Lab 1 - Credit Card Fraud Data Wrangling and EDA, Stat 415, Spring 2024

Xiaolong Liu

March 2024

1 Introduction

In this report, our focus will be on analyzing and cleaning a dataset comprising credit card transactions. Our main objective is to derive insights into the factors that could predict credit card fraud. The status of fraud will be considered the outcome variable throughout this analysis, anticipating its role as the target variable in a predictive model to be developed subsequently.

1.1 Data Description

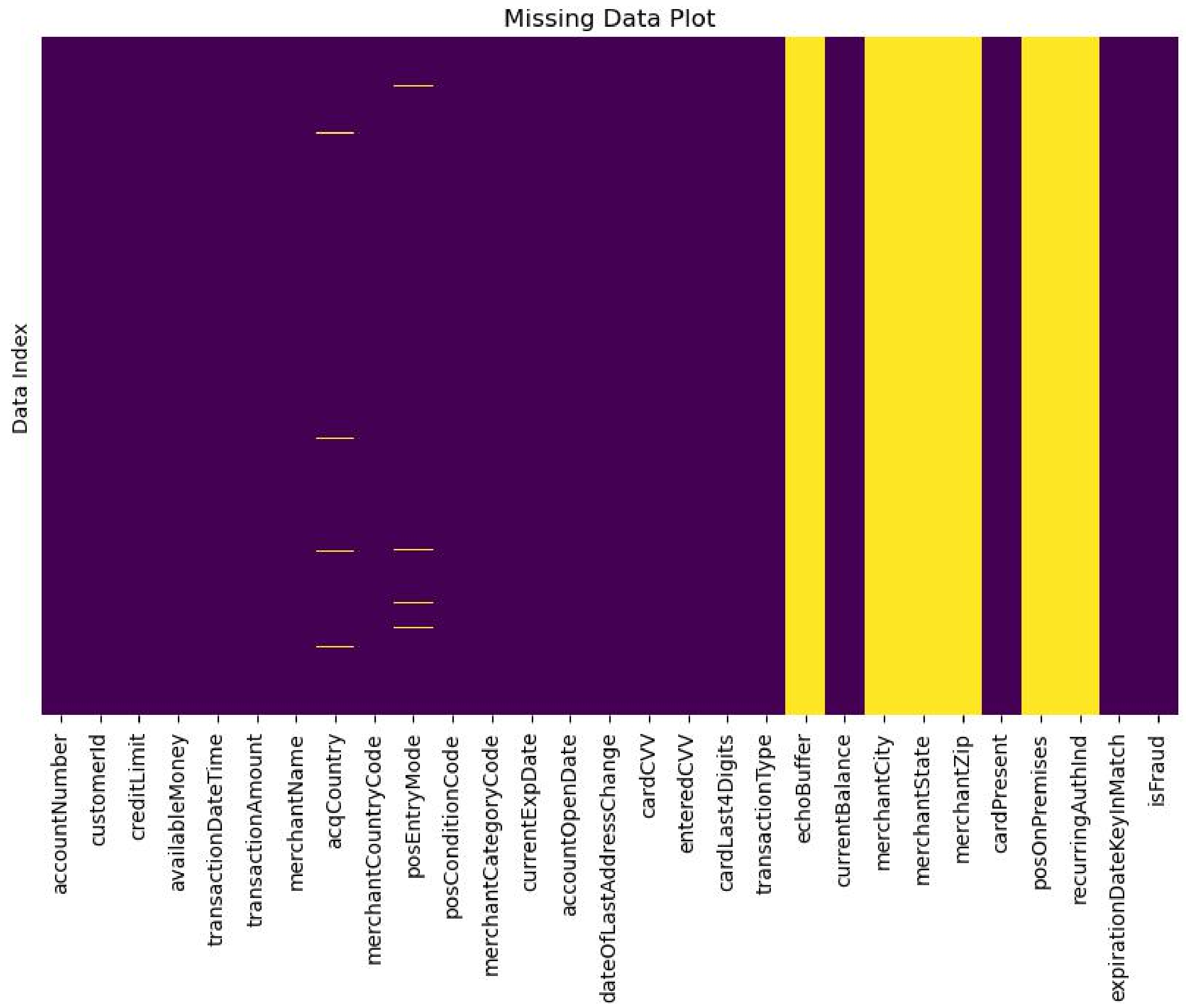
In total, our dataset comprises **786,363** observations and **29** variables. Below is the description of our variables:
accountNumber: a unique identifier for the customer account associated with the transaction
customerId: a unique identifier for the customer associated with the transaction
creditLimit: the maximum amount of credit available to the customer on their account
availableMoney: the amount of credit available to the customer at the time of the transaction
transactionDateTime: the date and time of the transaction
transactionAmount: the amount of the transaction
merchantName: the name of the merchant where the transaction took place
acqCountry: the country where the acquiring bank is located
merchantCountryCode: the country where the merchant is located
posEntryMode: the method used by the customer to enter their payment card information during the transaction
posConditionCode: the condition of the point-of-sale terminal at the time of the transaction
merchantCategoryCode: the category of the merchant where the transaction took place
currentExpDate: the expiration date of the customer's payment card
accountOpenDate: the date the customer's account was opened
dateOfLastAddressChange: the date the customer's address was last updated
cardCVV: the three-digit CVV code on the back of the customer's payment card
enteredCVV: the CVV code entered by the customer during the transaction
cardLast4Digits: the last four digits of the customer's payment card
transactionType: the type of transaction
echoBuffer: an internal variable used by the financial institution
currentBalance: the current balance on the customer's account
merchantCity: the city where the merchant is located
merchantState: the state where the merchant is located
merchantZip: the ZIP code where the merchant is located
cardPresent: whether or not the customer's payment card was present at the time of the transaction
posOnPremises: whether or not the transaction took place on the merchant's premises
recurringAuthInd: whether or not the transaction was a recurring payment

expirationDateKeyInMatch: whether or not the expiration date of the payment card was entered correctly during the transaction
isFraud: whether or not the transaction was fraudulent

2 Data Exploration

2.1 Preliminary Data Quality Checks

First, let's check the missingness of the dataset. Here's the missing plot for the dataset:



From the plot, we can observe that there is missingness in some variables. Specifically, data are entirely missing for the variables **echoBuffer**, **merchantCity**, **merchantState**, **merchantZip**, **posOnPremises**, and **recurringAuthInd**. Additionally, there is some missingness in the variables **acqCountry** with 0.58% missingness, **merchantCountryCode** with 0.09% missingness, **posEntryMode** with 0.52% missingness, **posConditionCode** with 0.052% missingness and **transactionType** with 0.09%.

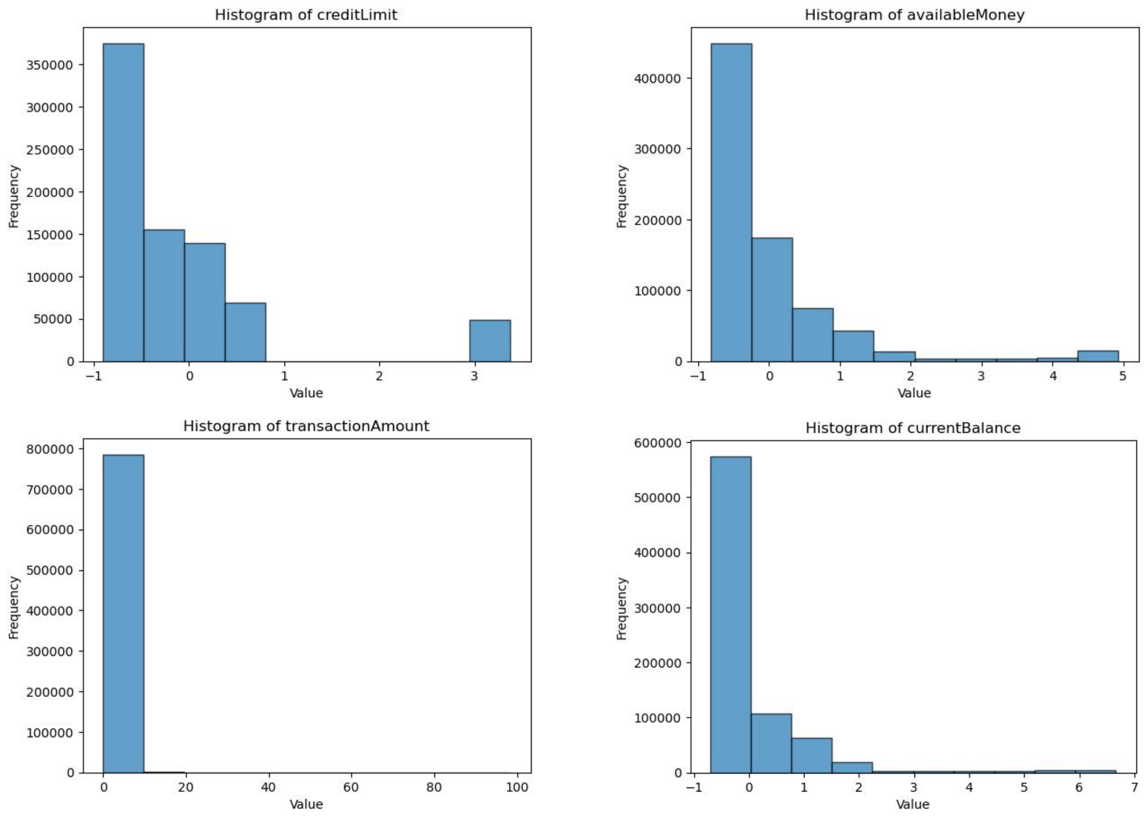
For columns with entirely missing data, we can directly drop them, as they contain no useful information. For columns with a small percentage of missingness, we can employ various techniques to impute the missing values. For instance, for missingness in categorical variables, we can assign a label such as 'Missing'. For other types of variables, we might consider using simulation or predictive methods to impute the missing values.

