Finance and Economics Discussion Series

Federal Reserve Board, Washington, D.C.
ISSN 1936-2854 (Print)
ISSN 2767-3898 (Online)

CardSim: A Bayesian Simulator for Payment Card Fraud Detection Research

Jeffrey S. Allen

2025-017

Please cite this paper as:
Allen, Jeffrey S. (2025). "CardSim: A Bayesian Simulator for Payment Card Fraud Detection Research," Finance and Economics Discussion Series 2025-017. Washington: Board of Governors of the Federal Reserve System, https://doi.org/10.17016/FEDS.2025.017.

CardSim: A Bayesian Simulator for Payment Card Fraud Detection Research

Jeffrey S. Allen*

February 2025

Abstract

Payment fraud has been high in recent years, and as criminals gain access to capability-enhancing generative AI tools, there is a growing need for innovative fraud detection research. However, the pace, diversity, and reproducibility of such research are inhibited by the dearth of publicly available payment transaction data. A few payment simulation methodologies have been developed to help narrow the payment transaction data gap without compromising important data privacy and security expectations. While these simulation approaches have enabled research advancements, more work is needed to generate datasets that reflect diverse and evolving fraud tactics. This paper introduces $CardSim$, a flexible, scalable payment card transaction simulation methodology that extends the small but emerging body of simulators available for payment fraud modeling research. $CardSim$ is novel in the extent to which it is calibrated to publicly available data and in its Bayesian approach to associating payment transaction features with fraud. The simulator's modular structure, which is operationalized in a corresponding software package, makes it easy to update based on new evidence about payment trends or fraud patterns.† After laying out the simulation methodology, I show how outputs can be used to test and evaluate machine learning workflows, modeling approaches, and interpretability frameworks that are relevant for payment card fraud detection.

Keywords: payment cards, fraud detection, Bayesian analysis, simulation, machine learning

JEL classification: C11, C15, C80, E42

* Federal Reserve Board. E-mail: jeff.allen@frb.gov. I would like to thank Sonja Danburg, Chuan Du, Jillian Mascelli, Kathy Wilson, Sarah Wright, Nathan Palmer, and Geoff Gerdes of the Federal Reserve Board and Kevin Foster, Curtis Koster, Jonathan Kuah, and Jair Filho of the Federal Reserve System for their feedback. The views expressed in this paper are solely those of the author and should not be interpreted as reflecting the views of the Federal Reserve Board.

† A Python package implementing the simulator methodology has been released with this paper for public use.

1 Introduction

Payment card fraud is a perennial concern for consumers, merchants, financial institutions, and policymakers. Fraud statistics show significant growth in payment card fraud in the United States since the COVID-19 pandemic. Survey data from the Federal Reserve Bank of Atlanta found that in 2023, 11.5 percent of credit card owners and 9.4 percent of debit card owners experienced card-related theft or fraud (Foster, Greene, and Stavins, 2024). These figures were more than double the rates before the pandemic. Similarly, the number of credit card fraud reports filed by consumers with the Federal Trade Commission (FTC) and the number of debit- and credit card-related suspicious activity reports filed by depository institutions with FinCEN were, respectively, 113 and 75 percent higher in 2023 than in 2019 (FTC, 2020, 2024; FinCEN, 2024).[1] Fraud was also the top operational risk concern cited by a plurality of financial institution risk officers (49 percent of respondents) in a 2023 Federal Reserve Financial Services (FRFS) survey, above cyber security, business disruptions, and third-party risk (FRFS, 2024).[2]

Developments in artificial intelligence (AI) complicate the fraud picture. On the one hand, easily accessible generative AI tools could augment malicious actors' ability to perpetuate fraud in payment systems by helping them to develop and deploy more convincing fraud schemes (FSSCC, 2024; FSB, 2024; U.S. Treasury, 2024). On the other hand, financial institutions and authorities use AI extensively for fraud detection, prevention, and response (FSB, 2017; Board of Governors and others, 2021; FSOC, 2023; OECD, 2023). Over the last two years, payment service providers have announced a wide array of AI-related fraud detection initiatives (for examples, see: Mastercard, 2024; Visa, 2024).

Academic research on payment card fraud detection has also grown. A Web of Science search shows 316 published studies on payment card fraud detection in the past 10 years.[3] A major constraint on the pace and diversity of fraud detection research is the lack of publicly available payment data. Payment transaction data are highly sensitive and economically valuable. The widespread use of transaction data for fraud detection research is understandably inhibited by data privacy imperatives and economic incentives. Some published studies use slices of data provided

[1] The FTC data did not capture debit card fraud reports in 2019.

[2] As of this writing, year-end 2024 data were not available for the sources cited here.

[3] Based on the following search string: ("payment card" OR "debit card" OR "credit card") AND "fraud detection".

by financial institutions or central banks under strict privacy and security controls. The inability to publish underlying data in these studies hampers reproducibility and research extensions. Many other studies use a core set of publicly available datasets. While these datasets have enabled significant innovation and knowledge sharing in the fraud detection discipline, they tend to be dated, small samples that are highly masked. These challenges can make it difficult to thoroughly compare machine learning (ML) approaches in the face of constantly evolving fraud patterns.

Simulations have emerged as a way to overcome some data gaps in this area. Although they are imperfect representations of reality and have their own limitations, simulations can help mitigate privacy and competitive concerns. Well-designed simulators can quickly generate very large samples and can be modified to reflect changing payment behaviors and fraud techniques. In recent years, researchers have introduced payment-related simulation methodologies to support research in payment fraud detection and related areas, such as anti-money laundering (AML) (Le Borgne and others, 2022; Lopez-Rojas, Elmir, and Axelsson, 2016; Suzumura and Kanezashi, 2021; Altman and others, 2023). While these simulators have enabled a wide range of researchers to test ML models for payment fraud detection, more work is needed to build methods that reflect the varied dynamics of fraud in payment systems.

This paper extends the small but emerging body of simulation methodologies geared toward fraud detection research by introducing $CardSim$, a flexible, scalable payment card transaction simulator that has three important features. First, most aspects of the simulator are calibrated to publicly available survey and economic data.[4] Second, the simulator embeds complex relationships between payment transaction features and fraud using Bayes' theorem. Third, the simulator is highly modular. Nearly every parameter can be easily adapted to capture changing payment and fraud trends. Collectively, these properties facilitate testing the strengths and weaknesses of ML models for fraud detection. The methodology is operationalized in a software package that has been published alongside this paper for public use.

The remainder of this paper proceeds as follows. In section 2, I review related work that seeks to address the payment transaction data gap. Section 3 lays out the simulation methodology in detail, which involves three key phases: developing payer and payee characteristics, running a

[4] Calibration sources include the Diary of Consumer Payment Choice (Foster, Greene, and Stavins, 2024), the Federal Reserve Payment Study (Board of Governors, 2024), and Gerdes, Greene, and Liu's (2018) payment fraud study.

transaction simulator, and using Bayes' theorem to generate a fraud flag. Next, section 4 walks through the results of a representative simulation. In section 5, I show how simulator outputs can be used to test and evaluate ML workflows and modeling approaches that are relevant for fraud detection. Section 6 concludes by summarizing the implications of this work and identifying areas for future research.

2 Related work

Data privacy standards and competitive economic pressures prevent the dissemination of sensitive payment transaction data for research purposes (Altman and others, 2023; BIS, 2023). While some researchers have access to slices of transaction data from financial institutions under tight controls, many use a limited set of published datasets. Grover and others (2022) catalogue the most used datasets for fraud detection and related issues. Four of nine datasets they identify relate to retail payments. One dataset published by the Machine Learning Group at the Universite libre de Buxelles (ULB) in collaboration with the payment processing firm Worldline on the Kaggle machine learning website (Dal Pozzolo, 2016) is a popular choice for fraud detection research.[5] The IEEE-CIS payment transaction dataset (Howard and others, 2019) is also widely used.

Even when researchers use real transaction data, challenges often remain. Published samples are typically small, non-current, and masked. These properties are evident in the ULB-Worldline data, which contain a relatively small number of highly redacted records that were carried out more than a decade ago. The feature variables are reduced to principal components to completely mask what they represent. Another important challenge is dealing with inconsistent or entity-specific labeling practices that limit generalizability. For example, fraud flags may reflect what happens after fraud is detected rather than the occurrence of fraud specifically.[6] The IEEE-CIS data has a fraud labeling framework that involves labeling all transactions associated with an account as fraudulent after fraud is first reported until the account is terminated or 120 days elapse.[7] While fraud detection research has benefitted tremendously from the publication of these

[5] As of this writing, the ULB- Worldline dataset is the second most upvoted dataset on Kaggle out of 434,571. Source: Kaggle (2025), "Datasets," https://www.kaggle.com/datasets?sort=votes, accessed February 16, 2025.

[6] Labeling can also be affected by jurisdiction-specific policies. For example, some jurisdictions may allow more widespread use of personally identifiable information, such as unique customer identifiers. This can enable customer transaction history to persist after certain account details, such as card number, have been terminated.

[7] For a discussion, see https://www.kaggle.com/c/ieee-fraud-detection/discussion/101203#589276.

datasets, their limitations can pose challenges for thoroughly assessing ML models that thrive on big datasets, investigating evolving fraud patterns, and testing emerging interpretability methods.

This paper builds on recent simulation methodologies that are meant to help narrow the payment transaction data gap and support fraud and money laundering detection research. Lopez-Rojas, Elmir, and Axelsson's (2016) $PaySim$ generates synthetic mobile money transactions using a baseline dataset of real payment transactions from an African mobile money service provider. Suzumura and Kanezashi's (2021) $AMLSim$ is a well-known graph-based simulation methodology for embedding complex money laundering typologies in payment transaction data. Building on $AMLSim$, Altman and others (2023) introduced $AMLWorld$, which includes more complex agent interactions than its predecessor. Much like $CardSim$, all transaction patterns in $AMLWorld$ are based on statistical distributions.

Many precedent simulators are agent-based models. $Cardsim$ has some features in common with agent-based models in that payers and payees interact. Ultimately, though, the methodology is more consistent with stochastic microsimulation models that assume constant behavior (Orcutt, 1957; Birkin and Wu, 2011). The payers and payees in $Cardsim$ do react to constraints like account balances, and their behavior does not change over time. In contrast to many microsimulation applications, $Cardsim$ is not geared toward public policy analysis.

$CardSim$ builds most heavily on the methodology developed by Le Borgne and others (2022). As in their simulator, $CardSim$ focuses on payment card transactions and uses payer and payee profiles to develop and run a daily transaction simulator. However, the core of $CardSim$ differs in important ways. Notably, most of the key simulation parameters are derived from publicly available survey and economic data, and the corresponding parameters used in the two approaches differ significantly. Additionally, Le Borgne and others (2022) embed three types of fraud typologies in their simulator.[8] $CardSim$ does not use typologies, which tend to be deterministic, to capture fraud. Rather, it uses Bayes' theorem and known properties of fraudulent transactions to associate payment transaction features with fraud. To my knowledge, $Cardsim$ is

[8] The typologies are: (1) High value: transactions above a certain value are labeled as fraud; (2) Compromised merchant: all transactions from randomly drawn merchants are labeled as fraud for a period of time; and (3) Compromised credentials: a fraction of payments made by a random selection of payers are labeled as fraudulent for two weeks, and corresponding payment amounts are scaled up by a factor of five.

the first Bayesian payment card fraud simulator, and it is more heavily dependent on publicly available survey data than related alternatives.

3 Simulation methodology

$CardSim$ focuses on generating synthetic transaction data and fraud patterns consistent with consumer-to-business (C2B) non-prepaid debit and credit card payments and unauthorized, third-party fraud.[9] The simulation proceeds in three phases. First, I develop payer and payee characteristics that establish geographic locations, typical payment patterns, and the ratio of payers to payees. Second, I run the core transaction simulator, which establishes the number of payments payers make each day and develops five transaction attributes. Finally, I generate a fraud label using Bayes theorem. Every parameter the simulator uses is adjustable. This modularity can enable researchers and practitioners to experiment with different parameters and simulate evolving payment and fraud trends. Sections 3.1-3.3 explain the three simulator phases in detail.

3.1 Payer and payee characteristics

The first phase of the simulator involves developing payer and payee characteristics. Focusing first on the payers, each is assigned an average number of daily card transactions, an average value and dispersion of debit and credit card payment amounts, and x, y coordinates. I derive the payer-specific payment characteristics using the Diary of Consumer Payment Choice (DCPC) (Foster, Greene, and Stavins, 2024). To generate the average number of daily card transactions, I draw m samples of size n from the weighted number of daily debit and credit card transactions reported by individuals in the 2022 and 2023 DCPC surveys.[10] In the simulator run underlying this paper, $m = 2{,}500$ and $n = 100$. I only retain individuals who have adopted a debit or credit card.

Next, I find the distribution of sample averages, which is depicted in Figure 1. The distribution's centrality is consistent with the average number of daily debit and credit card transactions from the DCPC, which stood at 0.76 in 2022 and 0.92 in 2023. For each payer, I draw a random value with replacement from the distribution of sample averages. As discussed later in section 3.2, the transaction simulator draws the daily number of card transactions for an individual

[9] For discussions of different fraud typologies, see Federal Reserve (2023) and Gerdes, Greene, and Liu (2018).

