Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет Комп'ютерних технологій і систем
(назва факультету)

Комп'ютерні інформаційні технології
(повна назва кафедри)

Пояснювальна записка
до кваліфікаційної роботи
магістра
(ступінь вищої освіти)

на тему: «Аналіз алгоритмів машинного навчання в ML.NET»

за освітньою програмою Інженерія програмного забезпечення
зі спеціальності: 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Виконав: студент групи: ПЗ2121

(підпис студента) / Андрій БУРЦЕВ /
(Ім'я ПРІЗВИЩЕ)

Керівник: (підпис) / Олександр ІВАНОВ /
(посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: (підпис) / Світлана ВОЛКОВА /
(посада, Ім'я ПРІЗВИЩЕ)

Консультанти:

Економіка (підпис) / Миколай ГНЕНИЙ /
(назва розділу) (посада, Ім'я ПРІЗВИЩЕ)

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

Студент (підпис)

Дніпро – 2022 рік

Міністерство освіти і науки України
Український державний університет науки і технологій

«Факультет Комп'ютерних технологій і систем»

(назва факультету)

«Комп'ютерні інформаційні технології»

(повна назва кафедри)

Пояснювальна записка

до кваліфікаційної роботи

магістра

(ступінь вищої освіти)

на тему: «Аналіз алгоритмів машинного навчання в ML.NET»

за освітньою програмою _____

зі спеціальності: 121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Виконав: студент групи:

_____ / Андрій БУРЦЕВ /

(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ / Олександр ІВАНОІ /

(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ / Світлана ВОЛКОВА /

(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Консультанти:

_____ _____ / /

(назва розділу) (підпис) (посада, Ім'я ПРІЗВИЩЕ)

_____ _____ / /

(назва розділу) (підпис) (посада, Ім'я ПРІЗВИЩЕ)

_____ _____ / /

(назва розділу) (підпис) (посада, Ім'я ПРІЗВИЩЕ)

_____ _____ / /

(назва розділу) (підпис) (посада, Ім'я ПРІЗВИЩЕ)

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

Студент _____

(підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

«Computer technologies and systems»
(faculty)

«Computer information technology»
(department)

Explanatory Note
to Master's Thesis
Master's
(higher education degree)

on the topic: «Analysis of machine learning algorithms in ML.NET »

according to educational curriculum

in the Speciality: «121 Software engineering»
(speciality and its code)

Done by the student of the group: / Andriy BURTSEV /
(name, surname)

Scientific Supervisor: / Oleksandr IVANOV /
(position, name, surname)

Normative controller: / Svitlana VOLKOVA /
(position, name, surname)

Supervisors

/ /
(Chapter title heading) (position, name, surname)

/ /
(Chapter title heading) (position, name, surname)

/ /
(Chapter title heading) (position, name, surname)

/ /
(Chapter title heading) (position, name, surname)

Dnipro – 2022

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет:
Кафедра:
Рівень вищої освіти:
Освітня програма:
Спеціальність:
(шифр та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри доцент Горячкін В. М.
(підпис) (Ім'я ПРІЗВИЩЕ)
Дата

З А В Д А Н Н Я

на кваліфікаційну роботу магістр
(ступінь вищої освіти)

студенту Бурцев Андрій Володимирович
(Прізвище, Ім'я По батькові)

1. Тема роботи: «Аналіз алгоритмів машинного навчання в ML.NET»

Керівник роботи: Іванов Олександр Петрович
(Прізвище, Ім'я, По батькові, науковий ступінь, вчене звання)

затверджені наказом від "___"_____2022 р. № _____

2. Строк подання студентом роботи: __.__.2022 р.
3. Вихідні дані до роботи:

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
4.1 Аналітична частина:

4.2 Основна частина:

4.3 Охорона праці та захист навколишнього середовища:

4.4 Економічна частина:

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав (підпис консультанта, дата)	Завдання прийняв (підпис студента, дата)
Техніко-економічні розрахунки	доц. Гнений М.В.		

КАЛЕНДАРНИЙ ПЛАН

КАЛЕНДАРНИЙ ПЛАН

№ зп	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вступ		
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	02.09.22 – 15.09.22	Від 70 джерел
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	16.09.22 – 10.10.22	
4	Постановка задачі, технічне завдання	11.10.22 – 17.10.22	30%
5	Техніко-економічні показники	18.10.22 – 19.10.22	
6	Розробка інструментальних засобів дослідження	20.10.22 – 07.11.22	
7	Виконання досліджень	08.11.22 – 14.11.22	60%
8	Оформлення тез доповідей	15.11.22 – 18.11.22	
9	Оформлення статті у фаховий журнал	19.11.22 – 22.11.22	
10	Оформлення пояснювальної записки	23.11.22 – 28.11.22	
11	Розробка демонстраційних матеріалів	29.11.22 – 05.12.22	100%
12	Подання кваліфікаційної роботи до кафедри		
13	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії		

Студент _____ Андрій БУРЦЕВ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ Олександр ІВАНОІ
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Кваліфікаційна робота складається з 110 стор., 30 рис., 5 додатків.

Об'єктом дослідження є алгоритми навчання машинного навчання.

Мета роботи: дослідити роботу алгоритмів у відкритому фреймворку ML.NET. Провести дослідження областей застосування та якість роботи алгоритмів. Класифікація типів завдань і формування рекомендацій використання доцільного алгоритму навчання до кожного типу. Проаналізувати чому даний алгоритм є кращим для вирішення конкретної задачі.

Методика дослідження: теоретичний аналіз наукових і літературних джерел, формально-логічний аналіз, узагальнення, абстрактування, конкретизація, моделювання та спостереження.

Перелік ключових слів: машинне навчання, алгоритми, ML.NET.

ЗМІСТ

ВСТУП

Початок роботи з машинним навчанням сьогодні включає круту криву навчання. При побудові моделей машинного навчання потрібно з'ясувати, яке завдання машинного навчання вибрати для сценарію (наприклад, класифікацію або регресію), як перетворити дані у формат, який можуть зрозуміти алгоритми ML (наприклад, текстові дані, числові вектори), і налаштувати ці алгоритми ML задля забезпечення найкращої продуктивності. Запитання "Який алгоритм машинного навчання використовувати?" все досі залишається відкритим. Вибір алгоритму залежить від обсягу, якості та природи даних. Він залежить від того, як розпоряджатись результатом. Він залежить від того, як з алгоритму були створені інструкції для комп'ютера, що реалізує його, та часу.

Коли говорять про машинне навчання, то часто мають на увазі штучні нейронні мережі (ІНП), що знову стали популярними, і глибоке навчання, які є моделями машинного навчання, тобто окремими випадками методів розпізнавання образів, дискримінантного аналізу, методів кластеризації тощо.

Машинне навчання є не лише математичною, а й прикладною, інженерною дисципліною. Практично жодне дослідження в галузі машинного навчання не обходиться без подальшого тестування на реальних даних для перевірки практичної працездатності методу, що розробляється.

Завдяки машинному навчанню програміст не зобов'язаний писати інструкції, що враховують усі можливі проблеми та містять усі рішення. Натомість у комп'ютер (або окрему програму) закладають алгоритм самостійного знаходження рішень шляхом комплексного використання статистичних даних, з яких виводяться закономірності та на основі яких робляться прогнози.

Технологія машинного навчання на основі аналізу даних бере початок у 1950 році, коли почали розробляти перші програми для гри в шашки. За минулі десятиліття загальний принцип не змінився. Зате завдяки вибуховому зростанню обчислювальних потужностей комп'ютерів багаторазово ускладнилися закономірності та прогнози, створювані ними, і розширилося коло проблем та завдань, які вирішуються з використанням машинного навчання.

Щоб запустити процес машинного навчання, для початку необхідно завантажити в комп'ютер DataSet (деяка кількість вихідних даних), у яких алгоритм вчитиметься обробляти запити. Наприклад, можуть бути фотографії собак і котів, на яких вже є мітки, що позначають, до кого вони відносяться. Після процесу навчання програма вже сама зможе розпізнавати собак і котів на нових зображеннях без змісту міток. Процес навчання триває і після виданих прогнозів, що більше даних ми проаналізували програмою, то більше точно вона розпізнає необхідні зображення.

Завдяки машинному навчанню комп'ютери вчяться розпізнавати на фотографіях та малюнках не лише обличчя, а й пейзажі, предмети, текст та цифри. Щодо тексту, то й тут не обійтися без машинного навчання: функція перевірки граматики зараз присутня у будь-якому текстовому редакторі і навіть у телефонах. Причому враховується як написання слів, а й контекст, відтінки сенсу та інші тонкі лінгвістичні аспекти. Більше того, вже існує програмне забезпечення, здатне без участі людини писати статті новин.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ДОСЛІДЖЕННЯ ЗА НАУКОВИМИ ЛІТЕРАТУРНИМИ ДЖЕРЕЛАМИ

1.1 Машинне навчання (ML)

Машинне навчання (ML) — це вивчення алгоритмів, що здатні самовдосконалюватися з досвідом. Якщо більш складніше описувати ML, то можна сказати шо це застосування алгоритмів для автоматичного знаходження закономірностей в даних і використання їх для прийняття великої кількості однотипних рішень, для яких певний відсоток помилок є допустимими. ML є підрозділ більш широкої галузі досліджень, що стосуються штучного інтелекту (AI). Задача машинне навчання — надати змогу комп’ютерам краще виконувати певні завдання по мірі того як вони отримують більше матеріалів для аналізу, без прямого алгоритмічного програмування.

ML перетинається з Data Mining (коли за допомогою цих алгоритмів ми намагаємося з бази даних витягнути певні інсайти) і зі статистичним аналізом (з якого, власне, ML походить), але є чимось більшим за все перелічене.

Машинне навчання буває трьох видів. Перший — «із вчителем», коли наш алгоритм має приклад результатів, яких ми від нього очікуємо, і підбирає правило, за яким із вхідних даних отримуються вихідні результати. Другий — «без вчителя», «ось тобі дані, сам з них щось витягни». Третій вид — «з підкріпленням», це схоже на навчання живої нейромережі, наприклад, собаки, коли в алгоритму є умови максимізації виграшу і він адаптується таким чином, щоб за будь-яких умов власний виграш максимізувати.

На даний час основним і найкрутішим видом ML є глибинне навчання (deep learning), коли ми складаємо кілька алгоритмів разом і те, що є вихідними даними для «нижнього шару» алгоритмів, стає вхідними даними для «верхнього». Так, наприклад, працює будь-яке серйозне розпізнавання образів.

1.2 Алгоритм

Алгоритм - це математичний опис, який використовується для створення моделі. Різні алгоритми пропонують моделі з різними характеристиками. Один алгоритм можна застосувати до різних завдань. *Навчальний алгоритм = алгоритм + завдання*.

Для ознайомлення приведемо приклад популярних алгоритмів: RandomizedPcaTrainer, OlsTrainer, LightGbmMulticlassTrainer, AveragedPerceptronTrainer.

1.3 Модель машинного навчання

Модель — це об'єкт (файл), що містить перетворення для отримання прогнозів на основі даних, що надано. Модель машинного дає можливість автоматично вчитися і вдосконалюватися на основі власного досвіду без явної участі людини. Основна мета моделі полягає в тому, щоб розробник зміг

використати переваги алгоритмів штучного інтелекту та машинного навчання для отримання додаткових конкурентних переваг.

Моделювання – це етап відбувається навчання моделі. Навчання моделей машинного навчання відбувається ітераційно – пробуються різні моделі, перебираються гіперпараметри, порівнюються значення обраної метрики та вибирається найкраща комбінація.

Всі моделі машинного навчання поділяються на навчання з учителем (supervised) та без учителя (unsupervised). У першу категорію входять регресійна та класифікаційна модель (рис.1).

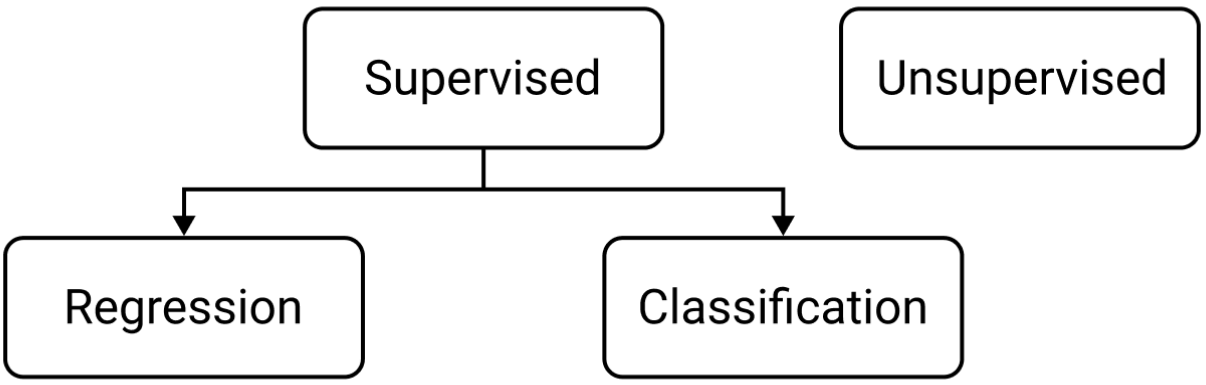

Рис.1 Фундаментальна сегментація моделей машинного навчання

Навчання з учителем є вивченням функції, яка перетворює вхідні дані у вихідні на основі прикладів пар введення-виводу. Це тип машинного навчання, в якому дані, які засовуються в модель маркуються. Маркування означає, що результат спостереження (тобто ряд даних) відомий. Наприклад, якщо модель намагається передбачити, чи піде завтра в ранці дощ чи ні, можуть бути такі змінні, як температура повітря, вологість, швидкість вітру. Якщо дані позначені, то змінна матиме значення 1, якщо дощ пішов, і значення 0, якщо дощу не було.

У регресійній моделі висновок є безперервним. Найпоширеніших типів регресійних моделей: лінійна регресія, дерево рішень, випадковий ліс, нейронна мережа.

Часто лінійна регресія стає першою моделлю машинного навчання, яку люди вивчають. Пов'язано це з тим, що її алгоритм (простіше кажучи рівняння) досить просто зрозуміти, використовуючи тільки одну змінну x (рис.2). Лінійна регресія чимось нагадує логістичну регресію, але використовується, коли цільова змінна - безперервна, а це означає, що вона може набувати практично будь-яке числове значення. Насправді, будь-яка модель з безперервною цільовою змінною може бути класифікована як регресія. Прикладом безперервної змінної може бути вартість продажу автомобіля.

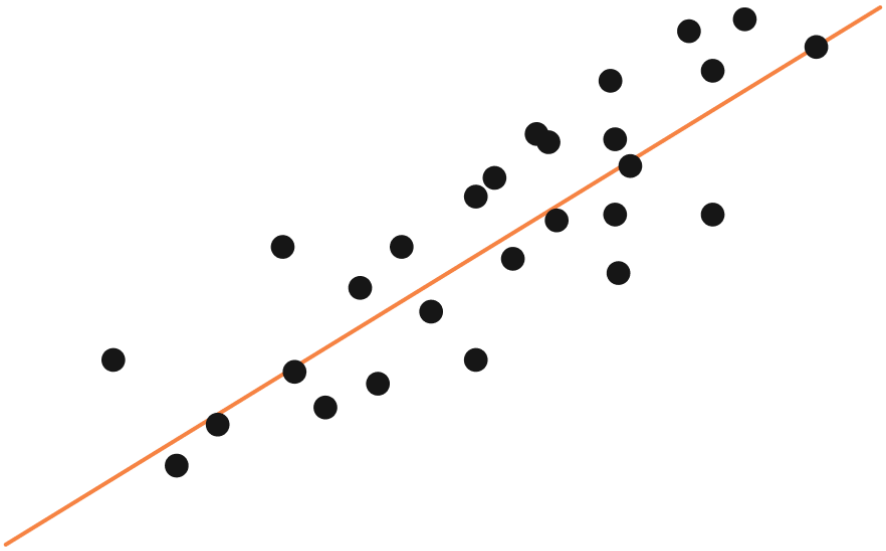

Рис.2 Графік лінійної регресії

Дерево рішень - популярна модель, що використовується в дослідженні операцій, стратегічному плануванні та машинному навчанні. Кожен прямокутник вище називається вузлом. Що більше вузлів, то більш точним буде дерево рішень. Останні вузли, у яких приймається рішення, називаються листям дерева. Дерева рішень інтуїтивні та прості у створенні, проте не надають точних результатів (рис.3).

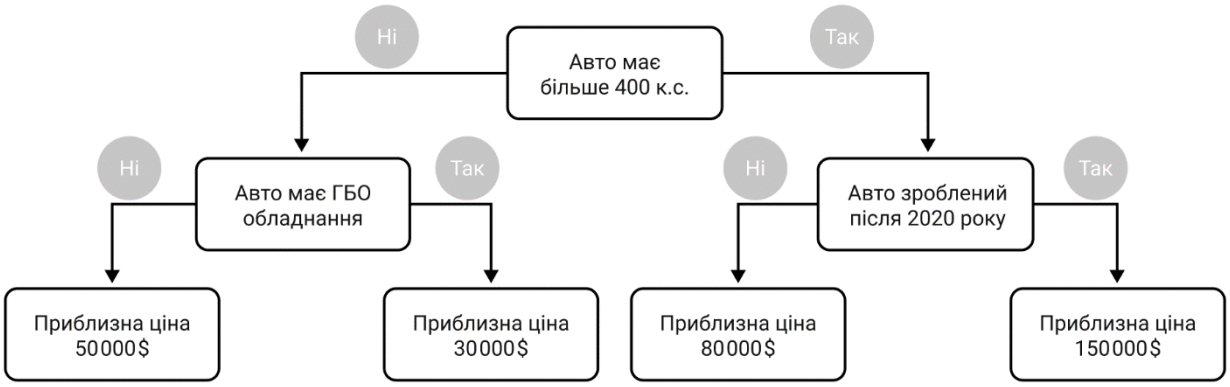

Рис.3 Графік дерева рішень

Випадковий ліс — це техніка методів, заснована на деревах рішень. Випадкові ліси включають створення кількох дерев рішень з використанням початкових наборів даних та випадковий вибір піднабору змінних на кожному етапі (рис.4). Потім модель вибирає моду (значення, яке зустрічається найчастіше) з усіх прогнозів кожного дерева рішень. Наприклад, є одне дерево рішень (друге), яке передбачає 0, але якщо покладатися на моду всіх 3 дерев, прогнозоване значення дорівнюватиме 1. У цьому полягає перевага випадкових лісів.

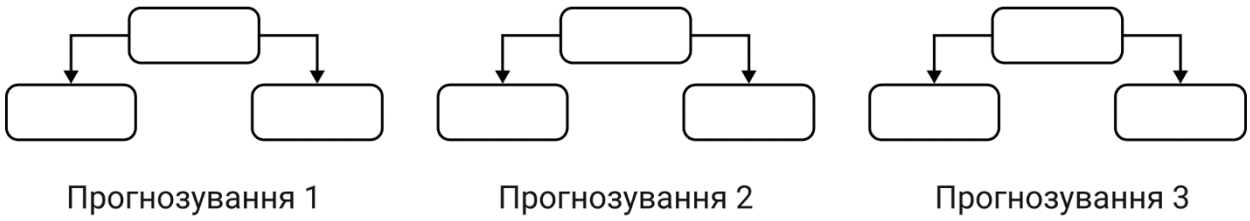

Рис.4 Графічне зображення випадкового лісу

Нейронна мережа це багатошарова модель, влаштована за системою людського мозку. Як і нейрони в нашому мозку, кола вище представляють вузли. Зеленим позначений шар вхідних даних, чорним приховані шари, а зеленим шар вихідних даних. Кожен вузол у прихованих шарах представляє функцію, якою проходять вхідні дані, що призводять до виходу синіх колах (рис.5).

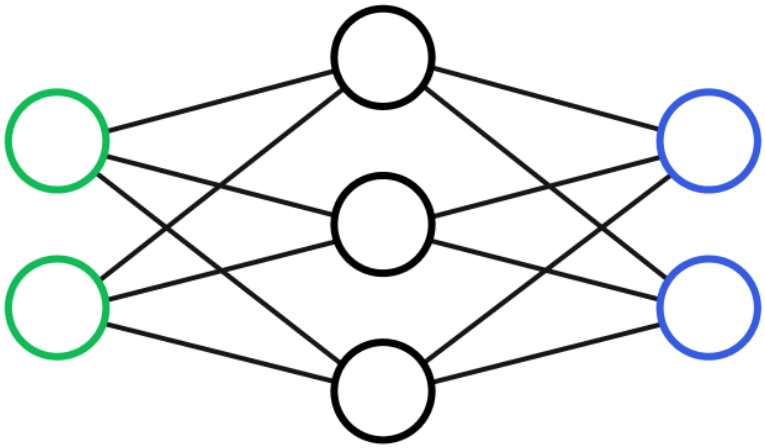

Рис.5 Графічне зображення випадкового лісу

У класифікаційних моделях висновок дискретним. Найпоширеніші типи класифікаційних моделей: логістична регресія, метод опорних векторів, метод Байєса.

Логістична регресія використовується на вирішення проблеми класифікації. Це означає, що цільова змінна (яка передбачається) складається з категорій. Ці категорії можуть бути так/ні, або щось подібне до числа від 1 до 10, яке позначає задоволеність клієнта. Модель логістичної регресії використовує рівняння для створення кривої з даними, а потім використовує цю криву, щоб спрогнозувати результати нового спостереження (рис.6).

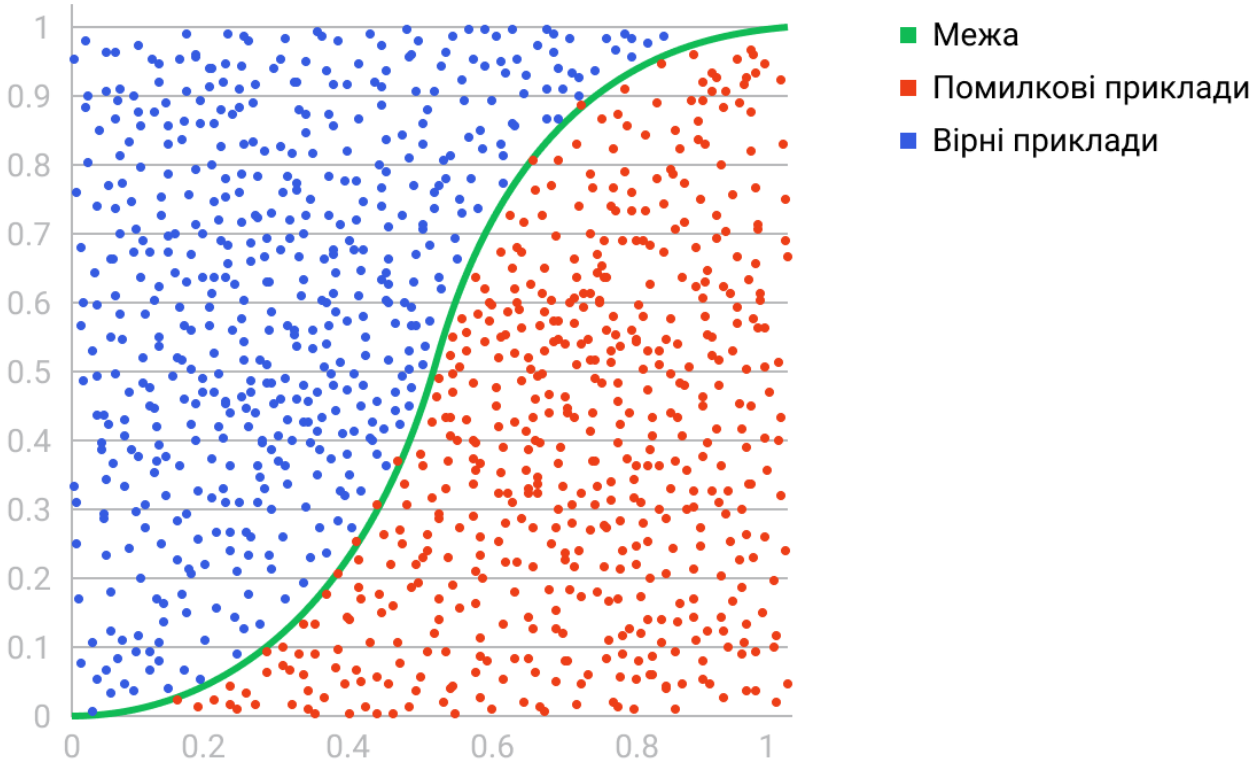

Рис.6 Графік логістичної регресії

Завдання лінійної регресії полягає у знаходженні лінії, яка найкраще відповідає даним. Розширення лінійної регресії включають множинну лінійну регресію (наприклад, пошук найбільш підходящої площини) і поліноміальну регресію (наприклад, пошук найбільш підходящої кривої).

Метод опорних векторів — це класифікаційний метод навчання з учителем, досить складний, але інтуїтивний на базовому рівні. Припустимо, що є два класу даних. Метод опорних векторів знаходить гіперплощину чи межу між двома класами даних, яка максимізує різницю між двома класами. Є безліч площин, які можуть поділити два класи, але тільки одна з них максимізує різницю або відстань між класами (рис.7).