After addressing the issue of entirely missing data—specifically, by removing columns that are entirely missing—we proceeded to examine the dataset for duplicated columns. Our investigation revealed that two columns, **accountNumber** and **customerId**, are identical in content but differ in name. Given that these columns contain exactly the same information, we can eliminate one of them to avoid redundancy in the dataset.

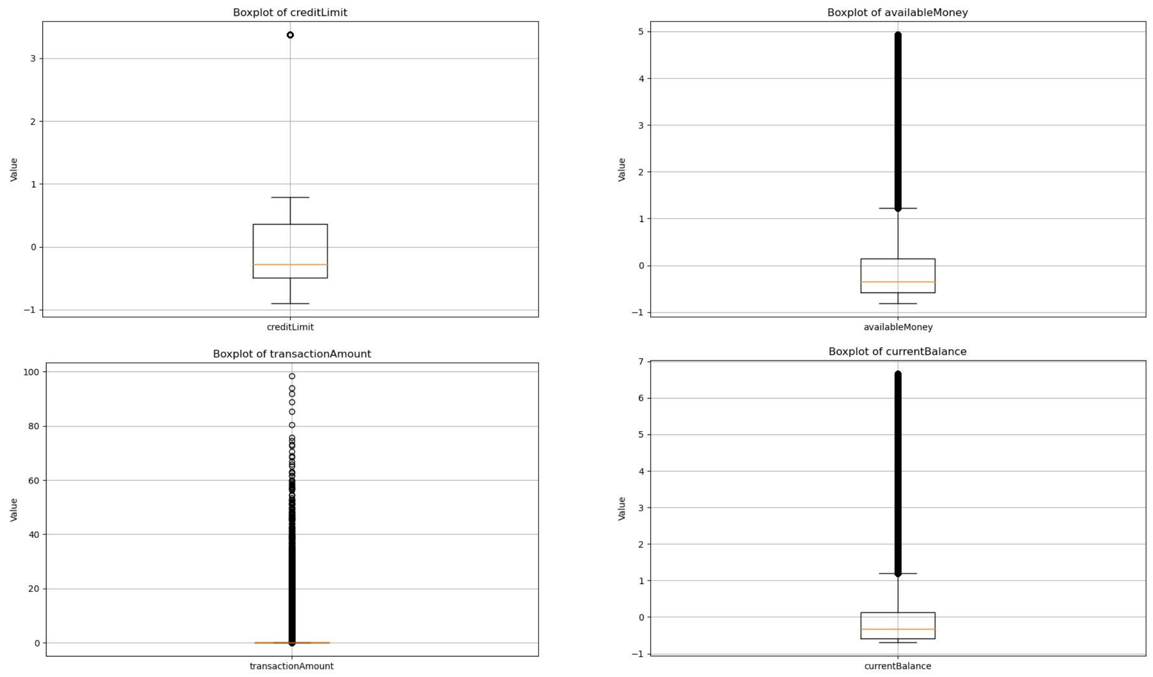
After doing that, we now have 22 variables.

2.2 Outliers in numerical variables

By analyzing the meaning of the variables, we will treat **creditLimit**, **availableMoney**, **transactionAmount**, **currentBalance** as the numerical variables. Before any analyzing, we will scale them with 'StandardScaler'.
Here're the histograms of these four variables:



From the above histograms, all may have outliers (note that there's an extremely large value for **transactionAmount**). We can also check the boxplots:



We also employed the Z-score method to ascertain the number of outliers present in each variable, setting the threshold at ±3. Among a total of **786,363** observations, the findings reveal a significant number of outliers in several variables:

- **creditLimit** exhibits **48,781** outliers,
- **availableMoney** is associated with **25,524** outliers,
- **transactionAmount** contains **1,833** outliers, and
- **currentBalance** has **18,183** outliers.

Currently, our stance is that outliers should not be indiscriminately discarded solely on the basis of their deviation from the norm. There exists a plausible hypothesis that these outliers may have a significant correlation with instances of fraud. Consequently, we advocate for a more nuanced approach, wherein the role and impact of outliers will be meticulously evaluated during the training phase of specific models. This analysis will inform our decision on whether to adjust or exclude these data points, ensuring that our methodology remains both rigorous and data-driven.
Outliers can significantly impact the performance of our models, leading to poor fit and biased results. Should further analysis confirm the need for remediation, one effective strategy for managing outliers involves establishing a big yet reasonable threshold. Values exceeding this threshold can then be capped at the threshold value itself, mitigating their influence.
What should be kept in mind is that there are several extreme values in **transactionAmount**. It is important to monitor these values in future analyses.

2.3 Dealing with missing data

In the first section of this chapter, we have already find that there is some missingness in the variables **acqCountry** with 0.58% missingness, **merchantCountryCode** with 0.09% missingness, **posEntryMode** with 0.52% missingness, and **posConditionCode** with 0.09% missingness.

Since all the missing values are in the categorical variables, we can simply add a new class called `missing` in the corresponding variables.

2.4 Time Variables

In the dataset, several time-related variables are present, namely **transactionDateTime**, **currentExpDate**, **accountOpenDate**, and **dateOfLastAddressChange**. Currently, these variables exhibit varied representations and are stored as objects, indicating that their format is not directly conducive to numerical or date-related operations, such as calculations involving dates.

Consequently, it is imperative to convert these variables to a datetime data type, a process that can be efficiently accomplished using the **to_datetime** function in **pandas**. Upon conversion, each variable becomes a datetime object. However, despite this transformation, directly incorporating these datetime objects into future models remains challenging due to their specialized data type.
The variable **transactionDateTime** encapsulates both date and time information, which are crucial for our analysis. To enhance usability, we partition this variable into two distinct variables: **transactionDate** for the date component and **transactionTime** for the time component. Subsequently, we standardize these variables by subtracting the minimum value observed within each respective variable from all other values, thereby recalibrating our dataset relative to these baselines.

For the variables **transactionDate**, **currentExpDate**, **accountOpenDate**, and **dateOfLastAddressChange**, we compute new values representing the difference in days from the minimum date. Similarly, for **transactionTime**, we calculate the new values as the difference in hours from the earliest time recorded. Through this process, we convert these **time delta** variables into continuous float variables. This transformation facilitates their integration into predictive models without forfeiting any temporal information.
In the end, we will use `StandardScaler` to scale them as usual.

2.5 Special treatment during data wrangling

Due to the unique characteristics of certain variables, special treatment is necessary. For instance, **cardCVV**, **enteredCVV**, and **cardLast4Digits** exhibit distinct properties that require individual consideration.
Furthermore, it is imperative to transform categorical variables into a format that is more conducive for analysis. This can be achieved through one-hot encoding, which also accommodates missing data by incorporating it as a distinct class.

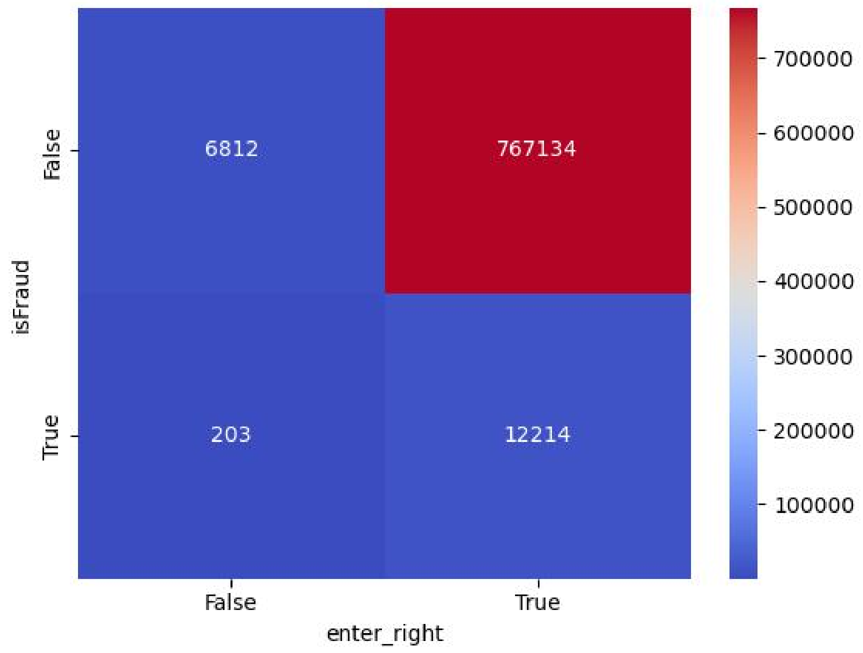
For variables such as **cardCVV** and **enteredCVV**, representing the information in the traditional manner may not be necessary. Alternatively, we can retain **cardCVV** in its original form and substitute **enteredCVV** with a Boolean variable, **enter_right**, to denote whether **enteredCVV** matches **cardCVV**.

