Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка
до кваліфікаційної роботи бакалавра
за спеціальністю 123 «Комп'ютерна інженерія»

на тему: СИСТЕМА ПЕРЕВІРКИ ДОСТОВІРНОСТІ ІНФОРМАЦІЇ З МЕРЕЖІ ІНТЕРНЕТ

Проектував _____ Д. М. Колєснік

Керівник роботи _____ О. А. Сенько

Нормоконтроль _____ Д. І. Кузнєцов

Завідувач кафедри _____ А. І. Купін

Кривий Ріг
2024

Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних систем та мереж

Ступінь вищої освіти бакалавр
Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри, голова циклової комісії

_____А. І. Купін

"____" _____20__року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

_____Колєснік Дмитро Мар`янович_____
(прізвище, ім'я, по батькові)

1. Тема роботи Система перевірки достовірності інформації з мережі Інтернет_
_____
_____
керівник роботи ____Сенько А.О._____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "___"_____20__року №___
2. Строк подання студентом роботи _____
3. Вихідні дані до роботи _____
масив даних з транзакціями користувачів платіжної системи._____
_____
_____
_____
_____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _ Дослідити поточний стан та особливості використання методів машинного навчання для виявлення шахрайських операцій, провести пошук даних для навчання алгоритмів, розробити математичні моделі випадкового лісу та градієнтного бустингу для класифікації транзакцій і на їх основі розробити інформаційну технологію для виявлення шахрайських операцій.

_____
_____
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)___Презентація PowerPoint_____
_____
_____
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1			
2			
3			

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1	Пошук підходящої літератури	12.03.24 - 18.03.24	Виконав
2	Розробка структури дипломної роботи та сортування інформації за розділами	20.03.24	Виконав
3	Огляд інформації що стосується тестування	24.03.24 – 28.03.24	Виконав
4	Написання першого розділу	01.04.24- 10.04.24	Виконав
5	Огляд існуючих інструментів	20.04.24	Виконав
6	Написання другого(аналітичного розділу)	23.04.24	Виконав
7	Створення тест плану з описом тест-кейсів	03.05.24	Виконав
8	Внесення результатів у технологію.	10.05.24	Виконав
9	Написання третього розділу	15.05.24	Виконав
10	Написання пояснювальної записки та загальне оформлення дипломної роботи	20.05.24	Виконав

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

ШАХРАЙСЬКІ ОПЕРАЦІЇ, МАШИННЕ НАВЧАННЯ, БІНАРНА КЛАСИФІКАЦІЯ, ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ

Пояснювальна записка : 65 с, 26 рис., 3 табл., 1 дод., 11 джерел.

Предметом дослідження є алгоритми класифікації транзакцій засновані на структурних моделях.
Об'єктом є відкритий набір даних про транзакції користувачів платіжної системи з веб-платформи Kaggle.
Метою роботи є розробка методів виявлення шахрайських операцій. Результатом є створена інформаційна технологія для виявлення шахрайських операцій.
Актуальність полягає в необхідності використання новітніх методів виявлення злочинних дій у банківській сфері, особливо в контексті глобалізації та діджиталізації банківських послуг. Методи машинного навчання є ефективним інструментом для виявлення шахрайських операцій.

Explanatory note

Explanatory note: 65 pages, 26 figures, 3 tables, 11 used sources.

The object of analysis is a system of automated testing of web applications.

The goal is to consider the theory of test automation and create a system of self-tests based on it.

The first section is devoted to the collection and analysis of information on automated and manual testing. In the second section, popular web application testing automation services were considered, their features were described, the main disadvantages and advantages of each of them were formed. In the third section, a self-testing system based on Katalon Studio was developed and implemented. The obtained results were analyzed.

Зміст

Вступ 7
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 8
1.1 Актуальність проблеми 8
1.2 Огляд проблематики виявлення шахрайств 9
1.3 Огляд існуючих досліджень 11
1.4 Постановка задачі 12
1.5 Висновок до розділу 13
РОЗДІЛ 2. МАТЕМАТИЧНІ МОДЕЛІ ВИЯВЛЕННЯ ШАХРАЙСЬКИХ ДІЙ 14
2.1 Обирання бінарної класифікації 15
2.2 Бустинг 23
2.3 Матриця помилок 27
2.4 Покращення результатів роботи моделей 30
2.5 Висновки до розділу 2 34
РОЗДІЛ 3. АНАЛІЗ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 36
2.1 Опис початкового набору даних 36
2.2 Змінна TransactionDT 41
2.3 Навчання моделей 49
2.4 Висновки до розділу 3 52
ВИСНОВКИ 54
ПЕРЕЛІК ПОСИЛАНЬ 55
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ 56

Вступ

Цифрова трансформація суспільства та економіки стає дедалі актуальнішою з кожним роком, не оминаючи банківський сектор. Миттєві грошові перекази та оплата товарів банківською картою стали невід'ємною частиною нашого повсякденного життя. На жаль, разом із зручностями та підвищенням рівня життя зростає й ризик шахрайства. Тому виявлення та запобігання шахрайству з картками є одним із найважливіших завдань сучасних платіжних провайдерів.

Індустрія запобігання шахрайству активно використовує широкий спектр методів машинного навчання для постійного вдосконалення своїх систем. Визначення найефективнішої моделі для виявлення шахрайських операцій є ключовим аспектом успішної побудови системи захисту платежів. Отже, це дослідження присвячене виявленню, аналізу та адаптації методів машинного навчання для виявлення шахрайських транзакцій.

Перший розділ роботи охоплює тематичні напрями, розглядає актуальність і проблематику задачі, а також її формалізацію. У цьому розділі також досліджуються основні принципи роботи платіжних систем та їх вразливості до шахрайських дій. Окремо висвітлюється важливість використання машинного навчання в сучасних умовах для підвищення ефективності виявлення шахрайства.

У другому розділі аналізуються існуючі підходи до визначення операційних ризиків на основі статистичних методів і методів штучного інтелекту. Розглядаються переваги та недоліки різних алгоритмів, таких як логістична регресія, дерева рішень, нейронні мережі та методи ансамблю. Також обговорюються сучасні тенденції та інновації в цій сфері.

Третій розділ містить практичні результати дослідження, зокрема, впровадження та тестування обраних моделей на реальних даних. Оцінюється ефективність різних методів, аналізуються їх результати та надаються рекомендації щодо покращення систем виявлення шахрайства. Окрім цього, обговорюються можливі шляхи подальшого розвитку технологій та їх адаптація до змінних умов ринку.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проблеми

Види фальшу з кредитками, на жаль, є незаперечним фактом сучасності. У 2021 році 65 гривень з кожного мільйона операцій по платіжним картокам були пов'язані з незаконними або підозрілими операціями. А за даними ЄС банку, показник у кілька разів вищий, досягаючи 0,036% у 2019 році [1]. Таким чином, можна стверджувати, що шахрайство стало глобальною проблемою.

Врахування законності, своєчасне виявлення та запобігання шахрайству є критичними викликами для компаній, що надають платіжні послуги. Прогалини в цій сфері призводять до значної репутаційної шкоди та збільшують втрати клієнтів і бізнесу через недостатню безпеку платіжних систем. Методи машинного навчання є одними з найперспективніших технологічних засобів у боротьбі проти фінансового шахрайства.

Алгоритми машинного навчання здатні розрізняти шахрайські транзакції від законних, не викликаючи підозр у користувачів. Ці алгоритми проводять обробку інформації швидше і точніше, ніж люди. Крім того, використання машинного навчання мінімізує людський фактор і дозволяє масштабувати рішення. Серед таких методів можна виділити нейронні мережі, методи ансамблю, дерева рішень та градієнтний бустинг. Кожен з цих підходів має свої переваги і недоліки, але загалом вони дозволяють значно підвищити ефективність систем виявлення шахрайства.

Міжнародні компанії в галузі забезпечення безпеки електронної комерції щороку вкладають значні ресурси в розробку та дослідження нейронних мереж, нечітких систем, еволюційних алгоритмів і колективного інтелекту для більш ефективної боротьби з фінансовим шахрайством. Вони активно впроваджують ці технології у свої продукти та послуги, забезпечуючи більш надійний захист для своїх клієнтів.

Одним із перспективних напрямів є розвиток систем самонавчання, які можуть адаптуватися до нових типів шахрайських дій у реальному часі. Такі системи здатні не лише аналізувати минулі транзакції, але й прогнозувати можливі загрози на основі виявлених аномалій. Це дозволяє запобігати шахрайству на ранніх стадіях, зменшуючи фінансові втрати та підвищуючи довіру клієнтів до платіжних систем.

Крім того, важливим аспектом є міжнародна співпраця та обмін інформацією між

фінансовими установами, правоохоронними органами та розробниками технологій. Спільні зусилля у боротьбі з шахрайством дозволяють швидше виявляти нові методи зловмисників та розробляти ефективні контрзаходи. У цьому контексті важливу роль відіграють міжнародні стандарти та регуляції, що встановлюють вимоги до безпеки платіжних систем та сприяють впровадженню передових технологій для запобігання шахрайству.

Отже, у сучасному світі шахрайство з кредитними картками є серйозною проблемою, яка потребує комплексного підходу до її вирішення. Використання методів машинного навчання, розвиток самонавчальних систем та міжнародна співпраця є ключовими елементами успішної боротьби з цією загрозою.

1.2 Огляд проблематики виявлення шахрайств

Відповідно до законодавства України, шахрайство визначається як придбання майна або майнових прав іншої особи шляхом обману або зловживання довірою. Давайте розглянемо деталі цієї злочинної діяльності, щоб краще зрозуміти основні проблеми, які можуть виникнути під час використання методів машинного навчання для виявлення шахрайських транзакцій.

Виявлення банківського шахрайства стикається з багатьма викликами та труднощами. Більша частина цього процесу залежить від інтелектуального аналізу даних, причому значна їх частка можливо буде недоступною для банків або фінансових установ через їх конфіденційність. Крім того, величезна кількість транзакцій, що відбуваються щодня, робить цей аналіз серйозною проблемою зі сторони наявних технологій і професійних навичок, які аналізують дані.

Методики боротьби з шахраями постійно прогресують, отож шахраї також змінюють свої підходи для досягнення своїх цілей. Їх вчинки, пов’язані з системою платежів, кредитними картами та реальними транзакціями, часто класифікуються як крадіжка особистих даних. Ці злочини включають незаконне володіння даними жертви. Зловмисники можуть видавати себе за цю особу різними способами, від злому облікового запису в Інтернет-банку до викрадення платіжних карток.

Для ефективного виявлення шахрайства необхідно перевіряти методи підробки послідовно та ретельно, зокрема ті, що використовуються для здійснення операцій по рахунку.

У таблиці 1.1 показано подібні шахрайські дії, класифіковані за типом збору персональних даних.

№	Вид шахрайства	Опис
1	Викрадення фізичної картки	Отримавши доступ до платіжної картки, шахрай може здійснювати покупки онлайн від імені власника картки, а також знімати готівкові кошти.
2	Заволодіння банківськими виписками	Банківські виписки, які містять детальну інформацію про всі операції по певному рахунку за вказаний період, надають шахраям повні дані про поведінку власника картки. Це дозволяє викрасти його особистість і отримати доступ до коштів з рахунку різними способами.
3	Проникнення у авторизований пристрій	Підключившись до загальної Wi-Fi мережі та перехопивши паролі від рахунків, або отримавши віддалений чи фізичний доступ до пристрою, авторизованого в банківському додатку, шахрай може отримати доступ до облікового запису з грошовим рахунком.
4	Фішинг	Метод соціальної інженерії, що полягає у надсиланні підроблених листів з проханням підтвердити авторизацію на певному сайті, а також форм, замаскованих під відомі додатки, для збору автентифікаційних даних користувачів з метою подальшого викрадення коштів.
5	Підробка карти	Платіжну картку можна підробити, зчитавши і скопіювавши дані з магнітної смужки справжньої кредитної або дебетової картки. Таку підроблену картку можна використовувати для оплати товарів і послуг, а також для зняття готівки у банкоматах, які не вимагають додаткової перевірки PIN-коду або чіпу картки.
6	Телефонне шахрайство (вішинг)	Вдаючи іншу особу, шахрай може здійснювати телефонні дзвінки, щоб отримати конфіденційну інформацію про банківський рахунок, обманом добитися підтвердження певної операції або безпосередньо переконати перевести йому кошти. Телефонне шахрайство є окремим видом фішингу.

Розглянувши класифікацію шахрайства, можна зробити висновок, що для його ефективного запобігання необхідно провести низку перевірок як операції, так і особи користувача, щоб втрата персональних даних не призвела до повної захоплення Масу.

Шахрай дізнається особу власника банківського рахунку.

Використання методів машинного навчання може бути ефективним кроком для перевірки транзакцій на потенційне шахрайство.

Говорячи про існуючі комерційні рішення цієї проблеми, варто згадати корпорацію «Веста», яка надає платіжні послуги для телекомунікаційних компаній.
Компанія розробила масштабовану систему для виявлення та усунення цифрового шахрайства на основі штучного інтелекту.
Унікальна програма може адаптуватися до схем шахрайства, що постійно змінюються, і захищати картки користувачів у режимі реального часу.

1.3 Огляд існуючих досліджень

Багато літератури про аномалії платіжних карток та виявлення шахрайства вже опубліковано та доступно для громадськості. Наприклад, детальний огляд Кліфтона Фуа та інших дослідників показав, що методи, які використовуються в цій галузі, включають інтелектуальний аналіз даних і автоматичне виявлення шахрайства [3]. Одним із висновків цього дослідження є те, що вчені іноді використовують занадто складні моделі, тоді як менш ресурсомісткі методи, як-от логістична регресія або байєсовські мережі, можуть бути так само ефективними.

В іншій статті, Суман, дослідник з GJUS&T у Хісар HCE, вивчав застосування контрольованих і неконтрольованих пристроїв, що навчаються автоматично для виявлення шахрайства з платіжними картками. Він порівняв ефективність моделей дерева рішень, методів опорних векторів та нейронних мереж. Незважаючи на те, що ці методи та алгоритми досягли певного успіху у виявленні шахрайства, їм не вдалося забезпечити тривале та стабільне рішення для виявлення шахрайських дій. [4]
Автори використовували різноманітні комбінацію методів та біометричних підходів, таких як додаткова двофакторна аутентифікація на основі біометричних даних (відбитків пальців, голосу, рис обличчя) або даних про поведінку (рухи миші на екрані комп’ютера). Основна проблема цього підходу полягає не лише в інтеграції складних багатофакторних систем аутентифікації, але й у розробці додаткових протоколів для обробки та зберігання даних користувачів.

Wen-Fang Yu та Na Wang провели дослідження, спрямоване на створення більш ефективних моделей класифікації транзакцій. Вони використовували аналіз викидів, виявлення викидів та алгоритми підсумовування відстані для точного передбачення шахрайських транзакцій на основі експерименту з емуляцією записів переказів по кредиткам комерційних банків. Майнінг викидів, як область аналізу даних, широко використовується у фінансовій та інтернет-сферах. Він передбачає виявлення об’єктів, що відрізняються від основної системи, тобто підозрілих транзакцій.

Yu та Wang брали атрибути поведінки клієнтів і на основі значень цих атрибутів обчислювали відстань між фактичним значенням і заданим значенням. Так гібридний алгоритм інтелектуального аналізу даних,

який вони застосували, зміг виявити підозрілі інциденти у наборі даних реальних карткових транзакцій. Цей метод, заснований на алгоритмі реконструкції мережі, здатний створювати уявлення про відхилення екземплярів від контрольної групи. Загалом, він показав свою ефективність для онлайн-транзакцій середнього розміру.

Проблема класової нерівності виникає у багатьох дослідженнях, де шахрайські транзакції становлять менше 10% від усіх транзакцій. Зазвичай такий метод автоматичного навчання використовуються для аналізу всіх транзакцій та виявлення підозрілих випадків. Ці випадки перевіряються працівниками банку або платіжної системи, які можуть звернутися до власника картки для підтвердження операції. Додатково, звіти працівників, що включають результати опитувань власників рахунків, продвигають покращення моделі в класі переказів та пошуку перемін в шахрайстві.

