Customer Identification Data Report

Adeline Casali

2023-10-15

Load packages, read in data, and select quasi-identifiers

In this section, I first loaded in all necessary packages. Next, I read in the data and select the quasi-identifiers to be shared with the telecommunications company our firm has a partnership with. I selected gender, age, years of education, and marital status to be analyzed in conjunction with the customer's tenure with the phone company. This will allow the partner company to determine which age, gender, marital status, and education statuses have the highest tenure, making them "good customers".

```
# Load packages
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(readxl)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
library(rmarkdown)
library(knitr)

# Read in dataset
customer_df <- read_excel("Data/Customer_Data.xlsx")

# Select variables to be shared
```

```
customer_selects_df <- customer_df %>%
  select(gender = Gender, age = Age, phoneco_tenure = PhoneCoTenure,
         education_years = EducationYears, marital_status = MaritalStatus)
```

Perform data quality control - Check for outliers and null values

As the first part of my analysis, I checked the data for outliers, null values, and other values that just don't make sense. For binary columns, I checked for numbers other than 1 or 0, and for non-binary columns I looked at the 5 lowest and highest results to determine if any were out of the ordinary.

```
cat(ifelse(any(customer_selects_df$gender != 0 & customer_selects_df$gender != 1),
           paste("Numbers in 'gender' column that are not 0 or 1:",
                 customer_selects_df$gender[customer_selects_df$gender != 0 & customer_selects_df$gende
           "None found in 'gender' column"), "\n")
```

```
## None found in ’gender’ column
```

```
cat(ifelse(length(customer_selects_df$gender[is.na(customer_selects_df$gender)]) > 0,
           paste("NULL or NA values in 'gender' column:",
                 customer_selects_df$gender[is.na(customer_selects_df$gender)],
                 sep = "\n"), "None found in 'gender' column"), "\n")
```

```
## None found in ’gender’ column
```

```
# No outliers or null values found in the gender column

customer_selects_df %>%
  arrange(age) %>%
  head(5) %>%
  print()
```

```
## # A tibble: 5 x 5
##   gender   age phoneco_tenure education_years marital_status
##    <dbl> <dbl>          <dbl>           <dbl>          <dbl>
## 1      0    18              2              13              0
## 2      1    18              6              13              1
## 3      1    18              2              13              1
## 4      1    18              7              13              1
## 5      0    18              7              13              1
```

```
customer_selects_df %>%
  arrange(desc(age)) %>%
  head(5) %>%
  print()
```

```
## # A tibble: 5 x 5
##   gender   age phoneco_tenure education_years marital_status
##    <dbl> <dbl>          <dbl>           <dbl>          <dbl>
## 1      0    79             71              12              1
## 2      1    79             72               7              0
## 3      0    79             67              16              0
## 4      1    79             45              12              1
## 5      0    79             34              12              1
```

```r
cat(ifelse(length(customer_selects_df$age[is.na(customer_selects_df$age)]) > 0,
           paste("NULL or NA values in 'age' column:",
                 customer_selects_df$age[is.na(customer_selects_df$age)],
                 sep = "\n"), "None found in 'age' column"), "\n")
```

```
## None found in ’age’ column
```

```r
# No outliers or null values found in the age column

customer_selects_df %>%
  arrange(phoneco_tenure) %>%
  head(5) %>%
  print()
```

```