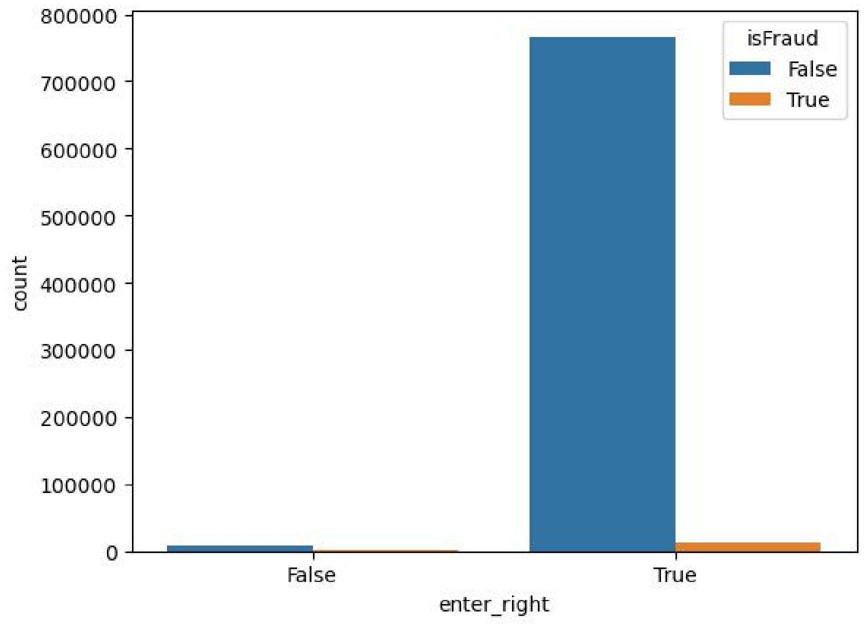
Utilizing the variable **cardLast4Digits** directly presents significant challenges, primarily because it is not numerical and its categorization is impractical due to the vast array of unique categories. Alternatively, we propose the creation of a new variable, **freq_last_4Digits**, representing the frequency of each value within **cardLast4Digits**. Currently, we opt not to discard **cardLast4Digits** to preserve the potential for leveraging its information in future analyses. We posit that **freq_last_4Digits** encapsulates a portion of the information inherent in **cardLast4Digits**, thereby offering a pragmatic approach to its utilization.

2.6 Relationship between cardCVV, enteredCVV, cardLast4Digits and the target variable

To analyze these relationships, we need to use the new variables we constructed from them in the previous section: **enter_right** and **freq_last_4Digits**.

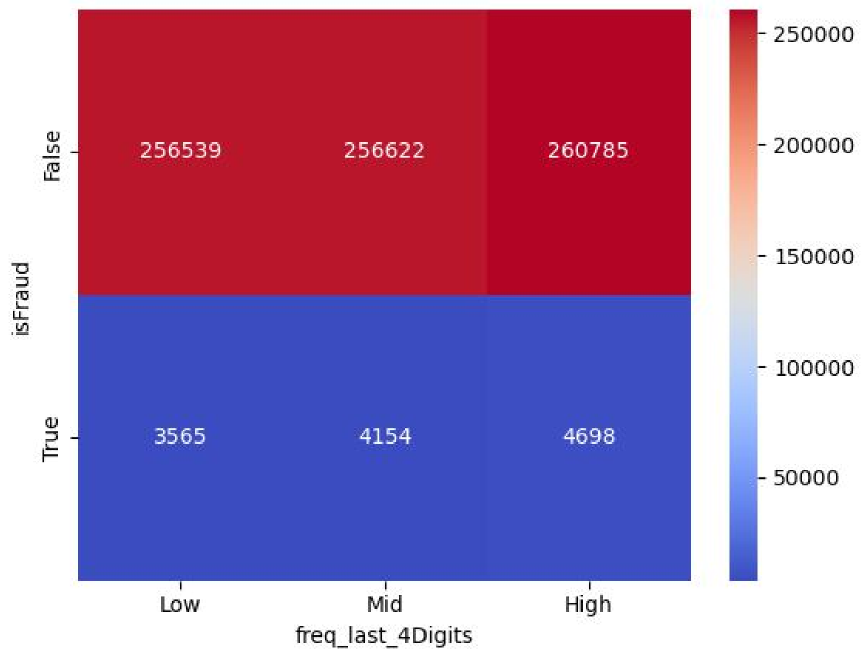
First, let's examine the relationship between **enter_right** and the target variable. Below is the contingency table and histogram between them. From this analysis, we can conclude that most transactions in this dataset with the correct CVV entry are not fraudulent. Similarly, for those with incorrect CVV entry, only a few are fraudulent. This may be due to our dataset not being balanced.



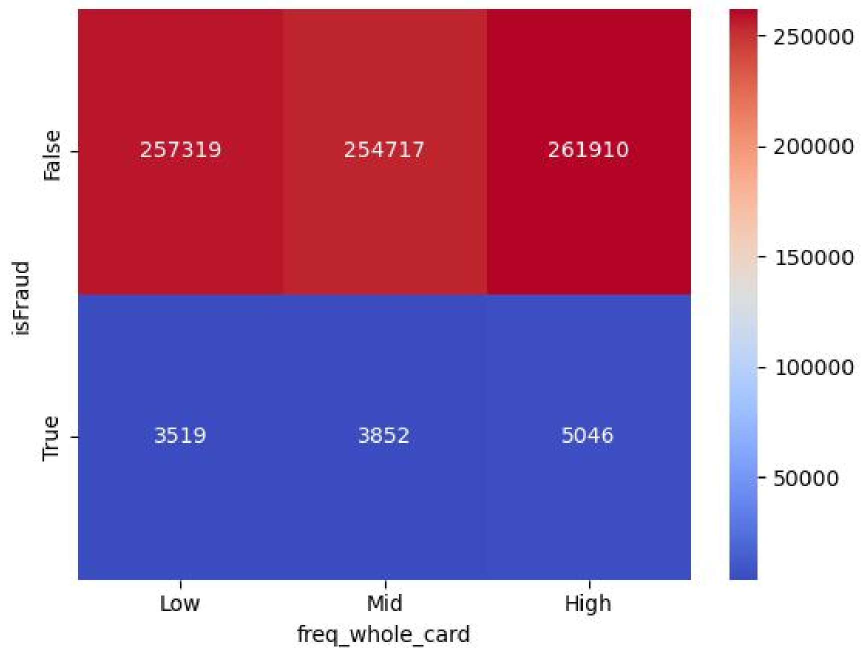


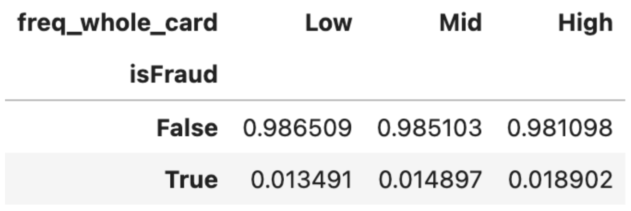
Next, we will analyze the relationship between **freq_last_4Digits** and the target variable. Direct analysis of this relationship is challenging; therefore, we have categorized **freq_last_4Digits** into three distinct groups: low, mid, and high frequency. The contingency table that outlines the relationship between

freq_last_4Digits and the target variable is presented below. We observe that this dataset is significantly unbalanced, making it difficult to conclusively determine the existence of any relationship.



We noticed that, different cards may have same last four digits, so just focus on this variable may not be correct. So we consider to combine **cardCVV** and **cardLast4Digits** together as the identifier of the card, and corresponding the **freq_whole_card**. We could find that the high the frequency, the more likely it's the fraud by calculating the proportion.

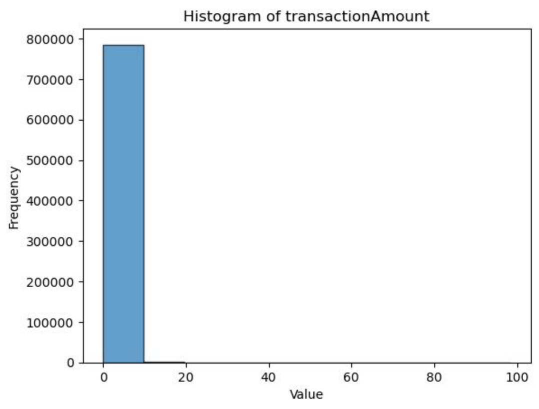


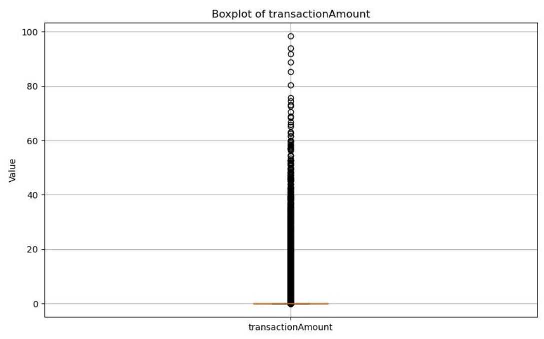


freq_whole_card	Low	Mid	High
isFraud			
False	0.986509	0.985103	0.981098
True	0.013491	0.014897	0.018902

