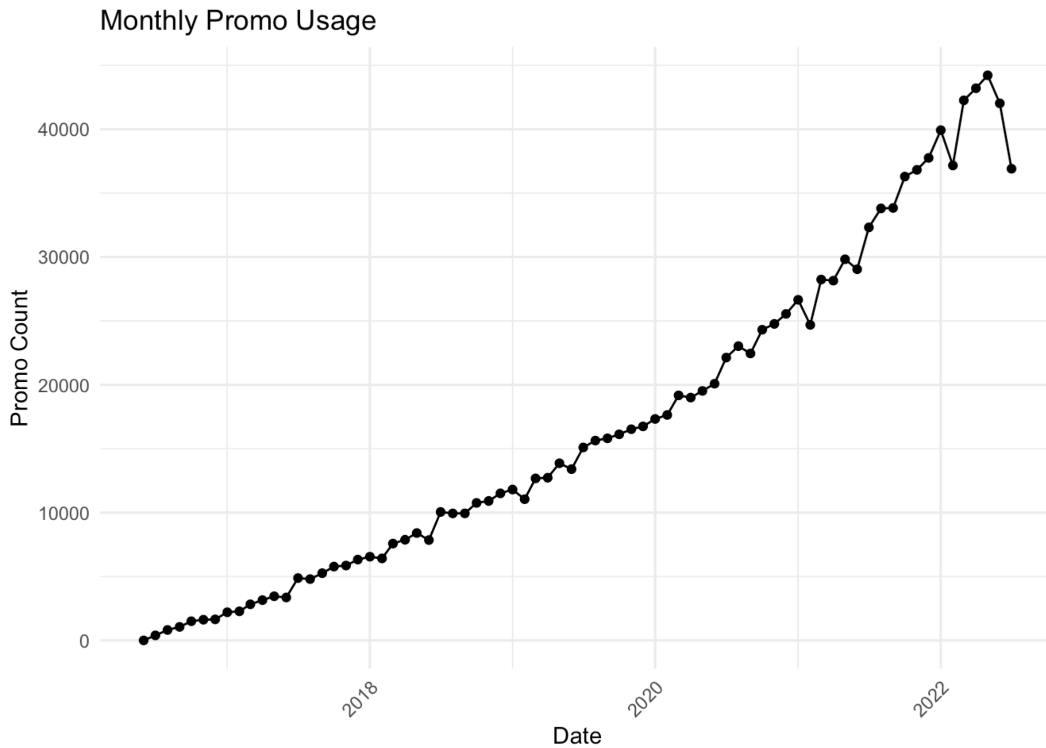
```
click_stream <- read.csv("data/click_stream_new.csv")
transaction <- read.csv("data/transaction_new.csv")
customer <- read.csv("data/customer.csv")
product <- read.csv("data/product.csv")
```

Visualization

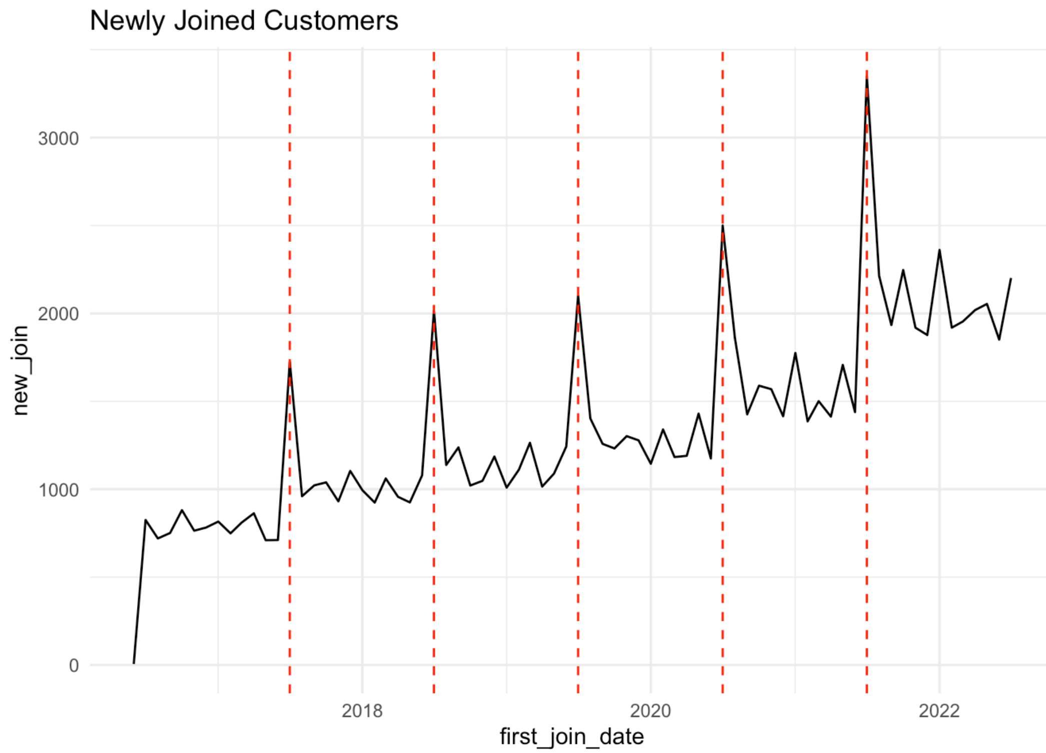
```
trans <- transaction |>
  mutate(event_time = ym(substr(created_at, 1, 7)))
#  select(-created_at)

monthly_sales <- trans |>
  select(event_time, total_amount) |>
  group_by(event_time) |>
  summarize(total_sales = sum(total_amount))

drop_down_points <- monthly_sales |>
  filter(total_sales < dplyr::lag(total_sales))

ggplot(monthly_sales, aes(x = event_time, y = total_sales)) +
  geom_line() +
  geom_point(data = drop_down_points, aes(y = total_sales), sha
  geom_vline(xintercept = as.numeric(as.Date("2022-05-01")), co
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Sales Over Time") +
  theme_minimal()
```




```r
promo_used <- trans |>
  mutate(promo = ifelse(promo_code != 0, 1, 0)) |>
  group_by(event_time) |>
  summarize(count_1_in_promo = sum(promo == 1))

ggplot(promo_used, aes(x = event_time, y = count_1_in_promo)) +
  geom_line() +
  geom_point() +
  labs(title = "Monthly Promo Usage",
       x = "Date",
       y = "Promo Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
new_cust <- customer |>
  mutate(first_join_date = ym(substr(first_join_date, 1, 7))) |>
  group_by(first_join_date) |>
  summarize(new_join = n())

ggplot(new_cust, aes(x = first_join_date, y = new_join)) +
  geom_line() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_vline(xintercept = c(as.numeric(as.Date("2017-07-01")),
                            as.numeric(as.Date("2018-07-01")),
                            as.numeric(as.Date("2019-07-01")),
                            as.numeric(as.Date("2020-07-01")),
                            as.numeric(as.Date("2021-07-01"))),
             color = "red", linetype = "dashed") +
  labs(title = "Newly Joined Customers") +
  theme_minimal()
```



