Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра мультимедійних систем

# Кваліфікаційна робота

освітній ступінь – магістр

на тему: **«СТВОРЕННЯ ДОСЛІДНИЦЬКОГО ПРОТОТИПУ ДЛЯ АНАЛІЗУ МОЖЛИВОСТЕЙ EXPLAINABLE AI»**

Виконав: студент 6-го року навчання,
Спеціальності
122 Комп’ютерні науки
Чалюк Андрій Олександрович
Керівник Олецький О. В.,_____
Рецензент ._____
(прізвище та ініціали)

Кваліфікаційна робота захищена з оцінкою _____
Секретар ЕК _____
“___” _____ 20____ р.

Київ – 2025

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ
Кандидат технічних наук, доцент
Олецький О.В.

_____
*(підпис)*
“_____” _____2025 р.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**
на дипломну роботу

студенту Чалюку Андрію факультету Інформатики 6 року навчання
ТЕМА **«СТВОРЕННЯ ДОСЛІДНИЦЬКОГО ПРОТОТИПУ ДЛЯ АНАЛІЗУ МОЖЛИВОСТЕЙ EXPLAINABLE AI»**

Зміст текстової частини магістерської роботи:

Зміст
Анотація
Вступ
Огляд літератури
Методологія
XOR
MNIST
Brain Tumor
Висновки
Список літератури

Дата видачі “___” _____ 2024р.

Керівник
Олецький О. В. _____
*(підпис)*

Завдання отримав

Чалюк А. О _____

*(підпис)*

**Тема:** Створення дослідницького прототипу для аналізу можливостей Explainable AI

**Календарний план виконання роботи:**

| № п/п | Перелік робіт | Термін виконання | Примітка |
|---|---|---|---|
| 1. | Отримання завдання на кваліфікаційну роботу | 17.10.2024 | |
| 2. | Огляд технічної літератури за темою роботи | 26.11.2024 | |
| 3. | Виконання аналізу сучасних рішень | 12.12.2024 | |
| 4. | Методологія та розробка прототипу | 16.02.2025 | |
| 5. | Впровадження та експерименти | 14.03.2025 | |
| 6. | Аналіз отриманих результатів | 08.04.2025 | |
| 7. | Написання текстової частини | 28.04.2025 | |
| 8. | Узгодження попередньої версії роботи з керівником | 21.05.2025 | |
| 9, | Внесення змін до кваліфікаційної роботи відповідно до зауважень наукового керівника | 25.05.2025 | |
| 10. | Попередній захист | 30.05.2025 | |
| 11. | Корегування роботи за результатами попереднього захисту | 06.06.2025 | |
| 12. | Захист кваліфікаційної роботи | 12.06.2025 | |

Науковий керівник _Олецький О. В.__

Виконавець кваліфікаційної роботи _Чалюк А. О.___

“_____” _____

# ЗМІСТ

# СПИСОК СКОРОЧЕНЬ

XAI – Explainable Artificial Intelligence

ІШІ – Штучний інтелект

TCAV – Testing with Concept Activation Vectors

GLM – Generalized Linear Model

GAM – Generalized Additive Model

LIME – Local Interpretable Model-agnostic Explanations

PDP – Partial Dependence Plot

ICE – Individual Conditional Expectation

c-ICE – centered Individual Conditional Expectation

ALE – Accumulated Local Effect

DiCE – Diverse Counterfactual Explanation

VAE – Variational Autoencoder

PCA – Principal Component Analysis

t-SNE – t-Distributed Stochastic Neighbor Embedding

KAN – Kolmogorov–Arnold Network

ACE – Automated Concept Extraction

CRP – Concept Relevance Propagation

DEC – Deep Embedded Clustering

ADEC – Adversarial Deep Embedded Clustering

IMSAT – Information-Maximization Clustering

GAN – Generative-adversarial Network

NLP – Natural Language Processing

MSE – Mean Squared Error

NMI – Normalized Mutual Information

ARI – Adjusted Rand Index

MLP – Multi Layer Perceptron

CNN – Convolutional Neural Network

LSTM – Long short-term memory

ResNet – Residual neural network

GAT – Graph Attention Network

NB – Naive Bayes

NN – Neural Network

CAM – Class Activation Maps

# АНОТАЦІЯ

У цій роботі оглянуто існуючі методи XAI. Проведено аналіз та оцінку цих методів на синтетичних та візуальних тестах. Розглянуто можливість їх використання для різного типу моделей. Проведено порівняння цих і методів за рядом критеріїв.

*Ключові слова: explainable AI, лінійні методи, деревовидні методи, counterfactual explanation, concept explanation, perturbation методи.*

# ВСТУП

**Актуальність.** За останнє десятиліття моделі машинного навчання досягли неабиякої продуктивності в широкому спектрі завдань: від розпізнавання зображень і обробки природної мови до медичної діагностики та фінансового прогнозування. Глибокі нейронні мережі регулярно перевершують людську точність на еталонних наборах даних, таких як ImageNet, а архітектури на основі трансформерів забезпечують масштабне розуміння мови. Проте, незважаючи на ці успіхи, їхні внутрішні процеси прийняття рішень залишаються значною мірою непрозорими і це викликає серйозні занепокоєння щодо довіри, підзвітності та справедливості, коли ці моделі приймають відповідальні рішення. У сценаріях з високими ставками – наприклад, у медицині для прогнозування хвороби на основі результатів сканувань або у фінансовій сфері для схвалення кредитів – знання того, що прогнозує модель, уже недостатньо. Зацікавлені сторони вимагають чітких, зрозумілих для людини обґрунтувань для перевірки, оскарження або уточнення автоматизованих рішень.

Ось чому пояснюваний штучний інтелект (Explainable AI, XAI) став критично важливою дисципліною: його методи спрямовані на подолання розриву між складною точністю сучасних моделей і прозорістю, необхідною для довіри, підзвітності та дотримання вимог. Виявляючи особливості, закономірності та концепції, що лежать в основі кожного прогнозу, XAI дає можливість аналітикам даних налагоджувати моделі, експертам певної області – перевіряти рішення, а кінцевим користувачам – розуміти, чому саме було отримано такий результат і, зрештою, приймати поведінку систем машинного навчання.

Наразі існує широкий спектр методів XAI, від методів атрибуції ознак (наприклад, карти значущості, пояснення значень Шеплі) до аналізу на основі концепцій (наприклад, TCAV). Однак багато підходів змушують шукати компроміс між інтерпретованістю та точністю, або ж розроблені для однієї модальності даних чи класу моделей, що ускладнює порівняння між методами.

**Мета дослідження.** Огляд відомих методів XAI, їх аналіз та оцінка на різноманітних тестах: синтетичних (XOR) та візуальних (MNIST, МРТ пухлин головного мозку), а також можливостей використання для різного типу моделей, та порівняння за кількома критеріями: точністю, достовірністю пояснення, обчислювальною вартістю і можливістю інтерпретації людиною.

**Об’єкт дослідження.** Моделі машинного навчання та механізми їх пояснення.

**Предмет дослідження.** Інтерпретованість моделей машинного навчання, інструменти для цього.

**Джерела дослідження.** Рецензовані статті та матеріали конференцій про методи та теоретичні засади XAI (наприклад, статті про LIME, SHAP, TCAV), електронні версії друкованої літератури, електронні ресурси, онлайн-документація та сховища коду, загальнодоступні еталонні набори даних (MNIST, Brain-Tumor),

# РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРИ

## 1.1. Класифікація методів XAI.

З огляду на поширення методів XAI, чітка таксономія має важливе значення для порівняння підходів і вибору правильного інструменту для конкретного випадку використання. Використовується тривимірну таксономію, вперше сформульовану Adadi & Berrada (2018)[1], пізніше вдосконалену Barredo Arrieta (2019)[2], яка організовує методи пояснюваності вздовж осей «Вбудовані» та «Постфактум», «Глобальні» та «Локальні», а також «Специфічні для моделі» та «Модель-агностичні».

Внутрішня інтерпретованість стосується моделей, які є прозорими і зрозумілими за своєю природою (наприклад, невеликі дерева рішень, лінійні регресії, прості списки правил). Їхні параметри і структуру можна безпосередньо перевірити і зрозуміти без додаткових методів пояснення.

Постфактичні пояснення застосовуються після навчання складних непрозорих моделей (наприклад, глибоких нейронних мереж). Ці методи розглядають модель як «чорну скриньку» і генерують пояснення. Наприклад, карти значущості або сурогатні моделі – без зміни ваг або архітектури оригінальної моделі.

Глобальні пояснення характеризують загальну поведінку моделі або межі прийняття рішень. Вони відповідають на питання на кшталт: «На які характеристики модель найбільше покладається з усіх вхідних даних?».

Локальні ж пояснення у свою чергу обґрунтовують один прогноз для одного запису даних. Вони пояснюють, чому модель передбачила саме цю мітку для цього конкретного прикладу. Локальні методи часто відхиляють вхідні дані у певному діапазоні або перевіряють градієнти навколо прикладу, що нас цікавить.

Специфічні для моделі методи використовують внутрішню інформацію моделі, таку як градієнти, активації або ваги уваги. Оскільки вони вимагають доступу до архітектури або параметрів моделі, вони не можуть бути застосовані до довільних систем.

Модель-агностичні методи розглядають модель суто як функцію «вхід-вихід», будуючи пояснення через спостереження за змінами, які відбуваються на виході моделі, після певних конкретних маніпуляцій з входами. Ця загальність досягається за рахунок потенційної втрати ефективності або достовірності.

Окрім цих трьох основних напрямків, нещодавні дослідження вводять додаткові відмінності для відображення нових потреб [3].

Сурогатні підходи навчають інтерпретовану модель (наприклад, дерево рішень) імітувати оригінал, тоді як прямі методи обчислюють пояснення з самої моделі (наприклад, на основі градієнта).

Деякі нові методи XAI використовують графіки причинно-наслідкових зв'язків або інтервенційні методи, на відміну від атрибуцій, що базуються виключно на кореляційних зв'язках.

Також розрізняють класифікацію за тим, чи призначені пояснення для експертів предметної області, такі пояснення повинні бути детальними і технічними, або ж для непрофесійних користувачів – спрощені, описові пояснення.

1.2. Лінійні методи.

Лінійні моделі є основою статистичного навчання і залишаються безцінними для пояснення завдяки своїй прозорій параметризації. У найпростішому випадку лінійна регресія підганяє зважену суму вхідних ознак для прогнозування безперервного результату; вивчені коефіцієнти безпосередньо кількісно оцінюють внесок кожної ознаки. Однак, коли ознаки сильно корелюють, звичайні оцінки лінійної регресії за методом найменших квадратів стають нестабільними і схильними до перенавчання.

Регуляризовані лінійні моделі, зокрема, Ridge та Lasso, вирішують цю проблему шляхом введення штрафів за величину коефіцієнта. Ridge регресія додає штраф $L_2$ ($\lambda \sum_i \beta_i^2$), який пропорційно зменшує всі коефіцієнти до нуля, зменшуючи дисперсію за рахунок невеликого зміщення. Lasso регресія

використовує штраф $L_1$ ($\lambda \sum_i |\beta_i|$), який не лише контролює перенавчання, але й забезпечує розрідженість, наближаючи багато коефіцієнтів точно до нуля.

Lasso Regularization Path відображає кожен коефіцієнт $\beta_i$ як функцію параметра регуляризації (сили штрафу) та візуально демонструє порядок, в якому ознаки входять або виходять з моделі, що дає уявлення про важливість і стабільність кожної ознаки.

Лінійна регресія передбачає, що цільова величина є неперервною та нормально розподіленою, але багато проблем пов'язані з бінарними мітками або дискретними ознаками. Generalized Linear Model (GLM) розширюють можливості інтерпретації для цих параметрів, вводячи функцію зв'язку, яка пов'язує лінійний предиктор з очікуваною ціллю. Наприклад, використання логіт-зв'язку дає логістичну регресію для бінарних результатів, де кожен коефіцієнт представляє зміну лог-шансів на одиницю збільшення його ознаки. Незважаючи на таку додаткову гнучкість, GLM зберігають ключову перевагу прямої інтерпретації коефіцієнтів [4].

Хоча GLM зберігають лінійність кожної ознаки, реальні взаємозв'язки часто є нелінійними. Generalized Additive Model (GAMs) вирішують цю проблему, замінюючи кожен коефіцієнт гладкою одновимірною функцією відповідної ознаки. Модель залишається адитивною сумою цих функцій, тому можна візуалізувати вплив кожного члена окремо - як правило, за допомогою кривої, що показує, як змінюється прогноз при зміні цієї ознаки. Таким чином GAM досягають балансу між інтерпретованістю та гнучкістю, дозволяючи даним виявляти нелінійні закономірності, не жертвуючи при цьому чіткістю пояснень, що ґрунтуються на окремих ознаках [5].

1.3. Методи засновані на деревах.

Дерева рішень є одними з найбільш прозорих моделей машинного навчання: рекурсивно розбиваючи простір ознак, вони дають набір зрозумілих людині правил «якщо-то», які відображають вхідні змінні на прогнози. Шлях від кореня до листка відповідає зрозумілому людині правилу («якщо ознака A > 0.5 і ознака B ≤ 1.2, то клас = X»), що робить кожне передбачення

простежуваним за допомогою кінцевого набору порогових умов. Дерева природним чином балансують між точністю та інтерпретованістю: неглибокі дерева з невеликою кількістю розгалужень дають компактні правила, але можуть не точно передавати чи взагалі не відповідати дійсності, тоді як глибокі дерева фіксують нюанси і дають кращі результати ціною більш складних наборів правил, через що страждають інтерпретованість результатів і загальність [6].

Щоб отримати глобальну інформацію з більш складних і точних, але непрозорих моделей, можна навчити дерево рішень як сурогатну модель: воно навчиться імітувати вихідні дані оригінальної моделі, надаючи приблизний, але зрозумілий вигляд її поверхні рішень. Сурогатне дерево підганяється під великий набір вхідних точок, позначених складнішою моделю, пояснення якої стоїть на меті, а потім обрізається до меншого розміру для кращої інтерпретованості. Незважаючи на деяку втрату точності, отримане дерево слугує компактним, глобальним наближенням оригінальної моделі – можна витягти її набір правил та візуалізувати її структуру [7].

RuleFit розвиває цю ідею, закладену у дерева рішень, видобуваючи правила з ансамблю дерев, а потім підганяючи розріджену лінійну модель за цими показниками правил. Спочатку він генерує сотні або тисячі правил-кандидатів шляхом підбору багатьох неглибоких дерев, а потім вибирає і зважує розріджену підмножину з них у регуляризованому лінійному ансамблі. Результатом є модель, прогноз якої розкладається на декілька правил, кожне з яких має вагу, що інтерпретується. Порівняно з одним деревом, RuleFit часто досягає вищої точності, забезпечуючи при цьому прозору структуру пояснення, основаного на правилах [8].

1.4. Модель-агностичні методи, основані на відхиленнях (perturbations).

Модель-агностичні методи, основані на відхиленнях, пояснюють довільні моделі машинного навчання, спостерігаючи, як контрольовані зміни на входах впливають на виходи. Оскільки вони не вимагають доступу до внутрішніх

параметрів або структури, їх можна однаково добре застосовувати до нейронних мереж, ансамблів дерев або будь-яких інших моделей.

LIME (Local Interpretable Model-agnostic Explanations) будує простий, інтерпретований сурогат (як правило, лінійну модель) в околиці окремого екземпляра. Випадковим чином впливаючи (збурюючи) на характеристики цього екземпляра і підганяючи сурогат для відтворення прогнозів «чорної скриньки», LIME присвоює вагу кожній характеристиці, що вказує на її локальну важливість. Ці ваги можна візуалізувати у вигляді гістограми, що дозволяє легко побачити, які ознаки найбільше вплинули на конкретний прогноз [9].

Anchors створюють високоточні правила «якщо-то» («якорі»), які гарантують, із заданою користувачем впевненістю, що будь-який екземпляр, який задовольняє правилу, отримає той самий прогноз моделі. Anchors зосереджуються на тому, щоб пояснення застосовувалися до чітко визначеної області вхідного простору, а не на створенні глобально достовірного наближення [10].

Permutation Feature Importance оцінює внесок кожної ознаки шляхом випадкового перемішування її значень у тестовому наборі та вимірює, наскільки погіршується продуктивність моделі (наприклад, точність або похибка). Значне падіння продуктивності вказує на те, що ознака була критичною, тоді як незначні зміни означають, що модель не покладалася на цю ознаку. На відміну від сурогатних моделей, цей підхід безпосередньо вимірює справжню чутливість моделі за рахунок переоцінки на переставленому наборі даних для кожної ознаки без повторного навчання, але цей метод може бути дуже ресурсозатратним.

Partial Dependence Plot (PDP) показує, як в середньому змінюється прогноз моделі, коли одна ознака змінюється в межах свого діапазону, а всі інші ознаки залишаються на своїх спостережуваних значеннях. Для побудови PDP потрібно для певної ознаки j вибирати сітку значень, що охоплює область значень цієї ознаки, для кожної точки сітки замінити ознаку j у кожному

тестовому прикладі на це значення та оцінити модель, усереднити прогнозовані результати для всіх змінених екземплярів. Нанесення цих середніх значень на сітку показує типовий вплив цієї ознаки - чи реакція моделі зростає лінійно, чи демонструє порогову поведінку, чи демонструє більш складні закономірності. PDP легко обчислювати та інтерпретувати, але коли ознаки корелюють, вони можуть змішувати ефекти від нереалістичних комбінацій ознак.

Графіки Individual Conditional Expectation (ICE) є схожими до PDP, проте відображають одну криву для кожного тестового прикладу, а не усереднюють. Це показує, як різні підгрупи поводяться незалежно – можливо, один кластер прикладів демонструє різке зростання, тоді як інший спадає – у PDP усереднення таких результатів б призвело б до їх втрати. Щоб чіткіше показати ці відмінності, використовують centered ICE (c-ICE): віднімаємо середній прогноз для кожного прикладу (по сітці) від усієї кривої. Це вирівнює всі криві навколо нуля, тому варіації нахилу і форми виділяються, роблячи ефекти взаємодії і неоднорідну поведінку відразу видимими [11].

Графіки Accumulated Local Effects (ALE) усувають ключовий недолік PDP, коли ознаки корелюють між собою: усереднення за малоймовірними комбінаціями ознак може ввести в оману. Замість цього ALE ділить діапазон ознаки на невеликі інтервали і вимірює зміну прогнозу при переході від нижньої до верхньої межі кожного інтервалу, але тільки для вибірок, які дійсно потрапляють в цей інтервал. Ці локальні відмінності потім накопичуються від найнижчого інтервалу вгору. Результатом є крива, яка відображає середній вплив ознаки, зберігаючи при цьому справжній розподіл даних, і за значно менших обчислювальних витрат, ніж обчислення повної PDP на дрібній сітці [12].

1.5. Методи на основі градієнтів.

Методи атрибуції на основі градієнта відстежують рішення моделі до її входів, обчислюючи карти чутливості - як правило, часткову похідну вихідної оцінки щодо кожного вхідного пікселя або ознаки. Ці методи не вимагають

перенавчання моделі і можуть бути застосовані як постфактичні методи до будь-якої диференційованої мережі.

Вперше представлений Simonyan та співавт., Saliency Map просто бере градієнт оцінки класу по відношенню до кожного вхідного пікселя і візуалізує абсолютне значення градієнта у вигляді теплової карти. Яскраві області показують, де невелика зміна інтенсивності пікселя найбільше вплине на прогнозовану оцінку. Хоча ці карти прості та швидкі, вони, як правило, зашумлені і виділяють дрібні деталі без семантичного контексту. Тому часто перед їх використанням пікселі на зображенні групують [13].

Guided Backpropagation вдосконалює Saliency, обнуляючи негативні градієнти під час зворотного проходу, ефективно відфільтровуючи гальмівні сигнали. Результатом є чіткіші, більш сфокусовані карти, які часто краще узгоджуються з людською інтуїцією. Однак, оскільки він все ще працює на рівні пікселів, він може пропустити більші патерни або не локалізувати ширші структури [14].

Sundararajan та співавт.. запропонували інтегровані градієнти (Integrated Gradients, IG) для вирішення проблеми насиченості градієнта. Замість того, щоб робити вибірку лише на вході, цей метод усереднює градієнти вздовж прямого шляху від еталоного зразка (наприклад, чорного зображення) до фактичного входу. Інтегровані градієнти дають більш плавні атрибуції, які краще відображають як локальні, так і глобальні ефекти, задовольняючи при цьому аксіоматичні властивості, такі як повнота (сума атрибуцій дорівнює вихідній різниці) [15].

Grad-CAM переміщує фокус з вхідних пікселів на внутрішні карти ознак. Беручи градієнти оцінки класу відносно активацій кінцевого згорткового шару, потім зважуючи та підсумовуючи ці карти ознак, Grad-CAM створює грубу, специфічну для класу теплову карту локалізації. Вона виділяє широкі області інтересу, такі як частини об'єкта, що робить її особливо корисною для згорткових нейронних мереж у задачах технічного зору. На відміну від карт на

рівні пікселів, Grad-CAM фіксує семантику більш високого рівня, але з нижчою просторовою роздільною здатністю [16].

Усі візуалізації на основі зворотного поширення мають спільну залежність від градієнтів мережі, що робить їх чутливими до шуму та негативних відхилень. Методи на рівні пікселів можуть бути складними для інтерпретації при накладенні на складні сцени, тоді як методи на рівні функціональних карт, такі як Grad-CAM, можуть бути занадто грубими для дрібних деталей. На практиці дослідники часто поєднують кілька підходів - наприклад, накладають результати Guided Backpropagation на теплові карти Grad-CAM, щоб збалансувати семантичну локалізацію з дрібнозернистою деталізацією. Ці градієнтні інструменти залишаються наріжним каменем пояснюваного глибокого навчання, формуючи основу для більш просунутих методів, заснованих на прототипах і концептах.

1.6. Контрфактичні пояснення.

Контрфактичні пояснення (Counterfactual explanations) мають на меті показати користувачам, як потрібно змінити вхідні дані, щоб змінити рішення моделі. Замість того, щоб розкривати внутрішню механіку класифікатора, вони надають інтуїтивно зрозумілі сценарії «що було б, якби». Це тісно пов'язано з людськими міркуваннями про причинно-наслідкові зв'язки, що робить контрфакти особливо привабливими у сферах з високими ставками, таких як фінанси або охорона здоров'я.

Diverse Counterfactual Explanation (DiCE) генерує набір альтернативних випадків, які змінюють передбачення моделі, спираючись на 3 умови:

- Близькість, щоб кожне контрфактичне пояснення було якомога ближчим до вихідних даних;
- Різноманітність, щоб запропонувати кілька якісно різних варіантів;
- Достовірність, яка гарантує, що кожен кандидат справді спричиняє бажану зміну результату.

DiCE формулює це як обмежений пошук у просторі ознак, використовуючи градієнтні оновлення та функції штрафу, що сприяють

різноманітності. Його реалізація з відкритим вихідним кодом підтримує табличні, графічні та текстові дані, що робить його універсальною відправною точкою для експериментів [17].

Variational autoencoders (VAE) вивчають гладке латентне представлення розподілу вхідних даних. Щоб згенерувати контрфакти, можна закодувати оригінальний екземпляр у його латентний простір, потім шукати в цьому латентному просторі його близький вектор, реконструйований вихід якого змінює рішення моделі. І декодувати цей код, щоб отримати реалістичний контрфактичний приклад, яка відповідає множині даних.

Оскільки VAE моделюють спільний розподіл усіх ознак, їхні контрфакти, як правило, дотримуються природньої кореляції. Вони також уникають неправдоподібних відхилень, які можуть спричинити прості градієнтні кроки. Однак якість значною мірою залежить від точності реконструкції та безперервності латентного простору [18].

Контрфактичні методи надають користувачам уявлення, які саме значення і як потрібно змінити, щоб отримати конкретний результат. Вони також безпосередньо ілюструють границі рішень, що полегшує інтерпретацію та налагодження моделі. Але вони можуть мати і недоліки:

- Неправдоподібні результати, якщо згенеровані екземпляри випадають з розподілу реальної множини даних (пом'якшуються за допомогою VAE або GAN).
- Обчислювальні витрати, особливо при пошуку в просторах високої розмірності або при забезпеченні різноманітності.
- Неунікальність, що змушує користувачів обирати між кількома пропозиціями без чітких критеріїв ранжування.

1.7. Зменшення розмірності для інтерпретації.

Високовимірні представлення - незалежно від того, чи це вхідні дані моделі, чи це високорозмірні векторні представлення (ембедінги) вивчених ознак, чи вектори атрибуції, отримані за допомогою методів XAI, - може бути важко перевірити безпосередньо. Методи зменшення розмірності дають змогу

спроектувати ці високорозмірні об'єкти у дво- або тривимірний простір, де можливо візуально оцінити такі закономірності, як структура кластеризації, відокремлюваність класів або взаємозв'язок між пояснюваними екземплярами. Вивчаючи ці проекції разом з прогнозами моделі або оцінками пояснення, дослідники отримують інтуїтивне розуміння того, як модель організовує інформацію.

Principal Component Analysis (PCA) - один з найпоширеніших лінійних методів. Він визначає ортогональні напрямки (компоненти) в даних, які фіксують найбільшу дисперсію, і проєктує кожну точку даних на головні компоненти. PCA є швидким, детермінованим і забезпечує чітке ранжування компонентів, що робить його особливо корисним для аналізу [19].

t-Distributed Stochastic Neighbor Embedding (t-SNE) - це нелінійна техніка, призначена для збереження локальних взаємозв'язків між сусідами. Вона перетворює високовимірні простори в умовні ймовірності, а потім оптимізує низьковимірний ембедінг так, щоб точки, які були близькими у вихідному просторі, залишалися близькими у проекції. Це робить t-SNE особливо користним у виявленні дрібнозернистої підструктури - наприклад, кластерів, що перекриваються, або перехідних шаблонів між класами, - які PCA може пропустити. Однак t-SNE чутливий до гіперпараметрів і може спотворювати глобальні взаємозв'язки, тому його краще використовувати в тандемі з більш стабільними лінійними проекціями. На практиці дослідники часто застосовують спочатку PCA, щоб зменшити розмірність (наприклад, до 50 компонент) перед запуском t-SNE, досягаючи балансу між обчислювальними витратами та якістю кінцевої візуалізації [20].

1.8. Мережа Колмогорова-Арнольда.

Мережа Колмогорова-Арнольда (Kolmogorov–Arnold Network, KAN) використовує класичну теорему суперпозиції Колмогорова–Арнольда, яка стверджує, що будь-яку неперервну функцію багатьох змінних можна розкласти на суми та суперпозиції одновимірних функцій, щоб побудувати невелику нейронну архітектуру, яку можна інтерпретувати. У ній немає

лінійних вагів – кожен ваговий параметр замінюється одновимірною функцією, параметризованою одновимірними сплайнами. KAN дає можливість отримати не лише числове передбачення і сплайни, але й символічну формулу, що описує модель.

Основна перевага KAN полягає в його теоретичній інтерпретованості: кожен прихований юніт відповідає окремій одновимірній функції, яку можна перевірити індивідуально. Однак значні обчислювальні витрати та чутливість до ініціалізації роблять його менш практичним, ніж простіші сурогати або пояснювачі на основі кластеризації.

1.9. Методи основані на прототипах і концептах.

У той час як більшість методів XAI пояснюють окремі прогнози в термінах низькорівневих ознак або відхилень, підходи на основі прототипів і концептів намагаються ґрунтувати пояснення на інтерпретованих людиною даних - або репрезентативних прикладах, або семантично значущих концептах.

Прототипні мережі (Prototypical Networks) вивчають набір векторів-зразків - прототипів - під час навчання і пояснюють нові зразки, показуючи найближчі прототипи разом з їхніми класовими мітками. У задачах комп’ютерного зору кожен прототип може бути зіставлений з фрагментом реального навчального зображення, що дозволяє користувачам бачити конкретні приклади, які підтримують класифікацію. Таким чином, мережі прототипів надають пояснення на основі конкретних прикладів: «ця цифра - 3, тому що вона схожа на ці збережені приклади». Однак вони, як правило, вимагають ретельного налаштування архітектури і не так легко узагальнюються до ситуативних налаштувань [22].

Testing with Concept Activation Vectors (TCAV) пропонує глобальний погляд на рівні концепту, вимірюючи, наскільки чутливий вихід моделі до заздалегідь визначеного концепту (наприклад, «смугастий візерунок» або «наявність петлі»). Збирається невеликий набір прикладів концептів, обчислюється лінійний класифікатор у просторі активації (CAV), а потім оцінюється, як часто переміщення вхідних даних у цьому напрямку змінює

прогнози. TCAV дає стислу оцінку важливості концепту для кожної пари концепт-клас, але покладається на заздалегідь підготовлені дані про концепт і не пояснює окремі випадки [23].

Automated Concept Extraction (ACE) автоматично виявляє візуальні концепти, кластеризуючи патерни активацій у згорткових шарах, а потім фільтрує кластери, щоб залишити ті, які можна інтерпретувати людиною. На відміну від TCAV, який вимагає ручного вибору концепції, ACE створює як прототипи концепцій, так і оцінки їхнього впливу на основі даних. Тим не менш, ACE зосереджується на візуальному виявленні концепту і не дає пояснень для кожної вибірки; він не пропонує механізму для об'єднання декількох прототипів концепту в одну узгоджену атрибуцію для індивідуального прогнозу [24].

Concept Relevance Propagation (CRP) поєднує локальні мапи релевантності (attribution maps) з глобальними поясненнями концептів, спочатку проєктує мапи релевантності (на рівні пікселів) на вивчені концепти, а потім поширюючи внески на рівні концептів на кінцевий результат. Цей двоетапний процес може виявити, які високорівневі концепти найбільше вплинули на рішення, але він успадковує шум і нестабільність градієнтних карт на першому етапі [25].

# РОЗДІЛ 2. МЕТОДОЛОГІЯ

## 2.1. Набори даних.

У цій роботі проводиться оцінка методів пояснення для трьох різних наборів даних (синтетичних, зображень і медичних зображень). Нижче підсумовано ключові властивості кожного набору даних та кроки попередньої обробки, що застосовуються перед навчанням та поясненням моделі.

Як найпростіший набір даних для дослідження поведінки різних моделей і методів, було створено синтетичний набір даних XOR. Кожен приклад складається з двох бінарних ознак $x_1$ та $x_2$, які визначають цільове значення як $y = x_1 \oplus x_2$. Для ускладення набір було доповнено 8 додатковими бінарними ознаками. Ці шумові ознаки не мають кореляції між цільовим значенням. Загалом створено 1000 прикладів: 800 записів складають навчальну вибірку і 200 – тестову. Оскільки усі ознаки бінарні, масштабування чи будь-яка інша попередня обробка (препроцесинг) даних не застосовувалась.

Набір даних MNIST складається з 70000 чорнобілих зображень рукописних цифр від 0 до 9: навчальний набір з 60000 зображень і тестовий набір з 10000 зображень. Кожне зображення має розмір 28×28 пікселів. Для моделей, що вимагають векторних вхідних даних (наприклад, лінійна регресія, GLM, GAM), ми переводимо кожне зображення у 784-вимірний вектор і нормалізуємо інтенсивність пікселів до [0,1]. Для згорткових нейронних мереж зображення зберігаються у двовимірному вигляді (28 × 28 × 1) з тією ж нормалізацією. Жодна додаткова аугментація (повороти, зсуви) не застосовується.

Також використано загальнодоступний набір даних «brain-tumour-MRI-scan» від Simezu [31], що містить 7023 T1-зважених МРТ-зрізів у чотирьох класах - гліома, менінгіома, пухлина гіпофіза та група без пухлин. 5710 зображень утворюють нашу навчальну вибірку і 1313 зображень - тестову. Оригінальні зображення мають різний розмір, тому вони зводяться до фіксованої роздільної здатності 128×128 і конвертуються в одноканальну шкалу сірого. Інтенсивність пікселів масштабується до [0,1]. Гармонізація

інтенсивності після цих кроків не проводиться, щоб зберегти природну варіабельність набору даних.

Ці три набори даних забезпечують комплексну основу для оцінки взаємозв'язку між точністю моделі та інтерпретованістю.

2.2. Метрики.

Щоб оцінити ефективність прогнозування і якість пояснення використовується досить багато метрик.

Mean Squared Error (MSE): середнє квадратичне відхилення між прогнозованими та істинними значеннями. Нижчий показник MSE свідчить про відповідність моделі даним [26].

Коефіцієнт детермінації ($R^2$): частка дисперсії цільового показника, яку пояснює модель. Вищі значення (до 1) вказують на більш пояснювальну модель. Від'ємні значення свідчать про те, що модель передбачає гірше, ніж тривіальна константна оцінка [26].

Accuracy (точність): частка правильних прогнозів для всіх випадків [26].

Precision: частка випадків, передбачених як позитивні (або певного класу), які дійсно є позитивними.

Recall: частка правильно ідентифікованих позитивних випадків (або певного класу) серед усіх випадків, які є справді позитивними.

$F_1$ Оцінка: середнє гармонійне значення precision та recall, що балансує обидва показники.

Для багатокласових завдань ми наводимо усереднені показники точності, precision, recall та $F_1$ оцінки (тобто усереднюємо кожну метрику однаково для всіх класів).

Purity: для кожного кластера частка членів, які мають найпоширенішу істинну характеристику, усереднена по кластерах. Вища purity означає, що кластери ближчі до істинних класів.

Normalized Mutual Information (NMI): вимірює спільну інформацію між мітками кластерів та істинними мітками, нормалізовану в діапазоні від 0 (немає спільної інформації) до 1 (ідеальний збіг) [27].

Adjusted Rand Index (ARI): порівнює парні узгодження між кластерами та істинними мітками з поправкою на випадковість. ARI може бути від'ємним, але досягає максимуму в 1 [27].

Silhouette Score: схожість для кожної точки даних порівнюється з її власним кластером та найближчим іншим кластером, усереднюючи за всіма точками. Оцінки варіюються від -1 (погане розділення) до 1 (добре розділення) [28].

Fidelity: наскільки точно спрощене або сурогатне пояснення відтворює вихідні ймовірності оригінальної моделі, коли важливі ознаки вилучаються або залишаються. Нижче середнє відхилення означає вищу точність.

Classification Fidelity: частка випадків, для яких сурогатне пояснення прогнозує той самий клас, що й оригінальна модель.

Stability: середня схожість пояснень для одного й того ж об’єкта при різних початкових станах випадкового генератора. В якості міри схожості використано коефіцієнт Жаккара (відношення перетину множин до їх об’єднання) між множинами ключових ознак, відібраних методом атрибуції.

Local fidelity: коефіцієнт детермінації $R^2$ між ймовірностями, передбаченими оригінальною моделлю, та апроксимацією лінійного сурогата у невеликому околі кожної точки. Тобто оцінює, наскільки добре проста модель відтворює поведінку складної моделі у маленькому околі навколо кожного прикладу.

Error Coverage: частка тестових прикладів, для яких механізм пояснення не дає правильного результату (не відтворює прогноз оригінальної моделі).

Complexity (Comp): середня кількість елементів у поясненні (наприклад, кількість ознак, виділених LIME, кількість правил у сурогатному дереві, кількість ненулевих коефіцієнтів у Permutation Importance), подана зі стандартним відхиленням. Простіші пояснення мають меншу складність.

