Coupon Usage Prediction Report

YONG YANG

Overview

This report utilizes the offline consumption dataset from Alibaba's O2O service to predict whether a user will use a coupon after receiving it. Given the large size of the dataset, which includes three parts online coupons, offline coupons, and user information I selected the relatively smaller offline dataset to accommodate my computer's performance limitations.

In terms of data processing, I first handled missing values and generated new features based on the discount rates. Considering the significant imbalance between positive and negative samples, I employed methods to balance the samples during data splitting. I then sequentially applied logistic regression, random forest for classification predictions, introducing cross-validation to improve the models' predictive performance.

Dataset Download and Description

from(https://tianchi.aliyun.com/dataset/137322?lang=en-us) This dataset provides real online and offline user consumption data from January 1, 2016 to June 30, 2016. Researchers are expected to predict the probability of customers redeeming a coupon within 15 days of receiving it.

Note: To protect the privacy of users and merchants, data is desensitized and under biased sampling.

Offline consumption & coupons Table offline_train.csv.zip

Field	Description
User_id	User ID
Merchant_id	Merchant ID
Coupon_id	Coupon ID, when Coupon_id = null, this means that a coupon has not been redeemed. In such case, Discount_rate and Date_received don't matter.
Discount_rate	Discount rate, range in [0,1]
Distance	500x, the distance from the nearest shop around the user for locations in which a user is most active. x range in [0,10]; 0 – less than 500 meters; 10 – more than 5 kilometers.
Date_received	Date the coupon is received
Date	Purchase date. When Date=null & Coupon_id!= null, users receive coupon but don't redeem it; When Date!=null & Coupon_id= null, purchase happened but no coupon had been received; When Date!=null & Coupon_id!= null, 'Date' in which the coupon was used.

```r
#install packages
options(repos = c(CRAN = "https://cran.rstudio.com/"))
#install.packages('tidyverse')
#install.packages('lubridate')
#install.packages('dplyr')
#install.packages('tidyr')
#install.packages('caret')
```

```r
#install.packages('ggplot2')
#install.packages('ROSE')
#install.packages('randomForest')
#install.packages("xgboost")
```

```r
#import packages
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- 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 errors
```

```r
library(lubridate)
library(ggplot2)
library(caret)
```

```
##      lattice
##
##    'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
# Read the CSV file
data <- read_csv("offline_train.csv")
```

```
## Rows: 1754884 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): Coupon_id, Discount_rate, Distance, Date_received, Date
## dbl (2): User_id, Merchant_id
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Display the first 10 rows of the dataset
view(head(data, 10))
head(data, 10)
```

```
## # A tibble: 10 x 7
##    User_id Merchant_id Coupon_id Discount_rate Distance Date_received Date
##      <dbl>       <dbl> <chr>     <chr>         <chr>    <chr>         <chr>
##  1 1439408         2632 null      null          0         null          20160217
##  2 1439408         4663 11002     150:20        1         20160528      null
##  3 1439408         2632 8591      20:1          0         20160217      null
##  4 1439408         2632 1078      20:1          0         20160319      null
##  5 1439408         2632 8591      20:1          0         20160613      null
##  6 1439408         2632 null      null          0         null          20160516
##  7 1439408         2632 8591      20:1          0         20160516      20160613
##  8 1832624         3381 7610      200:20        0         20160429      null
##  9 2029232         3381 11951     200:20        1         20160129      null
## 10 2029232          450 1532      30:5          0         20160530      null
```

```
# Check for missing values in the Discount_rate column
table(is.na(data$Discount_rate))
```

```
##
##   FALSE
## 1754884
```

```
# Display unique values in the Discount_rate column
unique(data$Discount_rate)
```

```
##  [1] "null"    "150:20"  "20:1"    "200:20"  "30:5"    "50:10"   "10:5"
##  [8] "100:10"  "200:30"  "20:5"    "30:10"   "50:5"    "150:10"  "100:30"
## [15] "200:50"  "100:50"  "300:30"  "50:20"   "0.9"     "10:1"    "30:1"
## [22] "0.95"    "100:5"   "5:1"     "100:20"  "0.8"     "50:1"    "200:10"
## [29] "300:20"  "100:1"   "150:30"  "300:50"  "20:10"   "0.85"    "0.6"
## [36] "150:50"  "0.75"    "0.5"     "200:5"   "0.7"     "30:20"   "300:10"
## [43] "0.2"     "50:30"   "200:100" "150:5"
```

```
# Replace "null" with NA in the Discount_rate column
data$Discount_rate <- ifelse(data$Discount_rate == "null", NA, data$Discount_rate)

# Recheck for missing values in the Discount_rate column
table(is.na(data$Discount_rate))
```

```
##
##   FALSE    TRUE
## 1053282  701602
```

```
# Recheck unique values in the Discount_rate column
unique(data$Discount_rate)
```

```
##  [1] NA        "150:20"  "20:1"    "200:20"  "30:5"    "50:10"   "10:5"
##  [8] "100:10"  "200:30"  "20:5"    "30:10"   "50:5"    "150:10"  "100:30"
## [15] "200:50"  "100:50"  "300:30"  "50:20"   "0.9"     "10:1"    "30:1"
## [22] "0.95"    "100:5"   "5:1"     "100:20"  "0.8"     "50:1"    "200:10"
## [29] "300:20"  "100:1"   "150:30"  "300:50"  "20:10"   "0.85"    "0.6"
## [36] "150:50"  "0.75"    "0.5"     "200:5"   "0.7"     "30:20"   "300:10"
## [43] "0.2"     "50:30"   "200:100" "150:5"
```

```r
# Check for missing values in the Distance column
table(is.na(data$Distance))
```

```
##
##   FALSE
## 1754884
```

```r
# Display unique values in the Distance column
unique(data$Distance)
```

```
##  [1] "0"    "1"    "null" "2"    "10"   "4"    "7"    "9"    "3"    "5"
## [11] "6"    "8"
```

