НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

«На правах рукопису»
УДК_____

«До захисту допущено»
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

«__»_____2023р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою
«Цифрові технології в енергетиці»
за спеціальністю 122 «Комп'ютерні науки»

на тему: Оптимізація алгоритмів машинного навчання для прискорення прогнозування на великих обсягах даних

Виконав: студент 2 курсу, групи ТР-21мп

Сергєєв Данило Вікторович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник _____доц., доц., к.т.н. Кублій Л. І._____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Студент _____
(підпис)

Київ – 2023

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ
Рівень вищої освіти другий (магістерський)
спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Цифрові технології в енергетиці»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
«___» _____ 2023р.

З А В Д А Н Н Я
на магістерську дисертацію студенту

Сергєєву Данилу Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Оптимізація алгоритмів машинного навчання для прискорення прогнозування на великих обсягах даних

науковий керівник дисертації Кублій Л.І. к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «06» листопада 2023 року № 5152-с

2. Термін подання студентом дисертації 18 грудня 2023року

3. Вихідні дані до роботи результатом роботи є програмний продукт, який автоматизує процеси оптимізації моделей та алгоритмів машинного навчання

4. Перелік питань, які потрібно розробити дослідити методи оптимізації у сфері машинного навчання; проаналізувати існуючі системи; створити власний програмний продукт; реалізувати практичні сценарії; випробувати створений програмний продукт

5. Орієнтований перелік ілюстративного матеріалу аналітичні та описові графіки для теоретичної частини дисертації, візуалізація алгоритмів, схеми архітектури застосунку, діаграма прецедентів можливостей користувачів, скриншоти інтерфейсу системи

6. Орієнтований перелік публікацій VII Міжнародна науково-практична конференція «Modern problems of science, education and society», 11-13.09.2023 Київ, Україна;
V Міжнародна науково-практична конференція «Global science: prospects and innovations», 28-30.12.2023 Ліверпуль, Великобританія

7. Дата видачі завдання «24» жовтня 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.08.2023	Виконано
2	Вивчення та аналіз задачі	01.08.2023–14.08.2023	Виконано
3	Збір інформації	15.08.2023–31.08.2023	Виконано
4	Аналіз вимог завдання, розробка методів і засобів розв'язання поставленої задачі	01.09.2023–30.09.2023	Виконано
5	Розробка та тестування програмного продукту	01.10.2023–14.10.2023	Виконано
6	Підготовка матеріалів магістерської роботи	15.10.2023–20.11.2023	Виконано
7	Захист програмного продукту	25.10.2023	Виконано
8	Передзахист	06.12.2023	Виконано
9	Захист		

Студент _____ Данило СЕРГЄЄВ
(підпис) (прізвище та ініціали)

Науковий керівник _____ Лариса КУБЛІЙ
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Актуальність теми. Швидкий розвиток сфери машинного навчання та поширення великих обсягів даних створюють вагомі виклики для ефективного прогнозування на великих обсягах інформації. Оптимізація алгоритмів машинного навчання стає невід'ємною частиною розв'язання цих завдань, особливо коли йдеться про прискорення процесу прогнозування.

Мета роботи. Основною метою даної роботи є розробка програмного продукту для оптимізації алгоритмів машинного навчання для швидкого та ефективного прогнозування на великих обсягах даних. Робота спрямована на покращення продуктивності і точності моделей машинного навчання при роботі з великими наборами інформації.

Завдання дослідження:

— опрацювати теоретичне підґрунтя алгоритмів машинного навчання;

— провести аналіз методів та підходів оптимізації моделей та алгоритмів машинного навчання;

— проаналізувати аналогічні програмні засоби;

— розробити програмний продукт під назвою ML Playground, спрямований на оптимізацію та спрощення взаємодії із машинним навчанням, а саме на пришвидшення прогнозування на великих обсягах даних;

— створити практичні сценарії використання розробленого програмного продукту;

— розробити стартап-проєкт до створеного програмного продукту.

Об'єкт дослідження. Методи оптимізації машинного навчання для великих обсягів даних.

Предмет дослідження. Прискорення прогнозування на великих обсягах даних із застосуванням оптимізації алгоритмів та моделей машинного навчання.

Практична цінність. Отримані результати дослідження можуть бути використані в різних сферах, таких як фінанси, медицина, інтернет-сервіси тощо, де швидкі та точні прогнози на великих обсягах даних є критично важливими. Оптимізовані алгоритми можуть сприяти покращенню якості прийняття рішень на

основі аналізу великих наборів інформації. Також вони можуть бути корисними для науковців і дослідників, які займаються вивченням великих даних і штучного інтелекту. Використання цих результатів може значно підвищити ефективність їхньої роботи і допомогти в розвитку нових технологій.

Апробація результатів дисертації. Основні положення даної роботи доповідалися та обговорювалися на:

— VII Міжнародній науково-практичній конференції «Modern problems of science, education and society», 11-13.09.2023, Київ, Україна;

— V Міжнародні науково-практичній конференції «Global science: prospects and innovations», 28-30.12.2023, Ліверпуль, Великобританія.

Дисертація складається із вступу, п’яти розділів, висновків і списку використаних джерел. Загальний обсяг дисертації становить 117 сторінок, у тому числі 97 сторінок основного тексту, 2 таблиці, 29 рисунків, 2 додатки, список використаних джерел містить 45 джерел.

Ключові слова: машинне навчання, алгоритми, оптимізація, великі обсяги даних, прогнозування.

ABSTRACT

Relevance of the topic. The rapid development of the field of machine learning and the proliferation of large volumes of data create significant challenges for effective forecasting on large amounts of information. Optimization of machine learning algorithms becomes an integral part of solving these tasks, especially when it comes to accelerating the forecasting process.

Objective of the study. The main objective of this work is to develop an application for optimizing machine learning algorithms for fast and effective forecasting on large volumes of data. The work is aimed at improving the productivity and accuracy of machine learning models when working with large sets of information.

Research tasks:

— to study the theoretical basis of machine learning algorithms;

— to analyze methods and approaches to optimizing models and algorithms of machine learning;

— to develop a software product called "ML Playground", aimed at optimizing and simplifying interaction with machine learning, namely accelerating forecasting on large volumes of data;

— to create practical scenarios for using the developed software product;

— to develop a startup project for the created software product.

Object of research. Methods of optimizing machine learning for large volumes of data.

Subject of research. Acceleration of forecasting on large volumes of data using optimization of algorithms and models of machine learning.

Practical value. The obtained research results can be applied in various fields such as finance, medicine, internet services, etc., where fast and accurate predictions on large datasets are critically important. Optimized algorithms can contribute to improving the quality of decision-making based on the analysis of large sets of information. They can also be beneficial for scientists and researchers involved in the

study of big data and artificial intelligence. The use of these results can significantly enhance the efficiency of their work and aid in the development of new technologies.

Approval of the dissertation results. The main provisions of this work were reported and discussed at:

— VII International Scientific and Practical Conference "Modern problems of science, education and society", 11-13.09.2023 Kyiv, Ukraine;

— V International Scientific and Practical Conference "Global science: prospects and innovations", 28-30.12.2023 Liverpool, United Kingdom.

The dissertation consists of an introduction, five sections, conclusions, and a list of used sources. The total volume of the dissertation is 117 pages, including 97 pages of the main text, 2 tables, 29 figures, 2 appendixes, the bibliography contains 45 sources.

Keywords: machine learning, algorithms, optimization, large volumes of data, forecasting.

ЗМІСТ

ВСТУП

У сучасному світі, де обсяги даних зростають експоненційно, аналіз та інтерпретація даних стають все більш актуальними. Одним з ключових інструментів для обробки та аналізу великих обсягів даних є машинне навчання.

Кілька десятиліть тому концепція комп'ютера як основного помічника людини була майже нереальною. З часом обсяг завдань, які може виконати цей електронний пристрій, і сфери його застосування значно розширилися. Наприклад, виникли такі нові його функції, як-от: автоматичне розпізнавання обличчя, забезпечення роботи апаратів самообслуговування в паркінгах тощо. Хоч можливо прогнозувати, що цей процес почне сповільнюватися але, навпаки, він «набирає обертів», що стимулює розробку більш досконалих технологій таких, як штучний інтелект.

Штучний інтелект включає ряд напрямків, одним з яких є машинне навчання. Машинне навчання дає можливість робити більш точні прогнози без необхідності додаткового програмування для досягнення цієї мети. Алгоритми машинного навчання використовують історичні дані як вхідні, щоб прогнозувати нові вихідні значення. Роль машинного навчання в сучасному світі не можна переоцінити, тому воно стало предметом даного наукового дослідження.

Актуальність машинного навчання обумовлена необхідністю прискорення процесу аналізу великих обсягів даних, що є особливо важливим у сферах, де час реакції реагування критичний, наприклад, у військовій й фінансовій сферах [1].

Класифікація, кластеризація, регресія є одними з найбільш поширених застосувань машинного навчання. Інші популярні застосування, як-от: виявлення шахрайства, фільтрацію спаму, виявлення загроз зловмисного програмного забезпечення, автоматизацію бізнес-процесів і прогнозування.

На сьогоднішній день, тема оптимізації алгоритмів машинного навчання є досить добре вивченою, проте, вона весь час розвивається у зв'язку з появою новітніх інформаційних технологій і методів обробки даних. У той самий час, існуючі розв'язки нерідко не є оптимальними і вичерпними для конкретних задач і специфічних обсягів даних.

Отже, необхідність проведення дослідження з даної теми обумовлена потребою в розробці нових і вдосконаленні існуючих засобів оптимізації алгоритмів машинного навчання для прискорення прогнозування за великими обсягами вхідних даних [1]. Результати дослідження дають можливість підвищити ефективність аналізу даних і зробити цей процес більш доступним і швидким.

Мета роботи — розробити програмний продукт ML Playground, який спростить процес розробки моделей машинного навчання, аналізу даних і розв'язання інших професійних задач, пов'язаних з машинним навчанням. Цей програмний продукт повинен мати інтуїтивно зрозумілий інтерфейс для створення, тренування і тестування моделей машинного навчання без необхідності написання складного коду. Це дасть можливість користувачам зосередитися на таких аспектах машинного навчання, як-от: вибір певних алгоритмів, налаштування параметрів моделі та інтерпретація результатів.

Платформа демонструє свою корисність в ряді сфер:

— спрощення розробки моделей машинного навчання;
— підтримка наукових досліджень в галузі машинного навчання;
— аналіз даних без програмування;
— оптимізація вибору алгоритмів і параметрів моделей;
— підвищення ефективності тренування і тестування;
— спрощення інтерпретації результатів;
— навчання та освіта в галузі машинного навчання;
— доступність для широкого кола користувачів;
— підтримка спільноти розробників;
— інтеграція з іншими інструментами.

Розроблений програмний продукт ML Playground є інноваційним кроком у галузі машинного навчання та надає зручний і доступний інструмент для розробників, дослідників і фахівців у галузі аналізу даних. Його функціонал і наявність зрозумілого користувацького інтерфейсу спрощує процес створення та оптимізації моделей.

1. ЗАДАЧА ОПТИМІЗАЦІЇ В МАШИННОМУ НАВЧАННІ

Прогрес у галузі штучного інтелекту відбувається з великою швидкістю, так само, як і розвиток глобальної мережі Інтернет. Зростання її інформаційних і комунікаційних потенціалів відбувається кожного дня. Людина щоденно взаємодіє з штучним інтелектом через електронні пристрої, наприклад, розблоковуючи свій мобільний телефон за допомогою розпізнавання обличчя або визначаючи оптимальний маршрут за допомогою навігатора.

Метою даної роботи є створення платформи ML Playground, яка повинна спростити процес оптимізації використання машинного навчання й поліпшити взаємодію із штучним інтелектом.

Для вдалого створення цієї платформи були визначені наступні завдання:

- проведення аналізу існуючих аналогічних систем;
- проведення аналізу алгоритмів машинного навчання;
- проєктування та розробка власного програмного продукту:
  - аналіз засобів розробки;
  - обрання засобів розробки;
  - проєктування бази даних;
  - розробка веб-застостунку;
  - використання машинного навчання;
  - створення візуалізацій та функцій експорту або імпорту;
- тестування та вдосконалення розробленої системи;
- випробування створеної системи;
- демонстрація роботи системи із реальними наборами даних.

1.1 Аналіз існуючих платформ

Під час аналізу існуючих платформ і подібних по функціоналу розглянуто наступні платформи, які надають можливість працювати з машинним навчанням: Google Colab, IBM Watson Studio, Microsoft Azure Machine Learning Studio, DataRobot, TensorFlow.

Усі зазначені вище платформи надають можливість взаємодіяти з машинним навчанням, змінюючи моделі та параметри. Також платформи мають візуалізацію.

Але жодна з проаналізованих платформ на момент написання дисертації не мала можливості взаємодіяти з наборами даних користувача. Варто зауважити, що наведені платформи лише наочно демонструють роботу машинного навчання, а не пропонують прямого їхнього використання.

За результатами аналізу можна визначити основні критерії платформи, яка повинна мати переваги відносно існуючих:

— можливість використання власних наборів даних;

— практичне застосування машинного навчання, наприклад, прогнозування;

— візуалізація результатів і метрик;

— можливість порівняння моделей.

Необхідно зазначити, що важлива підтримка різноманітних алгоритмів машинного навчання у створюваній платформі та їхня ефективність у різних сценаріях застосування. Платформа, яка пропонує широкий спектр алгоритмів та гарантує їхню високу ефективність у різних завданнях, може виявитися більш універсальною і відповідати потребам у різноманітних дослідженнях і застосуваннях машинного навчання. Такий аспект дає можливість користувачеві ефективно використовувати платформу для різноманітних викликів, що може бути важливим для комплексних досліджень і практичних застосувань.

Враховуючи зазначені критерії і вимоги до конкурентоспроможної платформи для оптимізації машинного навчання, потрібно зазначити необхідність забезпечення високого рівня гнучкості та легкості в інтеграції з іншими інструментами аналізу даних і роботи з великими обсягами даних. Такий функціонал може значно полегшити процеси обробки даних і розробки моделей, сприяючи взаємодії з іншими компонентами і продуктами. Таким чином, створювана платформа повинна бути не лише потужним інструментом для машинного навчання, але й гнучкою системою для роботи з великими обсягами даних.

1.2 Особливості застосування методів машинного навчання

Крім зазначених у попередньому пункті критеріїв, важливим аспектом предметної сфери є підтримка різноманітних алгоритмів машинного навчання. Також платформа повинна забезпечувати можливість використання передових моделей машинного навчання, щоб користувачі мали доступ до найновіших технологій у цій сфері.

Важливою характеристикою платформи має бути підтримка оцінювання роботи машинного навчання. Можливість використання реальних даних, порівняння моделей та результатів сприятиме швидкому розвитку, використанню та вдосконаленню методів навчання.

Функціонал експорту та імпорту даних повинен забезпечити швидку та зручну взаємодію із користувацькими наборами даних.

Також система повинна бути адаптована до використання в освітніх цілях. Забезпечення зручного інтерфейсу для студентів і викладачів, а також можливість проведення навчальних експериментів сприятиме засвоєнню підвалин машинного навчання та його практичному застосуванню.

Платформа повинна мати швидке та легке розгортання та масштабування, щоб забезпечити зручність користування як початківцям, так і досвідченим фахівцям у галузі машинного навчання.

Технологічні рішення, використані в програмному продукті, мають відповідати високим стандартам надійності, швидкості та сучасності. Вони повинні бути здатні аналізувати великі обсяги даних, гарантувати стабільність роботи та ефективність виконання завдань. Важливим є застосування сучасних інноваційних технологій, які дозволяють оптимізувати процеси обробки даних, забезпечуючи достатню швидкість й точність обчислень.

Використання сучасних технологій відіграє ключову роль у забезпеченні гнучкості та масштабованості системи. Це означає, що програмний продукт повинен бути здатний адаптуватися до змінних обставин, розширюватися та модернізуватися відповідно до потреб користувачів і стану розвитку технологій. Сучасні технології повинні гарантувати безперебійну роботу програмного

продукту, захист даних користувачів і стабільність роботи незалежно від зовнішніх факторів. Це досягається використанням надійних серверів, захищених протоколів зв’язку та ефективних механізмів резервного копіювання.

Висновки до розділу 1

Актуальність роботи насамперед визначається стрімким розвитком штучного інтелекту та його впливом на щоденне життя людини. Створення платформи ML Playground зумовлене необхідністю спрощення взаємодії користувачів із системами машинного навчання, що є важливим етапом у популяризації та розширенні сфер використання машинбного навчання.

Аналіз існуючих платформ акцентував важливість підтримки користувацьких наборів даних і практичного застосування машинного навчання. Відсутність можливості взаємодії із наборами даних у більшості аналогічних платформ свідчить про унікальність розробленої платформи.

Виняткова увага до технічних аспектів розробки, таких як швидкість й масштабованість, вказує на прагнення до створення ефективної та сучасної платформи. Використання передових технологій та їхня адаптація до змінних умов дають можливість забезпечити стабільну та безперебійну роботу платформи ML Playground в умовах стрімкого розвитку галузі штучного інтелекту.

Враховуючи ці факти, можна стверджувати, що робота має великий потенціал у напрямі популяризації застосування алгоритмів машинного навчання в різних сферах діяльності людини. Це сприяє впровадженню інновацій та допомагає розв’язувати складні завдання, які раніше були недосяжними для традиційних методів.

2. ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ

З метою створення програмної реалізації платформи ML Playground були проведені дослідження та аналіз можливих засобів розробки. Також виконане ретельне ознайомлення з різними моделями та методами використання машинного навчання. За результатами цієї роботи сформовано теоретичне підґрунтя розробки.

2.1 Теоретичні основи алгоритмів машинного навчання

Машинне навчання — це галузь штучного інтелекту, яка досліджує способи створення моделей, здатних навчатися, і алгоритми для їхнього створення та навчання. В якості навчання машина обробляє величезні масиви вхідних даних і знаходить у них закономірності. Тобто машина розв’язує складну багатопараметричну задачу не прямим способом, а може знайти закономірність у даних після навчання алгоритму на безлічі прикладів, таким чином знаходячи більш точні розв’язки задачі. Як результат – правильне прогнозування. Головною метою машинного навчання є автоматизація та оптимізація розв’язання складних професійних завдань в різних галузях людської діяльності частково чи повністю.

На сьогоднішній день сфера застосування машинного навчання постійно розширюється. За рахунок загальної інформатизації накопичується велика кількість даних у науці, виробництві, бізнесі, транспорті, медицині. Багато задач, пов’язаних з прогнозуванням, керуванням і прийняттям рішень, часто базуються на методах навчання з використанням прикладів. Раніше, коли було менше даних, такі задачі розв’язувалися іншими методами або залишалися нерозв’язаними.

2.1.1 Основні поняття і принципи машинного навчання

Машинне навчання ґрунтується на ряді базових понять і принципів, які відіграють визначальну роль у процесі розробки й застосування алгоритмів і моделей.

Одним з таких принципів є навчання на основі даних [3]. Його сутність полягає в тому, що алгоритми машинного навчання використовують саме

історичні дані для побудови моделей і прийняття рішень. Дані містять велику кількість прикладів, які дозволяють алгоритмам «навчитися» наявних закономірностей і надати максимально точні прогнози на нових даних.

Винятково важливим принципом є розділення на навчання й тестування. З цим принципом дані поділяються на два набори — один використовується для навчання моделі, а інший для перевірки її точності. Це допомагає уникнути зайвого перенавчання у випадку, коли модель добре працює на навчальних даних, але погано справляється на нових даних. Розділення на навчання і тестування є істотним кроком, оскільки воно дає можливість об’єктивно оцінити, наскільки добре модель здатна генералізувати на нові дані, що є важливим аспектом в оцінці її продуктивності й ефективності.

Крім того, важливо враховувати принципи вибору та інженерії ознак. Вибір правильних ознак або характеристик для навчання моделі суттєво впливає на результат. Інженерія ознак дає можливість створення нових ознак на основі наявних даних, що може покращити якість моделі.

Таким чином, основні принципи машинного навчання передбачають використання даних для навчання, розділення даних на навчальні й тестові набори, а також правильний вибір та інженерію ознак для покращення точності моделей.

Одним із важливих аспектів машинного навчання є регуляризація, призначена для керування складністю моделей і запобігання перенавчанню. Регуляризація містить додавання певних обмежень або штрафів до функції витрат моделі під час навчання. Це допомагає знизити ваги незначущих ознак і підтримувати більш загальні робочі моделі. Такий підхід робить моделі універсальнішими, спроможними генералізувати на нові дані та враховувати компроміс між точністю на навчальних даних і загальною придатністю для застосувань на практиці. Регуляризація є ключовим елементом успішного машинного навчання, оскільки вона допомагає досягти більш надійних і стійких результатів в різних задачах.

Дані, ознаки та алгоритми є базовими поняттями машинного навчання.

Дані — основна інформація, до якої входять певні вибірки даних, роботі з якими треба навчити систему. Самі дані залежать від задачі, яку треба розв’язати із застосуванням машинного навчання. Існують два методи збору даних: автоматичний та ручний. Автоматичний набагато швидший за ручний, але при цьому робить більшу кількість помилок. Саме якість вибірки визначає точність прогнозування.

Ознаки — характеристики та властивості, які повинна відстежувати система внаслідок навчання і від яких безпосередньо залежить вихідний результат.

Алгоритм — система послідовних операцій для вирішення певної задачі або метод вирішення. Під кожну конкретну задачу можна підібрати окремий витончений алгоритм.

2.1.2 Поняття алгоритмів машинного навчання

Алгоритми машинного навчання — це основа будь-якої системи машинного навчання. Вони використовуються для вивчення шаблонів у даних і створення моделей, які можуть робити прогнози або приймати рішення на основі цих шаблонів. Існує багато різних типів алгоритмів машинного навчання, кожен з яких має свої особливості та сфери застосування [4].

Серед найпоширеніших алгоритмів машинного навчання можна виділити лінійну регресію, дерева рішень, k-найближчих сусідів, наївний Баєсовий класифікатор, опорні вектори і нейронні мережі. Кожен з цих алгоритмів має свої переваги й недоліки, а також вимоги до даних. Наприклад, деякі алгоритми вимагають, щоб дані були нормалізовані або стандартизовані перед навчанням, тоді як інші можуть працювати з сировинними даними.

Вибір правильного алгоритму машинного навчання вимагає розуміння проблеми, яку дослідник намагаєтесь розв’язати, а також особливостей даних. Наприклад, якщо працювати з неперервними даними й мати метою прогнозування значення на основі інших ознак, то може знадобитися лінійна регресія. З іншого боку, якщо працювати з категоріальними даними і потрібно класифікувати приклади на основі їхніх ознак, то може знадобитися дерево рішень або наївний Баєсовий класифікатор.

Всі ці аспекти — вибір алгоритму, налаштування параметрів, обробка даних є важливими етапами в процесі машинного навчання. Вони вимагають глибокого розуміння принципів машинного навчання, а також досвіду і практичних навичок.

2.1.3 Завдання та алгоритми машинного навчання

Завдання машинного навчання — це процес, в рамках якого комп'ютерні системи навчаються вирішувати конкретні завдання без явного програмування. У машинному навчанні алгоритми аналізують великі обсяги даних, вивчають закономірності та роблять прогнози або приймають рішення на основі цих даних.

Цей підхід дає можливість системам самостійно адаптуватися до нових вхідних даних і покращувати свою продуктивність з часом. Завдання машинного навчання можуть включати в себе класифікацію об'єктів, прогнозування тенденцій, розпізнавання образів та багато інших завдань, які сприяють автоматизації прийняття рішень на основі даних.

Бінарна класифікація. Це завдання машинного навчання з наглядом, яке використовується для прогнозування, до яких з двох класів або категорій належить набір даних. Вхід для алгоритму класифікації — це набір даних, де кожна позначка є цілим числом 0 або 1.

Вихід для алгоритму бінарної класифікації — це класифікатор, який можна використовувати для прогнозування класу нових непозначених екземплярів. Приклади сценаріїв бінарної класифікації включають:

— визначення типу коментарів у соціальній мережі як «позитивний» чи «негативний»;

— визначення чи має авто певну несправність чи ні;

— прийняття рішення зазначити електронний лист як «спам» чи ні;

— визначення чи містить фотографія певний предмет, такий як дерево чи будівля.

Для досягнення найкращих результатів у бінарній класифікації тренувальні дані повинні бути збалансованими, тобто однакова кількість позитивних та негативних тренувальних даних. Відсутні значення слід обробляти перед тренуванням.

Відомі алгоритми, які використовують для бінарної класифікації:

— логістична регресія — використовує лінійну комбінацію змінних класифікаторів для оцінки ймовірності того, що результат буде 0 або 1;

— наївний класифікатор Баєса — це класифікатор з високим зміщенням та низькою дисперсією, що має переваги перед логістичною регресією та алгоритмами найближчих сусідів, коли маєте обмежену кількість даних для навчання моделі;

— метод опорних векторів — базується на статистичних підходах. Модель шукає гіперплощину, яка найліпше розділяється на два класи.

Мультикласова класифікація. Це завдання машинного навчання, призначене для прогнозування класу (категорії) набору даних. Вхід для алгоритму класифікації — це набір позначених прикладів. Кожна позначка зазвичай починається як текст. Потім її перетворюють у ключовий (числовий) тип.

Вихід для алгоритму класифікації — це класифікатор, який можна використовувати для прогнозування класу нових непозначених наборів даних.

Приклади сценаріїв мультикласової класифікації включають:

— класифікацію залізничних рейсів як «зарано», «вчасно» чи «запізно»;

— класифікацію відгуків на фільми за багатьма категоріями: «позитивний», «нейтральний», «негативний» тощо;

— класифікацію нерухомості за ціновими сегментами, як-от: «елітний», «середній», «економ-клас» тощо.

Відомі алгоритмі, які використовують для мультикласової класифікації:

— логістична регресія — розширення на більше ніж два класи;

— наївний класифікатор — адаптація наївних багатокласових завдань;

— метод опорних векторів — розширення на багатокласові сценарії;

— дерево рішень — використовується для класифікації на основі того, які були відповіді на попередні питання. Модель є формою навчання з учителем, що означає, що вона навчається і тестується на наборі даних, який містить бажану категоризацію;

— випадковий ліс — поєднує ліс дерев рішень, що згенерований на випадкових векторах входу та розділення вузлів на випадковий набір ознак;

— градієнтний бустінг — основна ідея полягає в тому, щоб послідовно будувати моделі, причому кожна наступна модель намагається зменшити помилки попередньої моделі.

Регресія. Це завдання машинного навчання з учителем, яке використовується для прогнозування значення мітки на основі набору пов’язаних ознак. Мітка може бути будь-яким дійсним значенням і не належить до скінченного набору значень, як у завданнях класифікації. Алгоритми регресії моделюють залежність мітки від пов’язаних ознак, щоб визначити, як зміниться мітка при зміні значень ознак.

Вхідними даними для алгоритму регресії є набір прикладів з мітками відомих значень.

Вихідним значенням алгоритму регресії є функція, яку можна використовувати для прогнозування значення мітки для будь-якого нового набору вхідних ознак.

Приклади сценаріїв регресії включають:

— прогнозування цін на будинки на основі атрибутів, таких як кількість спалень, розташування чи розмір;

— прогнозування майбутніх цін на акції на основі історичних даних та поточних тенденцій на ринку;

— прогнозування продажів продукту на основі бюджетів реклами.

Для розв’язання завдання регресії використовують більшість раніше перерахованих алгоритмів машинного навчання з певними адаптаціями.

Кластеризація. Це завдання машинного навчання без учителя, яке використовується для групування наборів даних у кластери, які мають схожі характеристики. Кластеризація може використовуватися для виявлення взаємозв’язків у наборі даних, які спочатку можливо не мають жодних представлених зав’язків.

Входи та виходи алгоритму кластеризації залежать від обраної методології. Це надає можливість використовувати підхід на основі розподілу, центроїда, зв’язку чи щільності.

Приклади сценаріїв кластеризації включають:

— розподіл за сегментами для гостей готелю на основі звичок і характеристик вибору готелів;

— розподіл за сегментами демографічних характеристик клієнтів для побудови рекламних кампаній;

— категоризація запасів на основі виробничих показників.

Відомі моделі, які використовують для мультикласової класифікації:

— ієрархічна кластеризація — створює дерево кластерів. Добре підходить для ієрархічних даних, таких як таксономії. Крім того, є перевага у тому, що кількість кластерів може бути вибрана шляхом відкидання вузлів дерева на відповідному рівні;

— кластеризація на основі центроїдів — виділяє дані в непідсистемні кластери, на відміну від ієрархічної кластеризації. Метод k-сусідів є найбільш поширеним. Такі моделі ефективні, але чутливі до початкових умов та викидів;

— кластеризація на основі щільності — з’єднує області високої щільності екземплярів в кластери. Має труднощі з даними різної щільності та високою розмірністю;

— кластеризація на основі розподілу — передбачає, що дані складаються з розподілів, таких як гаусові розподіли. Зі збільшенням відстані від центру розподілу ймовірність того, що точка належить до розподілу, зменшується. Модель ефективна за умови наявності типу розподілу даних.

Підсумок. Машинне навчання та його алгоритми роблять значний внесок у сучасне технологічне середовище. Розглядаючи різноманітні варіанти завдань у машинному навчанні, стає очевидною важливість правильного вибору кращого алгоритму, оскільки це визначально впливає на якість результатів та ефективність моделі [4]. Визначення коректного підходу до вибору алгоритму для конкретного завдання стає одним із ключових етапів у здобутку успішних рішень у машинному навчанні.

Це дає можливість не лише оптимізувати процес роботи алгоритмів, але і забезпечує більш глибоке розуміння впливу вибору алгоритмів на різні аспекти завдань машинного навчання. Розкриття потенціалу кожного алгоритму в

контексті конкретної задачі допомагає досягти оптимальних результатів і визначає успішність використання машинного навчання в різних областях та сценаріях.

2.1.4 Використання компактних моделей

Одним з ключових аспектів вдосконалення алгоритмів машинного навчання є застосування компактних моделей. Компактні моделі є результатом інтенсивних досліджень в галузі оптимізації алгоритмів, спрямованих на покращення швидкодії та ефективності прогнозування на великих обсягах даних [5].

Перевагою використання компактних моделей є їхня здатність забезпечувати значний рівень точності при відносно невеликому обсязі ресурсів, необхідних для їхньої роботи. Це особливо актуально в умовах великого обсягу даних, де швидкодія та ефективність моделі можуть бути вирішальними факторами.

