НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ

Кафедра математичних методів системного аналізу

До захисту допущено:

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___»_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Моделі і методи машинного навчання для виявлення аномалій у поведінці клієнтів банку»

Виконав: студент IV курсу, групи КА-05
Артеменко Євгеній Вячеславович _____

Керівник: д.т.н., професор
Недашківська Надія Іванівна _____

Консультант з економічної частини: к. е. н., доцент
Рощина Надія Василівна _____

Консультант з нормоконтролю: к.ф.-м.н.
Статкевич Віталій Михайлович _____

Рецензент: к.т.н., доцент кафедри системного проектування
НН ІПСА КПІ ім. Ігоря Сікорського,
Булах Богдан Вікторович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___»_____2024 р.

ЗАВДАННЯ

на дипломну роботу студенту

Артеменку Євгенію Вячеславовичу

1. Тема роботи «Моделі і методи машинного навчання для виявлення аномалій у поведінці клієнтів банку», керівник роботи Недашківська Надія Іванівна, д.т.н., професор, затверджені наказом по університету від «31» травня 2024 р. № 2240-с.

2. Термін подання студентом роботи: 11.06.2024.

3. Вихідні дані до роботи: набір даних транзакцій та набір даних про клієнтів банку.

4. Зміст роботи: огляд предметної області, математичні методи та моделі для виявлення аномалій у даних, реалізація програмного продукту та аналіз результатів, функціонально-вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу: моделі та методи виявлення аномалій, метрики для моделей класифікації та виявлення аномалій, метрики для моделей кластеризації, найкращі параметри для моделей класифікації, найкращі параметри для моделей кластеризації, найкращі параметри для моделей виявлення аномалії, показники якості оцінки класифікації, показники якості оцінки кластеризації, показники якості оцінки виявлення аномалій.

6. Консультанти розділів роботи[1]*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н. В., доцент, к.е.н.		

7. Дата видачі завдання: _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Формулювання тематики (напряму) дослідження	5.02.2024 - 26.02.2024	Виконано
2	Аналіз актуальності задач стосовно тематики дослідження	12.02.2024 - 04.03.2024	Виконано
3	Аналіз відомих результатів стосовно тематики дослідження	19.02.2024 - 11.03.2024	Виконано
4	Формулювання задач дослідження	11.03.2024 - 01.04.2024	Виконано
5	Уточнення теми дипломної роботи	01.04.2024 - 08.04.2024	Виконано
6	Збір статичних даних, попередній аналіз даних	01.04.2024 - 15.04.2024	Виконано
7	Розробка програмного продукту для виконання обчислювальних експериментів	15.04.2024 - 10.05.2024	Виконано
8	Виконання обчислювальних експериментів, аналіз та оформлення результатів	15.04.2024 - 15.05.2024	Виконано
9	Оформлення пояснювальної записки у цілому	15.05.2024 - 03.06.2024	Виконано
10	Підготовка презентації для захисту	22.05.2024 - 02.06.2024	Виконано
11	Попередній захист дипломної роботи	05.06.2024	Виконано

Студент Євгеній АРТЕМЕНКО

Керівник Надія НЕДАШКІВСЬКА

[1]* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломної роботи.

РЕФЕРАТ

Дипломна робота: 118 с., 27 рис., 18 табл., 2 додатки, 29 джерел.

КЛАСИФІКАЦІЯ, КЛАСТЕРИЗАЦІЯ, МАШИННЕ НАВЧАННЯ, ВИЯВЛЕННЯ АНОМАЛІЙ, НЕЙРОННА МЕРЕЖА.

У роботі розглянуто та проаналізовано найбільш поширені моделі та методи машинного навчання для виявлення аномалій.

Досліджено різні моделі та методи класифікації, кластеризації, виявлення аномалій. Робота обраних для дослідження методів була розглянута на практичній задачі, а саме виявлення аномалій у поведінці клієнтів банку.

Об’єктом дослідження стали дані про транзакції Credit Card Fraud Detection, та дані, зібрані однією з банківських установ Португалії Bank Marketing (анкетні, конфіденційні дані) та їх значення для швидкого та точного виявлення підозрілих транзакцій, клієнтів з аномальною поведінкою.

Предметом дослідження стали математичні методи машинного навчання для проведення класифікації, кластеризації, та виявлення аномалій на основі статистичних даних.

Одним із результатів роботи є програмне забезпечення, написане на мові програмування Python.

ABSTRACT

Bachelor thesis: 118 pages, 27 figures, 18 tables, 2 appendices, 29 references.

CLASSIFICATION, CLUSTERING, MACHINE LEARNING, ANOMALY DETECTION, NEURAL NETWORK.

The thesis examines and analyzes the most common models and methods of machine learning for anomaly detection.

Various models and methods of classification, clustering, and anomaly detection are investigated. The work of the selected research methods was examined on a practical task, specifically anomaly detection in bank customer behavior.

The object of the research is the data on transactions from the Credit Card Fraud Detection dataset and data collected by a Portuguese banking institution, Bank Marketing (questionnaire, confidential data), and their significance for the rapid and accurate detection of suspicious transactions and clients with anomalous behavior.

The subject of the research is the mathematical methods of machine learning for classification, clustering, and anomaly detection based on statistical data.

One of the results of the work is software written in the Python programming language.

ЗМІСТ

ВСТУП

У сучасному світі фінансові установи стикаються з численними викликами, пов'язаними з безпекою та надійністю своїх операцій. Одним із найбільш критичних аспектів є виявлення аномалій у поведінці клієнтів, що може свідчити про шахрайство або інші небажані дії. Традиційні методи аналізу даних часто не встигають за швидкими змінами та великою кількістю транзакцій, що проходять через банківські системи. У цьому контексті машинне навчання стає незамінним інструментом для виявлення аномалій, оскільки дозволяє автоматизувати процес аналізу та забезпечити високу точність виявлення підозрілих дій.

Розділ 1 містить огляд банківської сфери та керування ризиками.

Розділ 2 містить опис математичних моделей та методів класифікації, кластеризації, виявлення аномалій.

Розділ 3 містить результати моделей та їх порівняльний аналіз.

Розділ 4 містить проведений функціонально-вартісний аналіз програмного продукту.

Предметом виявлення аномалій у поведінці клієнтів банку є задача класифікації, кластеризації та виявлення точок даних, які не потрапили в жодний кластер.

РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основі визначення та терміни

Банківська установа - юридична особа, яка на підставі банківської ліцензії має виключне право надавати банківські послуги, за що беруть певну фінансову плату (відсотки).

Банківська ліцензія - документ, що видається регулюючим органом влади тієї країни, де фізично знаходиться банк, в порядку і на умовах, визначених законом країни, на підставі якого банки мають право здійснювати банківську діяльність.

Клієнт банку - це фізична, або, юридична особа, що користується послугами, які може надати банк.

Банківські операції - це конкретні дії банків, що надають певні банківські послуги. Ці терміни не мають точного визначення.

Банківський рахунок - рахунок, де обліковуються кошти, вимоги, зобов'язання банку стосовно клієнтів. За допомогою такого рахунку здійснювати переказ коштів, використовуючи банківські платіжні системи.

Банківська платіжна система (Payment System) - це система, що дає можливість швидко та безпечно виконувати транзакції між фізичними та/або юридичними особами. Найвідоміші міжнародні платіжні системи:

1. VISA International Service Association.
2. MasterCard.
3. PayPal.

Банківські послуги:

1. Створення особистих рахунків клієнтів - поточні (для фінансових операції), чи, депозитні (для вкладу своїх заощаджень задля того, щоб від банку отримати дохід у вигляді певних відсотків з вкладу).

2. Надання кредитних послуг - це може бути короткостроковий кредит, який призначений для вирішення певних тимчасових фінансових труднощів клієнта. Чи, довгострокові такі, як іпотечні кредити.
3. Інвестиційні послуги - це насамперед пенсійні фонди, мета яких полягає у тому, щоб банки надали можливість клієнтам зберігати свої кошти для майбутніх пенсійних виплат.
4. Платіжні послуги - тобто, здійснення платіжних операцій.
5. Клієнтська підтримка - це надання консультацій, пояснення тих, чи, інших деталей певних зобов'язань, обов’язків клієнту працівником банку.

Транзакція (transactions) - це одна із банківських операцій, зміст якої полягає в переказі заощаджень з одного рахунку на інший певними платіжними системами.

Кредитна карта - це пластикова карта, яку видав банк або фінансова установа, яка надає власнику можливість здійснювати транзакції на певну суму коштів, яку можна витрачати до певного ліміту. Основна особливість такої картки - власник має можливість використовувати позичені кошти (кредит) для здійснення покупок, а потім повертати ці кошти банку за умови відсутності боргу.

Кредит у банківській сфері - це фінансова угода, що надається банком, який надає позичальнику (клієнту) певну суму коштів на умовах погашення даної суми зі сплатою певних відсотків за користування кредитом протягом певного періоду часу.

Персональні дані, що збираються та обробляються працівниками Приват Банку[1]:

1. Анкетні дані клієнта.
    a. Фізичної особи (дані документа, що посвідчує особу, освіта, соціальний статус, контактна інформація).

   b. Юридичної особи (відомості про кінцевого бенефіціарного власника юридичної особи, відомості про осіб, уповноважених представляти інтереси акціонерів, учасників юридичної особи).
2. Дані, отримані в процесі надання послуг (дані договорів, укладених з банком).
3. Дані, отримані в процедурах фінансового моніторингу (дані про ділову репутацію, інформація про актуальність ліцензій, дозволів).
4. Дані, отримані в процесі банківського обслуговування.
5. Дані кредитної історії клієнта.
6. Дані, отримані з публічних джерел (інформація у відкритих джерелах щодо наявності кримінального провадження).
7. Дані, що отримуються в процесі роботи з кредитними зобов’язаннями.
8. Технічні дані сайтів та мобільних додатків.
9. Чутливі категорії персональних даних (зображення та голос клієнта).

1.2 Аналіз банківських операцій для виявлення аномальної поведінки клієнта

Аналіз банківських операцій для виявлення аномальної поведінки клієнта - це складний процес, який включає в себе кілька етапів та технологій для виявлення потенційних шахрайств, відмивання грошей. Розглянемо етапи аналізу.

1. Збір даних - банки збирають великий обсяг даних про транзакції клієнтів, включаючи суму, час, місце, одержувача коштів та інші деталі операцій.
2. Профілювання клієнтів - створюються профілі клієнтів на основі їхньої звичайної поведінки. Це може включати середні суми витрат, частоту транзакцій, географічне розташування та типи операцій.
3. Використання алгоритмів та моделей - для аналізу даних використовуються алгоритми машинного навчання та моделі виявлення

аномалій. Вони допомагають ідентифікувати відхилення від звичайної поведінки клієнтів.

4. Встановлення порогових значень - банки встановлюють порогові значення для різних типів трансакцій. Якщо операція перевищує ці пороги або відповідає певним критеріям ризику, вона позначається як підозріла.
5. Моніторинг у реальному часі - за це відповідають системи моніторингу в реальному часі, які постійно перевіряють транзакції на відповідність критеріям аномальної поведінки.
6. Розслідування підозрілих транзакцій - якщо виявляється підозріла транзакція, вона передається до відділу служби безпеки банку для подальшого розслідування. Цей етап може включати перевірку додаткових даних, зв'язок з клієнтом та інші заходи.
7. Звітність та регуляторні вимоги - банки зобов'язані звітувати про підозрілі транзакції перед регуляторними органами, такими як фінансові інспекції та правоохоронні органи. Це допомагає забезпечити дотримання законів проти відмивання грошей та фінансування тероризму.

1.3 Ризики, пов'язані з аномальною поведінкою клієнтів для банку

Ризики, пов'язані з аномальною поведінкою клієнтів для банків [2]:

1. Відповідність нормативним вимогам - можуть порушуватися клієнтами банку.
2. Фінансова злочинність - фінансування тероризму, шахрайство, крадіжки, відмивання грошей.
3. Кредитні ризики - пов’язаний з імовірністю неправильного прийняття рішення щодо надання позики клієнту, що залежить від його персональних даних.

4. Кіберризики - вразливість банківських онлайн-систем до кібер-атак, збоїв.

Управляти даними ризиками здатний генеративний штучний інтелект - штучний інтелект, здатний на основі вхідної інформації створювати схожу вихідну. Тобто, у банківській справі, задача генеративного штучного інтелекту - надавати рекомендації щодо вирішення проблеми, використовуючи знання щодо вирішення попередніх проблем.

1.4 Застосування методів штучного інтелекту в банківській сфері

Трансформація штучного інтелекту (ШІ) в банківській справі впливає на традиційні банки та необанки (банківські організації, що використовують цифрові технології для надання послуг клієнтам) [3]. Спостерігається тенденція збільшення ефективності та залучення клієнтів після переходу від класичного штучного інтелекту, який був орієнтований на дані, до просунутого, генеративного штучного інтелекту.

Еволюція штучного інтелекту в банківській справі розпочалася з автоматизації роботи та аналізу даних, що спочатку були складними завданнями. Сьогодні штучний інтелект застосовується в управлінні ризиками, запобіганні шахрайству та індивідуальній клієнтській службі [3]. Розробка генеративного штучного інтелекту обіцяє подальше покращення операцій та стратегій банків.

Генеративний штучний інтелект має потенціал для підвищення ефективності та результативності в управлінні ризиками. Він допоможе банкам зосередитися на стратегічному запобіганні ризикам, партнерстві з бізнес-напрямками та контролю за діями нових клієнтів. Це може призвести до створення центрів аналізу ризиків на основі штучного інтелекту, які забезпечуватимуть автоматизоване звітування та найвищу ефективність прийняття рішень, пов’язаних із ризиками.

Однак це не єдині області, де штучний інтелект допомагає банкам. Традиційні банки ставлять пріоритет на безпеку, організацію процесів та управління ризиками. Крім цього, до останнього часу задоволення споживачів було недостатнім. Використання штучного інтелекту в банківській справі значно покращило клієнтський досвід. Штучний інтелект змінив спосіб взаємодії між банками та клієнтами. Чат-боти, що працюють на основі штучного інтелекту, тепер є стандартом у сфері клієнтської служби для багатьох банків, надаючи миттєві відповіді клієнтам та цілодобові послуги.

Чат-бот Erica обробив 1,5 мільярда запитів з моменту запуску в 2018 році, надаючи 24/7-клієнтську підтримку, що зменшує час очікування та покращує задоволення клієнтів, швидко та майже безпомилково обробляючи запити та транзакції [3]. Нині банки залучають алгоритми штучного інтелекту для оцінки клієнтських даних, виявлення індивідуальних фінансових активів та надання індивідуальних порад, що дає можливість клієнтам приймати найкращі рішення та збільшує довіру клієнтів.

Прикладом можливості штучного інтелекту покращити клієнтську службу є використання його банком Barclays для виявлення шахрайства [3]. Система штучного інтелекту аналізує всі платіжні транзакції у режимі реального часу, знаходячи та перешкоджаючи потенційним шахрайствам, що захищає клієнтів та підвищує їхню оцінку заходів безпеки банку.

Glass (платформа дослідження та аналізу на основі ШІ) аналізує дані ринку та моделі банку, використовуючи техніки машинного навчання (МН) для виявлення ринкових тенденцій та прогнозування запитів клієнтів, тобто, для стратегічних фінансових інсайтів [3]. Штучний інтелект на основі даних про взаємодію клієнта з програмою онлайн-банкінгу має можливість коригувати та вдосконалювати інтерфейс програми, задля кращої відповідності індивідуальним потребам та поведінці користувачів.

Використовуючи ШІ, банки та необанки можуть працювати над створенням цифрового середовища, унікального для кожного користувача, що в свою чергу покращує загальний досвід банківської справи.

1.5 Постановка завдання

Буде досліджено такі задачі:

1. Проведення огляду та аналізу методів для виявлення аномалій у даних.
2. Знаходження найкращих параметрів для моделей та методів машинного навчання для виявлення аномальної поведінки клієнта.
3. Виявлення аномальної поведінки методами машинного навчання з вчителем та без вчителя, використовуючи два різних набори даних: Credit Card Fraud Detection, та Bank Marketing.
4. Формування висновків щодо використання того, чи іншого методу або моделі для виявлення аномалій на основі результатів метрик оцінки якості.

1.6 Висновки до розділу 1

Сучасні технології значним чином покращують роботу банківським працівникам, котрі спеціалізуються на виявлення нетипової поведінки клієнта, чи, певної банківської процедури. Навіть більше, ці ж технології у найближчому майбутньому можуть повністю замінити людський ресурс у вирішенні різних типів ризиків у банківській сфері.

В даному розділі наявний огляд застосування методів штучного інтелекту для покращення банківських операцій, при цьому гарантуючи безпеку та надійність у збереженні конфіденційних даних клієнтів. А також основні терміни, що зустрічаються у банківській сфері, та, ризики, що спричиняються поведінкою, не схожою на поведінки інших клієнтів.

РОЗДІЛ 2. МАТЕМАТИЧНІ МЕТОДИ ТА МОДЕЛІ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ У ДАНИХ

2.1 Поняття аномалії

Виявлення аномалій, відоме також як виявлення викидів, новизни та шуму - це завдання визначення за допомогою спеціалізованих моделей випадків, характеристики яких значно відрізняються від отриманих знань.

Виявлення викидів - це ідентифікація прикладу, що сильно не схожий на інші вхідні приклади. У цьому випадку, модель має доступ до повного набору даних з нормальними, так і аномальними прикладами.

Виявлення новизни - це ідентифікації нового прикладу у множині нових спостережень, що відповідні до спостережень у головному наборі даних. У цьому випадку, модель має доступ тільки до початкового набору прикладів без викидів, і має виявити нові шаблони в нових спостереженнях. Такі моделі можна додатково навчити або оновити, коли з’являться нові екземпляри доступні та належать до категорії інкрементних або потокових моделей.

Розглянемо типи аномалій [4]:

1. Непередбачувані аномалії - виникають через помилки (систематичні та / або випадкові) у процесі збору даних.
2. Навмисні аномалії - виникають через визначені дії. Ці аномалії можуть давати цінну інформацію про набір даних, бо можуть виявити унікальні прояви чи тенденції.

Розглянемо аномалії часових рядів [5]:

1. Глобальні точкові - приклади, які значно відхиляється від усіх решти прикладів з набору.
2. Локальні точкові - приклади, які суттєво відхиляється лише відносно підмножини прикладів з основного набору.

3. Групові - набір прикладів, які не дотримуються закономірностей, що спостерігаються в інших прикладах з набору.
4. Контекстні - приклади, які значно відрізняється від усіх решти прикладів відповідно до певного контексту (наприклад, сезонність).

Вкажемо важливість процедури виявлення аномалій:

1. Покращується якість даних - будуть більш репрезентативними для правильних закономірностей.
2. Розширюються можливості прийняття рішень - як наслідок, з’являються більш обґрунтовані рішення та покращення результатів.
3. Оптимізується машинного навчання - забезпечуються точні та надійні прогнози.

Виявлення викидів може бути складним завданням, оскільки аномалії у чистому вигляді зустрічаються рідко, а характеристики нормальної поведінки можуть бути складними та динамічними.

Розглянемо методи для виявлення аномалій:

1. Візуалізація - за допомогою діаграми розкиду, гістограм, чи просто графіків лінійної залежності, можна помітити деякі аномалії, а саме сезонність, тренди.
2. Статистичні методи - це можуть бути довірчі інтервали, z-статистика.
3. Алгоритми машинного навчання (МН).

Розглянемо техніки для виявлення аномалій:

1. Навчання без вчителя: набори даних без міток. Модель використовує набір для самостійного виявлення закономірностей або аномалій. Найчастіше зустрічається в сценаріях глибокого навчання (deep learning), які покладаються на штучні нейронні мережі.
2. Навчання з вчителем: мічені набори даних, які складаються як з нормальних, так і аномальних прикладів. Оскільки в таких наборах наявна така проблема, як незбалансованість класів ці методи виявлення аномалій використовуються не часто.

2.2 Поняття машинного навчання

Штучний інтелект - технології, що здатні імітувати роботу мозку людини задля вирішення певних проблем [6].

Машинне навчання (МН) - це субдисципліна штучного інтелекту, яка спеціалізується на використанні даних та алгоритмів, які імітують спосіб навчання людського мозку, при цьому покращуючи точність [6].

Відношення між штучним інтелектом (Artificial Intelligence) та машинним навчанням (Machine Learning) зображено на рис. 2.1:

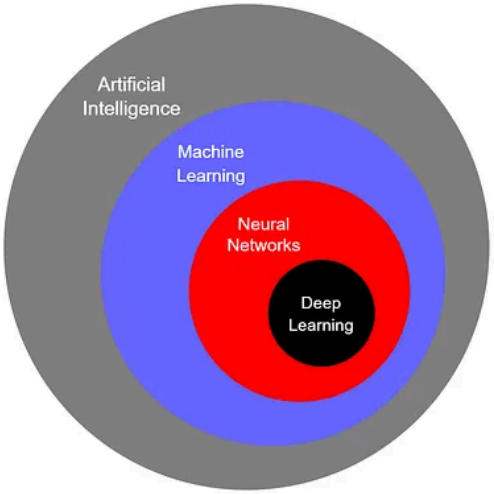


Рисунок 2.1 - Відношення між штучним інтелектом (Artificial Intelligence) та машинним навчанням (Machine Learning) [7]

Каліфорнійський університет у Берклі розбиває систему навчання алгоритму машинного навчання на три важливих етапи [8]:

1. Процес прийняття рішення: алгоритми машинного навчання призначені, щоб прогнозувати чи класифікувати на основі даних (мічених чи не мічених), та робити оцінку патернів взаємозв’язків між ознаками.
2. Функція оцінки помилок: оцінює прогноз моделі.
3. Процес оптимізації моделі: ваги моделі коригуються для зменшення розбіжності між правильним прогнозом і оцінкою моделі.

Алгоритм буде повторювати цей ітеративний процес «оцінки та оптимізації», поки не буде досягнуто порогу точності.

