Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА
БАКАЛАВРА

на тему *Метод прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконала: *студент групи КН-20-1* Підпис *Богдан КОРЧЕВСЬКИЙ*
Група виконавця Ім'я, ПРІЗВИЩЕ

Керівник: *зав. каф. КН, д.т.н., проф.* Підпис *Олександр БАРМАК*
Науковий ступінь, посада Ім'я, ПРІЗВИЩЕ

Нормоконтроль: *к.т.н., доц. каф. КН* Підпис *Руслан БАГРІЙ*
Науковий ступінь, посада Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
зав. кафедри КН, д.т.н., професор Підпис *Олександр БАРМАК*
Ім'я, ПРІЗВИЩЕ

13 06 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

(підпис)
д.т.н., професор Олександр БАРМАК
«16» 02 *2024* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Метод прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу»*

2. Завдання видано студенту *Богдану КОРЧЕВСЬКОМУ*
(ім'я, прізвище)

3. Керівник роботи *зав. каф. КН, д.т.н., проф. Олександр БАРМАК*
(посада, ім'я, прізвище)

4. Затверджено наказом університету від «15» 02 *2024* р. № 8

5. Дата видачі завдання студенту: «16» 02 *2024* р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи – покращення ефективності прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу. При прогнозуванні слід використовувати різноманітні фактори, які впливають на відвідуваність вебсайтів. Слід забезпечити виконання функцій додавання, редагування та перегляду всіх необхідних для роботи вебсервісу даних, й одержання наступних результатів роботи вебсервісу: автоматизоване редагування, додавання факторів необхідних для прогнозування відвідуваності вебсайтів, реалізація прогнозування за допомогою нейромережевого підходу, авторизація, реєстрація та додавання, редагування даних про вебсайт користувача.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником*	*грудень 2023*	*виконано*
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	*січень 2024*	*виконано*
3	*Робота над розділом 1 Аналіз методів, засобів та підходів до прогнозування відвідуваності вебсайтів*	*лютий 2024*	*виконано*
4	*Робота над розділом 2 Метод прогнозування відвідуваності вебсайтів нейромережевими засобами*	*березень 2024*	*виконано*
5	*Робота над розділом 3 Реалізація вебсервісу з методом прогнозування відвідуваності вебсайтів*	*квітень 2024*	*виконано*
6	*Оформлення пояснювальної записки згідно вимог*	*травень 2024*	*виконано*
7	*Попередній захист кваліфікаційної роботи бакалавра*	*травень 2024*	*виконано*
8	*Отримання відгуку керівника, рецензії, перевірка на плагіат, нормоконтроль та захист кваліфікаційної роботи бакалавра*	*червень 2024*	*виконано*

Виконавець: *студент групи КН-20-1* _____ *Богдан КОРЧЕВСЬКИЙ*
Група виконавця / Підпис / Ім'я, ПРІЗВИЩЕ

Керівник: *зав. каф. КН, д.т.н., проф.* _____ *Олександр БАРМАК*
Науковий ступінь, посада / Підпис / Ім'я, ПРІЗВИЩЕ

Анотація

Тема кваліфікаційної роботи бакалавра: *«Метод прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу»*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-20-1 Богдан КОРЧЕВСЬКИЙ*

Керівник кваліфікаційної роботи бакалавра: *завідувач кафедри КН, д.т.н., проф. Олександр БАРМАК*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
69	34	4	48	4

Метою кваліфікаційної роботи бакалавра є покращення ефективності прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу. При прогнозуванні було використано різноманітні фактори, які впливають на відвідуваність вебсайтів. Було забезпечено виконання необхідних для роботи вебсервісу функцій, та одержання наступних результатів роботи вебсервісу: автоматизоване редагування, додавання факторів необхідних для прогнозування відвідуваності вебсайтів, реалізація прогнозування за допомогою нейромережевого підходу, авторизація, реєстрація та додавання, редагування даних про вебсайт користувача.

Розроблений вебсервіс призначений для власників вебсайтів, які бажають покращити відвідуваність власних вебресурсів. Реалізований метод прогнозування відвідуваності вебсайтів дозволяє здійснювати прогнози, які можуть вплинути на подальший розвиток вебсайтів.

Практичними результатами використання розробленого вебсервісу є прогнозування відвідуваності вебсайтів.

Ключові слова: нейромережа, прогноз відвідуваності, вебсайт.

