Fraudulent Transactions Project

Harry Chang

2023-04-16

```r
library(readxl)
sample = read_excel("sample_dataset_data_scientist.xlsx")
```

```r
# Load required libraries
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
library(ggplot2)
library(caret)
```

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

```r
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-4
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.1

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

##
## Attaching package: 'randomForest'

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

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

```r
library(cluster)
library(fastDummies)
```

Exploratory Data Analysis

```r
library(dplyr)
# Calculate total transactions and flagged transactions per user
user_transactions <- sample %>%
  group_by(User_Id) %>%
  summarise(Total_Transactions = n(),
            Flagged_Transactions = sum(Flag))
```

```r
# Calculate the proportion of flagged transactions
user_transactions <- user_transactions %>%
  mutate(Proportion_Flagged = Flagged_Transactions / Total_Transactions) %>%
  filter(Flagged_Transactions > 0) # Keep only users with at least one flagged transaction

user_transactions
```

```
## # A tibble: 623 x 4
##    User_Id                    Total_Transacti~ Flagged_Transac~ Proportion_Flag~
##    <chr>                                 <int>            <dbl>            <dbl>
##  1 0044882a8beef484ff200161b~                8                8            1
##  2 00c32260b96f2baeae831d024~                9                3            0.333
##  3 00cb00057d78adfede9636226~                3                2            0.667
##  4 0201de4ff9af54c170d89cd54~                1                1            1
##  5 020ab591105a9c9989d43ae91~               24               23            0.958
##  6 023f150450d6f7bf13a1eadbc~                3                3            1
##  7 02924b7ebd9a43542332d4dfa~               10                9            0.9
##  8 0299d9a7313d63132cb1d9e65~                6                5            0.833
##  9 02a647c9d5a2e67d36e863a61~                4                3            0.75
## 10 02b9d1e4f04e22226a64e1406~                1                1            1
## # ... with 613 more rows
```

The above is a table that illustrates all users with at least one flagged transaction.

```
# Find the lowest proportion
lowest_proportion <- min(user_transactions$Proportion_Flagged)
lowest_proportion
```

```
## [1] 0.06818182
```

Among these users, the lowest proportion is 0.06818.

```
library(tidyr)

# Calculate missing values count and percentage
missing_values <- sample %>%
  gather(Feature, Value) %>% # Convert to long format
  mutate(Missing = is.na(Value)) %>% # Check if value is missing (NA)
  group_by(Feature) %>%
  summarise(Missing_Count = sum(Missing), # Count missing values
            Total_Count = n(), # Count total values
            Missing_Percentage = Missing_Count / Total_Count * 100) # Calculate percentage of missing v
```

```
## Warning: attributes are not identical across measure variables; they will be
## dropped
```

```
missing_values
```

```
## # A tibble: 19 x 4
##    Feature              Missing_Count Total_Count Missing_Percentage
##    <chr>                        <int>       <int>              <dbl>
##  1 Alpha2Code                       0      202389              0
##  2 ChannelType                      0      202389              0
##  3 ClientIP                         0      202389              0
##  4 CountryCode                      0      202389              0
##  5 Email_Id                    100721      202389             49.8
##  6 FirstEmailDate              102762      202389             50.8
##  7 FirstTransactionDate             0      202389              0
##  8 Flag                             0      202389              0
##  9 GeoIpCountry                   380      202389              0.188
## 10 ItemName                         0      202389              0
## 11 Merchant_Id                      0      202389              0
## 12 PaymentChannel                   0      202389              0
## 13 Price                            0      202389              0
## 14 Transaction_Id                   0      202389              0
## 15 TxnCompleteTime                  0      202389              0
## 16 TxnInitTime                      0      202389              0
## 17 UniquePaymentChannel             0      202389              0
## 18 UserAgent                        0      202389              0
## 19 User_Id                          0      202389              0
```

The above shows a table of all features, in order to tally how many missing values are there for each feature (if applicable). This is essential for preliminary data cleaning before exploratory data analysis is performed.

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
# Convert transaction dates to Date objects
sample$TxnInitTime <- ymd_hms(sample$TxnInitTime)
sample$FirstTransactionDate <- ymd(sample$FirstTransactionDate) %>% as.POSIXct() # Convert to datetime
```

```r
# Find the first fraudulent transaction for each user
first_fraud_transactions <- sample %>%
  filter(Flag == 1) %>%
  group_by(User_Id) %>%
  summarise(First_Fraud_TxnInitTime = min(TxnInitTime))
```

```r
# Calculate platform age at the time of first fraud transaction
platform_age <- first_fraud_transactions %>%
  left_join(sample %>% select(User_Id, FirstTransactionDate), by = "User_Id") %>%
  mutate(Platform_Age_Days = as.numeric(First_Fraud_TxnInitTime - FirstTransactionDate, units = "days"))
```

```
## Warning in left_join(., sample %>% select(User_Id, FirstTransactionDate), : Each row in `x` is expect
## i Row 1 of `x` matches multiple rows.
## i If multiple matches are expected, set `multiple = "all"` to silence this
##   warning.
```

```r
# Find the maximum platform age
max_platform_age <- max(platform_age$Platform_Age_Days, na.rm = TRUE)
max_platform_age
```

```
## [1] 840.2841
```