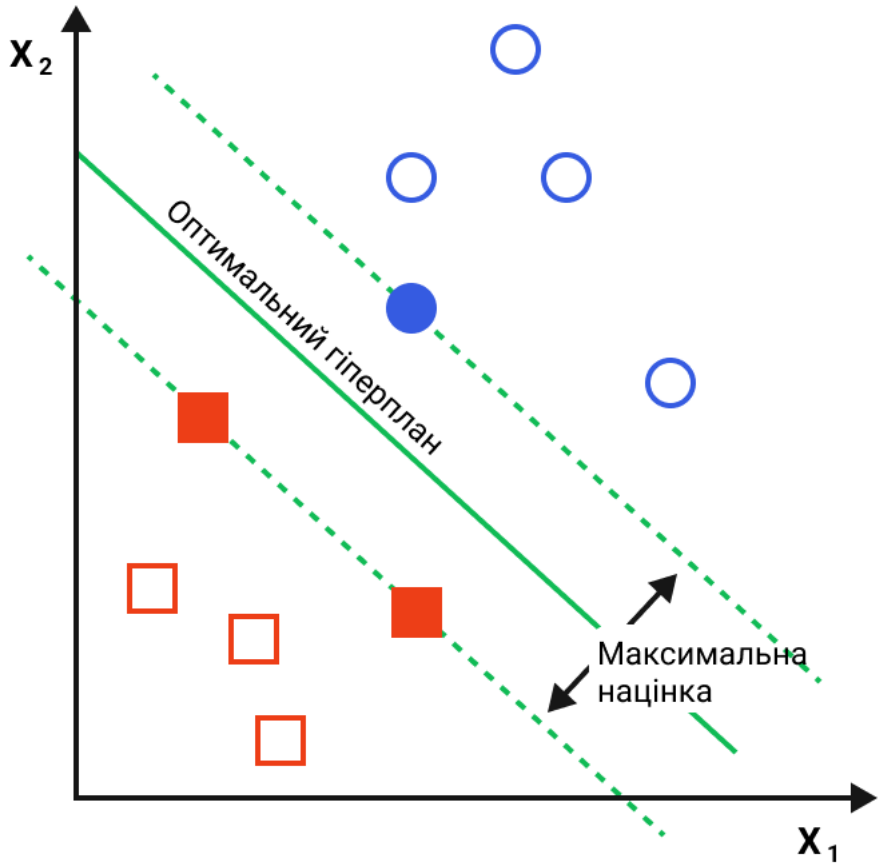

Рис.7 Графік опорних векторів

На відміну від навчання з учителем, навчання без вчителя використовується для того, щоб зробити висновки та знайти шаблони з вхідних даних без посилань на маркований результати. У навчанні без вчителя, не можна знати піде завтра в ранці дощ чи ні – тільки комп'ютер може знайти закономірності за допомогою моделі, щоб вгадати, що сталося або передбачити, що станеться. Два основних методи, що використовуються у навчанні без вчителя, включають кластеризацію та зниження розмірності.

Кластеризація - це техніка навчання без вчителя, яка включає групування або кластеризацію точок даних (рис.8). Найчастіше вона використовується для сегментації споживачів, виявлення шахрайства та класифікації документів. Поширені методи кластеризації включають кластеризацію за допомогою k-середніх, ієрархічну кластеризацію, зсув середнього значення та кластеризацію на основі щільності.

Коли використовується K кластеризація, необхідно почати з припущення, що у датасеті є K кластерів. Оскільки невідома, скільки груп насправді у даних, треба спробувати різні значення K і за допомогою візуалізації та метрик зрозуміти, яке значення K підходить. Метод K середніх найкраще працює із круговими кластерами однакових розмірів. Цей алгоритм спочатку вибирає найкращі точки даних K, щоб сформувати центр кожного кластера K. Потім, він повторює 2 наступні кроки для кожної точки:

1. Надає точку даних найближчому центру кластера.
2. Створює новий центр, взявши середнє значення всіх точок даних із цього кластера.

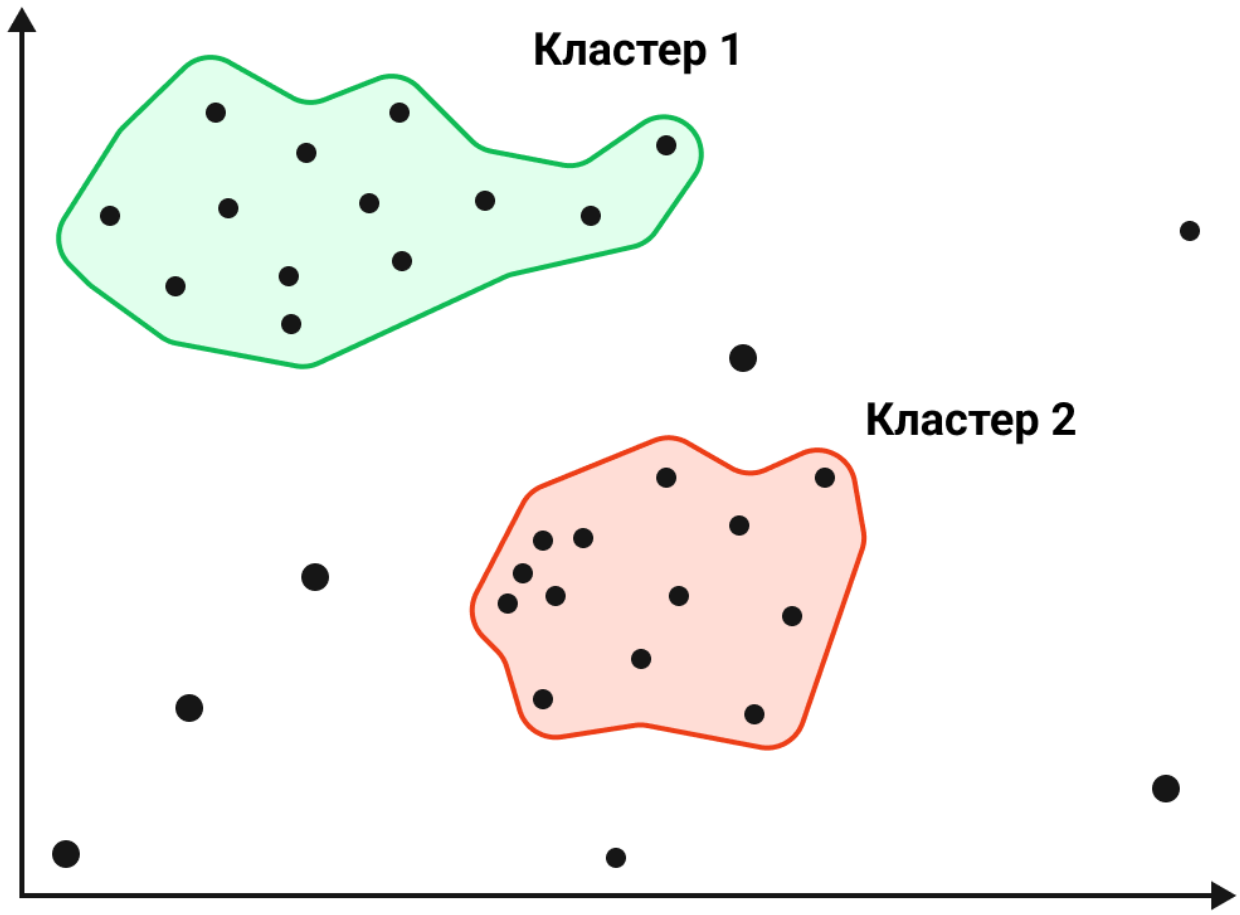

Рис.8 Графік кластеризації

1.4 Основні поняття ML.NET

ML.NET - це безкоштовна бібліотека машинного навчання від Microsoft, створена для розширення екосистеми .NET можливостями машинного навчання. Ця бібліотека надає можливість тривіальних інтеграцій з існуючими ML-моделями та пропонує чимало інструментів для створення власних моделей. Використовуючи цю функцію, можливо отримувати автоматичні прогнози на основі даних, доступних у додатку. Програми машинного навчання використовують для прогнозування закономірності, знайдені у даних, не будучи явно запрограмованими. Створену модель можна додати до програми та використовувати її для отримання прогнозів.

В основі ML.NET є модель машинного навчання. Ця модель визначає кроки, які потрібно виконати для отримання прогнозів на основі вхідних даних. За допомогою ML.NET можливо навчити модель, вказавши відповідний алгоритм, а також імпортувати попередньо навчені моделі TensorFlow і ONNX.

ML.NET працює у Windows, Linux та macOS з .NET Core або у Windows з .NET Framework. 64-розрядна версія підтримується на всіх платформах. 32-розрядна версія підтримується у Windows, за винятком функцій, пов'язаних із TensorFlow, LightGBM та ONNX.

За допомогою ML.NET можна прогнозувати наступні типи:

— класифікація/категоризація;

— регресія/прогнозування безперервних значень;

— виявлення аномалій;

— рекомендації;

— тимчасові ряди/послідовності;

— класифікація зображень.

1.5 Завдання у ML.NET

Двійкова класифікація – це завдання контрольованого машинного навчання, яке прогнозує розподіл елементів даних за двома категоріями. На вхід алгоритму класифікації подається набір прикладів з мітками, кожна з яких є цілим числом 0 або 1. Результатом роботи алгоритму двійкової класифікації є класифікатор, який вміє прогнозувати клас для нових екземплярів без мітки. Для отримання найкращих результатів навчання двійкової класифікації навчальні дані мають бути збалансовані, число позитивних та негативних навчальних даних має бути однаковим. Відсутні значення потрібно обробити до навчання. Навчити модель двійкової класифікації можливо використовуючи такі алгоритми:

- AveragedPerceptronTrainer;
- SdcaLogisticRegressionBinaryTrainer;
- SdcaNonCalibratedBinaryTrainer;
- SymbolicSgdLogisticRegressionBinaryTrainer;
- LbfgsLogisticRegressionBinaryTrainer;
- LightGbmBinaryTrainer;
- FastTreeBinaryTrainer;
- FastForestBinaryTrainer;
- GamBinaryTrainer;
- FieldAwareFactorizationMachineTrainer;
- PriorTrainer;
- LinearSvmTrainer.

Багатокласова класифікація - завдання контрольованого машинного навчання, яке прогнозує розподіл екземплярів даних за декількома категоріями. На вхід алгоритму класифікації подається набір прикладів із мітками. Кожна мітка зазвичай запускається як текст. Потім вона запускається через TermTransform, який перетворює її на тип ключа (числовий). Результатом роботи алгоритму класифікації є класифікатор, який вміє прогнозувати клас нових екземплярів без

мітки. Навчити модель багатокласової класифікації можливо використовуючи такі алгоритми:

— LightGbmMulticlassTrainer;

— SdcaMaximumEntropyMulticlassTrainer;

— SdcaNonCalibratedMulticlassTrainer;

— LbfgsMaximumEntropyMulticlassTrainer;

— NaiveBayesMulticlassTrainer;

— OneVersusAllTrainer;

— PairwiseCouplingTrainer.

Регресія - завдання контрольованого машинного навчання, яке прогнозує значення мітки набору пов'язаних компонентів. Мітка тут може набувати будь-якого значення, а не просто вибирається з кінцевого набору значень, як у задачах класифікації. Алгоритми регресії моделюють залежність міток від зв'язаних компонентів, щоб визначити закономірності зміни міток за різних значень компонентів. На вхід алгоритму регресії подається набір прикладів з відмітними знаками значень. Результатом роботи алгоритму регресії є функція, яка вміє прогнозувати значення мітки для будь-якого нового набору вхідних компонентів. Навчити модель регресії можливо використовуючи такі алгоритми:

— LbfgsPoissonRegressionTrainer;

— LightGbmRegressionTrainer;

— SdcaRegressionTrainer;

— OlsTrainer;

— OnlineGradientDescentTrainer;

— FastTreeRegressionTrainer;

— FastTreeTweedieTrainer;

— FastForestRegressionTrainer;

— GamRegressionTrainer.

Кластеризація - завдання неконтрольованого машинного навчання, яке групує окремі екземпляри даних у кластери зі схожими характеристиками. Кластеризацію можна також використовувати для визначення набору даних зв'язків, які неможливо логічно відстежити переглядом або спостереженням даних. Вхідні та вихідні дані для алгоритму кластеризації залежать від обраного методу. Можливо вибрати підхід на основі розповсюдження, центроїду, можливості підключення або густини. ML.NET в даний час підтримує тільки кластеризацію методом К-середніх на основі центроїду (KMeansTrainer).

Виявлення аномалій - це завдання створює модель виявлення аномалій з допомогою аналізу основних компонентів (PCA). Виявлення аномалій на основі PCA дозволяє створювати моделі у сценаріях, де легко отримати дані для навчання з одного класу, такого як допустимі транзакції, проте отримати достатню вибірку аномальних значень важко. Загальноприйнятий метод машинного навчання, PCA, часто використовується в розвідувальному аналізі даних, оскільки він розкриває внутрішню структуру даних і пояснює їх варіативність. PCA виконується шляхом аналізу даних із декількома змінними. Він виконує пошук кореляції між змінними та визначає поєднання значень, які найкраще фіксують відмінності результатів. Ці об'єднані значення компонентів використовуються створення більш компактних компонентів, званих головними компонентами.

Так як аномалії за визначенням досить рідкісні події, зі складанням репрезентативної вибірки даних, що використовуються для моделювання, можуть бути труднощі. Алгоритми, включені до цієї категорії, спеціально розроблені для вирішення основних проблем розробки та навчання моделей з використанням незбалансованих наборів даних. Навчити модель виявлення аномалій, можливо використовуючи алгоритм – RandomizedPcaTrainer.

Ранжування – це завдання створює засіб ранжирування з урахуванням набору прикладів з мітками. Цей набір прикладів складається з груп екземплярів, які можуть бути оцінені із заданими критеріями. Мітки ранжування для кожного екземпляра - {0, 1, 2, 3, 4}. Засіб ранжирування навчається ранжувати нові групи

екземплярів з невідомими оцінками кожного екземпляра. Алгоритми навчання ранжування ML.NET засновані на ранжируванні машинного навчання. Навчити модель ранжирування можливо використовуючи такі алгоритми:

— LightGbmRankingTrainer;

— FastTreeRankingTrainer.

Рекомендація – це завдання дозволяє створити список рекомендованих продуктів або служб. ML.NET використовує факторизацію матриці (MF), алгоритм спільної фільтрації для рекомендацій за наявності історичних даних про рейтинг продуктів у каталозі. Навчити модель рекомендацій можливо використовуючи алгоритм – MatrixFactorizationTrainer.

Прогнозування – це завдання прогнозування використовує попередні дані часових рядів, щоб робити прогнози майбутньому поведінці. Сценарії, які застосовуються до прогнозування, включають прогнозування погоди, сезонні прогнози продажу та діагностичне обслуговування. Навчити модель прогнозування можливо використовуючи алгоритм – ForecastBySsa.

Класифікація зображень – це завдання контрольованого машинного навчання, яке прогнозує розподіл зображень за декількома категоріями. Вхідні дані – це набір помічених прикладів. Кожна мітка зазвичай запускається як текст. Потім вона запускається через TermTransform, який перетворює її на тип ключа (числовий). Результатом роботи алгоритму класифікації зображень є класифікатор, який можна використовуватиме прогнозування класу нових зображень. Завдання класифікації зображень – це тип класифікації за кількома класами. Навчити модель класифікації зображень можливо використовуючи алгоритм – ImageClassificationTrainer.

Виявлення об'єктів – це завдання контрольованого машинного навчання, що використовується для прогнозування категорії зображення, а також надає прямокутник, що обмежує, для категорії всередині зображення. Замість класифікації одного об'єкта зображення, виявлення об'єктів може визначити кілька

об'єктів зображення. Навчання моделі виявлення об'єктів в даний час доступне тільки в Model Builder за допомогою машинного навчання Azure.

1.6 Огляд аналогів

TensorFlow – це бібліотека, створена Google, яка використовується для розробки систем, що використовують технології машинного навчання. Ця бібліотека включає реалізацію безлічі потужних алгоритмів, розрахованих на вирішення поширених завдань машинного навчання. Над цим інструментом розробники Google «билися» дуже багато часу, проте оприлюднили та зробили його загальнодоступним лише у 2015 році. Його попередником був проект DistBelief. З тих пір бібліотека TensorFlow досить швидко набула популярності серед розробників, які використовують машинне навчання у власних роботах. Бібліотека включає кілька важливих особливостей:

— Містить у своєму складі функцію, яка досить просто взаємодіє зі складними математичними виразами та обчисленнями.
— Забезпечує програмування глибоких нейронних мереж та машинного навчання різними алгоритмами.
— Містить функцію, яка обчислює різні набори даних, що дозволяє використовувати цей інструмент у різних сферах людської діяльності.

Бібліотека TensorFlow складається з багатьох інших дрібніших бібліотек. При необхідності до неї можна підключити інші бібліотеки, що не вистачає. Саме тому TensorFlow називають "екосистемою машинного навчання".

Розробники бібліотеки TensorFlow прагнули до того, щоб вона була б гнучкою, ефективною, розширюваною, переносимою. В результаті їй можна користуватися в різних обчислювальних середовищах - від тих, які формуються мобільними пристроями, до середовищ, представлених величезними кластерами. Бібліотека дозволяє швидко готувати до реальної роботи навчені моделі, що усуває необхідність створення особливих реалізацій моделей для комерційних цілей.

Ця бібліотека, завдяки спільним зусиллям усіх тих, хто працює над нею, підходить для вирішення завдань різних масштабів. Від тих, які постають перед самостійним розробником, до тих, які постають перед стартапами і навіть перед великими компаніями на зразок Google. З того моменту, як ця бібліотека стала відкритою, з листопада 2015 року, вона стала однією з найцікавіших бібліотек машинного навчання. Її дедалі частіше використовують під час проведення досліджень, розробки реальних додатків, під час навчання. TensorFlow постійно покращується, її постійно постачають чимось новим, оптимізують. Крім того, зростає і спільнота, яка сформована навколо цієї бібліотеки. Це дозволяє говорити про те, що вона має всі шанси на стабільний розвиток.

Загалом, якщо є задум використовувати можливості машинного навчання у власному програмному продукті, тоді можна скористатися можливостями TensorFlow. При цьому можна не замислюватися про те, чи підійде або не підійде.

Плюси:

— Відмінний фреймворк для створення нейронних мереж, які працюватимуть у комерції.
— Бере він оптимізацію ресурсів для обчислень.
— Велике товариство.
— За рахунок популярності вища ймовірність, що подібну проблему вже вирішили.

Мінуси:

— Складний у використанні та освоєнні.
— Необхідно постійно контролювати відео пам’ять, що використовується.
— Має свої стандарти.
— Погана документація.

Проекти, які використовують фреймворк TensorFlow:

— DeepSpeech - система розпізнавання мови.

— Mask R-CNN – модель, яка генерує обмежувальні рамки та маски сегментації для кожного об'єкта на зображенні.
— BERT - передбачена нейронна мережа, що використовується для вирішення задач обробки природної мови.

Keras – це бібліотека для Python, що дозволяє легко та швидко створювати нейронні мережі. Вона сумісна з TensorFlow, Theano, Microsoft Cognitive Toolkit та MXNet. Перші дві платформи найбільш використовуються розробки алгоритмів глибокого навчання, але досить складні в освоєнні. Keras ж, навпаки, є чудовим варіантом для тих, хто тільки починає вивчення нейронних мереж. Keras працює поверх TensorFlow, CNTK і Theano і надає інтуїтивно зрозумілий API, який, на думку наших інженерів, поки що є найкращим у своєму роді. Автор бібліотеки Франсуа Шолле, інженер Google, розробив Keras для того, щоб максимально спростити процес створення нейронних мереж. Наголос був на розширюваності, модульності та мінімалізмі з підтримкою Python.

Фреймворк націлений на оперативну роботу з нейромережами і є компактним, модульним та розширюваним. Підходить для невеликих проектів, тому що створити щось масштабне на ньому складно і він явно програватиме у продуктивності нейромереж того ж TensorFlow.

Keras був розроблений для користувачів, а не для машин – він використовує особливі методи: пропонує узгоджений та простий API, мінімізує кількість дій користувача, необхідних для вирішення найпоширеніших завдань. У разі виникнення помилок завжди можна звернутися на підтримку зворотного зв'язку. Можливість легко додавати нові класи, модулі та функції робить Keras відмінним засобом для проведення різноманітних досліджень. Всі моделі були написані мовою програмування Python, тому код завжди компактний і легко читається.

Плюси:

— Зручний у використанні.

— Легкий в освоєнні.

— Швидкорозвивається фреймворк.

— Гарна документація.

— Вбудований у TF.

Мінуси:

— Не підходить для великих проектів.

Проекти, які використовують фреймворк Keras:

— Mask R-CNN – модель, яка генерує обмежувальні рамки та маски сегментації для кожного об'єкта на зображенні

— Face Classification - алгоритм для розпізнавання осіб у режимі реального часу та класифікації емоцій та статі.

— YOLOv3 - нейронна мережа для виявлення об'єктів у режимі реального часу.

2 ОБҐРУНТУВАННЯ МЕТОДІВ ДОСЛІДЖЕННЯ

2.1 Обґрунтування доцільності розробки

За результатами аналізу доступних для вивчення джерел було зроблено висновок про недостатність актуального тематичного матеріалу з досліджень та методології проведення часових замірів, точності та доцільності використання конкретного алгоритму машинного навчання, що у свою чергу, підтвердило доцільність виконання даного дослідження.

2.2 Проблема вибору алгоритму навчання

В рамках наведених технік існує більше різних методів машинного навчання, починаючи від простих і інтуїтивних, закінчуючи настільки складними, що розбиратися в їх математиці досліднику не має сенсу.

Важливо розуміти, що серед десятків різних методів немає тих, які найкраще підходять всім завдань: прості моделі можуть працювати недостатньо точно, а складні – бути дуже важкими і складними у реалізації. Вибір методу під конкретне завдання майже завжди здійснюється методом спроб та помилок.

Існує таке поняття, як "No Free Lunch" теорема. Її суть полягає в тому, що немає такого алгоритму, який був би найкращим вибором для кожного завдання, що стосується навчання з учителем.

Наприклад, не можна сказати, що нейронні мережі завжди працюють краще, ніж дерева рішень, і навпаки. На ефективність алгоритмів впливає безліч факторів на кшталт розміру та структури набору даних. Тому доводиться пробувати багато різних алгоритмів, перевіряючи ефективність кожного на тестовому наборі даних, а потім вибирати найкращий варіант. Звісно ж, потрібно вибирати серед алгоритмів, що відповідають завданню.

Алгоритми машинного навчання можна описати як навчання цільової функції f, яка найкраще співвідносить вхідні змінні X та вихідну змінну Y: $Y = f(X)$. Ми не знаємо, що собою являє функція f. Адже якби знали, то використовували її прямо, а не намагалися навчити за допомогою різних алгоритмів.

2.3 Точність алгоритму навчання

2.3.1 Оцінки для двійкової класифікації.

Точність це частка правильних прогнозів за допомогою перевірочного набору даних. Це співвідношення числа правильно вгаданих та загальної кількості прикладів вхідних даних. Ця метрика працює добре, якщо існує аналогічна кількість вибірок, що належать кожному класу. Що ближче до 1,00, то краще. Точне значення 1,00 говорить про проблеми (зазвичай це витік міток та цілей, перенавчання або тестування за допомогою навчальних даних). Якщо тестові дані не збалансовані (більшість екземплярів відноситься до одного з класів), набір даних малий або оцінка підходить до значення 0,00 або 1,00, то точність не відображає фактичну ефективність класифікатора і потрібно перевірити додаткові метрики.

aucROC або площа під кривою оцінює площу під кривою, створену підсумовуванням частот істинно позитивних результатів і помилково позитивних результатів. Що ближче до 1,00, то краще. Для того, щоб модель була допустима, її значення має бути більше 0,50. Модель зі значенням AUC не вище 0,50 не застосовується.

aucPR або площа під кривою "точність - повнота": зручна міра успішного прогнозу, коли класи розрізняються (вкрай нерівномірно розподілені набори даних). Що ближче до 1,00, то краще. Високий рівень оцінки, близький до 1,00, показує, що класифікатор повертає точні результати (висока точність), а також повертає більшу частину всіх позитивних результатів (високий рівень повноти).

Показник F1 також називається збалансованою F оцінкою або F мірою. Це середнє гармонійне значення точності та повноти. Показник F1 корисний у тому випадку, якщо необхідно знайти баланс між точністю та повнотою. Що ближче до 1,00, то краще. Показник F1 досягає кращого значення 1,00 і гіршого - 0,00. Він повідомляє, наскільки точним є класифікатор.

2.3.2 Оцінки для багатокласової класифікації.

Мікросередня точність агрегує вклади всіх класів для обчислення середнього показника. Це частка екземплярів, які модель правильно спрогнозувала. Мікросереднє не враховує членство у класі. По суті, кожна пара "приклад - клас" однаково бере участь у метриці точності. Що ближче до 1,00, то краще. У задачі багатокласової класифікації підтримка мікроточності краще, ніж при точності макросів, якщо передбачається, що існує незбалансований клас (наприклад, у вас може бути кілька прикладів, відмінних від інших класів).