Interpretability Score (Mean H): показник зрозумілості для людини, що базується на довжині пояснення або, за наявності, на оцінках користувачів - вищі значення означають зрозуміліші пояснення. У нашому випадку оцінок

користувачів не було, тому значення метрики базувались виключно на довжині пояснення.

Deletion & Insertion AUC: два взаємодоповнюючі показники — один відстежує продуктивність моделі при послідовному видаленні елементів за ступенем важливості, а інший — при послідовному додаванні елементів. Площі під цими кривими відображають корисність пояснення [30].

Pearson Correlation (r): кореляція між картою атрибуції (наприклад, оцінками важливості) та еталонним сигналом (наприклад, вхідними градієнтами), усереднена за вибірками. Вищі значення вказують на краще узгодження з еталоном [29].

TCAV Score: частка прикладів у наборі концептів, для яких спрямована похідна вздовж цього концепту є додатною, кількісно оцінюючи важливість концепту для даного класу [23].

2.3. Неможливість існування сильного explainable AI

Означення: Сильний XAI – це механізм чи алгоритм A, який для будь-якої AI-системи f (розглянутої як функція f: X→Y) та будь-якого довільного вхідного значення x генерує формальне пояснення A(f, x) таке, що воно є:

Повним, тобто пояснює чому було отримано f(x) і воно містить стільки інформації, щоб однозначно відтворити мапування f на всіх можливих входах.

Коректним, тобто пояснення точно відображає алгоритм прийняття рішення системи f.

Якщо f є достатньо потужною, то вона може емулювати будь-яку машину Тьюринга, тобто є Turing-повною. Що у свою чергу означає, що для неї притаманні усі властивості обчислювальної універсальності.

Припустимо, що існує такий універсальний сильний механізм XAI A. Тоді для довільної нетривіальної властивості P (тобто властивості, що не виконується для всіх функцій або не відсутня для всіх функцій) можна побудувати алгоритм A*, який для заданої AI-системи f (Тюрінг-повної системи):

- Обчислює пояснення A(f, x) для деякого фіксованого x.

- Аналізує отримане пояснення з метою вирішення чи має f властивість P.

Що суперечить теоремі Райса, за якою не існує алгоритму, який міг би вирішити, чи має машина Тюрінга певну нетривіальну властивість [32].

Проте це стосується лише Тюрінг-повних моделей. Для певних класів моделей (наприклад, лінійних моделей, дерев рішень та інших моделей з обмеженою обчислювальною потужністю) можна надати повні пояснення, оскільки у цих випадках моделі не є Тюрінг-повними.

Але, наприклад, рекурентні нейронні мережі, через наявність зворотніх зв’язків можуть імітувати кроки обчислення Тьюринг-машини. Доведено, що рекурентна мережа з достатньою точністю ваг є еквівалентною машині Тьюринга [33].

Також це стосується лише повного пояснення роботи моделі. Апроксимовані і локальні пояснення є достатньо інформативними (наприклад, LIME або SHAP).

# РОЗДІЛ 3. XOR

## 3.1. Лінійні моделі

Для Linear, Ridge і Lasso регресій обраховано метрики MSE та $R^2$ (таблиця 3.1).

| | Linear Regression | Ridge Regression | Lasso Regression |
|---|---|---|---|
| MSE | 0.25 | 0.25 | 0.25 |
| $R^2$ Score | -0.01 | -0.01 | -0.00 |

*Таблиця 3.1. Значення метрик MSE та $R^2$ для різних типів регресії*

Mean Squared Error для всіх цих моделей становить 0.25 – типове значення, якщо модель для всіх прикладів передбачала результат близько 0.5, тобто модель не знайшла закономірності між входом і виходом. Про те саме свідчить $R^2$ метрика, яка повернула приблизно 0 (або навіть від'ємні значення) для усіх моделей, тобто не знашла кореляції між вхідними ознаками та цільовою функцією, що очікувано, враховуючи нелінійну природу XOR.

Щодо коефіцієнтів моделей, то Linear та Ridge дають подібні, невеликі за модулем коефіцієнти для $x_1$, $x_2$ та шумових змінних. Lasso з параметром $\alpha = 0.1$ обнулило всі коефіцієнти, це свідчить про те, що жодна з лінійних комбінацій ознак не дає відчутного покращення над простою константною оцінкою при такому рівні регуляризації (рисунок 3.1).

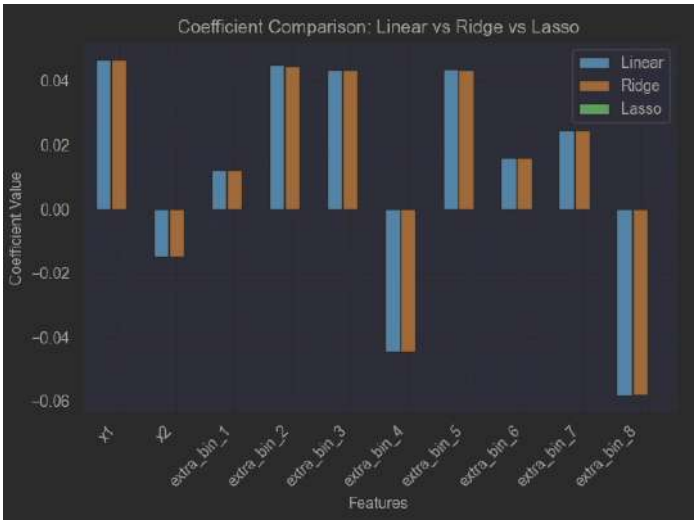



*Рисунок 3.1. Порівняння коефіцієнтів Linear, Ridge та Lasso регресій*

Для логістичної регресії отримані коефіцієнти відрізнялись за величиною, проте відношення між коефіцієнтами різних ознак зберігались (рисунок 3.2).

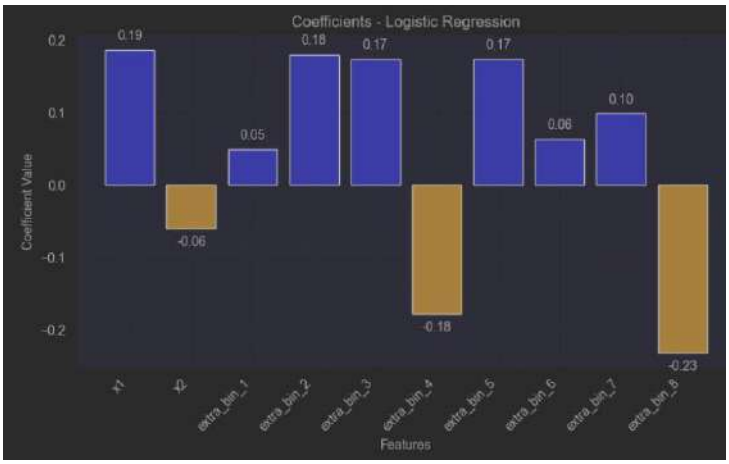



*Рисунок 3.2. Візуалізація коефіцієнтів Logistic Regression*

Lasso Regularization Path, показує як змінюються коефіцієнти Lasso при зміні параметра регуляризації. Він візуалізує, які ознаки є найбільш стійкими до обнулення, що вказує на їхню важливість для моделі.

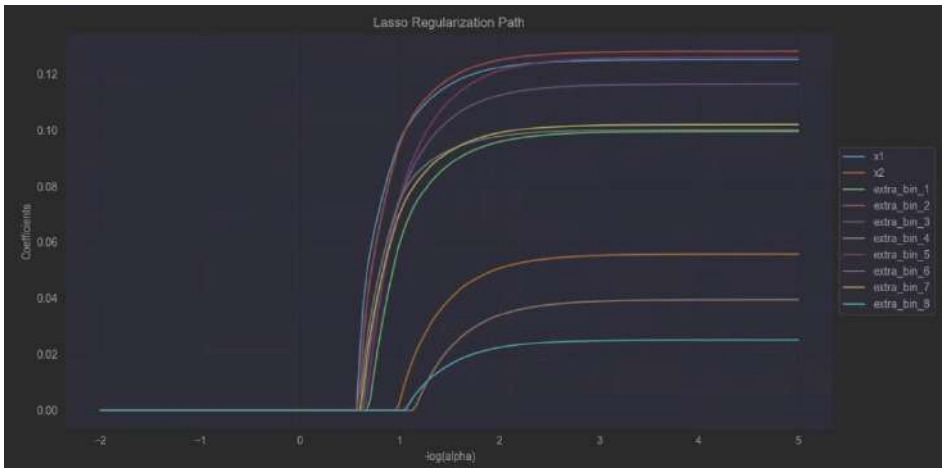



*Рисунок 3.3. Lasso Regularization Path*

Вісь x представляє -log(α), тобто зліва на право значення параметра регуляризації зменшується. Вісь y – значення коефіцієнтів кожної ознаки (рисунок 3.3).

Точка, де значення стає ненульовим, вказує, коли ця ознака входить в модель. Ознаки, які входять в модель з більшою регуляризацією, як правило, є більш важливими. Ознаки, коефіцієнти яких залишаються відносно стабільними в широкому діапазоні значень альфа, зазвичай є більш надійними предикторами. При низькій регуляризації коефіцієнти мають тенденцію збігатися до своїх звичайних значень за методом найменших квадратів (що еквівалентно стандартній лінійній регресії). Ознаки, лінії яких рухаються

схожими шляхами, можуть бути корельованими або мати схожі зв'язки з цільовою змінною.

Оскільки операція XOR є нелінійною, регресія передбачає середнє (0.5), що і дає MSE = 0.25 і $R^2 <= 0$. Для кращих результатів необхідно додати нелінійні ознаки (наприклад, $x_1 * x_2$ – у такому випадку отримуємо 100% точність).


*Рисунок 3.4. Logistic Regression з додаванням нелінійних змінних*

Для GLM з біноміальним розподілом (Binomial) отримано:

MSE: 0.2524, $R^2$: -0.0122, що дуже близько до результатів отриманих при лінійній регресії і свідчить, що модель не знайшла закономірності між вхідними ознаками і вихідною змінною.

Також використано Gamma GLM:

Ця модель зазвичай використовується для позитивних безперервних цільових змінних, тому вона погано підходить для вирішення XOR задачі. Отримані результати:

MSE = 110.49 – дуже висока помилка для змінної, що приймає значення лише 1 та 2 (для цієї задачі використано зміщення, для умови позитивності цільової змінної).

$R^2$ = -441.36 – показує, що модель працює гірше, ніж константна модель.

Проте навідміну від Binomial розподілу, у якому коефіцієнти мають приблизно однакові значення, для Gamma коефіцієнти $x_1$ та $x_2$ мають значно вищі значення, ніж шумові змінні, що вказує, що модель правильно знаходить змінні, які впливають на результат, але через гамма розподіл і неперервну

експоненційну природу методу не змогла побудувати булеву функцію (рисунок 3.5).

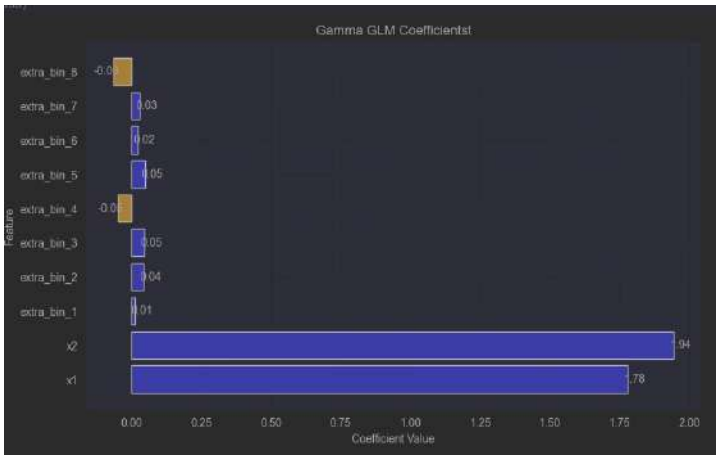



*Рисунок 3.5. Візуалізація коефіцієнтів Gamma GLM*

GLM також доцільно доповнити додатковими нелінійними ознаками. Для випадку з додатковою ознакою $x_1 \times x_2$ також отримуємо точний результат (рисунок 3.6).

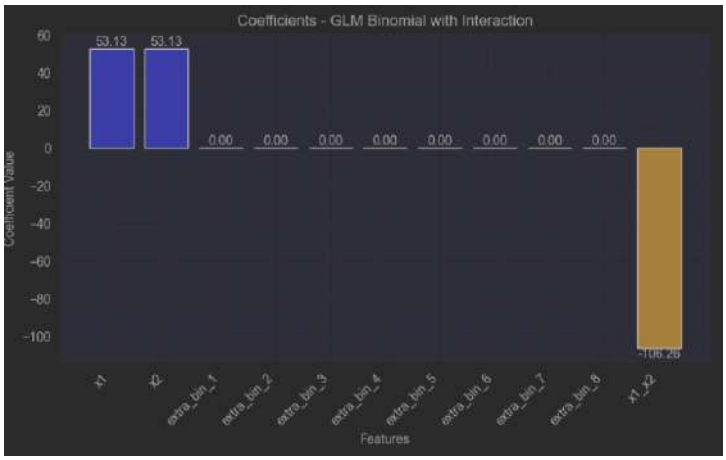



*Рисунок 3.6. Візуалізація коефіцієнтів Binomial GLM з нелінійними ознаками*

Для GAM отримано схожі до попередніх моделей результати:

| | MSE | $R^2$ |
|---|---|---|
| LinearGAM | 0.25 | 0 |
| LogisticGAM | 0.2494 | -1.2472 |

*Таблиця 3.2. Метрики для LinearGAM та LogisticGAM*

Partial dependence – це спосіб оцінити, як зміна значення певної ознаки (або кількох ознак) впливає на середній прогноз моделі, зафіксувавши або

усереднивши вплив усіх інших ознак. У GAM кожна ознака має власну нелінійну функцію [11].

Більшість кривих є лінійними і мають невеликий нахил. Це означає, що кожна окрема ознака має слабку залежність із виходом у (рисунок 3.7).

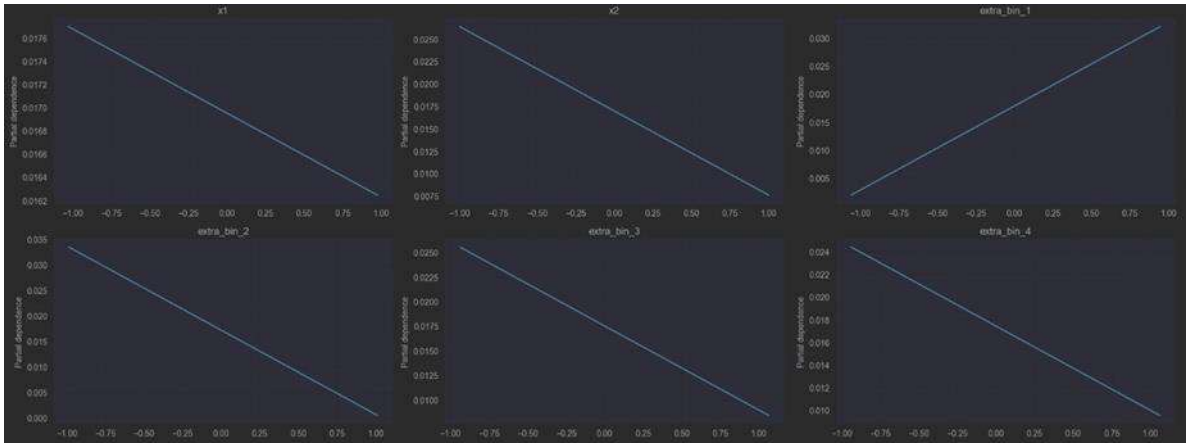



*Рисунок 3.7. Візуалізація кривих для LinearGAM*

3.2. Дерева рішень.

Дерева рішень (використані DecisionTreeClassifier і DecisionTreeRegressor параметрами з criterion=”gini” і “entropy”) можуть показувати досить різні результати в залежності від того, як згенеровані і розділені дані та які ознаки обираються для розділення. У XOR розбиття на будь-який один біт - сигнал або шум - дає ідентичну залежність (без зменшення ентропії). Після того, як зроблено початкове розбиття за шумовою ознакою, наступні розщеплення намагаються лише локально зменшити помилку.

Так, максимальної глибини 2 вистачає для точності у 100% для цієї задачі, проте часто на практиці алгоритм обирає неправильну ознаку для розгалуження, оскільки з точки зору ентропії розділення за кожною окремою ознакою дає приблизно однаковий результат. Зі збільшенням глибини точність зростає. Дерева без обмежень глибини ж дають точність 100% - вони усувають помилку навчання, оскільки розбивають шум необмежену кількість разів і можуть використати усі змінні, але це дає багато помилкових правил і вони можуть не узагальнюватись на тестові зразки. Також у таких дерев незалежно від їх структури важливими ознаками є лише $x_1$ і $x_2$ (рисунок 3.8).

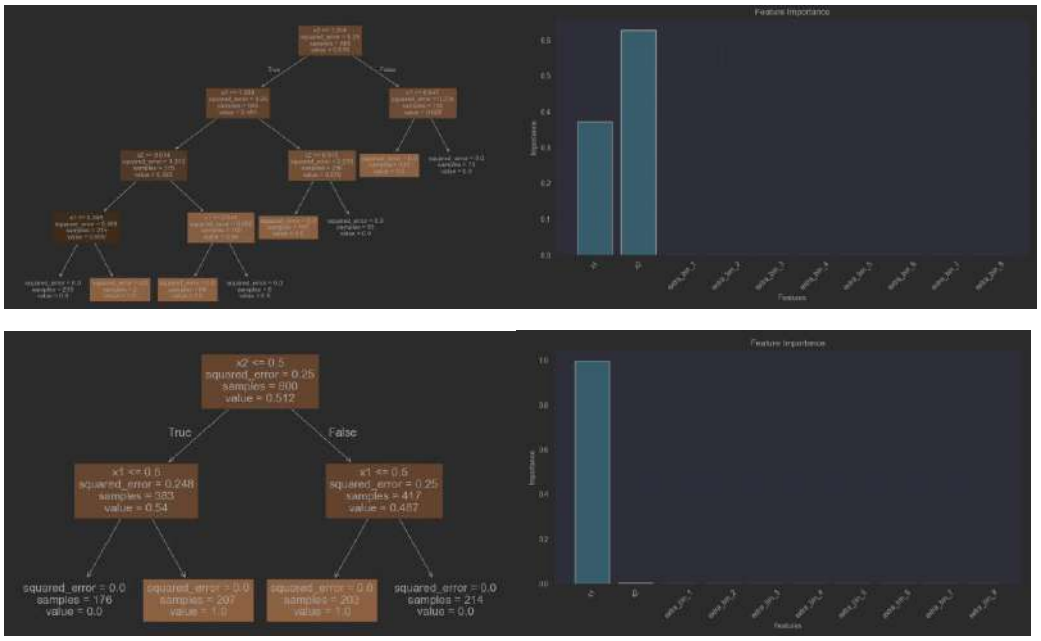



*Рисунок 3.8. Структури дерев рішень і візуалізації важливості ознак*

Для навченого дерева ми проходимо його внутрішню структуру вузлів, щоб витягти всі відображення лист-правило: кожне правило є кон'юнкцією тестів. Наприклад:

*Leaf 2: (f5 <= 0.500) and (f1 <= 0.500), predicted_class=1, n_samples=273, impurity=0.492*

*Leaf 3: (f5 <= 0.500) and (f1 > 0.500), predicted_class=0, n_samples=241, impurity=0.499*

*Leaf 5: (f5 > 0.500) and (f9 <= 0.500), predicted_class=1, n_samples=250, impurity=0.500*

*Leaf 6: (f5 > 0.500) and (f9 > 0.500), predicted_class=0, n_samples=236, impurity=0.490*

Щоб перевірити правильність отриманих правил, їх точність, синтезовано новий набір даних, вибравши по 50 прикладів на кожне правило. Для кожного правила:

- Зафіксовано ознаки, згадані в його антецеденті, у відповідному значенні 0/1.
- Випадковим чином призначаємо всі інші ознаки в {0, 1}.

Таким чином отримано синтетичну множину, мітки якої точно відповідають логіці дерева. Далі навчаємо MLP (один прихований шар з 10

нейронами) на оригінальному наборі даних, отримано точність 100% і оцінено його точність на синтетичній множині. Експеримент проводився зі зростаючою кількістю шумових ознак у оригінальному датасеті. У таблиці нижче наведено підсумки для дерев глибини 3 та необмеженої глибини:

| Noise Feats | Unrestricted rules | Synthetic samples | MPL accuracy on synthetic | Depth=3 rules | Synthetic samples | MPL accuracy on synthetic |
|---|---|---|---|---|---|---|
| 0 | 4 | 200 | 100% | 4 | 200 | 100% |
| 2 | 4 | 200 | 100% | 4 | 200 | 100% |
| 4 | 20 | 1000 | 100% | 8 | 400 | 50% |
| 6 | 28 | 1400 | 100% | 8 | 400 | 77.75% |
| 8 | 254 | 12700 | 91.93% | 8 | 400 | 50% |
| 10 | 284 | 14200 | 77.65% | 8 | 400 | 55.75% |
| 12 | 343 | 17150 | 70.12% | 8 | 400 | 51.50% |
| 14 | 280 | 14000 | 67.01% | 8 | 400 | 53.25% |
| 16 | 448 | 22400 | 63.32% | 8 | 400 | 49.25% |
| 18 | 576 | 28800 | 61.07% | 8 | 400 | 50.50% |

*Таблиця 3.3. Кількість правил, розмір синтетичних даних та точність MLP зі збільшенням кількості шумових характеристик*

Для дерева необмеженої глибини кількість правил і розмір синтетичної множини значно збільшуються зі зростанням шуму, а точність MLP на синтетичних даних поступово падає зі збільшення кількості правил.

Для дерева глибини 3 кількість правил обмежена 8, розмір синтетичної множини фіксований на 400 (по 20 на правило), але точність MLP сильно коливається (50-100 %), оскільки шум довільно зміщує перші три розбиття і чим більше шумових ознак, тим менша ймовірність, що серед обраних 3 будуть правильні. Це відбувається через згадану проблему з приблизно однаковою ентропією для кожної з ознак.

Далі застосовано алгоритм RuleFit (використано бібліотечну реалізацію RuleFitRegressor), який поєднує розріджену лінійну модель з функціями правил прийняття рішень, до набору даних XOR у двох конфігураціях: чистий RuleFit і з gradient boosting:

| Method | MSE |
|---|---|
| RuleFit | 0 |
| RuleFit + GradientBoosting | 0.16 |

*Таблиця 3.4. MSE для методів на основі RuleFit*

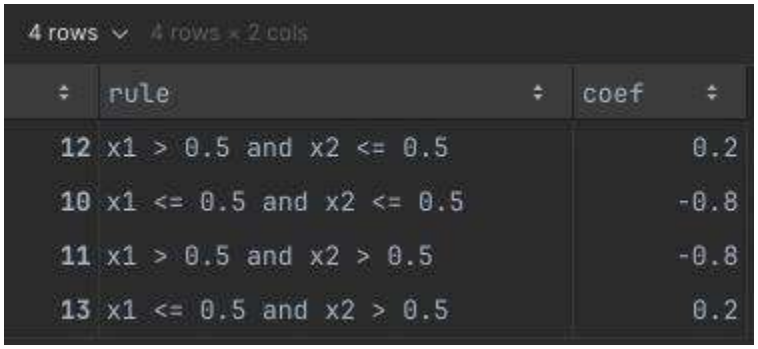



*Рисунок 3.9. Отримані правила RuleFit*

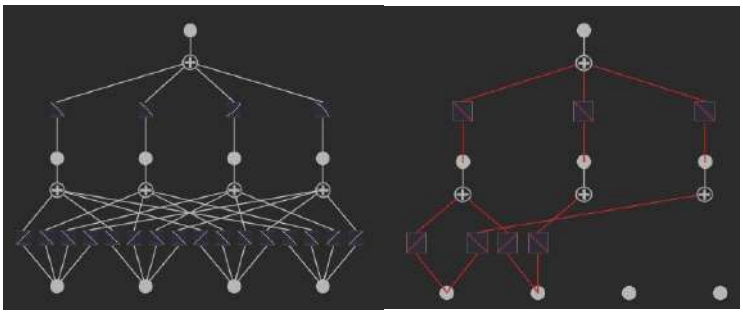
Ці правила описують логіку XOR лише в декількох правилах, що є добре зрозумілими для людей, і тому є гарною альтернативою моделям-чорним скринькам.

3.3. KAN

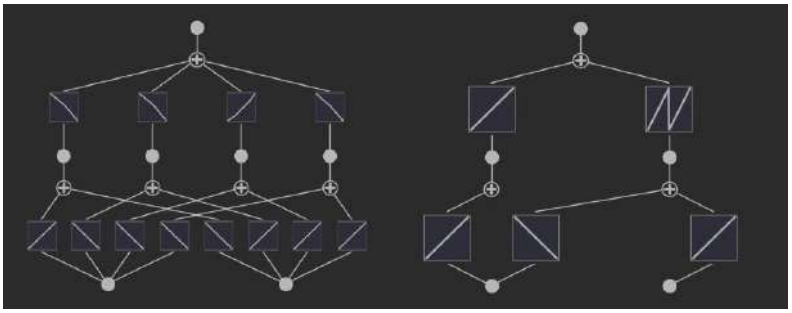
KAN розглядається як універсальний апроксиматор, який навчається представляти відображення XOR, проектуючи вхідні дані у латентний простір заданої розмірності, а потім рекомбінуючи їх за допомогою простих одновимірних активацій.

Реалізовано невелику мережу прямого поширення з одним прихованим шаром із 4 нейронів (для вхідної розмірності 2) і проведено експерименти з розширеними вхідними даними (розмірностями 5 і 10). KAN дає можливість отримати не лише числове передбачення і сплайни, але й символічну формулу, що описує модель.

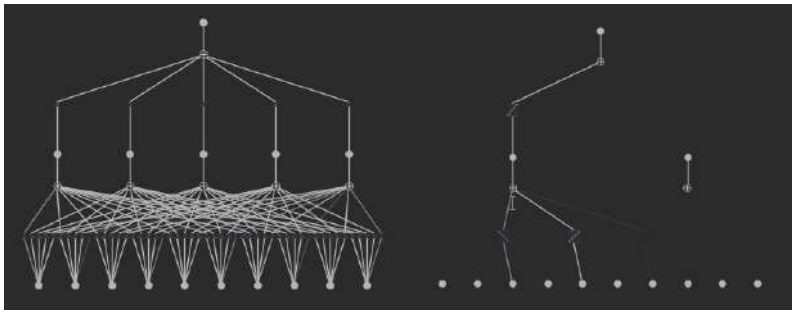
Для зазначених моделей отримано такі результати:



$y = -0.00834223457450588x_1 – 0.00439237828357661x_2 + 0.337767400254255$



$y = 0.00051449041264442 6x_1 + 0.000506331731172302x_2 + 0.493249092217568$



$y = -0.034807512287185x_3 + 0.13614309180078x_5 – 0.0047030777549413 2x_7 + 0.0224704404473567$

*Рисунок 3.10. Результати KAN*

У кожному випадку коефіцієнти є невеликими. Це означає, що мережа майже не залежить від вхідних значень. XOR є бінарною, тому істинна функція стрибає від 0 до 1 на межі розв'язку. Але підмережі KAN використовують плавні активації, тому як тільки вони наближаються до цього стрибка, їх градієнти зникають - навчання зупиняється. Мінімальна реалізація KAN просто не має достатньої кількості підмереж (або глибини) для виділення кожного кута квадранта істинності XOR. В результаті вона зводиться до тривіального константного розв'язку з кількома невеликами поправками.

### 3.4. Perturbation методи

Для застосування perturbation методів використовується класифікатор XGBoost (eXtreme Gradient Boosting з налаштуваннями за замовчуванням). Ця модель добре підходить для завдань класифікації завдяки своїй здатності працювати з нелінійними залежностями та автоматичному відбору ознак. На наборі даних XOR отримано 100% точність.

**LIME**

LIME пояснює передбачення моделі для конкретного екземпляру даних шляхом побудови локальної лінійної апроксимації навколо цього екземпляру.

Кроки алгоритму LIME:

- Обираємо приклад із набору даних, для якого прагнемо отримати детальне пояснення.
- Генеруємо додаткові зразки шляхом випадкового відхилення ознак навколо обраного прикладу та отримуємо передбачення від початкової складної моделі для цих нових зразків.
- Створюємо локальний датасет, що складається зі отриманих зразків з відхиленням і відповідних прогнозів.
- На отриманому локальному датасеті тренуємо просту (наприклад, лінійну) модель, при цьому кожен зразок зважується залежно від того, наскільки він близький до вихідного прикладу.
- Аналізуємо коефіцієнти простої моделі, аби зрозуміти, які ознаки найсуттєвіше вплинули на прогноз у межах цієї локальної околиці.

LIME використовує експоненційне ядро, яке визначає вагу кожного синтетичного зразка залежно від його близькості до обраного прикладу. Параметр kernel_width контролює розмір околиці:

- Мала ширина ядра означає, що локальна модель зважує майже виключно ті зразки, які дуже близькі до вихідного прикладу.

- Велика ширина ядра дає змогу враховувати й віддаленіші зразки, роблячи пояснення більш глобальними (менш чутливим до точкового прикладу).

Використано LIME для різних зразків.

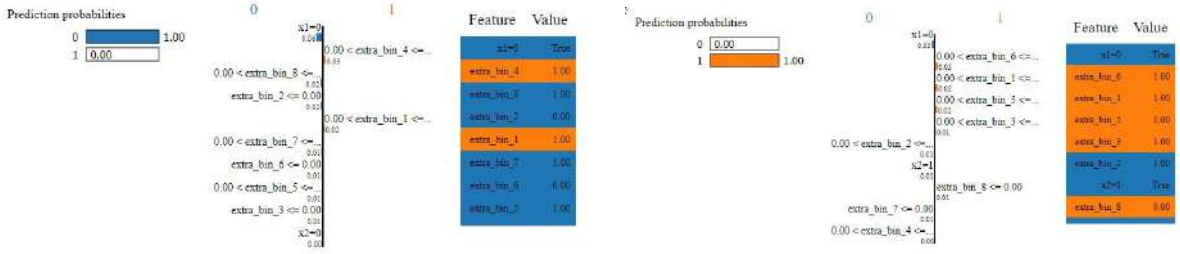



*Рисунок 3.11. Візуалізація результатів LIME*

У першому прикладі ймовірність класу 0 становить 100%, а ймовірність класу 1 – 0%. На вертикальній шкалі відображаються ознаки, які LIME визначив як важливі для передбачення класу. Список ознак праворуч показує кольором ознаки, як вони впливають на результат. Проте усі ваги є близькими до 0, тобто вони не мають сильного впливу на рішення.

**Anchors**

Anchors формують набір умов (логічних правил), які гарантують з високою ймовірністю однаковий прогноз моделі для всіх прикладів, що задовольняють ці умови.

Кроки алгоритму Anchors:

- Обираємо екземпляр, для якого бажаємо отримати детальне пояснення.
- Як і в LIME, алгоритм створює відхилення від прикладу, щоб перевірити, чи залишаються прогнози моделі сталими за невеликих змін у вхідних ознаках.
- Алгоритм формує початкову гіпотезу (набір умов), що може пояснювати, чому модель дає конкретний прогноз.
- Правило покроково уточнюється (додаються нові умови або змінюються існуючі), щоб збільшити частку зразків, які відповідають цьому правилу та отримують такий самий прогноз.

- Після кожного уточнення алгоритм оцінює, наскільки надійно правило утримує незмінним прогноз моделі для всіх (або переважної більшості) сусідніх прикладів.
- Результатом є набір умов, які з високою ймовірністю фіксують передбачення моделі. Наприклад, «x1 ≤ 0.5 та x2 = 1» може бути умовою, що «якорить» певний клас.

Приклад використання anchors (бібліотечна реалізація anchor_tabular):

*Anchor: 0.00 < extra_bin_2 <= 1.00 AND 0.00 < extra_bin_3 <= 1.00 AND 0.00 < extra_bin_5 <= 1.00 AND 0.00 < extra_bin_6 <= 1.00 AND 0.00 < extra_bin_1 <= 1.00 AND extra_bin_8 <= 0.00 AND extra_bin_7 <= 1.00 AND extra_bin_4 <= 1.00 AND x1 = 0 AND x2 = 1*

*Precision: 1.00*

*Coverage: 0.00*

Незважаючи на ідеальну точність для цієї вибірки, надзвичайно низьке покриття вказує на те, що якір є занадто специфічним, щоб слугувати загальним поясненням логіки XOR. Це ілюструє загальний компроміс: anchors можуть бути ідеально точними, але можуть бути недостатньо загальними.

**SHAP**

Значення Шейплі (Shapley values) - це метод з теорії коаліційних ігор (де гравці співпрацюють у коаліції), який показує, як справедливо розподілити виплати між гравцями залежно від їхнього внеску в загальну виплату.

Значення SHAP пояснюють прогноз, порівнюючи його з базовим значенням – середнім прогнозом для базового набору даних. Цей бейзлайн слугує відправною точкою для розуміння того, наскільки кожна ознака впливає на різницю між фактичним прогнозом і середнім прогнозом по фонових даних.

При обчисленні значень SHAP використовується набір фонових даних (background dataset) для оцінки очікуваного значення результату моделі. Це очікуване значення є середнім прогнозом по фоновому набору даних і представляє те, що модель передбачила б за відсутності конкретної інформації про даний екземпляр.

Значення SHAP вимірюють граничний внесок кожної ознаки в прогноз. Це передбачає розгляд того, що модель передбачила б, якби кожна ознака була відсутня або замінена значеннями з фонового розподілу. По суті, значення SHAP кількісно показують, який внесок кожна ознака робить у різницю між фактичним прогнозом і базовим значенням, отриманим з фонового набору даних.

Використання базового набору даних дозволяє SHAP наближено оцінити вплив вилучення ознаки шляхом усереднення багатьох можливих значень, які могла б прийняти ознака. Це більш ефективно з точки зору обчислень, ніж розгляд усіх можливих комбінацій значень ознаки.

Для демонстрації використано бібліотечну реалізацію (shap).