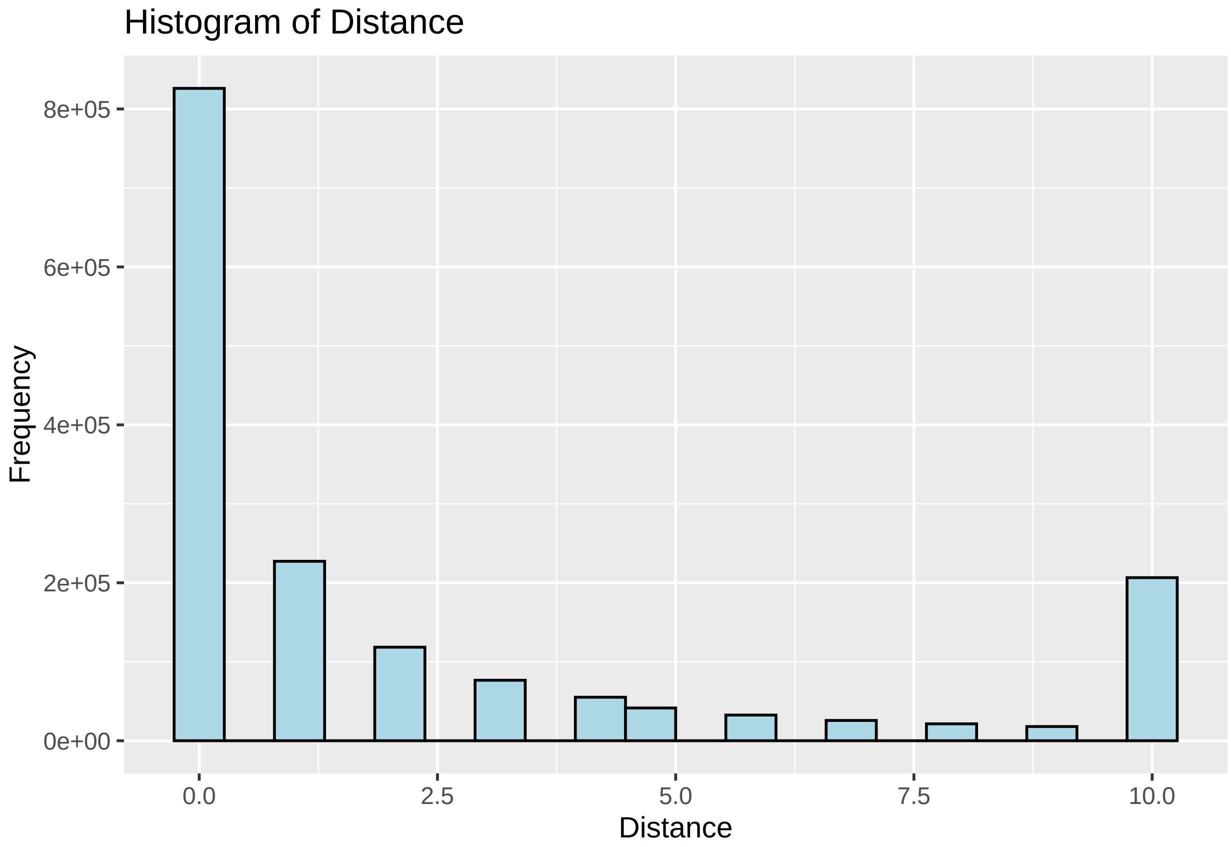
```r
# Replace "null" with NA in the Distance column
data$Distance <- ifelse(data$Distance == "null", NA, data$Distance)

# Recheck unique values in the Distance column
unique(data$Distance)
```

```
##  [1] "0"  "1"  NA   "2"  "10" "4"  "7"  "9"  "3"  "5"  "6"  "8"
```

```r
# Convert the Distance column to numeric
data$Distance <- as.numeric(data$Distance)
```

```r
# Plot a histogram of the Distance column
ggplot(data, aes(x = Distance)) +
  geom_histogram(bins = 20, color = "black", fill = "lightblue") +
  labs(x = "Distance", y = "Frequency", title = "Histogram of Distance")
```

```
## Warning: Removed 106003 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



The proportion of missing values here is relatively small, so it doesn't significantly impact the model. Therefore, we'll fill in the missing values with the median.

```
# Calculate the median of the Distance column
median_distance <- median(data$Distance, na.rm = TRUE)

# Fill missing values in the Distance column with the median
data$Distance[is.na(data$Distance)] <- median_distance
```

```
# Check if the 'Date' and 'Date_received' columns are in date format
class(data$Date)
```

```
## [1] "character"
```

```
class(data$Date_received)
```

```
## [1] "character"
```

```
# Display unique values in the Date and Date_received columns
unique(data$Date)
```

```
##   [1] "20160217" "null"     "20160516" "20160613" "20160626" "20160519"
##   [7] "20160606" "20160327" "20160115" "20160114" "20160521" "20160329"
##  [13] "20160321" "20160522" "20160322" "20160621" "20160523" "20160106"
##  [19] "20160324" "20160627" "20160130" "20160605" "20160614" "20160326"
##  [25] "20160221" "20160218" "20160128" "20160624" "20160612" "20160609"
##  [31] "20160331" "20160328" "20160415" "20160508" "20160517" "20160325"
##  [37] "20160601" "20160509" "20160123" "20160206" "20160214" "20160404"
##  [43] "20160524" "20160318" "20160408" "20160616" "20160610" "20160417"
##  [49] "20160205" "20160203" "20160406" "20160215" "20160119" "20160103"
##  [55] "20160105" "20160503" "20160520" "20160528" "20160409" "20160501"
##  [61] "20160615" "20160502" "20160112" "20160515" "20160604" "20160416"
##  [67] "20160228" "20160407" "20160323" "20160303" "20160425" "20160427"
##  [73] "20160607" "20160108" "20160315" "20160109" "20160424" "20160314"
##  [79] "20160118" "20160309" "20160121" "20160525" "20160608" "20160611"
##  [85] "20160110" "20160620" "20160222" "20160129" "20160312" "20160320"
##  [91] "20160504" "20160602" "20160511" "20160617" "20160223" "20160307"
##  [97] "20160102" "20160127" "20160210" "20160224" "20160628" "20160618"
## [103] "20160513" "20160107" "20160122" "20160216" "20160313" "20160305"
## [109] "20160308" "20160510" "20160527" "20160622" "20160420" "20160531"
## [115] "20160302" "20160514" "20160319" "20160117" "20160506" "20160410"
## [121] "20160131" "20160120" "20160124" "20160430" "20160512" "20160623"
## [127] "20160518" "20160126" "20160225" "20160630" "20160202" "20160529"
## [133] "20160530" "20160619" "20160412" "20160401" "20160226" "20160414"
## [139] "20160229" "20160316" "20160104" "20160428" "20160125" "20160426"
## [145] "20160505" "20160423" "20160411" "20160317" "20160204" "20160227"
## [151] "20160625" "20160507" "20160213" "20160413" "20160113" "20160422"
## [157] "20160526" "20160429" "20160301" "20160421" "20160209" "20160220"
## [163] "20160212" "20160405" "20160403" "20160304" "20160201" "20160207"
## [169] "20160116" "20160418" "20160211" "20160402" "20160101" "20160111"
## [175] "20160311" "20160310" "20160603" "20160629" "20160419" "20160208"
## [181] "20160219" "20160330" "20160306"
```

```r
unique(data$Date_received)
```

```
##   [1] "null"     "20160528" "20160217" "20160319" "20160613" "20160516"
##   [7] "20160429" "20160129" "20160530" "20160519" "20160606" "20160207"
##  [13] "20160421" "20160130" "20160412" "20160518" "20160327" "20160127"
##  [19] "20160215" "20160524" "20160523" "20160515" "20160521" "20160114"
##  [25] "20160321" "20160426" "20160409" "20160326" "20160322" "20160131"
##  [31] "20160125" "20160602" "20160128" "20160605" "20160607" "20160324"
##  [37] "20160601" "20160126" "20160124" "20160123" "20160201" "20160522"
##  [43] "20160203" "20160417" "20160415" "20160202" "20160206" "20160218"
##  [49] "20160611" "20160329" "20160510" "20160302" "20160526" "20160318"
##  [55] "20160205" "20160411" "20160520" "20160527" "20160317" "20160213"
##  [61] "20160505" "20160402" "20160211" "20160405" "20160408" "20160323"
##  [67] "20160204" "20160112" "20160430" "20160525" "20160609" "20160403"
##  [73] "20160325" "20160413" "20160210" "20160610" "20160414" "20160401"
##  [79] "20160109" "20160328" "20160420" "20160422" "20160615" "20160120"
##  [85] "20160614" "20160107" "20160508" "20160608" "20160603" "20160425"
##  [91] "20160424" "20160305" "20160330" "20160511" "20160504" "20160223"
##  [97] "20160404" "20160416" "20160118" "20160303" "20160212" "20160423"
## [103] "20160308" "20160228" "20160418" "20160509" "20160501" "20160428"
## [109] "20160427" "20160229" "20160512" "20160506" "20160117" "20160514"
## [115] "20160407" "20160410" "20160314" "20160116" "20160503" "20160502"
## [121] "20160531" "20160316" "20160331" "20160517" "20160222" "20160101"
## [127] "20160306" "20160604" "20160214" "20160406" "20160121" "20160313"
## [133] "20160225" "20160220" "20160110" "20160301" "20160105" "20160122"
## [139] "20160104" "20160113" "20160108" "20160115" "20160513" "20160208"
## [145] "20160612" "20160419" "20160103" "20160312" "20160209" "20160529"
## [151] "20160119" "20160227" "20160315" "20160304" "20160216" "20160507"
## [157] "20160311" "20160320" "20160102" "20160106" "20160224" "20160219"
## [163] "20160111" "20160310" "20160307" "20160221" "20160226" "20160309"
```