1.4 Постановка задачі

У дослідженні, присвяченому виявленню шахрайських транзакцій за допомогою методів машинного навчання, було проведено аналіз цієї області, а також розробку та впровадження сучасних підходів до вивчення математичних моделей для виявлення зловмисних транзакцій.

Отже, дослідження проводиться з ціллю розробки програми для виявлення хакерства, яка включає в себе результати вищевказаних шляхів.

Завдання викриття транзакцій в такому ключі можна назвати бінарною класифікації.

Використовуємо тестовий набір, що містить дані про транзакції, як вхідні дані для нашого моделювання. Результатом є набір значень для змінної isFraud, яка вказує на наявність шахрайства для кожної транзакції в тестовому наборі. Після розгляду існуючих рішень у сфері виявлення шахрайських транзакцій можна сформулювати низку вимог до майбутніх інформаційних технологій.

По-перше, нам потрібно виконати певну попередню обробку даних.

Це покращує передбачувану силу майбутніх моделей.

По-друге, нам потрібно побудувати модель класифікації, яка встановлює зв’язки в даних навчання з прийнятною точністю.

Ми також рекомендуємо запитувати можливість перегляду графіків залежностей у ваших навчальних зразках, що може бути корисним для ваших користувачів.

З розвитком інформаційних технологій стає необхідним виробка новіших моделей для обробки даних та проведення досліджень. Це вимагає використання мов програмування високого рівня.

1.5 Висновок до розділу

Цей розділ був присвячений актуальності теми виявлення шахрайських транзакцій, огляду існуючих рішень, постановки етапів та визначення подальшого дослідження.

За вказаною інформацією можна вважати, що шахрайські банківські операції представляють серйозну загрозу для сучасних платіжних систем. Це підкреслює необхідність подальшого розвитку інформаційних технологій для виявлення шахрайства.

Варто також зазначити, що проблема виявлення маніпуляцій розглянута як сценарій для розв'язання завдань бінарної класифікації за допомогою методів машинного навчання. У цьому розділі висунуті вимоги до майбутніх інформаційних технологій, які дозволять боротися з шахрайством в банківській сфері.

РОЗДІЛ 2. МАТЕМАТИЧНІ МОДЕЛІ ВИЯВЛЕННЯ ШАХРАЙСЬКИХ ДІЙ

Після вивчення предметної області та формулювання постановки проблеми, перейдемо до основних методів і математичних моделей, які будуть належним чином використовуватися в майбутніх інформаційних системах для відслідковування зловмисних операцій.

Базуючись на наборах даних, доступних для дослідження, і аналізі щодо виявлення шахрайства, варто спочатку зосередитися на сферах машинного навчання, таких як контрольоване навчання.

Навчання з репетиторами - це метод машинного навчання, що ґрунтується на використанні набору пар вхідних та вихідних даних для моделювання зв'язків у системі [6].

В цій парадигмі алгоритм працює з вхідними даними, що називаються міченими. Тобто обидва елементи є даними, присутніми в парі «вхід і вихід».

Лінійна регресія є однією з найпростіших керованих моделей машинного навчання, яку можна використовувати для аналізу помилок прогнозування моделі.

Нехай задано модель (2.1):

$$\bar{y} = X\bar{w} + \varepsilon \qquad (2.1)$$

де $\bar{y}$ - цільова змінна;

$\bar{w}$ - вектор параметрів моделі (також відомий як вектор вагів), X - матриця спостережень (ознак) розміром $n \times (m + 1)$ (враховуючи стовпець з одиниць);

ε - випадкова змінна, непрогнозована помилка моделі.

Отже, значення цільової змінної у складаються з певної функції $f(\bar{x})$, яка є детермінованою, та випадкової похибки ε. У цьому випадку похибки мають нормальний розподіл із середнім значенням нуль та певною дисперсією $\sigma 2$, тобто y розподілено за законом $N(f(\bar{x}), \sigma 2)$.

Необхідно апроксимувати невідому, але детерміновану функцію $f(\bar{x})$ функцією $f(\bar{x})$, яка є результатом навчання моделі і представляє собою точкову оцінку функції f . Отож, похибка в х може

бути розкладена за формулою (2.2):$Err(\bar{x}) = E[(y - \hat{f}(x))^2] = E[y^2] + E[\hat{f}^2] - 2E[y\hat{f}]$(2.2)

Оскільки

$$Ey^2 = Var(y) + E[y]^2 = \sigma^2 + f^2 ,$$

$$E[\hat{f}^2] = Var(\hat{f}) + E[f]^2 ,$$

$$E[y\hat{f}] = E[(f + \varepsilon)\hat{f}] = fE[\hat{f}] + E\varepsilon E[\hat{f}] = fE[\hat{f}] ,$$

Отримавши формулу (2.3) для похибки моделі

$$Err(\bar{x}) = E[(y - \hat{f}(x))^2] = \sigma^2 + f^2 + Var(\hat{f}) + E[f]^2 - 2fE[\hat{f}] =$$

$$= (f - E[\hat{f}])^2 + Var(\hat{f}) + \sigma^2 = Bias(\hat{f})^2 + Var(\hat{f}) + \sigma^2 (2.3)$$

Так, похибка будь-якої моделі у вигляді $y = f(x) + \varepsilon$ складається зі зсуву (зрушення), що може свідчити про наявність неправильних припущень про зв'язки в даних, зроблених під час процесу навчання моделі (це також відомо як недостатнє навчання). – Дисперсія (розкид) – помилка, яка вказує на дисперсію помилки.

Моделі з цією властивістю можуть імітувати випадковий шум, а не зв’язки в навчальних даних [7].

Як бачите, помилка мінімізується, коли і зміщення, і дисперсія зменшуються одночасно.

Однак зазвичай потрібно знайти компроміс між упередженими та нестабільністю.

Ця проблема також відома як компроміс зсуву дисперсії.

Цей розділ містить огляд існуючих контрольованих методів машинного навчання для вирішення задач бінарної класифікації, тобто проблем, де цільова змінна приймає лише два значення, які можуть бути представлені 0 і 1.

Після розгляду компромісів Машинне навчання розрізняє зміщення та дисперсію, тож враховуйте це під час розгляду своєї моделі.

2.1 Обирання бінарної класифікації

Розв'язання задачі бінарної класифікації передбачає знаходження функції, яка перетворює набір різноманітних прикладів у набір міток, що вказують на приналежність до одного з двох відомих класів. Формально, при наявності навчальної вибірки S з вхідними змінними A = $\{ai\}$, $i = 1, \ldots, n$, та цільовою змінною y, яка має невідомий фіксований розподіл D у розміченому просторі прикладів, мета алгоритмів вивчення класифікаторів полягає у створенні оптимального класифікатора з мінімальною узагальненою помилкою. Позначимо DT алгоритм вивчення класифікатора, DT(S) - класифікатор, навчений на наборі S, а $DT(S)(x)$ - результат класифікації для прикладу x. Узагальнена помилка - це

частка помилкових класифікацій за розподілом D. Цю помилку можна представити у вигляді виразу (2.4):

$$\varepsilon(DT(S), D) = \sum_{x,y \in U} D(x,y) L(y, DT(S)(x)) \quad (2.4)$$

де $L(y, DT(S)(x))$ - функція втрат вигляду (2.5):

$$L(y, DT(S)(x)) = \begin{cases} 0, y = DT(S)(x) \\ 1, y \neq DT(S)(x) \end{cases} \quad (2.5)$$

Індуктивний алгоритм (або учень у контексті машинного навчання) — це об'єкт, який приймає навчальний набір даних як вхідні дані та формує модель, яка узагальнює зв'язки між вхідними атрибутами та цільовими атрибутами.

Наприклад, індуктор може взяти як вхід певний навчальний кортеж із відповідною міткою класу його запису та створити класифікатор як результат.

У наступному підрозділі ми розглянемо популярні на даний момент алгоритми вивчення моделей класифікації.

Дерева рішень

Дерева рішень є одним з найбільш зрозумілих способів класифікації даних. Це кореневе, спрямоване дерево, схоже на блок-схему. Кожен внутрішній вузол представляє рішення про поділ, а кожен листок відповідає передбаченому класу. Дерево рішень можна розглядати як послідовність запитань або блоків if/else. На рисунку 2.1 показано приклад найпростішого дерева рішень.

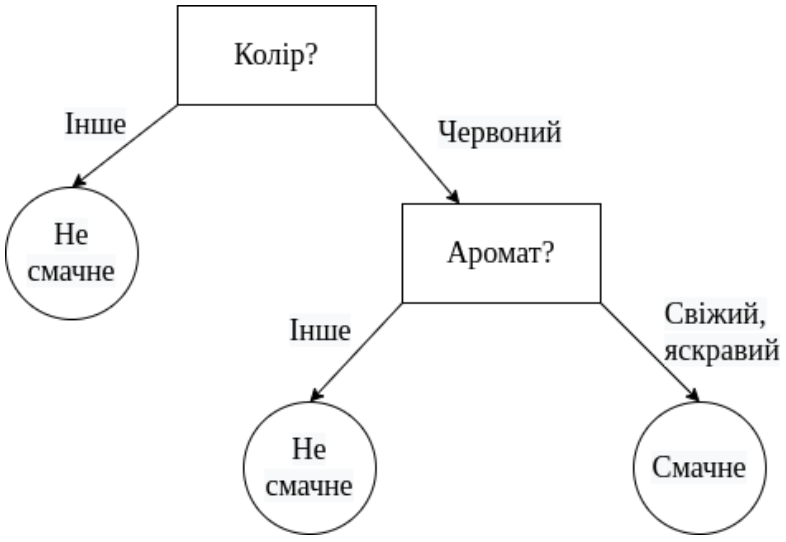


Рисунок 2.1 Найпростіше дерево рішень для вибору

Дерево можна навчити, розділивши вихідний набір розбиття даних на підмножини здійснюється шляхом перевірки значень атрибутів.

Цей процес рекурсивно повторюється для кожної похідної підмножини, що відомо як рекурсивне розбиття. Важливо відзначити, що задача побудови оптимального дерева рішень є дуже складною. Наприклад, доведено, що завдання побудови мінімального бінарного дерева для класифікації раніше невідомого набору даних за мінімальною кількістю тестів є NP-повним. Ці результати підтверджують, що використання оптимальних алгоритмів дерева

рішень можливе лише на невеликих наборах даних з невеликою кількістю гілок. Тому для вирішення цієї проблеми часто використовуються евристичні методи, зокрема методи «зверху вниз», які включають у себе алгоритм CART, з фаз вирощування та обрізки дерева.

Існує кілька загальновживаних алгоритмів для навчання дерев рішень, які розпочинаються з кореневого вузла. Більшість з них, таких як C4.5 або ID3, складаються з двох основних етапів: вирощування дерева - це процес розгалуження дерева від кореневого вузла за допомогою відповідної змінної з набору даних, орієнтований на максимізацію приросту інформації, та його обрізка.

На малюнку 2.2 нижче пояснюється принцип роботи алгоритму навчання дерева рішень, який використовує псевдокод для рекурсивної побудови дерева зверху (від кореня) до низу.

TreePruning(S, T, y)

```
S - навчальна вибірка
T - дерево, до якого застосовується обрізка
y - цільова змінна

DO
    Обрати вузол t дерева T, такий що його відсікання забезпечить
        максимальне збільшення значення деякого критерю оцінки
    IF t ≠ ∅ THEN T = pruned(T, t)
UNTIL t = ∅
RETURN T
```

Рисунок 2.2, аркуш 1 - Загальний опис процесу навчання дерева рішень

TreeGrowing(S, A, y, SplitCriterion, StoppingCriterion)

```
S - навчальна вибірка
A - набір вхідних змінних
y - цільова змінна
SplitCriterion - критерій розщеплення дерева
StoppingCriterion - критерій зупинки процесу "вирощування" дерева

Створення нового дерева T з єдиним кореневим вузлом

IF StoppingCriterion(S) THEN
    позначити T листом з міткою-найчастішим значенням y з набору S
ELSE
    ∀ aᵢ ∈ A знайти a, що забезпечує найкращий SplitCriterion(aᵢ, S)
    Розбиття вузлу t за значеннями змінної a
    FOR кожного значення vᵢ з a:
        Subtreeᵢ = TreeGrowing(vᵢ,S, A, y)
        З'єднати кореневий вузол tT з Subtreeᵢ з краями поміченими vᵢ
    END FOR
END IF
RETURN TreePruning(S,T,y)
```

Рисунок 2.2, аркуш 2 - Загальний опис процесу навчання дерева рішень

Приріст інформації, або втрата ентропії, є критерієм, який використовується в алгоритмах індукції для вибору розділень у дереві рішень. Ентропія Шеннона для системи з N можливими станами визначається формулою (2.6).

$$H = -\sum_{i=1}^{N} p_i \log_2 p_i \qquad (2.6)$$

де p_i – імовірність знаходження системи в і-му стані.

Ентропія є мірою невизначеності випадкової величини і використовується для визначення зміни невизначеності цільової змінної в контексті навчальних даних. Це означає, що ентропія вимірює, наскільки інформативним є певний атрибут або розділення даних за цим атрибутом. (2.7).

$$IG = -\sum_{y \in Y} p_y \log_2 p_y + \sum_{i \in S} \frac{|X_i|}{|X|} \sum_{y \in Y} p_{iy} \log_2 p_{iy} = \qquad (2.7)$$

$$= H_X(Y) - H_X(Y|S)$$

де X - навчальна вибірка;

Y - множина станів цільової змінної.

Чим більше зменшується невизначеність, тим більше інформації ми можемо отримати про залежність між змінними. Тому алгоритм спочатку розбиває дерево за ознакою, яка найбільше зменшує невизначеність цільової змінної.

Однак, критерій приросту інформації може бути вразливим до перенавчання моделі, особливо коли маємо справу з розгалуженнями, що мають велику кількість можливих значень. Це може призвести до того, що модель буде надто чутливою до навчальних даних і може погано узагальнювати залежності для нових даних.

Критерій Джині використовується в алгоритмі CART як критерій для розгалуження дерева. Цей критерій оцінює різницю вказує на ймовірність того, що випадково вибраний елемент з заданого набору буде неправильно класифікований, якщо його класифікувати випадковим чином. Критерій Джині дорівнює нулю, коли усі шляхи, що виходять з цього вузла дерева, ведуть до однієї категорії цільової змінної.

$$Gini(Y) = 1 - \sum_{ygY} p^2_y \qquad (2.8)$$

Наприклад, ритерій Джині можна розглядати як альтернативу ентропії для дерев рішень.

Після розгалуження вузлів дерева важливим кроком є його відсікання (pruning). Оскільки рекурсивне розгалуження може привести до перенавчання, процедура відсікання спрямована на вирішення цієї проблеми. Вона може замінювати піддерево більш простою версією або листовим вузлом.

Наприклад, алгоритм CART використовує а початковому етапі в процедурі відсікання за вартістю складності, що відома як cost complexity pruning, формується послідовність дерев рішень. $T0$, $T1$, . . . Tk, де $T0$ є вихідним деревом до проведення операції відсікання, а Tk складається тільки з кореневого вузла. На наступному етапі одне з цих дерев обирається як обрізане з урахуванням показника помилки узагальнення. Дерево Ti+1 Для отримання вони замінюють одне чи кілька піддерев батьківського дерева Ti відповідними листовими вузлами. Вибираються піддерева для відсікання на основі найменшого збільшення помилки при видаленні кожного листа.

$$error\ complexity = \frac{s(pruned(T,t),S) - s(T,S)}{|leaves(T)| - |leaves(pruned(T,t))|} \qquad (2.9)$$

де $\varepsilon(T, S)$- помилка дерева T на наборі S;

$|leaves(T)|$ - кількість листків у T;

pruned(T, t) - дерево, отримане заміною вузла t з дерева T відповідним листовим вузлом.

Обирається найкраще обрізане дерево за оцінками кожного з обрізаних дерев за похибкою узагальнення. Зазвичай відсікання за вартістю складності проводять на навчальній вибірці, а потім перевіряють результати моделі на тестовій вибірці. У випадку обмеженого обсягу даних також може використовуватися стратегія крос-валідації.