Preparing dataset

Training set

```
click_clean <- click_stream |>
  mutate(time = substr(event_time, 1, 4)) |>
  filter(event_name == 'BOOKING',
         payment_status == "Success",
         time == 2021) |> # extract the data in year 2021
  select(session_id, event_time, traffic_source, payment_status
  mutate(event_time = substr(event_time, 1, 10))

head(click_clean)
```

```
                            session_id event_time
traffic_source payment_status
1 7998797c-9337-4952-945b-713291cb561e 2021-01-01
MOBILE         Success
2 4dc77d60-0b99-4bf3-9317-3b2f637fe331 2021-01-18
MOBILE         Success
3 7ca647bc-bf0d-4b8c-b8d7-7c21116fc006 2021-01-26
MOBILE         Success
4 bd660101-7a41-4d98-a364-040ebf9d1134 2021-02-11
MOBILE         Success
```

```
5 18da9200-2cb3-4dc9-b883-229cab577e49 2021-02-19
WEB        Success
6 cdd6422c-b1a4-4fce-80df-4d7e679da553 2021-02-27
MOBILE        Success
```

```r
transaction_clean <- transaction |>
  mutate(time = substr(created_at, 1, 4)) |>
  filter(time == 2021,
         payment_status == "Success") |>
  select(-c(shipment_location_lat, shipment_location_long, book
  mutate(created_at = substr(created_at, 1, 10),
         shipment_date_limit = substr(shipment_date_limit, 1, 1(

head(transaction_clean)
```

```
  created_at customer_id                           session_id
payment_method
1 2021-01-08        4774 8f3c25e3-6529-469f-9b47-2acac609d93a
Credit Card
2 2021-02-24        4774 57b6d108-9ec4-48ef-8d66-cdd17c4942ba
Gopay
3 2021-04-12        4774 e8665111-3196-4edf-9a30-d1f2a4945b41
Credit Card
4 2021-05-29        4774 a4ad7fb7-6a09-450b-aa8d-97e28c0cda20
Credit Card
5 2021-07-15        4774 b0ab036c-5c29-417b-9c2e-1a41d872028a
LinkAja
6 2021-08-31        4774 b2fb0889-6e08-4bc5-b6ba-2ac71d1b3145
Credit Card
  payment_status promo_amount promo_code shipment_fee
shipment_date_limit
1        Success            0                       0
2021-01-11
2        Success         3855    BUYMORE        10000
2021-02-28
3        Success            0                       0
2021-04-14
4        Success            0                   10000
2021-06-03
5        Success         6325     AZ2022        10000
2021-07-17
6        Success            0                   10000
2021-09-05
  total_amount product_id quantity item_price time
1       174419      31302        1     174419 2021
2       213809      11599        1     207664 2021
3        93400      39204        1      93400 2021
```

```
4       164335       6572        1     154335 2021
5        75865      18001        1      72190 2021
6      2091956      10460        4     398497 2021
```

Merge Click and Trans

```r
# merge the transaction data with the click_stream data
df <- transaction_clean |>
  merge(click_clean, by = "session_id", all.x = TRUE) |>
  select(-(payment_status.y)) |>
  rename(payment_status = payment_status.x)

head(df)
```

```
                            session_id created_at customer_id
payment_method
1 00008e68-a4d4-4b5d-ab7a-9fbd44f7b7fd 2021-03-10        1681
Credit Card
2 0000b1a6-8dca-4fb3-b1db-ccde1791a330 2021-02-10        8544
Gopay
3 0000da55-698f-4d22-8b44-3ca192f2f961 2021-11-15       81829
Gopay
4 0000f2fc-1875-4c88-9398-5b7386b14ca6 2021-08-01       93614
Gopay
5 000130b9-07d1-4374-99b8-89d769c46c9f 2021-10-21       58359
Debit Card
6 00016815-cf29-45fa-8974-17c6796aab29 2021-04-30       63595
OVO
  payment_status promo_amount  promo_code shipment_fee
shipment_date_limit
1        Success            0                    10000
2021-03-14
2        Success            0                    25000
2021-02-12
3        Success            0                    10000
2021-11-16
4        Success         2365 WEEKENDSERU            0
2021-08-06
5        Success            0                    10000
2021-10-25
6        Success            0                        0
2021-05-03
  total_amount product_id quantity item_price time event_time
traffic_source
1       304725      57008        1     294725 2021 2021-03-10
MOBILE
2       172017      19353        1     147017 2021 2021-02-10
```

```
MOBILE
3       227518      13304        1     217518 2021 2021-11-15
MOBILE
4       522429      50025        1     524794 2021 2021-08-01
MOBILE
5       510384      28603        4     125096 2021 2021-10-21
MOBILE
6       164229       2219        1     164229 2021 2021-04-30
MOBILE
```

```
# drop the rows with NA values
df <- df |>
  na.omit()

head(df)
```

```
                            session_id created_at customer_id
payment_method
1 00008e68-a4d4-4b5d-ab7a-9fbd44f7b7fd 2021-03-10        1681
Credit Card
2 0000b1a6-8dca-4fb3-b1db-ccde1791a330 2021-02-10        8544
Gopay
3 0000da55-698f-4d22-8b44-3ca192f2f961 2021-11-15       81829
Gopay
4 0000f2fc-1875-4c88-9398-5b7386b14ca6 2021-08-01       93614
Gopay
5 000130b9-07d1-4374-99b8-89d769c46c9f 2021-10-21       58359
Debit Card
6 00016815-cf29-45fa-8974-17c6796aab29 2021-04-30       63595
OVO
  payment_status promo_amount  promo_code shipment_fee
shipment_date_limit
1        Success            0                    10000
2021-03-14
2        Success            0                    25000
2021-02-12
3        Success            0                    10000
2021-11-16
4        Success         2365 WEEKENDSERU            0
2021-08-06
5        Success            0                    10000
2021-10-25
6        Success            0                        0
2021-05-03
  total_amount product_id quantity item_price time event_time
traffic_source
1       304725      57008        1     294725 2021 2021-03-10
```

```
MOBILE
2        172017      19353         1     147017 2021 2021-02-10
MOBILE
3        227518      13304         1     217518 2021 2021-11-15
MOBILE
4        522429      50025         1     524794 2021 2021-08-01
MOBILE
5        510384      28603         4     125096 2021 2021-10-21
MOBILE
6        164229       2219         1     164229 2021 2021-04-30
MOBILE
```

We will convert the date-related columns into date datatype, exclude the total_amount column due to collinearity issues, and drop the promo_code column as certain promo codes may only apply to data from 2022 and not generalize to the test set. Additionally, we will initially label all customers as churned (TRUE) and then update the churn label to FALSE for rows satisfying the non-churn condition. Finally, we will create a new shipment_eta column to capture estimated shipment times.

```
df <- df |>
  mutate(created_at = as.Date(created_at)) |>
  mutate(shipment_date_limit = as.Date(shipment_date_limit)) |>
  mutate(event_time = as.Date(event_time)) |>
  arrange(event_time, session_id) |>
  mutate(churn = TRUE) |>
  relocate(churn, .before = "created_at") |>
  relocate(event_time, .before = "created_at") |>
  relocate(customer_id, .before = "created_at") |>
  mutate(shipment_eta = as.numeric(shipment_date_limit - created
  select(-c(created_at, payment_status, promo_code, shipment_da

head(df)
```

```
                            session_id churn event_time
customer_id
1 00ec7115-e23d-4c7d-8253-994a2522c74b  TRUE 2021-01-01
60111
2 02eea86a-33bc-4b09-bc5f-d6ad4a4b8835  TRUE 2021-01-01
73018
3 0303a291-c994-42f5-aae6-db85ce4da1b6  TRUE 2021-01-01
10474
4 040c9084-a9b2-48b8-9815-2d53445b1c5e  TRUE 2021-01-01
16482
5 042c21ea-e220-4940-82a4-8297e62f6444  TRUE 2021-01-01
13832
6 04baa42f-484d-40b6-adbb-f62fca15754d  TRUE 2021-01-01
```

```
23618
  payment_method promo_amount shipment_fee product_id quantity
item_price
1          Gopay            0        50000      57300        1
161945
2            OVO         5133        10000      17276        1
292885
3    Credit Card         6654        10000      47123        1
357107
4     Debit Card            0            0      39940        1
322259
5    Credit Card            0        10000      19407        1
288794
6    Credit Card         3700            0      19125        1
175924
  traffic_source shipment_eta
1         MOBILE            1
2         MOBILE            2
3            WEB            3
4         MOBILE            5
5         MOBILE            3
6         MOBILE            6
```

