Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Система виявлення шахрайських дій під час онлайн-платежів на основі технологій машинного навчання.

Виконав(ла): студент(ка) 4 курсу, групи СН-42
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Рудик В.І. (прізвище та ініціали)

Керівник (підпис) Липак Г.І (прізвище та ініціали)

Нормоконтроль (підпис) Липак Г.І. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
"27" січня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

студенту Рудик Володимир Іванович
(прізвище, ім'я, по батькові)

1. Тема роботи **Система виявлення шахрайських дій під час онлайн-платежів на основі технологій машинного навчання.**

Керівник роботи **к.соціо-комунікаційних н., доц. Липак Г.І.**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від "*07*" *травня* 2025 року № *4/7-444*

2. Термін подання студентом завершеної роботи 23 червня 2025 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
ВСТУП; 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ; 1.1 Недоліки традиційних систем виявлення шахрайства; 1.2 Процес виявлення шахрайства за допомогою алгоритмів машинного навчання; 1.3 Нові підходи до виявлення шахрайства з використанням машинного навчання; 2 АНАЛІЗ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ; 2.1 Gradient Boosting Machines (GBM); 2.2 Основні типи GBM; 2.3 Переваги та недоліки градієнтного бустингу; 2.4 Виявлення аномалій за допомогою алгоритму Isolation Forest; 2.5 Застосування алгоритму Isolation Forest; 2.6 Переваги та обмеження Isolation Forest; 3 РЕАЛІЗАЦІЯ СИСТЕМИ; 3.1 Загальний опис процесу розробки; 3.2 Опис програмного коду та результати виконання; 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ; ВИСНОВКИ; СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ; ДОДАТКИ

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема дослідження. 2. Актуальність дослідження. 3. Мета роботи. 4. Обмеження традиційних підходів. 5. Переваги ML у виявленні шахрайства. 6. Використовувані алгоритми. 7. Gradient Boosting Machines. 8. Проблеми та обмеження GBM.9. Використовуваний набір даних. 10. Вибір алгоритму машинного навчання. 11. Змінні type та amount набору даних. 12. Змінні type та amount набору даних. 13. Результати роботи моделі. 14. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Окіпний І.Б., к.т.н., доцент кафедри МТ		

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	24.01.2025 – 27.01.2025	*Виконано*
2.	Підбір джерел по темі роботи	28.01.2025 – 01.04.2025	*Виконано*
3.	Оформлення першого розділу	15.04.2025	*Виконано*
4.	Оформлення другого розділу	20.04.2025	
5.	Оформлення третього розділу	30.04.2025	*Виконано*
6.	Виконання завдання до підрозділу "Безпека		
7.	життєдіяльності, основи охорони праці"	15.05.2025	*Виконано*
8.	Оформлення кваліфікаційної роботи	07.06.2025	*Виконано*
9.	Перевірка на плагіат	07.06.2025	*Виконано*
10.	Нормоконтроль	09.06.2025	*Виконано*
11.	Попередній захист кваліфікаційної роботи	11.06.2025	*Виконано*
12.	Захист кваліфікаційної роботи	27.06.2025	
13.			
14.			
15.			

Студент _____ Рудик В.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Липак Г.І
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

"Система виявлення шахрайських дій під час онлайн-платежів на основі технологій машинного навчання." // Кваліфікаційна робота освітнього рівня "Бакалавр" // Рудик Володимир Іванович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-42 // Тернопіль, 2025 // с. – 60, рис. – 17, таблиць – 1, джерел – 50, додатків – 1, сторінок додатків – 21.

Ключові слова: машинне навчання, виявлення шахрайства, кредитні картки, логістична регресія, ансамблеві методи, Python

В умовах стрімкої цифровізації фінансового сектору та експоненційного зростання обсягів безготівкових платіжних операцій, проблема шахрайства з кредитними картками набуває критичної значущості. Ця стаття присвячена аналізу феномену фінансового шахрайства, дослідженню обмежень традиційних систем його виявлення та обґрунтуванню доцільності застосування алгоритмів машинного навчання як ефективного інструменту для ідентифікації та превенції несанкціонованих транзакцій.

Сучасна фінансова екосистема характеризується домінуванням електронних платежів, що, поряд із незаперечними перевагами, створює сприятливі умови для реалізації протиправних дій. Шахрайство з платіжними картками, що включає такі методи, як фішинг, скімінг, соціальна інженерія та компрометація баз даних, є однією з найсерйозніших загроз для фінансових інституцій та їх клієнтів, призводячи до значних економічних збитків. Ключова складність у протидії даному явищу полягає у динамічній еволюції шахрайських патернів та високій латентності таких операцій.

ANNOTATION

"Fraud Detection System during Online Payments Based on Machine Learning Technologies" // Qualification work of the educational level "Bachelor" // Rudyk Volodymyr // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group CH-42 // Ternopil, 2025 // p. – 60, fig. – 17, tables – 1, references – 50, annexes – 1, pages for annexes – 21.

Keywords: machine learning, fraud detection, credit cards, logistic regression, ensemble methods, Python

In the context of the rapid digitalization of the financial sector and the exponential growth of non-cash payment transactions, the problem of credit card fraud is gaining critical importance. This article is devoted to the analysis of the phenomenon of financial fraud, the study of the limitations of traditional systems for its detection and the justification of the feasibility of using machine learning algorithms as an effective tool for identifying and preventing unauthorized transactions.

The modern financial ecosystem is characterized by the dominance of electronic payments, which, along with undeniable advantages, creates favorable conditions for the implementation of illegal actions. Payment card fraud, which includes methods such as phishing, skimming, social engineering and database compromise, is one of the most serious threats to financial institutions and their customers, leading to significant economic losses. The key challenge in countering this phenomenon lies in the dynamic evolution of fraud patterns and the high latency of such transactions.

ЗМІСТ

ВСТУП

Шахрайство – це зростаюча загроза. Традиційного виявлення недостатньо. Машинне навчання – це нова зброя, яка революціонізує виявлення шахрайства.

У сучасному швидкозмінному бізнес-середовищі виявлення шахрайства стало критично важливим питанням для організацій у різних галузях. Оскільки фінансові операції все частіше переходять на цифрові платформи, ризик шахрайської діяльності зростає. Чи то шахрайство з кредитними картками, крадіжка особистих даних чи відмивання грошей, підприємства стикаються зі значними фінансовими втратами та репутаційною шкодою, якщо вони не виявляють та не запобігають шахрайській поведінці.

Важливість виявлення шахрайства базується на таких чинниках.

- Фінансові втрати. Шахрайська діяльність може призвести до значних фінансових втрат для бізнесу. Крадіжка коштів, несанкціоновані транзакції та фальшиві страхові заяви – все це впливає на прибуток.
- Довіра клієнтів. Випадки шахрайства підривають довіру клієнтів. Коли клієнти стикаються з шахрайством, вони можуть втратити впевненість у здатності організації захистити їхню конфіденційну інформацію.
- Дотримання законодавчих та нормативних вимог. Дотримання правил боротьби з шахрайством є надзвичайно важливим. Невиявлення та неможливість запобігти шахрайству може призвести до юридичних санкцій та шкоди репутації організації.
- Операційна ефективність. Ефективні процеси виявлення шахрайства зменшують ручну роботу та оптимізують операції, дозволяючи підприємствам зосередитися на основній діяльності.

Маніпулювання даними машинного навчання відіграє ключову роль у розумінні складних закономірностей та прихованих кореляцій шахрайської діяльності. Використання алгоритмів машинного навчання у виявленні та запобіганні шахрайству сприяє формуванню точніших висновків про потенційні

загрози. Це також допомагає підприємствам проактивно реагувати на ці загрози, вивчаючи історичні моделі шахрайства.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Недоліки традиційних систем виявлення шахрайства

Традиційні системи виявлення шахрайства (Fraud Detection Systems, FDS) здебільшого функціонують на основі детермінованих моделей, що базуються на наборі заздалегідь визначених правил (rule-based systems). Наприклад, таким правилом може бути блокування транзакції, що здійснюється з географічно віддаленої локації, нетипової для клієнта. Однак, такий підхід має суттєві обмеження.

1. Низька адаптивність. Системи на основі правил є статичними і не здатні оперативно реагувати на нові, раніше невідомі вектори атак.

2. Високий рівень хибнопозитивних спрацьовувань (False Positives). Жорсткість правил часто призводить до блокування легітимних операцій, що спричиняє незручності для клієнтів та репутаційні ризики для банків.

3. Високий рівень хибнонегативних спрацьовувань (False Negatives). Витончені шахрайські схеми можуть обходити встановлені правила, залишаючись невиявленими.

На противагу традиційним підходам, методи машинного навчання пропонують більш гнучкий та ефективний інструментарій для аналізу транзакційних даних. Моделі машинного навчання, зокрема з парадигми керованого навчання (supervised learning), здатні обробляти багатовимірні простори ознак (сума, час, місцезнаходження, тип терміналу, частота операцій тощо) для кожної транзакції в режимі реального часу.

1.2 Процес виявлення шахрайства за допомогою алгоритмів машинного навчання

Процес виявлення шахрайства за допомогою машинного навчання включає наступні етапи.

1. Підготовка даних. Формування навчальної вибірки на основі історичних даних, що містять верифіковані легітимні та шахрайські транзакції.

2. Навчання моделі. Використання алгоритмів для побудови класифікаційної моделі. Найбільш поширеними у цій галузі є:

• Логістична регресія. Простий та інтерпретований лінійний класифікатор.

• Дерева рішень та їх ансамблі (Випадковий ліс, Градієнтний бустинг). Демонструють високу точність завдяки агрегації результатів множини слабких моделей.

• Нейронні мережі (зокрема, глибокі нейронні мережі). Здатні виявляти складні нелінійні залежності у даних, що робить їх ефективними для ідентифікації витончених шахрайських патернів.

3. Оцінка та валідація. Розрахунок метрик якості моделі (точність, повнота, F1-score, площа під ROC-кривою) та її валідація на тестовій вибірці.

4. Імплементація. Впровадження моделі в продуктивну систему, де вона аналізує вхідний потік транзакцій і присвоює кожній з них імовірнісну оцінку (scoring) приналежності до класу шахрайських.

Попри високу ефективність, застосування машинне навчання пов'язане з певними методологічними викликами. Ключовою проблемою є асиметрія класів у наборах даних, де частка шахрайських операцій є вкрай малою. Це вимагає застосування спеціалізованих технік, таких як SMOTE (Synthetic Minority Over-sampling Technique) або адаптація вагових коефіцієнтів у функції втрат. Іншим важливим аспектом є необхідність безперервного моніторингу та періодичного перенавчання моделі (concept drift adaptation) для збереження її ефективності в умовах постійної зміни тактик зловмисників.

Застосування алгоритмів машинного навчання є обґрунтованим та перспективним напрямком для підвищення ефективності систем виявлення шахрайства з кредитними картками. Здатність до самонавчання, адаптації та виявлення складних патернів дозволяє моделям машинного навчання значно перевершувати традиційні системи на основі правил. Подальші дослідження у

цій сфері мають бути спрямовані на розробку гібридних моделей, вдосконалення технік роботи з незбалансованими даними та створення інтерпретованих моделей (Explainable AI), що підвищить довіру до автоматизованих систем прийняття рішень у фінансовому секторі.

1.3 Нові підходи до виявлення шахрайства з використанням машинного навчання

Перш ніж проаналізувати нові алгоритми машинного навчання, коротко розглянемо традиційні, які зазвичай використовуються: логістична регресія, дерева рішень та випадкові ліси. Ці моделі машинного навчання мали значний вплив на процеси виявлення шахрайства, надаючи організаціям практичні можливості прогнозування.

Логістична регресія, наприклад, – це статистична модель, яка широко використовується завдяки своїй простоті та ефективності. Метод дерев рішень, з іншого боку, робить прогнози на основі набору правил прийняття рішень. І, нарешті, алгоритм випадкового лісу поєднує кілька дерев рішень для генерації кінцевого результату.

Хоча ці традиційні алгоритми мають свої переваги, вони часто не справляються зі складними сценаріями шахрайства. Складні моделі шахрайства включають нелінійні зв'язки, багатовимірні дані та тактики, що постійно розвиваються. Як результат, виникають певні проблеми в застосуванні цих алгоритмів.

- Проблеми розробки ознак. Традиційні алгоритми значною мірою залежать від вручну створених ознак. Отримання відповідних ознак із необроблених даних може бути складним завданням, особливо коли шахраї постійно адаптують свої методи.
- Незбалансовані дані. Шахрайські транзакції зазвичай трапляються рідко порівняно з легітимними. Незбалансовані набори даних можуть призвести до упереджених моделей, які надають пріоритет точності для більшості.

• Динамічна шахрайська поведінка. Шахраї постійно вдосконалюють свою тактику. Традиційним алгоритмам важко встигати за швидкозмінними закономірностями.

Нові підходи: алгоритми машинного навчання.

1. Gradient Boosting Machines (GBM) – це метод ансамбевого навчання, який поєднує кілька слабких навчальних елементів (зазвичай дерева рішень) для створення сильної прогностичної моделі. Загальний алгоритм роботи цього алгоритму можна описати такими кроками.

1. Ініціалізація.

• GBM починається з початкового прогнозу (зазвичай середнього значення цільової змінної).

• Він обчислює залишки (різниці між фактичними та прогнозованими значеннями) для кожної точки даних.

2. Побудова дерев.

• GBM послідовно будує дерева рішень, кожне з яких має на меті виправити залишки з попереднього дерева.

• Дерева додаються одне за одним, причому кожне дерево зосереджується на помилках, що залишилися.

3. Зважена агрегація.

• GBM призначає ваги кожному дереву на основі його ефективності у зменшенні залишків.

• Остаточний прогноз – це зважена сума прогнозів з усіх дерев.

Головною перевагою GBM над традиційними методами є його здатність мінімізувати помилки в послідовних ітераціях. Це означає, що алгоритм навчається на помилках попередніх моделей та вдосконалює їх, ефективно підвищуючи точність прогнозів. Ця перевага дозволяє GBM обробляти різноманітні та складні набори даних, що робить його особливо актуальним для складної природи виявлення шахрайства.

Одним із помітних прикладів використання GBM є його використання Capital One, відомою банківською холдинговою компанією. Capital One

використовувала алгоритми GBM для покращення своїх можливостей виявлення шахрайства. За допомогою GBM їм вдалося успішно виявити потенційні шахрайські транзакції та запобігти їх просочуванню через мережу. Це свідчить про корисність та ефективність GBM у виявленні шахрайства, слугуючи маяком для інших підприємств та організацій, які прагнуть покращити власні стратегії виявлення шахрайства.

2. Застосування нейронних мереж (глибокого навчання) у виявленні шахрайства.

Одним із найсучасніших інструментів в арсеналі машинного навчання є нейронні мережі, алгоритмічна структура, змодельована за зразком людського мозку. Їхнє глибоке застосування в різних галузях, включаючи виявлення шахрайства, призвело до появи нової хвилі рішень, здатних обробляти складні шаблони даних.

Нейронні мережі, по суті, є обчислювальними системами, що складаються з взаємопов'язаних штучних нейронів або вузлів. Ці вузли імітують нейрони в мозку. Кожен вузол приймає набір вхідних даних, обробляє їх і передає результат далі. Ця функціональність дозволяє нейронним мережам навчатися та інтерпретувати сенсорні дані за допомогою машинного сприйняття, маркування або кластеризації необроблених вхідних даних.

У сфері виявлення шахрайства нейронні мережі, особливо нейронні мережі глибокого навчання, мають значний потенціал. Їхня здатність навчатися та розпізнавати складні закономірності у великих наборах даних відрізняє їх від багатьох інших моделей машинного навчання. Нейронні мережі можуть навчитися виявляти закономірності, пов'язані з шахрайськими транзакціями, з мінімальним явним програмуванням, що робить їх чудовим інструментом для виявлення та запобігання шахрайству.

Нейронні мережі чудово працюють у різних сценаріях.

- Послідовності транзакцій. Рекурентні нейронні мережі (RNN) можуть аналізувати послідовності транзакцій з часом, фіксуючи часові залежності.

• Виявлення шахрайства на основі зображень. Згорткові нейронні мережі (CNN) обробляють зображення (наприклад, відскановані чеки, посвідчення особи) для виявлення аномалій.

• Ансамблеві підходи. Поєднання нейронних мереж з іншими моделями (наприклад, GBM) може підвищити загальну точність виявлення шахрайства.

3. XGBoost: Extreme Gradient Boosting – це ще один передовий алгоритм машинного навчання, який створює хвилі у сфері виявлення шахрайства. Цей алгоритм вирізняється своєю гнучкістю, високою точністю та ефективністю в роботі з великими та складними наборами даних.

Ось кілька визначних прикладів успішного впровадження XGBoost у виявленні шахрайства:

1. Виявлення шахрайства з мобільними платежами.

• Дослідники запропонували фреймворк на основі XGBoost для виявлення шахрайства з мобільними платежами.

• Інтегруючи алгоритми виявлення викидів без нагляду та класифікатор XGBoost, вони досягли чудових результатів на великому наборі даних, що містить понад 6 мільйонів мобільних транзакцій.

2. Виявлення шахрайства з кредитними картками.

• XGBoost застосовувався до даних про транзакції за кредитними картками.

• Вони використовували виявлення викидів на основі суми відстаней для ефективного виявлення шахрайських транзакцій.

3. Набір даних Державної мережевої корпорації Китаю (SGCC).

• Дослідники вивчали інтеграцію генетичних алгоритмів з XGBoost для виявлення шахрайства в наборі даних SGCC.

• Ця інтеграція показала багатообіцяючі результати та відкриває шляхи для подальших досліджень.

Підсумовуючи, XGBoost – це цінний інструмент у боротьбі з шахрайством. Його здатність обробляти великі та складні набори даних, у поєднанні з надзвичайною адаптивністю та точністю, робить його популярним

вибором для компаній, які шукають надійне та ефективне рішення для виявлення шахрайства. Успіх алгоритму в різних галузевих змаганнях та реальних застосуваннях лише підкреслює його потенціал у революціонізуванні виявлення шахрайства.

4. Isolation Forest або iForest – це унікальний та переконливий алгоритм машинного навчання, розроблений для виявлення аномалій. Він базується на принципі, що аномалії або точки даних, що виходять за межі норми, «нечисленні та різні», що робить їх легшими для «ізоляції», ніж звичайні випадки.

Ізоляційні ліси особливо ефективні для виявлення шахрайських транзакцій завдяки таким факторам.

1. Високовимірні дані.

• Виявлення шахрайства часто включає високовимірні дані (наприклад, характеристики транзакцій, поведінку користувачів).

• Ізоляційні ліси ефективно обробляють такі дані, що робить їх придатними для реальних сценаріїв.

2. Відмінні аномалії.

• Шахрайські транзакції демонструють відмінні від законних шаблони поведінки.

• Ізольовані ліси чудово ізолюють ці рідкісні та незвичайні види поведінки.

3. Масштабованість та швидкість.

• Лінійна часова складність алгоритму та низьке використання пам'яті роблять його добре придатним для обробки великих наборів даних.

• У системах виявлення шахрайства в режимі реального часу швидкість має значення, і ізоляційні ліси забезпечують високий рівень безпеки.

Конкретним прикладом його ефективності є банківська галузь. Великий банк інтегрував алгоритм Isolation Forest у свою систему виявлення шахрайства. Результатом стала система, яка могла швидко виявляти потенційно шахрайські транзакції, тим самим збільшуючи швидкість реагування та значно зменшуючи фінансовий ризик.

Підсумовуючи, алгоритм Isolation Forest надає потужний інструмент для виявлення аномалій, включаючи шахрайську діяльність. Його здатність ізолювати окремі випадки робить його цінним активом для захисту від фінансових втрат та підтримки цілісності даних.

2АНАЛІЗ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

2.1 Gradient Boosting Machines (GBM)

Gradient Boosting Machines (GBM) – це ансамбль моделей, які використовують градієнтне підсилення порівняно з іншими алгоритмами, такими як AdaBoost. Більшість фахівців з обробки даних використовують їх у машинному навчанні (ML), оскільки алгоритм градієнтного підсилення створює високоточні моделі, які перевершують багато популярних альтернатив.

Gradient Boosting Machines – це алгоритм підвищення рівня достовірності для завдань регресії та класифікації, який використовує градієнтний спуск для мінімізації помилок та створення точніших прогнозів. Цей алгоритм будує ансамбль, навчаючи базовий навчальний елемент (модель) послідовно для прогнозування залишкових помилок попередньої моделі (див. рис. 2.1).

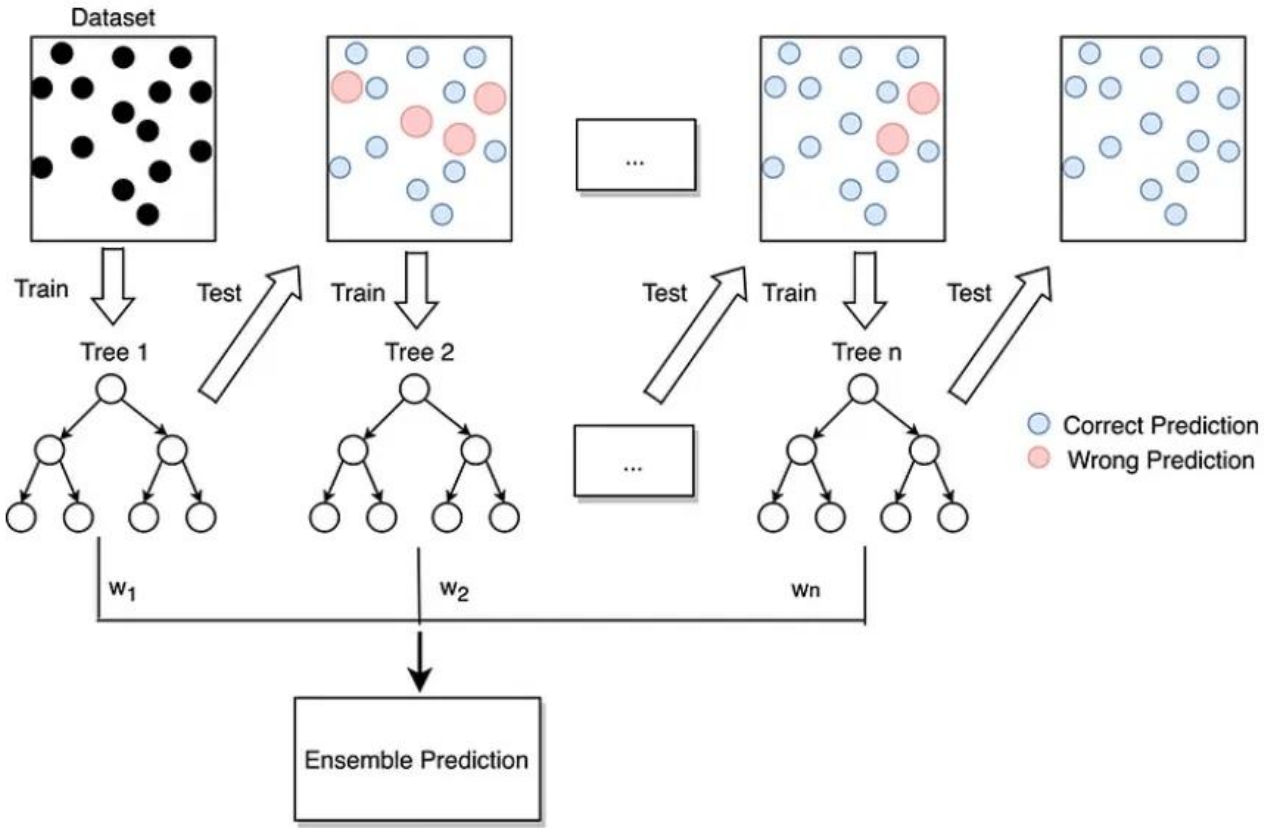


Рисунок 2.1 – Принцип роботи Gradient Boosting Machines

Gradient Boosting Machines мають три основні компоненти.