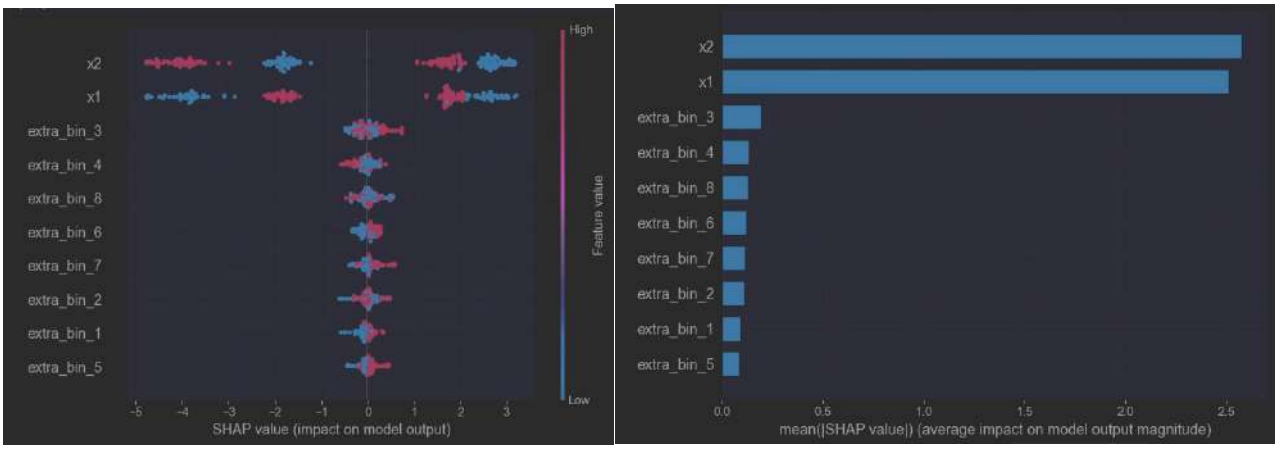



*Рисунок 3.12. Візуалізація значень SHAP*

SHAP було застосовано для кількісної оцінки середнього внеску кожної ознаки в рішення XOR для всіх тестових прикладів. Ознаки $x_1$ та $x_2$ мають найвищий середній вплив на прогнози моделі, усі ж інші ознаки мають майже нульовий середній вплив.

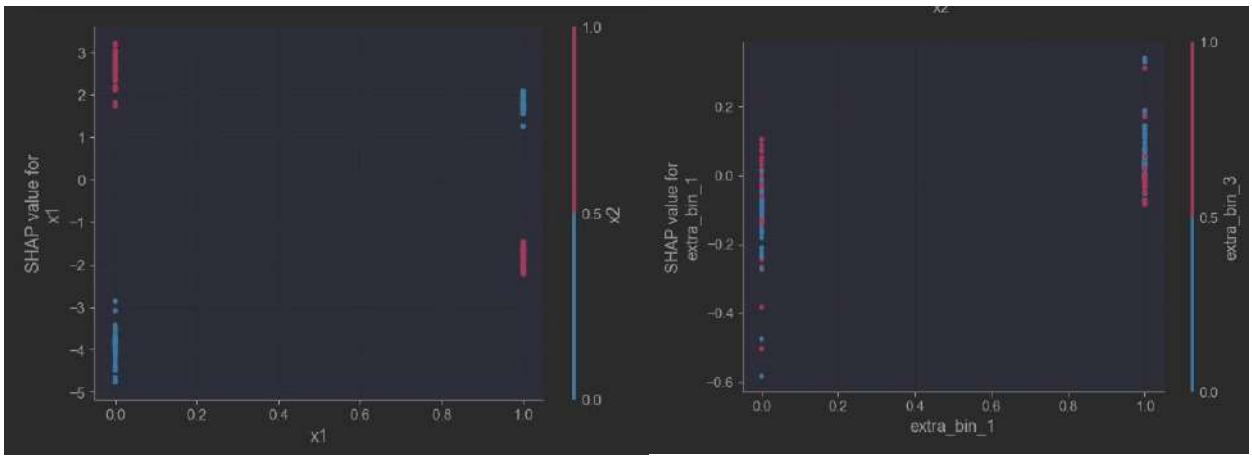


*Рисунок 3.13. Візуалізація значень SHAP для окремих ознак*

SHAP підтверджує, що $x_1$ та $x_2$ є єдиними факторами, що визначають вихід моделі, кожен з яких робить однаковий внесок, але в протилежних напрямках, залежно від значення іншої ознаки. Шумові ознаки «extra_bin» ефективно ігноруються, що ілюструє здатність SHAP відфільтровувати нерелевантні ознаки.

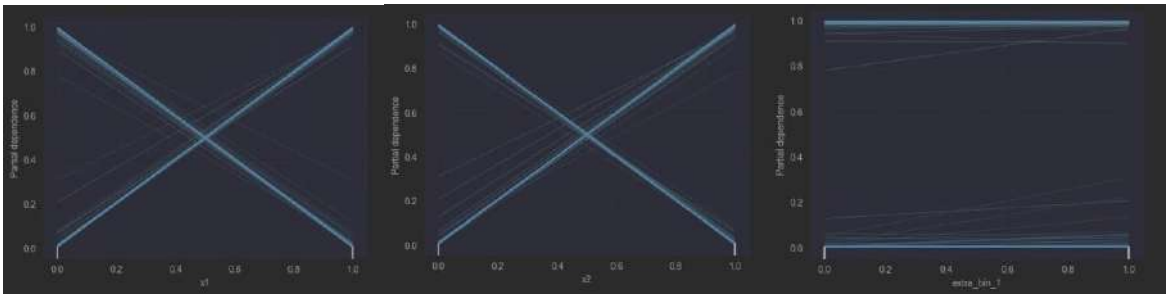
**ICE**

ICE будує одну лінію для кожного екземпляра, яка показує, як змінюється прогноз екземпляра при зміні ознаки.

Він дозволяє візуалізувати, як передбачення моделі змінюються при варіації однієї ознаки для кожного окремого спостереження.

Алгоритм:

1. Обирається одна ознака, вплив якої на прогноз потрібно дослідити. Інші ознаки залишаються зафіксованими на своїх спостережуваних значеннях.
2. Для кожного спостереження окремо змінюється обрана ознака через певний діапазон значень. При цьому інші ознаки зберігають свої початкові значення для даного прикладу.
3. Для кожного створеного варіанту обчислюється прогноз моделі. Таким чином, для кожного спостереження отримуємо послідовність прогнозів, залежну від варіації однієї ознаки.
4. Для кожного спостереження на графіку будується окрема лінія, що відображає залежність прогнозу від варіацій обраної ознаки. Це дозволяє побачити, як змінюється вплив ознаки на прогноз на рівні окремих прикладів.



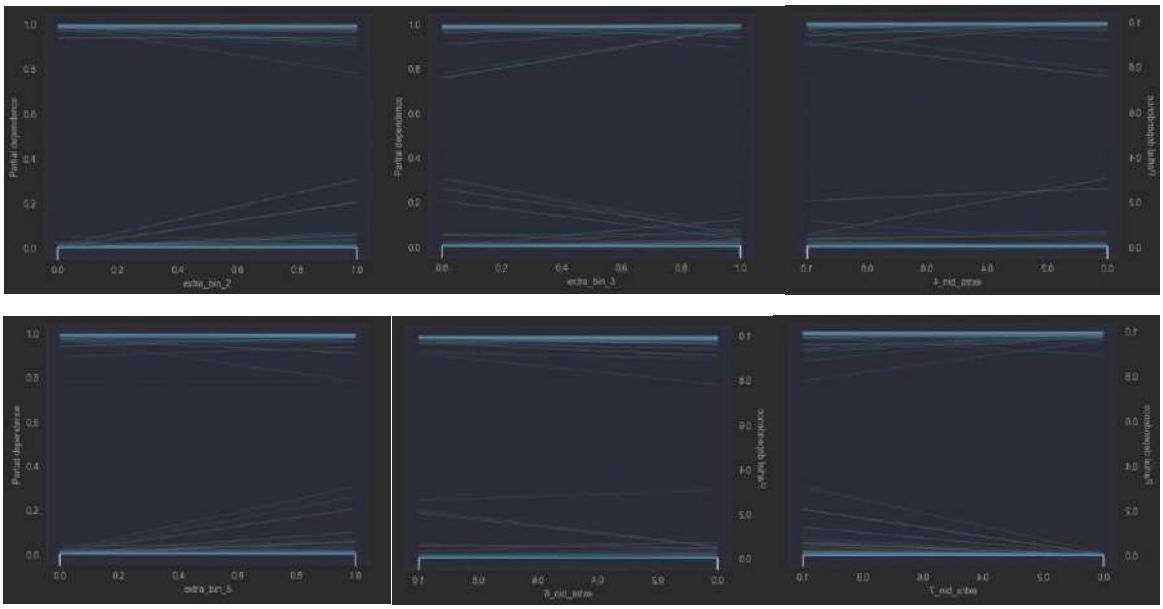


*Рисунок 3.14. Візуалізація ICE для ознак*

Як видно з графіків, шумові ознаки не впливають на вихід моделі (лінії залишаються горизонтальними – 0 або 1 на всьому діапазоні значень), значущі змінні ж у свою чергу напряму змінюють вихід моделі (рисунок 3.14).

**Partial Dependence Plot**

PDP показує середній вплив ознак на прогнозований результат моделі. Це середнє значення ліній ICE-діаграми і може показувати зв'язок між ознакою та цільовим показником.

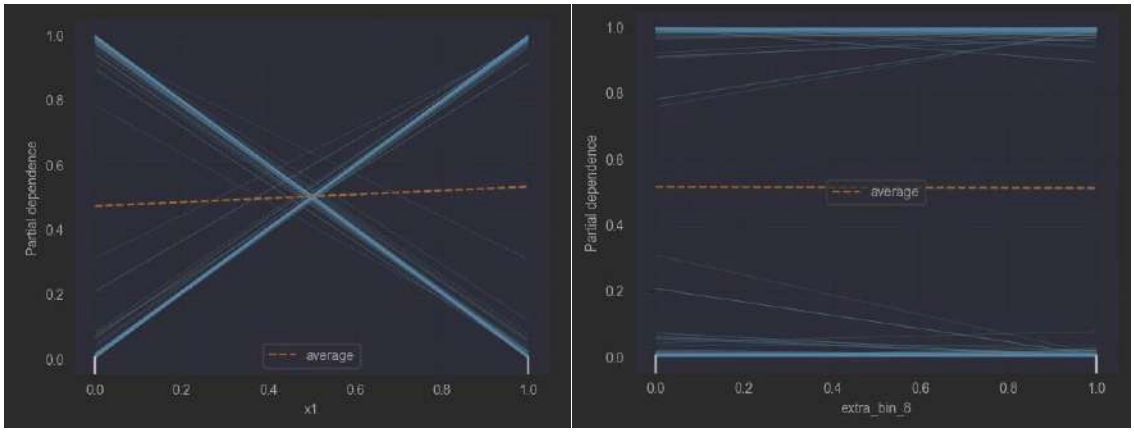



*Рисунок 3.15. Візуалізація PDP для ознак $x_1$ та extra_bin_8*

PDP для усіх ознак має однаковий вигляд – константно становить 0.5, оскільки усі лінії урівноважують одна одну (рисунок 3.15).

Також можна відцентрувати криві в певній точці функції і відобразити лише різницю в прогнозі до цієї точки – зобразити centered ICE.

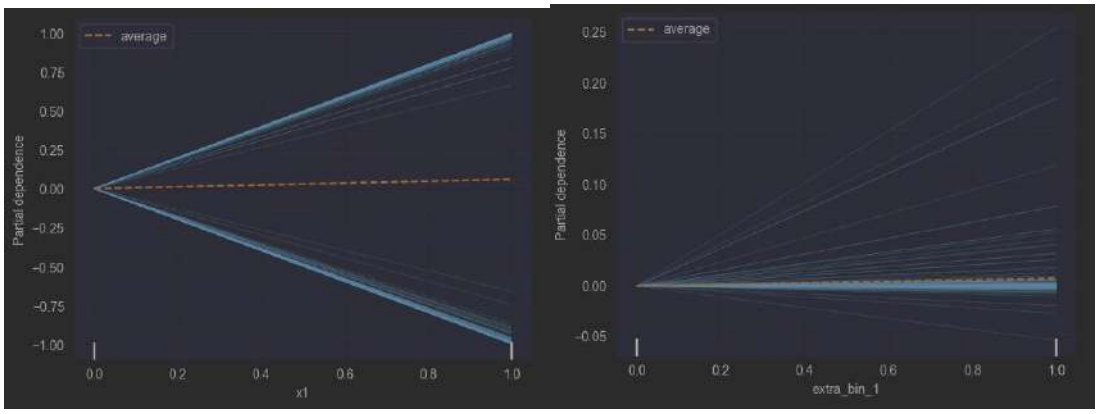



*Рисунок 3.16. Візуалізація PDP для ознак $x_1$ та extra_bin_1*

На цих графіках також чітко видно, що extra_bin змінні є паралельними, тобто не змінюють значення функції (рисунок 3.16).

**Permutation Feature Importance.**

Permutation Feature Importance базується на оцінці зміни якості роботи моделі при випадковому перемішуванні значень певної ознаки. Якщо ознака є важливою для моделі, то її перестановка повинна призвести до суттєвого погіршення показників моделі (наприклад, зниження точності або збільшення середньоквадратичної помилки). Якщо ж ознака має невеликий внесок, зміна її значень матиме незначний вплив на продуктивність моделі.

Алгоритм роботи:

1. На початковому етапі обчислюється базова продуктивність моделі на тестовій вибірці за обраною метрикою (accuracy, MSE, AUC тощо).
2. Для кожної ознаки окремо її значення перемішуються випадковим чином по всій тестовій вибірці. При цьому структура інших ознак залишається незмінною.
3. Модель знову робить передбачення на зміненій вибірці, і обчислюється нова продуктивність. Різниця між базовою оцінкою та оцінкою після перестановки служить показником важливості даної ознаки.
4. Чим більше погіршується якість роботи моделі після перестановки, тим важливішою вважається ознака для прийняття рішень моделлю.

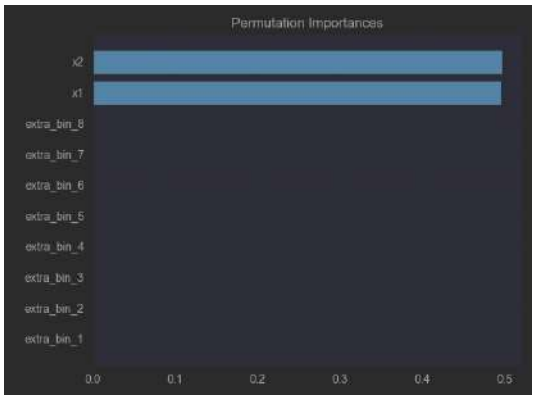



*Рисунок 3.17. Візуалізація permutation importance*

Ознаки $x_1$ та $x_2$ спричиняють найбільше падіння точності (~0.5) при перестановці, що підтверджує їхню однакову критичність. Всі шумові ознаки «extra_bin» дають майже нульовий вплив, що свідчить про те, що модель взагалі не покладається на них. Це підтверджує, що ці 2 вхідні ознаки є єдиними факторами, що визначають прогнозовану поведінку (рисунок 3.17).

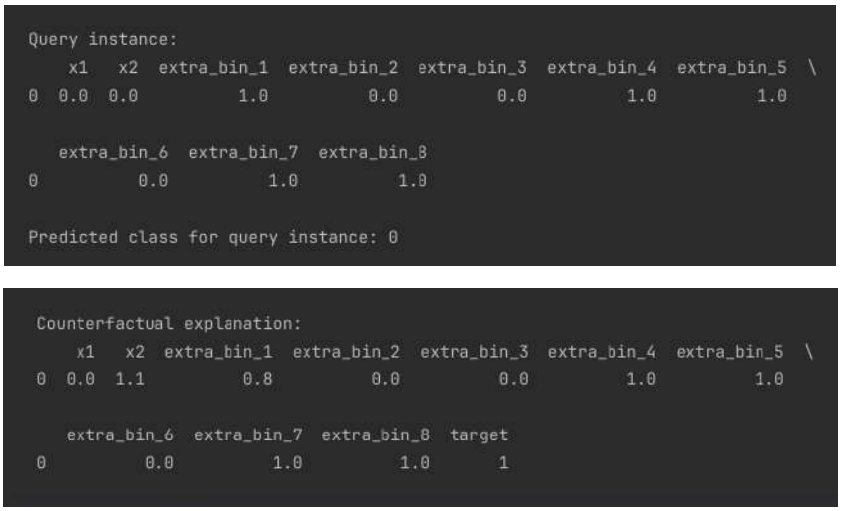
3.5. Counterfactual explanation.

Контрфактичні пояснення спрямовані на ідентифікацію мінімальних змін у вхідних даних, які призводять до іншого прогнозу моделі. Можливо створювати контрфактні приклади, змінюючи значення ознак екземпляра перед тим, як зробити новий прогноз.

Контрфактичне пояснення прогнозу описує найменшу зміну значень ознак, яка змінює прогноз на заздалегідь визначений результат.

Ми можемо використовувати алгоритм оптимізації для мінімізації втрат:

- Визначити функцію втрат.
- Ця функція втрат приймає на вході приклад, що нас цікавить, контрфактичний і бажаний (контрфактичний) результат.
- Знайти контрфактичне пояснення, яке мінімізує цю втрату, використовуючи алгоритм оптимізації.



```
Query instance:
    x1   x2  extra_bin_1  extra_bin_2  extra_bin_3  extra_bin_4  extra_bin_5  \
0  0.0  0.0          1.0          0.0          0.0          1.0          1.0

   extra_bin_6  extra_bin_7  extra_bin_8
0          0.0          1.0          1.0

Predicted class for query instance: 0
```

```
Counterfactual explanation:
    x1   x2  extra_bin_1  extra_bin_2  extra_bin_3  extra_bin_4  extra_bin_5  \
0  0.0  1.1          0.8          0.0          0.0          1.0          1.0

   extra_bin_6  extra_bin_7  extra_bin_8  target
0          0.0          1.0          1.0       1
```

*Рисунок 3.18. Counterfactual explanation*

Як видно з прикладу, для XOR задачі Counterfactual Explanations погано застосовні через дискретну природу датасету, оскільки змінені значення не входять в діапазон реальних значень. Проте навіть так це вказує, що важливою змінною є $x_2$, оскільки лише її зміни достатньо, щоб змінити результат моделі (рисунок 3.18).

3.6. PCA і t-SNE

PCA шукає ортогональні напрямки найбільшої дисперсії в даних і є лінійним методом. У задачі з XOR структура даних є нелінійною – логіка XOR створює таку конфігурацію, що класи не розділяються лінійно. Для 1-вимірного і 2-вимірного випадків немає чіткого розділення на кластери, проте для 3 вимірного існує розшарованість кластерів, що дозволяє їх візуально розділити (по осі z).

Метод t-SNE розроблено для збереження локальних відстаней між точками даних, що дозволяє виявити нелінійну структуру в даних. Він більш чутливе до локальних кластерів, навіть якщо загальна структура даних складна. Для 3 вимірного випадку чітко видно окремі кластери.

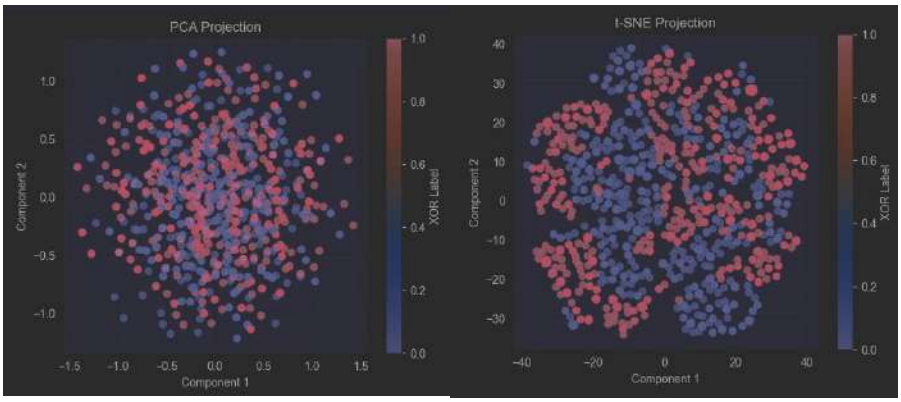



*Рисунок 3.19. 2-компонентні проекції PCA і t-SNE*

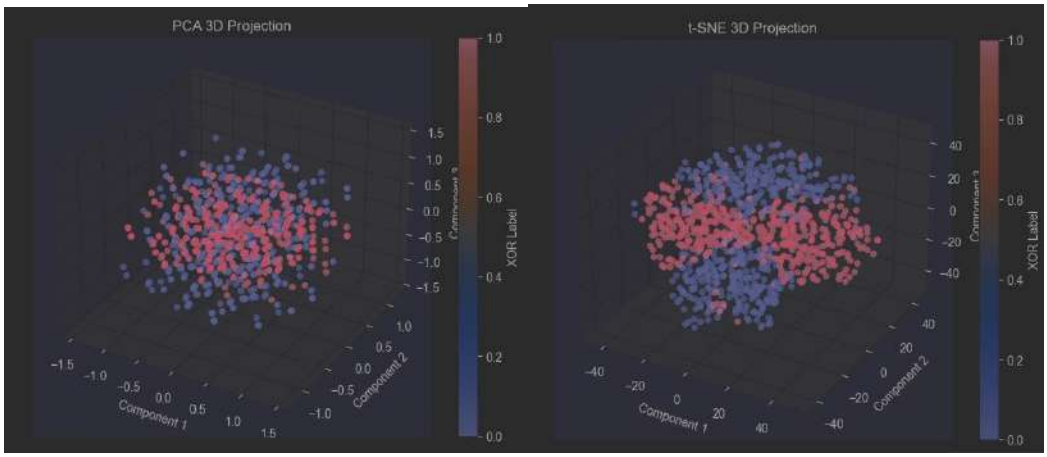



*Рисунок 3.20. 3-компонентні проекції PCA і t-SNE*

3.7. Гібридні підходи

Використано гібридні підходи на основі раніше зазначених методів. Отримано такі результати.

Використано ablation analysis - для кожної ознаки проводиться експеримент, де значення ознаки замінюються на медіану, після чого вимірюється падіння точності моделі. Це дозволяє отримати «золотий» показник важливості (чим більше падіння точності, тим важливіша ознака). Його результати збігаються з результатами гібридної моделі, яка використовує лінійну регресію для поєднання результатів з SHAP, Permutation та LIME для прогнозування нормалізованої ablation важливості.

| feature | shap | perm | lime | ablation_norm | meta_pred |
|---|---|---|---|---|---|
| x1 | 0.436787 | 0.480331 | 0.018728 | 0.466292 | 0.468938 |
| x2 | 0.447705 | 0.510870 | 0.281223 | 0.516854 | 0.514372 |
| extra_bin_1 | 0.008966 | 0.000000 | 0.142158 | 0.000000 | 0.004593 |
| extra_bin_2 | 0.010841 | 0.003106 | 0.027345 | 0.005618 | 0.003241 |
| extra_bin_3 | 0.025440 | 0.000000 | 0.178740 | 0.000000 | 0.001073 |
| extra_bin_4 | 0.003127 | 0.002588 | 0.063229 | 0.005618 | 0.006394 |
| extra_bin_5 | 0.013818 | 0.000000 | 0.053563 | 0.000000 | -0.000474 |
| extra_bin_6 | 0.022322 | 0.000000 | 0.059979 | 0.000000 | -0.002797 |
| extra_bin_7 | 0.017690 | 0.000000 | 0.050396 | 0.000000 | -0.001779 |
| extra_bin_8 | 0.013303 | 0.003106 | 0.124639 | 0.005618 | 0.006440 |

*Таблиця 3.5. Значення для кожної компоненти*

Створено інтегративну модель пояснення, яка поєднує локальні та глобальні методи оцінки важливості ознак:

SHAP – дозволяє отримати локальні оцінки впливу кожної ознаки на прогноз моделі,

Permutation Importance – глобальний підхід, який базується на зміні якості відповідей моделі при випадковому переставлянні значень ознак,

LIME – локальний метод, який генерує пояснення для окремих прогнозів,

Причинно-наслідковий аналіз (DoWhy) - для оцінки причинного ефекту кожної ознаки.

Глобальну сурогатну модель на основі дерева – для апроксимації поведінки «чорного ящика» моделі за допомогою інтерпретованої моделі (в даному випадку за допомогою дерева рішень).

Основна ідея полягає у формуванні метамоделі, що обчислює узагальнену оцінку впливу кожної ознаки за допомогою лінійної комбінації нормалізованих результатів окремих методик.

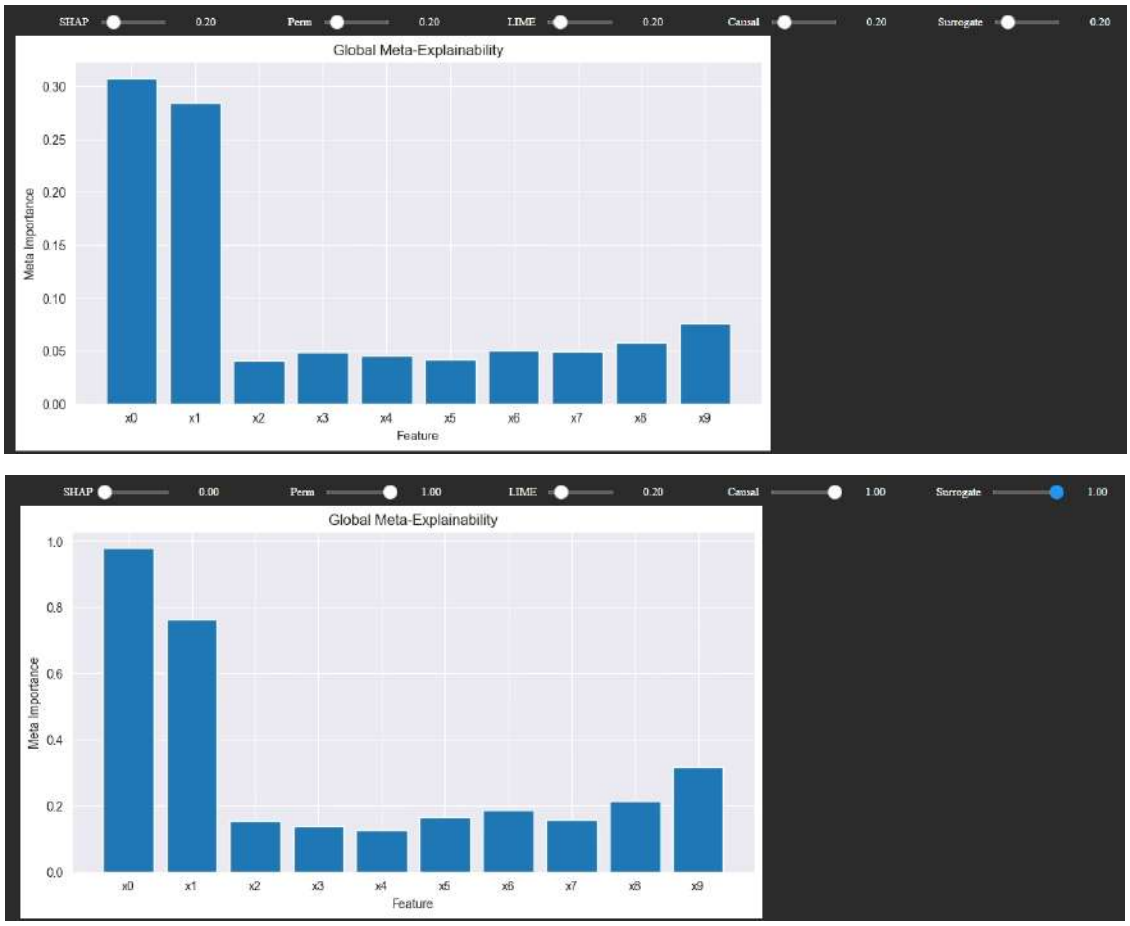



*Рисунок 3.21. Інтерактивна гібридна модель*

Щоб кількісно оцінити залежність моделі від концептів, які інтерпретуються людиною, застосовано TCAV до трьох простих бінарних концептів, отриманих на основі входів XOR.

Невелика логістична регресія (класифікатор CAV) була навчена на 20-мірних прихованих активаціях простої нейронної мережі, щоб розрізнити приклади, де кожна концепція присутня або відсутня (збалансована вибірка). Потім обчислено оцінки TCAV для кожного концепту, визначені як частка прикладів з тестового набору, для яких похідна за напрямком вздовж вектора концепту є додатною, окремо для класу 0 і для класу 1.

| Concept | TCAV (class 0) | TCAV(class 1) |
|---|---|---|
| $x_1$_is_1 | 0.4663 | 0.5024 |
| $x_2$_is_1 | 0.4663 | 0.4976 |
| $x_1$_AND_$x_2$ | 0.4819 | 0.0000 |

*Таблиця 3.6. Значення TCAV для XOR концептів*

Для концепцій з окремими ознаками ($x_1$_is_1 та $x_2$_is_1) обидва класи мають помірні значення TCAV (~0,47-0,50), що вказує на те, що межа рішення моделі приблизно однаково чутлива до кожного входу, що дорівнює 1, при прогнозуванні будь-якого з виходів.

Для об'єднаного поняття $x_1$_AND_$x_2$ оцінка для класу 0 становить ~0,48, тоді як для класу 1 вона дорівнює рівно 0,0. Це відображає логіку XOR: коли обидва входи дорівнюють 1, істинною міткою є 0. Отже, модель інтерналізувала шаблон AND як доказ лише для класу 0 (таблиця 3.6).

3.8. Порівняння методів і моделей

Натреновано 8 моделей для порівняння їх точності та інтепретованості за допомогою різних методів.

| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Decision Tree | 0.965 | 0.9444 | 0.9903 | 0.9668 |
| MLPClassifier | 1.000 | 1.000 | 1.000 | 1.000 |
| XGBoost | 1.000 | 1.000 | 1.000 | 1.000 |
| Deep MLP | 1.000 | 1.000 | 1.000 | 1.000 |
| BernoulliNB | 0.500 | 0.5152 | 0.4951 | 0.5049 |
| 1D-CNN | 0.975 | 0.9537 | 1.000 | 0.9763 |
| LSTM | 1.000 | 1.000 | 1.000 | 1.000 |
| TabTranformer | 1.000 | 1.000 | 1.000 | 1.000 |

*Таблиця 3.7. Порівняння точності моделей*

XGBoost, обидва варіанти MLP (MLPClassifier має 1 прихований шар з 64 нейронами, Deep MLP – 3 таких шари), LSTM і TabTransformer досягають 100% за кожною метрикою, що свідчить про відносно легке завдання класифікації. 1D-CNN досягає 97,5 % точності з ідеальним Recall, але з дещо нижчою точністю (0,9537), що свідчить про кілька хибних спрацьовувань. Поодиноке дерево рішень працює дуже добре (точність 96,5 %, $F_1 = 0{,}9668$), хоча дещо поступається ансамблям і глибоким мережам. Наївний байєсівський класифікатор Бернуллі (точність 50 %, $F_1 \approx 0{,}505$), що відображає його сильні

припущення про незалежність, коли він стикається з взаємозалежними ознаками (таблиця 3.7).

**Native importance**

Для дерев рішень Tree Feature Importance (FI) обчислює для кожної ознаки суму зменшень невпорядкованості (наприклад, ентропії), пов'язаних з кожним розбиттям цієї ознаки. Оскільки кожне розбиття розділяє дані і, таким чином, зменшує невпорядкованість у вузлах, підсумовування цих зменшень по дереву дає пряму оцінку того, наскільки кожна ознака впливає на загальну ефективність прогнозування. Ця величина доступна, як тільки дерево збудовано, не вимагає додаткових обчислень і забезпечує точне глобальне ранжування ознак. Його основним обмеженням є те, що він розглядає лише одновимірні розбиття - взаємодія між ознаками не враховується (що є критичним для XOR).

Поширюючи той самий принцип на ансамблі з градієнтним підсиленням, XGB FI агрегує приріст у зменшенні втрат, внесений кожною ознакою для всіх дерев в ансамблі. Хоча цей метод все ще дуже швидкий, він може дещо переоцінювати ознаки, використані в ранніх деревах, за рахунок тих, які вносять більш тонкий внесок у пізніших ітераціях. Тим не менш, як і Tree FI, він пропонує низьку вартість, точну глобальну міру важливості для будь-якої моделі з декількома деревами.

У нейронних мережах прямого поширення, а також в архітектурах на основі трансформерів, таких як TabTransformer, абсолютні значення ваг, що з'єднують кожну вхідну ознаку з першим прихованим шаром, дають швидке уявлення про глобальну важливість ознаки. Підсумовуючи або усереднюючи ці абсолютні значення ваги для кожного елемента, можна отримати миттєве ранжування без зайвих проходів вперед або назад. Недоліком є те, що видно лише залежності першого рівня; глибші нелінійні взаємодії в мережі не відображаються в цих оцінках.

Для наївних байєсівських моделей Бернуллі, які передбачають бінарні входи та умовну незалежність, можна виміряти зміну прогнозованої

ймовірності моделі при перевертанні кожної ознаки з 0 → 1 та 1 → 0. Усереднення абсолютної зміни в обох напрямках дає оцінку ΔP, яка безпосередньо кількісно показує, наскільки кожна бінарна ознака змінює вихідну ймовірність. Цей підхід використовує вбудовані в модель оцінки ймовірностей і є особливо інтуїтивно зрозумілим для дискретних або категоріальних даних, але він не узагальнюється для неперервних або сильно нелінійних моделей.

У будь-якій диференційовній моделі - наприклад, багатошарових персептронах, рекурентних мережах (LSTM), згорткових мережах (1D-CNN) або графових архітектурах (GAT) – можливо обчислити градієнт виходу (оцінку або ймовірність) щодо кожної вхідної ознаки. Якщо взяти абсолютне значення (або норму) цих градієнтів і усереднити їх по набору даних, можна отримати глобальну міру чутливості: ознаки з більшими середніми градієнтами є тими, до яких прогнози моделі є найбільш локально чутливими. Цей метод фіксує певну нелінійну поведінку, але може бути зашумленим - особливо в глибоких моделях або моделях, що базуються на увазі, - і вимагає зворотного розповсюдження через мережу для кожного входу.

| Model | Method | Runtime (s) | Comp | MeanImp |
|---|---|---|---|---|
| Decision Tree | Tree FI | 0 | 10 | 0.100 |
| MLPClassifier | InputWeightImp | 0.001 | 10 | 0.2573 |
| XGBoost | XGB FI | 0 | 10 | 0.100 |
| Deep MLP | InputWeightImp | 0 | 10 | 0.1513 |
| BernoulliNB | NB ProbDiff | 0 | 10 | 0.0344 |
| LSTM | InputGrapImp | 0.687 | 10 | 0.0098 |
| TabTranformer | InputWeightImp | 0.002 | 10 | 0.3965 |

*Таблиця 3.8. Native importance для моделей*

Comp – кількість ознак, важливість яких не дорівнює 0.

Дерево рішень (Tree FI) та XGBoost (XGB FI) мають середню важливість 0.10.

MLPClassifier показує відносно велике значення MeanImp = 0.257 для ваг першого шару, це може вказувати, що декілька вхідних даних непропорційно сильно впливають на прихований шар. Глибокий MLP, у свою чергу, має нижчий показник MeanImp = 0.151, що свідчить про те, що глибші архітектури ширше розподіляють важливість між входами.

BernoulliNB NB ProbDiff дає дуже мале значення MeanImp = 0.034, що узгоджується з припущенням про незалежність моделі - ефект перевертання (з 0 на 1 і навпаки) будь-якої окремої ознаки призводить лише до незначних змін у прогнозованій ймовірності.

Середній абсолютний градієнт LSTM становить 0.010, це може свідчити, що в середньому кожна ознака має низьку граничну чутливість.

TabTransformer - відображає змішану структуру з MeanImp = 0,397, що є найвищим показником серед усіх моделей. Це свідчить, що певні категоріальні ембединги впливають на прогнози набагато сильніше, ніж інші.