Застосування компактних моделей визначається як стратегічно важливий елемент оптимізації алгоритмів машинного навчання для досягнення швидкісних переваг при прогнозуванні на великих обсягах даних.

2.1.5 Вилучення несуттєвих параметрів моделі

Вилучення несуттєвих параметрів моделі у машинному навчанні — це техніка оптимізації моделей, яка полягає в усуненні зайвих параметрів чи ваг моделі з метою поліпшення її продуктивності та ефективності. Цей процес може відбуватися на етапі тренування або після завершення тренування.

Ідея вилучення у нейронних мережах, а саме у концепції видалення параметрів, які вважаються неважливими, існує вже досить давно. Перша пропозиція цієї ідеї була зроблена професором Яном Ле Куном у 1989 році.

Простий спосіб представити вилучення несуттєвих параметрів моделі — це усунення зв’язків або критеріїв ваги, які мають незначний вплив на вихідні дані моделі. Це може бути досягнуто різними методами, такими як встановлення нульових значень для критеріїв ваги чи видалення зв’язків, які мають малий ваговий коефіцієнт.

Виділяють дві основні категорії методу (рисунок 2.1):

— неструктуроване вилучення — проводиться на рівні окремих ваг. Отриманий шаблон активних ваг, загалом кажучи, має хаотичну структуру;

— структуроване вилучення — виконується на рівні цілих фільтрів або навіть шарів у нейронній мережі. Крім того, можна виконувати вилучення не окремих ваг критеріїв, а блоками, організованими за сусідством.

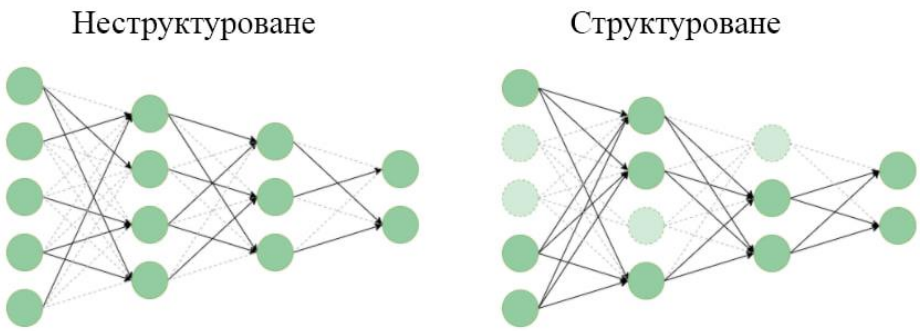


Рисунок 2.1 — Види методів вилучення параметрів моделі

Однією з переваг вилучення параметрів із моделі є зменшення обсягу моделі, що веде до меншого обсягу пам’яті, необхідного для її зберігання та ресурсів, витрачених на її реалізацію. Це зокрема корисно для мобільних і вбудованих систем з обмеженими ресурсами.

Порівняно з квантуванням, метод відсікання у моделі не тільки зменшує кількість бітів, використаних для представлення ваг, але й фактично видаляє зайві параметри. Він може бути особливо ефективним в умовах, коли точність моделі після відсікання може залишатися прийнятною для визначеної задачі.

Недоліками відсікання є втрата інтерпретованості моделі, оскільки важко розуміти, які саме функції або зв’язки видаляються. Також не завжди можна передбачити, як це вплине на взаємодію параметрів, і може виникнути ризик втрати точності.

У загальному метод відсікання виявляється корисним інструментом для оптимізації ресурсів і покращення продуктивності моделей в умовах обмежених обчислювальних ресурсів.

Метод Magnitude-based pruning. Метод відсікання на основі порівняння значень ваг вважається одним з простих та водночас ефективних підходів у оптимізації нейронних мереж. Метод вважається одним із найперших методів відсікання в нейронних мережах, запропонованим ще в кінці 80-х років. В основі цього методу лежить відкидання ваг з найменшим абсолютним значенням $|W_i|$. У матричних множеннях та операціях згортки неодмінно містяться операції подібного виду (сумування вважається за повторюваними індексами):

$$y_i = W_{ij} x_j$$

Вважаючи, що значення x_j в середньому мають величину першого порядку, можна відзначити, що ваги з найменшим абсолютним значенням роблять найменший внесок у суму.

При використанні методу відсікання на основі порівняння значень ваг на моделі ResNet50 та наборі даних ImageNet результати дослідження показують, що цей метод дає можливість ефективно відсікати ваги із значеннями, які роблять менший внесок, без втрати якості передбачень (рисунок 2.2) [6].

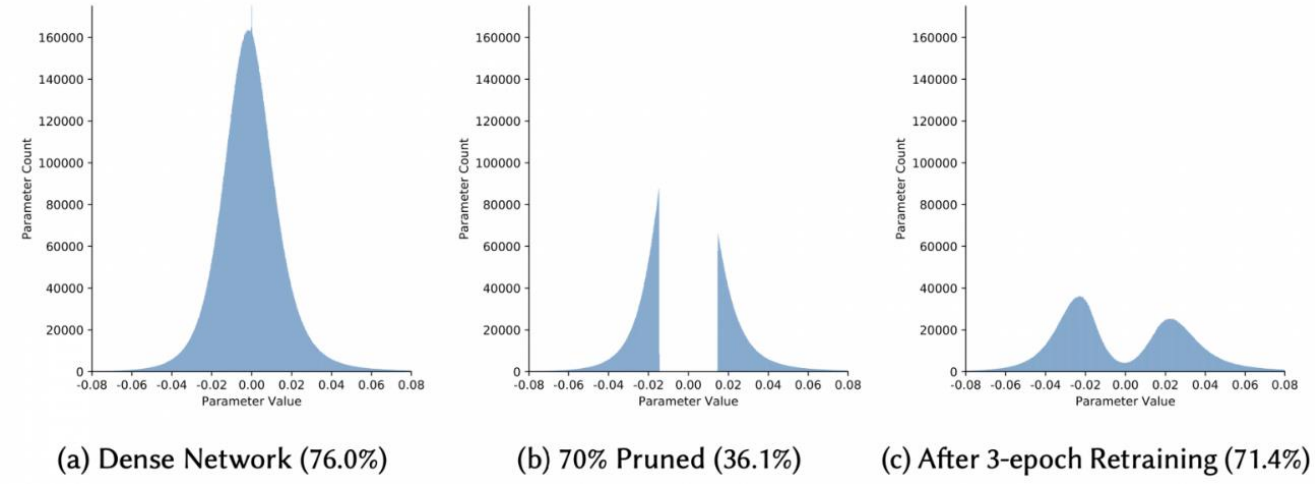


Рисунок 2.2 — Ефекти використання відсікання на основі порівняння значень ваг на моделі ResNet50 та наборі даних ImageNet [6]

Методи Optimal Brain Damage та Optimal Brain Surgeon. Ці методи оптимізації стосуються так званих алгоритмів оптимізації другого порядку, які ґрунтуються на використанні не лише градієнта, а й гессіана [7, 8].

Якщо розглядати функцію витрат $L(w)$ в будь який момент часу в процесі навчання, то можна апроксимувати її приріст (збурення) в околі точки мінімума за формулою Тейлора для функції багатьох змінних, обмежуючись членами до другого порядку:

$$\Delta L = L(w) - L(w_0) \approx \left(\frac{\partial L}{\partial w}\right)^T \Delta w + \frac{1}{2}\Delta w^T H \Delta w,$$

де символ T означає транспонування матриці;

$\frac{\partial L}{\partial w}$ — матриця Якобі, елементами якої є $\frac{\partial L}{w_i}$;

$\Delta w = w - w_0$ — вектор-матриця приросту;

H — матриця Гессе, елементами якої є похідні $\frac{\partial L}{\partial w_i \partial w_j}$.

Як правило, припускають, що вплив членів вищих порядків є незначним і тому їх ігнорують.

У методі Optimal Brain Damage (OBD) мета полягає в тому, щоб встановити один з параметрів (вагу), позначений w_i (скаляр), щоб мінімізувати приріст ΔL на кожній ітерації обрізання. Оскільки припускається, що у точці екстремума перший доданок (якобіан) дорівнює нулю, а матриця Гессе додатньо визначена, отримана в результаті оптимізаційна задача записується так:

$$\min_i \frac{1}{2}\Delta w^T H \Delta w,$$

за умови:

$$e_i^T \Delta w + w_i = 0,$$

де e_i — одиничний вектор, у якому на i-ий елемент дорівнює одиниці, а на інших місцях нулі.

Ця задача мінімізації традиційно може бути розв’язана методом множників Лагранжа знаходження умовного екстремуму функції багатьох змінних, щоб отримати «міру значущості» для приросту функції витрат, яка асоціюється з кожною вагою w_i:

$$\Delta L(w) = \frac{1}{2} \cdot \frac{w^2}{[H^{-1}]},$$

де через $\left[H^{-1}\right]ij$ позначено i-ий діагональний елемент оберненної матриці Гессе для функції витрат $L(w)$.

Щоб вибрати, які ваги відсікати, можна відсортувати ваги в порядку зменшення цієї «міри значущості» для функції витрат, причому вага з найменшим значенням є найкращим кандидатом для відсікання.

Цікаво, що ця процедура передбачає, що значення решти ваг також мають змінитися, і забезпечує відповідний оптимальний приріст Δw:

$$\Delta w = -\frac{w_i}{\left[H^{-1}\right]ii} \cdot H^{-1} \cdot e_i$$

У методі Optimal Brain Damage робиться припущення, що матриця Гессе — діагональна, тобто:

$$H = h_i \Delta_{ij}$$

Тоді критерій відбору зводиться до знаходження ваг з найменшим значенням $h_i^{\frac{1}{2}} w_i$, при цьому інші ваги не змінюються. Крім абсолютної величини ваг, враховується кривизна поверхні функції витрат (рисунок 2.3).

Зазначимо, що Optimal Brain Damage зводиться до методу відсікання на основі величини (magnitude-based pruning), коли всі h_i рівні [9].

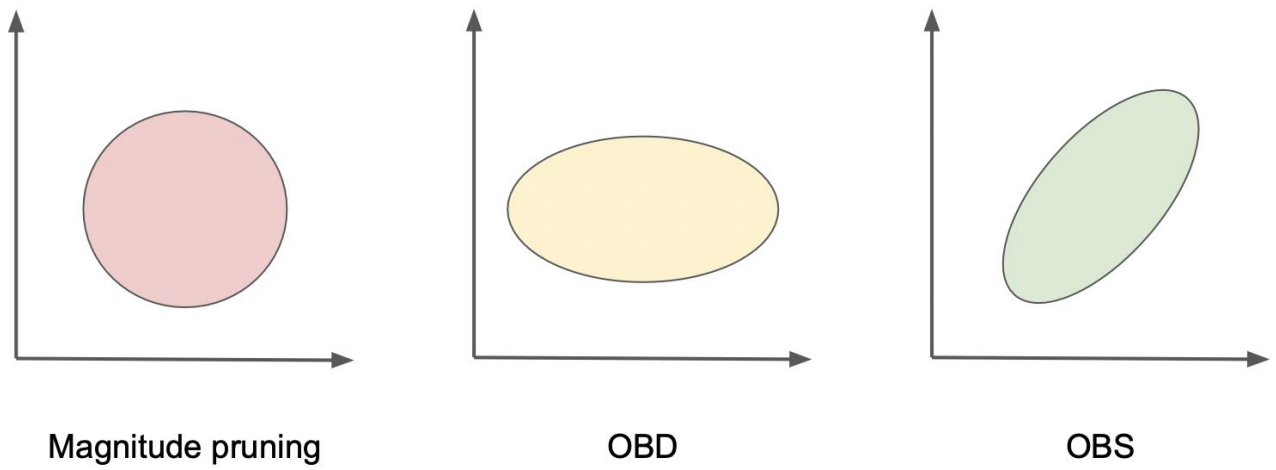


Рисунок 2.3 — Порівняння ліній рівня функції втрат для методів Magnitude-based pruning, Optimal Brain Damage та Optimal Brain Suregon

Приклад роботи різних методів, заснованих на розкладанні функції витрат за формулою Тейлора. На рисунку 2.4 подано графік функції:

$$L(x_1; x_2) = 2x_1^2 + 0{,}5x_2^2$$

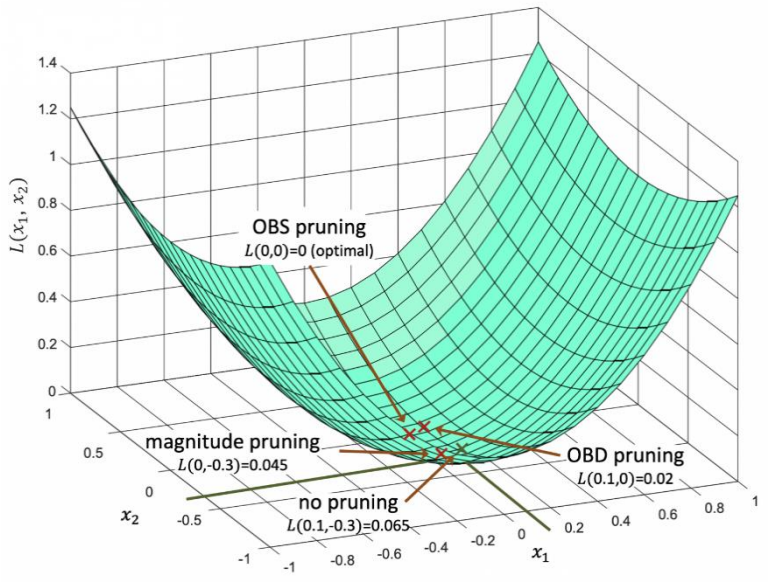


Рисунок 2.4 — Порівняння різних принципів відсікання для фіксованої квадратичної функції [9]

Припустимо, що за методом стохастичного градієнтного спуску знайшли апроксимацію мінімуму в точці $\left(x_1^*; x_2^*\right) = (0{,}1; -0{,}3)$.

Очевидно, що мінімум досягається в точці $(0; 0)$, а точка $(0{,}1; -0{,}3)$ береться для ілюстрації. Перший доданок в формулі Тейлора в цій точці дорівнює 0,1, але в алгоритмах оптимізації другого порядку прийнято припускати, що він є нульовим. Значення функції в цій точці $L(0{,}1; -0{,}3) = 0{,}065$. Відсікання за оцінкою абсолютних величин змінних дає змогу відкинути x_1, в результаті чого отримуємо $L(0; -0{,}3) = 0{,}045$. За методом OBD ваги діляться на відповідні діагональні елементи оберненої матриці Гессе, що призводить до відсікання x_2 і дає значення $L(0{,}1; 0) = 0{,}02$. Метод OBS оновить значення $x_1 = -0{,}1$, щоб врахувати той факт, що тепер x_2 дорівнює нулю . Таким чином, оновлена після відсікання точка буде $(0; 0)$, що призводить до оптимального значення $L(0; 0) = 0$ [10].

Отже, проаналізовані методи відсікання — дуже актуальна та цікава тема оптимізації машинного навчання, проте на даний момент використання методів відсікання на практиці досить обмежене [11, 12]. Наразі відсутня жодна жорстка теорія, яка гарантувала б успіх чи невдачу конкретної стратегії відсікання. Тим не менше, потенційна вигода та користь від методу занадто велика, і це стимулює подальші дослідження в цьому напрямку. Немає підстав сумніватися, що в

найближчі кілька років очікується активна наукова діяльність у цьому напрямку і поява багатьох інноваційних ідей.

2.1.6 Квантування

Сучасний стан галузі машинного навчання стимулює розвиток методів оптимізації виконання обчислень і пам’яті, що використовується для зберігання числових даних. Одним з таких методів є метод квантування моделей. Під квантуванням моделі мається на увазі зниження розрядності представлення її параметрів. Це дає можливість створити більш компактне представлення моделі та використовувати високопродуктивні обчислювальні операції на багатьох апаратних платформах. Зазвичай більшість моделей навчанні використовувати 32-бітні (32-розрядні) числа з плаваючою комою, щоб скористатися перевагами їхнього більш широкого діапазону значень також відомого як динамічний діапазон. Однак під час висновків цим моделям може знадобитися більше часу для прогнозування порівняно зі зниженою точністю висновків. У багатьох випадках краще використовувати 8-бітні (8-розрядні) цілі числа. Тоді, наприклад, перехід від 32-бітного до 8-бітного формату чисел зменшить розмір моделі в чотири рази, тому однією з очевидних переваг квантування є скорочення потрібного обсягу пам’яті.

Схематично квантування подане рисунком 2.5.

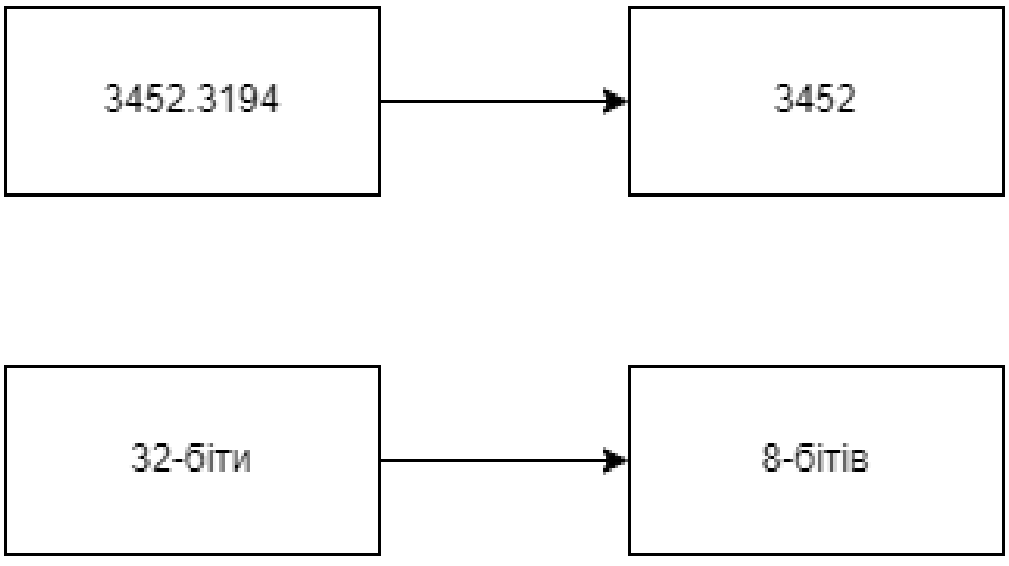


Рисунок — 2.5 Квантування

Техніка квантування має багато переваг:

— під час обробки 8-бітних цілих числових даних процесори використовують більш швидкі та доступні 8-бітні тензорні ядра для виконання операцій згортки та множення матриць. Це забезпечує більшу пропускну спроможність обчислень;

— переміщення даних із пам’яті в обчислювальні елементи вимагає часу, енергії та вибірково виділяє тепло. Зменшення точності даних активації та параметрів з 32-бітних чисел з плаваючою комою до 8-бітних цілих чисел призводить до 4-кратного скорочення обсягу даних, що заощаджує електроенергію та зменшує виділення тепла;

— зменшення обсягу пам’яті призводить до того, що модель потребує менше місця для зберігання, менше параметрів для оновлення, збільшується використання кешу тощо.

Математична модель квантування [13, 14]. Операція квантування відображає значення з плаваючою комою $x \in [\alpha; \beta]$ у цілочисельне значення $x_q \in [\alpha_q; \beta_q]$. Це відображення (процес квантування) задається функцією:

$$x_q = round\left(\frac{1}{c}x - d\right),$$

де c та d — сталі, які потрібно визначити. $round$ — це функція, яка застосовує певну політику округлення раціональних чисел до цілих.

Тоді обернене відображення (деквантування) визначається як:

$$x = c\left(x_q + d\right)$$

Сталі c та d знаходяться за умови, що α відображається в α_q та відповідно β відображається в β_q. Отже, потрібно розв’язати систему лінійних рівнянь:

$$\beta = c(\beta_q + d)$$
$$\alpha = c(\alpha_q + d)$$

розв’язок має наступний вид:

$$c = \frac{\beta - \alpha}{\beta_q - \alpha_q}$$
$$d = \frac{\alpha\beta_q - \beta\alpha_q}{\beta - \alpha}$$

Зазначимо, що практично значення 0 у числах з плаваючою комою відображається точно у ціле після квантування. Тоді з формули деквантування слідує, що d — ціле.

Отже, повинно бути:

$$x_q = round\ \ \frac{1}{c}0 - d\ \ = round(-d) = -round(d) = -d$$

Звідси отримуємо:

$$d = round(d) = round\left(\frac{\alpha\beta_q - \beta\alpha_q}{\beta - \alpha}\right)$$

Позначимо c як s (параметр масштабування), а $-d$ — як нульову точку z (параметр зсуву).

Тоді процес деквантування визначається як:

$$x = s\left(x_q - z\right)$$

Відповідно процес квантування буде мати вигляд:

$$x_q = round\ \ \frac{1}{s}x + z$$

Параметри s та z визначаються за формулами:

$$s = \frac{\beta - \alpha}{\beta_q - \alpha_q}$$

$$z = round\left(\frac{\beta\alpha_q - \alpha\beta_q}{\beta - \alpha}\right)$$

Зауважимо, що z — це ціле число, а s — це додатне число з плаваючою комою.

Відсікання значень. На практиці процес квантування може призводити до того, що значення x вийде за межі діапазону $[\alpha;\beta]$, тоді відповідне ціле квантоване значення x_q також вийде за межі діапазону $\left[\alpha_q;\beta_q\right]$.

Для усунення цього недоліку квантування застосовується додаткова процедура відсікання (clip):

$$x_q = clip\left(round\left(\tfrac{1}{s}x + z\right), \alpha_q, \beta_q\right),$$

де функція відсікання $clip(x; l; u)$ визначається, як

$$clip(x; l; u) = \begin{cases} l, x < l \\ x, l \leq x \leq u \\ u, x > u \end{cases}$$

Розглянуте відображення квантування носить назву афінного відображення квантування.

Симетричне і асиметричне квантування. Якщо необхідно відобразити значення з плаваючою комою $x \in [\alpha; \beta]$ у цілочисельні квантовані значення з симетричного інтервалу $x_q \in [\alpha_q; \beta_q]$, де величина параметра z вибирається рівною нулю, то математично це означає, що:

$$\alpha_q = -\beta_q$$

$$round\left(\frac{\beta\alpha_q - \alpha\beta_q}{\beta - \alpha}\right) = 0$$

Це призводить до того, що $\alpha = -\beta$. Таким чином, відображається симетричний відносно нуля діапазоном чисел з плаваючою комою $[\alpha; -\alpha]$ на цілочисельний діапазоном $[\alpha_q; -\alpha_q]$. У цьому випадку відображення має назву симетричного квантування, якщо ж $\alpha \neq -\beta$ — асиметричного квантування.

Відзначимо, що застосування асиметричного квантування дає можливість щільніше описати безліч значень даних і таким чином зробити обчислення більш точними.

2.1.7 Паралельна обробка

Використання можливостей паралельної обробки може покращити продуктивність моделі, особливо на сучасних багатоядерних процесорах. Паралельна обробка може бути реалізована за допомогою різних додаткових інструментів. Ці інструменти забезпечують можливості паралельної обробки в різних частинах конвеєра машинного навчання [15].

Важливо зазначити, що використання паралельної обробки може допомогти значною мірою зменшити час обчислень та покращити продуктивність алгоритмів машинного навчання. Однак, перед тим як використовувати паралельні можливості, важливо правильно розробити та оптимізувати конвеєр моделі для максимальної ефективності. Також, враховуючи особливості паралельної

обробки, слід бути обережним із використанням спільних ресурсів та уникати ситуацій спільного використання одних даних.

Паралельна обробка в різних етапах розробки моделі, включаючи підготовку даних, навчання моделі та оцінку результатів, може бути особливо корисною для великих обсягів даних, де оптимізація швидкості грає вирішальну роль у досягненні вдалих результатів.

Паралельне програмування. У сучасному інформаційному світі це стало важливою складовою для розв’язання задач, які вимагають великої обчислювальної потужності. Цей підхід полягає в одночасному виконанні кількох фрагментів програми або завдань, забезпечуючи оптимальне використання ресурсів. Величезний розвиток багатоядерних процесорів та паралельних обчислювальних систем створює унікальні можливості для впровадження паралельного програмування.

Однією з популярних моделей паралельного програмування є модель потоків (рисунок 2.6). Завдяки використанню потоків можливо розділити великі завдання на менші, що обробляються паралельно. Це забезпечує підвищену продуктивність, адже програма може використовувати всі доступні ресурси багатоядерних процесорів.

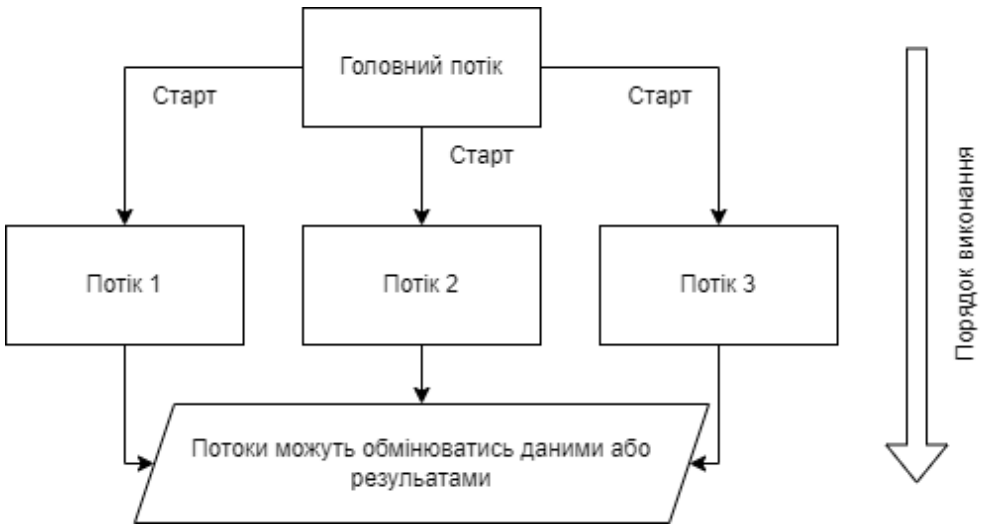


Рисунок 2.6 — Поведінка потоків у паралельному програмуванні

Проте паралельне програмування може викликати проблеми, такі як гонки за даними і блокування ресурсів (рисунок 2.7). Застосування правильних стратегій

синхронізації і керування потоками стає ключовим аспектом успішної реалізації програм із використанням паралельної обробки потоків [16].

У контексті розробки програмного забезпечення, яке взаємодіє з великими обсягами даних чи виконує складні обчислення, важливо ефективно використовувати можливості паралельного програмування. Сучасні інструменти і бібліотеки, такі як OpenMP для мов програмування C/C++ чи Java ForkJoinPool, надають зручні засоби для реалізації програм із паралельною обробкою потоків.

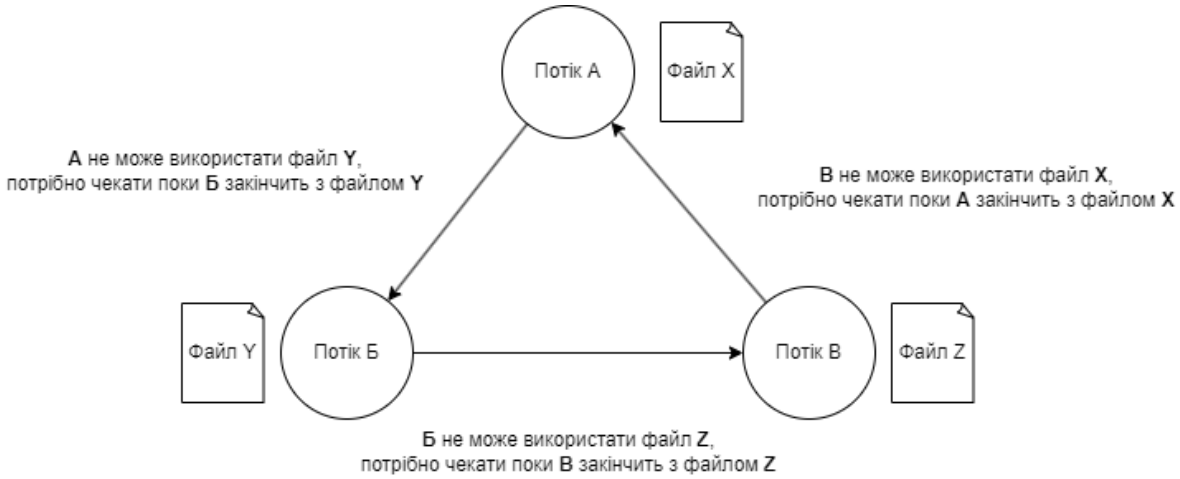


Рисунок 2.7 — Блокування ресурсів у паралельному програмуванні

Усі ці аспекти враховуються при розробці програмних рішень з використанням паралельного програмування для забезпечення максимальної ефективності, швидкодії та використання ресурсів обчислювальної системи.

2.1.8 Оптимізація гіперпараметрів

Оптимізація гіперпараметрів у моделях машинного навчання є важливим етапом, спрямованим на вибір оптимальних комбінацій параметрів моделі та гіперпараметрів, що не можуть бути оцінені самою моделлю, для досягнення максимальної ефективності та використання її потенціалу. У машинному навчанні існують два простори параметрів:

— параметри моделі — характеристики, які модель може адаптувати або налаштовувати на основі введених даних, такі як ваги критеріїв у нейронних мережах, що дає можливість моделі адекватно відповідати різноманітним сценаріям і вимогам;

— гіперпараметри — це параметри, які визначаються заздалегідь і не можуть бути автоматично налаштовані самою моделлю за допомогою наявних даних.

Математичне подання [17, 22]. Розглянемо, наприклад, що модель лінійної регресії визначена, як:

$$f(X) = Xw,$$

де $w = (w_0; w_1; \dots; w_n)$ — ваги моделі, $X = (x_{ij})$ — матриця, в якій кожен рядок містить ознаки одного об'єкта вибірки (для зручності можна вважати, що перший стовпчик у цій матриці є константним).

Ця модель може навчатися шляхом мінімізації наступного функціоналу:

$$L = |y - Xw|^2 + C|w|^2,$$

де, y — цільова змінна, C — коефіцієнт регуляризації.