Label the Target Variable

The function “f1” is to get all the records of the customer_id.

```
f1 <- function(id) {
  record <- df |>
    filter(customer_id == id)
  return(record)
}
```

```
# create a tibble and convert each the records of the customer_i
table <- tibble(customer = unique(df$customer_id)) |>
  rowwise() |>
  mutate(output = list(f1(customer))) |>
  ungroup()

head(table)
```

```
# A tibble: 6 × 2
  customer output
     <int> <list>
1    60111 <df [55 × 12]>
```

```
2     73018 <df [16 × 12]>
3     10474 <df [28 × 12]>
4     16482 <df [4 × 12]>
5     13832 <df [7 × 12]>
6     23618 <df [61 × 12]>
```

```
# unnest: to verify whether it returns to the original tibble
table |>
  unnest(cols = output) |>
  head()
```

```
# A tibble: 6 × 13
  customer session_id   churn event_time customer_id
payment_method promo_amount
     <int> <chr>        <lgl> <date>           <int> <chr>
<int>
1    60111 00ec7115-e2… TRUE  2021-01-01       60111 Gopay
0
2    60111 61f99687-29… TRUE  2021-01-12       60111 Credit
Card               0
3    60111 df65d934-d1… TRUE  2021-01-23       60111 Gopay
0
4    60111 25cd70af-45… TRUE  2021-02-03       60111 Gopay
0
5    60111 6ff5636d-cd… TRUE  2021-02-14       60111 LinkAja
0
6    60111 42e10ce3-fa… TRUE  2021-02-25       60111 Gopay
0
# ℹ 6 more variables: shipment_fee <int>, product_id <int>,
quantity <int>,
#   item_price <int>, traffic_source <chr>, shipment_eta <dbl>
```

It confirms that the number of rows becomes the same as the 'df' dataset before creating the tibble.

Here, "churn" is defined as customers who have not made any transactions within 30 days after their last transaction. The function "label" is to label churn in each records in each customer_id. As the definition of churn, if the customer did not make another transaction in 30 days, he/she is labelled as a churn. It's important to note that the session_id should be different, as multiple items purchased within the same transaction will have the same session_id.

```
label <- function(tibble_in) {
  n <- nrow(tibble_in)
  if(n>1){
```

```
    for (i in 1:(n-1)) {
      date_1 <- tibble_in[i, "event_time"]
      s_1 <- tibble_in[i, "session_id"]
      for (j in (i+1):n){
        date_2 <- tibble_in[j, "event_time"]
        s_2 <- tibble_in[j, "session_id"]
        if(as.logical(s_1!=s_2)&(as.numeric(date_2-date_1)<30))·
          tibble_in[i, "churn"] = FALSE
          next
        }
      }
    }
    }
  return(tibble_in)
  }
```

We optimized churn labeling in a large dataset by shifting from an inefficient O(n^2) approach to focusing solely on records with the same customer_id, achieving a time complexity of O(m^2), where m < 200. This significantly improved labeling speed for datasets with over 300,000 rows. Additionally, while initially finding the course content focused on tidyverse unengaging, we later recognized its value in developing crucial problem-solving skills during the project.

[Comparing the time complexities]

Direct Labeling on the entire dataset: O(n^2), where $n > 3 \times 10^5$

Using Rowwise Labeling: O(m^2), where $m < 2 \times 10^2$

Now, apply the function to the tibble, and unnest it.

```
table_1 <- table |>
  mutate(new_output = map(output, label)) |>
  unnest(cols = new_output)

head(table_1)
```

```
# A tibble: 6 × 14
  customer output          session_id churn event_time
customer_id payment_method
     <int> <list>          <chr>      <lgl> <date>
<int> <chr>
1    60111 <df [55 × 12]> 00ec7115-… FALSE 2021-01-01
60111 Gopay
2    60111 <df [55 × 12]> 61f99687-… FALSE 2021-01-12
60111 Credit Card
```

```
3    60111 <df [55 × 12]> df65d934-… FALSE 2021-01-23
60111 Gopay
4    60111 <df [55 × 12]> 25cd70af-… FALSE 2021-02-03
60111 Gopay
5    60111 <df [55 × 12]> 6ff5636d-… FALSE 2021-02-14
60111 LinkAja
6    60111 <df [55 × 12]> 42e10ce3-… FALSE 2021-02-25
60111 Gopay
# ℹ 7 more variables: promo_amount <int>, shipment_fee <int>,
product_id <int>,
#   quantity <int>, item_price <int>, traffic_source <chr>,
shipment_eta <dbl>
```

```
# the proportion of target
table(table_1$churn)
```

```
 FALSE   TRUE
203770 157566
```

```
# drop irrelevant columns.
table_2 <- table_1 |>
  select(-c(output, customer))

head(table_2)
```

```
# A tibble: 6 × 12
  session_id            churn event_time customer_id
payment_method promo_amount
  <chr>                 <lgl> <date>           <int> <chr>
<int>
1 00ec7115-e23d-4c7d-8… FALSE 2021-01-01       60111 Gopay
0
2 61f99687-2971-454f-a… FALSE 2021-01-12       60111 Credit
Card                0
3 df65d934-d1a3-4a04-9… FALSE 2021-01-23       60111 Gopay
0
4 25cd70af-459a-4624-8… FALSE 2021-02-03       60111 Gopay
0
5 6ff5636d-cd5d-40f9-9… FALSE 2021-02-14       60111 LinkAja
0
6 42e10ce3-fa16-4e92-a… FALSE 2021-02-25       60111 Gopay
0
# ℹ 6 more variables: shipment_fee <int>, product_id <int>,
quantity <int>,
#   item_price <int>, traffic_source <chr>, shipment_eta <dbl>
```

For the ‘product’ data, select the relevant columns and clean it for further analysis.

```
product_select <- product |>
  select(c(id, masterCategory, season, year, usage, productDisp
  filter(masterCategory != "") |>
  filter(!is.na(usage)) |>
  filter(usage != "") |>
  filter(season != "") |>
  mutate(brand = str_extract(productDisplayName, "\\w+")) |>
  select(-c(productDisplayName))

head(product_select)
```

```
     id masterCategory season year  usage      brand
1 15970        Apparel   Fall 2011 Casual     Turtle
2 39386        Apparel Summer 2012 Casual      Peter
3 59263    Accessories Winter 2016 Casual      Titan
4 21379        Apparel   Fall 2011 Casual Manchester
5 53759        Apparel Summer 2012 Casual       Puma
6  1855        Apparel Summer 2011 Casual   Inkfruit
```

For customer data, we will also select the relevant columns and clean it for further analysis.

```
customer_clean <- customer |>
  select(customer_id, gender, birthdate, device_version, home_lo
  mutate(birthdate = as.Date(birthdate),
         device_version = str_extract(device_version, "\\w+"))

head(customer_clean)
```

```
  customer_id gender  birthdate device_version
home_location
1        2870      F 1996-06-14         iPhone      Sumatera
Barat
2        8193      F 1993-08-16        Android         Jakarta
Raya
3        7279      M 1989-01-23           iPad Nusa Tenggara
Barat
4       88813      M 1991-01-05           iPad     Kalimantan
Timur
5       82542      M 2000-07-15         iPhone   Kalimantan
Selatan
6        5440      F 1989-01-09        Android         Jakarta
Raya
  first_join_date
```

```
1      2019-07-21
2      2017-07-16
3      2020-08-23
4      2021-10-03
5      2021-04-11
6      2021-05-30
```

Let's merge the combined transaction and click_stream data with the customer and product data tables.

```
final <- table_2 |>
  mutate(product_id = as.character(product_id)) |>
  left_join(product_select, by = c("product_id" = "id")) |>
  left_join(customer_clean, by = c("customer_id" = "customer_id"
  mutate(first_join_date = as.Date(first_join_date),
         age = as.integer(as.numeric(event_time-birthdate)/365)
         member_duration = as.numeric(event_time-first_join_date
  select(-c(birthdate, first_join_date))

head(final)
```

```
# A tibble: 6 × 22
  session_id            churn event_time customer_id
payment_method promo_amount
  <chr>                 <lgl> <date>           <int> <chr>
<int>
1 00ec7115-e23d-4c7d-8… FALSE 2021-01-01       60111 Gopay
0
2 61f99687-2971-454f-a… FALSE 2021-01-12       60111 Credit
Card                  0
3 df65d934-d1a3-4a04-9… FALSE 2021-01-23       60111 Gopay
0
4 25cd70af-459a-4624-8… FALSE 2021-02-03       60111 Gopay
0
5 6ff5636d-cd5d-40f9-9… FALSE 2021-02-14       60111 LinkAja
0
6 42e10ce3-fa16-4e92-a… FALSE 2021-02-25       60111 Gopay
0
# ℹ 16 more variables: shipment_fee <int>, product_id <chr>,
quantity <int>,
#   item_price <int>, traffic_source <chr>, shipment_eta <dbl>,
#   masterCategory <chr>, season <chr>, year <int>, usage
<chr>, brand <chr>,
#   gender <chr>, device_version <chr>, home_location <chr>,
age <int>,
#   member_duration <dbl>
```

