**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Західноукраїнський національний університет**
**Факультет комп'ютерних інформаційних технологій**
Кафедра комп'ютерної інженерії

**Рудик Владислава Всеволодівна**

# «Алгоритми формування елементів веб-сторінок на основі машинного навчання / Algorithms for generating web page elements based on machine learning»

спеціальність: 123 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерна інженерія
Кваліфікаційна робота

Виконав студент групи КІм-21
В.С. Рудик
_____

Науковий керівник:
к.т.н., доц. О. Й. Піцун
_____

Кваліфікаційну роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____Л. О. Дубчак

**Тернопіль – 2025**

# АНОТАЦІЯ

Рудик В. В. Алгоритми оптимального формування елементів веб-сторінок на основі машинного навчання. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 123 «Комп'ютерна інженерія», освітньо-професійна програма. Західноукраїнський національний університет, Тернопіль, 2025.

Робота написана обсягом 68 сторінок і містить 21 ілюстрацію, 3 таблиці, 2 додатка та 36 джерел за переліком посилань. Метою кваліфікаційної роботи є розроблення та впровадженні алгоритмів, що дозволяють автоматизувати процес формування елементів веб-сторінок різного рівня складності на основі аналізу метрик швидкодії та використання нейронних мереж.

Методи досліджень. Для розв'язання поставлених задач у кваліфікаційній роботі використано: методи: лінійної алгебри та аналітичної геометрії (для створення алгоритмів адаптивного підбору веб-сторінок); функціонального програмування (для проектування програмних засобів).

Результати дослідження: алгоритми формування елементів веб-сторінок на основі машинного навчання, методи класифікації інтерфейсів за метриками PageSpeed та ансамблеві підходи до визначення їхньої складності. Адаптивність забезпечують параметри пристрою, стан мережі й кластеризовані дані, що дозволяє автоматично обирати тип інтерфейсу та покращувати взаємодію користувача.

Результати роботи можуть бути застосовані для підвищення продуктивності та надійності веб-інтерфейсів у різних умовах використання.

Орієнтовні напрямки розвитку досліджень: створення сервісів для автоматичної генерації інтерфейсів на основі поведінкових і контекстних даних та покращення продуктивності за допомогою глибинного навчання і реальних даних користувачів.

КЛЮЧОВІ СЛОВА: АЛГОРИТМ, МАШИННЕ НАВЧАННЯ, АНАЛІЗ, ПРОДУКТИВНІСТЬ, ВЕБ-ІНТЕРФЕЙС, КЛАСТЕРИЗАЦІЯ.

# ANNOTATION

Rudyk V. V. Algorithms of optimal formation of web page elements based on machine learning. – Manuscript.

Qualification work for obtaining the degree of “Master” in specialty 123 “Computer Engineering”, educational and professional program. Western Ukrainian National University, Ternopil, 2025.

The work is written in 68 pages and contains 21 illustrations, 3 tables, 2 appendices and 36 sources according to the list of references. The purpose of the qualification work is the development and implementation of algorithms that automate the formation of web page elements of various complexity levels based on performance metrics analysis and the use of neural networks.

Research methods. To solve the tasks set in the qualification work, the following methods were used: linear algebra and analytical geometry (for creating algorithms of adaptive web page selection); functional programming (for designing software tools).

Research results: algorithms for forming web page elements using machine learning, methods for classifying interfaces by PageSpeed metrics, and ensemble approaches for determining their complexity. Adaptability is ensured by device parameters, network conditions, and clustered data, enabling automatic interface selection and improving user interaction.

The results of the work can be applied to enhance the performance and reliability of web interfaces under various usage conditions.

Approximate research development directions: development of services for automatic interface generation based on behavioral and contextual data, as well as further optimization of performance using deep learning and real user data.

KEYWORDS: ALGORITHM, MACHINE LEARNING, ANALYSIS, PERFORMANCE, WEB INTERFACE, CLUSTERING.

ЗМІСТ

# ВСТУП

Веб-технології стрімко інтегруються в повсякденне життя, трансформуючи способи доступу до інформації, отримання послуг та комунікації. Сучасні веб-ресурси вирізняються розмаїттям, високою зручністю, швидкодією та естетичним оформленням, що суттєво впливає на досвід користувачів. Зростаюча потреба у створенні інтуїтивно зрозумілих, функціональних та візуально привабливих веб-сторінок мотивує до пошуку нових підходів, спрямованих на оптимізацію та автоматизацію цього процесу. Зокрема, перспективним напрямом є застосування машинного навчання для створення елементів веб-інтерфейсів. Актуальність цього дослідження обумовлена необхідністю вдосконалення якості веб-ресурсів через автоматизацію їх проектування та адаптацію до різноманітних умов використання.

Існує значна кількість інструментів та методик для розроблення веб-інтерфейсів. Проте, навіть найкращі з них не завжди дозволяють досягти оптимального балансу між складністю дизайну, швидкодією та адаптивністю під різні пристрої та швидкість інтернет-з'єднання. Ручна розробка часто потребує значних часових і фінансових витрат, а також високої кваліфікації розробників. Використання алгоритмів машинного навчання може суттєво знизити ці витрати, автоматизуючи процеси аналізу та проектування веб-сторінок. Таким чином, розроблення алгоритмів формування елементів веб-сторінок на основі машинного навчання є важливим кроком до створення більш ефективних, швидких і доступних технологій розробки веб-ресурсів.

Мета даного дослідження полягає в розробленні та впровадженні алгоритмів, що дозволяють автоматизувати процес формування елементів веб-сторінок різного рівня складності на основі аналізу метрик швидкодії та використання нейронних мереж. Досягнення цієї мети передбачає виконання ряду завдань, серед яких вибір предметної області для дослідження, створення

HTML-шаблонів веб-сторінок різних рівнів складності, формування датасету на основі метрик PageSpeed Insights, навчання та тестування нейронної мережі для класифікації сайтів, а також визначення ключових характеристик веб-інтерфейсів для кожної категорії складності.

Об’єктом дослідження є процес створення та оптимізації веб-інтерфейсів із використанням машинного навчання.

Предметом дослідження виступають алгоритми формування елементів веб-сторінок різного рівня складності на основі аналізу показників швидкодії та інших метрик. Саме ця конкретна частина процесу створення веб-інтерфейсів дозволяє найбільш ефективно вирішувати задачі автоматизації та адаптації веб-ресурсів до потреб користувачів.

Методологія дослідження включає використання різних підходів і технологій. Для збору даних і оцінки метрик швидкодії було застосовано сервіс PageSpeed Insights. Формування HTML-шаблонів виконувалося на основі аналізу кращих практик у веб-дизайні. Для навчання нейронної мережі використовувались методи глибокого навчання, а також алгоритми класифікації. Аналіз ефективності розроблених рішень проводився за допомогою експериментального тестування та порівняння отриманих результатів із відомими підходами.

Наукова новизна даної роботи полягає у запропонованому алгоритмі до класифікації веб-ресурсів за рівнем складності з використанням нейронної мережі на основі метрик швидкодії. Розроблені алгоритми дозволяють визначати характерні елементи для веб-сторінок різної складності, що сприяє автоматизації процесу їх створення. Крім того, у роботі вперше реалізовано підхід до вибору типу веб-інтерфейсу для нового сайту, виходячи з вхідних параметрів, таких як тип пристрою, швидкість з’єднання тощо.

Практичне значення отриманих результатів полягає у можливості використання розроблених алгоритмів і методик у реальних умовах для автоматизації проектування веб-інтерфейсів. Це дозволить зменшити часові та фінансові витрати на розроблення веб-ресурсів, підвищити їх продуктивність

і адаптивність, а також забезпечити кращий користувацький досвід. Результати роботи можуть бути застосовані у веб-дизайні, розробці інтернет-магазинів, інформаційних сайтів та інших типів веб-ресурсів. Крім того, розроблені методи та алгоритми можуть стати основою для подальших досліджень у сфері використання машинного навчання для оптимізації веб-технологій.

Результати кваліфікаційної роботи призначені для використання в навчальному процесі та для проведення наукових досліджень в галузі класифікації даних засобами штучного інтелекту.

Кваліфікаційна робота складається із трьох розділів, висновків, списку використаної літератури та додатків.

У першому розділі було проаналізовано існуючі алгоритми машинного навчання без вчителя для класифікації числових даних показників метрик веб-сайтів.

У другому розділі були наведені алгоритмічні розробки для вибору кращої версії веб-сайту на основі зовнішніх показників базуючись на показниках продуктивності.

У третьому розділі були програмно реалізований модуль класифікації даних показників на основі ансамблевого підходу.

Результати роботи знайшли своє відображення у роботі Всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів «Інтелектуальні комп’ютерні системи та мережі» (Тернопіль, Україна, 25 листопада 2025 року).

# АНАЛІЗ ЗАСОБІВ РОЗРОБКИ ВЕБ-САЙТІВ

## 1.1 Аналіз типів сайтів

Сучасний веб-простір характеризується величезною різноманітністю веб-сайтів, кожен з яких виконує певну функцію та відповідає на конкретні потреби користувачів. Аналіз типів сайтів є важливим етапом для формування комплексного розуміння предметної області дослідження, що дозволяє виділити основні категорії веб-ресурсів, визначити їх ключові характеристики та особливості. Це, у свою чергу, сприяє розробленню оптимальних алгоритмів для автоматизації створення елементів веб-інтерфейсів [1].

Загалом веб-сайти можна класифікувати за декількома критеріями: метою використання, типом контенту, цільовою аудиторією та технічними характеристиками. Основні категорії сайтів включають інформаційні, корпоративні, електронні магазини, розважальні, соціальні мережі, персональні блоги та портфоліо, освітні ресурси та інтерактивні сервіси. Кожна з цих категорій має свої специфічні особливості, що впливають на їх дизайн, функціональність і швидкодію [2].

Інформаційні сайти є одним із найпоширеніших типів веб-ресурсів, які надають доступ до різноманітної інформації, зокрема новин, статей, довідкових матеріалів тощо. Такі сайти орієнтовані на швидкий пошук і зручне сприйняття інформації користувачами, тому їх дизайн зазвичай включає чітку структуру, логічну навігацію та акцент на текстовому контенті. Висока швидкодія є важливим критерієм для інформаційних ресурсів, оскільки велика кількість сторінок та користувачів потребують оптимізованих алгоритмів завантаження.

Корпоративні сайти створюються для представлення компаній, їхніх продуктів і послуг. Основними елементами таких сайтів є сторінки з інформацією про компанію, її історію, місію, перелік послуг, контактні дані та новини. У дизайні корпоративних сайтів зазвичай акцентується на брендовій

ідентичності, що відображається у виборі кольорової гами, шрифтів та графічних елементів. Крім того, важливими аспектами є забезпечення зручної навігації та адаптивності під різні пристрої.

Електронні магазини, або e-commerce сайти, є ще однією популярною категорією. Вони орієнтовані на продаж товарів і послуг через інтернет, тому основна увага приділяється функціональності, зокрема інтеграції платіжних систем, організації каталогу товарів, пошуку та фільтрації, а також можливостям взаємодії з клієнтами через відгуки, чати та інші засоби комунікації [3]. Дизайн електронних магазинів має бути інтуїтивно зрозумілим, щоб мінімізувати зусилля користувача при оформленні замовлення. Приклад електронного магазину можна побачити на рисунку 1.1.

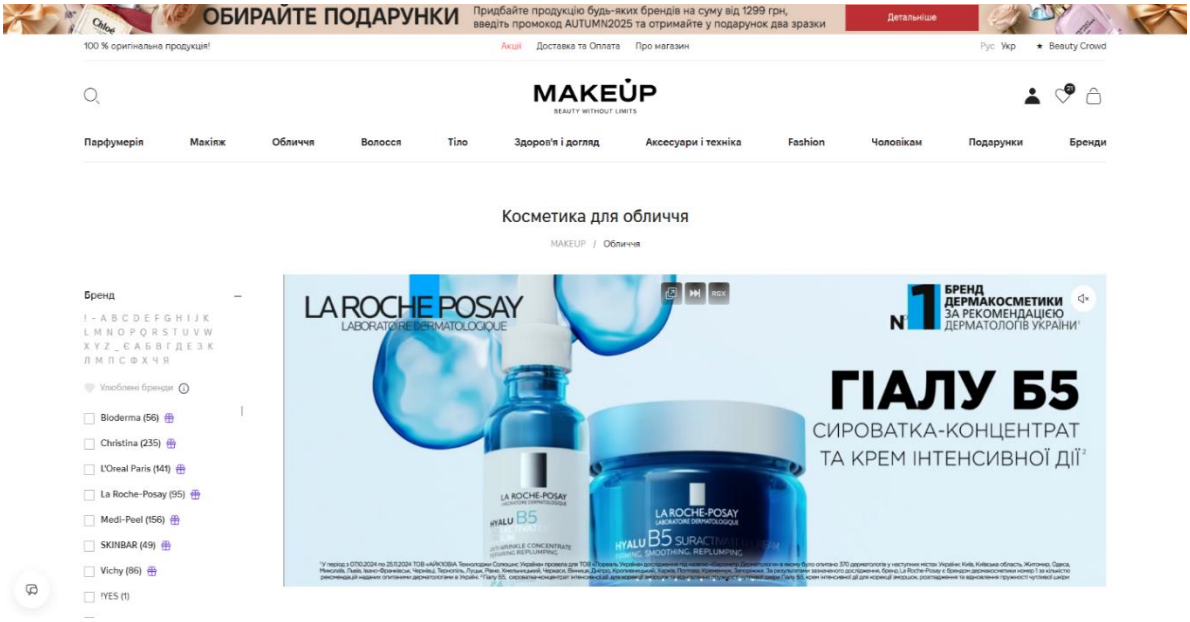


Рисунок 1.1 – Типовий приклад інтерфейсу електронного магазину

Розважальні сайти забезпечують користувачам доступ до контенту, створеного для відпочинку та розваг. Це можуть бути онлайн-кінотеатри, музичні платформи, сайти ігор або гумористичні ресурси. Основними характеристиками таких сайтів є привабливий дизайн, інтерактивні елементи та швидкий доступ до контенту. Оскільки велика кількість користувачів

відвідують такі ресурси одночасно, особливу увагу приділяють оптимізації швидкодії та масштабованості.

Соціальні мережі є одними з найбільш відвідуваних сайтів у світі. Вони надають користувачам можливість спілкуватися, ділитися інформацією, створювати спільноти та будувати мережі контактів [4]. Приклад соціальної мережі зображено нижче на рисунку 1.2. Особливості дизайну соціальних мереж включають простоту інтерфейсу, акцент на користувацькому контенті, адаптивність та інтеграцію з мобільними додатками. Соціальні мережі також активно використовують алгоритми машинного навчання для персоналізації контенту, рекомендацій та аналізу поведінки користувачів.