Під час мінімізації L ваги w налаштовуються за допомогою навчального набору, тобто вони є параметрами. У той же час величина коефіцієнта регуляризації визначається перед початком навчання, тобто вона є гіперпараметром.

$$L = |y - Xw|^2 + C|w|^2 \to min$$

Звідси C — гіперпараметр, w — параметр.

Як можна побачити на рисунку 2.8, якість моделі може значно варіюватися залежно від гіперпараметрів, тому існують різноманітні методи та інструменти для їхнього підбору. При цьому, незалежно від обраного методу підбору гіперпараметрів, оцінювання та порівняння моделей слід проводити ретельно.

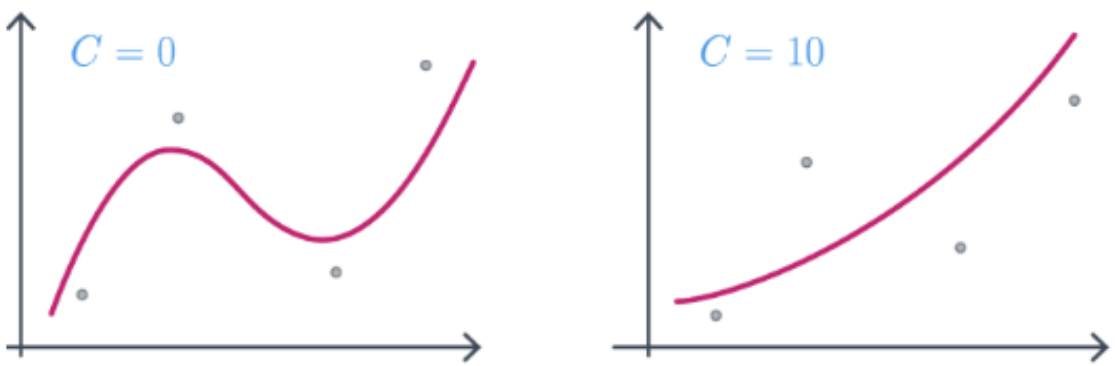


Рисунок 2.8 — Порівняння значень гіперпараметрів

Налаштування гіперпараметрів. Це процес визначення оптимальної комбінації гіперпараметрів, яка максимізує продуктивність моделі. Це здійснюється запуском кількох випробувань у межах одного процесу навчання. Кожне випробування — це повне виконання навчального застосування із значеннями обраних гіперпараметрів, встановленими в межах, які попередньо були вказані. Цей процес надасть набір значень гіперпараметрів, які найкраще підходять для моделі для отримання оптимальних результатів.

Ручний спосіб. Спосіб передбачає експериментування з різними наборами гіперпараметрів вручну, тобто кожний експеримент із заданим набором гіперпараметрів виконується людиною. Ця техніка вимагає потужного інструмента відстеження експериментів, який може відстежувати різноманітні змінні від зображень та історичних записів до системних метрик.

Існують сучасні інструменти для відстеження експериментів, які відповідають усім вимогам. У інструментах є можливість легко задавати гіперпараметри та переглядати всі види результатів даних, таких як зображення, метрики тощо.

Переваги ручної оптимізації гіперпараметрів:

— ручне налаштування гіперпараметрів означає більший контроль над процесом;

— якщо виконується дослідження або вивчення налаштування і його вплив на ваги мережі, ручне виконання оптимізації має сенс.

Недоліки ручної оптимізації гіперпараметрів:

— виконання вручну — це трудомісткий процес, оскільки може бути багато експериментів, а ведення контролю може виявитися витратним і займати багато часу;

— це не дуже практичний підхід, коли є багато гіперпараметрів для розгляду.

Автоматизований спосіб. Цей спосіб використовує вже існуючі алгоритми для автоматизації процесу. Спочатку треба визначити набір гіперпараметрів та обмеження для значень цих гіперпараметрів. Кожен алгоритм вимагає, щоб цей

набір був конкретною структурою даних, наприклад, часто використовують словники при роботі з алгоритмами.

Потім алгоритм виконує важку роботу. Він запускає ці експерименти і повертає найкращий набір гіперпараметрів, який забезпечить оптимальні результати.

Метод Grid Search [18]. Найбільш звичайний спосіб організувати перебір наборів гіперпараметрів — використовувати перебір по сітці (Grid Search):

— фіксуються декілька значень для кожного гіперпараметра;

— перебираються всі комбінації значень різних гіперпараметрів, при цьому на кожній з цих комбінацій модель навчається і тестується;

— обирається комбінація, на якій модель виявляє найкращу якість.

Приклади:

— для методу найближчих сусідів можна перебирати кількість сусідів (наприклад, від 1 до 20) і метрику, за якою буде вимірюватися відстань між об’єктами вибірки (евклідова, мангеттенська тощо);

— для дерев можна перебирати по сітці комбінації значення максимальної глибини дерева та різних критеріїв розгалуження (критерій Джині, ентропійний критерій тощо).

Перебір деяких значень гіперпараметрів можна вести за логарифмічною шкалою, оскільки це дає можливість швидше визначити правильний порядок параметра та одночасно значно зменшити час пошуку. Наприклад, так можна підбирати значення швидкості навчання для методу градієнтного спуску, значення константи регуляризації для лінійної регресії чи методу опорних векторів.

Одразу виникає розуміння обмеженості цього методу: якщо комбінацій параметрів занадто багато або кожне навчання або тестування потребує занадто тривалого часу, а отже, алгоритм потребує великих ресурсів часу.

У випадку, коли виникає дуже велика кількість комбінацій параметрів, спробувавши використати типові рішення, виникають такі проблеми:

— якщо взяти менше значень кожного гіперпараметра, тоді існує можливість пропустити найкращу комбінацію;

— якщо зменшити кратність у крос-валідації, тоді оцінка параметрів стане менш точною;

— якщо оптимізувати параметри послідовно і не перебирати їхні комбінації, то існує шанс отримати неоптимальне рішення.

Метод Random Search [19]. У цьому методі для кожного гіперпараметра задається розподіл, з якого вибирається його значення, і комбінація гіперпараметрів складається відбором з цих розподілів. Таким чином, завдяки випадковому вибору наступної комбінації гіперпараметрів є можливість знайти оптимальну комбінацію за меншу кількість ітерацій. На рисунку 2.9 добре ілюструється відмінність між пошуком по сітці та випадковим пошуком.

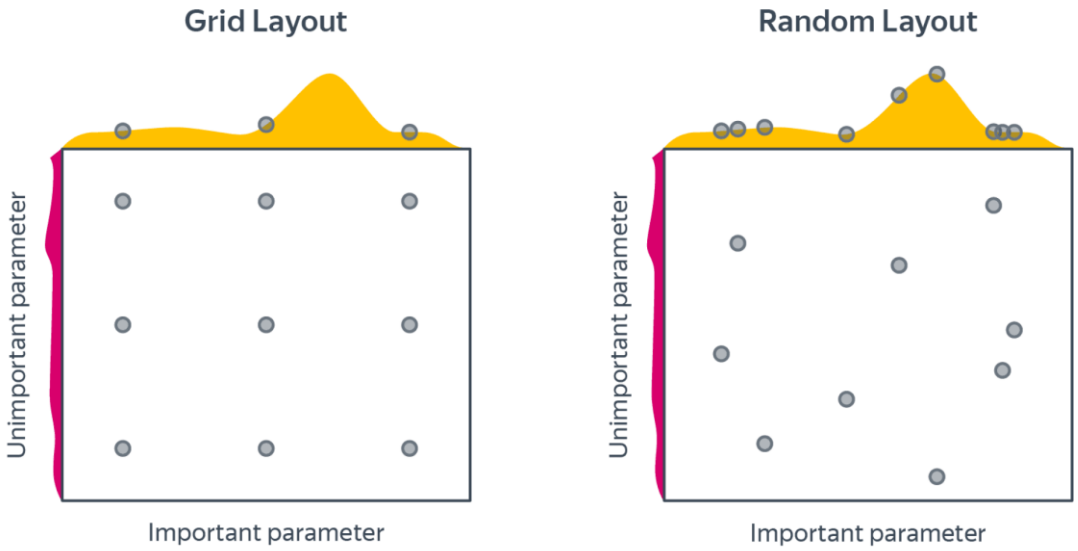


Рисунок 2.9 — Порівняння Grid Search та Random Search [19]

Якість моделі залежить від гіперпараметрів, які утворюють функцію багатьох змінних, яка описує деяку гіперповерхню [20]. Проте ця поверхня може від однієї змінної залежати значно менше, ніж від іншої. Якщо знати, який гіперпараметр важливіший для швидкості та точності моделі, то можна розглянути більше його можливих значень, але на практиці такий параметр невідомий, і потрібно розглядати деяку наперед визначену кількість значень для кожного гіперпараметра.

Випадковий пошук може за ті ж самі ітерації, що й Grid Search, розглянути більш різноманітні значення гіперпараметрів. Таким чином, він з більшою ймовірністю знайде ті значення, які найбільше впливають на якість моделі та з більшою ймовірністю знайде найкращу комбінацію значень гіперпараметрів.

Розглянемо випадок, коли є кінцева сітка гіперпараметрів (кожному гіперпараметру відповідає кінцева кількість значень). Наприклад, у сітці виділяється група розміру 6 від загальної кількості наборів гіперпараметрів, на якій модель досягає найкращої якості (можна уявно ранжувати всі набори за якістю в певний список і взяти перші найкращі 6 елементів цього списку). Тоді якийсь набір гіперпараметрів не потрапляє в цю групу з ймовірністю $(1 - 0{,}06)^n$.

Якщо відібрано n наборів, то кожен з них не потрапив в цю групу з ймовірністю $(1 - 0{,}06)^n$, відповідно, ймовірність того, що хоча б один невідібраний набір потрапив в найкращу групу, дорівнює $1 - (1 - 0{,}06)^n$.

Розв’яжемо нерівність:

$$1 - (1 - 0{,}06)^n \geq 0{,}94$$

звідси маємо, що при $n \geq 60$ ймовірність влучити в вибірку перших найкращих елементів не менше ніж 0,94.

Це у більшості випадків значно швидше, ніж перебір всіх комбінацій гіперпараметрів за допомогою Grid Search.

Якщо деякі гіперпараметри мають неперервний розподіл, то завжди можна припустити, що вже було відібрано з цих розподілів деяку кількість значень (рівну кількості ітерацій Random Search), далі слід вважати, що потрібно працювати з кінцевою сіткою.

Але, залишається залежність від самої сітки гіперпараметрів, і не будь-яка сітка має містити в собі глобальний максимум швидкості та точності моделі, або навіть гіперпараметри з інтервалу навколо нього.

Оптимізація Баєса [21]. Це ітераційний метод, що дає можливість оцінити оптимум функції, не диференціюючи її. Крім того, на кожній ітерації метод вказує, в якій наступній точці з найбільшою ймовірністю буде поліпшено поточну оцінку оптимуму. Це дає можливість значно скоротити кількість обчислень функції, кожне з яких може бути досить витратним у часі.

Підбір гіперпараметрів також можна сформулювати як задачу, яка може бути вирішена за допомогою оптимізації Баєса. Наприклад, нехай функція представляє собою значення валідаційних метрик у залежності від поточної комбінації гіперпараметрів. Обчислення цієї функції займає багато часу (необхідно навчити та провалідувати модель), та обчислити градієнти цієї функції за її змінними (гіперпараметрами) є неможливим.

Оптимізація Баєса складається з двох основних компонент:

— ймовірнісна модель — модель, яка наближає розподіл значень цільової функції в залежності від наявних історичних даних (часто вибирають гауссівські процеси);

— функція acquisition — функція, яка, опираючись на деякі статистики поточної ймовірнісної моделі, вказує, в якій наступній точці потрібно обчислити значення. Ця функція називається acquisition function. Вона повинна збалансувати між exploration і exploitation так:

- значення Exploration — дослідження тих точок, де дисперсія заданої ймовірнісної моделі велика;
- значення Exploitation — дослідження тих точок, де середнє значення ймовірності заданої моделі велике (і може служити оцінкою максимуму).

Простий приклад acquisition function — це сума середньої ймовірнісної моделі і стандартного відхилення з деякою вагою:

$$\alpha(x) = \mu(x) + \beta\sigma(x),$$

де x — точка у просторі, в якому оптимізуємо цільову функцію (у даному контексті — вектор значень гіперпараметрів).

На зазначеному рисунку 2.10 можна спостерігати обидві ключові компоненти, що формують acquisition function [21]. Перша компонента μ відображена синім графіком і представляє середнє значення ймовірнісної моделі, що є вказівником на центральний тенденції моделі. Друга компонента відображена сірою областю і представляє довірчий інтервал, ширина якого в кожній точці графіка пропорційна стандартному відхиленню ймовірнісної моделі, що визначає ступінь впевненості в прогнозах моделі. Такий двокомпонентний

підхід дає можливість враховувати як саме середнє значення, так і ступінь невизначеності моделі при прийнятті рішень.

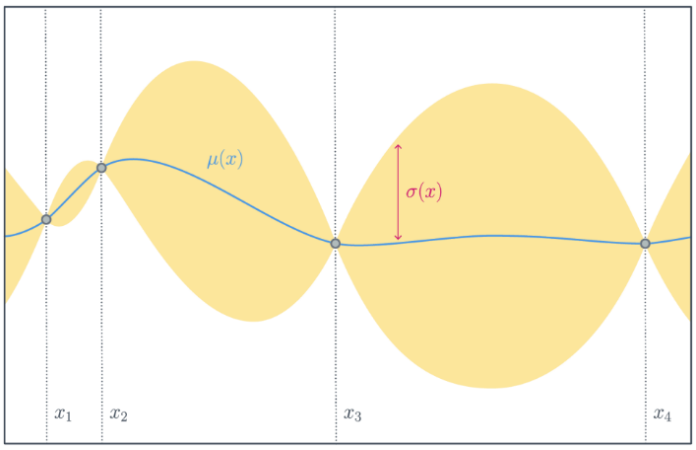


Рисунок 2.10 — Представлення acquisition function

Середнє значення моделі μ намагається наблизити шукану функцію f і точне значення f у тих точках, де значення f відомі. Довірчий інтервал має змінну ширину, оскільки чим далі розташована деяка точка від тих, де значення відомі, тим менше впевнена модель в тому, яке значення функції в цій точці, і тим ширший довірчий інтервал. Навпаки, у точках, де значення відомі, довірчий інтервал має нульовий радіус.

Оптимізація Баєса у загальному випадку представляє собою наступний алгоритм, для ефективної роботи якого, α — повинна бути легко обчислюваною та диференційовною:

— нехай S_t — множина попередніх спостережень цільової функції $f: \left(f(x_1); \ldots; f(x_t)\right)$, а $\alpha(\cdot)$ — деяка acquisition function;

— на ітерації $t+1$ обчислюється точка x_{t+1}, в якій потрібно провести наступне обчислення цільової функції: $x_{t+1} = \arg\max_{x \in X} \alpha(x|S_t)$;

— обчислюється значення $f\left(x_{t+1}\right)$ та оновлюється множина спостережень: $S_{t+1} = S_t; f\left(x_{t+1}\right)$;

— оновлюється статистична модель.

У поданому алгоритмі оптимізації Баєса (рисунок 2.11) використовується така логіка. Пунктирна лінія позначає цільову функцію, а суцільна лінія — графік середнього значення ймовірнісної моделі. Жовтий колір позначає довірчий інтервал моделі.

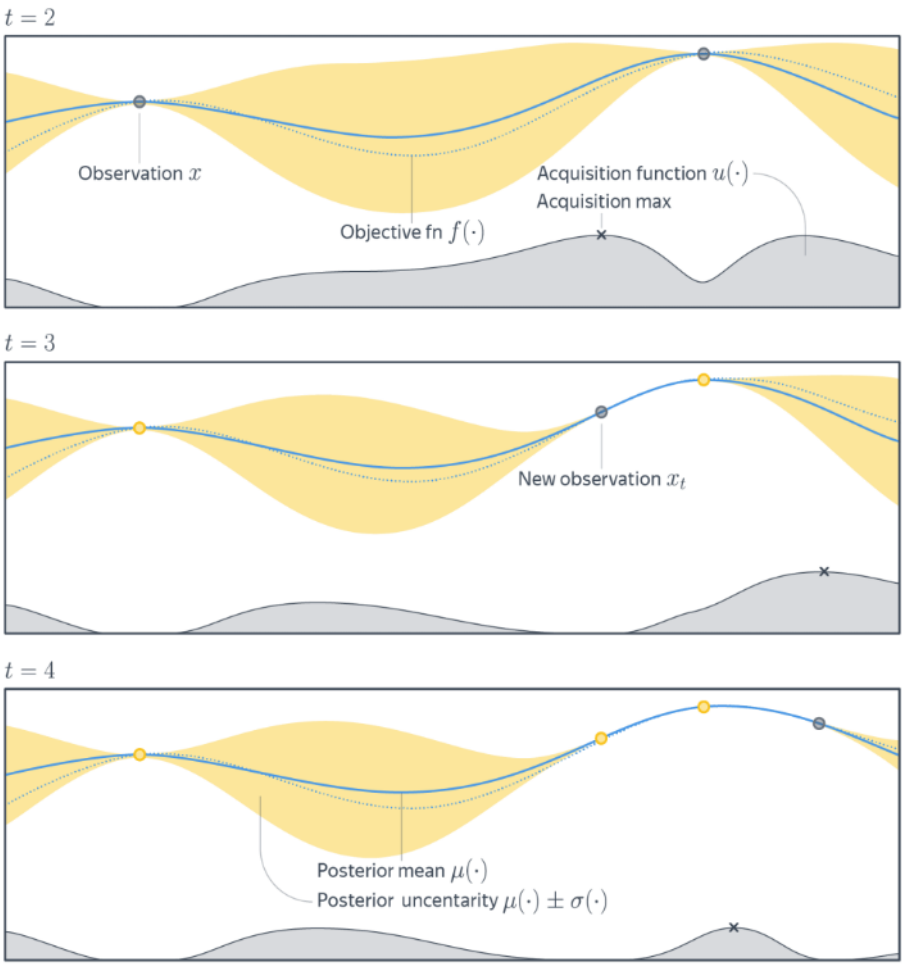


Рисунок 2.11 — Подання оптимізації Баєса у трьох ітераціях

Сірий графік знизу демонструє графік функції acquisition function. Її значення великі в областях, де ймовірнісна модель передбачає високі значення цільової функції (експлуатація), та в областях великої невизначеності ймовірнісної моделі (дослідження).

На кожній ітерації шукається точка максимуму функції acquisition function (чорний хрестик), і наступна ітерація відбувається в цій точці (сірий кружечок на графіку функції). На нижньому графіку виграє експлуатація, оскільки цільова функція правильно передбачила, що спостереження з невідомих областей слабо впливають на поточну оцінку максимуму f.

Підсумок. Розглянувши декілька методів, можна стверджувати, що при виборі методу оптимізації гіперпараметрів для моделей машинного навчання, важливо розуміти переваги та обмеження кожного з методів [23]:

— метод Grid Search — простий метод, добрий для невеликої кількості гіперпараметрів, але тривалий, особливо якщо не має можливості паралельно виконувати його;

— метод Random Search — поліпшення версія Grid Search, ефективніший завдяки випадковому перебору гіперпараметрів. Тривалий, не використовує результати попередніх ітерацій;

— оптимізація Баєса — використовує результати попередніх ітерацій, може моделювати внутрішні залежності між гіперпараметрами. Досягає вищої якості, але не так простий у паралельній реалізації, може бути тривалим, особливо з гауссівськими процесами і категоріальними гіперпараметрами [24].

2.1.9 Кешування і попереднє завантаження моделей

Використання кешування може сприяти уникненню повторного завантаження моделі при кожному виклику, що може покращити час відгуку застосунку [25].

Оптимізація ресурсів є критичним аспектом в контексті машинного навчання. Ініціювання завантаження моделей перед фактичним викликом може позитивно впливати на продуктивність системи та сприяти ефективній роботі алгоритмів машинного навчання.

Кешування для зменшення затримок важливе, де кожна мілісекунда має значення. Використання кешування може бути ключовим елементом для зменшення затримок при викликах моделей та оптимізації реакції системи.

Забезпечення консистентності є іншим важливим аспектом. Зберігання попередньо розрахованих результатів може запобігти непередбаченим змінам у моделі та гарантувати стабільність результатів.

При впровадженні кешування і попереднього завантаження моделей важливо аналізувати архітектурні особливості системи. Це дає можливість

ефективно взаємодіяти інтегрованим чином, уникати конфліктів та забезпечувати оптимальну працездатність.

Методи виявлення і керування кешуванням можуть бути застосовані для автоматичного оновлення кешу та гарантування актуальності даних в разі змін у моделі.

У випадках моделей з великим обсягом даних ефективне кешування може вимагати спеціального підходу. Розгляд асинхронного кешування і оптимізованих стратегій завантаження може сприяти підвищенню продуктивності системи.

Важливо враховувати аспекти безпеки та конфіденційності при використанні кешування. Заходи, як-от: шифрування та контроль доступу до кешованих даних, є ключовими для забезпечення високого рівня захисту інформації.

2.2 Засоби розробки

У ході розробки програмного продукту роботи використано сучасні технології та інструменти, які є стандартами у сфері машинного навчання. Основним інструментом для розробки моделей машинного навчання став ML.NET — фреймворк, який включає в себе сучасні засоби взаємодії та використання машинного навчання. Створений на платформі .NET із відкритим вихідним кодом. Платформа є популярною серед науковців та інженерів завдяки своїй простоті, гнучкості та підтримкою мови C#.

В якості бази даних використано PostgreSQL — потужну систему керування реляційними базами даних з відкритим вихідним кодом. Вона підтримує широкий спектр типів даних і має багатий набір функцій для обробки та аналізу даних.

Для взаємодії з базою даних використано бібліотеку Entity Framework, яка надає можливість керування і взаємодії з даними за допомогою C#.

Веб-застосунок створено за допомогою технології Blazor, а саме моделі Server. Це фреймворк для створення інтерактивних клієнтських веб-застосунків на .NET. Він дає можливість розробникам писати код мовою C# замість JavaScript. Інновація цього фреймворку — це те, що код програми виконується на

сервері. Клієнтський та серверний код взаємодіють через технологію SignalR для з’єднання зв’язку, яке встановлюється між браузером і сервером.

2.2.1 Середовище розробки JetBrains Rider

Середовище JetBrains Rider — це кросплатформене інтегроване середовище розробки, яке забезпечує високопродуктивну розробку .NET на C#, VB.NET, а також F#. Продукт Rider, створений компанією JetBrains, яка розробляє професійні інструменти для розробників програмного забезпечення, та є одним із найпотужніших інструментів для розробки .NET.

Одна із ключових особливостей Rider — його глибока інтеграція з технологією ReSharper, іншим популярним продуктом JetBrains. Технологія ReSharper додає багато корисних функцій і вдосконалень до Rider, включаючи розширену підтримку рефакторингу, код-аналізу, навігацію по коду та інше.

Середовище Rider підтримує широкий спектр технологій і мов програмування, включаючи більшість найвідоміших: .NET Core, Xamarin, Unity, ASP.NET, JavaScript, TypeScript, HTML, CSS, SQL.

Це робить Rider універсальним інструментом для розробників, які працюють з різними технологіями і платформами.

У середовище також вбудований налагоджувач (debugger), який дає можливість розробникам швидко та ефективно виявляти і виправляти помилки в коді. Він підтримує різні типи застосунків, включаючи консольні застосунки, застосунки для робочого столу, веб-застосунки тощо.

Наявні у середовищі функції включають в себе потужну систему керування версіями, яка підтримує такі технології, як: Git, Subversion, Perforce і Mercurial. Це дає можливість розробникам легко відстежувати зміни в коді, здійснювати завантаження коду в систему контролю версій, відміняти зміни тощо.

Однією з переваг Rider є його висока швидкість виконання операцій та відгуку інтерфейсу. Rider використовує інтелектуальну технологію кешування та індексації, що дає можливість забезпечити швидку роботу навіть з великими проєктами.

Гнучкі налаштування дають можлвість розробникам налаштовувати середовище розробки відповідно до своїх потреб. Це включає налаштування редактора коду, налаштування налагоджувача, налаштування системи керування версіями тощо.

Середовище Rider також має вбудовану підтримку для роботи з базами даних. Це включає підтримку мови SQL, можливість перегляду та редагування даних, виконання запитів до бази даних тощо.

Все перераховане визначає JetBrains Rider, як потужний інструмент для розробки .NET, який може задовольнити потреби навіть найвимогливіших розробників.

2.2.2 Платформа .NET

Платформа .NET є відкритим і універсальним засобом для розробки різноманітних типів програмного забезпечення, включаючи веб-застосунки, мобільні застосунки, стаціонарні Desktop застосунки, ігри та багато іншого [26, 28]. Компанія Microsoft розробила цю платформу і вперше представила у червні 2000 року.

Однією з ключових особливостей платформи є підтримка багатьох мов програмування. Можна використовувати C#, F#, VB.NET та багато інших мов для розробки програм на .NET. Більш того, кожен застосунок написаний однією мовою, може використовувати код, написаний на іншій мові, завдяки Common Language Runtime (загальномовне середовище виконання), скорочено CLR.

В основі платформи .NET вбудований Common Language Runtime (CLR), що дає можливість виконувати програмний код, написаний різними мовами програмування, як це видно з рисунка 2.12. Компонент CLR включає в себе не лише можливість інтеграції мов, але й надає низку сервісів, таких як керування пам’яттю, автоматичне очищення пам’яті, систему безпеки та обробку винятків, що робить його важливою складовою функціональності фреймворку.

Платформа .NET також забезпечує великий набір бібліотек та API, які полегшують розробку програмного забезпечення. До цього набору належать ADO.NET для роботи з базами даних, ASP.NET для розробки веб-застосунків,

WCF (Windows Communication Foundation) для розробки веб-сервісів та багато іншого.

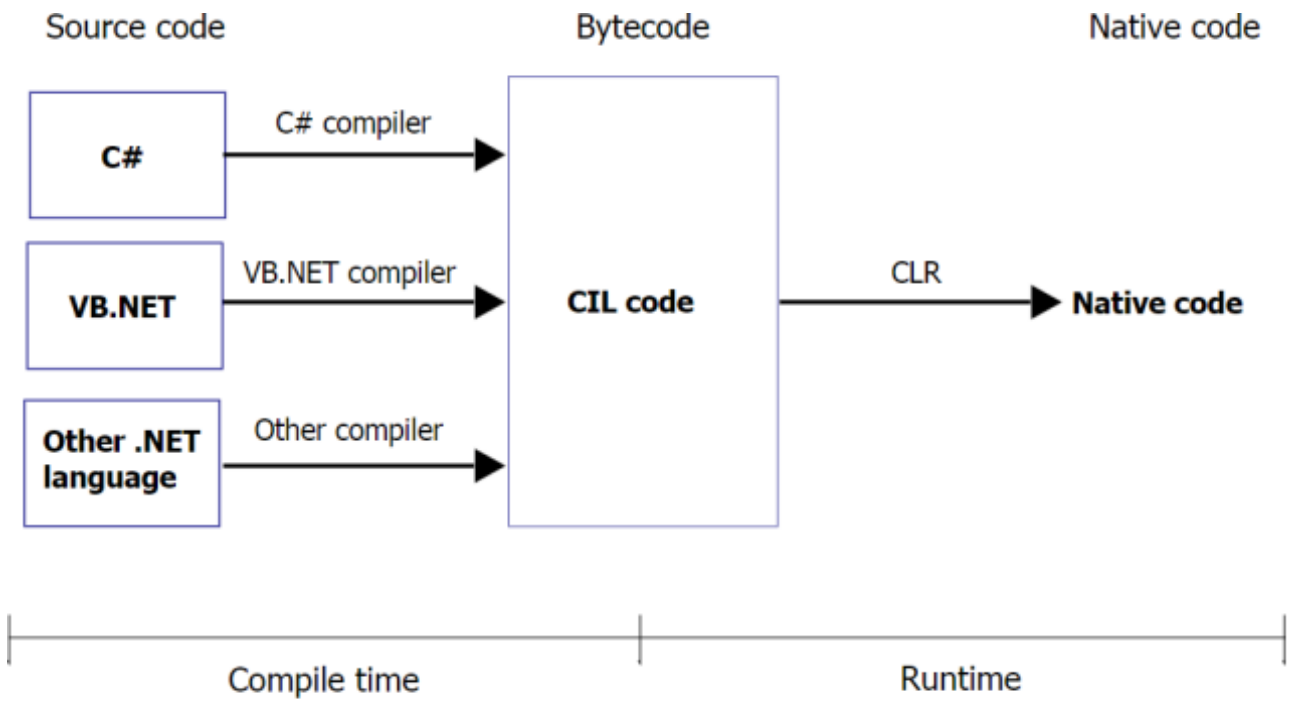


Рисунок 2.12 — Схема роботи CLR

Однією з основних переваг .NET є швидкість виконання та загальна ефективність використання для широкого спектру задач. Компонент CLR забезпечує виконання коду з високою продуктивністю, а оптимізовані бібліотеки дають можливість написати більш ефективний і стислий код з меншою ймовірністю кількості помилок.

Платформа .NET також відома своєю масштабованістю. Будь-який програмний продукт, розроблений з використанням .NET, може бути легко масштабований залежно від потреб використання. Фреймворк також підтримує розподілену обробку, що дає можливість застосунку ефективно використовувати ресурси системи.

Як проєкт з відкритим вихідним кодом, платформа .NET має активну спільноту розробників, які неперервно вносять зміни та вдосконалення. Існує чимало ресурсів для навчання та підтримки, включаючи документацію, приклади коду, бібліотеки, інструменти розробника тощо.

Платформа є досить привабливим засобом розробки для бізнесу, завдяки її кросплатформності. Завдяки новим формату платформи .NET Core і Xamarin, програми на платформі .NET можуть працювати в операційних системах Windows, macOS, Linux і навіть на мобільних операційних системах iOS і Android.

У загальному, платформа.NET є потужною, гнучкою та надійною і забезпечує широкий спектр можливостей для розробників програмного забезпечення. Вона включає в себе все необхідне для розробки високоякісних, надійних і безпечних застосунків, які можуть відповідати будь-яким комерційним потребам.

2.2.3 Бібліотека ML.NET

Бібліотека ML.NET є вільно розповсюджуваною, високопродуктивною, відкритою бібліотекою для машинного навчання, створеною корпорацією Microsoft [31, 32]. Ця бібліотека дає можливість використовувати машинне навчання без необхідності мати спеціальну експертизу в цій області. Вона підтримує різноманітні типи машинного навчання, включаючи виявлення аномалій, бінарну класифікацію, багатокласову класифікацію, регресію, ранжування і прогнозування часових рядів.