Машинне навчання охоплює широкий спектр завдань, які можна класифікувати за різними критеріями. Розглянемо основні задачі машинного навчання:

1. Класифікація - передбачає розподіл об'єктів у певні категорії або класи на основі їхніх характеристик. Приклади задач класифікації: розпізнавання спаму в електронній пошті, класифікацію зображень або діагностику захворювань на основі медичних даних.
2. Регресія - передбачення числових значень на основі вхідних даних. Задачі регресії: передбачення цін на нерухомість, прогнозування продажів або оцінка ризику кредитування.
3. Кластеризація - полягає у групуванні об'єктів на основі схожості між ними, без попередньо відомих міток. Приклади задач кластеризації: сегментація ринку, виявлення шахрайства або аналіз біологічних даних.
4. Пониження розмірності даних - спрямовані на зменшення кількості змінних у даних, зберігаючи при цьому важливу інформацію. Це допомагає візуалізувати дані, зменшити шум і покращити продуктивність моделей. Приклад задачі пониження розмірності даних - t-SNE.
5. Виявлення аномалій - спрямоване на ідентифікацію рідкісних або незвичайних спостережень, які відрізняються від більшості даних. Приклади включають виявлення шахрайських транзакцій.

2.3 Моделі і методи навчання з вчителем для виявлення аномалій

2.3.1 k-Nearest Neighbors

Алгоритм k-найближчих сусідів (k-Nearest Neighbors (k-NN)) є непараметричним методом навчання з вчителем, розроблений Евелін Фікс і Джозефом Ходжесом у 1951 році. Через деякий час, був дороблений Томасом Ковером. Метод k-NN використовується для класифікації (визначає приналежність точки до певного класу) та регресії (визначає числову властивість точки) [11]:

Переваги методу k-NN полягають у простоті реалізації й точності прогнозу і вимагає лише значення k (кількість найближчих сусідів) та метрику відстані між будь-якими двома точками з набору.

До недоліків можна віднести схильність до перенавчання, та, проблему великої розмірності даних.

Розглянемо метрики відстані між двома точками (нехай $x_i, x_j \in X$ -набору даних):

1. Відстань Махаланобіса - $d(x_i, x_j) = \sqrt{(x_i - x_j)^T S (x_i - x_j)}$, де S - матриця коваріації між x_i, x_j.
2. Відстань Чебишева - $d(x_i, x_j) = \max_k |x_{ik} - x_{jk}|$.
3. Відстань Мінковського - $d(x_i, x_j) = (\sum_{k=0}^{n} |x_{ik} - x_{jk}|^p)^{1/p}$, p - порядок ($p = 2$ - відстань Евкліда, $p = 1$ - відстань Манхетена, $p \to \infty$ - відстань Чебишева).
4. Відстань Евкліда - $d(x_i, x_j) = (\sum_{k=0}^{n} |x_{ik} - x_{jk}|^2)^{1/2}$.

Наведемо алгоритм k-NN:

1. Обирається довільне k та цільова маркована точка.

2. Обчислюється відстань від цільової маркованої точки до кожної з маркованих точок навчальної вибірки.
3. Обирається k точок навчальної вибірки з мінімальною відстанню.
4. Обирається клас точки, який найчастіше трапляється серед k найближчих сусідів (задача класифікації), або обчислюється середнє значення міток (задача регресії).

Приклад роботи алгоритму k-NN (класифікація) зображено на рис. 2.2.

Рисунок 2.2 - Приклад роботи алгоритму k-NN (класифікація) [9]

2.3.2 Gaussian Naive Bayes Classifier

Гаусівський наївний байєсівський метод класифікації (Gaussian Naive Bayes Classifier (GNB)) - метод класифікації, який заснований на ймовірнісному підході та розподілу Гауса [10]. Цей метод допускає, що усі ознаки є незалежними між собою, а також мають гаусівський розподіл.

Переваги методу GNB полягають у простоті реалізації. Недоліки - метод розраховує на незалежність між ознаками, що на практиці рідко трапляється, та виникають проблеми з категоріальними ознаками.

Формула нормального розподілу (Гаусівського):

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}}\exp\{\frac{-(x_i-\mu_y)^2}{2\sigma_y^2}\}.$$

σ_y - дисперсія y, μ_y - математичне сподівання y.

Теорема Байєса:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_{1,},...,x_n|y)}{P(x_{1,},...,x_n)}.$$

$P(y)$ - апріорна вірогідність гіпотези y.

$P(y|x_1, ..., x_n)$ - апостеріорна вірогідність гіпотези y при настанні події $\{x_1, ..., x_n\}$.

$P(x_1, ..., x_n|y)$ - вірогідність настання події $\{x_1, ..., x_n\}$ при істинності гіпотези y.

$P(x_1, ..., x_n)$ - повна вірогідність настання події $\{x_1, ..., x_n\}$.

Апостеріорна ймовірність для кожного класу і клас з максимальною ймовірністю:

$$P(y|x_1, ..., x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y) \Rightarrow \hat{y} = \arg\max_y P(y)\prod_{i=1}^{n} P(x_i|y).$$

Приклад гаусівського нормального розподілу та графіки коробчатих діаграм для Gaussian Naive Bayes Classifier зображено на рис. 2.3.

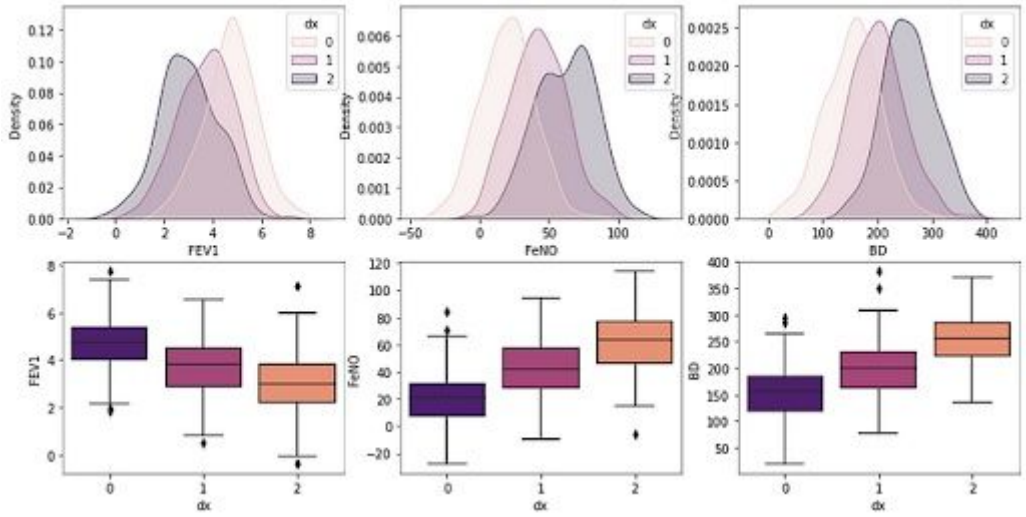


Рисунок 2.3 - Приклад гаусівського нормального розподілу та графіки коробчатих діаграм для Gaussian Naive Bayes Classifier [10]

Простота та ефективність Gaussian Naive Bayes Classifier роблять його привабливим для багатьох практичних задач. Однак, його припущення про незалежність ознак не завжди відповідає реальності, що може вплинути на точність класифікації. Незважаючи на це, алгоритм часто показує кращі результати навіть у випадках, коли це припущення порушується.

2.3.3 XGBoost

eXtreme Gradient Boost (XGBoost) - це сучасний, та дуже ефективний ітеративний ансамблевий алгоритм машинного навчання з вчителем, який поєднує у собі як послідовну, так і паралельну архітектуру [11]. Є розширеним варіантом Boosting Machine (який, у свою чергу, є підкласом Random Forest). Алгоритм XGBoost зображений на рис.2.4.

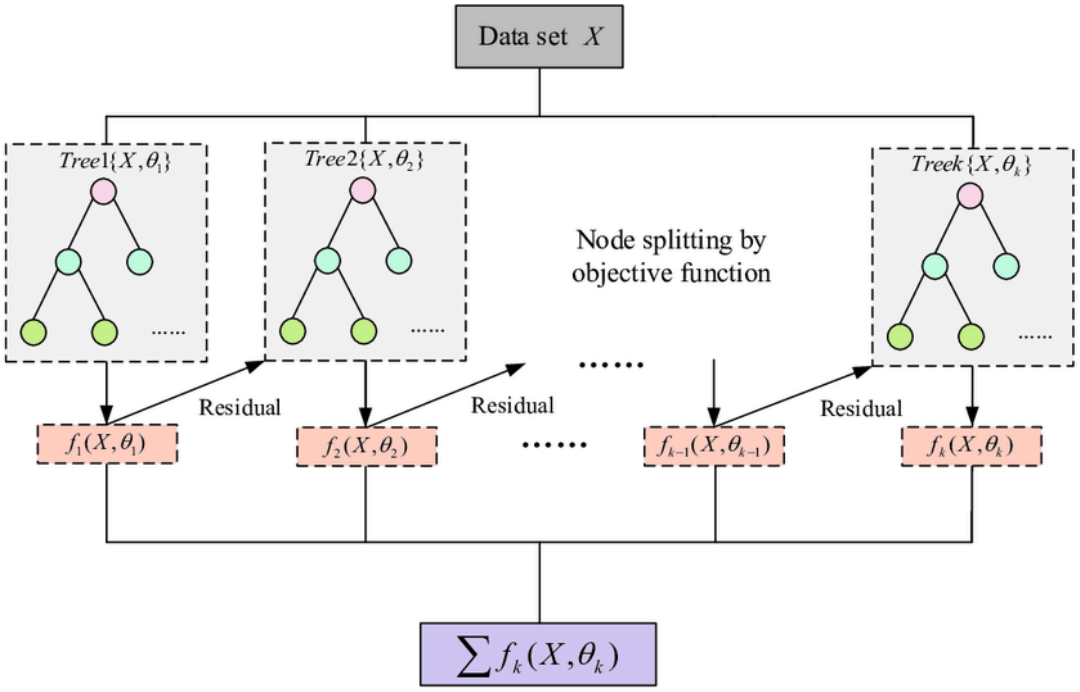


Рисунок 2.4 - Алгоритм XGBoost [12]

Послідовність заключається в тому, що даний алгоритм використовує адитивну стратегію, суть якої додавати по одному дереву за один раз та послідовно навчати слабкі (simple) дерева рішень. Це позитивно впливає на точність прогнозування моделі.

Паралельність призначена підвищити ефективність використання ресурсів. Для цього використовується градієнти першого та другого (Гессіан) порядків щоб мінімізувати цільову функцію втрат.

Розглянемо ключові аспекти XGBoost:

1. Градієнтний бустинг - це техніка, що створює послідовність моделей, зазвичай дерев рішень, де кожна нова модель намагається виправити помилки попередньої. Моделі комбінуються для покращення загальної продуктивності.
2. Регуляризація - XGBoost включає параметри регуляризації для контролю за складністю моделі та запобігання перенавчанню. L1 (Lasso) і L2 (Ridge) регуляризація застосовуються для зменшення складності моделі.

3. Паралельна обробка - XGBoost реалізує паралельне будування дерев рішень, що значно прискорює процес навчання порівняно з традиційними реалізаціями градієнтного бустингу.
4. Обробка пропусків - XGBoost ефективно обробляє відсутні значення у вхідних даних, автоматично знаходячи найкращі шляхи для пропущених значень під час навчання.
5. Зважування зразків - XGBoost дозволяє призначати ваги різним зразкам даних, що особливо корисно при роботі з незбалансованими наборами даних.
6. Shrinkage та Rate Dropout - XGBoost включає додаткові параметри, такі як shrinkage (параметр зменшення ваги нового дерева) та rate dropout (відсоток дерев, які випадково пропускаються під час навчання), що допомагають покращити продуктивність і зменшити ризик перенавчання.
7. Цільові функції - XGBoost підтримує широкий спектр цільових функцій (наприклад, регресія, класифікація, ранжування), що робить його універсальним для різних типів задач.

Розглянемо процес роботи XGBoost:

1. Ініціалізація - модель починає з передбачення, яке є середнім значенням цільової змінної для регресії або логарифмом відношення шансів для класифікації.
2. Побудова дерев - на кожній ітерації створюється нове дерево, яке намагається мінімізувати залишкову помилку попередньої моделі. Це досягається за допомогою обчислення градієнтів функції втрат для кожного зразка.
3. Оновлення передбачень - ваги для нових дерев додаються до загального передбачення. Процес повторюється, поки не досягне заданої кількості дерев або не буде досягнуто певного критерію зупинки.

4. Регуляризація та обрізка - під час навчання XGBoost використовує параметри регуляризації та обрізки, щоб зменшити ризик перенавчання, роблячи модель більш узагальненою.

Завдяки своїм перевагам, XGBoost широко використовується у задачах класифікації, регресії, виявлення аномалій, ранжування та інших областях. Його гнучкість та ефективність роблять його вибором багатьох фахівців з даних для побудови високопродуктивних моделей.

Прогнозований результат: $\hat{y} = \sum_{k=1}^{N} f_k(X, \theta_k)$, де f_k - результати k-го дерева.

Цільова функція: $F(\theta) = loss + reg$, де $loss = \sum_{i=1}^{n} l(y_i, \hat{y}_i)$ - функція втрат, $reg = \sum_{k=1}^{K} \Omega(f_k)$ - функція регуляризації.

Градієнтний бустинг - $F^{(t)} = F^{(t-1)} + \Omega(f^{(t)})$, t - номер ітерації.

2.3.4 Neural Networks

Нейронна мережа (Neural Network (NN)) - це обчислювальна модель, схожа до біологічних нейронних мереж, зокрема мозку. Вона складається з великої кількості взаємопов'язаних одиниць, які називаються нейронами або вузлами, що працюють разом для обробки інформації. Основна мета нейронної мережі - навчитися розпізнавати складні патерни в даних через процес навчання [16]. Схема нейронної мережі (a) та схема нейрону (b) зображені на рис. 2.5.

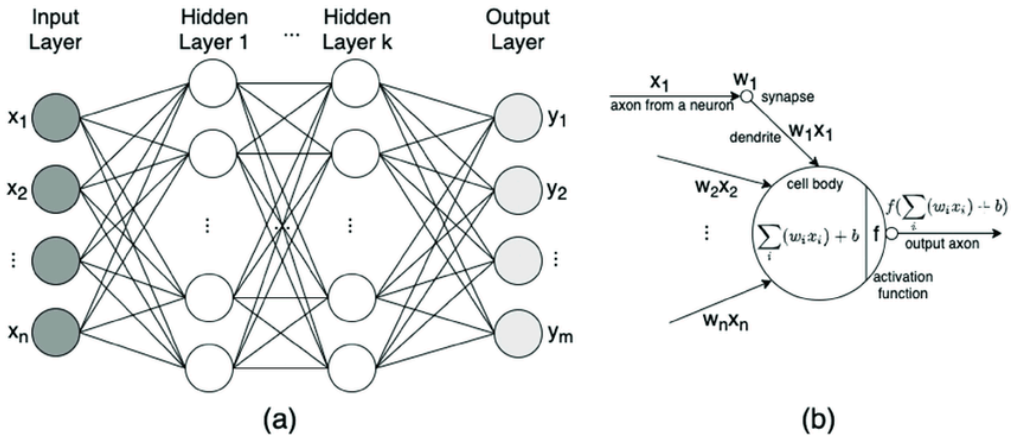


Рисунок 2.5 - Схема нейронної мережі (a) та схема нейрону (b) [13]

Нейронна мережа складається з трьох основних типів шарів:

1. Вхідний шар: це перший шар, який приймає початкові дані. Кількість нейронів у вхідному шарі відповідає кількості ознак або атрибутів у вхідних даних.
2. Приховані шари: ці шари розташовані між вхідним і вихідним шарами. Нейрони в прихованих шарах обробляють вхідні дані, застосовуючи ваги та функції активації для перетворення їх в корисні сигнали. Можливе використання кількох прихованих шарів, що створює багатошарову нейронну мережу, відому як багатошаровий перцептрон (MLP). Ці шари відповідають за виявлення складних патернів і залежностей у даних.
3. Вихідний шар: останній шар, який генерує прогноз або результат. Кількість нейронів у вихідному шарі залежить від специфіки задачі, наприклад, один нейрон для задач регресії або кілька нейронів для задач класифікації.

Кожен нейрон у нейронній мережі обчислює зважену суму своїх вхідних сигналів і передає результат через функцію активації, яка вводить нелінійність у модель і дозволяє нейронній мережі навчатися складним паттернам:

$$a_j^{(i)} = f(b_j + \sum_{i=1}^{n} w_{ji} x_i).$$

$a_j^{(i)}$ - вихід j-го нейрону i-го шару.

x_i - вхідні значення або ознаки.

w_{ji} - ваги.

b_i - байєс (зміщення) i-го шару.

Суть навчання нейронної мережі полягає в налаштуванні ваг з'єднань між нейронами з метою мінімізації помилки у прогнозах, що досягається за допомогою зворотного поширення помилки (backpropagation), а також одного з методів оптимізації - стохастичного градієнтний спуск (SGD) або Adam.

Під час навчання мережа отримує дані, проходить через всі шари, обчислює помилку на виході та коригує ваги, щоб зменшити цю помилку. Цей процес повторюється багаторазово з різними підходами до оптимізації, регуляризації та налаштування гіперпараметрів для покращення моделі.

Розглянемо функції активації:

1. Лінійна - $f(x) = x$
2. ReLU(Rectified Linear Unit) - $f(x) = \max(0, x)$
3. tanh - $f(x) = \tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$
4. Sigmoid - $f(x) = \sigma(x) = 1/(1 + e^{-x})$
5. SoftMax - $f(x) = e^x/(\sum_{k=1}^{K} e^{x_k})$

Архітектурні покращення містять в собі техніку Dropout, яка випадково відключає нейрони під час навчання, допомагає запобігти перенавчанню. Також до архітектурних покращень належать L1 та L2 регуляризація, що додають штраф за величину ваг у функцію втрат, зменшуючи складність моделі, а Data augmentation створює додаткові навчальні приклади шляхом застосування випадкових трансформацій до оригінальних даних.

Нейронні мережі використовуються для вирішення широкого спектру задач, включаючи обробку природної мови, класифікацію зображень,

розпізнавання мови, прогнозування часових рядів завдяки їхній здатності виявляти та узагальнювати складні патерни у великих обсягах даних.

2.4 Моделі і методи навчання без вчителя для виявлення аномалій

2.4.1 DBSCAN

Просторова кластеризація даних із шумом на основі щільності (Density-based spatial clustering of applications with noise (DBSCAN)) - алгоритм кластеризації, що запропонували Мартін Естер, Ганс-Петер Крігель, Йорґ Сандер та Сяовей Су у 1996 році.

Це алгоритм на основі щільності, де користувачеві необхідно визначити два параметри: ε та мінімальну кількість точок для створення кластера - $minPts$[14]:

1. Обирається випадкова базова точка P, і алгоритм обчислює відстань між цією точкою та кожною іншою точкою в наборі даних.
   1.1. Якщо відстань між двома точками в наборі даних $\leq$ ε, ці точки стають сусідами. Якщо в ε-околі P є принаймні стільки точок, скільки ми визначили як $minPts$ (вона називається щільною областю), створюється кластер.
   1.2. Якщо умова $minPts$ не виконується, це означає, усі точки позначені як шум. Однак ці шумові точки можуть бути знайдені пізніше в достатньому розмірі ε-околиці іншої точки та стати частиною іншого кластера.
2. Перевірка, чи може алгоритм розширити цей кластер, чи має перейти до наступної точки P' за межами цього кластера.
   2.1. Якщо перевірка позитивна, алгоритм розширює цей кластер для кожної точки в ε-околі P'.

2.2. Якщо кластер розширено до максимального розміру, кожна точка позначена як відвідана .

3. Якщо є ще недосліджені, то повернутися до пункту 1, інакше алгоритм припиняє свою роботу.

2.4.2 Gaussian Mixture Models

Суміш гаусівських моделей (Gaussian Mixture Models (GMM)) - використовується для визначення ймовірності належності точки до певного кластеру. Є методом м'якої кластеризації. Кількість моделей (гаусівських) у суміші - k (скільки і кластерів). Кожна модель описується математичним сподіванням μ, коваріаційною матрицею Σ, ймовірністю суміші (від цього залежить гаусівська функція) - π [15].

Нехай x - це набір точок даних, D - кількість вимірів.

Сума ймовірностей суміші: $\sum_{k=1}^{K} \pi_k = 1$.

Функція гаусівської щільності:

$$N(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}}\exp\{-\frac{1}{2}(x - \mu)^T\Sigma^{-1}(x - \mu)\}.$$

Функція щільності після логарифмування:

$$\ln N(x|\mu, \Sigma) = -\frac{D}{2}\ln 2\pi - \frac{1}{2}\ln\Sigma - \frac{1}{2}(x - \mu)^T\Sigma^{-1}(x - \mu).$$

$p(z_{nk} = 1|x_n)$ - вірогідність того, що точка належить k-му гаусівському розподілу.

Прихована змінна $z = \{z_1, ..., z_k\}$.

Рівняння ймовірності гаусівської суміші - $\pi_k = p(z_k = 1)$. $z_k = \{1, 0\}$ - 1, якщо точка належить k-му розподілу, 0 - інакше.

Рівняння функцій Гауса для z: $p(x_n|z) = \prod_{k=1}^{K} N(x_n|\mu_k, \Sigma_k)^{z_k}$.