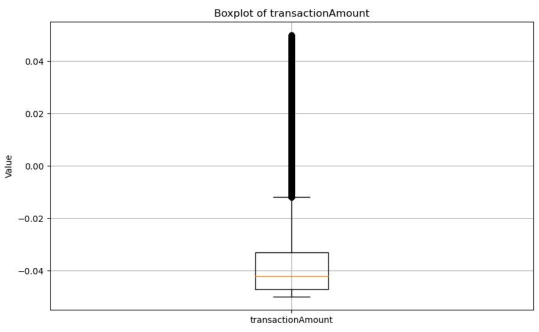
2.7 Analysis of transactionAmount

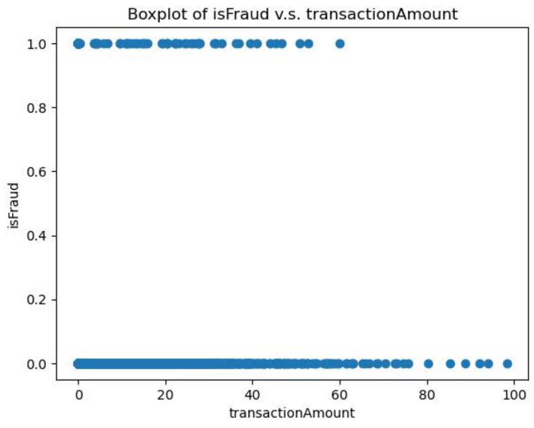
We have already drawn the histogram and boxplot of the (scaled) **transactionAmount** in Section 2 of this chapter, as follows. We briefly mentioned that it contains some extreme values. Now, we will explore this aspect further.





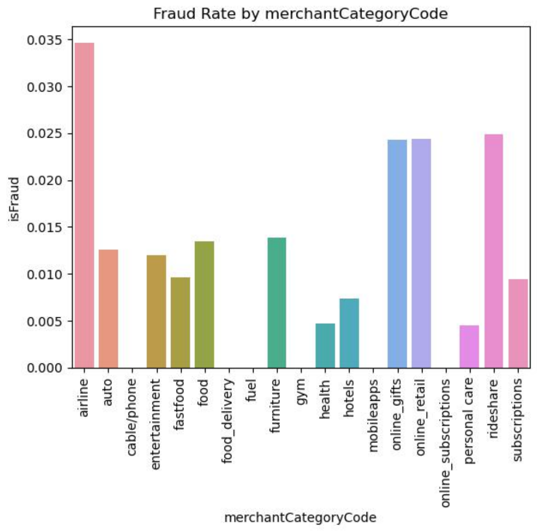
Since this data has been scaled, we will consider any value greater than 3 as an extreme value. Analyzing the boxplot might provide clearer insights. We observe a significant number of values exceeding 3 (1833 out of 786363, as mentioned in Section 2). Remarkably, many of these values surpass 60.
Let's examine the details more closely in the following plot. We find that most data points cluster around -0.04, with the majority being less than 0. Only the extreme values are notably high. This distribution pattern could carry critical information about the variable. It's possible that outliers (extremely large transaction amounts) are less likely to be fraudulent, given the heightened scrutiny they likely attract from banking institutions.

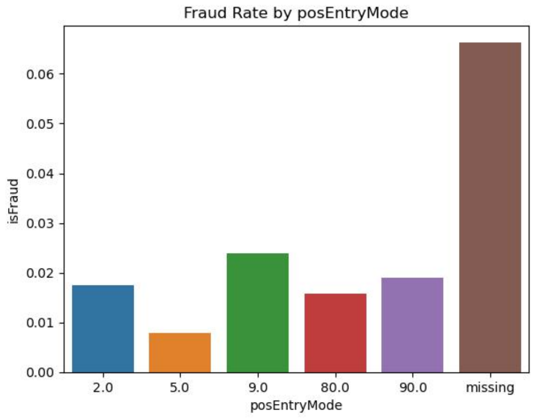


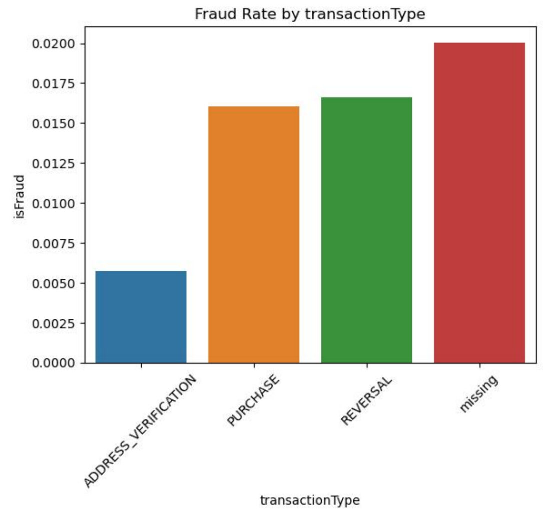


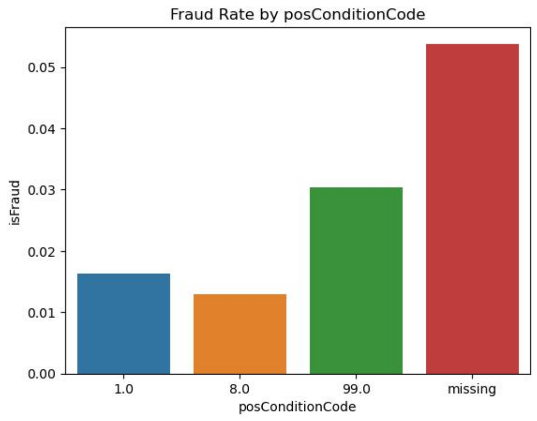
2.8 The relationship between isFraud and categorical predictors

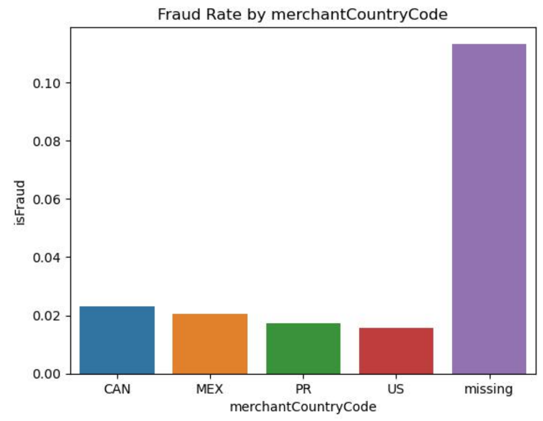
Here are the bar charts of the fraud rate for each category in **merchantCategoryCode**, **posEntryMode**, **transactionType**, **posConditionCode**, and **merchantCountryCode**.
For **merchantCategoryCode**, we observe that some categories exhibit higher fraud rates than others. Notably, categories such as **airline**, **online gifts**, **online retail**, and **rideshare** demonstrate elevated fraud rates compared to others. In the case of **posEntryMode**, the missing data has the highest fraud rate. Then mode 9 registers the highest fraud rate, whereas mode 5 records the lowest. Regarding **transactionType**, the missing data has the highest fraud rate. Then **purchase** and **reversal** exhibit comparably higher fraud rates, surpassing that of **address verification**. Within **posConditionCode**, the missing data has the highest fraud rate. Then code 99 is associated with the highest fraud rate. Lastly, for **merchantCountryCode**, the missing data has the highest fraud rate. Then the sequence of fraud rate magnitude is **CAN** surpassing **MEX**, followed by **PR**, and then **US**.
From the analysis presented above, it is evident that different categories within these variables exhibit varying probabilities of fraudulent transactions. Therefore, this valuable insight can be incorporated into our predictive model. Specifically, classes associated with a higher fraud rate could be assigned a greater weight, increasing their likelihood of being predicted as fraudulent compared to other classes.





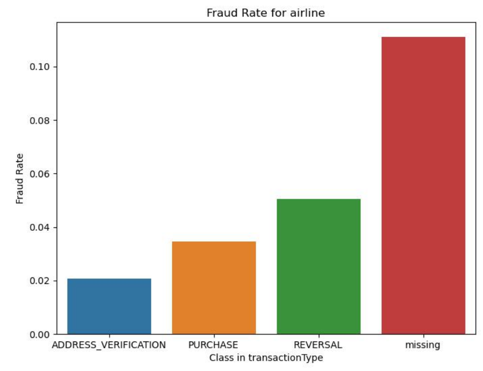


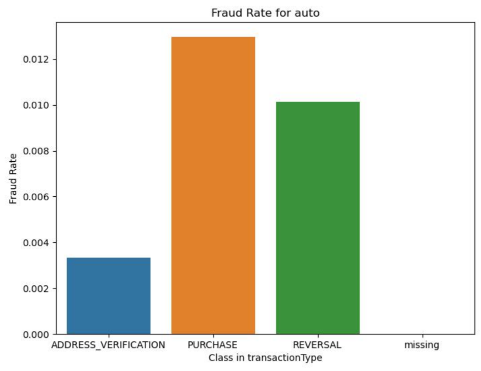


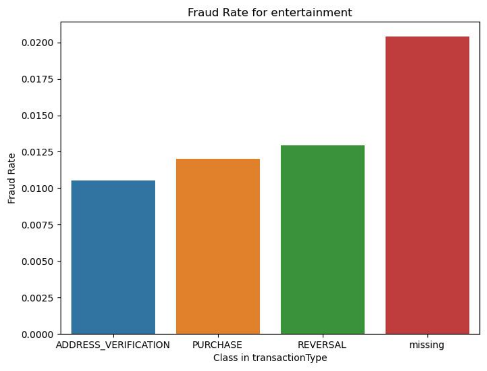


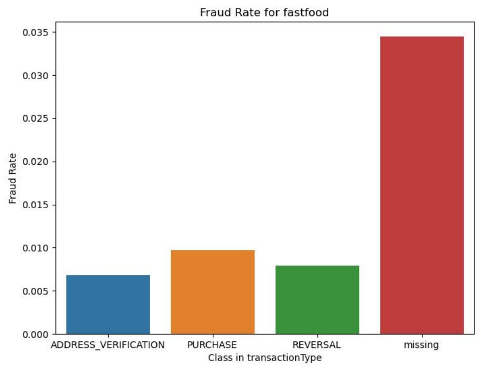
2.9 The relationship between isFraud and transactionType conditioned on merchantCategoryCode