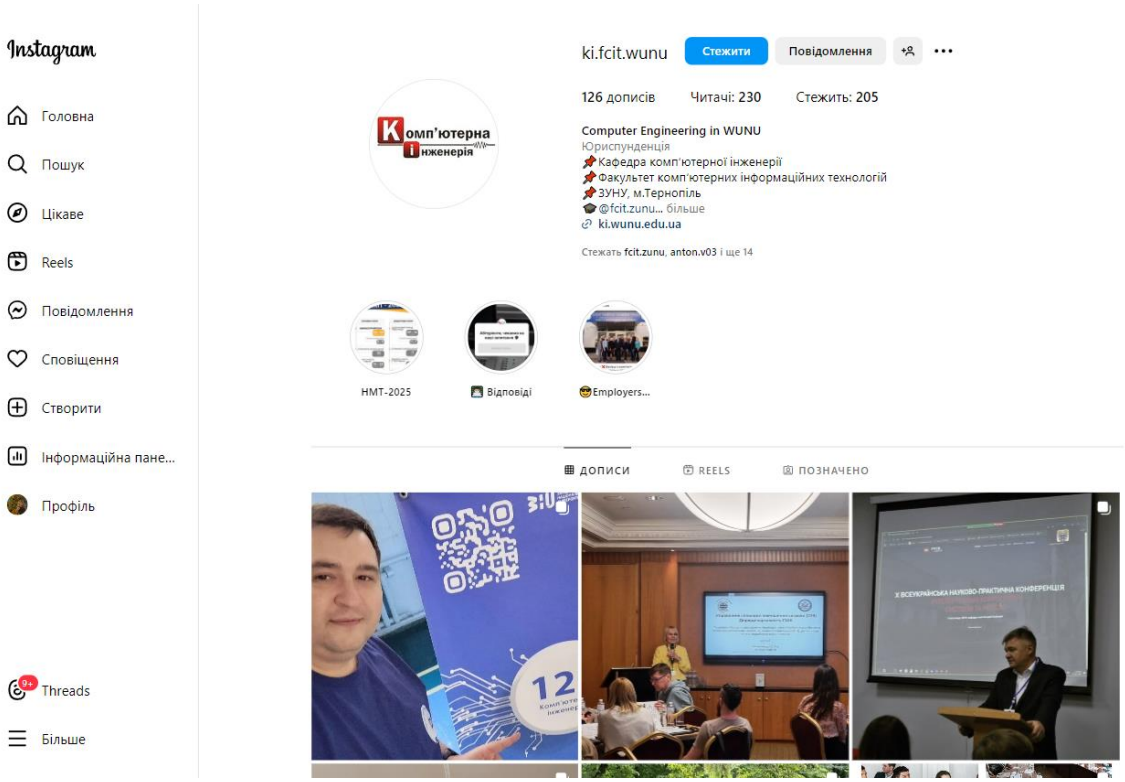


Рисунок 1.2 – Сторінка Instagram як приклад платформи соціальної взаємодії

Персональні блоги та портфоліо створюються для представлення індивідуальних досягнень, творчих робіт або ідей. Основний акцент у дизайні таких сайтів робиться на унікальності та самовираженні, що забезпечує

виділення серед інших ресурсів. Такі сайти часто мають мінімалістичний дизайн, що дозволяє користувачам зосередитись на контенті.

Освітні ресурси є важливим інструментом для навчання та розвитку. Вони включають онлайн-курси, відеоуроки, електронні бібліотеки та інші засоби навчання. Основні вимоги до таких сайтів — це зручність доступу до матеріалів, інтерактивність та підтримка користувачів у процесі навчання. Дизайн освітніх ресурсів зазвичай орієнтований на функціональність і мінімалізм, щоб забезпечити ефективне використання матеріалів.

Інтерактивні сервіси, такі як калькулятори, онлайн-карти чи платформи для спільної роботи, є ще однією важливою категорією веб-ресурсів. Основними характеристиками таких сайтів є складна функціональність, інтеграція з іншими сервісами та високий рівень інтерактивності. Дизайн інтерактивних сервісів має бути інтуїтивно зрозумілим і зручним, оскільки їх використання часто пов’язане з виконанням конкретних завдань.

Усі розглянуті типи сайтів мають свої унікальні особливості, що впливають на процес їх розроблення та подальшу експлуатацію. Аналіз цих характеристик дозволяє визначити ключові аспекти, які слід враховувати при створенні алгоритмів для автоматизації процесу формування елементів веб-інтерфейсів. Зокрема, важливо враховувати специфіку кожного типу сайту, його цільову аудиторію, технічні вимоги та функціональні особливості, щоб забезпечити ефективність і зручність використання кінцевого продукту.

## 1.2 Аналіз засобів штучного інтелекту

Засоби штучного інтелекту (ШІ) є ключовими інструментами, які трансформують підхід до вирішення складних задач в різних сферах, включаючи розробку веб-інтерфейсів. Одним із основних напрямів використання ШІ є алгоритми класифікації, які дозволяють аналізувати великі

обсяги даних і на основі цього створювати моделі для прийняття рішень [5]. Класифікація передбачає розподіл об’єктів або явищ за певними категоріями на основі їхніх характеристик. Наприклад, нейронні мережі, які є основою багатьох сучасних алгоритмів ШІ, використовуються для навчання моделей, що можуть передбачати категорію сайту залежно від його характеристик, таких як швидкодія, кількість елементів, інтерактивність тощо. Візуальне представлення роботи алгоритмів класифікації зображено на рисунку 1.3.

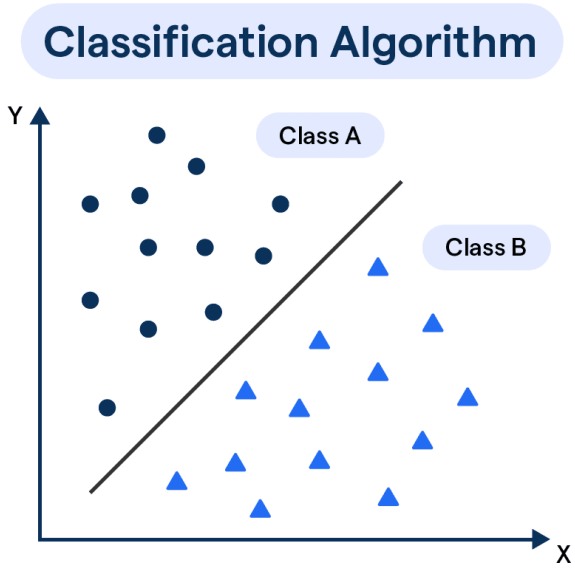


Рисунок 1.3 – Алгоритм класифікації

Крім класифікації, важливим інструментом є кластеризація, яка дозволяє групувати дані за схожими ознаками без попереднього визначення категорій [6]. Графічне представлення алгоритму кластеризації наведено на рисунку 1.4. Алгоритми кластеризації, такі як k-середніх (k-means) або ієрархічна кластеризація, знаходять своє застосування в аналізі користувацької поведінки на сайтах, визначенні найбільш відвідуваних розділів або оптимізації структури сторінки для підвищення її зручності. Наприклад, аналіз метрик PageSpeed Insights може бути використаний для кластеризації сайтів за рівнем їхньої продуктивності, що дозволяє виділити спільні риси для подальшого навчання моделей ШІ.

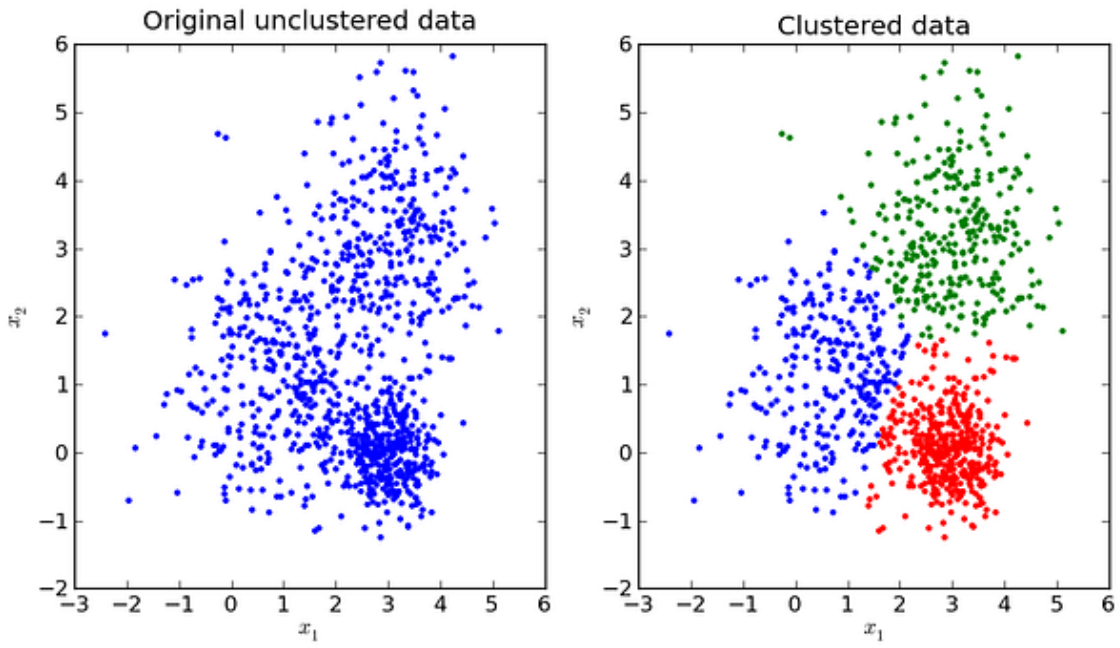


Рисунок 1.4 – Алгоритм кластеризації

Не менш важливим аспектом є використання нечіткої логіки, яка дозволяє моделювати процеси, що мають високу ступінь невизначеності. Нечітка логіка забезпечує можливість врахування різних параметрів, які можуть мати невизначені або розмиті межі, таких як якість підключення до інтернету чи тип пристрою користувача. Це дозволяє створювати адаптивні моделі, які підлаштовуються під конкретні умови та забезпечують високу точність результатів. Наприклад, на основі нечіткої логіки можна створювати системи, які визначають оптимальний рівень складності веб-інтерфейсу залежно від технічних параметрів.

Серед інших засобів ШІ, які заслуговують на увагу, варто виділити алгоритми глибокого навчання. Глибокі нейронні мережі здатні знаходити складні взаємозв’язки між даними, що робить їх незамінними для завдань аналізу великих наборів даних. Наприклад, convolutional neural networks (CNN) часто використовуються для обробки зображень, що може бути корисним при аналізі контенту сайтів, які містять велику кількість графічних елементів. Схематичне зображення принципів роботи глибоких нейронних мереж наведено на рисунку 1.5.

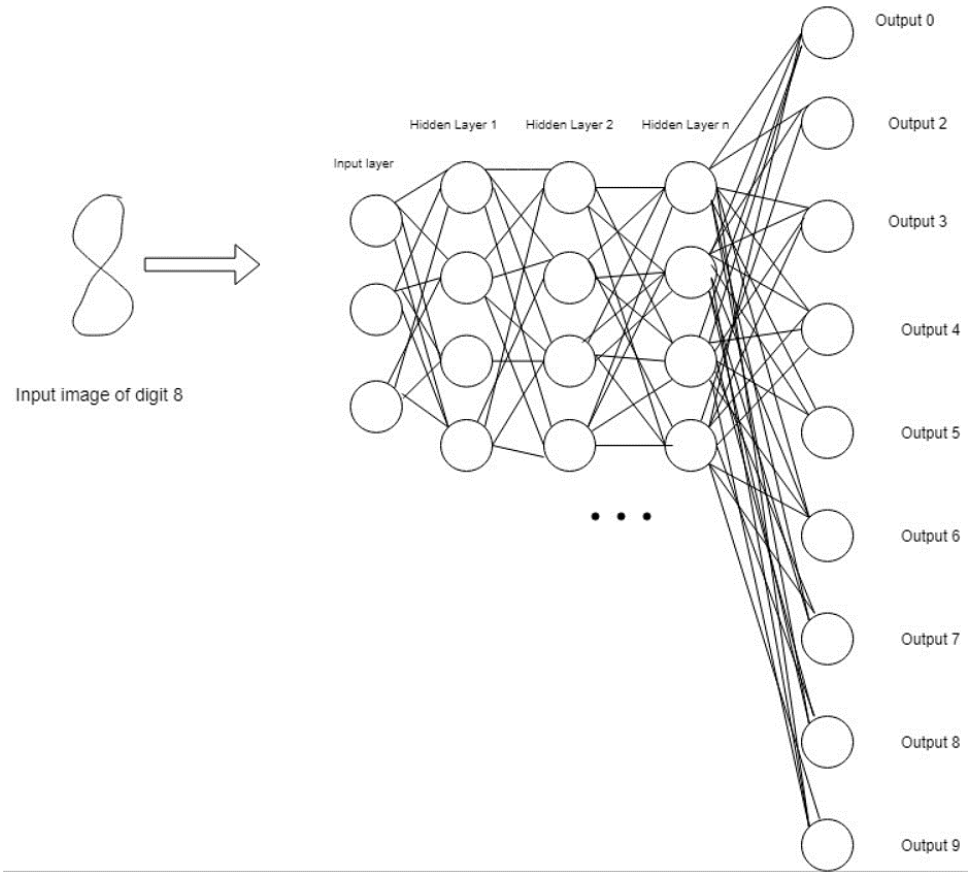


Рисунок 1.5 – Алгоритм глибокого навчання

Методи машинного навчання, такі як supervised learning, unsupervised learning та reinforcement learning, також мають важливе значення для розробки веб-інтерфейсів. У контрольованому навчанні (supervised learning) моделі створюються на основі позначених даних, що дозволяє досягти високої точності в задачах класифікації або регресії. Неконтрольоване навчання (unsupervised learning) надає можливість знаходити приховані закономірності в даних без наявності позначених прикладів [7]. Навчання з підкріпленням приводить перспективи для створення адаптивних веб-інтерфейсів, які вдосконалюються з часом на основі зворотного зв’язку від користувачів.

Для ефективного використання ШІ необхідно формувати якісні набори даних, які служать основою для навчання моделей [8]. У контексті даної роботи формування такого датасету базується на метриках PageSpeed Insights, які надають детальну інформацію про продуктивність сайтів. Використовуючи

ці дані, можна створювати моделі, які зможуть точно визначати категорію складності веб-інтерфейсу.

Засоби штучного інтелекту також дозволяють реалізувати алгоритми обробки природної мови (NLP), які корисні для аналізу текстового контенту на сайтах. Наприклад, NLP може використовуватися для визначення відповідності текстів ключовим запитам користувачів, що дозволяє підвищити релевантність результатів пошуку або запропонувати персоналізовані рекомендації. Це особливо важливо для сайтів, які містять великий обсяг текстової інформації, такої як блоги чи онлайн-магазини.

Іншим важливим аспектом є використання алгоритмів оптимізації, які допомагають автоматизувати процеси прийняття рішень [9]. Генетичні алгоритми та алгоритми мурашиних колоній є прикладами методів оптимізації, які можуть застосовуватися для вирішення складних задач, таких як оптимізація структури сайту або визначення найкращих маршрутів для користувачів. Ці алгоритми дозволяють знаходити рішення, які забезпечують максимальну ефективність та задоволення потреб користувачів.

Засоби ШІ також відкривають можливості для використання рекомендаційних систем, які базуються на аналізі даних про користувачів. Рекомендаційні системи дозволяють створювати персоналізовані пропозиції для користувачів, що значно підвищує їхній досвід взаємодії з сайтом. Наприклад, на основі поведінкових даних можна рекомендувати товари, які найбільше відповідають інтересам користувача.

Розвиток технологій ШІ також сприяє впровадженню віртуальних асистентів і чат-ботів на сайтах. Ці інструменти дозволяють автоматизувати обслуговування клієнтів, відповідаючи на запитання або надаючи допомогу в реальному часі. Використання чат-ботів, які базуються на алгоритмах ШІ, значно скорочує час реагування на запити та підвищує якість обслуговування.