Макросередня точність - це середня точність на рівні класу. Обчислюється точність кожного класу, а макроточність — це середнє цих значень. Власне, кожен клас однаково бере участь у цій метриці точності. Міноритарним класам призначається та ж вага, що й більшим. Макросереднє значення метрики призначає одну й ту саму вагу для кожного класу незалежно від того, скільки екземплярів

містить клас набору даних. Що ближче до 1,00, то краще. Вона обчислює метрики незалежно кожного класу і потім бере середнє значення (тому всі класи враховуються однаково).

Зазвичай мікроточність краще узгоджується з бізнес-потребами прогнозів машинного навчання. Якщо треба обрати одну метрику для визначення якості завдання багатокласової класифікації, слід вибрати мікроточність. Макроточність переважує невеликі команди у цьому прикладі; невеликі команди, які отримують лише 10 звернень на рік, враховуються нарівні з великою командою із 10 000 звернень на рік. У цьому випадку мікроточність краще корелює з бізнес-потребами: "скільки часу та грошей можна заощадити, автоматизуючи процес маршрутизації запитів до служби".

Логарифмічна втрата вимірює продуктивність моделі класифікації, де значення ймовірності прогнозу становить від 0,00 до 1,00. Втрата збільшується при відхиленні прогнозованої ймовірності від фактичного значення мітки. Чим ближче до 0,00, тим краще. У ідеальної моделі значення втрати дорівнює 0,00. Мета нашої моделі машинного навчання – звести до мінімуму це значення.

Редукцію логарифмічних втрат можна інтерпретувати як перевагу класифікатора над випадковим прогнозом. Має значення в діапазоні від -inf до 1,00 де 1,00 - ідеальний прогноз, а 0,00 - середні прогнози. Наприклад, якщо значення дорівнює 0,20, воно може інтерпретуватися як "ймовірність правильного прогнозу на 20% краще за випадкове вгадування".

2.3.3 Оцінки для завдань регресії та рекомендації.

Завдання регресії та рекомендації прогнозують число. У разі регресії число може бути будь-яким вихідним властивістю, яке впливають вхідні властивості. У разі рекомендації число зазвичай є значенням оцінки (наприклад, від 1 до 5) або рекомендацією "так/ні" (представлену 1 і 0 відповідно).

R-квадрат (R2) або коефіцієнт детермінації означає сукупну прогнозуючу здатність моделі в діапазоні від -inf до 1,00. 1,00 означає, що є ідеальний збіг, але

збіг може бути поганим, тому оцінки можуть бути негативними. Оцінка 0,00 означає, що модель прогнозує очікуване значення для позначки. Негативне значення R2 показує, що збіг відповідає тенденціям даних, а модель працює гірше, ніж випадкове припущення. Це можливе лише при використанні моделей нелінійної регресії або обмеженої лінійної регресії. R2 вимірює, наскільки реальні значення даних близькі до прогнозованих. Чим ближче до 1,00, тим вища якість. Тим не менш, іноді низькі значення (наприклад, 0,50) можуть бути повністю нормальні або достатні для сценарію, тоді як високі значення не завжди підходять і можуть бути підозрілими.

Абсолютна втрата, або середня абсолютна похибка (MAE) вимірює, наскільки прогнози близькі до фактичних результатів. Це середнє значення всіх помилок моделі, де помилка моделі - абсолютна відстань між значенням прогнозованої позначки і значенням правильної позначки. Ця помилка прогнозу обчислюється кожної записи перевірочного набору даних. Нарешті, середнє значення обчислюється всім зареєстрованих абсолютних помилок. Чим ближче до 0,00, тим вища якість. Середня абсолютна похибка використовує ту ж шкалу даних, що і дані, що вимірюються (тобто не нормалізується до певного діапазону). Абсолютна втрата, квадратична втрата та середньоквадратична втрата можуть використовуватися тільки для порівняння моделей для одного набору даних або наборів даних з аналогічним розподілом значень міток.

Квадрати з втратою даних або середні квадрати (MSE), також звані середнім квадратним відхиленням (МСД), повідомляють про те, наскільки близька Лінія регресії з набором значень тестів, приймаючи відстані від точок до лінії регресії (ці відстані є помилками E) та повертають їх . Квадрат дає більше ваги більшим відстаням. Цей показник завжди є невід'ємним, значення ближче до 0,00 краще. Залежно від даних може бути неможливим отримати дуже мале значення для середньоквадратичної помилки.

RMS-втрата (або середньоквадратична помилка (RMSE) ; також звана середньоквадратичним відхиленням, RMSD) вимірює різницю між значеннями,

прогнозованими моделлю, і значеннями, що спостерігаються в середовищі, що моделюється. RMS-втрата – це квадратний корінь із квадратної втрати; вона використовує ті ж одиниці, що й мітка, аналогічно абсолютній втраті, але надає більшої ваги більшої різниці. Середньоквадратична помилка зазвичай використовується в кліматології, прогнозуванні та регресійному аналізі для перевірки експериментальних результатів. Цей показник завжди є невід'ємним, значення ближче до 0,00 краще. RMSD — це міра точності порівняння помилок прогнозування різних моделей на конкретному наборі даних, а чи не між наборами даних, оскільки він залежить від масштабу.

2.3.4 Оцінки кластеризації.

Середня відстань між точками даних та центром призначеного ним кластера. Середня відстань - це міра близькості точок даних до центроїдів кластерів. Ця метрика вказує, наскільки "щільним" є кластер. Значення ближче до 0 краще. Чим ближче середня відстань до нуля, тим паче кластеризовані дані. Зверніть увагу, що ця метрика зменшуватиметься, якщо кількість кластерів збільшується. У крайньому випадку (де кожна окрема точка даних є власним кластером) вона дорівнюватиме нулю.

Середнє співвідношення відстаней усередині кластера та відстаней між кластерами. Що щільніше кластер і що далі кластери друг від друга, то нижче це значення. Значення ближче до 0 краще. Кластери, які знаходяться далі один від одного і є менш розподіленими, призведуть до кращої оцінки.

Ця метрика підходить, коли дані, які використовуються для навчання моделі кластеризації, також поставляються з контрольними мітками (тобто захищеною кластеризацією). Метрика "Нормалізована взаємна інформація" показує, чи однакові точки даних призначаються одному і тому ж кластеру, а різні точки даних - різним кластерам. Ця метрика набуває значення від 0 до 1. Значення ближче до 1 краще.

2.3.5 Метрики оцінки для ранжування

Дисконтований сукупний прибуток є мірою якості ранжування. Вона є похідною від двох припущень. Перше: високо релевантні елементи корисніші, якщо відображаються вище як ранжування. Друге: корисність відстежує релевантність, тобто чим вища релевантність, тим корисніший елемент. Дисконтована сукупна прибуток розраховується для конкретної позиції порядку ранжирования. Вона підсумовує оцінку релевантності, поділену на логарифм індексу ранжирування до потрібної позиції. Вона розраховується за допомогою формули ∑. Оцінки релевантності надаються алгоритму навчання ранжирування як контрольні позначки. Одне значення дисконтованого сукупного прибутку надається кожної позиції у таблиці ранжування, звідси і назва " Дисконтована сукупний прибуток " . Перевага надається вищим значенням

Нормалізація дисконтованого сукупного прибутку дозволяє порівнювати метрику списків ранжування різної довжини. Значення ближче до 1 краще.

2.3.6 Оцінки виявлення аномалій

Метрика "Площа під ROC-кривою" показує, наскільки добре модель розділяє аномальні та звичайні точки даних. Значення ближче до 1 краще. Тільки значення більше 0,5 демонструють ефективність моделі. Значення 0,5 або нижче вказують, що модель працює не краще ніж випадковий розподіл вхідних даних на категорії аномальних і звичайних.

Частота виявлення при кількості неправдиво-позитивних значень - це співвідношення кількості правильно визначених аномалій і загальної кількості аномалій в тестовому наборі, що індексуються по кожному помилково-позитивному результату. Таким чином, для кожного хибно-позитивного елемента існує значення частоти виявлення при кількості помилкових спрацьовувань. Значення ближче до 1 краще. Якщо помилкові спрацьовування відсутні, це значення 1.

3 РОЗРОБКА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ДЛЯ ДОСЛІДЖЕННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

Дане дослідження має на меті окремий аналіз кожної з розглянутих у попередньому пункті сценаріїв машинного навчання, тому інструментальні засоби, що входять до складу розробленого у результаті роботи програмного комплексу також будуть створюватися окремо для моделювання елементарних випадків кожної з розглянутих ситуацій. Проектування інструментальних засобів буде виконуватись за допомогою UML-діаграм. Unified Modeling Language, або UML — уніфікована мова візуального моделювання систем, у т.ч. програмного забезпечення. UML можливо використовувати для візуального моделювання за будь-яких обраних методології проектування.

3.1 Формалізація задачі

Формалізація задачі з розробки тематичного комплексу програмного інструментарію представлена у вигляді діаграми прецедентів. Діаграма прецедентів, або варіантів використання, використовується для визначення предметної області системи. Для програмного забезпечення у діаграмах прецедентів у еліпсах вказуються опції використання програми, що можуть виконуватись користувачем (у даному випадку він називається актором та позначається на діаграмі як чоловічок).

Позначка <<include>> використовується для дій, що включаються у більш загальну дію. Позначка <<extend>> використовується на позначення дій, якими інші дії можуть доповнюватися. Діаграма формалізації задачі представлена на рис.9 та містить короткийопис дій, які зможе виконувати користувач при безпосередній взаємодії з виконавчим файлом програми.

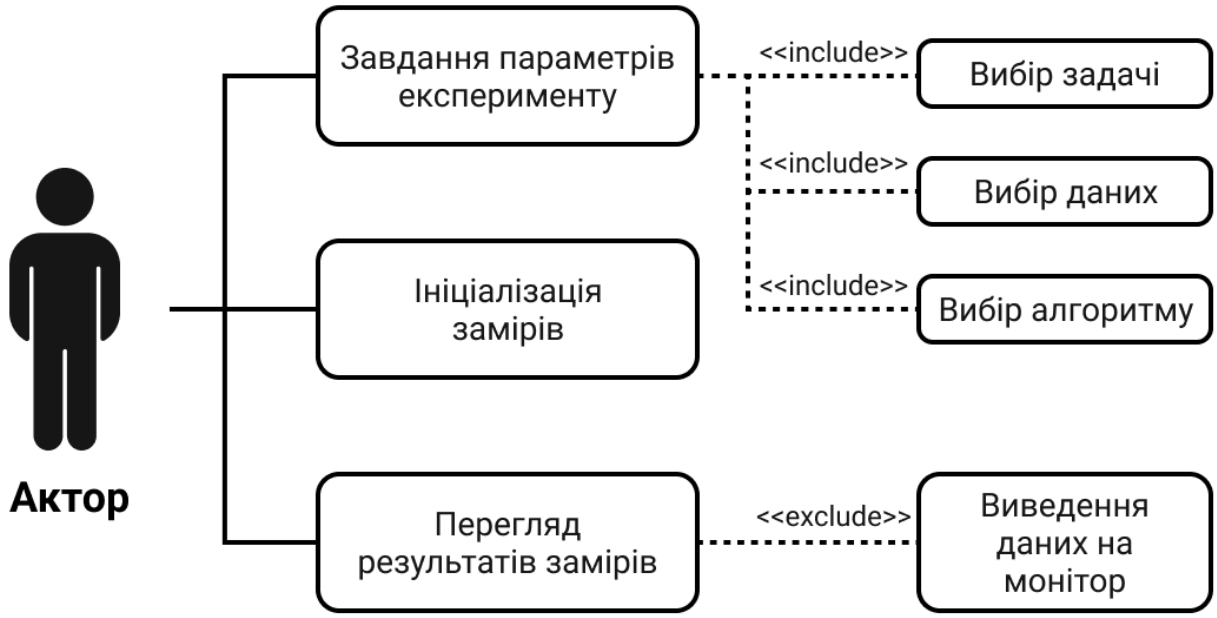

Рис.9 Діаграма варіантів використання

3.2 Вибір мови програмування та середовища розробки

За допомогою ML.NET можна створювати власні моделі машинного навчання за допомогою C# або F#, не виходячи з екосистеми .NET.

На сьогоднішній момент мова програмування C# одна з найпотужніших мов, що швидко розвиваються і затребуваних в IT-галузі. Зараз на ньому пишуться різні програми: від невеликих десктопних програм до великих веб-порталів і веб-сервісів, що обслуговують щодня мільйони користувачів.

C# є об'єктно-орієнтованим і в цьому плані багато перейняв у Java та C++. C# підтримує поліморфізм, успадкування, навантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і додатків, що розширюються. І C# продовжує активно розвиватися і з кожною новою версією з'являється все більше цікавих функціональностей.

3.3 Особливості мови C# для використання у розробці

Мова C# була створена спеціально для роботи з фреймворком .NET, проте саме поняття .NET дещо ширше. Фреймворк .NET представляє потужну платформу створення додатків. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов: поряд із C# це також VB.NET, C++, F#, а також різні діалекти інших мов, прив'язані до .NET, наприклад, Delphi.NET. Під час компіляції код будь-якою з цих мов компілюється в збірку загальною мовою CIL (Common Intermediate Language) - свого роду асемблер платформи .NET. Тому за певних умов ми можемо зробити окремі модулі одного додатка окремими мовами.

.NET є платформою, що переноситься (з деякими обмеженнями). Наприклад, остання версія платформи на даний момент – .NET 7 підтримується на більшості сучасних ОС Windows, MacOS, Linux. Використовуючи різні технології на платформі .NET, можна розробляти програми на мові C# для різних платформ - Windows, MacOS, Linux, Android, iOS, Tizen.

.NET представляє єдину всім підтримуваних мов бібліотеку класів. І який би додаток ми не збиралися писати на C# - текстовий редактор, чат або складний веб-сайт - так чи інакше ми використовуємо бібліотеку класів .NET.

Загальномовне середовище виконання CLR та базова бібліотека класів є основою цілого стеку технологій, які розробники можуть задіяти при побудові тих чи інших додатків. Наприклад, для роботи з базами даних у цьому стеку технологій призначено технологію ADO.NET та Entity Framework Core. Для побудови графічних додатків з багатим насиченим інтерфейсом – технологія WPF та WinUI, для створення більш простих графічних додатків – Windows Forms. Для розробки кросплатформових мобільних та десктопних програм - Xamarin/MAUI. Для створення веб-сайтів та веб-додатків - ASP.NET і т.д.

До цього варто додати активний Blazor - фреймворк, що розвивається і набирає популяність, який працює поверх .NET і який дозволяє створювати веб-додатки як

на стороні сервера, так і на стороні клієнта. А в майбутньому підтримуватиме створення мобільних додатків та, можливо, десктоп-додатків.

Згідно з рядом тестів веб-програми на .NET 7 у ряді категорій сильно випереджають веб-програми, побудовані за допомогою інших технологій. Програми на .NET 7 у принципі відрізняються високою продуктивністю.

Також слід відзначити таку особливість мови C# і фреймворку .NET, як автоматичне видалення сміття. А це означає, що нам здебільшого не доведеться, на відміну від C++, дбати про звільнення пам'яті. Вищезгадане загальномовне середовище CLR сама викличе збирач сміття та очистить пам'ять.

3.4 Особливості архітектури ML.NET

ML.NET починається з об'єкта MLContext. Загальний контекст для операцій ML.NET. Після створення екземпляра користувача він надає спосіб створення компонентів для підготовки даних, проектування ознак, навчання, прогнозування та оцінки моделі. Він також дозволяє вести журнал, керування виконанням та налаштовувати повторювані випадкові числа. Цей одноелементний об'єкт містить каталоги. Каталог - це фабрика завантаження та збереження даних, перетворень, інструкторів та компонентів операцій моделі. Кожен об'єкт каталогу має методи для створення різних типів компонентів. Для ззавантаження та збереження даних використовують компоненти DataOperationsCatalog та TransformsCatalog.

DataOperationsCatalog - це клас, який використовується для створення компонентів, які працюють з даними, але не є частиною конвеєра навчання моделі. Включає компоненти для завантаження, збереження, кешування, фільтрації, перетасовування та поділу даних.

TransformsCatalog – це клас, що використовується MLContext для створення екземплярів компонентів перетворення.

Каталоги для роботи з алгоритмами навчання: Двійкова класифікація (BinaryClassificationCatalog), Багатокласова класифікація

(MulticlassClassificationCatalog), Виявлення аномалій (AnomalyDetectionCatalog), Кластеризація (ClusteringCatalog), Прогнозування (ForecastingCatalog), Ранжування (RankingCatalog), Регресія (RegressionCatalog), Рекомендація (RecommendationCatalog).

А також каталог для що використовується MLContext для збереження та завантаження навчених моделей – ModelOperationsCatalog.

Через кожну із зазначених вище категорій можна перейти до відповідних методів створення. У Visual Studio каталоги відображаються за допомогою IntelliSense як зображено на рис.10.

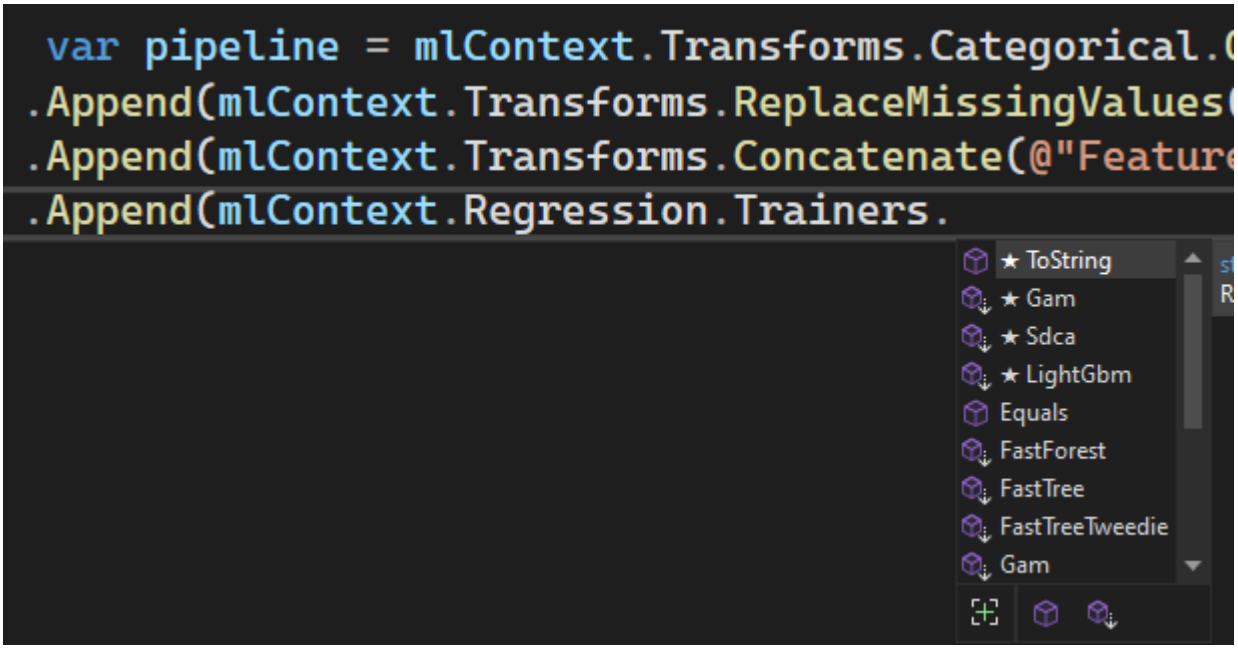

Рис.10 Можливості каталогу які можна застосувати

У кожному каталозі є набір методів розширення. Подивимося, як методи розширення використовуються під час створення конвеєра навчання. У поданому нижче фрагменті коду рис.11 Concatenate і OnlineGradientDescent це методи з каталогу. Кожен з них створює об'єкт IEstimator, який додається до конвеєра. Цей етап включає лише створення об'єктів. Виконання не відбувається.

```
var mlContext = new MLContext();

var reader = mlContext.Data.CreateTextLoader<ModelInput>(separatorChar: ',', hasHeader: true);
IDataView trainingDataView = reader.Load("heart.csv");

var pipeline = mlContext.Transforms.Categorical.OneHotEncoding(new[] { new InputOutputColumnPair(@"fbs", @"fbs"), new
.Append(mlContext.Transforms.ReplaceMissingValues(new[] { new InputOutputColumnPair(@"age", @"age"), new InputOutputCo
    new InputOutputColumnPair(@"cp", @"cp"),
    new InputOutputColumnPair(@"trtbps", @"trtbps"),
    new InputOutputColumnPair(@"chol", @"chol"),
    new InputOutputColumnPair(@"restecg", @"restecg"),
    new InputOutputColumnPair(@"thalachh", @"thalachh"), new InputOutputColumnPair(@"oldpeak", @"oldpeak"),
    new InputOutputColumnPair(@"slp", @"slp"),
    new InputOutputColumnPair(@"caa", @"caa"),
    new InputOutputColumnPair(@"thall", @"thall") }))
.Append(mlContext.Transforms.Concatenate(@"Features", new[] { @"fbs", @"exng", @"age", @"sex", @"cp", @"trtbps", @"cho
.Append(mlContext.Regression.Trainers.OnlineGradientDescent(
    new OnlineGradientDescentTrainer.Options()
    {
```

Рис.11 Створення конвеєра навчання

Після створення об'єктів у конвеєрі дані можна використовуватиме навчання моделі рис.12.

```
var model = pipeline.Fit(trainingDataView);
```

Рис.12 Навчання моделі

Виклик функції Fit() задіює вхідні навчальні дані для оцінки параметрів моделі. Це називається навчання моделі. Після виклику функції Fit() значення цих параметрів стають відомими.Отриманий об'єкт моделі реалізує інтерфейс ITransformer. Це означає, що модель перетворює вхідні дані на прогнози.

Вхідні дані можна перетворювати на прогнози все відразу або по черзі. Метод CreatePredictionEngine() рис.13 приймає вхідний та вихідний клас даних. Імена полів та/або атрибути коду визначають імена стовпців даних, які використовуються при навчанні моделі та прогнозуванні.

```
1 reference
private static PredictionEngine<ModelInput, ModelOutput> CreatePredictEngine()
{
    var mlContext = new MLContext();
    ITransformer mlModel = mlContext.Model.Load(MLNetModelPath, out var _);
    return mlContext.Model.CreatePredictionEngine<ModelInput, ModelOutput>(mlModel);
}
```

Рис.13 Створення прогнозу

3.5 Опис дослідницьких модулів

3.5.1 Лінійні алгоритми

Лінійні алгоритми створюють модель, яка обчислює оцінки з урахуванням лінійного поєднання вхідних даних, і набору вагових коефіцієнтів. Вагові коефіцієнти - це параметри моделі, що оцінюються під час навчання.

Лінійні алгоритми добре підходять для ознак, що є сепарабельними лінійно.

Перед навчанням за допомогою лінійного алгоритму необхідно нормалізувати ознаки. Це не дозволяє одній ознакі більш впливати на результат порівняно з іншими ознаками.

У загальному випадку лінійні алгоритми є масштабованими та швидкими, а також не вимагають великих витрат на навчання та прогнозування. Вони масштабуються за кількістю ознак і приблизно за розміром набору для навчання.

Лінійні алгоритми роблять кілька проходів за даними навчання. Якщо набір даних міститься в пам'ять, то додавання контрольної точки кешу до конвеєра ML.NET перед додаванням навчального алгоритму прискорить навчання.

3.5.1.1 L-BFGS алгоритми

L-BFGS алгоритми використовується при великій кількості ознак. Створює статистику навчання логістичної регресії, але не підходять для масштабується.

LbfgsMaximumEntropyMulti алгоритм використовуються у задачах двійкової та багатокласовой класифікацій.

```
// Data process configuration with pipeline data transformations
var pipeline = mlContext.Transforms.Categorical.OneHotEncoding(new []{new InputOutputColumnPair(@"language", @"language"),new InputOutputCol
                        .Append(mlContext.Transforms.ReplaceMissingValues(@"hasImage", @"hasImage"))
                        .Append(mlContext.Transforms.Text.FeaturizeText(@"title", @"title"))
                        .Append(mlContext.Transforms.Text.FeaturizeText(@"text", @"text"))
                        .Append(mlContext.Transforms.Text.FeaturizeText(@"title_without_stopwords", @"title_without_stopwords"))
                        .Append(mlContext.Transforms.Text.FeaturizeText(@"text_without_stopwords", @"text_without_stopwords"))
                        .Append(mlContext.Transforms.Concatenate(@"Features", new []{@"language",@"site_url",@"type",@"hasImage",@"title",@"
                        .Append(mlContext.Transforms.Conversion.MapValueToKey(@"label", @"label"))
                        .Append(mlContext.Transforms.NormalizeMinMax(@"Features", @"Features"))
                        .Append(mlContext.MulticlassClassification.Trainers.LbfgsMaximumEntropy(
                            l1Regularization:0.8556611994433955F,l2Regularization:0.929624221195336F,
                            labelColumnName:@"label",featureColumnName:@"Features"))
                        .Append(mlContext.Transforms.Conversion.MapKeyToValue(@"PredictedLabel", @"PredictedLabel"));
```

Рис.14 Використання LbfgsMaximumEntropyMulti

LbfgsPoissonRegressionTrainer алгоритм використовуються у задачах регресії.