Given the extensive number of classes within **merchantCategoryCode**, various methods have been employed to elucidate these relationships clearly. Initially, bar plots were generated for each class directly. With **merchantCategoryCode** encompassing 19 distinct classes, a total of 19 plots were created. To conserve space, four of these plots are presented below, while the remaining are detailed in the Appendix.

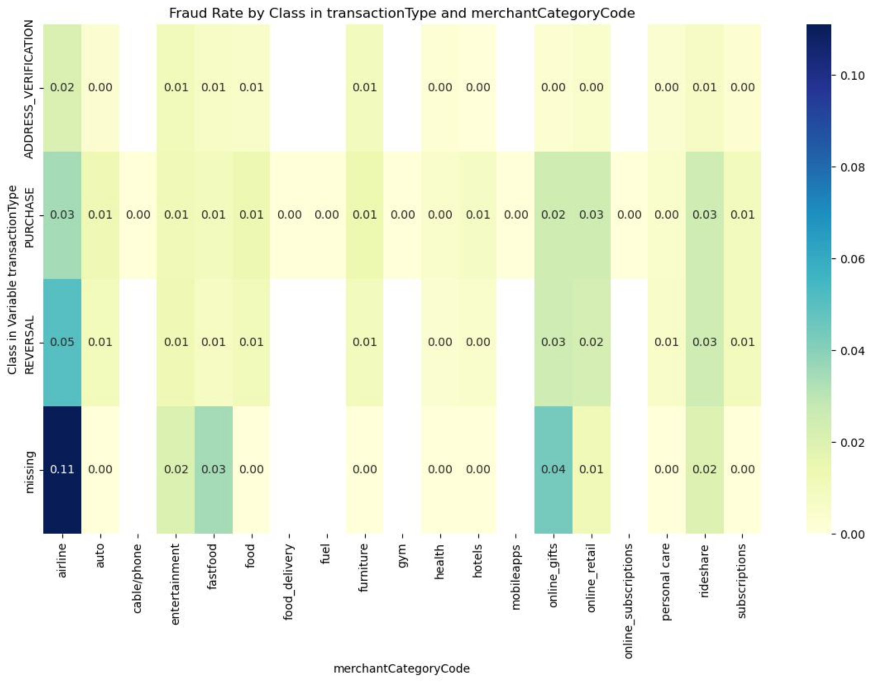








An alternative method for illustrating the relationship utilizes a heatmap. While slightly less clear, this approach is more space-efficient:

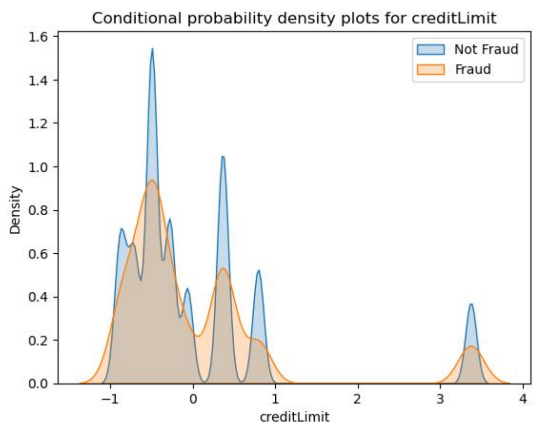


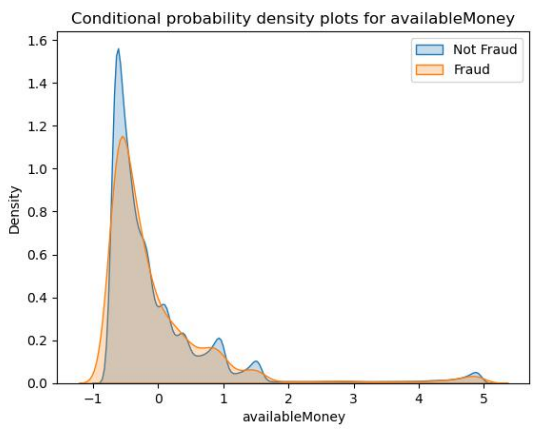
Class in Variable transactionType	airline	auto	cable/phone	entertainment	fastfood	food	food_delivery	fuel	furniture	gym	health	hotels	mobileapps	online_gifts	online_retail	online_subscriptions	personal care	rideshare	subscriptions
ADDRESS_VERIFICATION	0.02	0.00		0.01	0.01	0.01			0.01		0.00	0.00		0.00	0.00		0.00	0.01	0.00
PURCHASE	0.03	0.01	0.00	0.01	0.01	0.01	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.02	0.03	0.00	0.00	0.03	0.01
REVERSAL	0.05	0.01		0.01	0.01	0.01			0.01		0.00	0.00		0.03	0.02		0.01	0.03	0.01
missing	0.11	0.00		0.02	0.03	0.00			0.00		0.00	0.00		0.04	0.01		0.00	0.02	0.00

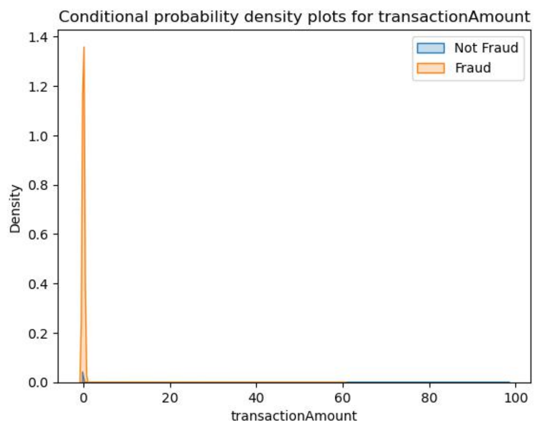
In conclusion, some classes within **merchantCategoryCode** lack fraud data in this dataset. Among the remainder, the fraud rate exhibits distinct patterns across the different classes of **merchantCategoryCode** when compared with the classes of **transactionType**. Naturally, in the majority of the scenarios, the category labeled as **missing** exhibits the highest incidence of fraud. This serves as a valuable reminder of the importance of considering variable interactions. Conclusions regarding the relationship between two variables should not be hastily drawn without accounting for the influence of additional variables.

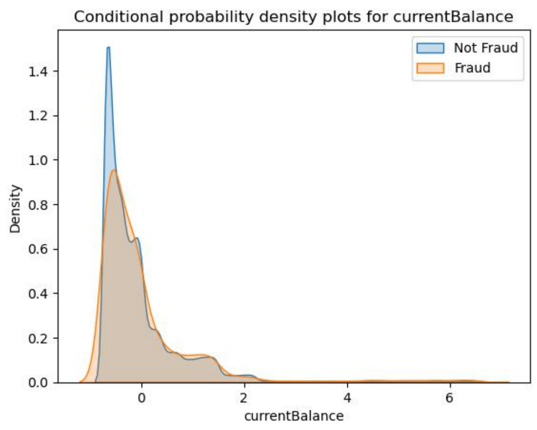
2.10 The relationships between numerical variables and the target variable

Initially, we will examine the conditional probability density plots for the variables **creditLimit**, **availableMoney**, **transactionAmount**, and **currentBalance**. With the exception of **transactionAmount**, which exhibits numerous extreme values obscuring clear patterns, the density distributions of the other variables, conditional on being 'fraud', closely resemble those conditional on 'not fraud'. However, distributions conditional on 'not fraud' tend to be more pronounced and spiky.