Не менш важливою є інтеграція алгоритмів аналізу поведінки користувачів, які дозволяють створювати динамічні та адаптивні веб-інтерфейси. Наприклад, аналіз даних про тривалість перебування на сторінці

або послідовність переходів між розділами сайту може бути використаний для вдосконалення структури та функціональності веб-інтерфейсу. Такі дані також можуть використовуватися для створення індивідуальних сценаріїв взаємодії з користувачами.

Штучний інтелект відкриває перед веб-розробниками нові горизонти, забезпечуючи засоби для автоматизації складних процесів і створення персоналізованих рішень. Завдяки класифікації стало можливим розподіляти веб-сайти за рівнем складності та пропонувати оптимальні підходи для їхньої оптимізації. Кластеризація дозволяє виявляти приховані взаємозв'язки між метриками продуктивності та поведінкою користувачів, сприяючи покращенню структури сайтів.

Алгоритми глибокого навчання, такі як нейронні мережі, стають незамінними для розробки інноваційних інтерфейсів, здатних адаптуватися до змінних умов. Нечітка логіка дає змогу моделювати ситуації з високим ступенем невизначеності, допомагаючи створювати максимально гнучкі та функціональні рішення. Це дозволяє зробити взаємодію користувача із сайтом більш комфортною та інтуїтивною.

Інтеграція цих технологій підвищує ефективність процесів розробки, створюючи продукти, що відповідають вимогам сучасного цифрового світу. Потенціал штучного інтелекту в цьому контексті практично безмежний, адже він здатний значно прискорити та спростити виконання навіть найскладніших задач.

## 1.3 Аналіз засобів оцінки продуктивності веб-ресурсів

Оцінка ефективності сучасних веб-ресурсів базується на використанні інструментів, які дозволяють глибоко аналізувати їхню продуктивність і зручність для користувачів [10]. В умовах зростаючої конкуренції та

постійного підвищення вимог користувачів до якості роботи сайтів, такі інструменти стають невід'ємною частиною процесу розробки й підтримки веб-ресурсів. Вони не тільки забезпечують можливість оцінки різних аспектів швидкодії, але й допомагають виявляти вузькі місця, що впливають на користувацький досвід, і пропонують рекомендації для їх усунення. Зокрема, популярні сервіси, як-от Google PageSpeed Insights, WebPageTest та Lighthouse, надають детальні метрики для аналізу, сприяючи створенню швидких і зручних сайтів.

Одним із найпоширеніших засобів є PageSpeed Insights, який надає детальну інформацію про продуктивність як десктопної, так і мобільної версій веб-сайту. Візуальне представлення інтерфейсу та ключових функцій цього інструменту зображено на рисунку 1.6.

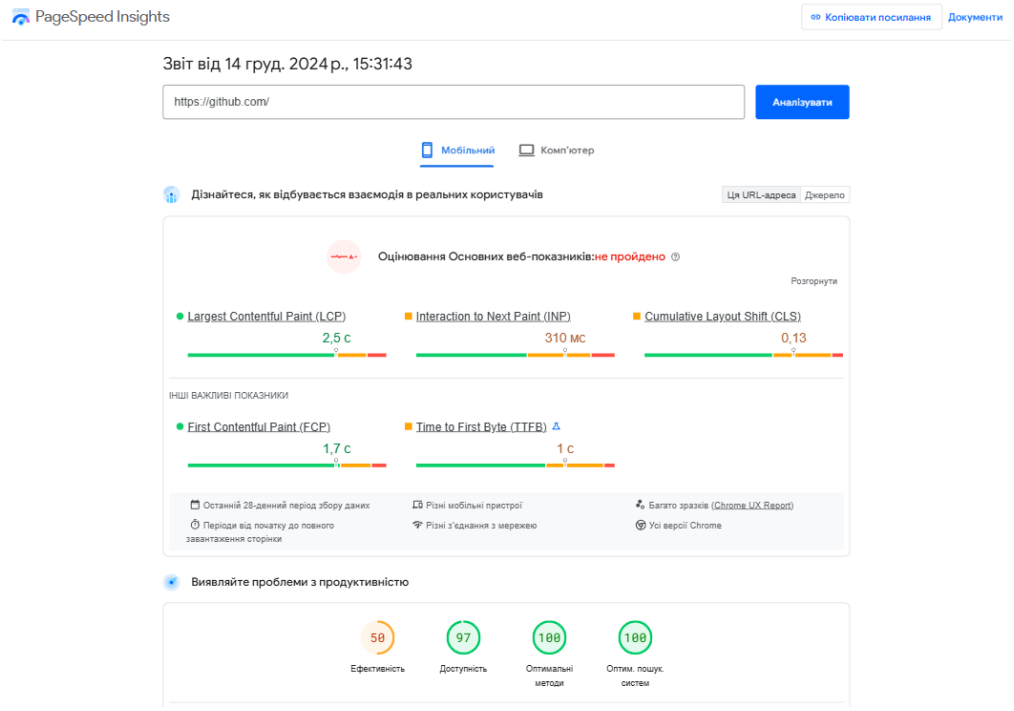


Рисунок 1.6 – Інструмент Google PageSpeed Insights

Основні метрики, що оцінюються цим інструментом, включають швидкість завантаження сторінки, час до першого малювання (First Paint) та час до першого інтерактивного елемента (First Interactive). Крім того,

PageSpeed Insights пропонує рекомендації щодо підвищення продуктивності, наприклад, оптимізація зображень, мінімізація JavaScript і CSS, а також використання кешування браузера. Ці рекомендації ґрунтуються на детальному аналізі поточного стану сайту, що дозволяє визначити найбільш проблемні аспекти та розробити план дій для їх усунення. Водночас оцінка продуктивності мобільних версій враховує такі аспекти, як швидкість підключення до мережі та енергоефективність, що є важливими для користувачів із повільними інтернет-з'єднаннями.

Ще одним популярним інструментом є WebPageTest, який дозволяє проводити глибокий аналіз завантаження веб-сторінок, зокрема, розподіляючи цей процес на окремі етапи [11]. Візуальне представлення інтерфейсу WebPageTest або результати аналізу наведено на рисунку 1.7.

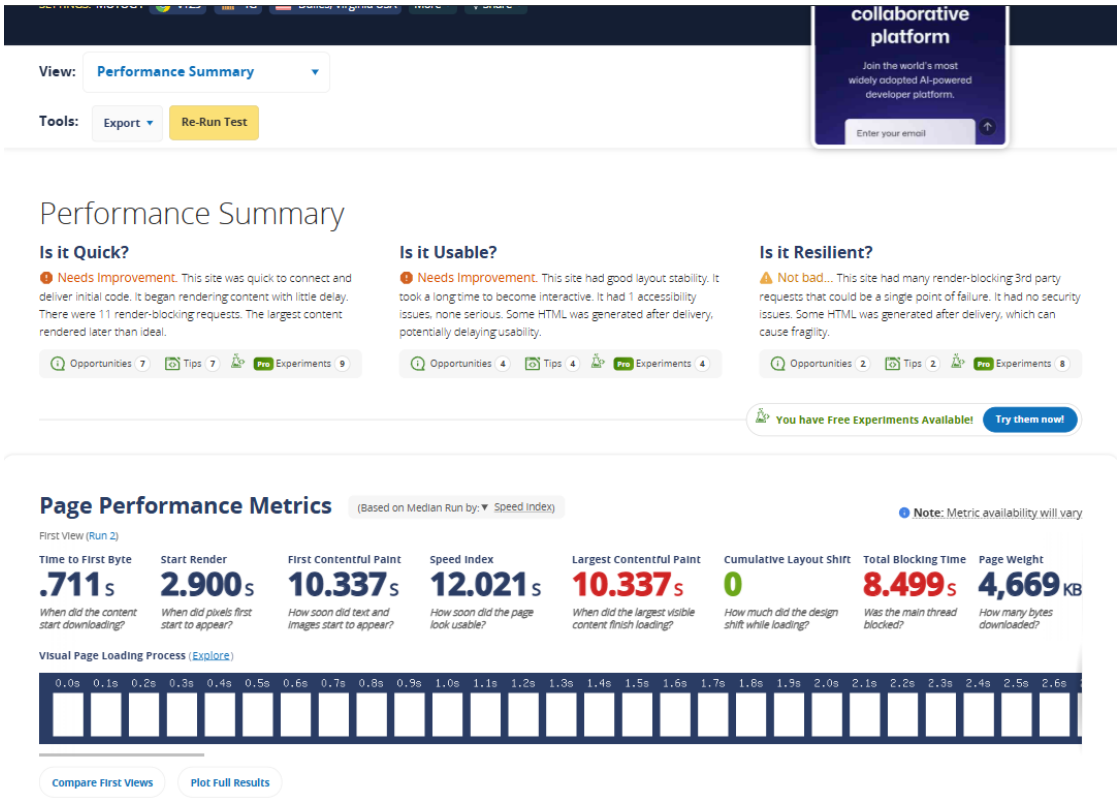


Рисунок 1.7 – Інструмент WebPageTest

WebPageTest надає розширені можливості для тестування, такі як вибір місця розташування сервера, емуляція швидкості мережі та використання

різних браузерів. Основні метрики, що аналізуються цим інструментом, включають час до завантаження першого байту (Time to First Byte, TTFB), загальний час завантаження сторінки, а також обсяги завантажуваних даних. Крім того, WebPageTest дозволяє створювати знімки екрана під час завантаження, що допомагає візуалізувати процес і визначити, які елементи сповільнюють швидкодію. Завдяки цьому інструменту розробники отримують змогу не лише визначати вузькі місця в продуктивності, але й експериментувати з різними стратегіями оптимізації.

Lighthouse, інтегрований у браузер Google Chrome, пропонує комплексний підхід до оцінки продуктивності та якості веб-ресурсів. Візуальне представлення прикладу звіту Lighthouse зображено на рисунку 1.8.

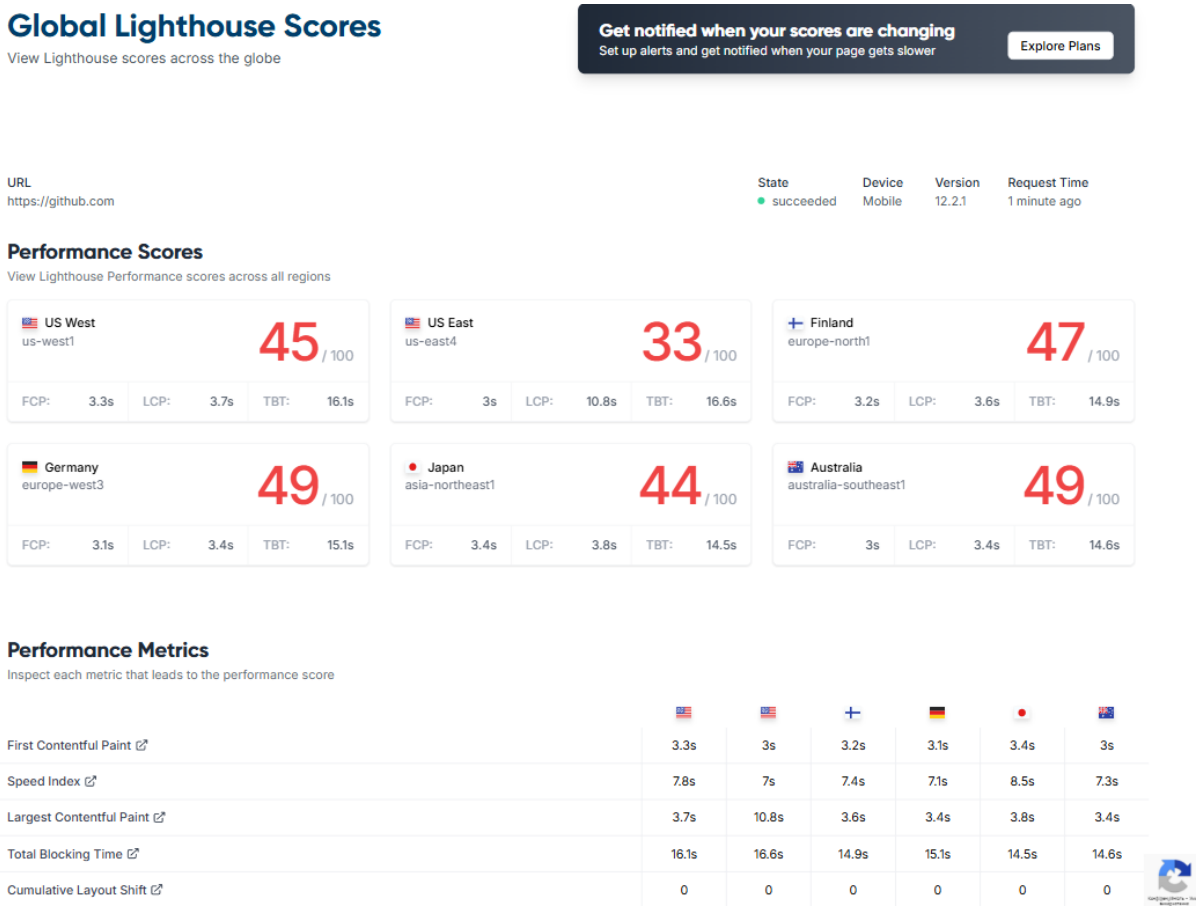


Рисунок 1.8 – Інсутрумент Lighthouse

Цей інструмент дозволяє аналізувати сайти за кількома напрямками, включаючи продуктивність, доступність, SEO та дотримання найкращих практик веб-розробки. Lighthouse надає розширені метрики, такі як Largest Contentful Paint (LCP), що відображає час завантаження найбільшого видимого елемента, і Cumulative Layout Shift (CLS), який вимірює стабільність візуального контенту під час завантаження. Завдяки цим метрикам розробники можуть виявляти проблеми, які впливають на сприйняття користувачами веб-ресурсу, і впроваджувати відповідні покращення. Крім того, Lighthouse генерує детальний звіт з рекомендаціями, які допомагають усувати виявлені недоліки. Особливістю цього інструменту є його орієнтація на сучасні стандарти веб-розробки, що робить його незамінним для створення адаптивних і швидкодіючих сайтів.

Додатковим інструментом для оцінки продуктивності є Pingdom, який дозволяє здійснювати моніторинг доступності та швидкодії веб-ресурсів у реальному часі. Візуальне представлення функціоналу Pingdom, включно з його інтерфейсом, зображено на рисунку 1.9.