## # A tibble: 5 x 5
##   gender   age phoneco_tenure education_years marital_status
##    <dbl> <dbl>          <dbl>           <dbl>          <dbl>
## 1      0    18              0              13              0
## 2      1    18              0              12              1
## 3      0    18              0              12              0
## 4      1    20              1              15              0
## 5      1    38              1              15              1
```

```r
customer_selects_df %>%
  arrange(desc(phoneco_tenure)) %>%
  head(5) %>%
  print()
```

```
## # A tibble: 5 x 5
##   gender   age phoneco_tenure education_years marital_status
##    <dbl> <dbl>          <dbl>           <dbl>          <dbl>
## 1      1    66             72              13              1
## 2      1    77             72              15              0
## 3      0    78             72              20              1
## 4      0    64             72              10              1
## 5      1    69             72               8              1
```

```r
cat(ifelse(length(customer_selects_df$phoneco_tenure[is.na(customer_selects_df$phoneco_tenure)]) > 0,
           paste("NULL or NA values in 'phoneco_tenure' column:",
                 customer_selects_df$phoneco_tenure[is.na(customer_selects_df$phoneco_tenure)],
                 sep = "\n"), "None found in 'phoneco_tenure' column"), "\n")
```

```
## None found in ’phoneco_tenure’ column
```

```
# No outliers or null values found in the phoneco_tenure column

customer_selects_df %>%
  arrange(education_years) %>%
  head(5) %>%
  print()
```

```
## # A tibble: 5 x 5
##   gender   age phoneco_tenure education_years marital_status
##    <dbl> <dbl>          <dbl>           <dbl>          <dbl>
## 1      0    69             69               6              1
## 2      0    48              4               6              0
## 3      0    75             64               6              0
## 4      1    45             48               6              1
## 5      1    36             40               6              1
```

```
customer_selects_df %>%
  arrange(desc(education_years)) %>%
  head(5) %>%
  print()
```

```
## # A tibble: 5 x 5
##   gender   age phoneco_tenure education_years marital_status
##    <dbl> <dbl>          <dbl>           <dbl>          <dbl>
## 1      0    45              4              23              0
## 2      0    75             26              23              0
## 3      1    29             46              23              1
## 4      1    70             60              23              1
## 5      1    49             31              22              1
```

```
cat(ifelse(length(customer_selects_df$education_years[is.na(customer_selects_df$education_years)]) > 0,
           paste("NULL or NA values in 'education_years' column:",
                 customer_selects_df$education_years[is.na(customer_selects_df$education_years)],
                 sep = "\n"), "None found in 'education_years' column"), "\n")
```

```
## None found in 'education_years' column
```

```
# No outliers or null values found in the education_years column

cat(ifelse(any(customer_selects_df$marital_status != 0 & customer_selects_df$marital_status != 1),
           paste("Numbers in 'marital_status' column that are not 0 or 1:",
                 customer_selects_df$marital_status[customer_selects_df$marital_status != 0 & customer_s
           "None found in 'marital_status' column"), "\n")
```

```
## None found in 'marital_status' column
```

```
cat(ifelse(length(customer_selects_df$marital_status[is.na(customer_selects_df$marital_status)]) > 0,
           paste("NULL or NA values in 'marital_status' column:",
                 customer_selects_df$marital_status[is.na(customer_selects_df$marital_status)],
                 sep = "\n"), "None found in 'marital_status' column"), "\n")
```

```
## None found in 'marital_status' column
```

```
# No outliers or null values found in the marital_status column
```