**Permutation importance**

Permutation importance - це метод діагностики моделі, який кількісно оцінює внесок кожної ознаки, вимірюючи падіння прогностичної продуктивності, коли значення цієї ознаки випадково переставляються місцями. На відміну від вбудованої (Native) важливості, вона відображає вплив ознаки на загальну поведінку моделі, фіксуючи взаємодії та нелінійності за рахунок додаткових обчислень.

| Model | Runtime (s) | Comp | MeanImp |
|---|---|---|---|
| Decision Tree | 0.1407 | 9 | 0.0926 |
| XGBoost | 1.1168 | 2 | 0.1004 |
| MLP | 0.2100 | 2 | 0.1004 |
| Deep MLP | 0.5318 | 2 | 0.1004 |
| TabTransformer | 0.7138 | 2 | 0.1010 |
| Bernoulli NB | 0.2654 | 10 | -0.0009 |
| 1D-CNN | 1.0208 | 5 | 0.0950 |

| | | | |
|---|---|---|---|
| LSTM | 1.0546 | 2 | 0.1010 |

*Таблиця 3.9. Permutation importance для моделей*

Дерева і неглибокі моделі сильно залежать від двох-дев'яти ознак - перестановка будь-якої з них знижує точність приблизно на 10%, тоді як згорткові фільтри розподіляють залежність на п'ять входів. Незначний приріст точності BernoulliNB при перестановці вказує на випадки, де модель не знайшла залежності у вхідних даних. Хоча permutation importance є досить наочною і надійною метрикою, її вартість у 0,2-1,2 с на модель може бути надто високою для великих систем або систем реального часу.

**Partial Dependence.**

PD показують для кожної ознаки середнє передбачення моделі, коли ця ознака фіксується на кожному можливому значенні, а всі інші ознаки змінюються відповідно до їх емпіричного розподілу.

| Model | Runtime (s) | Mean Range |
|---|---|---|
| Decision Tree | 0.0022 | 0.03 |
| XGBoost | 0.0386 | 0.0153 |
| MLPClassifier | 0.0024 | 0.0127 |
| Deep MLP | 0.0077 | 0.0113 |
| Bernoulli NB | 0.0061 | 0.0340 |
| 1D-CNN | 0.1133 | 0.0296 |
| LSTM | 0.1474 | 0.0112 |
| TabTransformer | 0.1293 | 0.0111 |

*Таблиця 3.10. Partial Dependence Plot метрики для моделей*

MeanRange показує, наскільки в середньому зміщується прогнозована ймовірність, коли ознака переходить від 0 до 1.

Дерево рішень і неглибокий MLP досягають майже нульової затримки (0,002 с). Ансамблеві моделі та глибокі архітектури мають вищі накладні витрати (0,04-0,21 с) через повторні прямі проходи. BernoulliNB і дерево рішень демонструють найбільші MeanRange (0,034 і 0,030), що відображає сильні одномірні залежності. Згорткові фільтри в 1D-CNN також дають значний

середній зсув (0,0296). На противагу цьому, MLP, Deep MLP, LSTM, TabTransformer демонструють більш розподілені репрезентації з меншими середніми діапазонами (0,011-0,013).

На практиці PDP забезпечують чітку візуалізацію ефектів ознак, показуючи як напрямок, так і величину, але їхня обчислювальна вартість зростає із збільшенням величиии моделі та кількістю точок сітки. Для даних високої розмірності або налаштувань, чутливих до затримок, можна зробити вибірку підмножини ключових ознак або зменшити роздільну здатність сітки, щоб збалансувати точність і час виконання.

**Surogate linear model.**

| Model | Runtime (s) | Comp | $R^2$ | RMSE | Cls. Fidelity |
|---|---|---|---|---|---|
| Decision Tree | 0.0008 | 9.9 | 0.1039 | 0.4714 | 0.6525 |
| XGBoost | 0.0224 | 10 | 0.1075 | 0.4609 | 0.6430 |
| MLP | 0.0011 | 10 | 0.1083 | 0.4649 | 0.6465 |
| Deep MLP | 0.0109 | 10 | 0.1078 | 0.4695 | 0.6505 |
| TabTransformer | 0.0544 | 9.9 | 0.1132 | 0.4674 | 0.6450 |
| Bernoulli NB | 0.0030 | 10 | 0.9984 | 0.0024 | 0.9935 |
| 1D-CNN | 0.0506 | 9.6 | 0.2494 | 0.1228 | 0.5950 |
| LSTM | 0.0570 | 9.9 | 0.1125 | 0.4605 | 0.6415 |

*Таблиця 3.11. Linear Surogate метрики для моделей*

Для нелінійних моделей (дерева, ансамблі, глибокі мережі) $R^2 = 0.1$ і fidelity = 65% показують, що лінійний сурогат охоплює лише невелику частку їх дисперсії. BernoulliNB, будучи за своєю суттю лінійною, досягає $R^2$ 1 і майже ідеальної fidelity. Сурогат для 1D-CNN показує кращі результати ($R^2 = 0.25$), але все ще залишає значну нелінійну структуру непоясненою.

| Model | Surrogate DT fidelity (depth 3) | Surrogate DT fidelity (unrestricted depth) |
|---|---|---|
| Decision Tree | 0.535 | 1 |

| | | |
|---|---|---|
| XGBoost | 0.510 | 0.965 |
| MLP | 0.510 | 0.965 |
| Deep MLP | 0.510 | 0.965 |
| Bernoulli NB | 0.770 | 0.985 |
| 1D-CNN | 0.555 | 0.96 |
| LSTM | 0.550 | 0.96 |

*Таблиця 3.12. DT surogate метрики для моделей*

Сурогатні моделі на основі дерев рішень без обмежень глибини мають fidelity більше 0.95, тобто вони майже повністю передають струкутуру моделей, у той час як невеликі дерева - лише близько 0.5.

Сурогатні моделі корисні, коли потрібні чіткі таблиці коефіцієнтів або прості правила прийняття рішень. Однак часто сурогатні моделі є лише грубим наближенням, і покладатися на його коефіцієнти слід з обережністю.

**Counterfactual explanation**

Контрфактичні пояснення шукають найменшу зміну вхідних даних, яка змінює передбачений моделлю клас. Для кожної моделі наведено середній час, необхідний для знаходження перевертання однієї ознаки, середню кількість змінених ознак та зміну передбачуваної ймовірності (ΔProb) зі стандартним відхиленням.

| Model | Runtime (s) | Avg. Feature Changed | Mean ΔProb | Std ΔProb |
|---|---|---|---|---|
| Decision Tree | 0.0007 | 1.0 | 1.000 | 0.000 |
| XGBoost | 0.0340 | 1.0 | 0.9836 | 0.0123 |
| MLP | 0.0007 | 1.0 | 0.9871 | 0.0077 |
| Deep MLP | 0.0013 | 1.0 | 0.9982 | 0.0011 |
| TabTransformer | 0.0935 | 1.0 | 0.9997 | 0.0004 |
| BernoulliNB | 0.0069 | 0.6 | 0.0283 | 0.0244 |
| 1D-CNN | 0.2004 | 1.0 | 0.2507 | 0.0514 |
| LSTM | 0.0933 | 1.0 | 0.9861 | 0.0088 |

*Таблиця 3.13. Counterfactual explanation метрики для моделей*

Дерева та більшість нейронних мереж потребують перевертання рівно однієї ознаки для інверсії рішення, що відображає лінійні або вирівняні по осях межі. BernoulliNB часто не вдається перевернути (середнє < 1), оскільки його припущення про незалежність обмежують його чутливість.

Для дерев, MLP та LSTM перевертання однієї ознаки збільшує прогнозовану ймовірність на приблизно 1.0, показуючи чіткі пороги прийняття рішень. CNN демонструє менший стрибок (≈ 0,25), оскільки його згорткові фільтри розподіляють сигнал по декількох входах.

Std ΔProb дуже низький для всіх, крім CNN і NB, що вказує на стабільні, передбачувані перевертання в моделя.

Ці результати підкреслюють, що контрфакти є швидкими і дієвими для багатьох моделей, але можуть призводити до часткових або нестабільних переворотів в архітектурах, де рішення приймаються на основі розподілених або імовірнісних внесків.

**Saliency / Gradient-Based**

Saliency будує карту важливості, обчислюючи абсолютні значення градфєнтів вихідної оціники моделі за вхідними ознаками так підсумоючи хїх вплив на прогноз. Ми обчислюємо середній час розрахунку значущості (saliency) для кожного прикладу, середню кількість ознак з ненульовим впливом, загальну суму та дисперсію зсувів ймовірності.

| Model | Runtime (s) | CompMean | Mean ΔProb | Std ΔProb |
|---|---|---|---|---|
| Decision Tree | 0.0018 | 2.1 | 2.1000 | 0.3000 |
| XGBoost | 0.0430 | 10.0 | 2.0115 | 0.0343 |
| MLPClassifier | 0.0019 | 10.0 | 2.0116 | 0.0220 |
| Deep MLP | 0.0142 | 10.0 | 2.0004 | 0.0030 |
| BernoulliNB | 0.0045 | 10.0 | 0.3406 | 0.0036 |
| 1D-CNN | 0.5080 | 10.0 | 0.7325 | 0.0896 |
| LSTM | 0.5161 | 10.0 | 1.9797 | 0.0090 |

| | | | | |
|---|---|---|---|---|
| TabTransform | 0.5187 | 10.0 | 2.0001 | 0.0007 |

*Таблиця 3.14. Saliency explanation метрики для моделей*

Цей метод є дуже швидким, глибокі та рекурентні моделі займають ≈0,5 с через повторні прямі проходження, інші ж – соті долі секунди.

Нейронні мережі та ансамблі позначають всі вхідні дані як суттєві, у той час як дерева лише ~2. Для усіх моделей притаманна низька дисперсія (< 0,09), що вказує на стабільні ознаки в більшості моделей.

**Interaction Strength (Середня H-статистика)**

H-статистика вимірює ступінь парної взаємодії між ознаками, де 0 вказує на суто адитивні ефекти, а значення, близькі до 1, вказують на сильну взаємодію. Для кожної моделі наведено час, необхідний для обчислення середньої H-статистики, та її результуюче значення.

| Model | Runtime (s) | Comp (pairs) | Mean H |
|---|---|---|---|
| Decision Tree | 0.5461 | 45 | 0.0571 |
| XGBoost | 3.2735 | 45 | 0.0354 |
| MLPClassifier | 0.6333 | 45 | 0.0327 |

*Таблиця 3.13. Interaction Strength метрики для моделей*

На кожній моделі обчислювались 45 пар ознак.

Mean H – це середня Interaction Strength на парах ознак.

Усі моделі демонструють низьку взаємодію ознак (Mean H < 0.06). Дерево рішень демонструє найвищу взаємодію. Через те, що оцінка проводиться для багатьох пар ознак, H-статистика є відносно повільною - особливо для XGBoost - і її слід використовувати у випадках, коли виявлення взаємодій між ознаками є критично важливим.

# РОЗДІЛ 4. MNIST

## 4.1. Лінійні моделі

Дотримуючись тієї ж схеми, що і для набору даних XOR, використано Linear, Ridge і Lasso регресії та метрики MSE та $R^2$.

| Model | MSE | $R^2$ |
|---|---|---|
| Linear | $9.71 * 10^{14}$ | $-1.16 * 10^{14}$ |
| Ridge ($\alpha = 1.0$) | 3.15 | 0.62 |
| Lasso ($\alpha = 0.1$) | 4.59 | 0.45 |

*Таблиця 4.1. Метрики для Regression моделей*

Для Linear regression отримано надзвичайно великі значення помилки, що вказує на гіршу роботу моделі у порівнянні з тривіальним прогнозом. Коефіцієнти мають дуже великі абсолютні значення, що вказує на перенавчання та нестабільність без регуляризації. На візуалізації майже вся карта коефіцієнтів виглядає однорідно, оскільки масштаб надто великий.

$L_2$-регуляризація допомогла стабілізувати модель і це суттєво знизило похибку. Покращило $R^2$ до 0.62, це означає, що модель пояснює близько 62% варіації цільової змінної.  На тепловій карті коефіцієнтів можна виділити певну структуру – виділяються зони, де колір пікселя впливає на результат класифікації.

$L_1$-регуляризація зводить до 0 значну кількість коефіцієнтів, що робить модель розрідженою. Це полегшує інтерпретацію (менше ознак із ненульовими вагами), але дещо погіршує точність порівняно з Ridge.

На мапі коефіцієнтів видно менше пікселів, які впливають на результат, проте вони формують певні кластери в місцях, що найбільше впливають на визначення цифр.

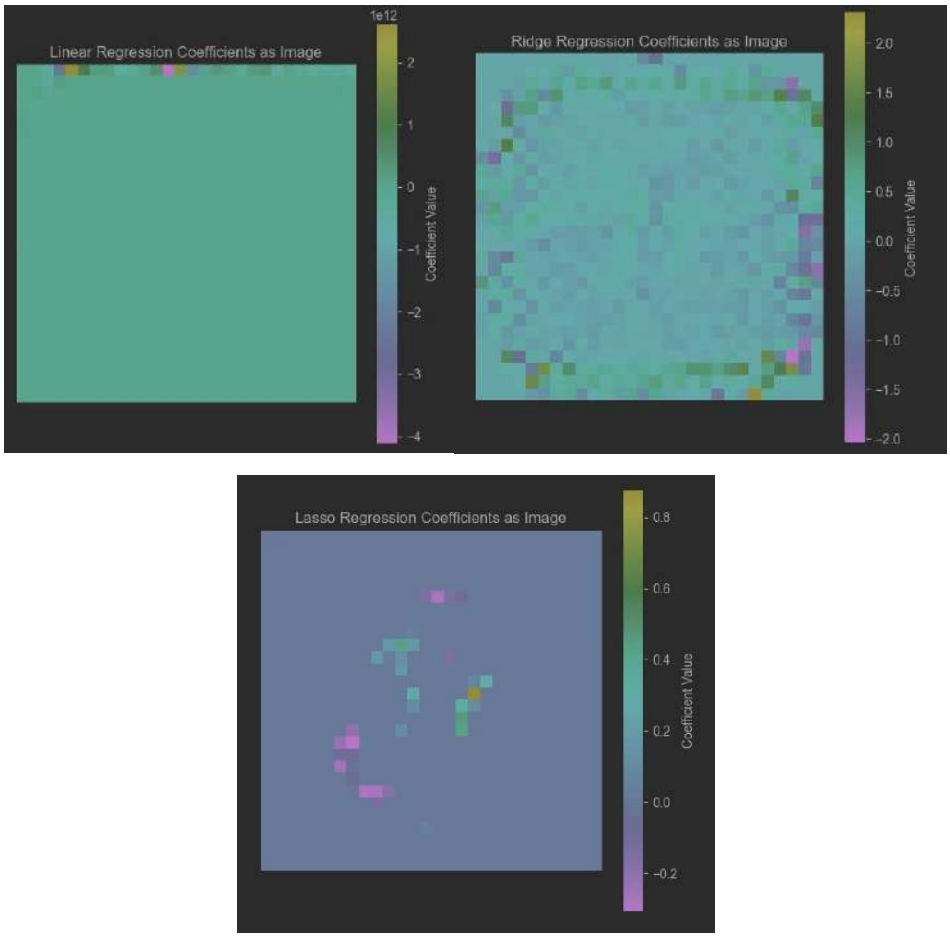



*Рисунок 4.1. Візуалізації коефіцієнтів для Regression моделей*

Як і для XOR датасету, було використано Lasso Regularization Path. Нижче наведено графіки ознак, які формують перші декілька рядків зображення (починаючи з верхнього лівого кутка). Вивід порядковий (всього таких рядків 28).

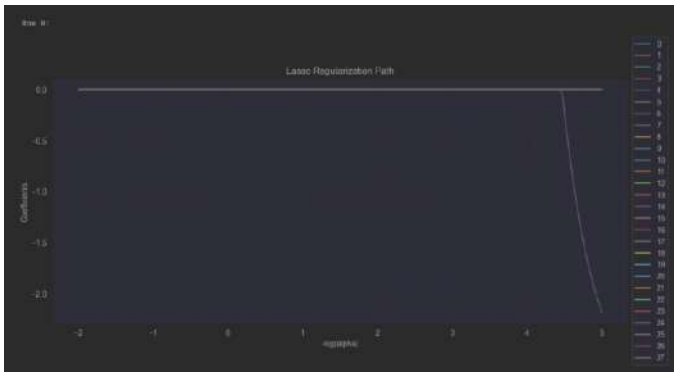


*Рисунок 4.2. Візуалізації Lasso Regularization Path*

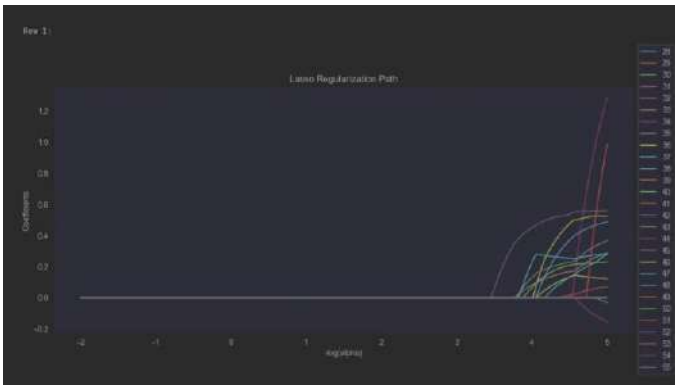



*Рисунок 4.3. Візуалізації Lasso Regularization Path*

Графіки демонструють, як зменшення сили регуляризації призводить до збільшення кількості активних коефіцієнтів, що відображає компроміс між точністю моделі та її інтерпретованістю. Також графіки демонструють значно складнішу природу даних у цьому випадку (криві можуть змінювати напрямок при зміні параметру регуляризації, чого не спостерігалось для XOR датасету).

GLM очікувано не дала хорошого результату:

MSE: 19.444

$R^2$: -1.3189

Отримані коефіцієнти мають великий розкид. Для кращої інтерпретації значення обмежено діапазоном [-0.1, 0.1].

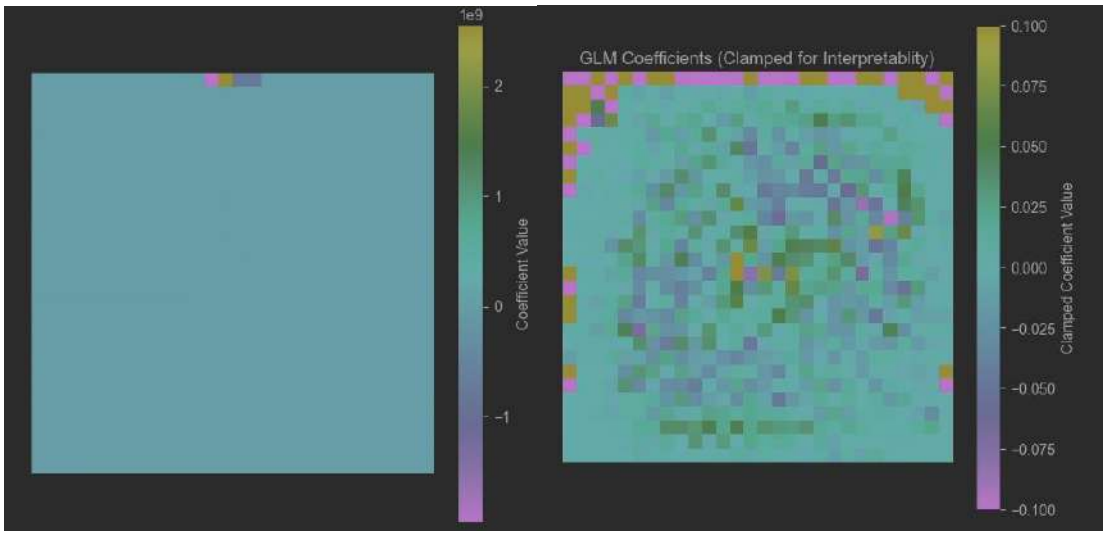



*Рисунок 4.4. Візуалізації коефіцієнтів GLM*
*(оригінальні і в діапазоні [-0.1, 0.1])*

На отриманій карті можна побачити, що важливіші пікселі зосереджені по центру зображення, оскільки у MNIST датасеті цифри на зображенні переважно центровані.

GAM з окремим сплайном для кожної змінної є дуже громіздким і важким для інтерпретації. Тому було використано PCA з 10 компонентами для зменшення кількості вхідних даних. Використано модель LinearGAM із 10-ма складовими. Кожна складова відповідає окремому головному компоненту PCA.

MSE = 3.9139 — середньоквадратична похибка помітно зменшилася порівняно з попередньою моделлю.

$R^2$ = 0.5332— понад 53% варіації цільової змінної пояснюється моделлю.

Effective DoF: 110.4 — сумарна гнучкість моделі. Кожен сплайн має в середньому 9–13 ступенів свободи, що дозволяє моделі виявляти досить складні нелінійні патерни.

Кожен терм s(i) статистично значущий ($p < 1.11 * 10^{-16}$), отже, всі PCA-компоненти мають внесок у передбачення.

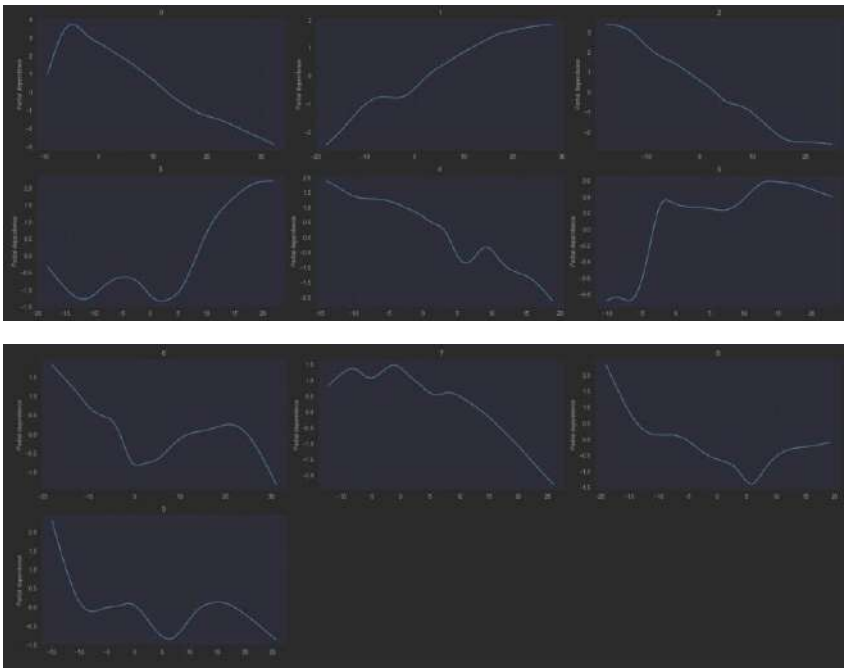


*Рисунок 4.5. Візуалізації сплайнів GAM*

На графіках показано, як змінюється передбачення моделі при варіації окремого компоненту PCA, коли інші компоненти зафіксовані.

Більшість кривих має виражену нелінійну форму (з горбами, падіннями тощо), свідчачи, що модель уловлює складні залежності між трансформованими ознаками (PCA) та цільовою змінною.

Оскільки PCA-компоненти — це лінійні комбінації вихідних пікселів (784), інтерпретувати їх «по пікселях» напряму складно. Зате з погляду моделі важливо, що кожний із 10 головних напрямків має суттєвий внесок і нелінійний вплив.

4.2. Дерева рішень.

Дерева рішень дають непоганий результат – отримано точність у 88% (з використанням DecisionTreeClassifier без обмежень глибини). При зміні максимальної глибини дерева (від 2 до 9) точність класифікації зростає від ~34 % до ~85 %, де виходить на плато.

Для візуалізації побудовано карти та зображення дерев з обженою глибиною (10) і без. Отримано такі результати:

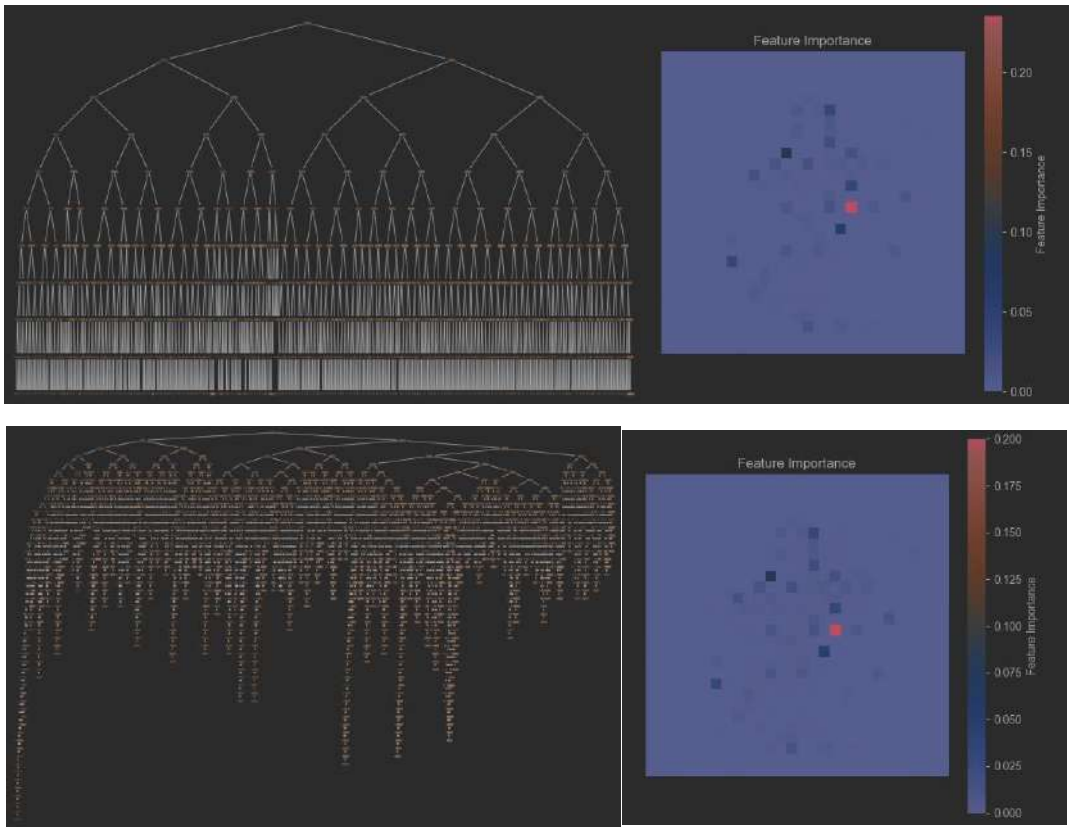



*Рисунок 4.6. Візуалізації дерев рішень і важливостей їх ознак*

Точність у обох випадках ~85% і карти важливості ознак дуже схожі, проте структура самих дерев кардинально відрязняється.

На натренованих деревах виділено правила розділення ознак. І аналогічно до XOR, для перевірки правильності отриманих правил та їх точності, синтезовано новий набір даних, вибравши по 20 прикладів на кожне правило. Так перевіримо його на MLP (1 прихований шар з 100 нейронами).

Досліджено, як глибина дерева та попередня обробка пікселів впливають на продуктивність моделі, складність правил та корисність синтетичних наборів даних, згенерованих за цими правилами. В усіх експериментах використовується розподіл повного набору даних MNIST 80/20, пікселі або розглядаються як неперервні значення в діапазоні [0,1], або бінаризуються з порогом 0.5.

Зі збільшенням глибини дерева класифікатор Decision-Tree (DT) досягає вищої точності як навчання, так і тестування, проте ціною значного збільшення кількості правил.

На всіх налаштуваннях MLP досягає майже ідеальної продуктивності на вихідних навчальних даних (100 %) і сильного узагальнення на тестовому наборі (97.8 % неперервних, 97.1 % двійкових даних). Така стабільність підкреслює, що MLP є високопродуктивною моделлю, здатною вивчити всю границю рішення MNIST незалежно від попередньої обробки.

Найглибші дерева дають великі синтетичні набори (оскільки при збільшенні дерева, збільшується і кількість правил, від якої залежить розмір набору даних), але точність MLP на цих синтетичних прикладах залишається низькою (12-17%), що набагато нижче тестової точності на реальних даних. Неглибокі дерева дають найменші синтетичні набори і найвищу синтетичну точність (17,11% для неперервних даних проти 10,47% для бінарних). Це свідчить про те, що надто складні набори правил ще гірше узагальнюють синтезовані вхідні дані.

Щодо попередньої обробки, то безперервна перевершує двійкове кодування на 1-8% у точності тесту дерева рішень і приблизно на 0,6% у точності тесту MLP.

Правила на безперервних даних дають дещо вищу продуктивність MLP на синтетичних вибірках на малих глибинах (17,1% проти 10,5%), але розрив зникає на деревах з більшою глибиною (12,62% проти 12,65% при необмеженій глибині), що відображає зростаючу надлишковість правил в обох кодуваннях.

Довжина правил зростає від 6 умов при глибині рівній 6 до >15 умов при необмеженій глибині, в результаті чого кожне правило визначає дуже обмежену область простору ознак.

Зі збільшенням кількості правил та складності умов об'єднання їхніх інтервалів не може адекватно покрити всю множину даних, що призводить до синтетичних вибірок, які MLP класифікує погано.

| | Depth=6 | Depth=10 | Depth=unlimited |
|---|---|---|---|
| DT train | 75.88% | 91.64% | 100.00% |
| DT test | 75.14% | 86.68% | 88.03% |
| MLP train | 100.00% | 100.00% | 100.00% |
| MLP test | 97.79% | 97.79% | 97.79% |
| Rules | 64 | 903 | 3159 |
| MLP on synthetic | 17.11% | 14.39% | 12.62% |
| Synthetic size | 1280 | 18060 | 63180 |

*Таблиця 4.2. Результати оцінки синтетичного набору на безперервних даних*

| | Depth=6 | Depth=10 | Depth=unlimited |
|---|---|---|---|
| DT train | 68.92% | 86.74% | 100.00% |
| DT test | 68.36% | 83.38 | 86.96% |
| MLP train | 100.00% | 100.00% | 100.00% |
| MLP test | 97.14% | 97.14% | 97.14% |
| Rules | 64 | 928 | 4547 |
| MLP on synthetic | 10.47% | 12.91% | 12.65% |
| Synthetic size | 1280 | 18560 | 90940 |

*Таблиця 4.3. Результати оцінки синтетичного набору на бінаризованих даних*

Точність на синтетичних даних є значно меншою від точності на реальних даних і цьому є декілька причин.

Рівномірно дискретизуючи інтервали (або фіксуючи біти у двійковому випадку), ігноруються справжні кореляції між пікселями, які характеризують рукописні цифри. В результаті, синтетичні вхідні дані часто лежать далеко від будь-якої множини реальних цифр.

Неглибокі дерева дають мало правил, які разом покривають лише частину реальних даних; більш глибокі дерева фрагментують простір на тисячі крихітних областей, багато з яких можуть перетинатися або залишати великі прогалини. В обох випадках синтезований набір не відображає щільність і різноманітність реальних прикладів.

Дерева з великою глибиною є надмірно пристосованими. Дерева необмеженої глибини виділяють дуже специфічні області (15+ тестів ознак на правило). Таке перенавчання означає, що кожне правило відповідає лише невеликій кількості реальних зразків - і коли ми вибираємо нові точки в цій області, вони більше не схожі на оригінали. І цьому не допомагає велика кількість прикладів, оскільки усі вони можуть бути дуже далекими від реальних.

Також синтетичний набір не додає жодного шуму чи змішування між регіонами, тому MLP бачить лише жорсткі патерни, з якими він не стикався під час навчання, що погіршує його здатність до узагальнення.

Також на результат може вплинути те, як генеруються дані. Однакова кількість синтетичних точок створюється для кожного правила, незалежно від того, скільки реальних прикладів потрапило в цей листок. Це може призвести до надмірної вибірки рідкісних або тривіальних регіонів і недостатньої вибірки найбільш репрезентативних, викривляючи пропорції класів і межі рішень.

Для покращення результатів було використано ансамбль невеликих дерев та балансування за розміром, і змінено вибірку в межах кожного листка. Це привело до підвищення точності до 45%.

Десять дерев глибиною 4 разом розбивають простір на багато листків середнього розміру - таким чином, синтетичні точки охоплюють більшу частину істинного розмаїття цифр, не перекриваючи крихітні регіони.

Гауссові зразки зберігають кореляції пікселів у листку (штрихи, петлі, відтінки). А також mixup змішує дві реальні цифри з додаванням невеликого випадкового збурення, створюючи правдоподібні проміжні варіанти.

Викидні області (з менше ніж 50 реальних зразків) ігноруються і для кожного листка генерується саме стільки синтетичних прикладів, скільки в ньому реальних, зберігаючи пропорції класів.

Разом це гарантує, що синтетичні приклади MNIST виглядають і розподіляються набагато ближче до оригіналів - так що MLP може правильно їх розпізнати, підвищуючи точність синтетичних даних з ~15% до ~45%.

| | |
|---|---|
| Rules | 56000 |
| MLP on systhetic data | 45.04% |

*Таблиця 4.4. Результати оцінки покращеного набору*

Щоб оцінити, чи може мережа навчатися виключно на вилучених з дерева областях, MLP навчалось лише на синтетичних прикладах і оцінювали його на реальному тестовому наборі MNIST, тобто провели зворотній до попереднього експеримент.

| | Depth=6 | Depth=10 | Depth=unlimited |
|---|---|---|---|
| Rules | 64 | 903 | 3159 |
| Samples | 1280 | 18060 | 63180 |
| MLP train | 28.23% | 34.62% | 48.82% |
| MLP test | 26.96% | 34.31% | 48.14% |

*Таблиця 4.5. Результати оцінки навчання на синтетичному наборі*

Навіть з більш ніж 63 000 синтетичних прикладів з дерева необмеженої глибини точність тесту залишається нижчою за 50 %. Більш того, точність навчання і тестування майже однакова на кожній глибині, що свідчить про навчання MLP на синтетичних шаблонах, які узагальнюють однаково (але

погано) реальні цифри. При використанні ансамблю з 10 дерев отримано трохи кращі результати.

| | Depth=6 | Depth=10 | Depth=unlimited |
|---|---|---|---|
| Rules | 640 | 8261 | 47075 |
| Samples | 12800 | 165220 | 941500 |
| MLP train | 46.79% | 43.86% | 53.00% |
| MLP test | 46.90% | 43.71% | 52.89% |

*Таблиця 4.6. Результати оцінки навчання на синтетичному наборі*

Для порівняння реалізовано та навчено спектрально-нормалізований AC-GAN із Hinge втратою для генерації якісніших синтетичних цифр. Генератор приймає 100-вимірний вектор шуму з ембедингом мітки класу, і проектує його через повнозв'язний шар у тензор розміру 127 × 7 × 7. Цей тензор поступово збільшується розміром двома блоками транспонованих згорток (кожний з BatchNorm) до зображення 1 × 28 × 28.

Дискримінатор складається з 3 згорткових шарів, які зменшують вхідне зображення до розміру 256 × 4 × 4. Цей тензор вирівнюється і пропускається через спільний повнозв'язний шар. З отриманого представлення отримуємо 2 гілки:

Гілка адвесаріальної голови – навчається за Hinge втратою для розрізнення реальних і синтетичних зразків.

Гілка класифікації – навчається за крос-ентропійною втратою для прогнозування міток.