From the above, it appears that the maximum platform age of a user (calculated as the number of days from the first transaction of the user) at the time they had the first fraud transaction is ~840 days.

```r
# Calculate total transactions and fraudulent transactions per Merchant_Id and ItemName
merchant_item_transactions <- sample %>%
  group_by(Merchant_Id, ItemName) %>%
  summarise(Total_Transactions = n(),
            Fraudulent_Transactions = sum(Flag))
```

```
## `summarise()` has grouped output by 'Merchant_Id'. You can override using the
## `.groups` argument.
```

```r
# Calculate the probability of fraudulent transactions
merchant_item_transactions <- merchant_item_transactions %>%
  mutate(Fraud_Probability = Fraudulent_Transactions / Total_Transactions)
```

```r
# Find the Merchant_Id and ItemName with the highest probability of fraudulent transactions
highest_fraud_prob <- merchant_item_transactions %>%
  filter(Fraud_Probability == max(Fraud_Probability))

highest_fraud_prob
```

```
## # A tibble: 304 x 5
## # Groups:   Merchant_Id [259]
##    Merchant_Id ItemName Total_Transactions Fraudulent_Transact~ Fraud_Probabili~
##          <dbl> <chr>                 <int>                <dbl>            <dbl>
##  1        1206 Item 23                2312                   17          0.00735
##  2        1210 Item 23                 257                    0          0
##  3        1210 Item 35                   2                    0          0
##  4        1210 Item 90                   2                    0          0
##  5        1211 Item 23               17875                    6          0.000336
##  6        1213 Item 35                 122                    1          0.00820
##  7        1833 Item 23                1264                    0          0
##  8        2214 Item 49                  62                    0          0
##  9        2214 Item 94                   5                    0          0
## 10        2215 Item 49                  45                    0          0
## # ... with 294 more rows
```

Based on the above, we can see that the combination of Item 23 and Merchant_Id has the highest probability of fraudulent transactions, at 0.00735.

```r
# Convert transaction initiation and completion times to datetime objects
sample$TxnInitTime <- ymd_hms(sample$TxnInitTime)
sample$TxnCompleteTime <- ymd_hms(sample$TxnCompleteTime)
```

```r
# Filter for fraudulent transactions
fraudulent_transactions <- sample %>%
  filter(Flag == 1)
```

```r
# Calculate transaction time for each fraudulent transaction
fraudulent_transactions <- fraudulent_transactions %>%
  mutate(Transaction_Time = as.numeric(TxnCompleteTime - TxnInitTime, units = "secs"))
```

```r
# Compute the average transaction time across each payment channel
average_transaction_time <- fraudulent_transactions %>%
  group_by(PaymentChannel) %>%
  summarise(Average_Transaction_Time = mean(Transaction_Time, na.rm = TRUE))

average_transaction_time
```

```
## # A tibble: 7 x 2
##   PaymentChannel   Average_Transaction_Time
##   <chr>                               <dbl>
## 1 PaymentChannel 1                     50.6
## 2 PaymentChannel 2                    137.
## 3 PaymentChannel 4                    123.
## 4 PaymentChannel 5                     71.6
```

```
## 5 PaymentChannel 6                    120.
## 6 PaymentChannel 7                    112.
## 7 PaymentChannel 8                    184
```

The above shows the average transaction time for fraudulent transactions using the respective payment channels.

```r
# Count the unique Email_Id and unique ClientIP for each user
user_unique_counts <- sample %>%
  group_by(User_Id) %>%
  summarise(Unique_Email_Count = n_distinct(Email_Id),
            Unique_ClientIP_Count = n_distinct(ClientIP))
```

```r
# Find the maximum number of unique Email_Id and unique ClientIP among all users
max_unique_email <- max(user_unique_counts$Unique_Email_Count, na.rm = TRUE)
max_unique_clientip <- max(user_unique_counts$Unique_ClientIP_Count, na.rm = TRUE)
```

```r
max_unique_email
```

```
## [1] 9
```

```r
max_unique_clientip
```

```
## [1] 32
```

The above shows that the maximum number of unique email IDs and client IPs each user has is 9 and 32 respectively.

```r
# Calculate average price for fraudulent transactions
average_fraud_price <- sample %>%
  filter(Flag == 1) %>%
  summarise(Average_Fraud_Price = mean(Price, na.rm = TRUE))
```

```r
# Calculate average price for non-fraudulent transactions
average_nonfraud_price <- sample %>%
  filter(Flag == 0) %>%
  summarise(Average_NonFraud_Price = mean(Price, na.rm = TRUE))
```

```r
# Compute the difference between average prices
price_difference <- average_fraud_price$Average_Fraud_Price - average_nonfraud_price$Average_NonFraud_P

price_difference
```

```
## [1] 11042.63
```

The average price differnce between fraudulent and non-fraudulent transactions is $11042.63.

```r
# Calculate TPV for all transactions and fraudulent transactions per country
country_tpv <- sample %>%
  group_by(GeoIpCountry) %>%
  summarise(Total_Purchase_Volume = sum(Price, na.rm = TRUE),
            Fraud_Purchase_Volume = sum(Price[Flag == 1], na.rm = TRUE))
```

```r
# Compute the percentage of TPV flagged as fraud for each country
country_tpv <- country_tpv %>%
  mutate(Fraud_Percentage = (Fraud_Purchase_Volume / Total_Purchase_Volume) * 100)

country_tpv
```

```
## # A tibble: 40 x 4
##    GeoIpCountry Total_Purchase_Volume Fraud_Purchase_Volume Fraud_Percentage
##    <chr>                        <dbl>                 <dbl>            <dbl>
##  1 AE                          11339                      0                0
##  2 AU                           2723                      0                0
##  3 BD                           2570                      0                0
##  4 BH                          15425                      0                0
##  5 BR                            217.                     0                0
##  6 CA                           3320                      0                0
##  7 DE                          22035                      0                0
##  8 FR                           9349                      0                0
##  9 GB                           2360                      0                0
## 10 GH                             80                      0                0
## # ... with 30 more rows
```