Розглянемо алгоритм максимізації очікування (expectation maximization):

1. Ініціалізуємо $\Theta = [\pi, \mu, \Sigma]^T$.
2. Е-крок. Оцінюємо:
$$Q(\Theta^*, \Theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} p(z_k = 1|x_n)[\ln \pi_k + \ln N(x_n|\mu_k, \Sigma_k)].$$
3. М-крок. Знаходимо $\theta^* = \arg\max_{\theta} Q(\theta^*, \theta)$. Для того, щоб у сумі π давали одиницю, потрібно додати множник Лагранжа:
$$Q(\Theta^*, \Theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk})[\ln \pi_k + \ln N(x_n|\mu_k, \Sigma_k)] - \lambda(\sum_{k=1}^{K} \pi_k - 1).$$
4. Знаходимо градієнт Q по Θ та прирівнюємо до 0: $\frac{\partial Q(\Theta^*, \Theta)}{\partial \Theta} = 0$. Розв'язавши це рівняння, знаходиться Θ^*.
5. Повертаємось до пункту 1, якщо Θ^* та/або Q не стабілізуються.

2.4.3 Local Outlier Factor

Локальний фактор викидів (Local Outlier Factor (LOF)) - даний алгоритм розробили Маркус М. Бройніг, Ганс-Пітер Крігель, Раймонд Т. Нг і Йорг Сандер у 2000 році. Він призначений для пошуку аномалій у наборах даних шляхом вимірювання локального відхилення певної точки від її сусідів. [16]. Параметри, що визначають поняття щільності - $MinPts$ (мінімальна кількість точок у кластері) і розмірність визначають поріг щільності (threshold) для роботи алгоритму кластеризації.

Точка p у наборі даних D є DB(pct, $dmin$) є викидом, якщо хоча б процент pct об'єктів у D лежить більше, ніж відстань $dmin$ від p, тобто потужність множини $\{q \in D \mid d(p, q) \leq dmin\}$ менше або дорівнює $(100 - pct)\%$ від розміру D.

Для будь-якого натурального числа k (k-відстань об’єкта p), позначена як $k\_distance(p)$, визначається як відстань $d(p, o)$ між p і $o \in D$ така, що:

1. принаймні k об’єктів $o' \in D \setminus \{p\}$ виконується $d(p, o') \leq d(p, o)$
2. щонайбільше для $k - 1$ об’єктів $o' \in D \setminus \{p\}$ виконується $d(p, o') < d(p, o)$.

Відстань досяжності об'єкта p щодо об'єкта o - $reach\_dist_k(p, o) = \max\{k\_distance(o), d(p, o)\}$.

Щільність локальної досяжності p (the local reachability density of p)

$$lrd_{MinPts}(p) = 1/\left(\frac{\sum_{o \in N_{MinPts}(p)} reach\_dist_{MinPts}(p, o)}{|N_{MinPts}(p)|}\right)$$

Local outlier factor - $LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}$

2.4.4 OPTICS

Метод впорядкування точок для визначення структури кластеризації (Ordering Points To Identify the Clustering Structure (OPTICS)) - метод кластеризації, основна ідея якого схожа на DBSCAN, але він вирішує одну з основних слабостей DBSCAN: проблему визначення значущих кластерів в наборах даних різної щільності [17].

Так само, як і DBSCAN, потребує параметри $MinPts$ та ε.

Для кожної точки визначається основна відстань $core\_dist_{\varepsilon, MinPts}(p)$, що є найменшою відстанню до $MinPts$.

Відстань досяжності від o до p $reach\_dist_{\varepsilon, MinPts}(o, p)$ - або відстань між o та p $(d(o, p))$, або основна відстань p $(core\_dist_{\varepsilon, MinPts}(p))$, в залежності від того, що більше. На рис. 2.6 відображено схематично відстані між точками.

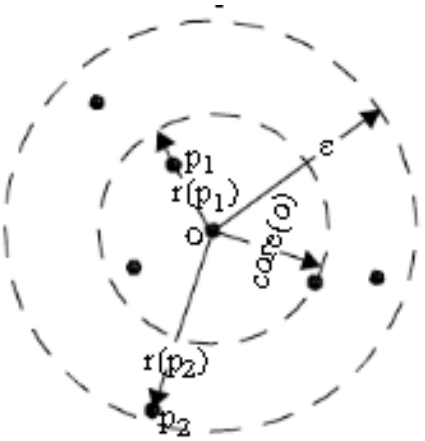


Рисунок 2.6 - Схематичне зображення відстані для алгоритму OPTICS [17]

Розглянемо алгоритм OPTICS (DB - база даних, ε, MinPts):

1. Для кожної точки p з бази даних відстань досяжності позначити як UNDEFINED
2. Для кожної необробленої точки p з бази даних знайти найближчих сусідів N в ε-околі, позначити p як оброблену, додати її у впорядкований список
   2.1. якщо основна відстань точки p != UNDEFINED, визначити seeds як пуста черга з пріоритетом, викликати функцію update (N, p, seeds, ε, MinPts)
      2.1.1. для кожної наступної q в Seeds знайти N' найближчих сусідів, позначити p як оброблену, додати її у впорядкований список
         2.1.1.1. якщо основна відстань q != UNDEFINED, то update (N', q, Seeds, ε, MinPts)

Функція update (N - найближчі сусіди p, p, Seeds, ε, MinPts):

1. Обчислюється основна відстань (core-distance) для точки p з урахуванням параметрів ε і MinPts.
2. Для кожної точки o з множини сусідів N:
   2.1. Якщо точка o ще не була оброблена (processed), то обчислюється нове значення відстані досяжності new-reach-dist, яке представляє

максимум між основною відстанню точки p і відстанню між точками p та o.

2.2. Якщо відстань досяжності для точки o є невизначеною, що означає, що точка o не знаходиться в черзі на обробку (Seeds), то встановлюється нове значення відстані досяжності для точки o та вона додається до черги Seeds.

2.3. У випадку, якщо відстань new-reach-dist менша за поточне значення відстані досяжності для точки o, то відстань досяжності для точки o оновлюється на нове значення, а також оновлюється її позиція у черзі Seeds, якщо точка o вже знаходиться в цій черзі.

2.4.5 BIRCH

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) - це алгоритм кластеризації, який ефективно працює з великими наборами даних. BIRCH створює компактне і збалансоване представлення даних, використовуючи деревоподібну структуру. Основні концепції та формули BIRCH включають кластерні особливості (CF) та кластерні вузли (CN).

Cluster Feature для набору точок $X = \{x_1, ..., x_т\}$ визначається трьома величинами:

1. CF-вектор суми (Linear Sum): сума всіх векторів точок у кластері.
2. CF-вектор суми квадратів (Squared Sum): сума квадратів всіх векторів точок у кластері.
3. CF-лічильник (Number of points): кількість точок у кластері.

Формула для CF: $CF = (N, LS, SS)$, де:

1. N - кількість точок у кластері.
2. $LS = \sum_{i=1}^{N} x_i$ - лінійна сума векторів точок.

3. $SS = \sum_{i=1}^{N} x_i^2$ - сума квадратів векторів точок.

Розглянемо властивості CF:

1. Адитивність. Нехай $CF_1 = (N_1, LS_1, SS_1)$, $CF_2 = (N_2, LS_2, SS_2)$:

   $CF_{total} = CF_1 + CF_2 = (N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2)$.

2. Центр кластера - $C = \frac{LS}{N}$.

3. Радіус кластера - $R = \sqrt{\frac{SS}{N} - (\frac{LS}{N})^2}$.

CF-дерево - це збалансоване деревоподібне представлення даних, яке складається з внутрішніх вузлів (містить кілька підвузлів і кластерних особливостей для кожного підвузла) і листових вузлів (містить кілька кластерних особливостей і точки даних, які асоціюються з цими кластерними особливостями)

Розглянемо алгоритм BIRCH:

1. Будівництво CF-дерева. По черзі вводити кожну точку даних у CF-дерево. Якщо поточний листовий вузол може вмістити нову точку (з урахуванням порогу обмеження розміру), оновити відповідну кластерну особливість. Інакше, розділити листовий вузол і створити новий внутрішній вузол.
2. Конденсація дерева - пройти по CF-дереву, видалити чи об'єднати незначні кластерні вузли для зменшення розміру дерева.
3. Глобальна кластеризація - застосувати будь-який традиційний алгоритм кластеризації (k-means) до набору кластерних особливостей, які представляють агреговані дані.

BIRCH ефективно обробляє великі набори даних завдяки своїй здатності зводити дані до компактної форми, що дозволяє виконувати

кластеризацію на значно меншому наборі кластерних особливостей замість оригінальних даних.

2.4.6 Isolation Forest

Ізоляційні ліси (Isolation Forest (IF)) - ансамблевий метод для виявлення аномалій. Був розроблений Фей Тоні Луі, Чжоу Чжіхуа, Кай Мін Тін у 2008 році.

Термін ізоляція означає “відокремлення екземпляра від решти екземплярів”. Так як аномалій “небагато і вони різні”, вони більш сприйнятливі до ізоляції. У випадковому дереві, представленому даними, розбиття екземплярів повторюється рекурсивно, доки всі екземпляри не будуть ізольовані. Це випадкове розділення створює помітні коротші шляхи для аномалій (рисунок 2.7), оскільки:

1. Менша кількість випадків аномалій призводить до меншої кількості розділів - коротші шляхи в структурі дерева (a);
2. Випадки з відмінними значеннями атрибутів більш ймовірно будуть розділені на ранніх етапах поділу (b);

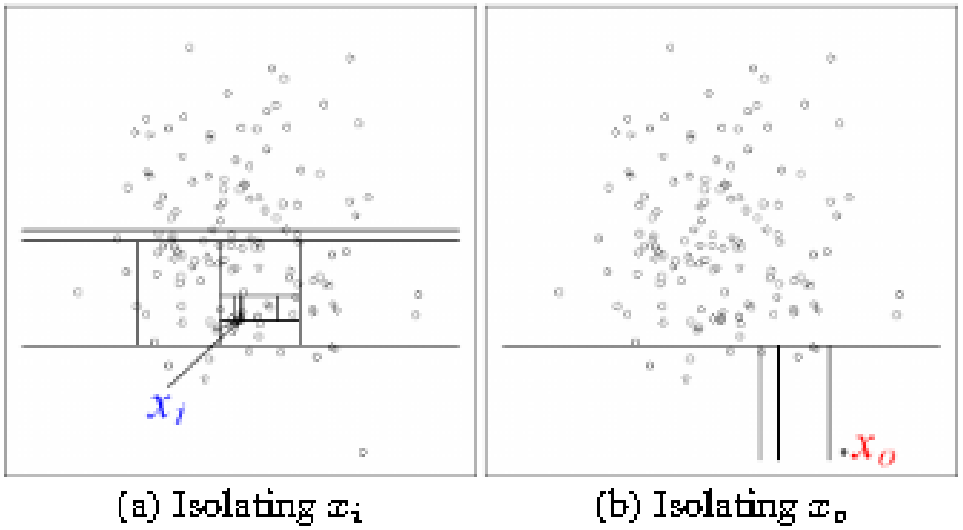


Рисунок 2.7 - Точки, ізольовані алгоритмом Isolation Forest. (a) нормальна точка вимагає дванадцяти випадкових поділів для ізоляції (b) аномалія вимагає лише чотирьох поділів для ізоляції

iTree - це повне двійкове дерево, де кожен вузол має 0 або 2 дочірніх вузли.

Отже, коли ліс випадкових дерев разом створює меншу довжину шляху для деяких конкретних точок, тоді вони, швидше за все, є аномаліями.

Розглянемо характеристики методу [20]:

1. Дозволяє будувати часткові моделі та використовувати підвибірки. Значна частина iTree, яка ізолює нормальні точки, не потрібна для виявлення аномалій, тому її не будують. Невеликий розмір вибірки створює кращі iTrees, зменшуючи ефект заболочування та маскування.
2. Не використовує вимірювання відстані чи щільності для виявлення аномалій, що усуває великі обчислювальні витрати.
3. Має лінійну часову складність із низькою константою та вимогами до пам'яті. Найефективніший існуючий метод забезпечує лише приблизну лінійну складність з високим використанням пам'яті.
4. Масштабується для обробки великих розмірів даних і проблем великої розмірності з багатьма нерелевантними атрибутами.

Нехай T вузол (зовнішній вузол без дочірніх вузлів або внутрішній вузол з одним тестом і точно двома дочірніми вузлами (T_l, T_r)) ізольованого дерева. Тест складається з атрибута q та розділеного значення ps таким чином, що тест $q < p$ ділить точки даних на T_l і T_r.

Довжина шляху $h(x)$ точки x вимірюється кількістю ребер x, що проходять через iTree від кореневого вузла, доки обхід не завершиться на зовнішньому вузлі.

Вибірка даних - $X = \{x_1, ..., x_n\}$, де n - кількість чисел з d -варійованого розподілу, щоб побудувати ізоляційне дерево. X рекурсивно ділиться шляхом випадкового вибору атрибута q та розділеного значення p , поки:

1. дерево не досягне межі висоти.
2. усі дані в наборі X будуть мати одні й ті самі значення.
3. $|X| = 1$

Якщо припустити, що всі екземпляри є різними, кожен екземпляр ізольовано до зовнішнього вузла, коли iTree готове повністю:

1. Кількість зовнішніх вузлів дорівнює n.
2. Кількість внутрішніх вузлів дорівнює $n - 1$.
3. Загальна кількість вузлів становить $2n - 1$.

Вимога до пам’яті обмежена і зростає лише лінійно з швидкістю n. Завдання виявлення аномалії - забезпечити рейтинг відображення ступені аномалії. Одним із способів виявлення аномалій - сортування точок даних відповідно до їх довжини шляху або балів аномалії (точками викиду в такому випадку будуть точки, що знаходяться у верхній частині списку).

Середня довжина шляху невдалого пошуку в бінарного дерева пошуку: $c(n) = 2(n - 1)(nH - 1)/n$

$H(i)$ - це гармонічне число, що оцінюється за допомогою $ln(i) + 0,5772156649$ (константа Ейлера).

$E(h(x))$ - середнє значенням $h(x)$ із набору ізольованих дерев, може приймати одне з значень:

1. $E(h(x)) \to c(n) \Rightarrow s \to 0.5$.
2. $E(h(x)) \to 0 \Rightarrow s \to 1$.
3. $E(h(x)) \to n - 1 \Rightarrow s \to 0$.

Змінна s монотонна до $h(x)$. Оцінка аномалії екземпляра x: $s(x, n) = 2^{-E(h(x))/c(n)}$. Взаємозв'язок між $E(h(x))$ та s, $0 < s \leq 1$, $0 < h(x) \leq n - 1$. На рис. 2.8 зображено залежність $E(h(x))$ від оцінки s.

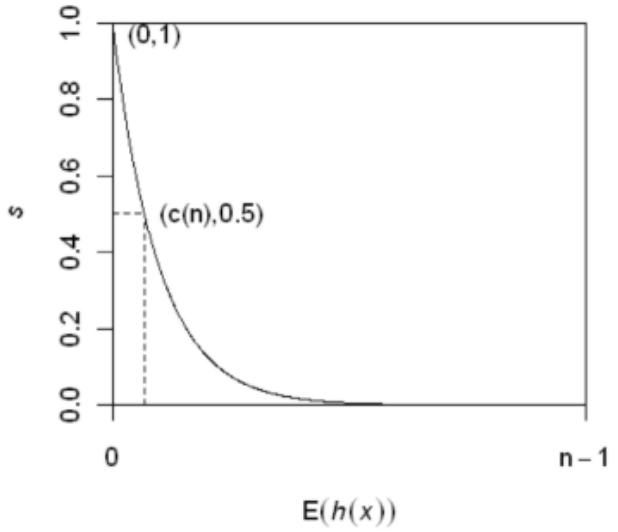


Рисунок 2.8 - Залежність $E(h(x))$ від оцінки s

Використовуючи оцінку аномалії s, можна зробити таку оцінку:

1. Якщо екземпляри повертають значення, близьке до 1, то вони точно є аномаліями.
2. Якщо екземпляри мають набагато менше ніж 0,5, то їх цілком безпечно розглядати як нормальні екземпляри.
3. Якщо всі екземпляри повертають s≈0,5, то повна вибірка не має чітких аномалій.

Потрібно звернути увагу, що обмеження висоти дерева автоматично встановлюється розміром підвибірки ψ: $l = ceiling(\log_2 \psi)$. ψ встановлюється зазвичай рівним 28 або 256.

Розглянемо алгоритм навчання $iForest(X, t, \psi)$ - складність рівна $O(t\psi \log \psi)$:

1. Вхідні дані: X, t - номер дерева, ψ - розмір підвибірки.
2. Вихідні дані: множина ізольованих дерев.
   2.1. Ініціалізувати $Forest$.
   2.2. Налаштувати межі висоти $l = ceiling(\log_2 \psi)$.
   2.3. Поки $1 \leq i \leq t$:
      2.3.1. $X' = sample(X, \psi)$
      2.3.2. $Forest = Forest \cup iTree(X', 0, l)$
   2.4. Повернути $Forest$

Алгоритм $iTree(X, e, l)$:

1. Вхідні дані: X, e - поточна висота дерева, l - межа висоти.
2. Вихідні дані: $iTree$.
   2.1. Якщо $e \geq l$ або $|X| \leq 1$:
      2.1.1. Повернути $exNode\{Size = |X|\}$.
   2.2. Інакше.
      2.2.1. Q - список атрибутів X.
      2.2.2. випадковим чином обрати $q \in Q$.
      2.2.3. випадковим чином обрати p з max та min значень q.
      2.2.4. $X_l = filter(X, q < p)$.
      2.2.5. $X_r = filter(X, q \geq p)$.
      2.2.6. Повернути.

$$inNode\{Left = iTree(X_l, e + 1, l),$$
$$Right = iTree(X_r, e + 1, l),$$
$$SplitAtt = q, 12: SplitValue = p\}.$$

Розглянемо алгоритм оцінювання $PathLength(x, T, e)$, його складність - $O(nt \log \psi)$:

1. Вхідні дані x - екземпляр, T - iTree, e - поточна довжина шляху (при першому виклику рівна 0)
2. Вихідні дані: довжина шляху для x
   2.1. Якщо T зовнішній вузол
      2.1.1. Повернути $e + c(T.size)$
   2.2. $a = T.splitAtt$
   2.3. Якщо $x_a < T.splitValue$
      2.3.1. Повернути $PathLength(x, T.left, e + 1)$
   2.4. Інакше
      2.4.1. Повернути $PathLength(x, T.right, e + 1)$

2.5 Метрики оцінки отриманих результатів

2.5.1 Для методів класифікації

TP - кількість істинно позитивних значень.

TN - кількість істинно негативних значень.

FP - кількість невірно позитивних значень.

FN - кількість невірно негативних значень.

Precision (точність) - здатність класифікатора не маркувати негативну точку як позитивну:

$$Precision = TP/(TP + FP).$$

Recall (повнота) - здатність класифікатора знаходити усі позитивні точки:

$$Recall = TP/(TP + FN).$$

F_1 - це середнє гармонічне значення точності та повноти, яке досягає свого найкращого значення при 1, а найгіршого - при 0:

$$F_1 = \frac{2 \times Precision}{Precision + Recall}.$$

ROC (receiver operating characteristic) крива - графік, що дозволяє оцінити ефективність бінарної класифікації, відображаючи відношення між часткою об'єктів, які правильно класифіковані, та часткою об'єктів, які помилково класифіковані, при зміні порогу класифікації.

AUC (Area under curve) - площа під ROC-кривою.

2.5.2 Для методів кластеризації

Якщо відомі розмічені дані:

1. **Однорідність (Homogeneity)** - максимальна, якщо кластер складається лише з об'єктів одного класу. Рівна одиниці мінус відношення ентропії класу за умови кластера до ентропії класу:

$$h = 1 - entropy(y_{true}|\hat{y})/entropy(y_{true}).$$

$entropy(p) = \sum_{i=1}^{N} p_i \ln p_i$, $\hat{y}$ - спрогнозовані мітки, y_{true} - істинні мітки.

2. **Повнота (Completeness)** - максимальна, якщо всі об'єкти з класу належать тільки до певного кластеру:

$$\mathrm{c} = 1 - entropy(\hat{y}|y_{true})/entropy(\hat{y}).$$

3. **V-measure** - середнє гармонічне метрик Homogeneity та Completeness:

$$v = (1 + \beta)hc/(h + c).$$

Якщо не розмічені дані:

1. **Силует (Silhouette)**:

$$(b(x) - a(x))/\ \max\{a(x), b(x)\}.$$

$a(x) = 1/(|C_X| - 1) \sum_{j \in C_X, x \neq j} d(x, j)$, C_X-кластер, якому належить точка x

, $b(x) = \min_{K \neq X} \frac{1}{|C_K|} \sum_{j \in C_K} d(x, j)$.

2. **Індекс Девіса-Булдіна (Davies-Bouldin DBI)** - $DBI = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j} \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}$, σ_i - середня відстань між точками у кластері i, c_i - центроїд кластеру i.
3. **Індекс Калінського-Харабаша (Calinski-Harabasz)** - $CH = \frac{Tr(B_k)}{Tr(W_k)} \times \frac{N-k}{k-1}$, $Tr(B_k)$ - слід міжкластерної матриці розкиду, $Tr(W_k)$ - слід матриці розкиду всередині кластеру, N - потужність вибірки, k - кількість кластерів.

2.6 Висновки до розділу 2

Даний розділ дав загальний огляд понять, методів та моделей, які використовуються для виявлення аномалій у даних.

Він почався з розгляду поняття аномалії та машинного навчання, подаючи базові відомості для подальшого розуміння теми. Після цього представлено моделі та методи навчання з вчителем для виявлення аномалій, такі як Isolation Forest, k-Nearest Neighbors, Gaussian Naive Bayes, XGBoost та Neural Networks. Кожен з цих підходів має свої переваги та недоліки, і вони можуть бути застосовані залежно від конкретних вимог та характеристик даних.

Далі розглянуто моделі та методи навчання без вчителя для виявлення аномалій, такі як DBSCAN, Gaussian Mixture Models, Local Outlier Factor, OPTICS та BIRCH. Ці методи не вимагають наявності міток аномалій у даних та можуть бути ефективними у виявленні складних аномалій або великих обсягів даних.

