МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра прикладної математики та моделювання складних систем

«До захисту допущено»

Завідувач кафедри ПМ та МСС

_____ Ігор Коплик
(підпис)

«___» _____20 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 113 «Прикладна математика»,

освітньо-професійної програми «Наука про дані та моделювання складних систем»

на тему: «Виявлення аномалій в операціях з кредитними картками»

Здобувачки групи ПМ.м-21 Пороскун Олени Олегівни

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Олена Пороскун
(підпис)

Керівник док. фіз.-мат. наук, професор Олександр Лисенко _____
(підпис)

Суми – 2023

АНОТАЦІЯ

Кваліфікаційна робота: 53 с., 31 рис., 17 джерел.

Мета роботи: провести аналіз сильно незбалансованих даних з метою виявлення аномалій на прикладі даних про транзакції. Визначити кращий алгоритм для аналізу сильно незбалансованих даних.

Об'єкт дослідження: дані про операції з кредитними картками, що містять шахрайські операції (0,2% від усього набору даних).

Предмет дослідження: характеристики методів та алгоритмів, які дозволяють провести аналіз сильно незбалансованих даних.

Методи аналізу: метод гаусівського виявлення аномалій, метод ізоляційного лісу.

У роботі створено алгоритм на основі функцій розподілу імовірності для виявлення аномальних (шахрайських) операцій з кредитними картками за даними https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud та проведено порівняльний аналіз створеного алгоритму з алгоритмом на базі ізоляційного лісу з бібліотеки scikit-learn. Для підвищення якості виявлення аномалій проведено попередню обробку даних, зокрема виділення впливових для розв'язку задачі ознак. Було проведено оптимізацію параметрів на валідаційних даних для підвищення якості моделей. Така попередня обробка даних дозволила підвищити f2-оцінку (f2-score) з 0.767285 до 0.834671. Отримано, що найвища якість виявлення аномалій на досліджуваних тестових даних досягається за допомогою метода гаусівського виявлення аномалій (f2-score дорівнює 81,65%).

Ключові слова: ВИЯВЛЕННЯ АНОМАЛІЙ, GAUSSIAN ANOMALY DETECTION, ISOLATION FOREST, F2-SCORE.

ЗМІСТ

ВСТУП

У контексті стрімкого розвитку технологій та електронних фінансів, а також зростання обсягів онлайн-транзакцій, використання кредитних карт не лише стало зручним, але й привабливим для фінансового шахрайства. Фінансові установи та компанії електронної комерції стикаються з викликом ефективного виявлення та запобігання незаконним діям, що можуть призвести до фінансових втрат та порушень конфіденційності даних клієнтів [1].

Ця робота присвячена вивченню та застосуванню сучасних методів виявлення аномалій в операціях з кредитними картками за допомогою технологій машинного навчання та аналізу великих даних. З врахуванням того, що фінансові шахраї часто вдосконалюють свої методи, важливим стає розробка інноваційних підходів, які забезпечать надійний захист та реагування на потенційні загрози.

Метою даного дослідження є розгляд сучасних підходів до виявлення аномалій в операціях з кредитними картками.

Результати цього дослідження мають потенціал значно покращити безпеку фінансових транзакцій та допомогти у виявленні аномалій, що дозволить забезпечити високий рівень довіри серед користувачів кредитних карт та зберегти фінансову стійкість у сучасному цифровому середовищі.

РОЗДІЛ 1
ВИЯВЛЕННЯ АНОМАЛІЙ (АНАЛІТИЧНИЙ ОГЛЯД)

1.1. Виявлення аномалій

Виявлення аномалій зазвичай означає виявлення викидів у наборі даних, який переважно складається з "нормальних" точок даних. Основна ідея полягає в тому, щоб виявити записи, які відрізняються від більшості даних і, зазвичай, є результатом помилкових операцій, дій системи або фінансових махінацій. Фінансові транзакції, які фіксують потік активів між сторонами, стають особливо цікавими для виявлення аномалій. При тривалому спостереженні за таким потоком можна виявити закономірності, що надає можливість визначити незвичайні події [1].

З розвитком обчислювальних машин стало можливим застосовувати методи машинного навчання, що раніше було неможливо, оскільки самі ідеї та алгоритми були створені ще у середині минулого століття. Зараз відмовляються від використання заздалегідь заданих правил у виявленні аномалій, оскільки вони не гнучкі та вимагають постійного втручання людини та налаштувань [2].

Використання статичних правил для виявлення аномалій при великій кількості операцій призводить до ускладнення системи виявлення через збільшену кількість можливих варіацій аномалій. Це ускладнює систему, робить її важкодоступною для модифікацій і усунення помилок у разі їх виникнення. Таким чином, автоматизація виявлення аномалій у фінансових операціях стає дуже актуальною та перспективною.

У сфері статистики та аналізу даних термін "аномалія" вказує на рідкісне спостереження, яке виявляє значні відхилення від основної маси даних і не відповідає чітко визначеному концепту нормальної поведінки. Існує можливість, що такі спостереження можуть бути породжені іншим

механізмом або не узгоджуються з іншими даними набору. Процес виявлення таких відхилень зазвичай називається виявленням аномалій.

Останнім часом машинне навчання все більше використовується для автоматизації цього процесу через навчання під наглядом (коли спостереження позначаються як нормальні або аномальні), напівнавчання під наглядом (коли маркується лише невелика частина спостережень) і навчання без нагляду (коли спостереження не маркуються).

Виявлення аномалій особливо ефективне в умовах, коли:

- Аномалії дуже рідкісні в наборі даних.
- Характеристики аномальних спостережень значно відрізняються від характеристик нормальних спостережень.
- Аномалії можуть виникати з різних (потенційно нових) причин.

Виявлення аномалій є важливим і корисним завданням, зокрема в контексті виявлення шахрайства з кредитними картками. Оскільки шахрайські транзакції є рідкісним явищем порівняно з автентичними, і методи, використовувані для шахрайства, постійно змінюються, системи виявлення аномалій є ефективним інструментом для розрізнення транзакцій, значення ознак яких істотно відрізняються від справжніх транзакцій [3].

1.2.Методи виявлення аномалій

Методи виявлення аномалій можна класифікувати за різними критеріями, такими як наявність учителя чи відсутність, тип навчального підходу, особливості вхідних даних та інші. Ось деякі загальні класифікації методів виявлення аномалій [4, 5]:

1. Методи, що базуються на статистичних моделях:
    - Метод гаусівського виявлення аномалій.
    - Метод машиною опорних векторів (Support Vector Machines - SVM) для виявлення аномалій.
2. Методи, що використовують групування (кластеризацію):

- Метод групування (кластеризації) k-means.

3. Методи, що використовують лише навчання на нормальних даних:
    - Метод однокрокових аномалій (One-Class SVM).
4. Методи, що використовують алгоритми машинного навчання:
    - Метод Isolation Forest.
5. Методи, що базуються на метриках відстані:
    - Метод з використанням метрик відстані (Distance-based methods).
6. Методи, що використовують глибоке навчання:
    - Метод з використанням моделей глибокого навчання (Deep Learning-based Anomaly Detection).

Ці класифікації не є взаємовиключними, і багато методів можуть мати елементи з кількох категорій. Вибір конкретного методу виявлення аномалій залежить від характеристик даних та вимог конкретного завдання.

Розглянемо деякі методи більш конкретно і виберемо метод, який найбільше підходить під наше завдання [6-11].

1.2.1. Метод гаусівського виявлення аномалій

Метод гаусівського виявлення аномалій (англ. Gaussian Anomaly Detection) - це метод машинного навчання, який використовує гаусівський розподіл (нормальний розподіл) для виявлення аномальних або несподіваних областей в даних. Основна ідея полягає в тому, що нормальні дані мають властивість бути сконцентрованими навколо середнього значення, тоді як аномалії зазвичай розташовані далеко від центральної маси даних.

Переваги:

1. Ефективність для нормально розподілених даних. Якщо дані мають нормальний розподіл, метод гаусівського виявлення аномалій може

бути ефективним, оскільки він базується на припущенні про гаусівський розподіл.

2. Простота реалізації. Метод гаусівського виявлення аномалій досить легко реалізується та розуміється. Він не вимагає великої кількості гіперпараметрів порівняно з іншими складнішими методами.

3. Працює при обмеженому об'ємі аномалій. Якщо аномалії становлять лише невеликий відсоток вибірки, метод гаусівського виявлення може добре працювати, оскільки він фокусується на знаходженні тих областей, які дещо відрізняються від нормального розподілу.

Недоліки:

1. Чутливість до розподілу даних. Якщо дані не мають нормального розподілу, метод може давати неточні результати. Він особливо неефективний у випадку, коли розподіл даних суттєво відрізняється від гаусівського.

2. Проблема збалансованості. Метод гаусівського виявлення припускає, що аномалії є рідкісними подіями. У випадку, коли аномалії становлять значну частину даних, він може недооцінювати їх.

3. Непередбачувані аномалії. Якщо аномалії мають нетипові форми або структури, які не можуть бути адекватно описані гаусівським розподілом, метод може несправедливо відхиляти їх.

4. Залежність від розмірності. При збільшенні кількості ознак метод може ставати менш ефективним через "прокляття розмірності" (curse of dimensionality) [6 - 8].

1.2.2. Метод групування (кластеризації) k-means

Метод групування (кластеризації) k-means - це алгоритм машинного навчання, призначений для розділення набору даних на групи (кластери) таким чином, щоб об'єкти всередині одного кластера були подібні між собою, а об'єкти з різних кластерів були різними.

Переваги:

1. Ефективність: K-means - це швидкий і ефективний алгоритм, особливо на великих наборах даних.
2. Простота реалізації: Легко реалізовується та зрозуміла концепція.
3. Ефективність для великих даних: Працює добре навіть на великих обсягах даних, особливо, якщо кількість кластерів не дуже велика.
4. Спроможність працювати з числовими даними: Якщо дані можна представити у вигляді числових ознак, то k-means може ефективно з ними працювати.

Недоліки:

1. Залежність від початкового вибору: Результати кластеризації можуть залежати від випадкового вибору початкових центрів кластерів. Це може привести до різних результатів при різних запусках алгоритму.
2. Неспроможність працювати з нелінійними формами кластерів: K-means припускає, що кластери мають форму круга або сфери, тому він може бути неефективним для деяких типів даних, особливо якщо кластери мають нелінійну форму.
3. Неспроможність працювати з різними розмірами та щільностями: K-means припускає, що всі кластери мають однакову щільність та діаметр. Це робить його менш ефективним для деяких типів даних.
4. Чутливість до викидів: Алгоритм чутливий до викидів, оскільки вони можуть значно впливати на розташування центрів кластерів.
5. Визначення кількості кластерів: Визначення оптимальної кількості кластерів може бути складною задачею і часто вимагає експертного втручання [9, 11].

1.2.3. Метод однокрокових аномалій (One-Class SVM)

Метод однокрокових аномалій — це метод машинного навчання, який використовується для виявлення аномалій у даних, коли навчальна вибірка містить тільки нормальні дані. Основна ідея полягає в тому, щоб визначити границі області, що містить нормальні дані, і визначити аномалії як ті точки, які виходять за ці межі.

Основні кроки роботи методу One-Class SVM:

1. Безкласове навчання. Модель One-Class SVM навчається тільки на нормальних даних, без використання аномальних даних. Мета полягає в тому, щоб створити границю, яка обмежить область, де повинні знаходитися нормальні дані.

2. Створення границі. SVM будує гіперплощину, яка максимально відокремлює нормальні дані від нульової точки (центру).