Ансамблеві моделі

Ансамблевий підхід у машинному навчанні передбачає навчання різних алгоритмів для вирішення однієї задачі та їх подальше об'єднання Для отримання кращих результатів використовується комбінація декількох алгоритмів. Ідея полягає в тому, що результат роботи кількох алгоритмів зазвичай є точнішим, ніж результат роботи лише одного алгоритму. Цю концепцію підтверджує теорема Кондорсе "про жюрі присяжних", згідно з якою ймовірність правильного прийняття рішення зростає зі збільшенням числа оцінюючих та наближається до одиниці, якщо кожен має незалежну думку і ймовірність прийняття вірного рішення для кожного з них перевищує 0.5.

$$\mu = \sum_{i=m}^{N} C_N^i p^i (1-p)^{N-i} \qquad (2.10)$$

де N - кількість присяжних,

p - ймовірність правильного рішення присяжного,

μ - ймовірність правильного рішення жюрі загалом,

m - мінімальна більшість жюрі, $m = [\frac{N}{2}] + 1$,

C_N^i — кількість комбінацій з N по i.

Формула (2.10) показує, що якщо p > 0.5 і, відповідно, μ > p, то при $N \to \infty$ $\mu \to 1$.

В машинному навчанні "членами жюрі" є "слабкі учні". Вони використовуються як компоненти для створення ансамблевої моделі. Існують три основні типи ансамблевих моделей за стратегією поєднання базових алгоритмів:
1. Стекінг: у цьому методі слабкі учні представлені двома чи більше алгоритмами, результати яких об'єднуються для прогнозу.

2. Бегінг (bagging): для незалежного навчання слабких учнів застосовується один алгоритм, і прогноз формується шляхом усереднення результатів цих учнів.

3. Бустинг (boosting): у цьому методі слабкі моделі навчаються почергово для коригування помилок попередніх моделей.

Бегінг один з найпростіших методів створення ансамблевих моделей. Основний принцип бегінгу полягає у використанні техніки бутстреп. За допомогою бутстреп методу з вихідного набору даних випадковим чином формуються кілька навчальних підвибірок. Для кожної такої підвибірки будується окремий навчальний алгоритм (базовий класифікатор або регресор). Результати базових моделей потім об'єднуються, наприклад, шляхом голосування для класифікації або усереднення для регресії, утворюючи остаточну ансамблеву модель прогнозування.
Бутстреп метод працює наступним чином: на вхід подається вибірка даних X величини N. З цієї вихідної вибірки випадковим чином з поверненням обираються N об'єктів. Тобто, кожен об'єкт може бути обраний більше одного разу з однаковою ймовірністю 1/N. Цей процес формування нової випадкової підвибірки повторюється кількістю значення M, після чого генеруються M різних підвибірок X1, X2, ... XM. Отже, бутстреп є методом створення множини нових навчальних вибірок шляхом випадкового відбору спостережень з вихідного набору даних з поверненням. Ці згенеровані підвибірки потім використовуються для побудови ансамблю моделей.
Перший крок бегінгу - генерація підвибірок методом бутстрепу ($X1, X2, \ldots XM$). Для кожної підвибірки розвивають різні класифікатори $ai(x)$, які можуть бути деревами рішень або моделями регресії. Останнім класифікатором є модель (2.11), яка об'єднує результати слабких моделей за допомогою усереднення.

$$a(x) = \frac{1}{M}\sum_{i=1}^{M} a_i(x) \qquad (2.11)$$

Ансамблеві моделі дозволяють зменшити ризик перенавчання навченої моделі, що призводить до зменшення різниці у помилках моделей, навчених на різних підмножинах даних. Це допомагає уникнути перенавчання. Наприклад, розглянемо задачу регресії з базовими алгоритмами $b1(x), \ldots, bn(x)$. Якщо ясно, що існує істинна функція відповіді y(x) для всіх об'єктів, на яких задано розподіл P. Тому помилку кожної функції регресії можна представити як (2.12).

$$\varepsilon_i = b_i(x) - y_i(x), i = 1, \ldots, n \qquad (2.12)$$

А середньоквадратична помилка базових алгоритмів – формулою 2.13 відповідно.

$$E_1 = \frac{1}{n} E_x \sum_{i=1}^{n} \varepsilon_i^2(x) \qquad (2.13)$$

Припустимо, що помилки незміщені та некорельовані. Ми продовжимо будувати функцію(2.14), що поєднує базовий алгоритм за допомогою усереднення.

$$a(x) = \frac{1}{n}\sum_{i=1}^{n} b_i(x) \qquad (2.14)$$

Її середньоквадратична помилка представлена формулою (2.15).

$$E_n = E_x\left(\frac{1}{n}\sum_{i=1}^{n} b_i(x) - y(x)\right)^2 = E_x\left(\frac{1}{n}\sum_{i=1}^{n} \varepsilon_i(x)\right)^2 =$$

$$= \frac{1}{n^2} E_x\left(\sum_{i=1}^{n} \varepsilon_i^2(x) + \sum_{j\neq i} \varepsilon_i(x)\varepsilon_j(x)\right) = \frac{1}{n} E_1 \qquad (2.15)$$

Таким чином, середній квадрат помилки зменшено у n разів, іншими словами - зменшено дисперсію класифікатора, адже загальну помилку можна розкласти за формулою (2.16).

$$Err(\vec{x}) = E\left[(y - f(x))^2\right] =$$

$$= (f - E[f])^2 + Var(f) + \sigma^2 \qquad (2.16)$$

Ефективність бегінгу пояснюється так, що базові алгоритми, навчені на різних підвибірках, мають велику різноманітність, щоб їхні помилки компенсували одна одну під час голосування. Цей підхід також дозволяє уникнути потрапляння об'єктів-викидів у навчальні розробки.

Випадковий ліс

Випадковий ліс — це мобільний ансамбль алгоритму машинного навчання, який можна використовувати для різних аспектів, пов’язаних з класифікацією та регресією. Ліс представляє собою групу незалежних дерев рішень, які потім об'єднуються для зменшення ризику перенавчання та отримання більш точних прогнозів для даних.

В розділі 2.1.2 використовується метод початкового завантаження для побудови зразка, на якому навчається базовий алгоритм (дерево рішень). Кількість дерев у лісі K є налаштовуваним параметром моделі і може бути вибрана на підставі досвіду або за допомогою методів оптимізації. Після навчання лісу для класифікації нових записів кожне дерево класифікує об'єкт до одного з класів., і в результаті вибирається клас, до якого належить об'єкт, визначається більшістю голосів дерев. Основна відмінність між випадковим лісом і класичним пакетуванням полягає в покращеному алгоритм побудови дерев рішень. Під час кожного відгалуження у відповідному вузлі дерева вибирається набір випадкових величин, за якими буде проводитися наступна фаза індукції. Цей додатковий елемент випадковості допомагає краще задовольнити вимоги моделі ансамблю щодо базової алгоритмічної різноманітності.

Методи, що використовуються у випадкових лісах, можна також застосовувати Однією з ключових переваг випадкового лісу є можливість порівняно легко підбирати важливість ознак, що дозволяє автоматично відбирати найбільш інформативні ознаки ефективність у роботі з великою кількістю вхідних змінних. Швидкість роботи також є однією з переваг випадкових лісів. Проте серед їх недоліків слід відзначити схильність моделей випадкового лісу до перенавчання.

2.2 Бустинг

Бустинг - це метод ансамблювання, що поєднує набір слабких моделей в один сильний класифікатор Модель отримується шляхом агрегації результатів цих моделей з урахуванням їх ваги, правила з окремих класифікаторів об'єднуються для формування сильного правила прогнозування.

Бустинг використовується у методі градієнтного бустингу (GBM), де зазвичай в якості слабких моделей використовуються дерева рішень. Класичний алгоритм GBM складається з наступних кроків:

1. Початкове значення GBM ініціалізується константою.

$$f_0 = arg\ min \sum_{i=1}^{n} L(y_i, \gamma)$$

2. На кожній ітерації

- Підрахувати псевдо залишки r_t

$$r_{it} = -\left|\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right|_{f(x)=f(x)}, i = 1, \ldots, n$$

- Побудувати новий базовий алгоритм $h_t(x)$ на псевдо залишках $\{(x_i, r_{it})\}_{i=1,\ldots,n}$
- Знайти оптимальний коефіцієнт ρ_t при $h_t(x)$ відносно вихідної функції втрат
- Зберегти

$$\hat{f}_t = \rho_t \cdot h_t(x)$$

- Оновити поточне наближення

$$f(x) \leftarrow \sum_{i=0}^{t} f_i(x)$$

3. В результаті отримано модель

$$f(x) = \sum_{i=0}^{M} f_i(x)$$

Зараз дуже популярна покращена версія градієнтного посилення XGBoost із екстремальним градієнтним посиленням, випадкова змінна використовується для вибору першої точки розбиття в одному класифікаторі, а інший класифікатор може вибрати іншу змінну для своєї першої точки розбиття в публічних бібліотеках.

Різниця між цим методом і класичним посиленням градієнта полягає в тому, що він використовує метод Ньютона у місці функцій навпроти градієнтного зпуску.

Для цього потрібна друга похідна функції втрат [11].

XGBoost також використовує методи регуляризації для контролю складності моделі, щоб уникнути перенавчання (наприклад, обмеженням кількості кінцевих вузлів у дереві).

Крім рандомізації кількості рядків, XGBoost також має можливість рандомізувати кількість функцій.

Light Gradient Boosting Machine (LGBM) — ще один варіант посилення градієнта, який останнім часом все частіше використовується.

Дослідники Microsoft успішно створили рішення, яке використовує XGBoost, одночасно використовуючи оптимізований алгоритм для отримання дерев рішень на основі гістограм.

Однією з особливостей LGBM, яка впливає на швидкість обчислень шляхом використання спеціального підходу до зменшення розміру навчальних вибірок перед їхнім використанням для навчання ансамблю дерев рішень. У міжнародній літературі ця евристика називається односторонньою вибіркою на основі градієнта.

Ідея полягає в тому, щоб зменшити вибірку (зменшити кількість вибірок) об’єктів із малими градієнтами на кожному кроці посилення об'єкти з великими градієнтами (які мають високий вплив на функцію втрат) сприяють більшому навчанню дерева рішень.

Таким чином, під час навчання моделі набір даних із високими значеннями градієнта зберігається у вихідній формі, і лише підмножина набору даних, що залишився, вибирається випадковим чином.

Виберіть a з набору даних із великим градієнтом і виберіть b із випадкового набору даних із малим градієнтом.

Потім ми використовуємо множник 1-a, щоб обчислити приріст інформації під

час навчання дерева рішень на вибірковому наборі даних.

Згідно з дослідженням Microsoft, використання цієї евристики *b* може призвести до скорочення часу навчання вдвічі за рахунок невеликого зниження якості моделі [12]. Ще однією особливістю ЛГБТМ є набір ексклюзивних функцій, яка полягає в об'єднанні кількох розріджених функцій в одну. Малюнок 2.3 чудово ілюструє цю ідею.

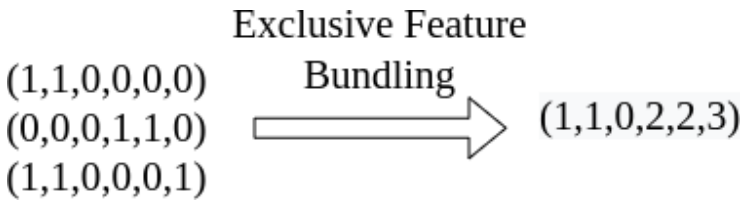


Рисунок 2.3 Принцип роботи евристики EFB

Оцінка якості роботи системи

Оцінка ефективності класифікатора є ключовою в машинному навчанні. Один із способів її здійснення - використання критеріїв оцінювання, які допомагають зрозуміти якість прогнозу та потребу у внесенні змін у рішення щодо обробки даних чи підбору гіперпараметрів.

Хоча існують інші критерії оцінки, такі як обчислювальна складність чи зрозумілість класифікатора, у цьому розділі ми зосередимося саме на критеріях оцінки прогностичної ефективності математичних моделей.

Підходи до розбиття датасету

Щоб навчити модель і оцінити її ефективність, потрібно визначити стратегію розбиття вихідного набору даних для подальшої роботи. Існують два основних підходи: розділення на навчальну та тестову вибірки та перехресна перевірка.

Ідея розділення зразків навчання та тестування (розподіл навчання та тестування), як випливає з назви підходу, полягає в тому, щоб певним чином розділити вихідний набір даних.

Отримана модель приймає тестові дані як вхідні дані, а метрики, описані далі в цьому розділі, оцінюють, наскільки добре модель узагальнює незалежні дані.

У той же час виникає дилема.

Збільшення розміру вашого тестового набору дозволяє точніше оцінити якість вашої моделі (враховуючи якомога більше випадків, коли ваша модель працює на невідомих даних).

При цьому необхідний навчальний набір також скорочується, що негативно позначається на підготовці моделі.

Перехресна перевірка, також відома як перехресна перевірка, — це набір методів для оцінки ефективності прогнозної моделі на реальних непідготовлених даних.

Одним із найбільш часто використовуваних методів перехресної перевірки є k-кратна перехресна перевірка.

Вибірку випадковим чином ділять на k рівних частин, а модель навчають і перевіряють k разів.

Частина K-1 використовується для навчання моделі, а частина, що залишилася, використовується для тестування щойно навченої моделі.

Тому кожна з k частин перевіряється лише один раз.

У результаті враховується середнє арифметичне отриманих оцінок якості моделі.

На малюнку 2.

4 показана ідея k-кратної перехресної перевірки з k=5.

Рисунок 2.4 K-fold cross-validation

Компанія Vesta надала учасникам змагання на Kaggle набори даних для навчання та тестування моделей. Учасники можуть використовувати крос-валідацію для оцінки якості моделі на навчальних даних перед порівнянням результатів на тестових даних з іншими учасниками. Таким чином, використовується комбінація крос-валідації та тестування на окремій вибірці.

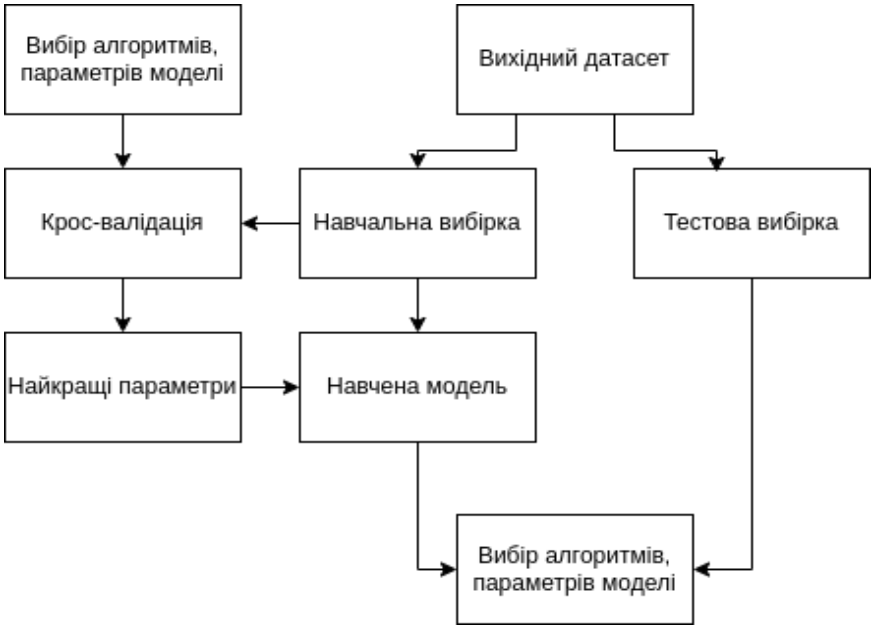


Рисунок 2.5 Стратегія навчання моделі

Точність класифікації

Точність - це один із критеріїв оцінювання якості класифікації моделі. Вона являє собою співвідношення кількості правильних передбачень, зроблених моделлю, до загальної кількості її передбачень. (2.17)

$$\frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \qquad (2.17)$$

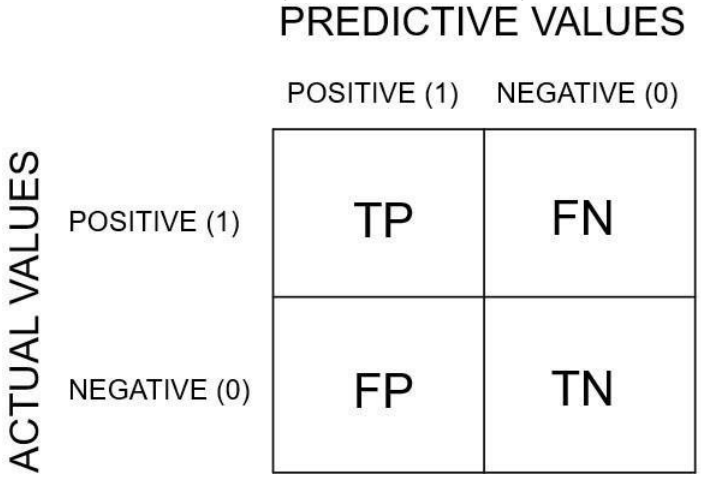
Хоча точність є простим і зрозумілим показником для оцінки класифікаційних моделей, вона має певні недоліки. При використанні цього критерію важливо усвідомлювати, що він може бути неадекватним для оцінювання роботи моделі на незбалансованих наборах даних. У випадках, коли один клас має значно більшу кількість прикладів, ніж інший (клас меншості), метрики точності можуть залишатися на високому рівні, незважаючи на можливо низьку прогностичну здатність моделі стосовно екземплярів класу меншості.