1. Функція втрат: вимірює різницю між прогнозованими та фактичними значеннями.

2. Базові (або «слабкі») учні: дерева рішень, побудовані послідовно, кожне з яких зосереджено на виправленні помилок, допущених попереднім деревом.

3. Адитивна модель: поєднує прогнози всіх базових учнів для отримання остаточного прогнозу.

Можна реалізувати GBM з ансамблем базових навчальних елементів, які можуть бути деревоподібними (дерева рішень) або недеревоподібними (лінійні моделі, нейронні мережі, методи опорних векторів (SVM)).

Ансамблі дерев є найпоширенішою реалізацією цієї методики. Дві відмінні характеристики градієнтного бустингу на основі дерев відрізняють його від інших підходів до моделювання.

1. Розробка ансамблю дерев рішень, де кожне дерево послідовно навчається для прогнозування залишків попереднього дерева, щоб вони могли компенсувати помилки.

2. Базові дерева учнів, або « слабкі учні», – це моделі в ансамблі з прогнозами трохи кращими, ніж випадкова припущення. Дерево може бути класифікатором або регресором, залежно від завдання.

Розглянемо покроковий процес тренувань GBM:

1. Ініціалізація моделі: підбір базової моделі навчання (дерево рішень) до наданого набору даних (який включає ознаки X та мітки y), щоб зробити початковий прогноз. Зазвичай це постійне значення, таке як середнє значення цільової змінної для регресії або логарифмічна втрата для класифікації. Це служить базовою лінією, на якій будуть удосконалюватися наступні моделі.

2. Ітеративно додаємо слабких учнів: додаємо нового «слабкого учня» в кожній ітерації, зазвичай це неглибоке дерево рішень. Цей новий учень зосереджуватиметься на помилках або залишках, залишених попередніми учнями.

3. Обчислення залишків: для кожного екземпляра в навчальному наборі обчислюємо залишкову помилку, яка є різницею між фактичним значенням та прогнозом поточної моделі.

4. Підбір слабкого учня до залишків: навчаємо нового слабкого учня (дерево рішень), використовуючи той самий набір даних, для прогнозування цих залишків. Міткою (або ціллю), на якій навчатиметься цей новий учень, будуть помилки з попередньої моделі. По суті, цей учень намагається виправити помилки існуючої моделі, прогнозуючи помилки, допущені попередньою моделлю. Обчислюємо залишки для нового учня, порівнюючи його прогнози з фактичними значеннями наявних даних.

5. Оновлення моделі: поєднуємо існуючу модель за допомогою схеми «зваженого голосування» з цим новим учнем. Зазвичай це робиться шляхом додавання прогнозів від нового учня, масштабованих за швидкістю навчання (параметр «скорочення» або розміру кроку), до існуючої моделі. Слід зауважити, що низька швидкість навчання означає, що модель оновлюється повільно, що призводить до більш надійної моделі, але вимагає більшої кількості дерев. Тим часом, вища швидкість навчання надає більшої важливості останнім прогнозам, що може дозволити ансамблю швидко адаптуватися до змін у даних.

6. Оптимізація функції втрат: використовуємо функцію втрат (наприклад, середньоквадратичну помилку для регресії або логістичні втрати для класифікації) для вимірювання ефективності моделі у прогнозуванні попередніх помилок. Для керування навчанням наступного учня використовується метрика, мінімізуючи функцію втрат та оновлюючи залишки.

7. Регуляризація: щоб запобігти перенавчанню, використовуємо методи регуляризації, такі як обмеження кількості дерев (ітерацій), глибини дерева та випадковості (наприклад, шляхом підвибірки ознак або екземплярів).

8. Повторюємо кроки 2–8 до досягнення збіжності: продовжуємо додавати слабкі навчальні елементи, доки не буде виконано критерій зупинки. Це може бути максимальна кількість дерев або поріг, за якого додавання більшої кількості дерев несуттєво зменшує функцію втрат.

9. Остаточна модель: остаточна модель GBM є сумою початкової моделі та всіх слабких учнів, кожен з яких робить невеликий внесок у остаточний прогноз.

Наведемо псевдокод для машини градієнтного підсилення (GBM) (лістинг 2.1)

Лістинг 2.1 – Псевдокод для машини градієнтного підсилення (GBM)

```
# Крок 1: Побудувати дерево рішень на основі ознак та поточної цілі
дерево = DecisionTree () # Ініціалізація дерева рішень
tree.fit (X, target) # Підігнати дерево рішень до ознак та поточної цілі

# Крок 2: Зробити прогнози, використовуючи поточне дерево рішень
tree_predictions = tree.predict (X)

# Крок 3: Обчислити залишки (різницю між цільовим значенням та прогнозами)
залишки = ціль - дерево_прогнозів

# Крок 4: Оновити ціль для наступної ітерації, використовуючи залишки (градієнтний спуск)
target = residuals # Оновіть ціль, щоб вона була залишками

# Крок 5: Додати прогнози поточного дерева рішень до ансамблю
ensemble_ predictions.append (tree_predictions)

# Об'єднати прогнози з усіх дерев рішень, щоб отримати остаточний прогноз
кінцевий_прогноз = сума(ансамбль_прогнозів)

# Масштабувати остаточний прогноз або застосувати швидкість навчання
final_prediction = швидкість_навчання * final_prediction
```

Узагальнену схему алгоритму GBM показано на рисунку 2.2.

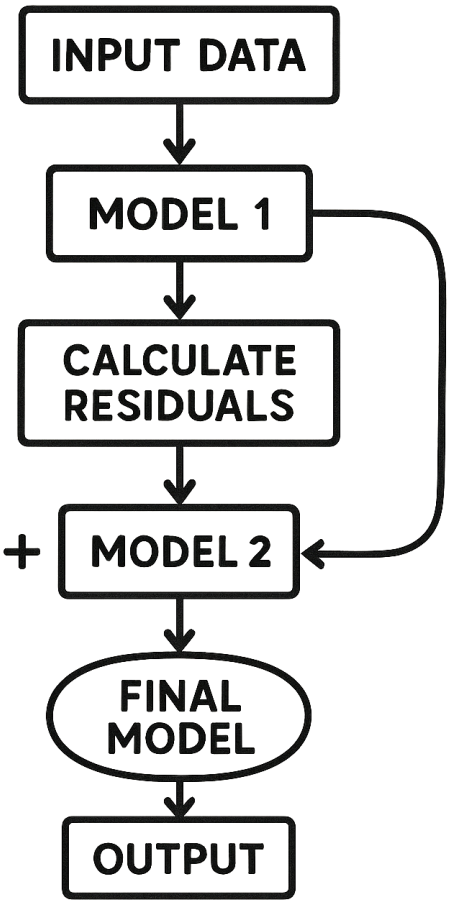


Рисунок 2.2 – Схема алгоритму GBM

Вибір реалізації градієнтного бустингу має вирішальне значення для оптимізації моделей машинного навчання. Різні типи GBM пропонують різну продуктивність, масштабованість та інтерпретованість. До міркувань належать використання пам'яті, обробка категоріальних ознак та стратегії навчання.

2.2 Основні типи GBM

Стандартний градієнтний бустинг – це базова форма градієнтного бустингу, де класифікатори або регресори дерев рішень, залежно від завдань, використовуються як базові навчальні елементи. Він послідовно додає дерева, кожне з яких виправляє залишки попередніх.

Стохастичний градієнтний бустинг – розширення стандартного градієнтного бустингу, яке включає випадковість у процес навчання. Він випадковим чином вибірково вибирає підмножину навчальних даних без заміни

перед вирощуванням кожного дерева (слабке навчання). Навчаючи кожне дерево на різній підмножині даних, SGB вводить випадковість у модель, що може допомогти зменшити дисперсію та запобігти перенавчанню.

XGBoost (Extreme Gradient Boosting). Це оптимізована реалізація градієнтного бустингу. Вона включає регуляризацію L1 (Lasso) та L2 (Ridge) для запобігання перенавчанню, обробці пропущених значень та обрізанню дерев. Крім того, вона добре обробляє великі набори даних та швидко виконує навчання. Це пояснюється тим, що вона оптимізує використання апаратних ресурсів під час виконання. Можна запускати XGBoost на багатопотоковій обробці на одній машині або на розподілених обчислювальних кластерах.

LightGBM. Тут використовується градієнтна одностороння вибірка (GOSS) та ексклюзивне об'єднання ознак (EFB) для оптимізації швидкості навчання та логічного висновку. Поєднання обох методів дозволяє легко розділити дані та досягти швидших обчислень, що дозволяє навчати моделі на великомасштабних наборах даних у кількох кластерах.

CatBoost. Найкраща реалізація для обробки категоріальних змінних. Вона використовує алгоритм під назвою «впорядковане підвищення», який сортує ознаки та застосовує градієнтне підвищення. CatBoost також обробляє відсутні значення та автоматично перетворює текстові ознаки на числові представлення, що робить його потужним інструментом для наборів даних з численними категоріальними ознаками.

Кожна реалізація GBM має свої сильні сторони та підходить для різних типів даних та наборів задач. Вибір між ними часто залежить від конкретних вимог, таких як розмір набору даних, типи ознак, обчислювальні ресурси та потреба в інтерпретованості моделі.

2.3 Переваги та недоліки градієнтного бустингу

Висока точність прогнозування. Градієнтне підвищення відоме своєю високою точністю прогнозування завдяки ефективній обробці категоріальних змінних, що робить його особливо ефективним у вирішенні складних та

нелінійних задач. Воно обробляє пропущені значення та автоматично перетворює текстові ознаки на числові представлення, досягаючи швидшої обчислювальної ефективності. Ці переваги сприяють його високій прогностичній ефективності.

Масштабованість. Алгоритми розробляють базові навчальні елементи послідовно, що забезпечує їхнє добре масштабування до великих наборів даних під час навчання та виконання логічного висновку.

Зменшення упередженості. Вони зменшують упередженість у прогнозах моделей завдяки своєму підходу до ансамбльного навчання, ітеративній корекції помилок, методам регуляризації та поєднанню слабких учнів.

Потребує менше попередньої обробки даних. GBM зазвичай не вимагають масштабування або нормалізації даних для їх вивчення під час навчання.

Стійкість до викидів та відсутніх даних. Використовується градієнтний спуск для виявлення та мінімізації впливу викидів, що може призвести до більш надійних прогнозів. GBM також обробляють відсутні значення як будь-яке інше значення, визначаючи, як розділити ознаку. Це робить його придатним для наборів даних, що містять відсутні значення або зашумлені, несумісні точки даних.

Вибір ознак та аналіз важливості. Алгоритм використовує регуляризацію для придушення ознак, які не сприяють точності прогнозування моделі. Або використовує підхід чистоти, такий як індекс Джині, для кількісної оцінки важливості групи ознак. Після того, як алгоритм побудує посилені дерева, можна отримати найважливіші оцінки для кожної ознаки в наборі даних.

Проблеми та обмеження GBM.

Схильний до перенавчання. Оскільки він неодноразово підганяє нові моделі до залишків попередніх моделей, що може призвести до надмірного виділення викидів або шуму в даних. Послідовність будує складніші моделі та збільшує ризик перенавчання, якщо не налаштувати параметри регуляризації.

«Чорна скринька», незрозумілі моделі. Прогнози моделей може бути важко інтерпретувати. Це пояснюється тим, що, хоча GBM і надають інформацію про

важливість ознак, на відміну від лінійних моделей, вони не мають коефіцієнтів або спрямованості. Вони показують лише те, наскільки важлива кожна ознака відносно інших. Це обмежує їх використання для застосувань у таких галузях, як банківська справа та охорона здоров'я.

Складність екстраполяції. Екстраполяція дозволяє моделі передбачати результати поза межами діапазону навчальних даних. Наприклад, лінійна регресія може зробити висновок з нижчих швидкостей, що автомобіль, що рухається зі швидкістю 60 км/год, проїжджає 120 км за 2 години, але градієнтна машина підвищення (GBM) потребує конкретних даних про цю 2-годинну подорож для точного прогнозування.

Вимоги до даних та обмеження. GBM зазвичай потребують достатньої кількості навчальних даних для вивчення складних закономірностей та ефективного виконання точних прогнозів.

Чутливість до гіперпараметрів. Продуктивність цього алгоритму може сильно залежати від обраних гіперпараметрів, а пошук оптимальних значень може зайняти багато часу та вимагати ретельних експериментів. Неправильний вибір гіперпараметрів може призвести до надмірного або недостатнього налаштування моделі, що впливає на її прогностичну здатність.

На завершення, Gradient Boosting Machines (GBM) – це потужні алгоритми машинного навчання, які довели свою високу ефективність у різних застосуваннях, включаючи розпізнавання зображень та сегментацію. Їхня здатність обробляти складні дані та генерувати точні прогнози робить їх безцінними в охороні здоров'я, автономних транспортних засобах та системах спостереження. Крім того, у поєднанні з обробкою природної мови (NLP), GBM може забезпечити ще більш повний аналіз, витягуючи інформацію з текстових та візуальних джерел даних.

2.4 Виявлення аномалій за допомогою алгоритму Isolation Forest

Виявлення аномалій є життєво важливим у всіх галузях, оскільки дозволяє виявити викиди в даних, які сигналізують про проблеми або унікальні висновки.

Ізольовані ліси пропонують потужне рішення, що ізолює аномалії від звичайних даних. У цьому розділі ми розглянемо реалізацію алгоритму Isolation Forest для виявлення аномалій за допомогою тестового набору даних, демонструючи його ефективність у виявленні викидів серед багатовимірних даних.

Аномалії, також відомі як викиди, – це точки даних, які суттєво відхиляються від очікуваної поведінки або норми в наборі даних. Їх вкрай важливо ідентифікувати, оскільки вони можуть сигналізувати про потенційні проблеми, шахрайську діяльність або цікаві відкриття. Виявлення аномалій відіграє життєво важливу роль у різних галузях, включаючи аналіз даних, машинне навчання та мережеву безпеку.

Існує три основні типи аномалій: точкові аномалії, контекстуальні аномалії та колективні аномалії.

1. Точкові аномалії (глобальні аномалії): це найпростіший тип, що представляє окремі точки даних, які є статистично незвичайними порівняно з рештою даних. Наприклад, транзакція кредитної картки з надзвичайно високою сумою може бути позначена як точка аномалія.

2. Контекстуальні аномалії (умовні аномалії): ці аномалії залежать від конкретного контексту або середовища, що їх оточує. Вони часто трапляються в даних часових рядів, де закономірності можуть змінюватися з часом. Прикладом є раптове підвищення температури взимку в метеорологічних даних.

3. Колективні аномалії: це групи пов'язаних точок даних, які разом демонструють аномальну поведінку, навіть якщо окремо вони можуть здаватися нормальними. Вони порушують загальний розподіл даних. Виявлення колективних аномалій часто вимагає складних алгоритмів на основі шаблонів, і вони зазвичай зустрічаються в динамічних середовищах, таких як дані мережевого трафіку.

Ізоляційний ліс – це алгоритм виявлення аномалій без учителя, особливо ефективний для високовимірних даних. Він працює за принципом, що аномалії є рідкісними та відмінними, що полегшує їх ізоляцію від решти даних. На відміну від інших методів, які профілюють нормальні дані, Ізоляційні ліси зосереджені

на ізоляції аномалій. В основі алгоритму Ізоляційного лісу лежить фундаментальна концепція, що аномалії значно відхиляються, що полегшує їх ідентифікацію.

Isolation Forest чудово виявляють аномалії завдяки унікальному підходу: ізоляції аномалій замість профілювання нормальних точок даних. Принцип роботи ізоляційних лісів визначено нижче:

1. Побудова дерев ізоляції. Алгоритм починається зі створення набору дерев ізоляції, зазвичай сотень або навіть тисяч. Ці дерева схожі на традиційні дерева рішень, але з ключовою відмінністю. вони не будуються для класифікації точок даних за певними категоріями. Натомість, дерева ізоляції спрямовані на ізоляцію окремих точок даних шляхом багаторазового розділення даних на основі випадково вибраних ознак та розділених значень.

2. Розділення за випадковими ознаками. Дерева ізоляції вводять випадковість у кожному вузлі дерева, вибирається випадкова ознака з набору даних. Потім вибирається випадкове значення розділення в діапазоні значень цієї конкретної ознаки. Ця випадковість допомагає гарантувати, що аномалії, які, як правило, відрізняються від більшості точок даних, не приховані в межах певних гілок дерева.

3. Ізоляція точок даних. Точки даних потім спрямовуються вниз по гілках дерева ізоляції на основі значень їхніх ознак.

• Якщо значення точки даних для вибраної ознаки падає нижче значення поділу, вона переходить до лівої гілки. В іншому випадку вона переходить до правої гілки.

• Цей процес продовжується рекурсивно, доки точка даних не досягне кінцевого вузла, який просто представляє ізольовану точку даних.

4. Оцінка аномалій: Ключова концепція ізоляційних лісів полягає в довжині шляху точки даних через дерево ізоляції.

• Аномалії, оскільки вони відрізняються від більшості, як правило, легше ізолювати. Вони потребують менше випадкових розщеплень, щоб досягти

кінцевого вузла, оскільки, ймовірно, виходять за межі типового діапазону значень для вибраних ознак.

• І навпаки, звичайні точки даних, які мають більше подібності одна з одною, можуть вимагати більше розщеплень на своєму шляху вниз по дереву, перш ніж їх буде ізольовано.

5. Розрахунок оцінки аномалії. Кожна точка даних оцінюється через усі дерева ізоляції в лісі.

• Для кожного дерева записується довжина шляху (кількість розщеплень), необхідна для виділення точки даних.

• Потім для кожної точки даних обчислюється оцінка аномалії шляхом усереднення довжин шляхів по всіх деревах ізоляції в лісі.

6. Виявлення аномалій. Точки даних з коротшими середніми довжинами шляху вважаються більш ймовірними аномаліями. Це пояснюється тим, що їх легше виділити, що свідчить про їх суттєві відхилення від основної маси даних. Встановлюється поріг для визначення оцінки аномалії, яка відділяє нормальні точки даних від аномалій. Цей поріг можна визначити на основі знань предметної області, експериментів або встановлених статистичних принципів.

Ключові висновки:

- Ізоляційні ліси використовують випадковість для ефективної ізоляції точок даних.
- Аномалії в середньому потребують меншої кількості розщеплень через свою особливу природу.
- Середня довжина шляху через усі дерева служить оцінкою аномалії.
- Нижчі бали вказують на вищу ймовірність наявності аномалії.

Розглянемо реалізацію алгоритму Isolation Forest для виявлення аномалій з використанням навчального набору даних про квіти ірисів зі scikit-learn. Ми збираємося виконати виявлення аномалій у транзакціях кредитних карток за допомогою алгоритму, виконавши такі кроки.

Крок 1: Імпорт бібліотек.

Лістинг 2.1 – Завантаження біліотек

```
# Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import IsolationForest
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
```

Крок 2: Завантаження та розділення набору даних.

Тепер ми завантажимо набір даних для виявлення аномалій кредитних карток та обмежимо кількість його рядків до 40000 для швидшої обробки. Потім ми стандартизуємо ознаки в наборі даних, виключаючи цільову змінну «Клас», за допомогою StandardScaler, гарантуючи, що кожна ознака має середнє значення 0 та стандартне відхилення 1. Далі він вибирає перші 40 000 рядків стандартизованих даних та перетворює їх у data frame. Нарешті, він відокремлює ознаки (X) від цільової змінної (y), де «X» містить усі стовпці, крім «Класу», а «y» містить лише стовпець «Клас», що вказує на статус шахрайства транзакції.

Лістинг 2.2 – Завантаження та попередня обробка набору даних

```
credit_data = pd.read_csv('creditcard.csv', nrows=40000) # https://www.kaggle.com/mlg-ulb/
creditcardfraud
scaler = StandardScaler().fit_transform(credit_data.loc[:,credit_data.columns!='Class'])
scaled_data = scaler[0:40000]
df = pd.DataFrame(data=scaled_data)
# Separate features and target variable
X = credit_data.drop(columns=['Class'])
y = credit_data['Class']
```

Крок 3: Підбір моделі.

Тепер час навчити нашу модель. По-перше, частка викидів у наборі даних визначається шляхом обчислення співвідношення шахрайських транзакцій ('Class' дорівнює 1) до нешахрайських транзакцій ('Class' дорівнює 0). Згодом створюється модель Ізоляційного лісу та підганяється до даних. Гіперпараметри для моделі Ізоляційного лісу визначаються наступним чином: --> 'n_estimators'

встановлюється на 100, що вказує на кількість базових оцінок в ансамблі, а 'contamination' присвоюється раніше розрахована частка викидів, що представляє очікувану частку викидів у наборі даних. Крім того, 'random_state' використовується для відтворюваності.

Лістинг 2.3 – Тренування моделі

```
# Determine the fraction of outliers
outlier_fraction = len(credit_data[credit_data['Class']==1])/
float(len(credit_data[credit_data['Class']==0]))
# Create and fit the Isolation Forest model
model =  IsolationForest(n_estimators=100, contamination=outlier_fraction, random_state=42)
model.fit(df)
```

Вивід лістингу 2.3:

```
IsolationForest(contamination=0.0026067776218167233, random_state=42)
```

Крок 4: Оцінка моделі.

Тепер ми оцінимо нашу модель на основі того, наскільки точно вона відокремлює викиди або потенційні аномалії, присутні в наборі даних. Отже, тут ми обчислимо оцінку аномалії з функції межі рішення моделі, а потім виведемо її точність.