Бібліотека ML.NET вперше була представлена у 2018 році і надалі разом з продовженням свого розвитку робить значний внесок у збільшення можливостей використання машинного навчання в .NET. Основна розробка бібліотеки ML.NET ведеться командою Microsoft Research, і вона активно використовується всередині компанії в продуктах Windows, Bing, PowerPoint і багатьох інших.

Особливістю бібліотеки ML.NET є те, що вона спроєктована з врахуванням потреб розробників .NET. Вона має чіткий і зрозумілий інтерфейс API, що дає можливість легко інтегрувати її в програми, створені на платформі .NET. Також вона має вбудовані алгоритми для обробки даних і навчання моделей, що значно спрощує процес розробки програм.

Одним із важливих елементів бібліотеки ML.NET є підтримка автоматичного машинного навчання, відомого як AutoML. Бібліотека не просто автоматизує процес вибору найефективнішої моделі для певного набору даних,

але також обробляє ці дані та тренує модель. Це значно полегшує процес розробки та виконання моделей машинного навчання. Таким чином, засіб AutoML в бібліотеці ML.NET допомагає розробникам зосередитися на розв'язанні конкретних проблем, а не на технічних деталях створення моделей. Бібліотека відкриває нові можливості для швидкого та ефективного використання машинного навчання в різних сферах [34].

Однією з ключових переваг ML.NET є його вбудована підтримка платформи .NET. Це означає, що є можливість використовувати знайомий синтаксис мови C#, інструменти та бібліотеки, щоб розробляти моделі машинного навчання, обирати найкращі та зберігати у зручному форматі (рисунок 2.13). Також існує можливість легко інтегрувати бібліотеку ML.NET в існуючі .NET-застосунки, без необхідності використовувати мови типу Python.

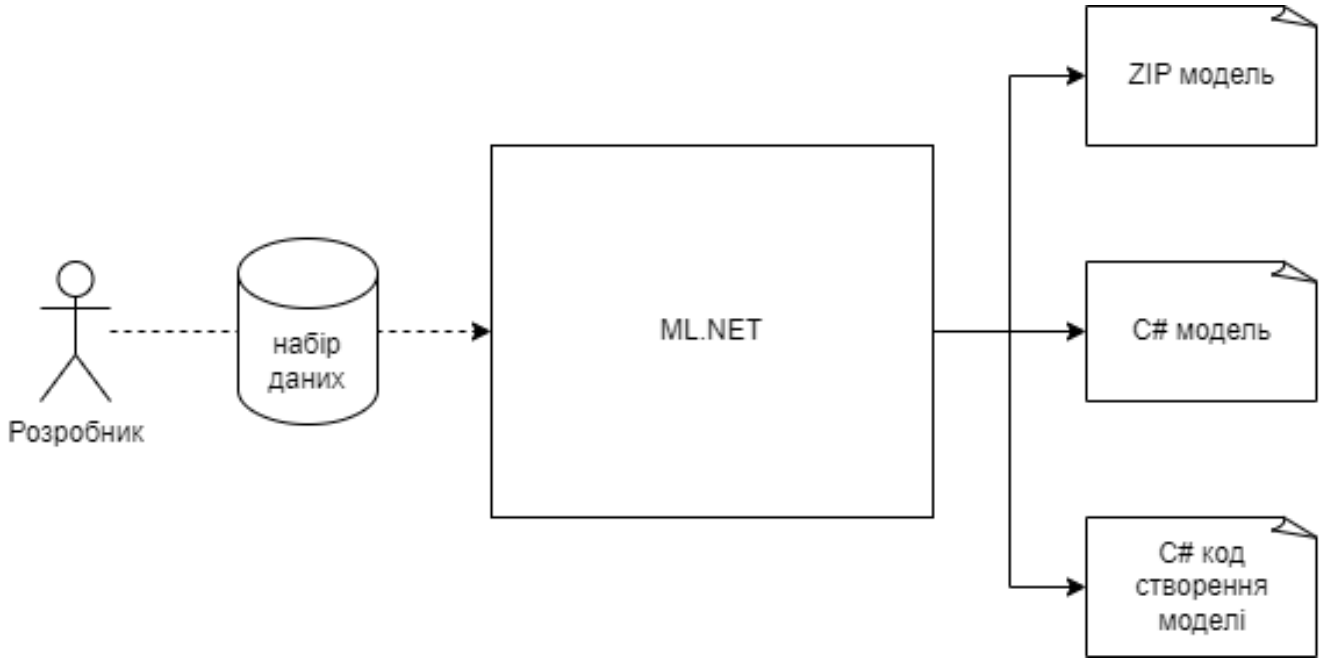


Рисунок 2.13 — Сценарій використання бібліотеки ML.NET розробником

Бібліотека ML.NET також спрямована на високу продуктивність і масштабованість. Вона включає оптимізовані алгоритми і колектори сміття із пам'яті, які забезпечують швидку обробку даних та навчання моделей. Бібліотека також підтримує паралелізацію та розподілене виконання, що дає можливість використовувати повну потужність обчислювальних ресурсів [33].

Ще однією важливою перевагою бібліотеки ML.NET є підтримка модуля трансформації даних. Модуль перетворення даних значно спрощує процес підготовки даних для навчання моделей, включаючи нормалізацію, дискретизацію і кодування даних. Модуль також підтримує великі набори даних і набори даних з пропущеними значеннями.

Цей засіб розробки також включає модуль для оцінки моделей, який допомагає визначити, наскільки добре працює модель. Цей модуль надає декілька різних метрик: точність, швидкість і математичні оцінки моделі, щоб допомогти визначити найкращу модель для певних потреб.

У загальному бібліотека ML.NET — це сильний і гнучкий інструмент для роботи з машинним навчанням в .NET. Її широкий функціонал допомагає розробникам швидко і ефективно розробляти та використовувати моделі машинного навчання, а підтримка .NET означає, що наявна можливість використовувати інші відомі інструменти і бібліотеки.

2.2.4 Технологія Blazor Server

Технологія Blazor, розроблена компанією Microsoft, надає можливість розробникам створювати інтерактивні веб-застосунки за допомогою мови програмування C# замість JavaScript. Технологія Blazor використовує WebAssembly, новий стандарт мережі Інтернет, який дає можливість виконувати код на боці клієнта з швидкістю, порівнянною зі звичайним серверним виконанням коду [26, 27].

Модель Server є однією з двох моделей розгортання, доступних для Blazor. Ця модель використовує SignalR для встановлення двостороннього зв’язку між браузером клієнта і сервером. Весь процес взаємодії користувача з застосунком відбувається на сервері, де результати обробки передаються назад до клієнта через SignalR. Це дає можливість забезпечити швидку та ефективну роботу застосунку, оскільки всі обчислення відбуваються на сервері, а не на клієнтському браузері (рисунок 2.14).

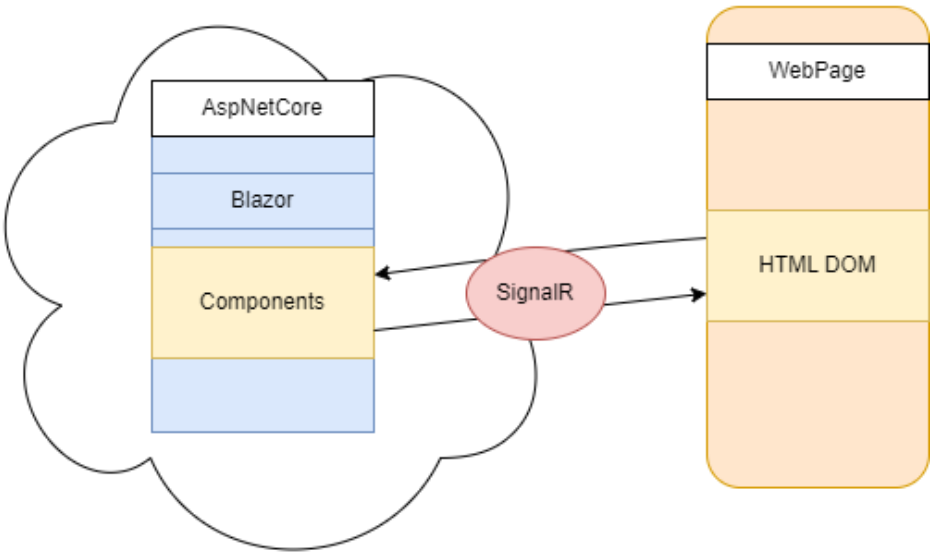


Рисунок 2.14 — Концепція Blazor Serer

Технологія Blazor була анонсована Microsoft у 2018 році як експериментальний проєкт. Вона була створена автором відомої бібліотеки Knockout.js. У 2019 році технологія Blazor, а саме модель Server була опублікована як частина платформи .NET Core.

Вона використовує компонентний підхід до розробки веб-додатків. Компоненти можуть бути успадковані, передавати дані один одному і реагувати на події користувача. Це робить код легким для розуміння і підтримки.

Засіб також підтримує формат Razor, що дає можливість розробникам використовувати мову розмітки сторінок HTML і мову C# у тому самому файлі (рисунок 2.15). Це робить код більш читабельним і дає можливість розробникам легко створювати динамічні веб-сторінки.

```
<h2>Products</h2>

<ul>
  @foreach (var p in products) {
     <li>
         @p.ProductName

         @if (p.UnitsInStock == 0) {
            @: (Out of stock!)
         }
     </li>
  }
</ul>
```

Рисунок 2.15 — Приклад написання коду Razor (C# та HTML)

Однією з ключових переваг моделі Server є те, що вона не вимагає великого завантаження пам’яті на боці клієнта. Всі важкі обчислення виконуються на сервері, що робить її ідеальною для слабких пристроїв або повільних мереж.

Засіб дає можливість розробникам використовувати всю потужність платформи .NET на сервері, включаючи взаємодію із базами даних, файловими системами і іншими серверними ресурсами. Це робить технологію Blazor гарним вибором для розробки вагомих та складних веб-застосунків.

Модель Server також підтримує гаряче перезавантаження (Hot reload), що дає можливість розробникам бачити зміни в реальному часі без необхідності перезавантажувати сторінку. Це значно підвищує продуктивність розробки.

Проте технологія має і свої недоліки [27]. Оскільки всі взаємодії користувача обробляються на сервері, це може призвести до затримок, особливо при повільних з’єднаннях із мережею Інтернет. Також, якщо з’єднання з сервером втрачено, веб-застосунок перестане працювати.

Модель Server також підтримує можливість використання виключно серверного завантаження сторінки, що дає можливість веб-застосункам завантажуватися швидше, ніж це можливо при клієнтському завантаженні сторінки. Це особливо корисно для маркетингу, оскільки пошукові системи можуть легко та швидко індексувати вміст сторінки.

В цілому, технологія Blazor і модель Server є потужною і гнучкою комбінацією інструментів для розробки веб-застосунків. Це дає можливість розробникам використовувати відому мову C# і платформу .NET, забезпечуючи при цьому високу продуктивність і швидкість роботи.

2.2.5 Бібліотека Radzen

Бібліотека Radzen є високоякісною бібліотекою для створення користувацьких інтерфейсів, який дає можливість розробникам створювати інтерактивні веб-застосунки з використанням великої кількості візуальних елементів та вже існуючих стилів. Бібліотека використовує компонентний підхід,

що дає можливість розробникам створювати веб-застосунки з використанням готових компонентів, що значно спрощує процес розробки.

Проєкт бібліотеки Radzen був створений компанією Radzen, яка спеціалізується на розробці інструментів для веб-розробників. Ця бібліотека була створена з метою спростити розробку веб-застосунків, зробити її швидшою та ефективнішою.

Однією з ключових особливостей бібліотеки Radzen є велика кількість готових компонентів із зручними та детальними налаштуваннями.

Бібліотека використовує оптимізований рендеринг, що дає можливість створювати високопродуктивні та швидкі веб-застосунків. Вона підтримує двосторонню прив’язку даних до компонентів, що дає можливість автоматично синхронізувати дані між інтерфейсом користувача та моделями даних.

Бібліотека включає в себе такі компоненти, як-от: таблиці, форми, діаграми, календарі та багато інших.

Всі ці компоненти (рисунок 2.16) легко налаштовуються та можуть бути використані для створення різноманітних веб-застосунків.

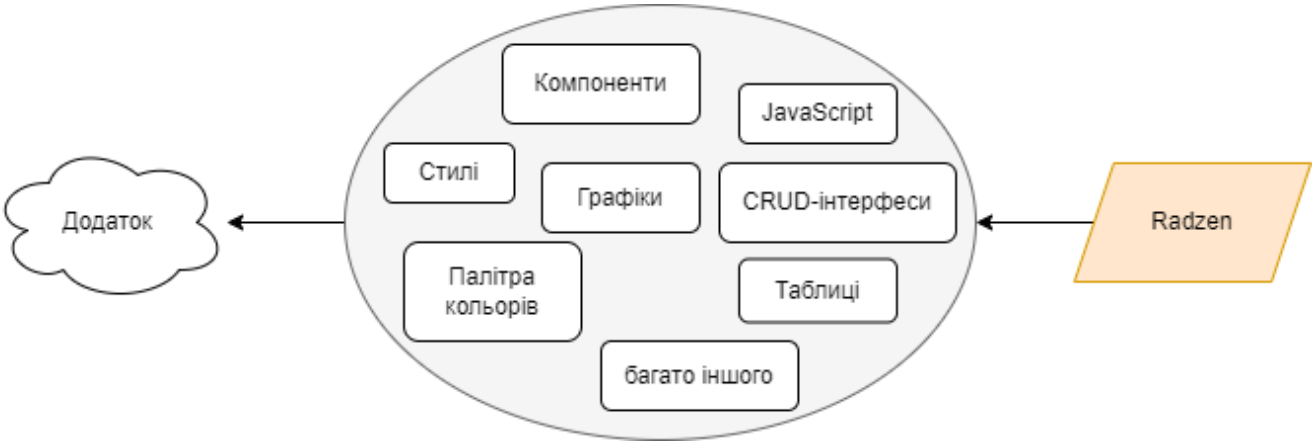


Рисунок 2.16 — Використання Radzen

Однією з переваг даного засобу є його підтримка технології Blazor та моделей Server і WebAssembly. Це означає, що бібліотека може використовуватися для створення як серверних, так і клієнтських веб-застосунків. Засіб підтримує розширення, що дає можливість розробникам додавати новий функціонал до своїх веб-застосунків.

Використовуючи бібліотеку Radzen, розробники можуть створювати веб-застосунки із адаптивними дизайном, які мають однаково коректний вигляд на будь-якому пристрої. Бібліотека включає в себе гнучку систему сітки (Gird), яка дає можливість легко створювати адаптивні макети під різні задачі.

Бібліотека також підтримує теми оформлення, які включають в себе налаштування палітри кольорів, налаштування анімацій, позицій елементів тощо. Розробники можуть вибрати з декількох вбудованих тем або створити свою власну тему для своїх веб-застосунків.

Всі ці особливості роблять бібліотеку потужним і гнучким інструментом для розробки клієнтських інтерфейсів веб-застосунків. Використовуючи бібліотеку, розробники можуть створювати високоякісні веб-застосунки швидко та ефективно.

2.2.6 Технологія PostgreSQL

Технологія PostgreSQL — одна з найпопулярніших систем керування базами даних з числа доступних на ринку, відома у світі завдяки своїм високим стандартам якості, надійності та стабільності [29].

Проєкт системи PostgreSQL був розроблений в 1986 році. Проєкт мав на меті розробити нові концепції керування базами даних. З того часу база даних PostgreSQL пройшла довгий шлях розвитку і вдосконалення.

Технологія PostgreSQL підтримує SQL, стандартну мову запитів для роботи з базами даних. Він також підтримує розширений SQL, що дає можливість використовувати більш складні запити та функції [30].

Вона може обробляти великі обсяги даних від невеликих баз даних до великих корпоративних систем. Технологія PostgreSQL також підтримує розподілені бази даних, що дає можливість розподіляти дані між кількома серверами для підвищення продуктивності та надійності.

Система має вбудовані механізми для запобігання втрати даних, включаючи транзакції з властивостями ACID, журналізацію, реплікацію та резервне копіювання. Ці функції забезпечують безперебійну роботу бази даних навіть у випадку відмови апаратного забезпечення або системи.

Наявна підтримка великої кількості типів даних, включаючи числові, текстові, булеві, дати/час, геометричні, бінарні та інші. Вона також дає можливість користувачам створювати свої власні типи даних. Це робить базу даних гнучким рішенням для різних задач обробки даних.

У базі даних є вбудована підтримка для концепції об’єктно-орієнтованого програмування, що дає можливість створювати складні типи даних і методи. Це робить систему керуваннями баз даних PostgreSQL потужним інструментом для розробки баз даних.

2.2.7 Бібліотека Entity Framework Core

Бібліотека Entity Framework Core (EF Core) — сучасна ORM (Object-Relational Mapping) бібліотека, розроблена компанією Microsoft. Вона є новітньою версією бібліотеки Entity Framework, яка була вперше представлена в 2008 році.

Вперше презентація повної версії бібліотеки EF Core була у 2016 році як частини платформи .NET Core.

Даний засіб надає можливість розробникам працювати з базами даних за допомогою .NET-об’єктів, а саме класів (рисунок 2.17). Це відбувається за допомогою технології LINQ (Language Integrated Query), що дає можливість виконувати запити до бази даних безпосередньо з програми із мовою програмування C# або Visual Basic.

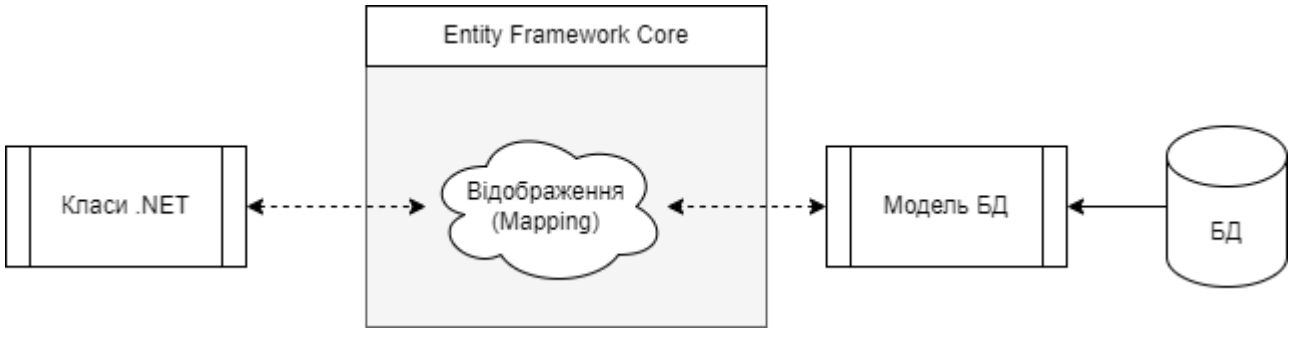


Рисунок 2.17 — Принцип роботи бібліотеки Entity Framework Core

Однією з ключових особливостей бібліотеки EF Core є її здатність підтримувати Code First — це підхід, який дає можливість розробникам створювати бази даних, використовуючи POCO (Plain Old CLR Objects) класи. Це

означає, що розробники можуть створювати моделі даних безпосередньо в коді, а бібліотека EF Core потім згенерує відповідні таблиці в базі даних.

Підхід Database First також присутній серед функціоналу фреймворку. Головна особливість цього підходу полягає у тому, що спочатку проєктується та створюється модель бази даних, а потім бібліотека генерує відповідні моделі даних. Це може бути корисним, коли розробники працюють з великими, складними базами даних, які вже існують.

Бібліотека включає в себе вбудовану підтримку міграцій баз даних, що надає можливість без проблем вносити зміни в структуру бази даних, при цьому не втрачаючи жодних даних. Це дуже зручно, оскільки не потребує додаткових зусиль для збереження інформації. Міграції для оновлення моделі бази даних можливо генерувати автоматично за допомогою бібліотеки. Міграція базується на змінах, які були внесені в моделі об’єктів .NET (рисунок 2.18). Це значно спрощує процес розробки, оскільки автоматизує одну з найбільш трудомістких частин роботи.

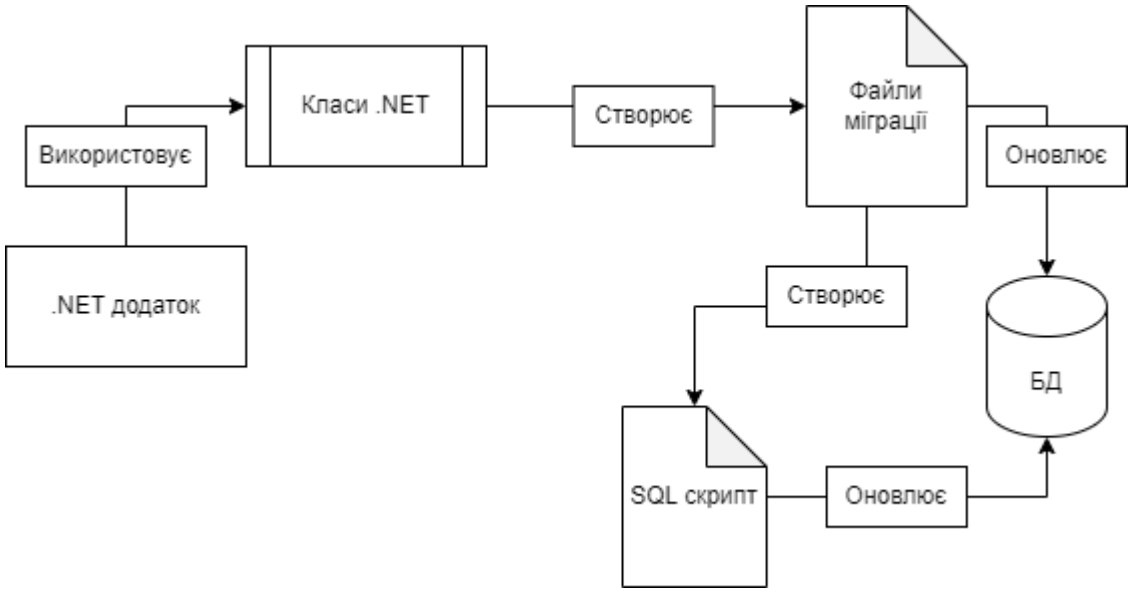


Рисунок 2.18 — Принцип роботи міграцій у бібліотеці Entity Framework Core

Типові шаблони проєктування Unit of Work та Repository закладені у базисах фремйворку, що дає можливість розробникам проєктувати, тестувати та легко маніпулювати з даними та архітектурними рішеннями. Ці шаблони

допомагають ізолювати бізнес-логіку від доступу до даних, що полегшує процес тестування та підтримки коду.

Даний засіб підтримує багато різних баз даних, включаючи SQL Server, SQLite, MySQL, PostgreSQL тощо. Бібліотека підтримує виконання звичайних SQL запитів — це робить бібліотеку гнучким рішенням для багатьох різних сценаріїв та може бути корисним для виконання складних операцій, які не можуть бути виконані за допомогою LINQ.

Оскільки, бібліотека EF Core є частиною платформи .NET Core, то це означає, що вона може бути використана на різних операційних системах, включаючи Windows, Linux і macOS. Це робить його гарним вибором для розробки крос-платформних застосунків.

Бібліотека має відкритий вихідний код, що означає те, що розробники можуть вільно використовувати, модифікувати та розповсюджувати його відповідно до своїх потреб. Крім того, він має велику спільноту розробників, яка постійно працює над модернізацією, вирішенням помилок та введенням сучасних функцій.

Загалом, бібліотека EF Core є потужною, гнучкою та легкою у використанні ORM, яка дає можливість розробникам працювати з базами даних на високому рівні абстракції. Вона має багато вбудованих функцій, що полегшують розробку застосунків, і підтримує широкий спектр баз даних, що робить її сучасним вибором для будь-якого проєкту.

2.2.8 Бібліотека NUnit

Бібліотека NUnit — засіб для модульного тестування у платформі .NET. Бібліотека була розроблена для підтримки тестування, використовуючи атрибути платформи .NET, що дає можливість легко писати код для тестування і контролювати його виконання. Бібліотека використовується для написання всіх видів тестів: модульних, інтеграційних, системних та інших. Засіб вперше був випущений у 2002 році, і з того часу він постійно оновлюється і вдосконалюється [37].

Вона відрізняється своєю гнучкістю, яка є однією з її ключових особливостей. Дана бібліотека надає користувачам можливість використовувати різноманітні види перевірок для того, щоб пересвідчитися в тому, що існуючий код працює відповідно до очікувань. В рамках цього засобу розробники мають можливість використовувати як прості перевірки, такі як AreEqual, так і більш складні, наприклад, That. Така гнучкість дає можливість адаптувати процес тестування під конкретні потреби і завдання, що робить бібліотеку NUnit незамінним інструментом для розробників.

Бібліотека NUnit, як інструмент для автоматизованого тестування, має вбудовану підтримку налаштування тестів. Це означає, що ви маєте можливість використовувати спеціальні атрибути, такі як SetUp і TearDown. Вони дають можливість виконати певний код перед початком кожного тесту і після його завершення відповідно. Ця функція може бути надзвичайно корисною, коли вам потрібно ініціювати або очистити ресурси, які використовуються в тестах.

Крім того, засіб має вбудовану підтримку паралельного виконання тестів. Це означає, що він може виконувати декілька тестів одночасно, що може значно скоротити час виконання тестових наборів (рисунок 2.19).

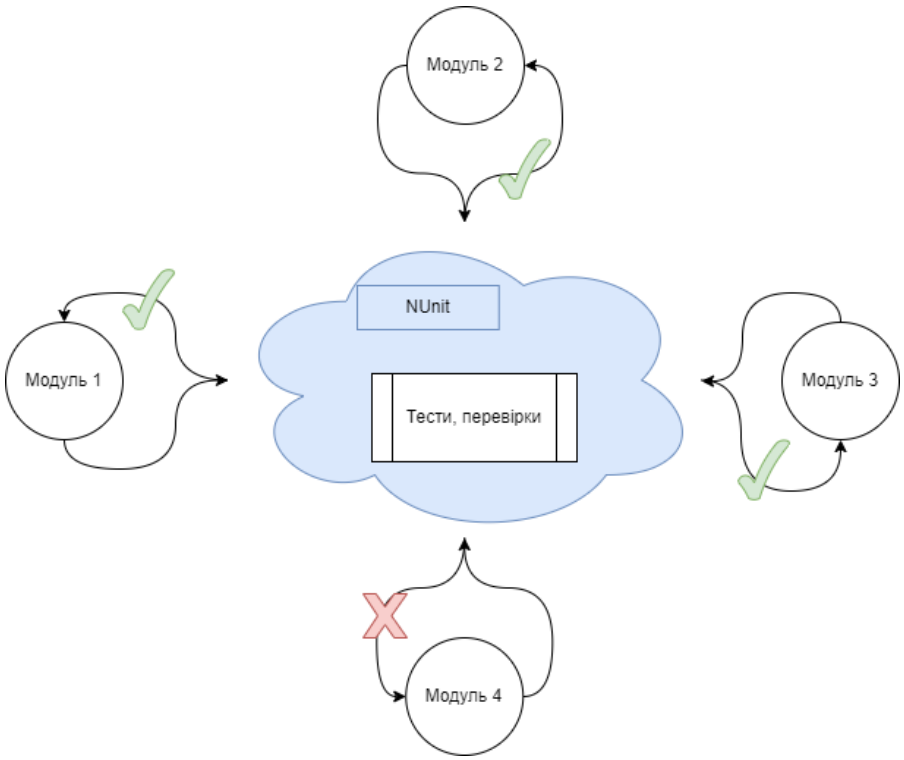


Рисунок 2.19 — Паралельне модульне тестування

Ця функція може бути особливо корисною, коли ви працюєте з великими тестовими наборами або коли вам потрібно швидко отримати результати тестування.

Однією з переваг бібліотеки є її широка сумісність з різними версіями платформи .NET. Вона підтримує всі версії платформи .NET від 2.0 і вище. Він працює з різними сучасними середовищами розробки

Бібліотека має вбудовану підтримку функціоналу генерації звітів про виконані тести. Звіти можуть бути представлені в різних форматах, включаючи формати XML і HTML. Це дає можливість легко аналізувати і ділитися результатами тестів із іншими членами команди розробників.

Засіб має налаштування для категорій тестів. Достатньо використовувати атрибут Category для групування схожих тестів. Це дає можливість виконувати лише певні групи тестів, що може бути корисним, коли виконується розробка великих програмних продуктів.

Наявність відкритого вихідного коду означає, що присутня можливість вільного використання, зміни та розповсюдження фреймворку. Він має активну спільноту користувачів та розробників, які доповнюють та вдосконалюють код фреймворку.

У підсумку, бібліотека NUnit є потужним і гнучким фреймворком для модульного тестування на платформі .NET. Він має багато важливих особливостей, що дозволяють легко та швидко створювати та виконувати тести. Його широка сумісність та активні оновлення роблять його корисним для будь-якого проєкту на платформі .NET.

Висновки до розділу 2

Теоретичні основи та засоби розробки — це дві основні частини розділу, які включають в себе розгляд багатьох понять про машинне навчання, методи та види оптимізації алгоритмів та засобів, використаних для створення програмного продукту. Було обґрунтовано вибір платформи .NET та технології Blazor для реалізації веб-застосунку, а бібліотеку ML.NET для всіх взаємодій із машинним навчанням.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ОПТИМІЗАЦІЇ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

Реалізація починається з проєктування і закінчується доставкою функціоналу готового програмного продукту до користувачів.

Проєктування має на меті заздалегідь визначити потрібний час і ключові аспекти створення програмного продукту, включаючи сценарії використання системи, використання бази даних, різні діаграми та архітектуру програми.

3.1 Проєктування програмного продукту

Вирішено спочатку провести аналіз потенційних сценаріїв використання продукту, а також всіх можливих користувачів системи. Це допоможе краще зрозуміти, який функціонал та задачі може виконувати програмний продукт. Він повинен включати в себе весь функціонал, який було визначено в завданні для магістерської дисертації, а саме:

— можливість використання власних наборів даних;

— практичне застосування машинного навчання (наприклад, прогнозування);

— візуалізація результатів і метрик;

— можливість порівняння моделей;

— експортування імпортування даних

— детальна конфігурація моделей

— автоматичне створення найкращих моделей

Крім основного функціоналу, в процесі проєктування додалися кілька додаткових функцій, які можуть виявитися корисними для користувачів і розробників.