Check current equivalence classes

Next, I checked the current equivalence classes. There were many classes with only one unique customer in them, so grouping is needed to decrease the uniqueness of those customers.

```
equivalence_classes <- customer_selects_df %>%
  group_by_all() %>%
  tally(name = "class_size")
print(head(equivalence_classes %>% arrange(class_size)))
```

```
## # A tibble: 6 x 6
## # Groups:   gender, age, phoneco_tenure, education_years [6]
##   gender   age phoneco_tenure education_years marital_status class_size
##    <dbl> <dbl>          <dbl>           <dbl>          <dbl>      <int>
## 1      0    18              0              12              0          1
## 2      0    18              0              13              0          1
## 3      0    18              2              11              0          1
## 4      0    18              2              13              1          1
## 5      0    18              3              11              0          1
## 6      0    18              4               8              0          1
```

```
# There are currently many equivalence classes with only one customer,
#and a maximum of 6 customers in an equivalence class
```

Perform grouping to decrease the number of unique customers

In this section, I performed grouping within the age, education_years, and phoneco_tenure columns to decrease the number of unique customers until there were no unique customers.

```
# Group age by a 15 year range, with <25 and 55+ as the upper and lowers
customer_selects_df$age <- cut(customer_selects_df$age,
                                     breaks = c(0, 30, 45, 60, 100),
                                     labels = c("<30", "30-44", "45-59", "60+"))

# Check equivalence classes
equivalence_classes <- customer_selects_df %>%
  group_by_all() %>%
  tally(name = "class_size")
print(head(equivalence_classes %>% arrange(class_size)))
```

```
## # A tibble: 6 x 6
## # Groups:   gender, age, phoneco_tenure, education_years [6]
##   gender age   phoneco_tenure education_years marital_status class_size
##    <dbl> <fct>          <dbl>           <dbl>          <dbl>      <int>
## 1      0 <30                0              12              0          1
## 2      0 <30                0              13              0          1
## 3      0 <30                1              12              0          1
## 4      0 <30                1              14              0          1
## 5      0 <30                1              15              1          1
## 6      0 <30                1              16              1          1
```

```
# There are still many unique customers, so more grouping needs to be done

# Group education_years by <15 (no college degree) or 15+ (college degree)
customer_selects_df$education_years <- cut(customer_selects_df$education_years,
                               breaks = c(0, 15, 100),
                               labels = c("<15", "15+"))

# Check equivalence classes
equivalence_classes <- customer_selects_df %>%
  group_by_all() %>%
  tally(name = "class_size")
print(head(equivalence_classes %>% arrange(class_size)))
```

```
## # A tibble: 6 x 6
## # Groups:   gender, age, phoneco_tenure, education_years [6]
##   gender age   phoneco_tenure education_years marital_status class_size
##    <dbl> <fct>          <dbl> <fct>                    <dbl>      <int>
## 1      0 <30                2 15+                          1          1
## 2      0 <30                3 <15                          1          1
## 3      0 <30                6 15+                          1          1
## 4      0 <30                8 15+                          1          1
## 5      0 <30               12 15+                          1          1
## 6      0 <30               17 <15                          1          1
```

```
# There are still many unique customers, so we will now group within the phoneco_tenure column

# Group phoneco_tenure by 24 months
customer_selects_df$phoneco_tenure <- cut(customer_selects_df$phoneco_tenure,
                                          breaks = c(-1, 24, 48, 72, 100),
                                          labels = c("<24", "24-47", "48-72", "72+"))

# Check equivalence classes
equivalence_classes <- customer_selects_df %>%
  group_by_all() %>%
  tally(name = "class_size")
print(head(equivalence_classes %>% arrange(class_size)))
```

```
## # A tibble: 6 x 6
## # Groups:   gender, age, phoneco_tenure, education_years [5]
##   gender age   phoneco_tenure education_years marital_status class_size
##    <dbl> <fct> <fct>          <fct>                    <dbl>      <int>
## 1      0 <30   48-72          15+                          0          2
## 2      1 <30   48-72          15+                          0          3
## 3      1 60+   <24            15+                          1          3
## 4      1 <30   48-72          <15                          0          4
## 5      0 <30   48-72          15+                          1          5
## 6      0 60+   <24            15+                          1          8
```

Add in column for probability of re-identification

Next, I added in a column with the probability of re-identification for each of the equivalence classes based on class size. I used this information to calculate the maximum and median probabilities of re-identification.

```r
equivalence_classes <- equivalence_classes %>%
  mutate(p_re_id = round(1/ class_size, 2))

# Calculate the maximum and median risks
p_max <- max(equivalence_classes$p_re_id)
cat("The maximum probability of re-identification is:", p_max, "\n")
```

```
## The maximum probability of re-identification is: 0.5
```

```r
p_med <- median(equivalence_classes$p_re_id)
cat("The median probability of re-identification is:", p_med, "\n")
```

```
## The median probability of re-identification is: 0.02
```