Лістинг 2.4 – Оцінка точності моделі

```
# Predict outliers
scores_prediction = model.decision_function(df)
y_pred = model.predict(df)
y_pred[y_pred == 1] = 0
y_pred[y_pred == -1] = 1
# Print the accuracy in separating outliers or anomalies
print("Accuracy in finding anomaly:",accuracy_score(y,y_pred))
```

Вивід лістингу 2.4:

```
Accuracy in finding anomaly: 0.997175
```

Крок 4: Візуалізація.

Тепер побудуємо графік нормальних та аномальних випадків будь-якої ознаки набору даних. Тут ми побудуємо графік ознаки «Кількість» набору даних, але можна просто змінити назву ознаки, щоб візуалізувати її результати.

Лістинг 2.5 – Візуалізація результатів

```
# Selecting the feature for y-axis
y_feature = credit_data['Amount']    # change the feature name to visualize another

# Adding the predicted labels to the original dataset
credit_data['predicted_class'] = y_pred

# Plotting the graph
plt.figure(figsize=(7, 4))
sns.scatterplot(x=credit_data.index, y=y_feature, hue=credit_data['predicted_class'], palette={0:
'blue', 1: 'red'}, s=50)
plt.title('Visualization of Normal vs Anomalous Transactions')
plt.xlabel('Data points')
plt.ylabel(y_feature.name)
plt.legend(title='Predicted Class', loc='best')
plt.show()
```

Вивід блоку візуалізації з лістингу 2.5 показано на рисунку 2.3.

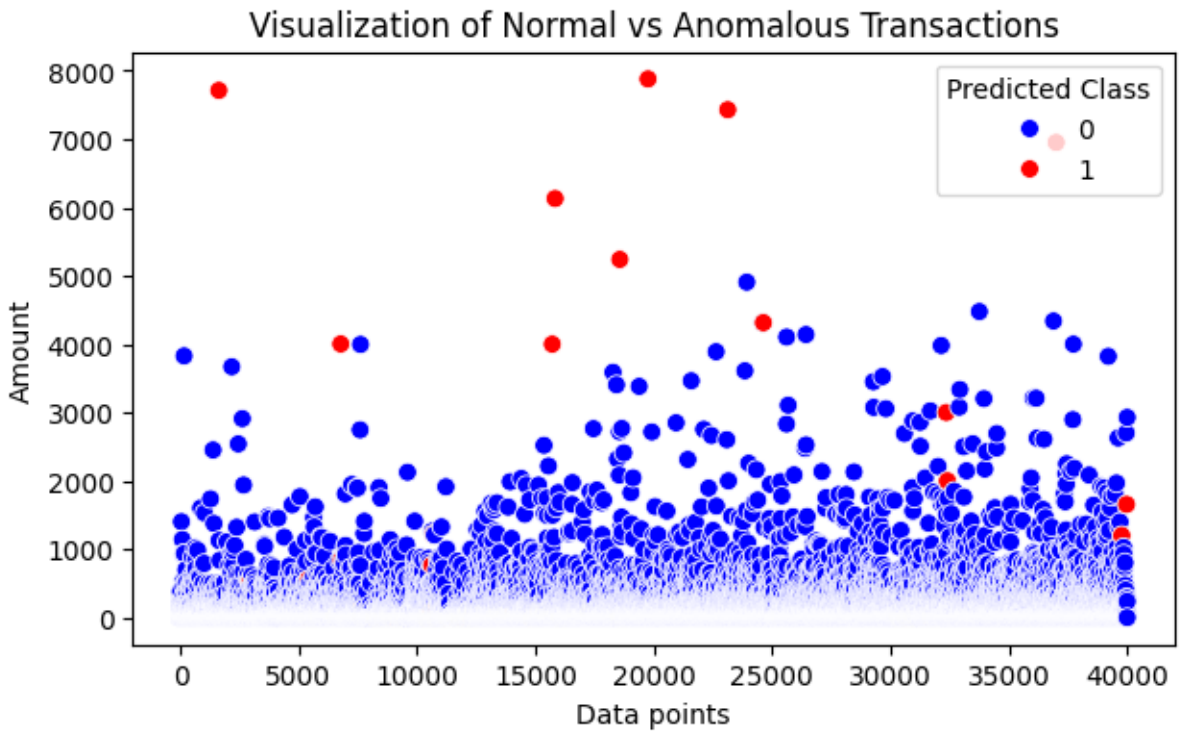


Рисунок 2.3 – Виявлення аномалій з використанням алгоритму Isolation Forest

З наведеного вище графіка чітко видно, що звичайні та аномальні випадки добре розділені з дуже незначним перекриттям.

2.5 Застосування алгоритму Isolation Forest

1. Кібербезпека.

Приклад: Виявлення шкідливого мережевого трафіку.

Сценарій. У корпоративній мережі Ізоляційний ліс може використовуватися для моніторингу вхідних та вихідних шаблонів трафіку.

Застосування: Якщо виникає аномалія, коли пристрій користувача раптово починає надсилати великий обсяг даних на зовнішній сервер у незвичний час або використовуючи незвичні протоколи, Ізоляційний ліс може виявити цю аномальну поведінку, оскільки він ізолює ці незвичайні шаблони від звичайного трафіку.

Перевага. Раннє виявлення таких аномалій може допомогти командам кібербезпеки зменшити потенційні загрози, такі як витік даних, поширення шкідливого програмного забезпечення або спроби несанкціонованого доступу.

2. Фінанси.

Приклад. Виявлення шахрайських транзакцій.

Сценарій. Фінансова установа щодня обробляє тисячі транзакцій з кредитними картками.

Застосування. Ізоляційний ліс може аналізувати дані транзакцій, щоб виявити аномалії, такі як транзакції, які суттєво відхиляються від типової поведінки користувача щодо витрат.

Перевага. Виділяючи транзакції, які потребують меншої кількості поділів (що вказує на їхню відмінність від звичайних транзакцій), Isolation Forest може виявляти потенційно шахрайські дії, такі як великі покупки в нетипових місцях або кілька транзакцій за короткий проміжок часу. Це допомагає фінансовим установам запобігати фінансовим втратам через шахрайство та захищає активи клієнтів.

3. Охорона здоров'я.

Приклад. Виявлення аномальних даних пацієнтів.

Сценарій. Лікарня керує електронними медичними картками (EHR) для своїх пацієнтів.

Застосування. Isolation Forest може аналізувати дані пацієнтів, щоб виявляти незвичайні закономірності, які можуть свідчити про медичне шахрайство або аномальні стани здоров'я.

Перевага. Наприклад, Isolation Forest може виявляти аномалії, такі як надзвичайно високі лабораторні результати, неочікувані комбінації діагнозів та методів лікування або аномальні життєво важливі показники. Це раннє виявлення може допомогти медичним працівникам розслідувати потенційні випадки страхового шахрайства, медичних помилок або нових проблем зі здоров'ям у пацієнтів, що призводить до своєчасного втручання та покращення догляду за пацієнтами.

2.6 Переваги та обмеження Isolation Forest

Ефективність та гнучкість: Isolation Forest демонструє надзвичайну стійкість, особливо у високовимірних наборах даних, завдяки своїй здатності видаляти аномалії шляхом випадкового поділу. На відміну від традиційних методів, таких як k-середні або ієрархічна кластеризація, йому не потрібно обчислювати відстань між точками даних. Ізоляційний ліс також залишається невеликим, що робить його високомасштабованим для завдань виявлення аномалій у реальному часі.

Толерантність до викидів: Однією з найпомітніших переваг Ізоляційного лісу є його толерантність до викидів. За своєю суттю, алгоритм чудово зменшує аномалії, виконуючи розділення, які розділяють повторювані точки даних. Це робить його особливо ефективним у випадках, коли аномалії невеликі або демонструють суттєві відмінності від норми. Крім того, оскільки сегментація лісу не спирається на методи на основі відстані, вона менш схильна до впливу викидів, забезпечуючи надійну продуктивність виявлення аномалій з різними наборами даних у різних галузях.

Легкість впровадження та інтерпретації: Ізоляція досить проста у впровадженні завдяки своїй простій конструкції та мінімальним накладним витратам. Простота алгоритму полегшує використання більшої кількості можливостей машинного навчання, що дозволяє швидко розгортати його в різних додатках. Крім того, бінарний характер розділення Ізоляційного лісу полегшує інтерпретацію, оскільки аномалії ідентифікуються на основі їх шляхів ізоляції в побудованих деревах. Така прозорість підвищує довіру до результатів виявлення та полегшує інтерпретацію після аналізу для прийняття рішень.

Обробка багатовимірних даних: Isolation Forest чудово справляється з обробкою багатовимірних даних, що створює проблеми для багатьох традиційних методів виявлення аномалій. Випадковим чином вибираючи ознаки для розділення, алгоритм ефективно пом'якшує недоліки розмірності, підтримуючи надійну продуктивність навіть у наборах даних з численними змінними. Це робить Isolation Forest добре придатним для таких застосувань, як обробка зображень, аналіз тексту та аналіз даних датчиків, де набори даних часто демонструють складні, багатовимірні структури.

Незважаючи на вагомі переваги, алгоритм ізоляційного лісу має свої потенційні обмеження, які обговорюються нижче.

Схильність до перенавчання. Хоча ізоляційний ліс часто стійкий до викидів, він може бути схильним до перенавчання, особливо при роботі з малими або сильно незбалансованими даними, за умови, що в різних випадках алгоритм може надмірно сегментувати дані, що призводить до надмірно неоднорідних дерев розділів, які не можуть добре узагальнювати на невидимі дані. Для зменшення цього ризику та забезпечення оптимальної продуктивності необхідні ретельне налаштування параметрів та процедури перехресної перевірки.

Обмежена чутливість до глобальних аномалій. Незважаючи на свою ефективність у виявленні локальних аномалій, ізоляційні ліси можуть мати труднощі з виявленням глобальних аномалій, які охоплюють кілька областей набору даних, оскільки алгоритм розділяє аномалії на основі їхніх індивідуальних характеристик, тому замість того, щоб розглядати глобальний

розподіл даних, потрібно використовувати альтернативні методи виявлення аномалій для виявлення шаблонів або розділення лісу з комбінацією методів попередньої обробки.

Вплив корельованих ознак. Ізоляційні ліси можуть погіршити продуктивність при роботі з наборами даних з дуже схожими ознаками. Розділення випадкових ознак у таких випадках може призвести до непотрібної сегментації, зменшуючи здатність алгоритму успішно ізолювати аномалії. Попередні кроки, такі як вибір ознак або зменшення розмірності, можуть допомогти вирішити цю проблему, покращуючи здатність алгоритму розрізняти ознаки шляхом зменшення надлишковості ознак.

Проблема з послідовними даними. Ізоляційні ліси за своєю суттю розроблено для наборів даних, які є незалежними та можуть зіткнутися з труднощами при застосуванні до порядкових або послідовних даних або даних часових рядів. Послідовні дані часто демонструють часові залежності та еволюціонуючі закономірності, що вимагають спеціалізованих підходів до виявлення аномалій. Хоча існують адаптації Ізоляційного лісу для послідовних даних, такі як розширення алгоритму для побудови дерев ізоляції вздовж часових послідовностей, ефективне усунення цього обмеження залишається актуальною областю досліджень у виявленні аномалій.

Алгоритм Isolation Forest пропонує ефективне рішення для виявлення аномалій, особливо в наборах даних з кількома вимірами. Він вирізняється тим, що ізолює викиди, а не профілює звичайні випадки, що робить його більш вправним у виявленні рідкісних випадків, які відрізняються від звичайної закономірності.

3 РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Загальний опис процесу розробки

Онлайн-платежі є найпопулярнішим методом транзакцій у світі сьогодні. Однак зі збільшенням онлайн-платежів також зростає шахрайство з платежами. Метою цього блокнота є навчання моделей машинного навчання для виявлення шахрайських та нешахрайських платежів. Набір даних зібрано з Kaggle, який містить історичну інформацію про шахрайські транзакції, яку можна використовувати для виявлення шахрайства в онлайн-платежах. Набір даних складається з 10 змінних:

- step: представляє одиницю часу, де 1 крок дорівнює 1 годині;
- type: тип онлайн-транзакції;
- amount: сума транзакції;
- nameOrig: клієнт, який починає транзакцію;
- oldbalanceOrg: баланс до транзакції;
- newbalanceOrig: баланс після транзакції;
- nameDest: одержувач транзакції;
- oldbalanceDest: початковий баланс одержувача до транзакції;
- newbalanceDest: новий баланс одержувача після транзакції;
- isFraud: шахрайська транзакція.

Бібліотеки Python, які використовувались в програмі: pandas, numpy, seaborn, matplotlib, tabulate, sklearn.

Через великий набір даних для виявлення шахрайства з онлайн-платежами використовувалися Random Forest та Naive Bayes.

Основні характеристики обраних для реалізації системи алгоритмів машинного навчання наведено в таблиці 3.1.

Таблиця 3.1 – Характеристики алгоритмів машинного навчання

	Accuracy	Precision	Recall	F1
Random Forest	0.999530	0.681524	0.93757	0.789301
Naive Bayes	0.567733	0.999190	0.00298	0.005942

Як бачимо з таблиці 3.1, найкращою моделлю є Random Forest для виявлення шахрайських та нешахрайських платежів.

3.2 Опис програмного коду та результати виконання

Спочатку в програмі імпортуються бібліотеки, необхідні для роботи програми.

Ось опис кожної бібліотеки одним реченням:

- numpy (np): ця бібліотека є фундаментальною для наукових обчислень у Python, надаючи потужні об'єкти масивів та інструменти для роботи з ними.
- pandas (pd): вона надає високоефективні, прості у використанні структури даних для аналізу даних та маніпуляцій з ними.
- matplotlib.pyplot (plt): це бібліотека для створення статичних, анімованих та інтерактивних візуалізацій у Python.
- seaborn (sns): базуючись на Matplotlib, ця бібліотека надає високорівневий інтерфейс для побудови привабливої статистичної графіки.
- sciру (sp): ця бібліотека пропонує безліч алгоритмів та інструментів для наукових та технічних обчислень, включаючи оптимізацію, обробку сигналів та статистику.
- tabulate: вона дозволяє легко форматувати табличні дані для друку, створюючи красиві та читабельні таблиці.
- random: цей вбудований модуль реалізує генератори псевдовипадкових чисел для різних розподілів.
- tensorflow (tf): це потужна відкрита бібліотека для машинного навчання та нейронних мереж, розроблена Google.

Програмний код наведено в додатках. Тут будемо відображати лише результати роботи програми та інтерпретувати їх.

Спочатку виконуємо аналіз набору даних з відомостями про банківські операції. Його ознаки (стопці) наведено у попередньому підпункті цього розділу. З використанням методів класу дата-фрейму мови Python отримаємо характеристики набору даних, показані на рисунках 3.1 та 3.2.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 10 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
dtypes: float64(5), int64(2), object(3)
memory usage: 485.4+ MB
```

Рисунок 3.1 – Відомості про набір даних

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
2889785	228	CASH_OUT	364310.96	C1882002610	127348.00	0.00	C1255784537	4814640.42	5178951.39	0
2805863	225	PAYMENT	20383.65	C149337475	403052.00	382668.35	M1589311961	0.00	0.00	0
1772525	162	CASH_IN	27891.95	C46285081	29584.00	57475.95	C376931188	640973.19	553796.55	0
6261030	608	PAYMENT	994.24	C1126305428	12549.64	11555.40	M595147503	0.00	0.00	0
3707836	277	PAYMENT	15309.73	C768133022	23391.50	8081.77	M1419231647	0.00	0.00	0

Рисунок 3.2 – Перші 5 рядків набору даних

Далі перевіряємо набір даних на предмет наявності порожних значень в кожному стовпцеві. Для цього користуємось функцією df.isnull().sum(). Результати показано на рисунку 3.3.

```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
dtype: int64
```

Рисунок 3.3 – Перевірка набору даних на порожні значення

Як видно з рисунка 3.3, набір даних не містить порожніх елементів. В іншому випадку довелось би локалізувати ці порожні значення та витирати відповідні рядки, щоби уникнути помилок класифікації операцій на основі пропущених даних.

Також в програмі виконано аналіз даних на предмет мінімальних та максимальних значень, виконано приведення типів даних до одного типу та здійснено перевірку, що набір даних не містить дублікатів.

Виконаємо тепер аналіз даних по окремих характеристиках. Для ознаки step, яка є фактично часовим ідентифікатором транзакції, порахуємо кількість унікальних значень та кількість кожного значення (див. рис. 3.4).

```
df['step'].value_counts()
```

Як бачимо, маємо 743 значення step, кожне з яких трапляється щонайменше двічі на весь набір даних.

```
19     51352
18     49579
187    49083
235    47491
307    46968
       ...
432        4
706        4
693        4
112        2
662        2
Name: step, Length: 743, dtype: int64
```

Рисунок 3.4 – Кількісний аналіз значень характеристики step

Покажемо тепер кількість різних типів транзакцій на основі значення характеристики type Набору даних. Для цього скористаємось функцією:

```
ax = sns.countplot(x='type', data=df, palette='PuBu')
```

З допомогою використанням стандартних функцій форматування стовпчикової діаграми отримаємо результуючу діаграму, показану на рисунку 3.5.

Як видно, зняття готівки, платежі, поповнення рахунку та грошові перекази є найбільш часто здійснюваними транзакціями.

З обсягами (сумами) транзакцій графік у вигляді розподілу сум транзакцій має правосторонню асиметрію. Це вказує на те, що більшість значень зосереджені навколо лівого хвоста розподілу, з довшим правим хвостом (див. рис. 3.6).

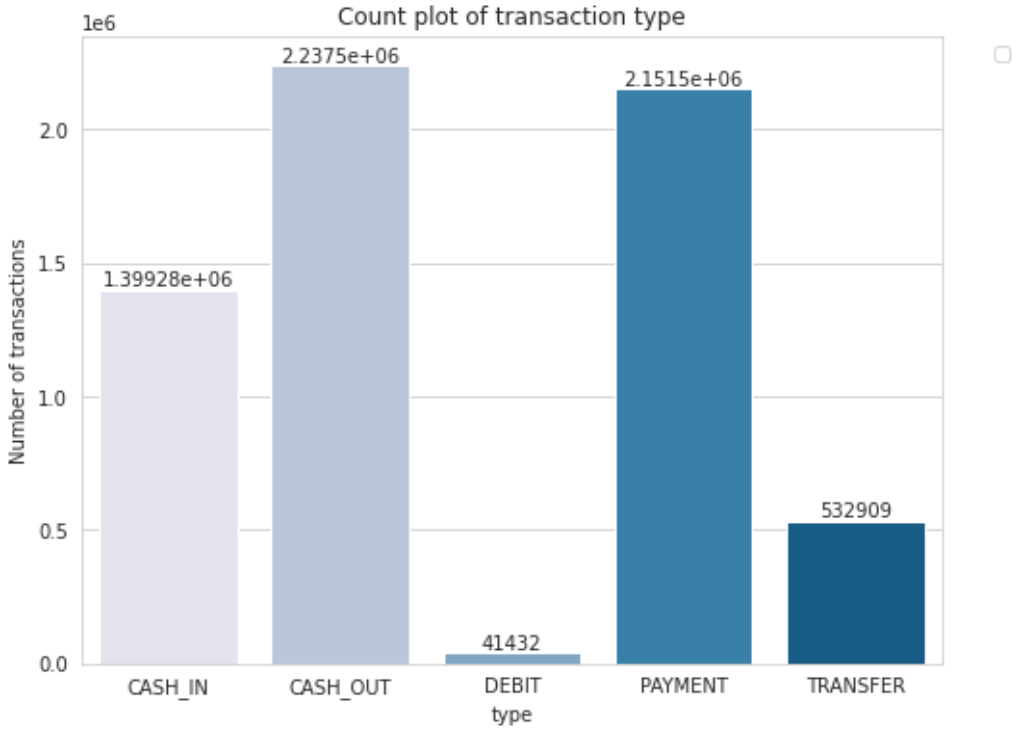


Рисунок 3.5 – Кількісне подання транзакцій різних типів

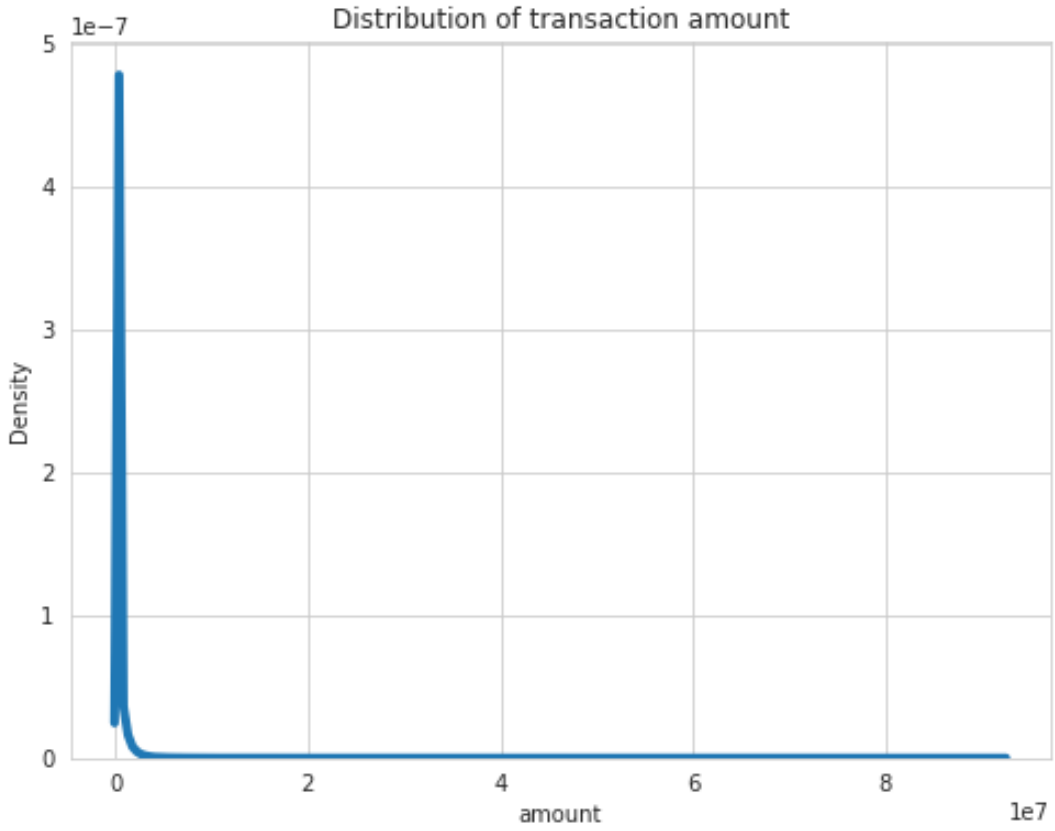


Рисунок 3.6 – Розподіл транзакцій за сумами

Аналогічний вигляд мають розподіли за сумою початкового балансу, сумою кінцевого балансу на боці джерела платежу та на стороні призначення платежу. Ц графіки наведені в додатках і в такому твердженні можна переконатися наочно.

Використовуючи частину нашого набору даних в якості навчальної вибірки, продемонструємо кількість операцій, котрі ідентифіковані, як шахрайські (див рисю 3.7):