[10] I use the DCPC's day-of-week weights to adjust the number and value of payment transactions used in the simulator.

from a Poisson distribution, which only takes a mean parameter. Thus, I do not generate a dispersion component for the number of daily transactions.

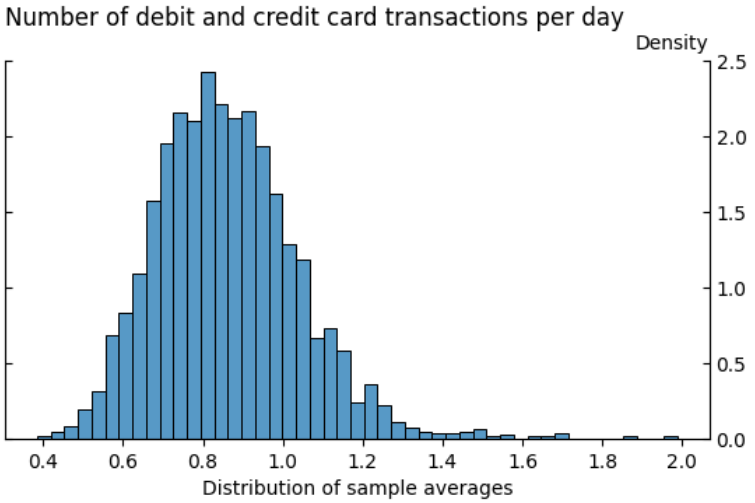

Figure 1. Distribution of card transaction sample averages. Figure depicts the distribution of sample averages of the daily number of card transactions for 2,500 samples of 100 observations drawn from the 2022 and 2023 DCPC surveys. The distribution is centered around 0.86 transactions per day.

Generating the payer-specific average value and dispersion of debit and credit card payment amounts is similar to, but slightly more complex than, finding the average daily transactions. As in the latter, I draw m samples of size n from the weighted debit and credit card transaction values, separately, from the combined 2022 and 2023 DCPCs, where $m = 5{,}000$ and $n = 200$ in this simulator run. Figure 2 shows the distribution of card payment amounts (left panel), zeroing in on those below $500, sample means (middle panel), and sample medians (right panel), with debit card distributions on the bottom row and credit card distributions on the top row. Unsurprisingly, the raw distributions of payment amounts are heavily skewed. The distributions of sample means are centered at approximately $62-72, while the distributions of sample medians are centered around $18-21. The latter appear more realistic as representative payment amounts for the payer-specific characteristics. Thus, I draw random values for each payer from the distributions of medians to serve as the payer's average payment amount for debit and credit card transactions.

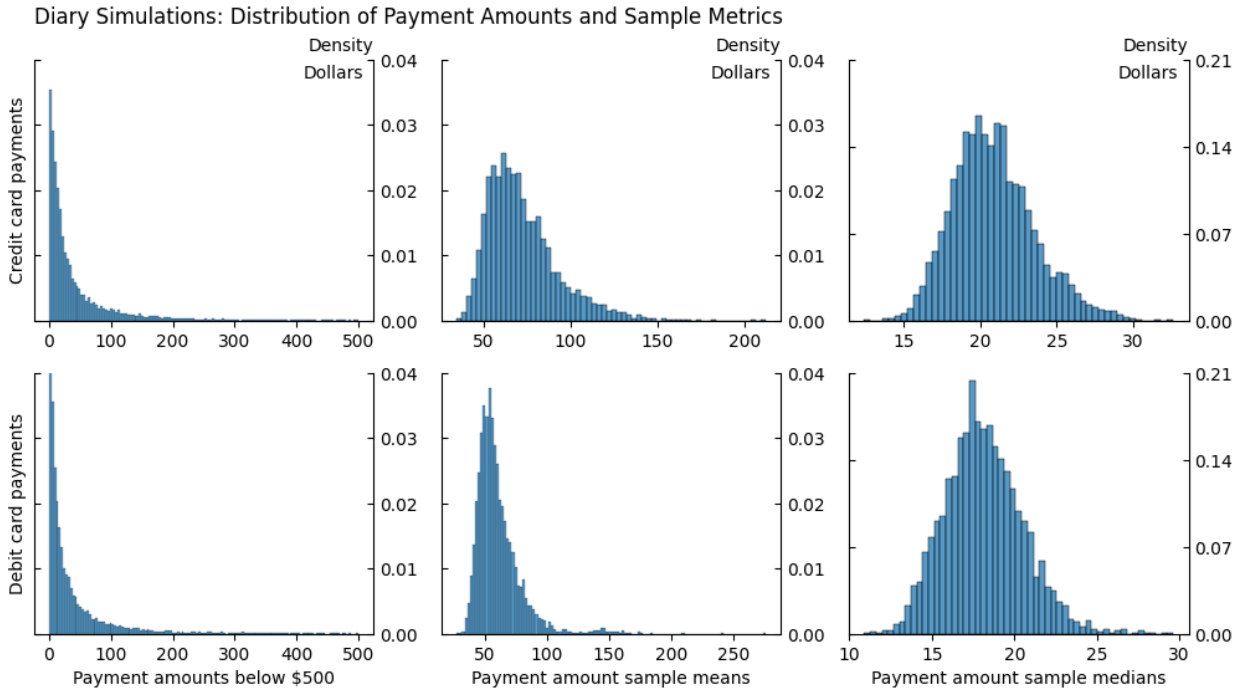

Figure 2. Distribution of payment amounts and sample metrics from the diary data and simulations. Figure depicts distributions of credit (top row) and debit (bottom row) card transactions and sample metrics. Left column captures the distribution of card payment amounts from the 2022 and 2023 DCPC surveys, zooming in on values below $500. The distributions are skewed heavily right. Middle column captures the distributions of sample means for 5,000 samples of size 200 drawn from the DCPC transactions. The credit card distribution is roughly normal, with some right skew, and centered around $72, while the debit card distribution, which is centered around $62, has more kurtosis. Right panel captures the distribution of sample medians for the same simulation parameters. Distributions are approaching normality and centered around $18-21.

In the transaction simulator (see section 3.2), I draw payment transaction values from a lognormal distribution, which requires a dispersion parameter. Because I am using the distribution of sample medians to draw average amounts, I draw a payer's representative payment amount dispersion from the distribution of scaled median absolute deviations (MAD) as a proxy for the standard deviation.[11] Figure 3 captures the distribution of scaled MAD for the debit and credit card samples, which are roughly normal and centered around $21-24.

[11] The scaled MAD multiplies the MAD by 1.4826 to better approximate the standard deviation (Rousseeuw and Croux, 1993).

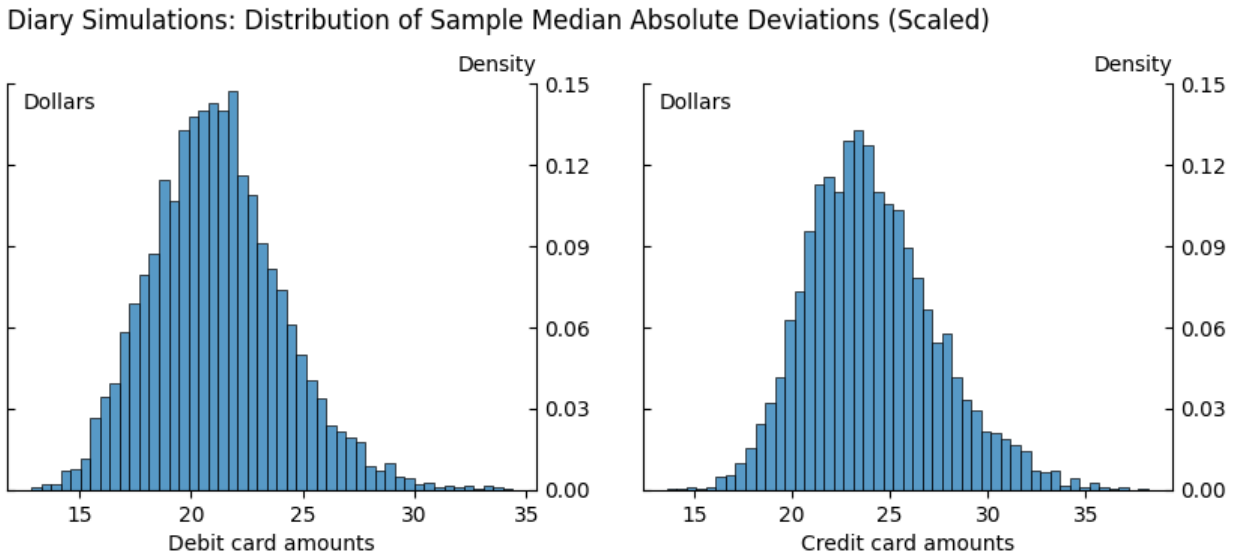

Figure 3. Distribution of sample median absolute deviations from the diary simulations. Figure depicts the distribution of scaled median absolute deviations for debit (left) and credit (right) card payment amounts from 5,000 samples of size 200 drawn from the 2022 and 2023 DCPC surveys. The distributions are roughly normal and centered around $21-23.

Turning to geography, payers and payees reside in square, $n\ x\ n$, grid, where each payer and payee is assigned x, y coordinates that are randomly drawn from a uniform distribution with a range of $[0, n)$, consistent with Le Borgne and others (2022). For this paper, I use $n = 200$. Geographic coordinates are the only payee-specific characteristics. The lack of payee-specific characteristics is a limitation of the methodology, and section 4.3 discusses this in more detail. I also assume a payer-to-payee ratio of 10:1. The ratio is based on estimates of the number of cardholding adults and businesses in the United States, for which I estimate a ratio of 7:1.[12] Because the number of business establishments used in the estimate includes non-retail establishments, 7:1 should be thought of as a lower bound on the ratio of cardholders to merchants. In the simulator, I assume a more conservative ratio of 10:1. Figure 4 captures the distribution of payers and payees in the grid.

[12] I calculate the percentage of consumers who have a debit or credit card from the 2023 survey of consumer payment choice (Foster, Greene, and Stavins, 2024) as 96.3 percent. I use the U.S. adult population of 262 million as of July 1, 2023 (U.S. Census Bureau, 2024a) to derive the number of cardholding adults as roughly 252 million. Finally, I use the total number of employer and non-employer business establishments in the United States of roughly 37 million in 2022 (U.S. Census Bureau, 2024b), the most recent data as of this writing, to calculate a cardholding adult to business establishment ratio of roughly 7:1.

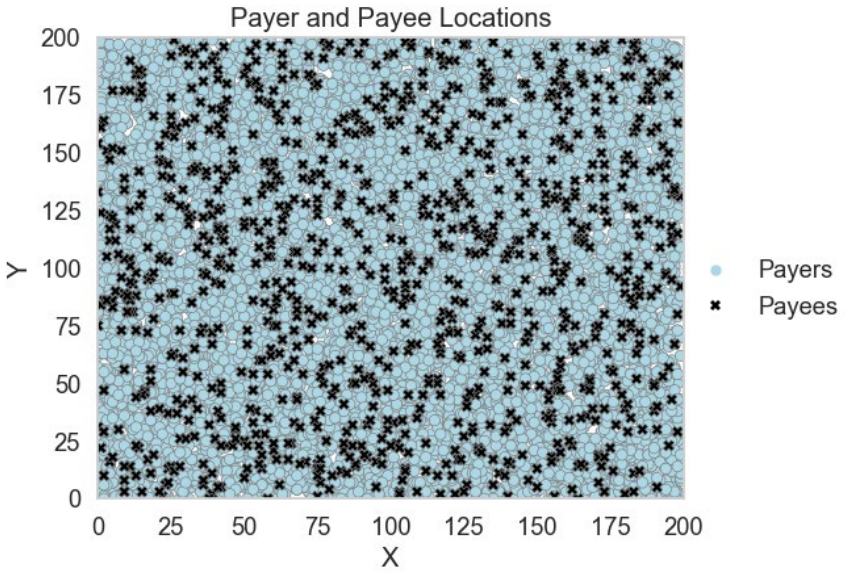

Figure 4. Payer and payee locations. The figure depicts the 200 X 200 grid in which payers and payees reside. There are 10,000 payers (light blue circles) and 1,000 payees (black x marks).

3.2 Transaction simulator

After establishing payer and payee characteristics, I run a transaction simulator that determines the number of daily payments, P, each payer makes and generates five features for each payment, which are defined in Table 1. The feature types or values are drawn from baseline distributions. The distributions for the number of daily payments, card type, location type, and payment amount are calibrated to real-world data, while the distributions for distance and time depend more heavily on assumptions about payment patterns, which are discussed in detail below.

Table 1. Payment transaction features

Feature	Description
Card type (C)	Debit or credit card
Location type (L)	In-person or remote
Amount (A)	Payment amount
Distance (D)	Distance between payer's home and payee
Time (T)	Time of day when transaction occurs

I assume that the daily number of payments a payer makes follows a Poisson distribution, which is a common assumption for simulating count data over a given unit of time. For each payer and for each day in the user-defined date range, I find the number of card payments, P, as

$$P \sim Poisson(\lambda), \quad (1)$$

where λ is the payer-specific average number of daily transactions derived from the DCPC.