3. Виявлення аномалій. Точки, які опиняються по той бік від цієї границі, вважаються аномальними.

4. Контроль параметрів. Метод має гіперпараметри, такі як ширина ядра та найменший зазначений розмір розглядуваної області, які можна налаштовувати для досягнення оптимальної ефективності.

Переваги:

- Ефективний для виявлення аномалій у випадках, коли навчальний набір містить тільки нормальні дані.
- Здатний працювати у високорозмірних просторах.

Недоліки:

- Може бути важко визначити оптимальні гіперпараметри.
- Залежність від представлення навчальних даних та їхньої структури [9, 11].

1.2.4. Метод ізоляційного лісу (Isolation Forest)

Метод ізоляційного лісу (Isolation Forest) - це алгоритм машинного навчання для виявлення аномалій, який був представлений у статті "Isolation Forest" Лю Фей Тоні, Кай Тінг і Чжоу Жі-Хуа у 2008 році. Цей метод базується на ідеї, що аномальні екземпляри даних можуть бути легше виявлені через їх швидше видалення або "ізоляцію" в дереві при порівнянні з нормальними екземплярами.

Основні кроки роботи методу Isolation Forest включають:

1. Випадковий вибір ізоляційних точок. З об'єктів у вибірці випадковим чином обирається певна кількість точок.
2. Рекурсивне розділення даних. Для кожної ізоляційної точки проводиться випадковий атрибут і випадковий поріг, за яким дані розділяються. Об'єкти з меншою кількістю даних після розділення просуваються глибше в дерево.
3. Повторення процесу. Кроки 1-2 повторюються рекурсивно для кожного вузла, поки всі об'єкти не стануть ізольованими точками або досягнеться максимальна глибина дерева.
4. Оцінка аномалій. Для кожного об'єкта розраховується середня глибина, на яку його ізолювали в дереві. Аномальні об'єкти мають тенденцію потрапляти в більш поверхневі вузли, отже, вони матимуть меншу середню глибину.
5. Порівняння з порогом. Задається поріг, і об'єкти, середня глибина яких менше порога, визначаються як аномалії.

Переваги:

- Не базується на відстані, щільності або моделі.
- Низьке споживання процесорного часу та пам'яті.
- Ефективність у виявленні аномалій.
- Обробка великорозмірних даних.
- Виявлення відокремлених аномалій.

- Масштабованість.

Недоліки:

- Не ефективний для групових аномалій. Метод не завжди ефективний для виявлення аномалій, які утворюють групи або кластери, оскільки об'єкти в групах можуть мати велику середню глибину в дереві.
- Залежність від параметрів. Деякі параметри, такі як кількість дерев, максимальна глибина та поріг ізоляції, можуть впливати на результати, і їх слід правильно налаштовувати.
- Не ефективний для неперевних аномалій. Метод може виявляти відокремлені аномалії краще, ніж ті, що утворюють гладку поверхню [12].

1.3.Метрики якості

Метрики якості можна застосовувати для наступних цілей [14]:

1. Знаходження функціонала похибки (при навчанні).
2. Підбір гіперпараметрів (при вимірі якості на крос-валідації).
3. Оцінювання моделі: чи придатна вона для вирішення задачі.

Будь-який прогноз щодо бінарної категоріальної цільової змінної підпадає під одну з чотирьох категорій:

- **True Positive (TP):** Модель класифікації правильно прогнозує, що результат буде позитивним
- **True Negative (TN):** Модель класифікації правильно прогнозує негативний результат
- **False Positive (FP):** Модель класифікації неправильно прогнозує позитивний результат
- **False Negative (FN):** Модель класифікації неправильно прогнозує негативний результат

Нехай TP, TN, FP і FN відповідно позначають кількість істинно-позитивних, істинно-негативних, хибно-позитивних і хибно-негативних прогнозів, зроблених певною класифікаційною моделлю. Нижче наводено визначення деяких метрик оцінювання, заснованих на цих чотирьох величинах.

$$\text{Accuracy} = \frac{\text{Кількість правильних прогнозів}}{\text{Кількість загальних прогнозів}} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{\text{Кількість правдивих позитивних прогнозів}}{\text{Кількість загальних позитивних прогнозів}} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{\text{Кількість правдивих позитивних прогнозів}}{\text{Кількість загальних позитивних випадків}} = \frac{TP}{TP + FN}$$

$$\text{Індекс Фаулкса-Меллоуза (FM)} = \text{Середнє геометричне значення Precision and Recall} = \sqrt{\text{Precision} \times \text{Recall}}$$

$$F_1\text{-Score} = \text{Середнє гармонійне значення Precision and Recall} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$F_\beta\text{-score} = \frac{(1 + \beta^2) \times \text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}} = \frac{(1 + \beta^2) \times TP}{(1 + \beta^2) \times TP + \beta^2 \times FN + FP},$$

де β - позитивний фактор, вибраний таким чином, що Recall в β в рази важливіше за Precision. Найпоширеніші значення β - 0.5, 1 та 2.

$$\text{Коефіцієнт кореляції Метьюса (MCC)} = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}.$$

Рис. 1.1 – Визначення деяких метрик оцінювання, заснованих на величинах TP, TN, FP, FN

На відміну від попередніх метрик, MCC варіюється від −1 (найгірший сценарій) до 1 (найкращий сценарій: ідеальне передбачення). Серед обговорених метрик для оцінювання моделей, зокрема для незбалансованих наборів даних, добре підходять MCC та $F1$-score, тоді як Precision and Recall також дає корисну інформацію.

Ми не будемо надавати великого значення метриці Accuracy у цьому проекті, оскільки вона дає хибні висновки, коли класи не збалансовані. В даній проблемі хибнонегативний результат (шахрайська транзакція класифікується як автентична) є більш небезпечним, ніж хибнопозитивний (автентична транзакція класифікується як шахрайська). У першому випадку шахрай може завдати ще більших фінансових збитків. У другому випадку

банк може здійснити перехресну перевірку автентичності транзакції від користувача картки після вжиття необхідних заходів для захисту картки. Враховуючи цей факт, ми використовуємо $F2$-оцінку для налаштування порогового параметру та вибору функцій у цій роботі. У термінах **TP**, **TN**, **FP** та **FN** він має вигляд

$$F_2\text{-score} = \frac{5 \times TP}{5 \times TP + 4 \times FN + FP}.$$

Рис. 1.2 – Визначення $F2$-оцінки

1.3.1. Частка правильних відповідей (accuracy)

У задачах класифікації як міру якості беруть частку неправильних відповідей $\frac{1}{l}\sum_{i=1}^{l}[a(x_i) \neq y_i]$.

У випадку задач класифікації зазвичай обирають метрики з орієнтацією на максимізацію, в той час як у задачах регресії, навпаки, спрямовані на мінімізацію. Таким чином, цю характеристику часто формулюють за допомогою наступного виразу [14]:

$$accuracy(a, X) = \frac{1}{l}\sum_{i=1}^{l}[a(x_i) = y_i] \quad (1.1)$$

1.3.2. Матриця похибок. Точність і повнота

Використання матриці помилок (чи похибок) дозволяє зручно класифікувати різні сценарії, що відображають взаємозв'язок між виведеним алгоритмом результатом і правильною відповіддю (див. рис. 1.3) [11].

	$y = 1$	$y = -1$
$a(x) = 1$	True Positive (TP)	False Positive (FP)
$a(x) = -1$	False Negative (FN)	True Negative (TN)

Рис. 1.3 – Матриця похибок

Коли алгоритм призначає об'єкт класу +1, це вказує на його успішне виконання. Якщо модель правильно визначила клас об'єкта, то це визначається як вірне позитивне спрацювання (TP); проте, якщо об'єкт насправді належить до іншого класу, то це вважається хибним позитивним спрацюванням (FP).

Якщо алгоритм повертає відповідь –1, це свідчить про пропуск об'єкта. Якщо при цьому відбувається пропуск об'єкта класу +1, це називається хибним пропуском (FN); водночас, якщо модель пропускає об'єкт класу –1, це рахується як істинний пропуск (TN).

Отже, можна виділити два види похибок: хибні позитивні спрацювання та хибні пропуски.

Розглянемо далі дві наступні метрики. Точність (precision) - це характеристика, що вказує, наскільки можна довіряти класифікатору, коли він робить позитивне спрацювання:

$$\text{precision}(a, X) = \frac{TP}{TP + FP} \tag{1.2}$$

Повнота (recall) – інша характеристика, яка вказує на те, на якій частці істинних об’єктів першого класу алгоритм виявляє їх спрацювання:

$$\text{recall}(a, X) = \frac{TP}{TP + FN} \tag{1.3}$$

1.3.3. F-міра

Для більш показових результатів використаємо гармонійне середнє, або F-міру [11]:

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{1.4}$$

Якщо виникає необхідність вибору між точністю та повнотою, розглядається розширена F-міра, у якої присутній параметр β:

$$F = (1 + \beta^2) \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \tag{1.5}$$

Наприклад, якщо встановити значення параметра β=0.5, то акцент буде зроблено на важливості повноти, а при встановленні β=2 – на точності.

1.3.4. PR-крива

Давайте розглянемо криву точності-повноти (PR-криву), яка використовується для оцінки віднесення об'єкта до певного класу. Повнота відображається на горизонтальній вісі, тоді як точність - на вертикальній вісі. Кожна точка на цій кривій відповідає конкретному класифікатору зі своїм власним значенням порогу. У випадках, де кількість об'єктів сягає кількох тисяч або більше, вигляд цієї кривої може бути подано так, як зображено на рис. 1.4:

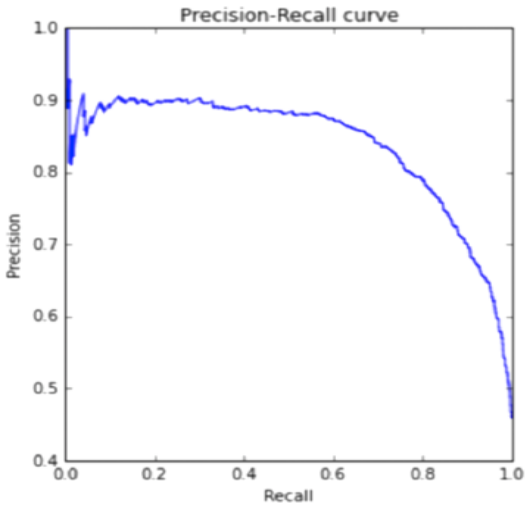


Рис. 1.4 – PR-крива

Необхідно відзначити, що PR-крива розпочинається з точки (0; 0) і завершується точкою (1; r), де r - частка об'єктів першого класу. У випадку ідеального класифікатора, коли існує значення порогу, при якому як точність, так і повнота дорівнюють 100%, крива проходить через точку (1; 1). Таким чином, чим кращі оцінки, тим ближче крива досягає цієї точки. Площа під цією кривою є мірою якості оцінок належності до першого класу. Ця метрика відома як площа під PR-кривою або AUC-PRC.

1.3.5. ROC-крива

ROC-крива - інший метод визначення якості оцінок належності до першого класу. Вона формується на вісях False Positive Rate (ось x) і True Positive Rate (ось y), обчислення яких можна виразити так:

$$FPR = \frac{FP}{FP + TN}, \quad TPR = \frac{TP}{TP + FN} \tag{1.6}$$

Якщо вибірка є великою, тоді ROC-крива виглядає так (рис. 1.5).