Перший крок — це створення діаграми прецедентів (рисунок 3.1). Цей елемент є одним з найважливіших у процесі проєктування програмного продукту, оскільки він допомагає ефективно і структуровано планувати та розробляти проєкт.

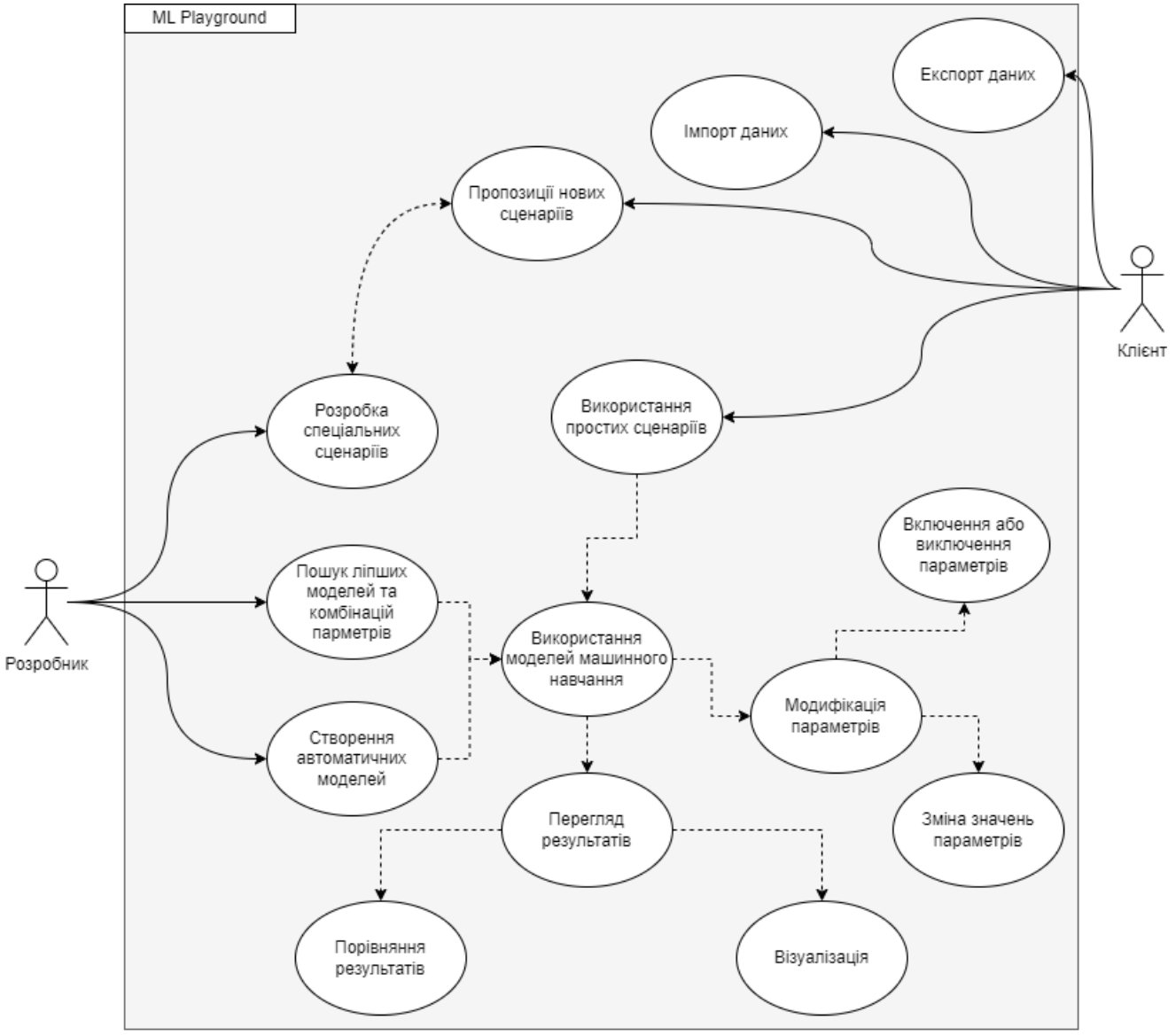


Рисунок 3.1 — Діаграма програмного продукту

Діаграма прецедентів виконує такі завдання у плануванні програмного продукту:

— ідентифікація функціональних вимог — діаграма дає можливість визначити основні функціональні вимоги системи на ранніх етапах розробки. Вона допомагає розуміти, як користувачі будуть використовувати систему та які функції вони очікують;

— комунікація із зацікавленими сторонами — може бути ефективним інструментом для комунікації між розробниками, менеджерами, тестувальниками та іншими зацікавленими сторонами. Вона дає можливість представити важливі аспекти взаємодії системи з оточенням без зайвих технічних деталей;

— визначення ролей та відповідальностей — допомагає визначити ролі різних елементів системи, таких як користувачі чи інші системи. Вона визначає, які взаємодії можуть мати ці ролі із системою та одне з одним;

— планування тестування — на основі діаграми використання можна розробити тести, які перевірять, чи виконує система очікувані функції;

— орієнтація на користувача — може допомогти команді розробників уникнути технічного жаргону та сконцентруватися на потребах та очікуваннях користувачів;

— визначення границь системи — дає можливість чітко визначити, як система взаємодіє із зовнішнім середовищем, визначаючи границі системи та обсяг її відповідальності.

Після фінальної постановки задач та вимог, затверджено наступні кроки розробки, серед яких визначення часових термінів реалізації, після цього обирались технології та засоби розробки, проєктування архітектури та інші кроки проєктування.

3.2 Архітектура програмного продукту

Для проєктування вибрано клієнт-серверну архітектуру, яка є типовою в розподілених системах, де функції системи розподілені між окремими комп’ютерами чи пристроями, відомими як клієнти та сервери. Цей підхід дає можливість ефективно керувати ресурсами, розділяти завдання та покращувати масштабованість системи.

Основні компоненти клієнт-серверної архітектури включають:

— клієнти — це пристрої або програми, які використовують ресурси та послуги, надані серверами. Клієнти можуть бути реалізовані у вигляді апаратного обладнання, програмних застосувань або їхніх комбінації;

— сервери — це пристрої або програми, які забезпечують ресурси, послуги або дані для клієнтів. Сервери можуть мати спеціалізовану функціональність, як-от: бази даних, веб-сервери, файлові сервери тощо;

— протоколи. Для взаємодії між клієнтами та серверами використовуються певні комунікаційні протоколи, такі як-от: HTTP (для веб-застосунків), TCP/IP, інші мережеві протоколи;

— розподіл функцій. Сервери зазвичай виконують функції, які пов’язані з обробкою та збереженням даних, виконанням обчислень чи наданням інших послуг. Клієнти зазвичай відповідають за взаємодію з користувачами та ініціацію запитів;

— вертикальна та горизонтальна масштабованість. Системи можуть бути масштабовані вертикально (збільшення ресурсів на окремих серверах) або горизонтально (додавання нових серверів).

Архітектура, зображена на рисунку 3.2, відкриває широкі можливості для створення ефективного розподіленого середовища, де різні компоненти системи можуть працювати паралельно та взаємодіяти між собою для досягнення необхідної функціональності.

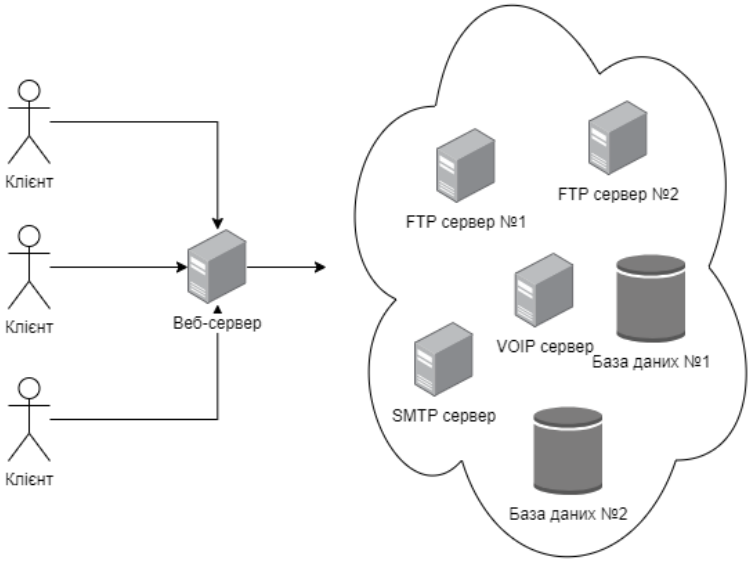


Рисунок 3.2 — Приклад використання розподіленої клієнт-серверної архітектури із використанням одного веб-сервера

Ця архітектура дає можливість розподілити завдання між різними вузлами системи, що надає можливість досягти більшої швидкодії та ефективності обробки даних. Кожен вузол може виконувати свою частину роботи незалежно від інших, що дає можливість збільшити продуктивність системи в цілому.

Далі вирішено структурувати код застосунку на три рівні (рисунок 3.3):

— веб-застосунок (Blazor Server);

— ядро (бізнес-логіка);

— тести (NUnit).

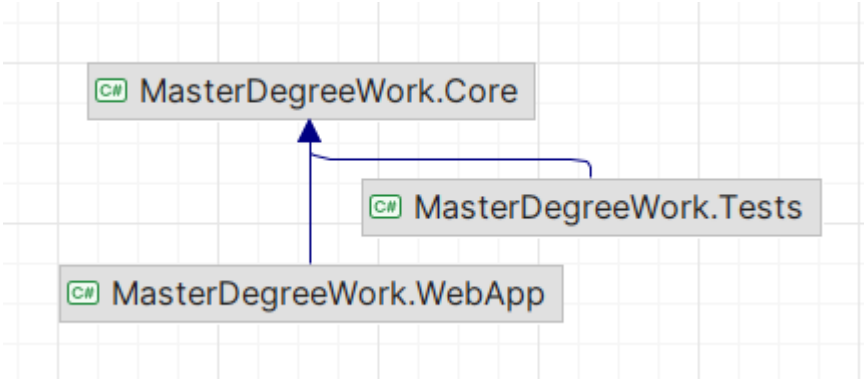


Рисунок 3.3 — Розбиття коду веб-застосунку на три рівні

Різноманітність рівнів у структурі програмного продукту, включаючи веб-застосунок, ядро (бізнес-логіка) та тести, є стратегічним рішенням, спрямованим на покращення організації та керування кодовою базою. Розбиття на такі рівні надає команді можливість працювати над проєктом більш системно та ефективно.

Перший рівень, веб-застосунок, фокусується на презентації та взаємодії з користувачем. Всі аспекти, пов’язані з відображенням інтерфейсу, обробкою запитів від користувачів і взаємодією з веб-сервером, концентруються на цьому рівні.

Другий рівень, ядро, або бізнес-логіка, визначає основні аспекти функціональності та обробки даних. Тут розміщена центральна логіка програмного продукту, яка відповідає за обчислення, маніпулювання даними та взаємодію з різними компонентами системи.

Третій рівень, тести, відіграє важливу роль у забезпеченні якості коду та функціональності. Тести дозволяють перевірити, чи працює код правильно та якісно, допомагаючи у виявленні та усуненні можливих помилок на ранніх етапах розробки.

Вирішення поділу коду на такі рівні також зменшує дублювання коду. Якщо фрагмент коду відповідає за певну функціональність, його можна легше використовувати та підтримувати у разі змін. Це забезпечує структурованість та легшу розширюваність проєкту.

Загалом, такий підхід дає можливість ефективніше працювати, розподіляючи завдання між різними рівнями відповідно до їхнього призначення. Це поліпшує легкість читання, розуміння коду новими розробниками та розширюваність кодової бази, що є критичними аспектами успішного розвитку та керування програмним продуктом.

3.3 Використання Blazor Server

Використання моделі Server для технології Blazor у програмному продукті надає кілька значущих переваг та функціональних можливостей:

— оновлення у режимі реального часу — Blazor Server взаємодіє з клієнтом за допомогою сигналізації (SignalR), що дає можливість отримувати миттєві оновлення на стороні клієнта без необхідності перезавантаження сторінки;

— компактність пересилання даних — Blazor Server передає тільки зміни між сервером і клієнтом, а не всю HTML-розмітку або JavaScript код. Це дає можливість зменшити обсяг пересилання даних та прискорити завантаження сторінок;

— спільне використання коду — спільно використовується код між клієнтською та серверною частиною, що спрощує розробку і зменшує кількість дубльованого коду. Це полегшує підтримку та оновлення програмного продукту;

— клієнт-сервер — Blazor Server використовує архітектурний підхід, де більша частина логіки застосунку виконується на сервері. Коли користувач взаємодіє із сторінкою, відбувається взаємодія з сервером за допомогою SignalR, що дає можливість передавати події та дані між клієнтом та сервером в режимі реального часу.

У програмному продукті є система сторінок. Створено базові сторінки, а саме: «RegressionPage.razor», «SpecialCase.razor», «Index.razor» тощо. Кожна

сторінка використовує певні компоненти із вже згаданого фреймворку користувацьких інтерфейсів Radzen.

Реалізовано інтерактивні таблиці включно із функціоналом фільтрації, сортування, навігацією за сторінками та іншими можливостями. Також у застосунку підтримується функціонал діалогових вікон, у яких реалізовано фунціонал взаємодії із даними.

Також система компонентів дає можливість створювати шаблони сторінок (Layout), котрі дублюють стилізацію, позиції деяких елементів на всіх сторінках, де є встановлено шаблон.

Робота з сервером реалізована з використанням асинхронних запитів, що дає можливість ефективно обробляти великий обсяг даних без блокування користувальницького інтерфейсу. Це сприяє плавності та продуктивності роботи застосунку навіть при роботі з великими обсягами інформації.

Однією з ключових особливостей є також можливість розширення функціоналу застосунку за допомогою власних компонентів та модулів. Це дає можливість легко адаптувати програмний продукт під конкретні потреби користувачів та впроваджувати нові функції, не змінюючи основний код.

3.4 Використання бази даних

Потреба використовувати базу даних у застосунку — мінімальна. Оскільки застосунок повинен демонструвати ефективність оптимізації машинного навчання, не потрібно зберігати надскладні структури даних у базі даних. Таким чином, програмний продукт має наступні переваги:

— швидкість і легкість;

— низькі витрати на серверну інфраструктуру;

— елементарне розгортання застосунку в будь-якому середовищі;

— легка масштабованість.

Але, все ж таки використання бази даних присутнє. База даних використовується для спеціальних сценаріїв, де використовуються складні структури даних, наприклад, для класифікації. Наприклад, для прогнозування

динаміки історичних даних із значеннями у базі даних присутня елементарна структура даних яка містить значення та дату (рисунок 3.4).

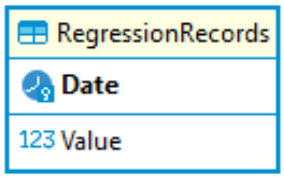


Рисунок 3.4 — Модель таблиці даних для прогнозування за допомогою регресії

Це надає можливість дуже швидко обробляти дані та уникати колізій, наприклад, дублікатів даних. Дата виступає ключем у цій структурі, а значення має тип числа з плаваючою точкою (float).

Для складних сценаріїв, які виконуються розробниками по запиту клієнтів, створюються наприклад, як для вже наявного розробленого сценарію прогнозування небезпечних ділянок доріг, так і більш складна структура (рисунок 3.5). Вона зберігає запис про ДТП, яке відбулось.

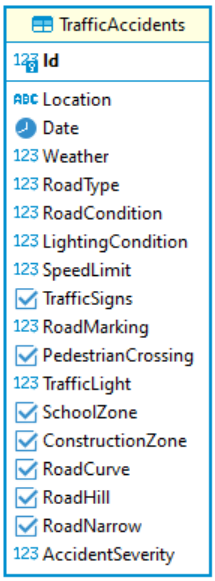


Рисунок 3.5 — Модель даних запису про ДТП

У структурі можна побачити наявність багатьох індикаторів та параметрів, як-от: обмеження швидкості, стан дороги, дорожню дугу, пішохідний перехід тощо.

За допомогою цих даних, можливо класифікувати рівень небезпеки на ділянці дороги за певним запитом, наприклад: розбита дорога, без знаків, без обмеження швидкості тощо.

3.5 Використання Entity Framework Core

Бібліотека Entity Framework Core використовується для взаємодії із базою даних, швидкою фільтрацією та відображенням моделей баз даних до класів. У проєкті веб-застосунку створено «EntityService» — сервіс, який виконує всі взаємодії із записами у коді мовою C#. Створення, редагування, видалення, вибірка, фільтрація — всі реалізовані у даному сервісі.

Він дає можливість швидко фільтрувати дані та відображати моделі баз даних у відповідні класи. Це значно спрощує роботу з даними і робить процес більш ефективним.

Це надає можливість забезпечити більшу гнучкість та контроль над процесом взаємодії з базою даних. Таким чином, «EntityService» стає важливим компонентом в структурі веб-застосунку, що сприяє ефективній роботі з даними.

Бібліотека Blazor, як відомо, не підтримує архітектуру клієнт-сервера, яка базується на HTTP-запитах. Замість цього, він використовує обмін даними через WebSocket. Це означає, що кожен клієнт підтримує постійне з’єднання з сервером протягом усієї сесії.

Ця особливість роботи бібліотеки Blazor вимагає створення нового «DbContext» (контексту бази даних) в межах бібліотеки Entity Framework Core для кожного клієнта при кожному запиті, як це показано на рисунку 3.6. Основна причина цього полягає в унікальності кожної сесії та стану бази даних при кожному запиті.

Таким чином, кожен запит та сесія, представлена клієнтом, може взаємодіяти з базою даних у своєму унікальному контексті. Це надає можливість забезпечити індивідуальний підхід до обробки даних для кожного користувача, що підвищує ефективність та гнучкість системи.

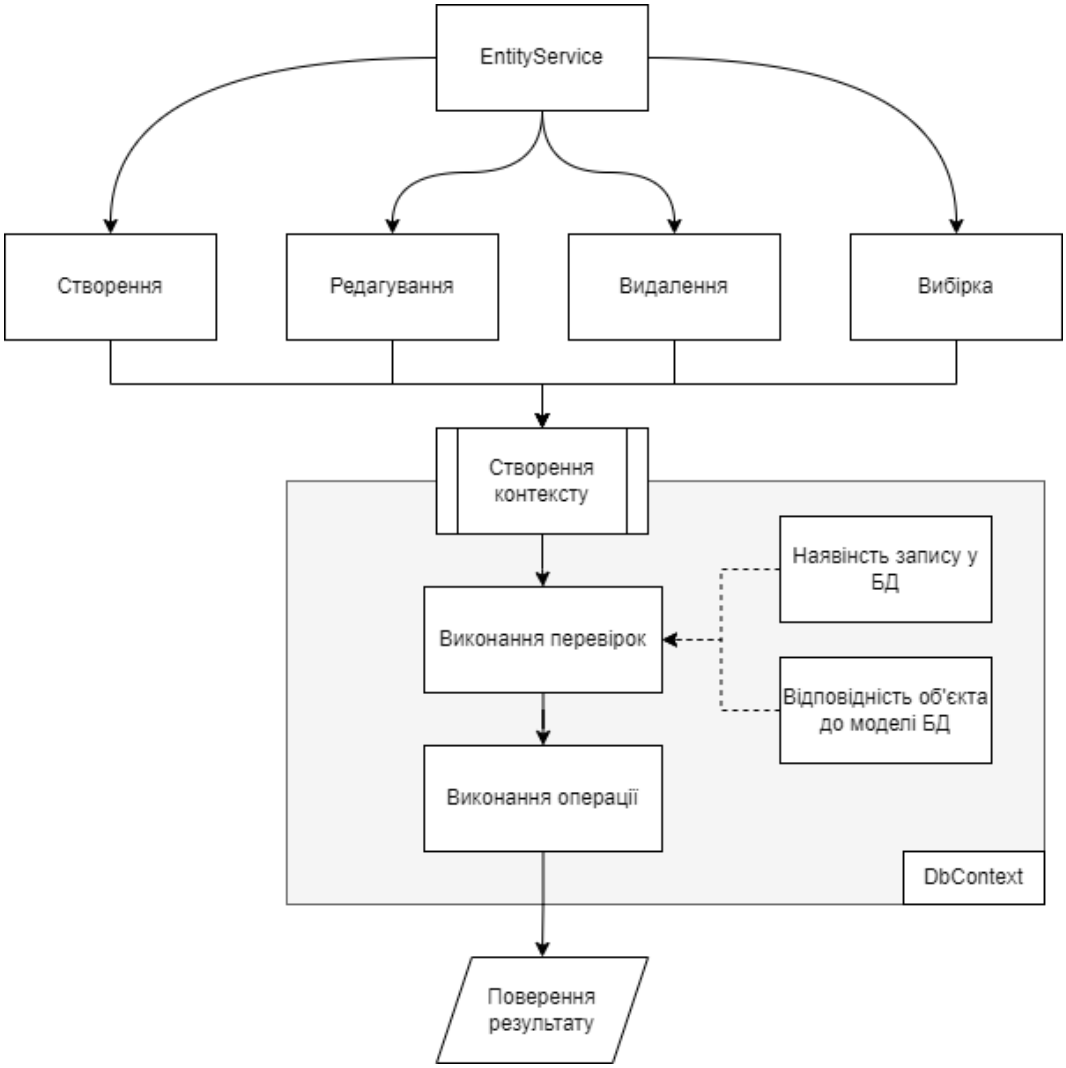


Рисунок 3.6 — Функціонал EntityService

Коли клієнт взаємодіє з застосунком і виконує операції в базі даних, важливо мати окремий «DbContex»t, оскільки контекст бази даних є однією з унікальних властивостей сесії. Використання окремого контексту для кожного клієнта дає можливість уникнути конфліктів та забезпечити ізоляцію даних між різними користувачами та компонентами.

Крім того, створення нового «DbContext» спрощує керування життєвим циклом контексту та забезпечує те, що він не буде утримувати занадто багато даних або надто довго існувати, зменшуючи тим самим можливість витоку ресурсів.

Таким чином, у межах Blazor Server та WebSocket створення нового «DbContext» для кожного клієнта допомагає забезпечити ефективне та ізольоване керування базою даних для кожної сесії.

3.6 Використання ML.NET

Бібліотека ML.NET використовується у проєкті для розв’язання задач машинного навчання, як-от: регресія, класифікація, кластеризація тощо.

Використання надає можливість легко маніпулювати даними, використовувати велику кількість моделей та параметрів, виконувати оцінювання моделі, проводити оптимізацію моделі, та багато інших функцій.

Створено сервіс «MLService», який виконує всі операції, пов’язані з машинним навчанням у системі. Сервіс приймає такі вхідні параметри: модель, параметри, значення параметрів, та набір даних. Він виконує певну послідовність операцій для отримання результату (рисунок 3.7). У сервісі присутній функціонал відстеження прогресу, який забезпечує відображення поточної операції.

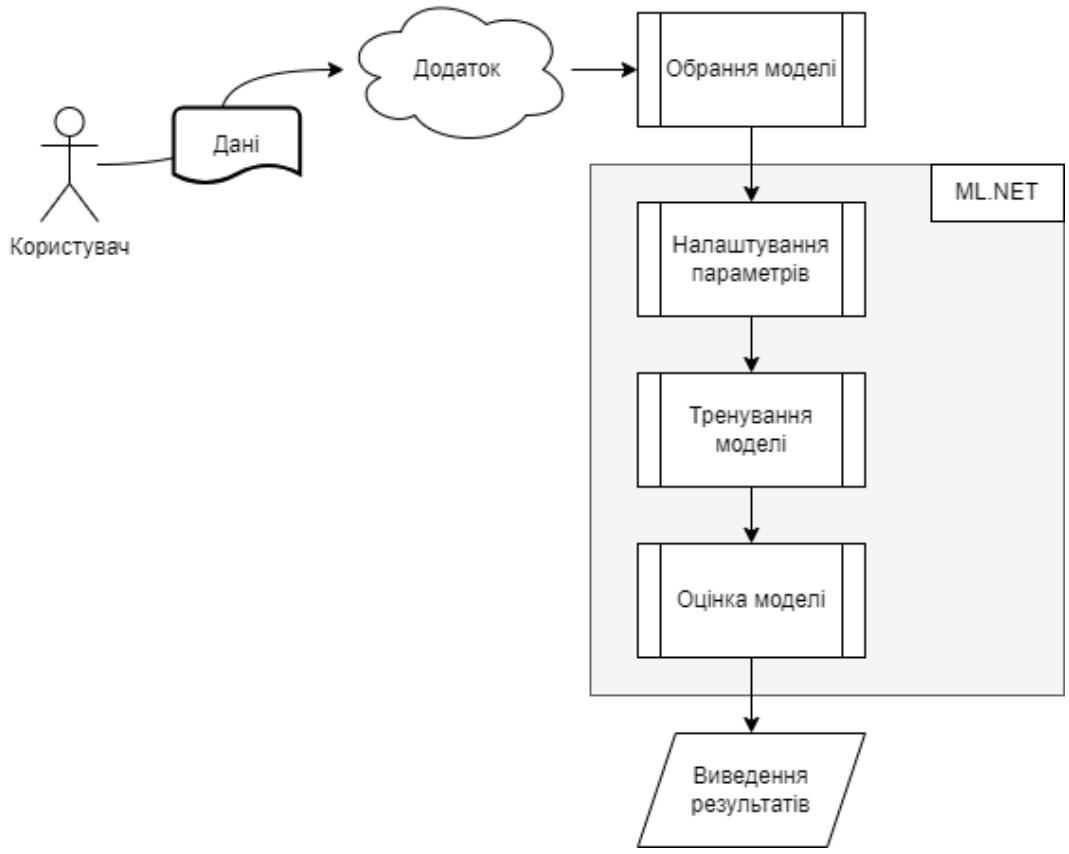


Рисунок 3.7 — Використання функціоналу бібліотеки ML.NET

Одна з основних операцій — це створення конвеєра ознак. Конвеєр ознак дає можливість визначити, які ознаки будуть використовуватися для навчання моделі.

Після створення конвеєра ознак, сервіс використовує надані дані. Це можуть бути дані з бази даних, CSV-файлів або будь-якого іншого джерела. Після завантаження даних, сервіс розділяє їх на тестову та навчальну частини. Тестова частина даних використовується для оцінки точності моделі після навчання.

Після розділення даних, виконується створення моделі. ML.NET надає багато вбудованих моделей для різних задач машинного навчання. Наприклад, для задачі класифікації — модель LightGbm, а для задачі регресії — модель FastForest. Також ML.NET має фунціонал створення AutoML моделей, які обирають найліпші з списку автоматично згенерованих за оцінками та із заданим обмеженням в часі на тренування.

Після створення моделі, сервіс виконує тренування моделі на навчальних даних. Тренування моделі полягає в тому, щоб модель «вивчила» залежності між ознаками та вихідними значеннями. Це може займати тривалий час, особливо, якщо використовувати великі обсяги даних.

Після тренування моделі, виконується оцінка точності за допомогою тестових даних. Оцінка моделі дає можливість зрозуміти, наскільки добре модель працює на нових даних. Це допомагає визначити, чи потрібно вносити зміни в модель або в набір даних.

3.7 Налаштування неперервної інтеграції і доставки

Налаштування неперервної інтеграції і доставки (CI/CD) є ключовим етапом у розробці програмного забезпечення, спрямованим на автоматизацію процесів розгортання та тестування. Це стратегічне рішення дає можливість швидко та безпечно впроваджувати новий код у виробниче середовище, забезпечуючи надійність та стабільність програмного продукту. Використано YAML pipelines — спосіб створення конвеєрів неперервної інтеграції/доставки.

На початковій стадії процесу налаштування основним завданням є налаштування автоматизованого тестування коду, як це зображено на рисунку 3.8. Цей процес передбачає створення спеціальних автоматичних тестів. Ці тести запускаються кожного разу, коли відбувається нове злиття коду. Головною метою

цих тестів є перевірка того, що внесені зміни не призводять до порушення вже існуючого функціоналу. Таким чином, автоматизоване тестування допомагає забезпечити стабільність та надійність роботи програмного коду після внесення будь-яких змін.

```
- job: TestJob
  dependsOn: BuildJob
  condition: eq(variables['Build.Reason'], 'Manual')
  steps:
  - task: VSTest@3
    inputs:
      testSelector: 'testAssemblies'
      testAssemblyVer2: |
        **\*test*.dll
        !**\*TestAdapter.dll
        !**\obj\**
      searchFolder: '$(System.DefaultWorkingDirectory)'
```

Рисунок 3.8 — YAML задача для виконання тестів

Після успішного завершення процесу інтеграції, наступний етап конвеєра — це компіляція застосунку, яка включає ряд важливих завдань для забезпечення правильної компіляції та готовності коду для розгортання (рисунок 3.9).

```
- job: 'Build'
        steps:
          - task: UseDotNet@2
            inputs:
              packageType: 'sdk'
              version: '7.x'
          - task: DotNetCoreCLI@2
            name: "Restore"
            inputs:
              command: 'restore'
              projects: '**/MasterDegreeWork.WebApp.csproj'
          - task: DotNetCoreCLI@2
            name: "Build"
            inputs:
              command: 'build'
              projects: '**/MasterDegreeWork.WebApp.csproj'
              arguments: '--configuration $(BuildConfiguration)'
          - task: DotNetCoreCLI@2
            name: "Publish"
            inputs:
              command: 'publish'
              publishWebProjects: false
              projects: '**/MasterDegreeWork.WebApp.csproj'
              modifyOutputPath: true
              zipAfterPublish: true
              nobuild: true
          - task: PublishBuildArtifacts@1
            displayName: 'Publish Artifact'
            inputs:
              PathtoPublish: '$(Build.ArtifactStagingDirectory)'
              ArtifactName: 'webApp'
```

Рисунок 3.9 — Задача для компіляції та публікації програми у форматі YAML

Одним із ключових аспектів цієї задачі є визначення та використання необхідної версії платформи .NET. Це гарантує, що застосунок буде взаємодіяти з необхідними бібліотеками та середовищем виконання.

Додатково, етап компіляції включає відновлення пакетів NuGet, яке є критичним для забезпечення наявності та актуальності всіх необхідних бібліотек. Це забезпечує, що проєкт буде коректно використовувати встановлені бібліотеки та їхні версії, що є важливим для уникнення конфліктів та забезпечення стабільної роботи програмного продукту.