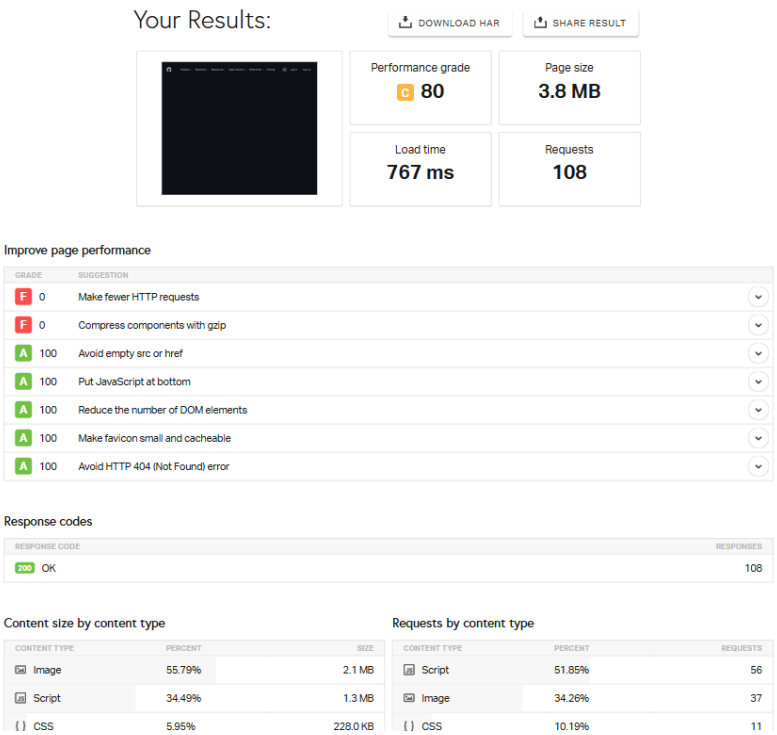


Рисунок 1.9 – Інструмент Pingdom

Pingdom надає розробникам дані про затримки в роботі серверів, час відгуку на запити та доступність веб-ресурсу з різних точок світу. Інструмент також пропонує можливості для створення інтерактивних графіків і звітів, які допомагають краще зрозуміти динаміку продуктивності сайту. Наприклад, розробники можуть аналізувати вплив змін у конфігурації сервера на час завантаження сторінки або оцінювати ефективність використання CDN.

Важливим аспектом використання інструментів оцінки продуктивності є розуміння взаємозв'язку між метриками, які вони надають, і реальним користувацьким досвідом. Наприклад, показник First Contentful Paint (FCP) демонструє, наскільки швидко користувач може побачити перший елемент контенту, тоді як Time to Interactive (TTI) вказує на момент, коли сторінка стає повністю функціональною. Поєднання цих метрик дозволяє отримати цілісне уявлення про продуктивність сайту та визначити області, які потребують покращення.

Крім того, сучасні інструменти оцінки продуктивності часто інтегруються з платформами для моніторингу, такими як New Relic або Datadog, що забезпечує розширені можливості для аналізу [12]. Наприклад, ці платформи дозволяють автоматично збирати дані про продуктивність сайту, створювати дашборди з ключовими показниками та отримувати сповіщення про можливі проблеми. Завдяки цьому розробники можуть оперативно реагувати на зміни в продуктивності та підтримувати високий рівень якості веб-ресурсу.

Важливо також враховувати, що різні інструменти оцінки продуктивності мають свої переваги та обмеження, які слід враховувати під час їхнього використання. Наприклад, PageSpeed Insights орієнтований на швидкий аналіз і надання рекомендацій, тоді як WebPageTest пропонує більше можливостей для глибокого дослідження. Lighthouse, своєю чергою, забезпечує комплексний підхід, включаючи аналіз доступності та SEO, що робить його ідеальним вибором для всебічної оцінки сайту.

Узагальнюючи, аналіз засобів оцінки продуктивності веб-ресурсів є важливим кроком для забезпечення їхньої ефективності та конкурентоспроможності. Інструменти, такі як PageSpeed Insights, WebPageTest, Lighthouse і Pingdom, надають розробникам потужні можливості для діагностики та оптимізації, допомагаючи створювати швидкі, адаптивні та зручні для користувачів сайти. Завдяки цим інструментам веб-розробники можуть не лише підвищувати якість своїх продуктів, але й забезпечувати їхнє довгострокове успішне функціонування в умовах високої конкуренції.

## 1.4 Висновки і постановка задачі

На основі аналітичного підходу здійснено аналіз типів веб-сайтів, алгоритмів штучного внтелекту для аналізу даних та аналіз засобів оцінки продуктивності веб-сайтів.

Для реалізації поставленої мети необхідно реалізувати такі завдання:

- провести порівняльний аналіз підходів до оцінки продуктивності роботи веб-сайтів, основні метрики для можливості виділення ключових критеріїв;
- провести порівняльний аналіз алгоритмів машинного навчання для класифікації числових даних;
- розробити алгоритм класифікації кількісних характеристик показників метрик продуктивності веб-сайтів на основі методу ансамблів;
- здійснити порівняльний аналіз отриманих результатів та вибір кращої стратегії для підбору алгоритмів класифікації.

# 2 АЛГОРИТМИ ФОРМУВАННЯ ЕЛЕМЕНТІВ ВЕБ-СТОРІНОК

## 2.1 Узагальнений алгоритм формування блоків веб-сторінок

На першому етапі здійснюємо аналіз параметрів роботи, зокрема параметрів що відносяться до характеристик пристрою та зовнішніх параметрів, таких як інтернет-з‘єднання.

Перед завантаженням сторінки відбувається визначення таких параметрів:

- Тип пристрою: (PC, планшет, мобільний)
- Роздільна здатність екрану.
- Швидкість інтернет-з'єднання через Network Information API (navigator.connection.effectiveType).
- Обсяг доступної пам’яті та продуктивність
- Кешування та історія взаємодії: аналіз попередніх сеансів користувача (локальне сховище).

У випадку якщо параметри не вдалось визначити відбувається повторне їх визначення. В іншому випадку здійснюється вказування значень по замовчуванню, стандартний ПК, середня швидкість завантаження, середній розмір пам’яті. За необхідні дані можуть зберігатись у базу даних для подальшого донавчання моделі на основі штучного інтелекту.

Додатковим аспектом аналізу параметрів є оцінка стану браузерного середовища, оскільки продуктивність рендерингу залежить не лише від фізичних характеристик пристрою, а й від поточного стану вкладки. Сюди належить кількість активних розширень браузера, рівень завантаження системи фоновими процесами та наявність активних блокувальників реклами, які змінюють структуру DOM. Також враховуються специфічні браузерні оптимізації, наприклад, підтримка нових API (IntersectionObserver, HTTP/3, Client Hints), можливість апаратного прискорення та наявність експериментальних функцій, що впливають на час рендерингу [13].

Ще одним важливим параметром є визначення реальної пропускної здатності мережі, а не тільки номінального значення effectiveType. В сучасних браузерах частина показників може кешуватися, тому система за необхідності виконує додаткові перевірки — наприклад, вимірює затримку між запитами малих ресурсів або аналізує поведінку TCP рукостискання. Це дозволяє уникнути помилок, коли з'єднання позначене як "4G", але реальна пропускна здатність деградує до рівня 3G через навантаження на мережу чи слабкий сигнал. У результаті формується фактичний профіль швидкодії мережі, який точніше відображає можливості користувача щодо сприйняття складних елементів веб-інтерфейсу.

На наступному етапі відбувається процес підбору парамтерів формування блоків веб-сторінко в залежності від вхідних характеристик. Блок правил сформований на основі попереднього навчання засобами машинного навчання та експерної оцінки фахівцем в галузі веб-дизайну та верстки сайтів.

Якщо завантаження сайту відбувається з персонального комп'ютера і швидкість інтернету є високою, то завантажується повнофункціональна версія сайту з усіма елементами, зокрема великими зображеннями, слайдерами, відео, складною логікою з використанням клієнтського JS та кастомного CSS.

У випадку якщо пристрій з якого заходять користувачі є мобільним і швидкість інтернет з'єднання є високою або середньою, то відбувається завантаження мобільної версії веб-сайту з полегшеним навантаженням у порівнянні з повнофункціональним сайтом.

У випадку якщо швидкість інтернету дуже низька і до того ж потужності пристрою не дозволяють у повній мірі завантажити повнофункціональний або мобільний сайт, відбувається завантаження легкої версії сайту з наявністю лише базових елементів веб-сторінки.

Можна використовувати ML-модель, яка на основі збережених даних прогнозуватиме найкращу структуру сторінки для кожного користувача [14, 15]. Така модель дозволяє точніше прогнозувати потрібну версію сторінки, орієнтуючись не лише на технічні параметри, але й на типову поведінку

користувача. Якщо система бачить, що користувач переважно відкриває сайт з мобільних пристроїв, модель обирає відповідний профіль продуктивності. Якщо ж історично оптимізована версія працювала повільно на певній групі пристроїв, параметри автоматично коригуються.

Вхідні параметри - Тип пристрою, швидкість інтернету, історія взаємодії, геолокація.

Вихідні параметри - Визначення оптимального компонування сторінки.

Алгоритм - Кластеризація (K-Means) або нейромережа (MLP) для вибору найкращого варіанту.

Запускаємо A/B тестування для оцінки ефективності різних стратегій рендерингу.

Аналізуємо продуктивність та досвід користувача за допомогою Google Lighthouse та Web Vitals.

Оптимізуємо розмір ресурсів (Lazy Loading, WebP-зображення, мінімізація JS/CSS) [16].

Після виводу сторінку користувачу відбувається моніторинг вибору для донавчання вибраної моделі та розширення вибірки.

Додатково існує можливість вибору певних блоків в залежності від типу з’єданння та кінцевого пристрою.

Узагальнений алгоритм наведено на рисунку 2.1.

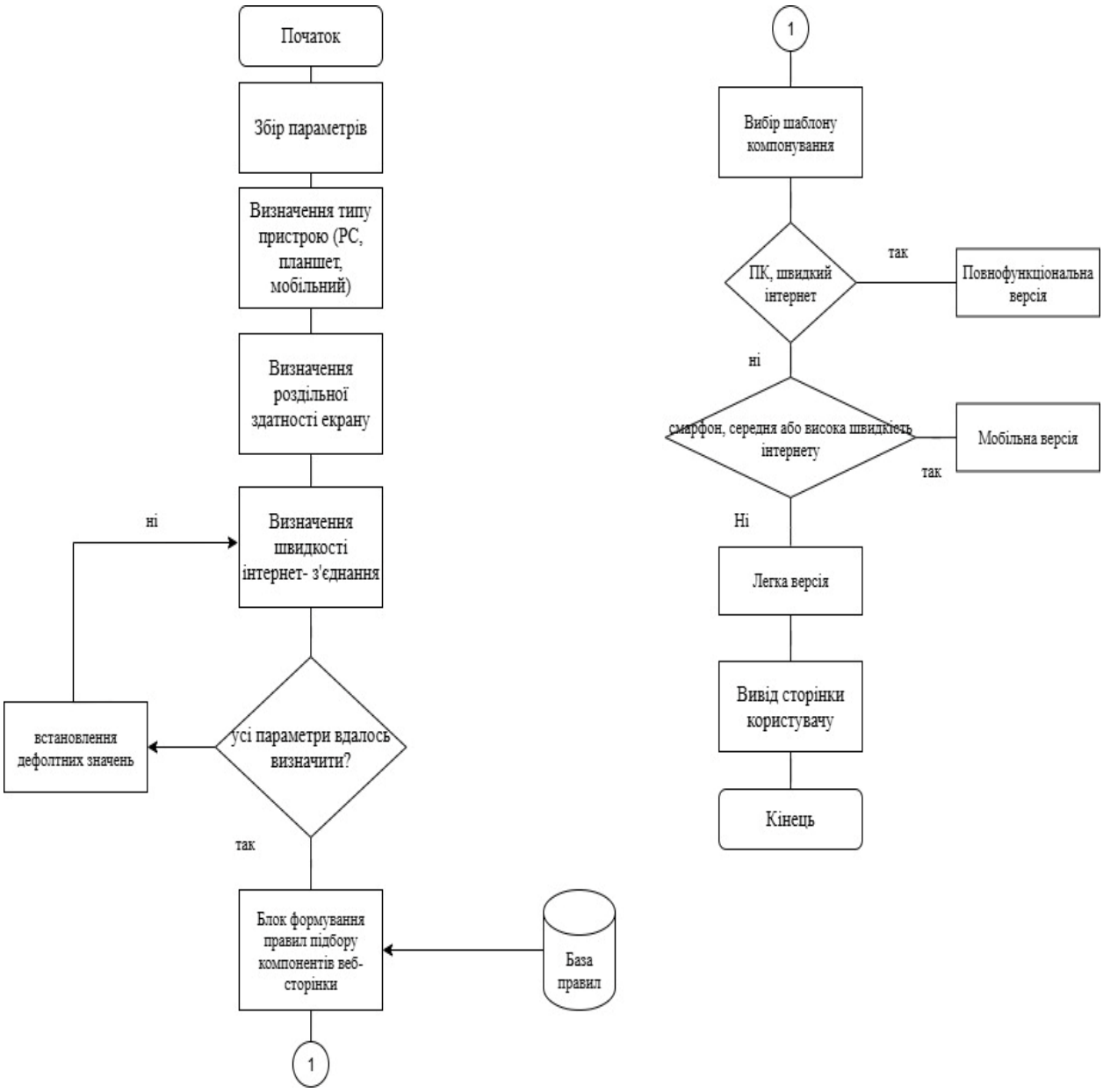


Рисунок 2.1 - Узагальнений алгоритм вибору типу сторінки в залежності від вхідних параметрів

Алгоритм складається з таких кроків:

1. Визначення параметрів інтернет з’єднання, типу пристрою.
2. Визначення роздільної здатності екрану.
3. Встановлення початкових значень при неможливості визначити зовнішні параметри.
4. Визначення типу вебсторінки в залежності від зовнішніх параметрів на основі попередньої класифікації методом ансамблів. В якості

правил підбору вебсторінок використано результати класифікації кількісних показників метрик продуктів на основі показників pagespeed, GTMetrix. Після оцінки характеристик кожного нового сайту відбувається віднесення до одного з трьох класів «легка сторінка» , «середня сторінка», «важка сторінка» та відповідно підвантаження необхідної версії сторінки.

Приклад поділу на 3 класи на основі попередньої класифікації можна сформулювати у вигляді таких правил:

```
First Contentful Paint (FCP)
Largest Contentful Paint (LCP)
IF FCP >2 с та LCP >2.5 с THEN class 1 (Light version)
IF  (FCP > 0.7c & FCP < 2c) & (LCP > 1.0c & LCP < 2c) THEN class
2 (Medium version)
IF  FCP < 0.7c & LCP < 1.0c THEN class 3 (Full version)
```

Якщо швидкість інтернету "2G" або пам'ять менше 2 ГБ, або менше 2 процесорних ядер → Вибираємо "Lite" (полегшена версія сайту).
Якщо пристрій мобільний або з'єднання "3G" → Вибираємо "Optimized" (оптимізована версія). В іншому випадку → Вибираємо "Full" (повноцінна версія сайту).

Приклад псевдокоду наведено нижче:

```
function getRenderingMode() {
    let connection = navigator.connection.effectiveType; // "4g",
"3g", "2g"
    let  device  =  /Mobi|Android/i.test(navigator.userAgent)  ?
"mobile" : "desktop";
    let memory = navigator.deviceMemory || 4; // В гігабайтах
    let cpuCores = navigator.hardwareConcurrency || 2;

    if (connection === "2g" || memory < 2 || cpuCores < 2) {
        return "lite"; // Полегшена версія
    } else if (device === "mobile" || connection === "3g") {
        return "optimized"; // Оптимізована версія
    } else {
        return "full"; // Повноцінна версія
    }
}
let mode = getRenderingMode();
console.log("Режим рендерингу:", mode);
```

Наскрізна логіка алгоритму дозволяє не тільки визначати режим рендерингу, а й змінювати його під час сесії. Наприклад, якщо протягом перегляду сторінки швидкість з’єднання різко падає, система може відключити частину динамічних елементів або зменшити частоту оновлення даних у реальному часі. Це забезпечує стабільність рендерингу та мінімізує ризик зависання інтерфейсу. У зворотній ситуації — коли умови покращуються — система може поступово активувати додаткові елементи, які були вимкнені на початку. Таким чином алгоритм працює не як одноразовий вибір, а як динамічна система адаптації.