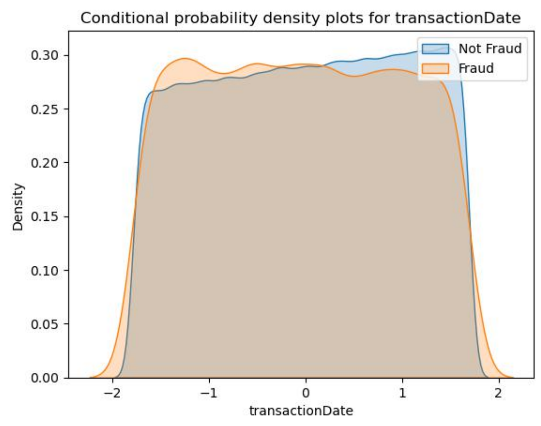


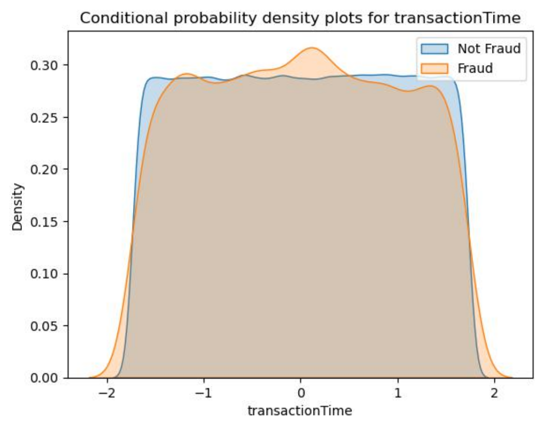


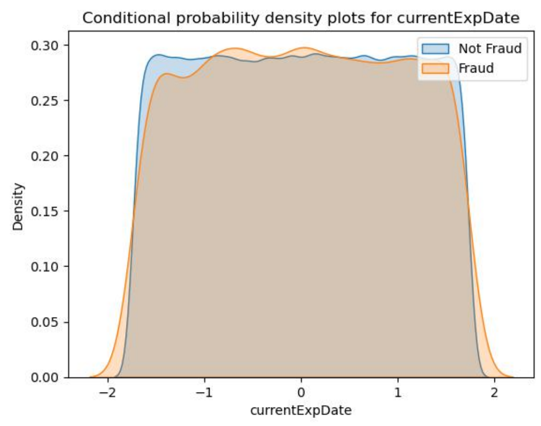


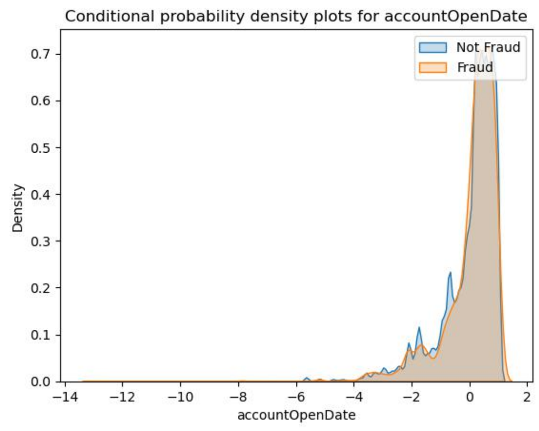


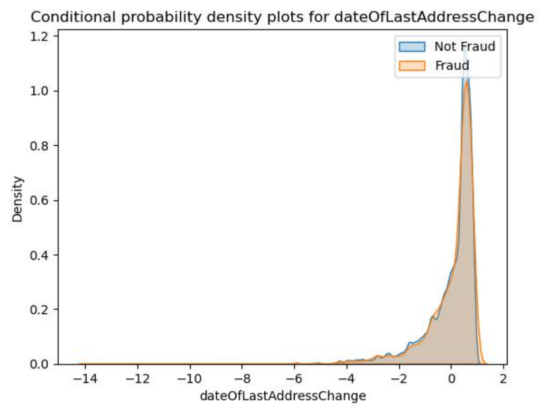
Subsequently, we will investigate the conditional probability density plots for the variables **transactionDate**, **transactionTime**, **currentExpDate**, **accountOpenDate**, and **dateOfLastAddressChange**. It is important to note that these variables were transformed into numerical values in Section 4 of this chapter. The density distributions for these variables, conditional on 'fraud', closely align with those conditional on 'not fraud'.





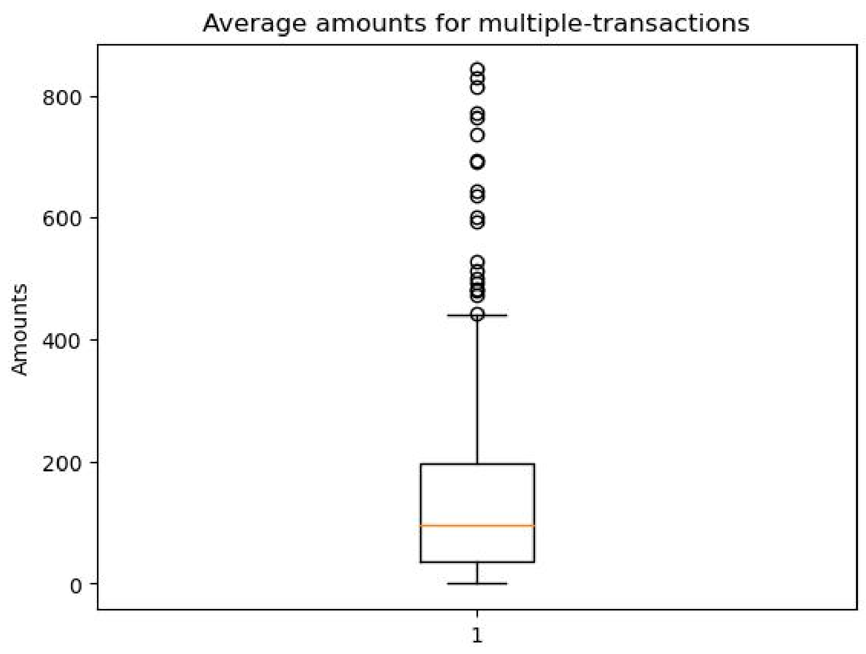




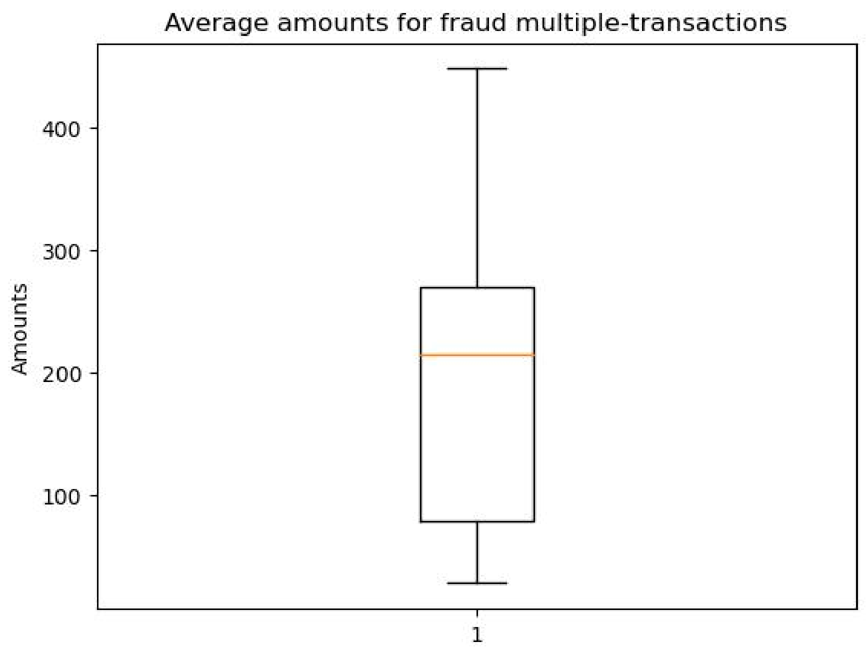


2.11 Identify multi-swipe transactions

In this section, we programmatically identify multi-swipe transactions by establishing specific criteria for their occurrence: identical account, identical amount, and within a 15-minute window. For this analysis, we utilize the original, unscaled data.
Consequently, the proportion of Multi-Swipe Transactions is approximately 0.12%, and the proportion of the Total Dollar Amount for Multi-Swipe Transactions stands at 0.091%. Furthermore, of all multi-swipe transactions, 0.127% are fraudulent.
Additionally, we calculate the average transaction amount for each set of multi-swipe transactions, grouped by account. An outlier transaction amounting to approximately $100,000 has been observed. Excluding this outlier for visualization purposes, we present the subsequent boxplot.



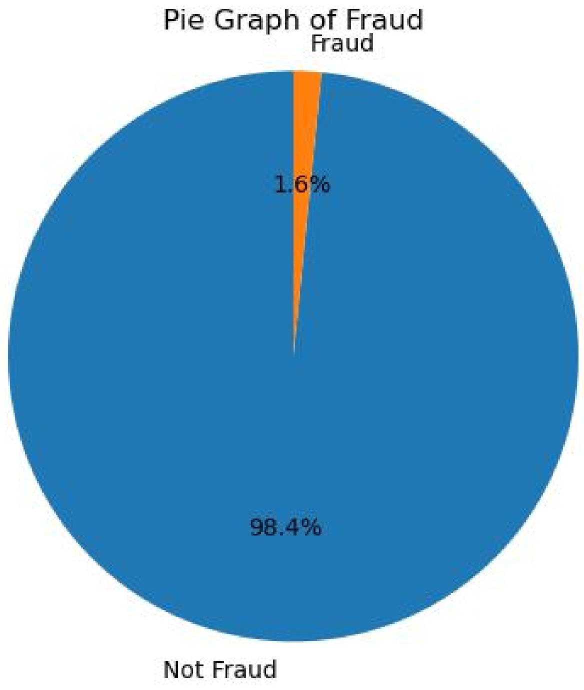
Additionally, a boxplot has been generated to depict the average transaction amount for each group of multi-swipe transactions classified as fraudulent, segregated by account. The plot reveals that the majority of fraudulent transactions of this nature involve modest amounts, typically around $200.