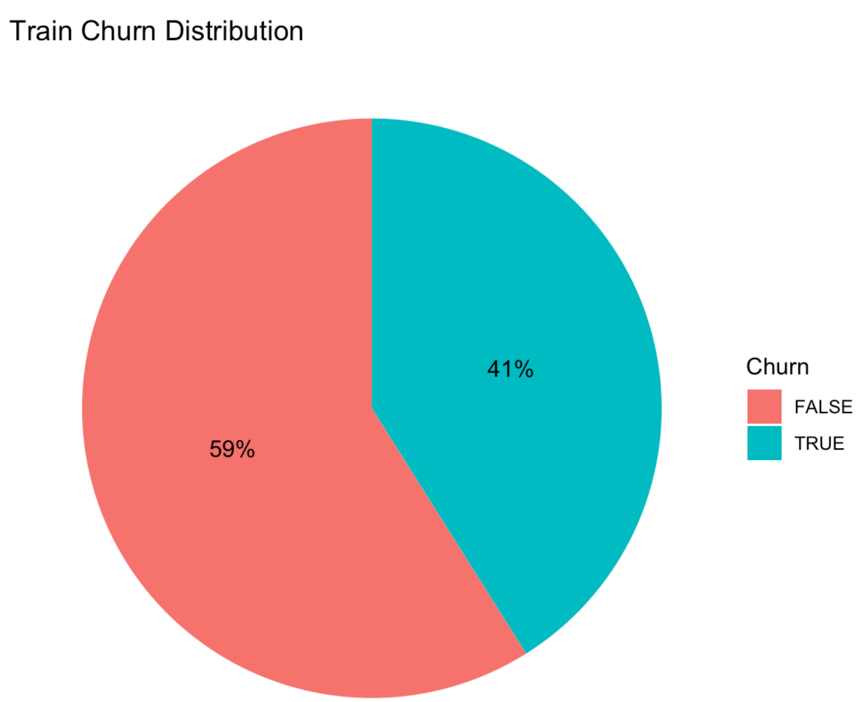
Next, we will convert relevant variables to factor variables and drop irrelevant columns. Additionally, we will exclude data from the last month to avoid an inaccurate churn label. For example, if a customer made a transaction on December 25th, 2021, and their next transaction was on January 1st, 2022, our labeling process would incorrectly mark them as churned since we only have data for 2021. However, they did not actually churn. By removing the last month's data, we can prevent this mislabeling.

```
train_set <- final |>
  na.omit() |>
  mutate(usage = as.factor(usage),
         brand = as.factor(brand),
         gender = as.factor(gender),
         payment_method = as.factor(payment_method),
         traffic_source = as.factor(traffic_source),
         masterCategory = as.factor(masterCategory),
         season = as.factor(season),
         device_version = as.factor(device_version),
         home_location = as.factor(home_location)) |>
  select(-c(session_id, customer_id, product_id)) |>
  mutate(product_year = 2021-year) |>
  relocate(product_year, .after=year) |>
  filter(event_time < as.numeric(as.Date("2021-12-01"))) |>
  select(-event_time, -year, -brand, -device_version) |>
  mutate(churn = as.factor(churn),
         home_location = str_extract(home_location, "\\S+"))

head(train_set)
```

```
# A tibble: 6 × 16
  churn payment_method promo_amount shipment_fee quantity
item_price
  <fct> <fct>                 <int>        <int>    <int>
<int>
1 FALSE Gopay                     0        50000        1
161945
2 FALSE Credit Card               0            0        1
244964
3 FALSE Gopay                     0            0        1
376160
4 FALSE Gopay                     0            0        1
307578
5 FALSE LinkAja                   0            0       18
146764
6 FALSE Gopay                     0        15000        2
386312
# ℹ 10 more variables: traffic_source <fct>, shipment_eta
```

```
<dbl>,
#   masterCategory <fct>, season <fct>, product_year <dbl>,
usage <fct>,
#   gender <fct>, home_location <chr>, age <int>,
member_duration <dbl>
```

```
library(ggplot2)
churn_df <- as.data.frame(table(train_set$churn)) |>
  mutate(Churn = Var1,
         Frequency = Freq) |>
  select(-Var1, -Freq) |>
  mutate(Percentage = Frequency / sum(Frequency) * 100)

ggplot(churn_df, aes(x = "", y = Frequency, fill = Churn)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") +
  labs(title = "Train Churn Distribution") +
  theme_void() +
  geom_text(aes(label = paste0(round(Percentage), "%")), positi
```



This pie chart shows the balanced target ratio in training set.