## 2.2 Алгоритм кластеризації вхідних показників веб-сайтів

Під час виконання кластеризації вихідні дані подають у вигляді впорядкованої структури — це може бути таблиця або набір числових векторів, залежно від типу задачі. Головне, щоб інформація була представлена так, аби можна було обчислити міру близькості чи подібності між елементами.

У більшості випадків кожен рядок відповідає окремому елементу (спостереженню), кожен стовпець — окремій характеристиці, числовій чи категоріальній.

Якщо маємо $n$ елементів та $m$ характеристик, то структура даних матиме розмір $n \times m$.

Нижче для наочності наведено ілюстративний приклад.

*Об'єкт Перша_Ознака Друга_Ознака Третя_Ознака*
*Object 4.5 7.8 1.1*
*Object 7.4 2.5 5.8*
*Object 9.4 5.6 7.5*

Коли дані для кластеризації подаються у форматі набору числових векторів, кожен об’єкт описується окремим вектором. У цьому випадку маємо список або масив, елементами якого є такі вектори. Приклад такої структури може виглядати так:

```
[
  [4.1, 0.9, 2.7],
  [1.8, 2.4, 3.6],
  [3.0, 1.1, 5.2]
]
```

Загальна схема роботи алгоритму включає такі етапи:

1) Представимо кількісні хаарктеристики веб сайтів у $X_{n\times m}$, де $m$ – порядок показника, і $n$ – значення.

2) Нормалізація даних наведена у наступній формулі:

$$\frac{x_{ij} - \bar{x}_j}{\sigma_j},$$

де $j$ – к-сть значень, $i$ – реалізація змінної $j$.

3) Обчислення кореляційної матриці (*R*).

4) Знаходження значень $\lambda_1$, $\lambda_2$, …, $\lambda_p$ матриці R з рівняння:

$$\det(R - \lambda E) = 0,$$

де $E$ – матриця.

5) Знаходження для кожного значення $\lambda_j$ вектора відповідно до рівняння:

$$(R - \lambda E)\cdot \vec{w} = 0,$$

де $\vec{w}$ – власний вектор.

6) Знаходження комбінації для базових аргументів $y_j$:

$$y_j = w_{1j}x_1 + w_{2j}x_2 + ... + w_{pj}x_p.$$

де $w_{ij}$ – факторні навантаження ознак.

7) Оцінювання ваги кожної головної компоненти та впорядкування їх за зростанням значущості. Значення внеску окремої компоненти визначається за такою формулою:

$$B_j = \frac{\lambda_j}{\lambda_1 + \lambda_2 + ... + \lambda_p},$$

Після скорочення даних наведено функції $p$:

$$X = \{X_1, X_2, ..., X_p\}, \text{ де } p << m.$$

8) Кількість кластерів визначається за допомогою методу «лікоть» (Elbow), що ґрунтується на аналізі значення мінімальної сумарної квадратичної помилки (MSSE) [17-19].

$$K = \min_{c_1,...,c_k} \sum_{i=1}^{n} \min \left\{ \|x_i - c_1\|^2, ..., \|x_i - c_k\|^2 \right\}.$$

9) метод K-means, для знаходження кластериних центрів

$$C_k = \frac{\sum_{x_i \in C_k} x_i}{|c_k|}.$$

10) На наступному етапі відбувається побудова дендрограми:

$$d_{\min}(C_i, C_j) = \min_{\substack{x_i \in C_i \\ x_j \in C_j}} d(x_i, x_j).$$

11) DBScan використовується для поділу на кластери:

$$N_\varepsilon(p) = \{q \in X | dist(p,q) \le \varepsilon\},$$

де $p$ – головна точки, $q$ – $\varepsilon$ - досяжна точка в кластері.

12) Якість кластеризації оцінюємо на основі ранд-індексу за виразом:

$$RI = \frac{TP + TN}{TP + FP + FN + TN},$$

де $TP$ – кількість істинних спрацьовувань;

$TN$ – кількість істинних негативів;

$FP$ – кількість помилкових спрацьовувань;

*FN* – кількість помилкових негативів.

Датасет показників продуктивності сайтів наведено на рисунку 2.2.

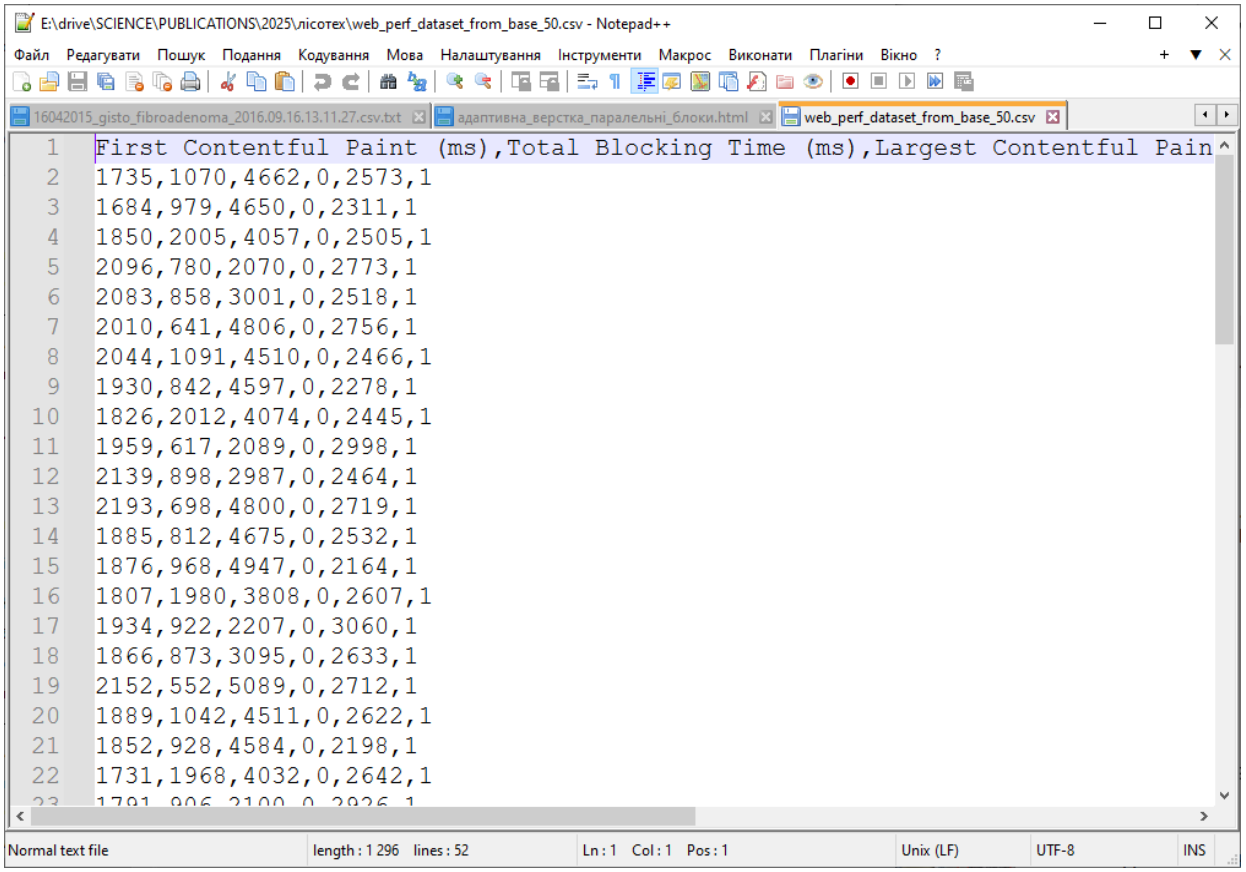


Рисунок 2.2 – Датасет показників продуктивності сайтів

Практичне застосування кластеризації в рамках системи рендерингу передбачає регулярне оновлення датасету та періодичну рекалібровку моделей [20-22]. Це означає, що процес кластеризації повинен бути автоматизований у вигляді пайплайну: збір — очистка — зведення ознак — обчислення компонентів — визначення k — кластеризація — оцінка. Такий підхід дозволяє швидко реагувати на зміну поведінки трафіку та підтримувати релевантність кластерів у часі.

## 2.3 Ансамблевий метод класифікації показників продуктивності веб-сайтів

До основних метрик, які використовувались для оцінки сайтів належить:

Cumulative Layout Shift — показник, який займається вимірюванням сумарного зсуву елементів інтерфейсу під час завантаження.

First Contentful Paint — показник, що відображає час від початку завантаження сторінки до етапу, коли браузер вперше відображає текст.

Total Blocking Time — сумарний час, коли основний потік браузера був заблокований довгими задачами, через що сторінка не могла швидко реагувати на взаємодії користувача.

Speed Index— оцінка швидкості відображення вмісту сторінки, що відображається для користувача.

Приклад сформованого датасету наведено у таблиці 2.1.

Таблиця 2.1 - Приклад сформованого датасету

| First Contentful Paint | Total Blocking Time | Largest Contentful Paint | Cumulative Layout Shift | Speed Index |
|---|---|---|---|---|
| 1735 | 1070 | 4662 | 0 | 2573 |
| 1684 | 979 | 4650 | 0 | 2311 |
| 1850 | 2050 | 4057 | 0 | 2505 |
| 2096 | 780 | 2070 | 0 | 2773 |
| 2083 | 858 | 3001 | 0 | 2518 |

Перед безпосередньою класифікацією дані проходять стандартну підготовку: обробку пропусків, виявлення та корекцію аномалій, нормалізацію числових ознак і кодування категоріальних полів при необхідності. Важливо також збалансувати класи або використати ваги під час навчання, якщо один із статусів значно переважає інші, щоб уникнути зміщення моделі в бік

домінуючого класу. Ці кроки забезпечують коректність відстаневих та ймовірнісних розрахунків у подальших алгоритмах.

Після додаткового опрацювання, кожному запису додається окреме поле «status», який знаходиться в межах від 0…2, що характеризує ступінь завантаженості веб-сайту.

Поле «status» формувалось з урахуванням комбінованого правилу, яке зважує ключові метрики: LCP, FCP та TBT, — таким чином формується узагальнений індекс завантаженості [23-25]. Для валідності ярлика використовувалося A/B порівняння із вручну промаркованими прикладами, а також перевірка узгодженості через крос-валідацію. Це дозволило зменшити шум у цільовому полі і підготувати стабільнішу мішень для ансамблевих методів.

Візуалізація параметру «Total Blocking Time» наведено на рисунку 2.3.

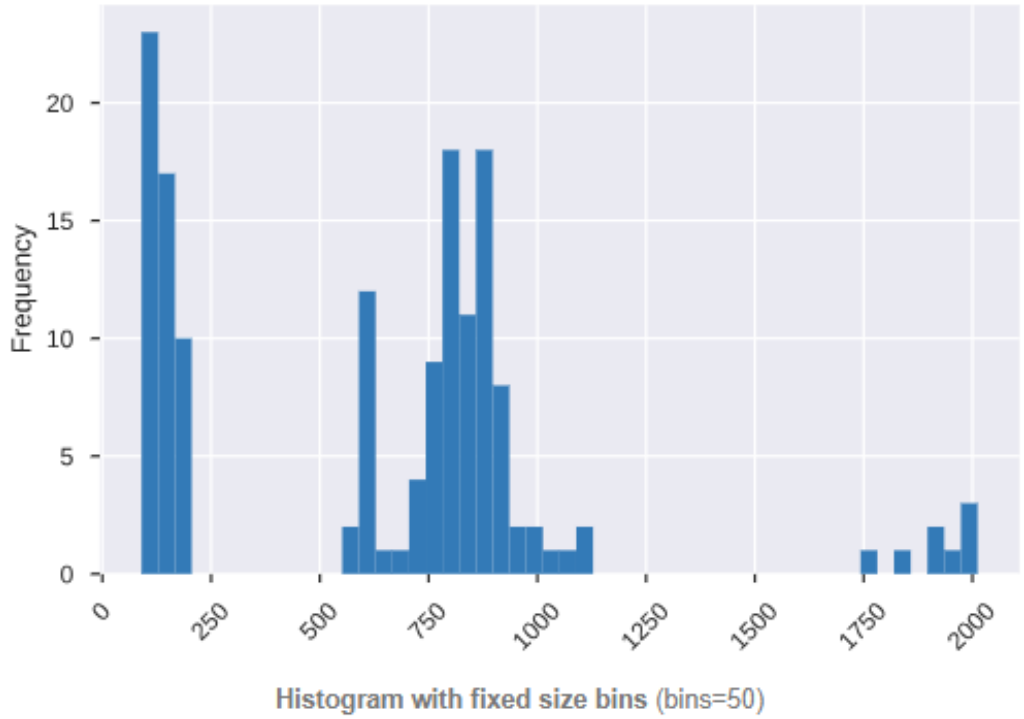


Рисунок 2.3 - Візуалізація параметру «Total Blocking Time»

Візуалізація параметру «Largest Contentful Paint» наведено на рисунку 2.4.

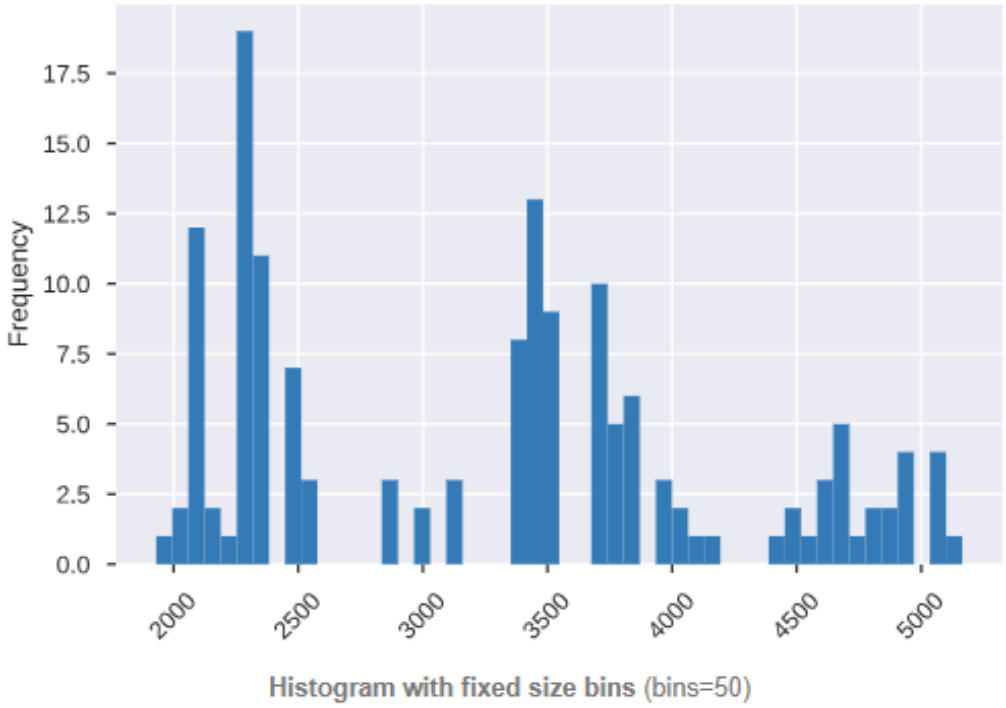


Рисунок 2.4. - Візуалізація параметру «Largest Contentful Paint»

Кореляційну матрицю наведено на рисунку 2.5.