Крім того, етап компіляції передбачає публікацію артефактів компіляції, що включає в себе усі необхідні файли та ресурси, готові для розгортання. Ці артефакти можуть бути використані наступними етапами процесу доставки та розгортання, забезпечуючи їхню доступність та структурованість.

Останній пункт задачі — завантаження файлів. Це передбачає перенесення зібраного коду та інших необхідних ресурсів на сервери або інші визначені місця, де застосунок буде готовий до розгортання та використання. Цей крок має велике значення для забезпечення доступності та швидкості розгортання виробленого продукту.

Таким чином, етап компіляції в конвеєрі неперервної інтеграції і доставки визначається рядом строго скоординованих завдань, які гарантують готовність програмного продукту до подальших етапів розробки та впровадження.

Висновки до розділу 3

Розглянуто всі необхідні сценарії, створено діаграми та детальний план роботи для створення програмного продукту. Розроблено комплекс програмного забезпечення, який включає у себе:

- веб-застосунок;
- сервіси взаємодії з базою даних і машинним навчанням;
- налаштування неперервної інтеграції/доставки;
- виконано тестування всіх наявних частин системи.

4. ПОДАННЯ СЦЕНАРІЇВ І ДЕМОНСТРАЦІЯ ФУНКЦІОНАЛУ

Існує велика кількість варіантів використання розробленої системи. Платформу можна використовувати для розв’язання різноманітних завдань у багатьох галузях. Одним із можливих напрямків є застосування системи для аналізу у військовій сфері або медицині. Вона може допомагати у ранньому виявленні ризиків, прогнозуванні результатів та вдосконаленні методів діагностики.

4.1 Вимоги для розгортання програмного продукту

Програмний продукт, який використовує програмні засоби так Blazor Server, PostgreSQL, ML.NET та EF Core, вимагає специфічних параметрів обчислювальної техніки і середовища розгортання для оптимальної продуктивності й ефективного використання ресурсів, а саме:

— процесор — мінімум чотириядерний процесор з можливістю багатозадачності для обробки паралельних запитань;

— оперативна пам’ять — не менше 8 гігабайт для швидкої обробки великої кількості даних та моделей машинного навчання;

— операційна система — сучасні версії операційних систем: Windows 10, 11 або Linux 20, 21;

— додатково встановлене середовище виконання .NET 7 Runtime.

— веб-сервер — наявність та налаштування високопродуктивного веб-сервера для роботи з технологією Blazor Server;

— швидке та стійке інтернет-з’єднання — для забезпечення безперервної роботи веб-застосунку;

— забезпечення наявності необхідних мережевих налаштувань — для забезпечення взаємодії між компонентами застосунку та базою даних;

— включення систем моніторингу та логування — для вчасного виявлення помилок та оптимізації продуктивності системи.

Якщо врахувати вказані вище вимоги, то програмний продукт забезпечить використання всіх функцій.

4.2 Сценарії використання програмного забезпечення

Далі подано два сценарії використання програмного продукту, а саме сценарії для задач мультикласової класифікації та регресії.

4.2.1 Класифікація небезпек на ділянці дороги

Класифікація небезпеки на ділянці дороги є важливим аспектом у попередженні дорожньо-транспортних пригод (ДТП). Застосування машинного навчання в цьому контексті може значно покращити точність та ефективність такої класифікації.

Машинне навчання має потенціал використовувати історичні дані про аварії на дорогах для виявлення певних закономірностей та тенденцій. Ці закономірності та тенденції можуть слугувати показниками потенційної небезпеки на конкретних ділянках дороги.

Дані, які використовуються для цього аналізу, можуть бути дуже різноманітними. Вони можуть включати інформацію про стан дорожнього покриття, обмеження швидкості на певній ділянці, наявність дорожніх знаків та освітлення.

Крім того, ці дані можуть включати інформацію про погодні умови, які можуть впливати на безпеку дорожнього руху. Це можуть бути дані про температуру, вологість, опади, видимість та інші погодні фактори.

Таким чином, машинне навчання може допомогти виявити потенційно небезпечні ділянки дороги, аналізуючи великі обсяги даних про різні фактори, які можуть впливати на безпеку дорожнього руху.

Наприклад, якщо ділянка дороги має поганий технічний стан, немає обмеження швидкості та відсутні дорожні знаки, алгоритми машинного навчання можуть визначити цю ділянку як потенційно небезпечну. Це дозволить вжити необхідних заходів для попередження можливих ДТП, як-от встановлення дорожніх знаків або обмеження швидкості.

Мультикласова класифікація дає можливість виконувати прогнозування де є більше, ніж два можливих варіанти класифікації (рисунок 4.1). Це означає, що система може класифікувати дані в один з кількох класів, замість того, щоб обмежуватися лише двома варіантами.

Цей вид класифікації особливо корисний у випадках, коли існує багато різних можливих результатів або коли дані можуть належати до більш ніж одного класу одночасно.

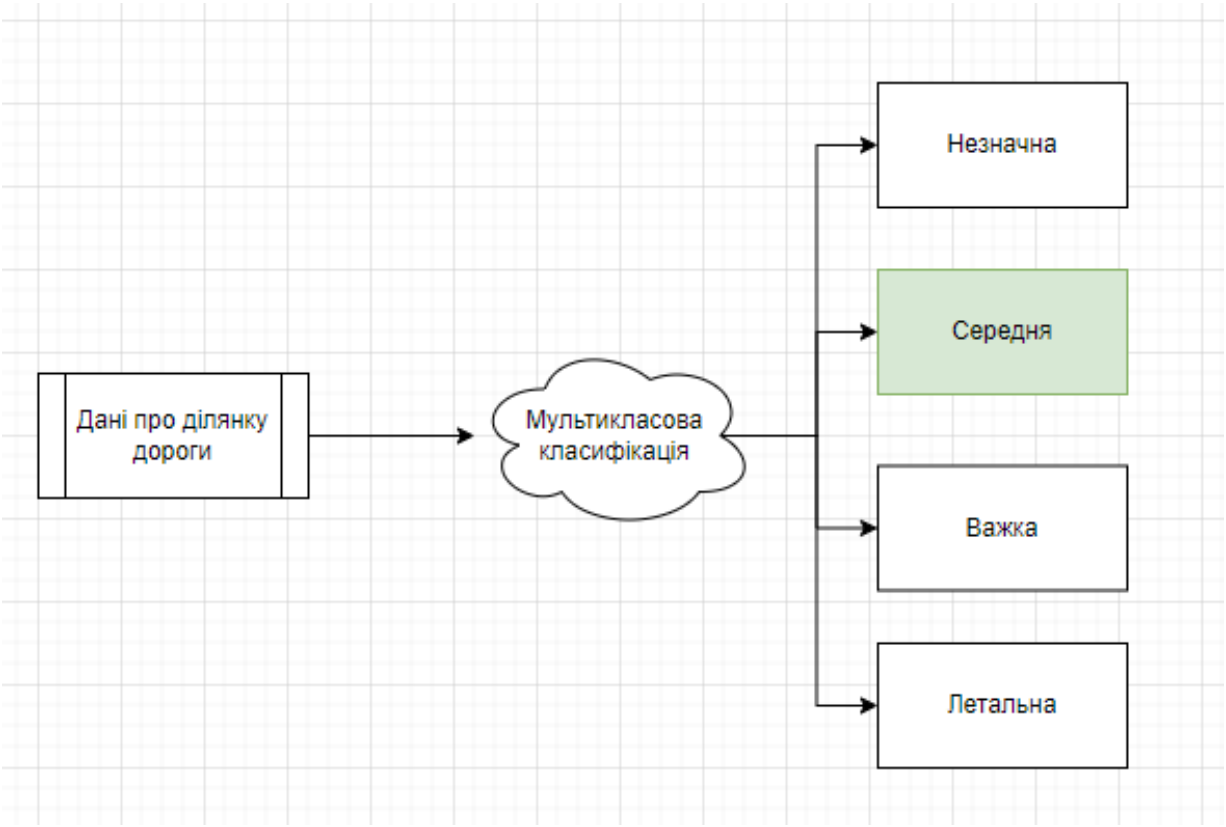


Рисунок 4.1 — Використання мультикласової класифікації

Це означає, що модель може прогнозувати різні рівні небезпеки для ділянки дороги, а не просто визначати, чи є ділянка небезпечною, чи ні.

Особливість використання мультикласової класифікації полягає в тому, що вона дає можливість моделі вивчити більш складні шаблони в даних. Наприклад, модель може вивчити, що деякі комбінації параметрів, таких як тип дороги та розмітка, можуть вказувати на високий рівень небезпеки, тоді як інші комбінації можуть вказувати на середній або низький рівень небезпеки.

Спочатку сформовано базу даних та моделі вхідних даних. Це дозволило зберігати великі обсяги даних про ДТП. Модель вхідних даних була розроблена таким чином, щоб відображати ключові параметри, які можуть впливати на небезпечність ділянки дороги, такі як стан дороги, тип дороги, обмеження швидкості тощо.

Також створено таблицю для відображення даних про ДТП. Вона дає можливість користувачам легко переглядати та аналізувати дані, що зберігаються в базі даних. Далі створено інтерфейс для записів ДТП, що дає можливість користувачам створювати, читати, оновлювати та видаляти записи в базі даних.

Переваги цього застосування полягають в його автоматизованому та об’єктивному характері. Модель машинного навчання, яка навчена на великому обсязі даних, здатна виявляти складні залежності та шаблони, які можуть бути неочевидними для людини. Крім того, використання такої моделі дає можливість стандартизувати процес оцінки небезпеки ділянок доріг, зменшуючи вплив суб’єктивних факторів.

Таким чином, цей сценарій демонструє, як застосування машинного навчання може допомогти розв’язати важливі соціальні проблеми.

4.2.2 Прогнозування навантаження на енергетичну систему

Опрацьовано сценарій прогнозування значення навантаження на енергетичну систему України, яке базується на основі вже наявних даних за минулий час. У сценарії використано набір даних із відкритих джерел, а саме — «Навантаження на енергетичну систему України». Дані були сформовані міжнародною місією «ENTSO-E» — це асоціація співпраці європейських операторів подачі електроенергії.

Використано структуру даних для регресії, яка містила дані про дату і значення. Це надає можливість проєктувати використання застосунку практично на будь-яку задачу, де є набір історичних даних.

Використовуючи досвід застосування мультикласової класифікації вирішено створити аналогічний шлях взаємодії із функціоналом машинного навчання, але вже для задачі регресії (рисунок 4.2).

Для демонстрації необхідно створити графічні візуалізації, а саме графіки. На графіках має бути візуалізовано історичні дані та доповнено однією лінією, яка візуалізує саме прогнозування з накладанням на реальні дані.

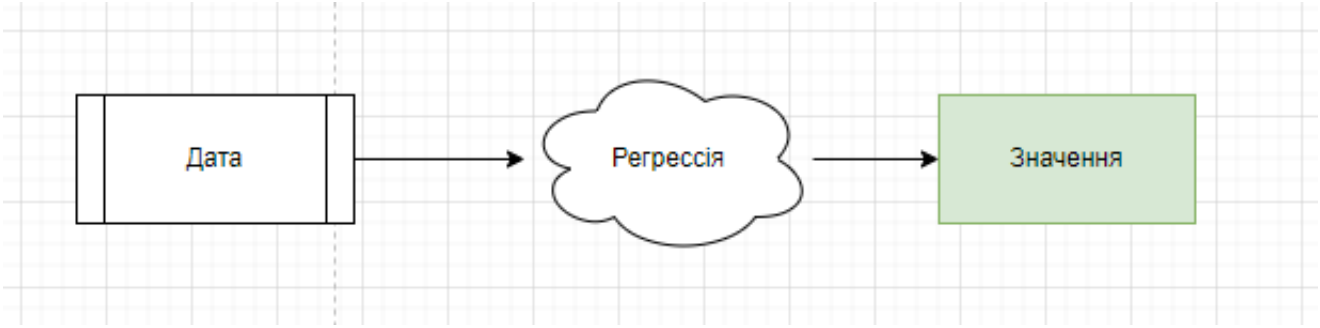


Рисунок 4.2 — Використання регресії

Таким чином, використання регресії виявилося вдалим для аналізу та прогнозування даних в контексті даного сценарію. Регресія дозволила не тільки розв’язати поставлену задачу, але й отримати важливі висновки, що можуть бути використані для подальшого планування та оптимізації роботи енергетичної системи.

4.3 Робота користувача з системою

При взаємодії з системою користувач спочатку потрапляє на сторінку конкретного сценарію, де він вже перебуває в середовищі, спеціально адаптованому для здійснення різноманітних завдань. Ця сторінка надає користувачеві відмінну можливість маніпулювати різними наборами даних та застосовувати методи машинного навчання безпосередньо в режимі реального часу.

Важливим аспектом є той факт, що всі елементи сторінки мають чітку структуру, що полегшує їхнє розуміння та навігацію. Кожен функціональний блок обладнаний інтуїтивно зрозумілими назвами, що сприяє зручності в користуванні. Враховуючи це, користувач може легко і швидко зорієнтуватися в інтерфейсі системи, не витрачаючи зайвого часу на пошук необхідних функцій чи опцій.

На сторінці конкретного сценарію користувач зустрічає головні функціональні елементи, які визначають його можливості та розширюють сферу використання системи. Ці ключові елементи, позначені відповідними цифрами на рисунку 4.3, визначаються :

1 — додавання запису до набору даних;

2 — експорт даних;

3 — імпорт даних;

4 — взаємодія із машинним навчанням;

5 — інтерактивна таблиця, яка візуалізує набір даних.

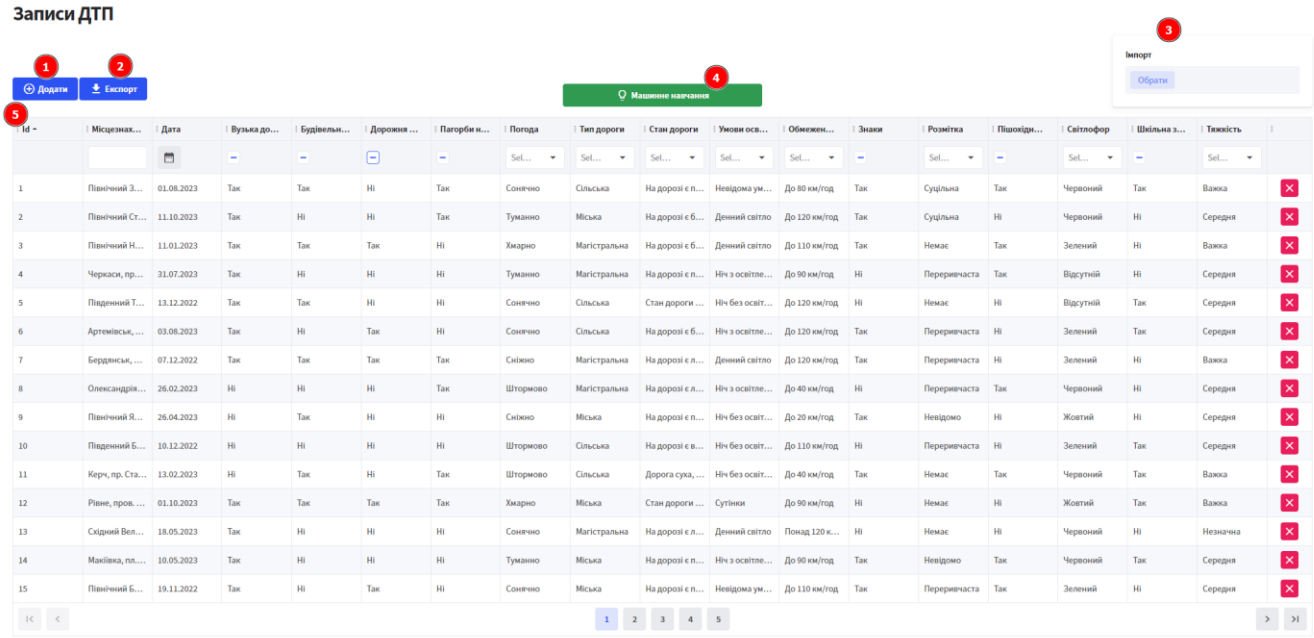


Рисунок 4.3 — Сторінка сценарію (Записи ДТП)

При додаванні запису до набору даних (рисунок 4.4) користувач може скористатися вбудованим інтерфейсом, який дає можливість введення необхідних параметрів. Такий підхід робить процес внесення нових даних максимально ефективним та легким для користувача, сприяючи точності та актуальності інформації в системі.

Інтерфейс представляє собою форму із полями, яка відкривається у діалоговому вікні. Це дає можливість залишатися на тій самій сторінці, без перезавантажень.

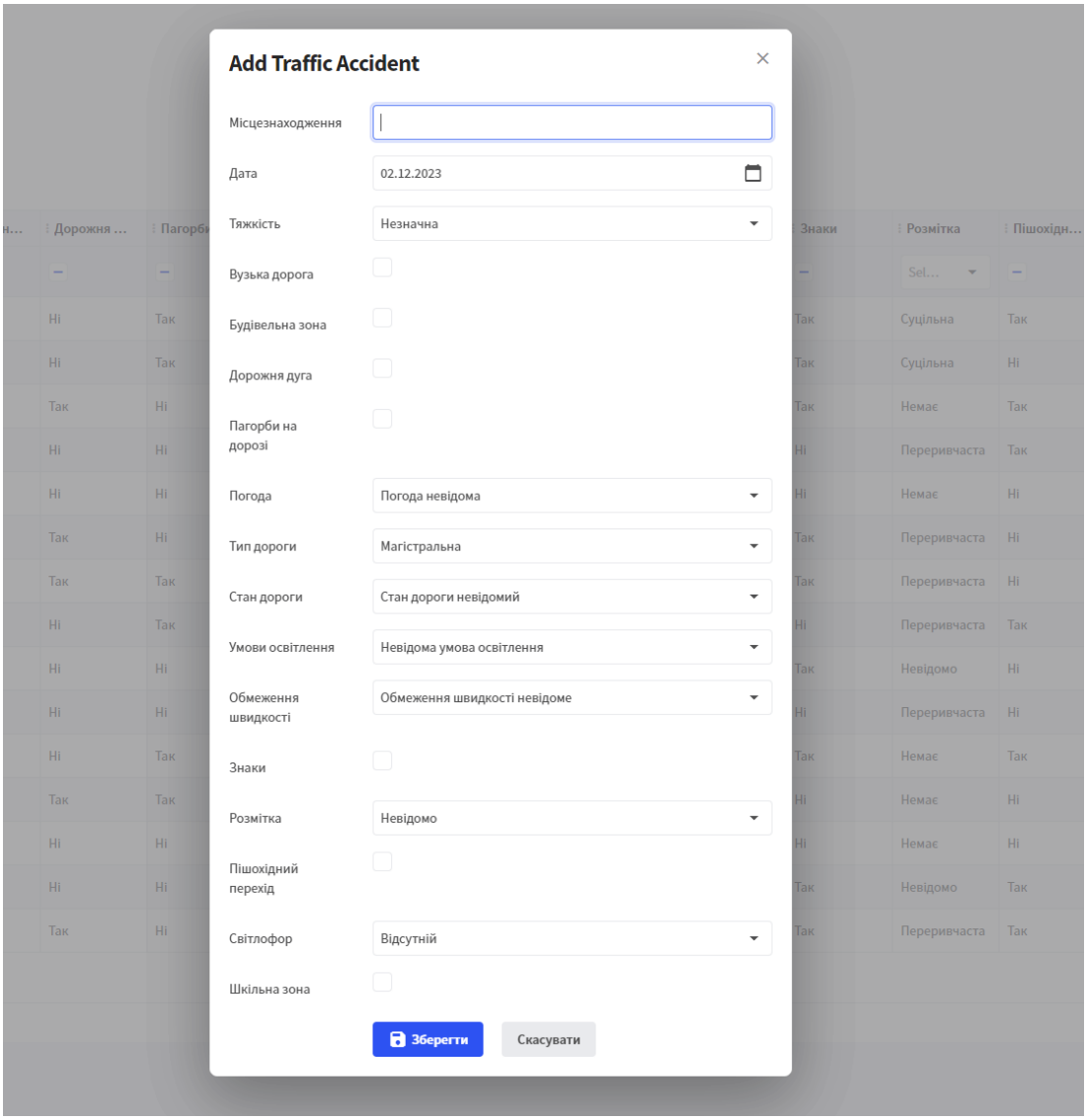


Рисунок 4.4 — Додавання нового запису

Загалом, дана система вирізняється не лише своєю функціональністю, але й великою увагою до зручності користування, що сприяє продуктивній та ефективній роботі користувачів у межах визначених сценаріїв використання.

4.3.1 Налаштування машинного навчання

Сценарії використання машинного навчання відрізняються своєю унікальністю, а кожен з них обладнаний власним інтерфейсом взаємодії з машинним навчанням. Цей інтерфейс включає в себе ряд ключових функцій, спрямованих на забезпечення ефективного та зручного користування процесом навчання моделей.

Усередині цього інтерфейсу доступні налаштування моделей, що дає можливість користувачам точно налаштувати параметри відповідно до вимог

конкретного сценарію. Важливим елементом інтерфейсу є можливість обрати необхідні параметри, що визначають специфіку завдання та орієнтацію моделі на досягнення конкретних результатів.

З рисунка 4.5 видно, що функціонал цього інтерфейсу ретельно структурований та включає в себе декілька ключових елементів, кожен з яких відіграє важливу роль у процесі використання машинного навчання:

1 — налаштування вхідних параметрів — дає можливість користувачеві визначити параметри, які будуть використовуватися як вхідні дані для навчання моделі. Тут можна задати ключові характеристики даних, що допомагає оптимізувати процес навчання;

2 — вихідні параметри — параметри чи результати, які мають бути отримані в результаті роботи моделі;

3 — обрання моделі (включаючи функцію AutoML) — користувач може вибрати конкретну модель для навчання або навіть використовувати автоматичний механізм машинного навчання (AutoML), який самостійно вибирає та налаштовує модель на основі вхідних даних;

4 — налаштування параметрів моделі — дає можливість користувачу налаштовувати різноманітні параметри моделі, щоб досягти оптимальної продуктивності і точності;

5 — налаштування значення фракції набору даних — дає можливість вибирати частку доступного набору даних для використання в процесі навчання моделі, що може бути корисним у випадку великих обсягів даних;

6 — кнопка «Старт» — ключовий елемент, який ініціює процес навчання моделі після того, як всі налаштування вже визначені;

7 — метрики моделі — виводить результати та метрики про продуктивність моделі, такі як точність, витрати тощо, для оцінки її ефективності;

8 — панель вводу даних для прогнозування — дає можливість користувачеві вводити нові дані для отримання прогнозів від навченої моделі;

9 — результати прогнозування — виводить результати прогнозування моделі на введених користувачем даних.

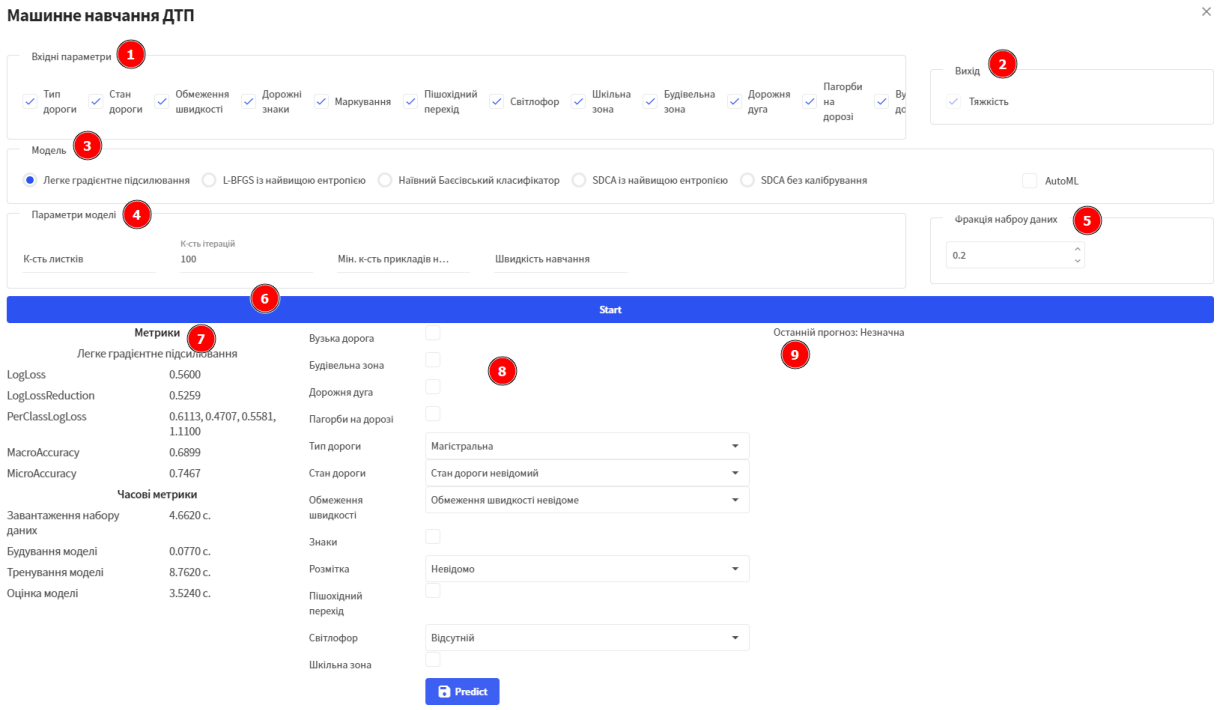


Рисунок 4.5 — Інтерфейс взаємодії з машинним навчанням

Цей детальний функціонал інтерфейсу налаштування машинного навчання створений таким чином, щоб забезпечити користувачу максимальний комфорт та повний контроль над усім процесом машинного навчання. Він дає можливість користувачу керувати всіма аспектами машинного навчання, роблячи його дійсно ефективним та гнучким. Це означає, що користувач може налаштувати процес навчання відповідно до своїх потреб та вимог. Такий підхід не тільки полегшує роботу з системою, але й дає можливість досягти кращих результатів.

4.3.2 Експорт та імпорт наборів даних

Велика увага приділяється зручності завантаження та обробці наборів даних. Для забезпечення максимального комфорту користувачів була впроваджена система експорту та імпорту з формату CSV та Microsoft Excel файлів.

Можливість експорту та імпорту даних у форматі CSV (рисунок 4.6) дає можливість легко обмінюватися інформацією між різними програмами та платформами. Користувачі можуть зручно зберігати свої набори даних у стандартному текстовому форматі, що спрощує обробку та аналіз.

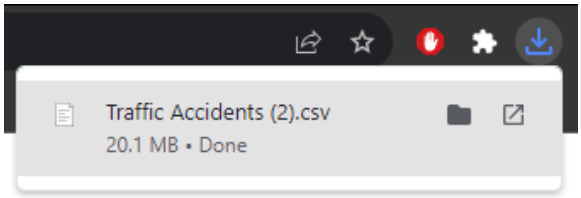
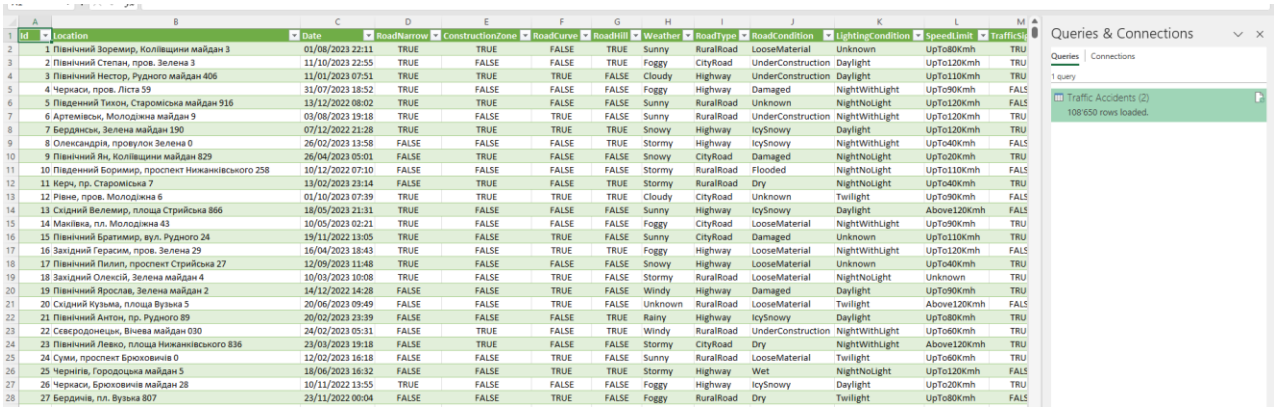


Рисунок 4.6 — Завантажений файл записів ДТП

Система також надає можливість експорту та імпорту даних безпосередньо у формат Microsoft Excel. Це особливо важливо для користувачів, які звикли використовувати Excel для зберігання та обробки даних (рисунок 4.7). Такий підхід полегшує взаємодію між системою машинного навчання та іншими робочими інструментами.