The above tabulates the total purchase volumes and fraud percentages grouped by country.

```r
# Count the unique emails for each user
user_unique_emails <- sample %>%
  group_by(User_Id) %>%
  summarise(Unique_Email_Count = n_distinct(Email_Id))
```

```r
# Filter users with more than 3 unique emails and count the total number of such users
users_more_than_3_emails <- user_unique_emails %>%
  filter(Unique_Email_Count > 3) %>%
  nrow()
```

```r
# Calculate the total number of transactions and the number of fraudulent transactions per count of uni
email_count_fraud <- sample %>%
  left_join(user_unique_emails, by = "User_Id") %>%
  group_by(Unique_Email_Count) %>%
  summarise(Total_Transactions = n(),
            Fraudulent_Transactions = sum(Flag))
```

```r
# Compute the percentage of fraudulent transactions for each group
email_count_fraud <- email_count_fraud %>%
  mutate(Fraud_Percentage = (Fraudulent_Transactions / Total_Transactions) * 100)
```

```r
users_more_than_3_emails
```

```
## [1] 28
```

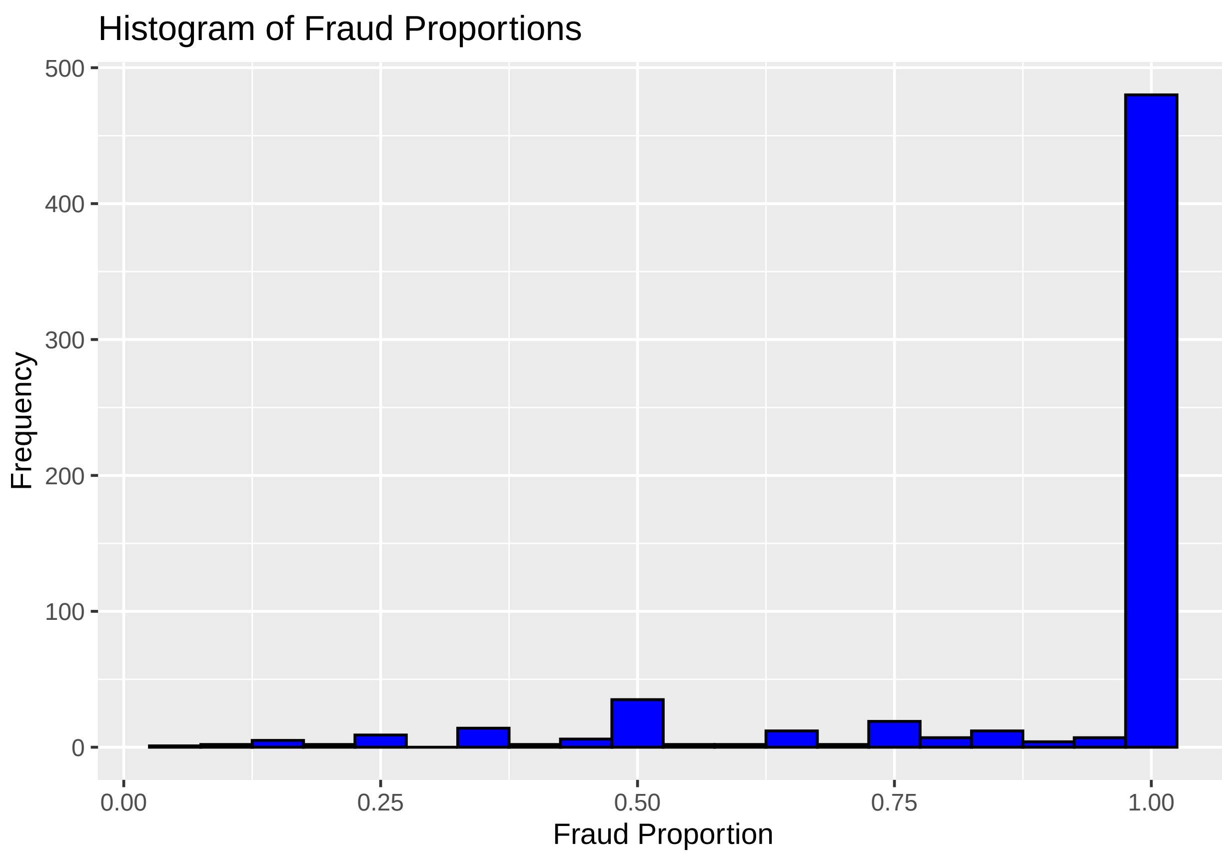
```r
email_count_fraud
```

```
## # A tibble: 8 x 4
##   Unique_Email_Count Total_Transactions Fraudulent_Transactions Fraud_Percentage
##                <int>              <int>                   <dbl>            <dbl>
## 1                  1             186360                    1980             1.06
## 2                  2              14670                     336             2.29
## 3                  3               1151                      59             5.13
## 4                  4                146                       0             0
## 5                  5                 21                       0             0
## 6                  6                 10                       9            90
## 7                  8                 16                       0             0
## 8                  9                 15                       5            33.3
```