Test set

For the test set, we perform the same procedures in data cleaning as those applied to the training set.

```r
library(tidyverse)
click_stream <- read.csv("data/click_stream_new.csv")
transaction <- read.csv("data/transaction_new.csv")
```

```r
# clean the click_stream data
click_clean <- click_stream |>
  mutate(time = substr(event_time, 1, 4)) |>
  filter(time == 2022,
         event_name == 'BOOKING',
         payment_status == "Success") |>
  mutate(time = substr(event_time, 1, 10),
         time = as.Date(time)) |>
  filter(time < as.Date("2022-03-01")) |>
  select(session_id, event_time, traffic_source, payment_status
  mutate(event_time = substr(event_time, 1, 10))

head(click_clean)
```

```
                            session_id event_time
traffic_source payment_status
1 78ffedab-febf-4b5b-9362-044ac40afa68 2022-01-05
WEB         Success
2 15965038-20b9-4484-b848-7eeca352b19a 2022-01-13
MOBILE         Success
3 7b71b1c4-e42e-4708-9e1f-31bcbb4cd60f 2022-01-21
MOBILE         Success
4 f584fd13-5ce7-4f1a-862e-2227f4fec0c5 2022-01-29
MOBILE         Success
5 69ac3214-b0d5-44a0-aced-d2123f894029 2022-02-06
MOBILE         Success
6 85c548c7-c16d-45a8-a08d-ee338a8ef26b 2022-02-14
MOBILE         Success
```

```r
# clean the transaction data
transaction_clean <- transaction |>
  mutate(time = substr(created_at, 1, 4)) |>
  filter(time == 2022,
         payment_status == "Success") |>
  mutate(time = substr(created_at, 1, 10),
         time = as.Date(time)) |>
  filter(time < as.Date("2022-03-01")) |>
  select(created_at, customer_id, session_id, payment_method, pa
  mutate(created_at = substr(created_at, 1, 10),
         shipment_date_limit = substr(shipment_date_limit, 1, 1

head(transaction_clean)
```

```
  created_at customer_id                           session_id
payment_method
1 2022-01-19        4774 5d3d5de2-3a5a-42e9-8f8c-512346a4c031
Credit Card
2 2022-01-09       58191 6240caad-6a41-4cf5-bcff-90df3090fabe
OVO
3 2022-01-24       58191 ef1e0fe1-ad41-4d19-976e-cc12a7729902
Gopay
4 2022-02-08       58191 33a331bc-5ee1-4c3f-812c-e3fb04237f1d
OVO
5 2022-02-23       58191 043c5f7b-3502-49f0-a4bd-18cd3bd2ca69
OVO
6 2022-01-09       76966 c7ea62e3-bb18-485d-b2f1-ad942f9075bd
Debit Card
  payment_status promo_amount  promo_code shipment_fee
shipment_date_limit
1        Success            0                     5000
2022-01-21
2        Success         4063 WEEKENDSERU        10000
2022-01-15
3        Success            0                    10000
2022-01-27
4        Success            0                        0
2022-02-13
5        Success            0                    10000
2022-03-01
6        Success            0                    10000
2022-01-11
  total_amount product_id quantity item_price
1      1078709      33212        3     357903
2       369557      16744        1     363620
3       109179      18231        1      99179
4       322802      44497        1     322802
5       185820      33513        1     175820
6       923940      33372        4     228485
```

```
# merge
df <- transaction_clean |>
  merge(click_clean, by = "session_id", all.x = TRUE) |>
  select(-(payment_status.y)) |>
  rename(payment_status = payment_status.x) |>
  na.omit() |>
  mutate(created_at = as.Date(created_at),
         shipment_date_limit = as.Date(shipment_date_limit)) |>
  mutate(event_time = as.Date(event_time),
         shipment_eta = as.numeric(shipment_date_limit - createc
  arrange(event_time, session_id) |>
```

```r
  mutate(churn = TRUE) |>
  relocate(churn, event_time, customer_id, .before = "created_at
  select(-c(created_at, payment_status, promo_code, shipment_da

head(df)
```

```
                            session_id churn event_time
customer_id
1 00098329-5a72-48b5-9ec2-42b914c0529f  TRUE 2022-01-01
10161
2 00098329-5a72-48b5-9ec2-42b914c0529f  TRUE 2022-01-01
10161
3 0020271e-e1b9-4656-8731-dbed78ee31e6  TRUE 2022-01-01
1073
4 00a772a5-221c-4874-89c5-f77ed51bd04d  TRUE 2022-01-01
49907
5 015a5071-538d-44a5-9fc5-15c34b18c382  TRUE 2022-01-01
18212
6 01bace48-ce78-4700-bc0a-c7ee523f6474  TRUE 2022-01-01
66360
  payment_method promo_amount shipment_fee product_id quantity
item_price
1        LinkAja         4105        15000      16765        1
544791
2        LinkAja         4105        15000      19200        1
310626
3            OVO            0            0      34556        1
170557
4     Debit Card            0            0      31142        1
76862
5            OVO            0        10000      49600        1
257421
6          Gopay         2832        10000      45822        1
180511
  traffic_source shipment_eta
1         MOBILE            5
2         MOBILE            5
3         MOBILE            4
4         MOBILE            2
5         MOBILE            4
6         MOBILE            2
```

```r
table <- tibble(customer = unique(df$customer_id)) |>
  rowwise() |>
  mutate(output = list(f1(customer))) |>
  ungroup()
```

```
head(table)
```

```
# A tibble: 6 × 2
  customer output
     <int> <list>
1    10161 <df [2 × 12]>
2     1073 <df [2 × 12]>
3    49907 <df [11 × 12]>
4    18212 <df [3 × 12]>
5    66360 <df [23 × 12]>
6    29934 <df [3 × 12]>
```

```
table |>
  unnest(cols = output) |>
  head()
```

```
# A tibble: 6 × 13
  customer session_id   churn event_time customer_id
payment_method promo_amount
     <int> <chr>        <lgl> <date>           <int> <chr>
<int>
1    10161 00098329-5a… TRUE  2022-01-01       10161 LinkAja
4105
2    10161 00098329-5a… TRUE  2022-01-01       10161 LinkAja
4105
3     1073 0020271e-e1… TRUE  2022-01-01        1073 OVO
0
4     1073 c2a4dee4-d9… TRUE  2022-02-22        1073 Debit Card
0
5    49907 00a772a5-22… TRUE  2022-01-01       49907 Debit Card
0
6    49907 c9297c63-f3… TRUE  2022-01-13       49907 Debit Card
5332
# ℹ 6 more variables: shipment_fee <int>, product_id <int>,
quantity <int>,
#   item_price <int>, traffic_source <chr>, shipment_eta <dbl>
```

```
table_1 <- table |>
  mutate(new_output = map(output, label)) |>
  unnest(cols = new_output)

head(table_1)
```

```
# A tibble: 6 × 14
  customer output         session_id churn event_time
customer_id payment_method
```

```
     <int> <list>         <chr>      <lgl> <date>
<int> <chr>
1    10161 <df [2 × 12]>  00098329-… TRUE  2022-01-01
10161 LinkAja
2    10161 <df [2 × 12]>  00098329-… TRUE  2022-01-01
10161 LinkAja
3     1073 <df [2 × 12]>  0020271e-… TRUE  2022-01-01
1073 OVO
4     1073 <df [2 × 12]>  c2a4dee4-… TRUE  2022-02-22
1073 Debit Card
5    49907 <df [11 × 12]> 00a772a5-… FALSE 2022-01-01
49907 Debit Card
6    49907 <df [11 × 12]> c9297c63-… FALSE 2022-01-13
49907 Debit Card
# ℹ 7 more variables: promo_amount <int>, shipment_fee <int>,
product_id <int>,
#   quantity <int>, item_price <int>, traffic_source <chr>,
shipment_eta <dbl>
```

```
table(table_1$churn)
```

```

FALSE  TRUE
35085 38571
```

```
table_2 <- table_1 |>
  select(-c(output, customer)) |>
  filter(event_time < as.Date("2022-02-01"))

head(table_2)
```

```
# A tibble: 6 × 12
  session_id             churn event_time customer_id
payment_method promo_amount
  <chr>                  <lgl> <date>           <int> <chr>
<int>
1 00098329-5a72-48b5-9… TRUE  2022-01-01       10161 LinkAja
4105
2 00098329-5a72-48b5-9… TRUE  2022-01-01       10161 LinkAja
4105
3 0020271e-e1b9-4656-8… TRUE  2022-01-01        1073 OVO
0
4 00a772a5-221c-4874-8… FALSE 2022-01-01       49907 Debit Card
0
5 c9297c63-f315-49e7-9… FALSE 2022-01-13       49907 Debit Card
5332
6 c9297c63-f315-49e7-9… FALSE 2022-01-13       49907 Debit Card
```

```
5332
# ℹ 6 more variables: shipment_fee <int>, product_id <int>,
quantity <int>,
#   item_price <int>, traffic_source <chr>, shipment_eta <dbl>
```

```
final <- table_2 |>
  mutate(product_id = as.character(product_id)) |>
  left_join(product_select, by = c("product_id" = "id")) |>
  left_join(customer_clean, by = c("customer_id" = "customer_id"
  mutate(first_join_date = as.Date(first_join_date),
         age = as.integer(as.numeric(event_time-birthdate)/365),
         member_duration = as.numeric(event_time-first_join_date
  select(-c(birthdate, first_join_date))

head(final)
```

```
# A tibble: 6 × 22
  session_id            churn event_time customer_id
payment_method promo_amount
  <chr>                 <lgl> <date>           <int> <chr>
<int>
1 00098329-5a72-48b5-9… TRUE  2022-01-01       10161 LinkAja
4105
2 00098329-5a72-48b5-9… TRUE  2022-01-01       10161 LinkAja
4105
3 0020271e-e1b9-4656-8… TRUE  2022-01-01        1073 OVO
0
4 00a772a5-221c-4874-8… FALSE 2022-01-01       49907 Debit Card
0
5 c9297c63-f315-49e7-9… FALSE 2022-01-13       49907 Debit Card
5332
6 c9297c63-f315-49e7-9… FALSE 2022-01-13       49907 Debit Card
5332
# ℹ 16 more variables: shipment_fee <int>, product_id <chr>,
quantity <int>,
#   item_price <int>, traffic_source <chr>, shipment_eta <dbl>,
#   masterCategory <chr>, season <chr>, year <int>, usage
<chr>, brand <chr>,
#   gender <chr>, device_version <chr>, home_location <chr>,
age <int>,
#   member_duration <dbl>
```

```
data <- final |>
  na.omit() |>
  mutate(usage = as.factor(usage),
         brand = as.factor(brand),
         gender = as.factor(gender),
```

```
         payment_method = as.factor(payment_method),
         traffic_source = as.factor(traffic_source),
         masterCategory = as.factor(masterCategory),
         season = as.factor(season),
         device_version = as.factor(device_version),
         home_location = as.factor(home_location)) |>
  select(-c(session_id, customer_id, product_id))

head(data)
```

```
# A tibble: 6 × 19
  churn event_time payment_method promo_amount shipment_fee
quantity item_price
  <lgl> <date>     <fct>                 <int>        <int>
<int>      <int>
1 TRUE  2022-01-01 LinkAja                4105        15000
1     544791
2 TRUE  2022-01-01 LinkAja                4105        15000
1     310626
3 FALSE 2022-01-01 Debit Card                0            0
1      76862
4 FALSE 2022-01-13 Debit Card             5332            0
1     253662
5 FALSE 2022-01-13 Debit Card             5332            0
2     203154
6 FALSE 2022-01-19 Debit Card                0        10000
1     205157
# ℹ 12 more variables: traffic_source <fct>, shipment_eta
<dbl>,
#   masterCategory <fct>, season <fct>, year <int>, usage
<fct>, brand <fct>,
#   gender <fct>, device_version <fct>, home_location <fct>,
age <int>,
#   member_duration <dbl>
```

```
data <- data |>
  mutate(churn = as.factor(churn))

head(data)
```

```
# A tibble: 6 × 19
  churn event_time payment_method promo_amount shipment_fee
quantity item_price
  <fct> <date>     <fct>                 <int>        <int>
<int>      <int>
1 TRUE  2022-01-01 LinkAja                4105        15000
1     544791
```