Next, I draw the card type, C, and location type, L, where $C \in \{debit, credit\}$ and $L \in \{in\ person, remote\}$, from the following Bernoulli distributions, which are derived from the Federal Reserve Payment Study (FRPS) (Board of Governors, 2024)[13]:

$$\Pr(C = debit) = 0.62, \qquad \Pr(C = credit) = 0.38, \tag{2a}$$

$$\Pr(L = in\ person) = 0.64, \qquad \Pr(L = remote) = 0.36. \tag{2b}$$

I then draw the payment amount, A, from a lognormal distribution, which is bound at zero and skewed right, similar to the real-world payment amounts captured in Figure 2:

$$A \sim Lognormal(\mu, \sigma), \tag{3}$$

where μ, σ are transformed versions of the payer-specific debit and credit card payment amount averages and dispersion parameters derived from the DCPC.[14]

Simulating distance and time is more complex. Starting with distance, for each transaction, a payer pays a payee in the grid depicted in Figure 4. I assume that payers are more likely to make in-person payments closer to home. I assume that payers still make many remote payments at close distances (for example, takeout food orders), but many others (for example, online shopping payments) are made to payees that are farther away. I operationalize the distance framework through several steps. First, I calculate the Euclidean distance between each payer and payee. Second, for each payer, I sort the payees by distance in ascending order. Third, for each transaction, I select a distance index, D_i, from $D = (D_1, D_2, \dots, D_n)$, where n is the number of payees in the system, from a triangular distribution:

$$D \sim Triangular(a, b, c)\ , \tag{4}$$

where $a = D_1$, $b = D_n$, $c = D_n \cdot Q$, Q is a quantile that depends on L, and $Q_{in-person} < Q_{remote}$.

[13] The data come from the FRPS Networks, Processors, and Issuers Payments Surveys (NPIPS), released in November 2024. The data are from calendar year 2022 and are based on non-prepaid debit and credit card transactions. The probabilities captured in equations 2a and 2b are equal to the proportion of card transactions belonging to a given category. For example, with 89.1 and 55.3 billion non-prepaid debit and credit card transactions reported in the NPIPS, I calculate the debit card probability as: $\frac{89.1}{89.1 + 55.3} = 0.62$.

[14] Specifically, given the average amount (m) and dispersion (s), $\mu = \ln\left(\frac{m^2}{\sqrt{m^2+s^2}}\right)$ and $\sigma = \sqrt{\ln\left(1 + \left(\frac{s^2}{m^2}\right)\right)}$

In words, the distance index maps to the sorted payees. The minimum (a) and maximum (b) are constant for all transactions. They simply represent the first and last distance indices. The mode (c), or peak, of the triangular distribution varies based on whether the transaction is in-person or remote. It is controlled by an index quantile, Q, which is chosen such that in-person payments are more likely to occur at closer distances than remote payments. The modal quantiles default to 0.01 for in-person payments and 0.5 for remote payments (Figure 5).[15] The default settings are arbitrary but configurable. Users can experiment with alternative parameters. Finally, after drawing a quantile and finding the distance index, I look up the associated payee and distance.

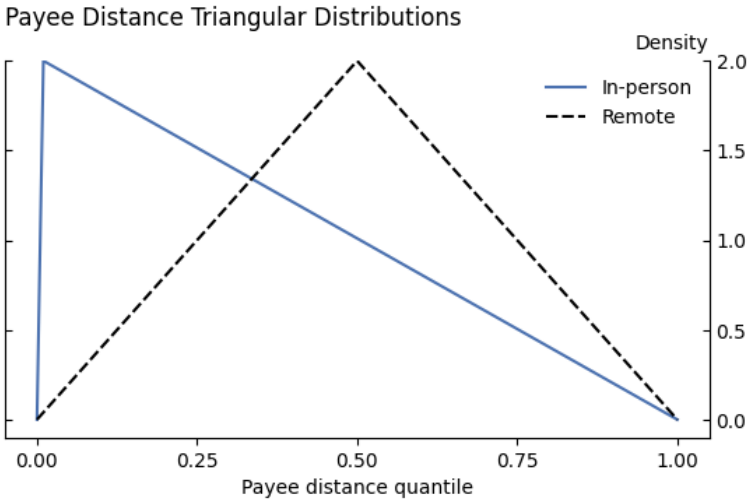

Figure 5. Payee distance triangular distributions. The figure depicts the quantiles associated with the triangular distributions underlying the payee distance simulation separately for in-person (blue solid line) and remote (black dashed line) payments. The minimum and maximum are constant for all observations. The mode, or peak, of the distribution is 0.01 and 0.5 for in-person and remote payments, respectively.

In simulating the transaction time, T, I assume that payments can occur any time of day, but they are more likely to occur during certain windows. Specifically, I assume there are significant spikes in payment activity during three windows: (1) around the morning commute as consumers pay for coffee and breakfast items; (2) lunchtime; and (3) early evening and dinnertime. I operationalize this by constructing a theoretical mixture distribution for transaction times throughout the day. I begin with a continuous probability density function (PDF) that is a weighted sum of component densities:

[15] The triangular distribution works well in simulating distance for several reasons. First, the underlying distribution is not known. Second, payers can, in principle, make payments at any distance within fixed boundaries. Finally, it is straightforward to shift the mode such that it reflects assumptions about payment distance patterns.

$$T \sim w_{baseline} Triangular(0, 12, 24) + \sum w_k Uniform(start_k, end_k)\,, \qquad (5)$$

where, $w_{baseline} + \sum w_k = 1$ and $k \in \{breakfast, lunch, dinner\}$.

The first component in equation 5 is a triangular distribution over the 24-hour period, peaking at noon, which captures the baseline daily transaction pattern. The remaining components are uniform distributions over the peak time windows. The weights and time windows are configurable, but they default to

$$w = \begin{cases} baseline = 0.4 \\ breakfast\ (7\ a.m. - 9\ a.m.) = 0.2 \\ lunch\ (11\ a.m. - 1\ p.m.) = 0.2 \\ dinner\ (6\ p.m. - 9\ p.m.) = 0.2. \end{cases}$$

In practice, evaluating the mixture PDF at each specific timestamp is computationally expensive, so I simplify the process by evaluating it for each hour and normalizing the results to a 0-1 interval, creating a discrete probability mass function (PMF) over the 24 hours of the day (Figure 6). I use the hourly probabilities to randomly select an hour for each transaction. Within each hour, I randomly select minutes and seconds from a uniform distribution to produce a final transaction time. While this approach is computationally efficient and a strong approximation, it sacrifices meaningful intra-hour variation, which could be valuable in modeling transactions that occur in quick succession.

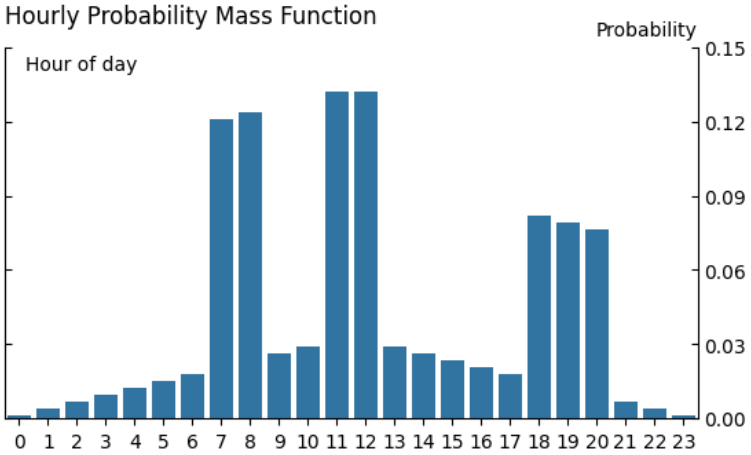

Figure 6. Hourly probability mass function. Figure depicts the probability that a payment transaction takes place in each hour of the day. The distribution has a baseline triangular shape that peaks at noon. It also has significant spikes in probability around breakfast, lunch, and dinner.

3.3 Fraud flag

The final phase of the simulation leverages Bayes' theorem to generate a ground truth fraud flag. The ability to label fraudulent transactions is not a given. Some financial institutions and payment systems have more established processes for labeling fraudulent transactions than others. The publicly available credit card fraud detection data posted by Dal Pozzolo (2016) includes a fraud flag, as does the simulation methodology developed by Le Borgne and others (2022). Importantly, the assumption that I can generate a fraud flag enables testing supervised learning classification approaches. Practitioners using real world data that do not have a fraud flag are largely confined to unsupervised anomaly detection approaches. With these considerations in mind, sections 3.3.1-3.3.2 discuss the two-step labeling process.

3.3.1 Marginal and conditional distributions

Using Bayes' theorem to produce a fraud flag requires having marginal and conditional distributions for each feature. For the binary variables, x_b, where $b \in \{C, L\}$, this is straightforward. The marginal probabilities, $P(x_b)$, were introduced in equations 2a and 2b and are derived from Board of Governors (2024). The conditional probabilities, $P(x_b|F)$, are derived from Gerdes, Greene, and Liu (2018) ("the payment fraud study").[16] Table 2 summarizes the card and location type probabilities that I use in applying Bayes' rule.

Table 2. Marginal and conditional probabilities of card and location type

Probability type:	Debit	Credit	In-person	Remote
Marginal: $P(x)$	0.62	0.38	0.64	0.36
Conditional: $P(x\|fraud)$	0.43	0.57	0.37	0.63

Notes: marginal probabilities derived from Board of Governors (2024); conditional probabilities derived from Gerdes, Greene, and Liu (2018); figures are based on non-prepaid debit and credit card transactions.

According to the FRPS, there are more debit than credit card transactions in the U.S. By contrast, the payment fraud study shows that among fraudulent payment card transactions, more are made with credit cards than debit cards. Location type has a similarly complex pattern. More payment card transactions are made in-person than remotely, but remote transactions account for

[16] Data from Gerdes, Greene, and Liu (2018) used to derive the conditional probabilities include: number of fraudulent credit card transactions (pg. 26, figure 19); number of fraudulent non-prepaid debit card transactions (pg. 27, figure 21); and distribution of fraudulent in-person and remote payments (pg. 29, figure 25).

a decisive majority of fraudulent transactions. This pattern owes, in part, to greater adoption of more secure payment methods for point-of-sale transactions, such as EMV chip cards and compatible card readers.

For the continuous variables, x_c, where $x_c \in \{A, D\}$, the marginal density, $pdf_{x_c}(u)$, is the PDF of x_c evaluated for each realized simulation value, u. I find the densities for payment amount and distance using the PDFs of the lognormal and triangular distributions defined in equations 3 and 4. The process for finding the conditional densities of these features, $pdf_{x_c|F}(u)$, is similar. I calculate the density at point u assuming that the value came from the fraud distribution, which is the baseline distribution shifted by a fraud factor. The assumed distribution of fraudulent payment amounts is

$$A_f \sim Lognormal(\mu_f, \sigma), \tag{6}$$

where u_f is a transformed version of the payer-specific average payment amounts scaled up by a fraud multiplier. I derive the multiplier from Gerdes, Greene, and Liu (2018, p. 27-28), who find that the average value of fraudulent non-prepaid debit and credit card payments are, respectively, 2 and 1.45 times the value of legitimate payments.

Similarly, the assumed distribution of fraudulent payment distances is

$$D_f \sim Triangular(a, b, c_f), \tag{7}$$

where I set the modal quantile for c_f to 0.75. Recall that the modal quantiles for in-person and remote payments are 0.01 and 0.5, respectively (see Figure 5). As in the marginal distribution, the fraudulent modal quantile is not empirically derived. It reflects the assumption that fraudulent payments are, on average, made to payees at farther distances from a payer's home.

Finally, for the transaction times, recall that a continuous mixture distribution (see equation 5) drives the underlying simulation process, but I discretize the distribution into an hourly PMF to improve computational performance (see Figure 6). The marginal probability of observing a transaction in a given hour is defined by the hourly PMF. I derive the conditional hourly distribution using the same process described in equation 5 and Figure 6, but I use different peak windows and mixture weights. In defining these elements, I assume that many fraudulent

transactions are likely to occur late at night and in the early morning. As with the marginal windows and weights, the precise settings are configurable, but they default to[17]

$$w_f = \begin{cases} baseline = 0.6 \\ late\ night\ (10\ p.m. - 12\ a.m.) = 0.3 \\ early\ morning\ (12\ a.m. - 5\ a.m.) = 0.1. \end{cases}$$

3.3.2 Bayesian estimation strategy

Given the marginal and conditional distributions of the five features, Bayes' theorem stipulates that we can derive the probability of fraud as

$$P(F|C, L, A, D, T) = \frac{P(C, L, A, D, T|F) \cdot P(F)}{P(C, L, A, D, T)}. \quad (8)$$

$Cardsim$ uses a naïve Bayes process to generate a fraud flag, which assumes conditional independence between predictor variables (Rish, 2001). In the present context, the assumption implies that the payment features are unrelated outside of their connection through fraud. Conditional independence is typically an unrealistic assumption, as it is in this case. Despite the strength of the assumption, naïve Bayes is a widely used classification algorithm, and the effect of assuming conditional independence often has a limited impact on performance (Zhang, 2004). Its scalability and capacity to generate probabilities based on priors and marginal and conditional distributions, without necessarily using pre-labeled data, which we do not have here, make it a good fit for the current use case. Using conditional independence and letting $P(\boldsymbol{x}|F)$ represent the product of the conditional probabilities and densities for the five features, we can find the probability of fraud by expanding the numerator and denominator of equation 8:

$$P(F|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|F) \cdot P(F)}{P(\boldsymbol{x}|F) \cdot P(F) + P(\boldsymbol{x}|\neg F) \cdot P(\neg F)}. \quad (9)$$

[17] In contrast to the marginal distribution, deriving the default weights for the distribution of fraudulent transaction times was an iterative process. I calibrated the weights defined in w_f to balance out the distribution of fraudulent payments across certain transaction windows. At 0.6, the baseline window is larger than one might expect if the goal is to concentrate fraudulent transactions late at night and in the early morning. However, because the marginal distribution includes significant spikes during certain windows and the Bayesian process discriminates between the marginal and conditional features in generating fraud probabilities, I found that using a relatively high weight during the baseline period was necessary to prevent unrealistic gaps in fraudulent payments during certain windows.