```
ax = sns.countplot(x='isFraud', data=df, palette='PuBu')
```

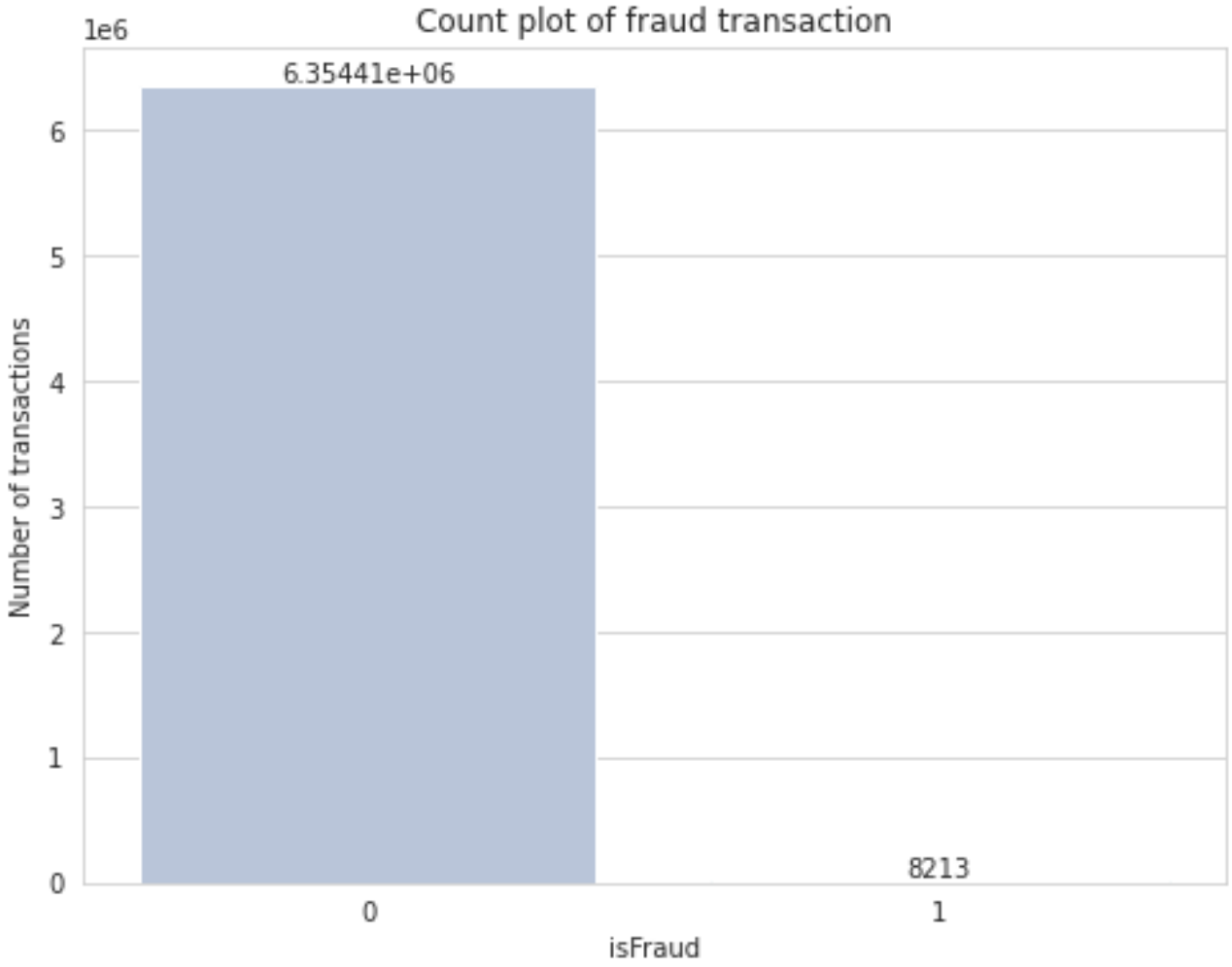


Рисунок 3.7 – Кількість "правильних" та шахрайських операцій на навчальній вибірці

Як видно з рисунка 3.7, нешахрайських транзакцій набагато більше, ніж шахрайських.

Проведемо двофактрний аналіз навчального набору даних. Для цього побудуємо діаграми з відображенням частки шахрайських операціях у всіх в перерізі типів операцій у абсолютних та відносних значеннях (рис. 3.8).

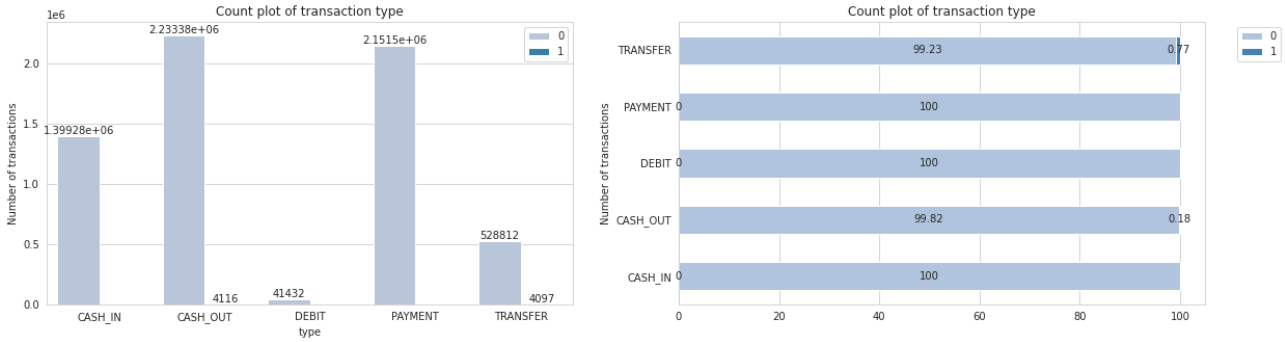


Рисунок 3.8 – Співвідношення шахрайських операцій до нешахрайських

Як бачимо шахрайські транзакції трапляються лише з дебетовими операціями та переказами. І їх кількість є дуже незначною. Детальніший аналіз циз характеристик набору даних наведено в додатках.

Якщо проаналізувати шахрайські транзакції в розрізі клієнтів, то бачимо, що їх кількість коливається від 1 до двох в різних користувачів (див. рис. 3.9).

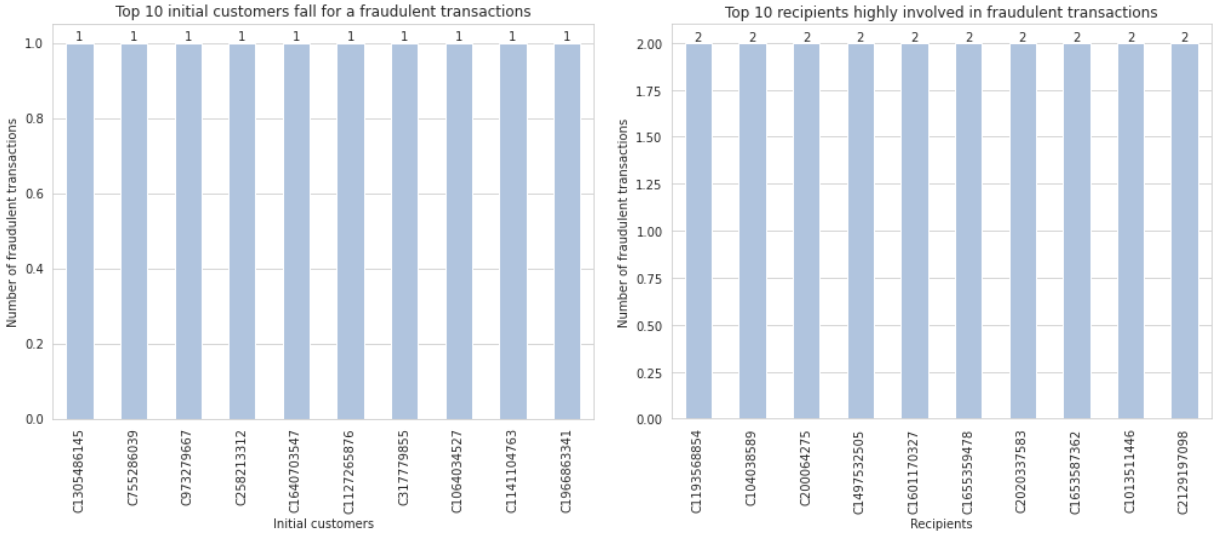


Рисунок 3.9 – Кількість шахрайських транзакцій в розрізі користувачів

Тепер виконаємо багатофакторний аналіз набору даних на предмет залежності (кореляції) різних факторів у шахрайських операціях. Результат показано на рисунку 3.10.

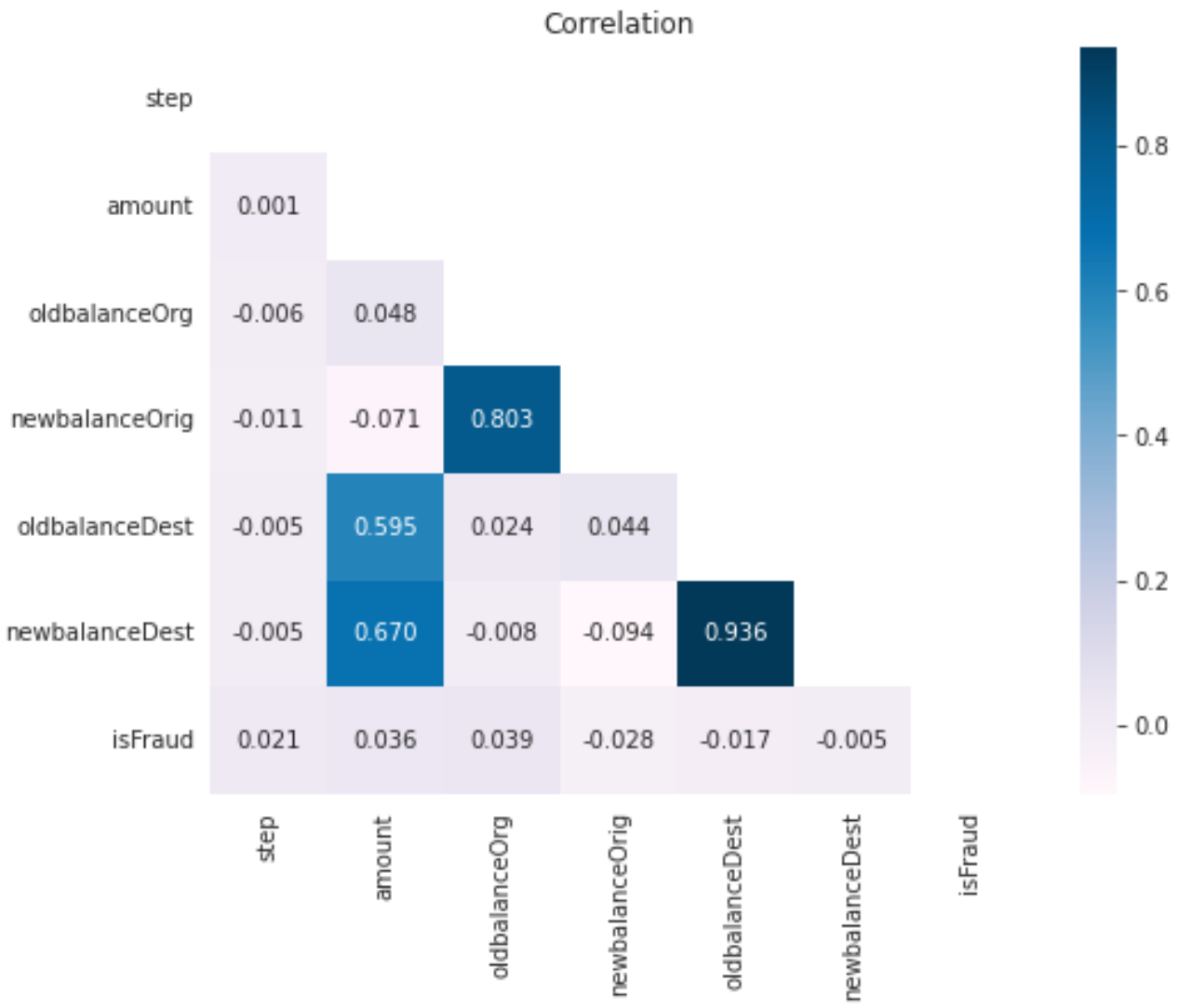


Рисунок 3.10 – Кореляційна матриця між факторами в наборі даних для шахрайських операцій

Таким чином, на основі рисунку 3.10 робимо висновок, що:

- між факторами oldbalanceOrg та newbalanceOrig є сильний позитивний зв'язок;
- між факторами oldbalanceDest та newbalanceDest є сильний позитивний зв'язок;
- між факторами oldbalanceOrg та amount є слабкий позитивний зв'язок;

• між факторами newbalanceOrig та amount є помірний позитивний зв'язок.

Тепер маємо всю необхідну інформацію для вибору моделі машинного навчання для виявлення шахрайських операцій на тестовому наборі даних. Через великий набір даних для виявлення шахрайства з онлайн-платежами використовуються випадковий ліс та логістична регресія зі збалансованою вагою класів. Детальний програмний код наведено в додатках. Конкретно в моделях логістичної регресії та випадкового лісу реалізовано класичний поділ набору даних на навчальний та тестовий, з використанням регуляризації для уникнення перенавчання моделі. Після виконання навчання моделі та її застосуванні до тестової вибірки отримаємо результати, показані на рисунку 3.11.

```
K-Fold Cross-Validation:

Random Forest Classifier:
Mean accuracy score: 0.985 (0.003)
Mean precision score: 0.975 (0.006)
Mean recall score: 0.996 (0.002)
Mean f1 score: 0.985 (0.003)
Mean roc_auc score: 0.998 (0.000)