2.12 The class imbalance

We ascertain that approximately 1.6% of the data represent fraud, while the remaining 98.4% are not fraudulent. This indicates a pronounced imbalance within the dataset. The potential implications include:

1) **Model Bias**: Standard algorithms are predisposed to favoring the majority class, potentially leading to a substantial incidence of false negatives for the minority class—that is, fraudulent transactions being misclassified as non-fraudulent.
2) **Performance Metrics**: Traditional performance metrics might not be suitable or could give a distorted view of a model's effectiveness. For example, model accuracy may seem elevated if the model predominantly predicts the majority class.
3) **Overfitting Risk**: There is a hazard that the model may become overfitted to the majority class and, consequently, lack adequate generalization to the minority class.

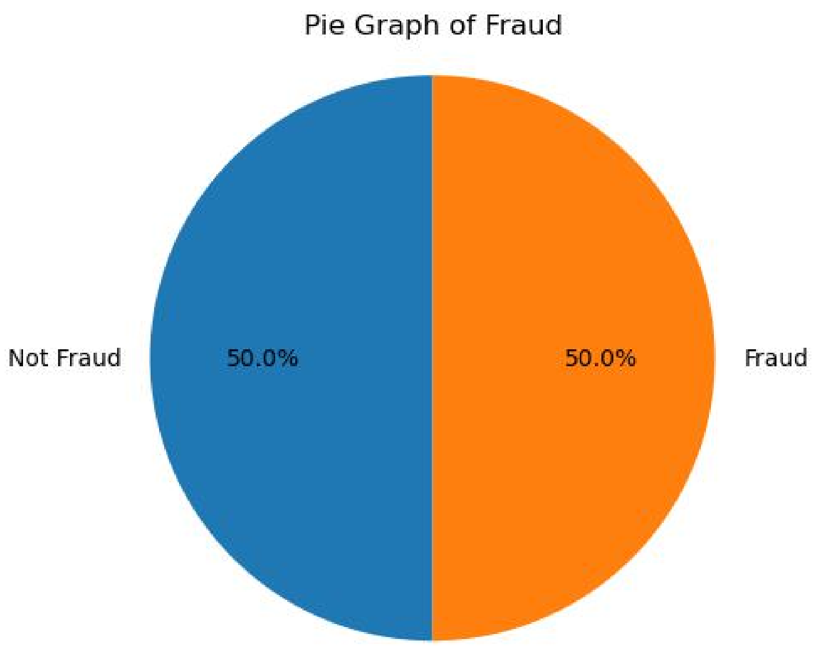


The strategies that may be employed to alleviate this issue are as follows:

- **Resampling the Dataset**: Employ techniques such as oversampling the minority class, undersampling the majority class, or applying a hybrid method known as SMOTE (Synthetic Minority Oversampling Technique) to rebalance the dataset.
- **Ensemble Methods**: Utilize a combination of various models to integrate their predictions, which can help counteract the class imbalance impact.

2.13 Mitigate class imbalance

We can apply Downsampling of the Majority Class. Following this procedure, both the fraudulent and non-fraudulent data categories will be equalized, each comprising $12,417$ observations. This quantity remains sufficient for conducting subsequent analyses.



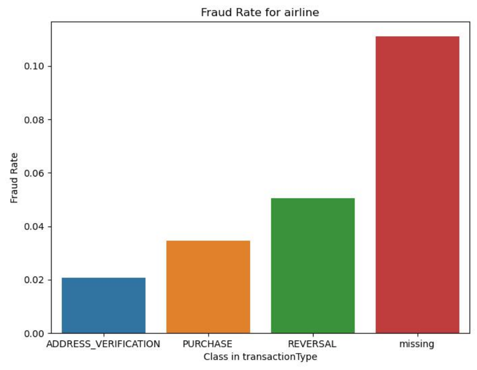
Downsampling the majority class helps to increase the sensitivity of the model to the minority class. The model's accuracy metric becomes more meaningful after addressing class imbalance. Standard accuracy can be misleading in imbalanced datasets since a model can achieve high accuracy by simply predicting the majority class. After downsampling, metrics like precision, recall, F1-score, and ROC-AUC become more indicative of the model's true performance.
Downsampling involves randomly removing instances from the majority class, which can lead to the loss of potentially valuable information that could be useful for the model to learn the underlying data patterns. And thus increasing the probability of overfitting.

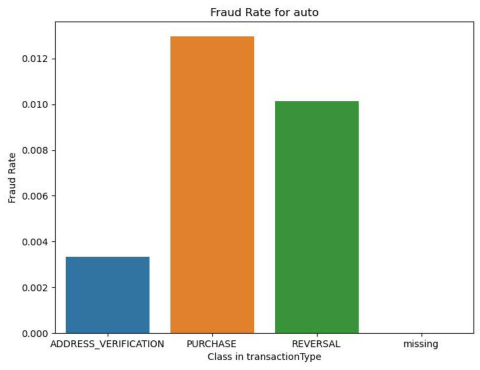
3 End

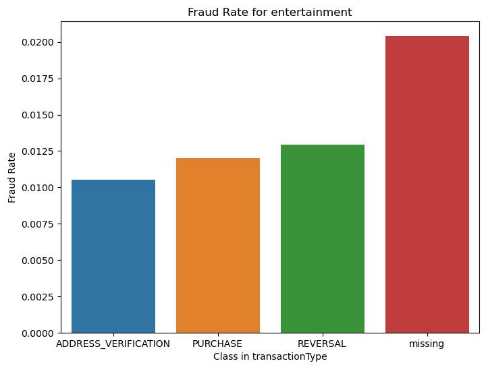
This concludes the content for Lab 1. Thank you for dedicating time to review this comprehensive report.

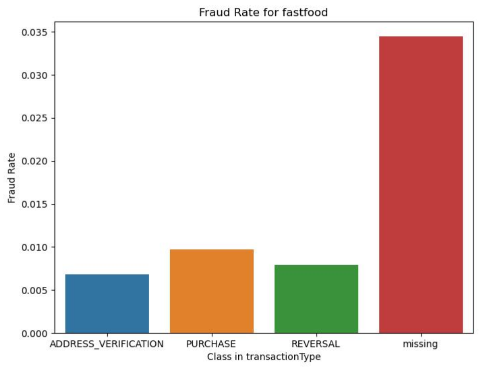
4 Appendix

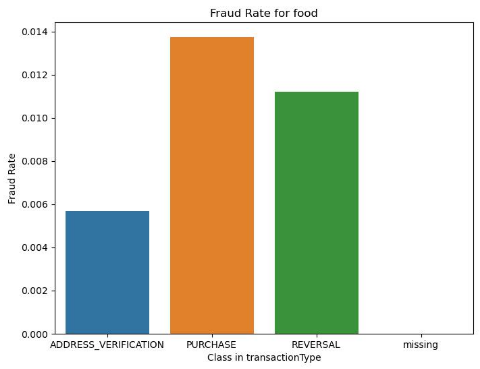
4.1 The relationship between isFraud and transactionType conditioned on merchantCategoryCode

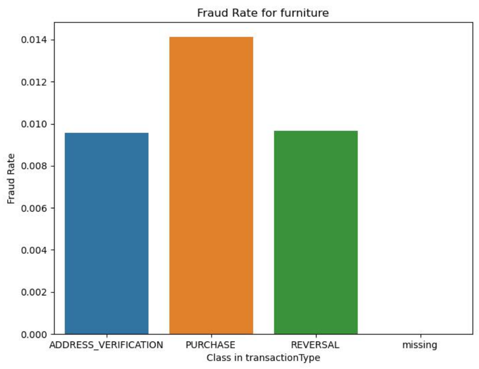


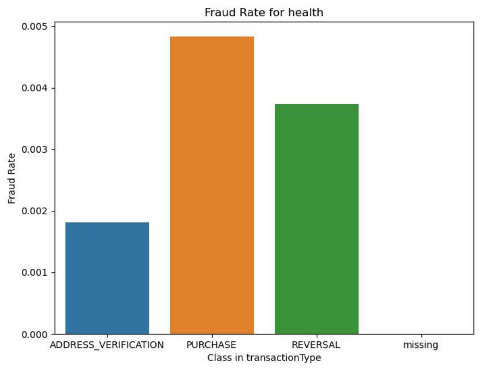


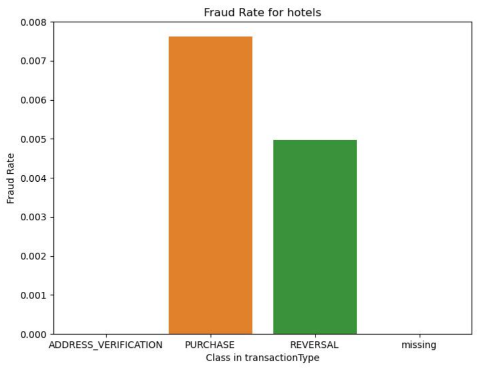


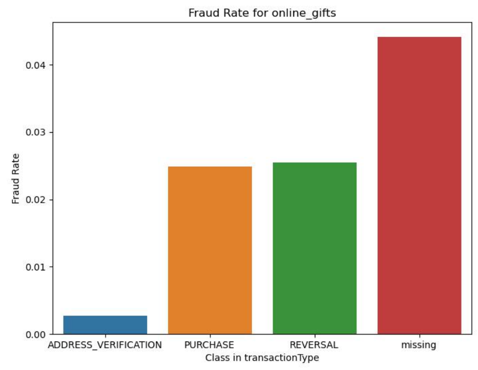


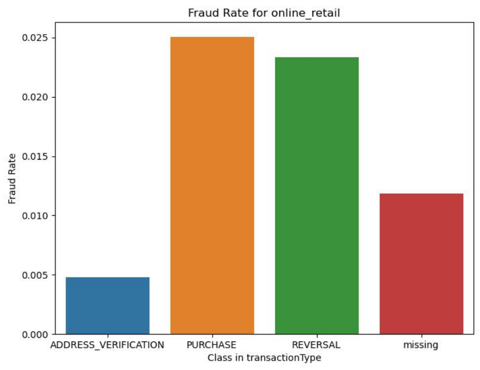

Fraud Rate for food
Fraud Rate
Class in transactionType
ADDRESS_VERIFICATION
PURCHASE
REVERSAL
missing
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014