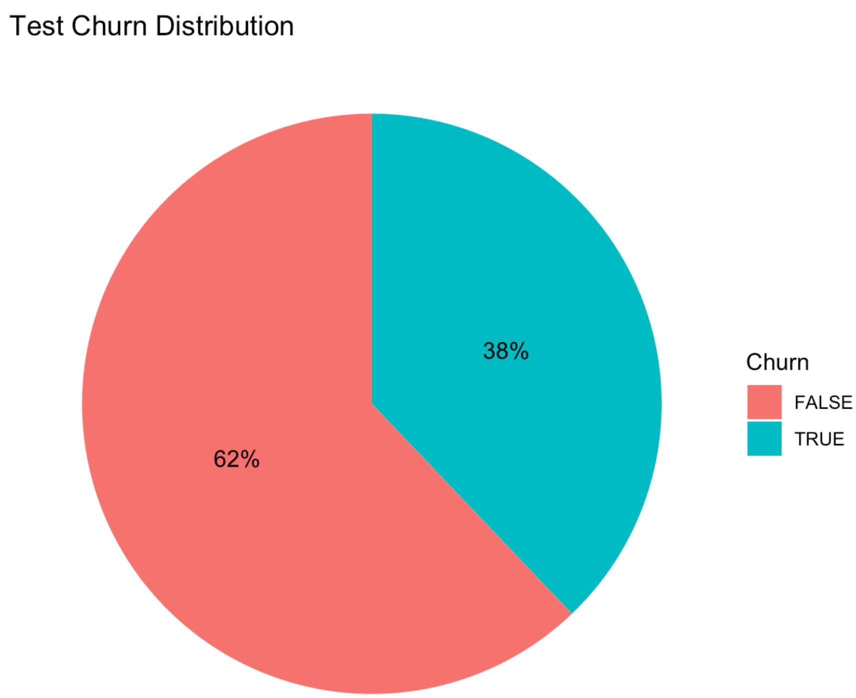
```
2 TRUE  2022-01-01 LinkAja                4105        15000
1     310626
3 FALSE 2022-01-01 Debit Card                0            0
1      76862
4 FALSE 2022-01-13 Debit Card             5332            0
1     253662
5 FALSE 2022-01-13 Debit Card             5332            0
2     203154
6 FALSE 2022-01-19 Debit Card                0        10000
1     205157
# ℹ 12 more variables: traffic_source <fct>, shipment_eta
<dbl>,
#   masterCategory <fct>, season <fct>, year <int>, usage
<fct>, brand <fct>,
#   gender <fct>, device_version <fct>, home_location <fct>,
age <int>,
#   member_duration <dbl>
```

```
test_set <- data |>
  mutate(product_year = 2021-year) |>
  relocate(product_year, .after=year) |>
  select(-event_time, -year, -brand, -device_version) |>
  mutate(churn = as.factor(churn),
         home_location = str_extract(home_location, "\\S+"))

head(test_set)
```

```
# A tibble: 6 × 16
  churn payment_method promo_amount shipment_fee quantity
item_price
  <fct> <fct>                 <int>        <int>    <int>
<int>
1 TRUE  LinkAja                4105        15000        1
544791
2 TRUE  LinkAja                4105        15000        1
310626
3 FALSE Debit Card                0            0        1
76862
4 FALSE Debit Card             5332            0        1
253662
5 FALSE Debit Card             5332            0        2
203154
6 FALSE Debit Card                0        10000        1
205157
# ℹ 10 more variables: traffic_source <fct>, shipment_eta
<dbl>,
#   masterCategory <fct>, season <fct>, product_year <dbl>,
```

```
usage <fct>,
#   gender <fct>, home_location <chr>, age <int>,
member_duration <dbl>
```

```
library(ggplot2)
churn_df <- as.data.frame(table(test_set$churn)) |>
  mutate(Churn = Var1,
         Frequency = Freq) |>
  select(-Var1, -Freq) |>
  mutate(Percentage = Frequency / sum(Frequency) * 100)

ggplot(churn_df, aes(x = "", y = Frequency, fill = Churn)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") +
  labs(title = "Test Churn Distribution") +
  theme_void() +
  geom_text(aes(label = paste0(round(Percentage), "%")), positi
```



This pie chart also shows the balanced target ratio in test set.

```
write.csv(train_set, file = "data/train.csv", row.names = FALSE
write.csv(test_set, file = "data/test.csv", row.names = FALSE)
```

Here's the period we used for training and test: training set: 2021-01 ~ 2021-12

test set: 2022-01

```r
train <- read.csv("data/train.csv") |>
  mutate(churn = as.factor(churn))
head(train)
```

```
  churn payment_method promo_amount shipment_fee quantity
item_price
1 FALSE          Gopay            0        50000        1
161945
2 FALSE    Credit Card            0            0        1
244964
3 FALSE          Gopay            0            0        1
376160
4 FALSE          Gopay            0            0        1
307578
5 FALSE        LinkAja            0            0       18
146764
6 FALSE          Gopay            0        15000        2
386312
  traffic_source shipment_eta masterCategory season
product_year  usage gender
1         MOBILE            1       Footwear Winter
9 Casual      F
2         MOBILE            2        Apparel   Fall
10 Casual      F
3         MOBILE            3    Accessories   Fall
10 Formal      F
4         MOBILE            3    Accessories Winter
6 Casual      F
5         MOBILE            1        Apparel   Fall
11 Casual      F
6         MOBILE            2        Apparel   Fall
9 Ethnic      F
  home_location age member_duration
1    Yogyakarta  33             289
2    Yogyakarta  33             300
3    Yogyakarta  33             311
4    Yogyakarta  33             322
5    Yogyakarta  33             333
6    Yogyakarta  33             344
```

```r
test <- read.csv("data/test.csv") |>
  mutate(churn = as.factor(churn))
head(test)
```

```
  churn payment_method promo_amount shipment_fee quantity
item_price
```

```
1  TRUE        LinkAja        4105       15000        1
544791
2  TRUE        LinkAja        4105       15000        1
310626
3 FALSE     Debit Card           0           0        1
76862
4 FALSE     Debit Card        5332           0        1
253662
5 FALSE     Debit Card        5332           0        2
203154
6 FALSE     Debit Card           0       10000        1
205157
  traffic_source shipment_eta masterCategory season
product_year  usage gender
1         MOBILE            5    Accessories Winter
6 Casual      M
2         MOBILE            5        Apparel   Fall
10 Casual      M
3         MOBILE            2        Apparel Summer
9 Formal      F
4            WEB            1       Footwear Summer
9 Casual      F
5            WEB            1        Apparel Summer
9 Ethnic      F
6         MOBILE            3       Footwear   Fall
10 Casual      F
  home_location age member_duration
1       Jakarta  36            1442
2       Jakarta  36            1442
3          Jawa  29             686
4          Jawa  29             698
5          Jawa  29             698
6          Jawa  29             704
```