Logistic Regression:
Mean accuracy score: 0.848 (0.007)
Mean precision score: 0.843 (0.008)
Mean recall score: 0.856 (0.005)
Mean f1 score: 0.849 (0.006)
Mean roc_auc score: 0.927 (0.004)
```

Рисунок 3.11 – Результати застосування моделей логістичної регресії (нижня частина рисунку) та випадкового лісу (верхня частина рисунку) для виявлення шахрайських операцій з онлайн-платежами

Як бачимо, модель навчання "Випадковий ліс" за результуючими показниками точності моделі переважає аналогічні параметри логістичної регресії. Тому виберемо саме випадковий ліс і проведемо глибше дослідження цієї моделі.

Використовуючи функції побудови матриці помилок (матриці невідповідностей) та побудови ROC-кривої (див. рис. 3.12), отримаємо результати, описані нижче.

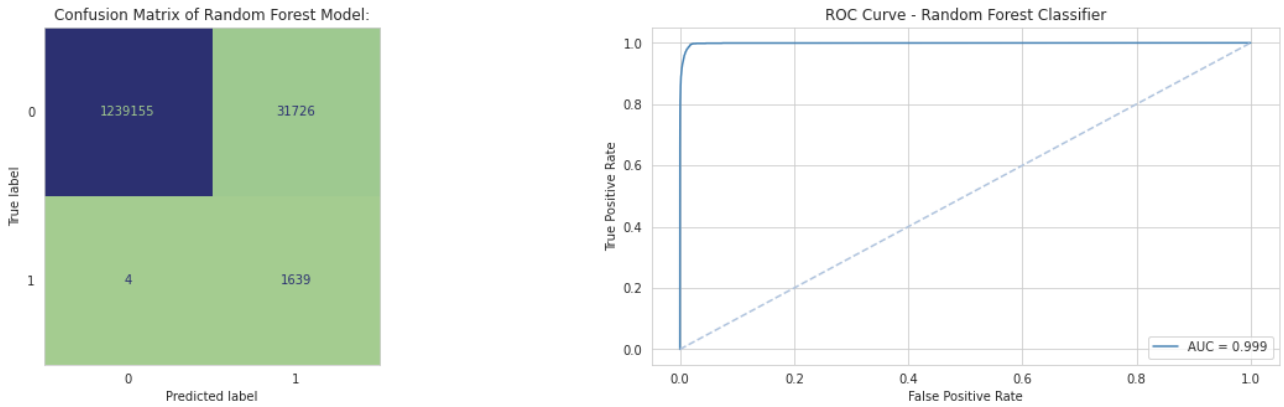


Рисунок 3.12 – Матриця невідповідностей та ROC-крива для моделі випадкового лісу

На основі цих відомостей бачимо, що:

1. Згідно з матрицею невідповідностей 1 239 155 платежів було правильно класифіковано як нешахрайські, а 31 726 осіб було неправильно класифіковано як нешахрайські.

2. Згідно з матрицею невідповідностей, 1 639 платежів було неправильно позначено як шахрайські, тоді як 4 платежі було правильно ідентифіковано як шахрайські.

В числових значеннях показники точності моделі показано на рисунку 3.13.

Random Forest Classifier:

	precision	recall	f1-score	support
Non-Fraud [0]	0.98	1.00	0.99	1239159
Fraud [1]	1.00	0.05	0.09	33365
accuracy			0.98	1272524
macro avg	0.99	0.52	0.54	1272524
weighted avg	0.98	0.98	0.96	1272524

Рисунок 3.13 – Показники точності моделі випадкового лісу для визначення шахрайських дій при онлайн-платежах

З рисунків 3.12 та 3.13 робимо висновок, що модель визначає 98% нешахрайських операцій та 100% шахрайських. Це також підтверджується показниками чутливості (recall), усередненого показника точності та чутливості (f1) та кількісних показників ідентифікації класів (support).

Алгоритм машинного навчання "Випадковий ліс" отримує найвищий бал серед усіх на основі використання перехресної перевірки К-середніх.

Найкращою моделлю є Випадковий ліс для виявлення шахрайських та нешахрайських платежів, оскільки AUC становить 0,999, що близько до 1. Це означає, що вона має хороший показник роздільності, і модель має 99,9% ймовірність розрізняти позитивні та негативні класи.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Поняття та об’єкт аналізу технічної безпеки

Безпеку визначають як стан діяльності людини, за яким з визначеною ймовірністю виключено прояв небезпек або ж відсутня надзвичай-на небезпека. Безпека праці – це стан умов праці людини, за яких відсутня дія небезпечних і шкідливих факторів.

Об’єктом аналізу безпеки праці є виробнича система "людина – машина – навколишнє середовище" (ЛМС), в якій в єдиній комплекс, створений для виконання певних функцій, поєднані технічні об’єкти, люди і навколишнє середовище, які взаємодіють між собою.

Основними компонентами виробничої системи є людина, машина, навколишнє середовище, взаємодія між якими має ґрунтуватись на дотри-манні відповідних правил, нормативних документів і бути керованою.

Система ЛМС є багаторівневою за ієрархією управління. Ієрархія поділяє людей на особу, яка формує завдання, організовує й управляє виробництвом, й особу, яка разом з технікою безпосередньо виконує це завдання. Таким чином, людина системи ЛМС більш високого рівня розглядає людину і техніку системи ЛМС більш низького рівня як єдиний компонент – своєрідну людину-машину, призначену для здійснення замислу.

Крім рівнів і компонентів в системі ЛМС доцільно виділити окремі стадії її життєвого циклу:

1. Стадія проектування (визначення завдань, формування вимог, розрахунок параметрів).
2. Стадія реалізації (коли у процесі виробництва перша стадія реалізується на практиці).
3. Стадія експлуатації (коли система ЛМС здійснює покладені на неї робочі функції).

Вірогідність нещасного випадку зростає, як тільки людина попадає в поле дії небезпечного або шкідливого фактору. Це небезпечні зони, що характеризуються певним видом небезпеки, її інтенсивністю, часом і простором дії.

Таким чином, з точки зору аналізу й управління небезпеками необхідно розглядати та аналізувати структурні елементи системи ЛМС – рівні (вищий і нижчий), компоненти і стадії життєвого циклу.

Взаємодія компонентів, що входять до системи ЛМС, може бути штатною і нештатною. Нештатна взаємодія може виявлятися у вигляді надзвичайної події – небажаних, незапланованих випадків, що порушують технологічний процес у відносно короткий відрізок часу. Відмова й інцидент, як правило, передують надзвичайній події, але можуть мати і самостійне значення. До головних моментів аналізу небезпек належить пошук відповідей на такі питання:

1. Які об’єкти є небезпечними.
2. Яким надзвичайним подіям можна запобігти.
3. Які надзвичайні події неможливо усунути і як часто вони мати-муть місце.
4. Яку шкоду не усунуті надзвичайні події можуть спричинити людям, об’єктам, навколишньому середовищу.

Пошук причин надзвичайних подій призводить до аналізу системи управління небезпеками (СУН) на виробництві. Ці системи обов’язково включають такі компоненти, як наявність інформації, зворотних зв’язків та алгоритми функціонування.

Наявність зворотних зв’язків й інформаційної системи дозволяє проводити збір даних щодо відхилень, відмов, проводити аналіз небезпек, порівнювати наслідки функціонування системи ЛМС з програмою управління небезпеками, приймати рішення. У виробничій системі ЛМС інформаційні функції виконують: рапорти інспекторів, акти розслідування нещасних випадків, аварій, протоколи атестації робочих місць тощо.

4.2 Розрахунок захисного заземлення

Захисне заземлення повинне забезпечити захист людей від ураження електричним струмом, при дотику до металевих частин, які можуть виявитися під напругою. Заземленням називається навмисне з'єднання електроустановок із заземлюючим пристроєм. Заземлювачем називається провідник, що перебуває в контакті із землею або її еквівалентом. Заземлюючим провідником називається провідник, що з'єднує заземлені частини із заземлювачем. Сукупність з'єднуючих провідників і заземлювачів називається заземлюючим пристроєм. Для установок потужністю не більше 100 кВт опір заземлюючого пристрою не повинне перевищувати 10 Ом, для установок потужністю більше 100 кВт – 4 Ом.

Розрахунок штучного заземлювального пристрою при відсутності природних заземлювачів.

Вихідні дані:

Захищуваний об’єкт – комп’ютерна мережа.

Захищуваний об’єкт – стаціонарний.

Напруга мережі – 220 В.

Виконання мережі – з глухозаземленою нейтраллю.

Тип заземлювального пристрою – вертикальні труби.

Розміри вертикальних заземлювачів:

Довжина – 6 м.

Діаметр труби – 0,60 м.

Товщина стінки труби – 0,06 м.

Висота труби – 0,6 м.

Відношення відстані між трубами до їхньої довжини:

$$\frac{L_B}{l_B} = 1 \tag{4.1}$$

Розмір горизонтального заземлювача (з'єднувальної стрічки): довжина $L_{Г} = L_{З.С.}$ $L_Г = L_{З.С.}$– згідно з розрахунком, м; ширина горизонтальної з'єднувальної стрічки b_c=0,04 м.

Глибина закладання вертикальних заземлювачів $h_В$=0,6.

Розміщення заземлювачів попередньо приймають за чотирикутним контуром при числі стержнів від 4 до 100 та в один ряд при числі стержнів від 2 до 20.

Грунт – супісок; склад – однорідний; вологість – мала; агресивність – нормальна.

Кліматична зона – II.

Розрахунок:

1. Визначаємо характеристику навколишнього середовища в приміщенні організації: за пожежною небезпекою згідно з ПУЕ воно відноситься до класу П-II; за вибухонебезпекою згідно з ПУЕ – до класу В-I; за ступенем ураження електричним струмом – без підвищеної та особливої небезпеки.

2. Визначаємо $R_Д$ – допустиме (нормативне) значення опору розтікання струму в заземлювальному пристрої, $R_Д \leq 4$ *Ом*.

3. Обраховуємо $K_{С.В.}$$К_{С.В.}$ – приблизне значення питомого опору ґрунту, що рекомендується для розрахунку – $\rho_{ТАБЛ} = 300$ *Ом · м* $\rho_{ТАБЛ} = 300$Ом*м.

4. Визначаємо $K_{С.В.}$ – коефіцієнт сезонності для вертикальних заземлювачів для денної кліматичної зони. За довідковою інформацією приймаємо $K_{С.В.}$=1,5.

5. Обраховуємо значення $К_{С.Г.}$ – коефіцієнт сезонності для горизонтального заземлювача згідно з кліматичною зоною. За довідковою інформацією приймаємо$К_{С.Г.} = 3,5$;$К_{С.Г.}$=3,5.

6. Визначаємо $\rho_{РОЗР.В.}$$\rho_{РОЗР.В.}$ – розрахунковий питомий опір ґрунту для вертикальних заземлювачів:

$$\rho_{РОЗР.В.} = \rho_{ТАБЛ} \cdot K_{С.В.} = 300 \cdot 1,5 = 450\ Ом \cdot м\,. \quad (4.2)$$

$$\rho_{РОЗР.В.} = \rho_{ТАБЛ} * К_{С.В.} = 300 * 1,5 = 450\ Ом \cdot м.$$

7. Розраховуємо $\rho_{РОЗР.Г}$ – розрахунковий питомий опір ґрунту для горизонтальних заземлювачів:

$$\rho_{РОЗР.Г} = \rho_{ТАБЛ.} * К_{С.Г.} = 300 * 3{,}5 = 1050\ Ом \cdot м.$$

$$\rho_{РОЗР.Г.} = \rho_{ТАБЛ} \cdot К_{С.Г.} = 300 \cdot 3{,}5 = 1050 Ом \cdot м. \quad (4.3)$$

8. Обраховуємо t – відстань від поверхні землі до середини вертикального заземлювача:

$$t = h_B + \frac{l_B}{2} = 0{,}6 + \frac{6}{2} = 3{,}6\ м. \quad (4.4)$$

9. Визначаємо R_B – опір, Ом, розтікання струму в одному вертикальному заземлювачі:

$$R_B = \frac{\rho_{РОЗ.В}}{2\pi l_B} \cdot \left(ln\frac{2l_B}{d} + 0{,}5 \cdot ln\frac{4t + l_B}{4t - l_B} \right) =$$
$$= \frac{450}{2\pi \cdot 6} \cdot \left(ln\frac{2 \cdot 6}{0{,}60} + 0{,}5 \cdot ln\frac{4 \cdot 3{,}6 + 6}{4 \cdot 3{,}6 - 6} \right) = 41{,}05\ Ом \quad (4.5)$$

10. Визначаємо $n_{т.в.}$ – теоретична кількість вертикальних заземлювачів без врахування коефіцієнта використання $\eta_{в.в.}$, тобто $\eta_{в.в.}$=1:

$$n_{Т.В.} = \frac{R_B}{R_Д \cdot \eta_{В.В}} = \frac{41{,}05}{4 \cdot 1} = 10{,}26\ шт. \quad (4.6)$$

11. Визначаємо $\eta_{в.в}$ – коефіцієнт використання вертикальних заземлювачів при розташуванні їх згідно з вихідними даними або за чотирикутним контуром при числі заземлення, $n_{т.в.}$=11 та при відношенні $\frac{L_B}{l_B} = 1$. За довідником приймаємо $\eta_{в.в}$ =0,42.

12. Визначаємо $n_{н.в}$ – необхідна кількість штук, вертикально однакових заземлювачів з врахування коефіцієнта використання:

$$n_{H.B} = \frac{R_B}{R_Д \cdot \eta_{B.B}} = \frac{41,05}{4 \cdot 0,42} = \frac{41,05}{1,68} = 24,43 \; шт. \qquad (4.7)$$

13. Визначаємо $R_{розр.в}$ – вертикальний опір, Ом, розтіканню струму у вертикальному заземленні при $n_{н.в} = 24,43$ без врахування з'єднувальної стрічки:

$$R_{РОЗР.В} = \frac{R_B}{n_{H.B} \cdot \eta_{B.B}} = \frac{41,05}{10,26} = 4 \; Ом. \qquad (4.8)$$

14. Визначаємо $L_в$ – відстань між вертикальним заземлювачами за відношенням $\frac{L_B}{l_B} = 1$, звідси

$$L_B = 1 \cdot l_B = 1 \cdot 6 = 6 \; м. \qquad (4.9)$$

15. Визначаємо $L_{з.с}$ – довжину, м, з'єднання стрічки горизонтального заземлювача:

$$L_{З.С} = 1,05 \cdot L_B (n_{H.B} - 1) = 1,05 \cdot 6 (24,43 - 1) = 147,60 \; м. \qquad (4.10)$$

16. Визначаємо $R_{г.з.с}$ – опір, Ом, розтікання струму в горизонтальному заземлювачі (з'єднувальній стрічці):

$$R_{Г.З.С} = \frac{\rho_{РОЗ.Г}}{2 \cdot \pi \cdot L_{З.С}} \cdot ln \frac{2 \cdot L_{З.С}^{\;2}}{2 \cdot b \cdot t} = \frac{1050}{2 \cdot 3,14 \cdot 147,61} \cdot ln \frac{2 \cdot (147,61)^2}{2 \cdot 0,04 \cdot 3,6} = 13,50 \; Ом. \qquad (4.11)$$

17. Визначаємо $\eta_{в.г}$ – коефіцієнт використання горизонтального заземлення при розташуванні вертикальних заземлювачів згідно з вихідними даними або за чотирикутним контуром при відношенні $\frac{L_B}{l_B}=1$ та необхідній кількості вертикальних заземлювачів $n_{н.в} = 24,43$.

За довжину приймаю $\eta_{в.г} = 0,19$.

18. Визначаємо $R_{розр.г}$ – розрахунковий опір, Ом, розтікання струму в горизонтальному заземленні (з'єднувальній стрічці) при числі електродів $n_г = 1$:

$$R_{РОЗР.Г} = \frac{R_{Г.З.С}}{n_Г \cdot \eta_{В.Г}} = \frac{13,5}{1 \cdot 0,19} = 71,05\ Ом\,. \qquad (4.12)$$

19. Визначаємо $R_{розр.в.г}$ – розрахунковий теоретичний опір, Ом, розтікання струму у вертикальному та горизонтальному заземленні:

$$R_{РОЗР.В.Г} = \frac{1}{\frac{1}{R_{РОЗ.В}}+\frac{1}{R_{РОЗ.Г}}} = \frac{1}{\frac{1}{4}+\frac{1}{71,05}} = 3,78\ Ом. \qquad (4.13)$$

20. Вибираємо матеріал та поперечний перетин з'єднувальних провідників. За довідковою інформацією вибираю голі мідні $S_м$ = 4 мм^2 провідники.

21. Вибираємо матеріал та поперечний перетин магістральної шини. За довідковою інформацією обираємо сталеву шину товщиною $\delta_с$ = 4мм і перетином не менше δ = 100 мм^2.

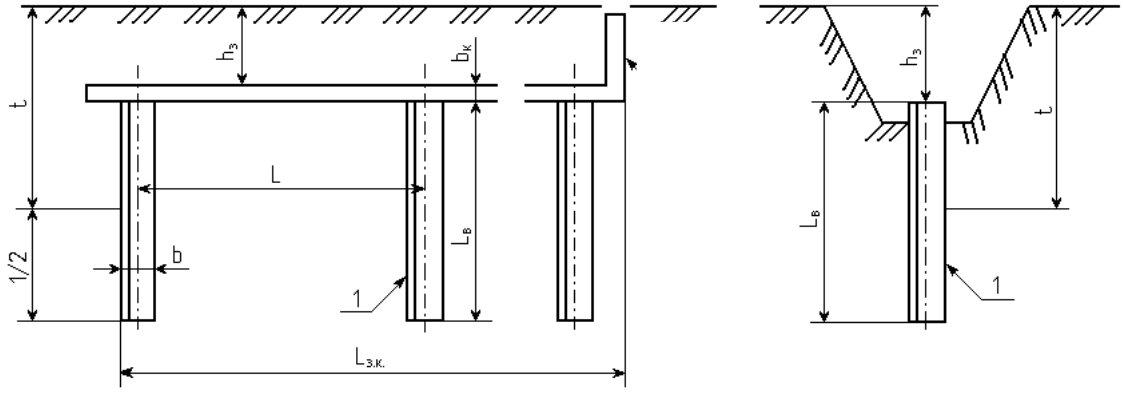


Рисунок 4.1 – Схема заземлювального контуру:

1 – вертикальний заземлювач; 2 – горизонтальний заземлювач;

$h_З$ – глибина закладання заземлювачів; L – відстань між заземлювачами;

$b_К$ – ширина квадрата; t – відстань від середини заземлювача до поверхні грунту; $L_{З,К}$ довжина горизонтального заземлювача; d – ширина кутника;

$L_А$ – довжина вертикального заземлювача.

Схема з’єднання обладнання з магістральною шиною та з’єднання магістральної шини з заземлювальним пристроєм (рисунок 4.1).

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було досліджено актуальну й критичну проблему сучасної цифрової фінансової інфраструктури – шахрайство під час онлайн-платежів. У вступі обґрунтовано необхідність пошуку нових підходів до ідентифікації шахрайської поведінки, зважаючи на стрімке зростання обсягів безготівкових транзакцій, а також обмежену ефективність традиційних rule-based систем. Було підкреслено, що зростаюча складність шахрайських схем, а також динамічність поведінки зловмисників, вимагає використання більш гнучких та адаптивних інструментів, серед яких ключове місце займають алгоритми машинного навчання.

У розділі 1 проведено детальний аналіз предметної області. Було з’ясовано, що традиційні системи, які базуються на жорстко визначених правилах, часто не встигають за динамікою шахрайської поведінки. Вони характеризуються високим рівнем хибнопозитивних і хибнонегативних рішень, що підриває як довіру клієнтів, так і фінансову стабільність компаній. У цьому контексті системи машинного навчання пропонують адаптивні моделі, здатні самостійно навчатися на історичних даних, ідентифікувати складні закономірності та швидко адаптуватися до нових умов.

У другому розділі досліджено сучасні алгоритми машинного навчання, що застосовуються у виявленні шахрайських дій. Розглянуто як традиційні (логістична регресія, дерева рішень, випадкові ліси), так і більш просунуті моделі, зокрема Gradient Boosting Machines (GBM), XGBoost та алгоритм Isolation Forest. Особливу увагу було приділено GBM як одному з найефективніших ансамблевих методів, здатному ітеративно вдосконалювати прогнозування завдяки мінімізації залишкових помилок. Наведено приклади успішного практичного впровадження.

Значне місце в дослідженні зайняв алгоритм Isolation Forest, орієнтований на виявлення аномалій у великих багатовимірних наборах даних. Його переваги включають масштабованість, низькі витрати ресурсів та здатність працювати в

режимі реального часу. Він успішно виявляє як точкові, так і колективні аномалії, що надзвичайно важливо для виявлення витончених шахрайських схем. Однак також було виявлено обмеження, зокрема складність роботи з часовими рядами та схильність до перенавчання при малій кількості даних.

У розділі 3 реалізовано прототип системи виявлення шахрайства. Для побудови класифікаційної моделі використано популярні бібліотеки Python (pandas, sklearn, seaborn та інші). Емпірично перевірено ефективність алгоритмів Random Forest та Naive Bayes, причому кращі результати показала модель Random Forest. Також проведено візуалізацію результатів, яка демонструє чітке розділення між легітимними та шахрайськими транзакціями.

У результаті виконання роботи було підтверджено, що застосування сучасних методів машинного навчання значно підвищує ефективність виявлення шахрайських транзакцій порівняно з традиційними підходами. Розроблена система здатна не лише ідентифікувати потенційно небезпечні транзакції, але й адаптуватися до нових видів загроз, що постійно еволюціонують. Це відкриває перспективи для подальшого вдосконалення систем виявлення шахрайства.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Готович, В. А., С. В. Марценко, and Т. Л. Щербак. "Створення мобільного апаратно-програмного пристрою моніторингу характеристик якості електроенергії." Збірник наукових праць Інституту проблем моделювання в енергетиці ім. ГЄ Пухова 70 (2014): 98-105.

2. Марценко, Сергій Володимирович. Математичне моделювання та статистичні методи обробки даних вимірювань в задачах моніторингу електронавантаження. Diss. Тернопільський національний технічний університет імені Івана Пулюя, 2011.

3. Lytvynenko, I., et al. "Simulation of gas consumption process based on the mathematical model in the form of cyclic random process considering the scale factors." 1st International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2021. 2021.

4. American Institute of Certified Public Accountants (AICPA). 1997. Consideration of Fraud in a Financial Statement Audit. Statement on Auditing Standards (SAS) No. 82. New York, NY: AICPA.

5. Association of Certified Fraud Examiners (ACFE). 2008. Report to the Nation on Occupational Fraud and Abuse. Austin, TX: ACFE.

6. Bayley, L., and S. Taylor. 2007. Identifying earnings management: A financial statement analysis (red flag) approach. Working paper ABN AMRO and University of New South Wales.

7. Beasley, M. 1996. An empirical analysis of the relation between the board of director composition and financial statement fraud. The Accounting Review 71 (4): 443–465.

8. Bell, T., and J. Carcello. 2000. A decision aid for assessing the likelihood of fraudulent financial reporting. Auditing: A Journal of Practice & Theory 19 (1): 169–184.

9. Beneish, M. 1997. Detecting GAAP violation: Implications for assessing earnings management among firms with extreme financial performance. Journal of Accounting and Public Policy 16: 271–309.

10. Beneish, M. 1999. Incentives and penalties related to earnings overstatements that violate GAAP. The Accounting Review 74 (4): 425–457.

11. Breiman, L. 1996. Bagging predictors. Machine Learning 24 (2): 123–140.

12. Breiman, L., J. Friedman, R. Olshen, and C. Stone. 1984. Classification and Regression Trees. Boca Raton, FL: Chapman and Hall/CRC Press.

13. Cecchini, M., H. Aytug, G. Koehler, and P. Pathak. 2010. Detecting management fraud in public companies. Management Science 56 (7): 1146–1160.

14. Chan, P. K., W. Fan, A. L. Prodromidis, and S. J. Stolfo. 1999. Distributed data mining in credit card fraud detection. IEEE Intelligent Systems and Their Applications 14 (6): 67–74.

15. Chawla, N. V. 2005. Data mining for imbalanced datasets: An overview. In The Data Mining and Knowledge Discovery Handbook, edited by Maimon, O., and L. Rokach, 853–867. Secaucus, NJ: Springer-Verlag New York, Inc.

16. Chawla, N. V., K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: Synthetic minority oversampling technique. Journal of Artificial Intelligence Research 16: 321–357.

17. Chen, C., and J. Sennetti. 2005. Fraudulent financial reporting characteristics of the computer industry under a strategic-systems lens. Journal of Forensic Accounting 6 (1): 23–54.

18. Dechow, P., R. Sloan, and A. Sweeney. 1996. Causes and consequences of earnings manipulations: An analysis of firms subject to enforcement actions by the SEC. Contemporary Accounting Research 13 (1): 1–36.

19. Dopuch, N., R. Holthausen, and R. Leftwich. 1987. Predicting audit qualifications with financial and market variables. The Accounting Review 62 (3): 431–454.

20. Drummond, C., and R. C. Holte. 2003. C4.5, class imbalance, and cost sensitivity: Why undersampling beats over-sampling. In The Proceedings of the Workshop on Learning from Imbalanced Datasets II, International Conference on Machine Learning, Washington, D.C.

21. Fan, A., and M. Palaniswami. 2000. Selecting bankruptcy predictors using a support vector machine approach. Neural Networks 6: 354–359.

22. Fanning, K., and K. Cogger. 1998. Neural network detection of management fraud using published financial data. International Journal of Intelligent Systems in Accounting, Finance and Management 7 (1): 21–41.

23. Feroz, E., T. Kwon, V. Pastena, and K. Park. 2000. The efficacy of red flags in predicting the SEC's targets: An artificial neural networks approach. International Journal of Intelligent Systems in Accounting, Finance and Management 9 (3): 145–157.

24. Fries, T., N. Cristianini, and C. Campbell. 1998. The Kernel-Adatron algorithm: A fast and simple learning procedure for support vector machines. In The Proceedings of the 15th International Conference on Machine Learning, Madison, WI.

25. Green, B. P., and J. H. Choi. 1997. Assessing the risk of management fraud through neural network technology. Auditing: A Journal of Practice & Theory 16 (1): 14–28.

26. Hall, M., and G. Holmes. 2003. Benchmarking attribute selection techniques for discrete class data mining. IEEE Transactions on Knowledge and Data Engineering 15 (3): 1–16.

27. Kaminski, K., S. Wetzel, and L. Guan. 2004. Can financial ratios detect fraudulent financial reporting? Managerial Auditing Journal 19 (1): 15–28.

28. Kirkos, E., C. Spathis, and Y. Manolopoulos. 2007. Data mining techniques for the detection of fraudulent financial statements. Expert Systems with Applications 32 (4): 995–1003.

29. Kotsiantis, S., E. Koumanakos, D. Tzelepis, and V. Tampakas. 2006. Forecasting fraudulent financial statements using data mining. International Journal of Computational Intelligence 3 (2): 104–110.

30. Lee, T. A., R. W. Ingram, and T. P. Howard. 1999. The difference between earnings and operating cash flow as an indicator of financial reporting fraud. Contemporary Accounting Research 16 (4): 749– 786.

31. Lin, J., M. Hwang, and J. Becker. 2003. A fuzzy neural network for assessing the risk of fraudulent financial reporting. Managerial Auditing Journal 18 (8): 657–665.

32. Perlich, C., F. Provost, and J. Simonoff. 2003. Tree induction vs. logistic regression: A learning-curve analysis. Journal of Machine Learning Research 4: 211–255.

33. Perols, J., and B. Lougee. 2009. Prior earnings management, forecast attainment, unexpected revenue per employee, and fraud. In The Proceedings of the American Accounting Association Western Region Annual Meeting, San Diego, CA.

34. Phua, C., D. Alahakoon, and V. Lee. 2004. Minority report in fraud detection: Classification of skewed data. SIGKDD Explorations 6 (1): 50–59.

35. Platt, J. 1999. Fast training of support vector machines using sequential minimal optimization. In Advances in Kernel Methods: Support Vector Learning, edited by Scholkopf, B., C. J. C. Burges, and A. J. Smola, 185–208. Cambridge, MA: MIT.

36. Prodromidis, A., P. Chan, and S. Stolfo. 2000. Meta-learning in distributed data mining systems: Issues and approaches. In Advances in Distributed and Parallel Knowledge Discovery, edited by Kargupta, H., and P. Chan, 81–114. Menlo Park, CA: AAAI/MIT.

37. Provost, F., and T. Fawcett. 1997. Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, Menlo Park, CA.

38. Provost, F., T. Fawcett, and R. Kohavi. 1998. The case against accuracy estimation for comparing induction algorithms. In The Proceedings of the Fifteenth International Conference on Machine Learning, Madison, WI.

39. Quinlan, J. R. 1993. C4.5: Programs for Machine Learning. San Francisco, CA: Morgan Kaufmann Publishers.

40. Shin, K. S., T. Lee, and H. J. Kim. 2005. An application of support vector machines in bankruptcy prediction model. Expert Systems with Application 28: 127–135.

41. Summers, S. L., and J. T. Sweeney. 1998. Fraudulently misstated financial statements and insider trading: An empirical analysis. The Accounting Review 73 (1): 131–146.

42. Uzun, H., S. H. Szewczyk, and R. Varma. 2004. Board composition and corporate fraud. Financial Analysts Journal 60 (3): 33–43.

43. Weiss, G. M. 2004. Mining with rarity: A unifying framework. ACM SIGKDD Explorations Newsletter 6 (1): 7–19.

44. West, D., S. Dellana, and J. Qian. 2005. Neural network ensemble strategies for decision applications. Computer and Operations Research 32 (10): 2543–2559.

45. Witten, I. H., and E. Frank. 2005. Data Mining: Practical Machine Learning Tools and Techniques. San Francisco, CA: Morgan Kaufmann Publishers.

46. Wolpert, D. 1992. Stacked generalization. Neural Networks 5 (2): 241–259.

47. Стручок, В. С., Стручок, О. С., & Мудра, Д. В. (2017). Навчальний посібник до написання розділу дипломного проекту та дипломної роботи ″Безпека в надзвичайних ситуаціях ″для студентів всіх спец. денної, заочної (дистанційної) та екстернатної форм навчання.

48. Стручок, В. С. (2022). Техноекологія та цивільна безпека. Частина "Цивільна безпека". Навчальний посібник.

49. Жидецький, В. Ц., Джигирей, В. С., & Мельников, О. В. (2000). Основи охорони праці. Львів: Афіша, 350, 132-136.

50. Навакатікян, О. О., Кальниш, В. В., & Стрюков, С. М. (1997). Охорона праці користувачів комп'ютерних відеодисплейних терміналів. О. Навакатікян.

ДОДАТКИ

Додаток А

Програмний код

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import scipy as sp
from tabulate import tabulate
import random
import tensorflow as tf

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
/kaggle/input/online-payment-fraud-detection/onlinefraud.csv
```

Exploratory Data Analysis

```
df = pd.read_csv('/kaggle/input/online-payment-fraud-detection/onlinefraud.csv')

df.drop('isFlaggedFraud', axis=1, inplace=True)

df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 10 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
dtypes: float64(5), int64(2), object(3)
memory usage: 485.4+ MB
```

- The dataset consists of 6,362,620 observations.

```
df.sample(5)
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
2889785	228	CASH_OUT	364310.96	C1882002610	127348.00	0.00	C1255784537	4814640.42	5178951.39	0
2805863	225	PAYMENT	20383.65	C149337475	403052.00	382668.35	M1589311961	0.00	0.00	0

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
1772525	162	CASH_IN	27891.95	C46285081	29584.00	57475.95	C376931188	640973.19	553796.55	0
6261030	608	PAYMENT	994.24	C1126305428	12549.64	11555.40	M595147503	0.00	0.00	0
3707836	277	PAYMENT	15309.73	C768133022	23391.50	8081.77	M1419231647	0.00	0.00	0

```
df.isnull().sum()
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
dtype: int64
```

- There are no missing values.