Scenario 1 Calculations

I used the probability of re-identification from above to calculate the maximum and median risks of re-identification under scenario 1.

```r
# Formula: p(re-id|attempt) * p(attempt)
# Being conservative and assuming the mitigating controls are low and the motives and capacity are high
p_attempt <- 0.6

# Calculation with max probability of re-identification
s1_max_risk <- ((p_max*p_attempt)/p_attempt)*p_attempt
cat("The maximum probability of re-identification for scenario 1 is:", s1_max_risk, "\n")
```

```
## The maximum probability of re-identification for scenario 1 is: 0.3
```

```r
# Calculation with median probability of re-identification
s1_med_risk <- ((p_med*p_attempt)/p_attempt)*p_attempt
cat("The median probability of re-identification for scenario 1 is:", s1_med_risk, "\n")
```

```
## The median probability of re-identification for scenario 1 is: 0.012
```

Scenario 2 Calculations

And repeated that with scenario 2.

```
# Formulas: p(re-id|acquanit) * p(acquaint), p(acquaint) = 1-(1-p)^m
p <- 0.97
m <- 150
p_acquaint <- 1-(1-p)^m

# Calculation with max probability of re-identification
s2_max_risk <- ((p_max*p_acquaint)/p_acquaint)*p_acquaint
cat("The maximum probability of re-identification for scenario 2 is:", s2_max_risk, "\n")
```

```
## The maximum probability of re-identification for scenario 2 is: 0.5
```

```
# Calculation with median probability of re-identification
s2_med_risk <- ((p_med*p_acquaint)/p_acquaint)*p_acquaint
cat("The median probability of re-identification for scenario 2 is:", s2_med_risk, "\n")
```

```
## The median probability of re-identification for scenario 2 is: 0.02
```

Scenario 3 Calculations

And scenario 3.

```
# Formula: p(re-id|breach)*p(breach)
# Being conservative, we will assume the likelihood of a breach is the same as that of a healthcare com
p_breach <- 0.27

# Calculation with max probability of re-identification
s3_max_risk <- ((p_max*p_breach)/p_breach)*p_breach
cat("The maximum probability of re-identification for scenario 3 is:", s3_max_risk, "\n")
```

```
## The maximum probability of re-identification for scenario 3 is: 0.135
```

```
# Calculation with median probability of re-identification
s3_med_risk <- ((p_med*p_breach)/p_breach)*p_breach
cat("The median probability of re-identification for scenario 3 is:", s3_med_risk, "\n")
```

```
## The median probability of re-identification for scenario 3 is: 0.0054
```

Scenario 4 Calculations

And finally, scenario 4.

```
# Formula: p(re-id)

# Calculation with max probability of re-identification
s4_max_risk <- p_max
cat("The maximum probability of re-identification for scenario 4 is:", s4_max_risk, "\n")
```

```
## The maximum probability of re-identification for scenario 4 is: 0.5
```

```r
# Calculation with median probability of re-identification
s4_med_risk <- p_med
cat("The median probability of re-identification for scenario 4 is:", s4_med_risk, "\n")
```

```
## The median probability of re-identification for scenario 4 is: 0.02
```

Create a table with results

Next, I took the risk calculations from the different scenarios and compiled a table with the maximum, median, and assessment of results from each of the scenarios. I based the assessment on a threshold of 33%, as discussed further in the certification report.

```r
risk_results <- data.frame(
  Results = c("Max Risk", "Median Risk", "Assessment"),
  S1 = c("30%", "1.2%", "Tolerable"),
  S2 = c("50%", "2%", "Unacceptable"),
  S3 = c("13.5%", "0.05%", "Tolerable"),
  S4 = c("50%", "2%", "Unacceptable")
)

risk_table <- kable(risk_results, format = "latex") %>%
  kable_styling(full_width = FALSE, latex_options = "HOLD_position")
risk_table <- risk_table %>%
  column_spec(1, background = "lightgray") %>%
  column_spec(2, background = "green") %>%
  column_spec(3, background = "red") %>%
  column_spec(4, background = "green") %>%
  column_spec(5, background = "red")

# Print the formatted table
risk_table
```