Рис.15 Використання LbfgsPoissonRegressionTrainer

Регресія Пуассона це параметризований метод регресії. Передбачається, що журнал умовного середнього значення залежної змінної слідує лінійній функції залежних змінних. Припускаючи, що залежні змінні наслідують розподіл Пуассона, параметри регресії можна оцінити, максимізуючи ймовірність отриманих спостережень.

У статистиці регресія Пуассона є узагальненою лінійною моделлю регресійного аналізу, який використовувується для моделювання даних та таблиць непередбачених ситуацій. Регресія Пуассона передбачає, що залежна змінна Y має розподіл Пуассона, і припускає, що логарифм її математичного сподівання може бути змодельоване лінійною комбінацією невідомих параметрів. Модель регресії Пуассона іноді називається лог-лінійною моделлю, особливо якщо вона використовується для моделювання таблиці непередбачених ситуацій.

LbfgsLogisticRegressionBinaryTrainer алгоритм використовуються у задачах двійкової класифікацій.

```
var pipeline = mlContext.Transforms.Text.FeaturizeText(@"comment", @"comment")
                        .Append(mlContext.Transforms.Concatenate(@"Features", @"comment"))
                        .Append(mlContext.Transforms.Conversion.MapValueToKey(@"toxic", @"toxic"))
                        .Append(mlContext.Transforms.NormalizeMinMax(@"Features", @"Features"))
                        .Append(mlContext.MulticlassClassification.Trainers.OneVersusAll
                        (binaryEstimator:mlContext.BinaryClassification.Trainers.LbfgsLogisticRegression(
                            l1Regularization:5.844774924428878F,
                            l2Regularization:2.34510133216183F,
                            labelColumnName:@"toxic",
                            featureColumnName:@"Features"),
                            labelColumnName: @"toxic"))
                        .Append(mlContext.Transforms.Conversion.MapKeyToValue(@"PredictedLabel", @"PredictedLabel"));
```

Рис.16 Використання LbfgsLogisticRegressionBinaryTrainer

Реалізована методика оптимізації заснована на обмеженій пам'яті методу Broyden-Fletcher-Goldfarb-Shanno (L-BFGS). L-BFGS - це квазі-Ньютонівський метод, який замінює витрати на витрати на обчислення гесійської матриці наближенням, але, як і раніше, користується швидкою конвергенцією, як метод Ньютон, де обчислюється повна матриця Гессіана. Так як наближення L-BFGS використовує лише обмежену кількість історичних станів для обчислення наступного напряму кроку, воно особливо підходить для проблем із вектором ознак з високою розмірністю. Число історичних станів є параметром користувача, використання більшого числа може призвести до кращого наближення до матриці Hessian, але і до більш високої вартості обчислень на крок.

Регуляризація – це метод, який може призвести до того, що проблема може виявитися гострішою, вводячи обмеження, які надають інформацію для доповнення даних і яка запобігає перенавченню шляхом покарання величини моделі, яка зазвичай вимірюється деякими нормними функціями. Такий підхід дозволяє покращити узагальнення моделі, реалізоване за допомогою вибору оптимальної складності компромісної частоті винятків. Регуляризация передбачає додавання штрафу, що з значеннями коефіцієнтів, до похибки гіпотези. До точної моделі з граничними коефіцієнтами буде застосовано більший штраф. При цьому для менш точної моделі з більш прийнятними значеннями штраф буде меншим.

3.5.1.2 Стохастичний подвійний покоординатний підйом

Алгоритми стохастичні подвійний покоординатно підйомні можна використовувати для двійкової класифікації, багатокласової класифікації та регресії. Не потрібне налаштування для забезпечення хорошої продуктивності.

SdcaMaximumEntropyMulticlassTrainer алгоритм використовуються у задачах двійкової та багатокласовой класифікацій.

```
// Data process configuration with pipeline data transformations
var pipeline = mlContext.Transforms.Categorical.OneHotEncoding(new []{new InputOutputColumnPair(@"language", @"language"),new InputOutputCol
                        .Append(mlContext.Transforms.ReplaceMissingValues(@"hasImage", @"hasImage"))
                        .Append(mlContext.Transforms.Text.FeaturizeText(@"title", @"title"))
                        .Append(mlContext.Transforms.Text.FeaturizeText(@"text", @"text"))
                        .Append(mlContext.Transforms.Text.FeaturizeText(@"title_without_stopwords", @"title_without_stopwords"))
                        .Append(mlContext.Transforms.Text.FeaturizeText(@"text_without_stopwords", @"text_without_stopwords"))
                        .Append(mlContext.Transforms.Concatenate(@"Features", new []{@"language",@"site_url",@"type",@"hasImage",@"title",@"
                        .Append(mlContext.Transforms.Conversion.MapValueToKey(@"label", @"label"))
                        .Append(mlContext.Transforms.NormalizeMinMax(@"Features", @"Features"))
                        .Append(mlContext.MulticlassClassification.Trainers.SdcaMaximumEntropy(
                            l1Regularization:0.8556611994433395F,l2Regularization:0.929624221195336F,
                            labelColumnName:@"label",featureColumnName:@"Features"))
                        .Append(mlContext.Transforms.Conversion.MapKeyToValue(@"PredictedLabel", @"PredictedLabel"));
```

Рис.17 Використання SdcaMaximumEntropyMulticlassTrainer

SdcaRegressionTrainer алгоритм використовуються у задачах регресії.

```
var pipeline = mlContext.Transforms.ReplaceMissingValues(new []{new InputOutputColumnPair(@"age", @"age"),new InputOutputColumnPair(@"anaemi
                        .Append(mlContext.Transforms.Concatenate(@"Features", new []{@"age",@"anaemia",@"creatinine_phosphokinase",@"diabete
                        .Append(mlContext.Regression.Trainers.Sdca(
                            new SdcaRegressionTrainer.Options()
                            {
                                LabelColumnName = @"platelets",
                                FeatureColumnName = @"Features",
                            }));
```

Рис.18 Використання SdcaRegressionTrainer

SdcaLogisticRegressionBinaryTrainer алгоритм використовуються у задачах двійкової класифікацій.

```
// Data process configuration with pipeline data transformations
var pipeline = mlContext.Transforms.ReplaceMissingValues(new []{new InputOutputColumnPair(@"fixed acidity", @"fixed acidity"),new InputOutpu
                        .Append(mlContext.Transforms.Concatenate(@"Features", new []{@"fixed acidity",@"volatile acidity",@"citric acid",@"r
                        .Append(mlContext.Transforms.Conversion.MapValueToKey(@"quality", @"quality"))
                        .Append(mlContext.MulticlassClassification.Trainers.OneVersusAll(
                            binaryEstimator:mlContext.BinaryClassification.Trainers.SdcaLogisticRegression(
                                new SdcaLogisticRegressionBinaryTrainer.Options()
                                {
                                    LabelColumnName=@"quality",
                                    FeatureColumnName=@"Features"}
                                ), labelColumnName: @"quality"))
                        .Append(mlContext.Transforms.Conversion.MapKeyToValue(@"PredictedLabel", @"PredictedLabel"));
```

Рис.19 Використання SdcaLogisticRegressionBinaryTrainer

Цей навчальний методи заснований на методі подвійного Стохастичного підйому координат (SDCA) — методи оптимізації з опуклими цільовими

функціями. Алгоритм можна масштабувати, оскільки це алгоритм навчання потокової передачі, як описано у кращому документі KDD.

Конвергенція виконується періодично шляхом примусової синхронізації між первинними та подвійними змінними в окремому потоці. Також надається кілька варіантів функцій втрат, таких як втрата петлі та логістична втрата. Залежно від використовуваної втрати, навчена модель може бути, наприклад, опорна векторна машина або логістична регресія. Метод SDCA поєднує декілька кращих властивостей, таких як можливість потокового навчання (без встановлення всього набору даних на згадку), досягнення розумного результату з кількома скануваннями всього набору даних (наприклад, див. експерименти в цьому документі) і не витрачаючи обчислень на нулі розріджених наборів даних.

SDCA – це алгоритм стохастичної та потокової оптимізації. Результат залежить від порядку навчальних даних, оскільки похибка зупинки недостатньо жорстка. При строго опуклій оптимізації оптимальне рішення унікальне і, отже, всі зрештою досягають того ж місця. Навіть у несучасних випадках отримаємо однаково добрі рішення від запуску до запуску. Для відтворюваних результатів рекомендується встановити для параметра Shuffle значення False і NumThreads значення 1.

Цей клас використовує емпіричну мінімізацію ризиків (тобто ERM) на формування завдання оптимізації, створеної з урахуванням зібраних даних. Зауважте, що емпіричний ризик зазвичай вимірюється шляхом застосування функції втрати до прогнозів моделі за зібраними точками даних. Якщо навчальні дані не містять достатньо точок даних, може відбутися перенавчання, щоб модель, створена ERM, добре описувала навчальні дані, але може прогнозувати правильні результати в незавиденных подіях. Регуляризация є поширеним методом полегшення такого явища шляхом покарання величини (звичайно вимірюваної функцією норм) параметрів моделі. Цей тренер підтримує еластичну чисту регуляризацію, яка карає лінійне поєднання L1-норм (LASSO), та L2-норм (ridge).

Регуляризація норм L1 та L2-норм мають різні ефекти та використання, які є взаємодоповнювальними у певних відносинах.

Поряд із реалізованим алгоритмом оптимізації нормалізація L1-норм може збільшити розрідженість ваги моделі. Для зарозумілих і розріджених наборів даних, якщо користувачі ретельно вибирають коефіцієнт L1-норм, можна досягти хорошої якості прогнозування з моделлю, яка має лише кілька ненульових ваг (наприклад, 1% загальної ваги моделі), не впливаючи на її потужність прогнозування. На відміну від цього, L2-норм не може збільшити розрідженість навченої моделі, але, як і раніше, може запобігти перенавчанню, уникаючи великих значень параметрів. Іноді використання L2-норм призводить до кращої якості прогнозування, тому користувачі можуть хотіти спробувати його і точно налаштувати коефіцієнти L1-норм і L2-норм. Концептуально використання L1-норм має на увазі, що розподіл всіх параметрів моделі є розподілом Laplace, а L2-нормою для них є гауссіанський розподіл.

Агресивна регуляризація (тобто присвоєння великих коефіцієнтів термінам регуляризації L1-норм або норм L2) може завдати шкоди прогнозній ємності за рахунок виключення важливих змінних з моделі. Наприклад, дуже великий коефіцієнт норм L1 може призвести до всіх параметрів нулі та призвести до тривіальної моделі. Тому вибір правильних коефіцієнтів регуляризації важливий практично.

3.5.2 Алгоритми дерева прийняття рішень

Алгоритми прийняття рішень створюють модель, яка містить ряд рішень: по суті, блок-схему для значень даних.

Для використання цього алгоритму не потрібні лінійно масштабовані ознаки. Крім того, ознаки не потрібно нормалізувати, оскільки окремі значення у векторі ознак використовуються незалежно у процесі прийняття рішень.

Алгоритми дерева прийняття рішень зазвичай дуже точні. За винятком узагальнених адитивних моделей (GAM), моделі дерева можуть мати недостатню пояснюваність, коли число ознак велике.

Алгоритми дерева прийняття рішень використовують більше ресурсів та гірше масштабуються порівняно з лінійними алгоритмами. Вони добре підходять для наборів даних, що містяться на згадку.

Розширені дерева прийняття рішень є ансамбль невеликих дерев, де кожне дерево оцінює вхідні дані і передає результат наступному дереву для уточнення оцінки і т. д., тобто кожне наступне дерево покращує результат попереднього.

3.5.2.1 Машина слабого градиентного бустинга

Машина слабкого градієнтного бустингу це алгоритми найшвидший і найточніший з навчальних алгоритмів дерев двійкової класифікації. Алгоритми Широкі можливості налаштування.

LightGbmRegressionTrainer алгоритм використовуються у задачах регресії.

```
var pipeline = mlContext.Transforms.ReplaceMissingValues(new []{new InputOutputColumnPair(@"age", @"age"),new InputOutputColumnPair(@"anaemi
                    .Append(mlContext.Transforms.Concatenate(@"Features", new []{@"age",@"anaemia",@"creatinine_phosphokinase",@"diabete
                    .Append(mlContext.Regression.Trainers.LightGbm(
                        new LightGbmRegressionTrainer.Options()
                        {
                            NumberOfLeaves=4,
                            MinimumExampleCountPerLeaf=41,
                            NumberOfIterations=4,
                            MaximumBinCountPerFeature=930,
                            LearningRate=1F,
                            LabelColumnName=@"platelets",
                            FeatureColumnName=@"Features",
                            Booster=new GradientBooster.Options()
                            {SubsampleFraction=0.393250737978472F,
                                FeatureFraction=0.654987102136424F,
                                L1Regularization=13537.6903874294F,
                                L2Regularization=0.0212905647969468F
                            }}));
```

Рис.20 Використання LightGbmRegressionTrainer

LightGBM - це відкритий код для реалізації дерева прийняття рішень по градієнту. LightGBM, скорочення від light gradient-boosting machine, — це безкоштовна розподілена структура для машинного навчання з відкритим вихідним кодом, розроблена Microsoft. Він заснований на алгоритмах дерева рішень і використовується для ранжування, класифікації та інших завдань машинного навчання. Розробка зосереджена на продуктивності та масштабованості.

3.5.2.2 Швидке дерево

Алгоритми швидкого дерева використовується для зображення даних з певними ознаками. Стійкий до незбалансованих даних. Мають широкі можливості налаштування.

FastTreeRegressionTrainer алгоритм використовуються у задачах регресії.

```
var pipeline = mlContext.Transforms.ReplaceMissingValues(new []{new InputOutputColumnPair(@"age", @"age"),new InputOutputColumnPair(@"sex", @"sex"),ne
                        .Append(mlContext.Transforms.Concatenate(@"Features", new []{@"age",@"sex",@"cp",@"trtbps",@"chol",@"fbs",@"restecg",@"thalach
                        .Append(mlContext.Regression.Trainers.FastTree(
                            new FastTreeRegressionTrainer.Options()
                            {
                                NumberOfTrees=16,
                                FeatureFraction=0.865867730409574F,
                                LabelColumnName=@"output",
                                FeatureColumnName=@"Features"
                            }));
```

Рис.21 Використання FastTreeRegressionTrainer

FastTree - це ефективна реалізація алгоритму підвищення градієнта MART. Градієнтний бустинг - це метод машинного навчання для проблем регресії. Він крок за кроком створює кожне дерево регресії, використовуючи стандартну функцію втрат для вимірювання помилок у кожному кроці та їх виправлення в наступному. Тому така модель прогнозування фактично є набором слабкіших моделей прогнозування. У задачах регресії при посиленні покроково створюється серія дерев, а потім за допомогою довільної функції втрат, що диференціюється, вибирається оптимальне дерево.

MART вивчає набір дерев регресії, який є деревом прийняття рішень із скалярними значеннями у листових вузлах. Дерево прийняття рішень (регресії) - це деревоподібна блок-схема, в якій на кожному внутрішньому вузлі приймається рішення, який із двох дочірніх вузлів використовувати далі, на основі одного із значень функції із вхідних даних. Значення повертається у кожному листовому вузлі. У внутрішніх вузлах рішення засноване на тесті x <= v, де x це значення функції у вхідному прикладі, а v одне з можливих значень цієї функції. Функції, які можуть створюватися за допомогою дерева регресії, є шматковими функціями-константами.

Набір дерев створюється шляхом обчислення (на кожному етапі) дерева регресії, що отримує наближення градієнта функції втрат і додає його до попереднього дерева з коефіцієнтами, які мінімізують втрати нового дерева. Вихідні дані набору, виробленого MART з урахуванням даного екземпляра, становлять суму трьох варіантів вихідних даних.

— Для проблеми бінарної класифікації вихідні дані перетворюються на ймовірність з використанням того чи іншого варіанту калібрування.
— Для проблем регресії вихідні дані є спрогнозоване значення функції.
— Для проблеми ранжирування екземпляри упорядковуються за вихідним значенням набору.

FastTreeTweedieTrainer алгоритм використовуються у задачах регресії.

```
var pipeline = mlContext.Transforms.ReplaceMissingValues(new []{new InputOutputColumnPair(@"age", @"age"),new InputOutputColumnPair(@"sex", @"sex"),ne
                    .Append(mlContext.Transforms.Concatenate(@"Features", new []{@"age",@"sex",@"cp",@"trtbps",@"chol",@"fbs",@"restecg",@"thalach
                    .Append(mlContext.Regression.Trainers.FastTreeTweedie(
                        new FastTreeTweedieTrainer.Options()
                        {
                            NumberOfTrees=16,
                            FeatureFraction=0.865867730409574F,
                            LabelColumnName=@"output",
                            FeatureColumnName=@"Features"
                        }));
```

Рис.22 Використання FastTreeTweedieTrainer

Модель підвищення Tweedie слідує математиці, встановленій у страховому преміум прогнозуванні через Градієнт. Підвищення градієнта зазвичай використовується з деревами рішень (особливо CART) фіксованого розміру як базові навчальні елементи. Для цього особливого випадку Фрідман пропонує модифікацію методу підвищення градієнта, що покращує якість відповідності кожного базового учня.

FastTreeBinaryTrainer алгоритм використовуються у задачах двійкової класифікацій.

```
var pipeline = mlContext.Transforms.ReplaceMissingValues(new []{new InputOutputColumnPair(@"fixed acidity", @"fixed acidity"),new InputOutpu
                        .Append(mlContext.Transforms.Concatenate(@"Features", new []{@"fixed acidity",@"volatile acidity",@"citric acid",@"r
                        .Append(mlContext.Transforms.Conversion.MapValueToKey(@"quality", @"quality"))
                        .Append(mlContext.MulticlassClassification.Trainers.OneVersusAll(
                            binaryEstimator:mlContext.BinaryClassification.Trainers.FastTree(
                                new FastTreeBinaryTrainer.Options()
                                {
                                    NumberOfLeaves=1599,
                                    MinimumExampleCountPerLeaf=2,
                                    NumberOfTrees=896,
                                    MaximumBinCountPerFeature=1024,
                                    LearningRate=1F,
                                    FeatureFraction=1F,
                                    LabelColumnName=@"quality",
                                    FeatureColumnName=@"Features"}
                                ), labelColumnName: @"quality"))
                        .Append(mlContext.Transforms.Conversion.MapKeyToValue(@"PredictedLabel", @"PredictedLabel"));
```

Рис.23 Використання FastTreeBinaryTrainer

3.5.2.3 Швидкий ліс

Алгоритми швидкого лиса відмінно підходить для даних із високим рівнем шуму.

FastForestRegressionTrainer алгоритм використовуються у задачах регресії.

```
var pipeline = mlContext.Transforms.ReplaceMissingValues(new []{new InputOutputColumnPair(@"age", @"age"),new InputOutputColumnPair(@"sex", @"sex"),new
                        .Append(mlContext.Transforms.Concatenate(@"Features",
                        new []{@"age",@"sex",@"cp",@"trtbps",@"chol",@"fbs",@"restecg",@"thalachh",@"exng",@"oldpeak",@"slp",@"caa",@"thall"}))
                        .Append(mlContext.Regression.Trainers.FastForest(
                            new FastForestRegressionTrainer.Options()
                            {
                                NumberOfTrees=16,
                                FeatureFraction=0.865867730409574F,
                                LabelColumnName=@"output",
                                FeatureColumnName=@"Features"
                            }));
```

Рис.24 Використання FastForestRegressionTrainer

FastForestBinaryTrainer алгоритм використовуються у задачах двійкової класифікацій.

```
var pipeline = mlContext.Transforms.ReplaceMissingValues(new []{new InputOutputColumnPair(@"fixed acidity", @"fixed acidity"),new InputOutpu
                        .Append(mlContext.Transforms.Concatenate(@"Features", new []{@"fixed acidity",@"volatile acidity",@"citric acid",@"r
                        .Append(mlContext.Transforms.Conversion.MapValueToKey(@"quality", @"quality"))
                        .Append(mlContext.MulticlassClassification.Trainers.OneVersusAll(
                            binaryEstimator:mlContext.BinaryClassification.Trainers.FastForest(
                                new FastForestBinaryTrainer.Options()
                                {
                                    LabelColumnName=@"quality",
                                    FeatureColumnName=@"Features"}
                                ), labelColumnName: @"quality"))
                        .Append(mlContext.Transforms.Conversion.MapKeyToValue(@"PredictedLabel", @"PredictedLabel"));
```

Рис.25 Використання FastForestBinaryTrainer

Дерева прийняття рішень - це непараметричні моделі, що виконують послідовність простих тестів на вхідних даних. Ця процедура прийняття рішень зіставляє їх з вихідними даними з навчального набору даних, вхідні дані якого

схожі на екземпляр, що обробляється. Рішення приймається на кожному вузлі структури даних двійкового дерева з урахуванням ступеня схожості, яка рекурсивно зіставляє кожен екземпляр через гілки дерева, доки не буде досягнуто відповідного листового вузол і не буде повернено вихідне рішення. Дерева прийняття рішень мають такі переваги:

— Вони ефективні з погляду обчислення та використання пам'яті під час навчання та прогнозування.
— Вони можуть представляти межі нелінійного ухвалення рішень.
— Вони виконують вбудований вибір ознак та класифікацію.
— Крім того, вони є стійкими за наявності шумових ознак.

Швидкий ліс – це реалізація випадкового лісу. Ця модель складається з сукупності дерев прийняття рішень. Кожне дерево у лісі прийняття рішень виводить розподіл Гауса шляхом прогнозування. По сукупності дерев виконується агрегування з метою знайти розподіл за Гауссом, найближчим до об'єднаного розподілу для всіх дерев моделі. Класифікатор лісу прийняття рішення складається із сукупності дерев прийняття рішень.

Взагалі, моделі сукупності забезпечують більше покриття і точність, ніж одне дерево прийняття рішень. Кожне дерево у лісі прийняття рішень виводить розподіл Гаусса.

3.5.2.4 Машина слабкого градієнтного бустингу

Алгоритми цього типу найшвидші і найточніші з навчальних алгоритмів дерев двійкової класифікації. Мають широкі можливості налаштування.

LightGbmBinaryTrainer алгоритм використовуються у задачах двійкової класифікацій.

```
var mlContext = new MLContext();

var reader = mlContext.Data.CreateTextLoader<ModelInput>(separatorChar: ',', hasHeader: true);
IDataView trainingDataView = reader.Load("heart.csv");

// Data process configuration with pipeline data transformations
var pipeline = mlContext.Transforms.Categorical.OneHotEncoding(new[] { new InputOutputColumnPair(@"fbs", @"fbs"), new InputOutputColumnPair(@"exng", @"exng")
                        .Append(mlContext.Transforms.ReplaceMissingValues(new[] { new InputOutputColumnPair(@"age", @"age"), new InputOutputColumnPair(@"sex"
                        .Append(mlContext.Transforms.Concatenate(@"Features", new[] { @"fbs", @"exng", @"age", @"sex", @"cp", @"trtbps", @"chol", @"restecg"
                        .Append(mlContext.BinaryClassification.Trainers.LightGbm(new LightGbmBinaryTrainer.Options
                        {
                            Booster = new GossBooster.Options
                            {
                                TopRate = 0.3,
                                OtherRate = 0.2
                            }
                        }));
```

Рис.26 Використання LightGbmBinaryTrainer

3.6 Тестування програми

При всій затребуваності машинного навчання, з поля зору замовників та виконавців подібних проектів, часто випадає область тестування в класичному її розумінні. Якщо тестуванню приділяється недостатньо уваги, це може призвести до некоректної роботи ML-систем і викликати розчарування замовника в технології як такої. Звичайно, самі фахівці з роботи з даними (Data Scientists) перевіряють свої моделі, використовуючи спеціальні метрики, проте реальність така, що сервіс не завжди здатний враховувати різні нюанси та вузькі місця виробництва. На жаль, в IT-галузі поки що немає усталених практик тестування ML-сервісів, тому, підбираючи тести, спираємося на власну експертизу та досвід фахівців, які знають галузеву специфіку компанії-замовника.

Ідея тестування полягає у використанні набору метрик, який дозволяє отримати повніше уявлення про межі якості моделі. Наприклад, ми можемо розділити валідаційний датасет за різними характеристиками - джерело даних, стать пацієнта, розмір зображення, розподіл таргету і рахувати метрики по цих сабсетах. Таким чином у нас утворюється кілька віртуальних валідаційних сетів. Можна створювати й окремі, що не перетинаються з основним датасетом - наприклад, регресіонний набір даних, що складається з прикладів, де ми колись помилялися, а потім зуміли вирішити проблему. Або демо-набір, що складається з прикладів, що використовуються для демонстрації можливостей роботи системи клієнтам.

Мінус цього підходу теж є. Навіть якщо вдалося значно покращити загальну якість моделі, існує велика ймовірність, що на якомусь із урізку даних або на конкретних прикладах предикати стали гіршими. Що робити у такій ситуації? Відповісти на це питання у вакуумі неможливо. У якихось ситуаціях можна зробити позначку на майбутнє, в якихось потрібно йти доопрацьовувати модель, в інших – реалізувати різні версії моделі та використовувати ту чи іншу залежно від характеристик вхідних даних.