Рисунок 2.5 – Кореляційна матриця

Візуальний аналіз (розподіли, дендрограми, кореляційні матриці) допоміг виявити взаємозв'язки між метриками — наприклад, сильну

кореляцію між LCP і Speed Index, що вказує на спільні джерела затримок. На основі цього було прийнято рішення про попередню агрегацію деяких ознак та зменшення розмірності перед кластеризацією, що поліпшило поведінку моделей і знизило ризик мультиколінеарності. Такі перетворення також спростили інтерпретацію результатів ансамблю.

Алгоритм побудови комітету:

5. Генерування випадкової підвибірки з повторенням розміром $n$ з навчальної вибірки

6. Випадковим чином обираються m предикаторів (ознак) із M

7. Побудова дерева рішень відбуваєтсья шляхом обрання ознак, на основі якої проводиться розбиття, не з усіх M ознак, а лише з m випадково вибраних

8. Розділення ознаки X на два класи $X_i \geq S_i$ та $X_i < S_i$

9. Виміряємо гомогенність у двох нових класах за допомогою критерію Джині

10. Оберемо таке значення «спліт-поінту» $S_i$ ознаки X, для якого досягнуто максимальної гомогенності класу.

11. Дерево будується до повного використання підвибірки без застосування процедури відсікання (на відміну від дерев рішень, створених за алгоритмами типу CART або C4.5).

12. Повертаємося до пункту 1, генеруємо нову вибірку та повторюємо кроки 2–4 для побудови наступного дерева.

13. Класифікація об'єктів здійснюється шляхом голосування: кожне дерево в ансамблі відносить об'єкт до одного з класів, і обирається той клас, який отримав найбільшу кількість голосів від дерев.

Параметри ансамблю (кількість дерев n, кількість предикторів m, глибина дерев) підбиралися експериментально з урахуванням компромісу між точністю і широкомасштабністю. Було застосовано крос-валідацію для вибору робочих значень, при цьому пріоритет надавався стійкості на різних підвибірках даних. Практично, невелике збільшення числа дерев підвищувало

стабільність прогнозів, але не приводило до значного зростання точності після певного порога, що враховано при остаточній конфігурації моделі.

*Hard voting*

Hard voting – це простий алгоритм, який можна використовувати для об’єднання прогнозів кількох класифікаторів. Простота жорсткого голосування робить його популярним вибором для практиків машинного навчання. Жорстке голосування базується на більшості голосів окремих класифікаторів. Якщо дані шумні, ймовірніше, що більшість голосів буде неправильною.

*Soft voting*

М’яке голосування — це алгоритм, який можна використовувати для об’єднання прогнозів кількох класифікаторів за допомогою ймовірності прогнозів. Алгоритм працює так, що спочатку кожен класифікатор призначає ймовірність для кожного класу. Тоді прогноз ансамблю є просто класом із найвищою загальною ймовірністю.

Вибір між жорстким (hard) та м’яким (soft) голосуванням визначався природою даних і потребою в калібровці ймовірностей [26]. За наявності надійних ймовірнісних прогнозів від базових класифікаторів soft voting давало кращі результати, оскільки дозволяло агрегувати прогнози з урахуванням упевненості окремих моделей. У випадках, коли окремі моделі були по-різному відчутні до шуму, застосовувалось hard voting як більш стійкий і простий механізм, або комбінувався з обмеженням довіри до окремих дерев.

Оцінка моделі здійснювалась комплексно: окрім ранд-індексу були використані силуетний коефіцієнт для кластеризації і метрики precision/recall для перевірки класифікації статусу [27, 28]. Також виконана перевірка узагальнюваності на відкладених датасетах, щоб уникнути перенавчання. У підсумку ансамблевий підхід показав кращу стійкість до шуму і більш передбачувану поведінку на нових даних порівняно з одиночними моделями.

## 2.4 Висновки до розділу 2

На основі алгоритмічного підходу розроблено узагальнений алгоритм формування блоків веб – сторінок, який дозволяє завантажувати максимально продуктивною версію.

На основі алгоритмічного підходу розроблено алгоритм кластеризації вхідних показників веб-сайтів на основі показників pagespeed для розподілу набору даних окремі частини.

Розроблено ансамблевий метод класифікації показників, що дозволяє максимально ефективно використати можливості різних алгоритмів навчання без вчителя.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1 Програмна реалізація модулю кластеризації для розподілу показників продуктивності веб-сайтів

Для ефективного аналізу продуктивності веб-сайтів застосовуються алгоритми кластеризації, що дозволяють групувати сайти за схожими показниками [29]. Це дає змогу виділити основні типи поведінки сторінок та визначити оптимальні стратегії їх оптимізації. Використання різних методів кластеризації забезпечує гнучкість у роботі з даними та підвищує точність подальшої класифікації [30]. Крім того, кластеризація допомагає виявити аномалії та відхилення у продуктивності, що може бути критично важливим для підтримки стабільності та швидкодії веб-сайтів у реальному часі [31].

Метод “лікоть” (Elbow) є одним із класичних підходів для визначення оптимальної кількості кластерів у задачах кластеризації. Його суть полягає в аналізі поведінки функції якості розбиття, зазвичай сумарної квадратичної помилки (Sum of Squared Errors, SSE) або мінімальної сумарної квадратичної помилки (MSSE), у залежності від кількості кластерів k.

При збільшенні k значення SSE монотонно зменшується, оскільки кожен новий кластер дозволяє точніше наблизити центроїди до відповідних об’єктів. Однак після певного порогу приріст якості суттєво зменшується. На графіку “кількість кластерів – значення SSE” ця точка проявляється як характерний перегин, що за формою нагадує “лікоть”.

Характеристику датасету, використану для аналізу наведено на рисунку 3.1.

| | FCP | TBT | SI | LCP |
|---|---|---|---|---|
| **count** | 100.000000 | 100.000000 | 100.000000 | 100.000000 |
| **mean** | 1.675452 | 0.497832 | 6.140811 | 2.473447 |
| **std** | 0.743724 | 0.293111 | 2.347410 | 0.880357 |
| **min** | 0.513805 | 0.006952 | 2.040493 | 1.043180 |
| **25%** | 0.983002 | 0.242005 | 4.215039 | 1.748845 |
| **50%** | 1.660356 | 0.505625 | 6.500439 | 2.529155 |
| **75%** | 2.325508 | 0.766184 | 8.018936 | 3.207333 |
| **max** | 2.967217 | 0.985650 | 9.920431 | 3.971515 |

Рисунок 3.1 - Характеристику датасету

Elbow метод наведено на рисунку 3.2.

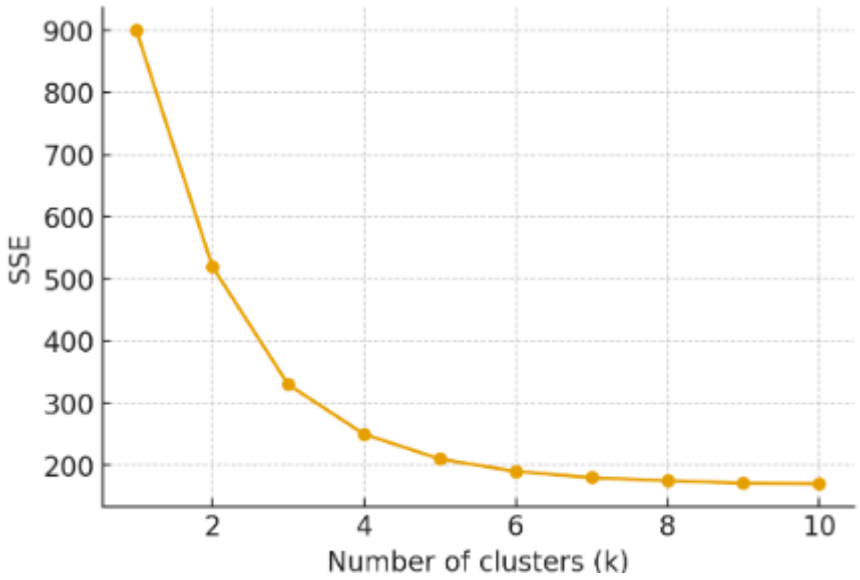


Рисунок 3.2 – Приклад Elbow методу

Крім графічного визначення оптимальної кількості кластерів, під час аналізу враховувалася також варіація всередині кластерів. Менші значення дисперсії вказують на більш однорідні групи, що підвищує точність подальшої класифікації веб-сайтів за рівнем продуктивності.

Результат кластеризації за критеріями LCP та SI наведено на рисунку 3.3.

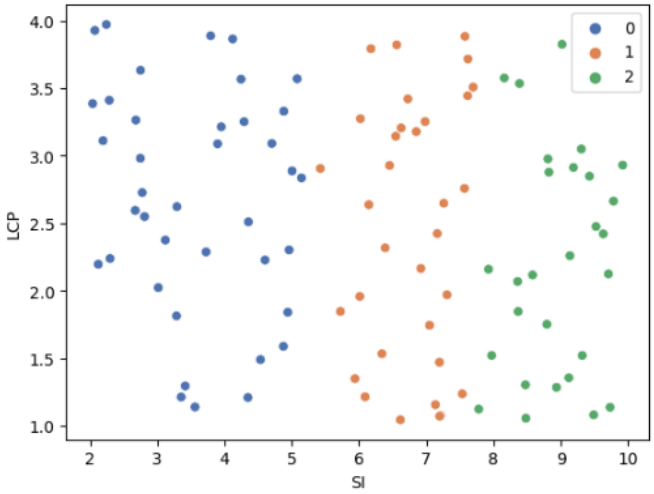


Рисунок 3.3 – Застосування алгоритму K-means для кластеризації на основі показників LCP та SI

На рисунку 3.4 подано результат кластеризації за показниками LCP та SI, виконаної методом Agglomerative Clustering.

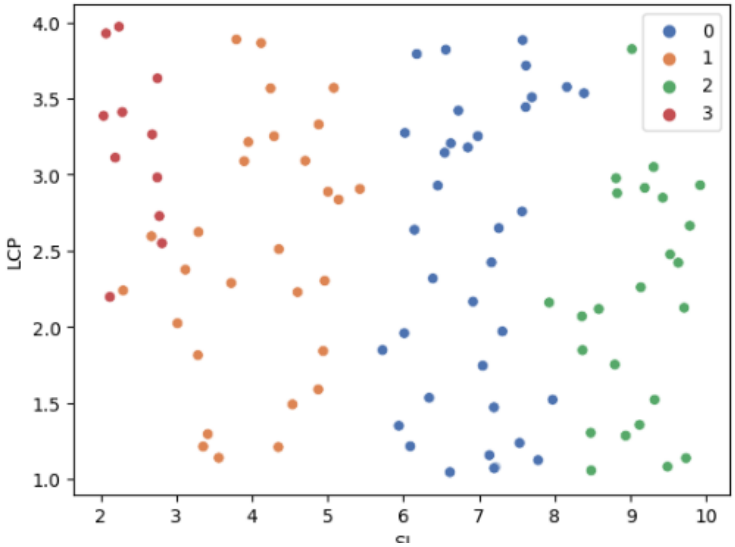


Рисунок 3.4 - Результат кластеризації за критеріями LCP та SI методом Agglomerative Clustering

Спостереження показали, що метод K-means добре виділяє основні групи за LCP та SI, але менш чутливий до невеликих аномалій у даних. Тому результати слід інтерпретувати з урахуванням можливого впливу шуму.

Birch Clustering дозволяє ефективно працювати з великими обсягами даних і швидко формувати кластери. Однак у разі нерівномірного розподілу даних деякі менші кластери можуть об’єднуватися із сусідніми, що потребує додаткової перевірки перед інтеграцією в алгоритм класифікації.

В цілому, проведена кластеризація підтвердила доцільність використання двох або трьох груп для подальшої обробки та вибору оптимального варіанту веб-інтерфейсу залежно від характеристик пристрою та мережевого підключення.

Додатково проведено порівняльний аналіз стабільності кластерів між методами K-means, Agglomerative і Birch. Було помічено, що Agglomerative більш чутливий до локальних варіацій даних, тоді як K-means і Birch демонструють кращу узгодженість при повторному запуску алгоритмів із різними початковими умовами [32, 33]. Це дозволяє оцінити надійність сформованих кластерів та вибрати найбільш стабільний підхід для інтеграції в модуль класифікації.

## 3.2 Розроблені версії графічного інтерфейсу

Створення веб-сайту, який демонструє три рівні складності дизайну та навантаженості, передбачає поєднання сучасних підходів до розробки інтерфейсів, креативного візуального стилю та технічної реалізації за допомогою базових, але потужних інструментів веб-розробки. Основна мета полягала в тому, щоб розробити лендінгову сторінку, що візуально і функціонально відповідатиме сучасним стандартам, та одночасно відобразити відмінності між варіантами сайту з високим, середнім і низьким рівнями складності. Обрана тематика — "Розумна клініка майбутнього", яка поєднує медичні технології, штучний інтелект і роботизовані протези, — дозволила створити унікальний приклад сучасного дизайну, що підкреслює

футуристичний характер і водночас залишається зрозумілим для користувачів. Основою верстки було обрано HTML та CSS, оскільки для цього завдання не було необхідності у використанні складних фреймворків або серверних технологій. Проте для реалізації деяких інтерактивних елементів були застосовані прості JavaScript-скрипти, які дозволили підвищити рівень динамічності сайту та додати плавні ефекти.

Під час розробки основна увага була зосереджена на створенні трьох рівнів дизайну. Високий рівень навантаженості передбачав наявність великої кількості мультимедійних компонентів, таких як відео на фоні головного блоку, анімовані елементи інтерфейсу, плавні переходи між секціями та інтерактивні блоки, що реагують на дії користувача. Це дозволило створити враження технологічної інноваційності, що ідеально відповідало тематиці сайту. Такий підхід вимагав ретельної оптимізації ресурсів, адже мультимедійні файли суттєво впливають на швидкість завантаження сторінки. Для цього використовувалися формати сучасних зображень (WebP) та відео з адаптивним налаштуванням роздільної здатності, що забезпечувало баланс між якістю та продуктивністю. Крім того, для створення анімацій було застосовано CSS-трансформації та ключові кадри, які дають можливість реалізовувати плавні рухи без значного навантаження на процесор.