У випадках класифікації, наприклад, виявлення шахрайських операцій, часто спостерігається проблема незбалансованості даних, коли кількість випадків шахрайства в наборі даних значно менше, ніж легальних транзакцій. Це означає, що для оцінювання прогностичної сили системи виявлення шахрайських операцій необхідно використовувати іншу метрику оцінювання, яка була б більш придатною для незбалансованих наборів даних.

2.3 Матриця помилок

Матриця помилок є корисним інструментом для аналізу результатів класифікаційної моделі. Вона візуалізує продуктивність алгоритму, надаючи інформацію про кількість правильних та неправильних передбачень для кожного класу.

У матриці помилок відображається кількість істинно позитивних (True Positive - TP) передбачень, коли модель правильно визначила приналежність екземпляра до позитивного класу. Також показуються істинно негативні (True Negative - TN)



передбачення, коли модель коректно класифікувала екземпляр як негативний. Крім того, матриця відображає кількість хибно позитивних (False Positive - FP) і хибно негативних (False Negative - FN) передбачень. Це означає невірну класифікацію негативних екземплярів як позитивних і позитивних екземплярів як негативних.

Рисунок 2.6 Заповнення матриці помилок

Цей інструмент дозволяє виявити помилки першого (FN) і другого (FP) типів, а також розрахувати деякі метрики, які детальніше відображають характеристики Звідси можна зробити висновок, що шахрайські банківські операції є серйозною проблемою для сучасних платіжних систем. Така ситуація підкреслює необхідність розвитку інформаційних технологій для виявлення шахрайства. (2.18-2.20):

$$Precision = \frac{TP}{TP + FP} \quad (2.18)$$

$$True\ positive\ rate(Recall) = \frac{TP}{FN + TP} \quad (2.19)$$

$$False\ positive\ rate = \frac{FP}{TN + FP} \quad (2.20)$$

Показники ROC, AUC

Криві ROC (Receiver Operating Characteristic) є важливим інструментом для візуалізації та оцінки точності роботи класифікаційних моделей.Вони ілюструють співвідношення між часткою істинно позитивних прогнозів (TPR, True Positive Rate) та часткою хибно позитивних прогнозів (FPR, False Positive Rate) для різних порогових значень класифікатора.

ROC-крива відображає відношення між FPR (False Positive Rate) та TPR (True Positive Rate).Ідеальною точкою на цій кривій є координати (0, 1), що означає досконалу класифікацію, коли всі позитивні екземпляри правильно визначені, а негативні екземпляри не помилково класифіковані як позитивні.

ROC-крива може бути корисним інструментом для визначення оптимальних класифікаторів.

Зазвичай вважається, що класифікатор має прийнятну якість, якщо його ROC-крива лежить переважно або повністю вище діагональної прямої y=x. Чим далі ROC-крива відхиляється від цієї прямої до верхнього лівого кута графіка, тим вищою є якість класифікатора.

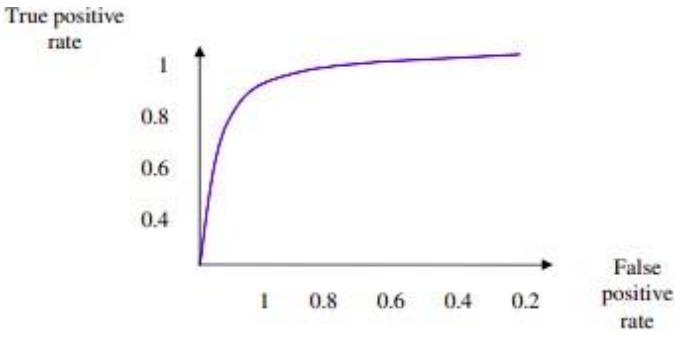


Рисунок. 2.7 Зображення ROC кривої

AUC (Area Under the Curve) - це площа під кривою ROC. Це узагальнена міра продуктивності бінарного класифікатора. Значення AUC = 1 означає, що модель здійснює ідеальну класифікацію вхідних даних, правильно розподіляючи всі позитивні та негативні екземпляри.

Важливо також враховувати, що значення AUC, близьке до 0.5, вказує на те, що прогнози моделі еквівалентні випадковому рівномірному розподілу в діапазоні від 0 до 1 і не залежать від вхідних даних. Іншими словами, така модель не має передбачувальної здатності, оскільки її прогнози рівноймовірно можуть бути як правильними, так і неправильними, не враховуючи ознаки вхідних екземплярів.

Отже, бажаним є отримати значення AUC максимально близьким до 1, що свідчитиме про високу якість класифікаційної моделі та її здатність ефективно розрізняти класи на основі вхідних даних.

2.4 Покращення результатів роботи моделей

Вибір відповідних алгоритмів машинного навчання для задачі бінарної класифікації є лише одним з факторів, що впливають на загальну якість моделі. Крім алгоритму, важливу роль відіграє належна підготовка вхідних даних та правильний підбір гіперпараметрів моделі. Розглянемо існуючі підходи до вирішення цих питань.

Попередня обробка даних може суттєво вплинути на продуктивність алгоритмів машинного навчання. Це включає такі етапи, як очищення даних від пропущених значень чи викидів, масштабування числових ознак, кодування категоріальних змінних тощо. Належна підготовка даних допомагає алгоритмам краще виявляти закономірності та покращує точність моделей.

Окрім підготовки даних, важливим аспектом є налаштування гіперпараметрів моделі. Гіперпараметри - це параметри, що не налаштовуються безпосередньо під час навчання моделі, але визначають її поведінку та архітектуру. Правильний підбір гіперпараметрів може значно покращити продуктивність моделі або, навпаки, призвести до погіршення результатів.

Отже, для отримання високоякісної моделі бінарної класифікації необхідно враховувати не лише сам алгоритм машинного навчання, а й ретельно підійти до питань підготовки даних та налаштування гіперпараметрів відповідно до характеристик задачі та вхідних даних.

Конструювання змінних

При вирішенні задачі виявлення шахрайських операцій з банківськими картками, набори даних про транзакції, на яких тренуються алгоритми, зазвичай містять схожий набір ознак. Це пояснюється існуванням міжнародних стандартів фінансової звітності та типових угод про обробку даних клієнтів банків і платіжних систем. Велике значення має якість даних та їх правильна обробка для успішного навчання алгоритмів машинного навчання. Для цього часто застосовуються методи трансформації наявних ознак та створення нових на їх основі.

Процес створення або вибору ознак (feature engineering) може бути розбитий на такі етапи:

1. Оцінка поточних ознак, наприклад, через мозковий штурм, для визначення їх придатності для використання в навчанні моделі.

2. Виходячи з результатів попереднього етапу, ухвалення рішень про додавання нових ознак або трансформацію існуючих. Це може включати числові перетворення, кодування категоріальних змінних, агрегування ознак або створення нових ознак на основі знань предметної області.

3. Оцінка ефективності виконаних перетворень за допомогою крос-валідації під час навчання моделей.

4. Прийняття рішення про впровадження запропонованих модифікацій на підставі результатів крос-валідації.

Цей процес можна повторювати ітеративно до тих пір, поки всі гіпотези щодо можливих трансформацій даних не будуть вичерпані і перевірені.

Налаштування гіперпараметрів моделей

Гіперпараметр моделі машинного навчання — це параметр, який визначає процес навчання моделі, але не налаштовується безпосередньо під час самого навчання. Прикладами гіперпараметрів можуть бути швидкість навчання, кількість шарів у нейронній мережі тощо.

Оптимальна робота багатьох алгоритмів машинного навчання значно залежить від того, як точно налаштовані їхні гіперпараметри.

Для конкретної задачі необхідно визначити простір пошуку гіперпараметрів, тобто набір гіперпараметрів та діапазонів їх можливих значень, які потрібно розглянути. Також слід обрати евристичний метод, який використовуватиметься для пошуку оптимальних комбінацій гіперпараметрів у заданому просторі пошуку.

Нехай A - алгоритм машинного навчання з H гіперпараметрами, а простір пошуку гіперпараметрів позначається як Θ = Θ1 × Θ2 × ... × Θh, де Θi - множина можливих значень і-го гіперпараметра. Тоді набір гіперпараметрів алгоритму A можна представити у вигляді вектора (θ1, θ2, ..., θh), де θi є елементом множини Θi. Має вигляд (2.21)

$$\theta =< \theta_1, \theta_2, \ldots, \theta_h >,\ \theta_i \epsilon\ \Theta_i \qquad (2.21)$$

Роботу алгоритму машинного навчання з набором гіперпараметрів θ на

конкретному навчальному наборі даних X з цільовою змінною Y можна оцінити за допомогою функції втрат $L(Y, \hat{f}(X, \theta))$, де $\hat{f}(X, \theta)$ - модель, побудована алгоритмом з гіперпараметрами θ на даних X.

Таким чином, задача налаштування гіперпараметрів зводиться до пошуку такого набору θ*, який мінімізує функцію втрат:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}}\, L(Y, \hat{f}(X, \theta))$$

Іншими словами, необхідно знайти оптимальні значення гіперпараметрів θ* з простору пошуку Θ, які забезпечують найменше значення функції втрат $L(Y, \hat{f}(X, \theta))$ на навчальних даних.

Для вирішення цієї оптимізаційної задачі використовуються різні методи оптимізації, такі як випадковий пошук, граділізований спуск, байєсівська оптимізація та інші. Метод Нелдера-Міда (або метод симплексного спуску) є широко використовуваним алгоритмом для вирішення задачі нелінійної оптимізації прямим пошуком. Одна ітерація цього алгоритму складається з наступних кроків:

1. Сортування контрольних точок x1, x2, ..., xn+1 за значеннями цільової функції f у порядку зростання: $f(x1) \leq f(x2) \leq ... \leq f(xn+1)$.

2. Відбиття. Обчислюється точка відбиття xr відносно центроїда $\overline{x}$ всіх точок, крім xn+1, за формулою: $xr = \overline{x} + \alpha(\overline{x} - xn+1)$. Якщо f(xr) краще за f(x1), але гірше за f(xn), замінюємо xn+1 на xr.

3. Розширення. Якщо $f(xr) < f(x1)$, обчислюється точка розширення $xe = \overline{x} + \beta(xr - \overline{x})$. Якщо $f(xe) < f(xr)$, $xn+1$ замінюється на xe, інакше - на xr.

4. Зовнішнє скорочення. Якщо $f(xn) \leq f(xr) < f(xn+1)$, обчислюється xoutside contraction $= \overline{x} + \gamma(xr - \overline{x})$. Якщо $f(xoc) \leq f(xr)$, $xn+1$ замінюється на xoc, інакше переходимо до кроку 6.

5. Внутрішнє скорочення. Якщо $f(xr) \geq f(xn+1)$, обчислюється xinside contraction $= \overline{x} - \gamma(xr - \overline{x})$. Якщо $f(xic) \leq f(xn+1)$, $xn+1$ замінюється на xic, інакше переходимо до кроку 6.

6. Стягування. Для $2 \leq i \leq n+1$ обчислюється $xi = x1 + \delta(xi - x1)$.

Ітераційний цикл завершується, коли виконується певний критерій збіжності, наприклад, коли стандартне відхилення значень функції в поточному симплексі стає меншим за заданий поріг.

2.5 Висновки до розділу 2

У цьому розділі було проаналізовано алгоритми та методики виявлення шахрайських банківських транзакцій. Додатково були проаналізовані метрики для оцінки ефективності класифікаційних моделей і описано процес налаштування їх гіперпараметрів.

На основі проведеного аналізу методів для класифікації транзакцій, у практичній частині роботи буде здійснено навчання описаних у розділі алгоритмів машинного навчання. При цьому будуть використані описані евристики та методи, такі як конструювання ознак, крос-валідація, налаштування гіперпараметрів тощо, з метою покращення якості

побудованих класифікаційних моделей для виявлення шахрайських операцій.

Отже, теоретичні відомості з цього розділу стануть підґрунтям для практичної реалізації та дослідження ефективності різних алгоритмів машинного навчання щодо поставленої задачі бінарної класифікації транзакцій.

РОЗДІЛ 3. АНАЛІЗ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

У практичній частині роботи для обробки набору даних про платіжні транзакції та побудови моделей класифікації використовувалася мова програмування Python у поєднанні з такими відкритими бібліотеками, як Pandas, SciPy, Scikit-learn. Основним середовищем розробки слугував Jupyter Notebook.

У цьому розділі детально описані основні етапи створення класифікаційних моделей для виявлення шахрайських транзакцій, включаючи підготовку даних, конструювання ознак, навчання алгоритмів машинного навчання та налаштування їх гіперпараметрів.

Також у розділі запропоновано інтегрувати побудовані класифікаційні моделі в інформаційну технологію для автоматизованого виявлення шахрайських операцій у режимі реального часу на основі нових надходжень транзакційних даних.

Отже, розділ охоплює практичну реалізацію підходів, описаних у теоретичній частині, та пропонує шляхи застосування розроблених моделей машинного навчання в інформаційній технології для ефективного виявлення шахрайських банківських транзакцій. У практичній частині роботи для обробки набору даних про платіжні транзакції та побудови моделей класифікації використовувалася мова програмування Python у поєднанні з такими відкритими бібліотеками, як Pandas, SciPy, Scikit-learn. Основним середовищем розробки слугував Jupyter Notebook.

У цьому розділі детально описані основні етапи створення класифікаційних моделей для виявлення шахрайських транзакцій, включаючи підготовку даних, конструювання ознак, навчання алгоритмів машинного навчання та налаштування їх гіперпараметрів.

Також у розділі запропоновано інтегрувати побудовані класифікаційні моделі в інформаційну технологію для автоматизованого виявлення шахрайських операцій у режимі реального часу на основі нових надходжень транзакційних даних.

Отже, розділ охоплює практичну реалізацію підходів, описаних у теоретичній частині, та пропонує шляхи застосування розроблених моделей машинного навчання в інформаційній технології для ефективного виявлення шахрайських банківських транзакцій.

2.1 Опис початкового набору даних

Набір даних, який розглядається в цій роботі, надано компанією Vesta Corporation, яка безпосередньо спеціалізується на наданні платіжних послуг.