Нарешті, розділ завершується розглядом метрик оцінки результатів для якості моделей, які використовуються для виявлення аномалій.

Для задач кластеризації використовуються Precision, Recall, F1, AUC, ROC.

Для задач кластеризації використовуються коефіцієнт силуету, індекс Калінського-Харабаша та індекс Девіса-Булдіна - не потребують додаткової розмітки. Ті, що потребують додаткової розмітки - Homogeneity, Completeness, V-measure.

Ці метрики дозволяють оцінити ефективність моделей та визначити їхню придатність для конкретних завдань.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Опис використаних програмних засобів

3.1.1 Python

Python - це високорівнева, інтерпретована, мультипарадигмальна мова програмування зі строгою синтаксичною структурою, яка зосереджується на простому коді та простоті використання. Розроблена Гвідо ван Россумом і вперше випущена у 1991 році, Python швидко став однією з найпопулярніших мов програмування завдяки своїй ефективності, розширюваності та широкому спектру застосувань.

Розглянемо переваги Python:

1. Простота вивчення та використання: синтаксис Python простий і зрозумілий, що робить його ідеальним для початківців і професіоналів. За допомогою Python можна швидко створювати функціональні програми.
2. Велика спільнота та підтримка: Python має велику аудиторію розробників по всьому світу, що підтримують велику кількість бібліотек, модулів та інструментів для розширення його функціональності. Завдяки цьому Python підтримується на багатьох платформах.
3. Універсальність: Python підходить для різноманітних задач, а саме математичні обчислення, обробка даних, машинне навчання.
4. Великий вибір бібліотек: Python має велику кількість сторонніх бібліотек, таких як Pandas, Seaborn, NumPy, що допомагають розробникам прискорити розробку програм.

5. Інтеграція з іншими мовами: Python інтегрується з іншими мовами програмування, що дозволяє розробникам використовувати Python для розширення функціональності їхніх додатків.

Розглянемо недоліки Python:

1. Повільність: Python може бути повільним порівняно з іншими мовами програмування, такими як C++ або Java, особливо у випадках, коли велика кількість обчислень потребується виконати в реальному часі.
2. Обмеження в ресурсах: у деяких випадках Python може бути обмежений для розробки великих систем або програм, які вимагають великої кількості ресурсів.
3. Неідеальна підтримка мобільних платформ: хоча існують фреймворки, такі як Kivy, для розробки мобільних додатків на Python, підтримка для мобільних платформ не така повна, як для веб-або десктопних додатків.

3.1.2 Scikit-learn

Scikit-learn (скорочено від "scipy toolkit learn") - це відкрита бібліотека машинного навчання для Python [21]. Вона побудована на основі інших популярних бібліотек Python, таких як NumPy, SciPy та matplotlib, і надає простий та ефективний інтерфейс для виконання багатьох різноманітних завдань машинного навчання, таких як класифікація, регресія, кластеризація, зниження розмірності.

Основні переваги бібліотеки Scikit-learn включають:

1. Простий у використанні інтерфейс: Scikit-learn пропонує послідовний та легко зрозумілий API, що спрощує роботу з алгоритмами машинного навчання навіть для початківців.
2. Багатий вибір алгоритмів: бібліотека містить широкий спектр алгоритмів машинного навчання, включаючи методи класифікації, регресії, кластеризації, виявлення аномалій, підбору параметрів та інші.
3. Підтримка векторизованих операцій: використання NumPy та SciPy в якості базових бібліотек дозволяє Scikit-learn працювати з

векторизованими операціями, що призводить до ефективної обробки великих обсягів даних.

4. Документація та спільнота: Scikit-learn має добре документовану API та розширену спільноту користувачів, що сприяє швидкій підтримці та вирішенню проблем.
5. Відкритий код і безкоштовність: Scikit-learn є відкритою бібліотекою, що означає, що розробник може вільно використовувати, змінювати та поширювати її безкоштовно.

Хоча Scikit-learn має багато переваг, варто враховувати, що вона не підтримує побудову складних моделей або велику кількість даних, які можуть бути оброблені більш потужними інструментами, такими як TensorFlow або PyTorch. Однак, для більшості задач машинного навчання, особливо на початкових етапах дослідження, Scikit-learn є чудовим інструментом.

3.1.3 Seaborn

Seaborn - це бібліотека, надбудована над matplotlib для візуалізації даних [29]. Вона надає високорівневий інтерфейс для створення привабливих та інформативних статистичних графіків. Seaborn спрощує процес візуалізації даних, дозволяючи швидко створювати високоякісні графіки зі стандартними налаштуваннями, такими як кольорова палітра та стилізація.

Основні переваги бібліотеки Seaborn включають:

1. Простий у використанні інтерфейс: Seaborn має простий та зрозумілий API для створення різноманітних графіків з мінімальними зусиллями.
2. Привабливість графіків за замовчуванням: графіки, створені за допомогою Seaborn, зазвичай виглядають більш привабливо та професійно за замовчуванням, що полегшує їх використання в друкованих матеріалах або веб-сайтах.
3. Підтримка статистичних графіків: Seaborn має спеціальні функції для створення різноманітних статистичних графіків, таких як графіки

розподілу, бокс-плоти, діаграми розсіювання зі згладжуванням, що дозволяє швидко виводити та аналізувати дані.

4. Інтеграція з Pandas: Seaborn легко інтегрується з бібліотекою Pandas, що дозволяє безпосередньо використовувати DataFrame для візуалізації даних.
5. Корисні додаткові функції: бібліотека містить додаткові функції для візуалізації регресійних моделей, взаємодії з категоріальними даними та фасилітуючи порівняння між різними групами даних.

Незважаючи на ці переваги, варто зазначити, що Seaborn спеціалізується на статистичній візуалізації та не надає таких розширених можливостей для налаштування графіків, як matplotlib. Однак, вона чудово підходить для швидкої та ефективної візуалізації даних у великих дослідженнях чи аналізах.

3.1.4 Numpy

NumPy - це бібліотека, яка надає підтримку для роботи з багатовимірними масивами і матрицями [25]. Вона є однією з основних бібліотек у сфері наукових обчислень та аналізу даних у Python.

Розглянемо деякі основні можливості NumPy:

1. Масиви: NumPy надає об'єкт ndarray, який представляє собою багатовимірний масив однакового типу даних. Ці масиви дозволяють здійснювати ефективні операції з великими об'ємами даних.
2. Універсальні функції: NumPy містить велику колекцію універсальних функцій, які дозволяють виконувати математичні операції на масивах. Це включає арифметичні операції, тригонометричні функції, логарифми, експоненти та багато інших.
3. Індексація і зрізи: NumPy надає потужні засоби для індексації та вибору підмасивів з масивів, включаючи різноманітні методи індексації та булеві маски.

4. Векторизація: NumPy дозволяє виконувати векторні операції без необхідності використання циклів, що дозволяє здійснювати операції швидше та більш ефективно.
5. Лінійна алгебра: NumPy містить функції для роботи з лінійною алгеброю, такі як множення матриць, знаходження власних значень та векторів, розв'язання систем лінійних рівнянь та інші.

NumPy є важливим інструментом для розвитку наукових досліджень, обробки даних та створення складних алгоритмів у Python.

3.1.5 Pandas

Pandas - це бібліотека для обробки та аналізу даних, яка містить у собі спеціалізовані структури даних та функції, які дозволяють легко та ефективно маніпулювати табличними даними, що часто використовуються у наукових дослідженнях, аналізі даних та розвідці даних [24].

Основні структури даних у бібліотеці Pandas:

1. DataFrame: DataFrame - це основна структура даних в Pandas. Вона представляє собою двовимірну табличну структуру даних з рядками та стовпцями, схожу на аркуш даних у Excel чи таблицю у базі даних. DataFrame дозволяє легко індексувати, фільтрувати та агрегувати дані.
2. Series: Series - це одновимірна маркирована структура даних, що містить однорідні дані. Вона може бути розглянута як стовпець у DataFrame або один рядок у таблиці даних.

Основні можливості бібліотеки Pandas:

1. Завантаження та збереження даних: Pandas надає зручні функції для імпорту та експорту даних з різних форматів, включаючи CSV, Excel, SQL, JSON, HTML та інші.
2. Маніпуляція даними: вибірка, фільтрація, групування, об'єднання, зміна формату та очистка даних.

3. Обробка відсутніх значень: Pandas надає можливості для обробки та роботи з відсутніми значеннями в даних, що дозволяє ефективно управляти пропущеними даними.
4. Візуалізація даних.

Бібліотека Pandas є незамінним інструментом для роботи з даними у Python, особливо у випадках, коли потрібно виконати аналіз, очистку, обробку та візуалізацію табличних даних.

3.1.6 XGBOOST

XGBoost - це бібліотека для машинного навчання, яка надає інструменти для побудови та навчання моделей градієнтного бустингу [27]. Вона реалізована для декількох мов програмування, включаючи Python, R, Java, Scala та інші, але найбільш популярною і широко використовуваною є версія для Python.

Основні особливості бібліотеки XGBoost включають:

1. Підтримка градієнтного бустингу: XGBoost базується на алгоритмі градієнтного бустингу, який є потужним методом ансамблювання для підвищення якості прогнозування.
2. Широкий вибір функцій втрат і метрик оцінки: XGBoost надає різноманітні функції втрат і метрики оцінки, що дозволяють вибирати найкращий підхід для конкретної задачі.
3. Регуляризація: вбудована підтримка регуляризації допомагає уникнути перенавчання та покращити загальну роботу моделі.
4. Інтерпретованість моделі: XGBoost надає інструменти для аналізу важливості ознак і пояснення прогнозів, що дозволяє зрозуміти, як модель приймає рішення.
5. Підтримка паралельного навчання: XGBoost може працювати в паралельному режимі, що робить його ефективним у використанні на багатоядерних та розподілених системах.

Загалом, XGBoost - це потужна та ефективна бібліотека, яка займає важливе місце в арсеналі інструментів для машинного навчання та аналізу даних.

3.1.7 Google Colaboratory

Google Colaboratory, часто скорочено як Google Colab, це безкоштовна платформа на основі хмарних обчислень, яка дозволяє створювати, виконувати та спільно користуватися ноутбуками Jupyter. Вона пропонує середовище, в якому можна писати та виконувати код Python, а також проводити аналіз даних, навчання моделей машинного навчання, візуалізацію даних та багато іншого, все це безпосередньо в браузері, без необхідності встановлення та налаштування середовища розробки на власному комп'ютері.

Основні особливості Google Colab включають:

1. Хмарне середовище: використання хмарних обчислень Google дозволяє запускати код в середовищі Google, що забезпечує високу доступність та швидкість роботи.
2. Безкоштовні ресурси GPU та TPU.
3. Інтеграція з Google Drive: розробник може зберігати свої ноутбуки та дані на Google Drive та легко звертатися до них з Google Colab.
4. Спільна робота: Google Colab підтримує спільну роботу над ноутбуками Jupyter, що дозволяє ділитися кодом та результатами з іншими користувачами та отримувати зворотній зв'язок.
5. Широкий вибір бібліотек та фреймворків у Python.

Google Colab є дуже популярним середовищем для навчання, досліджень та роботи з даними, особливо для студентів, дослідників та аналітиків даних, які шукають зручний та доступний інструмент для виконання своїх завдань.

3.2 Попередня обробка наборів даних

3.2.1 Credit Card Fraud Detection

Набір даних містить транзакції, здійснені за допомогою кредитних карток у вересні 2013 року європейцями, які відбулися за два дні. Серед них 492 шахрайства з 284 807 звичайних операції. Тобто, даний датасет є незбалансованим.

Атрибути:

1. V1 - V28 - основні конфіденційні компоненти, отримані за допомогою PCA(Principal component analysis - техніка зниження розмірності даних).
2. Time - час(с) між кожною транзакцією і першою транзакцією.
3. Amount - сума транзакцій.

Class - цільова змінна, яка приймає значення 1 у разі шахрайства і 0 протилежному випадку.

Перші п’ять рядків датасету зображено на рис. 3.1.

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

Рисунок 3.1 - Перші п’ять рядків набору даних Credit Card Fraud Detection

Як видно з рис. 3.1, значення ознак V1-V28 перших п’яти прикладів є не дуже великими, проте Amount може бути рівним як 149.02, так і 2.08.

Описова статистика датасету Credit Card Fraud Detection до стандартизації зображена на рис. 3.2.

	count	mean	std	min	25%	50%	75%	max
Time	283726.0	94811.08	47481.05	0.00	54204.75	84692.50	139298.00	172792.00
V1	283726.0	0.01	1.95	-56.41	-0.92	0.02	1.32	2.45
V2	283726.0	-0.00	1.65	-72.72	-0.60	0.06	0.80	22.06
V3	283726.0	0.00	1.51	-48.33	-0.89	0.18	1.03	9.38
V4	283726.0	-0.00	1.41	-5.68	-0.85	-0.02	0.74	16.88
V5	283726.0	0.00	1.38	-113.74	-0.69	-0.05	0.61	34.80
V6	283726.0	-0.00	1.33	-26.16	-0.77	-0.28	0.40	73.30
V7	283726.0	0.00	1.23	-43.56	-0.55	0.04	0.57	120.59
V8	283726.0	-0.00	1.18	-73.22	-0.21	0.02	0.33	20.01
V9	283726.0	-0.00	1.10	-13.43	-0.64	-0.05	0.60	15.59
V10	283726.0	-0.00	1.08	-24.59	-0.54	-0.09	0.45	23.75
V11	283726.0	0.00	1.02	-4.80	-0.76	-0.03	0.74	12.02
V12	283726.0	-0.00	0.99	-18.68	-0.41	0.14	0.62	7.85
V13	283726.0	0.00	1.00	-5.79	-0.65	-0.01	0.66	7.13
V14	283726.0	0.00	0.95	-19.21	-0.43	0.05	0.49	10.53
V15	283726.0	0.00	0.91	-4.50	-0.58	0.05	0.65	8.88
V16	283726.0	0.00	0.87	-14.13	-0.47	0.07	0.52	17.32
V17	283726.0	0.00	0.84	-25.16	-0.48	-0.07	0.40	9.25
V18	283726.0	0.00	0.84	-9.50	-0.50	-0.00	0.50	5.04
V19	283726.0	-0.00	0.81	-7.21	-0.46	0.00	0.46	5.59
V20	283726.0	0.00	0.77	-54.50	-0.21	-0.06	0.13	39.42
V21	283726.0	-0.00	0.72	-34.83	-0.23	-0.03	0.19	27.20
V22	283726.0	-0.00	0.72	-10.93	-0.54	0.01	0.53	10.50
V23	283726.0	0.00	0.62	-44.81	-0.16	-0.01	0.15	22.53
V24	283726.0	0.00	0.61	-2.84	-0.35	0.04	0.44	4.58
V25	283726.0	-0.00	0.52	-10.30	-0.32	0.02	0.35	7.52
V26	283726.0	0.00	0.48	-2.60	-0.33	-0.05	0.24	3.52
V27	283726.0	0.00	0.40	-22.57	-0.07	0.00	0.09	31.61
V28	283726.0	0.00	0.33	-15.43	-0.05	0.01	0.08	33.85
Amount	283726.0	88.47	250.40	0.00	5.60	22.00	77.51	25691.16

Рисунок 3.2 - Описова статистика для змінних в датасеті Credit Card Fraud Detection до стандартизації

Щоб отримати статистику, як на рис. 3.2 для детальнішого дослідження ознак набору даних Credit Card Fraud Detection, використаємо функцією describe з бібліотеки pandas, що дає інформацію по кожній з ознак про кількість прикладів, математичне сподівання (mean), стандартне відхилення (std), мінімальні (min) та максимальні (max) значення, а також 0.25-квартиль, 0.5-квартиль, 0.75-квартиль.

Як видно з рис. 3.2, ознаки V1-V28 мають приблизно нормальний розподіл - $N(\mu \approx 0, \sigma \approx 1)$, де N - нормальний Гаусівський розподіл, μ - математичне сподівання, σ - стандартне відхилення. Time та Amount мають великі діапазони значень - [0, 172792] та [0, 25691] відповідно. Кількість видалених дублікатів у датасеті рівна 1081. Пропуски в даних відсутні. Також, набір даних потрібно стандартизувати за допомогою StandardScaler з бібліотеки scikit-learn, задля того, щоб підвищити ефективність моделей та методів. Кореляційна матриця набору даних зображена на рис. 3.3.

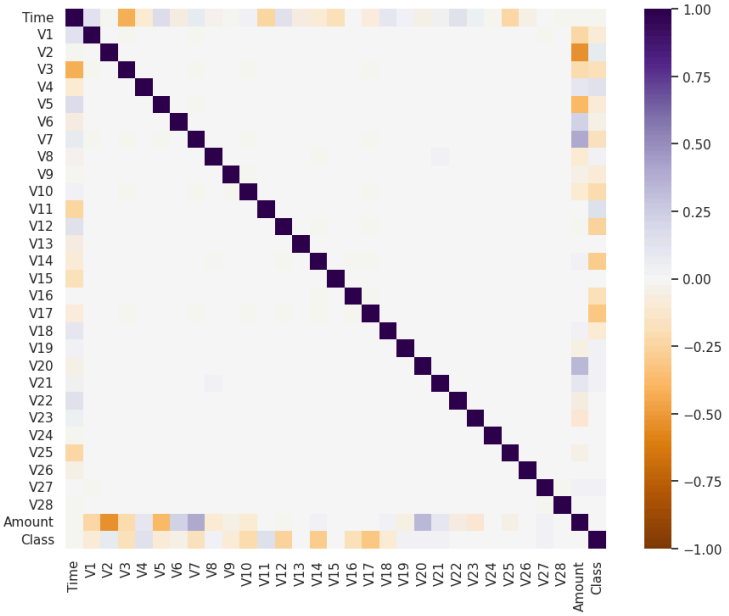


Рисунок 3.3 - Кореляційна матриця набору даних Credit Card Fraud Detection

Тобто, змінні V1 - V28 не мають високу кореляцію між собою, а, отже, дані не є мультиколінеарними.

Розподіл ознак V[1,2,3,4,7,10,11,12,14,16,17,18] зображено на рис. 3.4.

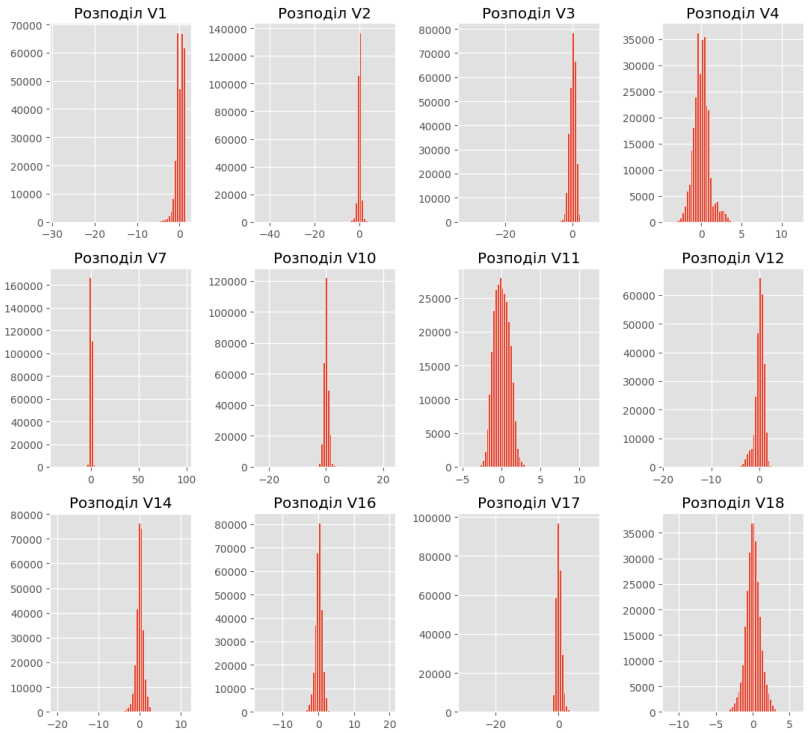


Рисунок 3.4 - Розподіл ознак V[1,2,3,4,7,10,11,12,14,16,17,18] з датасету Credit Card Fraud Detection

3.2.2 Bank Marketing

Дані були зібрані шляхом телефонних опитувань в португальській банківській установі.

Основні атрибути:

1. age - вік клієнта.
2. job - вид діяльності.
3. marital - сімейний статус.
4. education - рівень освіти.
5. default - наявність простроченого кредиту.
6. balance - середній щорічний баланс.
7. housing - наявність іпотеки.
8. loan - наявність персонального кредиту.
9. contact - тип комунікації.
10. day_of_week - день тижня останнього контакту.
11. month - місяць останнього контакту.
12. duration - тривалість останнього контакту у секундах.
13. campaign - кількість контактів з працівником установи
14. pdays - кількість днів, які минули після останнього контакту
15. previous - кількість контактів, зроблених у попередній компанії
16. poutcome - результат попередньої маркетингової кампанії.

Окрім цих атрибутів, є ще emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m, nr.employed, про які не було надано жодної інформації.

y (надалі або y, або Class) - цільова змінна, яка означає, чи надали клієнту строковий депозит.

Перші 5 рядків з датасету Bank Marketing зображені на рис. 3.5.

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	fri	...	2	999	0	nonexistent	-1.8	92.893	-46.2	1.313	5099.1	no
1	39	services	single	high.school	no	no	no	telephone	may	fri	...	4	999	0	nonexistent	1.1	93.994	-36.4	4.855	5191.0	no
2	25	services	married	high.school	no	yes	no	telephone	jun	wed	...	1	999	0	nonexistent	1.4	94.465	-41.8	4.962	5228.1	no
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	fri	...	3	999	0	nonexistent	1.4	94.465	-41.8	4.959	5228.1	no
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	mon	...	1	999	0	nonexistent	-0.1	93.200	-42.0	4.191	5195.8	no

5 rows × 21 columns

Рисунок 3.5 - Перші 5 рядків з датасету Bank Marketing

Розмір датасету - 4119 прикладів. З них ті, що Class рівний - 3668 прикладів, а тих, у яких Class є рівним одиниці - 451 приклади. Отже, і в цьому наборі даних наявний дисбаланс класів.