Після навчання порівняно роботу і якість генерації з допомогою MLP (один прихований шар з 200 нейронами):

| Train → Test | Accuracy |
|---|---|
| Real → Real | 98.14% |
| Real → Synthetic | 98.50% |
| Synthetic → Synthetic | 100% |
| Synthetic → Real | 86.47% |

| | |
|---|---|
| Real + Synthetic → Real | 97.68% |

*Таблиця 4.7. Результати точності навчання і тестування на синтетичному датасеті з викоирстанням AC-GAN*

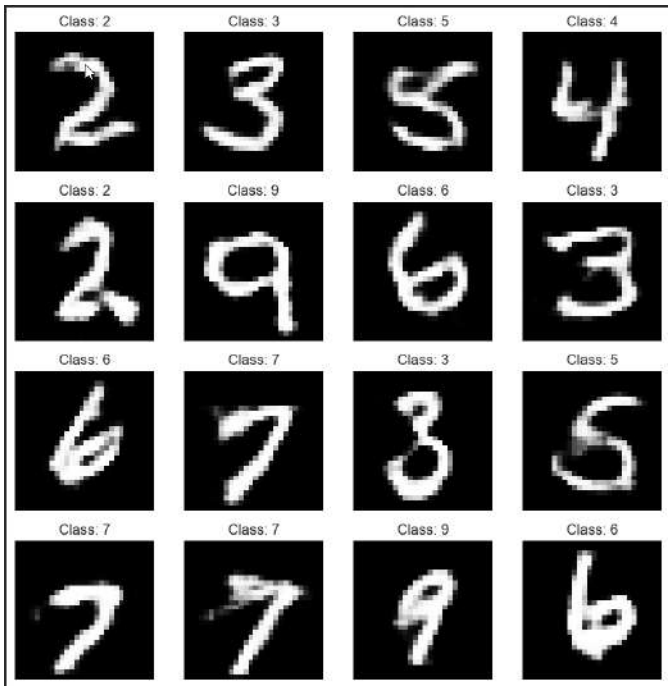



*Рисунок 4.7. Візуалізація згенерованих цифр*

Датасет згенерований з допомогою GAN значно реалістичніша за rule-based синтетику. Додавання синтетики до реальних даних майже не погіршує якість (97.7 % і 98.14 %), що свідчить про високу відповідність GAN-генерацій розподілу MNIST.

GAN-синтетика дозволяє створювати контрольовані, але візуально правдоподібні приклади для тестування explainability-методів. Якщо XAI-метод стабільний на GAN-синтетиці й узагальнюється на реальні дані, це підтверджує його достовірність і точність пояснень.

**RuleFit**

Після навчання RuleFitRegressor виводить перелік лінійних умов-правил із відповідними ваговими коефіцієнтами на кшталт:

*X291 > 0.0098 and X510 <= 0.4902,0.05*

Кожне таке правило у межах моделі робить додатний або від’ємний внесок у фінальний прогноз залежно від знака і величини коефіцієнта.

Для кожної ознаки, яка входить в правило, додається вага правила, Отримане зображення можна порівняти з мапою вагових коефіцієнтів для Lasso Regression.

Після навчання отримано MSE = 3.73.

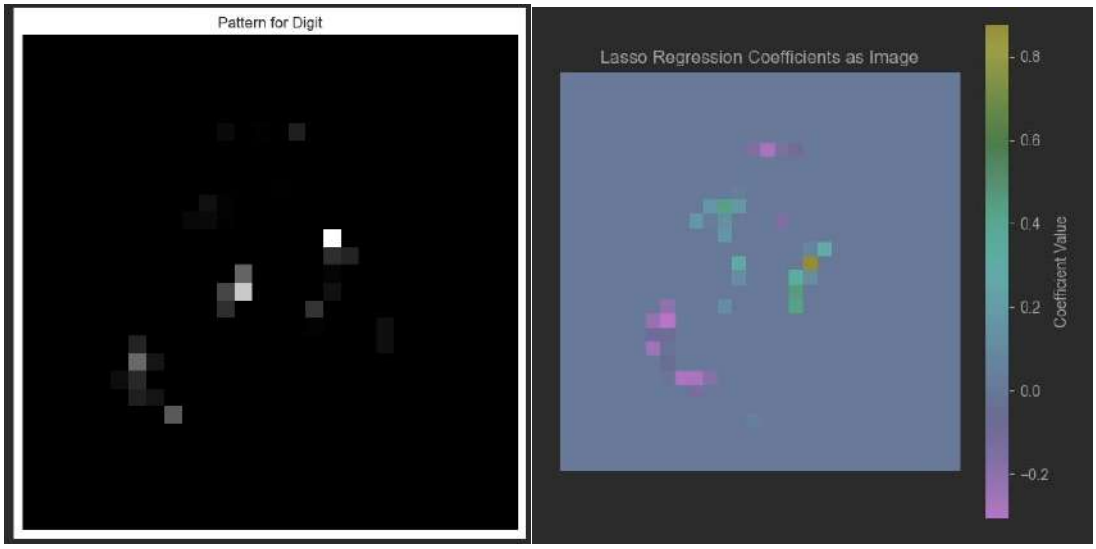



*Рисунок 4.8. Візуалізації Rulefit і порівняння з візуалізацією Lasso*

Основна причина схожості полягає в тому, що RuleFit все ще використовує той самий розв'язувач Lasso для оцінки ознак. Він включає вихідні інтенсивності пікселів разом з усіма похідними правилами. Коли більшість видобутих правил є просто пороговими значеннями для окремих пікселів і їх поєднанням, ці правила, поводяться так само, як трохи більш зашумлені версії самих вихідних пікселів. Штраф Lasso розглядає обидва типи ознак однаково, тому він, як правило, вибирає той самий набір інформативних пікселів - незалежно від того, чи є вони в вигляді пікселів, чи об'єднані в піксельні правила.

Оскільки використані дерева були неглибокими, було дуже мало правил, які об'єднували кілька пікселів у значущі взаємодії. В результаті коефіцієнти правил здебільшого зводилися до тих самих точок, які обирає Lasso.

Схожі результати отримано при використанні RuleFit з gradient boosting. У цьому випадку Mean Squared Error становив 2.88.

4.3. Perturbation методи

**SHAP**

Використано MLP модель з 1 прихованим шаром і 100 нейронами для оцінки цього методу. Отримана точність на тестовій вибірці складає 96.44%.

Побудовано візуалізацію з колірним кодуванням, де червоним позначено пікселі, які позитивно впливають на прогноз для певного класу (в даному випадку для класу 3). Аналіз такої візуалізації дозволяє побачити, які регіони в зображенні найбільш типові для певної цифри.

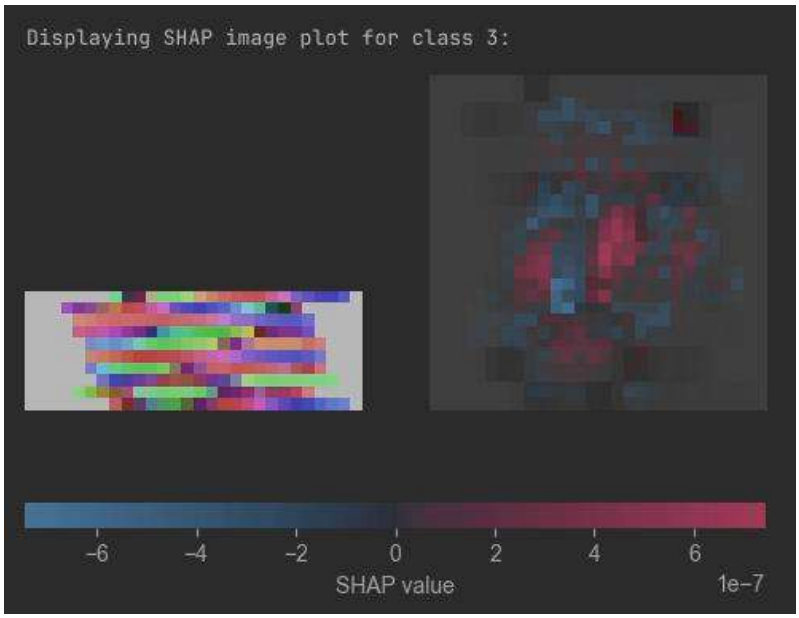



*Рисунок 4.9. Віалізація SHAP з колірним кодуванням класу 3*

**PDP**

Як і для XOR датасету використано бібліотечну реалізацію для PDP. Було побудовано графік залежності прогнозу (для цільового класу 1) від значень пікселів.

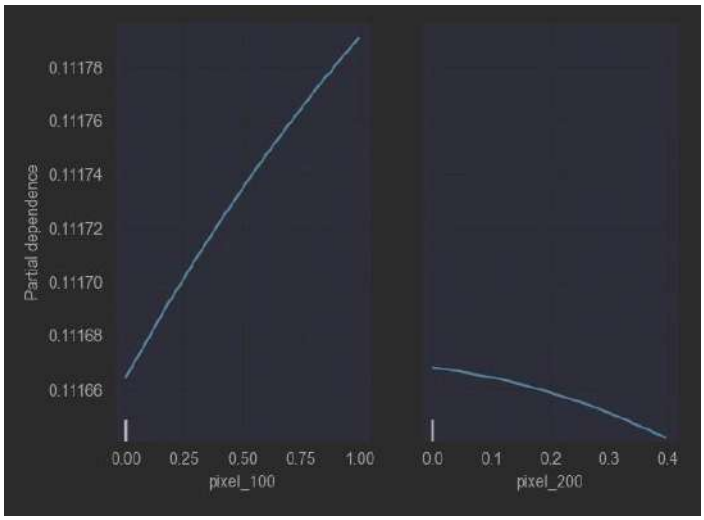



*Рисунок 4.10. Графік залежності прогнозу від значень пікселів 100 та 200*

Аналогічно, цей метод було застосовано до усіх пікселів на зображені:

1) Існують ознаки, які не впливають на визначення класу зображення. Зазвичай це пікселі, які розташовані на краю зображення, оскільки у MNIST датасеті цифри переважно центровані.

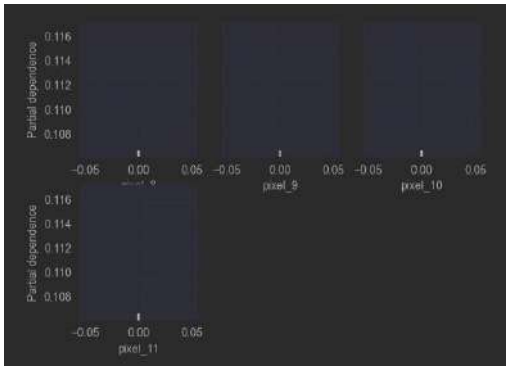



*Рисунок 4.11. Графіки PDP для пікселів 8-11*

2) Інші ознаки мають або лінійний характер, або є поліномом невеликого ступеню.

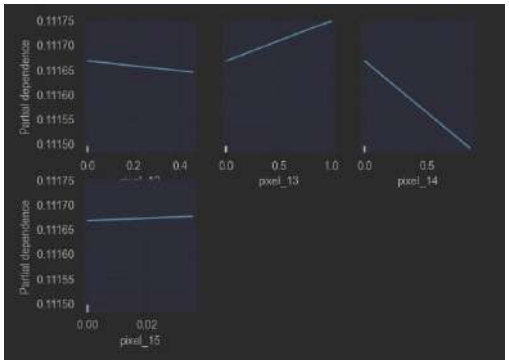



*Рисунок 4.12. Графіки PDP для пікселів 12-15*

3) Всі ознаки мають невеликий вплив при зміні значення, а також можуть впливати не на всьому діапазоні.

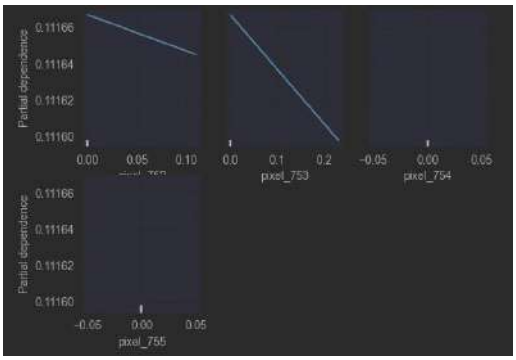



*Рисунок 4.13. Графіки PDP для пікселів 752-755*

Також використано Partial Dependence Plots для побудови графіків залежності ймовірності класу від інтенсивності конкретного пікселя. Нижче наведено такі графіки для пікселя 100 і класів 0, 1 і 2.

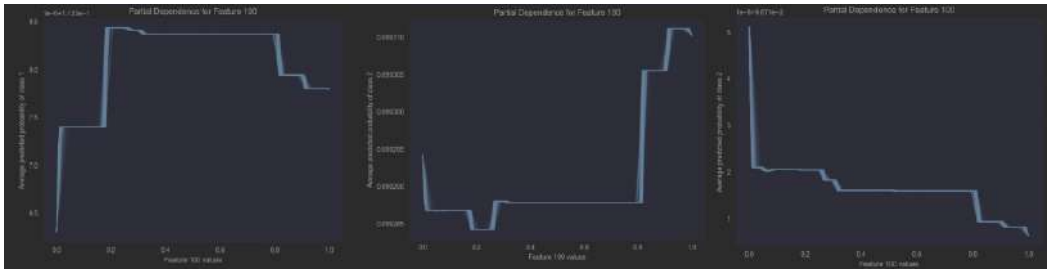


*Рисунок 4.14. Графіки залежності ймовірності класу від інтенсивності пікселя 100*

XGBoost модель, яка використана для перевірки цього методу (точність 98%), приймає рішення за допомогою порогових умов. При проходженні через поріг прогноз різко змінюється, що відображається як сходинки. Фіксуючи інші ознаки, зміна лише однієї може призвести до переключення гілок дерева у багатьох прикладах, що дає різкі зміни середнього прогнозу.

Ці ж графіки підтверджують, що ознаки, які знаходяться на границі зображення слабко або взагалі не впливають на прогнозування класу.

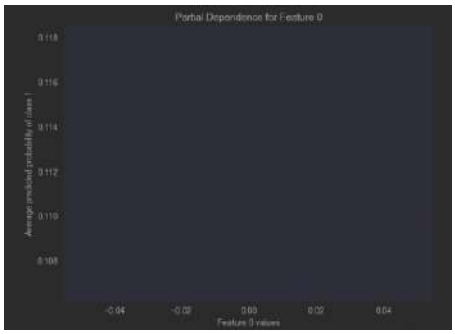



*Рисунок 4.15. Графіки залежності ймовірності класу від інтенсивності пікселя 0*

**ICE**

Більшість ліній на графіках розташовані горизонтально біля 0 чи 1. Якщо для певного прикладу модель практично видає результат, що це зображення має клас 1 або його не має (наприклад, ймовірність = 0.99 або 0.01), то при зміні одного пікселя (з 0 до 1) результат може лишатися майже незмінним. Тобто, для цього прикладу інтенсивність конкретного пікселя майже не впливає на рішення щодо класу 1.

У деяких випадках (близько, 1–2% ліній) спостерігається не зовсім пласка, а трохи похила чи зламана лінія. Це означає, що для конкретного прикладу по мірі збільшення значення пікселя модель змінює ймовірності класів. Чим більша амплітуда нахилу, тим сильніший вплив цього пікселя на рішення для того прикладу. Проте, якщо таких ліній невелика кількість серед усіх зазначених, ця ознака не є критичною.

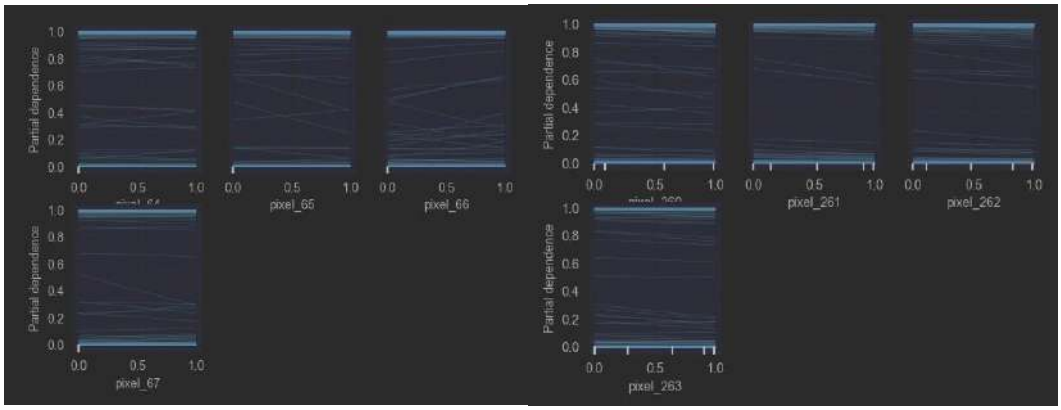



*Рисунок 4.16. Приклади графіків ICE для різних пікселів*

**Permutation importance**

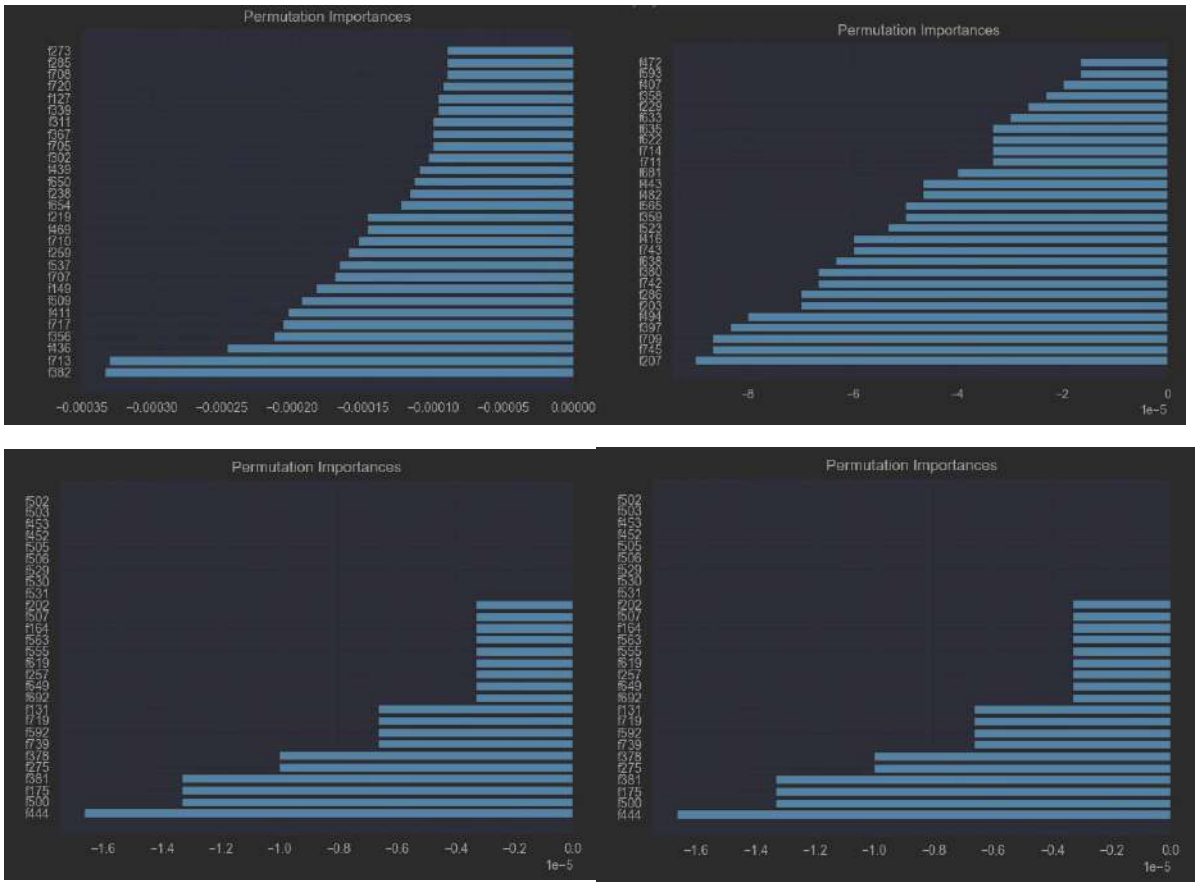



*Рисунок 4.17. Permutation importance*

Якщо після перемішування деяких пікселів показник точності моделі навіть зростає (або не змінюється) — це дає від’ємну або близьку до нуля важливість. Такі пікселі можуть бути фактично не використані моделлю (або містити шум), і їхня випадкова перестановка не шкодить роботі алгоритму.

Негативне значення не означає, що піксель зайвий: радше говорить про те, що в контексті XGBoost даний піксель не покращує передбачення — а інколи його вимкнення навіть знімає шум і трохи підвищує точність.

4.4. Counterfactual explanation

Натреновано NN-модель з точністю 98%. Цю нейромережу було використано для наступних методів.

Для генерації контрафактичних прикладів було використано бібліотеку Dice ML.

Нижче наведено приклад зміни пікселів для перетворення початкового зображення цифри 7 і перетвореного, яке класифікується як 0, 1, 2, 3 та 4 відповідно:

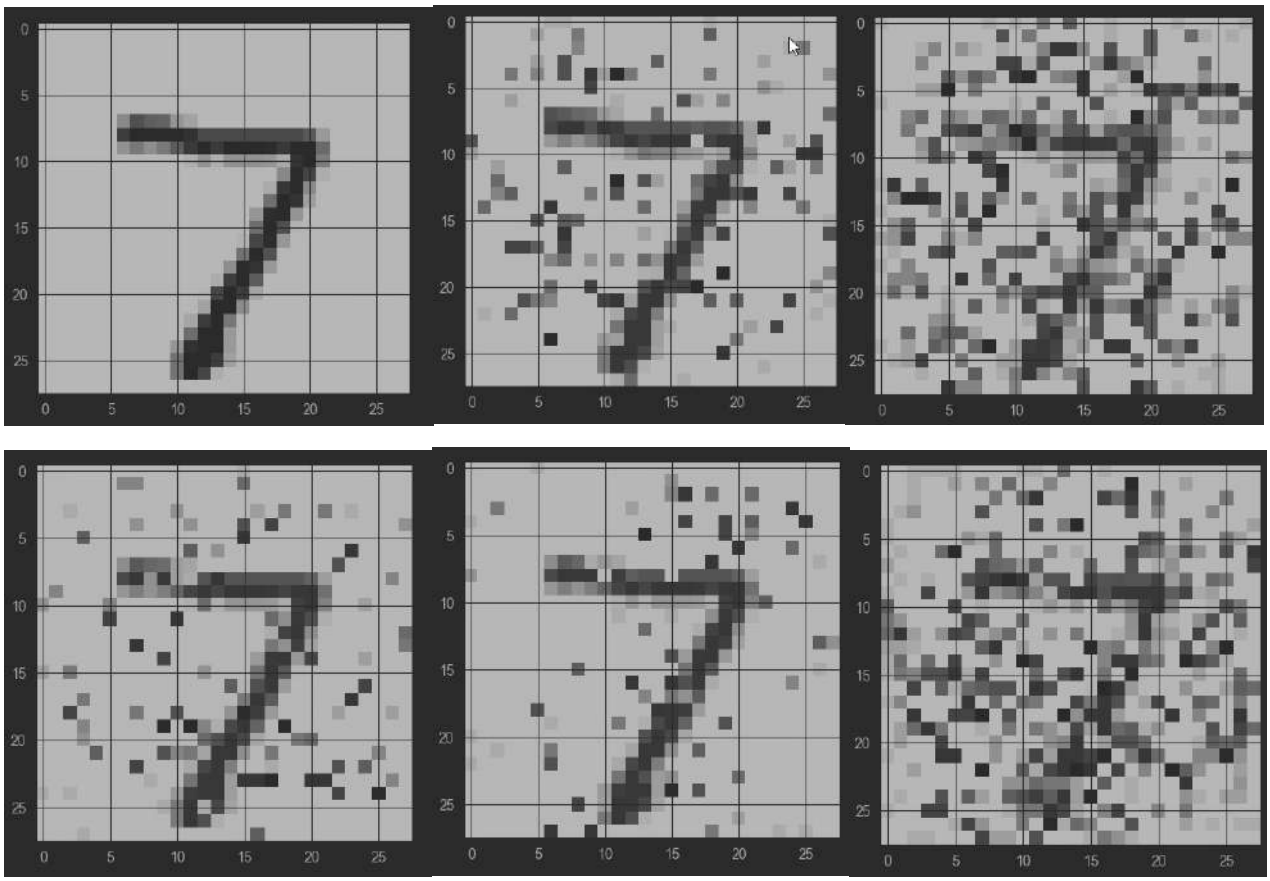


*Рисунок 4.18. Приклади застосування Dice ML*

Оскільки зображення чорнобілі, з малою роздільною здатністю і досить простим алгоритмом пошуку конрафактичного прикладу, зміни сильно помітні на кожному з зображень.

Для покращення було використано інший підхід.

Алгоритм складається з 2 частин:

VAE (Variational Autoencoder) вчиться стискати зображення в латентний простір і потім відновлювати їх назад (енкодер і декодер).

Класифікатор просто навчається розпізнавати цифри.

Алгоритм бере оригінальне зображення і перетворює його в латентний вектор z через VAE (енкодер). Потім цей вектор z змінюють, щоб класифікатор вирішив, ніби це інший клас, і одночасно z залишався близьким до початкового.

Результати демонстрації моделі з невеликою кількістю епох навчання для цифри 7:

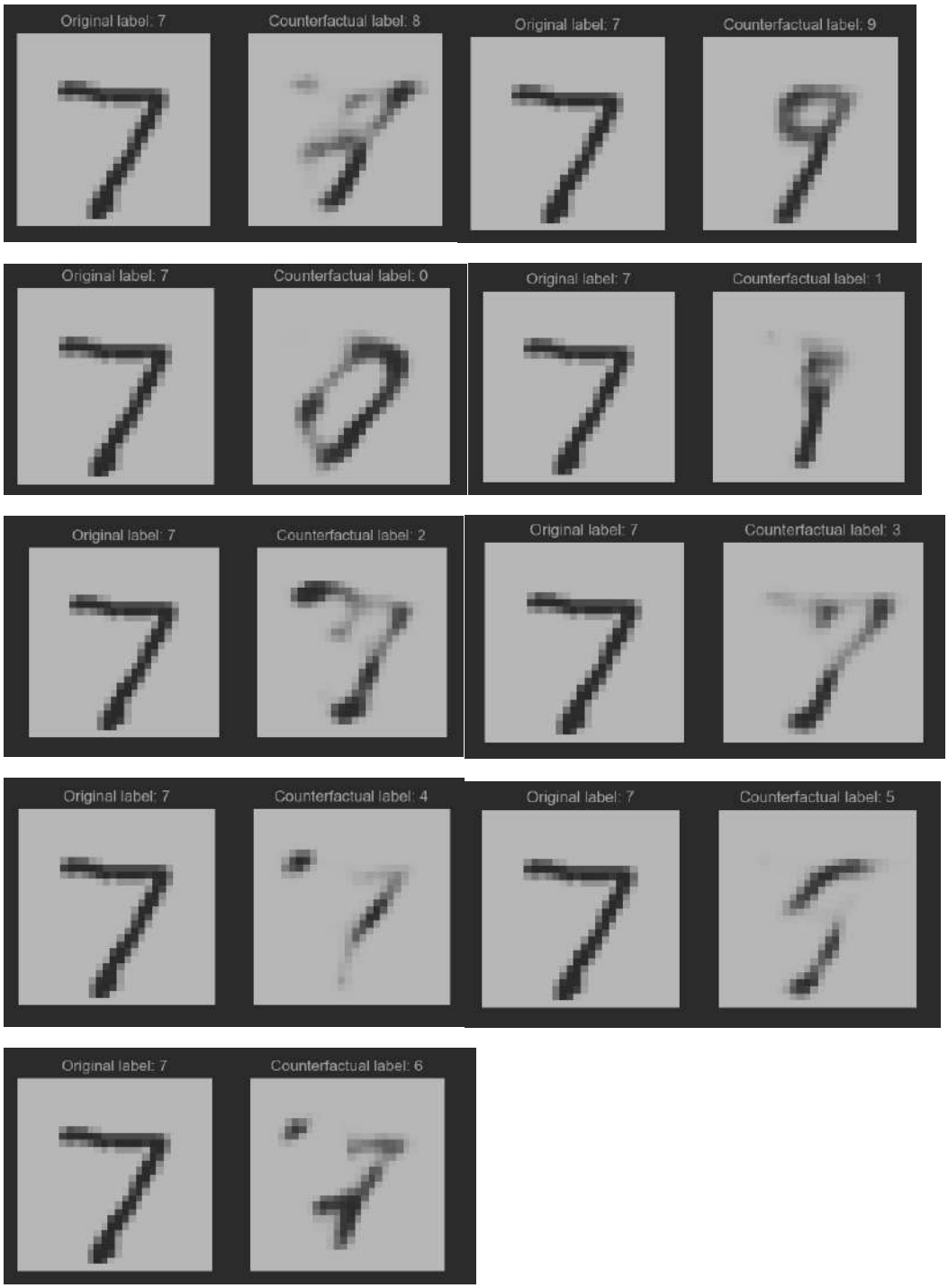



*Рисунок 4.19. Приклади застосування VAE*

За допомогою такого підходу чітко видно, які частини зображення впливають на класифікацію.

4.5. Градієнтні методи

**Saliency map**

Saliency map відображає, наскільки вихід моделі залежить від кожного пікселя (або ознаки) під час прийняття рішення. Основна ідея методу полягає в

обчисленні градієнту виходу моделі відносно вхідних пікселів. Якщо невелика зміна пікселя сильно впливає на прогноз, то градієнт буде великим — це свідчить про високу важливість такого пікселя.

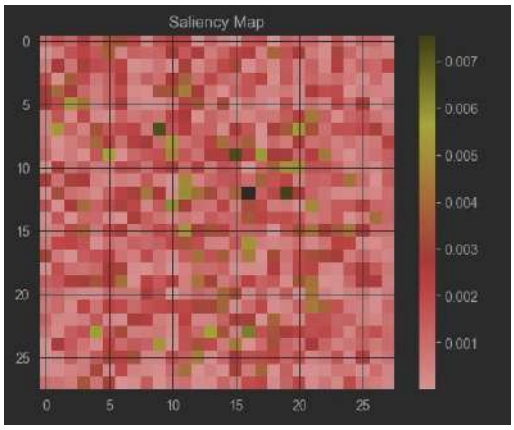



*Рисунок 4.20. Приклад використання saliency map*

Saliency map дає змогу побачити, які саме області зображення модель використовує для розпізнавання. Якщо модель орієнтується не на суттєві деталі (наприклад, фон або шуми), можна робити висновки щодо її правильної/неправильної інтерпретації вхідних даних.

Наведений приклад показує, що найважливіші пікселі знаходяться в центрі зображення, оскільки цифри у наборі даних центровані. Також варто відзначити, що їх патерн (з певним шумом) збігається з візуалізацією важливості ознак дерев рішень.

4.6. PCA і t-SNE

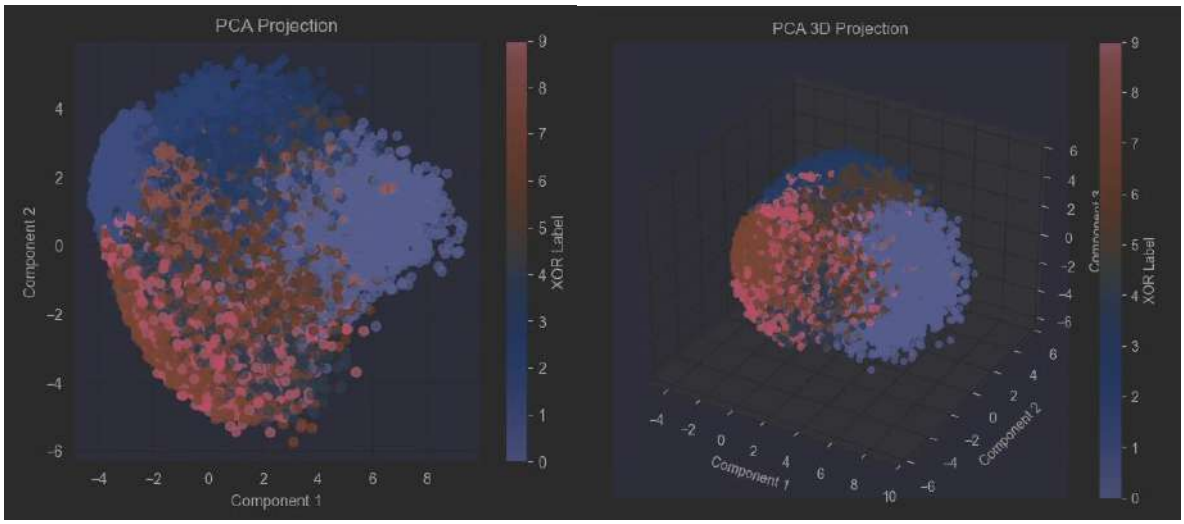



*Рисунок 4.21. 2- і 3-компонентна візуалізація PCA*

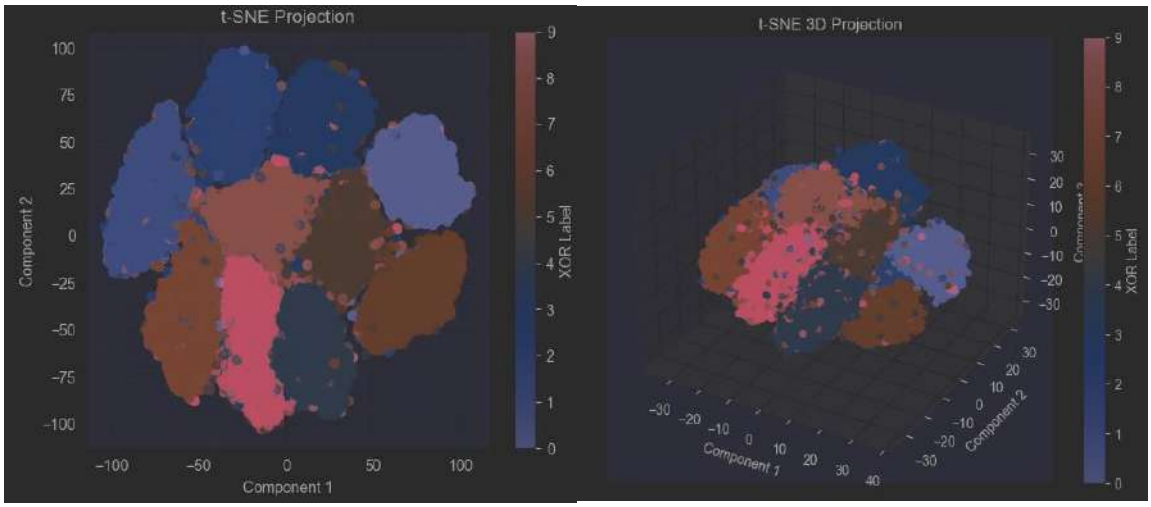



*Рисунок 4.22. 2- і 3-компонентна візуалізація t-SNE*

Після отримання ознак з моделі CNN (2 шарова CNN модель), їх можна також зменшити до 2D-простору за допомогою PCA чи t-SNE. PCA у такому випадку дає змогу оцінити, наскільки CNN модель генерує компактне представлення, в якому подібні зображення (одного класу) групуються поруч, а t-SNE показує більш детальний локальний поділ, що дозволяє краще відокремити групи зображень за класами.