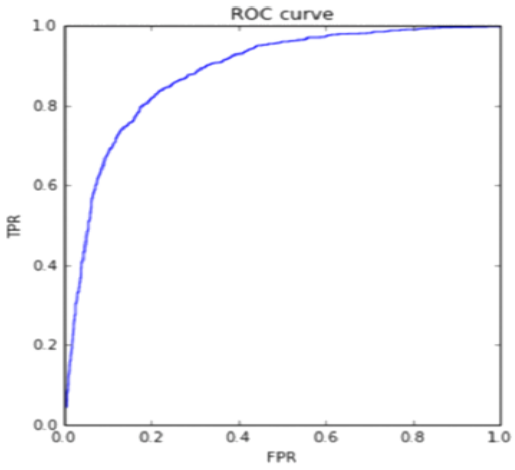


Рис. 1.5 – Крива ROC в реальних задачах з десятками тисяч об’єктів

Крива починається з точки $(0; 0)$ і приходить в точку $(1; 1)$. Водночас крива повинна пройти через точку $(0; 1)$, якщо існує ідеальний класифікатор. Чим краще будуть оцінки, тим ближче крива буде до цієї точки, а площа під кривою буде характеризувати якість оцінок приналежності до першого класу. Ця метрика має назву площа під ROC-кривою, або AUC-ROC.

РОЗДІЛ 2
МАТЕМАТИЧНА МОДЕЛЬ ВИЯВЛЕННЯ АНОМАЛІЙ

2.1. Метод гаусівського виявлення аномалій

Багатовимірний гаусівський розподіл використовується для виявлення аномалій, щоб знайти ймовірність кожного прикладу і на основі деякого порогового значення вирішити, чи потрібно позначати аномалію чи ні. Вираз для обчислення параметрів гаусівської моделі - це μ і σ, де μ - це середнє значення кожної ознаки, а σ обчислює коваріаційну матрицю. Ці два параметри використовуються для обчислення ймовірності p(x).

Цей метод використовується для моделювання спільного розподілу n-вимірних випадкових величин. Це узагальнення одновимірного нормального розподілу на більш високі виміри. При виявленні аномалій він використовується для моделювання розподілу нормальних даних, а потім обчислення ймовірності появи нових точок даних. Якщо ймовірність нижче певного порогу, точка даних позначається як аномалія [6-8].

2.1.1. Загальна постановка задачі

Дані:

Маємо навчальний набір даних $X_{train} = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$, де $x^{(i)}$ - вектор ознак для *i*-го прикладу.

Модель:

Модель базується на припущенні, що нормальні події утворюють багатовимірний нормальний розподіл. Тобто, ознаки вважаються нормально розподіленими, і ми використовуємо їх для створення моделі щільності розподілу ймовірностей $p(X)$.

Параметри моделі:

Параметри моделі - це середнє значення μ і коваріаційна матриця Σ. Ці параметри оцінюються на основі навчальної вибірки.

Функція щільності розподілу:

Для багатовимірного нормального розподілу, функція щільності розподілу ймовірностей виглядає як [6-8]:

$$p(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(X-\mu)^T \Sigma^{-1}(X-\mu)} \quad (2.1)$$

Тут:

- $p(X)$ - це ймовірність події X, де X - вектор ознак;
- n - кількість ознак (розмірність вектору ознак);
- Σ - коваріаційна матриця;

Коваріаційна матриця - це матриця, яка містить коваріації між парами ознак у наборі даних. Коваріація вимірює ступінь, до якої дві ознаки змінюються разом. Коли коваріація позитивна, це означає, що збільшення однієї ознаки пов'язано із збільшенням іншої. Якщо коваріація від'ємна, то збільшення однієї ознаки пов'язане із зменшенням іншої. Якщо коваріація дорівнює нулю, це вказує на відсутність лінійної залежності між ознаками.

Коваріаційна матриця Σ для набору даних X розмірності $m \times n$ (де m - кількість спостережень, n - кількість ознак) обчислюється наступним чином:

$$\Sigma = \frac{1}{m-1} \sum_{i=1}^{m} (X_i - \bar{X})^T (X_i - \bar{X}) \quad (2.2)$$

Тут:

- X_i - вектор ознак для і-го спостереження;
- $\bar{X}$ - вектор середніх значень ознак по всьому набору даних;

- $(X_i - \bar{X})^T$ - транспонований вектор відхилень від середніх для і-го спостереження.

Отримана матриця містить коваріації між усіма парами ознак. Якщо дані стандартизовані (мають одинакові середні та стандартні відхилення), то коваріаційна матриця також буде матрицею кореляцій, де кожен елемент - це кореляція між відповідними ознаками.

- μ - вектор середніх значень.

Функція щільності розподілу виражає теоретичну ймовірність отримання конкретного вектора ознак X за умови, що дані розподілені за багатовимірним нормальним розподілом. При виявленні аномалій, ми порівнюємо значення цієї ймовірності з певним пороговим значенням: якщо вона менша за поріг, то приклад вважається аномальними.

Оцінка параметрів:

Середнє значення μ і коваріаційна матриця Σ обчислюються на основі навчальної вибірки.

Виявлення аномалій:

Порівнюється значення щільності розподілу $p(X)$ для кожного прикладу з певним пороговим значенням. Приклади, для яких $p(X)$ менше порогового значення, вважаються аномальними [6-8].

2.1.2. Детальний розбір функцій

Одновимірний нормальний розподіл:

Щільність розподілу ймовірностей одновимірного нормального розподілу із середнім μ та стандартним відхиленням σ має вигляд [6, 7]

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (2.3)$$

де:

- x - вхідне спостереження,
- μ - середнє значення, яке визначається за формулою

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (2.4)$$

де

- n - кількість спостережень,
- x_i - значення ознаки (спостереження) i;

- σ - стандартне відхилення, яке визначається за формулою

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}} \quad (2.5)$$

Функція щільності розподілу ймовірностей визначає ймовірність того, що випадкова змінна x матиме значення близьке до μ.

Добуток нормальних функцій щільності ймовірності:

Функція, яка обчислює добуток щільностей розподілу для набору змінних ознак:

$$f(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} f(x_i) \quad (2.6)$$

де:

- *xi* - значення ознаки *i*,
- *n* - кількість ознак.

Ця функція враховує статистичну незалежність між ознаками.

Модель виявлення аномалій:

Модель виявлення аномалій порівнює отриманий добуток щільностей розподілу для конкретного спостереження з певним пороговим значенням щільності. Якщо отримана щільність менше порогу, спостереження вважається аномальним.

Налаштування порогового значення щільності:

Параметр α використовується як порогове значення щільності, і виконується оптимізація його значення за допомогою F2-оцінки.

Модель виявлення аномалій буде визначати аномалії на основі порівняння отриманої щільності з встановленим пороговим значенням [7].

2.2. Метод групування (кластеризації) k-середніх (k-means)

Метод групування, наприклад, k-середніх (k-means), використовує математичну модель для розділення даних на кластери. В основі цього методу лежить метричний принцип, який мінімізує суму квадратів відстаней між кожною точкою та центроїдом свого кластера [9, 11].

2.2.1. Загальна постановка задачі

Нехай маємо множину X з n точок даних у d-вимірному просторі:

$$X = \{x_1, x_2, \dots, x_n\}, x_i \in R^d \quad (2.7)$$

Метою є розбити дані на K кластерів (де K - задане число) так, щоб мінімізувати суму квадратів відстаней в кожному кластері. Кожен кластер характеризується своїм центроїдом.

Формально, постановка задачі виглядає наступним чином:

I. Центроїди:

$$C = \{C_1, C_2, \dots, C_n\}, C_i \in R^d \quad (2.8)$$

II. Призначення кластерів:

Маємо функцію відстані, наприклад, евклідову відстань $\|x - Ck\|$. Кожна точка x_j призначається кластеру i, якщо:

$$i = argmin_k \left\|x_j - C_k\right\|^2 \quad (2.9)$$

III. Мінімізація суми квадратів відстаней:

Мінімізуємо суму квадратів відстаней для всіх точок та кластерів:

$$J(C) = \sum_{i=1}^{K} \sum_{j=1}^{n} \left\|x_j - C_i\right\|^2 \quad (2.10)$$

IV. Оновлення центроїдів:

Оновлюємо центроїди для кожного кластера, використовуючи середнє арифметичне точок у кластері:

$$C_i = \frac{1}{|S_i|} \sum_{j \epsilon S_i} x_j \quad (2.11)$$

V. Повторюємо призначення кластерів та оновлення центроїдів до досягнення збіжності або до досягнення максимальної кількості ітерацій.

Задача полягає в пошуку оптимальних значень для центроїдів, що мінімізують функціонал відстані $J(C)$.

2.3. Метод однокрокових аномалій (One-Class SVM)

Метод однокрокових аномалій (One-Class SVM) є алгоритмом машинного навчання, який використовується для виявлення аномалій або викидів в даних. Цей метод працює в умовах навчання без учителя і в основному базується на навчанні моделі лише на нормальних даних [9, 11].

2.3.1. Загальна постановка задачі

Дано:

- $X \in R^{m \times n}$: Навчальний набір даних, що містить тільки "нормальні" приклади.
- $\phi(x)$: Функція відображення в простір вищої розмірності (зазвичай нелінійна функція).
- w: Ваговий вектор, який характеризує нормальний простір.
- b: Зсув (bias), який також характеризує нормальний простір.

Мета:

- Максимізувати розмежувальну відстань між "нормальними" прикладами і гіперплощиною, що їх розділяє.

У моделі One-Class SVM використовується гіперплощина у просторі вищої розмірності, яка розділяє "нормальні" дані від потенційних аномалій. Функція рішення $R(x)$ для нового прикладу x визначається наступним чином:

$$R(x) = sgn(\langle w, \phi(x) \rangle - b) \tag{2.12}$$

де:

- $\langle \cdot, \cdot \rangle$: Скалярний добуток.
- sgn(·): Функція знаку (sgn(z)=1, якщо z>0, sgn(z)=−1, якщо z<0, sgn(z)=0, якщо z=0).

Мета навчання One-Class SVM полягає у знаходженні параметрів w і b, які максимізують розмежувальну відстань між "нормальними" прикладами і гіперплощиною. Після тренування, модель може використовуватися для визначення того, чи є нові екземпляри аномальними чи нормальними.

2.4. Метод ізоляційного лісу (Isolation Forest)

Isolation Forest - це алгоритм для виявлення аномалій у наборі даних. Він використовує метод дерева рішень для ізоляції аномалій шляхом випадкового розбиття даних і розділення їх на частини, намагаючись виділити аномалії, які повинні бути меншими та меншими групами. Аномальні екземпляри будуть швидше відокремлені, тобто вони зазвичай

потраплять в різні гілки дерева при меншій глибині порівняно з нормальними даними.

2.4.1. Загальна постановка задачі

Дано:

Набір даних, який складається з нормальних та можливих аномальних зразків.

Задача:

Розробити модель, яка здатна визначити, які зразки є аномалійними, використовуючи метод ізоляційного лісу.

Основні кроки вирішення задачі включають побудову дерева (або кількох дерев), визначення глибини ізоляції для кожного зразка та порівняння цих значень для прийняття рішення щодо того, чи є зразок аномальним. Метод використовує ефект, що аномальні точки швидше ізолюються в дереві порівняно з нормальними точками.

Метод ізоляційного лісу (Isolation Forest) використовує ансамбль дерев для виявлення аномалій у наборі даних. Основні формули та кроки методу можна виразити наступним чином:

1. Побудова ізоляційного дерева (Isolation Tree).