```
#library(lubridate)

# Convert the Date and Date_received columns to date format
data$Date <- ymd(data$Date)
```

```
## Warning: 977900 failed to parse.
```

```
data$Date_received <- ymd(data$Date_received)
```

```
## Warning: 701602 failed to parse.
```

```
# Check the conversion results for Date and Date_received
unique(data$Date)
```

```
##   [1] "2016-02-17" NA           "2016-05-16" "2016-06-13" "2016-06-26"
##   [6] "2016-05-19" "2016-06-06" "2016-03-27" "2016-01-15" "2016-01-14"
##  [11] "2016-05-21" "2016-03-29" "2016-03-21" "2016-05-22" "2016-03-22"
##  [16] "2016-06-21" "2016-05-23" "2016-01-06" "2016-03-24" "2016-06-27"
##  [21] "2016-01-30" "2016-06-05" "2016-06-14" "2016-03-26" "2016-02-21"
##  [26] "2016-02-18" "2016-01-28" "2016-06-24" "2016-06-12" "2016-06-09"
##  [31] "2016-03-31" "2016-03-28" "2016-04-15" "2016-05-08" "2016-05-17"
##  [36] "2016-03-25" "2016-06-01" "2016-05-09" "2016-01-23" "2016-02-06"
##  [41] "2016-02-14" "2016-04-04" "2016-05-24" "2016-03-18" "2016-04-08"
##  [46] "2016-06-16" "2016-06-10" "2016-04-17" "2016-02-05" "2016-02-03"
##  [51] "2016-04-06" "2016-02-15" "2016-01-19" "2016-01-03" "2016-01-05"
##  [56] "2016-05-03" "2016-05-20" "2016-05-28" "2016-04-09" "2016-05-01"
##  [61] "2016-06-15" "2016-05-02" "2016-01-12" "2016-05-15" "2016-06-04"
##  [66] "2016-04-16" "2016-02-28" "2016-04-07" "2016-03-23" "2016-03-03"
##  [71] "2016-04-25" "2016-04-27" "2016-06-07" "2016-01-08" "2016-03-15"
##  [76] "2016-01-09" "2016-04-24" "2016-03-14" "2016-01-18" "2016-03-09"
##  [81] "2016-01-21" "2016-05-25" "2016-06-08" "2016-06-11" "2016-01-10"
##  [86] "2016-06-20" "2016-02-22" "2016-01-29" "2016-03-12" "2016-03-20"
##  [91] "2016-05-04" "2016-06-02" "2016-05-11" "2016-06-17" "2016-02-23"
##  [96] "2016-03-07" "2016-01-02" "2016-01-27" "2016-02-10" "2016-02-24"
## [101] "2016-06-28" "2016-06-18" "2016-05-13" "2016-01-07" "2016-01-22"
## [106] "2016-02-16" "2016-03-13" "2016-03-05" "2016-03-08" "2016-05-10"
## [111] "2016-05-27" "2016-06-22" "2016-04-20" "2016-05-31" "2016-03-02"
## [116] "2016-05-14" "2016-03-19" "2016-01-17" "2016-05-06" "2016-04-10"
## [121] "2016-01-31" "2016-01-20" "2016-01-24" "2016-04-30" "2016-05-12"
## [126] "2016-06-23" "2016-05-18" "2016-01-26" "2016-02-25" "2016-06-30"
## [131] "2016-02-02" "2016-05-29" "2016-05-30" "2016-06-19" "2016-04-12"
## [136] "2016-04-01" "2016-02-26" "2016-04-14" "2016-02-29" "2016-03-16"
## [141] "2016-01-04" "2016-04-28" "2016-01-25" "2016-04-26" "2016-05-05"
## [146] "2016-04-23" "2016-04-11" "2016-03-17" "2016-02-04" "2016-02-27"
## [151] "2016-06-25" "2016-05-07" "2016-02-13" "2016-04-13" "2016-01-13"
## [156] "2016-04-22" "2016-05-26" "2016-04-29" "2016-03-01" "2016-04-21"
## [161] "2016-02-09" "2016-02-20" "2016-02-12" "2016-04-05" "2016-04-03"
## [166] "2016-03-04" "2016-02-01" "2016-02-07" "2016-01-16" "2016-04-18"
## [171] "2016-02-11" "2016-04-02" "2016-01-01" "2016-01-11" "2016-03-11"
## [176] "2016-03-10" "2016-06-03" "2016-06-29" "2016-04-19" "2016-02-08"
## [181] "2016-02-19" "2016-03-30" "2016-03-06"
```

```r
unique(data$Date_received)
```

```
##   [1] NA           "2016-05-28" "2016-02-17" "2016-03-19" "2016-06-13"
##   [6] "2016-05-16" "2016-04-29" "2016-01-29" "2016-05-30" "2016-05-19"
##  [11] "2016-06-06" "2016-02-07" "2016-04-21" "2016-01-30" "2016-04-12"
##  [16] "2016-05-18" "2016-03-27" "2016-01-27" "2016-02-15" "2016-05-24"
##  [21] "2016-05-23" "2016-05-15" "2016-05-21" "2016-01-14" "2016-03-21"
##  [26] "2016-04-26" "2016-04-09" "2016-03-26" "2016-03-22" "2016-01-31"
##  [31] "2016-01-25" "2016-06-02" "2016-01-28" "2016-06-05" "2016-06-07"
##  [36] "2016-03-24" "2016-06-01" "2016-01-26" "2016-01-24" "2016-01-23"
##  [41] "2016-02-01" "2016-05-22" "2016-02-03" "2016-04-17" "2016-04-15"
##  [46] "2016-02-02" "2016-02-06" "2016-02-18" "2016-06-11" "2016-03-29"
##  [51] "2016-05-10" "2016-03-02" "2016-05-26" "2016-03-18" "2016-02-05"
##  [56] "2016-04-11" "2016-05-20" "2016-05-27" "2016-03-17" "2016-02-13"
##  [61] "2016-05-05" "2016-04-02" "2016-02-11" "2016-04-05" "2016-04-08"
##  [66] "2016-03-23" "2016-02-04" "2016-01-12" "2016-04-30" "2016-05-25"
##  [71] "2016-06-09" "2016-04-03" "2016-03-25" "2016-04-13" "2016-02-10"
##  [76] "2016-06-10" "2016-04-14" "2016-04-01" "2016-01-09" "2016-03-28"
##  [81] "2016-04-20" "2016-04-22" "2016-06-15" "2016-01-20" "2016-06-14"
##  [86] "2016-01-07" "2016-05-08" "2016-06-08" "2016-06-03" "2016-04-25"
##  [91] "2016-04-24" "2016-03-05" "2016-03-30" "2016-05-11" "2016-05-04"
##  [96] "2016-02-23" "2016-04-04" "2016-04-16" "2016-01-18" "2016-03-03"
## [101] "2016-02-12" "2016-04-23" "2016-03-08" "2016-02-28" "2016-04-18"
## [106] "2016-05-09" "2016-05-01" "2016-04-28" "2016-04-27" "2016-02-29"
## [111] "2016-05-12" "2016-05-06" "2016-01-17" "2016-05-14" "2016-04-07"
## [116] "2016-04-10" "2016-03-14" "2016-01-16" "2016-05-03" "2016-05-02"
## [121] "2016-05-31" "2016-03-16" "2016-03-31" "2016-05-17" "2016-02-22"
## [126] "2016-01-01" "2016-03-06" "2016-06-04" "2016-02-14" "2016-04-06"
## [131] "2016-01-21" "2016-03-13" "2016-02-25" "2016-02-20" "2016-01-10"
## [136] "2016-03-01" "2016-01-05" "2016-01-22" "2016-01-04" "2016-01-13"
## [141] "2016-01-08" "2016-01-15" "2016-05-13" "2016-02-08" "2016-06-12"
## [146] "2016-04-19" "2016-01-03" "2016-03-12" "2016-02-09" "2016-05-29"
## [151] "2016-01-19" "2016-02-27" "2016-03-15" "2016-03-04" "2016-02-16"
## [156] "2016-05-07" "2016-03-11" "2016-03-20" "2016-01-02" "2016-01-06"
## [161] "2016-02-24" "2016-02-19" "2016-01-11" "2016-03-10" "2016-03-07"
## [166] "2016-02-21" "2016-02-26" "2016-03-09"
```