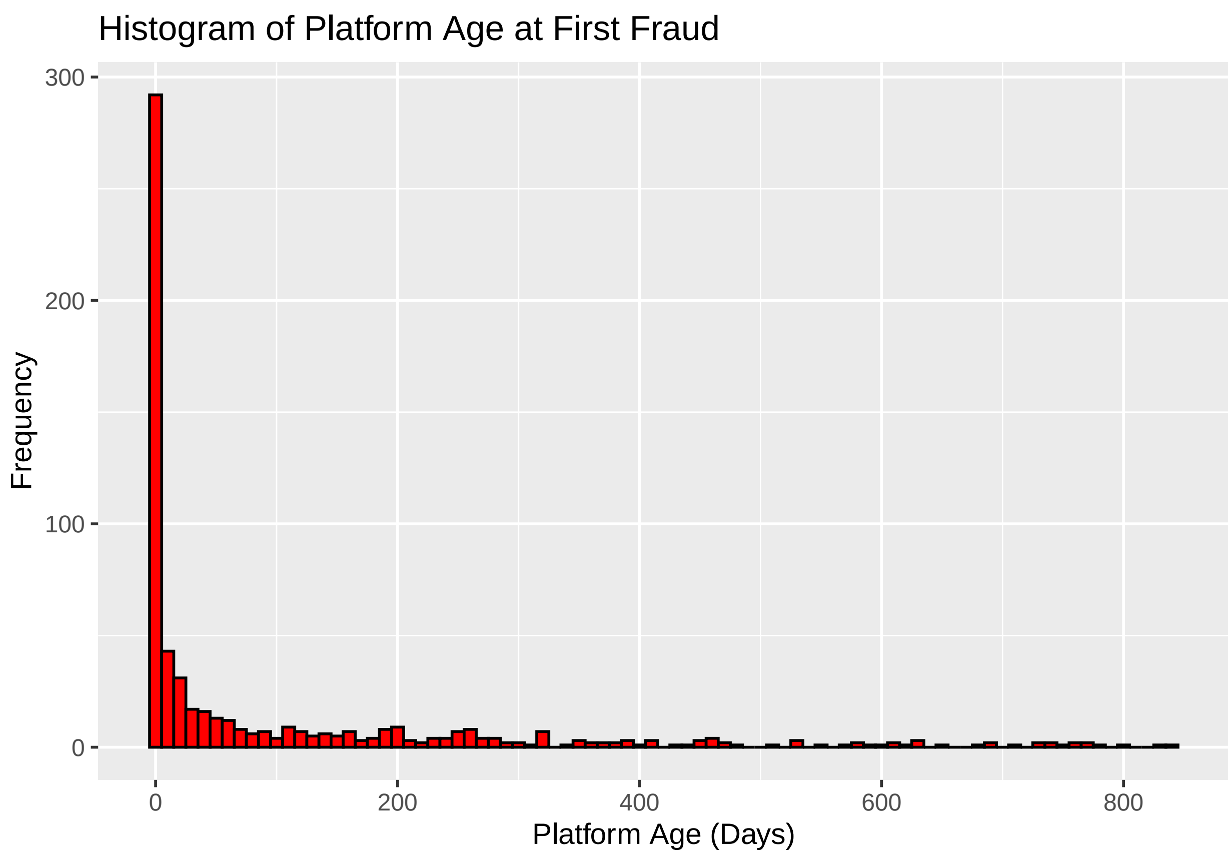
The above tells us that there are 28 users with more than 3 unique emails, and a table is also displayed, showcasing the percentage of fraudulent transactions, grouped by unique email count.

Visualizations

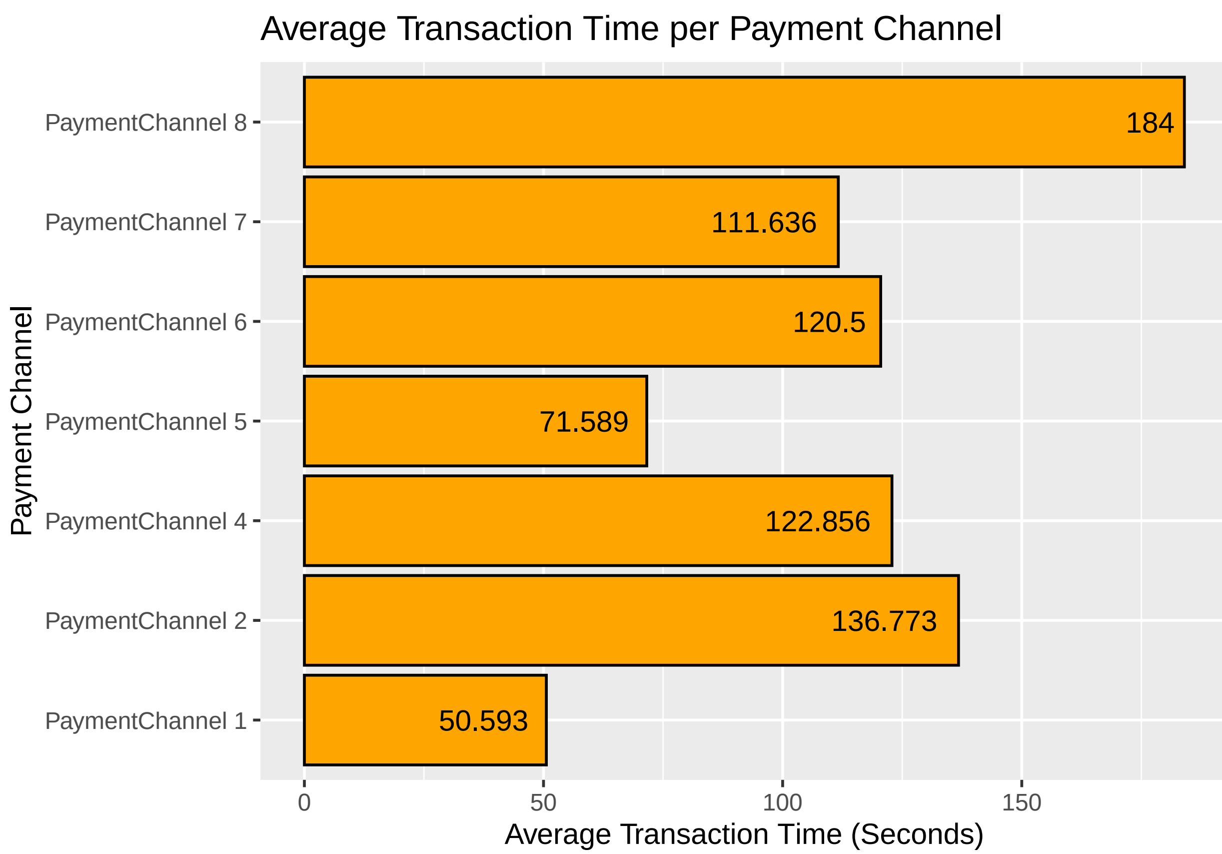
```r
# Q1: Proportion of flagged transactions per user with at least one flagged transaction
q1_data <- sample %>%
  group_by(User_Id) %>%
  summarise(Total_Transactions = n(),
            Fraud_Transactions = sum(Flag)) %>%
  filter(Fraud_Transactions > 0) %>%
  mutate(Fraud_Proportion = Fraud_Transactions / Total_Transactions)

q1_plot <- ggplot(q1_data, aes(x = Fraud_Proportion)) +
  geom_histogram(binwidth = 0.05, fill = "blue", color = "black") +
  labs(title = "Histogram of Fraud Proportions",
       x = "Fraud Proportion",
       y = "Frequency")

q1_plot
```