В окрему підкатегорію можна винести тести на робото здатність та стійкість до атак.

Стійкість до змагальних атак. Досить рідкісний та специфічний тест. У цьому випадку ми намагаємося в режимах white box (повний доступ до моделі) та black box (доступ тільки до пророцтв) підібрати деяку трансформацію вхідних даних, яка кардинально змінить вивід системи.

Можна різними способами доповнити вхідні дані та виміряти зміни в предикатах, створити спеціальні тестові набори даних із рідкими або випадками поза доменом. При релізі нової версії корисно також порівняти її предикати з прогнозами попередньої версії. Якщо це не таке глобальне вдосконалення моделі, то різкі здвиги повинні викликати підвищення. Сильні зміни на конкретних прикладах також можна проінспектувати вручну.

4 ДОСЛІДЖЕННЯ АЛГОРИТМІВ У ВІДКРИТОМУ ФРЕЙМВОРКУ ML.NET

4.1 Вихідні умови експериментів

Експерименти з дослідження алгоритмів у відкритому фреймворку ML.NET повинні проводитись для їх аналізу перед отриманням статистичних даних, що потім будуть використовуватися для формування результативних висновків.

Для поточного експерименту було виділено умови проведення на основі доступних для використання ресурсів, тому існує вірогідність того, що для більш

ранніх або пізніх версій фреймворку ML.NET результати проведення аналогічних експериментів можуть суттєво відрізнятися. У даному розділі будуть розглянуті умови проведення експерименту.

4.1.1 Опис програмно-апаратного середовища для проведення експерименту

Для проведення експерименту було заздалегідь обрано мову програмування та середовище виконання, а саме: мова програмування C#, середовище виконання Microsoft Visual Studio Community 2022 (64-bit) - Version 17.3.6.

Інтегроване середовище розробки Visual Studio – це багатофункціональна програма, яка підтримує багато аспектів розробки програмного забезпечення. Visual Studio це майданчик для написання, налагодження та складання коду, а також подальшої публікації програм. Крім стандартного редактора і відладчика, які є в більшості середовищ IDE, Visual Studio включає компілятори, засоби автозавершення коду, графічні конструктори і багато інших функцій для поліпшення процесу розробки.

Дослідження алгоритмів проводиться на фізичному ЕОМ. Фізична машина представляє собою персональний IBM сумісний комп‘ютер зі встановленою фреймворком ML.NET та VisualStudio 2020. У таблиці 4.1 наведений перелік деяких системних характеристик для машини та їх значення.

Таблиця 4.1 — Системні характеристики використаного у дослідженні ЕОМ

Елемент	Значення
Назва ОС	Майкрософт Windows 11 Pro
Версія	10.0.22000 Збірка 22000
Виробник ОС	Microsoft Corporation
Назва системи	DESKTOP-SL8H8H6
Виробник	Micro-Star International Co., Ltd.
Модель	Pulse GL76 11UDK

Тип	x64-based PC
Обліковий номер системи	17L2.3
Процесор	11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz, 2304 МГц, ядер 8, логічних процесорів 16
Версія BIOS/Дата	American Megatrends International, LLC. E17L2IMS.311, 08.04.2022
Версія SMBIOS	3.3
Версія вбудованого контролера	255.255
Модель BIOS	UEFI
Виробник системної плати	Micro-Star International Co., Ltd.
Тип системної плати	MS-17L2
Версія системної плати	REV:1.0
Роль платформи	Мобільний
Стан безпечного завантаження	Увімкнути
Апаратнозалежний рівень (HAL)	Версія = "10.0.22000.1219"
Часовий пояс	Фінляндія (зима)
Установлена фізична пам’ять (ОЗП)	16,0 ГБ
Загальний обсяг фізичної пам'яті	15,7 ГБ
Доступно фізичної пам'яті	6,18 ГБ
Усього віртуальної пам'яті	29,7 ГБ
Доступно віртуальної пам'яті	14,5 ГБ
Розмір файлу довантаження	14,0 ГБ
Захист ПДП ядра	Увімкнути
Безпека на основі віртуалізації	Запущено
Безпека на основі віртуалізації: доступні властивості безпеки	Підтримка базової візуалізації, Безпечне завантаження, Захист ПДП, Тільки читання коду UEFI, SMM Security Mitigations 1.0, Керування

	виконанням залежно від режиму, Віртуалізація APIC
Безпека на основі віртуалізації: налаштовані служби	Застосована гіпервізором цілісність коду
Безпека на основі віртуалізації: запущені служби	Застосована гіпервізором цілісність коду
Політика керування програмами в Захиснику Windows	Застосовано
Політика застосування користувацького режиму для керування програмами в Захиснику Windows	Вимкнути

4.1.2 Опис підходу для визначення часу роботи алгоритму

У наведеній нижче таблиці 4.2 вказано узагальнений середній час, необхідний для отримання хорошої ефективності набору прикладів наборів даних на локальному комп'ютері.

Таблиця 4.2 — Системні характеристики використаного у дослідженні ЕОМ

Розмір набору даних	Середній час навчання
0–10 МБ	10 с
10–100 МБ	10 хв
100–500 МБ	30 хв
500 МБ — 1 ГБ	60 хв
Більше 1 ГБ	Більше 3 годин

Це орієнтовні числа. Точна тривалість навчання залежить від:

— кількості ознак (стовпців), що використовуються як вхідні дані для моделі;

— типу стовпців;

— завдання машинного навчання;

— продуктивності ЦП, диска та пам'яті комп'ютера, що використовується для навчання.

4.2 Проведення експерименту

Для проведення експериментів було обрано наступні датасети.

Набір даних аналізу та прогнозування серцевого нападу. Набір має 268 строк даних. Про цей набір даних:

— Вік: вік пацієнта;

— Стать: стать пацієнта;

— exang: стенокардія, спричинена фізичним навантаженням (1 = так; 0 = ні);

— ca: кількість великих судин (0-3);

— cp: Тип болю в грудях;

— Значення 1: типова стенокардія;

— Значення 2: атипова стенокардія;

— Значення 3: неангінальний біль;

— Значення 4: безсимптомний.

— trtbps: артеріальний тиск у спокої;

— chol: холестеральний у мг/дл, отриманий за допомогою датчика IMT;

— fbs: (цукор у крові натщесерце > 120 мг/дл) (1 = вірно; 0 = помилково);

— rest_ecg: результати електрокардіографії в спокої;

— Значення 0: нормально;

— Значення 1: наявність аномалії зубця ST-T;

— Значення 2: демонструє ймовірну або певну гіпертрофію лівого шлуночка за критеріями Естеса.

— thalach: досягнута максимальна частота серцевих скорочень;

— ціль: 0 - менший шанс серцевого нападу, 1 - більший шанс серцевого нападу.

Набір даних прогнозування вартості оренди житла. Набір має 4746 строк даних. Про цей набір даних:

— BHK: Кількість спалень, хол, кухня;

— Оренда: Оренда Будинку/Квартири/Квартири;

— Розмір: Розмір будинків/квартир/квартир у квадратних футах;

— Поверх: Будинки/Квартири/Квартири, на яких поверх і загальна кількість поверхів (Приклад: Перший з 2, 3 з 5 тощо);

— Тип зони: Розмір будинків/квартир/квартир;

— Район Місцевість: Місцевість будинків/квартир/квартир;

— Місто: місто, де розташовані будинки/квартири/квартири;

— Статус меблювання: статус меблювання будинків/квартир/квартир: мебльовані, напівмебльовані чи немебльовані;

— Бажаний орендар: тип орендаря, якому віддає перевагу власник або агент;

— Санвузол: Кількість санвузлів;

— Контактна особа: до кого слід звертатися для отримання додаткової інформації щодо будинків/квартир/квартир.

Набір даних аналізу справжніх чи підроблених вакансій. Набір має 17826 строк даних. Про цей набір даних:

— job_id: унікальний ідентифікатор вакансії;

— title: назва запису оголошення про роботу;

— location: географічне розташування оголошення про роботу;

— department: корпоративний відділ (наприклад, продажі);

— salary_range: орієнтовний діапазон зарплати;

— company_profile: короткий опис компанії;

— description: детальний опис оголошення про роботу;

— requirements: перелік вимог до вакансії;

— benefits: зараховані пільги, запропоновані роботодавцем;

— telecommuting: вірно для посад дистанційної роботи;

— hascompanylogo: правда, якщо присутній логотип компанії;

— has_questions: правда, якщо присутні контрольні запитання;

— Employment_type: повна зайнятість, частична зайнятість, контракт тощо;

— required_experience: керівник, початковий рівень, стажер тощо;

— required_education: ступінь доктора, магістра, бакалавра тощо;

— industry: автомобільна промисловість, IT, охорона здоров'я, нерухомість тощо;

— function: консалтингу, проектування, дослідження, продажів тощо;

— fraudulent: ознака класифікації.

Прогноз сонячної радіації. Набір має 32786 строк даних. Про цей набір даних:

— Сонячна радіація: ват на метр квадратний;

— Температура: градуси Фаренгейта;

— Вологість: відсоток;

— Барометричний тиск: Hg;

— Напрямок вітру: град;

— Швидкість вітру: милі на годину;

— Схід/захід сонця: гавайський час;

Класифікація якості вина. Набір має 1599 строк даних. Про цей набір даних на основі фізико-хімічних тестів:

— фіксована кислотність;

— летюча кислотність;

— лимонна кислота;

— залишковий цукор;

— хлориди;

— вільний діоксид сірки;

— сірчистий газ загальний;

— щільність;

— pH;

— сульфати;

— спирт;

— якість: "добре" і "погано" на основі балів >5 і <5.

Класифікація небажаної електронної пошти (спаму). Набір має 959 строк даних. Про цей набір даних:

— Тіло повідомлення;

— Мітки.

Класифікація фейкових новин на основі джерел. Набір має 22635 строк даних. Про цей набір даних:

— author: автор;

— title: назва/заголовок;

— text: текст;

— language: мова;

— site_url: посилання на сайт;

— type: тип статті;

— label: справжній чи підроблений;

— title_without_stopwords: заголовок без стоп-слів;

— text_without_stopwords: текст без стоп-слів;

— hasImage: чи є в статті зображення чи ні.

Набір даних IMDB із 50 тисяч оглядів фільмів. Набір має 25000 строк даних. Про цей набір даних:

— text: огляд;

— sentiment: відтінок огляду (негативний\позитивний);

—

4.3 Результати експерименту

4.3.1 Регресія

У першому експерименті модель навчалась прогнозувати серцевий напад у пацієнтів. Результаті цього навчання приведено нижче у таблицях (4.3.1, 4.3.2, 4.3.3), найкращі результати показав алгоритм – LightGbmRegressionTrainer.

Таблиця 4.3.1 — Експеримент №1. Набір даних аналізу та прогнозування серцевого нападу.

Алгоритми навчання	R-квадрат	Абсолютна втрата	Квадратична втрата	RMS-втрата
LbfgsPoissonRegressionTrainer	0,3127	0,33	0,15	0,38
LightGbmRegressionTrainer	0,4196	0,27	0,13	0,35
SdcaRegressionTrainer	0,3576	0,31	0,14	0,37
FastTreeRegressionTrainer	0,4104	0,25	0,13	0,36
FastTreeTweedieTrainer	0,3647	0,27	0,13	0,36
FastForestRegressionTrainer	0,4181	0,29	0,13	0,35

Таблиця 4.3.2 — Експеримент №1. Набір даних аналізу та прогнозування серцевого нападу.

Алгоритми навчання	R-квадрат	Абсолютна втрата	Квадратична втрата	RMS-втрата
LbfgsPoissonRegressionTrainer	0,3053	0,33	0,15	0,38
LightGbmRegressionTrainer	0,4413	0,26	0,12	0,35
SdcaRegressionTrainer	0,3854	0,29	0,13	0,36
FastTreeRegressionTrainer	0,3893	0,28	0,13	0,36
FastTreeTweedieTrainer	0,3876	0,26	0,14	0,37
FastForestRegressionTrainer	0,4101	0,27	0,13	0,35

Таблиця 4.3.3 — Експеримент №1. Набір даних аналізу та прогнозування серцевого нападу.

Алгоритми навчання	R-квадрат	Абсолютна втрата	Квадратична втрата	RMS-втрата
LbfgsPoissonRegressionTrainer	0,3181	0,33	0,15	0,38
LightGbmRegressionTrainer	0,4589	0,26	0,12	0,34
SdcaRegressionTrainer	0,3930	0,29	0,13	0,36
FastTreeRegressionTrainer	0,3485	0,29	0,14	0,37
FastTreeTweedieTrainer	0,3757	0,28	0,13	0,36
FastForestRegressionTrainer	0,4222	0,28	0,13	0,35

У другому експерименті модель навчалась прогнозувати ціни на житло. Результаті цього навчання приведено нижче у таблицях (4.3.4, 4.3.5, 4.3.6), найкращі результати показав алгоритм – LightGbmRegressionTrainer.

Таблиця 4.3.4 — Експеримент №2. Набір даних прогнозування оренди житла.

Алгоритми навчання	R-квадрат	Абсолютна втрата	Квадратична втрата	RMS-втрата
LbfgsPoissonRegressionTrainer	0,1937	13705,40	2197375564,40	46876,17
LightGbmRegressionTrainer	0,6433	15169,43	972176787,17	31179,75
SdcaRegressionTrainer	0,5249	17634,97	1294745415,08	35982,57
FastTreeRegressionTrainer	0,6448	16105,67	968111031,24	31114,48
FastTreeTweedieTrainer	0,5309	11837,06	1278450358,33	35755,42
FastForestRegressionTrainer	0,7534	12381,91	672084245,24	25924,59

Таблиця 4.3.5 — Експеримент №2. Набір даних прогнозування оренди житла.

Алгоритми навчання	R-квадрат	Абсолютна втрата	Квадратична втрата	RMS-втрата

LbfgsPoissonRegressionTrainer	0,1958	13709,97	2191803569,95	46816,70
LightGbmRegressionTrainer	0,7624	11802,29	647680909,48	25449,58
SdcaRegressionTrainer	0,5268	17598,87	1289728171,67	35912,79
FastTreeRegressionTrainer	0,7227	12262,10	755635672,89	27488,83
FastTreeTweedieTrainer	0,6908	12124,92	842770398,40	29030,51
FastForestRegressionTrainer	0,7402	12220,81	707971147,78	26607,73

Таблиця 4.3.6 — Експеримент №2. Набір даних прогнозування оренди житла.

Алгоритми навчання	R-квадрат	Абсолютна втрата	Квадратична втрата	RMS-втрата
LbfgsPoissonRegressionTrainer	0,1750	13946,20	2248510914,54	47418,47
LightGbmRegressionTrainer	0,6120	16152,19	1057531114,31	32519,70
SdcaRegressionTrainer	0,5256	17577,13	1292793978,13	35955,44
FastTreeRegressionTrainer	0,6662	13372,99	909758165,40	30162,20
FastTreeTweedieTrainer	0,7076	12465,75	796796449,18	28227,58
FastForestRegressionTrainer	0,7511	12000,55	678257447,97	26043,38

У третьому експерименті модель навчалась прогнозувати чи є оголошення про роботу справжнім чи ні. Результаті цього навчання приведено нижче у таблицях (4.3.4, 4.3.5, 4.3.6), найкращі результати показав алгоритм – LbfgsPoissonRegressionTrainer.

Таблиця 4.3.7 — Експеримент №3. Набір даних справжніх чи підроблених вакансій.

Алгоритми навчання	R-квадрат	Абсолютна втрата	Квадратична втрата	RMS-втрата

LbfgsPoissonRegressionTrainer	0,6662	0,03	0,01	0,12
LightGbmRegressionTrainer	0,1186	0,08	0,04	0,19
SdcaRegressionTrainer	-0,0005	0,09	0,04	0,21
FastTreeRegressionTrainer	0,1016	0,05	0,04	0,20
FastTreeTweedieTrainer	-1,4893	0,32	0,11	0,33
FastForestRegressionTrainer	0,6437	0,04	0,02	0,12

Таблиця 4.3.8 — Експеримент №3. Набір даних справжніх чи підроблених вакансій.

Алгоритми навчання	R-квадрат	Абсолютна втрата	Квадратична втрата	RMS-втрата
LbfgsPoissonRegressionTrainer	0,7250	0,03	0,01	0,11
LightGbmRegressionTrainer	0,1186	0,08	0,04	0,19
SdcaRegressionTrainer	-0,0005	0,09	0,04	0,21
FastTreeRegressionTrainer	0,1016	0,05	0,04	0,20
FastTreeTweedieTrainer	-0,5926	0,25	0,07	0,26
FastForestRegressionTrainer	0,6476	0,04	0,01	0,12

Таблиця 4.3.9 — Експеримент №3. Набір даних справжніх чи підроблених вакансій.

Алгоритми навчання	R-квадрат	Абсолютна втрата	Квадратична втрата	RMS-втрата
LbfgsPoissonRegressionTrainer	0,7598	0,03	0,01	0,11
LightGbmRegressionTrainer	0,1541	0,08	0,04	0,20
SdcaRegressionTrainer	0	0,09	0,05	0,22
FastTreeRegressionTrainer	0,1262	0,06	0,04	0,20
FastTreeTweedieTrainer	-1,0491	0,30	0,09	0,31
FastForestRegressionTrainer	0,6748	0,04	0,02	0,12

У четвертому експерименті модель навчалась прогнозувати рівень сонячної радіації. Результаті цього навчання приведено нижче у таблицях (4.3.10, 4.3.11, 4.3.12), найкращі результати показав алгоритм – LightGbmRegressionTrainer.

Таблиця 4.3.10 — Експеримент №4. Набір даних прогноз сонячної радіації.

Алгоритми навчання	R-квадрат	Абсолютна втрата	Квадратична втрата	RMS-втрата
LbfgsPoissonRegressionTrainer	0,8869	52,03	11263,34	106,13
LightGbmRegressionTrainer	0,3337	208,83	66337,23	257,56
SdcaRegressionTrainer	0,1803	221,31	81611,61	285,68
FastTreeRegressionTrainer	0,1481	166,00	84815,37	291,23
FastTreeTweedieTrainer	-0,3981	204,54	139190,67	373,08
FastForestRegressionTrainer	0,7998	84,27	19927,48	141,16

Таблиця 4.3.11 — Експеримент №4. Набір даних прогноз сонячної радіації.

Алгоритми навчання	R-квадрат	Абсолютна втрата	Квадратична втрата	RMS-втрата
LbfgsPoissonRegressionTrainer	0,8778	53,19	12169,67	110,32
LightGbmRegressionTrainer	0,9107	56,02	8887,88	94,28
SdcaRegressionTrainer	0,2563	215,43	74038,99	272,10
FastTreeRegressionTrainer	0,1481	166,00	84815,37	291,23
FastTreeTweedieTrainer	-0,1155	186,46	111063,84	333,26
FastForestRegressionTrainer	0,7953	86,71	20378,84	142,75

Таблиця 4.3.12 — Експеримент №4. Набір даних прогноз сонячної радіації.

Алгоритми навчання	R-квадрат	Абсолютна втрата	Квадратична втрата	RMS-втрата
LbfgsPoissonRegressionTrainer	0,6258	112,03	38847,60	197,10
LightGbmRegressionTrainer	0,8775	66,97	12718,15	112,77
SdcaRegressionTrainer	0,3286	210,29	69705,80	264,02
FastTreeRegressionTrainer	0,8799	66,78	12474,51	111,69
FastTreeTweedieTrainer	0,7291	77,25	28132,00	167,73
FastForestRegressionTrainer	0,6785	115,66	33379,88	182,70

4.3.2 Двійкова класифікація

У п’ятому експерименті модель навчалась класифікувати якість вина. Результаті цього навчання приведено нижче у таблицях (4.3.13, 4.3.14, 4.3.15), найкращі результати показав алгоритм – FastTreeBinaryTrainer.

Таблиця 4.3.13 — Експеримент №5. Набір даних класифікації якості вина.

Алгоритми навчання	Мікроточність	Макроточність
SdcaLogisticRegressionBinaryTrainer	0,4969	0,5000
LbfgsLogisticRegressionBinaryTrainer	0,7673	0,7676
LightGbmBinaryTrainer	0,7673	0,7673
FastTreeBinaryTrainer	0,7987	0,7989
FastForestBinaryTrainer	0,7987	0,7991
SdcaMaximumEntropyMulti	0,4969	0,5000
LbfgsMaximumEntropyMulti	0,7610	0,7612

Таблиця 4.3.14 — Експеримент №5. Набір даних класифікації якості вина.

Алгоритми навчання	Мікроточність	Макроточність
SdcaLogisticRegressionBinaryTrainer	0,4969	0,5000
LbfgsLogisticRegressionBinaryTrainer	0,7610	0,7613
LightGbmBinaryTrainer	0,7107	0,7111
FastTreeBinaryTrainer	0,8050	0,8051
FastForestBinaryTrainer	0,7862	0,7864
SdcaMaximumEntropyMulti	0,4969	0,5000
LbfgsMaximumEntropyMulti	0,7610	0,7612

Таблиця 4.3.15 — Експеримент №5. Набір даних класифікації якості вина.

Алгоритми навчання	Мікроточність	Макроточність
SdcaLogisticRegressionBinaryTrainer	0,4843	0,5000
LbfgsLogisticRegressionBinaryTrainer	0,7862	0,7871
LightGbmBinaryTrainer	0,7987	0,7993
FastTreeBinaryTrainer	0,8491	0,8505
FastForestBinaryTrainer	0,7987	0,7997
SdcaMaximumEntropyMulti	0,4843	0,5000
LbfgsMaximumEntropyMulti	0,7862	0,7871

У шостому експерименті модель навчалась класифікувати електрону пошту. Результаті цього навчання приведено нижче у таблицях (4.3.16, 4.3.17, 4.3.18), найкращі результати показав алгоритм – LbfgsMaximumEntropyMulti.

Таблиця 4.3.16 — Експеримент №6. Набір даних класифікація електронної пошти.

Алгоритми навчання	Мікроточність	Макроточність
SdcaLogisticRegressionBinaryTrainer	0,4063	0,5000
LbfgsLogisticRegressionBinaryTrainer	0,9876	0,9902
LightGbmBinaryTrainer	0,7949	0,7633
FastTreeBinaryTrainer	0,8423	0,8531
FastForestBinaryTrainer	0,8286	0,8023
SdcaMaximumEntropyMulti	0,4063	0,5000
LbfgsMaximumEntropyMulti	0,9876	0,9902

Таблиця 4.3.17 — Експеримент №6. Набір даних класифікація електронної пошти.

Алгоритми навчання	Мікроточність	Макроточність
SdcaLogisticRegressionBinaryTrainer	0,4063	0,5000
LbfgsLogisticRegressionBinaryTrainer	0,9622	0,9535
LightGbmBinaryTrainer	0,7883	0,7501
FastTreeBinaryTrainer	0,9103	0,9194
FastForestBinaryTrainer	0,8665	0,8517
SdcaMaximumEntropyMulti	0,4063	0,5000
LbfgsMaximumEntropyMulti	0,9876	0,9902

Таблиця 4.3.18 — Експеримент №6. Набір даних класифікація електронної пошти.

Алгоритми навчання	Мікроточність	Макроточність
SdcaLogisticRegressionBinaryTrainer	0,1274	0,5000
LbfgsLogisticRegressionBinaryTrainer	0,9771	0,9139
LightGbmBinaryTrainer	0,8726	0,5000
FastTreeBinaryTrainer	0,9665	0,8892
FastForestBinaryTrainer	0,9660	0,8971
SdcaMaximumEntropyMulti	0,1274	0,5000
LbfgsMaximumEntropyMulti	0,9772	0,9142

У сьомому експерименті модель навчалась класифікувати фейкові новини. Результаті цього навчання приведено нижче у таблицях (4.3.19, 4.3.20, 4.3.21),

найкращі результати показали два алгоритми LbfgsLogisticRegressionBinaryTrainer та LbfgsMaximumEntropyMulti.

Таблиця 4.3.19 — Експеримент №7. Набір даних класифікація фейкових новин на основі джерел.

Алгоритми навчання	Мікроточність	Макроточність
SdcaLogisticRegressionBinaryTrainer	0,8996	0,7765
LbfgsLogisticRegressionBinaryTrainer	0,9694	0,9870
LightGbmBinaryTrainer	0,9694	0,9870
FastTreeBinaryTrainer	0,9694	0,9870
FastForestBinaryTrainer	0,9651	0,9739
SdcaMaximumEntropyMulti	0,9563	0,9476
LbfgsMaximumEntropyMulti	0,9694	0,9870

Таблиця 4.3.20 — Експеримент №7. Набір даних класифікація фейкових новин на основі джерел.

Алгоритми навчання	Мікроточність	Макроточність
SdcaLogisticRegressionBinaryTrainer	0,9340	0,8245
LbfgsLogisticRegressionBinaryTrainer	0,9858	0,9863
LightGbmBinaryTrainer	0,9857	0,9863
FastTreeBinaryTrainer	0,9857	0,9863
FastForestBinaryTrainer	0,9811	0,9716
SdcaMaximumEntropyMulti	0,9670	0,9275
LbfgsMaximumEntropyMulti	0,9858	0,9863

Таблиця 4.3.21 — Експеримент №7. Набір даних класифікація фейкових новин на основі джерел.