Тому набір даних містить різні характеристики кожної транзакції.
Нижче наведено опис кожної змінної.
1) TransactionDT – кількість часу, що минув від системного набору «нуля» до виконання транзакції.
2) TransactionAMT – сума транзакції в доларах США.
3) ProductCD – Код платного продукту.
4) Картка1 – Картка6 – дані картки, такі як тип, категорія, назва банку, країна тощо.
5) Адреса – адреса виставлення рахунку, регіон і країна.
6) dist – адреса транзакції, відстань між IP-адресами.
7) P_ (R_)emaildomain – домен електронної пошти відправника (P) і одержувача (R).
8) DeviceType – тип пристрою, який використовувався під час транзакції.
9) DeviceInfo – інформація про пристрій, така як його бренд та агент користувача браузера.
10) isFraud – змінна, яка вказує, чи була транзакція шахрайською.
11) Від C1 до C14 – різноманітні показники, включаючи кількість адрес, пов'язаних з цією картою. Конкретні значення цих змінних не розкриті постачальником даних.
12) D1–D15 – змінні, які вказують на часові інтервали між різними подіями та транзакціями. Наприклад, кількість днів між транзакціями.
13) M1-M9 – змінні, що представляють різноманітні збіги, наприклад, ім'я або адресу, як вказано на картці.
14) Vxxx – функції, розроблені компанією Vesta, які включають рейтинги та інші зв'язки між об'єктами.
15) id_12 – id_38 – це інформація про підключення до мережі (IP, ISP, проксі тощо) та цифровий підпис (UA/браузер/ОС/версія тощо), яка пов'язана з транзакцією.
Згідно з описом запису від Vesta, логіка позначення транзакції як шахрайська, коли система отримує сповіщення про зняття цього платежу та наступну транзакцію, яка відбувається за обліковим записом користувача, адресою електронної пошти або платежем, і ця транзакція вважається шахрайською (isFraud=1).
Адреси, безпосередньо пов’язані з цими атрибутами, також позначаються як недійсні.
Якщо жоден із зазначених вище звітів не буде створено та не виявлено протягом 120 днів, транзакція вважається законною (isFraud=0).
Vesta надала два окремі файли: навчальний набір даних та тестовий набір даних.
Тестові дані використовуються для оцінки та порівняння результатів моделей між учасниками на платформі Kaggle.
Загалом Початковий набір даних містить 434 змінні, з яких 340 є так званими V-змінними, які Vesta розрахувала для існуючих транзакцій.
Це означає, що необхідно провести докладний аналіз змінних та їх зв'язків і, можливо, вирішити скоригувати розмір набору даних.
Початковий огляд даних

Для створення моделі з високою точністю прогнозування необхідно докладно проаналізувати дані.

На початковому етапі роботи з наданим датасетом виникла проблема великого обсягу даних, загальний розмір csv-файлів становив близько 1.3 ГБ. Для прискорення обробки даних було впроваджено функцію оптимізації, що використовує більш ефективні типи даних для представлення змінних із метою економії пам'яті.
Наступним кроком буде аналіз вихідного набору даних, який включатиме:

1. Перевірка змінних на відсутність значень.
2. Оцінку збалансованості класів (цільової змінної).
3. Аналіз впливу незалежних змінних на цільову змінну (ознаку шахрайства).

Такий розвідувальний аналіз даних є необхідним етапом для виявлення потенційних проблем та особливостей датасету, які потрібно врахувати під час підготовки даних та подальшого навчання моделей машинного навчання.

Аналіз відсутніх значень.

Коли я перевірив набір даних на відсутність значень, я отримав такі результати:
Для категоріальних змінних:
- Більшість "змінних M", за винятком M5, мають пропущені значення.
- Поля id_07, id_08, а також id_21-27 містять значну кількість порожніх записів. - id_01, id_12, map1 , map2 містить кілька пропущених значень Відсутні значення для розподілених категоріальних змінних показано на рисунку 3.1.

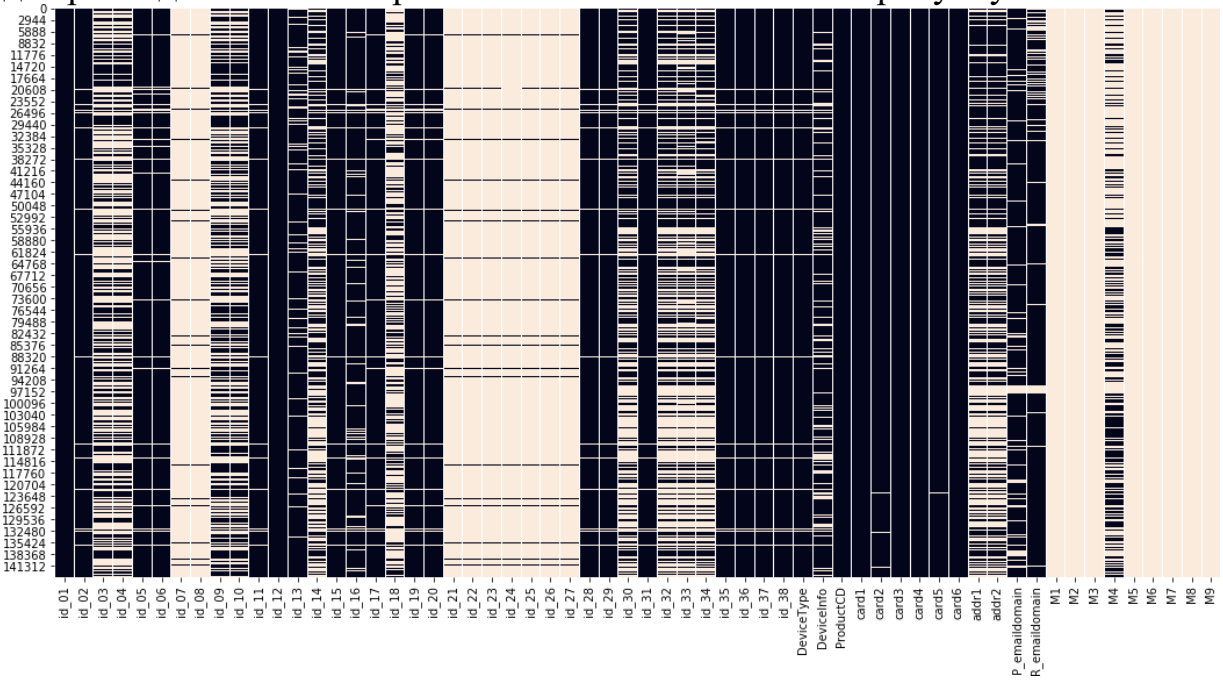


Рисунок 3.1

Для неперервних змінних:
- Основна інформація (час, розмір, код продукту) представлена повністю.
- dist1 і dist2 мають багато пропущених значень.
- Змінна C може бути повністю представленою.

- Змінні D дуже розріджені, за винятком D1.

На малюнку 3.2 показано розподіл пропущених значень для неперервних змінних.

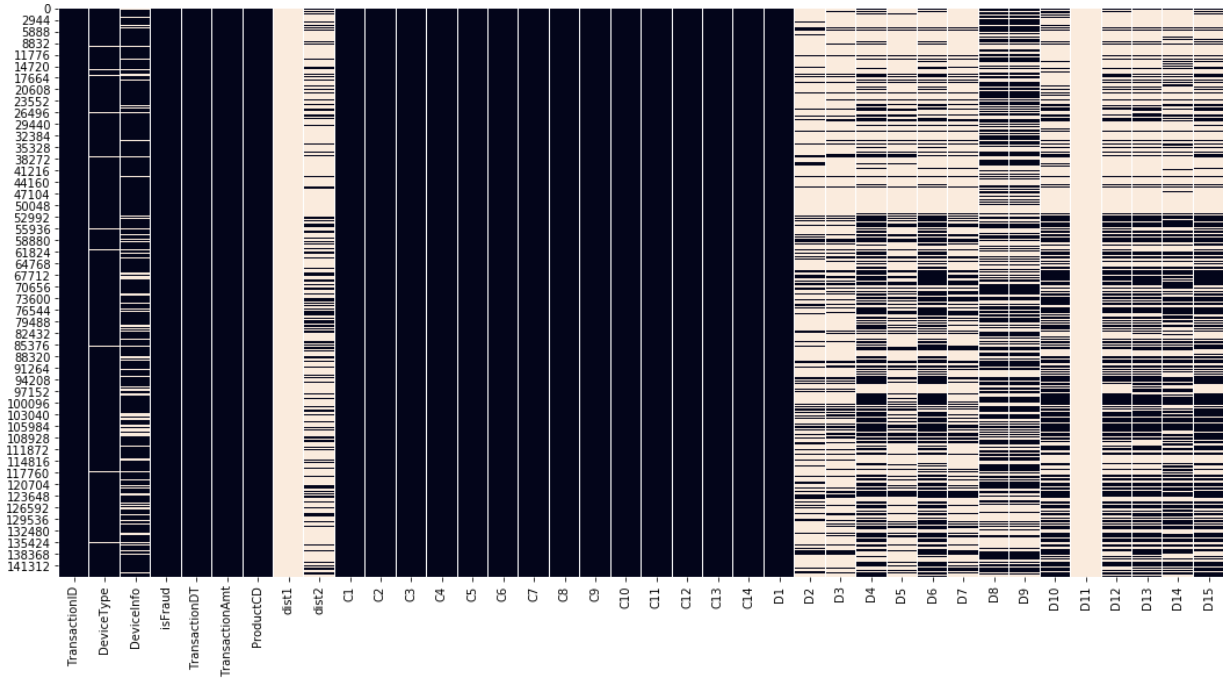


Рисунок 3.2

Щодо даних, наданих Vesta, ми спостерігаємо деякі закономірності в прогалинах у цих стовпцях (наприклад, прогалини у V322-399 ідентичні).

Це може свідчити про наявність мультиколінеарності в наборі даних.

Обрання "змінної V" може позитивно підвищити точність прогнозування майбутніх моделей.

Відсутні значення для цих змінних показані на малюнку 3.3.

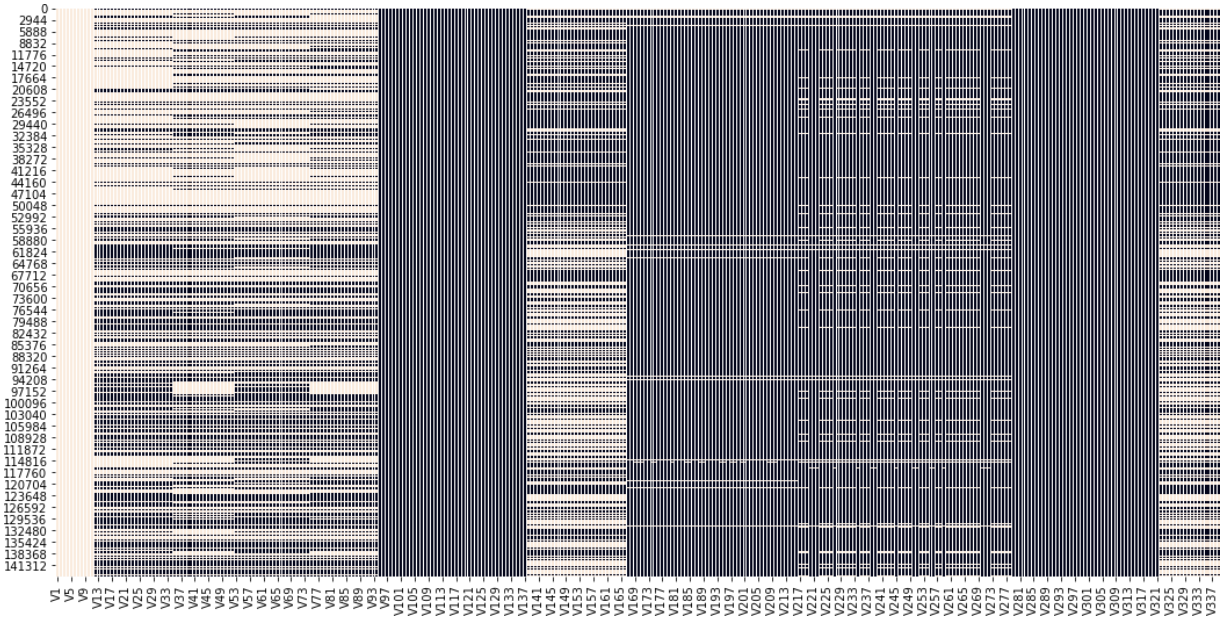


Рисунок 3.3 Пропущені значення змінних від компанії Vesta.

Деякі змінні датасету було ретельно проаналізовано для вивчення їх поведінки.

Цікавим спостереженням є залежність між статусом транзакції (шахрайська чи легальна) та типом оплаченого продукту. Згідно з рисунком 3.4, шахрайські операції дещо частіше відбуваються для транзакцій, пов'язаних з продуктами групи C.

Виявлення таких залежностей між незалежними змінними та цільовою змінною під час розвідувального аналізу даних є важливим, оскільки дозволяє краще зрозуміти природу даних та особливості, що можуть впливати на точність класифікаційних моделей. Ця інформація може бути корисною для подальшого конструювання ознак та інтерпретації результатів моделювання.

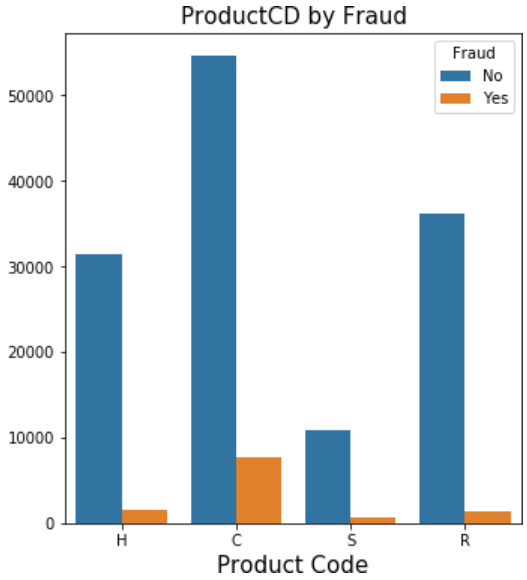


Рисунок 3.4 Залежність виявлення шахрайства від типу товару у транзакціях.

Якщо включити дані про суму транзакції у цю діаграму, ми побачимо на малюнку 3.5, що продукт C є в середньому продуктом з найнижчою ціною.

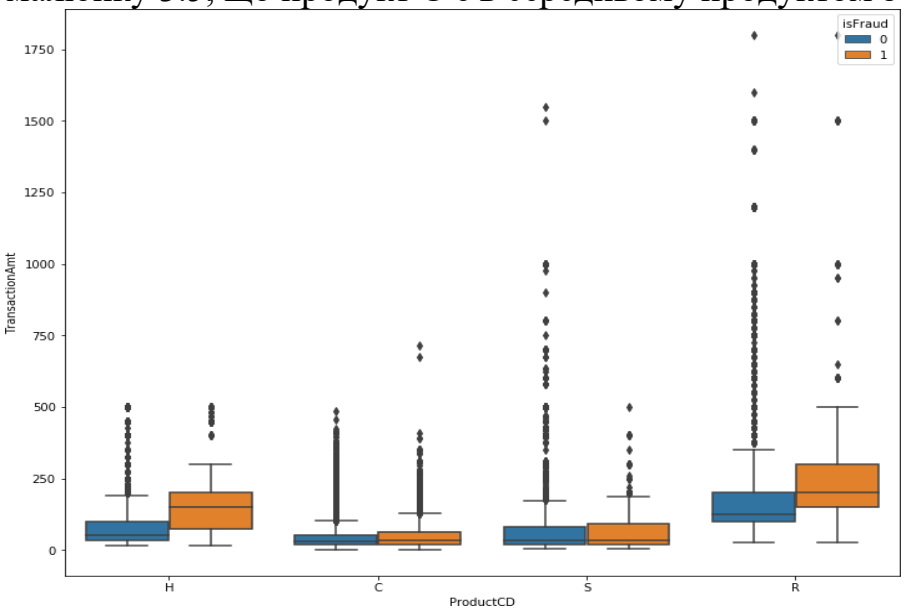


Рис. 3.5 Залежність від шахрайства від суми та товару транзакції.

2.2 Змінна TransactionDT

Змінна `transactionDT` містить інформацію про дату та час кожної транзакції. Аналіз цих змінних ускладнюється тим, що вони відображають час, що пройшов з певної початкової дати або події(мітки часу), не з 1 січня 1970 року, як це типово для дат у більшості поширених мов програмування, а з випадкового значення з цього моменту.

Опорна точка допомагає врахувати циклічні або повторювані зміни в моделі.

На малюнку 3.6 показано, що транзакції з навчального та тестового наборів даних відбуваються в різні періоди, які не перетинаються.

При цьому всі «тестові» операції матимуть пізнішу дату, ніж «навчальна».

При цьому всі транзакції виконуються протягом 396 днів, або приблизно 13 місяців.

Якщо розглянути стрибки на обох кінцях графіка, можна зробити припущення, що вони спричинені святковими періодами, наприклад, Різдвом та Чорною п'ятницею, коли зазвичай збільшується кількість покупок. Давайте перевіримо цю гіпотезу.