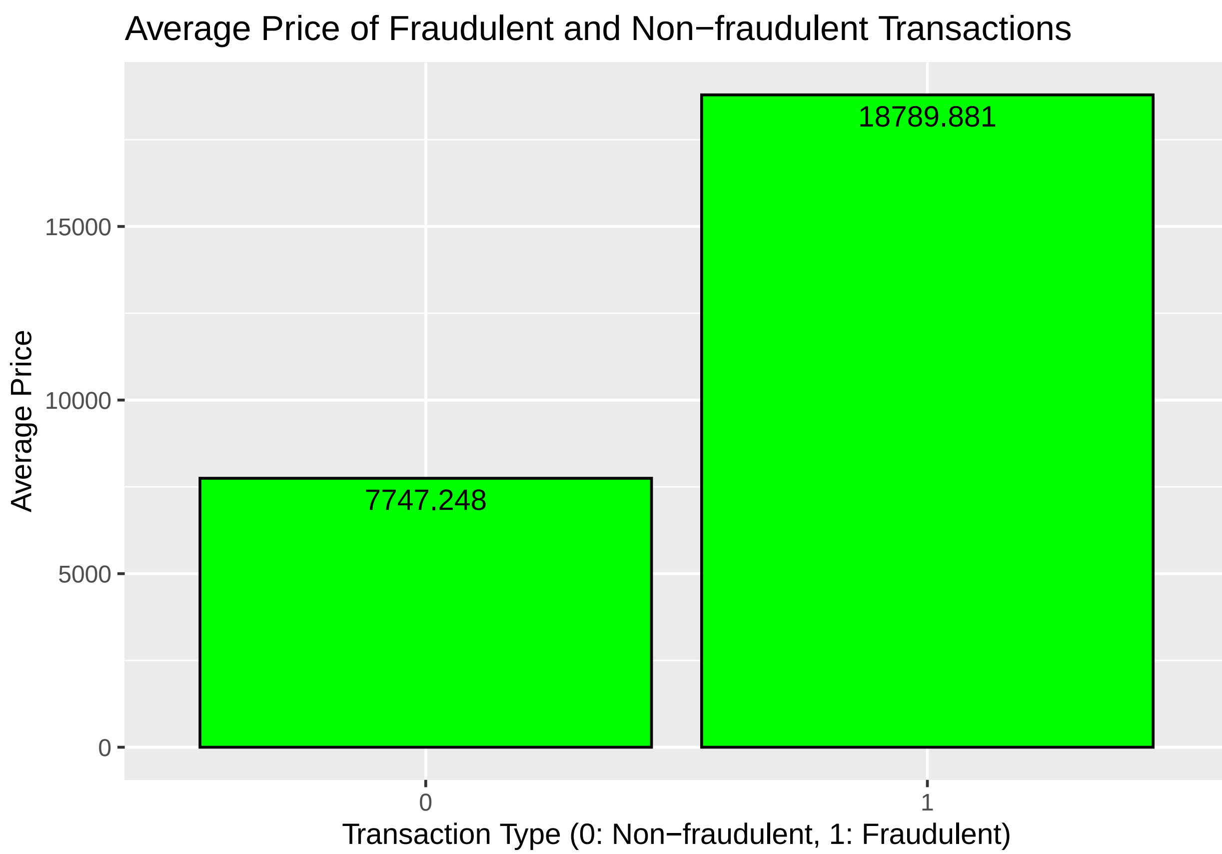

```r
# Q4: Maximum platform age of a user at the time they had the first fraud transaction
q4_data <- sample %>%
  filter(Flag == 1) %>%
  group_by(User_Id) %>%
  arrange(User_Id, TxnInitTime) %>%
  filter(row_number() == 1) %>%
  mutate(Platform_Age_Days = as.numeric(TxnInitTime - FirstTransactionDate, units = "days"))

q4_plot <- ggplot(q4_data, aes(x = Platform_Age_Days)) +
  geom_histogram(binwidth = 10, fill = "red", color = "black") +
  labs(title = "Histogram of Platform Age at First Fraud",
       x = "Platform Age (Days)",
       y = "Frequency")

q4_plot
```