Алгоритми навчання	Мікроточність	Макроточність
SdcaLogisticRegressionBinaryTrainer	0,5895	0,2500
LbfgsLogisticRegressionBinaryTrainer	0,9694	0,9870
LightGbmBinaryTrainer	0,9476	0,9381
FastTreeBinaryTrainer	0,9651	0,9823
FastForestBinaryTrainer	0,9563	0,9476
SdcaMaximumEntropyMulti	0,9563	0,9476
LbfgsMaximumEntropyMulti	0,9694	0,9870

У восьмому експерименті модель навчалась класифікувати настрій рецензії на фільм. Результаті цього навчання приведено нижче у таблицях (4.3.22, 4.3.23, 4.3.24), найкращі результат показав алгоритм - LbfgsLogisticRegressionBinaryTrainer.

Таблиця 4.3.22 — Експеримент №8. Набір даних IMDB із 50 тисяч оглядів фільмів.

Алгоритми навчання	Мікроточність	Макроточність
SdcaLogisticRegressionBinaryTrainer	0,4979	0,5000
LbfgsLogisticRegressionBinaryTrainer	0,8911	0,8910
LightGbmBinaryTrainer	0,6840	0,6832
FastTreeBinaryTrainer	0,6840	0,6831
FastForestBinaryTrainer	0,7447	0,7446
SdcaMaximumEntropyMulti	0,4979	0,5000
LbfgsMaximumEntropyMulti	0,8899	0,8899

Таблиця 4.3.23 — Експеримент №8. Набір даних IMDB із 50 тисяч оглядів фільмів.

Алгоритми навчання	Мікроточність	Макроточність
SdcaLogisticRegressionBinaryTrainer	0,4973	0,5000
LbfgsLogisticRegressionBinaryTrainer	0,8911	0,8910
LightGbmBinaryTrainer	0,6840	0,6832
FastTreeBinaryTrainer	0,6840	0,6831
FastForestBinaryTrainer	0,7439	0,7446
SdcaMaximumEntropyMulti	0,4979	0,5000
LbfgsMaximumEntropyMulti	0,8898	0,8899

Таблиця 4.3.24 — Експеримент №8. Набір даних IMDB із 50 тисяч оглядів фільмів.

Алгоритми навчання	Мікроточність	Макроточність
SdcaLogisticRegressionBinaryTrainer	0,4979	0,5000
LbfgsLogisticRegressionBinaryTrainer	0,8911	0,8910
LightGbmBinaryTrainer	0,7227	0,7223
FastTreeBinaryTrainer	0,7841	0,7840
FastForestBinaryTrainer	0,7509	0,7507
SdcaMaximumEntropyMulti	0,4979	0,5000
LbfgsMaximumEntropyMulti	0,8898	0,8898

4.3.3 Висновки результатів експерименту

Діаграма з результатами 1 експерименту:

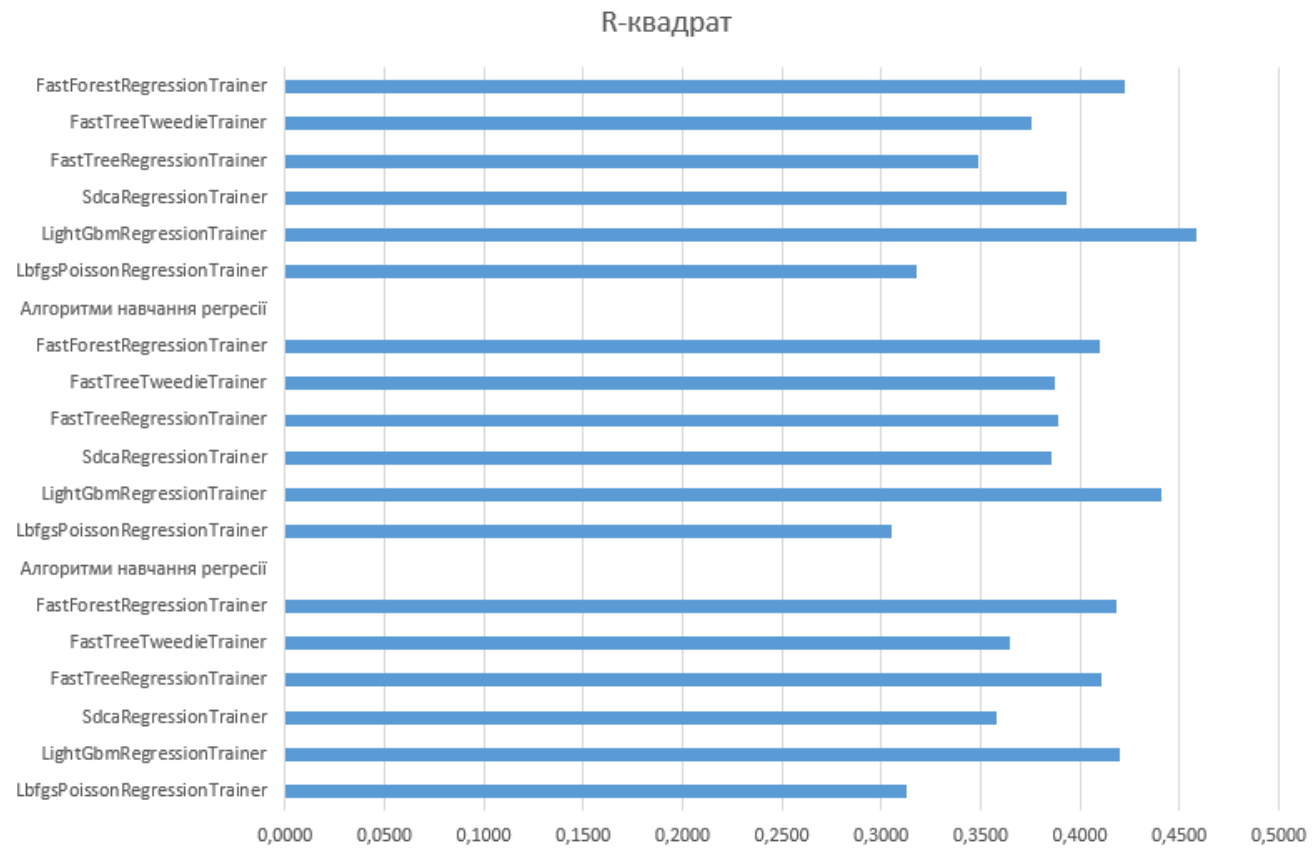

Рис.27 Діаграма експерименту №1

Діаграма з результатами 2 експерименту:

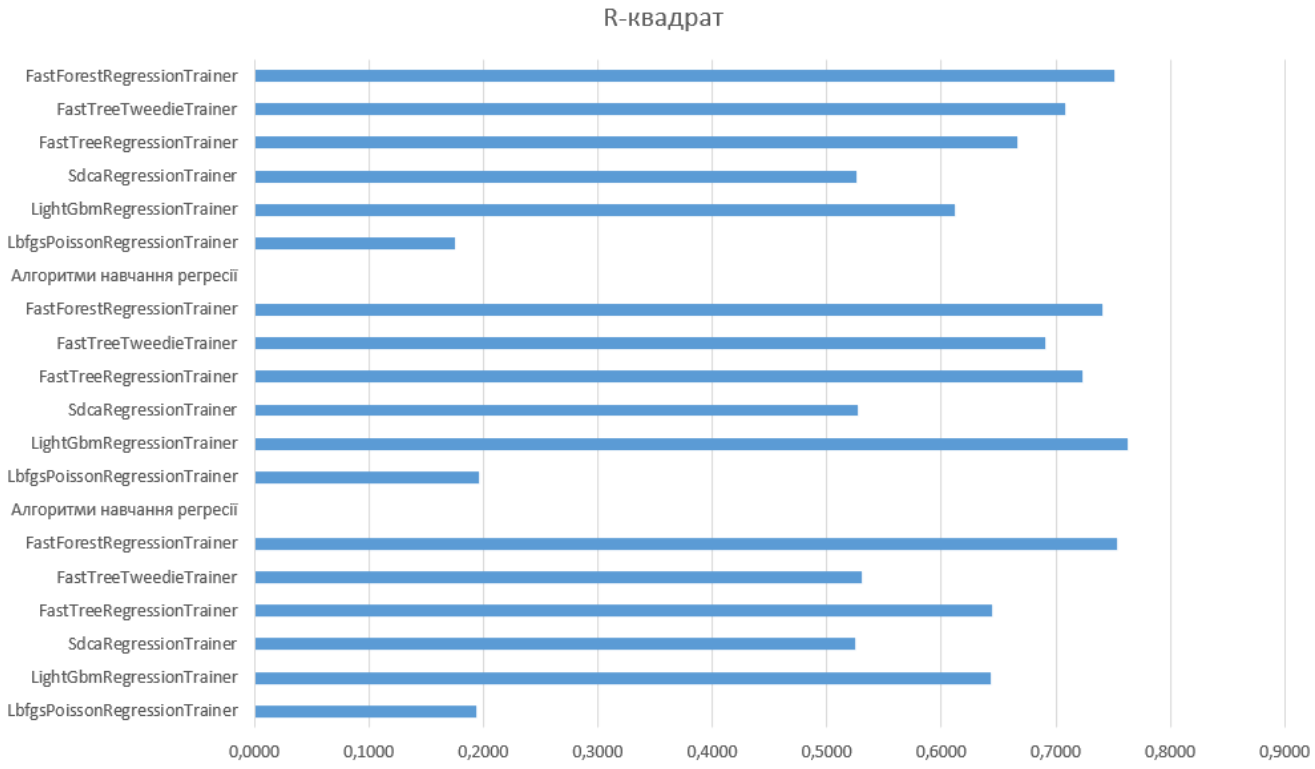

Рис.28 Діаграма експерименту №2

Діаграма з результатами 3 експерименту:

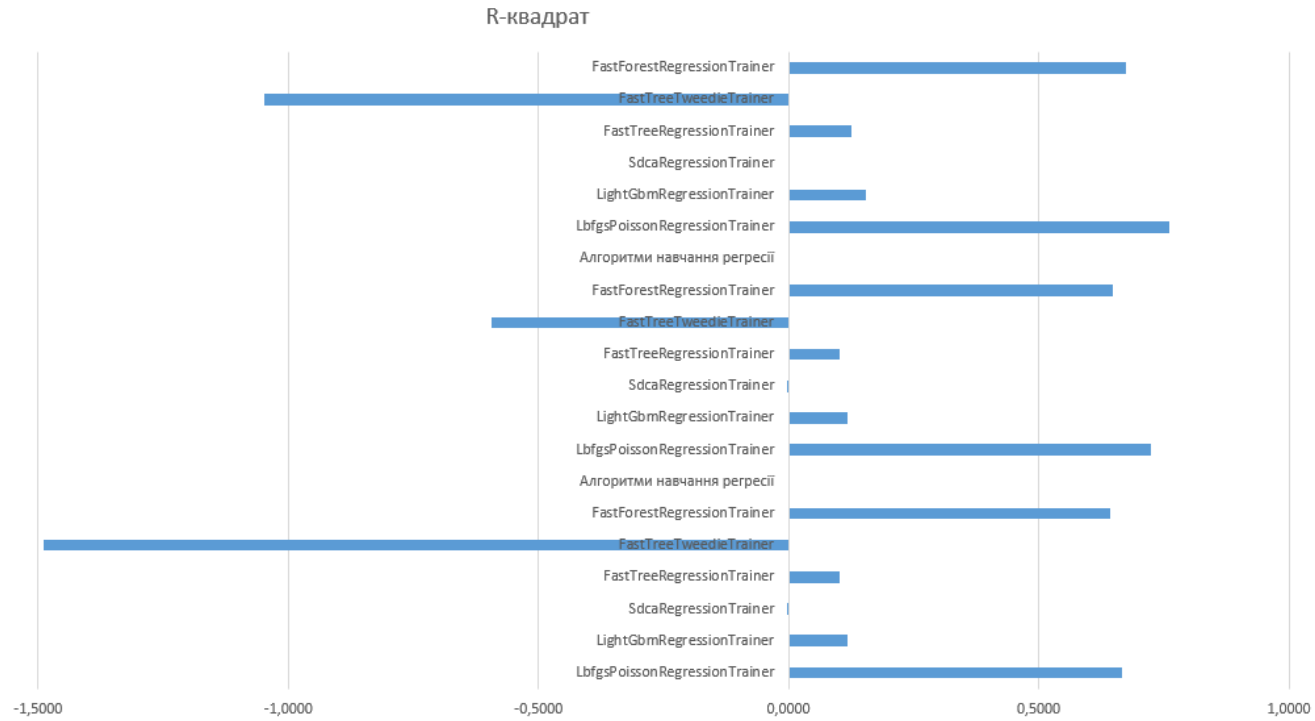

Рис.29 Діаграма експерименту №3

Діаграма з результатами 4 експерименту:

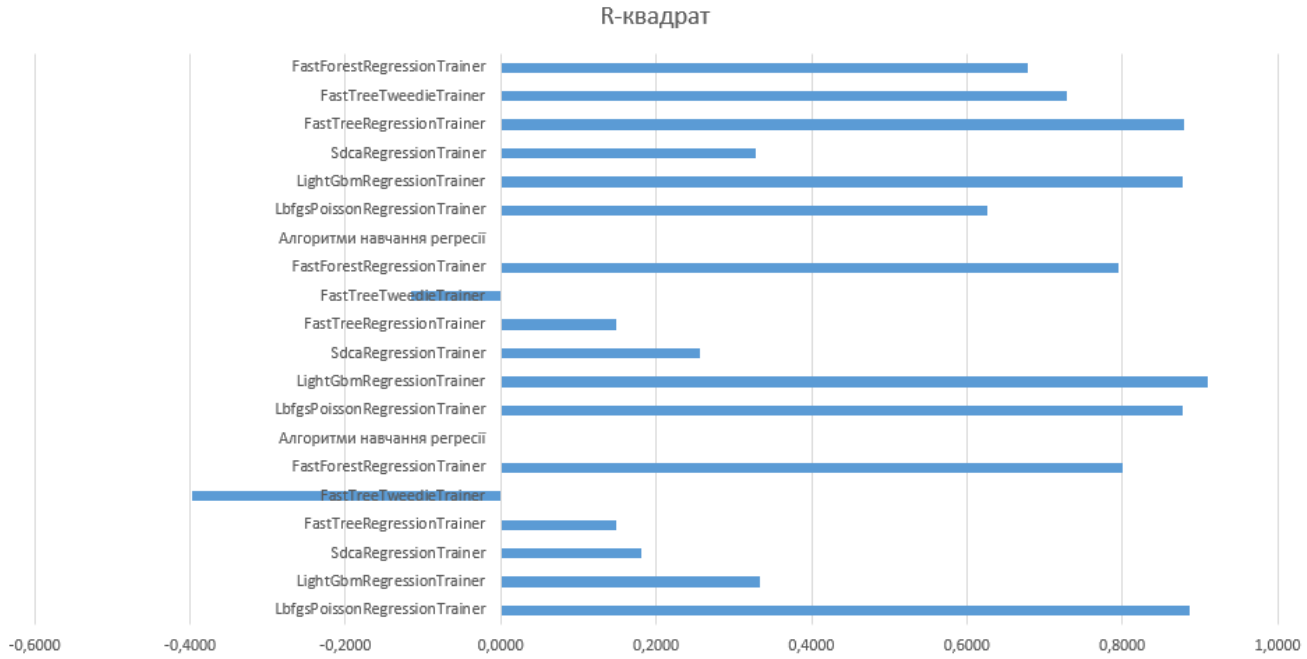

Рис.30 Діаграма експерименту №4

Діаграма з результатами 5 експерименту:

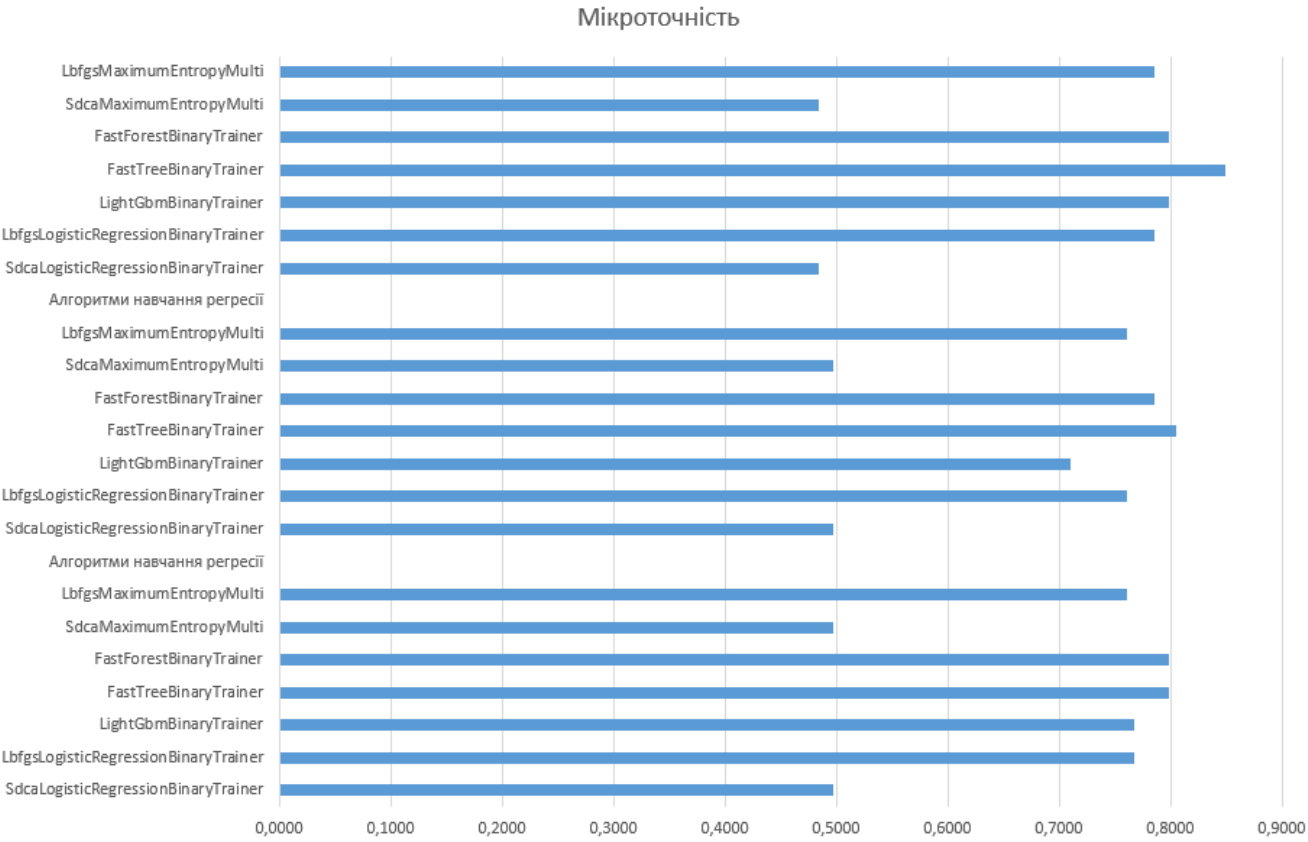

Рис.31 Діаграма експерименту №5

Діаграма з результатами 6 експерименту:

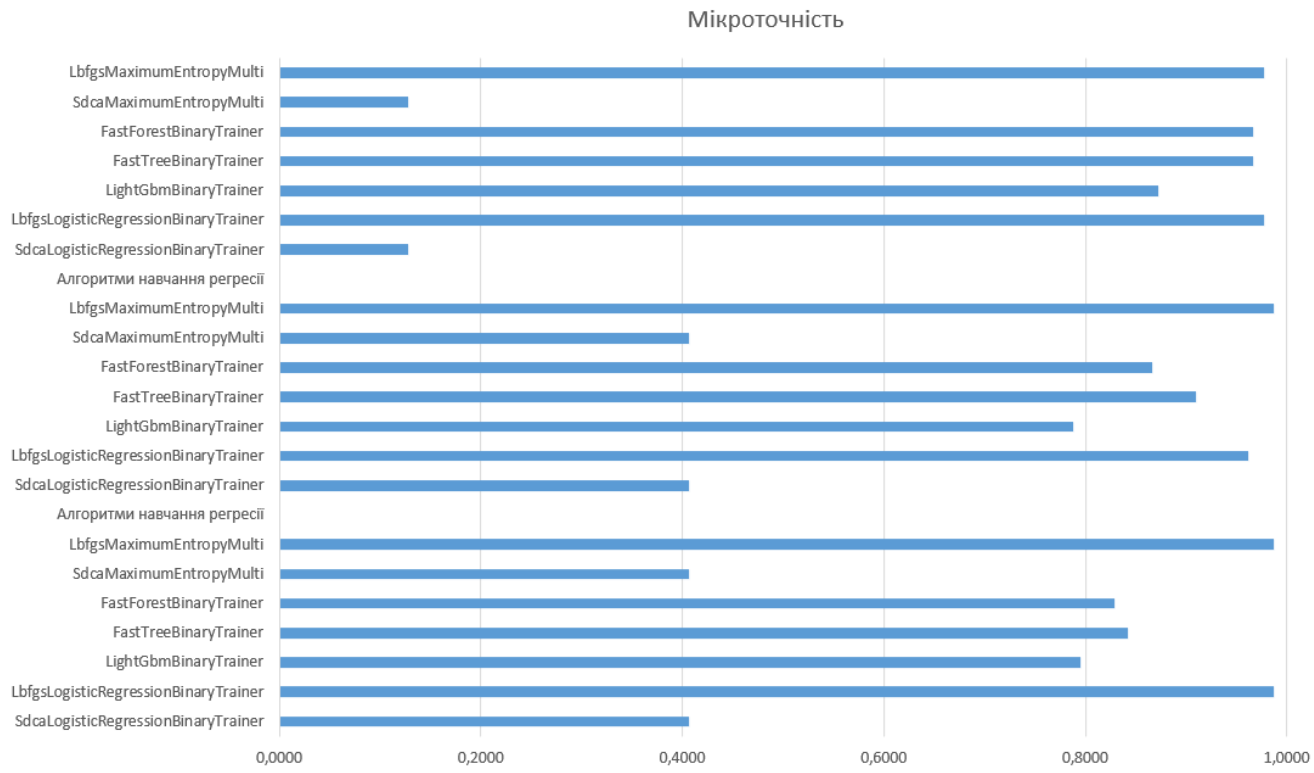

Рис.32 Діаграма експерименту №6

Діаграма з результатами 7 експерименту:

Рис.33 Діаграма експерименту №7

Діаграма з результатами 8 експерименту:

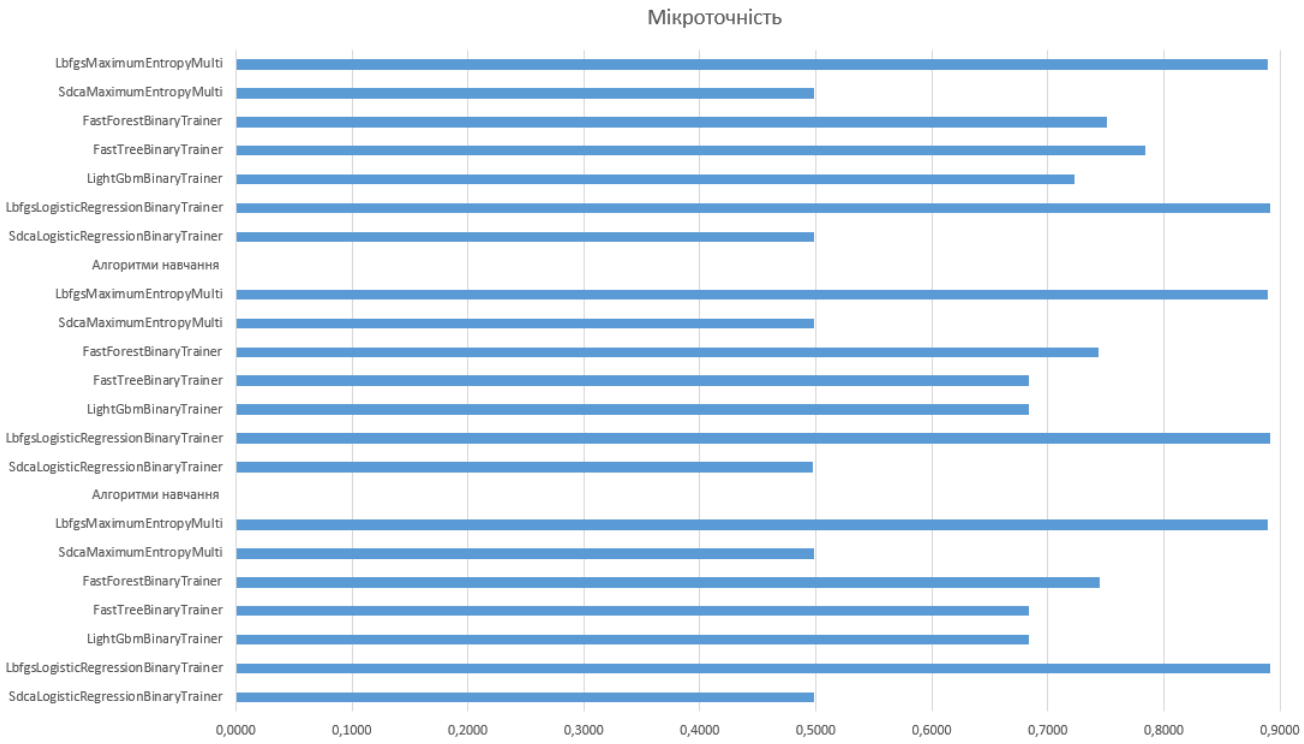

Рис.34 Діаграма експерименту №8

Підвівши підсумки першого, другого, третього, четвертого експериментів, можна сказати, що краще всього в прогнозуванні значення мітки набору пов'язаних компонентів показує себе алгоритм FastForestRegressionTrainer.