```r
# Recheck if the 'Date' and 'Date_received' columns are in date format
class(data$Date)
```

```
## [1] "Date"
```

```r
class(data$Date_received)
```

```
## [1] "Date"
```

```r
# Replace "null" with NA in the Coupon_id column
data$Coupon_id <- ifelse(data$Coupon_id == "null", NA, data$Coupon_id)
#unique(data$Coupon_id)
table(is.na(data$Coupon_id))
```

```
##
##   FALSE    TRUE
## 1053282  701602
```

```r
# Check if both Coupon_id and Date_received columns are either both NA or both not NA
data$both_null <- is.na(data$Coupon_id) & is.na(data$Date_received)

# Count the rows where both columns are NA
sum(data$both_null)
```

```
## [1] 701602
```

```r
# Count the rows where neither column is NA
sum(!data$both_null)
```

```
## [1] 1053282
```

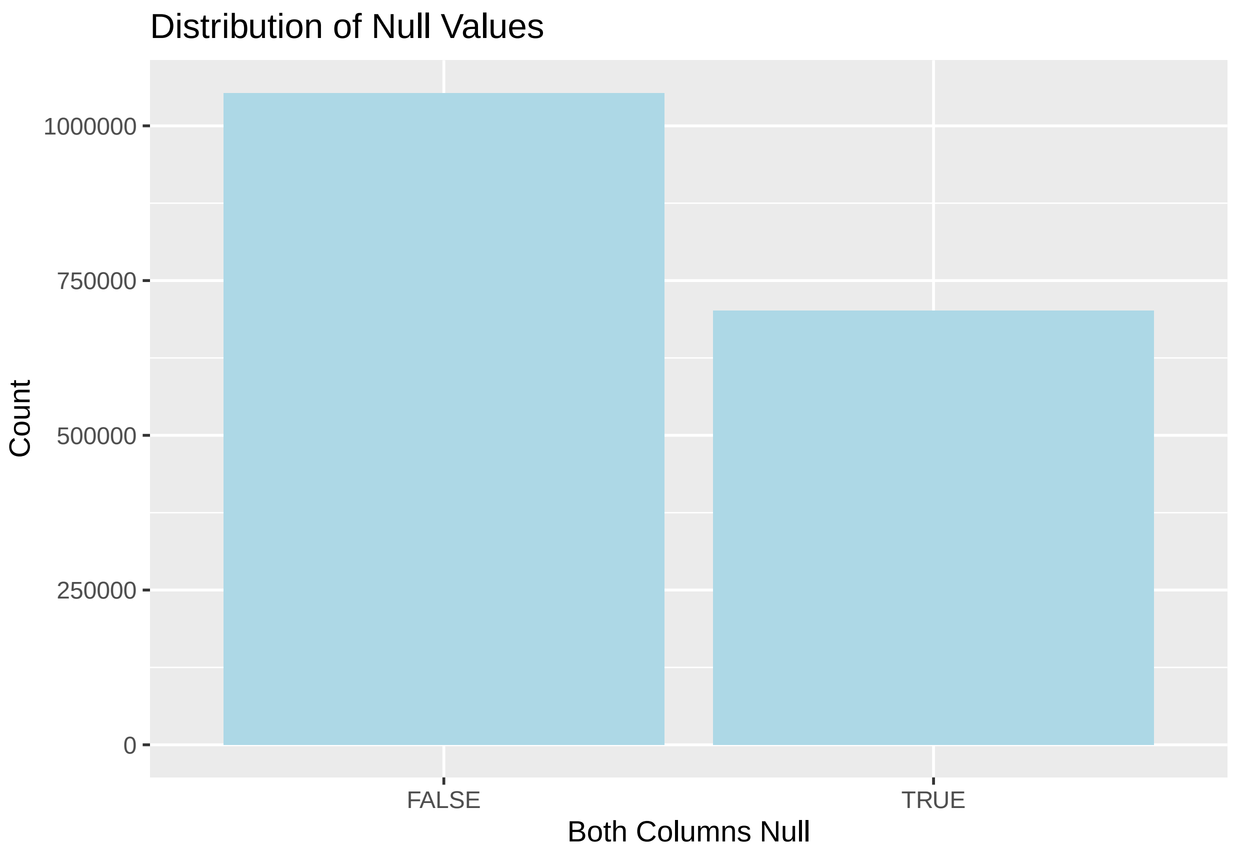
```r
# Display the distribution of rows where both columns are NA or not
table(data$both_null)
```

```
##
##   FALSE    TRUE
## 1053282  701602
```

```r
# Identify rows where one column is NA and the other is not
data$one_null <- is.na(data$Coupon_id) != is.na(data$Date_received)

# Count the rows where one column is NA and the other is not
sum(data$one_null)
```

```
## [1] 0
```

```r
# Plot a bar chart showing the distribution of rows where both columns are NA or not
ggplot(data, aes(x = both_null)) +
  geom_bar(fill = "lightblue") +
  labs(x = "Both Columns Null", y = "Count", title = "Distribution of Null Values")
```



```r
# Specify the date format for the Date and Date_received columns
data$Date <- as.Date(data$Date, format = "%Y-%m-%d")
data$Date_received <- as.Date(data$Date_received, format = "%Y-%m-%d")
```

Remove all instances where CouponId is empty, which means removing users who consumed directly without claiming a coupon. Focus on users who consumed with coupons, and examine the impact of coupons on user consumption.