Id	Location	Date	RoadNarrow	ConstructionZone	RoadCurve	RoadHill	Weather	RoadType	RoadCondition	LightingCondition	SpeedLimit	TrafficSt
1	Північний Зоремир, Колівщини майдан 3	01/08/2023 22:11	TRUE	TRUE	FALSE	TRUE	Sunny	RuralRoad	LooseMaterial	Unknown	UpTo80Kmh	TRU
2	Північний Степан, пров. Зелена 3	11/10/2023 22:55	TRUE	FALSE	FALSE	TRUE	Foggy	CityRoad	UnderConstruction	Daylight	UpTo120Kmh	TRU
3	Північний Нестор, Рудного майдан 406	11/01/2023 07:51	TRUE	TRUE	TRUE	FALSE	Cloudy	Highway	UnderConstruction	Daylight	UpTo110Kmh	TRU
4	Черкаси, пров. Ліста 59	31/07/2023 18:52	TRUE	FALSE	FALSE	FALSE	Foggy	Highway	Damaged	NightWithLight	UpTo90Kmh	FALS
5	Південний Тихон, Староміська майдан 916	13/12/2022 08:02	TRUE	TRUE	FALSE	FALSE	Sunny	RuralRoad	Unknown	NightNoLight	UpTo120Kmh	FALS
6	Артемівськ, Молодіжна майдан 9	03/08/2023 19:18	TRUE	FALSE	TRUE	FALSE	Sunny	RuralRoad	UnderConstruction	NightWithLight	UpTo120Kmh	TRU
7	Бердянськ, Зелена майдан 190	07/12/2022 21:28	TRUE	TRUE	TRUE	TRUE	Snowy	Highway	IcySnowy	Daylight	UpTo120Kmh	TRU
8	Олександрія, провулок Зелена 0	26/02/2023 13:58	FALSE	FALSE	FALSE	TRUE	Stormy	Highway	IcySnowy	NightWithLight	UpTo40Kmh	FALS
9	Північний Ян, Колівщини майдан 829	26/04/2023 05:01	FALSE	TRUE	FALSE	FALSE	Snowy	CityRoad	Damaged	NightNoLight	UpTo20Kmh	TRU
10	Південний Боримир, проспект Нижанківського 258	10/12/2022 07:10	FALSE	FALSE	FALSE	FALSE	Stormy	RuralRoad	Flooded	NightNoLight	UpTo110Kmh	FALS
11	Керч, пр. Староміська 7	13/02/2023 23:14	FALSE	TRUE	FALSE	FALSE	Stormy	RuralRoad	Dry	NightNoLight	UpTo40Kmh	TRU
12	Рівне, пров. Молодіжна 6	01/10/2023 07:39	TRUE	TRUE	TRUE	TRUE	Cloudy	CityRoad	Unknown	Twilight	UpTo90Kmh	FALS
13	Східний Велемир, площа Стрийська 866	18/05/2023 21:31	TRUE	FALSE	FALSE	FALSE	Sunny	Highway	IcySnowy	Daylight	Above120Kmh	FALS
14	Макіївка, пл. Молодіжна 43	10/05/2023 02:21	TRUE	FALSE	FALSE	FALSE	Foggy	CityRoad	LooseMaterial	NightWithLight	UpTo90Kmh	TRU
15	Північний Братимир, вул. Рудного 24	19/11/2022 13:05	TRUE	FALSE	TRUE	FALSE	Sunny	CityRoad	Damaged	Unknown	UpTo110Kmh	TRU
16	Західний Герасим, пров. Зелена 29	16/04/2023 18:43	TRUE	FALSE	TRUE	TRUE	Foggy	Highway	LooseMaterial	NightWithLight	UpTo120Kmh	FALS
17	Північний Пилип, проспект Стрийська 27	12/09/2023 11:48	TRUE	FALSE	FALSE	FALSE	Snowy	Highway	LooseMaterial	Unknown	UpTo40Kmh	TRU
18	Західний Олексій, Зелена майдан 4	10/03/2023 10:08	TRUE	FALSE	TRUE	FALSE	Stormy	RuralRoad	LooseMaterial	NightNoLight	Unknown	TRU
19	Північний Ярослав, Зелена майдан 2	14/12/2022 14:28	FALSE	FALSE	TRUE	FALSE	Windy	Highway	Damaged	Daylight	UpTo90Kmh	TRU
20	Східний Кузьма, площа Вулька 5	20/06/2023 09:49	FALSE	FALSE	TRUE	FALSE	Unknown	RuralRoad	LooseMaterial	Twilight	Above120Kmh	FALS
21	Північний Антон, пр. Рудного 89	20/02/2023 23:39	FALSE	FALSE	FALSE	TRUE	Rainy	Highway	IcySnowy	Daylight	UpTo80Kmh	TRU
22	Северодонецьк, Вічева майдан 030	24/02/2023 05:31	FALSE	TRUE	FALSE	TRUE	Windy	RuralRoad	UnderConstruction	NightWithLight	UpTo60Kmh	TRU
23	Північний Левко, площа Нижанківського 836	23/03/2023 19:18	FALSE	TRUE	FALSE	FALSE	Stormy	CityRoad	Dry	NightWithLight	Above120Kmh	TRU
24	Суми, проспект Брюховичів 0	12/02/2023 16:18	FALSE	FALSE	TRUE	FALSE	Sunny	RuralRoad	LooseMaterial	Twilight	UpTo60Kmh	TRU
25	Чернігів, Городоцька майдан 5	18/06/2023 16:32	FALSE	FALSE	TRUE	TRUE	Stormy	Highway	Wet	NightNoLight	UpTo120Kmh	FALS
26	Черкаси, Брюховичів майдан 28	10/11/2022 13:55	TRUE	FALSE	FALSE	FALSE	Foggy	Highway	IcySnowy	Daylight	UpTo120Kmh	TRU
27	Бердичів, пл. Вулька 807	23/11/2022 00:04	FALSE	FALSE	TRUE	FALSE	Foggy	RuralRoad	Dry	Twilight	UpTo80Kmh	FALS

Рисунок 4.7 — Обробка даних у Microsoft Excel

Щодо можливостей масового завантаження, система розроблена з урахуванням потреб користувачів у роботі з великими обсягами інформації. Підтримка масового завантаження дає можливість ефективно обробляти великі набори даних чи бази даних, прискорюючи процес введення та підготовки великої кількості даних для аналізу та навчання моделей.

Що стосується автоматичного визначення структури даних, система видає вражаючу функціональність. Під час імпорту файлів (рисунок 4.8) у форматах CSV та Excel система не лише виконує їхню обробку, але й автоматично проводить аналіз та визначення структури даних.

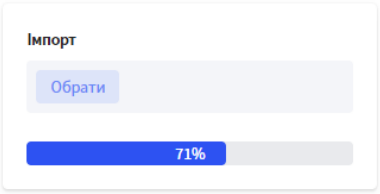


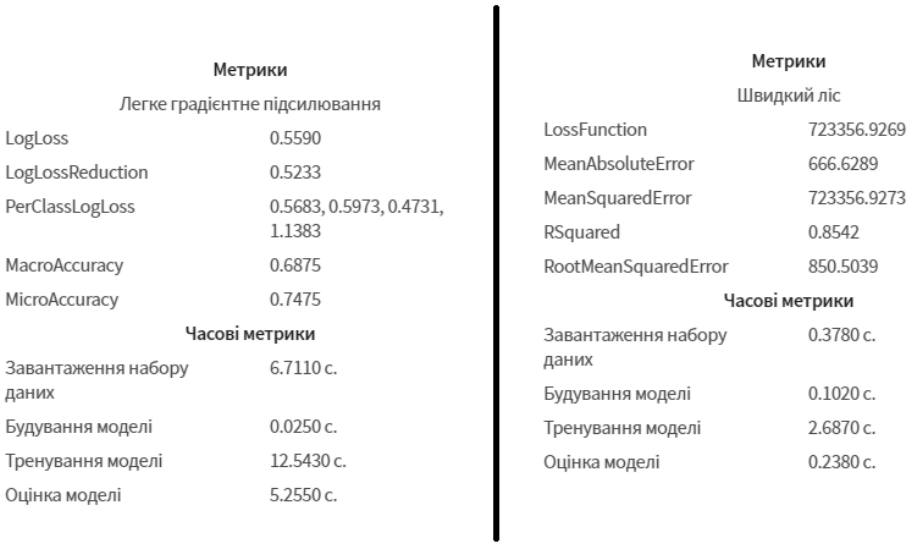
Рисунок 4.8 — Прогрес імпортування даних

Це робить процес завантаження та підготовки даних не лише простим, а й інтуїтивно зрозумілим, забезпечуючи користувачам додатковий рівень зручності та швидкості у використанні системи машинного навчання.

4.3.3 Візуалізація результатів

Візуалізація результатів та метрик в системі є важливою частиною, яка допомагає користувачам краще розуміти та інтерпретувати результати моделей. Вона дає можливість відображати складні дані та моделі в простій та зрозумілій формі. Візуалізація допомагає виявити шаблони, тенденції та відхилення, які можуть бути непомітними при аналізі сировинних даних.

Система надає різні метрики для моделей регресії та мультикласової класифікації (рисунок 4.9) [35]. Метрики для моделей регресії включають, наприклад, середню абсолютну помилку, середньоквадратичну помилку, корінь середньоквадратичної помилки та коефіцієнт детермінації [36]. Ці метрики допомагають оцінити, наскільки добре модель регресії передбачає незалежні змінні.



Метрики	
Легке градієнтне підсилювання	
LogLoss	0.5590
LogLossReduction	0.5233
PerClassLogLoss	0.5683, 0.5973, 0.4731, 1.1383
MacroAccuracy	0.6875
MicroAccuracy	0.7475
Часові метрики	
Завантаження набору даних	6.7110 с.
Будування моделі	0.0250 с.
Тренування моделі	12.5430 с.
Оцінка моделі	5.2550 с.

Метрики	
Швидкий ліс	
LossFunction	723356.9269
MeanAbsoluteError	666.6289
MeanSquaredError	723356.9273
RSquared	0.8542
RootMeanSquaredError	850.5039
Часові метрики	
Завантаження набору даних	0.3780 с.
Будування моделі	0.1020 с.
Тренування моделі	2.6870 с.
Оцінка моделі	0.2380 с.

Рисунок 4.9 — Метрики мультикласової класифікації та регресії

Для регресії система також надає графіки (рисунок 4.10), на яких можна легко побачити тенденції та прогнозування значень на наступні кроки. Ці графіки допомагають візуалізувати відносини між змінними та передбачуваними значеннями. Вони також допомагають виявити будь-які відхилення або аномалії в даних.

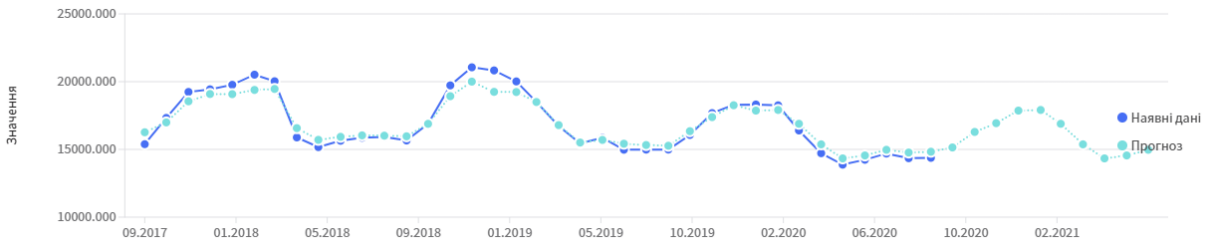


Рисунок 4.10 — Графік результатів прогнозування для регресії

Візуалізація результатів та метрик в системі не тільки полегшує інтерпретацію результатів, але й допомагає виявити будь-які проблеми або недоліки в моделі. Наприклад, якщо модель регресії має високий коефіцієнт детермінації, але велику середньоквадратичну помилку, це може вказувати на проблему з моделлю, як-от перенавчання.

Таким чином, візуалізація результатів та метрик в системі є важливим інструментом для аналізу та інтерпретації моделей. Вона допомагає користувачам краще розуміти результати моделей, виявляти проблеми та вдосконалювати моделі.

Висновки до розділу 4

Детально проаналізовано та розроблено два сценарії використання програмного продукту для задач мультикласової класифікації і регресії:

— класифікація небезпеки на ділянці дороги;

— прогнозування навантаження на енергетичну систему.

Також розглянуто використання системи та налаштування машинного навчання для великих обсягів даних, що включає в себе оптимізацію алгоритмів для швидкості і точності обробки великих наборів інформації. Представлено

можливість експортування та імпортування даних у систему, що сприяє легкому обміну інформацією між різними платформами і джерелами даних.

Продемонстровано візуалізацію метрик оцінювання моделі машинного навчання та результатів прогнозування для мультикласової класифікації і регресії, що сприяє зрозумілій інтерпретації отриманих результатів та допомагає користувачам приймати рішення на основі результатів та оцінок моделей.

5. СТАРТАП-ПРОЄКТ

Розробка стартап-проєкту є важливою частиною магістерської дисертації. Цей розділ має на меті дослідити та висвітлити ключові маркетингові аспекти, які впливають на успішне впровадження власного бізнес-проєкту [38].

Це включає в себе аналіз потенційного ринку, визначення цільової аудиторії, розробку стратегії просування та визначення каналів комунікації з клієнтами. Також важливим елементом є розробка фінансової моделі проєкту, включаючи прогнозування доходів та витрат, оцінку ризиків та можливостей.

Всі отримані практичні навички та знання — можуть бути використані при створенні власного бізнесу в майбутньому.

5.1 Аналіз ідеї

На початковому етапі розробки стартап-проєкту ML Playground виконується глибокий аналіз основної ідеї, який включає в себе ретельне вивчення її концепції, потенційної корисності та можливостей реалізації. Розглянемо загальні дані в таблиці 5.1:

Таблиця 5.1 — Загальні дані стартап-проєкту ML Playground

Зміст ідеї	Напрями застосування	Вигоди для користувача
Інноваційний та інтерактивний інструмент для машинного навчання, який спрощує розробку моделей через зручний інтерфейс. Він дає можливість працювати з різними наборами даних, використовувати різні моделі та оптимізувати їх	Освіта та навчання	Легкість використання
	Прогнозування та аналіз даних	Інтерактивність
	Дослідження та розвиток	Широкий вибір моделей
	Стартапи та розробники	Детальне налаштування
	Медичні дослідження	

Аналіз загальних даних стартап-проєкту, представлених у таблиці 5.1, вказує на широкий спектр можливостей та напрямків застосування інноваційного інтерактивного інструмента для машинного навчання. Зміст ідеї визначається як інноваційний та спрощений інструмент для розробки моделей, пристосований для використання в різних галузях.

Щодо напрямків застосування, стартап спрямований на освіту та навчання, де можливості використання моделей машинного навчання можуть значно полегшити процес навчання та розробки нових педагогічних методик. Прогнозування та аналіз даних стають іншим ключовим напрямком, де інтерактивність і широкий вибір моделей можуть забезпечити точні та ефективні результати.

Вигоди для користувача, зазначені у таблиці, включають легкість використання, що може спростити взаємодію з інструментом для широкого кола користувачів. Інтерактивність та широкий вибір моделей дозволяють користувачам персоналізувати та оптимізувати свої рішення, забезпечуючи більш гнучке та ефективне використання інструмента.

Таким чином, загальний аналіз стартап-проєкту вказує на його множинність застосувань та потенційну вигоду для різноманітних галузей, від освіти до медичних досліджень, зробивши його перспективним інструментом для широкого спектру користувачів [39].

5.2 Концепція програмного продукту

Продукт ML Playground створений як інноваційний інструмент для зручної роботи з машинним навчанням, який повинен охопити різні аудиторії користувачів. Концепція продукту базується на поєднанні легкості використання, інтерактивності та розширеної функціональності для задоволення потреб користувачів.

Основною ідеєю є створення комфортного та інтуїтивно зрозумілого інтерфейсу. Користувачі будуть мати можливість без зусиль пристосовуватися до

використання ML Playground, навіть якщо вони новачки у сфері машинного навчання.

Застосунок буде надавати можливість перегляду результатів роботи моделі в реальному часі. Це створить візуально привабливий процес експериментування та сприятиме кращому розумінню впливу налаштувань на результат

Ключовим компонентом є фреймворк ML.NET, який забезпечує використання різних моделей та параметрів для машинного навчання. Він дозволить користувачам вибирати та налаштовувати моделі з урахуванням їхніх потреб та завдань.

Продукт має сценарії, спеціально адаптовані для освітніх завдань, наукових досліджень, бізнес-аналітики та розробки, забезпечуючи різноманітність застосувань.

Концепція передбачає можливість інтеграції ML Playground з іншими інструментами, що дозволить розширити можливості користувачів та полегшити спільну роботу над певними задачами.

Засоби порівняння та візуалізації результатів дозволять користувачам систематично вдосконалювати свої моделі та робити більш обґрунтовані вибори моделей для своїх потреб.

Концепція стартап-проєкту ML Playground спрямована на створення доступного та потужного інструмента для роботи з машинним навчанням. Враховуючи різноманіття застосувань та користувацьких потреб, програмний продукт має потенціал стати важливим ресурсом для навчання, наукових досліджень та розвитку у сфері штучного інтелекту.

5.3 Перспективи реалізації на ринку

Даний стартап-проєкт має величезний потенціал для успішної реалізації на ринку платформ та інструментів для взаємодії з машинним навчанням та аналізом даних. З врахуванням стрімкого розвитку цих галузей, попит на інноваційні та легкі у використанні інструменти стає все більшим [40].

Нижче наведено перспективи продукту на ринку та потенційні стратегії для здобуття конкурентної переваги:

— зростання споживчого попиту — машинне навчання швидко входить у різні сфери життя, і велика кількість користувачів шукає прості та ефективні інструменти для експериментів. Продукт ML Playground може задовольнити цей попит, надаючи доступну та динамічну платформу для навчання та розробки моделей;

— освіта та навчання — у сфері освіти ML Playground може стати невід'ємним інструментом для викладачів та студентів. Забезпечення легкого доступу до різноманітних моделей та можливості експериментувати з ними робить програмний продукт ідеальним для вивчення машинного навчання в академічних установах;

— професійне використання — для професіоналів у галузях машинного навчання та аналізу даних ML Playground може стати інструментом для швидкого прототипування та тестування моделей. Забезпечення можливості спільної роботи та обміну результатами може значно полегшити колективну роботу робочих груп;

— запровадження в бізнесі — для бізнес-середовища ML Playground може стати інструментом для впровадження машинного навчання у бізнес-процеси. Відділи аналізу даних можуть використовувати застосунок для створення та оптимізації моделей для рішень, пов'язаних з маркетингом, стратегією та фінансами;

— загальні тенденції ринку — сучасний ринок вимагає інструментів, які не лише ефективні, але й інтуїтивно зрозумілі. Продукт ML Playground відповідає цьому попиту, надаючи інтерактивне середовище, де користувачі можуть експериментувати та вдосконалювати свої моделі без глибоких знань у галузі програмування;

— глобальна конкуренція та вибіркові переваги — з конкурентною сферою слід враховувати попит на інструменти, які надають широкий функціонал разом з легкістю використання. Забезпечення постійного оновлення списку моделей машинного навчання, високого рівня безпеки та інтеграції з іншими відомими

інструментами може стати ключовими факторами вибору ML Playground на ринку;

— маркетинг та партнерство — ефективна стратегія маркетингу та партнерства з іншими ключовими гравцями в галузі може значно підсилити присутність ML Playground на ринку. Рекламні кампанії, участь у конференціях та співпраця з університетами можуть сприяти популяризації та впровадженню продукту.

Проєкт ML Playground має широкі перспективи на ринку машинного навчання та аналізу даних. З адаптованою концепцією для різних сфер використання та стратегіями залучення користувачів, він може стати важливим гравцем у своєму сегменті ринку, надаючи ефективний та доступний інструмент для розвитку та застосування моделей машинного навчання.

5.4 План реалізації

Підготовка маркетингової стратегії для ML Playground є важливим кроком у введенні продукту на ринок. У світі швидко зростаючого попиту на інструменти машинного навчання та аналізу даних, ефективна стратегія може забезпечити успіх та визнання від користувачів.

Визначення цільової аудиторії. Проєкт ML Playground спрямований в першу чергу на діяльність освіти та науки, тому студенти та викладачі мають пріоритет. Також великий відсоток аудиторії — це приватний бізнес.

Як можна бачити на наведеному рисунку 5.1, враховано такі сфери, як медицина та розробники. Це означає, що вони також були включені в аналіз. Машинне навчання в наш час відіграє важливу роль в медицині, що робить цю сферу однією з ключових в контексті використання цієї технології. Відповідно, можна припустити, що приблизно 20 відсотків аудиторії, що вивчає машинне навчання, пов’язані з медициною.

Щодо розробників, то вони становлять активну спільноту професіоналів у сфері комп'ютерних наук або програмування. Вони не лише використовують машинне навчання в своїй роботі, але й намагаються самостійно вивчати цю

область. Крім того, серед розробників є і просто звичайні люди, які хочуть ознайомитись з машинним навчанням та навчитись його використовувати.

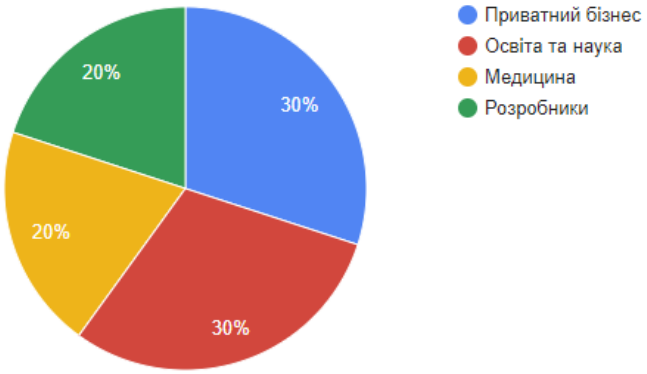


Рисунок 5.1 — Аудиторія стартап-проєкту

Аналіз потенційних конкурентів. На даний момент сфера машинного навчання має велику популярність, і вже існує досить багато стартапів на цю тематику, але розглянемо найвідоміші платформи:

— Google Colab — це безкоштовна платформа від Google для написання та запуску коду Python у середовищі Jupyter Notebook. Вона також має можливості для машинного навчання та обробки даних;

— IBM Watson Studio — платформа від IBM, яка надає інструменти для машинного навчання, аналізу даних та спільної роботи над проєктами;

— Microsoft Azure Machine Learning Studio — інструмент для створення, тренування та валідації моделей машинного навчання у хмарному середовищі Azure;

— DataRobot — платформа для автоматизації процесу побудови моделей машинного навчання;

— TensorFlow Playground — інтерактивна веб-платформа для експериментування з мережами глибокого навчання, яка може конкурувати в аспекті інтерактивності та легкості використання.

Порівняно з конкурентами, ML Playground вигідний завдяки своєму комфортному інтерфейсу та спрямуванню на широкий спектр користувачів. Інтерактивність, легкість використання та фокус на навчанні та дослідженнях

роблять його конкурентоспроможним на ринку інструментів машинного навчання.

Маркетингова кампанія. Важливою частиною розробки стартап-проєкту є планування маркетингової кампанії [40]. Це збільшить клієнтську базу та контракти на впровадження продукту. Вдала маркетингова кампанія робить продукт більш популярним та відомим. Імідж платформи ML Playground формується не лише його функціональністю, але й сприйняттям його користувачами та ринком. Правильно спроєктований бренд та ефективна комунікація можуть вивести продукт на новий рівень відомості та визнання [41].

Розглянемо основні позиції які будуть у маркетинговій компанії:

— стратегія бренду та позиціонування — визначившись з усіма унікальними характеристиками, потрібно створити значущий бренд, який відображатиме цінності та переваги платформи. Це буде чітко відрізняти ML Playground від конкурентів;

— маркетинговий зміст — потрібно створити якісний зміст, що відображає можливості та переваги платформи. У використанні можуть бути: блоги, відеоуроки, інфографіка тощо;

— соціальні мережі та онлайн присутність — активне оновлення профілів у соціальних мережах з акцентом на професійних спільнотах. Використання реклами для збільшення охоплення аудиторії та привертання нових користувачів;

— партнерства та колаборації — партнерство з університетами, бізнес-школами та дослідницькими установами [42]. Можна залучати відомих експертів у сфері машинного навчання для співпраці та рекомендацій;

— електронна пошта та розсилки — електронна пошта для регулярних оновлень, новин та спеціальних пропозицій [43]. Розсилки повинні бути цікавими та корисними для користувачів;

— участь у професійних заходах — потрібно брати участь у різноманітних заходах для підвищення визнання та залучення нових клієнтів;

— відгуки та рекомендації — активний збір позитивних відгуків від користувачів та публікація їх на веб-сайтах і в соцмережах;

— аналіз та оптимізація — систематичний аналіз результатів маркетингових компаній та внесення коректив у власну.

Дотримання цих ключових елементів гарантує, що маркетингова кампанія ML Playground буде сприяти збільшенню продажів та привертанню нових клієнтів.

Фінансове планування. Слід узяти до уваги різні аспекти, щоб забезпечити успішний запуск та стабільний розвиток проєкту, зокрема, витрати на розробку продукту, маркетинг та рекламу, витрати на персонал, інфраструктуру та обслуговування тощо [44, 45].

Таблиця 5.2 детально відображає розподіл витрат на різні категорії, що включають витрати на розробку програмного продукту, маркетинг та рекламу, витрати на персонал, інфраструктуру та обслуговування.

Таблиця 5.2 — Фінансовий план

Категорія	Підкатегорія	Сума
Витрати на розробку продукту	Реалізація програмного продукту	180 000 грн
	Технічне обладнання та ПЗ	150 000 грн
	Ліцензії та юридичні витрати	20 000 грн
Маркетинг	Рекламні кампанії	80 000 грн
	Розробка та підтримка маркетингового веб-сайту	30 000 грн
	Участь у конференціях	7 000 грн
Витрати на персонал	Зарплати для членів команди	190 000 грн
	Витрати на навчання	20 000 грн
	Витрати на відпочинок	15 000 грн
	Витрати на премії	20 000 грн
Інфраструктура та обслуговування	Оренда серверів	30 000 грн
	Технічна підтримка	10 000 грн
	Стратегія залучення фінансування	15 000 грн
Всього		767 000 грн

У категорії «Витрати на розробку продукту» виділяються важливі аспекти, такі як реалізація проєкту, технічне обладнання та програмне забезпечення, а також юридичні витрати та ліцензії. Це включає витрати на технічні засоби, необхідні для розробки продукту, а також правове забезпечення, щоб уникнути можливих юридичних проблем.

Категорія «Маркетинг» включає витрати на рекламні кампанії, розробку та підтримку маркетингового веб-сайту та участь у конференціях. Це важливо для створення пізнаванності продукту та привертання уваги цільової аудиторії.

Витрати на персонал включають зарплати для членів команди, витрати на навчання та відпочинок. Забезпечення задовільних умов праці та професійного розвитку персоналу є ключовим аспектом для збереження та мотивації команди.

Інфраструктура та обслуговування охоплюють витрати на оренду серверів, технічну підтримку та стратегію залучення фінансування. Ці витрати важливі для забезпечення ефективної роботи інфраструктури та забезпечення стійкості діяльності проєкту.

Загальна сума витрат на рік складає 767 000 грн, що відображає повний обсяг необхідних ресурсів для реалізації та успішного функціонування стартап-проєкту.

Висновки до розділу 5

В ході створення стартап-проєкту ML Playground виконано такі кроки:

- сформовано та проаналізовано ідею проєкту;
- розглянуто перспективи реалізації на ринку;
- розглянуто потенційних конкурентів;
- визначено аудиторію користувачів проєкту;
- сплановано маркетингову кампанію;
- сформовано фінансовий план.

ВИСНОВКИ

У магістерській дисертації розглянуто та проаналізовано широкий спектр методів і підходів оптимізації алгоритмів машинного навчання для ефективного прогнозування на великих обсягах даних. Теоретичний аналіз дав можливість визначити ключові аспекти, які впливають на продуктивність і точність моделей машинного навчання.

Проведений аналіз методів оптимізації та їхній порівняльний огляд дав змогу вибрати оптимальний підхід для розв’язання конкретних завдань прискорення прогнозування на великих обсягах даних.

Розроблено та реалізовано програмний продукт ML Playground, який демонструє високий рівень швидкодії та точності при використанні оптимізованих алгоритмів машинного навчання.

Застосування розробленого продукту в практичних сценаріях підтверджує його високу адаптивність до різних вимог і відмінну пристосованість до викликів, що ставляться перед сучасними технологіями в галузі енергетики та дорожньої безпеки.

Цей продукт демонструє не лише потужний потенціал для застосування в енергетиці та дорожній безпеці, але й відзначається гнучкістю і універсальністю, що відкриває нові горизонти для розгортання в інших промислових секторах та наукових галузях.

Розглянуто перспективи подальшого розвитку за тематикою дослідження, зокрема, розширення функціоналу програмного продукту і вдосконалення оптимізованих алгоритмів.

Створено стартап-проєкт, у якому проаналізовано перспективи ідеї розробленого програмного продукту. Виконано розгляд потенційних конкурентів, які існують на даний момент, та доведено конкурентоспроможність продукту. Проведено аналіз ринку програмних продуктів у сфері машинного навчання та сформовано маркетингову стратегію. Також створено приблизний фінансовий план для реалізації проєкту.

Розроблений програмний продукт та оптимізовані алгоритми мають практичний потенціал для вдосконалення процесів прогнозування на великих обсягах даних та відкривають перспективи для подальших досліджень у цій області.

Результати виконання магістерської дисертації були представлені на VII міжнародній науково-практичній конференції «Modern problems of science, education and society», 11-13 вересня 2023 року, Київ, Україна (Додаток А) і на V міжнародній науково-практичній конференції «Global science: prospects and innovations», 28-30 грудня 2023 року, Ліверпуль, Великобританія (Додаток Б).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кононова К. Ю. Машинне навчання: методи та моделі: підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка». К. Ю. Кононова. Харків: ХНУ імені В. Н. Каразіна, 2020. 301 с. URL: https://karazin.ua/storage/static-content/source/documents/vydavnytstvo/2021/navchalni-vydannia/Kononova_.pdf
2. Mathematics for Machine Learning. Marc Peter Deisenroth. Cambridge University Press, 2020. 398 с.
3. Машинне навчання. Комп’ютерний практикум. Л. М. Олещенко. КПІ ім. Ігоря Сікорського, 2022. 92 с. URL: https://ela.kpi.ua/handle/123456789/48731
4. Методи та системи штучного інтелекту: Навчальний посібник для студентів напряму підготовки 6.050101 «Комп’ютерні науки». А.С. Савченко, О. О. Синельніков. НАУ, 2017. 190 с. URL: https://pdf.lib.vntu.edu.ua/books/2020/Savchenko_2017_176.pdf
5. Порівняльний аналіз методів оптимізації функціоналу якості моделей машинного навчання. Н. Н. Шаповалова, О. Г. Рибальченко, Д. І. Куропятник. Вісник Криворізького національного університету, 2018. 104-111 с. URL: http://nbuv.gov.ua/UJRN/Vktu_2018_46_23
6. Learned Threshold Pruning. Kambiz Azarian, Yash Sanjay Bhalgat, Jinwon Lee, Tijmen Blankevoort. arXiv, 2020. 12 с. URL: https://arxiv.org/pdf/2003.00075.pdf
7. Optimal Brain Damage — Advances in Neural Information Processing Systems 2. Yann LeCun, John Denker, Sara Solla. Morgan Kaufmann Publishers Inc, 1990. 598-605 с. URL: https://dl.acm.org/doi/10.5555/2969830.2969903
8. Optimal Brain Surgeon and general network pruning. B. Hassibi, D.G. Stork, G.J. Wolff. IEEE, 1993. 293-299 с. URL: https://ieeexplore.ieee.org/abstract/document/298572/
9. Sparsity in deep learning: pruning and growth for efficient inference and training in neural networks — The Journal of Machine Learning Research Volume 22, Issue 1. Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, Alexandra Peste.