- Випадковий вибір ознаки та порогу. Вибір ознаки та порогу здійснюється випадковим чином для кожного вузла дерева.

- Рекурсивний розподіл даних. Область даних розбивається на дві частини за вибраною ознакою та порогом. Процес розбиття повторюється рекурсивно для обох частин досягнення максимальної глибини або до того моменту, коли всі дані відокремлюються.

- Оцінка глибини ізоляції. Глибина ізоляції вимірюється шляхом підрахунку, скільки розділень (рівнів у дереві) потрібно, щоб ізолювати зразок. Чим менше глибина ізоляції, тим більше ймовірність того, що зразок є аномалією.

2. Визначення аномалій.

- Середня глибина ізоляції. Для кожного зразка розраховується глибина ізоляції. Середня глибина ізоляції обчислюється як середнє значення глибини для всіх зразків.

- Оцінка аномалії. За допомогою середньої глибини ізоляції та константи c, яка визначається середньою глибиною бінарного дерева, розраховується оцінка аномалії $s(x,n)$. Ця оцінка вказує, наскільки зразок відрізняється від середнього.

3. Порівняння з порогом.

- Класифікація: Зразки класифікуються як аномальні чи нормальні залежно від того, чи перевищує оцінка аномалії встановлений поріг.

Глибина ізоляції (Path Length):

$$h(x) = -\frac{2}{c}\log\left(\frac{l(x)}{n}\right) \tag{2.13}$$

де $l(x)$ - глибина ізоляції зразка x, n - середня глибина ізоляції для всіх зразків, c - середня глибина бінарного дерева.

Оцінка аномалії:

$$s(x,n) = 2^{-\frac{E(h(x))}{c}} \quad (2.14)$$

де $E(h(x))$ - середнє значення глибини ізоляції для всіх зразків, c - середня глибина бінарного дерева.

Мета - визначити значення аномалій $s(x,n)$ та поріг для класифікації зразків на аномальні та нормальні [12].

РОЗДІЛ 3
ПРОВЕДЕННЯ КОМП'ЮТЕРНОГО ЕКСПЕРИМЕНТУ

Для вирішення поставленого завдання використовувався пакет програм Python.

3.1. Опис набору даних

Набір даних містить інформацію про транзакції, здійснені за допомогою кредитних карток європейськими власниками карток за два конкретні дні вересня 2013 року. У ньому представлено транзакцій на загальну суму 284807, з яких 492 були шахрайськими. Очевидно, що набір даних дуже незбалансований, позитивний клас (шахрайські транзакції) становить лише 0,173% від усіх транзакцій. Стовпці в наборі даних виглядають наступним чином:

- Тривалість. кількість секунд, що минула між транзакцією та найпершою транзакцією.
- V1 - V28. Отримано в результаті перетворення за допомогою аналізу головних компонент (PCA) вихідних даних, які не доступні з міркувань конфіденційності.
- Сума. Сума транзакції.
- Клас. Статус транзакції щодо автентичності. Клас автентичної (або шахрайської) транзакції приймається за 0 (або 1) [15].

3.2. Підготовка даних

Підготовка даних - це процес обробки та організації вхідних даних перед їхнім використанням для навчання моделей машинного навчання або проведення аналізу. Цей процес може включати в себе кілька етапів:

1. Очищення даних: Вилучення аномальних або непотрібних значень, вирішення пропущених даних та виправлення помилок у даних.
2. Вибір функцій (ознак): Вибір найважливіших функцій для аналізу або моделювання. Це може включати в себе відкидання зайвих функцій або вибір тих, які найкраще впливають на цільовий показник.
3. Масштабування і нормалізація: Приведення всіх функцій до одного масштабу для забезпечення стабільності та ефективності алгоритмів машинного навчання.
4. Розділення даних: Розділення набору даних на тренувальні, валідаційні та тестові набори для навчання, налаштування та оцінювання моделі.
5. Обробка категоріальних даних: Конвертування категоріальних змінних у формат, зручний для обробки алгоритмами машинного навчання.
6. Обробка дисбалансу класів: У випадках, коли класи в наборі даних є незбалансованими, можуть застосовуватися методи для балансування класів, такі як oversampling або undersampling.

Підготовка даних важлива для успішного навчання моделей та отримання надійних результатів при аналізі даних. Вона допомагає уникнути проблем, таких як перенавчання, покращує витрати часу та ресурсів, і допомагає зробити моделі більш інтерпретованими та загальними.

У наступній частині коду, що зображена на рис. 3.1, ми беремо спочатку і розбиваємо дані за цільовими класами, тобто належністю даних до звичайних транзакцій, а які до шахрайських.

Далі розбиваємо за цільовою функцією на незалежні та залежну змінні.

Наступним кроком готуємо навчальні вибірки для автентичного та шахрайського класів. Також об’єднуємо дані для побудови валідаційної та тестової вибірки [16].

Ввод [4]:

```
# Розбиття даних за цільовими класами
data_0, data_1 = data[data['Class'] == 0], data[data['Class'] == 1]

# Розбиття за функцією-ціллю
X_0, y_0 = data_0.drop('Class', axis = 1), data_0['Class']
X_1, y_1 = data_1.drop('Class', axis = 1), data_1['Class']

# Розбиття автентичного класу та побудова навчальної вибірки
X_train, X_test, y_train, y_test = train_test_split(X_0, y_0, test_size = 0.2, random_state = 40)
X_val, X_test, y_val, y_test = train_test_split(X_test, y_test, test_size = 0.5, random_state = 40)
data_val_1, data_test_1 = pd.concat([X_val, y_val], axis = 1), pd.concat([X_test, y_test], axis = 1)

# Розділення шахрайського класу
X_val, X_test, y_val, y_test = train_test_split(X_1, y_1, test_size = 0.5, random_state = 40)
data_val_2, data_test_2 = pd.concat([X_val, y_val], axis = 1), pd.concat([X_test, y_test], axis = 1)

# Об'єднання даних для побудови валідаційної та тестової вибірки
data_val, data_test = pd.concat([data_val_1, data_val_2], axis = 0), pd.concat([data_test_1, data_test_2], axis = 0)
X_val, y_val = data_val.drop('Class', axis = 1), data_val['Class']
X_test, y_test = data_test.drop('Class', axis = 1), data_test['Class']
```

Рис. 3.1 – Тренувально-валідаційно-тестовий поділ даних

Далі зобразимо графічно розподіл автентичних та шахрайських транзакцій між навчальним, валідаційним та тестовим набором (див. рис. 3.3).

Ввод [5]:

```
# Розподіл автентичних та шахрайських транзакцій між навчальним, валідаційним та тестовим набором
labels = ['Train', 'Validation', 'Test']
values_0 = [len(y_train[y_train == 0]), len(y_val[y_val == 0]), len(y_test[y_test == 0])]
values_1 = [len(y_train[y_train == 1]), len(y_val[y_val == 1]), len(y_test[y_test == 1])]
fig = make_subplots(rows = 1, cols = 2, specs = [[{'type': 'domain'}, {'type': 'domain'}]])
fig.add_trace(go.Pie(values = values_0, labels = labels, hole = 0.5, textinfo = 'percent', title = "Authentic"),
              row = 1, col = 1)
fig.add_trace(go.Pie(values = values_1, labels = labels, hole = 0.5, textinfo = 'percent', title = "Fraudulent"),
              row = 1, col = 2)
text_title = "Розподіл автентичних та шахрайських транзакцій між навч., валід. та тест. набором"
fig.update_layout(height = 500, width = 800, showlegend = True, title = dict(text = text_title, x = 0.5, y = 0.95))
fig.show()
```

Рис. 3.2 – Розподіл автентичних та шахрайських транзакцій між навчальним, валідаційним та тестовим набором

Далі час для наступного кроку – налаштування ознак. Почнемо з розгляду ознаки часу. Розкладемо час на дні. години, хвилини та секунди.

Налаштування ознак

Час

Ввод [7]:

```
# Час розкладання
for df in [X_train, X_val, X_test]:
    df['Day'], temp = df['Time'] // (24*60*60), df['Time'] % (24*60*60)
    df['Hour'], temp = temp // (60*60), temp % (60*60)
    df['Minute'], df['Second'] = temp // 60, temp % 60
X_train[['Time', 'Day', 'Hour', 'Minute', 'Second']].head()
```

Out[7]:

	Time	Day	Hour	Minute	Second
19594	30401.0	0.0	8.0	26.0	41.0
124712	77397.0	0.0	21.0	29.0	57.0
167920	118964.0	1.0	9.0	2.0	44.0
47377	43191.0	0.0	11.0	59.0	51.0
41731	40804.0	0.0	11.0	20.0	4.0

Рис. 3.3 – Налаштування ознак (час)

Далі порівняємо візуально гістограми часу та годин. Отже, з графіку (див. рис. 3.4) видно, що краще взяти замість ознаки часу – години.

Ввод [8]:

```
# Візуалізація
fig, ax = plt.subplots(1, 2, figsize = (15, 6), sharey = False)
sns.histplot(data = X_train, x = 'Time', bins = bins_train, ax = ax[0])
sns.histplot(data = X_train, x = 'Hour', bins = 24, ax = ax[1])
ax[1].set_ylabel(" ")
plt.suptitle("Гістограми часу та години", size = 14)
plt.tight_layout()
plt.show()
```

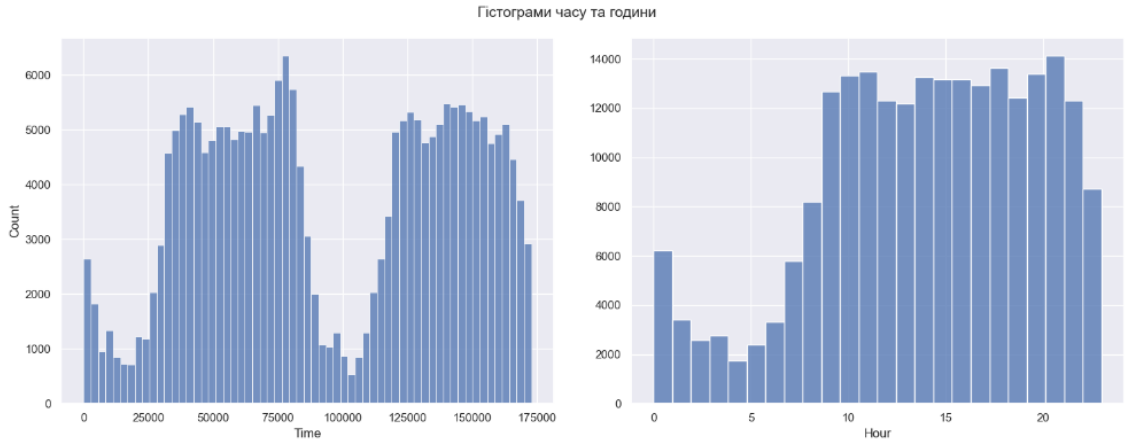


Рис. 3.4 – Візуалізація налаштування ознак (час)

Далі розглянемо ознаку Суми (або кількості, або Amount). Розподіл Amount має екстремально позитивну асиметрію. Ми застосуємо перетворення $x \mapsto \log(x+0.001)$ до цього стовпчика і сформуємо новий

стовпчик Amount_transformed. Додатну константу 0.001 додано, щоб врахувати транзакції з нульовою сумою, що призводить до log0, невизначеної величини. Також зобразимо гістограму нетрансформованої та трансформованої ознаки (див. рис. 3.5).