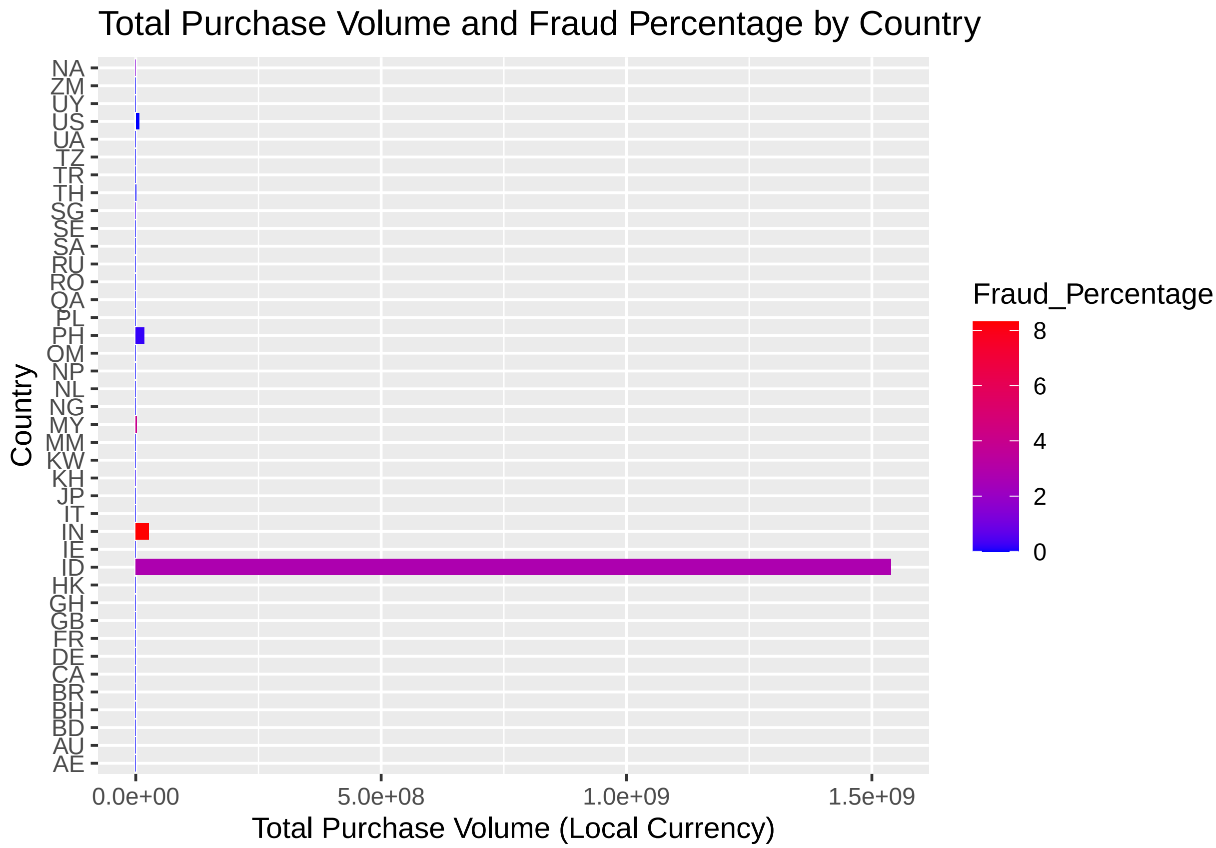

```r
# Q6: Average transaction time for fraudulent transactions across each payment channel
q6_data <- sample %>%
  filter(Flag == 1) %>%
  mutate(Transaction_Time = as.numeric(TxnCompleteTime - TxnInitTime, units = "secs")) %>%
  group_by(PaymentChannel) %>%
  summarise(Average_Transaction_Time = mean(Transaction_Time, na.rm = TRUE))

q6_plot <- ggplot(q6_data, aes(x = PaymentChannel, y = Average_Transaction_Time)) +
  geom_bar(stat = "identity", fill = "orange", color = "black") +
  geom_text(aes(label = round(Average_Transaction_Time,3), hjust = 1.2), colour = "black")+
  labs(title = "Average Transaction Time per Payment Channel",
       x = "Payment Channel",
       y = "Average Transaction Time (Seconds)") +
  coord_flip()

q6_plot
```




```r
# Q7: Difference between average price of fraudulent and non-fraudulent transactions
q7_data <- sample %>%
  group_by(Flag) %>%
  summarise(Average_Price = mean(Price, na.rm = TRUE))

q7_plot <- ggplot(q7_data, aes(x = factor(Flag), y = Average_Price)) +
  geom_bar(stat = "identity", fill = "green", color = "black") +
  geom_text(aes(label = round(Average_Price,3), vjust = 1.5), colour = "black")+
  labs(title = "Average Price of Fraudulent and Non-fraudulent Transactions",
       x = "Transaction Type (0: Non-fraudulent, 1: Fraudulent)",
       y = "Average Price")

q7_plot
```