Виконавець: *студент групи КН-20-1* _____ *Богдан КОРЧЕВСЬКИЙ*

Група виконання | Підпис | Ім'я, ПРІЗВИЩЕ

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
БД	База даних
GPU	Graphics processing unit
RNN	Recurrent neural networks
DNN	Deep Neural Network
CNN	Convolutional neural network
SEO	Search engine optimization
VAE	Variational Autoencoders
LSTM	Long Short-Term Memory
ПЗ	Пояснювальна записка
API	Application programming interface
JSON	JavaScript Object Notation
CSV	Comma-separated values
CSRF	Cross-Site Request Forgery
XSS	Cross Site Scripting
SSO	Single Sign-On
JS	JavaScript
TS	TS TypeScript
HTTP	HyperText Transfer Protocol
REST	Representational State Transfer
SQL	Structured Query Language
LDAP	Lightweight Directory Access Protocol
VS	Visual Studio

Вступ

Кваліфікаційна робота бакалавра присвячена розробці методу прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу та реалізації його у вигляді вебсервісу.

Актуальність. У сучасному інтернет-просторі відвідуваність вебсайтів визначає їх успішність та вплив на аудиторію. Розвиток технологій та величезна кількість інформації роблять конкуренцію серед вебресурсів жорсткою. Тому актуальним стає питання розробки ефективного методу прогнозування відвідуваності вебсайтів з метою підвищення їх ефективності та конкурентоспроможності.

Об’єкт дослідження – процес прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу, який включає в себе аналіз поведінки користувачів, оцінку впливу різних факторів на відвідуваність та розробку моделей, що можуть передбачати ці показники.

Предмет дослідження – нейромережеві методи прогнозування відвідуваності вебсайтів, алгоритми та засоби, що використовуються для прогнозування відвідуваності вебсайтів подані у вигляді вебсервісу.

Мета кваліфікаційної роботи бакалавра – покращення ефективності прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу реалізованого у вигляді вебсервісу.

Завдання кваліфікаційної роботи бакалавра. Провести аналіз предметної області прогнозування відвідуваності вебсайтів. Виконати аналіз методів прогнозування за допомогою нейромережевого підходу. Виконати аналіз наявних рішень щодо подібних задач. Розробити метод прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу. Спроєктувати структуру вебресурсу з автоматизованим прогнозуванням відвідуванності вебсайтів за допомогою нейромережевого підходу. Спроєктувати структуру бази даних вебресурсу. Виконати вибір засобів розробки вебресурсу з прогнозуванням відвідуванності вебсайтів за допомогою нейромережевого

підходу. Виконати програмну реалізацію вебресурсу. Провести тестування розробленого вебресурсу з автоматизованим прогнозуванням відвідуванності вебсайтів за допомогою нейромережевого підходу. Виконати дослідження ефективності розробленого методу прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу.

Розділ 1 Аналіз методів, засобів та підходів до прогнозування відвідуваності вебсайтів

1.1 Огляд методів прогнозування відвідуваності вебсайтів

Прогнозування відвідуваності вебсайтів є важливим завданням для багатьох компаній та веброзробників. Це допомагає розуміти поведінку користувачів, виявляти тенденції та здійснювати налагодження стратегій маркетингу та розвитку. У цьому розділі розглядаються основні методи прогнозування відвідуваності вебсайтів, включаючи статистичні методи, машинне навчання та нейронні мережі.

Перш за все слід розглянути статистичні методи [1]. Статистичні методи застосовують в усіх областях, які передбачають ухвалювання рішень, щоби отримувати точні висновки з консолідованого [2] масиву даних для ухвалення рішень в умовах невизначеності на основі статистичної методології.

Одним з найпоширеніших статистичних методів прогнозування відвідуваності вебсайтів – це метод заснований на аналізі часових рядів [3]. Він використовує історичні дані про відвідуваність для прогнозування майбутньої активності. Цей підхід дозволяє виявляти сезонні тенденції, циклічні зміни та інші закономірності у поведінці користувачів.

Також варто приділити увагу методам машинного навчання [4]. Машинне навчання стало потужним інструментом у прогнозуванні відвідуваності вебсайтів. Моделі, побудовані на основі алгоритмів машинного навчання, можуть враховувати більш широкий спектр факторів, що впливають на відвідуваність, такі як маркетингові кампанії, вміст сайту, погода тощо. Серед популярних методів – лінійна регресія, випадковий ліс [5], градієнтний бустинг [6] тощо.