We do not initially have the complement, $P(\boldsymbol{x}|\neg F)$, but the simulator derives it by rearranging the law of total probability for each feature x_i:

$$P(x_i|\neg F) = \frac{P(x_i) - P(x_i|F) \cdot P(F)}{P(\neg F)}, \tag{10}$$

where all the elements on the right-hand side of equation 10 are available.

Ultimately, I use the odds form of Bayes' theorem (Gelman and others, 2014, p. 8; Hastie, Tibshirani, and Friedman, 2009, p. 234), which, at a high level, is: $Posterior\ Odds = Prior\ Odds + Likelihood\ Ratio$. In the present case, the odds form is given by

$$\frac{P(F|\boldsymbol{x})}{P(\neg F|\boldsymbol{x})} = \frac{P(F)}{P(\neg F)} \cdot \frac{P(\boldsymbol{x}|F)}{P(\boldsymbol{x}|\neg F)}. \tag{11}$$

The odds form facilitates operationalizing the labeling methodology. I generate fraud labels using a naïve Bayes ranking approach, wherein I rank each record by posterior odds in descending order and label the top $n\%$ as fraud. Ranking is a very common application of naïve Bayes. It is motivated, in part, by the fact that the precise probabilities generated by Bayes' rule with multiple features are often poorly calibrated, but the relative magnitude of the predictions are meaningful (Zhang and Su, 2004). Further, the rank order of the posterior odds is equivalent to that of the posterior probabilities, and it is easier to generate odds in a computational setting. The odds form enables splitting out the prior odds (see equation 11) and breaking down the likelihood ratio into individual likelihood ratios for each feature (see equation 12). This modular approach simplifies the calculation considerably, is easy to update, and facilitates the use of fast, array-based programming methods.

$$Likelihood\ ratio = LR_C \ldots \times \ldots LR_T = \frac{P(C|F)}{P(C|\neg F)} \ldots \times \ldots \frac{P(T|F)}{P(T|\neg F)} \tag{12}$$

4 Results

This section presents the characteristics of an example $CardSim$ dataset based on the following high-level simulation parameters: 10,000 payers, 1,000 payees, and 365 days. With these parameters, the simulator produced 3.16 million records. The output includes eight primary fields that are summarized in Table 3. Section 4.1 examines feature distributions in the example dataset,

and 4.2 explores how computational performance varies with the number of payers and the number of days. Section 4.3 addresses some limitations of the simulation methodology.

Table 3. Simulator output fields

Field	Description
date_time	The transaction datetime; transactions are sorted chronologically
payer_id	A unique identifier for the payer
payee_id	A unique identifier for the payee
amount	The payment amount
credit_card	1 if the payment is made using a credit card and 0 if using a debit card
remote	1 if the payment is made remotely and 0 if in-person
payee_distance	The Euclidean distance between the payer's home and the payee
fraud	1 if the payment is fraudulent and 0 if it is legitimate

Importantly, in the example simulator run, I assume both a prior fraud probability and a fraud ranking threshold of one percent. These parameters are separate in the simulator, but they both default to one percent. As discussed in 3.3.2, I use a ranking approach to label fraud transactions. A one percent fraud ranking threshold guarantees that one percent of transactions will be labeled as fraud. This assumption is high by many official sector and industry estimates (Gerdes, Greene, and Liu, 2018; EBA and ECB, 2024; Australian Payments Network, 2023). However, publicly available fraud statistics likely underestimate fraud because they often focus on fraudulent transactions that have cleared and settled. This means that they may not capture attempted fraud that did not clear and settle, including denied transactions and those in which the payment amount was returned to the payer before settlement (Gerdes, Greene, and Liu, 2018). Furthermore, some fraudulent payment card transactions occur without a cardholder's knowledge, and some victims, especially in vulnerable elderly segments, choose not to report fraud. The one percent rate also preserves the challenges of modeling with extreme class imbalance. Nevertheless, as with other parameters in the simulator, the prior fraud rate and ranking threshold can be tuned.

4.1 Feature distributions

Figure 7 depicts the simulation results for the distributions of card (left panel) and location (right panel) types across payment class. The distributions of features for the legitimate transactions are identical to the marginal probabilities presented in Table 2. This is unsurprising given the low fraud rate and over three million records in the simulator output. The distributions of fraudulent transactions are more skewed toward the feature with the higher conditional probability. For

example, as captured in Table 2, $P(C = credit|F) = 0.57$, while the realized proportion of fraudulent transactions made with a credit card in the simulator is 0.69. A similar pattern holds for the location type.

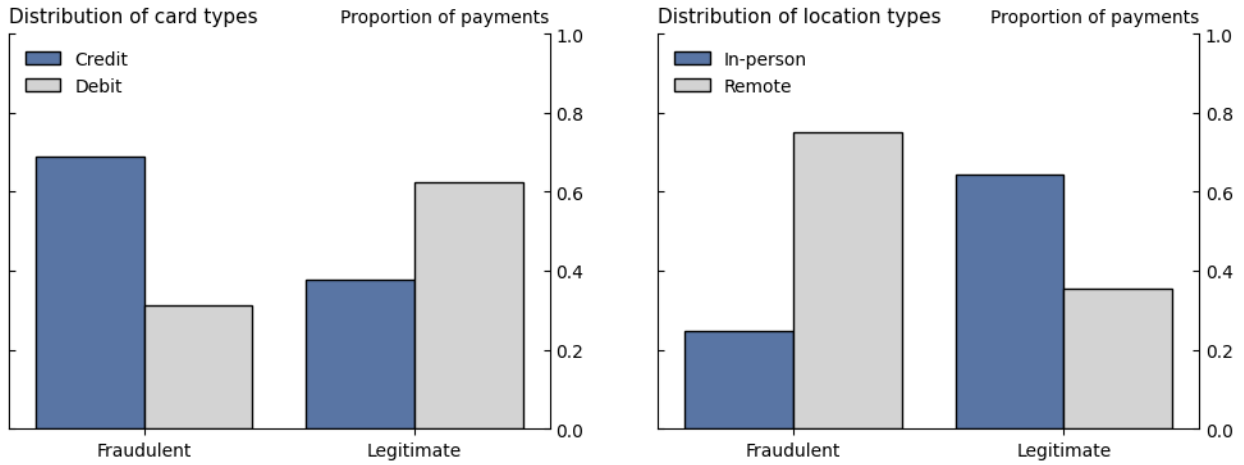

Figure 7. Feature distributions: card and location types. Left panel captures the distribution of card type (credit, debit) across payment class. Credit card payments (blue bars) account for 69 percent of fraudulent transactions and 38 percent of legitimate transactions. Debit card payments (gray bars) account for 31 percent of fraudulent transactions and 62 percent of legitimate transactions. Right panel captures the distribution of location type (in-person, remote) across payment class. In-person payments (blue bars) account for 25 percent of fraudulent transactions and 64 percent of legitimate transactions. Remote payments (gray bars) account for 75 percent of fraudulent transactions and 36 percent of legitimate transactions.

There are at least two explanations for why the naïve Bayes approach favors feature types with higher conditional probabilities in predicting fraud. The first revolves around the strength of the conditional independence assumption. Naïve Bayes uses each feature in isolation, but the features are likely correlated outside of their relationship through fraud. Thus, the model puts too much weight on certain feature outcomes, when in reality, at least part of the signal from those features is redundant information. A second reason may be due to ranking with incomplete predictive power. The simulator ranks transactions based on posterior odds, but we do not have a complete set of features that are relevant for predicting fraud. Thus, naïve Bayes may be giving too much weight to the features that are present in the model. This pattern is easiest to see with card and location types because it is straightforward to compare outcomes using simple proportions, but there is evidence that the skew persists with the other features as well.

Figure 8 focuses on the payment amounts generated by the simulator. The top panel captures the distributions of debit (left) and credit (right) card payments, zooming in on those

below $200. The distributions are lognormal and skewed right. The global average debit and credit card payment amounts are $18.15 and $20.90, respectively. The bottom panel plots kernel densities of legitimate and fraudulent payment amounts separately for debit (left) and credit (right) card payments. For both card types, the distribution of legitimate payment amounts is much more concentrated at lower values than fraudulent payments. The difference in the distributions of legitimate and fraudulent debit card payment amounts is more extreme than that of credit card payments, consistent with the higher debit card fraud value multiplier discussed in 3.3.1.

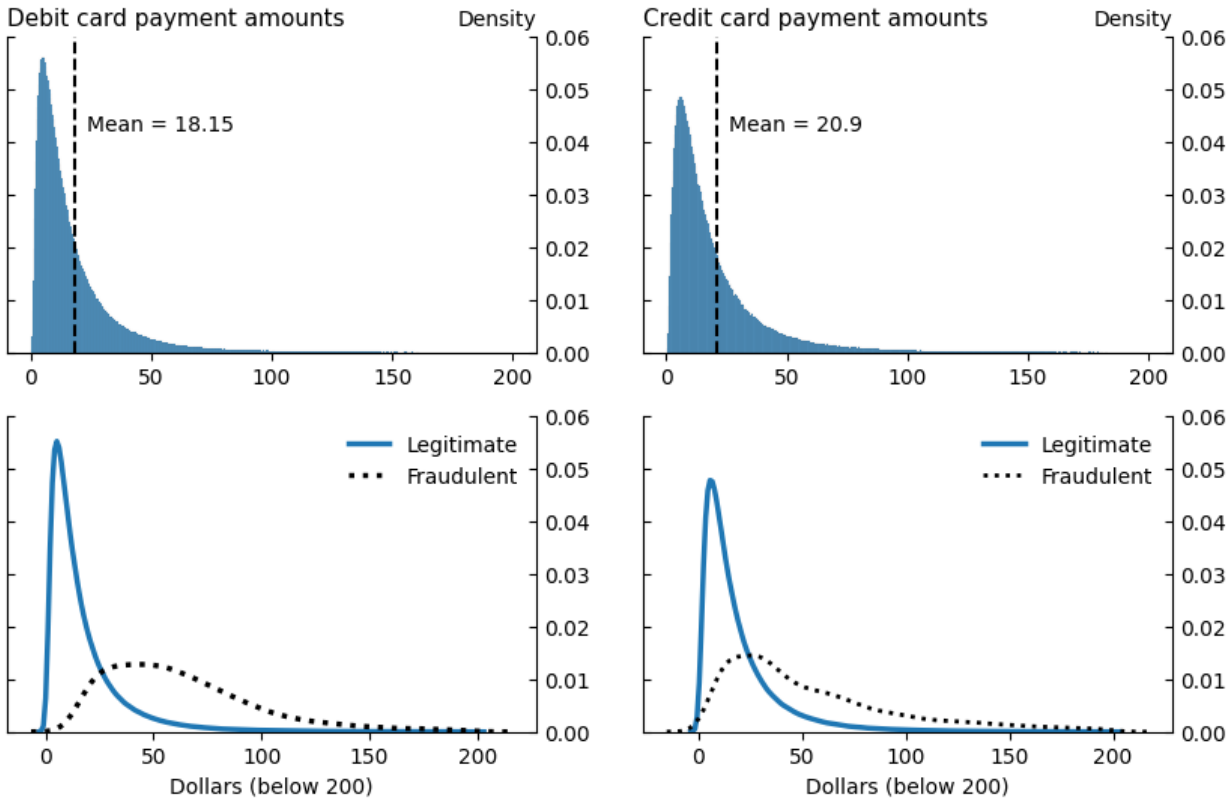

Figure 8. Feature distributions: payment amount. Top panel depicts the distribution of simulated debit (left) and credit (right) card transaction amounts, zooming in on payments below $200. The global means for debit and credit card transactions are $18.15 and $20.90, respectively. Bottom panel depicts kernel density estimates for the distributions of payment amounts for legitimate (solid blue line) and fraudulent (dotted black line) transactions separately for debit (left) and credit (right) card payments. The difference in the legitimate and fraudulent payment amount distribution is more extreme for debit than for credit card payments.