```
fraud_min_max = [
    ['amount', df.amount.min(), df.amount.max()],
    ['oldbalanceOrg', df.oldbalanceOrg.min(), df.oldbalanceOrg.max()],
    ['newbalanceOrig', df.newbalanceOrig.min(), df.newbalanceOrig.max()],
    ['oldbalanceDest', df.oldbalanceDest.min(), df.oldbalanceDest.max()],
    ['isFraud', df.isFraud.min(), df.isFraud.max()]
]

print(
    tabulate(
        fraud_min_max,
        headers=['columns', 'min value', 'max value'],
        showindex=True,
        tablefmt='github',
        numalign='right'
    )
)
```

```
|    | columns        |   min value |   max value |
|----|----------------|-------------|-------------|
|  0 | amount         |           0 | 9.24455e+07 |
|  1 | oldbalanceOrg  |           0 |  5.9585e+07 |
|  2 | newbalanceOrig |           0 |  4.9585e+07 |
|  3 | oldbalanceDest |           0 | 3.56016e+08 |
|  4 | isFraud        |           0 |           1 |
```

```
# Downcast numerical columns with smaller dtype
for col in df.columns:
    if df[col].dtype == 'float64':
        df[col] = pd.to_numeric(df[col], downcast='float')
    if df[col].dtype == 'int64':
        df[col] = pd.to_numeric(df[col], downcast='unsigned')

# Use category dtype for categorical column
df['type'] = df['type'].astype('category')
# Check duplicate values
df.duplicated().sum()
```

```
0
```

- There are no duplicate values.

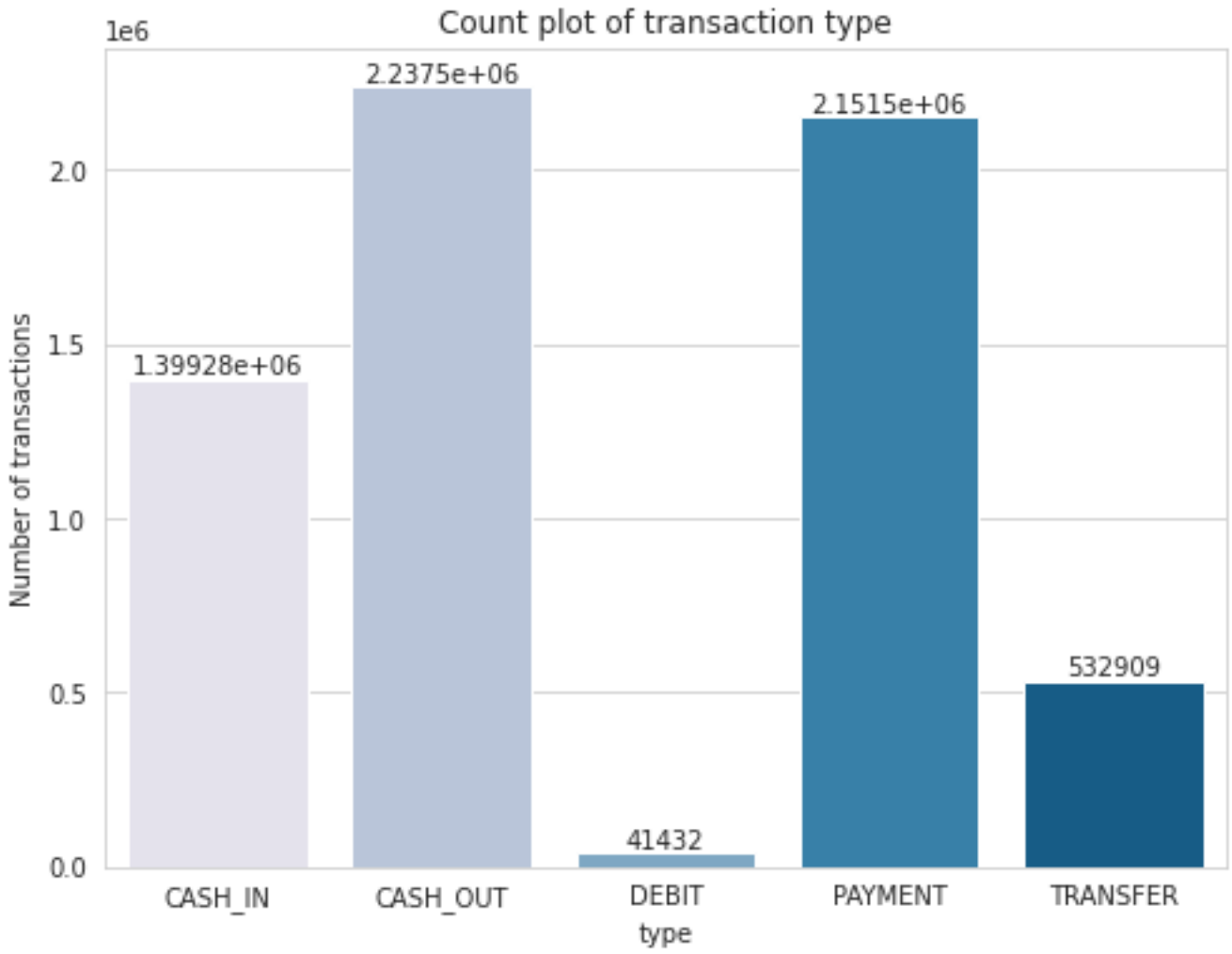
```
sns.set_style('whitegrid')
plt.rcParams['figure.figsize'] = (8,6)
```

Univariate data visualization

```
df['step'].value_counts()
19     51352
18     49579
187    49083
235    47491
307    46968
       ...
432        4
706        4
693        4
112        2
662        2
Name: step, Length: 743, dtype: int64
```

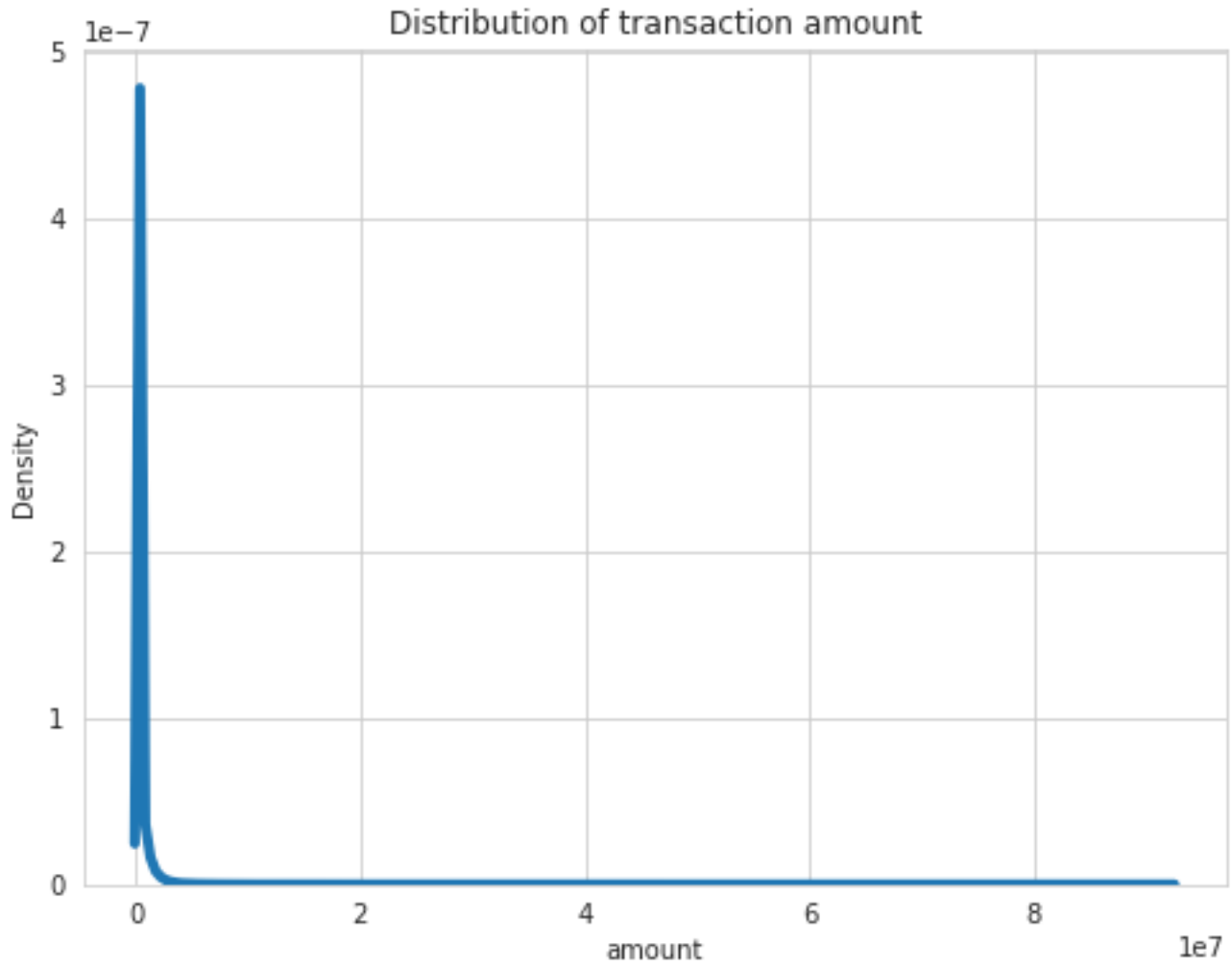
- There are **743** steps, and every step has **at least 2** occurrences.

```
ax = sns.countplot(x='type', data=df, palette='PuBu')
for container in ax.containers:
    ax.bar_label(container)
plt.title('Count plot of transaction type')
plt.legend(bbox_to_anchor=(1.05,1), loc='upper left')
plt.ylabel('Number of transactions')
Text(0, 0.5, 'Number of transactions')
```



- **Cash out** is **the most numerous** transaction type, followed by payment, cash in, transfer and debit types.

```
sns.kdeplot(df['amount'], linewidth=4)
plt.title('Distribution of transaction amount')
Text(0.5, 1.0, 'Distribution of transaction amount')
```

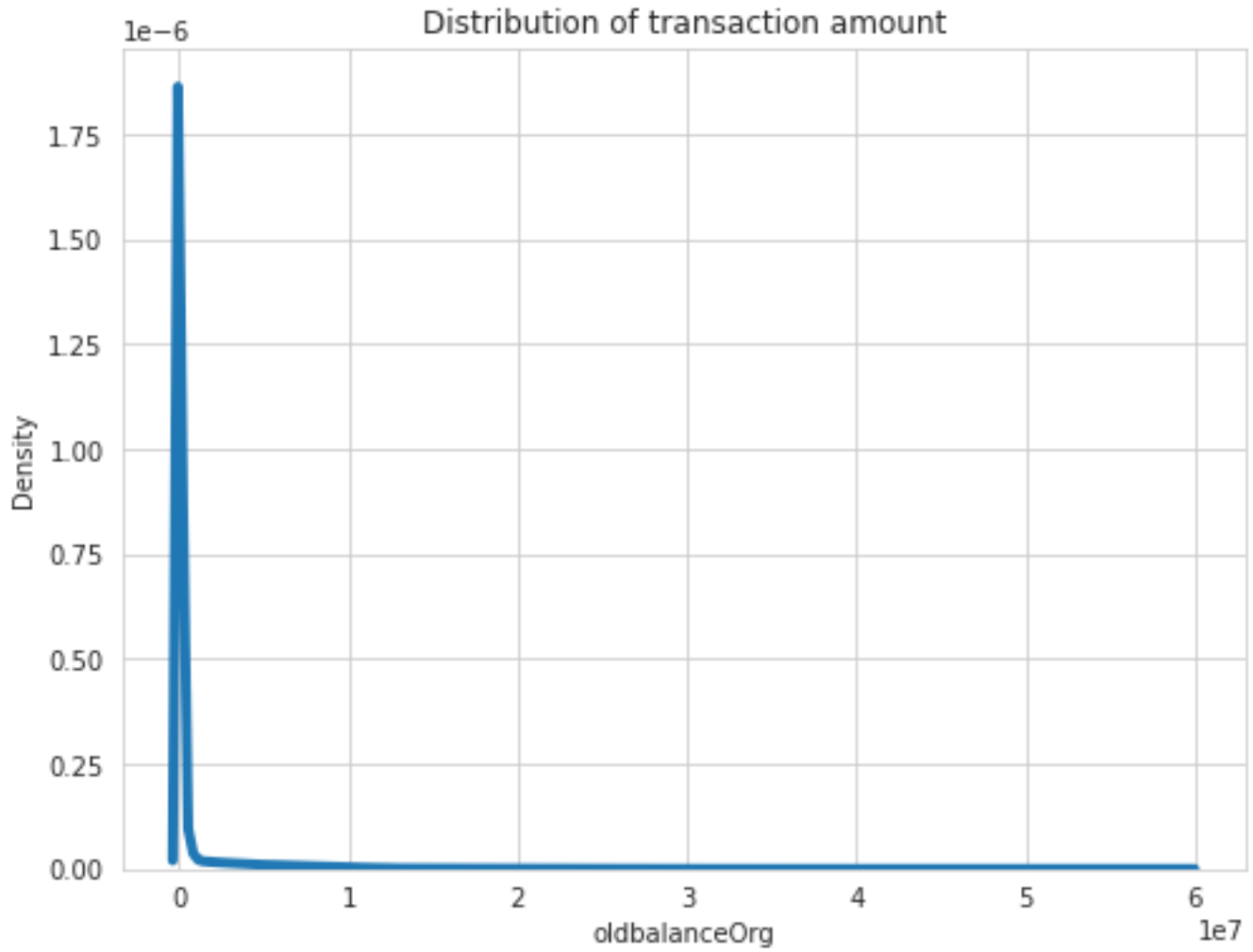


- The distribution of transaction amounts is **right skewed**.
- This indicates that most values are clustered around the left tail of the distribution, with the longer right tail.
- (mode < median < mean)

```
df['nameOrig'].value_counts()
C1902386530    3
C363736674     3
C545315117     3
C724452879     3
C1784010646    3
              ..
C98968405      1
C720209255     1
C1567523029    1
C644777639     1
C1280323807    1
Name: nameOrig, Length: 6353307, dtype: int64
```

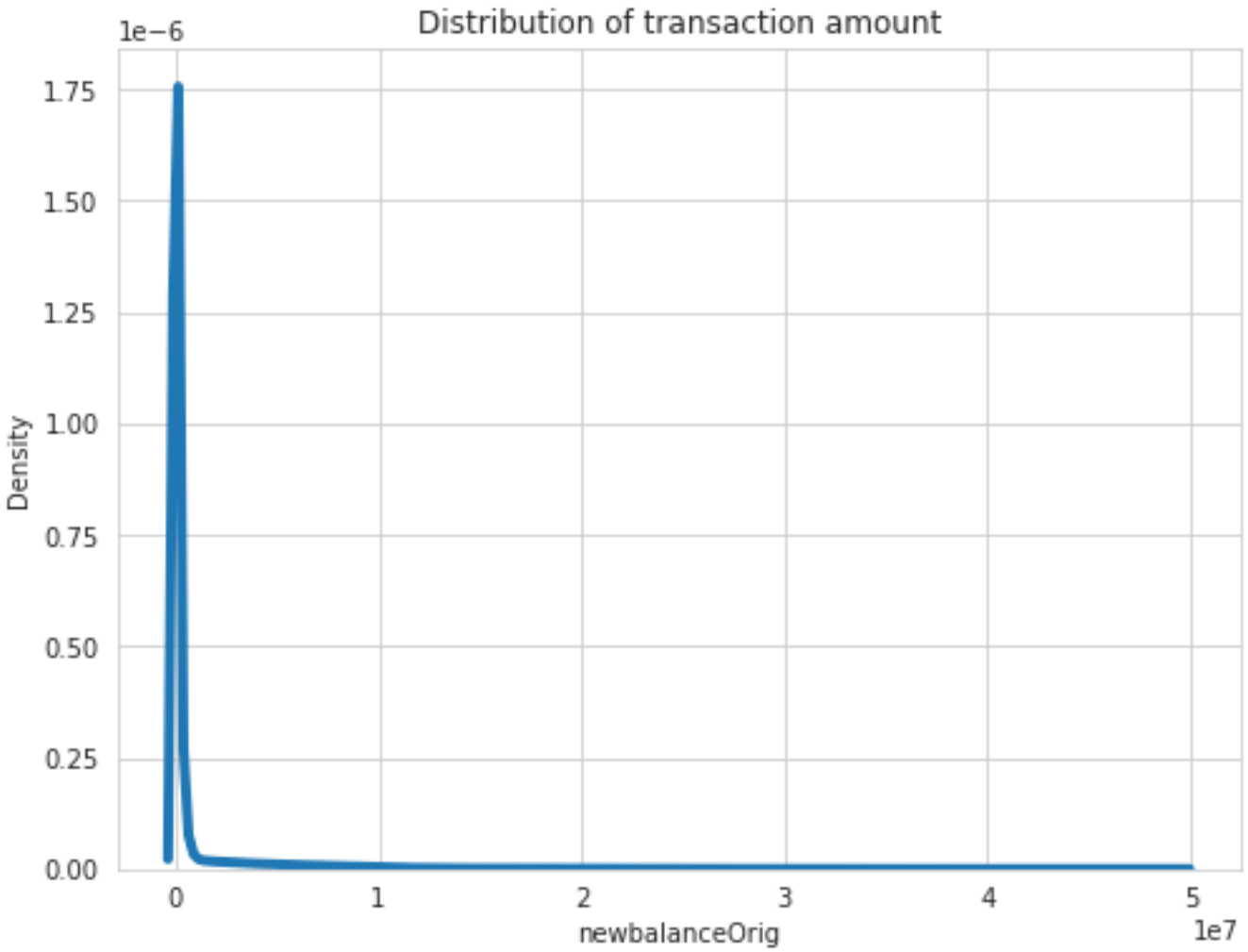
- There are **6353307** initial customers, and every step has **at least 1** occurrence.

```
sns.kdeplot(df['oldbalanceOrg'], linewidth=4)
plt.title('Distribution of transaction amount')
Text(0.5, 1.0, 'Distribution of transaction amount')
```



- The distribution of pre-transaction balances of the initial customers is **right skewed**.

```
sns.kdeplot(df['newbalanceOrig'], linewidth=4)
plt.title('Distribution of transaction amount')
Text(0.5, 1.0, 'Distribution of transaction amount')
```

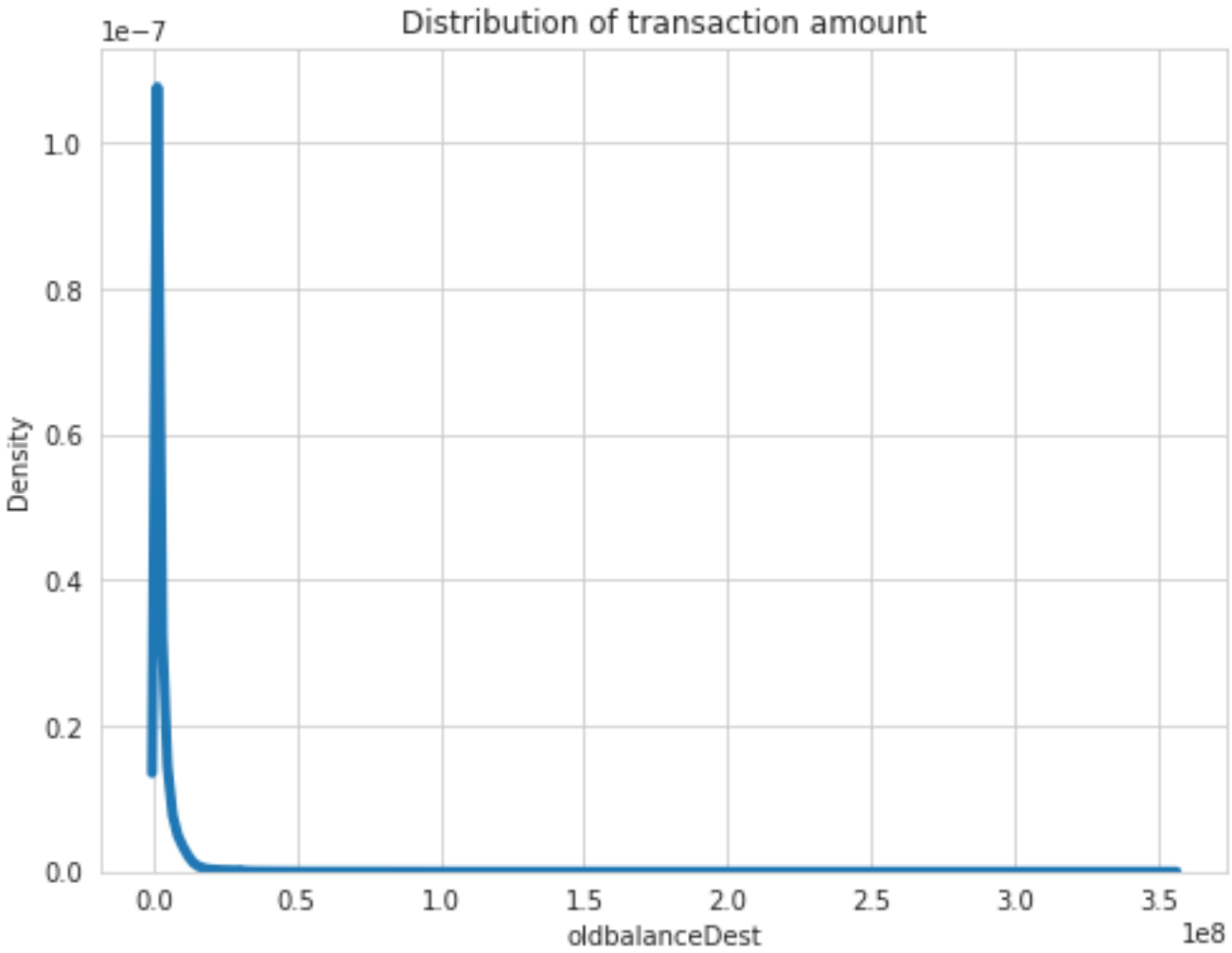


- The distribution of post-transaction balances of the initial customers is **right skewed**.