Results	S1	S2	S3	S4
Max Risk	30%	50%	13.5%	50%
Median Risk	1.2%	2%	0.05%	2%
Assessment	Tolerable	Unacceptable	Tolerable	Unacceptable

Create a table with risk ranges

Finally, I created a table with the percentage of customers at each risk range for each scenario.

```r
# Scenario 1
equivalence_classes <- equivalence_classes %>%
  mutate(s1_risk = round(((p_re_id*p_attempt)/p_attempt)*p_attempt, 2))
# Scenario 2
equivalence_classes <- equivalence_classes %>%
  mutate(s2_risk = round(((p_re_id*p_acquaint)/p_acquaint)*p_acquaint, 2))
# Scenario 3
equivalence_classes <- equivalence_classes %>%
  mutate(s3_risk = round(((p_re_id*p_breach)/p_breach)*p_breach, 2))
# Scenario 4
equivalence_classes <- equivalence_classes %>%
  mutate(s4_risk = round(p_re_id, 2))

# Create risk ranges
equivalence_classes$s1_risk <- cut(equivalence_classes$s1_risk,
                               breaks = c(-1, 0.05, 0.10, 0.20, 0.33, 0.5, 1),
                               labels = c("<5%", "<10%", "<20%", "<33%", "<50%", ">50%"))
equivalence_classes$s2_risk <- cut(equivalence_classes$s2_risk,
                                   breaks = c(-1, 0.05, 0.10, 0.20, 0.33, 0.5, 1),
                                   labels = c("<5%", "<10%", "<20%", "<33%", "<50%", ">50%"))
equivalence_classes$s3_risk <- cut(equivalence_classes$s3_risk,
                                   breaks = c(-1, 0.05, 0.10, 0.20, 0.33, 0.5, 1),
                                   labels = c("<5%", "<10%", "<20%", "<33%", "<50%", ">50%"))
equivalence_classes$s4_risk <- cut(equivalence_classes$s4_risk,
                                   breaks = c(-1, 0.05, 0.10, 0.20, 0.33, 0.5, 1),
                                   labels = c("<5%", "<10%", "<20%", "<33%", "<50%", ">50%"))

# Count the class size for each risk range
s1_sum_class_size <- equivalence_classes %>%
  group_by(s1_risk) %>%
  summarize(percent_risk = (sum(class_size)/5000)) %>%
  ungroup()
s2_sum_class_size <- equivalence_classes %>%
  group_by(s2_risk) %>%
  summarize(percent_risk = (sum(class_size)/5000)) %>%
  ungroup()
s3_sum_class_size <- equivalence_classes %>%
  group_by(s3_risk) %>%
  summarize(percent_risk = (sum(class_size)/5000)) %>%
  ungroup()
s4_sum_class_size <- equivalence_classes %>%
  group_by(s4_risk) %>%
  summarize(percent_risk = (sum(class_size)/5000)) %>%
  ungroup()

# Create a table with risk ranges
risk_ranges <- data.frame(
  Risk = c("<5%", "<10%", "<20%", "<33%", "<50%", ">50%"),
  S1 = c("99.2%", "0.05%", "0.03%", "0", "0", "0"),
  S2 = c("96.5%", "2.7%", "0.06%", "0.02%", "0", "0"),
  S3 = c("99.8%", "0.02%", "0", "0", "0", "0"),
  S4 = c("96.5%", "2.7%", "0.06%", "0.02%", "0", "0")
)

risk_ranges_table <- kable(risk_ranges, format = "latex") %>%
  kable_styling(full_width = TRUE, latex_options = "HOLD_position")
risk_ranges_table
```

Risk	S1	S2	S3	S4
<5%	99.2%	96.5%	99.8%	96.5%
<10%	0.05%	2.7%	0.02%	2.7%
<20%	0.03%	0.06%	0	0.06%
<33%	0	0.02%	0	0.02%
<50%	0	0	0	0
>50%	0	0	0	0