Розподіл ознак contact, education, job, poutcome зображено на рис. 3.6.

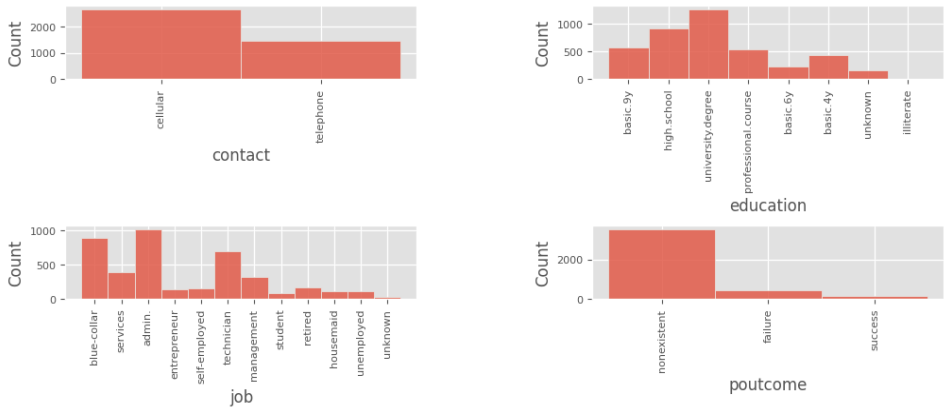


Рисунок 3.6 - Розподіл ознак contact, education, job, poutcome

Проаналізувавши рис. 3.8, доходимо до висновку, що кількість тих клієнтів, що вказували у полі contact (контакт) значення cellular (стільниковий телефон) є більшою за 20000. Найбільша кількість клієнтів мають university.degree (університетський ступінь) за ознакою education (освіта). За ознакою job (місце роботи) більшість клієнтів вказували admin.

Кодуємо дані за допомогою OrdinalEncoder з бібліотеки scikit-learn.

Основні статистики для датасету (кількість прикладів, математичне сподівання (mean), стандартне відхилення (std), мінімальні (min) та максимальні (max) значення, а також 0.25-квартиль, 0.5-квартиль, 0.75-квартиль) до стандартизацій Bank Marketing зображено на рис. 3.7.

	count	mean	std	min	25%	50%	75%	max
age	4119.0	40.11	10.31	18.00	32.00	38.00	47.00	88.00
job	4119.0	3.82	3.61	0.00	1.00	3.00	7.00	11.00
marital	4119.0	1.18	0.61	0.00	1.00	1.00	2.00	3.00
education	4119.0	3.78	2.15	0.00	2.00	3.00	6.00	7.00
default	4119.0	0.20	0.40	0.00	0.00	0.00	0.00	2.00
housing	4119.0	1.08	0.98	0.00	0.00	2.00	2.00	2.00
loan	4119.0	0.35	0.74	0.00	0.00	0.00	0.00	2.00
contact	4119.0	0.36	0.48	0.00	0.00	0.00	1.00	1.00
month	4119.0	4.29	2.31	0.00	3.00	4.00	6.00	9.00
day_of_week	4119.0	2.01	1.39	0.00	1.00	2.00	3.00	4.00
duration	4119.0	256.79	254.70	0.00	103.00	181.00	317.00	3643.00
campaign	4119.0	2.54	2.57	1.00	1.00	2.00	3.00	35.00
pdays	4119.0	960.42	191.92	0.00	999.00	999.00	999.00	999.00
previous	4119.0	0.19	0.54	0.00	0.00	0.00	0.00	6.00
poutcome	4119.0	0.92	0.37	0.00	1.00	1.00	1.00	2.00
emp.var.rate	4119.0	0.08	1.56	-3.40	-1.80	1.10	1.40	1.40
cons.price.idx	4119.0	93.58	0.58	92.20	93.08	93.75	93.99	94.77
cons.conf.idx	4119.0	-40.50	4.59	-50.80	-42.70	-41.80	-36.40	-26.90
euribor3m	4119.0	3.62	1.73	0.64	1.33	4.86	4.96	5.04
nr.employed	4119.0	5166.48	73.67	4963.60	5099.10	5191.00	5228.10	5228.10
Class	4119.0	0.11	0.31	0.00	0.00	0.00	0.00	1.00

Рисунок 3.7 - Описова статистика для змінних датасеті Bank Marketing до стандартизації

Як видно з рис. 3.7, лише ознаки loan, previous,emp.var.rate мають приблизно нормальний розподіл - $N(\mu \approx 0, \sigma \approx 1)$. age, duration, pdays, cons.price.idx, nr.employed мають великі діапазони значень. Тому, стандартизуємо, за допомогою StandardScaler з бібліотеки scikit-learn.

Матриця кореляцій між ознаками, та, цільовою змінною, зображена на рис. 3.8.

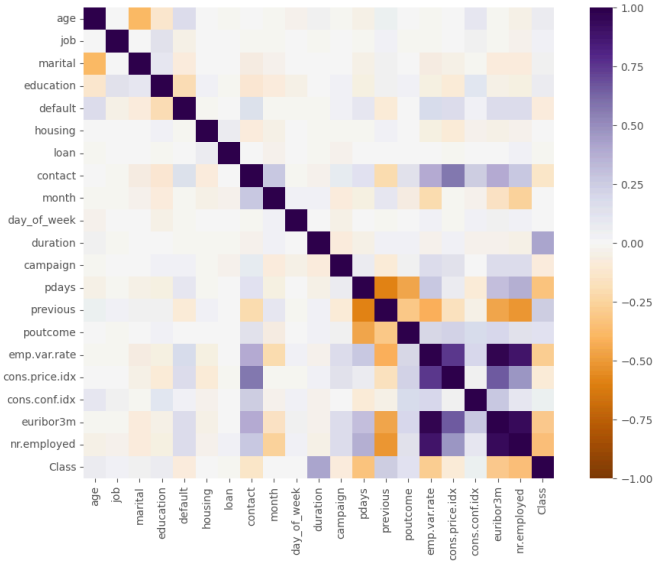


Рисунок 3.8 - Матриця кореляцій між ознаками, та, цільовою змінною до стандартизації датасету Bank Marketing

3.2.3 Покращення наборів даних

По-перше, потрібно позбутися дисбалансу класів. Є декілька підходів до вирішення цієї проблеми:

1. Взяти більше зразків із класу меншин з набору даних.
2. Змінити функцію втрат.
3. Oversampling (Підвищити вибірку) класу меншості.
4. Undersampling (Понизити вибірку) класу більшості.

Отже, буде обрано метод Undersampling для обох наборів даних. Після процедури Undersampling, розмір датасету Credit Card Fraud Detection зменшився аж до 946 прикладів, які рівномірно розподілені по двух класах. Аналогічно, датасет Bank Marketing зменшився, проте кількість прикладів рівна 902, і, приклади рівномірно розподілилися по класах.

Кореляційні матриці після Undersampling для наборів даних Credit Card Fraud Detection (зліва), та Bank Marketing (справа) зображені на рис. 3.9.

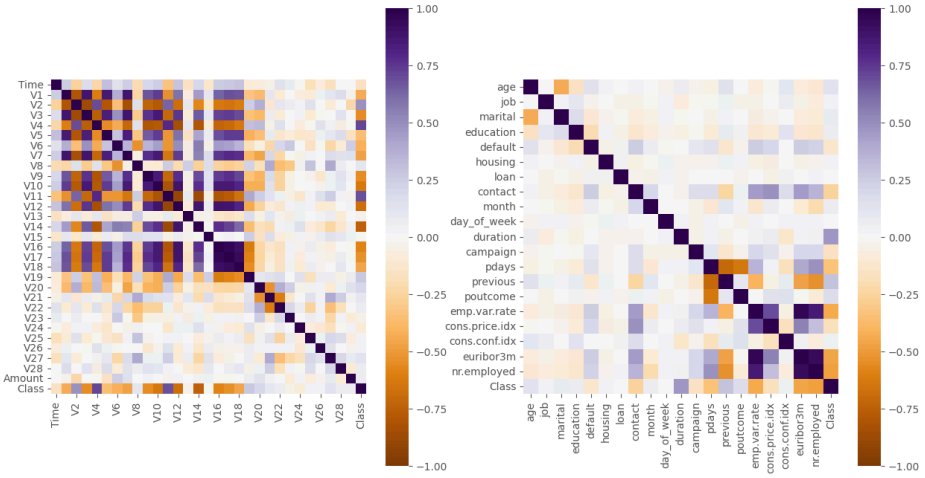


Рисунок 3.9 - Кореляційні матриці після Undersampling для наборів даних Credit Card Fraud Detection (зліва), та Bank Marketing (справа)

По-друге, датасет Credit Card Fraud Detection має 30 ознак, а датасет Bank Marketing - 16. Для методів кластеризації DBSCAN, OPTICS, BIRCH це є проблемою, тому потрібно знизити розмірність наборів даних, спроектувавши їх на площину 2-D. Для цього скористаємося технікою

нелінійного пониження розмірності t-distributed stochastic neighbor embedding (t-SNE).

Графіки точок на 2-D площині, утворених за допомогою методу t-SNE для Credit Card Fraud Detection (зліва) та Bank Marketing (справа) зображені на рис. 3.10.

Рисунок 3.10 - Графіки точок на 2-D площині, утворених за допомогою методу t-SNE для Credit Card Fraud Detection (зліва) та Bank Marketing (справа)

Можна помітити, що на рис. 3.10 точки обох наборів даних не дуже чітко, але поділені на декілька кластерів.

3.3 Аналіз отриманих результатів

У якості моделей та методів МН з учителем (класифікація) було взято k-NN Classifier, Gaussian NB, XGBoost Classifier, та, MLP Classifier(NN).

У якості моделей та методів навчання без учителя було взято:

1. Для задачі кластеризації - DBSCAN, BIRCH, Gaussian Mixture Model, OPTICS.
2. Для задачі виявлення аномалій - Isolation Forest, Local Outlier Factor.

Усі моделі та методи реалізовані у спеціалізованій бібліотеці scikit-learn.

Насамперед, моделям потрібно задати найкращі гіперпараметри. Перед тренуванням, набори були поділені на тестову та тренувальну вибірки.

Щоб було простіше, нехай набір Credit Card Fraud Detection позначається як CFD, а набір Bank Marketing - BM.

Для кожної моделі класифікації побудуємо сітку значень кожного з гіперпараметрів та скористаємося RandomizedSearchCV з бібліотеки scikit-learn.

Найкращі гіперпараметри для алгоритму k-NN Classifier після навчання на наборах CFD та BM вказані в табл. 3.1.

Таблиця 3.1 - Найкращі гіперпараметри для алгоритму KNN після навчання на наборах CFD та BM

Параметр	Опис параметру	CFD	BM
n_neighbors	Кількість сусідів	19	12
weights	Функція ваг для прогнозування	distance	distance
metric	Метрика для обрахунку відстані між точками	minkowski	minkowski

Найкращі гіперпараметри для алгоритму Gaussian Naive Bayes після навчання на наборах CFD та BM вказані у табл. 3.2.

Таблиця 3.2 - Найкращі гіперпараметри для алгоритму Gaussian Naive Bayes після навчання на наборах CFD та BM

Параметр	Опис параметру	CFD	BM
priors	Апріорні ймовірності класів	None	None
var_smoothing	Частка найбільшої дисперсії всіх ознак, яка додається до дисперсій для стабільності розрахунку.	0.93	0.78

Найкращі гіперпараметри для моделі XGBoost після навчання на наборах CFD та BM вказані у табл. 3.3.

Таблиця 3.3 - Найкращі гіперпараметри для моделі XGBoost після навчання на наборах CFD та BM

Параметр	Опис параметру	CFD	BM
subsample	Доля випадково вибраних прикладів з навчального набору даних, які будуть використані для побудови кожного дерева	0.8	0.7
learning_rate	Коефіцієнт швидкості навчання	1	0.00316
n_estimators	Кількість слабких моделей (дерев рішень), які будуть побудовані і комбіновані в процесі навчання	150	160
max_depth	Визначає максимальну глибину кожного дерева в моделі	6	7

Найкращі гіперпараметри для нейронної мережі MLPClassifier після навчання на наборах CFD та BM вказані у табл. 3.4.

Таблиця 3.4 - Найкращі гіперпараметри для нейронної мережі MLP Classifier після навчання на наборах CFD та BM

Параметр	Опис параметру	CFD	BM
hidden_layer_sizes	i-й елемент представляє кількість нейронів в i-му прихованому шарі	(50,)	(70,)
activation	Функція активації для прихованого шару	tanh	tanh
solver	Розв'язувач для оптимізації ваги	sgd	adam
alpha	Сила регуляризації L2	0.00001	0.00001
learning_rate	Коефіцієнт швидкості навчання	constant	adaptive

Для Gaussian Mixture Models та BIRCH потрібно підібрати параметр k-кількість кластерів за допомогою методу ліктя (Elbow Method). Суть його полягає у запуску алгоритму K-means для різних значень k, для кожного k розраховується сумарна відстань від кожної точки до центру її кластера - внутрішньо-кластерна сума квадратів відстаней, WCSS). На графіку по осі X відкладається k а по осі Y - WCSS, або, distortion. Точка ліктя, де спостерігається різке уповільнення зменшення WCSS, вказує на оптимальну кількість кластерів.

Графік методу ліктя для набору даних CFD зображено на рис. 3.11.

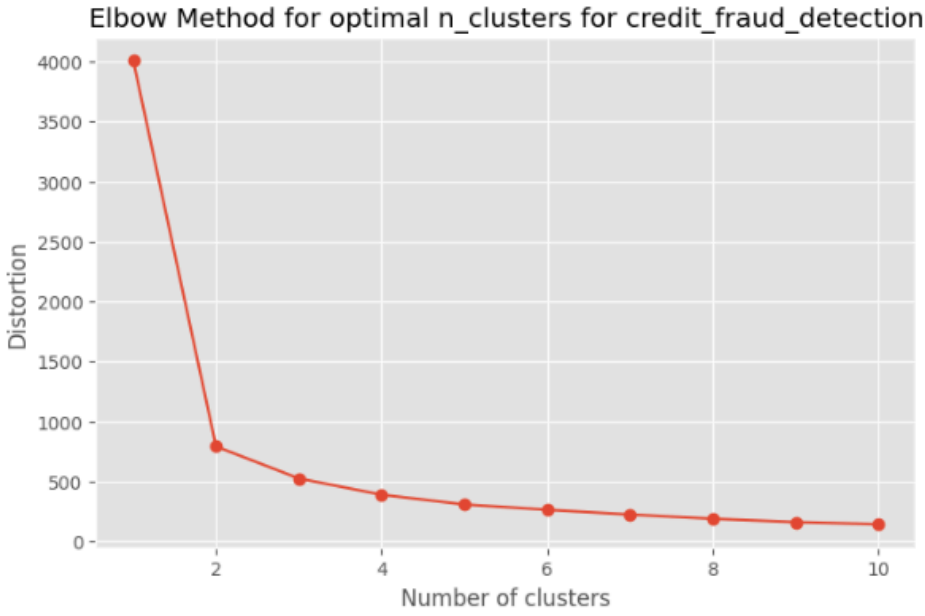


Рисунок 3.11 - Графік методу ліктя для набору даних CFD

Графік методу ліктя для набору даних BM зображено на рис. 3.12.

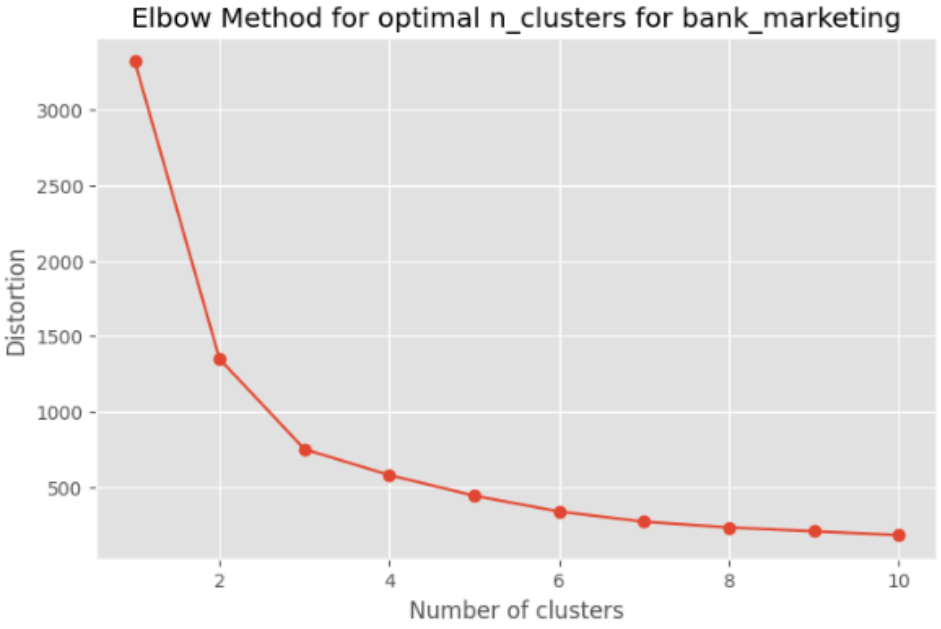


Рисунок 3.12 - Графік методу ліктя для набору даних BM

Отже, найкраще значення кількості кластерів для обох наборів даних $k = 2$.

Для DBSCAN та OPTICS потрібно підібрати гіперпараметр ε, за допомогою метода k-distance (в нашому випадку, $k = 2$). Дані - компоненти, котрі були отримані методом t-SNE. Для кожної точки обчислюється відстань

до її k-го найближчого сусіда, і ці відстані сортуються у порядку зростання. Потім будується графік k-distance, де по осі X відкладаються точки даних, а по осі Y - відповідні, k-відстані. Точка різкого зростання на графіку визначає оптимальне значення ε, що вказує на зміну щільності кластерів.

Графік методу k-distance для набору даних CFD зображено на рис. 3.13.

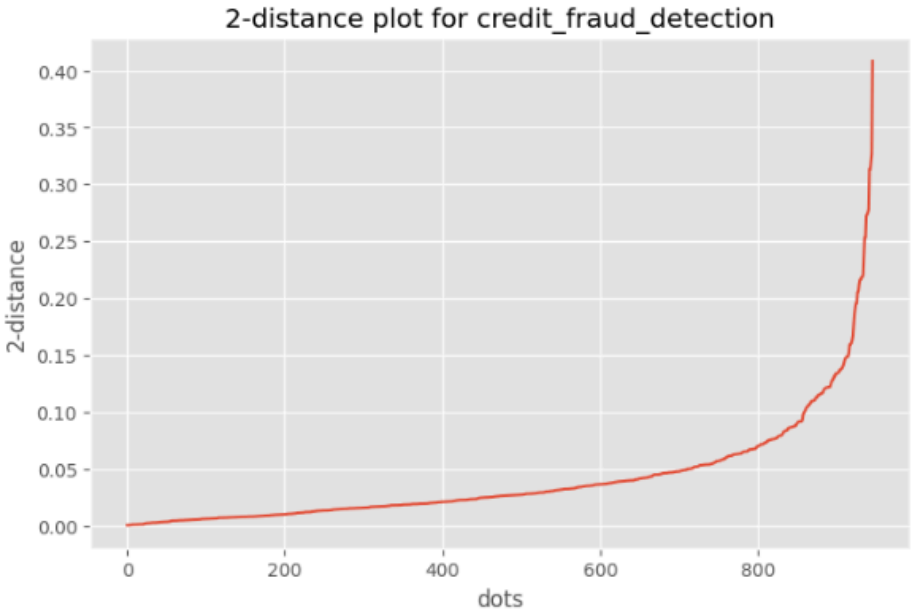


Рисунок 3.13 - Графік методу k-distance для набору даних CFD

Графік методу k-distance для набору даних BM зображено на рис. 3.14.

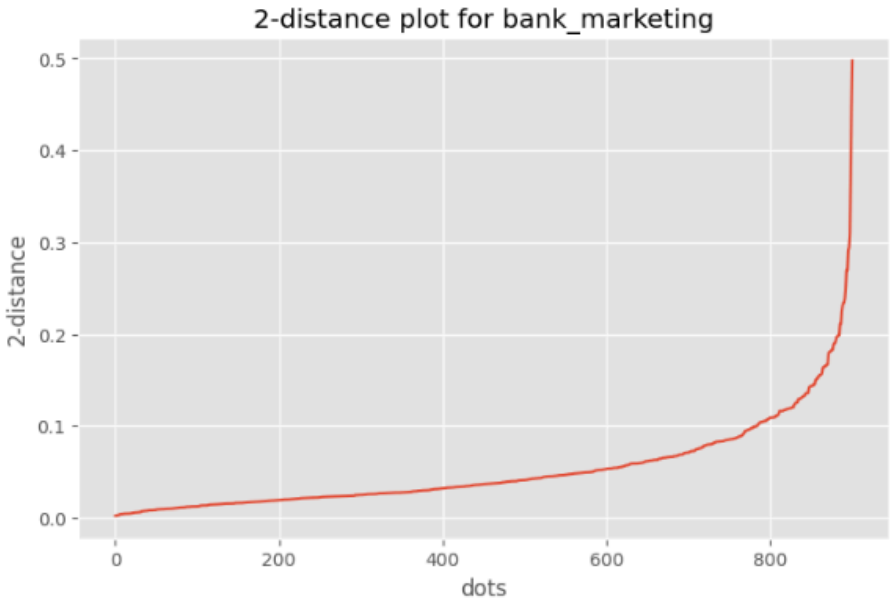


Рисунок 3.14 - Графік методу k-distance для набору даних BM

Отже, ті моделі, що вчаться на даних CFD, значення $\varepsilon \in [0.07, 0.23]$, а для тих, що вчаться на даних BM - $\varepsilon \in [0.07, 0.25]$. Далі, ще точніше налаштовуємо параметр ε за допомогою методу перебору.