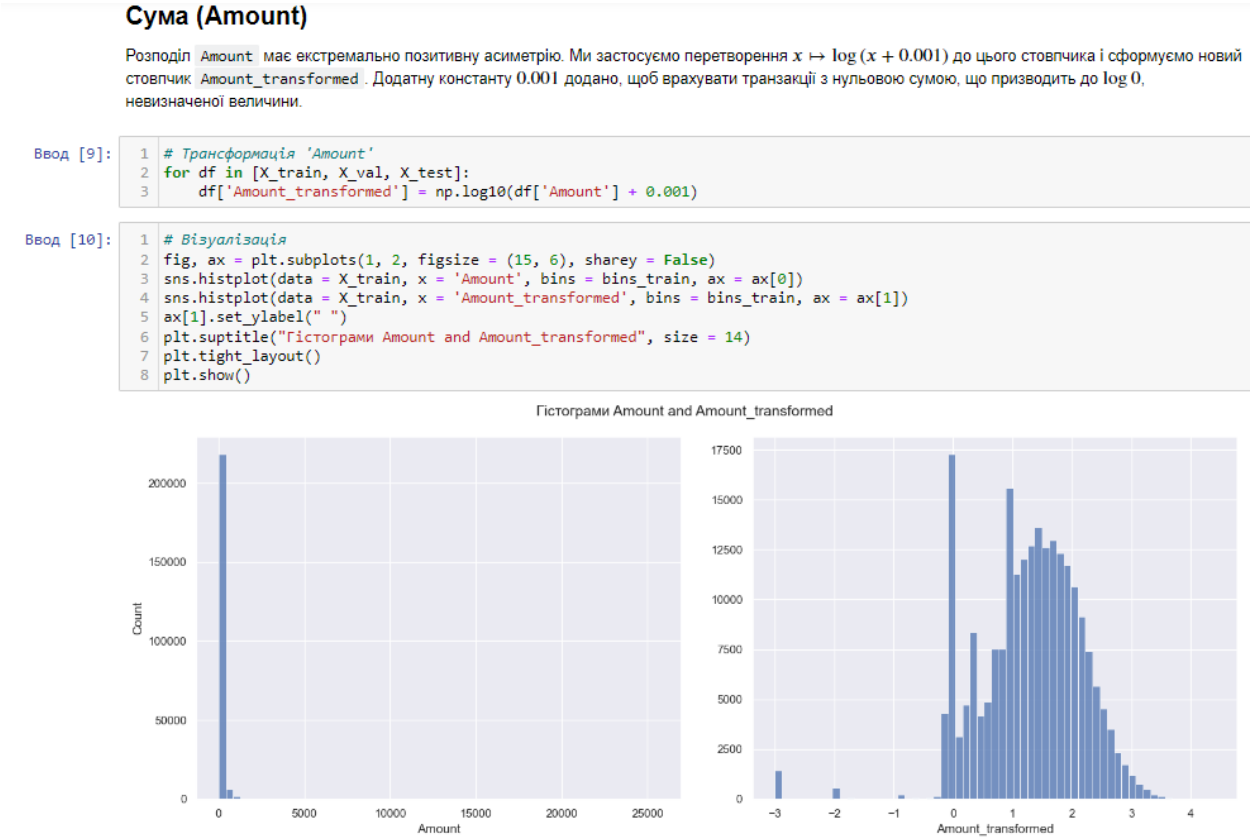


Рис. 3.5 – Налаштування ознак (сума)

Наступним кроком видалимо непотрібні стовпці (див. рис. 3.6).

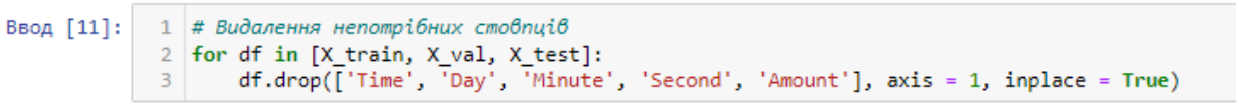


Рис. 3.6 – Видалення непотрібних ознак (стовпців)

Далі йде процес вибору ознак.

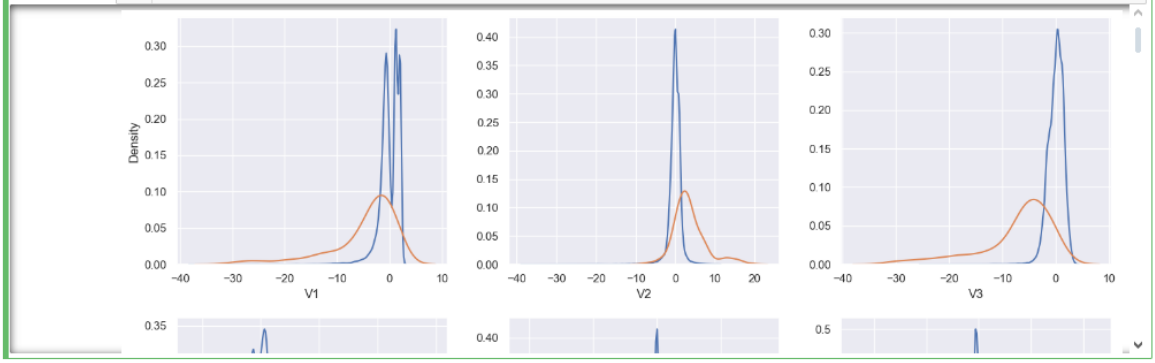
Висока розмірність створює труднощі для виявлення аномалій, оскільки зі збільшенням кількості атрибутів або ознак зростає і кількість даних, необхідних для точного узагальнення, що призводить до розрідженості даних, коли точки даних стають більш розкиданими та ізольованими. Ця розрідженість даних зумовлена непотрібними змінними

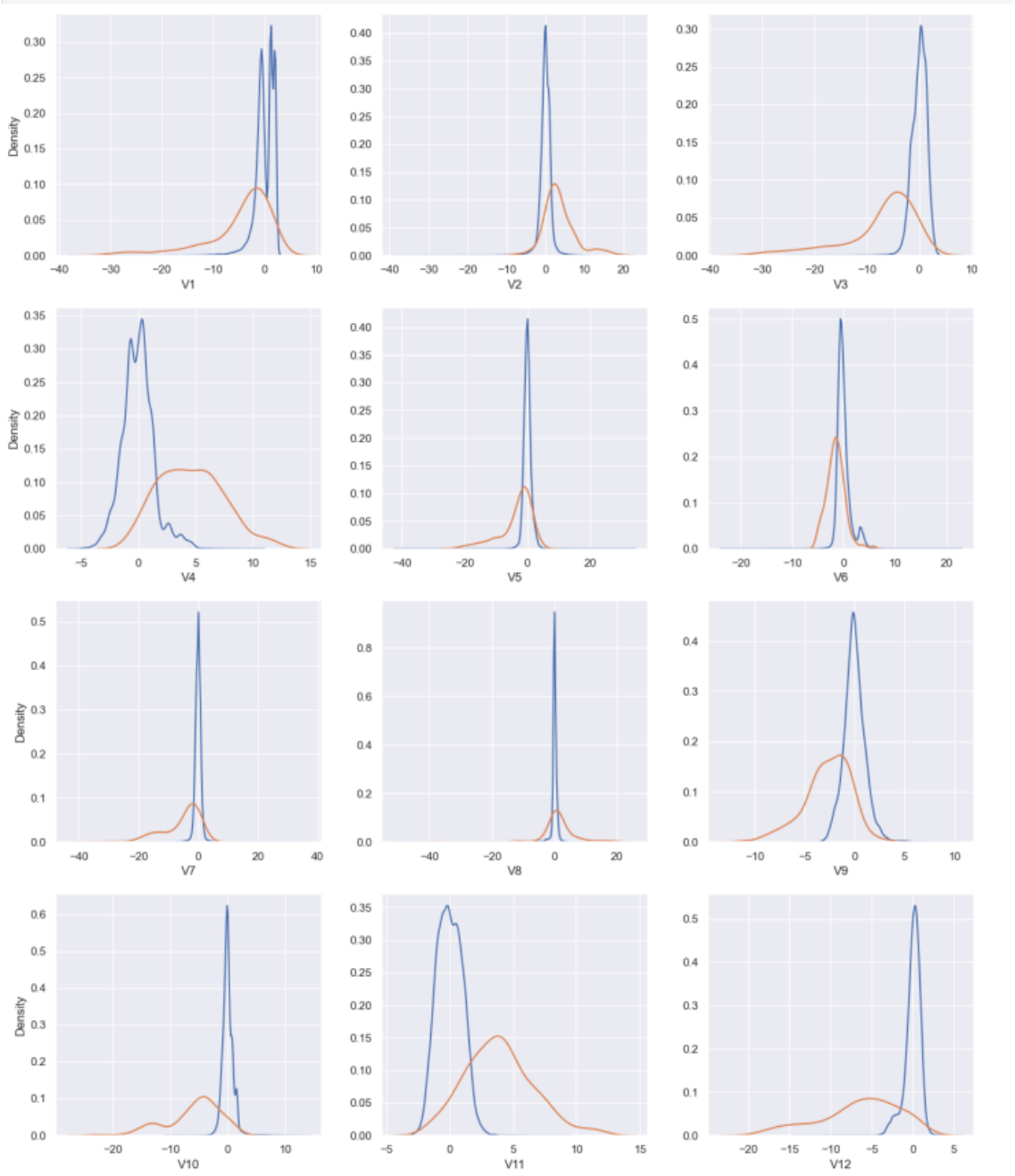
або високим рівнем шуму від численних нерелевантних атрибутів, які приховують справжні аномалії. Ця проблема широко відома як прокляття розмірності.

У задачі, яку ми розглядаємо, ми маємо 30 ознак. Ми прагнемо залишити лише ті з них, які суттєво допомагають розрізняти автентичні та шахрайські транзакції. Зокрема, ми порівнюємо розподіл кожної ознаки для обох цільових класів. Якщо ознака має схожий розподіл для автентичних і шахрайських транзакцій, то вона, швидше за все, не зробить значного внеску в процес класифікації транзакції як *автентичної* або *шахрайської*. Однак, якщо ознака має дуже різний розподіл для різних цільових класів, то вона відіграє набагато важливішу роль у тому ж процесі. Ми будуємо графіки розподілів і вибираємо ознаки, які мають досить чіткий розподіл між цільовими класами (див. рис. 3.7).