Turning to distance, Figure 9 depicts kernel densities for the distributions of Euclidean distances between a payer's home and the payee for legitimate and fraudulent transactions separately for in-person (left) and remote (right) payments. The distribution of legitimate in-person payments is more concentrated at closer distances than legitimate remote payments. For both types

of payments, fraudulent payments tend to occur at farther distances, but the gap is more pronounced for in-person payments. These results are consistent with the modal quantiles selected for the triangular distributions associated with the marginal and conditional distance distributions.

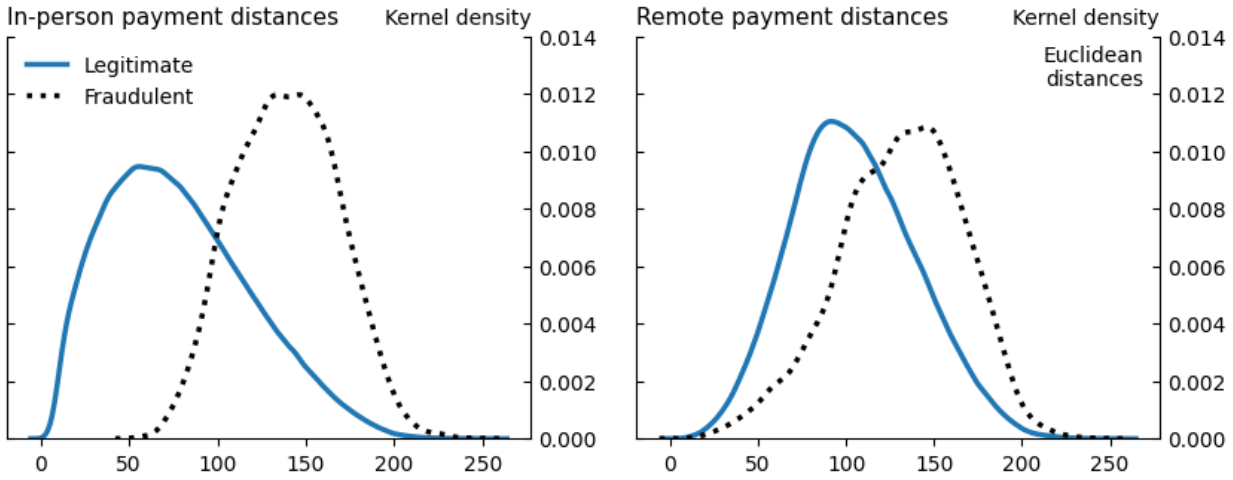

Figure 9. Feature distributions: payee distances. Figure depicts kernel density estimates for the distributions of payee distances for legitimate (solid blue line) and fraudulent (dotted black line) transactions separately for in-person (left) and remote (right) payments. The distribution of legitimate in-person payments is more concentrated at closer distances than legitimate remote payments. For both types of payments, fraudulent transactions are more widely distributed.

Finally, Figure 10 captures the proportion of legitimate (left) and fraudulent (right) payments occurring within each hour. For both payment classes, transactions occur throughout the day. The distribution of legitimate payment times mirrors the hourly PMF presented in Figure 6. There is a baseline daily transaction pattern following a triangular distribution that peaks at noon and significant intraday seasonality, with spikes in payment activity around breakfast, lunch, and dinner. Fraudulent payments peak late at night and early in the morning, as expected, but the relative balance of transactions in these windows differs from the conditional mixture distribution weights, which were higher for the late-night window. The reason is that there are more payments overall in the early hours, and that effect overpowers the window weights.

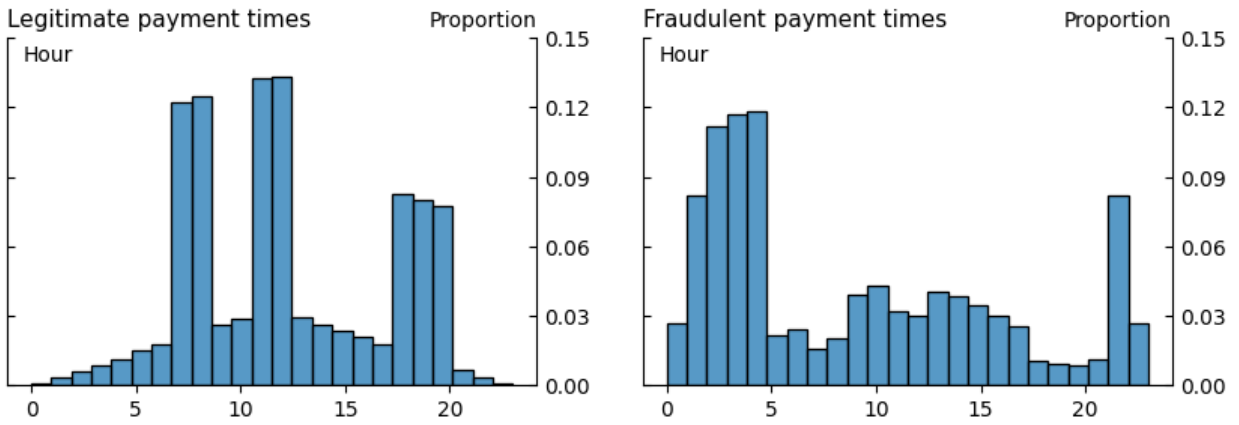

Figure 10. Feature distributions: transaction time. The figure captures the distributions of simulated payment times within each hour for legitimate (left) and fraudulent (right) transactions. Legitimate and fraudulent transactions occur throughout the day, but the former peak around breakfast, lunch, and dinner, while the latter peak late at night and in the early morning.

4.2 Computational performance

Cardsim uses an array-based programming approach (Oliphant, 2007; Harris and others, 2020) that supports scalability and computational efficiency. The simulation underlying the results discussed in 4.1, which produced 3.16 million records, ran in about 10 seconds.[18] Figure 10 shows how the simulator's performance varies with the number of payers and the number of days a user specifies. Performance is much more sensitive to the number of payers (top left) than the number of days (top right). Holding the number of days and payer-to-payee ratio constant at 365 and 10:1, respectively, the runtime grows from 3 seconds with 5,000 payers to 171 seconds with 45,000 payers, and the relationship is non-linear. The runtime only grows from 7 seconds with 180 days to 27 seconds with 1,620 days, holding the number of payers constant at 10,000. New payers and, by connection, payees introduce more complexity into the environment, which enriches the simulation but comes at a higher computational cost.

Meanwhile, the number of records the simulator produces grows linearly with both the number of payers (bottom left) and the number of days (bottom right). For both parameter ranges, the number of records grows from roughly 1.6 million to 14 million at a constant rate. A good strategy for increasing the complexity and size of the simulation output would be to use both the number of payers and days in concert.

[18] All performance tests were carried out on a cloud-based analytics instance that had the equivalent of about 34 Gigabytes of memory and eight CPUs.

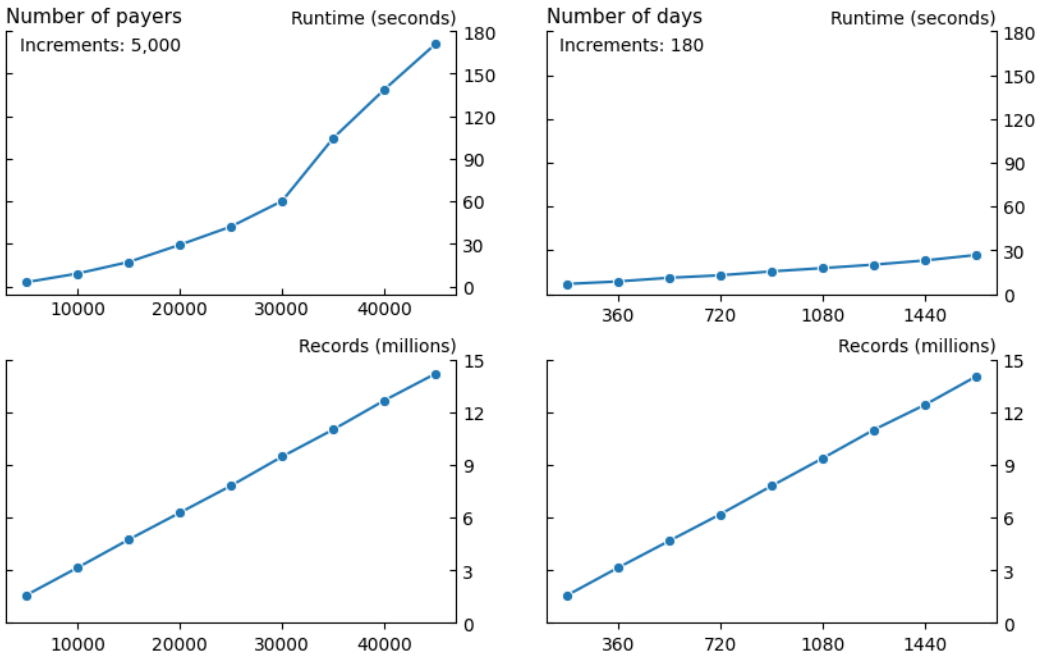

Figure 11. Simulator performance. Figure illustrates how simulator performance varies with the number of payers and number of days. Left column focuses on the number of payers, ranging from 5,000 to 45,000 in increments of 5,000. Right column focuses on the number of days, ranging from 180 to 1,620 in increments of 180. The top row illustrates how runtime, in seconds, reacts to varying the parameters. The runtime grows non-linearly from 3 to 171 seconds over the payers range, while it only grows from 7 to 27 seconds over the days range. The bottom row illustrates how the number of records grows in response to the parameters. There is a linear relationship between the number of records and both parameters. The records grow from roughly 1.6 to 14 million. Tests were carried out on a cloud-based analytics instance that had the equivalent of about 34 Gigabytes of memory and eight CPUs.

4.3 Limitations

Contrary to some existing simulation methodologies, the simulator does not attempt to capture fraud typologies, such as spending sprees and compromised merchants. Payment class draws are independent events. Whether or not a given payment is flagged as fraudulent has no effect on subsequent transactions. Rather, the simulator focuses on correlating important payment properties with fraud via Bayes' theorem. Integrating fraud patterns based on typologies tends to be more deterministic and rule based. An advantage of the *Cardsim* approach is that ML models are generally better at uncovering probabilistic relationships than deterministic patterns. Therefore, the simulation methodology could help researchers and practitioners gain a better understanding of model strengths and weaknesses. The disadvantage is that certain types of common feature engineering approaches in fraud detection focused on recency and frequency of payments make

no difference in testing models with the simulator output. $Cardsim$ is meant to complement, not replace, other methodologies.

The focus on unauthorized fraud and use of a simplified set of payer and payee characteristics also limit the ability to test several relevant modeling and feature engineering approaches. For example, the complexity of authorized fraud schemes may shift the focus to identifying fraudulent payee networks, rather than fraudulent transactions. Additionally, financial institutions and payment system operators typically have access to a richer set of demographic and economic indicators for payers and payees within a payment network. In the real world, fraud detection models might create profiles of payers and merchants based on certain economic and demographic factors, such as the payer's age or income and the merchant's size or retail category.

More broadly, any simulator is an imperfect representation of reality. While the parameters selected in the simulator are chosen carefully and are meant to be plausible, some are not based on real world data or economic theory. As discussed throughout, the parameters used in the simulator can be modified. Indeed, the simulation methodology can be used to test models under a wide range of fraud patterns. The main goal of the simulator is to enable researchers and practitioners to test the mechanics of important fraud detection modeling approaches. Section 5 demonstrates that $CardSim$ outputs can be used to accomplish this.

5 Using the data for machine learning

Having generated payment transaction data, I show how the data can be used to test and evaluate ML models and workflows for fraud detection. This section demonstrates data processing, model training and testing, and tuning methodologies on the simulator output. As discussed in 3.3, the simulator assumes that a fraud flag is available for model training. This assumption enables testing supervised learning-based classification approaches, which are the primary focus of the model testing. The section concludes by interpreting feature importance using the SHAP framework.

5.1 Data processing

It is straightforward to apply common ML data partitioning strategies to the simulated data. In this paper, I use a proportional split approach to divide the data into training and testing sets, where the training set contains 70 percent of the 3.16 million transactions produced by the simulator, and the test set contains the remaining 30 percent. In some ML settings, it may be appropriate to randomly

sample records for the training and testing splits, but payment transactions are temporal in nature. Thus, it is more realistic to split the data chronologically to ensure that the testing data are both out of sample and out of time.[19]

The simulator output enables carrying out common feature engineering approaches that can have a meaningful impact on predictive accuracy. Researchers can use the timestamp to create various intraday seasonality indicators. In the model testing, I use four time-of-day features defined in table 4. Additionally, transformations are commonly applied to skewed variables to compress their distribution. The simulated payment amounts are skewed right. The model tests use the natural log of payment amounts as a predictor. Because I use a lognormal distribution to draw payment values, the log of the raw distribution will be normally distributed. Indeed, this is the definition of a lognormal distribution. In practice, the natural log of payment card transaction values may not be precisely normal, but the transformation is likely to behave similarly.[20]

Table 4. Time-of-day features

Feature	Definition
late	1 if a transaction occurs between 10 p.m. and 5 a.m., 0 otherwise
breakfast	1 if a transaction occurs between 7 a.m. and 9 a.m., 0 otherwise
lunch	1 if a transaction occurs between 11 a.m. and 1 p.m., 0 otherwise
dinner	1 if a transaction occurs between 6 p.m. and 9 p.m., 0 otherwise

It is also possible to create some behavioral analytics predictors for the payment amount variable. For example, I create an indicator that captures deviations of log payment amounts from payer-card type averages. In contrast to the payment amount feature, the deviation feature is personalized for individual payers. Figure 12 depicts the distribution of the deviation variable for credit and debit card payments in the training dataset. Crucially, when using aggregate metrics, such as means and medians, in developing features, one should only use the aggregation metric calculated in the training data to transform variables in both the training and test sets. In the real world, forecasters will only have access to training data metrics ex ante. Using global aggregations inclusive of the test data would be considered data leakage in a model testing context.