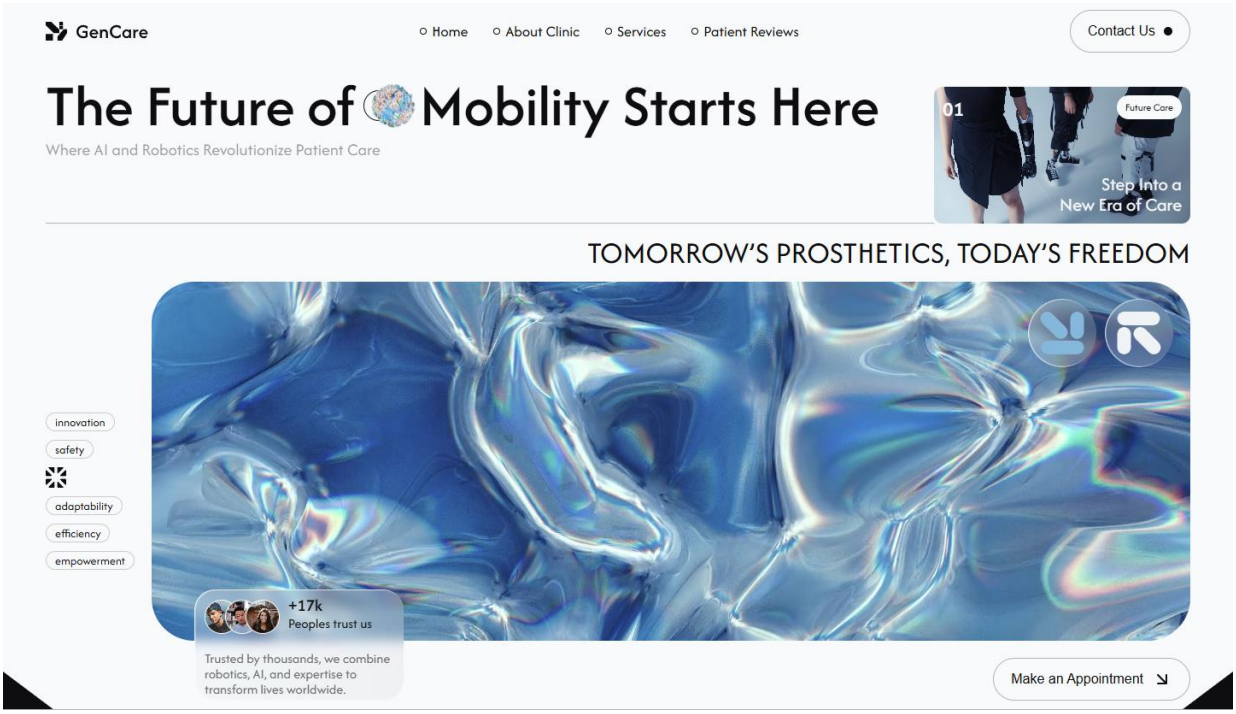


Рисунок 3.5 - Результат розробки сайту високого рівня складності

Варіант із середнім рівнем складності мав меншу кількість мультимедіа і використовував більш статичні елементи, проте зберігав привабливий сучасний вигляд та схожість із варіантом високого рівня навантаженості. Тут були застосовані оптимізовані зображення та базові ефекти при наведенні, що забезпечували інтерактивність, але без перевантаження браузера користувача. Відео було замінене на статичні зображення, щоб зменшити навантаження на продуктивність сторінки. У цьому варіанті також використовувалися базові переходи між секціями та мінімальні скрипти для динамічних компонентів, таких як меню або форми. Такий підхід дозволяв зберегти швидкість завантаження на високому рівні навіть при повільному з'єднанні, забезпечуючи баланс між візуальною насиченістю та продуктивністю.

Легкий варіант сайту був максимально спрощеним та орієнтованим на мінімалістичний підхід. Він містив лише необхідні елементи без додаткових ефектів, а також статичні зображення за необхідності без складних обробок чи фільтрів. Тут було повністю відсутнє фонове відео, а колірна схема та типографіка залишалися чистими та стриманими. Це забезпечувало миттєве завантаження навіть за умов низької швидкості інтернету. Такий підхід є

корисним для демонстрації того, як сайт може виглядати в умовах максимальної оптимізації, коли головний акцент робиться на швидкодії та простоті. Для зменшення навантаження на пристрій були використані лише системні шрифти без підключення додаткових зовнішніх бібліотек.

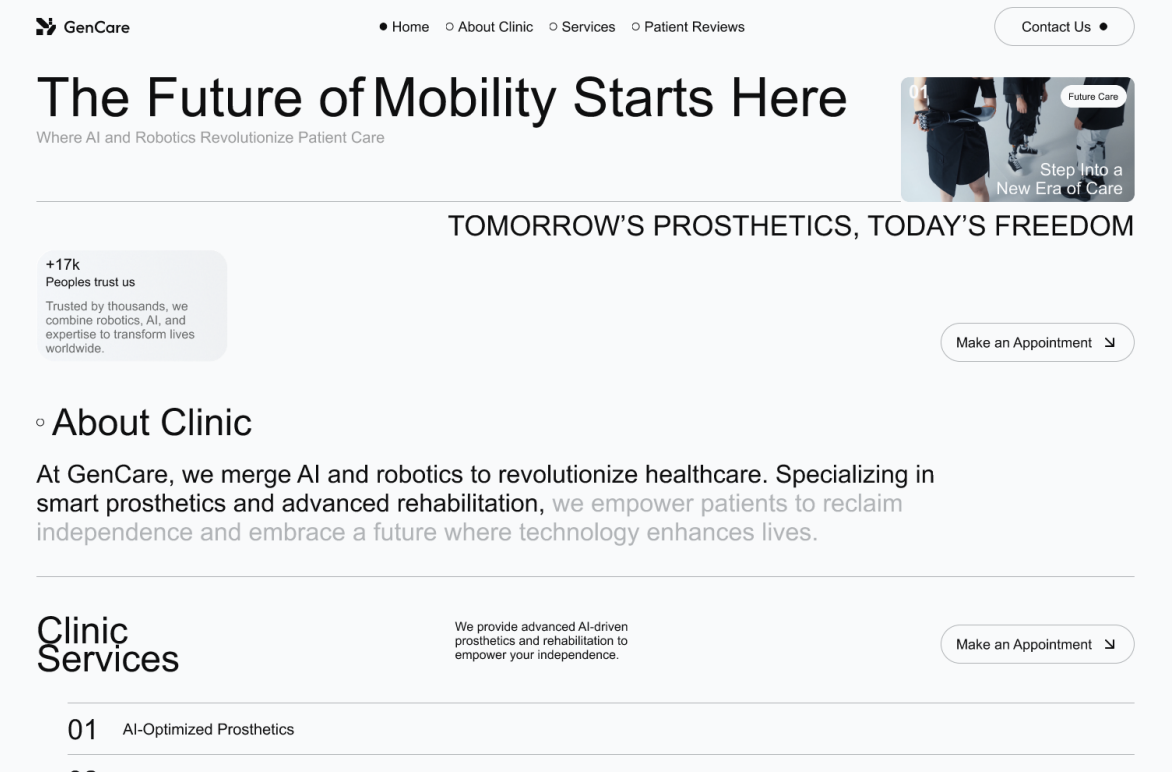


Рисунок 3.6 - Результат розробки сайту низького рівня складності

Процес розробки кожного варіанту розпочинався зі створення макету у Figma. Це дозволяло наочно побачити структуру сторінки, розташування блоків, підбір шрифтів і кольорових рішень, перш ніж переходити до кодування. У випадку з дизайном для високого рівня навантаженості використовувалися складніші композиційні рішення: багатошарові елементи, контрастні кольори, градієнти та інтегровані зображення високої роздільної здатності. Для середнього та легкого рівнів дизайн спрощувався шляхом зменшення кількості декоративних компонентів і більшої уваги до порожнього простору. Цей підхід дозволив відпрацювати різні сценарії

використання однієї й тієї ж тематики, демонструючи, як дизайн можна адаптувати під різні технічні умови.

Важливим етапом стала верстка HTML-структури, що базувалася на семантичних тегах. Це забезпечувало не лише зручність у розробці, але й краще сприйняття сторінки пошуковими системами. Структура сторінки складалася з головного блоку з заголовком і фоновим відео для варіанту високого рівня, секції "Про нас", блоку з послугами, секції переваг, відгуків або успішних історій пацієнтів, а також форми для запису на консультацію та футера з навігацією. Кожна з цих секцій була ретельно пропрацьована для відповідності обраному стилю. Для додаткових елементів, таких як іконки та візуальні маркери, використовувалися векторні формати SVG, які не втрачають якості при масштабуванні і швидко завантажуються.

CSS-оформлення було організова но за допомогою методології BEM, що дозволяло підтримувати код у впорядкованому вигляді. Для високого рівня складності застосовувалися ефекти затемнення, тіні, градієнтні фони та складні анімаційні переходи. При середньому рівні використовувалися більш прості стилі та обмежена кількість кольорових акцентів. У легкому варіанті стилі були зведені до мінімуму: базові кольори, стандартні відступи та прості текстові блоки без декоративних деталей. Особливу увагу було приділено адаптивності, щоб сторінка коректно відображалася на різних пристроях, від великих моніторів до мобільних телефонів.

Результат розробки адаптивної версії сайту наведено на рисунку 3.8.

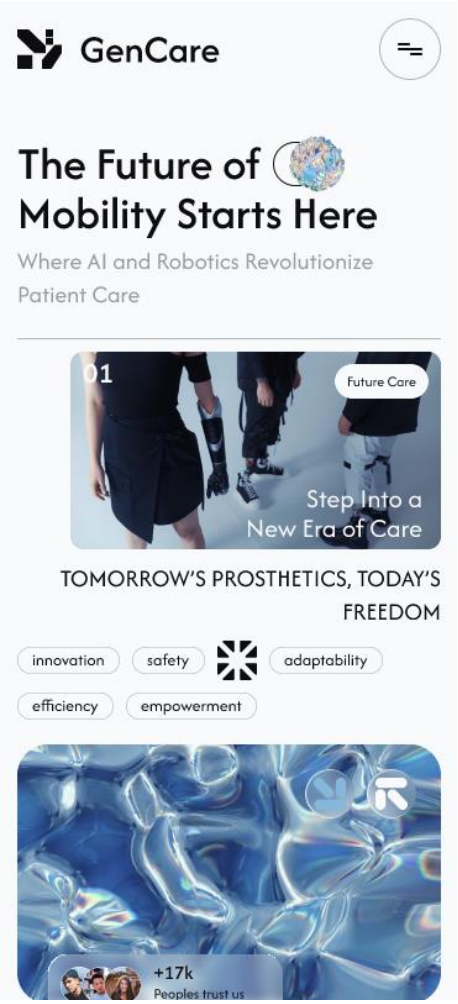


Рисунок 3.7 - Результат розробки адаптивності вебсайту

JavaScript використовувався лише для реалізації ключових інтерактивних елементів. Наприклад, у варіанті з високою складністю це були анімації при прокрутці, плавні переходи між секціями та інтерактивні лічильники, що підвищували динамічність сторінки.

Скрипт для плавної прокрутки до елементів сторінки наведено нижче.

```
document.querySelectorAll('a[href^="#"],
button[href^="#"]').forEach(el => {
    el.addEventListener('click', e => {
        e.preventDefault();
        const targetId = el.getAttribute('href');
        const target = document.querySelector(targetId);

        if (target) {
            target.scrollIntoView({ behavior: 'smooth' });
        }
```

```
    });
});
```

У середньому варіанті збережено повну функціональність ключових сценаріїв, таких як обробка форм та базові інтерактивні елементи, але зменшено навантаження за рахунок спрощення анімацій та обмеження додаткових клієнтських перевірок. У легкому варіанті скрипти були мінімальними, що дозволяло уникнути затримок у відображенні контенту. Додатково були застосовані прийоми оптимізації, зокрема, асинхронне підключення зовнішніх файлів і використання відкладеного завантаження зображень.

Важливим аспектом створення цих варіантів стало тестування продуктивності. Для кожного рівня сайту проводилася перевірка за допомогою Google Lighthouse та PageSpeed Insights. Це дозволяло вимірювати ключові метрики, такі як час завантаження першого контенту, інтерактивність та стабільність верстки. У результаті було підтверджено, що легкий варіант забезпечує майже миттєве завантаження, середній підтримує баланс між візуальною привабливістю і швидкодією, а високий рівень, хоча й вимагав більше часу на завантаження, залишався прийнятним завдяки проведеним оптимізаційним заходам.

Таблиця 3.1 – Результати тестування продуктивності різних варіантів сайту

| Показник/Варіант | Легкий | Середній | Складний |
|---|---|---|---|
| First Contentful Paint (мс) | 450 | 850 | 1 200 |
| Largest Contentful Paint (мс) | 800 | 1 500 | 2 700 |
| Total Blocking Time (мс) | 50 | 180 | 650 |
| Cumulative Layout Shift | 0,01 | 0,03 | 0,05 |
| Speed Index | 500 | 1 200 | 2 500 |
| Page Load Time (с) | 1,2 | 2,5 | 4,0 |

Продовження таблиці 3.1

| | | | |
|---|---|---|---|
| Кількість HTTP-запитів | 18 | 35 | 72 |
| Енергоспоживання на мобільних (ум. од.) | 1 | 2 | 4 |
| Робота кешування | Висока | Середня | Висока |

Розробка трьох варіантів сайту дозволила на практиці продемонструвати, як сучасні технології веб-розробки можуть використовуватися для створення гнучких рішень, що відповідають різним технічним умовам і потребам користувачів. Створення мультимедійного, середнього та полегшеного варіантів не лише показало відмінності між підходами до дизайну, але й дало змогу застосувати алгоритмічні методи вибору варіантів сторінки залежно від характеристик пристрою та інтернет-з'єднання. Робота стала прикладом комплексної реалізації, що поєднала в собі візуальну естетику, технічну оптимізацію та адаптивність, роблячи сайт доступним і зручним для максимально широкої аудиторії.

Тестування адаптивності та продуктивності вебсайту стало одним із ключових етапів практичної реалізації, адже саме воно визначає, наскільки ефективно створені версії сайту здатні працювати в різних умовах, та формує основу для алгоритму підстановки блоків залежно від параметрів користувача. В рамках роботи було проведено аналіз усіх трьох варіантів дизайну: повнофункціонального, оптимізованого та полегшеного. Кожен із цих варіантів мав різний рівень насиченості контентом і графічними елементами, що безпосередньо впливає на швидкість завантаження, час відгуку й рівень навантаження на пристрої користувачів.

Метою тестування було не лише виявити можливі вузькі місця кожної версії сайту, а й визначити оптимальні умови, за яких кожна з них працює найбільш ефективно. Важливим було забезпечення того, щоб користувач із будь-яким рівнем технічних характеристик пристрою чи швидкістю інтернет-з'єднання міг комфортно завантажити сайт і взаємодіяти з його основними

елементами. Тобто ключовим завданням стало створення такої системи адаптації, яка автоматично підбиратиме найбільш відповідний набір блоків, виходячи з фактичних параметрів користувача, навіть якщо його інтернет-з'єднання є вкрай повільним або пристрій має низьку продуктивність.

Першим етапом тестування стала перевірка адаптивності верстки. Усі три версії сайту були протестовані на різних типах пристроїв: настільних комп'ютерах із великими моніторами, ноутбуках, планшетах та смартфонах із різними роздільними здатностями екранів. Це дозволило переконатися, що блоки сторінки коректно перебудовуються під доступний простір, а шрифти та інтерактивні елементи зберігають читабельність і зручність використання незалежно від розміру екрана. Під час цього етапу було виявлено й виправлено дрібні проблеми зі зміщенням окремих елементів у мобільній версії та з відображенням великих зображень на екранах високої щільності (Retina-дисплеї).

Наступним аспектом стало тестування продуктивності при різних швидкостях інтернет-з'єднання. Для цього було змодельовано кілька сценаріїв роботи: високошвидкісне підключення (Wi-Fi або 4G/5G), середнє (3G) та низькошвидкісне (2G або нестабільне мобільне з'єднання). У кожному випадку вимірювалися ключові метрики, такі як час завантаження сторінки (Page Load Time), затримка при першій взаємодії (First Input Delay, FID), показник Largest Contentful Paint (LCP) та загальна кількість HTTP-запитів. Ці показники дозволили оцінити, наскільки швидко користувач може отримати доступ до основного контенту сайту та почати взаємодію з ним.