Ввод [12]:

```
# Порівняння розподілів ознак для різних цільових класів
data_val = pd.concat([X_val, y_val], axis = 1)
data_val_0, data_val_1 = data_val[data_val['Class'] == 0], data_val[data_val['Class'] == 1]
cols, ncols = list(X_val.columns), 3
nrows = math.ceil(len(cols) / ncols)
fig, ax = plt.subplots(nrows, ncols, figsize = (4.5 * ncols, 4 * nrows))
for i in range(len(cols)):
    sns.kdeplot(data_val_0[cols[i]], ax = ax[i // ncols, i % ncols])
    sns.kdeplot(data_val_1[cols[i]], ax = ax[i // ncols, i % ncols])
    if i % ncols != 0:
        ax[i // ncols, i % ncols].set_ylabel(" ")
plt.tight_layout()
plt.show()
```




Density
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

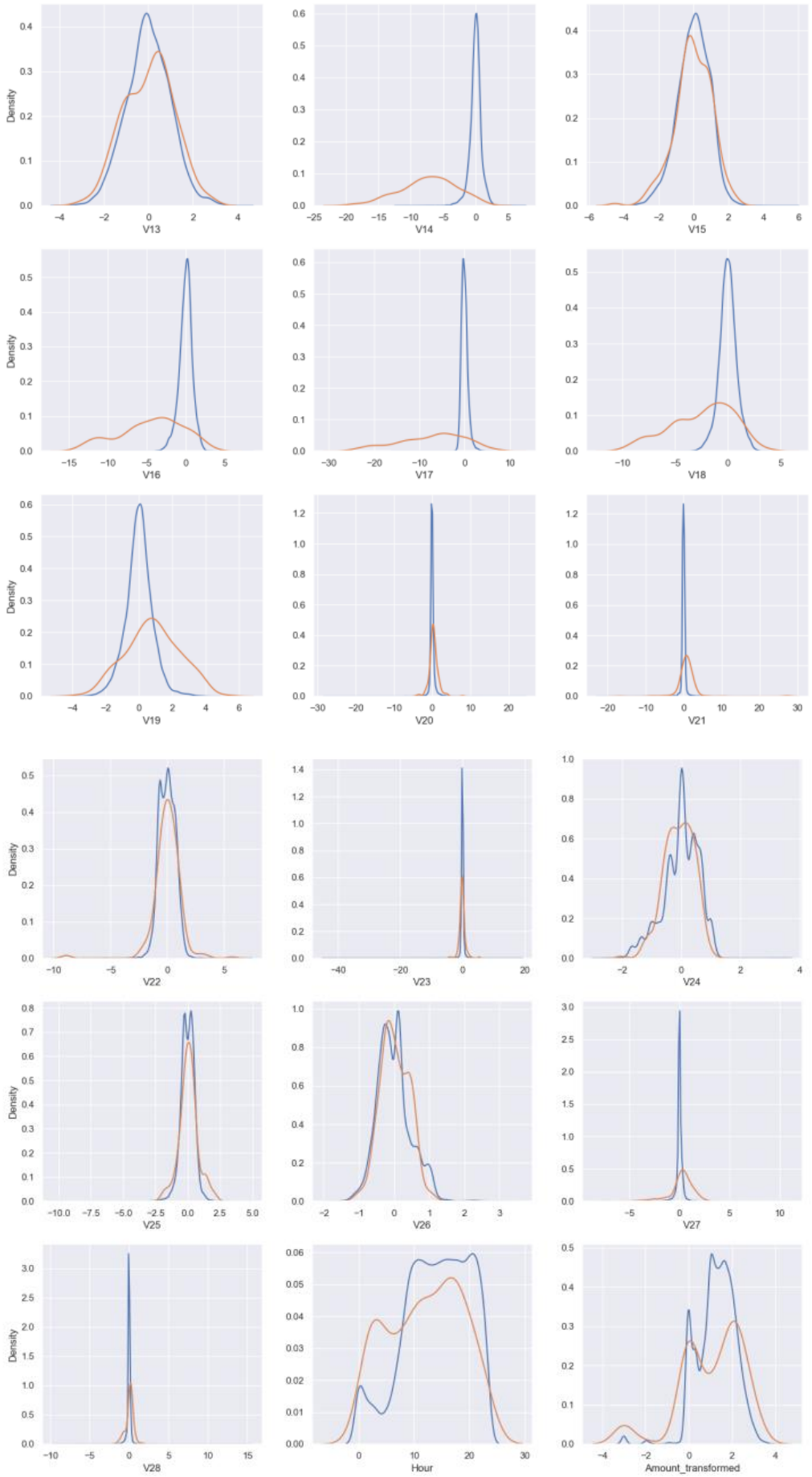


Рис. 3.7 – Порівняння розподілів ознак для різних цільових класів

Отже, завдяки графікам розподілам ознак ми обираємо з 30 ознак 9 ознак (див. рис. 3.8).

Ввод [13]:
```
# Вибір ознак
cols = ['V4', 'V11', 'V12', 'V14', 'V16', 'V17', 'V18', 'V19', 'Hour']
X_train_fs, X_val_fs, X_test_fs = X_train[cols], X_val[cols], X_test[cols]
X_train_fs.head()
```

Out[13]:

	V4	V11	V12	V14	V16	V17	V18	V19	Hour
19594	-0.706232	2.027925	0.535822	0.250769	0.773615	0.449717	-1.963208	0.613481	8.0
124712	1.474933	-1.154523	0.263527	0.316174	-1.029415	1.030772	-0.438839	0.529080	21.0
167920	4.840766	-2.242431	0.034829	-0.546349	-0.070375	1.033695	0.531801	1.215045	9.0
47377	0.565273	-0.157045	-0.548790	0.419194	0.183518	-0.681323	0.911357	1.318132	11.0
41731	-0.428860	-0.580964	-0.609099	-0.187948	1.226723	0.104368	-0.995711	0.420557	11.0

Рис. 3.8 – Вибір важливих ознак

Далі переходимо до процесу впровадження виявлення аномалій. Розглянемо метод заснований на щільності розподілу ймовірностей.

Щільність розподілу ймовірностей одновимірного нормального розподілу із середнім μ та стандартним відхиленням σ має вигляд

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right), \quad x \in \mathbb{R}; \ \mu \in \mathbb{R}, \ \sigma > 0.$$

Рис. 3.9 – Визначення функції щільності ймовірності

Тоді запишемо визначення цієї функції на мові Python (див. рис. 3.10).

Ввод [14]:
```
# Звичайна функція щільності ймовірності
def normal_density(x, mu, sigma):
    """
    Обчислює одновимірну нормальну щільність розподілу ймовірностей (pdf) із середнім значенням mu, стандартним відхиленням
    Аргументи:
      x (скаляр) : вхідне спостереження
      mu (скаляр): середнє значення
      sigma (скаляр): стандартне відхилення (> 0)
    Повертає
      f (скаляр): значення одновимірного нормального pdf
    """
    assert sigma > 0, "Середньоквадратичне відхилення має бути позитивним"
    f = (1 / (sigma * np.sqrt(2 * np.pi))) * np.exp(- (1 / 2) * ((x - mu) / sigma)**2)
    return f
```

Рис. 3.10 – Визначення звичайної функції щільності ймовірності

Наступна функція обчислює добуток таких одновимірних нормальних щільностей. Це можна розглядати як об'єднану щільність розподілу ймовірностей ряду змінних ознак, кожна з яких має одновимірний

нормальний розподіл і є статистично незалежною від інших ознак (див. рис. 3.11).

Ввод [15]:

```python
# Добуток нормальних функцій щільності ймовірності
def normal_product(x_vec, mu_vec, sigma_vec):
    """
    Обчислює добуток одновимірних нормальних густин
    Аргументи:
      x_vec (array_like, shape (n,)) : вектор вхідних спостережень
      mu_vec (array_like, shape (n,)) : вектор середніх
      sigma_vec (array_like, shape (n,)): вектор стандартних відхилень (> 0)
    Повертає
      f (скаляр): добуток одновимірних нормальних щільностей
    """
    assert min(sigma_vec) > 0, "Середньоквадратичне відхилення має бути позитивним"
    assert len(mu_vec) == len(x_vec), "Довжина середнього вектора не відповідає довжині вхідного вектора"
    assert len(sigma_vec) == len(x_vec), "Довжина вектора середньоквадратичного відхилення не відповідає довжині вхідного ве
    f = 1
    for i in range(len(x_vec)):
        f = f * normal_density(x_vec[i], mu_vec[i], sigma_vec[i])
    return f
```

Рис. 3.11 – Визначення добутку нормальних функцій щільності ймовірності

Далі ми обчислюємо вектор середніх значень і вектор стандартних відхилень для ознак у навчальній вибірці. Ці оцінки характеризують спільну функцію щільності ймовірності ознак, яка буде використовуватися для виявлення аномальних спостережень (див. рис. 3.12).

Ввод [16]:

```python
# Підбір моделі
mu_train, sigma_train = X_train_fs.mean(), X_train_fs.std()
```

Ввод [17]:

```python
# Функція для прогнозування аномалії на основі порогового значення щільності ймовірності
def model_normal(X, epsilon):
    """
    Модель виявлення аномалій
    Args:
      X (DataFrame, shape (m, n)): DataFrame ознак
      epsilon (скаляр) : порогове значення щільності (> 0)
    Повертається:
      y (array_like, shape (m,)): передбачені мітки класів
    """
    y = []
    for i in X.index:
        prob_density = normal_product(X.loc[i].tolist(), mu_train, sigma_train)
        y.append((prob_density < epsilon).astype(int))
    return y
```

Рис. 3.12 – Підбір моделі та визначення функції для прогнозування аномалії на основі порогового значення щільності ймовірності

Наступним кроком буде налаштування порогових значень на валідаційному наборі.

Спочатку ми побудуємо деякі функції для обчислення та відображення матриці помилок, а також для обчислення $F2$-оцінки, враховуючи істинні мітки та передбачувані мітки цілі (див. рис. 3.13-14).

Ввод [18]:
```python
# Функція для обчислення матриці помилок
def conf_mat(y_test, y_pred):
    """
    Обчислює матрицю помилок
    Аргументи:
      y_test (array_like): істинні двійкові (0 або 1) мітки
      y_pred (array_like): передбачені двійкові (0 або 1) мітки
    Повертається:
      confusion_mat (масив): двовимірний масив, що представляє матрицю помилок 2x2
    """
    y_test, y_pred = list(y_test), list(y_pred)
    count, labels, confusion_mat = len(y_test), [0, 1], np.zeros(shape = (2, 2), dtype = int)
    for i in range(2):
        for j in range(2):
            confusion_mat[i][j] = len([k for k in range(count) if y_test[k] == labels[i] and y_pred[k] == labels[j]])
    return confusion_mat
```

Ввод [19]:
```python
# Функція для друку матриці помилок
def conf_mat_heatmap(y_test, y_pred):
    """
    Виводить матрицю помилок
    Аргументи:
      y_test (array_like): істинні двійкові (0 або 1) мітки
      y_pred (array_like): передбачені двійкові (0 або 1) мітки
    Повертає:
      Нічого, виводить теплову карту, що представляє матрицю помилок 2x2
    """
    confusion_mat = conf_mat(y_test, y_pred)
    labels, confusion_mat_df = [0, 1], pd.DataFrame(confusion_mat, range(2), range(2))
    plt.figure(figsize = (6, 4.75))
    sns.heatmap(confusion_mat_df, annot = True, annot_kws = {"size": 16}, fmt = 'd')
    plt.xticks([0.5, 1.5], labels, rotation = 'horizontal')
    plt.yticks([0.5, 1.5], labels, rotation = 'horizontal')
    plt.xlabel("Predicted label (Прогнозована мітка)", fontsize = 14)
    plt.ylabel("True label (Справжня мітка)", fontsize = 14)
    plt.title("Confusion Matrix (Матриця помилок)", fontsize = 14)
    plt.grid(False)
    plt.show()
```

Рис. 3.13 – Визначення функції для обчислення матриці помилок та функція для її друку

Ввод [20]:
```python
# Функція для обчислення та повернення f2_score
def f2_score(y_test, y_pred):
    """
    Обчислює точність за істинними та передбачуваними двійковими (0 або 1) мітками
    Аргументи:
      y_test (array_like): істинні двійкові (0 або 1) мітки
      y_pred (array_like): передбачені двійкові (0 або 1) мітки
    Повертається:
      f2 (float): точність, отримана з y_test та y_pred
    """
    confusion_mat = conf_mat(y_test, y_pred)
    tn, fp, fn, tp = confusion_mat[0, 0], confusion_mat[0, 1], confusion_mat[1, 0], confusion_mat[1, 1]
    f2 = (5 * tp) / ((5 * tp) + (4 * fn) + fp)
    return f2
```

Рис. 3.14 – Визначення функції для обчислення та повернення f2_score

Далі перейдемо до налаштування порогового значення щільності. Тобто порівняємо результати F2-оцінки, що залежить від значення щільності і знайдемо оптимальний, при якому F2-оцінки максимальна (див. рис. 3.15).