```r
# Remove rows where Coupon_id is NA
data <- data[!is.na(data$Coupon_id), ]
view(head(data))
head(data)
```

```
## # A tibble: 6 x 9
##   User_id Merchant_id Coupon_id Discount_rate Distance Date_received Date
##     <dbl>       <dbl> <chr>     <chr>            <dbl> <date>        <date>
## 1 1439408        4663 11002     150:20               1 2016-05-28    NA
## 2 1439408        2632 8591      20:1                 0 2016-02-17    NA
## 3 1439408        2632 1078      20:1                 0 2016-03-19    NA
## 4 1439408        2632 8591      20:1                 0 2016-06-13    NA
## 5 1439408        2632 8591      20:1                 0 2016-05-16    2016-06-13
## 6 1832624        3381 7610      200:20               0 2016-04-29    NA
## # i 2 more variables: both_null <lgl>, one_null <lgl>
```

```r
# Display unique values in the Coupon_id column
#unique(data$Coupon_id)
table(is.na(data$Coupon_id))
```

```
##
##   FALSE
## 1053282
```

```r
# Display unique values in the Discount_rate column
unique(data$Discount_rate)
```

```
##  [1] "150:20"  "20:1"    "200:20"  "30:5"    "50:10"   "10:5"    "100:10"
##  [8] "200:30"  "20:5"    "30:10"   "50:5"    "150:10"  "100:30"  "200:50"
## [15] "100:50"  "300:30"  "50:20"   "0.9"     "10:1"    "30:1"    "0.95"
## [22] "100:5"   "5:1"     "100:20"  "0.8"     "50:1"    "200:10"  "300:20"
## [29] "100:1"   "150:30"  "300:50"  "20:10"   "0.85"    "0.6"     "150:50"
## [36] "0.75"    "0.5"     "200:5"   "0.7"     "30:20"   "300:10"  "0.2"
## [43] "50:30"   "200:100" "150:5"
```

```r
# Check the class (data type) of the Discount_rate column
class(data$Discount_rate)
```

```
## [1] "character"
```

The original Discount_rate has two types of values: ‘200:20’ means 20 off for every 200 spent, and ‘0.8’ means 20% off the original price. Based on the Discount_rate column, generate four new feature columns: consumption threshold (discount_threshold), discount amount (discount_amount), new discount rate (new_discount_rate), and discount strength (discount_strength).

```r
library(dplyr)
library(tidyr)


# Generate features
data <- data %>%
  mutate(Discount_rate = as.character(Discount_rate)) %>%  # Ensure Discount_rate is of character type
  separate(Discount_rate, c("discount_threshold", "discount_amount"), sep = ":", convert = TRUE, fill =
  mutate(
    discount_threshold = ifelse(is.na(discount_threshold), 0, discount_threshold),  # Fill NA with 0 in
    discount_amount = ifelse(is.na(discount_amount), 0, discount_amount),  # Fill NA with 0 in discount_
    new_discount_rate = ifelse(discount_threshold == 0, as.numeric(Discount_rate),
                               1 - discount_amount / discount_threshold),  # Calculate the new discount
    discount_strength = 1 / new_discount_rate  # Calculate the discount strength
  )
```

```r
# View the processed data
view(head(data))
head(data)
```

```
## # A tibble: 6 x 12
##   User_id Merchant_id Coupon_id discount_threshold discount_amount Distance
##      <dbl>       <dbl> <chr>                  <dbl>           <dbl>    <dbl>
## 1 1439408        4663 11002                    150              20        1
## 2 1439408        2632 8591                      20               1        0
## 3 1439408        2632 1078                      20               1        0
## 4 1439408        2632 8591                      20               1        0
## 5 1439408        2632 8591                      20               1        0
## 6 1832624        3381 7610                     200              20        0
## # i 6 more variables: Date_received <date>, Date <date>, both_null <lgl>,
## #   one_null <lgl>, new_discount_rate <dbl>, discount_strength <dbl>
```

```r
# Check the transformation results
glimpse(data)
```

```
## Rows: 1,053,282
## Columns: 12
## $ User_id            <dbl> 1439408, 1439408, 1439408, 1439408, 1439408, 183262~
## $ Merchant_id        <dbl> 4663, 2632, 2632, 2632, 2632, 3381, 3381, 450, 6459~
## $ Coupon_id          <chr> "11002", "8591", "1078", "8591", "8591", "7610", "1~
## $ discount_threshold <dbl> 150, 20, 20, 20, 20, 200, 200, 30, 20, 50, 20, 10, ~
## $ discount_amount    <dbl> 20, 1, 1, 1, 1, 20, 20, 5, 1, 10, 1, 5, 10, 30, 20,~
## $ Distance           <dbl> 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 2, 0, 10, 10, 2, 0~
## $ Date_received      <date> 2016-05-28, 2016-02-17, 2016-03-19, 2016-06-13, 20~
## $ Date               <date> NA, NA, NA, NA, 2016-06-13, NA, NA, NA, NA, NA, NA~
## $ both_null          <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FA~
## $ one_null           <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FA~
## $ new_discount_rate  <dbl> 0.8666667, 0.9500000, 0.9500000, 0.9500000, 0.95000~
## $ discount_strength  <dbl> 1.153846, 1.052632, 1.052632, 1.052632, 1.052632, 1~
```

```r
colnames(data)
```

```
##  [1] "User_id"            "Merchant_id"        "Coupon_id"         
##  [4] "discount_threshold" "discount_amount"    "Distance"          
##  [7] "Date_received"      "Date"               "both_null"         
## [10] "one_null"           "new_discount_rate"  "discount_strength"
```

```r
# Determine if the coupon was used
data <- data %>%
  mutate(
    Coupon_used = ifelse(!is.na(Coupon_id) & !is.na(Date), "Yes", "No")
  )

colSums(is.na(data))
```

```
##            User_id        Merchant_id          Coupon_id discount_threshold 
##                  0                  0                  0                  0 
##    discount_amount           Distance      Date_received               Date 
##                  0                  0                  0             977900 
##          both_null           one_null  new_discount_rate  discount_strength 
##                  0                  0                  0                  0 
##        Coupon_used 
##                  0
```

```r
str(data)
```

```
## tibble [1,053,282 x 13] (S3: tbl_df/tbl/data.frame)
##  $ User_id           : num [1:1053282] 1439408 1439408 1439408 1439408 1439408 ...
##  $ Merchant_id       : num [1:1053282] 4663 2632 2632 2632 2632 ...
##  $ Coupon_id         : chr [1:1053282] "11002" "8591" "1078" "8591" ...
##  $ discount_threshold: num [1:1053282] 150 20 20 20 20 200 200 30 20 50 ...
##  $ discount_amount   : num [1:1053282] 20 1 1 1 1 20 20 5 1 10 ...
##  $ Distance          : num [1:1053282] 1 0 0 0 0 0 1 0 0 0 ...
##  $ Date_received     : Date[1:1053282], format: "2016-05-28" "2016-02-17" ...
##  $ Date              : Date[1:1053282], format: NA NA ...
##  $ both_null         : logi [1:1053282] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ one_null          : logi [1:1053282] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ new_discount_rate : num [1:1053282] 0.867 0.95 0.95 0.95 0.95 ...
##  $ discount_strength : num [1:1053282] 1.15 1.05 1.05 1.05 1.05 ...
##  $ Coupon_used       : chr [1:1053282] "No" "No" "No" "No" ...
```

```r
# Check the ratio of positive and negative samples
table(data$Coupon_used)
```

```
##
##     No    Yes
## 977900  75382
```

```r
# Select features
X <- data[, c("discount_threshold", "discount_amount", "Distance", "new_discount_rate", "discount_streng

# Select the target variable
y <- factor(data$Coupon_used)

head(X)
```

```
## # A tibble: 6 x 5
##   discount_threshold discount_amount Distance new_discount_rate
##                <dbl>           <dbl>    <dbl>             <dbl>
## 1                150              20        1             0.867
## 2                 20               1        0             0.95
## 3                 20               1        0             0.95
## 4                 20               1        0             0.95
## 5                 20               1        0             0.95
## 6                200              20        0             0.9
## # i 1 more variable: discount_strength <dbl>
```