У випадку швидкого інтернет-з'єднання повнофункціональна версія сайту показала стабільні результати, завантажуючи великі зображення, відео та інтерактивні елементи без відчутних затримок. Середня версія демонструвала баланс між швидкодією та візуальними ефектами, а полегшена завантажувалася практично миттєво навіть на слабкому сигналі. Проте під час тестів із низькою швидкістю мережі (2G) було зафіксовано значні затримки у відображенні повнофункціональної версії, що підтвердило важливість

алгоритму автоматичного підбору блоків: у таких умовах система повинна замінювати важкі елементи на полегшені, щоб уникнути перевантаження сторінки.

Додатково оцінювалася кількість HTTP-запитів, оскільки велика кількість одночасних запитів при завантаженні сторінки значно впливає на її швидкодію. Було встановлено, що у повнофункціональній версії сайту цей показник удвічі вищий порівняно з полегшеною, через наявність великої кількості стилів, скриптів і медіа. Оптимізована версія зменшувала кількість запитів завдяки використанню попередньо стиснутих ресурсів і об'єднаних стилів. Ці результати підтвердили необхідність застосування технік оптимізації, зокрема кешування ресурсів і відкладеного завантаження (Lazy Loading), для підвищення ефективності в реальних умовах використання.

Важливим етапом стала перевірка показника FID, який визначає час між першою взаємодією користувача зі сторінкою (наприклад, натисканням кнопки) та моментом, коли сайт реагує на цю дію. У повнофункціональній версії через велику кількість анімацій і скриптів FID іноді перевищував рекомендовані значення, особливо на старих смартфонах. У полегшеній версії затримка була мінімальною, що демонструє перевагу спрощеного підходу для слабких пристроїв. На основі цих даних було сформовано правило: якщо пристрій має невелику кількість процесорних ядер або обмежену пам'ять, система автоматично віддає пріоритет полегшеній версії для забезпечення швидкого відгуку.

Окремо було проведено аналіз енергоспоживання на мобільних пристроях під час взаємодії з різними версіями сайту. Високий рівень використання процесора через важкі анімації або скрипти призводив до швидшого розряджання акумулятора, що є критично важливим для користувачів мобільних мереж. Полегшена версія сайту, завдяки мінімальній кількості динамічних елементів, забезпечувала значно менше енергоспоживання, що також стало вагомим аргументом на користь її використання для слабших пристроїв або ситуацій із низьким зарядом батареї.

Також тестувалася робота кешування, яка безпосередньо впливає на повторне завантаження сайту. Було перевірено, як зберігаються й використовуються статичні ресурси, включно з CSS, зображеннями та скриптами. При повторних відвідуваннях сторінки час завантаження значно зменшувався завдяки правильно налаштованим заголовкам кешу. Це підтвердило важливість використання кешування як частини загальної стратегії оптимізації для будь-якої версії сайту.

На основі отриманих результатів було побудовано адаптивну модель підстановки блоків. Вона враховує показники швидкості інтернету, продуктивності пристрою, енергоспоживання, кількості запитів, часу завантаження та FID. Алгоритм автоматично аналізує вхідні дані та обирає, які блоки відображати: повнофункціональні, оптимізовані чи полегшені. Завдяки цьому сайт залишається доступним і зручним для всіх користувачів незалежно від їхніх технічних умов. У підсумку навіть при вкрай повільному з’єднанні сторінка завантажується швидко, а основний контент стає доступним практично миттєво, що й було головною метою цього етапу роботи.

## 3.3 Результати тестування алгоритмів класифікації показників веб-сайтів

Результати класифікації показників продуктивності веб-сайтів, поділений за рівнем складності на три частини наведено у таблиці 1. Результати отримано експериментальним чином.

Для виконання експериментів використано таке програмне забезпечення:

- Мова програмування: Python;
- Бібліотеки: numpy, pandas, keras
- Дві відеокарти NVIDIA Tesla T4. кожна з яких має 16 ГБ відеопам'яті.

Таблиця 3.2 – Порівняльний аналіз алгоритмів класифікації

| Алгоритм | Навчальна точність | Тестова точність |
|---|---|---|
| RidgeClassifier | 86.33 | 83.0 |
| Logistic Regression | 99.33 | 97.0 |
| k-Nearest Neighbors | 59.67 | 32.0 |
| Stochastic Gradient | 38.67 | 25.5 |
| VotingClassifier-hard voiting | 99.33 | 98.0 |
| VotingClassifier-soft voting | 100.00 | 83.5 |

Аналізуючи дані у вищенаведеній таблиці, можна зробити висновок, що hard та soft VotingClassifier показують кращі результати у порівнянні з одиничним використанням загальновідомих алгоритмів та підходів [34, 35]. Такий підхід дозволяє більш якісніше класифікувати показники продуктивності веб-сайтів та відносити їх до певної категорії для подальшого завантаження відповідно до типу пристрою, інтернет з'єднання тощо.

Після виконання класифікації також було помічено, що VotingClassifier проявляє стабільність навіть при наявності деякого шуму у вхідних даних, що може виникати через варіації метрик продуктивності на різних пристроях або при різних умовах мережі.

Крім того, результати показують, що використання ансамблевого підходу дозволяє зменшити ймовірність помилкової класифікації у порівнянні з одиночними моделями, що особливо важливо для систем, які автоматично підбирають оптимальні варіанти сторінок для користувачів.

Практичне застосування цих алгоритмів дозволяє інтегрувати систему класифікації у процес вибору адаптивного інтерфейсу, підвищуючи загальну ефективність та зручність використання веб-сайтів незалежно від технічних характеристик користувацьких пристроїв.

## 3.4 Висновки до розділу

На основі підходів до реалізації програмного забезпечення, розроблено програмний модуль кластеризації даних для розподілу показників продуктивності веб-сайтів, що дозволяє розподілити на 3 класи.

Проаналізовано алгоритми класифікації даних, зокрема ансамблеві підходи до голосування для отримання якіснішого результату. Результати показали точність у 98% на тестовій вибірці;

# ВИСНОВКИ

У даній роботі розроблено алгоритм підбору оптимальних версій веб сторінок на основі показників продуктивності та попередньо навчених даних на основі ансамблів. За результатами проведених досліджень можна зробити такі висновки:

- проведено аналіз існуючих підходів до вимірювання продуктивності веб-сайтів для можливості подальшого формування датасету показників сайтів, що дозволило виділити перелік ключових параметрів оцінки продуктивності сайтів для подальшої класифікації;
- проаналізовано алгоритми класифікації даних, зокрема ансамблеві підходи до голосування для отримання якіснішого результату. Результати показали точність у 98% на тестовій вибірці;
- Розроблено ансамблевий метод класифікації показників, що дозволяє максимально ефективно використати можливості різних алгоритмів навчання без вчителя.
- розроблено узагальнений алгоритм вибору типу сторінки в залежності від вхідних параметрів, що дозволяє автоматично підбирати версії веб сторінок відповідно до умов;

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. K. Mani. Machine learning models in web applications. Journal / Conference, 2025.
2. Калініченко, В. С.; Ковтунець, О. В. Вимірювання ефективності веб-додатку. Аналіз найважливіших показників ефективності. — КПІ ім. Ігоря Сікорського, SoftTech-2021. https://ela.kpi.ua/server/api/core/bitstreams/65a34cd5-3e66-4a82-9c3e-a142c334eedb/content
3. Amjad, Mahfida, Md Tutul Hossain, Rakib Hassan, and Md Abdur Rahman. "Web application performance analysis of E-commerce sites in Bangladesh: an empirical study." International Journal of Information Engineering and Electronic Business 13, no. 2 (2021): 47-54.
4. M. Santiago, et al. Integrating Machine Learning with Web Intelligence for Web-based Search and Recommendation Systems. 2024, pp. 44−53 pdfs.semanticscholar.org/dd2d/ac3d6c679c5c95487636beca96ded4331465.pdf
5. Ivan Chaus, Tetyana Marusenkova. Front-end Framework for Building Applications With Adaptive User Interfaces Using Machine Learning Methods. CSN, 2024, 252-267. DOI: 10.23939/csn2024.02.252.
6. Madhulatha, T. Soni. An Overview on Clustering Methods. arXiv preprint, 2012. 14 c. arXiv:1205.1117.
7. Bolón-Canedo, Verónica, Noelia Sánchez-Marono, and Amparo Alonso-Betanzos. "Data classification using an ensemble of filters." Neurocomputing 135 (2014): 13-20. https://doi.org/10.1016/j.neucom.2013.03.067
8. Jafarzadeh, Hamid, Masoud Mahdianpari, Eric Gill, Fariba Mohammadimanesh, and Saeid Homayouni. 2021. "Bagging and Boosting Ensemble Classifiers for Classification of Multispectral, Hyperspectral and PolSAR Data: A Comparative Evaluation" Remote Sensing 13, no. 21: 4405. https://doi.org/10.3390/rs13214405
9. Khan, Azal Ahmad, Omkar Chaudhari, and Rohitash Chandra. "A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation." Expert Systems with Applications 244 (2024): 122778.

10. O. V. Klochko. The Use of Data Mining To Predict Web Performance. https://www.scribd.com/document/20161572/The-Use-of-Data-Mining-to-Predict-Web-Performance
11. L. Jin and L. Chen, "Exploring the Impact of Computer Applications on Cross-Border E-Commerce Performance," in IEEE Access, vol. 12, pp. 74861-74871, 2024, doi: 10.1109/ACCESS.2024.3385017
12. Heričko, Tjaša, Boštjan Šumak, and Saša Brdnik. "Towards representative web performance measurements with google lighthouse." In Proceedings of the 2021 7th student computer science research conference, p. 39. 2021.
13. Akhtar, Naveed; et al. Identifying the Intents Behind Website Visits by Employing Unsupervised Machine Learning Models. Annals of Data Science, 2024. DOI:10.1007/s40745-024-00586-5.
14. Zhang, Yuzhen, Jingjing Liu, and Wenjuan Shen. 2022. "A Review of Ensemble Learning Algorithms Used in Remote Sensing Applications" Applied Sciences 12, no. 17: 8654. https://doi.org/10.3390/app12178654
15. Pitsun O. Comparative analysis of the value-semantic sphere of Ukrainian volunteers using AI. CEUR Workshop ProceedingsConference - 2025 https://ceur-ws.org/Vol-3974/short06.pdf
16. Березький О. М., Піцун О. Й., Мельник Г. М., Дацко Т. В. Застосування методу лінійної регресії для аналізу кількісних характеристики цитологічних зображень. Український журнал інформаційних технологій. 2021, т. 3, № 1. С. 73–77.
17. T. H. Zhang, et al. “K-Means Clustering: Performance, Limitations and Metrics in Low-Resource Environments.” Journal of Software Engineering and Applications, 2024, 17, pp. 817-831. DOI: 10.4236/jsea.2024.1711045.
18. Alqurashi, T.; Wang, W. Clustering ensemble method. International Journal of Machine Learning & Cybernetics, vol. 10, 2019, pp. 1227–1246. DOI: 10.1007/s13042-017-0756-7.

19. J. P. Ntayagabiri, et al. “Clustering Techniques in Unsupervised Learning: Review of K-means, Fuzzy C-means, BIRCH & EM.” Open Journal of Applied Sciences, 2023, 13, 1163–1177. DOI:10.4236/ojapps.2023.137092.
20. Ahn, Hongshik, Hojin Moon, Melissa J. Fazzari, Noha Lim, James J. Chen, and Ralph L. Kodell. "Classification by ensembles from random partitions of high-dimensional data." Computational Statistics & Data Analysis 51, no. 12 (2007): 6166-6179.
21. Kilaj, Michal; Bani-Khalid, Salim. An Intelligent Web Service Composition and Resource-Optimization Method Using K-Means Clustering and Knapsack Algorithms. Mathematics (MDPI), 2023, 9 (17). DOI:10.3390/math9172182.
22. Ping Dai, et al. “Design and Performance Evaluation of Efficient Clustering Algorithms for Big Data Applications.” Applied Mathematics and Nonlinear Sciences, 2025. DOI: 10.2478/amns-2025-0056.
23. Mahajan, Palak, Shahadat Uddin, Farshid Hajati, and Mohammad Ali Moni. 2023. "Ensemble Learning for Disease Prediction: A Review" Healthcare 11, no. 12: 1808. https://doi.org/10.3390/healthcare11121808
24. Z. Farhadi, M. A. Khashei, et al. An Ensemble Framework to Improve the Accuracy of Regression Using Elastic Net and Clustering. Applied Sciences, 2022, 12(20):10608. DOI: 10.3390/app122010608.
25. Albert Bifet, et al. Low-latency multi-threaded ensemble learning for dynamic big data streams. https://scispace.com/pdf/low-latency-multi-threaded-ensemble-learning-for-dynamic-big-35y4891cqb.pdf
26. Dasari, A.K., Biswas, S.K., Thounaojam, D.M., Devi, D., Purkayastha, B. (2023). Ensemble Learning Techniques and Their Applications: An Overview. In: Kumar, A., Mozar, S., Haase, J. (eds) Advances in Cognitive Science and Communications.
27. Torabi, Majid; Nur Izura Udzir; Mohd Taufik Abdullah; Yaakob, Razali. International Journal of Advanced Computer Science and Applications; West Yorkshire Vol. 12, Iss. 5, (2021). DOI:10.14569/IJACSA.2021.0120566

28. Le-vus, Ye. V., & Vasyli-uk, R. B. (2022). Re-com-men-da-ti-on al-go-rithm using da-ta clus-te-ring. Uk-ra-ini-an Jo-ur-nal of In-for-ma-ti-on Techno-logy, 4(2), 18–24. https://doi.org/10.23939/ujit2022.02.018
29. Pitsun O. Comparative analysis of stress factors of humanities and technical specialities students . CEUR Workshop ProceedingsConference - 2024 https://ceur-ws.org/Vol-3716/paper6.pdf
30. Li, Hongmin; Ye, Xiucai; Imakura, Akira; Sakurai, Tetsuya. LSEC: Large-scale spectral ensemble clustering. arXiv preprint, 2021. 16 с. DOI:10.48550/arXiv.2106.09852.
31. Fournier, Quentin; Ezzati-Jivan, Naser; Aloise, Daniel; Dagenais, Michel R. “Automatic Cause Detection of Performance Problems in Web Applications.” - 2021 https://er.knutd.edu.ua/handle/123456789/17660
32. Shalev-Shwartz, Shai; Ben-David, Shai. Clustering (Chapter 22), у книзі Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 2014, стор. 264–277. DOI:10.1017/CBO9781107298019.023.
33. Oleshchenko L.M., Burchak P.V. “Software System Architecture Development for Intelligent Analysis of Web Application Performance Metrics” Том 35, № 4, 2024. https://www.tech.vernadskyjournals.in.ua/journals/2024/4_2024/24.pdf
34. Gomes, Heitor Murilo, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. "A survey on ensemble learning for data stream classification." ACM Computing Surveys (CSUR) 50, no. 2 (2017): 1-36. https://doi.org/10.1145/305492
35. I. D. Mienye and Y. Sun, "A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects," in IEEE Access, vol. 10, pp. 99129-99149, 2022, doi: 10.1109/ACCESS.2022.3207287.
36. Березький О.М., Мельник Г.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня “Магістр”. Спеціальність: 123 - Комп’ютерна інженерія. Магістерська програма - Комп’ютерна інженерія". Тернопіль: ЗУНУ, 2024. 33 с.

ДОДАТОК А
Лістинг коду програми для класифікації засобами ансамблів

ДОДАТОК Б
Світлокопії публікацій