JMLR.org, 2021. 10882-11005 с. URL: https://dl.acm.org/doi/abs/10.5555/3546258.3546499

10. Learning to Prune Deep Neural Networks via Layer-wise Optimal Brain Surgeon — Advances in Neural Information Processing Systems 30. X Dong, S Chen, S Pan. Curran Associates Inc., 2017. 4860-4874 с. URL: https://proceedings.neurips.cc/paper/2017/file/c5dc3e08849bec07e33ca353de62ea04-Paper.pdf
11. Machine Learning Based Optimized Pruning Approach for Decoding in Statistical Machine Translation — IEEE Access. Debajyoty Banik, Asif Ekbal, Pushpak Bhattacharyya. IEEE, 2018. 1736-1751 с. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8588318
12. Comparing Biases for Minimal Network Construction with Back-Propagation — Advances in Neural Information Processing Systems 1. Stephen Hanson, Lorien Pratt. Morgan Kaufmann Publishers Inc, 1989. 177-185 с. URL: https://proceedings.neurips.cc/paper/1988/hash/1c9ac0159c94d8d0cbedc973445af2da-Abstract.html
13. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, Dmitry Kalenichenko. IEEE, 2018. 2704-2713 с. URL: http://openaccess.thecvf.com/content_cvpr_2018/html/Jacob_Quantization_and_Training_CVPR_2018_paper.html
14. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation. Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, Paulius Micikevicius. arXiv, 2020. 20 с. URL: https://arxiv.org/pdf/2004.09602.pdf
15. Parallel approaches to machine learning — A comprehensive survey. Sujatha R. Upadhyaya. Elsevier BV, 2012. 9 с. URL: https://readingxtra.github.io/docs/ml-gpu/survey-jpdc.pdf
16. Concurrency in C# Cookbook: Asynchronous, Parallel, and Multithreaded Programming. Stephen Cleary. O’Reilly Media, 2019. 251 с.

17. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. Yoshua Bengio, Nicholas Léonard, Aaron Courville. arXiv, 2013. 12c. URL: https://arxiv.org/pdf/1308.3432.pdf
18. Random Search for Hyper-Parameter Optimization. James Bergstra, Yoshua Bengio. JMLR.org, 2012. 25 c. URL: https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf
19. Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS. Petro Liashchynskyi, Pavlo Liashchynskyi. arXiv, 2019. 11 c. URL: https://arxiv.org/pdf/1912.06059.pdf
20. Taking the Human Out of the Loop: A Review of Bayesian Optimization — Proceedings of the IEEE. Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, Nando de Freitas. IEEE, 2015. 148-175 c. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7352306
21. A Tutorial on Bayesian Optimization. Peter I. Frazier. arXiv, 2018. 22 c. URL: https://arxiv.org/pdf/1807.02811.pdf
22. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, Ameet Talwalkar. JMLR.org, 2018. 52 c. URL: https://jmlr.org/papers/volume18/16-558/16-558.pdf
23. Algorithms for Hyper-Parameter Optimization — Advances in Neural Information Processing Systems 24. James Bergstra, Rémi Bardenet, Yoshua Bengio, Balázs Kégl. Curran Associates Inc., 2011. 2546-2554 c. URL: https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf
24. Advanced Lectures on Machine Learning. Carl Edward Rasmussen and Christopher K. I. Williams. The MIT Press, 2006. 266 c. URL: https://gaussianprocess.org/gpml/chapters/RW.pdf
25. Learn to Cache: Machine Learning for Network Edge Caching in the Big Data Era — IEEE Wireless Communications. Zheng Chang, Lei Lei, Zhenyu Zhou, Shiwen Mao, Tapani Ristaniemi. IEEE, 2018. 28-35 c. URL: https://par.nsf.gov/servlets/purl/10086524

26. Microsoft Blazor: Building Web Applications in .NET. Peter Himschoot. Apress, 2020. 303 с.
27. Blazor Server vs. Blazor WebAssembly — Building Single Page Applications in .NET Core 3. Michele Aponte. Apress, 2020. 120 с. URL: https://link.springer.com/chapter/10.1007/978-1-4842-5747-0_2
28. C# 11 and .NET 7 – Modern Cross-Platform Development Fundamentals: Start building websites and services with ASP.NET Core 7, Blazor, and EF Core 7. Mark J. Price. Packt Publishing, 2022. 818 с.
29. PostgreSQL: Introduction and Concepts. Bruce Momjian. Addison-Wesley, 2000. 462 с. URL: https://www.foo.be/docs-free/aw_pgsql_book.pdf
30. Practical PostgreSQL. Joshua Drake, John Worsley. O'Reilly Media, 2002. 636 с.
31. Programming ML.NET. Dino Esposito, Francesco Esposito. Microsoft Press, 2022. 49 с.
32. ML.NET Revealed: Simple Tools for Applying Machine Learning to Your Applications. Sudipta Mukherjee. Apress, 2020. 232 с.
33. Mastering .NET Machine Learning: Master the art of machine learning with .NET and gain insight into real-world applications. Jamie Dixon. Packt Publishing, 2016. 358 с.
34. Hands-On Machine Learning with ML.NET: Getting started with Microsoft ML.NET to implement popular machine learning algorithms in C#. Jarred Capellman. Packt Publishing, 2020. 298 с.
35. Metrics for Multi-Class Classification: an Overview. Margherita Grandini, Enrico Bagli, Giorgio Visani. arXiv, 2020. 17 с. URL: https://arxiv.org/pdf/2008.05756.pdf
36. Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology. Alexei Botchkarev. arXiv, 2019. 37 с. URL: https://arxiv.org/ftp/arxiv/papers/1809/1809.03006.pdf
37. Pragmatic Unit Testing in C# with NUnit. Andy Hunt, Dave Thomas, Matt Hargett. Pragmatic Bookshelf, 2007. 239 с.
38. Modelling the marketing of high-tech start-ups — Journal of Targeting, Measurement and Analysis for Marketing. Dragana Popovic. Palgrave Macmillan

Ltd., 2006. 14 с. URL: https://link.springer.com/content/pdf/10.1057/palgrave.jt.5740186.pdf

39. Стартап-проект. Рекомендації до виконання розділу магістерської дисертації «Розроблення стартап-проекту». П. В. Круш, Н. А. Шевчук, О. І. Андрусь. КПІ ім. Ігоря Сікорського, 2019. 50 с. URL: https://ela.kpi.ua/handle/123456789/27914
40. Основи інтернет-маркетингу. Кордзая, Н. Р. ОЛДІ-ПЛЮС, 2018. 184 с. URL: https://card-file.ontu.edu.ua/handle/123456789/10633
41. Інтернет-маркетинг в Україні: передумови виникнення, особливості становлення, перспективи розвитку. А. В. Семенова. Економічний вісник НТУУ «КПІ, 2013. с. 413–417
42. Управління стартапами. О. А. Гавриш, К. О. Бояринова, М. О. Кравченко, К. О. Копішинськ. КПІ ім. Ігоря Сікорського, 2020. 716 с.
43. Менеджмент стартап-проектів: навчально-методичний комплекс з дисципліни. О. А. Шевчук, К. О. Бояринова, Н. В.Рощина. КПІ ім. Ігоря Сікорського, 2023. 140 с.
44. The Lean Startup: How Today’s Entrepreneurs Use Continuous Innovationto Create Radically Successful Businesses. R. Eric. Crown Books: New York, USA, 2011.
45. Джерела фінансування стартапів в Україні. Інвестиції: практика та досвід. Гейдор А. П., Бізбіз Т. М. ТОВ «ДКС Центр», 2020. с. 73–78.

ДОДАТОК А

ОБРОБКА ДАНИХ В РЕАЛЬНОМУ ЧАСІ
НА ВЕЛИКИХ ОБСЯГАХ ДАНИХ

Тези на VII Міжнародну науково-практичну конференцію «Modern problems of science, education and society», 11-13 вересня 2023 року, м. Київ, Україна

УКР.НТУУ“КПІ ім. Ігоря Сікорського” _ІАТЕ_ЦТЕ_ ТР-21мп

Аркушів 7

Київ — 2023

ОБРОБКА ДАНИХ В РЕАЛЬНОМУ ЧАСІ НА ВЕЛИКИХ ОБСЯГАХ ДАНИХ

Сергєєв Данило Вікторович
студент кафедри цифрових технологій в енергетиці
НТУУ «КПІ ім. Ігоря Сікорського»
serhieiev.danylo@gmail.com

Вступ. У стрімкому та динамічному світі, де очікування клієнтів все вище, а людські ресурси все цінніші, машинне навчання і наука про великі дані відіграють вирішальну роль у розвитку. Цифрова технологізація робочого процесу життєво необхідна для збереження передових позицій в конкурентному середовищі будь-якої компанії.

Великі дані – це величезні обсяги різноманітної цифрової інформації, які надходять дуже швидко і не можуть бути оброблені за допомогою звичайних методів. Проте, вони дозволяють виявляти закономірності між подіями, які за іншими обставинами були б незрозумілими. З правильно сформульованим запитом можна отримати відмінні результати для оптимізації різних сфер діяльності [3]. Точно так, великі дані стали найціннішим ресурсом у сучасній економіці, що відображається у зміні списку найбільш дорогоцінних компаній за ринковою капіталізацією. На сьогодні інформація є таким же важливим стратегічним ресурсом, як і традиційні: Земля, повітря і вода. Країни по всьому світу поступово усвідомлюють настання епохи великих даних і розвивають галузь Big Data.

Мета дослідження. Проаналізувати особливості обробки даних в реальному часі на великих обсягах даних.

Матеріали та методи дослідження. Дослідження базувалось на теоретичному аналізі наукової літератури, а також аналізі та узагальненні отриманих даних. Для досягнення поставлених завдань використовувались методи порівняльного аналізу та класифікації. Добір матеріалу, застосування

методів індуктивного і логічного аналізу, а також статистичні методи аналізу літературних даних здійснювались з використанням системного підходу.

Постановка проблеми. Однак говорячи про незаперечні переваги застосування технологій Big Data на сучасних підприємствах, не варто забувати про існуючі обмеження застосування цих технологій в сучасних умовах. Сьогодні всі галузі та галузі в різному ступені піддаються впливу великих даних і використовують їх. Однак багато підприємств або організацій, які впроваджують великі дані, не мають успіху. Існує ще багато обмежень і проблем в застосуванні великих даних всередині підприємства. Згідно з дослідженням, існують загальні проблеми з невдалою реалізацією великих даних, тому часто використовують обробку даних в реальному часі.

Система реального часу (СРЧ) – це система, яка повинна реагувати на події у зовнішньому оточенні або впливати на нього, виконуючи свої завдання протягом обмеженого періоду часу. Поняття «реальний час» вказує на тимчасовий показник, який можна виміряти за фізичним годинником, навіть у відмінності від логічного часу, який відображає лише послідовність подій. Отже, опрацювання даних «в реальному часі» означає, що система обробляє інформацію настільки швидко, що час обробки може бути порівняний або менший за час передачі даних мережею. Якщо система вимагає часових параметрів для своєї роботи, то вона функціонує в режимі реального часу [2].

Дослідження методів програмної реалізації машинного навчання та ефективної обробки на великих обсягах даних є досить актуальною темою, оскільки потреба у використанні комп'ютерної техніки, програмних продуктів високої якості з кожним днем зростає все більше. Застосування машинного навчання може значно прискорити та підвищити ефективність прийняття рішень, прогнозувати та аналізувати поведінку системи, завдяки чому можна буде працювати в режимі реального часу і уникнути небажаних ситуацій.

Виклад основного матеріалу. Алгоритми машинного навчання стають більш ефективними в міру зростання обсягів навчальних даних. І об'єднуючи великі дані з машинним навчанням, ми отримуємо подвійну вигоду: алгоритми

Machine Learning допомагають нам справлятися з безперервним потоком даних, а обсяги і різноманітність потоків даних прокачують алгоритми, роблячи їх більш досконалими. Давайте подивимося, як в ідеалі повинен працювати процес інтеграції цих двох технологій.

Зв'язка Big Data I Machine Learning вже дає свої плоди в масштабних IT-проектах. Ці технології доповнюють один одного, роблячи роботу окремих додатків і цілих систем по-справжньому ефективною. Тому поглиблені знання в цих областях, розуміння того, як працюють ці технології в зв'язці, буде конкурентною перевагою при роботі в компаніях, що займаються обробкою і аналізом великих масивів даних [1].

На даний час існує безліч різноманітних алгоритмів машинного навчання. Рішення вибору алгоритму завжди залишається на увазі спеціалістів. Ті чи інші алгоритми завжди дають різну точність. Деякі алгоритми краще працюють за однією задачею, а деякі з іншою. Завжди треба пам'ятати – у світі машинного навчання ніколи не буває єдиного способу розв'язання проблеми. Завжди є кілька відповідних алгоритмів, і вам потрібно вибрати, який із них підходить краще. Виділяють два основні класи задач машинного навчання – алгоритми з вчителем (supervised learning) та без вчителя (unsupervised learning).

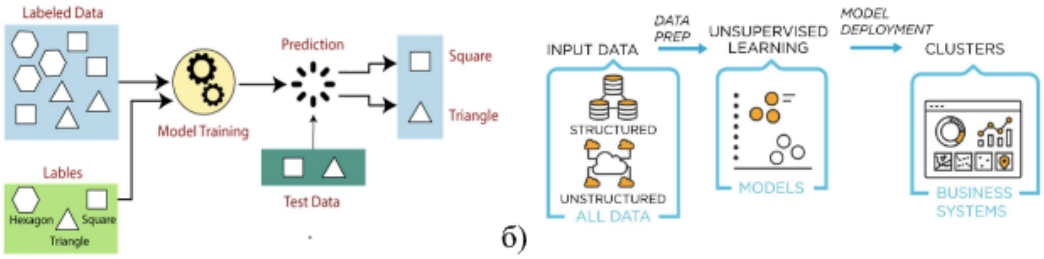


Рис. 1. Схематичне зображення машинного навчання – а) алгоритму з вчителем (supervised learning) та б) без вчителя (unsupervised learning)

Алгоритми навчання з вчителем засновані на тому, що дані які подаються на вхід, містять в собі правильні рішення, які називаються мітками. Типовий приклад, це задача класифікації, та регресії.

Алгоритм класифікації на основі вхідних даних з вірними мітками класів, повинна навчитися вірно класифікувати нові вхідні приклади. Алгоритм регресії повинен передбачати цільове числове значення. На основі предикторів (заданий набір цільових ознак) алгоритм повинен передбачити якесь числове значення. Щоб навчити алгоритм потрібно на вхід подати багато прикладів, враховуючи набір цільових ознак. Регресія дуже добре працює коли дані залежать від часу. Ще є таке зауваження, що алгоритм регресії, може розв'язувати задачу класифікації. Цей алгоритм називається логістичною регресією. Логістична регресія використовується для класифікації, бо вона може спрогнозувати значення, яке відповідає ймовірності належності до певного класу.

В навчанні без вчителя дані не позначені правильними відповідями. Задача стоїть в тому, що система сама вчиться без правильних відповідей, знаходити якісь приховані, цікаві закономірності. Наприклад, алгоритми кластеризації можуть виявити непересічні групи об’єктів. Цей алгоритм буде розділяти точки до кожного з кластерів, поки вони не будуть сформовані. В одному кластері будуть дуже схожі об’єкти, а в різних кластерах об’єкти будуть дуже сильно відрізнятись один від одного.

Основною задачею сучасної роботи з великими обсягами даних є їх ефективне стиснення. Стиснення полягає в тому, що воно включає перетворення рядка символів у певному представленні (наприклад, ASCII) у новий рядок (наприклад, бітів), що містить ту саму інформацію, але довжина якого є значно меншою. Таким чином, індексація великого обсягу даних є важливою частиною бази даних, що створена для підвищення продуктивності обробки запитів.

Для розв'язання цієї проблеми було розроблено алгоритми зменшення розмірності, такі як: аналіз основних компонентів (PCA, Jolliffe, 2011), багатомірного масштабування (MDS, Wickelmaier, 2003), стохастичного включення сусідів (tSNE, SNE, Maaten, Van Der & Hinton, 2008; Hinton & Roweis, 2003). Одним із найперших підходів до зменшення розмірності наборів

даних є алгоритм Isomap (Samko, Marshall & Rosin, 2006), який підходить для ізометричного відображення. Алгоритм Isomap шукає менший розмір, який підтримує геодезичні відстані між усіма точками.

Багатомірне масштабування (MDS, (Wickelmaier, 2003)) шукає представлення даних, коли відстані можна порівняти з відстанями в початковому просторі. Цей алгоритм використовують для аналізу даних подібності або відмінності. MDS намагається моделювати дані подібності або відмінності як відстані в геометричних просторах. Існує два типи алгоритму MDS: метричний та неметричний.

Також існує підхід до візуалізації T-розподілене вкладення стохастичної близькості або t-SNE (t-SNE це t-distributed stochastic neighbor embedding). Дана техніка надає можливість візуалізації великих обсягів даних, надаючи кожній точці даних розташування на дво- чи тривимірній карті за допомогою нелінійного зниження розмірності. Основне використання підходу t-SNE полягає у ефективному зображені кластерів. У контексті обраного датасету Fashion-MNIST, даний підхід матиме особливу перевагу, через те, що всі 10 класів датасету можна чітко розрізнити, порівняти на схожість між собою та візуалізувати новий підхід до навчання автокодувальників.

Для зниження розмірності великих обсягів даних можуть використовуватись готові реалізації алгоритмів бібліотеки Scikit-learn (Pedregosa et al., 2011; Documentation, 2018) – Isomap, MDS та t-SNE. Результати роботи алгоритмів візуально наведено на рис. 2.

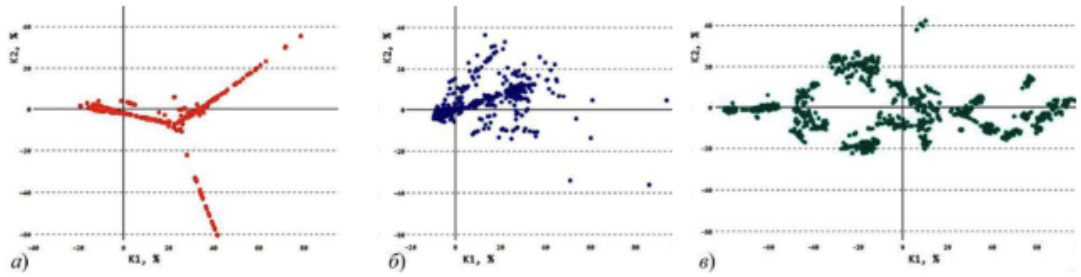


Рис. 2. Результат спрощення набору даних алгоритмами:

а) Isomap; б) MDS; в) t-SNE

Як видно з рис. 2, у результатах роботи алгоритмів спостерігаємо значні відмінності. Результатом роботи алгоритму Isomap (рис. 2 а) спостерігаємо три яскраво виражені промені. Якщо проводити візуалізацію спрощення великих обсягів даних методом MDS (рис. 2 б) видно локальні скупчення точок. Ці скупчення мають відмінну форму, порівняно з результатами Isomap. Однак також можна побачити три промені. Внаслідок роботи t-SNE (рис. 2 в) відсутня променева структура, а самі точки представлено у вигляді скупчень (кластерів).

Найкращої візуалізації способом зниження розмірності вдалося досягти за допомогою алгоритму t-SNE. Кластери з матеріалів, які сформовано за допомогою алгоритму, були найбільш чіткими та очевидними.

Однією з основних труднощів при використанні алгоритму t-SNE є правильний вибір відповідних гіперпараметрів, таких як perplexity (здивування) та learning rate (швидкість навчання), які можуть значно вплинути на кінцеву візуалізацію. Невірний вибір гіперпараметрів може спричинити спотворені візуалізації, які не відображають дійсну структуру даних. Крім того, сам алгоритм t-SNE може вимагати значних обчислювальних ресурсів, особливо для великих наборів даних, що робить його менш придатним для використання в реальному часі або в інтерактивних програмах.

Висновки. Проаналізовано шляхи оптимізації алгоритмів машинного навчання для прискорення прогнозування на великих обсягах даних за рахунок алгоритму t-SNE. Було введено апроксимацію Барнса-Хата, яка прискорює обчислення t-SNE і дозволяє використовувати його на великих наборах даних. Ще однією покращеною версією є алгоритм UMAP (Uniform Manifold Approximation and Projection), який комбінує переваги t-SNE з іншими методами зменшення розмірності, щоб забезпечити більш точні та інтерпретовані візуалізації. Подальші дослідження в цій області можуть спрямовуватися на розробку нових способів покращення алгоритму t-SNE.

Список літератури

1. Аксютіна Е. М., Бєлов Ю. С. Огляд архітектур і методів машинного навчання для аналізу великих даних. *Електронний журнал: наука, Техніка та освіта.* 2016. №1 (5). С. 132–139.
2. Боровський А. А. Перспективи застосування технологій машинного навчання до обробки великих масивів історичних даних. *Кібернетика та програмування.* 2015. № 1. С. 77–114.
3. Фісун М., Дворецький М., Дворецька С. Побудова моделей для оптимізації структури бази даних вузла у корпоративних інформаційних системах. *ITKI,* 2020. vol 48, № 2, С. 52–60.

ДОДАТОК Б

OPTIMIZATION IN ML. NET

Тези на V Міжнародну науково-практичну конференцію «Global science: prospects and innovations», 28-30 грудня 2023 року, м. Ліверпуль, Великобританія

УКР.НТУУ“КПІ ім. Ігоря Сікорського” _ІАТЕ_ЦТЕ_ ТР-21мп

Аркушів 8

Київ — 2023

УДК 004.43:517.91

OPTIMIZATION IN ML. NET

Larysa Kublii
Candidate of Technical Sciences
Associate Professor at the Department of Digital Technologies in Energy
Danylo Serhieiev
master's degree student
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv

Abstract. It is emphasized that optimization in ML. NET can be implemented using a variety of techniques and strategies aimed at improving the speed, scalability, and efficiency of machine learning models. Methods such as the use of compact models, quantization, parallel processing, selection of optimized algorithms, caching and preloading of models, monitoring and optimization of resource use are described.

Key words: Optimization, caching, pruning, ONNX, model preloading, Quantization, machine learning.

Introduction. In 2018, Microsoft developed ML. NET, which is a machine learning framework for C# and F# programming languages. Over time, this library has undergone significant changes and acquired new functions for identifying patterns in data. With the help of ML. NET, a machine learning model can be trained by specifying the appropriate algorithm, or import pre-trained TensorFlow and ONNX models [1].

Currently, ML. NET is an open-source cross-platform machine learning environment for developers. ML. NET includes ModelBuilder (an easy-to-use user interface tool in Visual Studio) and a CLI that make it easy to build custom machine learning (ML) models using automatic machine learning (AutoML). Machine learning optimization in ML. NET is the process of adjusting hyperparameters to minimize the cost function using one of the optimization methods. It is important to

minimize the cost function because it describes the discrepancy between the true value of the estimated parameter and what the model predicted.

Presenting main material. In this study, we will analyze the main types of optimization methods in ML. Optimization in ML. NET can be implemented using a variety of techniques and strategies aimed at improving the speed, scalability, and efficiency of machine learning models. Among the analyzed methods, the following can be mentioned: use of compact models, quantization, parallel processing, selection of optimized algorithms, caching and preloading of models, monitoring and optimization of resource use.

We will provide a more detailed description of the listed methods.

1. The use of compact models. Converting models to the ONNX format is a significant step in optimizing machine learning, as it not only reduces the size of the models but also significantly enhances their efficiency and computational speed. This is achieved through more efficient use of memory and acceleration of the inference process, especially in environments with limited resources.

Optimizing models in ML.NET using ONNX opens possibilities for integration with various computing environments, including cloud services, edge computing, and mobile platforms. Such versatility provides a broader range of model applications, from personalized applications on mobile devices to complex analytical systems on servers. The compactness and efficiency of models optimized through ONNX make them ideal for applications where response speed and low power consumption are important, such as in IoT systems and real-time processing.

Furthermore, ONNX promotes better compatibility between different machine learning tools, allowing developers to easily import and export models between various platforms. This enables the use of advanced machine learning techniques developed for one platform in other applications and environments. Such an approach provides great flexibility and the ability to adapt to the rapidly changing requirements of the modern world of digital technologies.

When deploying a model, it is important to choose formats that ensure compact dimensions of the model. For example, using the ONNX format can help reduce the size of the model and ensure efficient transmission over the network. An example of a software implementation of the optimization process can be the following code:

```
// An example of converting a model to ONNX format
var onnxModel = model. ToOnnx (Model. OnnxConverter. TensorFlow);
```

1. Pruning models in ML. NET. ML. NET supports pruning models [2], that is, removing unnecessary parameters or layers. This can be an important optimization step to obtain smaller and more efficient models:

```
// The prune of the model
var prunedModel = model. Transform. Prune();
```

In ML. NET compression and encoding techniques can be used to reduce model size. However, it is important to note that in some cases this may lead to a loss of accuracy:

```
// The compression of the model
var compressedModel = model. Transform. Compress();
```

2. Quantization. Using quantization reduces the number of bits used to represent model weights. This can significantly reduce memory requirements and improve data transfer rates [3]. An example of a software implementation of the optimization process can be the following code:

```
// The quantization of the model
var quantizationSettings = new QuantizationSettings() { QuantizationBits = 8 };
var compactModel = model. Transform. Conversion. ConvertToCompact
(quantizationSettings);
```

Take advantage of dynamic loading of models in ML. NET to load only the parts of the model that are needed for a particular task, which can help reduce the amount of memory used:

```
// The dynamic loading of models
var dynamicModel = Model. Load («dynamic_model. onnx»);
```

3. Parallel processing. Using parallel processing capabilities can improve model performance, especially on modern multi-core processors. Parallel processing in ML. NET can be achieved using a Microsoft library ML. Parallel. This

library provides parallel processing capabilities in different parts of your machine learning pipeline. The examples of using parallel processing in ML. NET are given below:

```
sing Microsoft. ML. Parallel;
// Creating the parallel object
var parallelOptions = new ParallelOptions();
parallelOptions. MaxDegreeOfParallelism = Environment. ProcessorCount;
// The use of the parallel object in the pipeline
var pipeline = new LearningPipeline();
pipeline. Add (new TextLoader (dataPath). CreateFrom<Data>());
pipeline. Add (new ColumnCopier((«Label», «Label»)));
pipeline. Add (new TextFeaturizer («Features», «Text»));
pipeline. Add (new FastTreeBinaryClassifier() { NumLeaves = 5, NumTrees = 5 });
// The use of the parallel processing to train a model
var model = pipeline. Train<Data, FastTreeBinaryClassificationModel>(data, parallelOptions);
```

Using the ParallelML. NET library makes it easy to introduce parallelism into different stages of the machine learning pipeline. In general, parallel processing in ML. NET can be useful for speeding up model training and performance evaluation, particularly when the user has large amounts of data or complex models.

4. Selection of optimized algorithms. It is important to choose optimized algorithms for machine learning tasks. For example, you can use LightGBM or TensorFlow for faster performance.
5. Caching and preloading models. Using caching can help avoid reloading the model on each call, which can improve the application's response time. Caching models in ML. NET can be implemented using memory caching systems such as MemoryCache in C#:

```
using Microsoft. Extensions. Caching. Memory;
// Creating a caching object
var memoryCache = new MemoryCache (new MemoryCacheOptions());
// Model loading and caching
var model = LoadModel();
memoryCache. Set («MyModelKey», model);
```

```
// Retrieving a model from the cache
var cachedModel = memoryCache. Get («MyModelKey») as ITransformer;
if (cachedModel != null)
// Using the model
var predictionEngine = mlContext. Model. CreatePredictionEngine<Data,
Prediction>(cachedModel);
var prediction = predictionEngine. Predict (new Data());
```

Models can be preloaded periodically or at certain events. For example, if the user knows that the data for training the model is periodically updated, then the model can be preloaded after each data update:

```
// Preloading the model when updating data
var updatedModel = TrainNewModel();
memoryCache. Set («MyModelKey», updatedModel);
```

6. Optimization of Hyperparameters. Optimizing the model's hyperparameters can improve its performance and reduce training time [4]. In ML. NET optimization of hyperparameters can be performed using Cross-Validation:

```
/ An example of using cross-validation to optimize hyperparameters
var crossValidationResults = mlContext. Regression. CrossValidate (data, pipeline,
numberOfFolds: 5);
var averageMetrics = crossValidationResults
.Select (fold => fold. Metrics. MeanSquaredError)
.Average();
```

This helps avoid overtraining and evaluate the performance of the model on different subsamples of the data.

7. Using the Grid Search method to sort through various combinations of hyperparameters, you can perform optimization to obtain the best values:

```
// An example of using Grid Search to optimize hyperparameters
var gridSearch = mlContext. Regression. TrainTestSplit (data)
.Append (mlContext. Regression. Trainers. FastTree (new FastTreeRegressionTrainer.
Options
NumberOfLeaves = 5,
MinimumExampleCountPerLeaf = 2
}));
```

– Monitoring and optimizing the use of resources. You should constantly monitor the use of resources (memory, processor time) and take measures for optimization.

Conclusions. The history of creation and peculiarities of work in ML. NET are analyzed. Methods of optimization in ML. NET are listed and characterized. It is emphasized that with the above strategies, the user can achieve significant optimization of models in ML. NET, which is especially important for applications where resources are limited, or high performance is required.

References

1. How to choose an ML. NET algorithm. URL: https://learn.microsoft.com/en-us/dotnet/machine-learning/how-to-choose-an-ml-net-algorithm. Accessed on: 21.11.2023.
2. Optimal Brain Damage. Yann Le Cun, John S. Denker and Sara A. Solla. URL: https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf. Accessed on: 21.11.2023.
3. Quantization for Neural Networks. Lei Mao. URL: https://leimao.github.io/article/Neural-Networks-Quantization/. Accessed on: 22.11.2023.
4. AutoML: Methods, Systems, Challenges. Chapter 1: Hyperparameter Optimization. URL: https://www.automl.org/wp-ontent/uploads/2019/05/AutoML_Book_Chapter1.pdf. Accessed on: 22.11.2023.