Modeling

Model 1

logistic regression with all variables: A logistic regression, fit using maximum likelihood, with churn as the response and all other variables as explanatory variables. The recipe normalizes all the numeric predictors. Moreover, it replace any home_location that occurs in less than 1 percent of the data with an “other” category.

```r
recipe_1 <- recipe(data = train,
                   formula = churn ~ .) |>
  step_normalize(all_numeric_predictors()) |>
  step_other(home_location,
             threshold = 0.01,
             other = "other")

parsnip_1 <- logistic_reg() |>
  set_mode("classification") |>
  set_engine("glm")

workflow_1 <- workflow() |>
  add_model(parsnip_1) |>
  add_recipe(recipe_1)
```

```r
glm_fit <- workflow_1 |>
  fit(data = train)

glm_fit
```

```
══ Workflow [trained] ══════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: logistic_reg()

── Preprocessor ────────────────────────────────────────────────────────────────
─
2 Recipe Steps

• step_normalize()
• step_other()

── Model ───────────────────────────────────────────────────────────────────────
────────

Call:  stats::glm(formula = ..y ~ ., family = stats::binomial,
data = data)

Coefficients:
                (Intercept)      payment_methodDebit Card
                 -0.3523648                    -0.0247119
        payment_methodGopay         payment_methodLinkAja
                  0.0194529                     0.0086615
          payment_methodOVO                  promo_amount
```

```
                  0.0003333                     0.0284403
               shipment_fee                      quantity
                  0.0079213                     0.0036154
                 item_price              traffic_sourceWEB
                  0.0068722                     0.0040416
               shipment_eta         masterCategoryApparel
                  0.0008440                    -0.0019783
     masterCategoryFootwear      masterCategoryFree Items
                 -0.0014353                     0.0096854
         masterCategoryHome   masterCategoryPersonal Care
                  1.4549904                    -0.0753189
masterCategorySporting Goods                 seasonSpring
                 -0.0612083                     0.0669303
               seasonSummer                  seasonWinter
                  0.0064204                    -0.0020377
               product_year                   usageEthnic
                 -0.0023075                    -0.0004080
                usageFormal                     usageHome
                 -0.0195514                            NA
                 usageParty             usageSmart Casual
                 -0.1282483                    -0.0086525
                usageSports                   usageTravel
                 -0.0237835                    -0.0870972
                    genderM           home_locationJakarta
                  0.0165542                     0.0017327
         home_locationJawa        home_locationKalimantan
                 -0.0431480                     0.0001054
     home_locationKepulauan          home_locationLampung
                 -0.1372291                     0.0880367
        home_locationMaluku             home_locationNusa
                 -0.0978956                    -0.0221477
         home_locationPapua          home_locationSulawesi
                 -0.0102430                     0.1101835
      home_locationSumatera       home_locationYogyakarta
                 -0.1480163                    -0.0210246
         home_locationother                           age
                  0.0820716                     0.0062866
            member_duration
                  0.1138840

Degrees of Freedom: 307954 Total (i.e. Null);  307913 Residual

...
and 2 more lines.
```

```
library(yardstick)
glm_predict <- predict(glm_fit,
```

```r
                            test,
                            type = "class")

df1 <- data.frame(tru = test$churn, est = glm_predict$.pred_cla
confusion_mat <- conf_mat(df1, truth = tru, estimat = est)

confusion_mat
```

```
          Truth
Prediction FALSE  TRUE
     FALSE 22342 13610
     TRUE    134    93
```

```r
acc <- accuracy(df1, truth = tru, estimat = est)
acc
```

```
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.620
```

Model 2

A logistic regression, fit using the **lasso with a penalty = 0.1**, with arr_delay_over_30 as the response and all other variables as explanatory variables. Use the same recipe as in A. Call this workflow workflow_B.

```r
parsnip_2 <- logistic_reg(penalty = 0.01) |>
  set_mode("classification") |>
  set_engine("glmnet")

workflow_2 <- workflow() |>
  add_model(parsnip_2) |>
  add_formula(churn ~ .)
```

```r
glmnet_fit <- workflow_2 |>
  fit(data = train)

glmnet_fit
```

```
══ Workflow [trained] ══════════════════════════════════════════════════════════

Preprocessor: Formula
Model: logistic_reg()
```

```
── Preprocessor ────────────────────────────────────────────────────────────────
─
churn ~ .

── Model ───────────────────────────────────────────────────────────────────────
────────

Call:  glmnet::glmnet(x = maybe_matrix(x), y = y, family = 
"binomial")

   Df %Dev    Lambda
1   0 0.00 0.0287500
2   1 0.04 0.0262000
3   1 0.08 0.0238700
4   1 0.11 0.0217500
5   1 0.13 0.0198200
6   1 0.15 0.0180600
7   1 0.17 0.0164500
8   1 0.18 0.0149900
9   1 0.20 0.0136600
10  1 0.20 0.0124500
11  1 0.21 0.0113400
12  1 0.22 0.0103300
13  1 0.22 0.0094150
14  1 0.23 0.0085790
15  2 0.23 0.0078170
16  2 0.24 0.0071220
17  3 0.25 0.0064900
18  5 0.25 0.0059130
19  8 0.27 0.0053880
20  8 0.28 0.0049090
21  8 0.29 0.0044730
22  8 0.29 0.0040760
23 10 0.30 0.0037140
24 10 0.31 0.0033840
25 11 0.31 0.0030830
26 11 0.32 0.0028090
27 12 0.32 0.0025600
28 12 0.33 0.0023320
29 13 0.33 0.0021250
30 15 0.33 0.0019360
31 17 0.34 0.0017640
32 19 0.34 0.0016080
33 21 0.34 0.0014650
34 21 0.34 0.0013350
```

```
35 23 0.35 0.0012160
36 25 0.35 0.0011080
37 25 0.35 0.0010100
38 25 0.35 0.0009199
39 26 0.35 0.0008382
40 27 0.35 0.0007637
```

```
glmnet_predict <- predict(glmnet_fit,
                          test,
                          type = "class")

df2 <- data.frame(tru = test$churn, est = glmnet_predict$.pred_c
confusion_mat <- conf_mat(df1, truth = tru, estimat = est)

confusion_mat
```

```
          Truth
Prediction FALSE  TRUE
     FALSE 22342 13610
     TRUE    134    93
```

```
acc2 <- accuracy(df2, truth = tru, estimat = est)
acc2
```

```
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.621
```