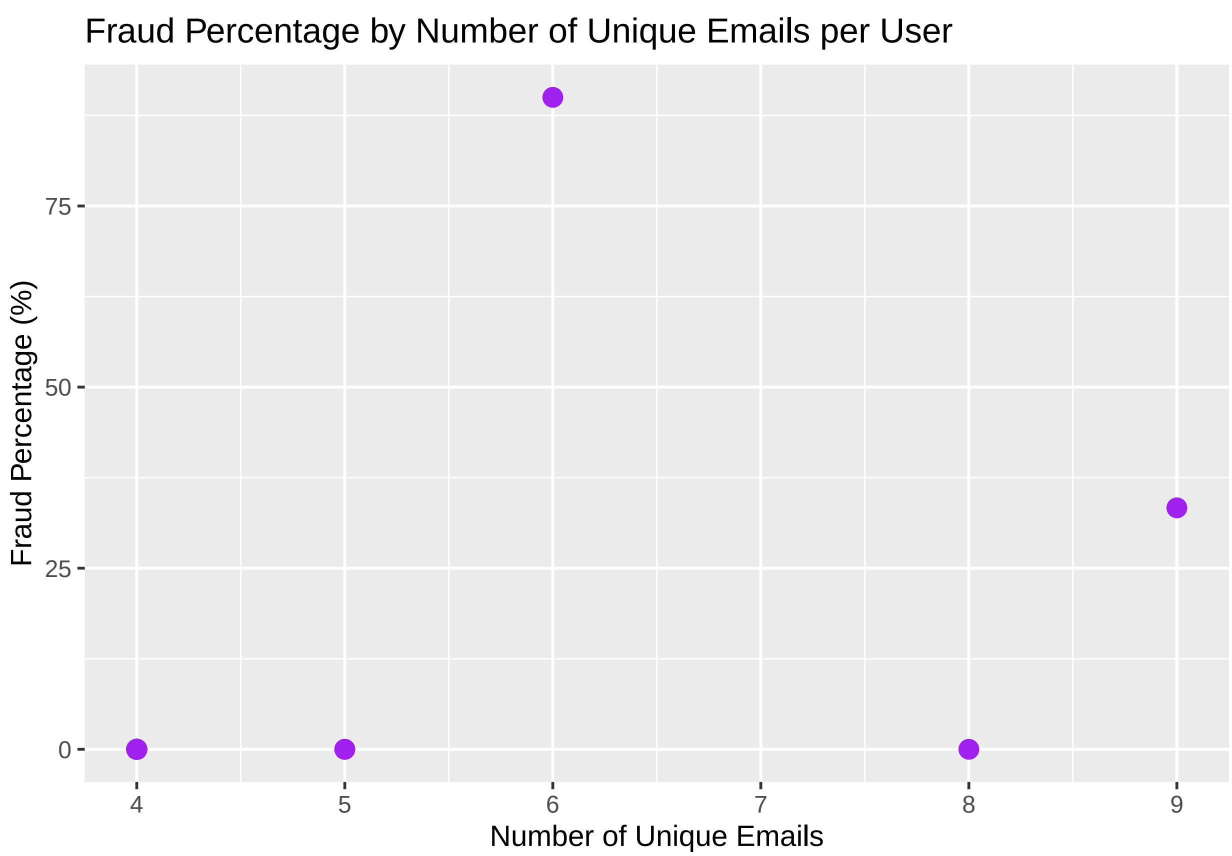

```r
# Q8: TPV in local currency grouped by country and percentage of TPV flagged as fraud
q8_data <-sample %>%
  group_by(GeoIpCountry) %>%
  summarise(Total_Purchase_Volume = sum(Price, na.rm = TRUE),
            Fraud_Purchase_Volume = sum(Price[Flag == 1], na.rm = TRUE)) %>%
  mutate(Fraud_Percentage = (Fraud_Purchase_Volume / Total_Purchase_Volume) * 100)

q8_plot <- ggplot(q8_data, aes(x = GeoIpCountry, y = Total_Purchase_Volume, fill = Fraud_Percentage)) +
  geom_col() +
  scale_fill_gradient(low = "blue", high = "red") +
  labs(title = "Total Purchase Volume and Fraud Percentage by Country",
       x = "Country",
       y = "Total Purchase Volume (Local Currency)") +
  coord_flip()

q8_plot
```




```
# Q9: Number of User_Id's with more than 3 unique emails and percentage of fraudulent transactions grou
q9_data <- sample %>%
  group_by(User_Id) %>%
  summarise(Unique_Emails = n_distinct(Email_Id),
            Total_Transactions = n(),
            Fraud_Transactions = sum(Flag)) %>%
  filter(Unique_Emails > 3) %>%
  mutate(Fraud_Percentage = (Fraud_Transactions / Total_Transactions) * 100)

q9_plot <- ggplot(q9_data, aes(x = Unique_Emails, y = Fraud_Percentage)) +
  geom_point(size = 3, color = "purple") +
  geom_smooth(method = "loess", se = FALSE, color = "darkblue") +
  labs(title = "Fraud Percentage by Number of Unique Emails per User",
       x = "Number of Unique Emails",
       y = "Fraud Percentage (%)")

q9_plot
```

```
## `geom_smooth()` using formula = 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : at 3.975

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : radius 0.000625
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : all data on boundary of neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at 3.975

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.025

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : zero-width neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 4

## Warning: Computation failed in `stat_smooth()`
## Caused by error in `predLoess()`:
## ! NA/NaN/Inf in foreign function call (arg 5)
```



Preprocessing data for machine learning methods

```
# Preprocess the data
sample <- sample %>%
  filter(!is.na(Flag)) %>%
  select(Flag, Price, Merchant_Id, PaymentChannel, ChannelType, UniquePaymentChannel, GeoIpCountry)

# Handle categorical variables using one-hot encoding
dummies_model <- dummyVars(~ ., data = sample)
sample <- data.frame(predict(dummies_model, newdata = sample))

# Split the dataset into training (80%) and testing (20%) sets
set.seed(42)
train_index <- createDataPartition(sample$Flag, p = 0.8, list = FALSE)
train_data <- sample[train_index, ]
test_data <- sample[-train_index, ]

# Normalize the data
pre_process <- preProcess(train_data, method = c("center", "scale"))
```

```
## Warning in preProcess.default(train_data, method = c("center", "scale")): These
## variables have zero variances: GeoIpCountryNL
```

```
train_data_norm <- predict(pre_process, train_data)
test_data_norm <- predict(pre_process, test_data)
```