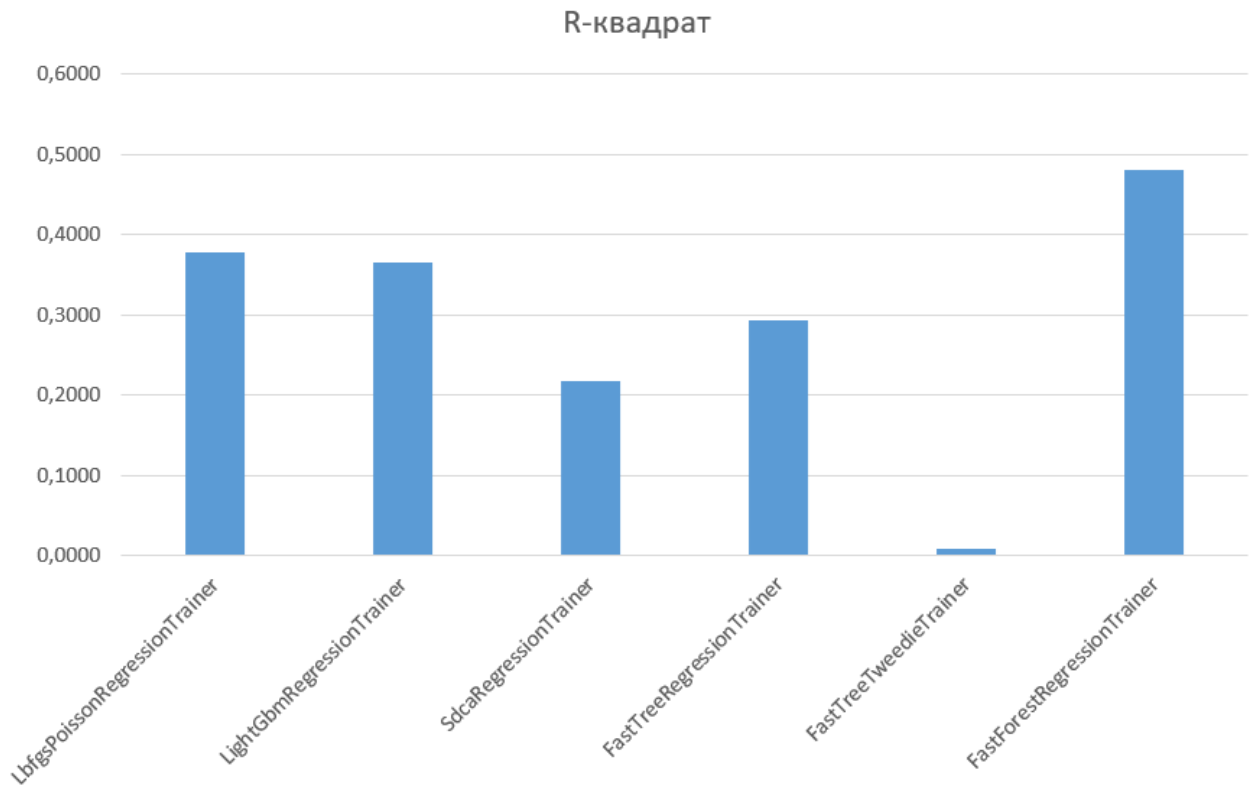

Рис.35 Діаграма сценарію регресії

Підвівши підсумки п’ятого, шостого, сьомого, восьмого експериментів, можна сказати, що краще всього в прогнозуванні розподілу екземплярів даних за двома або кількома класами (категоріями). показують себе алгоритми LbfgsLogisticRegressionBinaryTrainer та LbfgsMaximumEntropyMulti.

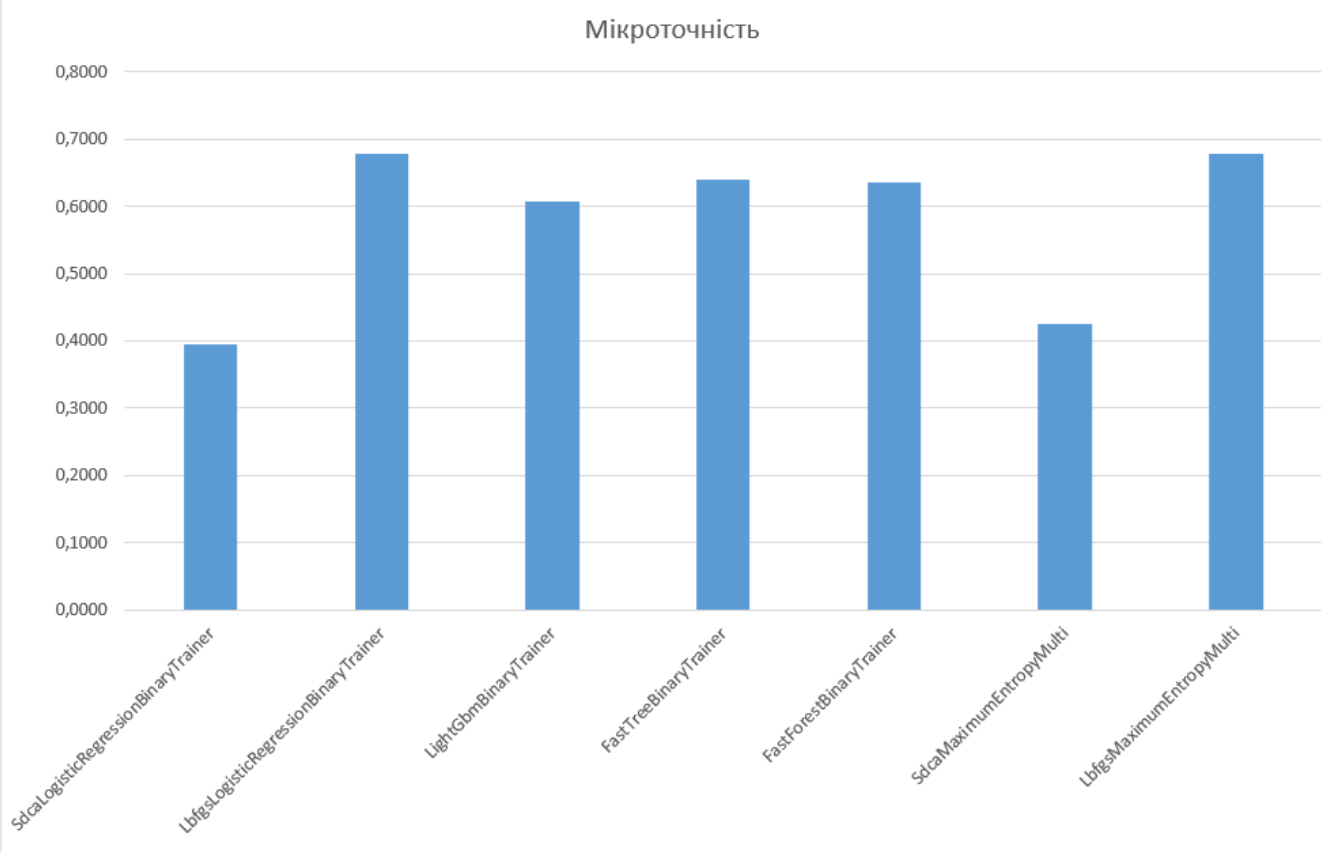

Рис.35 Діаграма сценарію класифікацій

5 ТЕХНІКО–ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ПРОЕКТУ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Техніко-економічне обґрунтування – це комплект розрахункових та аналітичних документів, які включають у себе основні технічні, організаційні, розрахункові та інші показники, що дозволяють оцінити доцільність та економічну стійкість проекту.

Метою такого обґрунтування є доказ того, що обрані технології, затверджений процес виробництва та організація роботи є оптимальною. Також важливо аргументувати, що даний проект не буде збитком на економічному

рівні.

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях.

$$E = a \times S^b \times EAF \quad (5.1)$$

де S^b – обсяг коду;

E – витрати праці на проект;

EAF – фактор уточнення витрат.

Для простих систем, a = 2,4; b = 1,05.

Розмір програмного коду було підраховано за допомогою інструменту вбудованного у Visual Studio 2022 Community – Code Metrics Results.

Загальний обсяг програмного коду становить 130 рядків (мала кількість коду обумовлена використанням готової бібліотеки) (рис. 36).

Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Source code	Lines of Executable code
MastersWork (Debug)	79	32	1	42	130	74
myApp	79	32	1	42	130	74
Program.ModelOutput	100	2	1	1	8	2
Program.ModelInput	78	28	1	2	60	56
Program	60	2	1	40	127	16

Рисунок 36 – Code Metrics Results

$$E = 2{,}4 \times 0{,}130^{1{,}05} \times 1 = 0{,}281$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 0,281 людино-місяці.

Нижче наведені розрахунки вартості розробки програмного забезпечення «Дослідження алгоритмів у відкритому фреймворку ML.NET». Основними витратами визначено такі критерії як заробітна плата, відрахування на соціальні потреби, накладні та додаткові витрати, витрати на персональний комп’ютер на програмні засоби.

Розрахуємо заробітну плату інженера-програміста, для оцінки основної заробітної плати. Згідно статистиці сайту вакансій Dou, середня зарплатня інженера-програміста на 2022 рік, становить близько 26400 гривень, а мінімальна 20000 гривень, максимальне значення – близько 34600 гривень. Графік заробітної плати (рис. 37).

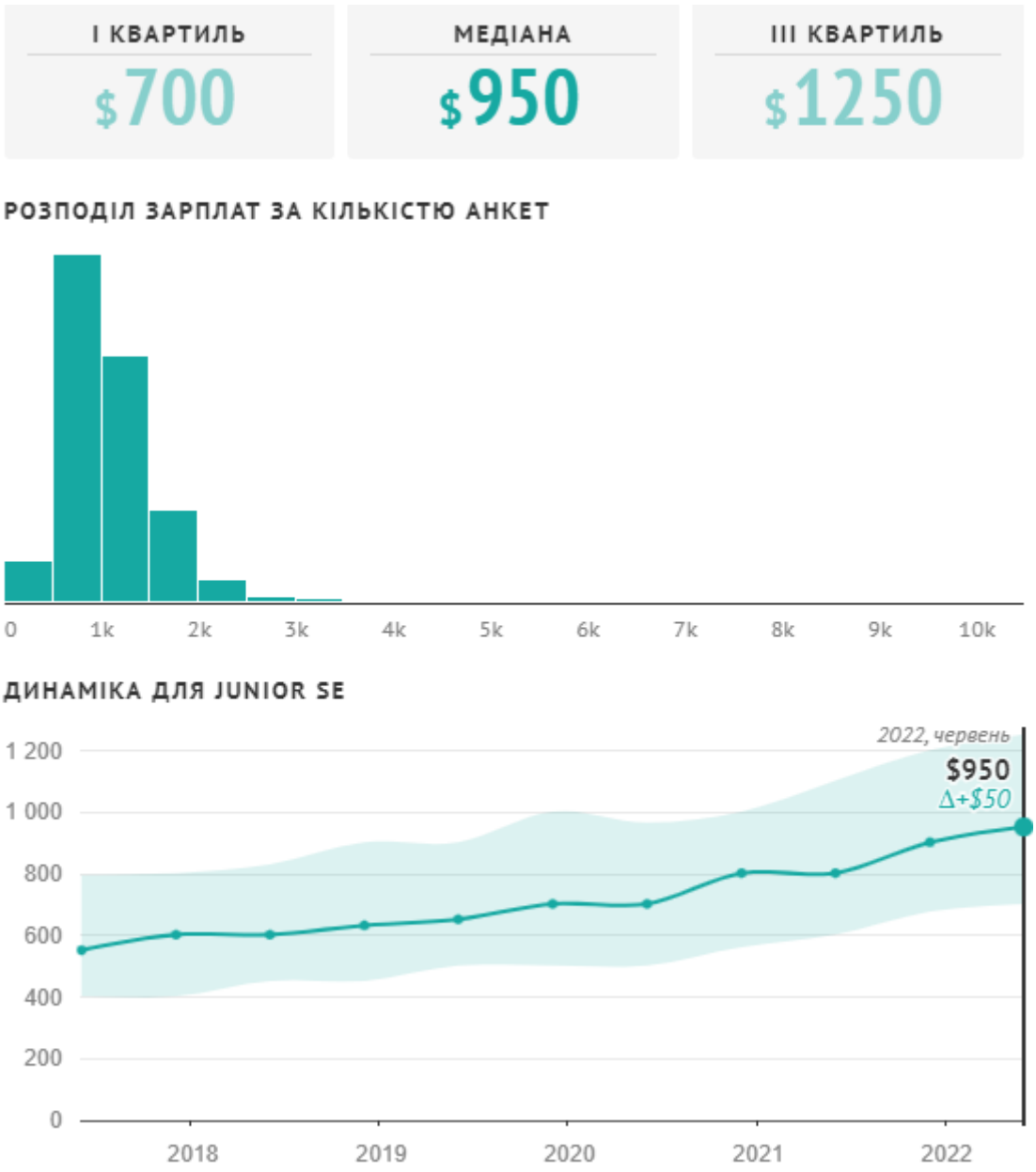

Рисунок 37 – Зарплатня інженера-програміста за Dou

Розрахунок заробітної платні проводиться по формі (табл. 5.1.)

Таблиця 5.1 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати грн
			чол	місяців	

1	інженер-програміст	26400	1	0,281	7418,4

Описаний в проекті програмний додаток був розроблений одним програмістом в період з 01.10.22 до 01.12.22, що становить 4 робочих тижні. Витрати робочого часу прийняті за 40 годин у тиждень. Погодинна ставка кваліфікованого інженера–програміста складає 30,91 грн/год, згідно середній зарплаті. Таким чином, витрачено робочого часу:

$$t_{розробки} = N_{чол} \times N_{тиж} \times N_{год} \quad (5.2)$$

де $N_{чол}$ – кількість виконавців, чол;

$N_{тиж}$ – тривалість розробки;

$N_{год}$ – витрати робочого часу, год;

$$t_{розробки} = 1 \times 4 \times 40 = 160 \text{ чол/год} \quad (5.2)$$

ОЗП визначається за формулою:

$$ОЗП = t_{розробки} \times N \times ККВ \quad (5.3)$$

де $t_{розробки}$ – витрати праці у чол/год;

N – погодинна ставка;

ККВ – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. Коефіцієнт кваліфікації розробника (k) – ступінь підготовленості виконавця до дорученої йому роботі (він визначається залежність від стажу праці та становить:

– для працюючих до 2 років- 0,75;

– від 2 до 3 років 1,0;

– від 3 до 5 років - 1,1-1,2;

– від 5 до 7 років - 1,3-1,4;

– понад 7 років - 1,5-1,6.

В даному випадку $ККВ$ приймається 0,75. ОЗП складає:

$$ОЗП = 160 \times 30{,}91 \times 0{,}75 = 3709{,}2$$

Єдиний соціальний внесок на 2022 рік складає 22%. Відрахування на соціальні потреби встановлюються у відсотках від суми заробітної плати:

$$С_{соц} = \frac{ОЗП \times 22\%}{100\%} \quad (5.4)$$

$$С_{соц} = \frac{3709{,}2 \times 22\%}{100\%} = 816 \text{ грн.} \quad (5.4)$$

Отримані результати підсумовуються. Вони складають 4525 грн. та визначають основні прямі витрати.

Накладні витрати враховують такі витрати для роботи над проектом як: опалення, електроенергія, амортизація будівель, зарплату адміністративного персоналу і т.д.

$$С_{накл} = \frac{(ОЗП + С_{соц}) \times 35\%}{100\%} \quad (5.5)$$

$$С_{накл} = \frac{(3709{,}2 + 816) \times 35\%}{100\%} = 1583{,}75 \text{ грн.} \quad (5.5)$$

Протягом усього терміну використання нової техніки підприємство щорічно витрачає певні кошти, пов’язані з її експлуатацією.

Експлуатаційні витрати на персональний комп’ютер визначаються протягом терміну розробки програмного засобу в залежності від вартості комп’ютеру. В експлуатаційні витрати входять:

- вартість витратних матеріалів;
- витрати на ремонт;
- заробітна плата ремонтника;
- оренда приміщення;

– додаткові витрати – прибирання приміщення, охорона, оренда, комунальні послуги;

– амортизаційні витрати на персональний комп'ютер і програмне забезпечення;

– витрати на електроенергію ($С_{ел}$), які визначаються за формулою:

$$С_{ел} = Р \times В \times Т_{розр} \quad (5.6)$$

– Р – потужність комп’ютера та допоміжних споживачів електричної енергії, приймається 0,45 кВт/год;

– В – вартість 1 кВт/годин для непобутових споживачів, становить близько 1,68 грн [4];

– $Т_{розр}$ – час роботи з ЕВМ, приймається рівним робочому часу.

Витрати на електроенергію визначаються так:

$$С_{ел} = 0{,}45 \times 1{,}68 \times 160 = \ 121 \text{ грн.} \quad (5.6)$$

Витрати на витратні матеріали ($С_{вм}$) протягом всього терміну експлуатації приблизно 10% від вартості комп’ютеру. Вартість робочої станції приймається 30000 грн. (беремо середній за ціною портативний ноутбук), термін експлуатації – 5 років, відповідно до Податкового кодексу України [6]. Отже, можна визначити ці витрати за період створення програмного засобу:

$$С_{вм} = В_{ком} \times \frac{N_Д}{N_{експ} \times 365} \times \frac{10\%}{100\%} \quad (5.7)$$

де $B_{КОМ}$ – вартість персонального комп’ютеру;

$N_Д$ – кількість днів розробки програмного продукту;

$N_{експ}$ – термін експлуатації персонального комп’ютеру.

Витрати на витратні матеріали визначаються так:

$$С_{вм} = 30000 \times \frac{30}{5 \times 365} \times \frac{10\%}{100\%} = 49 \text{ грн.} \quad (5.7)$$

Заробітна плата ремонтника ($C_{рем}$) визначена наступним чином: на ремонт 50 комп’ютерів потрібен один інженер-системотехнік. За даними порталу Dou, середньомісячна заробітна плата інженера-системотехніка в Україні на 2022 рік становить 29000 грн.

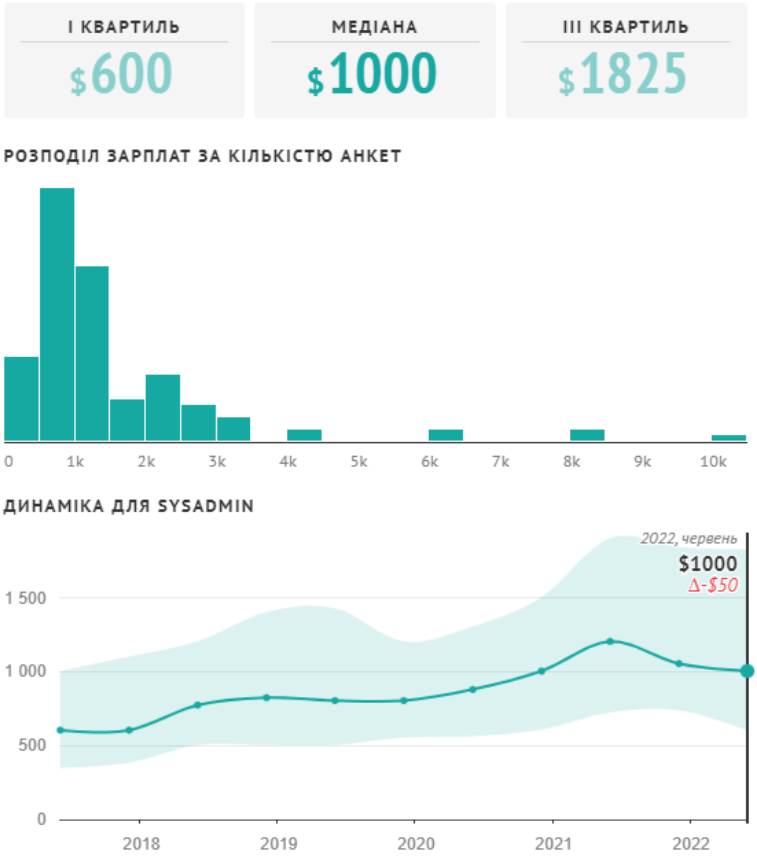

Рисунок 38 – Статистика зарплати інженера Dou

Тоді в перерахунку на один комп'ютер його заробітна плата за період розробки програмного продукту складає:

$$C_{рем} = \frac{C_{рем}'}{N_{ком}} \cdot T_{міс} \qquad (5.8)$$

де $C_{рем}'$ – середньомісячна заробітна плата;

$N_{КОМ}$ – кількість комп'ютерів на одного ремонтника.

$T_{мес}$ – час розробки програмного продукта, міс.

Заробітна плата ремонтника ($C_{рем}$) буде складати:

$$C_{рем} = \frac{29000}{50} \cdot 0{,}281 = 163 \text{ грн.} \qquad (5.8)$$

За статистикою витрати на комплектуючи вироби (СКОМ) для ремонту

персонального комп’ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$СКОМ = СВМ = 49 \text{ грн.} \quad (5.9)$$

Амортизаційні відрахування на персональний комп’ютер визначені з положення, що амортизаційний період в даний час дорівнює терміну морального старіння обчислювальної техніки і складає приблизно 5 роки, що являється терміном корисного використання техніки.

$$АКП = В_{КОМ} \times \frac{N_Д}{N_{експ} \times 365} \quad (5.10)$$

$$АКП = 30000 \times \frac{0,281}{5 \times 12} = 234,17 \text{ грн.} \quad (5.10)$$

Для функціонування персонального комп’ютера використовувалася операційна система Windows 11 Home, для написання програмного забезпечення – програмне середовище VisualStudio Community 2022, яка надана користувачам безкоштовно, тому приймаємо його значення за 0.

$$АКП = 5000 \times \frac{0,281}{5 \times 12} = 23,41 \text{ грн.} \quad (5.10)$$

Розрахунок амортизаційних відрахувань на програмне забезпечення зведений в табл. 5.2. Додаткові витрати ($C_{ДОД}$): прибирання приміщень, охорона, комунальні послуги важко оцінити точно і прийняти рівними 50% заробітної плати інженера-програміста, тобто 3700 гривень на місяць.

Оренду приміщень для однієї людини приймемо рівною 3000 гривень на місяць, з квадратурою 20м2 та 50м2. Тобто за весь період розробки (0,281) – 843 грн. Сумарні експлуатаційні витрати на один персональний комп’ютер складають:

$$С_{експ} = С_{ел} + С_{ВМ} + С_{рем} + АКП + АПО + С_{ор} + С_{дод} \quad (5.11)$$

$$С_{експ} = 121 + 49 + 163 + 234{,}17 + 23{,}41 + 843 + 3700 = 5134 \text{ грн.} \quad (5.11)$$

Результати розрахунків зведено у табл. 5.2 - 5.3.

Таблиця 5.2 – Використовуване програмне забезпечення

Найменування програмного забезпечення	Вартість програмного забезпечення, грн	Джерело придбання	Амортизаційні відрахування, грн
Windows 10	5000	https://soft.rozetka.com.ua/ua/microsoft_kw9_00661/p323700280/?gclid=CjwKCAiAheacBhB8EiwAItVO2-YEpZsIkCobqktRwRuzIH4RR_FTFmnO_hVaiEK8cxgD9TwN84ZU7hoC7RMQAvD_BwE	23,41
PHPStorm 2019	0	https://www.jetbrains.com/ru-ru/phpstorm/download/#section=windows	0
Всього:	5000		23,41

Таблиця 5.3 – Експлуатаційні витрати на ПК і ПЗ.

Найменування витрат	Витрати, грн
Витрати на електроенергію	121
Вартість витратних матеріалів	49
Витрати на ремонт	163
Амортизація персонального комп’ютера	234,17
Амортизація програмного забезпечення	23,41
Оренда приміщення	843

Додаткові витрати	3700
Всього	5134

Таким чином, витрати на створення програмного продукту складають:

$$С_{розробки} = ОЗП + С_{соц} + С_{накл} + С_{експ} \quad (5.12)$$

$$С_{розробки} = 3709{,}2 + 816 + 1583{,}75 + 5134 = 11243 \text{ грн.} \quad (5.12)$$

Розрахунок витрат зведено у табл. 5.4.

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, грн
Основна заробітна плата	3709,2
Відрахування на соціальні потреби	816
Накладні витрати	1583,75
Експлуатаційні витрати	5134
Всього	11243

За результатами розрахунків, приблизна вартість розробки складає 11243 грн.