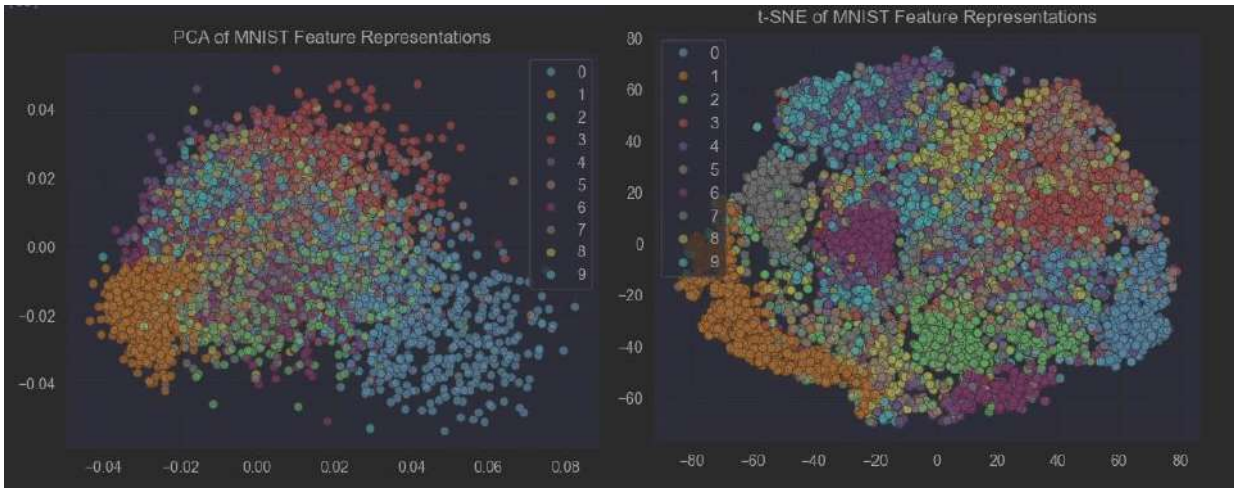



*Рисунок 4.23. 2-компонентна візуалізація ознак CNN з PCA та t-SNE*

Використовуючи 1D PCA, точки малюються по осі X (значення першої компоненти) з додатковим випадковим зсувом по осі Y. Це дозволяє візуально оцінити, як єдина компонента розподіляє дані й демонструє потенційні розділення класів.

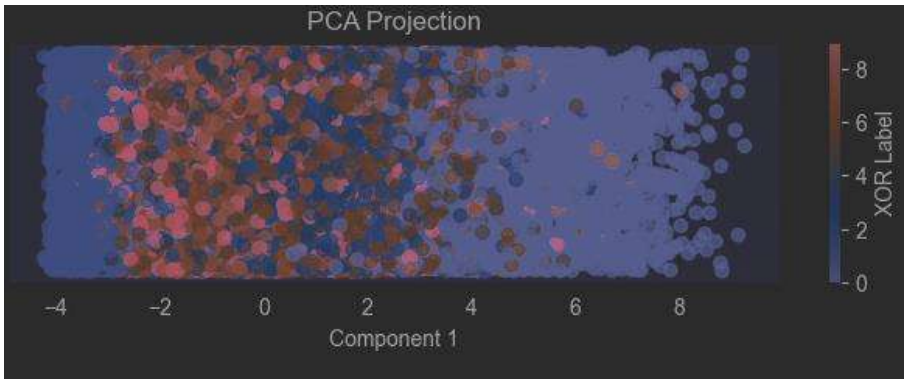



*Рисунок 4.24. 1-компонентна візуалізація PCA*

4.7. Візуалізація CNN.

У CNN кожен фільтр реагує на певні локальні патерни: це можуть бути контури, текстури або специфічні частини зображення, тому може бути корисно візуалізувати їх.

Аналіз сітки активацій дозволяє визначити, які саме області зображення активізуються в певних шарах. Наприклад, шари на початку мережі (ближче до вхідного) можуть реагувати на краї або базові форми, тоді як глибші шари здатні захоплювати більш абстрактні характеристики.

Для CNN з двома згортковими шарами та двома повнозв’язними:

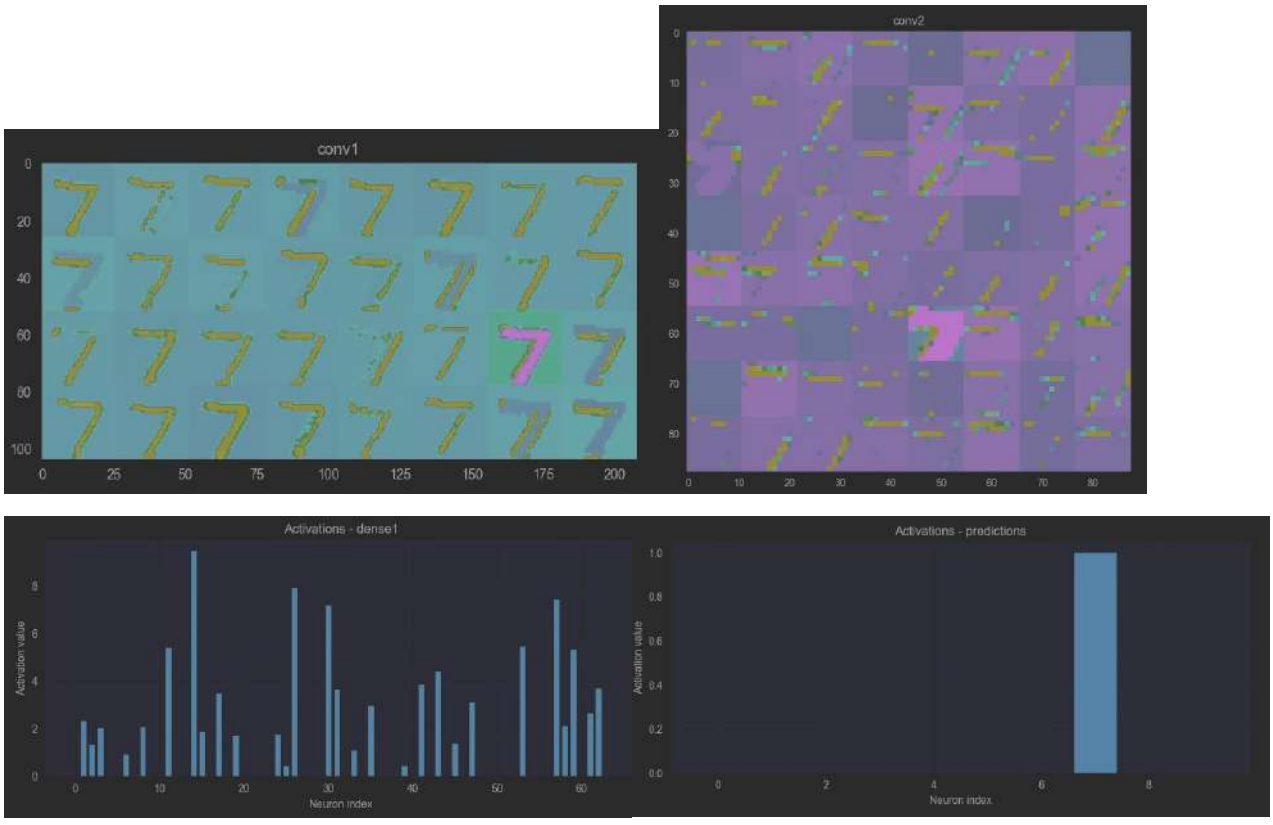



*Рисунок 4.25. Візуалізація шарів CNN*

Також цікавим і корисним є пошук еталонного зображення – зворотній процес до класифікації зображення – для заданих ймовірностей класів побудувати зображення.

Алгоритм починається з випадкового шумового зображення. Потім, за допомогою градієнтного підйому, змінюються значення пікселів так, щоб максимізувати відповідь моделі для обраного класу (наприклад, цифри 0, 1, 2 тощо).

Модель поступово змінює зображення, щоб воно містило патерни, що найбільше схожі на ті ознаки, за якими модель розпізнає певну цифру. Це можна уявити як пошук оптимального зображення, яке викликає найбільшу активність нейрона для потрібного класу.

Ці зображення допомагають зрозуміти, на які деталі модель звертає увагу, що сприяє прийняттю рішення. Це важливо для інтерпретації та перевірки працездатності моделі, особливо коли хочуть переконатися, що модель відрізняє вхідні дані за релевантними ознаками, а не за шумом.

Для візуалізованої раніше CNN це виглядатиме так:

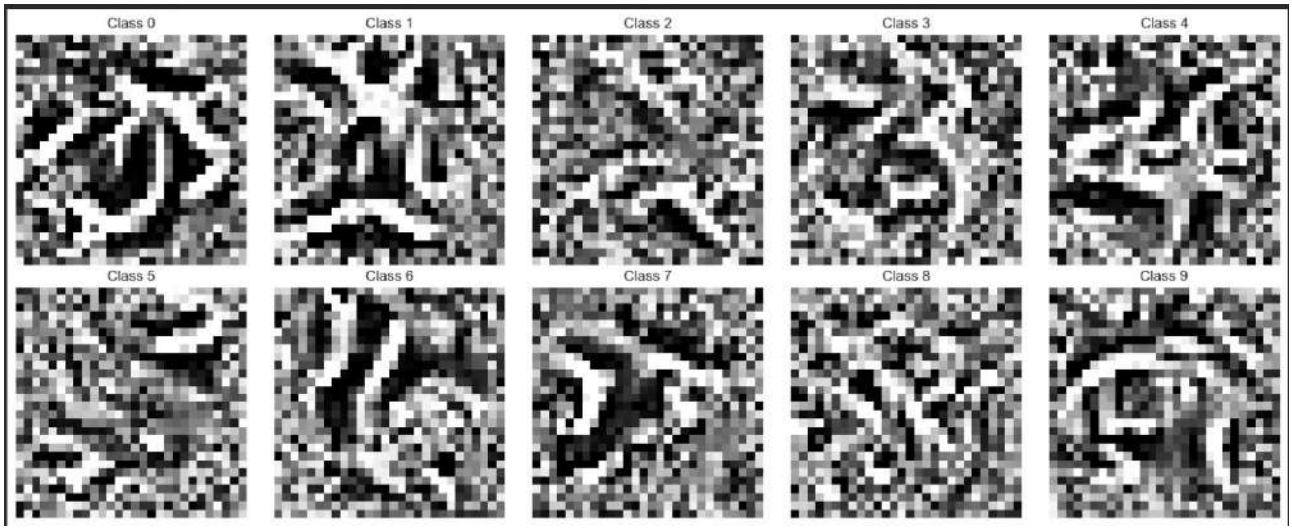



*Рисунок 4.26. Візуалізація еталонних зображень CNN*

На цих зображеннях можна впізнати узагальнені фрагменти і обриси цифр, що генерувались.

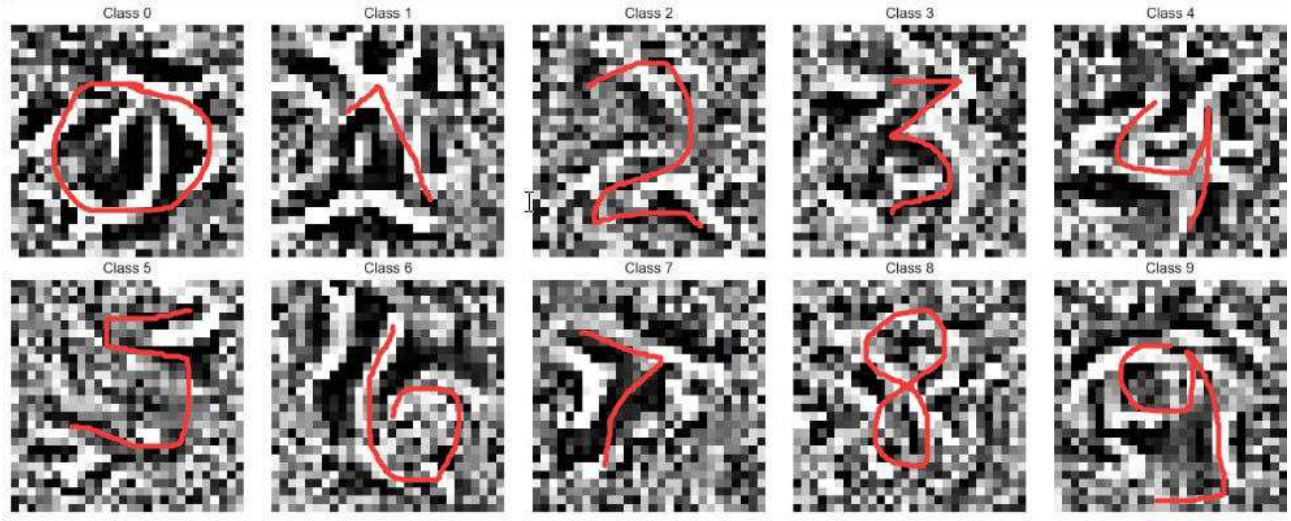



*Рисунок 4.27. Візуалізація еталонних зображень CNN*

При збільшенні глибини моделі патерни стають складнішими і зникає закономірність між реальною цифрою і еталонним зображенням. Так для складнішої моделі – з 6 згортковими шарами, MaxPooling, BatchNormalization і 3 Dense шарами з Dropout візуалізація виглядає абстрактно.

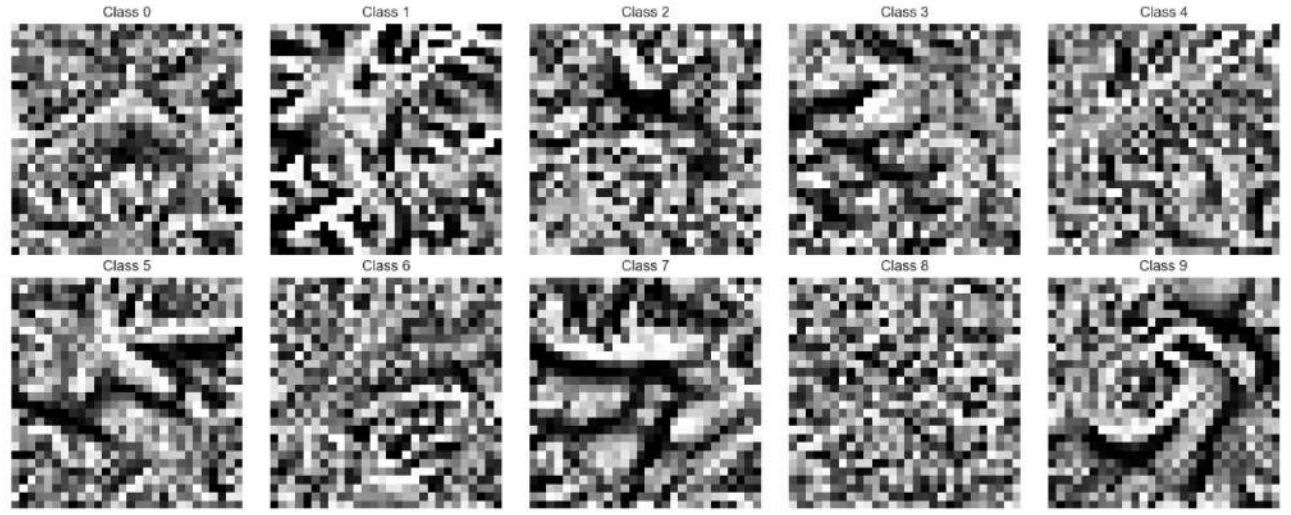



*Рисунок 4.28. Візуалізація еталонних зображень глибокої CNN*

При використані ResNet, еталонні зображення починають більше нагадувати шум:

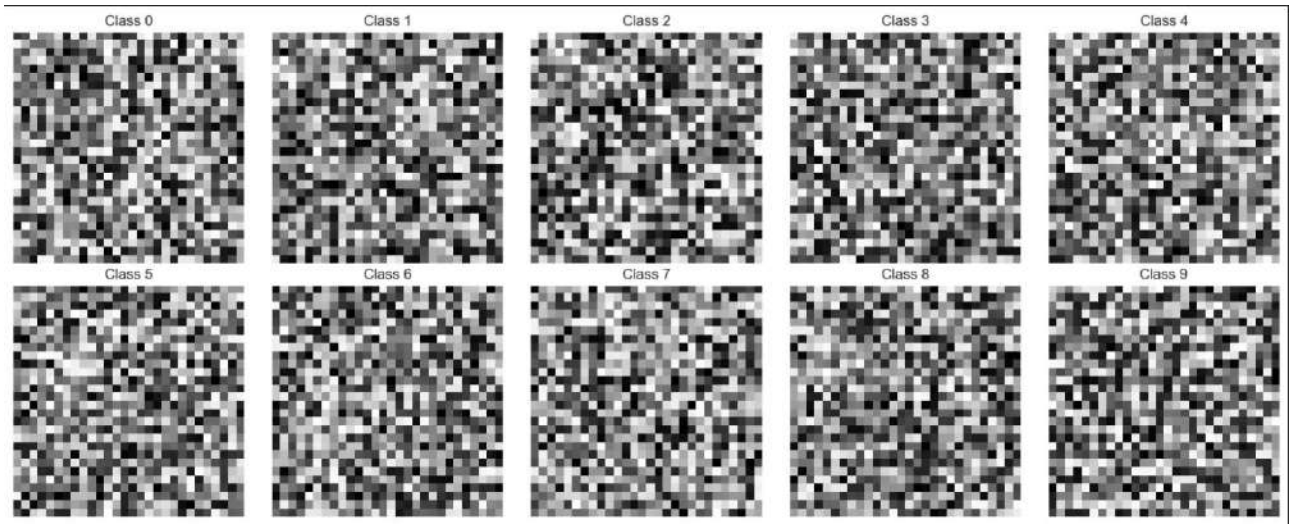



*Рисунок 4.29. Візуалізація еталонних зображень ResNet*

4.8. Порівняння методів і моделей.

Як і для XOR використано DT, XGBoost, MLP (1 прихований шар, 64 нейрони), Deep MLP (3 призованих шари по 64 нейрони), BernouliNB, LSTM, проте решту моделей замінено на GaussianNB, 2D-CNN через велику обчислювальну вартість.

| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Decision Tree | 0.8743 | 0.8725 | 0.8725 | 0.8725 |
| MLPClassifier | 0.9745 | 0.9743 | 0.9744 | 0.9743 |
| Deep MLP (3 * 64) | 0.9724 | 0.9723 | 0.9720 | 0.9721 |
| XGBoost | 0.9766 | 0.9766 | 0.9764 | 0.9764 |
| BernoulliNB | 0.8343 | 0.8349 | 0.8314 | 0.8321 |
| GaussianNB | 0.5539 | 0.6804 | 0.5460 | 0.5084 |
| 2D-CNN | 0.9850 | 0.9852 | 0.9850 | 0.9850 |
| LSTM | 0.9844 | 0.9845 | 0.9842 | 0.9843 |

*Таблиця 4.7. Точність моделей*

У таблиці наведено точність класифікації, досягнуту кожною моделлю на тестовому наборі MNIST. Неглибоке дерево рішень досягає лише 87.43% точності, тоді як Бернуллі та гаусівський наївний Байєс показують ще більш скромні результати – 83.43% та 55.39% відповідно, що відображає їхню

нездатність вловлювати просторові залежності штрихів у піксельному просторі. Проте у порівнянні з набором даних XOR, вони дають значно кращі результати, тобто ознаки мають певну незалежність одна від одної (не повну, але значно вищу, ніж для XOR). На відміну від них, усі нейронні та ансамблеві методи перевищують 97% точності: один прихований шар MLP досягає 97.45 %, глибший MLP (3*64) – 97.24%, а дерева з градієнтним підсиленням – 97.66%. Архітектури, які явно використовують структуру зображення, а саме 2D-CNN (98.50%) і LSTM (98.44 %), ще більше підвищують продуктивність, демонструючи переваги локальних рецептивних полів і послідовної інтеграції ознак.

**Native importance**

| Model | Method | Runtime (s) | Comp | MeanImp |
|---|---|---|---|---|
| DT | Tree FI | 0 | 508 | 0.001276 |
| MLPClassifier | InputWeightImp | 0 | 784 | 0.0948 |
| Deep MLP | InputWeightImp | 0 | 784 | 0.095671 |
| XGBoost | XGB FI | 0.0025 | 541 | 0.001276 |
| BernoulliNB | NB ProbDiff | 10.994 | 784 | 0.013332 |

*Таблиця 4.8. Native importance MNIST*

Усі методи обчислення власної важливості (Native методи) працюють за незначний час - навіть NB ProbDiff, найповільніший з них (11 с), значно швидший за багато інших методів оцінки важливості на 784-вимірних даних. Comp (кількість ненулевих ознак) відображає, що нейромережеві методи аналізу важливості за першими вагами враховують усі 784 вхідні ваги, тоді як деревовидні моделі обробляють лише ті ознаки, на які вони фактично розбиті (508 для DT і 541 для XGBoost).

Для деревовидних методів середнє значення важливості виявилось дуже малим ($1.3 * 10^{-3}$). Це може вказувати на те, що жоден окремий піксель не домінує, оскільки інформація розподіляється між багатьма розщепленнями. Для обох варіантів MLP середні значення ваги першого шару (~0,095) майже на два порядоки більші.

Власна важливість дає глобальне ранжування пікселів за їхнім загальним впливом на класифікацію. Однак на практиці ці необроблені оцінки на піксельному рівні важко інтерпретувати для складних наборів даних (наприклад, для MNIST).

**Permutation importance**

Permutation importance оцінює зниження продуктивності моделі, коли одна вхідна ознака випадковим чином перемішується, порушуючи тим самим зв'язок між цією ознакою та ціллю. У наших експериментах застосовано важливість перестановок до всіх 784 пікселів у восьми класифікаторах, щоб кількісно оцінити внесок кожного пікселя.

| Model | Runtime (s) | Comp | MeanImp |
|---|---|---|---|
| DT | 13.6851 | 464 | 0.003518 |
| MLPClassifier | 20.9283 | 536 | 0.000142 |
| Deep MLP | 25.3162 | 535 | 0.000087 |
| XGBoost | 128.591 | 433 | 0.000096 |
| BernoulliNB | 71.8078 | 548 | -0.000111 |
| GaussianNB | 753.747 | 680 | 0.001676 |
| 2D-CNN | 368.452 | 386 | 0.000264 |
| LSTM | 508.013 | 384 | 0.000012 |

*Таблиця 4.9. Permution importance MNIST*

Першим помітним результатом є час виконання. Навіть традиційно швидким моделям, таким як дерево рішень, потрібно понад 13 с, щоб виконати перестановку ознак і оцінити їх, тоді як GaussianNB займає більше 12 хвилин (753 с). Глибокі моделі (MLP, Deep MLP, LSTM, 2D-CNN) перевищують 20 с кожна, а XGBoost – 128 с. Таке масштабування вказує на те, що без групування ознак обчислення Permutation importance є надто затратним для обчислень на даних зображень високої розмірності.

По-друге, середні значення важливості зникаюче малі майже для всіх моделей. Дерево рішень показує найвищу середню важливість (0,00352), що означає падіння точності в середньому на 0,35 % при перестановці пікселя.

GaussianNB слідує за ним з показником 0,00168, але ймовірнісні моделі, такі як BernoulliNB, насправді демонструють дещо від'ємне середнє значення (-0,00011). Це свідчить про те, що випадкова перестановка пікселів може іноді підвищити точність, порушуючи пристосування до шуму. Нейронні мережі (MLP, Deep MLP), XGBoost, LSTM і 2D-CNN фіксують середню важливість у діапазоні від $10^{-3}$ до $10^{-4}$, підтверджуючи, що жоден піксель не має вирішального впливу на класифікацію; натомість розпізнавання цифр відбувається на основі розподілених високорівневих шаблонів штрихів.

**Partial Dependence**

Partial Dependence (PD) оцінює, як в середньому реагує прогнозована моделлю ймовірність правильного класу, коли ми змінюємо один піксель, залишаючи всі інші пікселі при їхньому емпіричному спільному розподілі.

| Model | Runtime (s) | Comp | Mean Range |
|---|---|---|---|
| DT | 0.0339 | 784 | 0 |
| MLPClassifier | 0.0497 | 784 | 0.000302 |
| Deep MLP | 0.0535 | 784 | 0.000207 |
| XGBoost | 0.1261 | 784 | 0.000129 |
| BernoulliNB | 0.1673 | 784 | 0.000310 |
| GaussianNB | 1.5103 | 784 | 0.000507 |
| 2D-CNN | 1.1738 | 784 | 0.000659 |
| LSTM | 1.3031 | 784 | 0.000414 |

*Таблиця 4.10. Partial dependence MNIST*

Для всіх моделей час виконання PDP залишається скромним - трохи менше двох секунд – навіть для вхідних даних високої розмірності (784 пікселі). Тим не менш, спостережувані значення MeanRange зникаюче малі (порядку $10^{-3}$ або менше для нейронних моделей, і точно нуль для дерева рішень). Така закономірність вказує на те, що жоден окремий піксель не має сильного впливу на класифікацію; натомість, передбачення виникає в результаті розподіленої взаємодії між багатьма пікселями.

Результати PDP підтверджують, що граничні ефекти на необроблені пікселі для MNIST фактично дорівнюють нулю. Отже, будь-яка значуща інтерпретація повинна ґрунтуватися на характеристиках вищого рівня – наприклад, суміжних штрихах, орієнтації країв – а не на значеннях окремих пікселів.

**Surogate Linear Model**

У таблиці наведено оцінки простої лінійної регресії, навченої імітувати передбачення кожного класифікатора на MNIST. Оцінено точність за допомогою середнього абсолютного значення коефіцієнта (MeanCoef), точності класифікації (ClsFidelity, частка випадків, для яких сурогат прогнозує ту саму мітку) та покриття помилок (ErrCoverage, частка випадків, коли сурогат неправильно класифікує).

| Model | Runtime (s) | Comp | Mean Coef | Cls Fidelity | Err Coverage |
|---|---|---|---|---|---|
| DT | 0.0044 | 0 | 0.000 | 0.00025 | 1 |
| MLP | 0.0041 | 681.5 | 0.005111 | 0.00075 | 1 |
| Deep MLP | 0.0042 | 659.8 | 0.004109 | 0.00075 | 1 |
| XGBoost | 0.0068 | 459.6 | 0.001526 | 0 | 1 |
| Bernoulli NB | 0.0065 | 381.1 | 0.001160 | 0 | 1 |
| Gaussian NB | 0.0146 | 0 | 0.000 | 0 | 1 |
| 2D-CNN | 0.0652 | 539 | 0.002037 | 0 | 1 |
| LSTM | 0.0665 | 670.7 | 0.004362 | 0 | 1 |

*Таблиця 4.11. Surogate Linear Model for MNIST*

Ці результати демонструють фундаментальне обмеження лінійних сурогатів на дуже вимірних даних зображень: із 784 незалежних пікселів розріджена лінійна регресія за замовчуванням видає тривіальний або дуже простий лінійний результат замість відтворення складної граничної функції

оригінального класифікатора (Clsfidelity майже нульова). На практиці, перш ніж сурогатні моделі зможуть дати значущі, зрозумілі людині коефіцієнти для задач обробки зображень, необхідно спочатку зменшити розмірність - за допомогою групування ознак, PCA або векторів активації концепцій, проте в результаті такої попередньої обробки втрачається зв'язок з оригінальними ознаками.

**Counterfactual explanation**

| Model | Runtime (s) | Comp Mean | Mean ΔProb | Std ΔProb |
|---|---|---|---|---|
| DT | 0.0263 | 1.0 | 1.000 | 0.000 |
| MLP | 0.2855 | 0.05 | 0.026799 | 0.116812 |
| Deep MLP | 0.1541 | 0.05 | 0.030864 | 0.134531 |
| XGBoost | 0.2207 | 0.05 | 0.009523 | 0.041510 |
| Bernoulli NB | 0.1383 | 0.15 | 0.072289 | 0.179389 |
| Gaussian NB | 0.0039 | 1.0 | 1.000 | 0.000 |
| 2D-CNN | 37.7414 | 0.0 | 0.000 | 0.000 |
| LSTM | 33.8701 | 0.1 | 0.043282 | 0.135821 |

*Таблиця 4.12. Counterfactual explanation for MNIST*

Контрфактичний аналіз показує, що невеликі моделі (Decision Tree, GaussianNB) міняють у середньому 1 піксель, щоб повністю змінити прогноз (CompMean = 1.0, ΔProb = 1.0) менш ніж за 30 мс (що показує їх нестабільшість і недостовірність). На противагу цьому, CNN повністю не спрацьовують (CompMean = 0), а LSTM потребують десятки секунд, щоб знайти дуже слабкі фліпи (MeanΔProb ≈ 0.04). BernoulliNB знаходиться між цими крайнощами (CompMean = 0.15, MeanΔProb ≈ 0.07). Загалом, сирі піксельні контрфакти мають сенс лише для простих моделей; для глибоких класифікаторів зображень потрібно збурювати концепції вищого рівня, а не окремі пікселі.

**Saliency (Gradient Based)**

| Model | Runtime (s) | Comp Mean | Mean ΔProb | Std ΔProb |
|---|---|---|---|---|
| DT | 0.0987 | 0 | 0.000 | 0.000 |

| | | | | |
|---|---|---|---|---|
| MLP | 0.1438 | 349.45 | 1.275890 | 4.969080 |
| Deep MLP | 0.1457 | 234.55 | 0.539371 | 1.342880 |
| XGBoost | 0.2056 | 442.95 | 0.820501 | 2.938480 |
| Bernoulli NB | 0.1817 | 313.45 | 7.988150 | 19.854700 |
| Gaussian NB | 0.1871 | 16.30 | 0.359629 | 0.547816 |
| 2D-CNN | 37.8490 | 631.55 | 0.570742 | 1.256610 |
| LSTM | 36.9800 | 769.75 | 5.174350 | 15.038100 |

*Таблиця 4.13. Saliency for MNIST*

У таблиці наведено градієнтну ефективність для кожного класифікатора в MNIST. Невеликі моделі, такі як Decision Tree, не дають значущого градієнта (MeanΔProb = 0 за 0.10 с), тоді як GaussianNB дає невеликі, але стабільні зсуви на рівні пікселів (MeanΔProb = 0.36 за 0.19 с). MLPClassifier і XGBoost генерують більші величини значущості (MeanΔProb = 1.28 і 0.82), але з дуже високою дисперсією, а Deep MLP знаходиться між ними (MeanΔProb = 0.54 з помірним рівнем шуму). Критично важливо, що 2D-CNN і LSTM займають 37 с на пояснення і все ще демонструють зашумлені карти, що робить градієнтний аналіз сирих пікселів як обчислювально складним, так і ненадійним - підкреслюючи потребу в поясненнях вищого рівня або агрегованих ознак, а не окремих пікселів.

4.9. Concept explanation (TCAV)

Щоб виявити високорівневі, зрозумілі людині патерни, зафіксовані кластерами ознак, застосовано TCAV до кожного з десяти кластерів, отриманих за допомогою K-mean на 128-вимірних ембедінгах найтренованої нейромережі з 1 прихованим шаром. Визначено десять концептів форми штрихів, таких як «loops», «one_loop», «two_loops», «straight», «curves», «diagonals», «horizontals», «verticals», «cross», «symmetry», в результаті групування цифр з цими характеристиками ("loops": {0,6,8,9}, "one_loop": {6,9}, "two_loops": {8}, "straight": {1,4,7}, "curves": {2,3,5}, "diagonals": {4,7}, "horizontals": {2,5,8}, "verticals": {1,4,6}, "cross": {4}, "symmetry": {0,1,3,8}).

Для кожного поняття навчено простий лінійний класифікатор відрізняти його позитивні приклади від усіх інших, а потім для кожного кластера виміряно частку зразків, градієнти яких позитивно збігаються з вектором поняття.

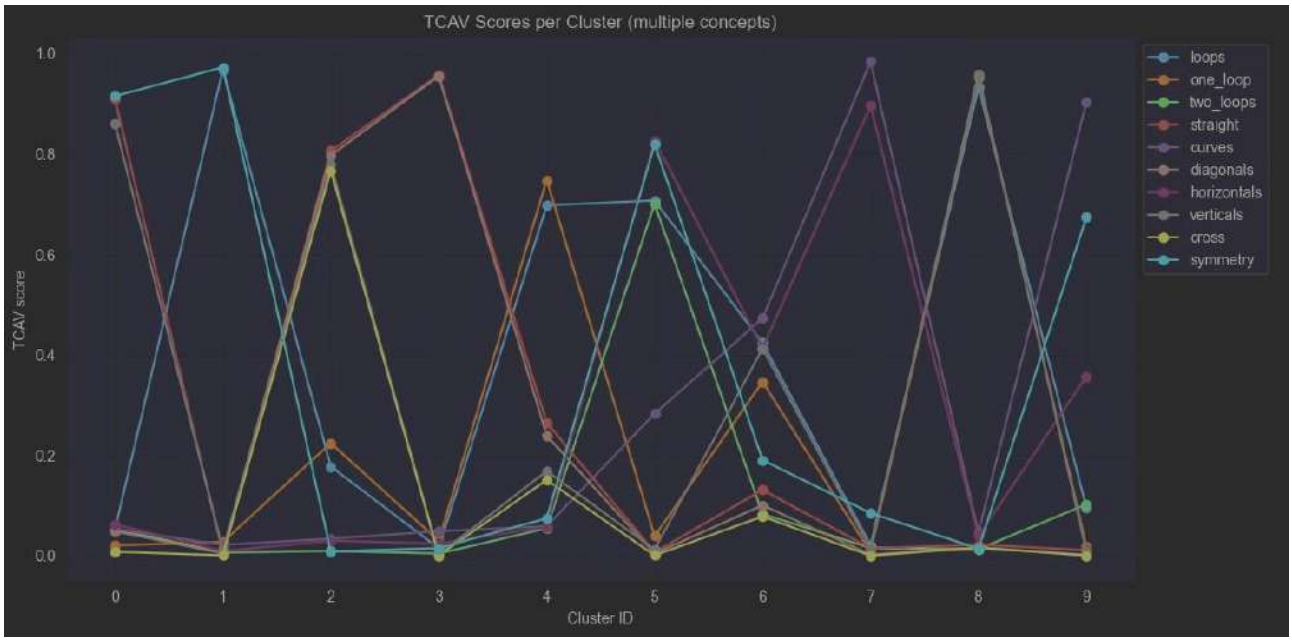



*Рисунок 4.30. Візуалізація оцінок TCAV для усіх концепцій у кластерах*

Нижче на рис. 4.21 показано оцінки TCAV для всіх концептів у кластерах, які зображено у вигляді ліній. Можна побачити, що кластери 0 і 1 мають дуже високі оцінки для «loops», кластер 2 - для «two_loops », кластер 3 - для «straight» і так далі. Це свідчить про те, що кожен кластер дійсно групує зображення за домінуючими атрибутами штрихів. На рисунку 4.22 ті самі дані перекодовано у вигляді теплової карти, що дає змогу легко виявити поняття, які визначають кожен кластер.