[19] One can also apply k-folds cross-validation, which involves training and testing the model k times, leaving out a different slice of the data as the test set at each iteration. While this approach is generally more rigorous than the proportional split method, it is difficult to enforce chronology and may not be appropriate in the fraud detection context. I also tested the models using 5-folds cross-validation. The results are very similar.

[20] The distribution of the log of raw payment values from the DCPC data used in the simulation are bell shaped but have a good bit more kurtosis than the distribution of the log of simulated transaction values.

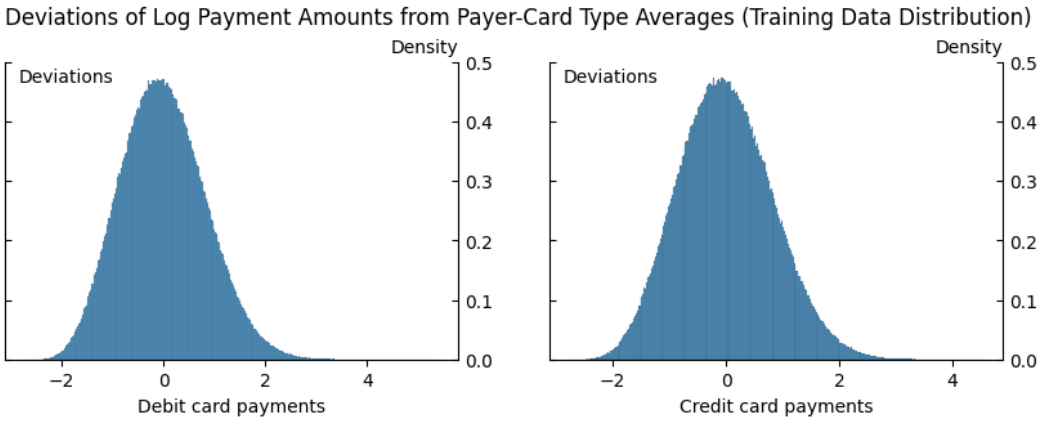

Figure 12. Distribution of the payment amount deviations feature by card type. The figure depicts distributions for deviations of log payment amounts from payer-card type averages separately for debit (left) and credit (right) card transactions. The distributions are normal.

5.2 Evaluation metrics

Fraud detection is a class imbalance challenge—that is, there are usually far more observations in the majority class of the target variable than in the minority class. In other words, fraud is rare. The simulator captures this through a default 99:1 ratio of legitimate to fraudulent transactions (Figure 13). Class imbalance makes it difficult for ML models to learn the distinguishing characteristics of minority class observations. This is a common challenge in machine learning. Some techniques can help mitigate the issue, including various re-sampling and asymmetric misclassification cost methods. Section 5.4 illustrates an implementation of the former.

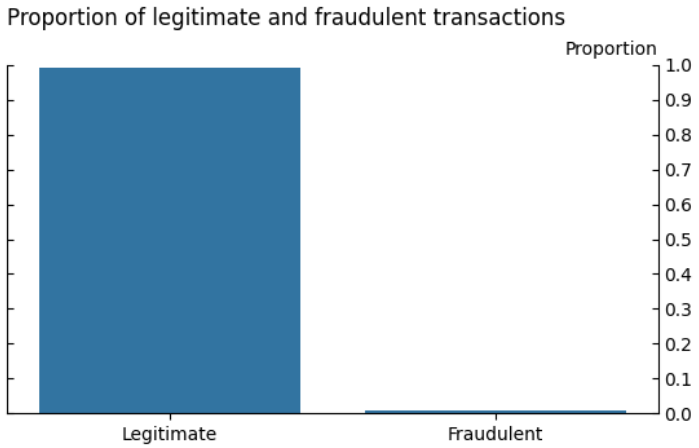

Figure 13. Distribution of transaction class in the simulator output. The figure captures the proportion of legitimate and fraudulent transactions generated by the simulator. There is a 99:1 ratio of legitimate to fraudulent transactions.

All the metrics I use to judge the model results presented in section 5.3 are derived from the confusion matrix. Figure 14 illustrates a common form of the confusion matrix. Perhaps the most important implication of class imbalance is that some evaluation metrics are more meaningful than others in the presence of imbalance. Table 5 summarizes the four metrics I use to judge the relative performance of the competing models that I test.

Actuals	Predictions	
	0	**1**
0	True negatives (TN)	False positives (FP)
1	False negatives (FN)	True positives (TP)

Figure 14. The confusion matrix. Figure depicts a common form of the confusion matrix, which captures the intersection of actual and predicted classes.

Table 5. Accuracy metrics used to evaluate model performance

Metric	Description
Precision	Otherwise known as "positive predicted value." Captures the share of fraud predictions that were fraudulent. Calculated as: TP / (TP + FP).
Recall	Otherwise known as "sensitivity" or "true positive rate." Captures the share of fraudulent transactions correctly identified by the model. Calculated as: TP / (TP + FN).
F1 score	A metric that summarizes precision and recall. Calculated as the harmonic mean of precision and recall.
Area under the precision-recall curve (AUPRC)	Represents the area under the curve that plots pairs of precision and recall for a range of classification thresholds.

In 5.3, I show results for four other performance metrics that are uninformative or controversial in the class imbalance context. First, overall accuracy represents the proportion of correctly classified records. A model can achieve 99 percent accuracy in this dataset by simply predicting the majority class (legitimate) every time. Such a model would be considered highly accurate in a strict sense, but it is worthless in fraud detection because it picks up on no fraudulent transactions. Second, the false positive rate (FPR), which is calculated as FP / (FP + TN) (see figure 14), is typically very low with extreme class imbalance because true negatives are usually the most dominant modeling outcome, leading to a very large denominator. Third, specificity is calculated as TN / (TN + FP), or, equivalently, 1 – FPR. For the same reasons that FPR is typically very low, specificity is usually very high. Finally, the area under the receiver operating characteristics (ROC) curve is frequently used in judging classifier performance. It is similar to the AUPRC (see Table 5), but the ROC curve represents pairs of true and false positive rates for various classification

thresholds. A considerable body of research shows that the area under the ROC (AUROC) curve can be highly inflated in the presence of class imbalance, in part because the FPR is often extremely low (Davis and Goadrich, 2006; Jeni, Cohn, and De La Torr, 2013; Saito and Rehmsmeier, 2015; Sofaer, Hoeting, and Jarnevich, 2019).

5.3 Model training and performance

I train three classification models on the training data that are commonly used in supervised learning: logistic regression, XGBoost, and a multilayer perceptron (MLP). I refer to these as the candidate models. Table 6 summarizes these approaches, which represent three different types of models. I compare the candidate models' performance to a Gaussian Naïve Bayes (Gaussian NB) model, a common benchmark for classification. The benchmark model is very similar to the naïve Bayes approach used to construct the data in the first place but with two important differences. First, it uses a normal distribution for the continuous variables—in this case, distance and payment amount—which differs from the lognormal and triangular distributions used to simulate these variables. Second, the benchmark model has access to the engineered variables—specifically, the time-of-day features (see Table 4) and the payment amount deviations (see Figure 12).

Table 6. Overview of candidate models

Model (type)	Description
Logistic regression (linear)	Widely used method for modeling binary response variables. Involves estimating the probability of an event as a linear function of a set of predictor variables after applying non-linear transformations to the response and predictor variables to ensure predictions are probabilities ranging from 0-1. Advantages of logistic regression include speed, relative simplicity, and interpretability. The main drawback is typically lower predictive accuracy than more advanced ML approaches.
XGBoost (tree-based ensemble)	A scalable gradient boosting machine that fits a sequence of decision trees, where each tree focuses on the errors of the ensemble of previous trees. XGBoost is a widely deployed ML approach for business use cases because of its predictive and computational performance. It is generally more interpretable than neural networks but less so than logistic regression.
Multilayer perceptron (neural network)	A fully connected feed-forward neural network that involves connecting an input layer with several nodes (the predictors) to an output layer (the outcomes) through one or more hidden layers with several nodes, which apply activation functions to transform the input data. The model is often quite accurate and capable of capturing complex relationships, but it is less interpretable, slower, and may be more prone to overfitting than the logistic regression and XGBoost.

After training the models, I evaluate their predictive accuracy over the testing set. Table 7 summarizes model performance across the key metrics defined in Table 5 above. For robustness, I measure the average performance of the models over five different training and testing sets using the same parameters but different reproducibility seeds. The candidate models display 76-81 percent precision and 57-65 percent recall. That is, 76-81 percent of the fraud predictions the models make are fraudulent, and the models pick up on 57-65 percent of fraudulent transactions. Overall, the XGBoost and MLP neural network outperform the logistic regression. The neural network demonstrates the best precision, while XGBoost has the best recall by a decisive margin. The candidate models are much more balanced than the benchmark Gaussian NB, which sacrifices a lot of precision for very high recall.

Table 7. Summary of model performance

Model	Precision	Recall	F1	AUPRC
Gaussian NB	0.183	0.832	0.300	0.512
Logistic	0.756	0.569	0.650	0.750
XGBoost	0.774	0.647	0.705	0.816
MLP	0.807	0.585	0.676	0.807

Note: average performance across five simulator runs.

The F1 score and AUPRC summarize performance across precision and recall. The candidate models have much higher summary scores than the benchmark model. Among the candidate models, the XGBoost has the highest overall summary scores, but its AUPRC is comparable to that of the neural network. It is helpful to visualize the AUPRC through a precision-recall plot, which captures precision-recall pairs for various classification thresholds (Figure 15). The areas under the XGBoost (blue dashed line) and neural network (black dotted line) curves are comparable and greater than the area under the logistic regression curve (orange solid line).

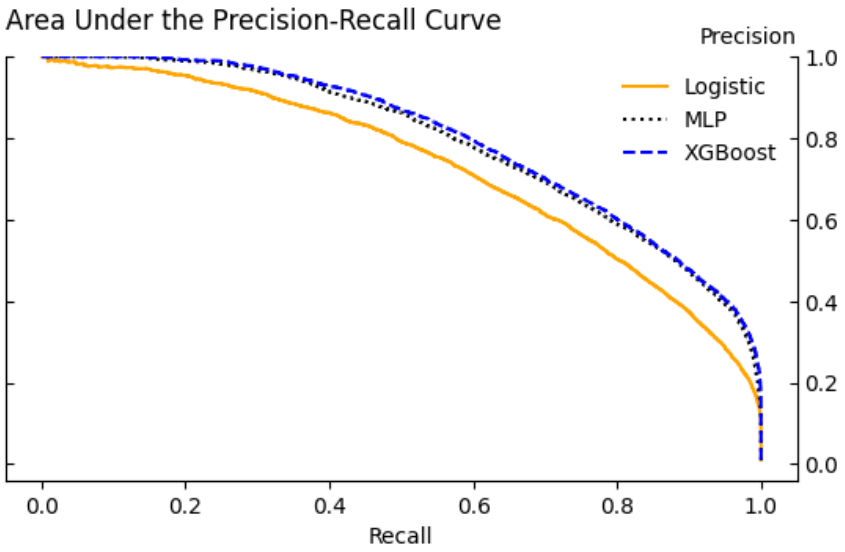

Figure 15. Precision-recall plot. Figure captures precision-recall pairs for various classification thresholds for the logistic regression (orange solid line), XGBoost (blue dashed line), and MLP neural network (black dotted line). The AUPRC for the XGBoost and neural network are comparable and greater than that of the logistic regression.

Speed is an important consideration in addition to predictive accuracy. Table 8 summarizes the candidate models' computing time, in seconds, for inference and training. The former captures the time it takes to make predictions for the roughly 947,000 records in the test set. The latter represents the time needed to fit the models to the roughly 2.2 million records in the training set. The logistic regression is the fastest on the inference dimension, while the XGBoost and neural network are comparable. In some ML applications, inference time matters more than training time because models may not be trained very often. Because fraud patterns are constantly changing, fraud models are likely to be re-trained frequently. In terms of training time, XGBoost is the most performant, while the neural network shows considerable performance overhead. It was about 5.5 and 14.1 times slower than the logistic regression and XGBoost, respectively.

Table 8. Inference and training time (in seconds)

Model	Inference	Training
Logistic	0.12	20.65
XGBoost	0.84	8.11
MLP	0.88	114.32

Notes: average of five model runs; average testing and training set records: 947,314 and 2,210,401.