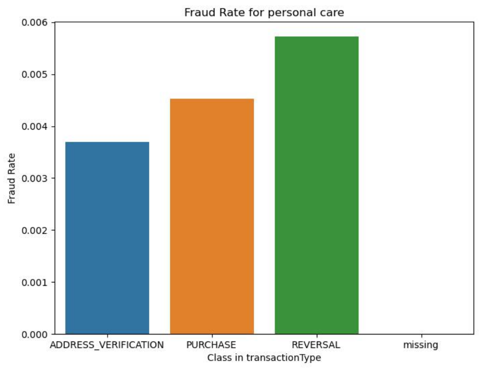

Fraud Rate for furniture
Fraud Rate
Class in transactionType
ADDRESS_VERIFICATION
PURCHASE
REVERSAL
missing
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014

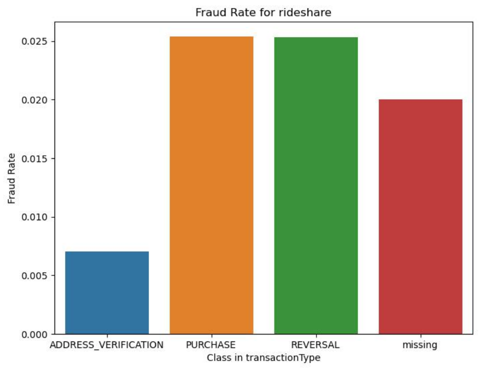

Fraud Rate for health
Fraud Rate
Class in transactionType
ADDRESS_VERIFICATION
PURCHASE
REVERSAL
missing
0.000
0.001
0.002
0.003
0.004
0.005

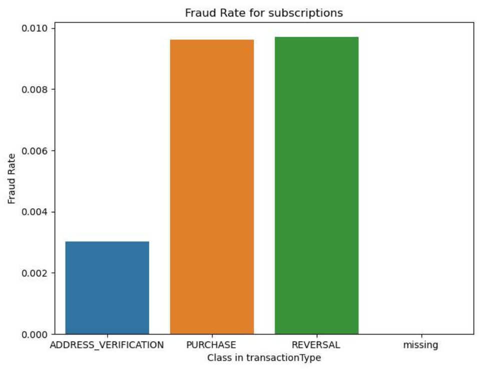

Fraud Rate for hotels
Fraud Rate
Class in transactionType
ADDRESS_VERIFICATION
PURCHASE
REVERSAL
missing
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008

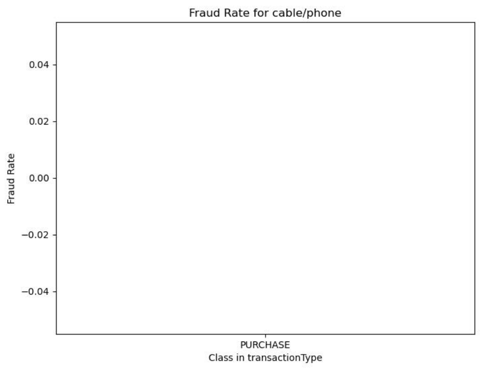

Fraud Rate for online_gifts
Fraud Rate
0.00
0.01
0.02
0.03
0.04
ADDRESS_VERIFICATION
PURCHASE
REVERSAL
missing
Class in transactionType

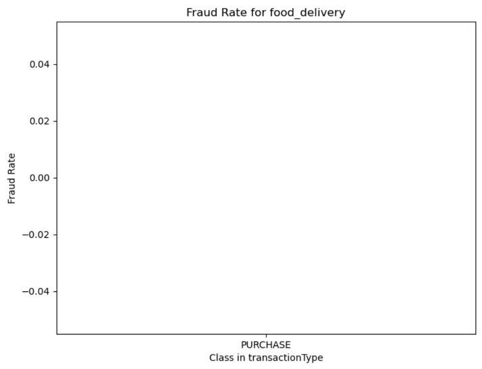

Fraud Rate for online_retail
Fraud Rate
0.000
0.005
0.010
0.015
0.020
0.025
ADDRESS_VERIFICATION
PURCHASE
REVERSAL
missing
Class in transactionType

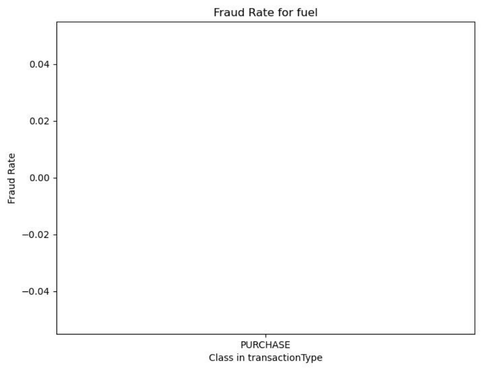

Fraud Rate for personal care
Fraud Rate
0.000
0.001
0.002
0.003
0.004
0.005
0.006
ADDRESS_VERIFICATION
PURCHASE
REVERSAL
missing
Class in transactionType

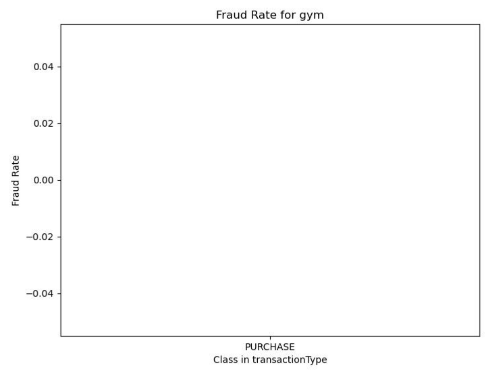

Fraud Rate for rideshare
Fraud Rate
0.000
0.005
0.010
0.015
0.020
0.025
ADDRESS_VERIFICATION
PURCHASE
REVERSAL
missing
Class in transactionType

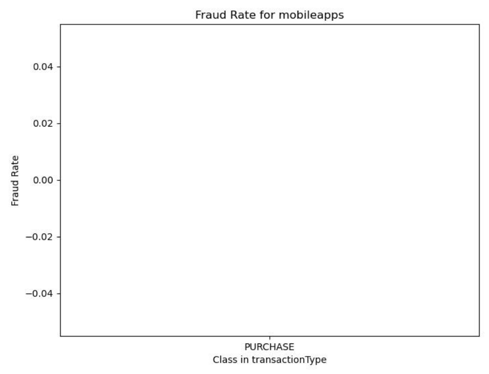

Fraud Rate for subscriptions
Fraud Rate
0.010
0.008
0.006
0.004
0.002
0.000
ADDRESS_VERIFICATION
PURCHASE
REVERSAL
missing
Class in transactionType

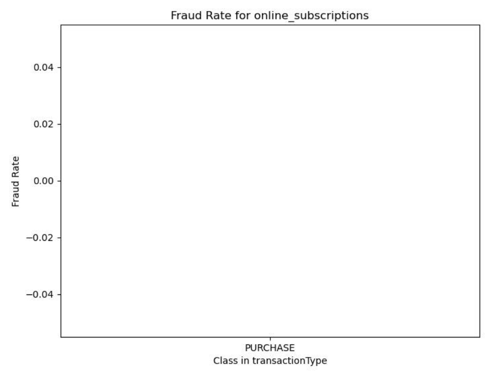

Fraud Rate for cable/phone
Fraud Rate
0.04
0.02
0.00
−0.02
−0.04
PURCHASE
Class in transactionType


Fraud Rate for food_delivery
Fraud Rate
0.04
0.02
0.00
−0.02
−0.04
PURCHASE
Class in transactionType


Fraud Rate for fuel
Fraud Rate
0.04
0.02
0.00
−0.02
−0.04
PURCHASE
Class in transactionType


Fraud Rate for gym
Fraud Rate
0.04
0.02
0.00
−0.02
−0.04
PURCHASE
Class in transactionType


Fraud Rate for mobileapps
Fraud Rate
0.04
0.02
0.00
−0.02
−0.04
PURCHASE
Class in transactionType


Fraud Rate for online_subscriptions
Fraud Rate
0.04
0.02
0.00
−0.02
−0.04
PURCHASE
Class in transactionType