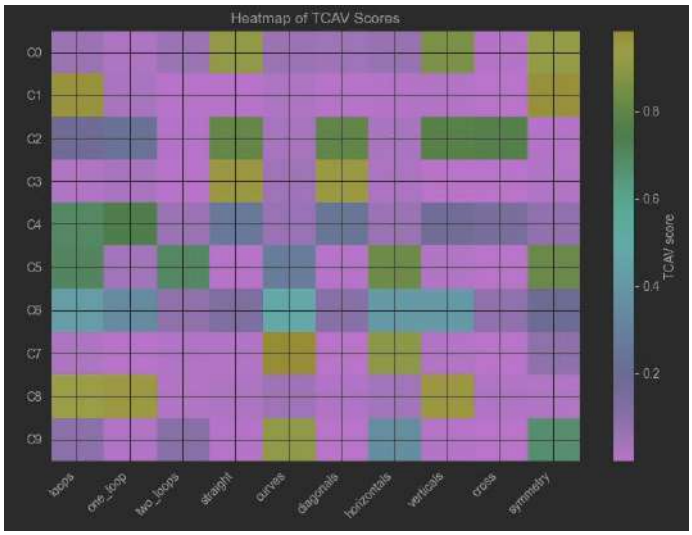


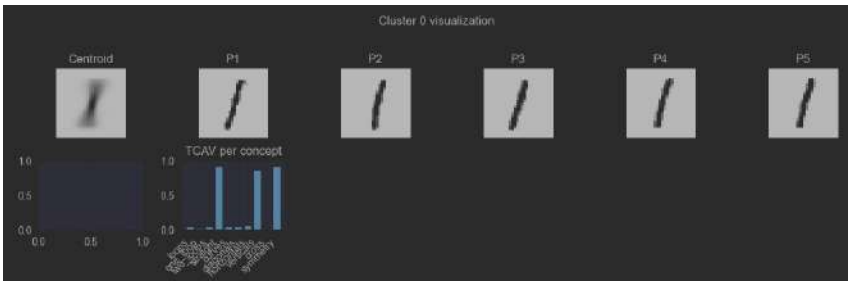

*Рисунок 4.31. Теплова карта оцінок TCAV для усіх концепцій у кластерах*

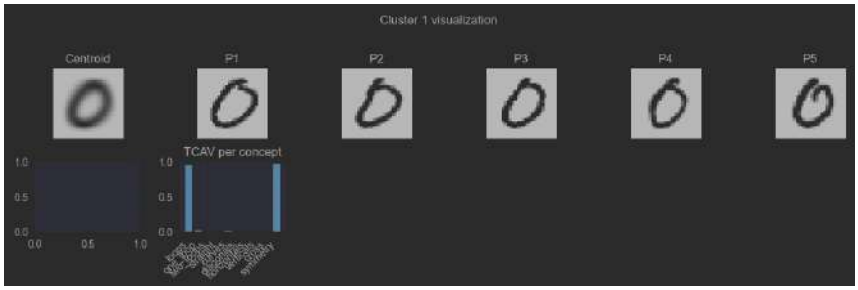
Щоб повернути ці семантичні асоціації назад у піксельну область, візуалізовано кожен кластер:

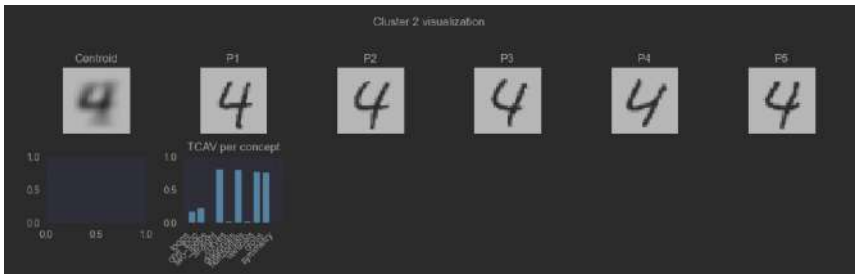
- Центроїдне зображення (середнє значення всіх членів),
- П'ять найближчих прототипів (реальні зображення цифр, найближчі до центроїда)
- Гістограму оцінок TCAV для десяти концептів.

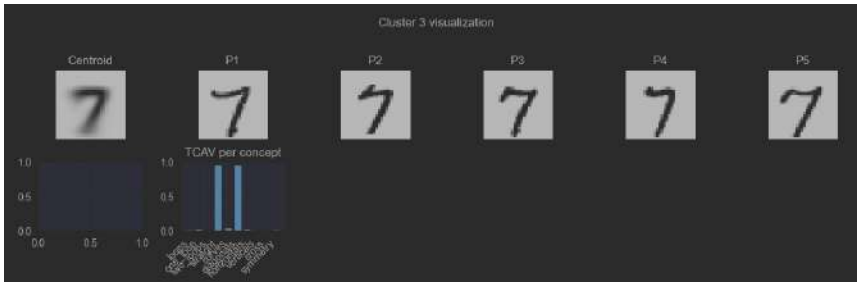
З цікавих результатів можна звернути увагу на цифру 4 (2 кластер) для неї було визначено лише концепції «straight», «diagonals», «verticals» та «cross», але до них додались концепції «loop» та «one_loop» з не великою вагою, оскільки у наборі даних MNIST є зображення цифри 4 з різним написанням – з замкнутими кінцями, що формують петлю, і відкритими.

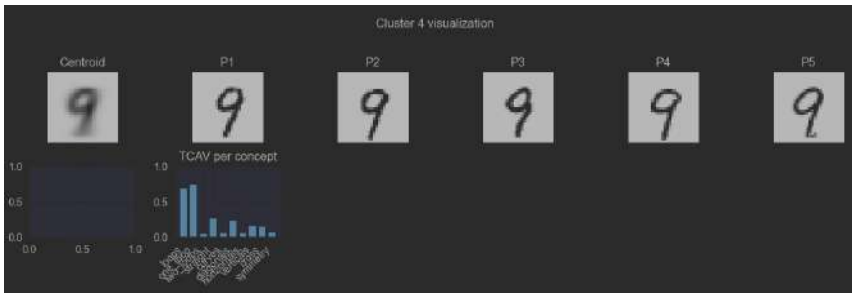
Також можна виокремити 2 кластери, у яких екземпляри виглядають як цифра 6 (кластери 6 і 8). Це пов'язано з різним написання цифри 6, у одному з кластерів вона групується з цифрою 5 (що видно з центроїди) через написання з розривом, а у іншому лише переважно цифри 6 з петлею. Це також може вказувати на проблему розпізнавання цих цифр нейронною мережею, з якої взято ембедінги.


Cluster 0 visualization
Centroid
P1
P2
P3
P4
P5
TCAV per concept


Cluster 1 visualization
Centroid
P1
P2
P3
P4
P5
TCAV per concept


Cluster 2 visualization
Centroid
P1
P2
P3
P4
P5
TCAV per concept


Cluster 3 visualization
Centroid
P1
P2
P3
P4
P5
TCAV per concept


Cluster 4 visualization
Centroid
P1
P2
P3
P4
P5
TCAV per concept

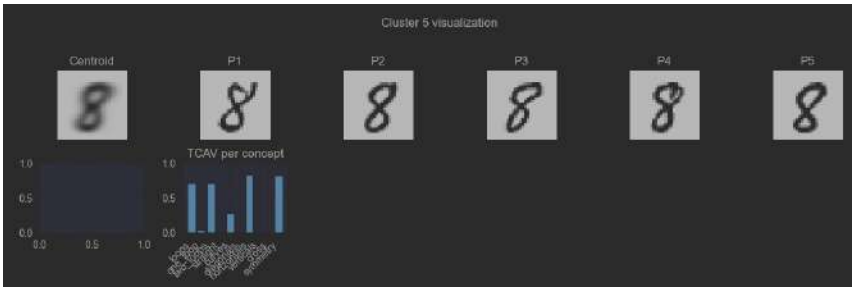

Cluster 5 visualization
Centroid
P1
P2
P3
P4
P5
TCAV per concept


*Рисунок 4.32. Візуалізація центроїдів кластерів, їх прикладів і концепцій*

# РОЗДІЛ 5. BRAIN TUMOR

5.1. Лінійні моделі.

Як і для попередніх 2 задач використовувались Linear, Lasso, Ridge Regression з метриками MSE і $R^2$.

| Model | MSE | $R^2$ |
|---|---|---|
| Linear Regression | 0.61 | 0.53 |
| Ridge (α = 0.1) | 0.64 | 0.51 |
| Lasso (α = 1.0) | 1.14 | 0.13 |
| Lasso (α = 0.001) | 0.40 | 0.70 |

*Таблиця 5.1. Метрики для Regression моделей*

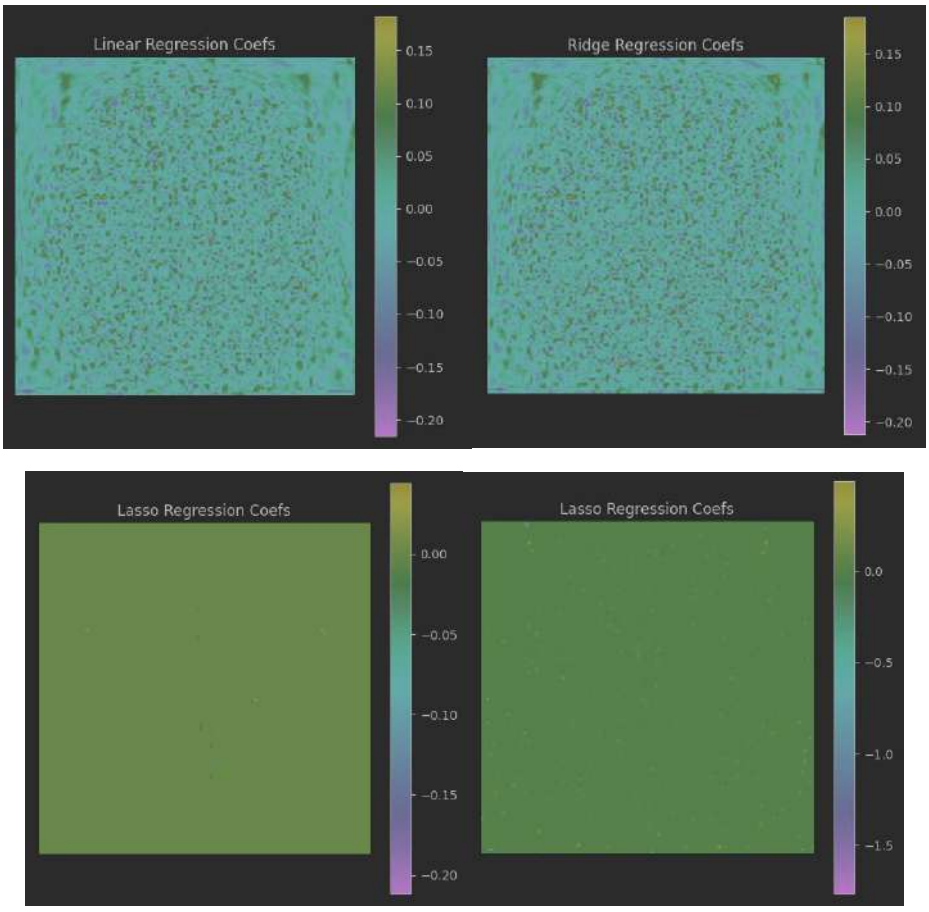



*Рисунок 5.1. Візуалізація коефіцієнтів Regression моделей*

Для Linear і Ridge моделей ваги розкидані по всьому зображенню, без чіткої структури і являють собою переважно випадковий шум.

Для Lasso більшість коефіцієнтів мають значення очікуване значення 0.

Lasso regularization path має вигляд:

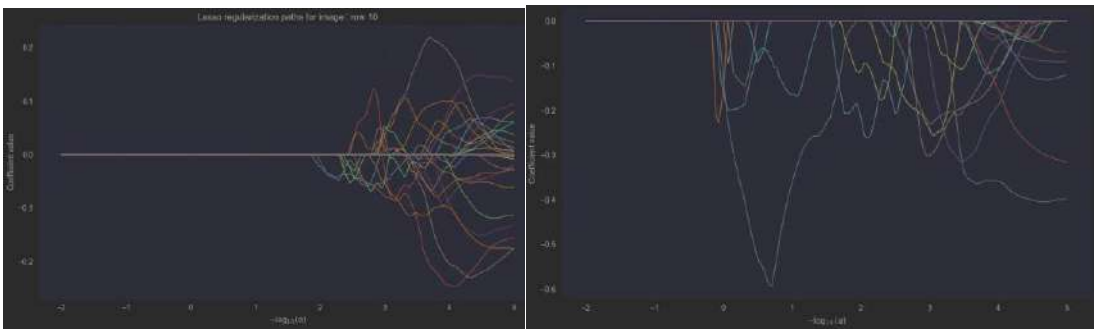


*Рисунок 5.2. Lasso Regularization Path*

Шляхи є значно складнішими, ніж у XOR чи MNIST датасету, оскільки зображенням мають значно більшу кількість ознак і складнішу структуру. Шляхи для перших і останніх рядків разюче відрізняються від всіх інших оскільки в основному відповідають за фон зображення.

5.2. Дерева рішень

Для дерева рішень з необмеженою глибино отримано точність у 89%.

Візуалізація дерева виглядає значно компактніше, ніж для MNIST, незважаючи на значно більшу кількість ознак. Це може стверджувати, що лише невелика кількість ознак насправді впливає на результат.

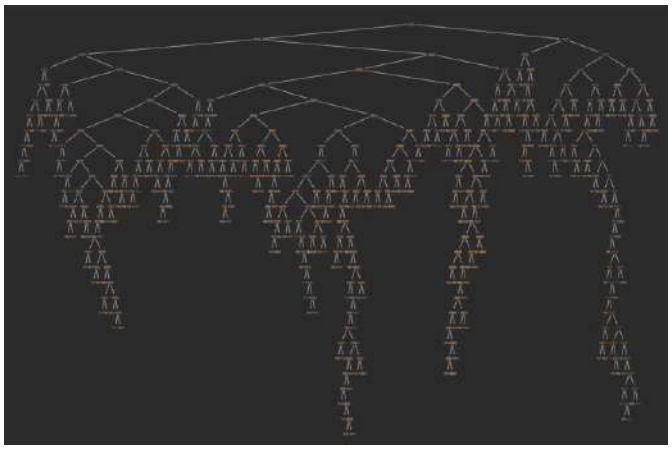


*Рисунок 5.3. Візуалізація дерева рішень*

Про це ж говорить мапа важливості ознак, на котрій майже усі ознаки мають важливість 0.

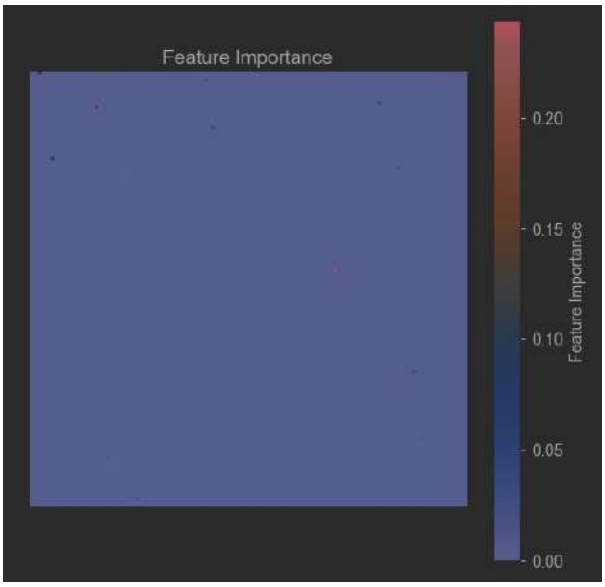



*Рисунок 5.4. Візуалізація важливості ознак дерева рішень*

Як і раніше з отриманого дерева рішень було виділено правила і створено синтетичний датасет (20 прикладів на правило), та перевірено його точність за допомогою MLP. Знову порівняно неперервне та двійкове кодування пікселів, три глибини дерев і вимірюємо, наскільки добре MLP, навчений на реальних даних, переносить їх на синтетичні приклади.

| | Depth=6 | Depth=10 | Depth=unlimited |
|---|---|---|---|
| DT train | 86.47% | 98.11% | 100.00% |
| DT test | 79.63% | 88.71% | 89.93% |
| MLP train | 99.26% | 98.67% | 99.95% |
| MLP test | 89.47% | 90.69% | 91.08% |
| Rules | 51 | 179 | 222 |
| MLP on synthetic | 15.20% | 17.88% | 18.47% |
| Synthetic size | 1020 | 3580 | 4440 |

*Таблиця 5.2. Результати оцінки синтетичного набору на безперервних даних*

| | Depth=6 | Depth=10 | Depth=unlimited |
|---|---|---|---|
| DT train | 50.54% | 61.40% | 99.98% |
| DT test | 49.50% | 59.27% | 78.64% |
| MLP train | 99.96% | 99.96% | 99.96% |
| MLP test | 89.47% | 88.94% | 88.33% |

| | | | |
|---|---|---|---|
| Rules | 42 | 129 | 786 |
| MLP on synthetic | 36.31% | 37.56% | 21.77% |
| Synthetic size | 840 | 2580 | 15720 |

*Таблиця 5.3. Результати оцінки синтетичного набору на бінаризованих даних*

Глибші дерева підвищують точність реальних даних (DT test з 79.6 до 89.9 % на безперервних ознаках; з 49.5 до 78.6 % - на бінарних), але також збільшують кількість правил (з 51 до 222 на безперервних ознаках; з 42 до 786 – на бінарних).

При неперервному кодуванні синтетична точність зростає незначно (з 15.2 до 18.5 %), незважаючи на більшу кількість правил, що відображає те саме погане покриття і перенавчання, яке спостерігається в MNIST.

Двійкове кодування дає набагато кращі результати (36-38% на глибині 6 та 10), перш ніж впасти на необмеженій глибині (21.8 %). Це може свідчити про те, що порогові ознаки фіксують більш стабільні, узагальнюючі шаблони в цьому наборі даних.

На відміну від MNIST, де безперервна дискретизація дещо перевершила бінарну, тут бінарна попередня обробка значно покращує якість синтетичних даних на малих глибинах. Це свідчить про те, що для цього набору даних дискретні розбиття краще узгоджуються з пороговими значеннями ознак, що визначають клас.

Другий набір даних підтверджує, що неглибокі набори правил (глибина=10) і правильна попередня обробка можуть дати синтетичні вибірки, які частково переносяться на MLP - особливо, коли ознаки бінаризовані. Однак навіть найкраща синтетична ефективність (38%) залишається набагато нижчою за точність тесту на реальних даних (90%), що підкреслює потребу в удосконаленні з урахуванням щільності або генерації для подолання цього розриву.

Поєднання ансамблю неглибоких дерев (10 дерев з depth = 4) із гаусівською генерацією й mixup прикладами дало синтетичний набір обсягом

56010 зразків. Завдяки цьому MLP навчається на реалістичніших синтетичних даних і підвищує точність з приблизно 31 % до 52.7 %.

Як і раніше перевірено зворотню задачу – навчання MLP на синтетичному наборі, перевірка на реальному. Отримано такі результати:

| | Depth=6 | Depth=10 | Depth=unlimited |
|---|---|---|---|
| Rules | 51 | 179 | 222 |
| Synthetic Size | 1020 | 3580 | 4440 |
| Synth. MLP train | 29.41% | 17.96% | 23.86% |
| Synth. MLP test | 31.12% | 21.51% | 24.41% |

*Таблиця 5.4. Результати оцінки тренування MLP на синтетичному наборі*

З чотирьох класів лише правила глибини 6 перевищують базовий рівень (31 %). Для датасету отриманого з правил ансамблю дерев точність становить 53%.

Як і для MNIST використано AC-GAN для покращеної генерації синтетичного набору. Отримано такі результати:

| Train → Test | Accuracy |
|---|---|
| Real → Real | 93.44% |
| Real → Synthetic | 71.95% |
| Synthetic → Synthetic | 100% |
| Synthetic → Real | 51.56% |
| Real + Synthetic → Real | 93.90% |

*Таблиця 5.5. Результати оцінки навчання і тренування MLP на синтетичному наборі*

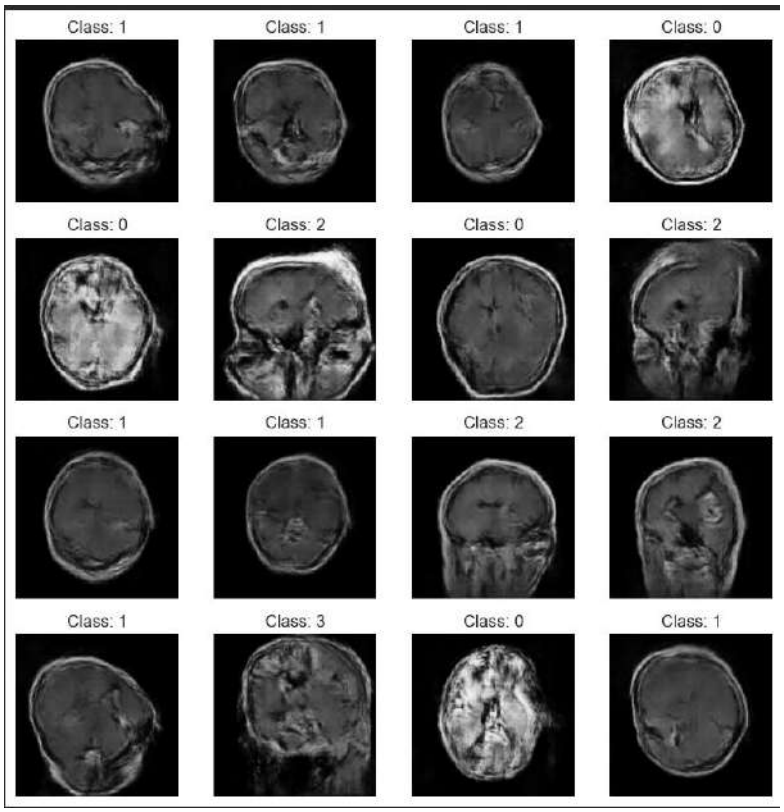



*Рисунок 5.5. Візуалізація синтетичних прикладів*

Отримано результати значно кращі за аналогічні для набору даних на основі правил, проте поступаються аналогічним для MNIST, що свідчить про більш складну природу цього датасету. Також варто відмітити, що точність навчання на комбінації реального і синтетичного датасетів навіть дещо зросла, у порівнянні з навчанням лише на реальних даних. Це свідчить до синтетичні дані у цьому випадку допомагають узагальнити результати.

**Rulefit**

RuleFit також опирається на дуже обмежений набір ознак.

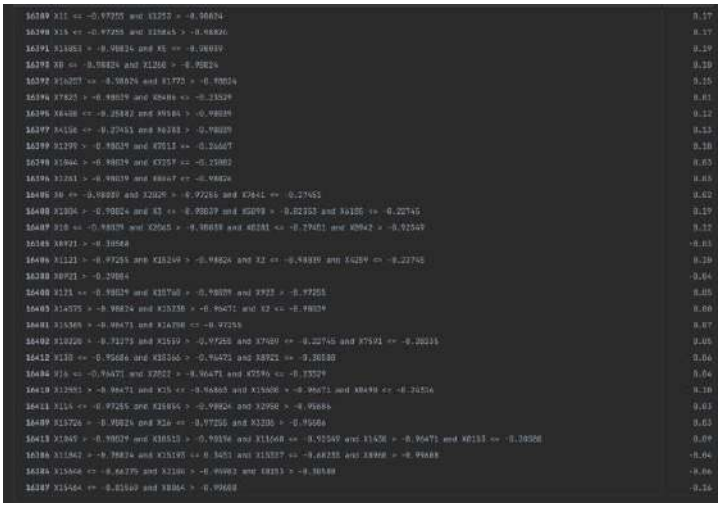


*Рисунок 5.6. Rulefit правила і візуалізація*

Цей метод також присвоює отриманим правилам невеликі значення, через що на мапі важливості навіть важливіші пікселі слабко вирізняються.

5.3. Peturbation методи

Натреновано модель XGBClassifier з точністю 95% для використання у локальних методах.

**Lime**

LIME дає дуже розкидані пояснення, які важко співвіднести з патологією (не видно зв’язної ділянки пухлини). Також усі пояснення мають дуже низьків значення ($\sim 10^{-3}$). Це означає, що усі із цих пікселів виділених як критерії, насправді мало впливають на рішення моделі у локальній апроксимації. Для покращення локальної інтерпретованості можна об’єднати пікселі у групи.

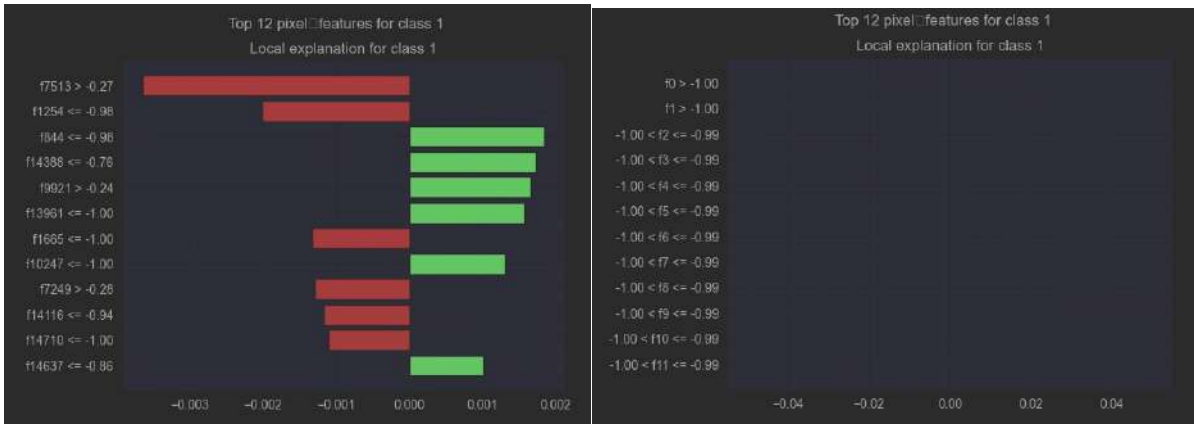



*Рисунок 5.7. Приклади візуалізація LIME*

**SHAP**

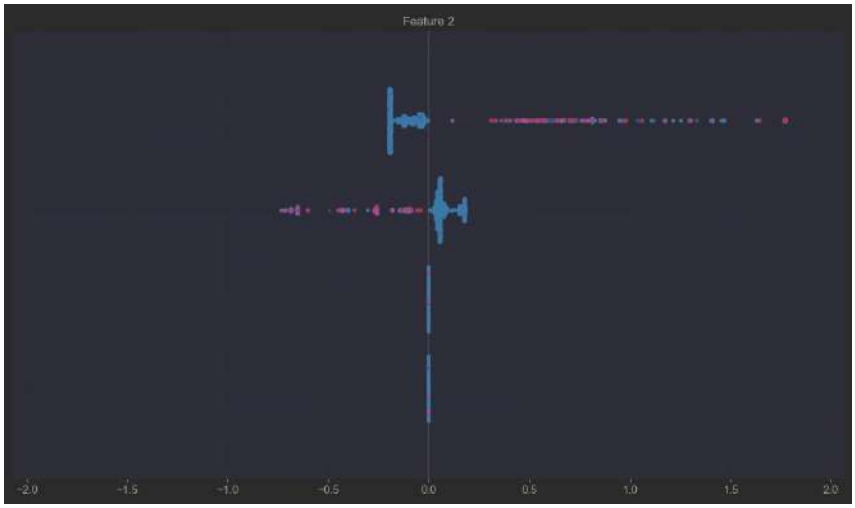



*Рисунок 5.8. Приклади візуалізація SHAP*

5.4. Градієнтні методи.

**Saliency map.**

Для подальшої роботи натреновано згорткову нейромережу з 3 згортковими шарами та MaxPooling і 2 Dense шарами. Точність отриманої моделі становила 92%.

Центральна зона мозку (біля середини зображення) помітно яскравіша — модель звертає на неї підвищену увагу. Цілком ймовірно, що тут і знаходиться патологічний осередок (пухлина), і саме зміни інтенсивності в цій області сигналізують моделі про наявність певного класу.

Низька чутливість у периферії (темні області) означає, що зміни цих пікселів практично не впливають на рішення.

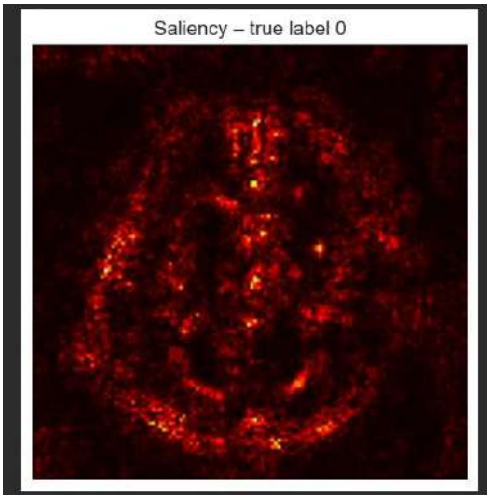



*Рисунок 5.9. Приклади візуалізація Saliency map*

**Integrated Gradients**

Інтегровані градієнти (Integrated Gradients) — це метод інтерпретації нейромереж, який дозволяє зрозуміти, які вхідні пікселі (або ознаки) найбільше впливають на рішення моделі. Ідея полягає в тому, щоб розкласти передбачену ймовірність класу від деякого базового зображення до реального прикладу на суму внесків окремих пікселів.

Δ - це різниця між сумою атрибуцій і різницею вихідних логітів (completeness error). Чим ближче Δ до нуля, тим краще інтегральна апроксимація відповідає вимозі completeness.

У даному випадку Δ становить приблизно 0.4, що є нормальним показником для складного зображення розміром 128 x 128.

Окружність центральної частини мозку чітко простежується в усіх трьох випадках: модель бачить у ній ключову інформацію, швидше за все — пухлинний осередок або характерну аномалію.

Збільшення n_steps робить карту гладшою й згладжує артефакти, проте й вартість обчислень зростає.

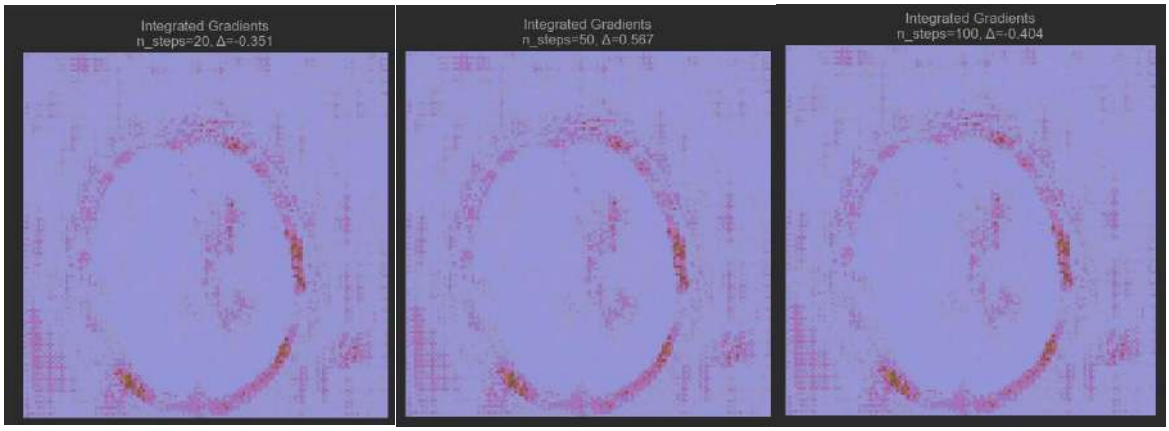



*Рисунок 5.10. Приклади візуалізація інтегральних градієнтів*

**Grad-CAM**

Grad-CAM (Gradient-weighted Class Activation Mapping) — метод візуальної інтерпретації CNN-моделей, який дає змогу локалізувати вхідні зони, що найбільше впливають на конкретне класове передбачення.

Червоні зони означають, що саме ці області сильно зміщують мережу в бік поточного класу (згідно з градієнтною важливістю). Зелені/сині ділянки

мають менший внесок. Grad-CAM показує, де на МР-знімку модель бачить ознаки пухлини.

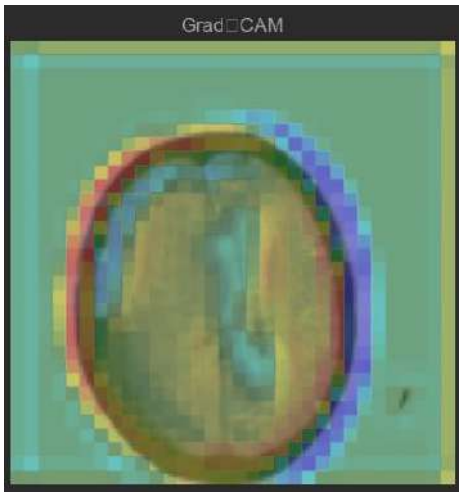



*Рисунок 5.11. Приклади візуалізація Grad-CAM*

5.5. Counterfactual Explanations.

Для цього екперименту використано Dice ML

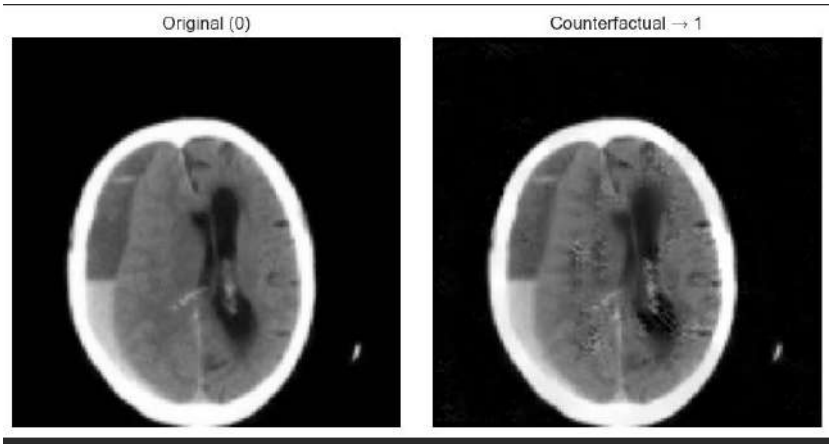



*Рисунок 5.12. Приклади візуалізація Conterfactual explanation*

Внесені мінімальні зміни (зображення дещо затемнене і розмите) у ділянках, де модель очікує знайти ознаки 1 класу. Найкращим і найінформативнішим є порівняння зображень з пухлинами з знімками здорових мізків.

Натомість при використанні VAE-контрафакту отримано не реалістичне зображення, що вказує на значно складнішу природу датасету у порівнянні з MNIST і потребу у покращеній архітектурі і довшому навчанні моделі (модель мала таку ж кількість епох навчання як для MNIST).