Model 3

A logistic regression, fit using the **top 5 principal components** of the remaining numerical variables.

```
parsnip_3 <- logistic_reg() |>
  set_mode("classification") |>
  set_engine("glm")

recipe_3 <- recipe(churn ~ .,
                   data = train) |>
  step_normalize(all_numeric_predictors()) |>
  step_other(home_location,
             threshold = 0.01,
             other = "other") |>
  step_pca(all_numeric_predictors(),
```

```
            num_comp = 5) |>
  step_dummy(all_nominal_predictors())


workflow_3 <- workflow() |>
  add_model(parsnip_3) |>
  add_recipe(recipe_3)
```

```
glm_fit_pca <- workflow_3 |>
  fit(data = train)

glm_fit_pca
```

```
══ Workflow [trained] ══════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: logistic_reg()

── Preprocessor ────────────────────────────────────────────────────────────────
─
4 Recipe Steps

• step_normalize()
• step_other()
• step_pca()
• step_dummy()

── Model ───────────────────────────────────────────────────────────────────────
────────

Call:  stats::glm(formula = ..y ~ ., family = stats::binomial,
data = data)

Coefficients:
                  (Intercept)                           PC1
                   -3.507e-01                     9.866e-02
                          PC2                           PC3
                    6.459e-03                    -2.203e-03
                          PC4                           PC5
                   -1.336e-02                    -2.138e-03
    payment_method_Debit.Card           payment_method_Gopay
                   -2.297e-02                     1.875e-02
        payment_method_LinkAja             payment_method_OVO
                    7.224e-03                     9.536e-04
            traffic_source_WEB         masterCategory_Apparel
```

```
                    3.301e-03                      5.623e-05
     masterCategory_Footwear     masterCategory_Free.Items
                    1.633e-03                      6.735e-03
         masterCategory_Home  masterCategory_Personal.Care
                    1.435e+00                     -8.655e-02
masterCategory_Sporting.Goods                 season_Spring
                   -4.571e-02                      6.111e-02
               season_Summer                  season_Winter
                    2.872e-03                     -1.371e-02
                usage_Ethnic                   usage_Formal
                   -3.803e-04                     -1.943e-02
                  usage_Home                    usage_Party
                           NA                     -1.282e-01
            usage_Smart.Casual                 usage_Sports
                   -1.029e-02                     -2.248e-02
                usage_Travel                       gender_M
                   -8.810e-02                      1.946e-02
       home_location_Jakarta             home_location_Jawa
                    2.144e-03                     -4.167e-02
    home_location_Kalimantan        home_location_Kepulauan
                    3.239e-04                     -1.357e-01
       home_location_Lampung           home_location_Maluku
                    8.836e-02                     -1.014e-01
          home_location_Nusa            home_location_Papua
                   -1.798e-02                     -6.811e-03
      home_location_Sulawesi         home_location_Sumatera
                    1.143e-01                     -1.494e-01
    home_location_Yogyakarta            home_location_other
                   -2.257e-02                      8.463e-02

Degrees of Freedom: 307954 Total (i.e. Null);  307916 Residual
Null Deviance:      417000
Residual Deviance: 415900   AIC: 416000
```

```
glm_pca_predict <- predict(glm_fit_pca,
                           test,
                           type = "class")

df3 <- data.frame(tru = test$churn, est = glm_pca_predict$.pred_
confusion_mat <- conf_mat(df3, truth = tru, estimat = est)

confusion_mat
```

```
          Truth
Prediction FALSE  TRUE
     FALSE 22351 13624
     TRUE    125    79
```

```
acc3 <- accuracy(df3, truth = tru, estimat = est)
acc3
```

```
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.620
```

Predictions

```
workflow_names <- c("logsitic",
                    "logistic_lasso",
                    "logistic_pca")

workflow_objects <- list(workflow_1,
                         workflow_2,
                         workflow_3)

workflows_tbl <- tibble(work_names = workflow_names,
                        work_objects = workflow_objects)

workflows_tbl
```

```
# A tibble: 3 × 2
  work_names     work_objects
  <chr>          <list>
1 logsitic       <workflow>
2 logistic_lasso <workflow>
3 logistic_pca   <workflow>
```

```
accuracy <- tibble(Model = c("Model1", "Model2", "Model3"),
                   Accuracy = c(0.6201111,0.6212444,0.6199729))

accuracy
```

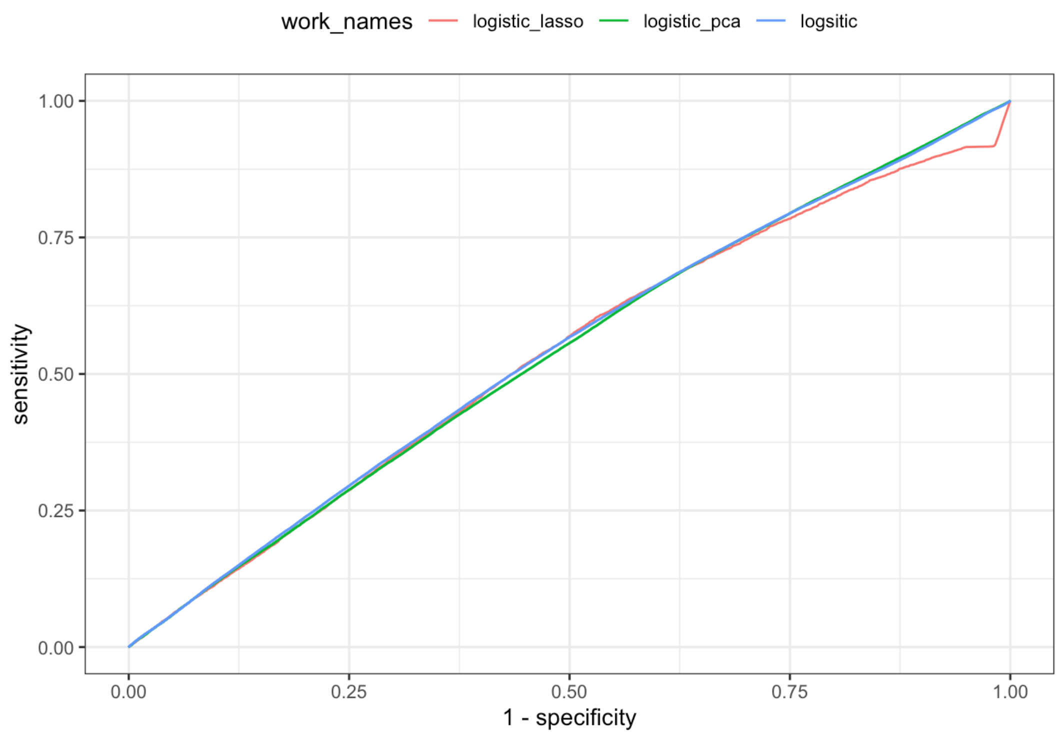
```
# A tibble: 3 × 2
  Model  Accuracy
  <chr>     <dbl>
1 Model1    0.620
2 Model2    0.621
3 Model3    0.620
```

Model 3, using logistic regression with **lasso with a penalty = 0.1**, has the highest accuracy here.

```r
set.seed(1)
workflows_tbl <- workflows_tbl |>
  rowwise() |>
  mutate(fits = list(fit(work_objects,
                         train)))
```

```r
workflows_resub_prob <- workflows_tbl |>
  mutate(predictions = list(predict(fits,
                                    train,
                                    type = "prob")))

predictions_resub  <- workflows_resub_prob |>
  select(work_names,
         predictions) |>
  unnest(cols = c(predictions)) |>
  cbind(churn = train |>
          pull(churn))
```

```r
roc_all <- predictions_resub |>
  group_by(work_names) |>
  roc_curve(truth = churn,
            .pred_TRUE,
            event_level = "second")

roc_all |>
  ggplot(aes(x = 1- specificity,
             y = sensitivity,
             color = work_names)) +
  geom_path() +
  theme(legend.position = "top")
```



The ROC graph’s almost linear pattern indicates that the AUC score close to 0.5. It reflects poor model performance suggesting classification performance not significantly better than random guessing.

Conclusion

The AUC score close to 0.5 for the evaluated models indicates poor predictive performance, essentially suggesting random classification. Although the logistic_lasso model demonstrated slightly higher accuracy compared to others, the ROC curve analysis revealed instances where the AUC fell below 0.5, further highlighting the limitations of the models. Selecting the best model remains challenging at this point.

In a real-world scenario, these findings have practical significance for the company. Deploying models with poor predictive performance could lead to ineffective decision-making and wasted resources. For example, in a customer churn prediction scenario, relying on inaccurate models may result in misidentification of at-risk customers, leading to ineffective retention strategies and potential loss of revenue.

However, there is an opportunity for improvement. Conducting feature engineering to create additional features, fine-tuning model parameters, or exploring alternative algorithms could enhance predictive performance. By investing in further analysis and modeling efforts, the company can develop a more robust and effective solution for predicting customer behavior, ultimately leading to better-informed decision-making and improved business outcomes.