Logistic Regression

```
# Fit logistic regression model
train_data_norm$Flag <- as.factor(train_data_norm$Flag)
logistic_model <- glm(Flag ~ ., data = train_data_norm, family = "binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# Predict on test data
test_data_norm$pred_logistic <- predict(logistic_model, test_data_norm, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

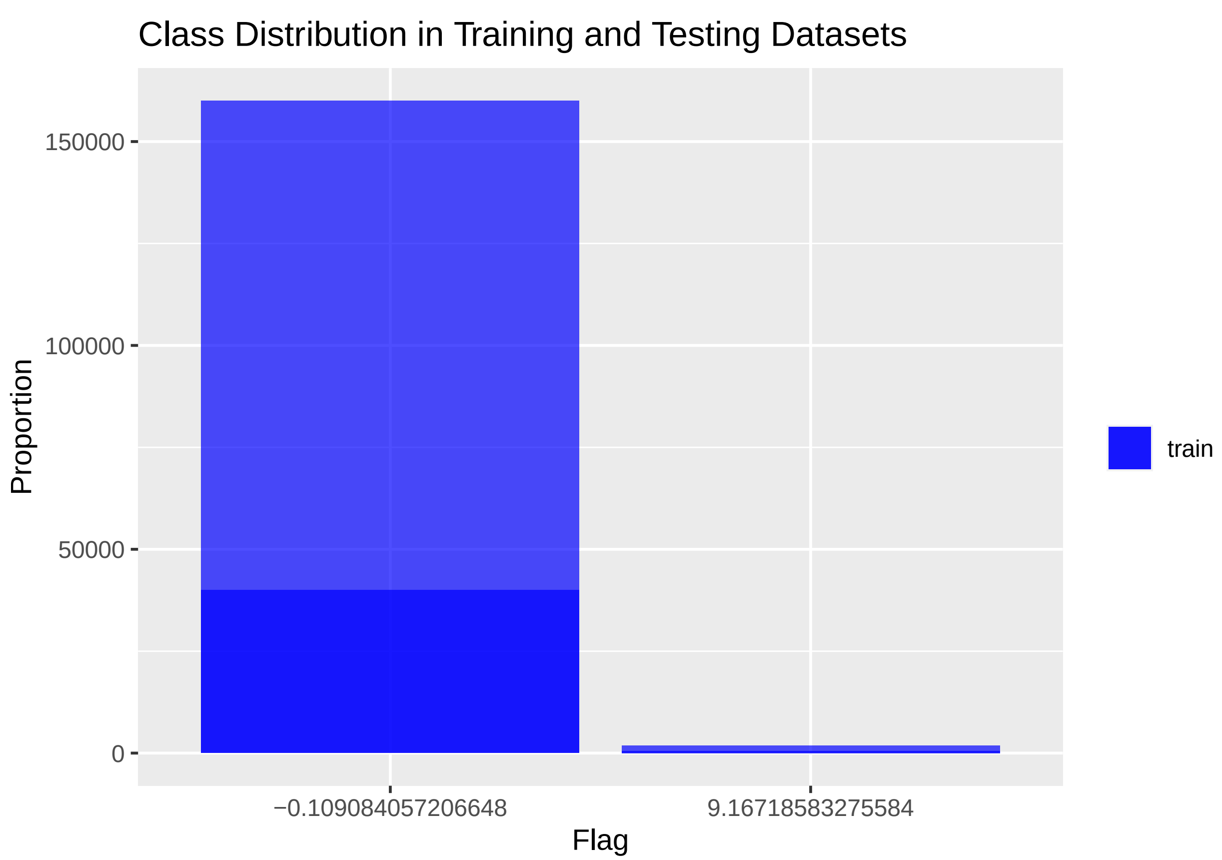
```
# Convert test_data_norm$Flag to factor
test_data_norm$Flag <- as.factor(test_data_norm$Flag)

# Calculate performance metrics only when there are overlapping levels
pred_labels <- as.factor(ifelse(test_data_norm$pred_logistic > 0.5, 1, 0))
if (any(levels(pred_labels) %in% levels(test_data_norm$Flag))) {
```

```
  conf_matrix_logistic <- confusionMatrix(pred_labels, test_data_norm$Flag)
  accuracy_logistic <- conf_matrix_logistic$overall["Accuracy"]
} else {
  cat("No overlapping levels between predicted and true labels.")
}
```

```
## No overlapping levels between predicted and true labels.
```

```
# Calculate class distribution in training and testing datasets
train_class_dist <- prop.table(table(train_data_norm$Flag))
test_class_dist <- prop.table(table(test_data_norm$Flag))
```

```
# Plot class distribution in training and testing datasets
ggplot(mapping = aes(x = factor(Flag), fill = "train")) +
  geom_bar(data = train_data_norm, alpha = 0.7, position = "identity") +
  geom_bar(data = test_data_norm, alpha = 0.7, position = "identity") +
  scale_fill_manual(name = "", values = c("train" = "blue", "test" = "red")) +
  labs(x = "Flag", y = "Proportion", title = "Class Distribution in Training and Testing Datasets")
```



Since there is a huge imbalance in the dataset, logistic regression might not be appropriate for use to predict fraudulent transactions in this case.