```r
head(y)
```

```
## [1] No  No  No  No  Yes No
## Levels: No Yes
```

```r
glimpse(y)
```

```
##  Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 1 1 1 ...
```

```r
glimpse(X)
```

```
## Rows: 1,053,282
## Columns: 5
## $ discount_threshold <dbl> 150, 20, 20, 20, 20, 200, 200, 30, 20, 50, 20, 10, ~
## $ discount_amount    <dbl> 20, 1, 1, 1, 1, 20, 20, 5, 1, 10, 1, 5, 10, 30, 20,~
## $ Distance           <dbl> 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 2, 0, 10, 10, 2, 0~
## $ new_discount_rate  <dbl> 0.8666667, 0.9500000, 0.9500000, 0.9500000, 0.95000~
## $ discount_strength  <dbl> 1.153846, 1.052632, 1.052632, 1.052632, 1.052632, 1~
```

```r
table(y)
```

```
## y
##     No    Yes
## 977900  75382
```

```r
summary(X)
```

```
##  discount_threshold discount_amount     Distance      new_discount_rate
##  Min.   :  0.20     Min.   :  0.00   Min.   : 0.000   Min.   :0.3333
##  1st Qu.: 30.00     1st Qu.:  5.00   1st Qu.: 0.000   1st Qu.:0.8333
##  Median : 50.00     Median :  5.00   Median : 1.000   Median :0.8500
##  Mean   : 78.47     Mean   : 10.37   Mean   : 2.904   Mean   :0.8471
##  3rd Qu.:100.00     3rd Qu.: 10.00   3rd Qu.: 5.000   3rd Qu.:0.9000
##  Max.   :300.00     Max.   :100.00   Max.   :10.000   Max.   :1.0000
##  discount_strength
##  Min.   :1.000
##  1st Qu.:1.111
##  Median :1.176
##  Mean   :1.199
##  3rd Qu.:1.200
##  Max.   :3.000
```

```r
cor(X)
```

```
##                    discount_threshold discount_amount      Distance
## discount_threshold          1.0000000       0.8274816  0.1483796738
## discount_amount             0.8274816       1.0000000  0.1758509221
## Distance                    0.1483797       0.1758509  1.0000000000
## new_discount_rate           0.2421811      -0.1472984 -0.0064179239
## discount_strength          -0.2400951       0.1093832  0.0009436533
##                    new_discount_rate discount_strength
## discount_threshold       0.242181119     -0.2400951473
## discount_amount         -0.147298352      0.1093832064
## Distance                -0.006417924      0.0009436533
## new_discount_rate        1.000000000     -0.9725801949
## discount_strength       -0.972580195      1.0000000000
```

```r
# Load machine learning packages
library(caret)
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```
library(ggplot2)
```

```
# Data splitting
set.seed(42)
train_index <- createDataPartition(y, p = 0.75, list = FALSE)
X_train <- X[train_index, ]
X_test <- X[-train_index, ]
y_train <- y[train_index]
y_test <- y[-train_index]
```

```
# Use the ROSE package for a combination of oversampling and undersampling
balanced_data <- ROSE(Coupon_used ~ ., data = cbind(X_train, Coupon_used = y_train), seed = 42)$data
X_resampled <- balanced_data[, !(names(balanced_data) %in% c("Coupon_used"))]
y_resampled <- balanced_data$Coupon_used
```

Logistic regression model

```
# Convert Coupon_used to factor and specify the reference level
balanced_data$Coupon_used <- factor(balanced_data$Coupon_used, levels = c("No", "Yes"))
```

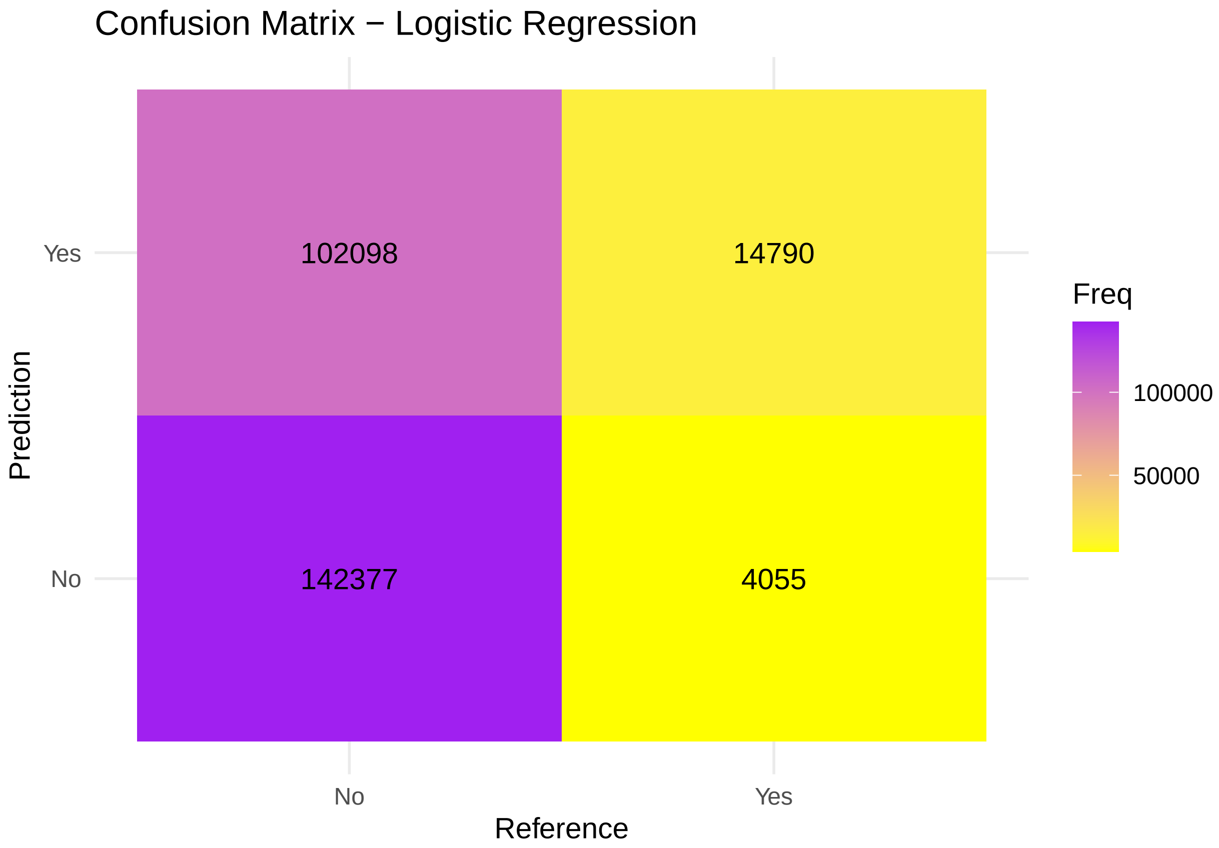
```
# Train the logistic regression model
log_reg <- glm(Coupon_used ~ ., data = balanced_data, family = binomial)

# Make predictions on the test set
y_pred_log <- predict(log_reg, newdata = X_test, type = "response")

# Adjust the threshold according to specific needs, here using 0.5 as an example
y_pred_log_class <- ifelse(y_pred_log > 0.5, "Yes", "No")
```