```
df['nameDest'].value_counts()
C1286084959    113
C985934102     109
C665576141     105
C2083562754    102
C248609774     101
              ...
M1470027725      1
M1330329251      1
M1784358659      1
M2081431099      1
C2080388513      1
Name: nameDest, Length: 2722362, dtype: int64
```

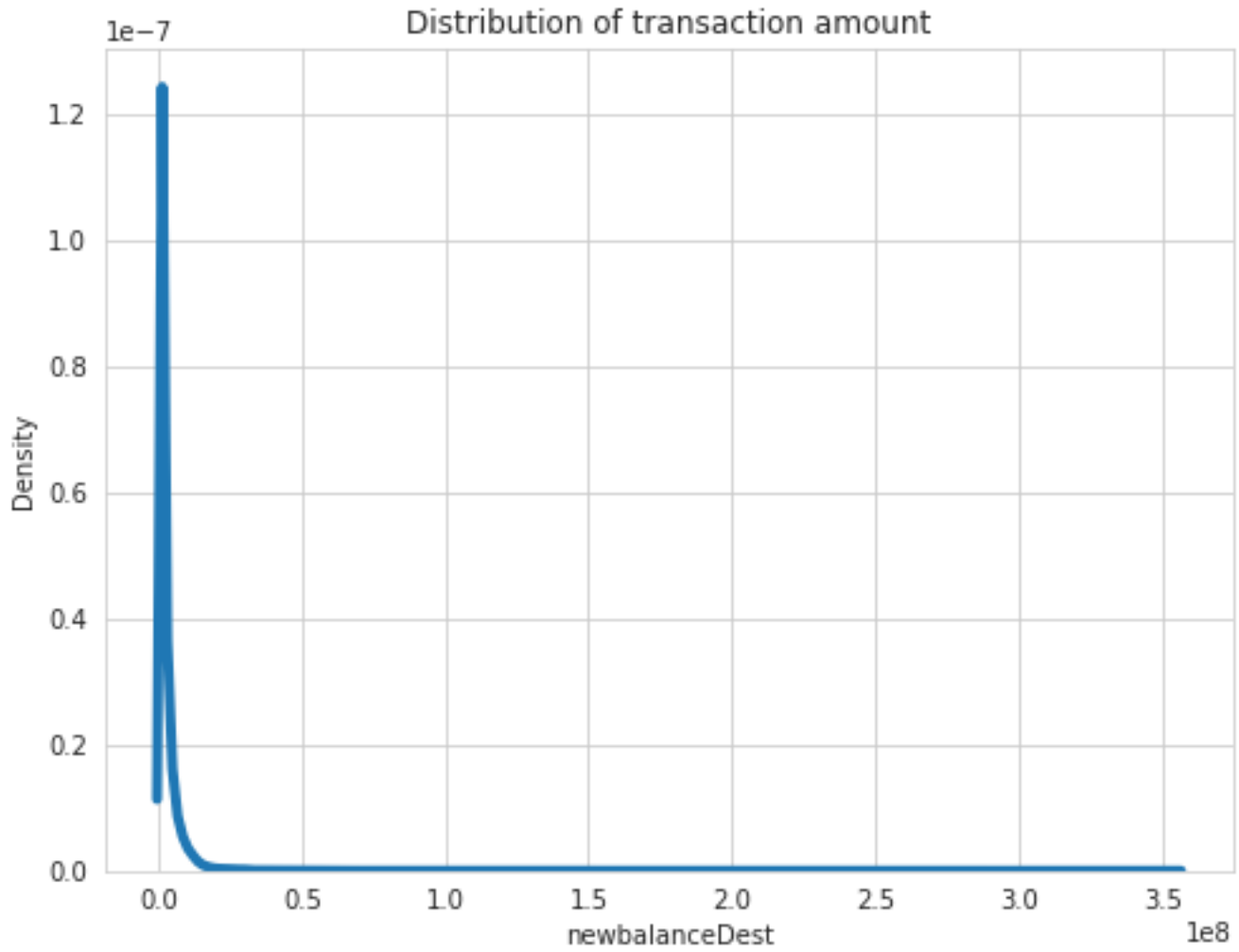
- There are **2722362** recipients, and every step has **at least 1** occurrence.

```
sns.kdeplot(df['oldbalanceDest'], linewidth=4)
plt.title('Distribution of transaction amount')
Text(0.5, 1.0, 'Distribution of transaction amount')
```



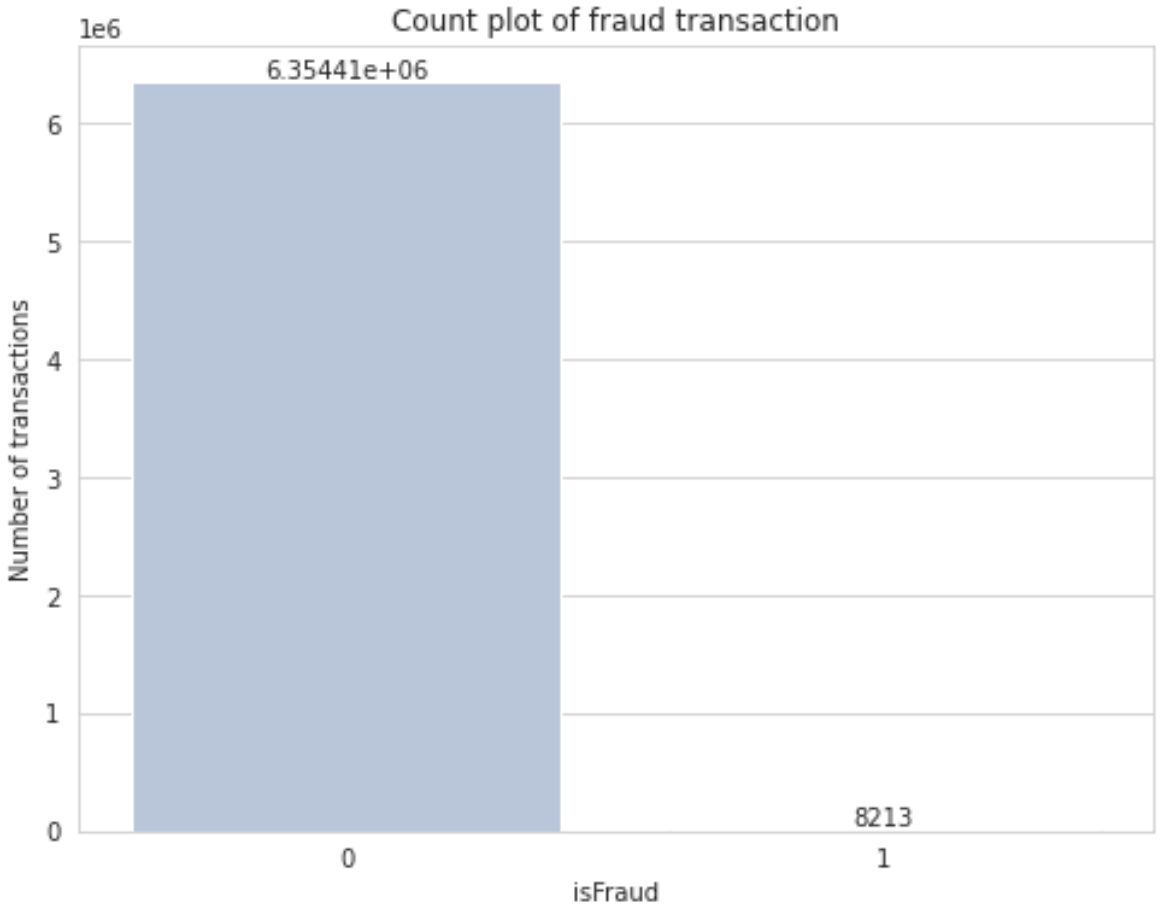
- The distribution of pre-transaction balances of the recipient is **right skewed**.

```
sns.kdeplot(df['newbalanceDest'], linewidth=4)
plt.title('Distribution of transaction amount')
Text(0.5, 1.0, 'Distribution of transaction amount')
```



- The distribution of post-transaction balances of the recipient is **right skewed**.

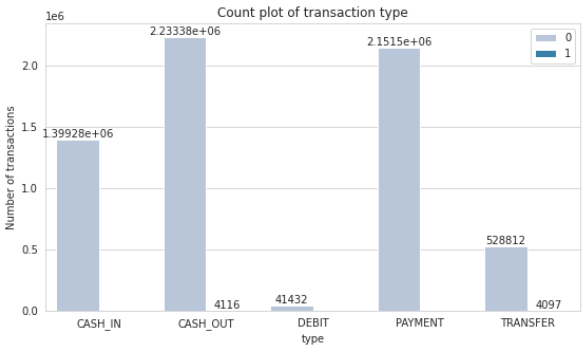
```
ax = sns.countplot(x='isFraud', data=df, palette='PuBu')
for container in ax.containers:
    ax.bar_label(container)
plt.title('Count plot of fraud transaction')
plt.ylabel('Number of transactions')

del ax
```

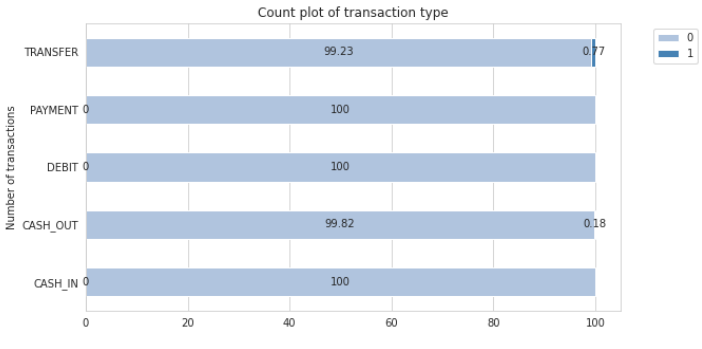


- There are much **more non-fraudulent transactions** than fraudulent transactions.

Bivariate data visualization

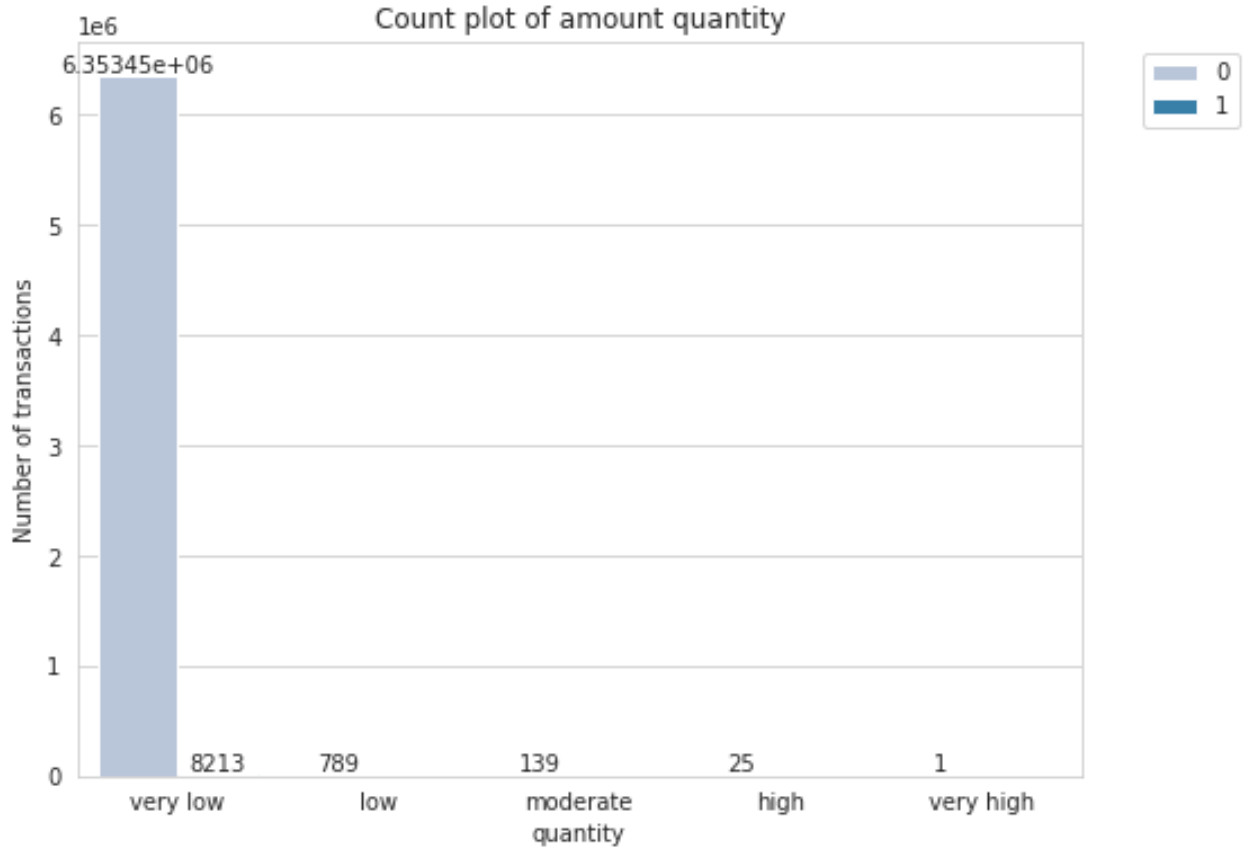
```
fig, ax = plt.subplots(1,2,figsize=(20,5))

sns.countplot(x='type', data=df, hue='isFraud', palette='PuBu', ax=ax[0])
for container in ax[0].containers:
    ax[0].bar_label(container)
ax[0].set_title('Count plot of transaction type')
ax[0].legend(loc='best')
ax[0].set_ylabel('Number of transactions')

df2 = df.groupby(['type', 'isFraud']).size().unstack()
df2.apply(lambda x : round(x/sum(x)*100, 2), axis=1).plot(kind='barh',
stacked=True, color=['lightsteelblue', 'steelblue'], ax=ax[1])
for container in ax[1].containers:
    ax[1].bar_label(container, label_type='center')
ax[1].set_title('Count plot of transaction type')
ax[1].legend(bbox_to_anchor=(1.05,1), loc='upper left')
ax[1].set_ylabel('Number of transactions')
ax[1].grid(axis='y')
```





- Fraudulent transactions only occur in debit and transfer types.

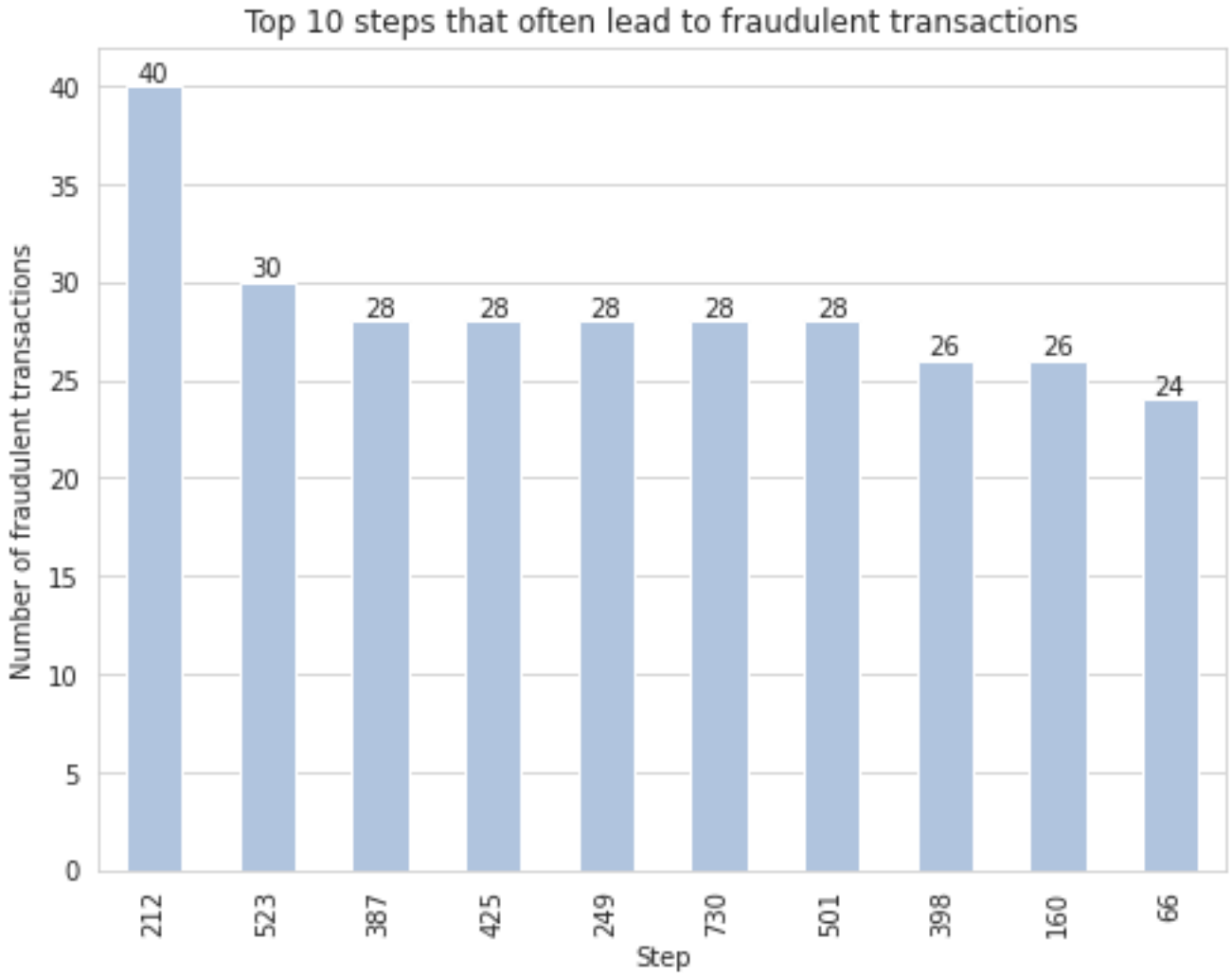
```
df['quantity'] = pd.cut(df['amount'], 5, labels=['very low', 'low', 'moderate',
'high', 'very high'])

ax = sns.countplot(x='quantity', data=df, hue='isFraud', palette='PuBu')
for container in ax.containers:
    ax.bar_label(container)
plt.title('Count plot of amount quantity')
plt.legend(bbox_to_anchor=(1.05,1), loc='upper left')
plt.ylabel('Number of transactions')
Text(0, 0.5, 'Number of transactions')
```



- All fraudulent transactions fall into the category of very low amounts.
- This suggests that in most cases, small transactions are more prone to fraudulent transactions.

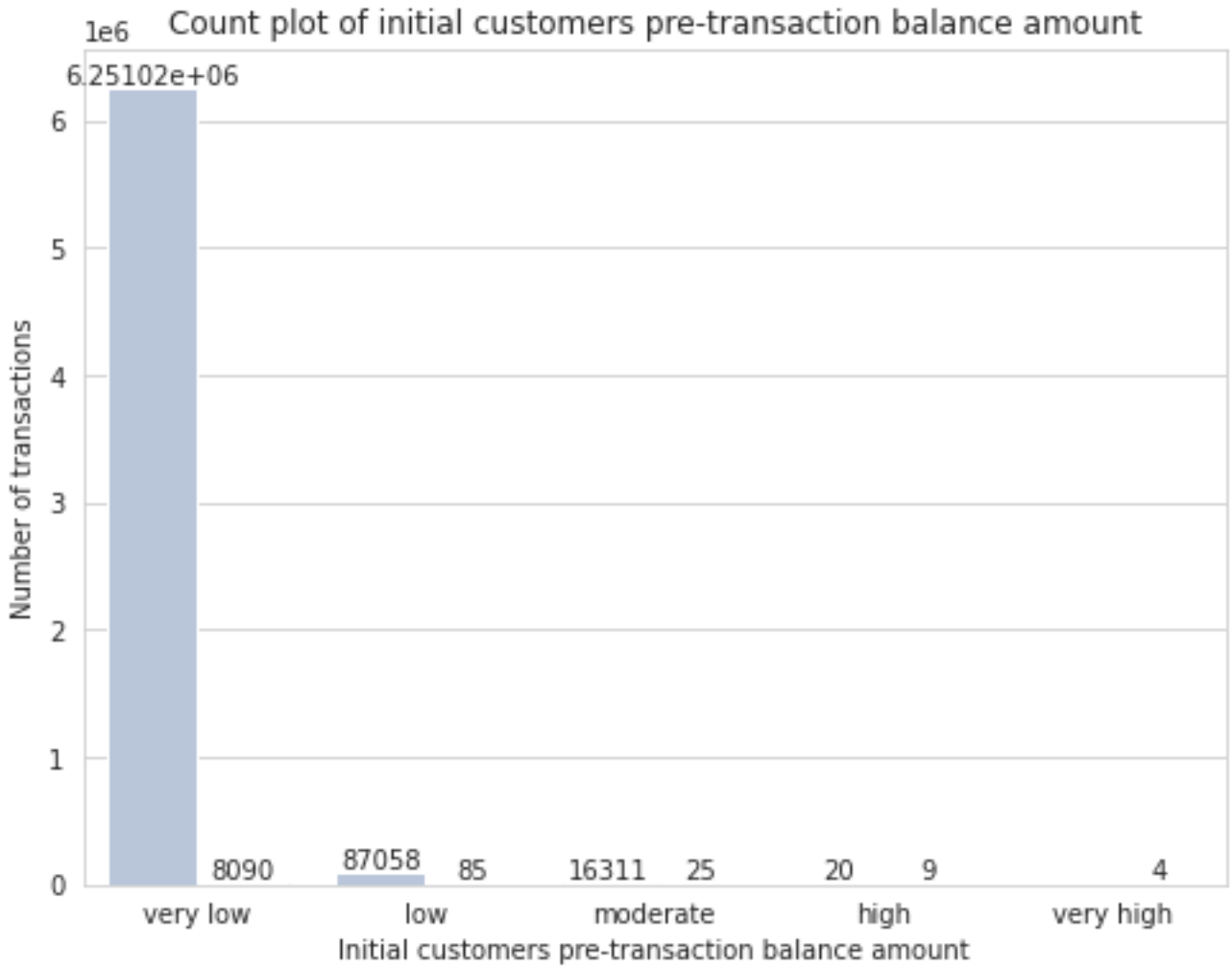
```
df1 = df[df['isFraud']==1]
df2 = df1['step'].value_counts().head(10)
ax = df2.plot(kind='bar', color='lightsteelblue')
```