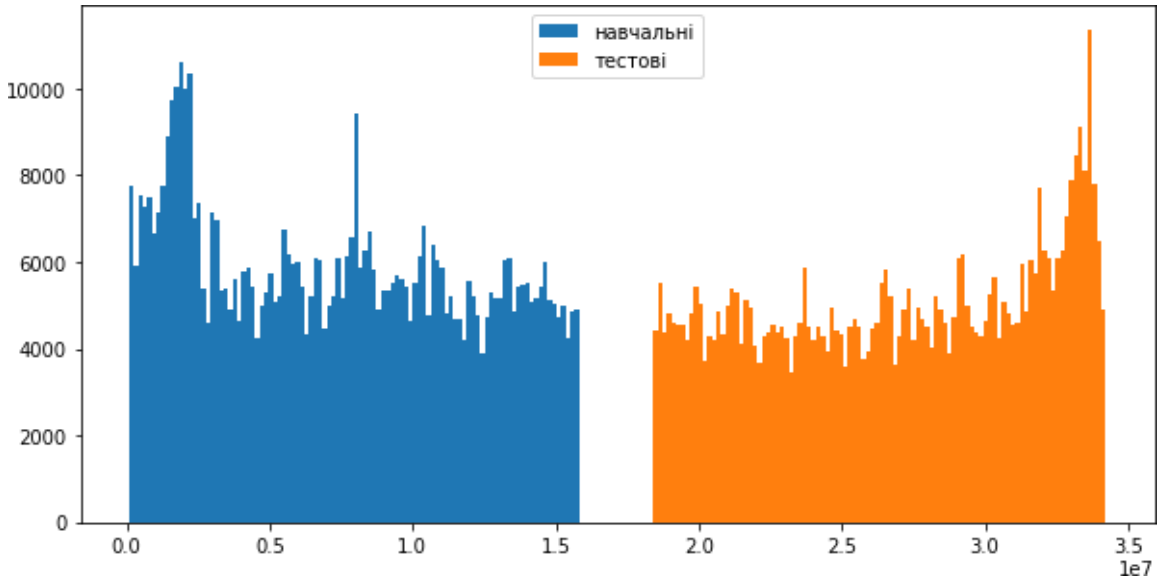


Рисунок 3.6 Дата транзакції в навчальній та тестовій вибірках представлена на графіку.

Мінімальне значення змінної становить 86400, що відповідає кількості секунд у добі, оскільки TransactionDT вимірюється в секундах. Давайте подивимося на рік, у якому відбулася транзакція.

У приклад візьмемо змінну "deviceInfo". Один з приладів є Samsung Galaxy S8 (SAMSUNG SM-G892A Build/NRD90M), який, за даними Вікіпедії, був випущений у США 21 квітня 2017 року. У таблиці 3.1 наведені операції, здійснені на цьому телефоні на другий день зазначеного періоду.

Можна припустити, що ця операція відбулася після квітня 2017 року.

Таблиця 3.1 Пристрої, використані в транзакціях ухронологічному порядку

DeviceInfo	TransactionDT
SAMSUNG SM-G892A Build/NRD90M	86506
iOS Device	86535
Windows	86549
MacOS	86620
SM-G930V Build/NRD90M	87445

В результаті бачимо, день зв еликою кількістю транзакцій у наборі даних було 395 транзакцій на обліковий запис.

Щоб перевірити наші гіпотези щодо святкового сезону, ми використовуємо дані Google про покупки в США з 1 квітня 2017 року по 31 грудня 2018 року.

Ці дані показані на рисунку 3.7 і демонструють, що пік активності спостерігався у листопаді 2017 та 2018 років.

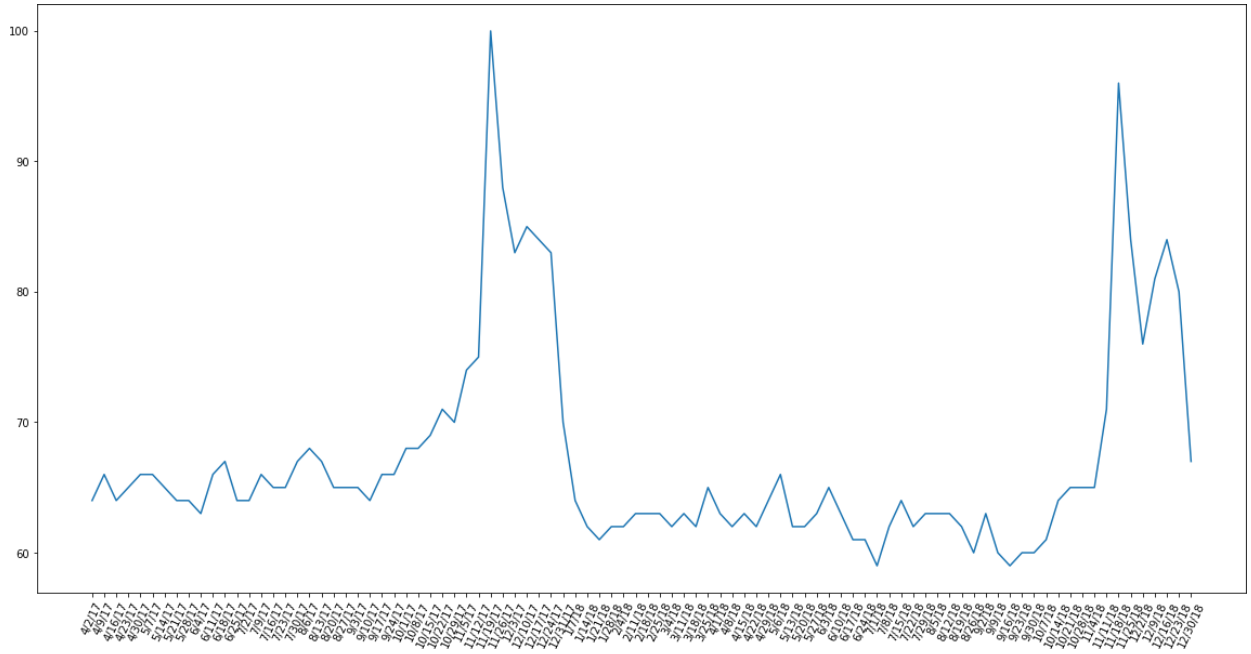


Рисунок 3.7 Розподіл покупок у США згідно з даними компанії Google.

Після побудови частотної діаграми транзакцій за днями (рисунок 3.8), виявлено, що піки у датасеті можуть справді відповідати Чорній П'ятниці 2017 і 2018 років. Це підтверджується обчисленнями.

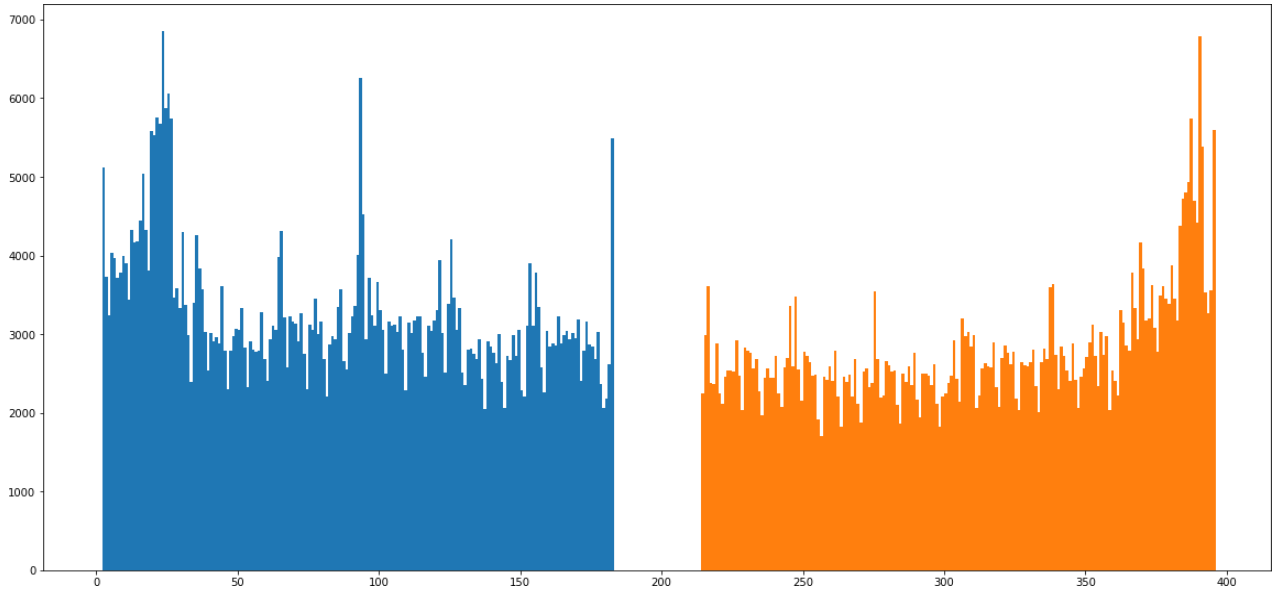


Рисунок 3.8

З графіка видно, що кількість транзакцій у пікові дні перевищує 6000 у навчальній вибірці та понад 5500 у тестовій вибірці.

Усвідомлення цього дасть вам кілька днів.

Ними виявилися 23, 25, 93, 387, 390 і 395 із заданих днів.

. Далі ми отримуємо такі дані (у форматі РР/ММ/ДД) з найбільшою кількістю транзакцій (Малюнок 3.9):

datetime.date(2017, 11, 23),

datetime.date(2017, 11, 25),

datetime.date(2018, 2, 1),

datetime.date(2018, 11, 23).

datetime.date(2018, 11, 26),

datetime.date(2018, 11, 27),

datetime.date(2018, 12, 1)

Рисунок 3.9 Результати конвертації змінної transactionDT

Нижче, 23 листопада 2017 року – відома як Чорна п’ятниця, і 26 листопада 2018 року – Кіберпонеділок.

Таким чином, аналіз Google і закономірності, виявлені в наборі даних, припускають, що зворотний відлік для набору даних починається 1 листопада 2017 року.

Цільова змінна

Після аналізу пропорцій шахрайських операцій у датасеті стає очевидним, що він є незбалансованим, як показано на нижченаведеному рисунку 3.10.

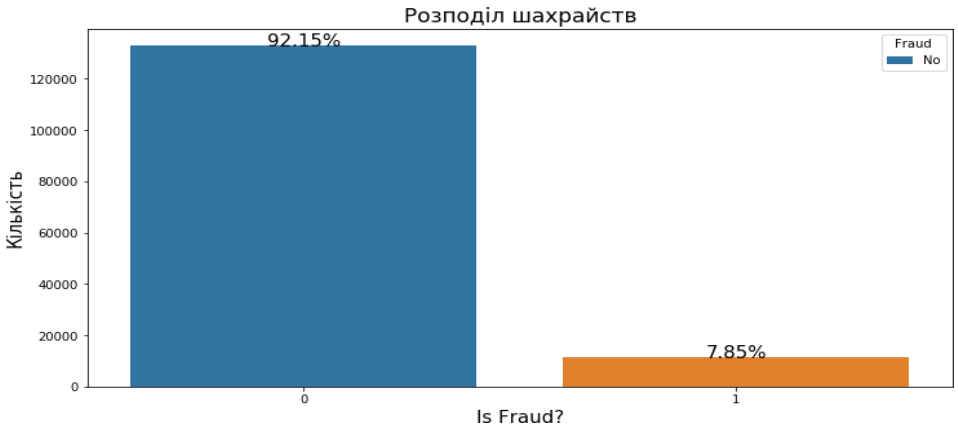


Рисунок 3.10 Розподіл операцій у датасеті

Для вирішення проблеми незбалансованості класів у датасеті, де випадків шахрайських транзакцій було значно менше, ніж легальних, було вирішено використовувати метод оверсемплінгу SMOTE (Synthetic Minority Over-sampling Technique) для класу ризикових транзакцій.

Метод SMOTE генерує синтетичні приклади меншинного класу, в цьому випадку шахрайських транзакцій, шляхом інтерполяції наявних прикладів цього класу. Таким чином кількість прикладів збільшується, зрівнюючи кількість екземплярів обох класів.

Використання оверсемплінгу меншинного класу допомагає вирішити проблему незбалансованості даних, яка може негативно впливати на продуктивність алгоритмів при побудові класифікаційних моделей.

Обробка “V”-змінних

У зв'язку з відсутніми значеннями в наборі даних, є підозра на наявність взаємозалежності у змінної 'V'.

Набір даних містить до 400 таких змінних.

З’ясуємо, які кореляції корельовані, створивши кореляційну матрицю (рис.3.11-3.15).