```
# Evaluate the logistic regression model performance
# Use the confusionMatrix function from the caret package for comprehensive evaluation
confusion_matrix <- confusionMatrix(data = as.factor(y_pred_log_class), reference = as.factor(y_test),
print(confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     No    Yes
##        No  142377   4055
##        Yes 102098  14790
##
##                Accuracy : 0.5969
##                  95% CI : (0.595, 0.5987)
##     No Information Rate : 0.9284
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.108
```

```
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.78482
##             Specificity : 0.58238
##          Pos Pred Value : 0.12653
##          Neg Pred Value : 0.97231
##              Prevalence : 0.07157
##          Detection Rate : 0.05617
##    Detection Prevalence : 0.44390
##       Balanced Accuracy : 0.68360
##
##        'Positive' Class : Yes
##
```

```
# Visualize the confusion matrix of logistic regression
cm_log <- confusionMatrix(factor(y_pred_log_class), factor(y_test))
ggplot(data = as.data.frame(cm_log$table),
       aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq))) +
  scale_fill_gradient(low = "yellow", high = "purple") +
  theme_minimal() +
  labs(title = "Confusion Matrix - Logistic Regression")
```



Random Forest model

```r
# Load the randomForest library
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
##    'randomForest'
```

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

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

```r
# Train a Random Forest model
rf_model <- randomForest(
  x = X_resampled,
  y = as.factor(y_resampled),
  ntree = 50,
  mtry = sqrt(ncol(X_resampled)),
  importance = TRUE
)
```

```r
# Predict on the test set using the trained Random Forest model
rf_pred <- predict(rf_model, newdata = X_test)
```

```r
# Evaluate the model performance using a confusion matrix
confusionMatrix(as.factor(rf_pred), as.factor(y_test))
```

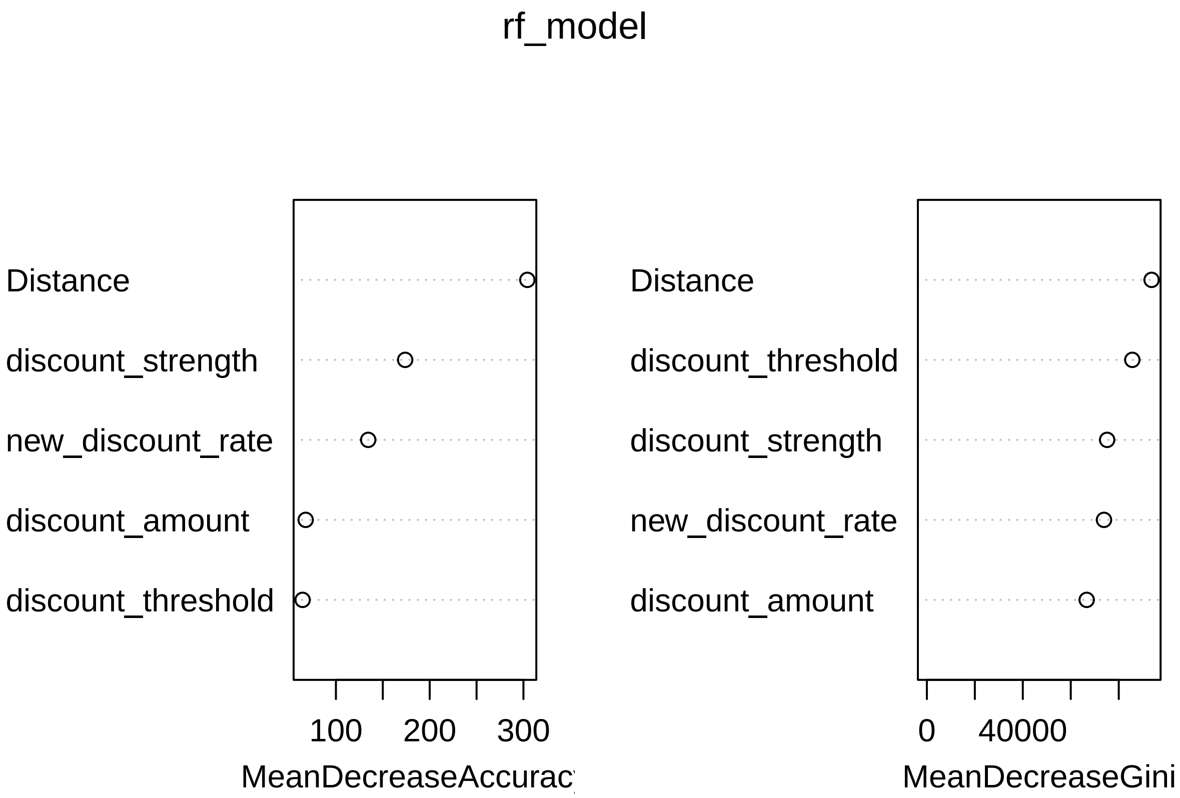
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     No    Yes
##        No  180004   6845
##        Yes  64471  12000
##
##                Accuracy : 0.7292
##                  95% CI : (0.7275, 0.7309)
##     No Information Rate : 0.9284
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1547
##
##  Mcnemar's Test P-Value : <2e-16
```

```
##
##             Sensitivity : 0.7363
##             Specificity : 0.6368
##          Pos Pred Value : 0.9634
##          Neg Pred Value : 0.1569
##              Prevalence : 0.9284
##          Detection Rate : 0.6836
##    Detection Prevalence : 0.7096
##       Balanced Accuracy : 0.6865
##
##        'Positive' Class : No
##
```

```
# View feature importance from the Random Forest model
importance(rf_model)
```

```
##                           No        Yes MeanDecreaseAccuracy MeanDecreaseGini
## discount_threshold  24.13125   24.43364             64.45430         85652.30
## discount_amount     40.52572   30.37855             67.74025         66631.93
## Distance           216.54919  207.97572            304.14335         93696.05
## new_discount_rate  206.22484 -202.10811            134.44259         73845.17
## discount_strength  277.54103 -250.02162            173.84578         75155.06
```

```
varImpPlot(rf_model)
```



```r
# Install and load the xgboost package
#install.packages("xgboost")
library(xgboost)
```

```
##
##    'xgboost'

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

```r
# Define model training control parameters
ctrl <- trainControl(
  method = "cv",
  number = 5,              # 5-fold cross-validation
  search = "grid",
  verboseIter = TRUE,
  allowParallel = TRUE
)
```

```r
# Define the grid of hyperparameters for XGBoost
xgb_grid <- expand.grid(
  nrounds = c(50, 100),
  max_depth = c(3, 6),
  eta = c(0.1, 0.3),
  gamma = 0,
  colsample_bytree = 0.8,
  min_child_weight = 1,
  subsample = 0.8
)
```

```r
# Train the XGBoost model using the train function
xgb_model <- train(
  x = as.matrix(X_resampled),
  y = y_resampled,
  method = "xgbTree",
  trControl = ctrl,
  tuneGrid = xgb_grid,
  objective = "binary:logistic",
  eval_metric = "auc",
  verbose = FALSE,
  nthread = parallel::detectCores() - 1
)
```

```
## + Fold1: eta=0.1, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:17:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold1: eta=0.1, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold1: eta=0.1, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:18:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold1: eta=0.1, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold1: eta=0.3, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:18:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold1: eta=0.3, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold1: eta=0.3, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:19:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold1: eta=0.3, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold2: eta=0.1, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:19:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold2: eta=0.1, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold2: eta=0.1, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:20:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold2: eta=0.1, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold2: eta=0.3, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:20:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold2: eta=0.3, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold2: eta=0.3, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.
## [16:21:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold2: eta=0.3, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold3: eta=0.1, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:21:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold3: eta=0.1, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold3: eta=0.1, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:22:05] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold3: eta=0.1, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold3: eta=0.3, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:22:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold3: eta=0.3, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold3: eta=0.3, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:23:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold3: eta=0.3, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold4: eta=0.1, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:23:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold4: eta=0.1, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold4: eta=0.1, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:23:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold4: eta=0.1, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold4: eta=0.3, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:24:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold4: eta=0.3, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold4: eta=0.3, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:24:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold4: eta=0.3, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold5: eta=0.1, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:25:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold5: eta=0.1, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold5: eta=0.1, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:26:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold5: eta=0.1, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold5: eta=0.3, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:26:44] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold5: eta=0.3, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## + Fold5: eta=0.3, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.