Значення min_samples не менше за кількість ознак у датасеті плюс одиниця. Тобто, для DBSCAN та OPTICS обираємо min_samples = 30.

Найкращі гіперпараметри для алгоритму DBSCAN після навчання на наборах CFD та BM відображено у табл 3.5.

Таблиця 3.5 - Найкращі гіперпараметри для алгоритму DBSCAN після навчання на наборах CFD та BM

Параметр	Опис параметру	CFD	BM
min_samples	кількість точок в околицях для точки, яка розглядається як основна точка	30	30
eps	максимальна відстань між двома точками, щоб визначити, чи знаходяться вони разом	0.25	0.24

Найкращі гіперпараметри для алгоритму OPTICS після навчання на наборах CFD та BM відображено у табл 3.6.

Таблиця 3.6 - Найкращі гіперпараметри для алгоритму OPTICS після навчання на наборах CFD та BM

Параметр	Опис параметру	CFD	BM
min_samples	кількість точок в околицях для точки, яка розглядається як основна точка	30	30
eps	максимальна відстань між двома точками, щоб визначити, чи знаходяться вони разом	0.25	0.24
cluster_method	метод для виокремлення кластерів	dbscan	dbscan

Для BIRCH для обох наборів даних встановлюємо n_clusters = 2.

Для Gaussian Mixture Models для обох наборів даних встановлюємо n_components = 2.

Для Isolation Forest, перед тренуванням на обох наборах даних, обираємо максимальну кількість зразків, які використовуються для побудови кожного дерева max_samples = 450 (приблизна кількість прикладів одного

класу з кожного набору даних), частку аномальних (outliers) зразків у даних contamination = 0.4 (зумовлено нечітким поділом на кластери (рис. 3.10)).

Для Local outlier Factor кількість сусідів, які використовуються для оцінки локальної щільності n_neighbors = 450 (приблизна кількість прикладів одного класу з кожного набору даних), частку аномальних (outliers) зразків у даних contamination = 0.4 (зумовлено нечітким поділом на кластери (рис. 3.10)).

У задачі класифікації, у наборі CFD клас з аномаліями (шахрайствами) є клас, мітки якого рівні 1, у наборі BM клас з аномальною поведінкою клієнта (ті, що мають строковий депозит) - також клас, мітки якого рівні 1 (yes).

Метрики якості оцінки класифікації для моделей, натренованих на наборі даних CFD занесені у табл. 3.7.

Таблиця 3.7 - Метрики якості оцінки класифікації для моделей, натренованих на наборі даних CFD

Метрика \ Модель	k-NN	GNB	XGBoost	MLP
precision	**0.989011**	0.946809	0.960396	0.969388
recall	0.882353	0.872549	**0.950980**	0.931373
f1-score	0.932642	0.908163	**0.955665**	0.950000
auc	0.935495	0.907865	**0.952763**	0.948641
time	**0.004270**	0.003338	0.100535	1.24862

На наборі даних CFD, найкраще за метрикою precision (точність) показала модель k-NN. За метрикою recall (повнота), F1 та AUC - XGBoost. За часом тренування - k-NN.

Метрики якості оцінки класифікації для моделей, натренованих на наборі даних BM занесені у табл. 3.8.

Таблиця 3.8 - Метрики якості оцінки класифікації для моделей, натренованих на наборі даних BM

Метрика \ Модель	k-NN	GNB	XGBoost	MLP
precision	0.822222	0.777778	**0.869565**	0.858696
recall	0.831461	0.786517	**0.898876**	0.887640
f1-score	0.826816	0.782123	**0.883978**	0.872928
auc	0.828774	0.784563	**0.884221**	0.873168
time	0.003785	**0.002917**	0.136773	1.445863

На наборі даних BM, найкраще за метрикою precision (точність), recall (повнота), F1 та AUC - XGBoost. За часом тренування - GNB.

У задачі кластеризації, для DBSCAN, OPTICS аномалії - точки, що не ввійшли до жодного кластеру. Для BIRCH, GMM - у наборі CFD кластер з аномаліями (шахрайствами) - кластер, який містить точки з мітками, рівними 1, у наборі BM кластер з аномаліями (ті, що мають строковий депозит) - кластер, мітки точок рівними 1. Метрики якості оцінки кластеризації для моделей, натренованих на наборі даних CFD занесені у табл. 3.9.

Таблиця 3.9 - Метрики якості оцінки кластеризації для моделей, натренованих на наборі даних CFD

Метрика \ Модель	DBSCAN	BIRCH	OPTICS	GMM
Homogeneity	0.563117	**0.634229**	0.557172	0.600773
Completeness	0.278934	**0.638404**	0.277608	0.620114
V-measure	0.373071	**0.636310**	0.370577	0.610290
Silhouette	0.317278	0.632344	0.311412	**0.702990**
Calinski Harabasz	510.301	2383.55	487.970	**3830.11**
Davies Bouldin	1.241661	0.513626	1.230544	**0.430935**
Час навчання	**0.014093**	0.066946	1.203735	0.102389
Кількість кластерів	4	2	4	2
Кількість аномальних точок	284	0	291	0

В даному випадку (табл. 3.9) найкраще себе проявили ті моделі, що потребують вказати кількість кластерів. За ознакою Homogeneity, Completeness, V-measure найкраща модель - BIRCH, а за Silhouette, Calinski Harabasz, та Davies Bouldin Index - GMM. Проте, показники метрик обох моделей майже однакові.

Метрики якості оцінки кластеризації для моделей, натренованих на наборі даних BM занесені у табл. 3.10.

Таблиця 3.10 - Метрики якості оцінки кластеризації для моделей, натренованих на наборі даних BM

Метрика \ Модель	DBSCAN	BIRCH	OPTICS	GMM
Homogeneity	0.201625	0.161761	**0.201855**	0.148533
Completeness	0.105240	**0.162212**	0.105125	0.148586
V-measure	0.138295	**0.161986**	0.138250	0.148560
Silhouette	0.429848	0.523932	0.428890	**0.531673**
Calinski Harabasz	640.133	1296.135	642.671	**1319.153**
Davies Bouldin	1.437576	0.748525	1.361602	**0.734398**
Час навчання	**0.015035**	0.031627	1.150348	0.134964
Кількість кластерів	6	2	6	2
Кількість аномальних точок	15	0	16	0

В даному випадку (табл. 3.10) також найкраще себе проявили ті моделі, що потребують вказати кількість кластерів. За ознакою Homogeneity, Completeness, V-measure найкраща модель - BIRCH, а за Silhouette, Calinski Harabasz, та Davies Bouldin Index - GMM. Проте, показники метрик обох моделей майже однакові.

Для задачі виявлення аномалій, аномальними точками є мітки класів меншості обох наборів даних, як CFD, так і BM (клас, рівний 1). Моделі LOF та IF після прогнозування повертають набір міток зі значеннями -1 або 1, де -1 означає аномальну точку, а 1 - нормальну точку.

Метрики якості оцінки для моделей виявлення аномалій, що натреновані на наборі даних CFD занесені у табл. 3.11.

Таблиця 3.11 - Метрики якості оцінки для моделей виявлення аномалій, що натреновані на наборі даних CFD

Метрика \ модель	IF	LOF
Precision	0.677249	**0.968254**
Recall	0.541226	**0.773784**
F1-Score	0.601645	**0.860165**
AUC	0.641649	**0.874207**
Час тренування	0.540928	**0.110185**

Метрики якості оцінки для моделей виявлення аномалій, що натреновані на наборі даних BM занесені у табл. 3.12.

Таблиця 3.12 - Метрики якості оцінки для моделей виявлення аномалій, що натреновані на наборі даних BM

Метрика \ Модель	IF	LOF
Precision	0.603878	**0.714681**
Recall	0.483370	**0.572062**
F1-Score	0.536946	**0.635468**
AUC	0.583149	**0.671840**
Час тренування	0.528810	**0.154757**

Як видно з табл. 3.11 та табл. 3.12 найкраща модель для виявлення аномалій - це LOF, причому, за усіма метриками якості!

3.4 Висновки до розділу 3

У цьому розділі було детально розглянуто процес реалізації програмного продукту для виявлення аномалій та проведено аналіз отриманих результатів. Основні аспекти роботи включають використання популярних і ефективних інструментів для реалізації алгоритмів машинного навчання та аналізу даних, таких як Python, Scikit-learn, Seaborn, Numpy,

Pandas, XGBoost, та Google Collaboratory. Це забезпечило гнучкість, швидкість та надійність при розробці та тестуванні моделей.

Для ефективного навчання моделей було проведено ретельну попередню обробку даних на прикладі двох основних наборів: Credit Card Fraud та Bank Marketing.

Попередня обробка включає виявлення та видалення пропущених значень, стандартизацію, а також кодування категоріальних змінних, що є критично важливим для покращення якості моделей та їх здатності до виявлення аномалій.

Також наведені результати метрик оцінки якості моделей і методів з та без вчителя для пошуку аномалій. Найкраще себе для обох наборів у класифікації проявила модель XGBoost, кластеризації - BIRCH та GMM, у виявленні аномалій - LOF.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В даному розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на дослідженні демографічного стану.

Дана реалізація буде сприяти проведенню усіх необхідних досліджень, що дасть змогу якісно дослідити питання не лише в Україні, проте у всьому світі.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір самої програми.

F_2 – якісний аналіз даних.

F_3 – графічні показники.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) Python.

б) C++.

Функція F_2.

а) Застосування вбудованих функцій.

б) Створення своїх обчислень значень.

Функція F_3:

а) Використання шаблонних графіків.

б) Створення своїх.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

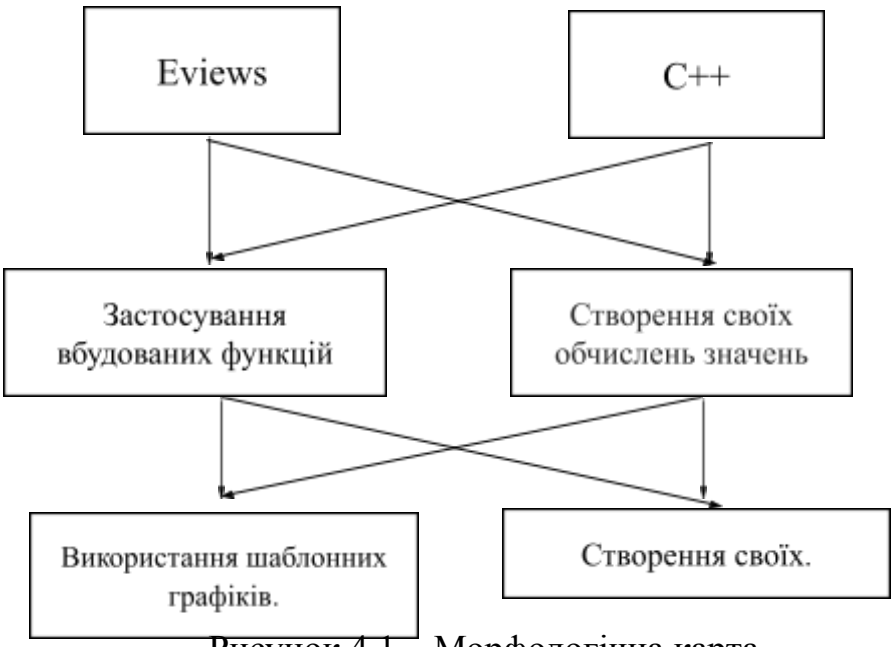


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1 - Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	*А*	Величезна кількість бібліотек , легко інтегрується з іншими мовами програмування	Використовує велику кількість пам’яті, що може бути критично для обмежених систем
	Б	Ефективне використання пам’яті, я значно швидшим завдяки низькорівневому управлінню пам’яттю	Більш складний синтаксис та управління пам'яттю, потребує багато часу для написання та відлагодження коду
F_2	*А*	Дозволяє швидко виконувати аналіз, надійні	Обмежена можливість налаштування під специфічні потреби
	Б	Повна свобода у створенні функцій для специфічних задач	Вимагає більше часу та ресурсів для розробки і тестування
F_3	*А*	Швидке створення візуалізацій без необхідності розробки з нуля.	Можуть виглядати стандартно і не відображати унікальні аспекти даних.
	Б	Повна свобода у створенні графіків для специфічних задач	Потребує більше знань та навичок у візуалізації даних

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1:

Перевагу даємо загальнодоступності. Для спрощення роботи по написанню коду варіант Б має бути відкинутий.

Функція F_2:

Програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Функція F_3:

Реалізація першого варіанту є сприйнятливою для програми. Це варіант А.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

F_1а – F_2а – F_3а

F_1а – F_2б – F_3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- *X1* – швидкодія мови програмування;
- *X2* – об’єм пам’яті для обчислень та збереження даних;
- *X3* – час навчання даних;
- *X4* – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2 - Основні параметри програмного продукту

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	60	80	110
Об’єм пам’яті	X2	Мб	60	50	30
Час попередньої обробки даних	X3	мс	80	70	60
Потенційний об’єм програмного коду	X4	кількість рядків коду	500	450	416

За даними таблиці 4.3 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

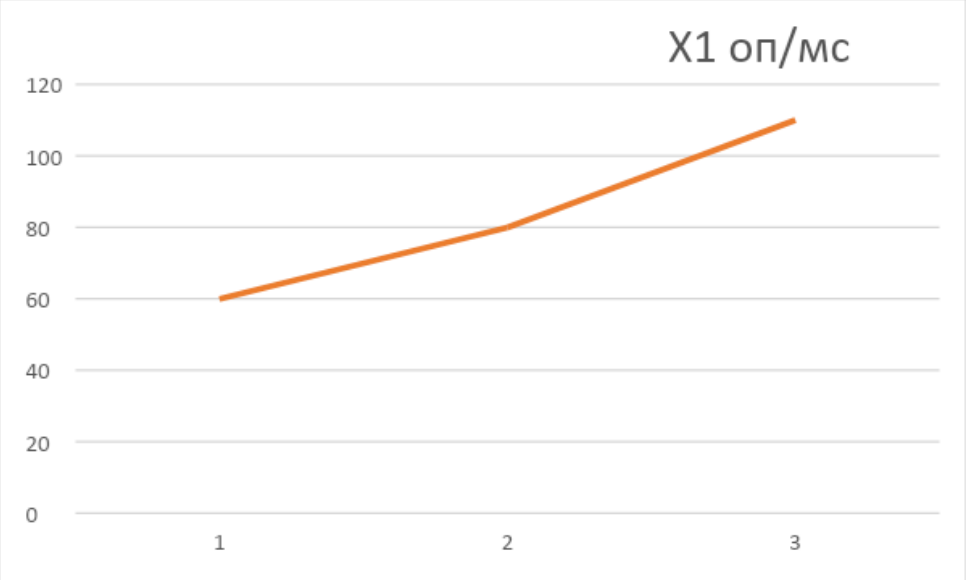


Рисунок 4.2 – X1, швидкодія мови програмування

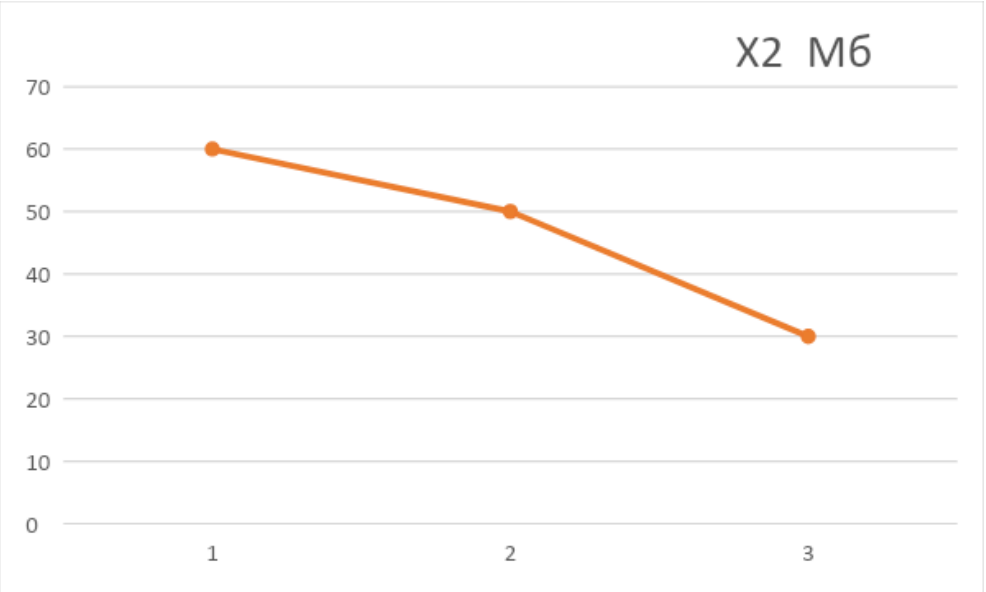


Рисунок 4.3 – X2, об’єм пам’яті

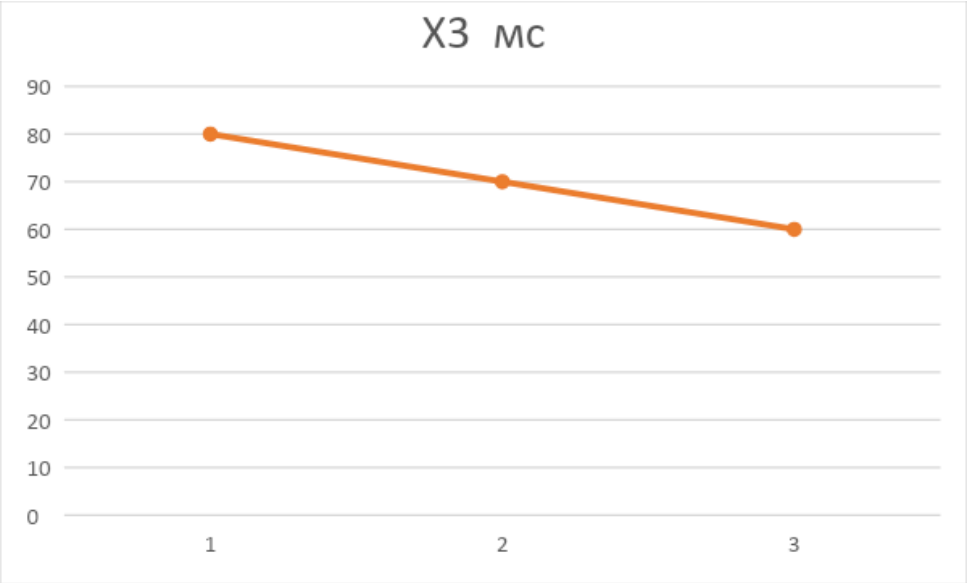


Рисунок 4.4 – X3, час попередньої обробки даних

Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	1	2	2	1	1	1	2	10	-7.5	56.25
X2	Об’єм пам’яті	Mb	3	4	3	3	4	3	4	24	6.5	42.25
X3	Час попередньої обробки даних	мс	2	1	1	2	2	2	1	11	-6.5	42.25
X4	Потенційний об’єм програмного коду	Кількість рядків коду	4	3	4	4	3	4	3	25	7.5	56.25
	Разом		10	10	10	10	10	10	10	70	0	197

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \ (4.1)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17.5 \ (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \ (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 197. \ (4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2\left(n^3 - n\right)} = \frac{12 \cdot 197}{7^2\left(4^3 - 4\right)} = 0.754 > W_k = 0.67. \ (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 - Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	<	<	<	<	<	0.5
X1 і X3	<	>	>	<	<	<	>	<	0.5
X1 і X4	<	<	<	<	<	<	<	<	0.5
X2 і X3	>	>	>	>	>	>	>	>	1.5
X2 і X4	<	>	<	<	>	<	<	<	0.5
X3 і X4	<	<	<	<	<	<	<	<	0.5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \{1.5 \text{ при } X_i > X_j \, 1, \ 0 \text{ при } X_i = X_j, \ 0.5 \text{ при } X_i < X_j \, . \ (4.6)$$

З отриманих числових оцінок переваги складемо матрицю А=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \quad (4.9)$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \quad (4.10)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1	0.5	0.5	0.5	2.5	0.16	9.25	0.16	34.125	0.16
X2	1.5	1	1.5	0.5	4.5	0.28	16.25	0.28	59.125	0.28
X3	1.5	0.5	1	0.5	3.5	0.22	12.25	0.21	41.875	0.2
X4	1.5	1.5	1.5	1	5.5	0.34	21.25	0.35	77.875	0.36
Всього:					16	1	59	1	213	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Об'єм пам'яті), *X3* (час попередньої обробки даних) та *X4* (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j}, \quad (4.11)$$

де n – кількість параметрів;

$K_{\text{в}i}$ – коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	100	26	0.16	4.16
F3	А	X2	91	23	0.28	6.44
	Б	X3	28	19	0.2	3.8
F4	А	X4	27	30	0.36	10.8

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}\left[F_{1k}\right] + K_{\text{ТУ}}\left[F_{2k}\right] + \ldots + K_{\text{ТУ}}\left[F_{zk}\right], \quad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 4.16 + 6.44 + 10.8 = 21.4 ;$$

$$K_{K2} = 4.16 + 3.8 + 10.8 = 18.76 .$$

Як видно з розрахунків, кращим є 2 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_О = T_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М}, \quad (4.13)$$

де $T_Р$ – трудомісткість розробки ПП;

$К_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_Р$ =37 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.8. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.9. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 37 \cdot 1.8 \cdot 0.9 = 59.94 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =29 людино-днів, $К_П$ =0.9, $К_{СК}$ = 1, $К_{СТ}$ =0.8:

$$Т_2 = 29 \cdot 0.9 \cdot 0.8 = 20.88 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$Т_I = (59.94 + 20.88 + 4.8 + 20.88) \cdot 8 = 852 \text{ людино-годин.}$$

$$Т_{II} = (59.94 + 20.88 + 6.91 + 20.88) \cdot 8 = 868.88 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 25000 грн., один аналітик в області даних з окладом 3000. Визначимо середню зарплату за годину за формулою:

$$СЧ = \frac{М}{T_m \cdot t} \text{грн.,} \quad (4.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$СЧ = \frac{2*25000+30000}{3 \cdot 21 \cdot 8} = 158.73 \text{ грн.} \quad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$СЗП = С_ч \cdot Т_i \cdot КД, (4.16)$$

де $С_ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 158.73 \cdot 852 \cdot 1.2 = 162285.55$ грн.