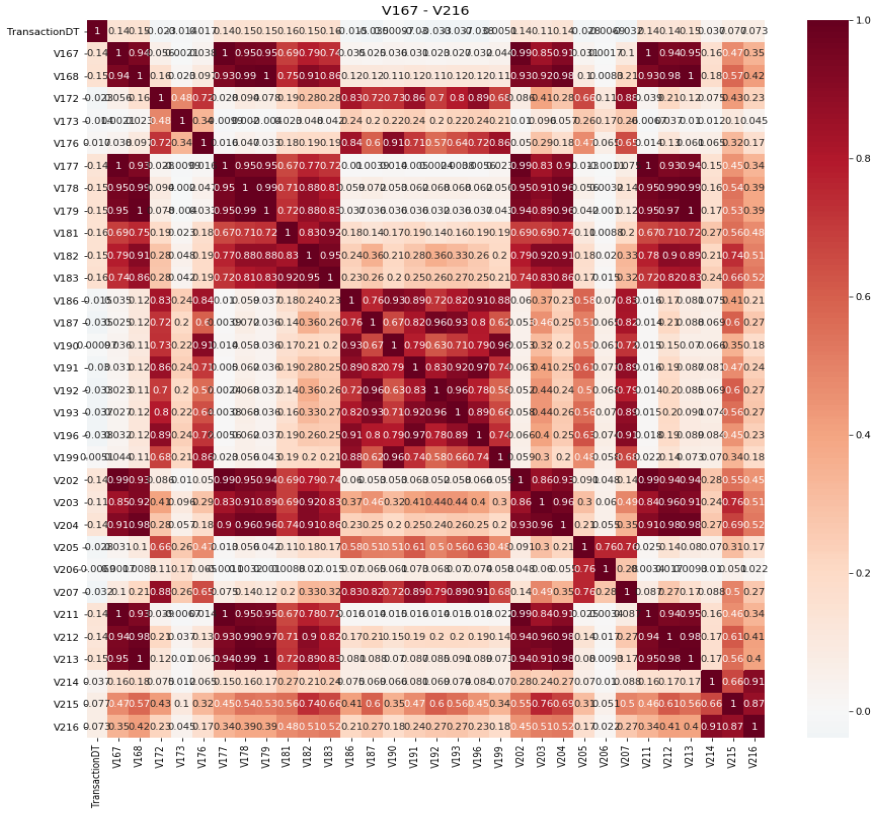


Рисунок 3.11 Кореляційна матриця V167-V216

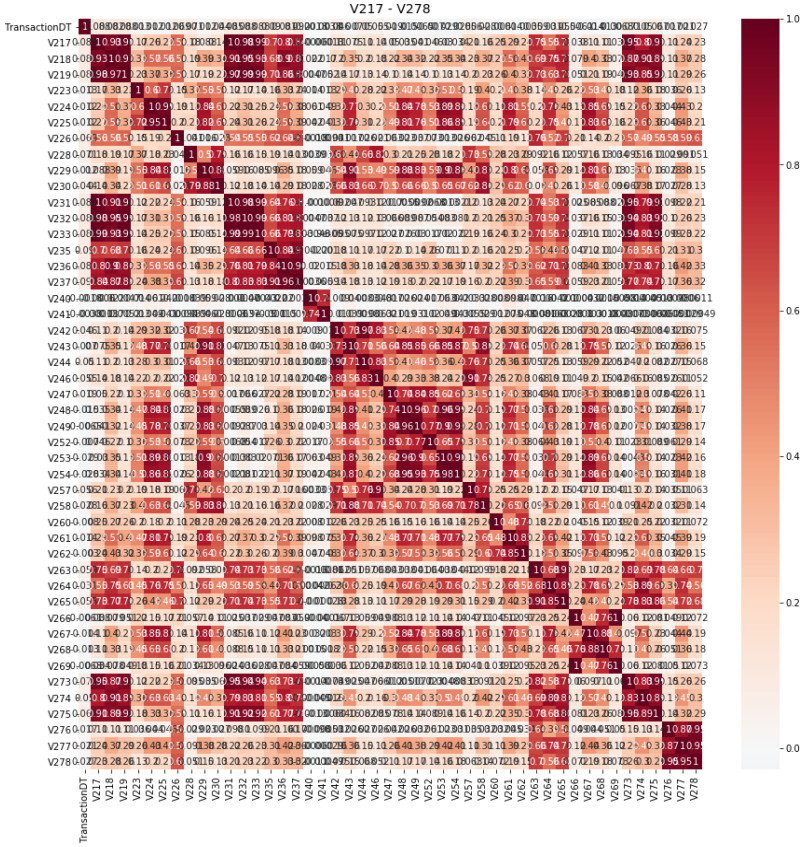


Рисунок 3.12 Кореляційна матриця V217-278

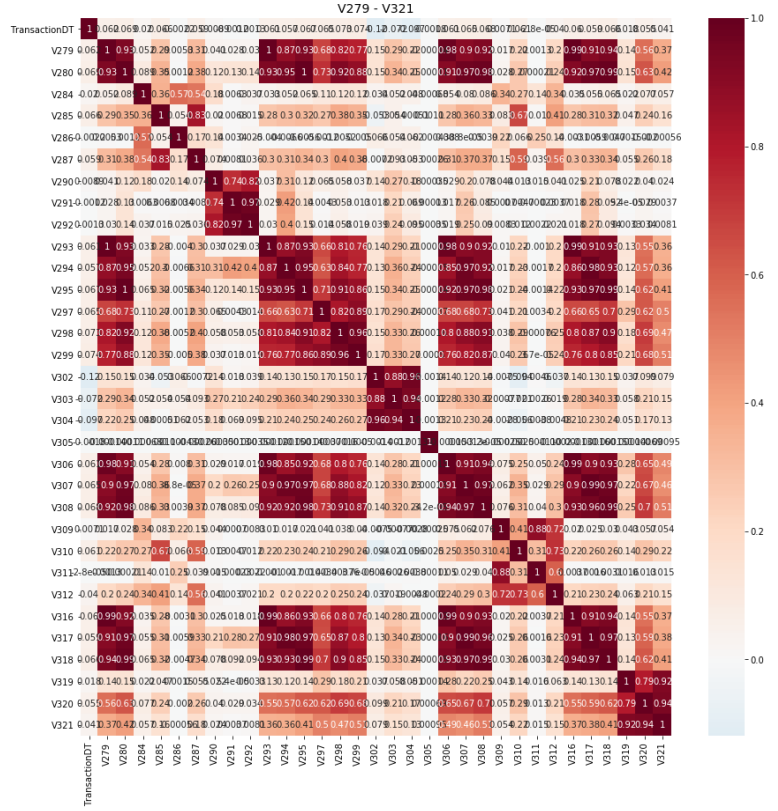


Рисунок 3.13 Кореляційна матриця V279-V321

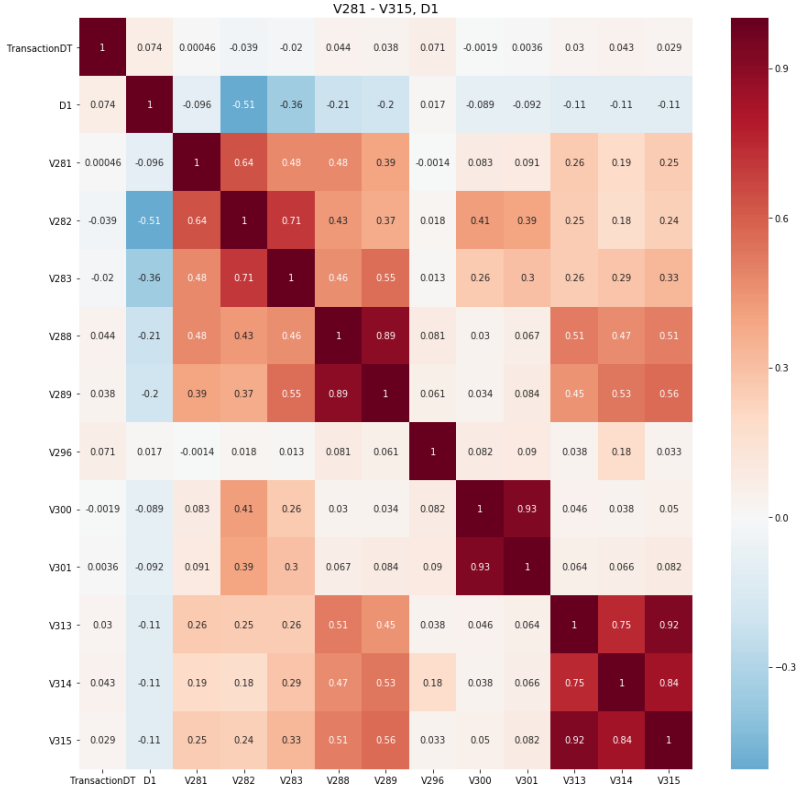


Рисунок 3.14 Кореляційна матриця V281-V315

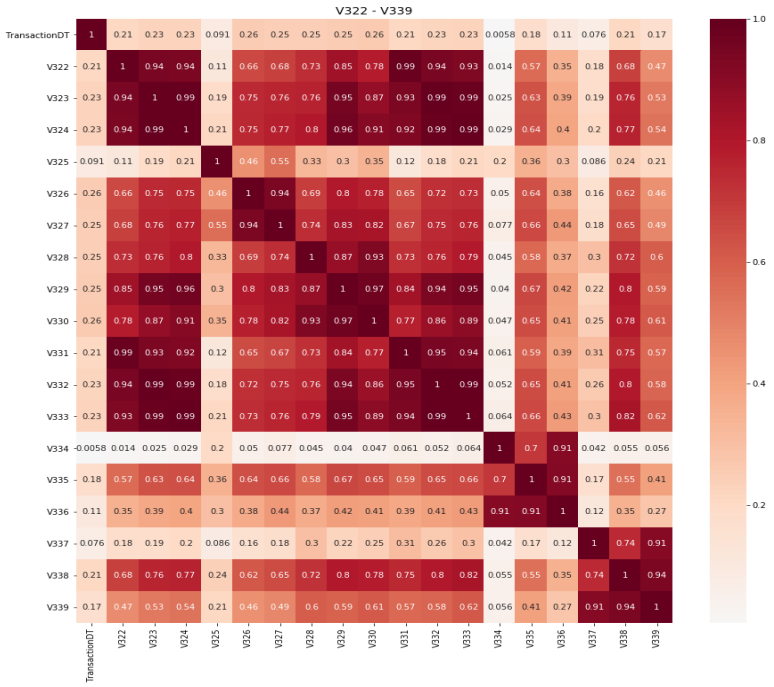


Рисунок 3.15 Кореляційна матриця V322-V339

Як бачите, між цими змінними існує велика мультиколінеарність.

Кількість змінних V можна зменшити до 128 шляхом диференціації груп змінних V відповідно до шаблону відсутніх значень (роблячи їх рівними за кількістю) і враховуючи їх кореляційну матрицю.

Їх кореляційна матриця показана на рисунку 3.16.

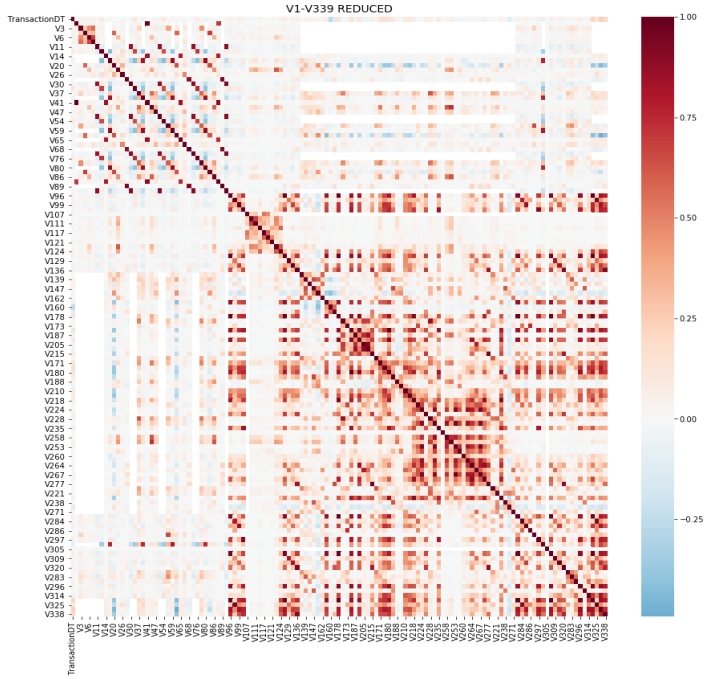


Рисунок 3.16 Кореляційна матриця після зменшення мультиколінеарності

Підготовка даних до побудови моделей

Перед безпосереднім навчанням моделі було виконано ряд етапів підготовки даних.

По-перше, стовпці з більш ніж 20% відсутніми значеннями були видалені.

Використовуючи попередній аналіз, кількість змінних V також було зменшено з 339 до 128.

Ідентифікатори транзакцій також видалено, оскільки ці дані не корисні для прогнозів.

Другим кроком було створення нового набору змінних для набору даних.

Зокрема, такі змінні, як день тижня, місяць і номер дня з початку зворотного відліку, були створені на основі дат, отриманих зі змінних TransactionDT (точок відліку для транзакцій, доступних у наборі даних).

Такі змінні виявилися більш значущими для навчання моделі, ніж секунди від конкретної контрольної точки.

Крім того, була створена нова змінна «номер картки» на основі змінних card1-card3 і Card5 шляхом об’єднання їхніх рядкових значень.

Цей підхід був спробою ізолювати людей, які виконували транзакції, щоб створена модель враховувала поведінку клієнта, а не аналізувала кожну транзакцію окремо від історії клієнта.

По-третє, були заповнені відсутні значення.

— це медіана для безперервних змінних і мітка переходу для категоріальних змінних.

Наступним кроком було перетворення категоріальних змінних, тобто кодування міток.

Це було віддано перевагу над гарячим кодуванням через більший розмір набору даних.

Усі категоріальні дані тепер будуть зіставлені з цілими числами.

На останньому етапі підготовки даних вибірку було стандартизовано, і на цій стадії було використано надмірну вибірку SMOTE для вирішення проблем дисбалансу класів.

2.3 Навчання моделей

На даний момент було навчено дві моделі: Random Forest Classifier і XGBoost Classifier.

Щоб налаштувати гіперпараметри моделі, ми розробили можливість вибирати найкращу модель шляхом порівняння за допомогою метрики AUC.

Я коротко поясню концепцію встановлення гіперпараметрів.

Для кожного набору гіперпараметрів навчальні зразки поділяються на сім частин, і кожна частина по черзі ділиться на навчальні та перевірочні зразки.

Перевірте якість прогнозів за допомогою отриманої метрики AUC моделі та обчисліть середнє значення AUC для кожного набору гіперпараметрів.

Пошук мінімуму ($-\Sigma AUCi$ / 7) серед моделей, побудованих у вказаному просторі гіперпараметрів, виконується за допомогою методу Нелдера-Міда.

Таке гніздо обрано у спробі навчитися та передбачити для кожного окремого періоду з урахуванням сортування даних за часом.

У таблиці нижче наведені значення AUC для наших моделей.

Алгоритм	AUC
Random Forest Classifier	0.887
XGBoost Classifier	0.891

Таблиця 3.2 Метрика AUC навчених моделей

XGBoost Classifier Tuned	0.897
LightGBM	0.919

Як бачите, алгоритм Light Gradient Boosting Machine навчився найкраще класифікувати транзакції.

Цей результат можна покращити, включивши додатковий аналіз вихідного зразка.

Взявши до уваги об'єм обчислень, необхідних для цього кроку, можна зробити висновок, що доцільніше зосередитися на аналізі розвідувальних даних.

Зосередьтеся на конкретних наборах даних.

Отже, найпродуктивнішою виявилася ЛГБТМ з такими параметрами, як показано на рисунку 3.17.

```
Параметри LGBM:
num_leaves: 737
min_child_weight: 0.17
feature_fraction: 0.72
bagging_fraction: 0.72
min_data_in_leaf: 179
objective: binary
max_depth: -1
learning_rate: 0.006
boosting_type: gbdt
bagging_seed: 13
metric: auc
verbosity: -1
reg_alpha: 0.3299927210061127
reg_lambda: 0.3885237330340494
random_state: 4
```

Рис. 3.17 Параметри найкращої моделі

Використання результатів моделювання

Після підготовки початкових даних і аналізу поведінки моделі, навченої на цих даних, ІТ можна використовувати для виявлення транзакцій.

На малюнку 3.18 показана функціональна схема виявлення шахрайства, побудованої на корпоративному наборі даних Vesta.

Ідея полягає в тому, щоб обробити транзакцію, ввести її в попередньо навчену модель і в результаті зробити висновок про те, чи належить транзакція до класу шахрайства чи ні.

Рис. 3.18, аркуш 1 Структурна схема інформаційної технології

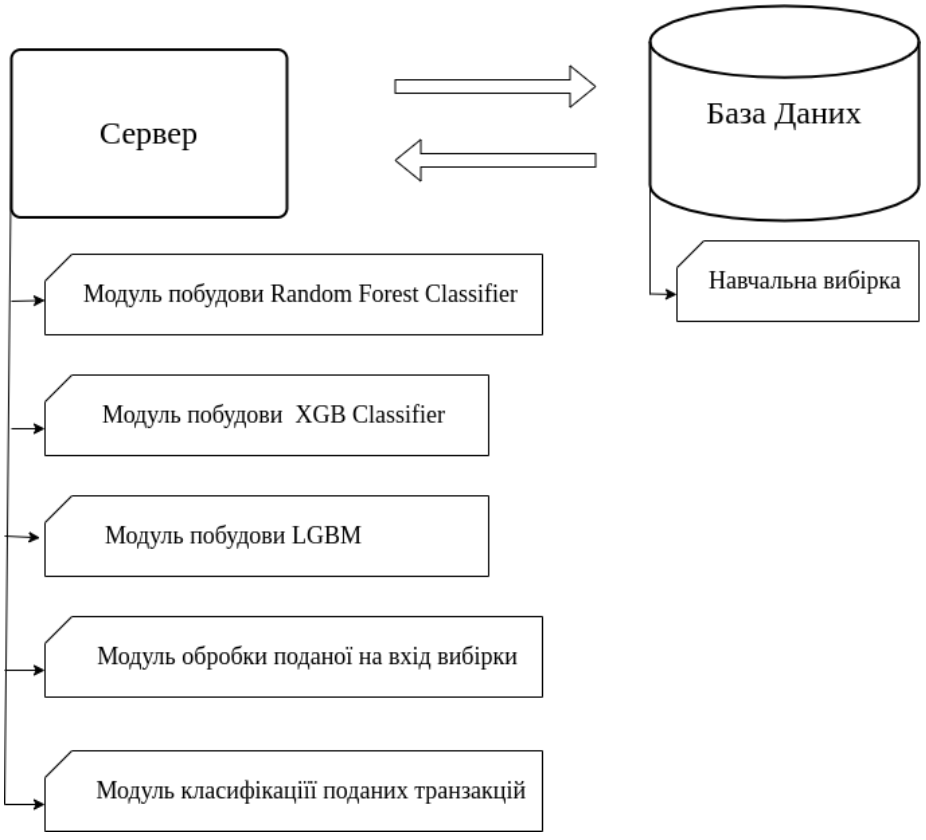


Рис. 3.18, аркуш 2 Структурна схема інформаційної технології

Ця інформаційна технологія може бути інтегрована в системи безпеки банківських систем і різноманітних електронних платіжних систем.

Обробляючи транзакції до їх підтвердження, підприємства зменшують ризик втрати клієнтів через менш безпечні платіжні системи.

Один із способів інтеграції цієї інформаційної технології показано на рисунку 3.19.