As introduced in 5.2, several metrics are uninformative or can be problematic in a class imbalance setting. Table 9 captures the overall accuracy, FPR, specificity, and AUROC metrics for

the three candidate models, the benchmark model, and a dummy model, which simply predicts the majority class (zero) every time. The dummy model achieves 99 percent accuracy, but it identifies no fraudulent transactions. The models have extremely low FPR and high specificity. When it comes to AUROC, the candidate models show very strong and indistinguishable performance. The ROC curve plots pairs of true and false positive rates for various classification thresholds. In this case, it appears that the AUROC is inflated because of the extremely low FPR.

Table 9. Uninformative or problematic metrics in the class imbalance context

Model	Accuracy	FPR	Specificity	AUROC
Dummy	0.990	0.000	1.000	0.500
Gaussian NB	0.961	0.038	0.962	0.970
Logistic	0.994	0.002	0.998	0.996
XGBoost	0.995	0.002	0.998	0.997
MLP	0.994	0.001	0.999	0.997

It is easier to see the drivers of the metrics listed in Table 9 by reviewing a confusion matrix for one of the models. Figure 16 shows the MLP confusion matrix. TN are by far the most common modeling result, over 160 times bigger than the nearest class.[21] This derives from the dominance of legitimate transactions in the data.

Actuals	Predictions	
	0	**1**
0	TN: 936,278	FP: 1,590
1	FN: 3,649	TP: 5,755

Figure 16. MLP confusion matrix. Figure depicts the confusion matrix results for the MLP neural network. The matrix illustrates the dominance of true negatives.

5.4 Tuning and re-sampling

Practitioners can pursue various methods to refine fraud detection models and calibrate them to specific objectives. I show how to implement two types of model refinements using XGBoost as context. First, I perform a grid search to optimize several hyperparameters. Grid search involves fitting models with various combinations of hyperparameters and finding the set that yields the best performing model for a given objective. I use recall as the performance objective. Table 10 identifies and describes the four hyperparameters that I include in the grid search. The table also

[21] The MLP also has very few false positives, which helps explain the high precision captured in table 7.

captures the defaults for those parameters and the value the search selects as optimal. The routine selects the default parameter for boosting rounds and the max tree depth.

Table 10. Hyperparameters for the XGBoost grid search

Hyperparameter	Choice set	Description
Boosting rounds	{50, 100, **200**}	How many trees to build.
Max depth	{3, **6**, 10}	Maximum number of splits per tree.
Subsample	{0.6, **0.8**, 1}	Fraction of features to randomly sample and use as predictors for each tree. Random feature sampling can help protect against overfitting.
Learning rate	{**0.1**, 0.3, 0.6}	Step size at each iteration in minimizing the loss function. Bigger values speed up the model but risk stepping over the best minimization path. Smaller values slow the model but are more likely to find optimal pathways.

Note: Bold indicates parameter was selected during grid search. Underlined indicates default parameter.

To help address class imbalance, I also test randomly re-sampling the training data such that there is a 25:1, rather than 99:1, ratio of legitimate to fraudulent transactions. Table 11 captures the accuracy metrics for the for the XGBoost models using grid search (Tuned) and re-sampling (Down sampled) compared to the default model. The tuned model performance is virtually identical to that of the default model, with a very modest gain in recall. The re-sampling results are more compelling. The model shows considerable gains in recall to the detriment of precision. If practitioners have a higher tolerance for false positives and have prioritized identifying fraudulent transactions, re-sampling could be a promising strategy. In some cases, multiple models that target different objectives may be deployed, and an ensemble of predictions could be used.

Table 11. Model performance for the tuned and down sampled models

Model	Precision	Recall	F1	AUPRC
Default	0.775	0.643	0.703	0.813
Tuned	0.773	0.646	0.703	0.816
Down sampled	0.569	0.856	0.683	0.805

5.5 Interpretability

Recent years have seen advancements in research on “explainable AI” or “interpretable machine learning,” which revolves around summarizing how ML models transform inputs into outputs

(Molnar, Casalicchio, and Bischl, 2020; Linardatos, Papastefanopoulos, and Kotsiantis, 2021).[22] The SHapley Additive exPlanations (SHAP) interpretability framework (Lundberg and Lee, 2017), which is based on a game theoretic approach to allocating the marginal contribution of each predictor to model outcomes, has become widely used in supervised learning approaches, such as tree-based ensembles and neural networks. The framework is attractive because it enables global and local estimates of predictor importance. The former involves summarizing the overall effect of a predictor, while the latter revolves around interpreting the influence of predictors on individual predictions. Another useful property of the SHAP framework is that it addresses both directionality and magnitude. Like many other novel ML interpretability approaches, SHAP does not quantify the uncertainty of predictor importance estimates.

I show how SHAP values can be used to interpret global and local predictor importance for the XGBoost model. I also demonstrate some of the challenges of SHAP's application to XGBoost. Critically, the results presented here are sensitive to the $CardSim$ calibrations and should not be interpreted for any substantive significance. The purpose is to demonstrate the mechanics of the SHAP approach, rather than specific insights about the nature of fraud. SHAP values can actually serve as a guide for tuning the simulator's parameters. If they show that a given predictor is driving model outcomes more than would be expected under specific circumstances, end-users can adjust the default parameters accordingly.

Figure 17 displays a SHAP swarm plot that summarizes, in log-odds, how each feature moves a prediction from a base value, which represents the expected value of the prediction in the absence of any feature variables (Lundberg and Lee, 2017, p. 5). The summary plot lists predictors in order of importance. The payer-specific payment amount deviations variable has the biggest overall effect on predictions, while the location type indicator has the smallest effect. Each point on the plot represents a SHAP value for an instance of the predictions. I calculate the SHAP values over the test set. Therefore, each predictor has one SHAP value point for each of the records in the test set. The relative value of each point is represented on a color scale, where blue captures lower values, and orange captures higher values. SHAP values greater than zero increase the predicted

[22] There is some disagreement over whether explainability and interpretability are synonymous. Some authoritative accounts treat the terms interchangeably (for example, Molnar, 2020), while others treat interpretability as broader than explainability (for example, Linardatos, Papastefanopoulos, and Kotsiantis, 2021)

probability in the direction of fraud. Values lower than zero decrease the predicted probability toward legitimacy.

Tying these elements together, the summary plot shows that higher payment amount deviations generally increase the predicted probability and thus push the prediction toward fraud. To take another example, recall that the late indicator is a binary variable that takes on a value of one when transactions occur between 10 p.m. and 5 a.m. and zero otherwise. Falling outside of the late window tends to put downward pressure on the predicted probability toward legitimacy.

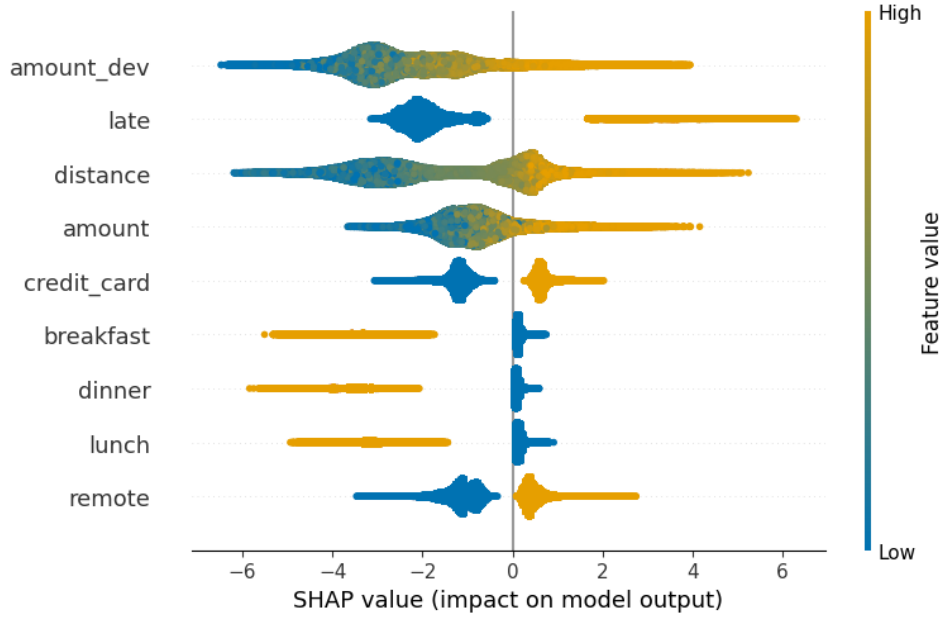

Figure 17. SHAP summary plot. Figure captures SHAP values for each predictor for every record in the test data. Features are listed in order of global importance on the left-hand y-axis. The feature value scale is on the right-hand side of the plot. Blue represents lower values of a feature. Orange represents higher values of a feature. The x-axis is a SHAP value scale that ranges from about -6 to 6 in log-odds. Payment amount deviation is the most important predictor according to the SHAP summary. Higher payment amount deviations are associated with higher SHAP values—that is, they increase the predicted probability in the direction of fraud. Lower payment amount deviations tend to decrease the predicted probability in the direction of legitimacy. The location type indicator is the least important by the SHAP metrics.

SHAP also enables local interpretation for individual predictions through force plots. Figure 18 captures two force plots. The top panel is a force plot for a low predicted probability (legitimate), and the bottom panel represents a high predicted probability (fraud). Directionality is captured by the arrows, where arrows pointing left force the prediction down, and arrows pointing right force the prediction up. The size of the arrows represent magnitude. Recall that payment amounts and deviations are in log terms. In the top panel, the below average payment amount

deviation, close distance, and breakfast indicator are important in forcing the prediction downward. In the bottom panel, the far distance, late indicator, and high payment amount deviation are the most important variables forcing the prediction upward toward fraud.

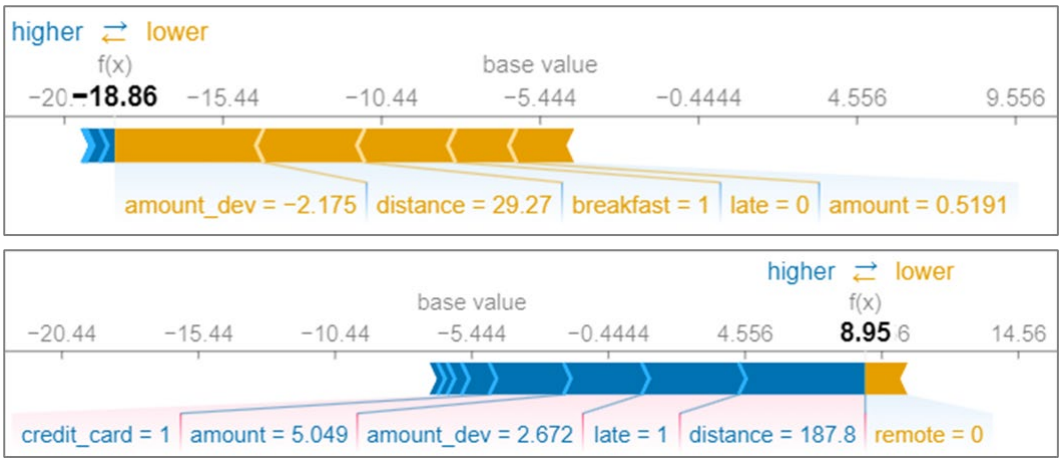

Figure 18. SHAP force plots. Figure captures two force plots, which enable local interpretability. The top panel is a force plot for a test set record that has a low predicted probability and is classified as legitimate. The bottom panel captures a fraud prediction. Left pointing (orange) arrows force the prediction downward, while right pointing (blue) arrows force the prediction upward. The size of the arrows corresponds to the magnitude of predictor importance. In both cases, payment amount deviations, distance, and time-of-day flags play an important role in driving the prediction.

Interpreting the SHAP base value is challenging with XGBoost. As discussed above, the base value should approximate the expected value in the absence of any feature variables. With one percent fraudulent transactions in the training data, the expected value should be about .01, or, in log-odds: $\log\left(\frac{0.01}{0.99}\right) = -4.6$. As depicted in Figure 18, the base value for the XGBoost is -5.444. The discrepancy arises from how SHAP and XGBoost are operationalized in software. The SHAP base value is calculated directly from the fitted model structure. XGBoost weighs samples traveling down tree branches differently than many other tree-based ML approaches and does not internally store the count of samples that traversed each branch.[23] Thus, the base value calculated from a fitted XGBoost may differ from what is expected.[24] It is hard to say what impact this issue has on interpreting SHAP for XGBoost models, but practitioners should consider the nuance when leveraging this popular interpretability method for fraud detection.

[23] The discrepancy is described by Lundberg, one of the creators of SHAP, on a GitHub issues board here: https://github.com/shap/shap/issues/29#issuecomment-388121951.

[24] As a comparison, I fit another tree-based ML model, the random forest, to the training data, and the SHAP base value, in probability terms, is about 0.01, in line with my expectation.

6 Conclusion

High levels of fraud in payment systems and concerns about the potential for Generative AI to augment malicious actors' fraud capabilities point to a need for more diverse and creative fraud detection research. But researchers' ability to test fraud detection models and disseminate findings are constrained by the lack of publicly available payment transaction data. Due to their sensitivity and economic value, it is unlikely that payment transaction data will become widely available for testing and research. In recent years, simulation methodologies have helped narrow the payment transaction data gap without comprising important privacy and security imperatives.