II. $С_{ЗП} = 158.73 \cdot 868.88 \cdot 1.2 = 165500.79$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 162285.55 \cdot 0.22 = 35702.821$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 165500.79 \cdot 0.22 = 36410.17$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 25000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 25000 \cdot 0,2 = 60000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 60000 \cdot (1 + 0.2) = 72000 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 72000 \cdot 0,22 = 15840 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 20000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.4 \cdot 0.25 \cdot 20000 = 7000 \text{ грн.},$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.4 \cdot 20000 \cdot 0.08 = 2240 \text{ грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.35 =$$

$$= 627.2 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 627,2 \cdot 0,24 \cdot 0,32 \cdot 5,31 = 255.78 \text{ грн.},$$

де N_C – середньо-споживча потужність приладу;

$K_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 20000 \cdot 0.67 = 13400 \quad грн.$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, (4.17)$$

$$С_{ЕКС} = 72000 + 15840 + 7000 + 2240 + 255.78 + 13400 = 110735.78 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 110735.78 / 627{,}2 = 176.56 \quad грн/год.$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М-Г} \cdot T, (4.18)$$

I. $С_М = 176.56 \cdot 852 = 150429.12$ грн.

II. $С_М = 176.56 \cdot 868.88 = 153409.45$ грн.

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0.67, (4.19)$$

I. $С_Н = 162285.55 \cdot 0.67 = 108731.32$ грн.

II. $С_Н = 165500.79 \cdot 0.67 = 110885.53$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н, (4.20)$$

I. $С_{ПП} = 162285.55 + 35702.821 + 150429.12 + 108731.32 = 457148.81$ грн.

II. $С_{ПП} = 165500.79 + 36410.17 + 153409.45 + 110885.53 = 466205.94$ грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР_j} = К_{К_j} / С_{Ф_j}, (4.21)$$

$К_{ТЕР1} = 21,4 / 457148.81 = 4.6812 \cdot 10^{-5}$,

$К_{ТЕР2} = 18.76 / 466205.94 = 4.0234 \cdot 10^{-5}$.

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 4.6812 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 4.6812 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- вибір програмного продукту – Python;
- реалізація важливої постановки з допомогою вбудованих функцій;
- використання стандартного інтерфейсу для побудови значень.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку реалізацію програми та доступний функціонал для роботи.

4.8 Висновки до розділу 4

В даному розділі було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту.

В результаті виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують.

На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

В ході роботи було виконано комплексний огляд предметної області, пов'язаної з виявленням аномалій у поведінці клієнтів банку. Було детально розглянуто та проаналізовано основні методи машинного навчання, як з учителем, так і без нього. Зокрема, досліджено ефективність алгоритмів k-nearest neighbors, Gaussian Naive Bayes, XGBoost для навчання з учителем та DBSCAN, OPTICS, Gaussian Mixture Models, Local Outlier Factor, BIRCH, Isolation Forest для навчання без учителя.

Для оцінки якості класифікації застосовувалися такі метрики, а саме: Precision, Recall, F1-Score, AUC. Для кластеризації використовувалися такі метрики, а саме: Homogeneity, Completeness, V-measure, Silhouette, Calinski Harabasz, Davies Bouldin, що дозволило об'єктивно порівняти результати різних методів.

Розроблено програмний продукт на мові програмування Python, проведено його тестування на реальних наборах даних (Credit Card Fraud Detection, Bank Marketing), а також здійснено попередню обробку та покращення наборів даних для підвищення точності моделей.

Проаналізовано результати метрик якості для кожної моделі.

Проведено функціонально-вартісний аналіз програмного продукту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ПриватБанк. Політика приватності. URL: https://privatbank.ua/personal-information. (дата звернення 21.04.2024)
2. McKinsey & Company. How Generative AI Can Help Banks Manage Risk and Compliance. McKinsey & Company URL: https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/how-generative-ai-can-help-banks-manage-risk-and-compliance (дата звернення 21.04.2024).
3. Forbes Technology Council. How Artificial Intelligence Is Reshaping Banking. Forbes. URL: https://www.forbes.com/sites/forbestechcouncil/2024/02/23/how-artificial-intelligence-is-reshaping-banking(дата звернення 21.04.2024).
4. Bouchama, Rabah. Formal Verification and Control of Autonomous Systems. Theses, HAL. URL: https://theses.hal.science/tel-03651493v1/document(дата звернення 21.04.2024).
5. IBM. Anomaly Detection. IBM. URL: https://www.ibm.com/topics/anomaly-detection (дата звернення 21.04.2024).
6. IBM. Artificial Intelligence. IBM. URL: https://www.ibm.com/topics/artificial-intelligence (дата звернення 21.04.2024).
7. Medium Data Science 365. The Relationship Between AI, ML, NNs, and DL. Medium. URL: https://medium.com/data-science-365/the-relationship-between-ai-ml-nns-and-dl-60bd40069908 (дата звернення 21.04.2024).
8. UC Berkeley School of Information. What Is Machine Learning? UC Berkeley School of Information. URL:

https://ischoolonline.berkeley.edu/blog/what-is-machine-learning (дата звернення 21.04.2024).

9. IBM. k-Nearest Neighbors. IBM. URL:https://www.ibm.com/topics/knn (дата звернення 21.04.2024).
10. Martins, C. Gaussian Naive Bayes Explained With Scikit-Learn. Built In. URL: https://builtin.com/artificial-intelligence/gaussian-naive-bayes (дата звернення 21.04.2024).
11. Chen, Tianqi. XGBoost: Its Genealogy, Its Architectural Features, and Its Innovation. Towards Data Science. URL: https://towardsdatascience.com/xgboost-its-genealogy-its-architectural-features-and-its-innovation-bf32b15b45d2 (дата звернення 21.04.2024).
12. Crop Yield. Estimation Using Sentinel-3 SLSTR, Soil Data, and Topographic Estimation Using Sentinel-3 SLSTR, Soil Data, and Topographic Features Combined with Machine Learning Modeling: A Case Study of Nepal. 2023. AgriEngineering Vol.5. No. 4. P. 1766-1788 . DOI: 10.3390/agriengineering5040109. URL: https://www.researchgate.net/publication/374564114_Crop_Yield_Estimation_Using_Sentinel-3_SLSTR_Soil_Data_and_Topographic_Features_Combined_with_Machine_Learning_Modeling_A_Case_Study_of_Nepal (дата звернення 21.04.2024).
13. Srinivasan Selvaraj. Industry 4.0 Interoperability, Analytics, Security, and Case Studies. 2021. In book: Industry 4.0 Interoperability, Analytics, Security and Case StudiesPublisher: CRC press, Taylor FrancisGroup. URL: https://www.researchgate.net/publication/355485828_Industry_40_Interoperability_Analytics_Security_and_Case_Studies (дата звернення 21.04.2024).
14. Merk Adam, Cal Piotr, Wozniak Michal. Distributed DBSCAN Algorithm – Concept and Experimental Evaluation. 2017, P. 472-480. DOI: 10.1007/978-3-319-59162-9_49 (дата звернення 21.04.2024).

15. Wong William. Gaussian Mixture Model. Built In. URL: https://builtin.com/articles/gaussian-mixture-model (дата звернення 21.04.2024).
16. Breunig Markus M., et al. LOF: Identifying Density-Based Local Outliers. LMU. URL: https://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf (дата звернення 21.04.2024).
17. Ankerst Mihael, Breunig Markus, Kröger Peer, Sander Joerg. OPTICS: Ordering Points to Identify the Clustering Structure. Sigmod Record. 1999. 28. P. 49-60. 10.1145/304182.304187 (дата звернення 21.04.2024).
18. Zhang Tian, et al. BIRCH: An Efficient Data Clustering Method for Very Large Databases. CS.SFU. URL: https://www2.cs.sfu.ca/CourseCentral/459/han/papers/zhang96.pdf (дата звернення 21.04.2024).
19. Blondel Vincent, et al. Modularity in Community Detection. CORE. URL: https://core.ac.uk/download/pdf/80994866.pdf (дата звернення 21.04.2024).
20. Liu Fei Tony, Ting Kai, Zhou Zhi-Hua. Isolation Forest. 2009. P. 413 - 422. DOI: 10.1109/ICDM.2008.17 (дата звернення 21.04.2024).
21. Scikit-Learn. Model Evaluation. Scikit-Learn. URL: https://scikit-learn.org/stable/modules/model_evaluation.html (дата звернення 21.04.2024).
22. Недашківська Н.І. Методи і моделі інтелектуального аналізу даних. Практикум. Навчальний посібник. Київ : КПІ ім. Ігоря Сікорського, 2019. 71 с. URL: https://ela.kpi.ua/handle/123456789/53764 (дата звернення 21.04.2024).
23. Nayak Ankur A. Hands-On Unsupervised Learning Using Python. Amazon. URL:https://www.amazon.com/Hands-Unsupervised-Learning-Using-Python/dp/1492035645 (дата звернення 21.04.2024).
24. Pandas - Python Data Analysis Library. URL: https://pandas.pydata.org (дата звернення 21.04.2024).

25. NumPy. NumPy. URL: https://numpy.org (дата звернення 21.04.2024).
26. Matplotlib - Visualization with Python. Matplotlib - Visualization with Python. URL: https://matplotlib.org (дата звернення 21.04.2024).
27. Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics Vol. 29, No. 5 (Oct., 2001), pp. 1189-1232. DOI: 10.1214/aos/1013203451 (дата звернення 21.04.2024)..
28. Недашківська Н.І. Інтелектуальний аналіз даних : Практикум. Навч. посіб. для студ. спеціальності 124 «Системний аналіз», освітніх програм «Системний аналіз і управління», «Системний аналіз фінансового ринку». Київ : КПІ ім. Ігоря Сікорського. 2021. 105 с. URL: https://ela.kpi.ua/handle/123456789/53763 (дата звернення 21.04.2024).
29. Waskom M. L., seaborn: statistical data visualization. Journal of Open Source Software. 6(60). 2021. 3021. DOI: 10.21105/joss.03021 (дата звернення 21.04.2024).

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

```
# -*- coding: utf-8 -*-
"""code

Automatically generated by Colab.

Original file is located at

https://colab.research.google.com/drive/1XGWbFtlG2_npBG67MTm4sIaj2O7C
MmHc
"""

import pandas as pd, numpy as np, seaborn as sns, xgboost as xgb,
matplotlib.pyplot as plt,time

from sklearn.ensemble import IsolationForest
from sklearn.neighbors import KNeighborsClassifier,
LocalOutlierFactor, NearestNeighbors
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.cluster import DBSCAN, Birch, OPTICS, KMeans
from sklearn.mixture import GaussianMixture

from sklearn.preprocessing import StandardScaler, OrdinalEncoder
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.model_selection import train_test_split,
RandomizedSearchCV
from sklearn.metrics import precision_recall_fscore_support, \
precision_score, recall_score, f1_score, classification_report,
roc_curve, auc,\
homogeneity_score, completeness_score, v_measure_score,\
silhouette_score, calinski_harabasz_score,
davies_bouldin_score,adjusted_rand_score

from imblearn.under_sampling import RandomUnderSampler
plt.style.use('ggplot')

"""# EDA Credit Card Fraud"""

!kaggle datasets download -d mlg-ulb/creditcardfraud
!unzip creditcardfraud.zip

df_cfd = pd.read_csv('creditcard.csv')

# df_cfd.info()
```

```
print(f"Кількість видалених дублікатів у датасеті:
{df_cfd.duplicated().sum()}")
df_cfd.drop_duplicates(inplace=True)

df_cfd.describe().round(2).T

ss_cfd = StandardScaler()
df_cfd.iloc[:,:-1] = ss_cfd.fit_transform(df_cfd.iloc[:,:-1])

plt.figure(figsize=(12, 8))
sns.heatmap(df_cfd.corr(),cmap='PuOr',vmin=-1.0,
vmax=1.0,square=True)

def plt_dist_or_box(flag='box'):
  fig, axes = plt.subplots(nrows=3, ncols=4, figsize=(11,17))

  features = [f'V{i}' for i in [1,2,3,4,7,10,11,12,14,16,17,18]]

  for i, ax in enumerate(axes.flat):
    if flag == 'distribution':
      ax.hist(df_cfd[features[i]], bins=60, linewidth=0.5,
edgecolor="white")
    else:
      sns.boxplot(ax=ax, data=df_cfd, x='Class', y=features[i])
    ax.set_title(f"Розподіл {features[i]}")

  plt.tight_layout()

"""# EDA [Bank
Marketing](https://archive.ics.uci.edu/dataset/222/bank+marketing)"""

df_bm = pd.read_csv('bank-additional.csv',sep=';')

df_bm.info()

df_bm.rename(columns={'y':'Class'},inplace=True)

plt.figure(figsize=(12, 4))
for i,col in enumerate(['contact','education','job','poutcome']):
  ax = plt.subplot(2, 2, i + 1)
  sns.histplot(df_bm[col])
  ax.tick_params(axis='x', rotation=90, labelsize=8)
  ax.tick_params(axis='y', labelsize=8)
  plt.subplots_adjust(hspace=2, wspace=0.5)

pd.DataFrame({
    'num_nan': df_bm.isna().sum(),
    'num_uniq_val': df_bm.agg('nunique')})

df_bm.drop_duplicates(inplace=True)
```

```
oe_bm = OrdinalEncoder()
obj_cols = df_bm.select_dtypes(include=['object']).columns
df_bm[obj_cols] = oe_bm.fit_transform(df_bm[obj_cols])

scaler_bm = StandardScaler()
df_bm.iloc[:,:-1] = scaler_bm.fit_transform(df_bm.iloc[:,:-1])

plt.figure(figsize=(12, 8))
sns.heatmap(df_bm.corr(),cmap='PuOr', vmin=-1.0,
vmax=1.0,square=True)

"""# Undersampling imbalanced datasets & TSNE"""

us_data_repo = {}

for df,name in zip([df_cfd, df_bm],['credit_fraud_detection',
'bank_marketing']):
  X = df.drop('Class',axis=1)
  y = df['Class']
  undersample =
RandomUnderSampler(sampling_strategy='all',random_state=0)
  X, y = undersample.fit_resample(X, y)
  tsne = TSNE(n_components=2,perplexity=500, random_state=42)
  X[['c1','c2']] = tsne.fit_transform(X)

  us_data_repo[name] = [X, y]

plt.figure(figsize=(15,8))
for i,(name_data, data) in enumerate(us_data_repo.items()):
    X, y = data
    X = X.iloc[:,-2:]

    plt.subplot(1,2,i+1)
    plt.scatter(X['c1'][y==0],X['c2'][y==0],label='Class 0')
    plt.scatter(X['c1'][y==1],X['c2'][y==1],label='Class 1')
    plt.xlabel('Component 1')
    plt.ylabel('Component 2')
    plt.title(f'TSNE for {name_data}')
    plt.legend()

for name_data, data in us_data_repo.items():
    X, y = data
    print(name_data, y.value_counts(),sep='\n')

plt.figure(figsize=(15,8))
for i,(name_data, data) in enumerate(us_data_repo.items()):
    X, y = data
    X = X.iloc[:,:-2]
    df = X
```

```
    df['Class'] = y
    plt.subplot(1,2,i+1)
    sns.heatmap(df.corr(),cmap='PuOr', vmin=-1.0,
vmax=1.0,square=True)

"""#Supervised learning for anomaly detection"""

classification_models = {
    'credit_fraud_detection':{
        'knn': KNeighborsClassifier(),
        'gnb': GaussianNB(),
        'xgboost':xgb.XGBClassifier(),
        'nn':MLPClassifier()
    },
    'bank_marketing':{
        'knn': KNeighborsClassifier(),
        'gnb': GaussianNB(),
        'xgboost':xgb.XGBClassifier(),
        'nn':MLPClassifier()
    }
}

param_grids = {
    'knn': {
        'n_neighbors': np.arange(1, 31),
        'weights': ['uniform', 'distance'],
        'metric': ['euclidean', 'manhattan', 'minkowski']
    },
    'gnb': {
        'var_smoothing': np.logspace(-9, 0, 100)
    },
    'xgboost': {
        'n_estimators': np.arange(50, 300, 50),
        'max_depth': np.arange(3, 10),
        'learning_rate': np.logspace(-5, 0, 3),
        'subsample': np.arange(0.5, 1.0, 0.1)
    },
    'nn': {
        'hidden_layer_sizes': [(50, ), (70, ),(50,10),(70,10)],
        'activation': ['tanh', 'relu'],
        'solver': ['sgd', 'adam'],
        'alpha': np.logspace(-5, -1, 3),
        'learning_rate': ['constant', 'adaptive']
    }
}

classification_metrics = {}

for name_data, models in classification_models.items():
    X, y = us_data_repo[name_data]
```

```
    X = X.iloc[:,:-2]

    X_train, X_test, y_train, y_test = \
    train_test_split(X, y , test_size=0.2, random_state=42)

    classification_metrics[name_data] = {}
    for model_name, model in models.items():
        random_search = RandomizedSearchCV(
            model,
            param_distributions=param_grids[model_name],
            n_iter=50, cv=5, n_jobs=-1, random_state=42)
        random_search.fit(X,y)

        model.set_params(**random_search.best_params_)
        start_time = time.time()
        model.fit(X_train, y_train)
        end_time = time.time()

        pred = model.predict(X_test)

        fpr, tpr, _ = roc_curve(y_test, pred)

        classification_metrics[name_data][model_name] = {
            'precision': precision_score(y_test, pred),
            'recall': recall_score(y_test, pred),
            'f1-score': f1_score(y_test, pred),
            'auc': auc(fpr, tpr),
            'time': end_time - start_time
        }

        print(f"Task: {name_data}, Model: {model_name}")
        print(f"Best parameters: {random_search.best_params_}")
        print(f"Best score: {random_search.best_score_}\n")

pd.DataFrame(classification_metrics['credit_fraud_detection'])

pd.DataFrame(classification_metrics['bank_marketing'])

"""# Unsupervised learning for anomaly detection"""

for name_data, data in us_data_repo.items():
    X, y = data
    X = X.iloc[:,-2:]
    distortions = []
    for k in range(1, 11):
        kmeans = KMeans(n_clusters=k,n_init=10)
        kmeans.fit(X)
        distortions.append(kmeans.inertia_)
    plt.figure(figsize=(8, 5))
    plt.plot(range(1, 11), distortions, marker='o')
```

```
    plt.xlabel('Number of clusters')
    plt.ylabel('Distortion')
    plt.title(f'Elbow Method for optimal n_clusters for {name_data}')

"""Для обох наборів даних оптимальна кількість кластерів k=2."""

for name_data, data in us_data_repo.items():
  k = 2
  X, y = data
  X = X.iloc[:,-2:]
  nbrs = NearestNeighbors(n_neighbors=k).fit(X)
  distances, _ = nbrs.kneighbors(X)


  k_distances = distances[:, -1]
  k_distances_sorted = np.sort(k_distances)

  plt.plot(np.arange(len(X)), k_distances_sorted)
  plt.xlabel('dots')
  plt.ylabel(f'{k}-distance')
  plt.title(f'{k}-distance plot for {name_data}')
  plt.show()

"""Для credit_fraud_detection eps = [0.07, 0.23]

Для bank_marketing eps = [0.07, 0.25]

Підбираємо найкращі eps за допомогою bruteforce:
"""

brut_eps = {}
epsilon = {
    'credit_fraud_detection': np.arange(0.07,0.23,0.01),
    'bank_marketing': np.arange(0.07,0.25,0.01)
}
for name_data, data in us_data_repo.items():
    X, y = data
    X = X.iloc[:,-2:]
    brut_eps[name_data] = {}
    metrics = {}
    for name_m, model in zip(['dbscan','optics'],[DBSCAN, OPTICS]):
      model_v = []
      model_eps = []
      for eps in epsilon[name_data]:

        params = {
          'eps':eps,
          'min_samples':3
        }
```

```
            if name_m == 'optics':
              params['min_cluster_size'] = 400
            m = model(**params)

            pred = m.fit_predict(X)
            model_v.append(v_measure_score(y, pred))
            model_eps.append(eps)

          metrics[name_m+'_v_measure'] = model_v

        brut_eps[name_data] = {
            'eps': epsilon[name_data],
            'dbscan_v_measure': metrics['dbscan_v_measure'],
            'optics_v_measure': metrics['optics_v_measure']
        }

    pd.DataFrame(brut_eps['credit_fraud_detection'])

    pd.DataFrame(brut_eps['bank_marketing'])

    clustering_models ={
        'credit_fraud_detection':{
            'dbscan': DBSCAN(
                eps=0.22,
                min_samples=3
            ),
            'birch': Birch(n_clusters=2),
            'optics': OPTICS(
                eps=0.2,
                min_samples=3,
                min_cluster_size=400
            ),
            'gmm': GaussianMixture(n_components=2)
        },
        'bank_marketing':{
            'dbscan': DBSCAN(
                eps=0.07,
                min_samples=3
            ),
            'birch': Birch(n_clusters=2),
            'optics': OPTICS(
                eps=0.16,
                min_samples=3,
                min_cluster_size=400
            ),
            'gmm': GaussianMixture(n_components=2)
        },
    }

    clustering_metrics = {}
```

```
for name_data, models in clustering_models.items():
    X, y = us_data_repo[name_data]

    X = X.iloc[:,-2:]
    clustering_metrics[name_data] = {}

    for model_name, model in models.items():

        start_time = time.time()
        pred = model.fit_predict(X)
        end_time = time.time()

        clustering_metrics[name_data][model_name] = {
            'Homogeneity': homogeneity_score(y, pred),
            'Completeness': completeness_score(y, pred),
            'V-measure': v_measure_score(y, pred),
            'Silhouette': silhouette_score(X, pred),
            'Calinski Harabasz': calinski_harabasz_score(X, pred),
            'Davies Bouldin': davies_bouldin_score(X, pred),
            'time': end_time - start_time
        }

        print(f"Task: {name_data}, Model: {model_name}")

pd.DataFrame(clustering_metrics['credit_fraud_detection'])

pd.DataFrame(clustering_metrics['bank_marketing'])

"""max_samples та n_neighbors приблизно дорівнює кількості прикладів
в одному з кластерів."""

anomaly_detect_models ={
    'credit_fraud_detection':{
        'if': IsolationForest(
            max_samples=470
            ),
        'lof': LocalOutlierFactor(
            n_neighbors=470
            ),
    },
    'bank_marketing':{
        'if': IsolationForest(
            max_samples=450
            ),
        'lof': LocalOutlierFactor(
            n_neighbors=450,
            contamination=0.5
            ),
    },
```

```
}

anomaly_metrics = {}

for name_data, models in anomaly_detect_models.items():
    X, y = us_data_repo[name_data]

    X = X.iloc[:,:-2]

    anomaly_metrics[name_data] = {}
    for model_name, model in models.items():

        start_time = time.time()
        pred = model.fit_predict(X)
        end_time = time.time()


        if model_name != 'gmm':
          pred[pred == 1] = 0
          pred[pred == -1] = 1

        fpr, tpr, _ = roc_curve(y, pred)

        anomaly_metrics[name_data][model_name] = {
            'precision': precision_score(y, pred),
            'recall': recall_score(y, pred),
            'f1-score': f1_score(y, pred),
            'auc': auc(fpr, tpr),
            'time': end_time - start_time
        }

        print(f"Task: {name_data}, Model: {model_name}")

pd.DataFrame(anomaly_metrics['credit_fraud_detection'])

pd.DataFrame(anomaly_metrics['bank_marketing'])
```

ДОДАТОК Б. ПЕРЕЛІК ГРАФІЧНОГО МАТЕРІАЛУ ДО ЗАХИСТУ

Моделі і методи машинного навчання для виявлення аномалій у поведінці клієнтів банку

Виконав: студент IV курсу, групи КА-05

Артеменко Євгеній Вячеславович

Керівник: д.т.н., професор

Недашківська Надія Іванівна

Актуальність

Актуальність даної роботи полягає у тому, що сучасні банки все більше залучають технології штучного інтелекту, а саме моделі та методи машинного навчання для виявлення аномальної поведінки серед клієнтів банку, використовуючи всю наявну інформацію про них.