6 ЗАГАЛЬНИЙ ВИСНОВОК

Під час виконання роботи було досліджено роботу алгоритмів у відкритому фреймворку ML.NET. Було детально розглянуто принципи та ідеї, на яких побудована машинне навчання.

У роботі було проаналізовано два види сценарію машинного навчання, сценарій регресії та сценарій багато класової класифікації. Було проведено 12 експериментів, по 6 експериментів за кожен сценарій навчання машини.

Було приділено багато уваги тестовим даним для екскрементів. Дані підбиралися різних розмірів та різних за типом наповнення.

У роботі також було розглянуто аналогічні фреймворки для розробки машинного навчання.

7 БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. тра-нсп. ім. акад. В. Лазаряна, 2009. - 38 с.
2. НПАОП 0.00-7.15-18. Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями.
3. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень.
4. Інженерія програмного забезпечення [Текст] : навчальний посібник / В. І. Шинкаренко, О. В. Горбова, О. П. Іванов, В. О. Андрющенко, В. Я. Нечай; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Дніпро, 2019. – 140 с.
5. ML.Net // Документація ML.NET // https://learn.microsoft.com/ru-ru/dotnet/machine-learning/how-does-mldotnet-work (дата звернення: 01.12.2021).
6. Робота: програміст в Україні. Вакансії і робота — Dou.ua // Сервіс пошуку роботи Dou.ua. URL: https://dou.ua/ (дата звернення: 02.12.2021).
7. Мінфін. Нові тарифи на електроенергію. https://index.minfin.com.ua/ua/tariff/electric/index.php/
8. Компьютерные комплектующие // Інтернет-магазин техніки Rozetka. URL: https://hard.rozetka.com.ua/ua/ (дата звернення: 03.12.2021).
9. Мінфін // Сервіс конвертації валют Мінфін. URL: https://www.xe.com/currencyconverter/convert/?Amount=1&From=USD&To=UAH (дата звернення: 01.12.2021).
10. Аренда офиса Днепр, снять офис — объявления OLX.ua Днепр // Сервіс оголошень OLX. URL: https://www.olx.ua/d/uk/nedvizhimost/kommercheskaya-nedvizhimost/arenda-

kommercheskoy-nedvizhimosti/ofsn-primschennya/dnepr/ (дата звернення: 03.12.2021)

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
18.12.22

«TOOLKIT ML.NET»

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.0170437-01

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
18.12.22

Керівник розробки
_____ Олександр ПЕТРОВИЧ
18.12.22

Виконавець
_____ Андрій БУРЦЕВ
18.12.22

Нормоконтролер
_____ Світлана ВОЛКОВА
18.12.22

2022

ЗАТВЕРДЖЕНО

1116130.0170437-01

«TOOLKIT ML.NET»
Технічне завдання

1116130.0170437-01

Листів 7

2022

ЗМІСТ

ВСТУП

Програмний комплекс ««TOOLKIT ML.NET» являє собою набір утиліт, призначених для дослідження алгоритмів у відкритому фреймворку ML.NET. Даний програмний комплекс дасть нам змогу отримати характеристики навченої моделі машинного навчання.

Необхідність розробки даного програмного забезпечення обумовлена недостатністю викладення методології проведення аналогічних досліджень в існуючих статтях та наукових роботах, присвячених ML.NET.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Основою для розробки є наказ ректора Українського державного університету науки і технології Радкевич А.В. «Про затвердження тем та призначення керівників дипломних проектів» №01206 ст. від 10.12.2022 р.

Тема проекту: «TOOLKIT ML.NET».

Керівник дипломного проекту: Іванов Олександр Петрович

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення програми полягає у виконанні багатократних замірів точності алгоритмів навчання моделі. А саме алгоритмів які задіяні у сценаріях регресії та двійкової класифікацій.

Експлуатаційне призначення програми полягає у наданні наглядного статистичного аналізу точності алгоритмів навчання моделі.

3. ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Продукт повинен являти собою систему з підтримкою обробки даних з файлів типу «.csv». Серед необхідних функцій:

— Можливість обрати потрібний сценарій навчання

— Можливість отримати необхідний алгоритм навчання

Формат вхідних даних – файл з типом розширення «.csv».

Формат вихідних – дані у числовому форматі, які показують точність моделі.

3.2 Вимоги до надійності

Програма повинна виводити помилку користувачу при будь яких нештатних помилках. Також програма повинна попереджувати користувача про не вірні данні які знаходяться у файлі з даним.

Апаратна частина представлена серверами, що розташовані в приміщенні. Температура експлуатації від 21 С до 25 С, відносна вологість 50-60%.

3.3 Вимоги до складу і параметрів технічних засобів

Програмний продукт розрахований для функціонування на ПК, що має такі характеристики:

— Маніпулюючий пристрій миша;

— Клавіатура;

— USB-роз’єм стандарту 3.0 і вище;

— Процесор від Core I3 та внутрішня пам’ять пам’ять від 8гб.

3.4 Вимоги до інформаційної і програмної сумісності

Користувачу для використання сервісу необхідний встановити відкритий фреймворк ML.Net та мати операційна систему Windows (починаючи с версій Windows 7)

3.5 Вимоги до маркування і упаковки

Передача офлайн версії продукту також потребує унікального маркування. Розроблений штамп представлено на рис. 1.

Упакування офлайн версії проходитиме після проходження контролю якості командою тестувальників, а також пробного запуску на змодельованому сервері.

Рис. 1. Штамп продукту

3.6 Вимоги до транспортування і зберігання

Продукт буде розгорнуто віддалено на сервері, тому транспортування проходитиме на рівні електронної мережі по захищеному з’єднанню.

Зберігання продукту здійснюватиметься на орендованих хостингах, крім цього локальну копію розробленого продукту буде передано на жорстокому USB-носії при передачі продукту за контрактом.

3.7 Вимоги до програмної документації

До складу програмної документації мають входити:

- специфікація;
- текст програми;
- опис програми.
- керівництво користувача.

Вся документація до програми повинна задовольняти вимогам державного стандарту до оформлення програмних документів.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. ML.Net // Документація ML.NET // https://learn.microsoft.com/ru-ru/dotnet/machine-learning/how-does-mldotnet-work (дата звернення: 01.12.2021).
2. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор
Українського державного
університету науки і технології
Анатолій РАДКЕВИЧ

«TOOLKIT ML.NET»

Робочий проект
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.0170437-01-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
18.12.22

Керівник розробки
_____ Олександр ПЕТРОВИЧ
18.12.22

Виконавець
_____ Андрій БУРЦЕВ
18.12.22

Нормоконтролер
_____ Світлана ВОЛКОВА
18.12.22

2022

ЗАТВЕРДЖЕНО
1116130.0170437-01-ЛЗ

«TOOLKIT ML.NET»

Специфікація

Листів 2

2022

Позначення	Найменування	Примітка
	Документація	
1116130.0170437–01–ЛЗ	Лист затвердження	
1116130.0170437–01 12 01-ЛЗ	Лист затвердження	
1116130.0170437–01 12 01	Текст програми	
1116130.0170437–01 13 01-ЛЗ	Лист затвердження	
1116130.0170437–01 13 01	Опис програми	
1116130.0170437–01 І3 01-ЛЗ	Лист затвердження	
1116130.0170437–01 І3 01	Керівництво користувача	

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор
Українського державного
університету науки і технології
Анатолій РАДКЕВИЧ

«TOOLKIT ML.NET»

Опис програми
1116130.0170437-01 13 01

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
18.12.22

Керівник розробки
_____ Олександр ПЕТРОВИЧ
18.12.22

Виконавець
_____ Андрій БУРЦЕВ
18.12.22

Нормоконтролер
_____ Світлана ВОЛКОВА
18.12.22

2022

ЗАТВЕРДЖЕНО
1116130.0170437-01 13 01

«TOOLKIT ML.NET»

Опис програми

Листів 14

2022

АНОТАЦІЯ

Документ 1116130.0170437-01 13 01 «TOOLKIT ML.NET». Опис програми» входить до складу програмної документації на програму для реалізації клієнтської частини та серверу.

У даному документі представлений опис програми клієнтської частини системи: функціональне призначення, опис логічної структури, використані технічні засоби, виклик і завантаження, вхідні і вихідні дані, порядок роботи з програмою. Програми написані мовою C#. Об'єм пам'яті, що займають програми комплексу, складає 147 Мб. Конфігурація комп'ютера стандартна.

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ

Програма представляє серверну частину електронної системи «TOOLKIT ML.NET» та призначена для реалізації маніпуляції з системою. За допомогою даної програми користувачі можуть зчитувати інформацію та працювати з нею через консольну частину програми.

Для функціонування даного програмного продукту необхідно, щоб на користувацькому комп‘ютері було встановлено наступне програмне забезпечення:

- ОС — Microsoft Windows 7 або пізніша;
- Відкритий фреймворк ML.NET.

Програмний додаток розробляється з використанням мови програмування C#.

2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Задачі, що розв‘язуються даним програмним забезпеченням, пов‘язані з отриманням точності моделі після її навчання. Інформація має бути подана структуровано та зрозуміло.

Функціональні обмеження накладаються тільки у тому разі, якщо користувач не маю опиту роботи з мовою програмування C#.

3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1. Алгоритм програми

На рис. 1 представлено алгоритм роботи програми.

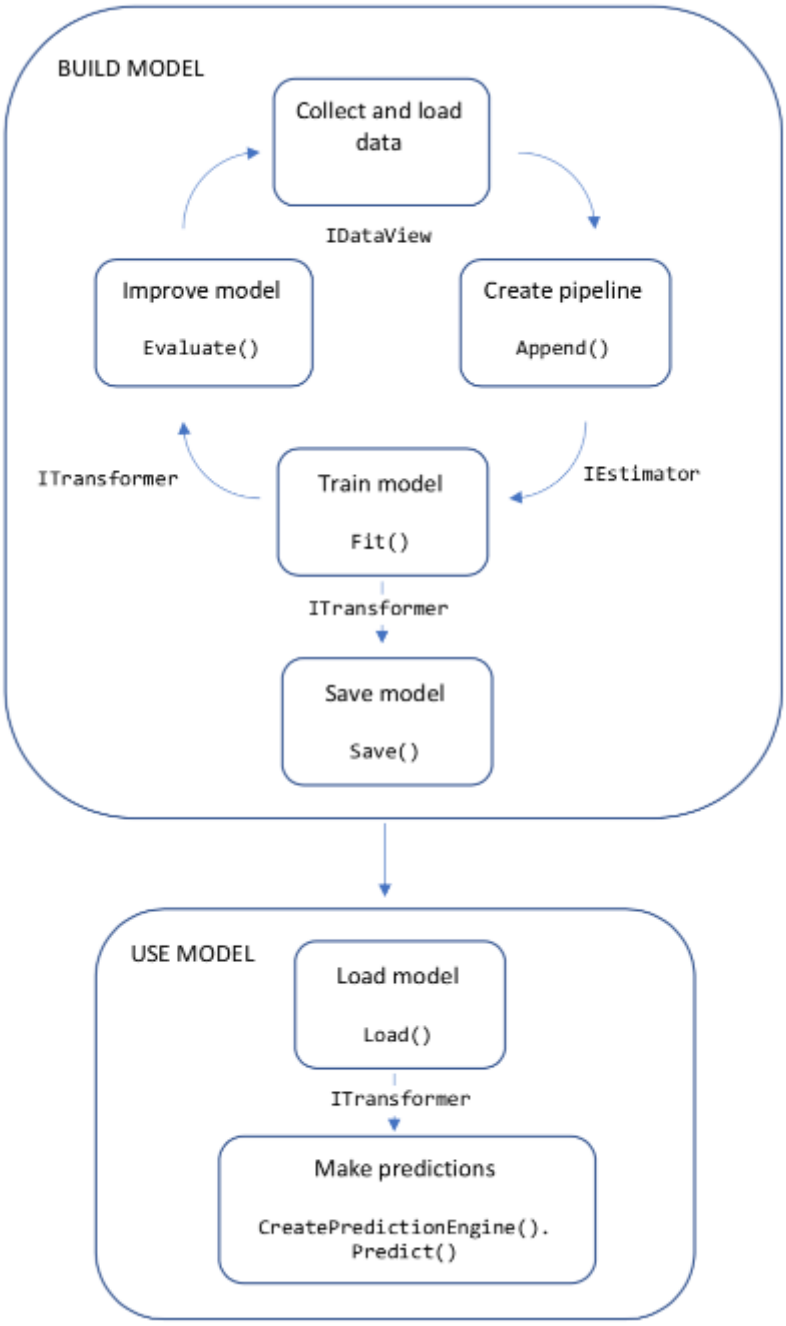

Рис. 1. Алгоритм роботи програми

3.2. Використані методи

На наведеній вище схемі показано структуру коду програми, а також ітеративний процес розробки моделі:

— Збір та завантаження навчальних даних в об'єкт IDataView

— Вказівка конвеєра операцій для отримання функцій та застосування алгоритму машинного навчання

— Навчання моделі шляхом виклику функції Fit() для конвеєра

— Оцінка моделі та ітерації для її покращення

— Збереження моделі у двійковому форматі для використання у додатку

— Завантаження моделі назад в об'єкт ITransformer

— Прогнозування за допомогою функції CreatePredictionEngine.Predict()

3.3. Структура програми

На рисунку 2 відображені основні програмні модулі та взаємозв‘язок між ними.

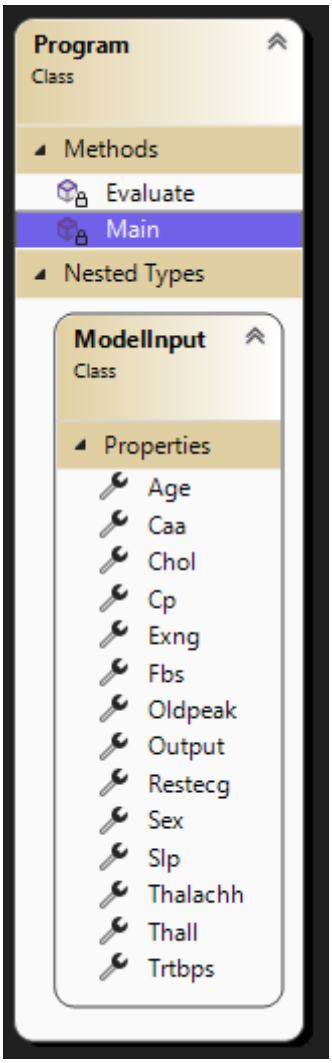

Рис. 2. Взаємозв‘язок програмних модулів

Для розробки даного програмного забезпечення в поєднанні з програмними модулями на мові програмування C# використовуються готову бібліотеку «ML.Net» версія 2.0.0.

4. ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для функціонування програмного забезпечення «Клієнт» достатньо мати робочий ПК, що має наступні характеристики:

- маніпулюючий пристрій миша;
- клавіатура;
- стійке інтернет-з’єднання ;
- USB-роз’єм стандарту 3.0 і вище.

5. ВИКЛИК ТА ЗАВАНТАЖЕННЯ

Перед початком роботи користувач повинен впевнитися у тому, що виконавчі файли даного програмного комплексу знаходяться на користувацькому комп‘ютері та їх можливо використовувати.

Для початку роботи з програмним комплексом користувач повинен запустити командну строку та ввести після крапки та скісної риски назву модулю для запуску, а також параметри та їх необхідні значення налаштування роботи модулю.

6. ВХІДНІ ТА ВИХІДНІ ДАНІ

У більшості випадків доступні дані не підходять для навчання моделі машинного навчання у їхньому вихідному вигляді. Необроблені дані необхідно підготувати (піддати попередньому обробленню), перш ніж їх можна буде використовувати для пошуку параметрів вашої моделі. Можливо, дані доведеться перетворити з рядкових значень на числові, звільнити від надмірної інформації, зменшити чи збільшити їх розмір, масштабувати чи нормалізувати.

Для завантаження даних із файлу використовуйте метод LoadFromTextFile з моделлю даних для даних, що завантажуються. Оскільки параметр separatorChar за замовчуванням розділяється знаками табуляції, змініть його, якщо потрібно для файлу даних. Якщо у файлі є заголовок, надайте параметру hasHeader значення true, щоб ігнорувати перший рядок у файлі та почати завантаження даних з другого рядка. Файл повинен мати формат розширення «.csv»

Вихідними даними даної системи будуть показники точності моделі. Користувачеві буде надано інформацію в текстовому виду у консолі.

7. ПОРЯДОК РОБОТИ З ПРОГРАМОЮ

Програма спроектована таким чином, щоб користувачі могли мати постійний доступ до інформації на сервері у будь-який час. Доступ до даної програми надається цілодобово.

Порядок роботи системного адміністратора, який є відповідальним за перевірку та схвалення користувацьких дій, а також за надання консультацій щодо користування програмою, визначається окремо відповідно до розпорядження установи, у якій планується використання даного програмного продукту. Приклад порядку роботи системного адміністратора:

- Технічний огляд серверу перед використанням — 8:00 - 8:15
- Отримання, перевірка та внесення інформації, що надходить від користувачів; обробка запитів користувачів щодо користування системою — 8:15 - 15:00
- Складання звітності за виконану роботу та плану роботи на наступний день — 15:00 - 17:00
- Передача звітності та узгодження плану роботи з керівництвом — 17:00

8. ПОВІДОМЛЕННЯ

В таблиці 1 приведені повідомлення користувачеві програми, які можуть виникати під час роботи з програмою.

Таблиця 1 — повідомлення користувачеві

Текст повідомлення	Опис ситуації	Рекомендовані дії
«System.FormatException: 'Parsing failed with an exception: Stream reading encountered exception'»	Користувач не надав файл для завантаження.	Додати потрібний файл з даними у програму.
«System.InvalidOperationException: 'No valid training instances found, all instances have missing features.'»	Користувач надав файл з некоректними даними.	Користувачу потрібно надати файл з перевірними даними.

БІБЛІОГРАФІЧНИЙ СПИСОК

3. ML.Net // Документація ML.NET // https://learn.microsoft.com/ru-ru/dotnet/machine-learning/how-does-mldotnet-work (дата звернення: 01.12.2021).
4. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
10.06.22

БОТ ДЛЯ РОБОТИ З РОЗВАЖАЛЬНИМ КОНТНЕТОМ «МЕМОВОТ»

Текст програми
ЛИСТ
ЗАТВЕРДЖЕННЯ
1116130.0170437-01 13 01-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
18.12.22

Керівник розробки
_____ Олександр ПЕТРОВИЧ
18.12.22

Виконавець
_____ Андрій БУРЦЕВ
18.12.22

Нормоконтролер
_____ Світлана ВОЛКОВА
18.12.22

2022

ЗАТВЕРДЖЕНО

1116130.0170437-01 13 01-ЛЗ

«TOOLKIT ML.NET»

Текст програми

1116130.0170437-01 13 01-ЛЗ

Листів 6

2022

АНОТАЦІЯ

Документ 44165850.95103–01–ЛЗ «TOOLKIT ML.NET». Текст програми» входить до складу програмної документації на програму для реалізації клієнтської частини та серверу.

У даному документі представлений текст програми клієнтської частини системи. Програми написані мовою C#. Об'єм пам'яті, що займають програми комплексу, складає 100 Мб. Конфігурація комп'ютера стандартна. Операційна система Windows 11.

ЗМІСТ

1. СХЕМА ВЗАЄМОДІЇ МОДУЛІВ

На рис. 1 представлено схему взаємодії модулів.

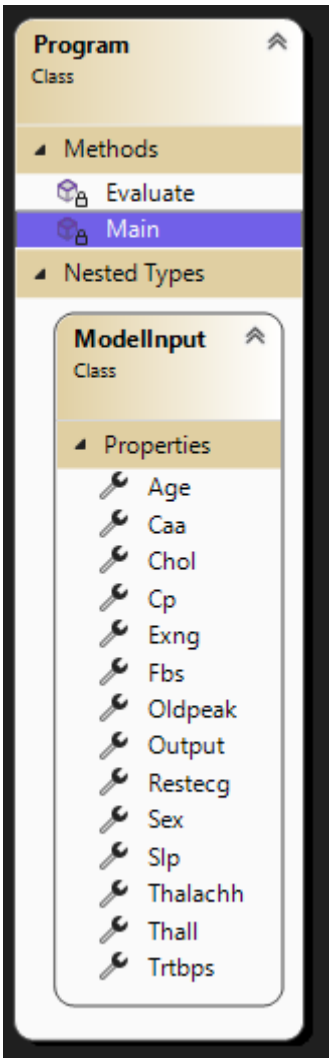

Рисунок 1.1 — Схема взаємодії модулів програмного комплексу

2. ТЕКСТ ПРОГРАМИ

2.1. Модуль "Main"

```
using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Trainers;

namespace myApp
{
   class Program
   {
      #region model input class
      public class ModelInput
      {
         [LoadColumn(0)]
         [ColumnName(@"age")]
         public float Age { get; set; }

         [LoadColumn(1)]
```

```
    [ColumnName(@"sex")]
    public float Sex { get; set; }

    [LoadColumn(2)]
    [ColumnName(@"cp")]
    public float Cp { get; set; }

    [LoadColumn(3)]
    [ColumnName(@"trtbps")]
    public float Trtbps { get; set; }

    [LoadColumn(4)]
    [ColumnName(@"chol")]
    public float Chol { get; set; }

    [LoadColumn(5)]
    [ColumnName(@"fbs")]
    public bool Fbs { get; set; }

    [LoadColumn(6)]
    [ColumnName(@"restecg")]
    public float Restecg { get; set; }

    [LoadColumn(7)]
    [ColumnName(@"thalachh")]
    public float Thalachh { get; set; }

    [LoadColumn(8)]
    [ColumnName(@"exng")]
    public bool Exng { get; set; }

    [LoadColumn(9)]
    [ColumnName(@"oldpeak")]
    public float Oldpeak { get; set; }

    [LoadColumn(10)]
    [ColumnName(@"slp")]
    public float Slp { get; set; }

    [LoadColumn(11)]
    [ColumnName(@"caa")]
    public float Caa { get; set; }

    [LoadColumn(12)]
    [ColumnName(@"thall")]
    public float Thall { get; set; }

    [LoadColumn(13)]
    [ColumnName(@"output")]
    public float Output { get; set; }

}

#endregion

static void Main(string[] args)
```

```
    {
        var mlContext = new MLContext();

        var reader = mlContext.Data.CreateTextLoader<ModelInput>(separatorChar: ',', hasHeader: true);
        IDataView trainingDataView = reader.Load("heart.csv");

        var pipeline = mlContext.Transforms.Categorical.OneHotEncoding(new[] { new InputOutputColumnPair(@"fbs",
@"fbs"), new InputOutputColumnPair(@"exng", @"exng") })
        .Append(mlContext.Transforms.ReplaceMissingValues(new[] { new InputOutputColumnPair(@"age", @"age"),
new InputOutputColumnPair(@"sex", @"sex"), new InputOutputColumnPair(@"cp", @"cp"), new
InputOutputColumnPair(@"trtbps", @"trtbps"), new InputOutputColumnPair(@"chol", @"chol"), new
InputOutputColumnPair(@"restecg", @"restecg"), new InputOutputColumnPair(@"thalachh", @"thalachh"), new
InputOutputColumnPair(@"oldpeak", @"oldpeak"), new InputOutputColumnPair(@"slp", @"slp"), new
InputOutputColumnPair(@"caa", @"caa"), new InputOutputColumnPair(@"thall", @"thall") }))
        .Append(mlContext.Transforms.Concatenate(@"Features", new[] { @"fbs", @"exng", @"age", @"sex", @"cp",
@"trtbps", @"chol", @"restecg", @"thalachh", @"oldpeak", @"slp", @"caa", @"thall" }))
        .Append(mlContext.Regression.Trainers.OnlineGradientDescent(
            new OnlineGradientDescentTrainer.Options()
            {
                NumberOfIterations = 7,
                LearningRate = 0.218590876185363F,
                LabelColumnName = @"output",
                FeatureColumnName = @"Features",
                // Change the loss function.
                LossFunction = new TweedieLoss(),
                // Give an extra gain to more recent updates.
                RecencyGain = 0.1f,
                // Turn off lazy updates.
                LazyUpdate = false,
                // Specify scale for initial weights.
                InitialWeightsDiameter = 0.2f
            }
            ));

        var model = pipeline.Fit(trainingDataView);

        Evaluate(mlContext, model);
    }

    private static void Evaluate(MLContext mlContext, ITransformer model)
    {
        IDataView dataView = mlContext.Data.LoadFromTextFile<ModelInput>("heart.csv", hasHeader: true,
separatorChar: ',');

        var predictions = model.Transform(dataView);

        var metrics = mlContext.Regression.Evaluate(predictions, "output", "Score");

        Console.WriteLine();
        Console.WriteLine($"*************************************************");
        Console.WriteLine($"*       Model quality metrics evaluation         ");
        Console.WriteLine($"*------------------------------------------------");
        Console.WriteLine($"*       RSquared Score:      {metrics.RSquared:0.##}");
        Console.WriteLine($"*       Root Mean Squared Error:      {metrics.RootMeanSquaredError:#.##}");
        Console.WriteLine($"*************************************************");
    }
```

```
    /*
     RSquared - Возвращает значение R-квадрата модели, которое также называется
коэффициентом определения. R-Squared ближе к 1 указывает на лучшую модель.
     */
  }
}
```