```
for container in ax.containers:
    ax.bar_label(container)
plt.title('Top 10 steps that often lead to fraudulent transactions')
plt.ylabel('Number of fraudulent transactions')
plt.xlabel('Step')
plt.grid(axis='x')

del ax, df2
```



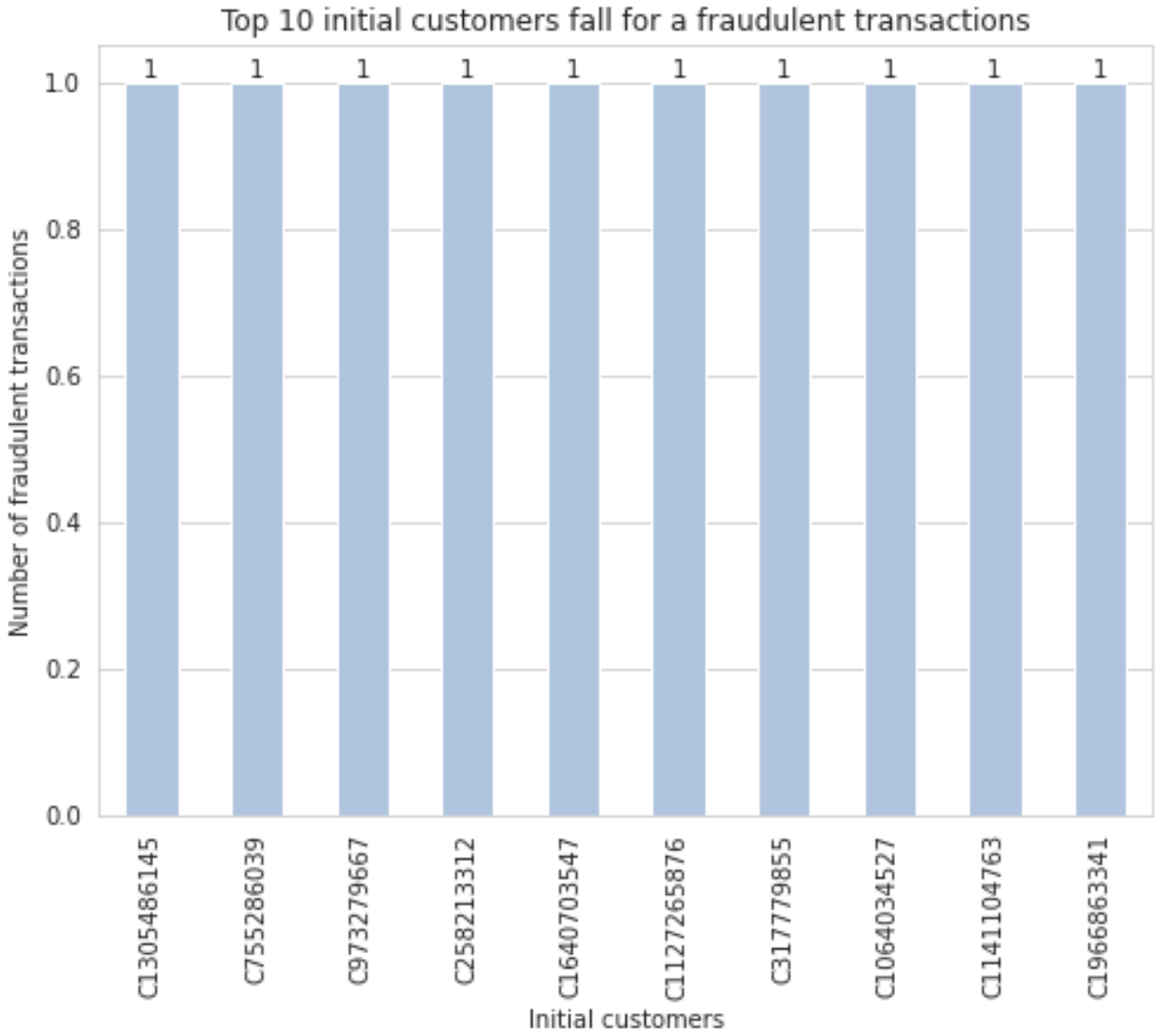
- **Step 212** has the highest number of fraudulent transactions, 40 cases.
- This indicates that Step 212 is the step that will most likely lead to fraudulent transactions.

```
df['oldbalanceOrg_amt'] = pd.cut(df['oldbalanceOrg'], 5, labels=['very low',
'low', 'moderate', 'high', 'very high'])

ax = sns.countplot(x='oldbalanceOrg_amt', data=df, hue='isFraud', palette='PuBu')
for container in ax.containers:
    ax.bar_label(container)
plt.title('Count plot of initial customers pre-transaction balance amount')
plt.legend(bbox_to_anchor=(1.05,1), loc='upper left')
plt.ylabel('Number of transactions')
plt.xlabel('Initial customers pre-transaction balance amount')
Text(0.5, 0, 'Initial customers pre-transaction balance amount')
```



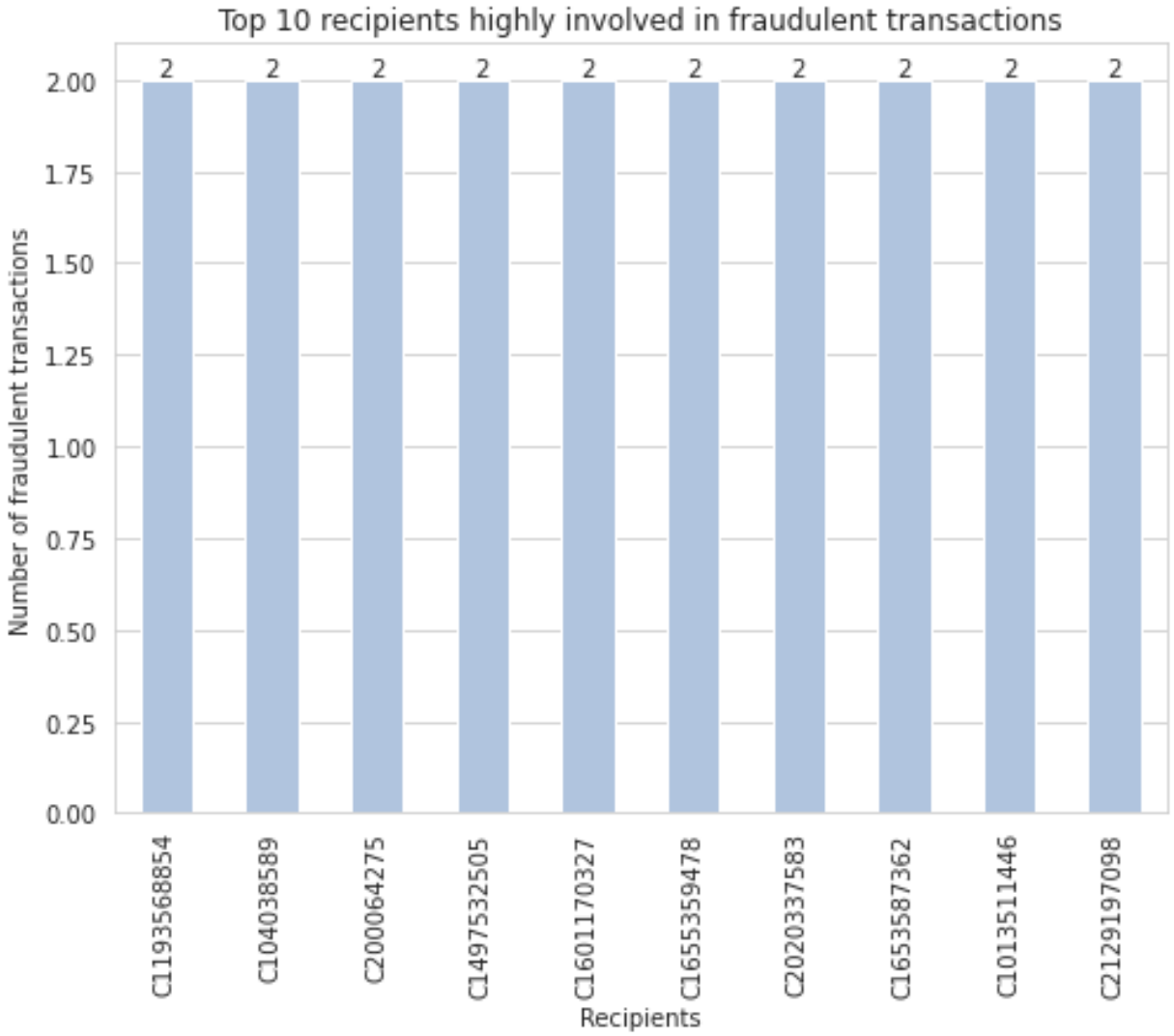
- Initial customers with **very low pre-transaction balances** has the highest number of fraudulent transactions.
- This means that initial customers with very low pre-transaction balances may be more likely to fall for a fraudulent transaction.

```
df2 = df1['nameOrig'].value_counts().head(10)
ax = df2.plot(kind='bar', color='lightsteelblue')
for container in ax.containers:
    ax.bar_label(container)
plt.title('Top 10 initial customers fall for a fraudulent transactions')
plt.ylabel('Number of fraudulent transactions')
plt.xlabel('Initial customers')
plt.grid(axis='x')

del ax, df2
```



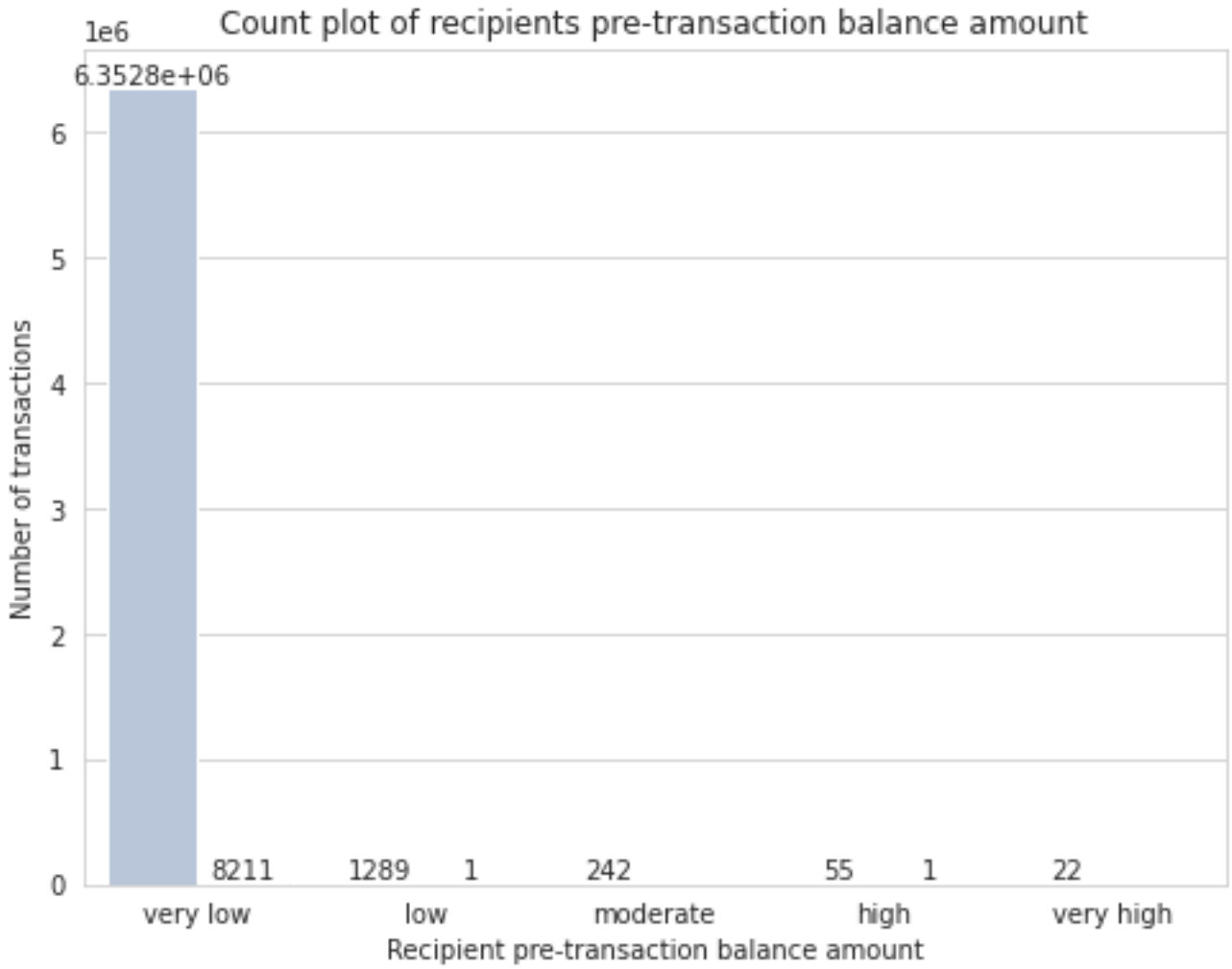
- Initial customers are scammed of **at most one** for a fraudulent transaction.

```
df2 = df1['nameDest'].value_counts().head(10)
ax = df2.plot(kind='bar', color='lightsteelblue')
for container in ax.containers:
    ax.bar_label(container)
plt.title('Top 10 recipients highly involved in fraudulent transactions')
plt.ylabel('Number of fraudulent transactions')
plt.xlabel('Recipients')
plt.grid(axis='x')

del ax, df2
```



- Suspicious recipients only have a **maximum of 2** involved fraudulent transactions.

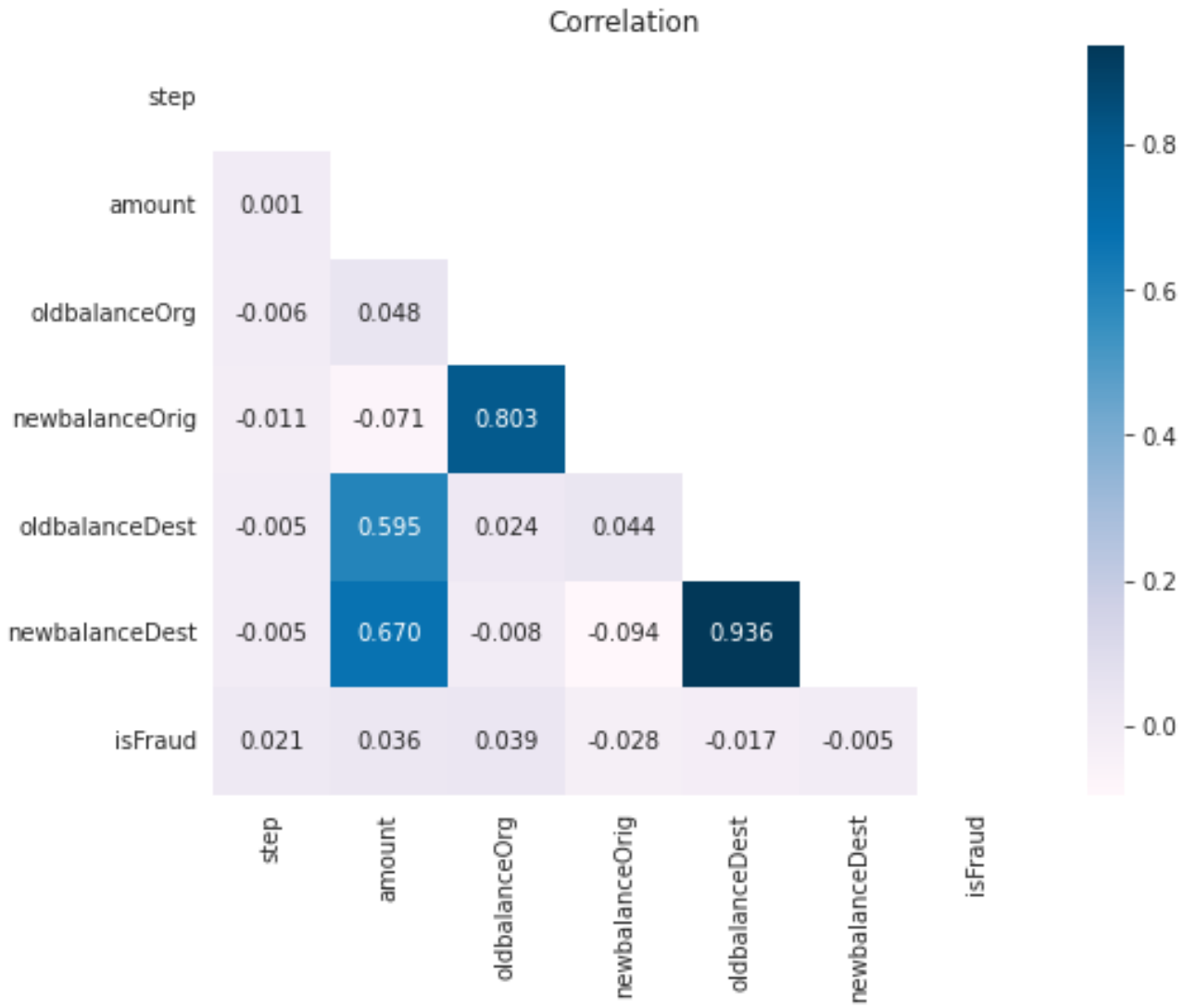
```
df['oldbalanceDest_amt'] = pd.cut(df['oldbalanceDest'], 5, labels=['very low',
'low', 'moderate', 'high', 'very high'])

ax = sns.countplot(x='oldbalanceDest_amt', data=df, hue='isFraud', palette='PuBu')
for container in ax.containers:
    ax.bar_label(container)
plt.title('Count plot of recipients pre-transaction balance amount')
plt.legend(bbox_to_anchor=(1.05,1), loc='upper left')
plt.ylabel('Number of transactions')
plt.xlabel('Recipient pre-transaction balance amount')
Text(0.5, 0, 'Recipient pre-transaction balance amount')
```



- Recipients with **very low pre-transaction balances** has the highest number of fraudulent transactions.
- This implies that recipients with very low pre-transaction balances may be more susceptible to fraudulent transactions.

Multivariate data visualization

```
corr_matrix = df.corr('spearman')
sns.heatmap(corr_matrix, cbar=True, annot=True, mask =
np.triu(np.ones_like(corr_matrix, dtype = bool)), fmt='.3f', cmap='PuBu')
plt.title('Correlation')
Text(0.5, 1.0, 'Correlation')
```



- oldbalanceOrg and newbalanceOrig has **strong positive** relationship.
- oldbalanceDest and newbalanceDest has **strong positive** relationship.
- oldbalanceOrg and amount has **weak positive** relationship.
- newbalanceOrig and amount has **moderate positive** relationship.

Model Building

```
# Data preprocessing
df['type'] = df['type'].map({'PAYMENT':0, 'CASH_IN':1, 'DEBIT':2, 'CASH_OUT':3,
'TRANSFER':4})
```

- Due to the large dataset, Random Forest and Logistic Regression with balanced class weight are used to identify online payment fraud.

```
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import cross_val_score
```

```
from sklearn.metrics import classification_report, roc_curve, auc,
ConfusionMatrixDisplay


seed = 42
np.random.seed(seed)
random.seed(seed)
tf.random.set_seed(seed)


X = df.copy()
X.drop(['nameOrig', 'newbalanceOrig', 'nameDest', 'newbalanceDest', 'quantity',
'oldbalanceOrg_amt', 'oldbalanceDest_amt'], axis=1, inplace=True)
y = X.pop('isFraud')


# Stratified train-test split
skfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=seed)
for train_idx, test_idx in skfold.split(X,y):
    X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]


sc = StandardScaler()
scaled_train = sc.fit_transform(X_train)
scaled_test = sc.transform(X_test)
X_train = pd.DataFrame(scaled_train, index=X_train.index, columns=X_train.columns)
X_test = pd.DataFrame(scaled_test, index=X_test.index, columns=X_test.columns)


X_train, y_train =
RandomUnderSampler(sampling_strategy='majority').fit_resample(X_train, y_train)
def model_comparison_evaluate(classifiers, X, y):
    print('K-Fold Cross-Validation:\n')
    for name, model in classifiers.items():
        print('{}:'.format(name))

        scoring = ['accuracy', 'precision', 'recall', 'f1', 'roc_auc']

        for score in scoring:
            scores = cross_val_score(model, X, y, scoring=score, cv=skfold,
n_jobs=-1)
            print('Mean {} score: {:.3f} ({:.3f})'.format(score, scores.mean(),
scores.std()))

        print('\n')
classifiers = { 'Random Forest
Classifier':RandomForestClassifier(class_weight='balanced', random_state=seed),
                'Logistic Regression': LogisticRegression(class_weight='balanced',
random_state=seed)
              }
model_comparison_evaluate(classifiers, X_train, y_train)
K-Fold Cross-Validation:

Random Forest Classifier:
Mean accuracy score: 0.985 (0.003)
Mean precision score: 0.975 (0.006)
Mean recall score: 0.996 (0.002)
Mean f1 score: 0.985 (0.003)
Mean roc_auc score: 0.998 (0.000)
```

```
Logistic Regression:
Mean accuracy score: 0.848 (0.007)
Mean precision score: 0.843 (0.008)
Mean recall score: 0.856 (0.005)
Mean f1 score: 0.849 (0.006)
Mean roc_auc score: 0.927 (0.004)
```

```
model = RandomForestClassifier(class_weight='balanced', random_state=seed)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
y_pred_score = model.predict_proba(X_test)[:,1]
print('Random Forest Classifier:')
print(classification_report(y_pred, y_test, labels=[0,1], target_names=['Non-Fraud
[0]', 'Fraud [1]']), '\n')
```

```
fig, ax = plt.subplots(1, 2, figsize=(20,5))
ax[0].set_title('Confusion Matrix of Random Forest Model:')
ConfusionMatrixDisplay.from_predictions(y_test, y_pred, colorbar=False,
values_format='', cmap='crest', ax=ax[0])
ax[0].grid(False)
```

```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_score)
roc_auc = auc(fpr, tpr)
ax[1].set_title('ROC Curve - Random Forest Classifier')
ax[1].plot(fpr, tpr, label = 'AUC = %0.3f' % roc_auc, c='steelblue')
ax[1].plot([0,1],[0,1],'--', c='lightsteelblue')
ax[1].legend(loc='lower right')
ax[1].set_ylabel('True Positive Rate')
ax[1].set_xlabel('False Positive Rate')
Random Forest Classifier:
              precision    recall  f1-score   support

Non-Fraud [0]       0.98      1.00      0.99   1239159
    Fraud [1]       1.00      0.05      0.09     33365

     accuracy                           0.98   1272524
    macro avg       0.99      0.52      0.54   1272524
 weighted avg       0.98      0.98      0.96   1272524
```

```
Text(0.5, 0, 'False Positive Rate')
```