Постановка завдання

1. Проведення аналізу та огляду методів для виявлення аномалій у даних.
2. Знаходження найкращих параметрів для моделей та методів машинного навчання для виявлення аномальної поведінки клієнта.
3. Виявлення аномальної поведінки методами машинного навчання з вчителем та без вчителя, використовуючи два різних набори даних: Credit Card Fraud Detection (CFD), та Bank Marketing (BM).
4. Формування висновків щодо використання того, чи іншого методу або моделі для виявлення аномалій на основі результатів метрик оцінки якості.

Предмет, об'єкт та мета досліджень

Предметом дослідження стали математичні методи машинного навчання для проведення класифікації, кластеризації, та виявлення аномалій на основі статистичних даних.

Об’єктом дослідження стали дані про транзакції Credit Card Fraud Detection (CFD), та дані, зібрані однією з банківських установ Португалії Bank Marketing (BM) (анкетні, конфіденційні дані) та їх значення для швидкого та точного виявлення підозрілих транзакцій, клієнтів з аномальною поведінкою.

Метою дослідження є розробка програмного продукту для дослідження різних моделей та методів класифікації, кластеризації, виявлення аномалій. Робота обраних для роботи методів була розглянута на практичній задачі, а саме виявлення аномалій у поведінці клієнтів банку.

Поняття аномалії

Виявлення аномалій, відоме також як виявлення викидів, новизни та шуму - це завдання визначення за допомогою спеціалізованих моделей випадків, характеристики яких значно відрізняються від отриманих знань.

Розглянемо типи аномалій:

1. Непередбачувані аномалії - виникають через помилки (систематичні та / або випадкові) у процесі збору даних.
2. Навмисні аномалії - виникають через визначені дії. Ці аномалії можуть давати цінну інформацію про набір даних, бо можуть виявити унікальні прояви чи тенденції.

Моделі та методи виявлення аномалій

З учителем	Без учителя
k найближчих сусідів (KNN)	DBSCAN
Гаусівський наївний байєсівський метод класифікаці (GNB)	OPTICS
XGBoost	BIRCH
Нейронні мережі (NN або MLP)	Суміш гаусівських моделей (GMM)
	Ізоляційні ліси (IF)
	Локальний фактор викидів (LOF)

Метрики для моделей класифікації та виявлення аномалій

1. **Precision:** $Precision = TP/(TP + FP)$
2. **Recall:** $Recall = TP/(TP + FN)$
3. **F1:** $F_1 = \frac{2 \times Precision}{Precision + Recall}$
4. **AUC.**

TP - кількість істинно позитивних значень.

TN - кількість істинно негативних значень.

FP - кількість невірно позитивних значень.

FN - кількість невірно негативних значень.

Метрики для моделей кластеризації

Якщо відомі розмічені дані:

1. **Homogeneity:** $h = 1 - entropy(y_{true}|\hat{y})/entropy(y_{true})$
2. **Completeness:** $c = 1 - entropy(\hat{y}|y_{true})/entropy(\hat{y})$
3. **V-measure:** $v = (1 + \beta)hc/(h + c)$

$$entropy(p) = \sum_{i=1}^{N} p_i \ln p_i$$

$\hat{y}$ - спрогнозовані мітки, y_{true} - істинні мітки

Якщо не розмічені дані:

1. **Index Davies-Bouldin :** $DBI = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j} \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}$, σ_i - середня відстань між точками у кластері i , c_i - центроїд кластеру i
2. **Silhouette:** $(b(x) - a(x))/\max\{a(x), b(x)\}$

   $a(x) = 1/(|C_x| - 1)\sum_{j \in C_x, x \neq j} d(x, j)$, C_x-кластер, якому належить точка x, $b(x) = \min_{K \neq X} \frac{1}{|C_K|}\sum_{j \in C_K} d(x, j)$
3. **Index Calinski-Harabasz:**

   $CH = \frac{Tr(B_k)}{Tr(W_k)} \times \frac{N-k}{k-1}$, $Tr(B_k)$ - слід міжкластерної матриці розкиду, $Tr(W_k)$ - слід матриці розкиду всередині кластеру, N - потужність вибірки, k - кількість кластерів.

Опис набору даних Credit Card Fraud Detection

Набір даних містить транзакції що відбулися за два дні і були здійснені за допомогою кредитних карток у вересні 2013 року європейцями. Серед них 492 шахрайства з 284 807 звичайних операції. Тобто, одразу можна зробити висновок, що даний датасет є незбалансованим

Атрибути:

1. V1 - V28 - основні конфіденційні компоненти, отримані за допомогою PCA (Principal component analysis - техніка зниження розмірності даних).
2. Time - час(с) між кожною транзакцією і першою транзакцією.

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

Опис набору даних Bank Marketing

Дані були зібрані за допомогою опитування через телефонні дзвінки) португальської банківської установи.

y (надалі або y, або Class) - цільова змінна, яка означає, чи надали клієнту строковий депозит.

Кількість прикладів у наборі - 4119. З них ті, що Class рівний - 3668 прикладів, а тих, у яких Class є рівним одиниці - 451 приклади.

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	fri	...	2	999	0	nonexistent	-1.8	92.893	-46.2	1.313	5099.1	no
1	39	services	single	high.school	no	no	no	telephone	may	fri	...	4	999	0	nonexistent	1.1	93.994	-36.4	4.855	5191.0	no
2	25	services	married	high.school	no	yes	no	telephone	jun	wed	...	1	999	0	nonexistent	1.4	94.465	-41.8	4.962	5228.1	no
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	fri	...	3	999	0	nonexistent	1.4	94.465	-41.8	4.959	5228.1	no
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	mon	...	1	999	0	nonexistent	-0.1	93.200	-42.0	4.191	5195.8	no

5 rows × 21 columns

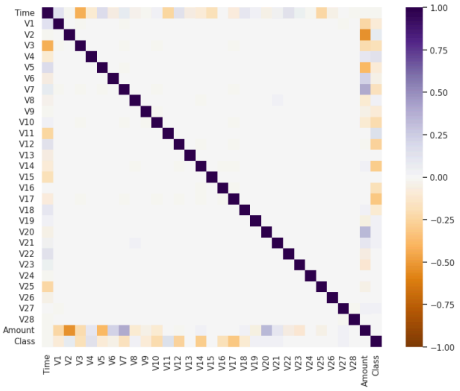
Опис змінних набору даних Bank Marketing

Основні атрибути: age - вік клієнта; job - вид діяльності; marital - сімейний статус; education - рівень освіти; default - наявність простроченого кредиту; balance - середній щорічний баланс; housing - наявність іпотеки; loan - наявність персонального кредиту; contact - тип комунікації; day_of_week - день тижня останнього контакту; month - місяць останнього контакту; duration - тривалість останнього контакту у секундах; campaign - кількість контактів з працівником установи; pdays - кількість днів, які минули після останнього контакту; previous - кількість контактів, зроблених у попередній кампанії; poutcome - результат попередньої маркетингової кампанії.

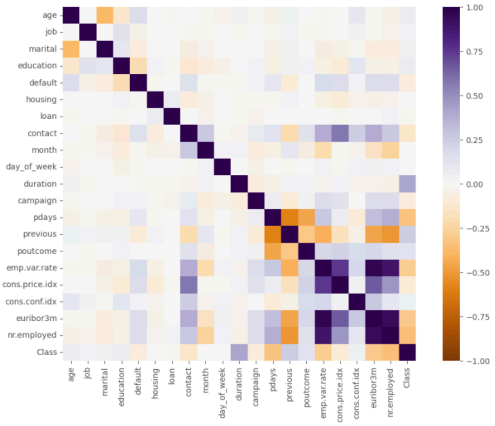
Окрім цих атрибутів, є ще emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m, nr.employed, про які не було надано жодної інформації.

Кореляційні матриця ознак для наборів даних

Для Credit Card Fraud Detection:



Для Bank Marketing:

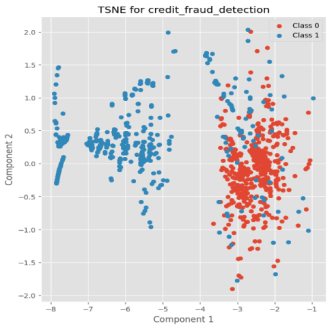


Покращення наборів даних

Категоріальні змінні кодуються. Видаляються дублікати, пропуски. Стандартизується матриця ознак.

Проблему незбалансованості класів вирішемо за допомогою методу пониження вибірки класу більшості (тобто, для наборів даних залишаємо лише ту кількість прикладів з класу більшості, яка рівна кількості прикладів з класу меншості)

Також, за допомогою методу пониження розмірності даних (t-SNE), спроектуємо набори на 2-d площину (для кращої кластеризації).



Пояснення маркування аномалій у наборах

У задачі класифікації, у наборі CFD клас з аномаліями (шахрайствами) є клас, мітки якого рівні 1, у наборі BM клас з аномальною поведінкою клієнта (ті, що мають строковий депозит) - також клас, мітки якого рівні 1 (yes).

У задачі кластеризації, для DBSCAN, OPTICS аномалії - точки, що не ввійшли до жодного кластеру. Для BIRCH, GMM - у наборі CFD кластер з аномаліями (шахрайствами) - кластер, який містить точки з мітками, рівними 1, у наборі BM кластер з аномаліями (ті, що мають строковий депозит) - кластер, мітки точок рівними 1.

Для задачі виявлення аномалій, аномальними точками є мітки класів меншості обох наборів даних, як CFD, так і BM (клас, рівний 1). Моделі LOF та IF після прогнозування повертають набір міток зі значеннями -1 або 1, де -1 означає аномальну точку, а 1 - нормальну точку.

Підбір найкращих гіперпараметрів для моделей класифікації

```
param_grids = {
    'knn': {
        'n_neighbors': np.arange(1, 31),
        'weights': ['uniform', 'distance'],
        'metric': ['euclidean', 'manhattan', 'minkowski']
    },
    'gnb': {
        'var_smoothing': np.logspace(-9, 0, 100)
    },
    'xgboost': {
        'n_estimators': np.arange(50, 300, 50),
        'max_depth': np.arange(3, 10),
        'learning_rate': np.logspace(-5, 0, 3),
        'subsample': np.arange(0.5, 1.0, 0.1)
    },
    'nn': {
        'hidden_layer_sizes': [(50, ), (70, ),(50,10),(70,10)],
        'activation': ['tanh', 'relu'],
        'solver': ['sgd', 'adam'],
        'alpha': np.logspace(-5, -1, 3),
        'learning_rate': ['constant', 'adaptive']
    }
}
```

Для пошуку найкращих параметрів з param_grids, використаємо RandomizedSearchCV (n_iter=50, cv=5).

knn - k - найближчих сусідів.
gnb - гаусівський наївний байєсівський класифікатор
nn - нейронна мережа

Найкращі параметри для моделей класифікації

KNN

Параметр	Опис параметру	CFD	BM
n_neighbors	Кількість сусідів	19	12
weights	Функція ваг для прогнозування	distance	distance
metric	Метрика для обрахунку відстані між точками	minkowski	minkowski

GNB

Параметр	Опис параметру	CFD	BM
priors	Апріорні ймовірності класів	None	None
var_smoothing	Частка найбільшої дисперсії всіх ознак, яка додається до дисперсій для стабільності розрахунку.	0.93	0.78

XGBoost

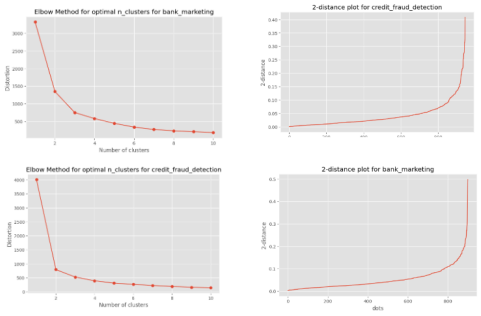
Параметр	Опис параметру	CFD	BM
subsample	Доля випадково вибраних прикладів з навчального набору даних, які будуть використані для побудови кожного дерева	0.8	0.7
learning_rate	Коефіцієнт швидкості навчання	1	0.00316
n_estimators	Кількість слабких моделей (дерев рішень), які будуть побудовані і комбіновані в процесі навчання	150	160
max_depth	Визначає максимальну глибину кожного дерева в моделі	6	7

NN (MLP)

Параметр	Опис параметру	CFD	BM
hidden_layer_sizes	i-й елемент представляє кількість нейронів в i-му прихованому шарі	(50,)	(70,)
activation	Функція активації для прихованого шару	tanh	tanh
solver	Розв'язувач для оптимізації ваги	sgd	adam
alpha	Сила регуляризації L2	0.00001	0.00001
learning_rate	Коефіцієнт швидкості навчання	constant	adaptive

Найкращі гіперпараметри для моделей кластеризації

Найкраще значення кількості кластерів, обране за допомогою метода ліктя (elbow method), для обох наборів даних **k=2** (для BIRCH n_clusters=2 та для GMM n_components=2). За допомогою методу k-distance, звужуємо можливі значення eps. Отже, ті моделі, що вчаться на даних CFD, значення eps лежить на проміжку [0.07,0.23], а для тих, що вчаться на даних BM - [0.07,0.25]. min_samples >= за кількість ознак у датасеті плюс 1. Тобто, для DBSCAN та OPTICS обираємо min_samples = 30.



DBSCAN, OPTICS

Параметр	Опис параметру	CFD	BM
min_samples	кількість точок в околицях для точки, яка розглядається як основна точка	30	30
eps	максимальна відстань між двома точками, щоб визначити, чи знаходяться вони разом	0.25	0.24
cluster_method (лише для OPTICS)	метод для виокремлення кластерів	dbscan	dbscan

17

Підбір найкращих гіперпараметрів для моделей виявлення аномалій

Для Isolation Forest, перед тренуванням на обох наборах даних, обираємо максимальну кількість зразків, які використовуються для побудови кожного дерева max_samples = 450 (приблизна кількість прикладів одного класу з кожного набору даних)

Для Local outlier Factor кількість сусідів, які використовуються для оцінки локальної щільності n_neighbors = 450 (приблизна кількість прикладів одного класу з кожного набору даних),

Частка аномальних (outliers) зразків у даних contamination = 0.4 (зумовлена нечітким поділом на кластери).

Показники якості оцінки класифікації

Для моделей, натренованих на наборі даних CFD:

Метрика \ Модель	k-NN	GNB	XGBoost	MLP
precision	**0.989011**	0.946809	0.960396	0.969388
recall	0.882353	0.872549	**0.950980**	0.931373
f1-score	0.932642	0.908163	**0.955665**	0.950000
auc	0.935495	0.907865	**0.952763**	0.948641
time	**0.004270**	0.003338	0.100535	1.24862

Для моделей, натренованих на наборі даних BM:

Метрика \ Модель	k-NN	GNB	XGBoost	MLP
precision	0.822222	0.777778	**0.869565**	0.858696
recall	0.831461	0.786517	**0.898876**	0.887640
f1-score	0.826816	0.782123	**0.883978**	0.872928
auc	0.828774	0.784563	**0.884221**	0.873168
time	0.003785	**0.002917**	0.136773	1.445863

Показники якості оцінки кластеризації

Для моделей, натренованих на наборі даних CFD:

Метрика \ Модель	DBSCAN	BIRCH	OPTICS	GMM
Homogeneity	0.563117	**0.634229**	0.557172	0.600773
Completeness	0.278934	**0.638404**	0.277608	0.620114
V-measure	0.373071	**0.636310**	0.370577	0.610290
Silhouette	0.317278	0.632344	0.311412	**0.702990**
Calinski Harabasz	510.301	2383.55	487.970	**3830.11**
Davies Bouldin	1.241661	0.513626	1.230544	**0.430935**
Час навчання	**0.014093**	0.066946	1.203735	0.102389
Кількість кластерів	4	2	4	2
Кількість аномальних точок	284	0	291	0

Для моделей, натренованих на наборі даних BM:

Метрика \ Модель	DBSCAN	BIRCH	OPTICS	GMM
Homogeneity	0.201625	0.161761	**0.201855**	0.148533
Completeness	0.105240	**0.162212**	0.105125	0.148586
V-measure	0.138295	**0.161986**	0.138250	0.148560
Silhouette	0.429848	0.523932	0.428890	**0.531673**
Calinski Harabasz	640.133	1296.135	642.671	**1319.153**
Davies Bouldin	1.437576	0.748525	1.361602	**0.734398**
Час навчання	**0.015035**	0.031627	1.150348	0.134964
Кількість кластерів	6	2	6	2
Кількість аномальних точок	15	0	16	0

Показники якості оцінки виявлення аномалій

Для моделей, натренованих на наборі даних CFD:

Метрика \ Модель	IF	LOF
Precision	0.677249	**0.968254**
Recall	0.541226	**0.773784**
F1-Score	0.601645	**0.860165**
AUC	0.641649	**0.874207**
Час тренування	0.540928	**0.110185**

Для моделей, натренованих на наборі даних BM:

Метрика \ Модель	IF	LOF
Precision	0.603878	**0.714681**
Recall	0.483370	**0.572062**
F1-Score	0.536946	**0.635468**
AUC	0.583149	**0.671840**
Час тренування	0.528810	**0.154757**

Результати по роботі

На обох наборах даних, найкраще себе проявили ті моделі моделі кластеризації, що потребують вказати кількість кластерів. За ознакою Homogeneity, Completeness, V-measure найкраща модель - BIRCH, а за Silhouette, Calinski Harabasz, та Davies Bouldin Index - GMM. Проте, показники метрик обох моделей майже однакові.

Найкраща модель для виявлення аномалій в обох наборах даних - це LOF, причому, за усіма метриками якості!

Найкраща модель для класифікації в обох наборах даних - це XGBoost.

Висновки

В ході роботи було виконано комплексний огляд предметної області, пов'язаної з виявленням аномалій у поведінці клієнтів банку. Було детально розглянуто та проаналізовано основні методи машинного навчання, як з учителем, так і без нього. Зокрема, досліджено ефективність алгоритмів k-nearest neighbors, Gaussian Naive Bayes, XGBoost для навчання з учителем та DBSCAN, OPTICS, Gaussian Mixture Models, Local Outlier Factor, BIRCH, Isolation Forest для навчання без учителя.

Для оцінки якості кластеризації застосовувалися такі метрики, а саме та класифікації застосовано відповідні метрики Precision, Recall, F1-Score, AUC та для кластеризації - Homogeneity, Completeness, V-measure, Silhouette, Calinski Harabasz, Davies Bouldin, що дозволило об'єктивно порівняти результати різних методів. Було реалізовано програмний продукт на мові програмування Python, проведено його тестування на реальних наборах даних (Credit Card Fraud Detection, Bank Marketing), а також здійснено попередню обробку та покращення наборів даних для підвищення точності моделей.

Проаналізовано результати метрик якості для кожної моделі.

Проведено функціонально-вартісний аналіз програмного продукту.

Подальші дослідження

Подальші дослідження повинні зосереджуватися на покращенні гіперпараметрів моделей та методів для виявлення аномалій у поведінці клієнтів банку. Крім цього, дослідити, як вплинуть нові ознаки на якість класифікації, кластеризації та виявлення аномалій на основі статистичних даних. Також, дослідити інші спеціалізовані моделі та методи, що підходять для даної задачі.

Дякую за увагу!