Ввод [21]:
```python
# Налаштування порогового значення щільності
alpha_list, f2_list, f2_max, alpha_opt, y_val_pred_opt = [], [], 0.0, 0.0, np.zeros(len(y_val))
for alpha, j in itertools.product(np.arange(0.001, 0.051, 0.001), range(1)):
    y_val_pred = model_normal(X_val_fs, epsilon = alpha**X_val_fs.shape[1])
    f2 = f2_score(y_val, y_val_pred)
    alpha_list.append(alpha)
    f2_list.append(f2)
    if f2 > f2_max:
        alpha_opt = alpha
        y_val_pred_opt = y_val_pred
        f2_max = f2
```

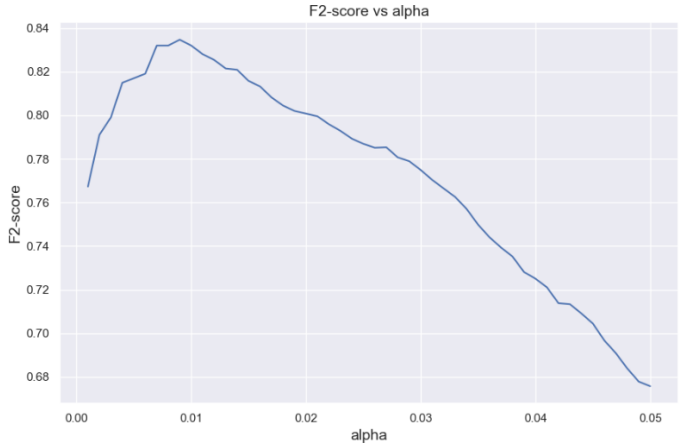
```
100% 50/50 [01:58<00:00, 2.40s/it]
```

Рис. 3.15 – Процес налаштування порогового значення щільності

3.3. Результати моделювання методом Gaussian Anomaly Detection

Тепер побудуємо графіка залежності F2-оцінки від альфа. Також виведемо результати оптимізації порогового значення щільності (див. рис. 3.16).

Ввод [22]:
```
# Побудова графіка залежності F2-оцінки від альфа
plt.figure(figsize = (9, 6))
plt.plot(alpha_list, f2_list)
plt.xlabel("alpha", fontsize = 14)
plt.ylabel("F2-score", fontsize = 14)
plt.title("F2-score vs alpha", fontsize = 14)
plt.tight_layout()
plt.show()
```



Ввод [23]:
```
# Підсумок налаштування
print(pd.Series({
    "Optimal alpha": alpha_opt,
    "Optimal F2-score": f2_score(y_val, y_val_pred_opt)
}).to_string())
```

```
Optimal alpha       0.009000
Optimal F2-score    0.834671
```

Рис. 3.16 – Побудова графіка залежності F2-оцінки від альфа та виведення результатів оптимізації порогового значення щільності

Далі побудуємо матрицю помилок для прогнозів на валідаційному наборі даних (див. рис. 3.17).

```
Ввод [24]:
# Матриця помилок для прогнозів на валідаційному наборі
conf_mat_heatmap(y_val, y_val_pred_opt)
```

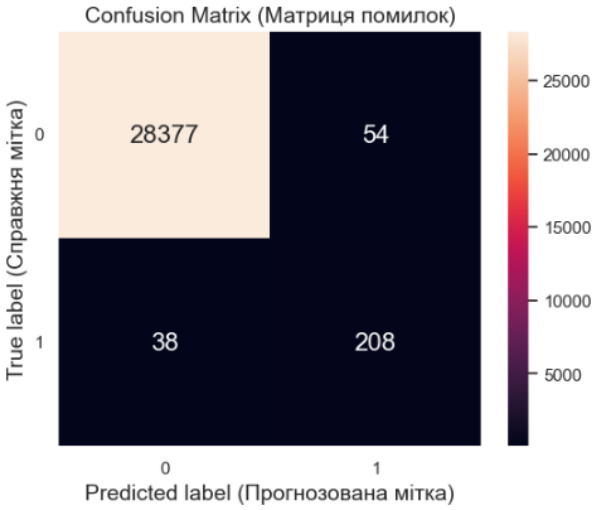


Рис. 3.17 – Підсумок налаштування та матриця помилок для прогнозів на валідаційному наборі

Перейдемо до прогнозування та оцінки на тестовому наборі даних.

Створимо функцію, що обчислить та надрукує метрики оцінки якості нашої моделі. Серед цих метрик, будуть наступні: Accuracy, Precision, Recall, F1-score, F2-score, MCC (див. рис. 3.18).

```
def evaluation(y_test, y_pred):
    confusion_mat = conf_mat(y_test, y_pred)
    tn, fp, fn, tp = confusion_mat[0, 0], confusion_mat[0, 1], confusion_mat[1, 0], confusion_mat[1, 1]
    mcc_value = matthews_corrcoef(y_test, y_pred)
    mcc_value = np.clip(mcc_value, -1.0, 1.0)
    print(pd.Series({
        "Accuracy": (tp + tn) / (tn + fp + fn + tp),
        "Precision": tp / (tp + fp),
        "Recall": tp / (tp + fn),
        "F1-score": (2 * tp) / ((2 * tp) + fn + fp),
        "F2-score": (5 * tp) / ((5 * tp) + (4 * fn) + fp),
        "MCC": mcc_value
    }).to_string())
```

```
# Прогнозування та оцінка на тестовому наборі
y_test_normal = model_normal(X_test_fs, epsilon = alpha_opt**X_test_fs.shape[1])
evaluation(y_test, y_test_normal)
```

```
Accuracy     0.996687
Precision    0.798419
Recall       0.821138
F1-score     0.809619
F2-score     0.816492
MCC          0.808030
```

Рис. 3.18 – Визначення функції для обчислення та друку метрик оцінювання та прогнозування та оцінка на тестовому наборі

Отже, на тестовому наборі даних ми маємо наступні результати (див. табл. 1.1, рис. 3.19).

Табл. 1.1 – Результати оцінки алгоритму на тестовому наборі

Accuracy	0.996687
Precision	0.798419
Recall	0.821138
F1-score	0.809619
F2-score	0.816492
MCC	0.808030

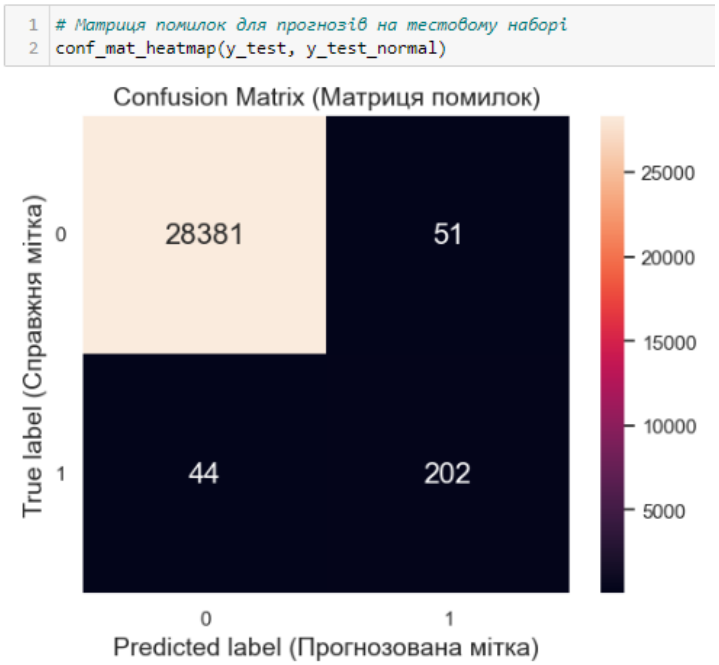


Рис. 3.19 – Матриця помилок для прогнозів на тестовому наборі

Далі використаємо вже створені нами функції для гаусівського виявлення аномалій для визначення ймовірності аномалій у просторі для двох ознак V4 та V11, а потім відобразимо точки на графіку, виділяючи аномальні точки червоним кольором.

```
# Параметри гаусівської моделі:
# середнє значення (μ) та матриця коваріації (Σ) обчислюються на тренувальному наборі даних.
# ЦЕ параметри густини для нормальних (тренувальних) даних.
mu_train2, sigma_train2 = X_train_fs[['V4', 'V11']].mean(), X_train_fs[['V4', 'V11']].std()

# Оцінка густини для точок в просторі тестових даних
density_values = normal_density(X_test_fs['V4'], mu_train2['V4'], sigma_train2['V4']) * \
                 normal_density(X_test_fs['V11'], mu_train2['V11'], sigma_train2['V11'])

# Задаємо поріг для визначення аномалій
# threshold = alfa^k, alfa = const, k - кількість ознак
threshold = alpha_opt**X_test_fs[['V4', 'V11']].shape[1]  # використали порогове значення alpha_opt з нашого алгоритму

# Визначаємо аномалії на основі порогу
anomalies = density_values < threshold

# Отримаємо кількість нормальних  та аномальних  значень
count_anomalies = anomalies.value_counts()

# Виведемо результати
print("Кількість нормальних значень:", count_anomalies[False])
print("Кількість аномальних значень:", count_anomalies[True])


# Побудуємо графік, виділяючи аномалії
plt.scatter(X_test_fs['V4'], X_test_fs['V11'], c='gray', alpha=0.5, label='Нормальні точки')
plt.scatter(X_test_fs['V4'][anomalies], X_test_fs['V11'][anomalies], c='red', label='Аномалії')

plt.xlabel('Ознака V4')
plt.ylabel('Ознака V11')
plt.title('Графік нормальних даних та аномалій')
plt.legend()
plt.show()
```

Рис. 3.20 – Код для побудови графіка аномалій та нормальних даних

```
Кількість нормальних значень: 28449
Кількість аномальних значень: 229
```