This paper seeks to advance fraud detection research by introducing a new simulation methodology and corresponding software package, $CardSim$. The methodology embeds probabilistic relationships between payment transaction features and fraud propensity using publicly available data and Bayes' theorem. It is also highly modular, and its parameters can be calibrated to changing payment behaviors and fraud trends. As demonstrated in this paper, the simulator can be used to test and evaluate important types of ML workflows, models, and interpretability approaches that are relevant for fraud detection.

As in all simulators, $CardSim$ makes a range of necessary simplifications. These simplifications and several of the limitations identified in section 4.3 point to areas for future research and development. From a methodological standpoint, future payment card simulation methodologies could bring together the types of fraud typologies embedded in Le Borgne and others' (2022) simulator and the probability-based calibration approach underlying $CardSim$. Additionally, as in most other payment fraud simulation approaches, $CardSim$ assumes that a fraud label can be generated, but these labels are not available to some practitioners. Thus, another relevant extension would be to build a simulation methodology geared toward unsupervised anomaly detection approaches. This could involve simulating a richer set of payment features that might be available in a modern payment message, while forgoing the fraud label. Finally, one of the most obvious limitations of $CardSim$ is that it only focuses on the payment card system. Future simulation methodologies could focus on enabling fraud detection research in other important electronic payment systems, such as ACH, wire, and instant payments.

References

Altman, Erik, Jovan Blanuša, Luc Von Niederhäusern, Béni Egressy, Andreea Anghel, and Kubilay Atasu. (2024). "Realistic synthetic financial transactions for anti-money laundering models." Advances in Neural Information Processing Systems, 36, https://proceedings.neurips.cc/paper_files/paper/2023/hash/5f38404edff6f3f642d6fa5892479c42-Abstract-Datasets_and_Benchmarks.html.

Australian Payments Network (2023). "Payment fraud statistics: Jul 22 - Jun 23." Sydney: Australian Payments Network, https://www.auspaynet.com.au/resources/fraud-statistics/July-2022-June-2023#summary-table.

Bank for International Settlements (BIS) (2023). "Project Aurora: The Power of Data, Technology, and Collaboration to Combat Money Laundering Across Institutions and Borders." Basel: Bank for International Settlements, https://www.bis.org/publ/othp66.pdf.

Birkin, Mark, and Belinda Wu (2011). "A review of microsimulation and hybrid agent-based approaches." In: Heppenstall, A., Crooks, A., See, L., Batty, M. (eds) Agent-Based Models of Geographical Systems. Springer, Dordrecht, 5168, https://doi.org/10.1007/978-90-481-8927-4_3.

Board of Governors of the Federal Reserve System (2024). "The Federal Reserve Payments Study: Cards and Alternative Payments, 2021 and 2022." Washington: Board of Governors, https://www.federalreserve.gov/paymentsystems/fr-payments-study.htm.

Board of Governors of the Federal Reserve System, Bureau of Consumer Financial Protection, Federal Deposit Insurance Corporation, National Credit Union Administration, and Office of the Comptroller of the Currency (2021). "Request for Information and Comment on Financial Institutions' Use of Artificial Intelligence, Including Machine Learning." *Federal Register*, vol. 86, no. 60 (March 31), pp. 16837-16842, https://www.govinfo.gov/content/pkg/FR-2021-03-31/pdf/2021-06607.pdf.

Dal Pozzolo, Andrea (2016). "Credit card fraud detection dataset." Kaggle, https://www.kaggle.com/mlg-ulb/creditcardfraud.

Davis, Jesse, and Mark Goadrich (2006). "The relationship between Precision-Recall and ROC curves." In Proceedings of the 23rd international conference on Machine learning, pp. 233-240., https://doi.org/10.1145/1143844.1143874.

European Banking Authority (EBA) and European Central Bank (ECB) (2024). "2024 report on payment fraud." Frankfurt: European Central Bank, https://www.ecb.europa.eu/press/intro/publications/pdf/ecb.ebaecb202408.en.pdf.

Federal Reserve (2023). FraudClassifier^{SM} Model. The Federal Reserve: FedPayments Improvement. Accessed December 20, 2023, https://fedpaymentsimprovement.org/strategic-initiatives/payments-security/fraudclassifier-model/.

Federal Reserve Financial Services (FRFS) (2024). "Key Findings from the 2023 Federal Reserve Financial Services Survey of Risk Officers." Federal Reserve Financial Services, https://www.frbservices.org/binaries/content/assets/crsocms/news/research/2023-risk-officer-survey.pdf.

Federal Trade Commission (FTC) (2020). "Consumer Sentinel Network Data Book 2019." Washington: Federal Trade Commission, https://www.ftc.gov/reports/consumer-sentinel-network-data-book-2019.

Federal Trade Commission (FTC) (2024). "FTC Consumer Sentinel Network: All Fraud Reports by Payment Method." Washington: Federal Trade Commission, https://public.tableau.com/app/profile/federal.trade.commission/viz/FraudReports/FraudFacts.

Financial Crimes Enforcement Network (FinCEN) (2024). "SAR Stats." Washington: FinCEN, https://www.fincen.gov/reports/sar-stats (accessed September 9, 2024).

Financial Services Sector Coordinating Council (2024). "Artificial Intelligence in the Financial Sector: Cybersecurity and Fraud Use Cases and Risks," in U.S. Department of the Treasury (2024). "Managing artificial intelligence-specific cybersecurity risks in the financial services sector." Washington: Department of the Treasury, Annex A., https://home.treasury.gov/system/files/136/Managing-Artificial-Intelligence-Specific-Cybersecurity-Risks-In-The-Financial-Services-Sector.pdf.

Financial Stability Board (FSB) (2017). "Artificial intelligence and machine learning in financial services: Market developments and financial stability implications." Basel: Financial Stability Board, https://www.fsb.org/wp-content/uploads/P011117.pdf.

Financial Stability Board (FSB) (2024). "The financial stability implications of artificial intelligence." Basel: Financial Stability Board, https://www.fsb.org/uploads/P14112024.pdf.

Financial Stability Oversight Council (FSOC) (2023). Annual Report, 2023. Washington: Department of the Treasury, https://home.treasury.gov/system/files/261/FSOC2023AnnualReport.pdf.

Foster, Kevin, Claire Greene, and Joanna Stavins (2024). "2023 Survey and Diary of Consumer Payment Choice." Federal Reserve Bank of Atlanta, https://doi.org/10.29338/rdr2024-01.

Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald E. Rubin (2014). *Bayesian data analysis, third edition*. CRC Press, Taylor & Francis Group: Boca Raton, FL.

Gerdes, Geoffrey R., Claire Greene, and May X. Liu (2018). "Changes in U.S. Payments Fraud from 2012 to 2016: Evidence from the Federal Reserve Payments Study." Washington: Board of Governors of the Federal Reserve System, https://www.federalreserve.gov/publications/files/changes-in-us-payments-fraud-from-2012-to-2016-20181016.pdf.

Grover, Prince, Julia Xu, Justin Tittelfitz, Anqi Cheng, Zheng Li, Jakub Zablocki, Jianbo Liu, and Hao Zhou (2022). "Fraud Dataset Benchmark and Applications." arXiv preprint arXiv:2208.14417, https://doi.org/10.48550/arXiv.2208.14417.

Harris, Charles R., K. Jarrod Millman, Stéfan J. Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser and others (2020). "Array programming with NumPy." Nature, 585(7825): 357-362., https://doi.org/10.1038/s41586-020-2649-2.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The elements of statistical learning, second edition*. Springer: New York, NY, https://doi.org/10.1007/978-0-387-84858-7.

Howard, Addison, Bernadette Bouchon-Meunier, IEEE CIS, inversion, John Lei, Lynn@Vesta, Marcus2010, Hussein Abbass (2019). "IEEE-CIS Fraud Detection." Kaggle, https://www.kaggle.com/c/ieee-fraud-detection/overview.

Jeni, László A., Jeffrey F. Cohn, and Fernando De La Torr (2013). "Facing imbalanced data--recommendations for the use of performance metrics." In 2013 Humaine association conference on affective computing and intelligent interaction, pp. 245-251., https://doi.org/10.1109/ACII.2013.47.

Le Borgne, Yann-Aël, Wissam Siblini, Bertrand Lebichot, and Gianluca Bontempi (2022). *Reproducible Machine Learning for Credit Card Fraud Detection-Practical Handbook*. Brussels: Université libre de Bruxelles, https://fraud-detection-handbook.github.io/fraud-detection-handbook/Foreword.html.

Linardatos, Pantelis, Vasilis Papastefanopoulos, and Sotiris Kotsiantis (2021). "Explainable ai: A review of machine learning interpretability methods." Entropy, 23(1): 18, https://doi.org/10.3390/e23010018.

Lopez-Rojas, Edgar, Ahmad Elmir, and Stefan Axelsson (2016). "PaySim: A financial mobile money simulator for fraud detection." In: The 28th European Modeling and Simulation Symposium-EMSS, Larnaca, Cyprus. 2016, https://www.msc-les.org/proceedings/emss/2016/EMSS2016_249.pdf.

Lundberg, Scott M., and Su-In Lee (2017). "A unified approach to interpreting model predictions." Advances in neural information processing systems 30, https://doi.org/10.48550/arXiv.1705.07874.

Mastercard (2024). "Mastercard accelerates card fraud detection with generative AI technology." Purchase, NY: Mastercard, Press Releases, May 22, https://www.mastercard.com/news/press/2024/may/mastercard-accelerates-card-fraud-detection-with-generative-ai-technology/.

Molnar, Christoph (2020). *Interpretable Machine Learning*. Lulu.com, https://christophm.github.io/interpretable-ml-book/.

Molnar, Christoph, Giuseppe Casalicchio, and Bernd Bischl (2020). "Interpretable machine learning–a brief history, state-of-the-art and challenges." In: Koprinska, I., et al. ECML PKDD 2020 Workshops. ECML PKDD 2020. Communications in Computer and Information Science, vol 1323. Springer, Cham., https://doi.org/10.1007/978-3-030-65965-3_28.

OECD (2023). "Generative artificial intelligence in finance." OECD Artificial Intelligence Papers, No. 9, OECD Publishing, Paris, https://doi.org/10.1787/ac7149cc-en.

Oliphant, Travis E (2007). "Python for scientific computing." Computing in science & engineering, 9(3): 10-20., https://doi.org/10.1109/MCSE.2007.58.

Orcutt, Guy H (1957). "A new type of socio-economic system." The review of economics and statistics, 39(2): 116-123, https://doi.org/10.2307/1928528.

Rish, Irina (2001). "An empirical study of the naive Bayes classifier." In IJCAI 2001 workshop on empirical methods in artificial intelligence, 3(22): 41-46, https://www.dors.it/documentazione/testo/201911/10.1.1.330.2788.pdf.

Rousseeuw, Peter and Christophe Croux (1993). "Alternatives to the median absolute deviation." Journal of the American Statistical association, 88 (424): 1273-1283, https://doi.org/10.2307/2291267.

Saito, Takaya, and Marc Rehmsmeier (2015). "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets." PLoS ONE 10(3): e0118432, https://doi.org/10.1371/journal.pone.0118432.

Sofaer, Helen R., Jennifer A. Hoeting, and Catherine S. Jarnevich (2019). "The area under the precision-recall curve as a performance metric for rare binary events." Methods in Ecology and Evolution, 10(4): 565-577., https://doi.org/10.1111/2041-210X.13140.

Suzumura, Toyotaro and Hiroki Kanezashi (2021). "Anti-Money Laundering Datasets," https://github.com/IBM/AMLSim.

U.S. Census Bureau (2024a). "National Population by Characteristics: 2020-2023, Median Age and Age by Sex." Suitland, MD: U.S. Census Bureau, https://www.census.gov/data/tables/time-series/demo/popest/2020s-national-detail.html.

U.S. Census Bureau (2024b). "QuickFacts: United States." Suitland, MD: U.S. Census Bureau, https://www.census.gov/quickfacts/fact/table/US/PST045223 (accessed July 18, 2024).

U.S. Department of the Treasury (2024). "Managing artificial intelligence-specific cybersecurity risks in the financial services sector." Washington: Department of the Treasury, https://home.treasury.gov/system/files/136/Managing-Artificial-Intelligence-Specific-Cybersecurity-Risks-In-The-Financial-Services-Sector.pdf.

Visa (2024). "Visa's Growing Services Business Infused with New AI-Powered Products." San Francisco: Visa, Press Releases, March 27, https://usa.visa.com/about-visa/newsroom/press-releases.releaseId.20516.html.

Zhang, Harry (2004). "The optimality of naïve Bayes." Aa 1, no. 2, https://cdn.aaai.org/FLAIRS/2004/Flairs04-097.pdf.

Zhang, Harry and Jiang Su (2004). "Naive Bayesian Classifiers for Ranking." In: Boulicaut, JF., Esposito, F., Giannotti, F., Pedreschi, D. (eds) Machine Learning: ECML 2004. ECML 2004. Lecture Notes in Computer Science, vol 3201. Springer, Berlin, Heidelberg., https://doi.org/10.1007/978-3-540-30115-8_46.