## [16:27:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## - Fold5: eta=0.3, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
## Aggregating results
## Selecting tuning parameters
## Fitting nrounds = 100, max_depth = 6, eta = 0.3, gamma = 0, colsample_bytree = 0.8, min_child_weight

## Warning in check.booster.params(params, ...): The following parameters were provided multiple times:
##  objective
##   Only the last value for each of them will be used.
```

```r
# Print the trained model and best hyperparameters
print(xgb_model)
```

```
## eXtreme Gradient Boosting
##
## 789962 samples
##      5 predictor
##      2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 631970, 631970, 631970, 631969, 631969
## Resampling results across tuning parameters:
##
##   eta  max_depth  nrounds  Accuracy   Kappa
##   0.1  3           50      0.7148243  0.4296019
##   0.1  3          100      0.7223740  0.4447058
##   0.1  6           50      0.7332998  0.4665583
##   0.1  6          100      0.7404166  0.4808018
##   0.3  3           50      0.7281768  0.4563168
##   0.3  3          100      0.7370544  0.4740792
##   0.3  6           50      0.7423344  0.4846417
##   0.3  6          100      0.7440485  0.4880744
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##
## Tuning parameter 'min_child_weight' was held constant at a value of 1
##
## Tuning parameter 'subsample' was held constant at a value of 0.8
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were nrounds = 100, max_depth = 6, eta
##  = 0.3, gamma = 0, colsample_bytree = 0.8, min_child_weight = 1 and subsample
##  = 0.8.
```

```r
print(xgb_model$bestTune)
```

```
##   nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
## 8     100         6 0.3     0              0.8                1       0.8
```

```r
# Make predictions on the test data
xgb_pred <- predict(xgb_model, newdata = as.matrix(X_test))
levels(xgb_pred)
```

```
## [1] "No"  "Yes"
```

```r
# Optionally, convert predictions to numeric and display the first 10
xgb_pred_numeric <- predict(xgb_model, newdata = as.matrix(X_test))
head(xgb_pred_numeric, n = 10)
```

```
##  [1] No  No  No  No  No  Yes Yes No  Yes No
## Levels: No Yes
```

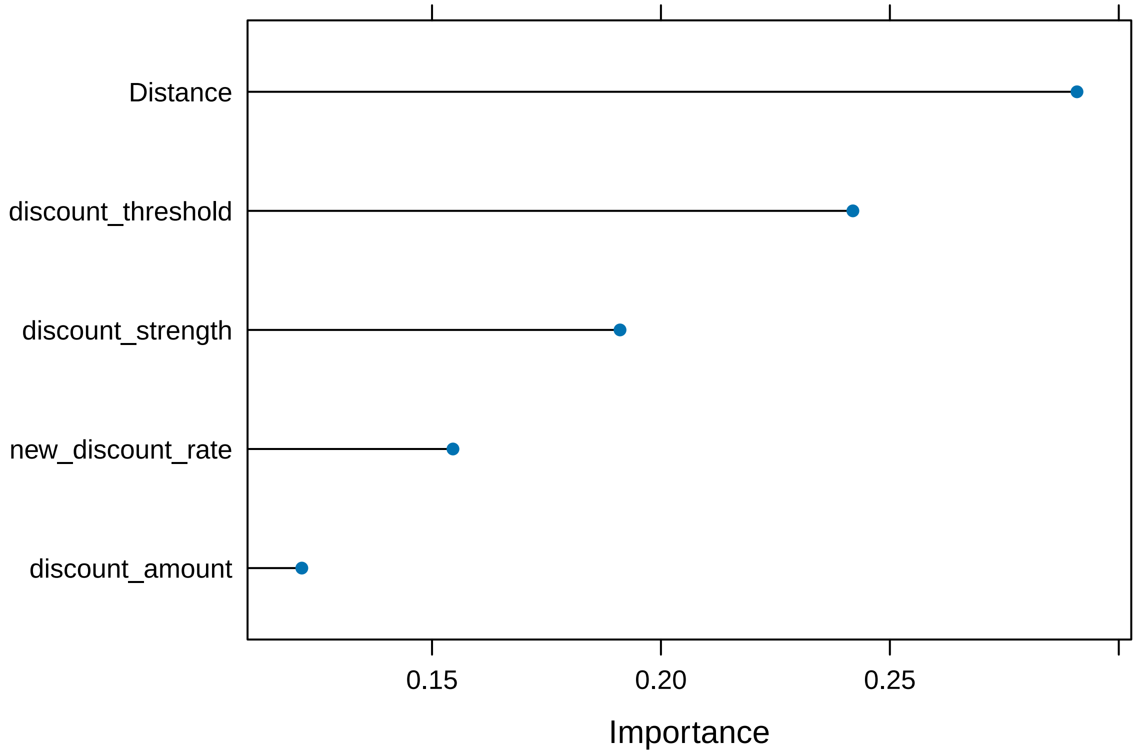
```r
# Convert numeric predictions to factors and calculate the confusion matrix
xgb_pred_factor <- factor(xgb_pred_numeric, levels = c("No", "Yes"))
cm <- table(Predicted = xgb_pred_factor, Actual = y_test)
print(cm)
```

```
##          Actual
## Predicted     No    Yes
##       No  173462   6101
##       Yes  71013  12744
```

```r
# Calculate and display feature importance
importance <- varImp(xgb_model, scale = FALSE)
print(importance)
```

```
## xgbTree variable importance
##
##                    Overall
## Distance            0.2909
## discount_threshold  0.2419
## discount_strength   0.1911
## new_discount_rate   0.1546
## discount_amount     0.1216
```

```r
plot(importance)
```



Conclusion

This report aims to predict the likelihood of a user visiting a store after receiving a coupon, using multiple models. The results indicate that the models can accurately identify negative samples (coupons not used), but struggle with identifying positive samples (coupons used). The primary reason for this is the substantial imbalance between positive and negative samples, leading to suboptimal performance when handling imbalanced data.

In practical applications, techniques such as sampling can be used to balance the positive and negative samples. If the original dataset exhibits a severe imbalance, it may be necessary to discard some data or seek higher-quality data sources. Additionally, due to computational performance constraints, I reduced the depth of trees in the random forest model and did not apply cross-validation, while in the XGBoost model, I also reduced the number of parameters and cross-validation rounds. If computational resources permit, increasing the complexity of these parameters could further enhance model performance.

Overall, while the XGBoost model showed some improvement in identifying positive samples compared to the random forest, the improvement was not significant. However, both models outperformed the logistic regression model.

Future

demonstrating better performance with more complex models.

...

Data Source: The dataset used in this report is from Alibaba's O2O service, available at Tianchi.