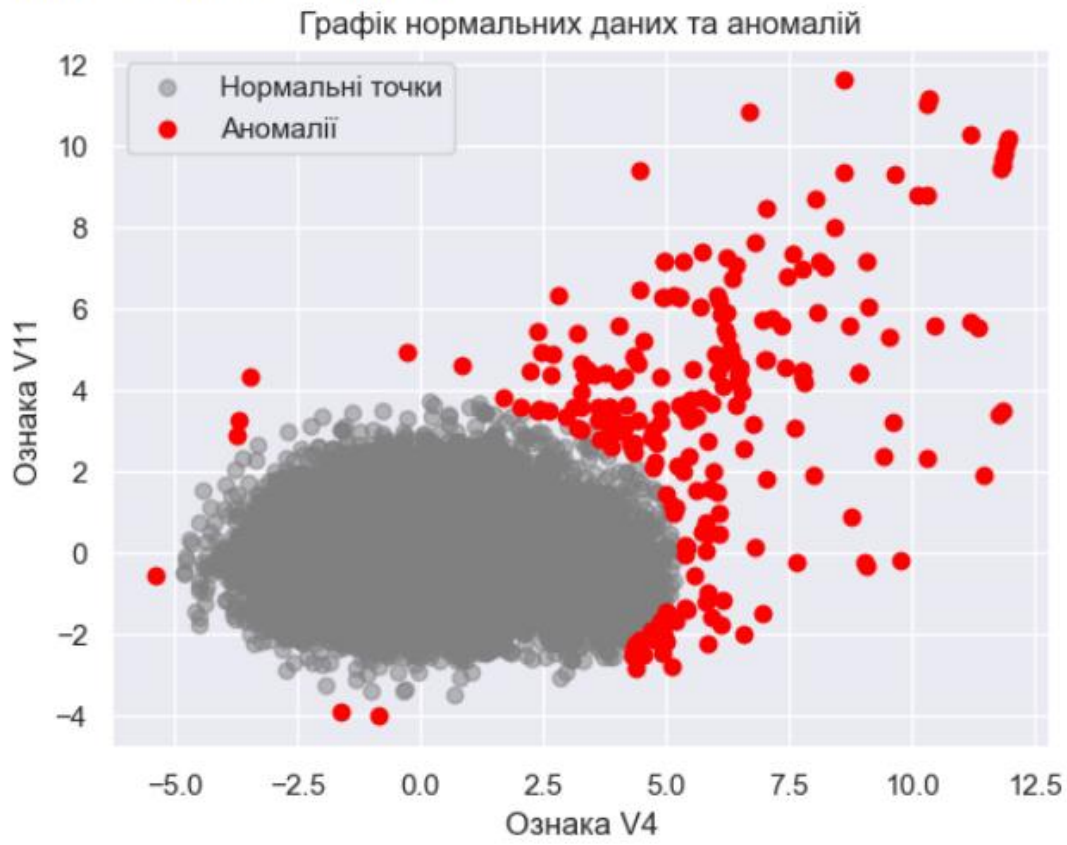


Рис. 3.21 – Графік аномалій та нормальних даних

3.4. Порівняння результатів з методом Isolation Forest

Спочатку на основі f2-оцінки підберемо оптимальний параметр забруднення (contamination) для моделі побудованої на основі вбудованого алгоритму Isolation Forest. Підбір відбувається на валідаційних даних.

```
# Задайте діапазон значень contamination для перевірки
param_grid = {'contamination': [0.001, 0.005, 0.01, 0.05, 0.1, 0.2]}

# Визначте модель та метрику для оцінки
model = IsolationForest()

# Використовуйте make_scorer для визначення своєї метрики
scorer = make_scorer(f2_score)

# Виконайте пошук по сітці з використанням крос-валідації
grid_search = GridSearchCV(model, param_grid=param_grid, scoring=scorer, cv=5)
grid_search.fit(X_val_fs, y_val)

# Виведіть оптимальні значення та їх ефективність
print("Optimal contamination:", grid_search.best_params_['contamination'])
print("Best f2_score:", grid_search.best_score_)
```

```
Optimal contamination: 0.001
Best f2_score: 0.008888114325548972
```

Рис. 3.22 – Код для пошуку оптимального значення параметру для алгоритму Isolation Forest та результат його виконання

Побудуємо модель на основі оптимального параметру contamination та оцінимо її якість.

```
# Ініціалізація та навчання моделі
isolation_forest = IsolationForest(contamination=grid_search.best_params_['contamination'])
isolation_forest.fit(X_train_fs)

# Прогнозування аномалій на валідаційних даних
predicted_labels = isolation_forest.predict(X_val_fs)
predicted_labels = (predicted_labels == -1).astype(int)

# Розрахунок метрик на валідаційних даних
evaluation(y_val, predicted_labels)
```

```
Accuracy     0.996583
Precision    0.873737
Recall       0.703252
F1-score     0.779279
F2-score     0.731810
MCC          0.782233
```

Рис. 3.23 – Код для побудови моделі та оцінки її якості разом з результатами оцінки моделі на валідаційних даних

Оцінимо якість моделі на тестових даних.

```
# Прогнозування аномалій на тестових даних
predicted_labels_test = isolation_forest.predict(X_test_fs)
predicted_labels_test = (predicted_labels_test == -1).astype(int)

# Розрахунок метрик на тестових даних
evaluation(y_test, predicted_labels_test)
```

```
Accuracy     0.996374
Precision    0.877660
Recall       0.670732
F1-score     0.760369
F2-score     0.703925
MCC          0.765545
```

Рис. 3.24 – Код для побудови моделі та оцінки її якості разом з результатами оцінки моделі на тестових даних

Табл. 1.2 – Результати оцінки алгоритму Isolation Forest на тестовому наборі

Accuracy	0.996374
Precision	0.877660
Recall	0.670732
F1-score	0.760369
F2-score	0.703925
MCC	0.765545

Побудуємо графік аномальних і нормальних даних для 2 ознак.

```python
# Виведіть кількість нормальних і аномальних даних
normal_count = len(predicted_labels_test[predicted_labels_test == 0])
anomaly_count = len(predicted_labels_test[predicted_labels_test == 1])
print("Кількість нормальних значень:", normal_count)
print("Кількість аномальних значень:", anomaly_count)

# Додайте дані для графіку
X1, X2 = X_test_fs['V4'], X_test_fs['V11']
plt.scatter(X1, X2, c=predicted_labels_test, cmap='viridis', label='Нормальні дані')
plt.scatter(X1[predicted_labels_test == 1], X2[predicted_labels_test == 1], c='red', marker='x', label='Аномалії')

# Позначте осі та додайте легенду
plt.xlabel('Ознака V4')
plt.ylabel('Ознака V11')
plt.title('Графік нормальних даних та аномалій')
plt.legend()
plt.show()
```

Рис. 3.25 – Код для побудови графіка аномалій та нормальних даних

```
Кількість нормальних значень: 28490
Кількість аномальних значень: 188
```

Рис. 3.26 – Графік аномалій та нормальних даних

Далі порівняємо результати двох моделей.

Табл. 1.3 – Результати оцінки алгоритму Gaussian Anomaly Detection та Isolation Forest на тестовому наборі

	Gaussian Anomaly Detection	Isolation Forest
Accuracy	0.996687	0.996374
Precision	0.798419	0.877660
Recall	0.821138	0.670732
F1-score	0.809619	0.760369
F2-score	0.816492	0.703925
MCC	0.808030	0.765545

Як бачимо з таблиці 1.3, метод гаусіувського виявлення аномалій (Gaussian Anomaly Detection) в цілому краще виявляє аномалії, ніж метод ізоляційного лісу (Isolation Forest). Тому висновки щодо результату будуть сконцентровані на першому методі.

ВИСНОВКИ

У роботі створено алгоритм на основі функцій розподілу імовірності для виявлення аномальних (шахрайських) операцій з кредитними картками. Сутність його полягає в визначенні параметрів одновимірних гаусівських розподілів для кожної впливової характеристики на даних без аномальних транзакцій (аномальні транзакції є викидами і суттєво б спотворювали б характеристики таких розподілів). Далі досліджуючи дані з аномальними транзакціями за допомогою функції розподілу, що є добутком парціальних функцій розподіли знаходимо пороги, які дозволяють відділити звичайні транзакції від аномальних. Отримана система перевірялась на тестовому наборі і показала високі показники якості.

Результати експериментів показують, що метод гаусівського виявлення аномалій є більш ефективний у виявленні аномалій порівняно з методом ізоляційного лісу (Isolation Forest) із бібліотеки scikit-learn .

Під час проведення дослідження було виявлено, що дані сильно незбалансовані, оскільки шахрайські транзакції виникають рідше у порівнянні з автентичними і складають 0,2% від усіх даних.

Використовуючи порівняння розподілу кожної ознаки для цільових класі відібрали такі ознаки: V4, V11, V12, V14, V16, V17, V18, V19 та Hour. Визначили поріг, отриманий на основі F2-оцінки.

Аналіз метрик якості класифікації, таких як Accuracy, Precision, Recall, F1-score, F2-score та MCC, показав високий рівень ефективності моделі, що побудована на гасівських розподілах. Зокрема, висока F2-оцінка (81.65%) свідчить про вдале визначення порогу для виявлення шахрайських транзакцій, навіть на фоні сильної назбалансованості даних.

Таким чином, можемо стверджувати, що метод гаусівського виявлення аномалій дозволяє ефективно визначати шахрайські транзакції навіть за умови сильно незбалансованих даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Frąckiewicz, M. (2023). ШІ та прогнозна аналітика для розслідування шахрайства: використання машинного навчання для розслідування та виявлення фінансових злочинів. Штучний інтелект. [https://ts2.space/uk/ші-та-прогнозна-аналітика-для-розслід/]

2 (2023). Боротьба з онлайн-шахрайством. Interkassa Blog. [https://interkassa.com/blog/borotba-z-onlajn-shahrajstvom]

3 Гриценко, К.Г. (2023). Аналіз методів виявлення шахрайства у банках, що здійснюються персоналом банку. Наукова стаття. Сумський державний університет, Кафедра економічної кібернетики. [https://doi.org/10.32843/infrastruct34-48]

4 Smith, J., & Jones, A. (2018). "Credit Card Fraud Detection Using Machine Learning: A Review." Journal of Financial Analytics, 15(2), 45-63

5 Johnson, R., & Brown, S. (2016). "A Survey of Credit Card Fraud Detection Techniques: Data and Technique Perspective." International Journal of Data Science and Analytics, 12(4), 321-340.

6 Kumar, A. (2019, February 10). Wondering how to build an anomaly detection model? Make your own anomaly detection model from scratch in python. Towards Data Science. [https://towardsdatascience.com/wondering-how-to-build-an-anomaly-detection-model-87d28e50309]

7 Amos, S. (2017, September 12). Anomaly detection algorithm implemented in Python. [https://udohsolomon.github.io/_posts/2017-09-12-Anomaly-detection/]

8 DeepLearning.AI.. Supervised Machine Learning: Regression and Classification. Викладачі: Andrew Ng, Aarti Bagul, Geoff Ladwig. Курс входить до Спеціалізації Machine Learning. Coursera. [https://www.coursera.org/learn/machine-learning/]

9 Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Publisher: Springer. Hardcover, 738 pages. Language: English. ISBN-10: 0387310738, ISBN-13: 978-0387310732. Published on August 17, 2006.

10 Chandola, V., Banerjee, A., & Kumar, V. (2009). "Anomaly Detection: A Survey." *ACM Computing Surveys*, 41(3). Retrieved from [https://www.researchgate.net/publication/220565847_Anomaly_Detection_A_Survey] (DOI: 10.1145/1541880.1541882).

11 Wasserman, L. (2022). "All of Statistics: A Concise Course in Statistical Inference". Publisher: Springer. Hardcover, 649 pages. Language: English. Published on November 19, 2022.

12 Zhiguo Ding and Minrui Fei. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. IFAC Proceedings Volumes, 46(20):12–17, 2013.

13 James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). "An Introduction to Statistical Learning." Springer; 1st ed. 2013, Corr. 7th printing 2017 edition. Hardcover, 440 pages. Publisher: Springer; 1st ed. 2013, Corr. 7th printing 2017 edition (June 25, 2013). Language: English. ISBN-10: 1461471370, ISBN-13: 978-1461471370.

14 Harrington P. Machine learning in action / P. Harrington. — Dublin: Manning Publications, 2012. — 382 p.

15 MACHINE LEARNING GROUP - ULB. (2017). Credit Card Fraud Detection. Kaggle. [https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud]

16 Ghosh, S. (2022). Anomaly Detection in Credit Card Transactions. Kaggle. [https://www.kaggle.com/code/sugataghosh/anomaly-detection-in-credit-card-transactions]

17 Ghosh, S. (2022). Credit Card Fraud Detection - Part 2. Modeling. Kaggle. [https://www.kaggle.com/code/sugataghosh/credit-card-fraud-detection-part-2-modeling]