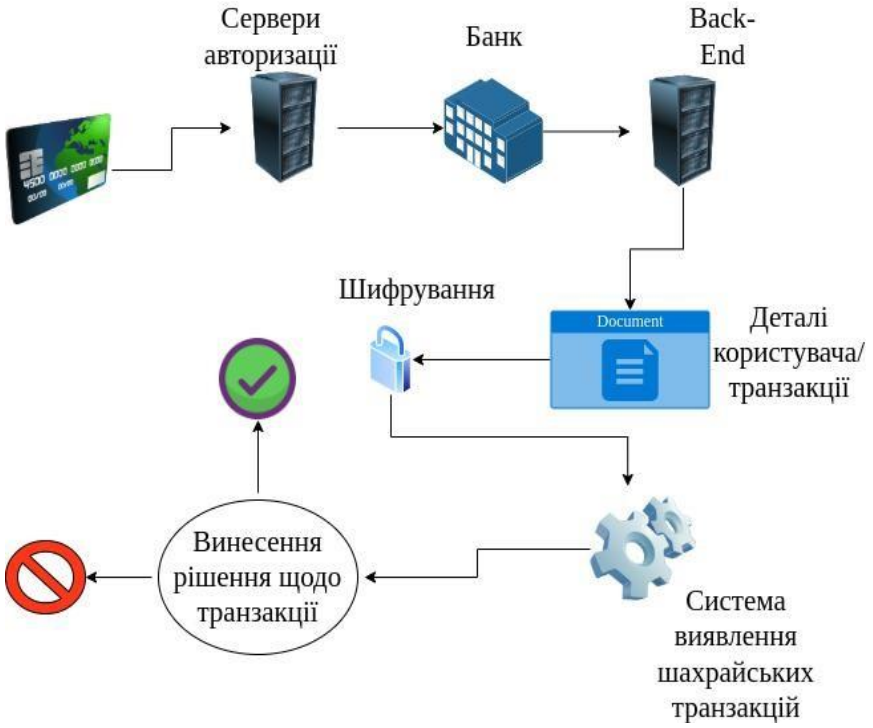


Рисунок 3.19 Приклад інтеграції ІТ

2.4 Висновки до розділу 3

У ході виконання практичної частини роботи було проведено такі ключові кроки:

1. Підготовка вихідного набору даних про транзакції до навчання моделей, включаючи обробку пропусків, кодування категоріальних змінних, масштабування числових змінних та вирівнювання незбалансованості класів за допомогою оверсемплінгу.

2. Налаштування гіперпараметрів для обраних алгоритмів машинного навчання з метою покращення їх продуктивності.

3. Порівняння результатів роботи побудованих класифікаційних моделей за метрикою AUC (Area Under Curve).

4. Інтеграція отриманих моделей в запропоновану інформаційну технологію для виявлення шахрайських транзакцій у режимі реального часу.

У якості подальших досліджень можна працювати над підвищенням прогнозної сили існуючих моделей, наприклад, шляхом додаткового конструювання ознак чи застосування ансамблевих методів. Також можна розглянути використання інших алгоритмів машинного навчання в рамках запропонованої інформаційної технології.

Окрім виявлення шахрайств, дану інформаційну технологію можна розширити для реалізації системи підтримки прийняття рішень в банківській сфері на основі результатів класифікаційних моделей.

ВИСНОВКИ

Під час переддипломної практики було створено початкові варіанти розділів магістерської дисертації. Після вивчення предметної області та розгляду математичних методів виявлення шахрайських операцій було обрано алгоритми навчання моделей у третьому - практичному розділі роботи. Було проведено аналіз даних щодо транзакцій платіжної системи Vesta Corporation. На основі цих даних застосовано методи машинного навчання для побудови моделей класифікації транзакцій для виявлення шахрайських операцій. Розробка виконувалась за допомогою мови програмування Python, з використанням редактора Jupyter як основного середовища розробки. Запропоновано метод налаштування гіперпараметрів моделей. Тестування проводилося за допомогою метрики AUC та показало хороші результати на навчальній та тестовій вибірках. На основі виконаного моделювання було запропоновано інформаційну технологію для виявлення шахрайських транзакцій. Ця інформаційна технологія може бути використана для імплементації системи підтримки прийняття рішень у банківських установах та системах забезпечення онлайн платежів. У рамках подальших досліджень можна працювати над підвищенням точності класифікації створених моделей, а також залученням інших алгоритмів машинного навчання до вирішення поставленої задачі - наприклад, нейронних мереж, та створенням варіантів ансамблів попередньо навчених алгоритмів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Інтернет-представництво Національного банку України, URL: https://bank.gov.ua/ua/news/all/startuye-informatsiyna-kampaniya-natsionalnogo-banku-z-platijnoyi-bezpeki-shahraygudbay (дата звернення:жовтень 2022).

2. Mohri M., Rostamizadeh A., Talwalkar A. Foundations of Machine Learning. Second edition. Cambridge, MA. MIT Press, 2018. 488p.

3. Vapnik V.N. The Nature of Statistical Learning Theory. Springer, 2000, 313 p.

4. Кузнєцова Н.В. Аналіз фінансово-економічних даних. Частина 1:Дерева рішень *Навчально методичні матеріали до курсу лекцій*. Київ, 2017, 59 с.

5. Rokach L., Maimon O. Data mining with decision trees: theory and applications *Machine Perception and Artificial Intelligence*. Vol. 81, 2014. 305 p.

6. Friedman J.H. Greedy Function Approximation: A Gradient Boosting Machine", *The Annals of Statistics.* Vol. 29, pp.1189-1232, 2001.

7. Why XGBoost wins every Machine Learning Competition. URL: https://syncedreview.com/2017/10/22/tree-boosting-with-xgboost-why-does-xgboost-win-every-machine-learning-competition/ (accessed in October, 2022).

8. Ke G., Finley T., Wang T., Chen W., Ma W., Ye Q., Lin T. LightGBM: A Highly Efficient Gradient Boosting Decision Tree, *31st Conference on Neural Information Processing Systems*, Long Beach, CA, USA, 2017.

9. N. V. Kuznietsova Scoring Technology for Risk Assessment of Fraud inBanking, *Selected Papers of the XVI International Scientific and Practical Conference* "Information Technology and Security", CEUR Workshop Proceedings.Vol. 1813, pp. 54–61, 2016.

10. Nelder J.A., Mead R. A simplex method for function minimization *The Computer Journal*, Vol. 7, pp. 308–313, 1965.

11. Статистика покупок в США за даними компанії Google. URL: https://trends.google.com/trends/explore?cat=18&date=2017-04-01%202018-12-31&geo=US (дата звернення 27.10.2022)

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
import pandas as pdimport numpy as np

from matplotlib import pyplot as pltimport seaborn as sns

from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_splitfrom sklearn.preprocessing import
StandardScaler from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier

from sklearn import preprocessing
from sklearn.metrics import confusion_matrix, roc_auc_score
from sklearn.model_selection import StratifiedKFold, cross_val_score, KFoldfrom xgboost
import XGBClassifier
import xgboost as xgb

from hyperopt import fmin, hp, tpe, Trials, space_eval, STATUS_OK, STATUS_RUNNING
from functools import partial

import warnings warnings.filterwarnings('ignore')

def reduce_mem_usage(df):
start_mem = df.memory_usage().sum() / 1024**2
print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))
for col in df.columns: col_type = df[col].dtype

if col_type != object: c_min = df[col].min() c_max = df[col].max()
if str(col_type)[:3] == 'int':
```

```
if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max: df[col] =
    df[col].astype(np.int8)
elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max: df[col] =
    df[col].astype(np.int16)
elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max: df[col] =
    df[col].astype(np.int32)
elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max: df[col] =
    df[col].astype(np.int64)
    else:
if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max: df[col] =
    df[col].astype(np.float16)
elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max: df[col] =
    df[col].astype(np.float32)
    else:
    df[col] = df[col].astype(np.float64)

    else:
    df[col] = df[col].astype('category')

    end_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage after optimization is: {:.2f} MB'.format(end_mem)) print('Decreased by
    {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))return df
def datetime_trans(train,start_date='2017-11-30'):
    startdate=datetime.datetime.strptime(start_date,"%Y-%m-%d")
    train['TransactionDT']=train['TransactionDT'].fillna(train['TransactionDT'].mean())
    train['date']=train['TransactionDT'].apply(lambda x : datetime.timedelta(seconds=x)+startdate)
    train['weekday']=train['date'].apply(lambda x :x.weekday)
```

```
train['month']=(train['date'].dt.year-2017)*12+train['date'].dt.month
train['hour']=train['date'].apply(lambda x :x.hour) train['day']=(train['date'].dt.year-
2017)*365+train['date'].dt.dayofyear
train['year_weekday']=train['date'].apply(lambda x : str(x.year)+'_'+str(x.weekday()))
train['weekday_hour']=train['date'].apply(lambda x :str(x.weekday())+'_'+str(x.hour))
date_col=['weekday','month','day','hour','year_weekday','weekday_hour'] datetime_trans(train)
datetime_trans(test)
train_transaction = pd.read_csv('../input/ieee-fraud-detection/train_transaction.csv')
print(train_transaction.shape)
train_transaction = reduce_mem_usage(train_transaction)train_transaction.head()
train_identity = pd.read_csv('../input/ieee-fraud-detection/train_identity.csv')
print(train_identity.shape)
train_identity = reduce_mem_usage(train_identity)train_identity.head()
train_df = pd.merge(train_transaction, train_identity, how='left')print(train_df.shape)
len_train_df = len(train_df)
del train_transaction, train_identitytrain_df.head()
test_transaction = pd.read_csv('../input/ieee-fraud-detection/test_transaction.csv')
print(test_transaction.shape)
test_transaction = reduce_mem_usage(test_transaction)

test_identity = pd.read_csv('../input/ieee-fraud-detection/train_identity.csv')
print(test_identity.shape)
test_identity = reduce_mem_usage(test_identity)

test_df = pd.merge(test_transaction, test_identity, how='left')
test_df.columns = train_df.drop('isFraud', axis=1).columnsprint(test_df.shape)
del test_transaction, test_identity plt.figure(figsize=(8, 4)) plt.hist(train_df['TransactionDT'],
```

```
label='Train')plt.hist(test_df['TransactionDT'], label='Test') plt.ylabel('Count')
plt.title('Розподіл дати транзакції')plt.legend()
plt.tight_layout()plt.show()

mv = combined_df.isnull().sum()/len(combined_df) combined_mv_df =
combined_df.drop(columns=mv[mv>0.2].index)del combined_df, train_df, test_df
print(combined_mv_df.shape)
num_mv_df = combined_mv_df.select_dtypes(include=np.number)print(num_mv_df.shape)

cat_mv_df = combined_mv_df.select_dtypes(exclude=np.number)print(cat_mv_df.shape)
del combined_mv_df

imp_median = SimpleImputer(missing_values=np.nan, strategy='median')
num_df = pd.DataFrame(imp_median.fit_transform(num_mv_df),
columns=num_mv_df.columns)del num_mv_df
print(num_df.shape)

imp_max = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
cat_df = pd.DataFrame(imp_max.fit_transform(cat_mv_df), columns=cat_mv_df.columns)del
cat_mv_df
print(cat_df.shape)

combined_df_cleaned = pd.concat([num_df, cat_df], axis=1)del num_df, cat_df

print(f'Total missing values: {combined_df_cleaned.isnull().sum().sum()}')
print(combined_df_cleaned.shape)
combined_df_cleaned.head()
```

```
    combined_df_encoded = pd.get_dummies(combined_df_cleaned, drop_first=True)
    print(combined_df_encoded.shape)
    del combined_df_cleaned combined_df_encoded.head()

    import time
def objective(params):time1 = time.time()params = {
    'max_depth': int(params['max_depth']), 'gamma': "{:.3f}".format(params['gamma']),
    'subsample': "{:.2f}".format(params['subsample']),
    'reg_alpha': "{:.3f}".format(params['reg_alpha']),
    'reg_lambda': "{:.3f}".format(params['reg_lambda']), 'learning_rate':
    "{:.3f}".format(params['learning_rate']), 'num_leaves': '{:.3f}'.format(params['num_leaves']),
    'colsample_bytree': '{:.3f}'.format(params['colsample_bytree']), 'min_child_samples':
    '{:.3f}'.format(params['min_child_samples']),'feature_fraction':
    '{:.3f}'.format(params['feature_fraction']), 'bagging_fraction':
    '{:.3f}'.format(params['bagging_fraction'])
    }

    print("\n################ New Run #################")
    print(f"params = {params}")
    FOLDS = 7
    count=1
    skf = StratifiedKFold(n_splits=FOLDS, shuffle=True, random_state=42)

    tss = TimeSeriesSplit(n_splits=FOLDS) y_preds = np.zeros(submission.shape[0]) y_oof =
    np.zeros(X_train_smote.shape[0])score_mean = 0
for tr_idx, val_idx in tss.split(X_train_smote, y_train_smote):clf = xgb.XGBClassifier(
    n_estimators=600, random_state=4, verbose=True,tree_method='gpu_hist',
```

```
    **params

    )

    X_tr, X_vl = X_train_smote.iloc[tr_idx, :], X_train_smote.iloc[val_idx, :]y_tr, y_vl =

    y_train_smote.iloc[tr_idx], y_train_smote.iloc[val_idx]


    clf.fit(X_tr, y_tr)
    #y_pred_train = clf.predict_proba(X_vl)[:,1]#print(y_pred_train)
    score = make_scorer(roc_auc_score, needs_proba=True)(clf, X_vl, y_vl)# plt.show()
    score_mean += score
    print(f'{count} CV - score: {round(score, 4)}')count += 1
    time2 = time.time() - time1
    print(f"Total Time Run: {round(time2 / 60,2)}")gc.collect()
    print(f'Mean ROC_AUC: {score_mean / FOLDS}')del X_tr, X_vl, y_tr, y_vl, clf, score
    return -(score_mean / FOLDS)
params = {'num_leaves': int((2**10)*0.72),'min_child_weight': 0.17,
    'feature_fraction': 0.72,
    'bagging_fraction': 0.72,
    'min_data_in_leaf': 179,'objective': 'binary', 'max_depth': -1,
    'learning_rate': 0.006, "boosting_type": "gbdt","bagging_seed": 13, "metric": 'auc',
    "verbosity": -1,
    'reg_alpha': 0.3299927210061127,
    'reg_lambda': 0.3885237330340494,
    'random_state': 4,

    }
    print('Параметри LGBM:')
  for key,value in params.items():print(key + ': ' + str(value))
    FOLDS = 7
```

```
folds = KFold(n_splits=NFOLDS)

columns = train.columns splits = folds.split(train, y_)

y_preds = np.zeros(test.shape[0])y_oof = np.zeros(train.shape[0]) score = 0

feature_importances = pd.DataFrame() feature_importances['feature'] = columns

for fold_n, (train_index, valid_index) in enumerate(splits):

X_train, X_valid = train[columns].iloc[train_index], train[columns].iloc[valid_index]y_train,

y_valid = y_.iloc[train_index], y_.iloc[valid_index]

dtrain = lgb.Dataset(X_train, label=y_train) dvalid = lgb.Dataset(X_valid, label=y_valid)

   clf = lgb.train(params, dtrain, 10000, valid_sets = [dtrain, dvalid], verbose_eval=200,
early_stopping_rounds=300)

feature_importances[f'fold_{fold_n + 1}'] = clf.feature_importance()

y_pred_valid = clf.predict(X_valid)y_oof[valid_index] = y_pred_valid

print(f"Fold {fold_n + 1} | AUC: {roc_auc_score(y_valid, y_pred_valid)}")

score += roc_auc_score(y_valid, y_pred_valid) / NFOLDSy_preds += clf.predict(test) /

NFOLDS

del X_train, X_valid, y_train, y_validgc.collect()

print(f"\nMean AUC = {score}")

feature_importances['average'] = feature_importances[[f'fold_{fold_n + 1}' for fold_n in
range(folds.n_splits)]].mean(axis=1)

plt.figure(figsize=(16, 16))

sns.barplot(data=feature_importances.sort_values(by='average', ascending=False).head(50),
x='average',y='feature');

plt.title('50 TOP feature importance over {} folds average'.format(NFOLDS));
```