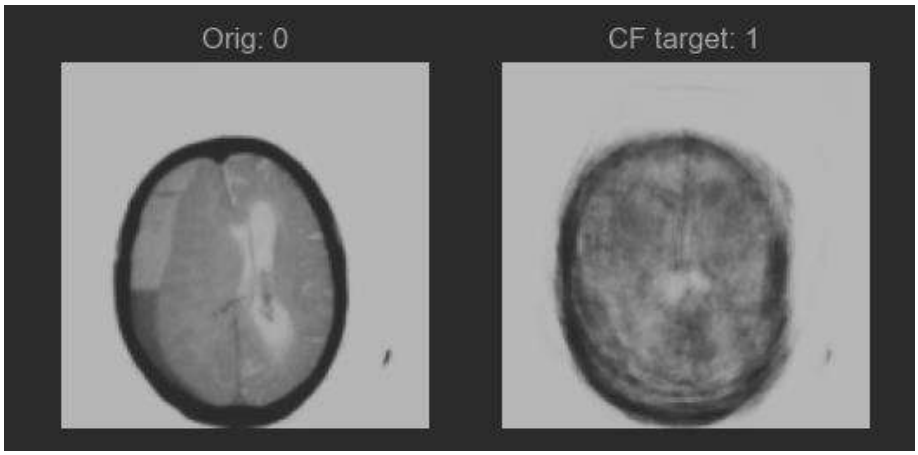



*Рисунок 5.13. Приклади візуалізація Conterfactual explanation з VAE*

5.6. PCA і t-SNE.

З візуалізації PCA та t-SNE для 2 та 3 компонентів видно, що зображення формують досить компактні кластери, що робить використання зменшення розмірності дуже корисним у цьому випадку для кращого застосування інших методів XAI.

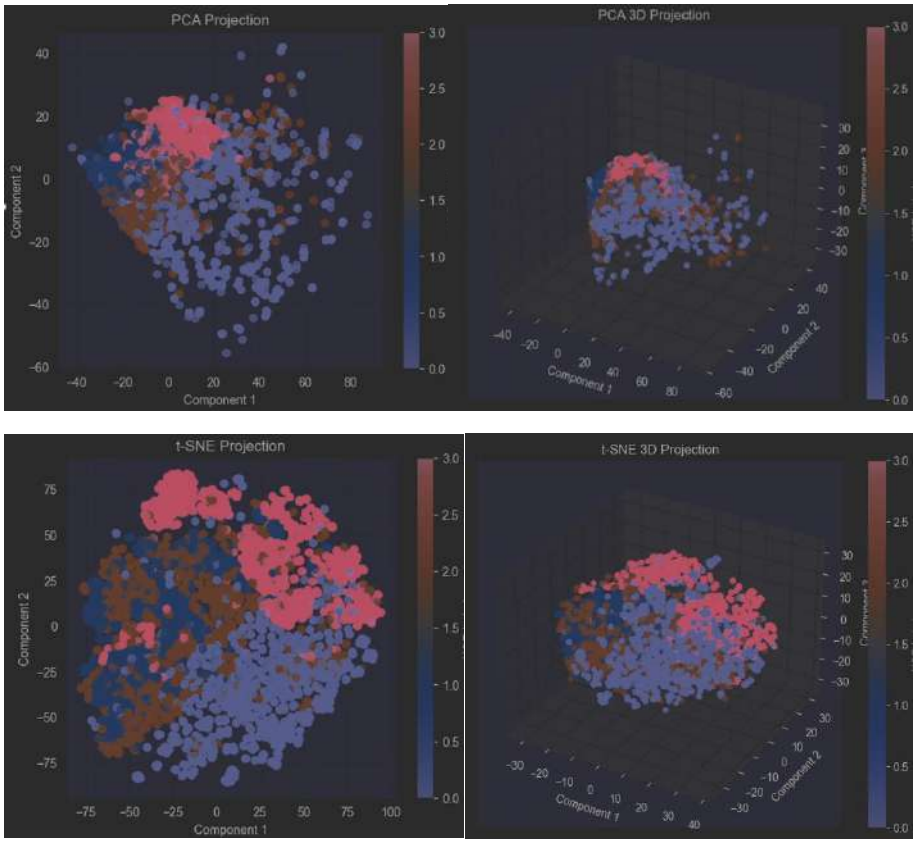



*Рисунок 5.14. Приклади візуалізація PCA та t-SNE*

5.7. Concept explanation (TCAV).

Аналогічно до MNIST для виявлення високорівневих патернів зображень, закодованих кластерами ознак CNN (2 згорткові шари), застосовано TCAV із шістнадцятьма концепціями - від текстури («high_entropy») та яскравості («bright_image» проти «dark_image») до просторової асиметрії («left_heavy» / «top_heavy») та специфічних ознак пухлин («necrotic_core», «ring_enhancement»).

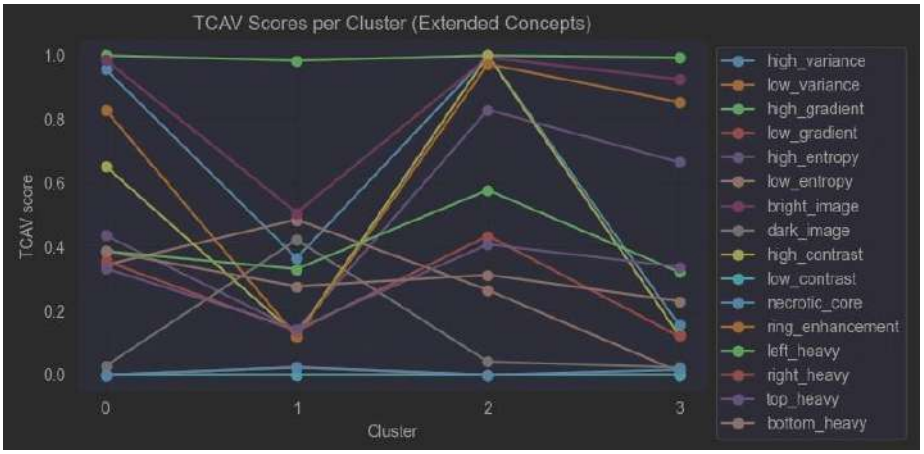



*Рисунок 5.15. Візуалізація оцінки TCAV кожної концепції*

На рис. 5.15. показано оцінку TCAV кожної концепції для кластерів 0-3. Виникає кілька чітких тенденцій:

Майже всі кластери мають оцінку близько 1.0 для high_gradient і high_contrast, що відображає сильну залежність моделі від інформації про межі та границі.

Кластер 2 виділяється рівномірно високими показниками bright_image, high_variance, high_entropy та ring_enhancement, що відповідає його прототипам нормальних, добре контрастних зрізів мозку.

Кластер 1 демонструє підвищену necrotic_core і low_variance, але низьку ring_enhancement, що відповідає великим некротичним гліомам.

Кластер 0 поєднує помірне ring_enhancement з low_entropy і темним зображенням, що відповідає ознакам менінгіоми.

Кластер 3 демонструє найвищі показники за параметрами high_entropy, left_heavy та top_heavy, що відображає скани гіпофіза у його прототипах.

Аналогічно до MNIST створено теплову карту (Рисунку 5.16.), яка об'єднує ці лінійні графіки в сітку «кластер * концепція», що дозволяє легко визначити домінуючі атрибути кожного кластера.

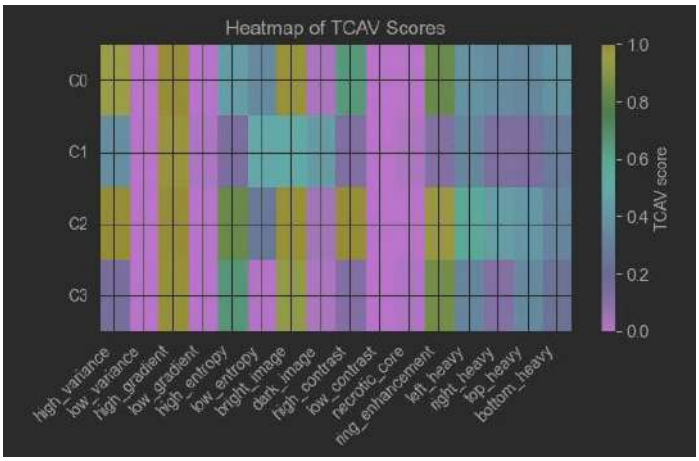



*Рисунок 5.16. Теплова карта оцінки TCAV кожної концепції*

Візуалізація на рівні кластерів.

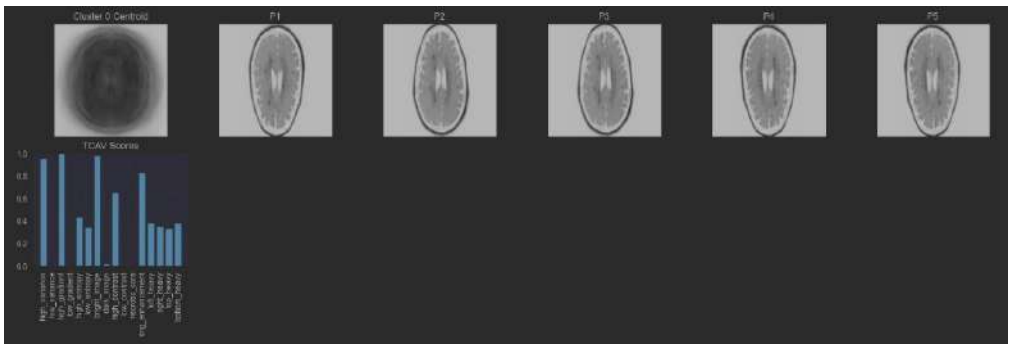



*Рисунок 5.17. Візуалізація кластеру 0 (менінгіома)*

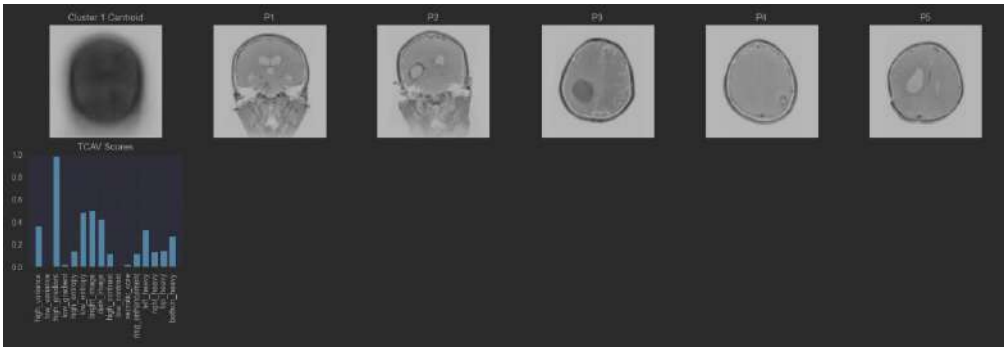



*Рисунок 5.18. Візуалізація кластеру 1 (гліома)*

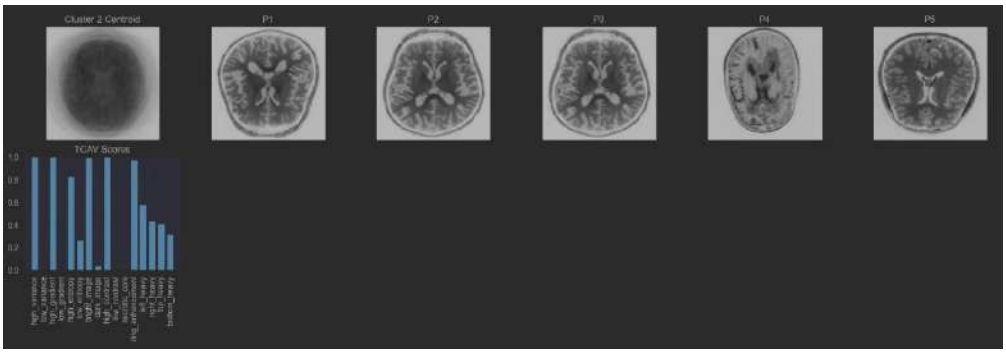



*Рисунок 5.19. Візуалізація кластеру 2 (здорові)*

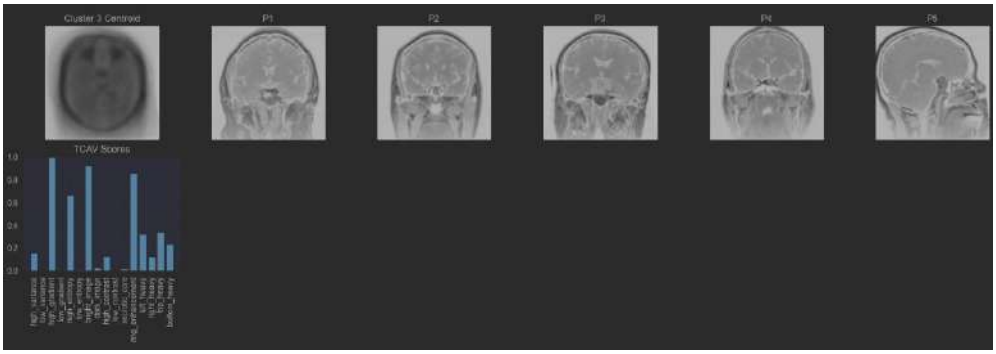



*Рисунок 5.20. Візуалізація кластеру 3 (гіпофіз)*

5.8. Порівняння моделей.

**Точність моделей**

| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Decision Tree | 0.877856 | 0.872967 | 0.870390 | 0.871379 |
| MLP | 0.870328 | 0.865401 | 0.863755 | 0.859459 |
| Deep MLP | 0.856598 | 0.861721 | 0.845891 | 0.845525 |
| XGBoost | 0.954996 | 0.953751 | 0.951176 | 0.951892 |
| Gaussian NB | 0.565980 | 0.549212 | 0.569749 | 0.539014 |
| Bernoulli NB | 0.583524 | 0.582924 | 0.595082 | 0.570557 |
| CNN | 0.928299 | 0.927128 | 0.925349 | 0.925002 |
| LSTM | 0.921434 | 0.916586 | 0.915318 | 0.915909 |

*Таблиця 5.6. Результати точності для моделей*

Найвищу точність демонструє XGBoost (≈0,955) зі збалансованими precision/recall/F1 ≈0,952. Далі йдуть CNN (≈0,928) та LSTM (≈0,921), після них — Decision Tree (≈0,878), MLP (≈0,870) і Deep MLP (≈0,857) – хоча Deep MLP має значно складнішу архітектуру (3 приховані шари з 512, 256, 128

нейронами), модель показує трохи гірше за MLP з (1 прихованим шаром і 128 нейронами на ньому). Gaussian NB (≈0,566) і Bernoulli NB (≈0,584) значно відстають за всіма метриками.

**Native importances**

| Model | Method | Runtime (s) | Comp | MeanImp |
|---|---|---|---|---|
| DecisionTree | Tree FI | 0 | 275 | 0.008904 |
| MLPClassifier | InputWeightImp | 0.0131 | 16384 | 0.011761 |
| DeepMLP | InputWeightImp | 0.0051 | 16384 | 0.023031 |
| XGBoost | XGB FI | 0 | 4252 | 0.005553 |
| GaussianNB | NB ProbDiff | 1874.22 | 1511 | 0.010153 |
| BernoulliNB | NB ProbDiff | 2961.30 | 0 | 0 |

*Таблиця 5.7. Результати Native importance для моделей*

Серед усіх моделей найшвидше оцінюють важливість DecisionTree і XGBoost (Runtime ≈0 с), натомість NB ProbDiff у GaussianNB і особливо в BernoulliNB вимагає понад 30 хв (≈1874 с та 2961 с). За показником Comp (кількість ненульових ознак) найменше ознак залучає DecisionTree (275) і XGBoost (4252), тоді як обидва MLP використовують усі 16384 ознаки, а GaussianNB — 1511. Що стосується середньої важливості (MeanImp), DeepMLP демонструє найвищий внесок (~0,0230), далі йдуть MLPClassifier (~0,0118) і GaussianNB (~0,0102), у DecisionTree — ~0,0089, у XGBoost — ~0,0056, а в BernoulliNB вона взагалі нульова. Таким чином, MLP-моделі дають потужніші оцінки важливості, але за рахунок повного охоплення ознак, тоді як деревовидні алгоритми працюють швидше й із меншою кількістю ознак, хоч демонструють менші середні значення.

**Permutation importance**

| Model | Runtime (s) | Comp | MeanImp |
|---|---|---|---|
| DecisionTree | 701.973 | 96 | 0.011762 |
| MLPClassifier | 2461.970 | 92 | -0.000325 |
| Deep MLP | 16.5825 | 75 | 0.000069 |

| | | | |
|---|---|---|---|
| XGBoost | 5001.37 | 49 | 0.000287 |
| Gaussian NB | 24555.30 | 86 | -0.000481 |
| Bernoulli NB | 5295.29 | 39 | -0.000102 |
| CNN | 5308.02 | 491 | -0.000100 |
| LSTM | 10536.30 | 1285 | 0.000035 |

*Таблиця 5.8. Результати Permutation importance для моделей*

Deep MLP найшвидший (≈16,6 с) при найнижчому Comp (75). DecisionTree працює ≈702 с, має середню важливість ≈0,0118 — найвищу серед усіх. XGBoost і CNN потребують ≈5000 с, їх MeanImp ≈0,00029 і −0,00010 відповідно. LSTM займає ≈10536 с із Comp 1285 і MeanImp ≈0,000035. MLPClassifier (≈2462 с) і Gaussian NB (≈24555 с) показують від’ємні MeanImp, як і Bernoulli NB (≈5295 с), що свідчить про випадкове поліпшення точності при перестановці ознак. У порівняні з результатами отриманими на MNIST помітно значно більший час виконання, що очікувано при збільшені вимірності даних у BrainTumor датасеті. Проте, незважаючи на це, у BrainTumor отримано менше значення Comp майже для всіх моделей, що підтверджує результати отримані при візуалізації ознак з допомогою DecisionTree про невелику кількість важливих ознак.

**Partial dependence**

| Model | Runtime (s) | Comp | MeanRange |
|---|---|---|---|
| Decision Tree | 0.2912 | 100 | 0 |
| MLPClassifier | 0.7125 | 100 | 0.000636 |
| DeepMLP | 0.1737 | 200 | 0.000215 |
| XGBoost | 0.4346 | 100 | 0.000374 |
| Gaussian NB | 4.9700 | 100 | 0.000086 |
| Bernoulli NB | 1.1372 | 100 | 0.000326 |
| CNN | 12.4388 | 1000 | 0.001014 |
| LSTM | 0.9078 | 16384 | 0.000213 |

*Таблиця 5.9. Результати Partial dependence для моделей*

Для BrainTumor час виконання PD залишається доволі низьким: Decision Tree ≈0,29 с, Deep MLP ≈0,17 с, XGBoost ≈0,43 с, MLPClassifier ≈0,71 с, LSTM ≈0,91 с, Gaussian NB ≈4,97 с, Bernoulli NB ≈1,14 с і CNN ≈12,44 с.

Comp для більшості моделей складає 100 (Deep MLP – 200, CNN – 1000, LSTM – 16384). При цьому MeanRange у всіх випадках дуже малий (від 0 у Decision Tree до 0,001014 у CNN), що аналогічно до результів на інших наборах даних свідчить про те, що жодна окрема ознака (піксель) не має суттєвого впливу на прогноз: зміна значення одного пікселя лише незначно змінює ймовірність правильного класу.

**Linear Surrogate**

| Model | Runtime (s) | Comp | Mean $R^2$ | Mean RSME | Mean Coef | CLS Fidelity | Err Coverage |
|---|---|---|---|---|---|---|---|
| DT | 0.1601 | 0 | 1.000 | 0.000 | 0.000 | 0.0003 | 1 |
| MLP | 0.1609 | 84 | 0.891691 | 0.010354 | 0.004215 | 0.0725 | 1 |
| Deep MLP | 0.0584 | 186.7 | 0.529697 | 0.075552 | 0.009281 | 0.0764 | 0.8479 |
| XGBoost | 0.1216 | 94 | 0.551824 | 0.057285 | 0.010860 | 0.01575 | 0.923 |
| Gaussian NB | 0.4242 | 0 | 1.000 | 0.000008 | 0.000025 | 0 | 1 |
| Bernoulli NB | 0.1002 | 0 | 1.000 | 0.000 | 0.000 | 1 | 1 |
| CNN | 2.6666 | 0 | 0.999993 | 0.000 | 0.000 | 0 | 1 |
| LSTM | 0.3105 | 0 | 0.999999 | 0.000008 | 0.000025 | 0 | 1 |

*Таблиця 5.10. Результати Linear surrogate для моделей*

Як і для набора даних MNIST, ці результати свідчать про те, що лінійні сурогати майже не відтворюють класифікацію BrainTumor без попередньої редукції розмірності. Для Decision Tree, Gaussian NB, Bernoulli NB, CNN і LSTM $R^2 \approx 1$ і RMSE≈0, але всі Mean Coef=0 і Cls Fidelity≈0 (окрім Bernoulli NB), тобто сурогат видає константне передбачення й не відповідає оригінальній моделі. MLP, Deep MLP і XGBoost мають трохи ненульових коефіцієнтів (Comp

від 84 до 186, Mean Coef ≈ 0.004–0.011), але їх Cls Fidelity лишається надзвичайно низькою (0.015–0.076), що підтверджує нездатність лінійної моделі передбачити складні, розподілені рішення без зменшення розмірності.

**Counterfactual explanation**

| Model | Runtime (s) | CompMean | Mean ΔProb | Std ΔProb |
|---|---|---|---|---|
| MLP | 0.3975 | 0 | 0.000 | 0.000 |
| Deep MLP | 0.2312 | 0 | 0.000 | 0.000 |
| XGBoost | 0.7474 | 0 | 0.000 | 0.000 |
| Gaussian NB | 0.1431 | 0.20 | 0.200 | 0.400 |
| Bernoulli NB | 0.3409 | 0 | 0.000 | 0.000 |
| CNN | 12.3334 | 0.05 | 0.000374 | 0.001632 |
| LSTM | 113.876 | 0.05 | 0.000933 | 0.004069 |

*Таблиця 5.11. Результати Counterfactual explanation для моделей*

Для BrainTumor MLP, Deep MLP, XGBoost і BernoulliNB не знаходять жодного однопіксельного контрфакту (CompMean = 0, ΔProb = 0), тобто поодинокі пікселі не здатні змінити їхній прогноз. CNN та LSTM дають дуже невеликі зсуви ймовірності (MeanΔProb ≈ 0,00037 у CNN і ≈ 0,00093 у LSTM) за ≈12,3 с і ≈113,9 с відповідно. Це вказує, що більшість складних моделей для BrainTumor потребує масових змін вхідних даних (або не реагує на зміну поодиноких пікселів), а GaussianNB залишається найбільш чутливою до шума.

**Saliency (Gradient based)**

| Model | Runtime (s) | CompMean | Mean ΔProb | Std ΔProb |
|---|---|---|---|---|
| Decision Tree | 0.0214 | 0 | 0.000 | 0.000 |
| MLP | 0.2956 | 98.9 | 0.34593 | 0.383243 |
| Deep MLP | 0.0087 | 200 | 0.18473 | 0.353557 |
| XGBoost | 0.8145 | 15.6 | 0.007552 | 0.009368 |
| Gaussian NB | 0.1779 | 0.20 | 0.000176 | 0.000763 |
| Bernoulli NB | 0.3405 | 0 | 0.000 | 0.000 |
| CNN | 13.1587 | 969.6 | 3.80962 | 5.27847 |

| | | | | |
|---|---|---|---|---|
| LSTM | 120.485 | 15414.8 | 8.19177 | 8.67264 |

*Таблиця 5.12. Результати saliency importance для моделей*

Decision Tree і BernoulliNB не реагують на зміну одного пікселя (CompMean = 0, ΔProb = 0). GaussianNB має мінімальну чутливість (CompMean ≈ 0,2 пікселя, Mean ΔProb ≈ 0,00018) за ≈0,18 с. XGBoost реагує дуже слабко (CompMean ≈ 15,6 пікселів, Mean ΔProb ≈ 0,0076, Std ≈ 0,0094) за ≈0,81 с. Deep MLP обчислює градієнти за ≈0,009 с із помірним рівнем шуму (CompMean = 200, Mean ΔProb ≈ 0,1847, Std ≈ 0,3536). MLP показує більшу чутливість (CompMean ≈ 98,9 пікселя, Mean ΔProb ≈ 0,3459, Std ≈ 0,3832) за ≈0,30 с. CNN і LSTM вимагають суттєво більше часу (≈13,16 с і ≈120,49 с), змінюючи в середньому ≈969,6 й ≈15414,8 пікселів і демонструючи високі Mean ΔProb (≈3,81 і ≈8,19) із дуже великою дисперсією (Std ≈ 5,28 і ≈ 8,67). Це означає, що для глибоких мереж у задачі BrainTumor навіть градієнтні методи виходять надто повільними й надміру шумними, а простіші моделі часто зовсім не мають значущого локального градієнта.

# ВИСНОВКИ

У межах цієї роботи був проведений комплексний аналіз методів Explainable AI (XAI). Це дослідження оцінює і порівнює широко використовувані підходи до пояснюваності (від лінійних моделей і деревних сурогатів до градієнтних і контрфактичних методів) у трьох принципово різних доменах:

- XOR (синтетичний набір даних) тестує прості логічні відносини в умовах шуму.
- MNIST (рукописні цифри) перевіряє пояснення для архітектур класифікації, починаючи від невеликих CNN до RNN і TabTransformers.
- Brain-Tumor MRI (медична візуалізація) оцінює поведінку XAI на об'ємних медичних сканах з чотирма діагностичними класами.

Ця міждоменна оцінка — надає уявлення про надійність і загальність методів, а також показує компроміс між точністю моделей та їх інтерпретованістю. Жоден з підходів не домінує всіма метриками, вибір методу залежить від задачі, типу даних і моделі до якої він застосовується.

Також продемонстровано неможливість існування універсального сильного XAI, який для кожної можливої моделі наведе повне і коректне пояснення.

Проведено перевірку правильності та повноти правил дерев рішень. Отримано правила дерев рішень (та їх ансамблів) для генерації синтетичних наборів даних і перевірено на них простий MLP – результат показав, що навіть дерева з досить високою тоністю (100 % точність для XOR та близько 90% для MNIST та BrainTumor) дають точність тестування MLP лише 50 %–20 % (для XOR аналогічний результат отримуємо при збільшенні кількості шумових ознак), що показує, як обмежену інтерпретованість такого підходу, так і обмеженість сурогатних моделей на їх основі. Такі ж результати дав зворотній тест з навчанням MLP на синтетичних даних з подальшою оцінкою на реальних.

Аналіз лінійних методів, дерев рішень та сурогатних моделей на їх основі, показав, що вони є простими, швидкими й добре інтепретованими для простих і табличних задач, проте слабкими для складних сценаріїв. Для їх ефективного використання на складних наборах даних потрібне застовування методів зменшення розмірності, проте у такому випадку страждає інтерпретованість через втрату зв'язку з реальними вхідними ознаками.

Проведено детальне порівняння методів на основі видхидень та градієнтів – продемонстровано, що LIME/Anchors та Permutation Importance працюють у всіх доменах, але страждають від високих обчислювальних витрат та нестабільності у високовимірні, корельованих даних. Saliency, Integrated Gradients та Grad-CAM виявляють чіткі локальні атрибуції на зображеннях, але їх рівень шуму та інтерпретованість сильно залежать від складності мережі та даних.

Пряме порівняння DiCE та VAE-базованих контрфактичних методів демонструє компроміси між швидкістю та реалістичністю на MNIST та Brain-Tumor. Також згенеровано еталонні зображення для малих та глибоких CNN та ResNet, показуючи, як складність моделі впливає на рівень абстракції та інтерпретованість моделі. Для невеликих моделей (або певних частин моделі) візалізація еталонних результатів дозволяє приблизно зрозуміти, на що спирається модель при ухвалені рішень.

Оцінено TCAV на кластеризованих ембадінгах – шляхом кластеризації ембедінгів MLP/CNN та присвоєння зрозумілих для людини понять (наприклад, «петлі», «вертикальні лінії»), можна кількісно оцінити, які семантичні поняття керують кожним кластером, поєднуючи зрозумілі людині концепції та латентні представлення. Використання таких методів є дуже корисним для генерації зрозумілих людям пояснень, проте вони теж є досить дорогими з точки зору обчислень і потребують означення необхідних концепції людиною (хоча певна робота у напрямку автоматизованого створення концепцій була пророблена [24][25]).

Також розроблено прототип, який дозволяє в режимі реального часу зважувати кілька пояснювачів (LIME, SHAP, Permutation Importance, PDP тощо), що дає можливість одночасно порівнювати локальні та глобальні пояснення в одному інтерактивному інтерфейсі користувача.

# СПИСОК ЛІТЕРАТУРИ


1. A. Adadi and M. Berrada, “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI),” *IEEE Access*, vol. 6, pp. 52138–52160, 2018. https://doi.org/10.1109/ACCESS.2018.2870052
2. A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, et al., “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI,” *Information Fusion*, vol. 58, pp. 82–115, 2020. https://doi.org/10.1016/j.inffus.2019.12.012
3. G. Schwalbe and B. Finzel, “A Comprehensive Taxonomy for Explainable Artificial Intelligence: A Systematic Survey of Surveys on Methods and Concepts,” *Data Mining and Knowledge Discovery*, vol. 37, no. 1, pp. 1–59, 2023. https://doi.org/10.1007/s10618-022-00867-8
4. J. A. Nelder and R. W. M. Wedderburn, “Generalized Linear Models,” Journal of the Royal Statistical Society: Series A (General), vol. 135, no. 3, pp. 370–384, 1972. https://doi.org/10.2307/2344614
5. T. Hastie and R. Tibshirani, “Generalized Additive Models,” Statistical Science, vol. 1, no. 3, pp. 297–318, 1986. https://doi.org/10.1214/ss/1177013604
6. L. Breiman, J. Friedman, R. Olshen & C. Stone, Classification and Regression Trees, Wadsworth, 1984
7. J. Craven & J. Shavlik, “Extracting Tree-Structured Representations of Trained Networks,” in Advances in Neural Information Processing Systems (NIPS), 1996.
8. J. H. Friedman & B. Popescu, “Predictive Learning via Rule Ensembles,” The Annals of Applied Statistics, vol. 2, no. 3, pp. 916–954, 2008.
9. M. T. Ribeiro, S. Singh & C. Guestrin, ““Why Should I Trust You?” Explaining the Predictions of Any Classifier,” in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144. https://doi.org/10.1145/2939672.2939778
10. M. T. Ribeiro, S. Singh & C. Guestrin, “Anchors: High-Precision Model-Agnostic Explanations,” in Proceedings of the AAAI Conference on Artificial

Intelligence, vol. 32, no. 1, 2018, pp. 1527–1535. https://ojs.aaai.org/index.php/AAAI/article/view/11491

11. N. Goldstein, A. Kapelner, J. Bleich & E. Pitkin, "Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation," J. Comput. Graph. Statist., vol. 24, no. 1, pp. 44–65, 2015. https://doi.org/10.1080/10618600.2014.907095
12. D. W. Apley & J. Zhu, "Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models," J. R. Stat. Soc. B, vol. 82, no. 4, pp. 1059–1086, 2020. https://doi.org/10.1111/rssb.12377
13. K. Simonyan, A. Vedaldi & A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," Workshop at Int. Conf. on Learning Representations, San Diego, CA, USA, 2014. https://arxiv.org/abs/1312.6034
14. J. T. Springenberg, A. Dosovitskiy, T. Brox & M. Riedmiller, "Striving for Simplicity: The All Convolutional Net," Workshop at Int. Conf. on Learning Representations, San Diego, CA, USA, 2015. https://arxiv.org/abs/1412.6806
15. M. Sundararajan, A. Taly & Q. Yan, "Axiomatic Attribution for Deep Networks," in Proc. International Conf. on Machine Learning, Sydney, Australia, Jul. 2017, pp. 3319–3328. http://proceedings.mlr.press/v70/sundararajan17a.html
16. R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh & D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," in 2017 IEEE Int. Conf. on Computer Vision (ICCV), Venice, Italy, Oct. 2017, pp. 618–626. https://doi.org/10.1109/ICCV.2017.74
17. R. K. Mothilal, A. Sharma & C. Tan, "Explaining Machine Learning Classifiers Through Diverse Counterfactual Explanations," in Proc. 2020 Conf. on Fairness, Accountability, and Transparency (FAccT '20), Barcelona, Spain, Jan. 27–30, 2020, pp. 607–617. https://doi.org/10.1145/3351095.3372850

18. D. P. Kingma & M. Welling, “Auto-Encoding Variational Bayes,” in 2nd Int. Conf. on Learning Representations (ICLR), Banff, Canada, Apr. 2014. https://arxiv.org/abs/1312.6114
19. I. T. Jolliffe, Principal Component Analysis, 2nd ed., Springer Series in Statistics, Springer-Verlag, New York, 2002. https://doi.org/10.1007/b98835
20. L. J. P. van der Maaten & G. E. Hinton, “Visualizing High-Dimensional Data Using t-SNE,” J. Mach. Learn. Res., vol. 9, no. nov, pp. 2579–2605, 2008. http://www.jmlr.org/papers/v9/vandermaaten08a.html
21. Z. Liu, Y. Wang, S. Vaidya, F. Rüehle, J. Halverson, M. Soljačić, T. Y. Hou & M. Tegmark, “KAN: Kolmogorov–Arnold Networks,” in International Conference on Learning Representations (ICLR) 2025, accepted Feb. 9, 2025, arXiv:2404.19756. https://arxiv.org/abs/2404.19756
22. C. Chen, O. Li, A. J. Barnett, J. Su, and C. Rudin, “This Looks Like That: Deep Learning for Interpretable Image Recognition,” NeurIPS (Demonstrations), 2019. https://arxiv.org/abs/1806.10574
23. B. Kim, M. Wattenberg, J. Gilmer, C. J. Cai, J. Wexler, F. B. Viégas, and R. Sayres, “Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors,” in Proc. 35th ICML, vol. 80, pp. 2668–2677, 2018. https://proceedings.mlr.press/v80/kim18d.html
24. A. Ghorbani, J. Wexler, J. Y. Zou, and B. Kim, “Towards Automatic Concept-based Explanations,” in Advances in Neural Information Processing Systems, vol. 32, pp. 9273–9282, 2019. https://arxiv.org/abs/1902.03129
25. R. Achtibat, M. Dreyer, I. Eisenbraun, S. Bosse, T. Wiegand, W. Samek, and S. Lapuschkin, “From Attribution Maps to Human-Understandable Explanations through Concept Relevance Propagation,” in Proc. NeurIPS, 2022. https://arxiv.org/abs/2206.03208
26. T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, 2nd ed., Springer, 2009.
27. N. X. Vinh, J. Epps, and J. Bailey, “Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for

Chance," J. Mach. Learn. Res., vol. 11, pp. 2837–2854, 2010. https://jmlr.org/papers/v11/vinh10a.html

28. P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," Comput. Appl. Math., vol. 20, pp. 53–65, 1987. https://doi.org/10.1016/0377-0427(87)90125-7
29. S.-M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in Proc. NeurIPS, 2017, pp. 4768–4777. https://arxiv.org/abs/1705.07874
30. M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic Attribution for Deep Networks," in Proc. ICML, 2017, pp. 3319–3328. https://arxiv.org/abs/1703.01365
31. Simezu. (2023). brain-tumour-MRI-scan [Dataset]. Hugging Face. https://huggingface.co/datasets/Simezu/brain-tumour-MRI-scan
32. Rice, H. G. (1953). Classes of recursively enumerable sets and their decision problems. Transactions of the American Mathematical Society, 74(2), 358–366.
33. Hyötyniemi, Heikki (1996). "Turing machines are recurrent neural networks". Proceedings of STeP '96/Publications of the Finnish Artificial Intelligence Society: 13–24.