Одним з найбільш ефективних інструментів у прогнозуванні складних залежностей у відвідуваності вебсайтів є нейромережі. Вони можуть адаптуватися до змінних умов та виявляти складні неявні закономірності. Для прогнозування відвідуваності вебсайтів застосовуються різноманітні архітектури

нейронних мереж, включаючи звичайні штучні нейронні мережі, рекурентні нейронні мережі та згорткові нейронні мережі.

Нейронні мережі [7] – це математичні моделі, які намагаються імітувати роботу людського мозку. Вони складаються з нейронів [8], які взаємодіють між собою за допомогою зв'язків. Ці зв'язки мають вагу, яка визначає важливість сигналу, що йде від одного нейрона до іншого. Нейронні мережі використовуються в багатьох сферах, включаючи розпізнавання образів, мовлення, управління, прогнозування та багато інших.

Нейронна мережа складається зі шарів нейронів. Перший шар називається вхідним, останній – вихідним, а всі інші – прихованими шарами. Кожен нейрон у шарі з'єднаний з кожним нейроном у наступному шарі. Ця структура дозволяє мережі виявляти складні залежності у вхідних даних.

Для роботи нейронної мережі вхідні дані подаються на вхідний шар, після чого вони проходять через приховані шари, де обчислюються ваги зв'язків між нейронами. На виході вихідного шару отримується результат роботи мережі. Для навчання нейронної мережі використовують алгоритми оптимізації, такі як зворотнє поширення помилки [9], які дозволяють коригувати ваги зв'язків таким чином, щоб мережа навчалася відповідати на вхідних даних.

Нейронні мережі поділяють на декілька видів. Найпоширеніші види:

- Перцептрон [10] – це найпростіший тип нейронної мережі, який складається з одного або декількох шарів нейронів, які працюють як бінарний класифікатор; використовується для задач класифікації;
- Рекурентні нейронні мережі [11] (RNN) – цей тип мережі має зворотний зв'язок, що дозволяє їй зберігати інформацію про попередні стани в процесі роботи; використовується для аналізу послідовностей даних, таких як мовлення або часові ряди;
- Згорткові нейронні мережі [12] (CNN) – цей тип мережі використовується для обробки вхідних даних зі структурою сітки, таких як зображення; CNN мають спеціальні шари згортки та пулінгу, що дозволяють виявляти важливі функції в даних;

– Глибокі нейронні мережі [13] (DNN) – це тип мережі, який має багато шарів нейронів між вхідним та вихідним шарами які DNN використовуються для складних задач, таких як розпізнавання образів або голосу;
– Модульні нейронні мережі [14] – цей тип мережі складається з окремих модулів, які можуть бути навчені окремо та пізніше об'єднані для рішення більш складних задач;
– Автокодери [15] – це тип мережі, який використовується для автоматичного витягування корисних функцій з даних. Вони використовуються для пониження розмірності даних.

У таблиці 1.1 наведені переваги та недоліки найпоширеніших видів нейромереж.

Таблиця 1.1 – Переваги та недоліки нейромереж різної архітектури

	Перцептрон	RNN	CNN
Переваги	Простота реалізації, добре підходить для завдань бінарної класифікації та лінійних задач.	здатність враховувати послідовності даних, широко використовується в. задачах обробки природної мови та часових рядів.	Ефективність у роботі зі. зображеннями, вміння виявляти просторові шаблони.
Недоліки	Обмежена здатність до навчання складних нелінійних залежностей, потребує великої кількості даних.	Проблема зникновення та вибуху градієнту, обмежена пам'ять для довгих послідовностей.	Потребують багато даних для навчання, можуть бути складні у використанні для інших типів даних.

Продовження таблиці 1.1 – Переваги та недоліки нейромереж різної архітектури

	DNN	Модульні нейронні мережі	Автокодери
Переваги	Здатність моделювати складні залежності, висока точність у багатьох задачах	Зручність у побудові та розширенні, можливість використання готових модулів для побудови складних архітектур.	Використовуються для зменшення розмірності даних та відновлення даних, ефективні для роботи з великими об'ємами даних.
Недоліки	Потребують великої обчислювальної потужності та об'єму даних для навчання, можуть страждати від перенавчання.	Можуть виникати проблеми зі сумісністю модулів, потребують уважного проектування.	Можуть втрачати деяку інформацію під час зменшення розмірності даних, вимагають уважного налаштування гіперпараметрів.

У випадку прогнозування відвідуваності вебсайтів нейронні мережі можуть бути навчені на історичних даних про відвідуваність, враховуючи різноманітні фактори, такі як день тижня, час доби, свята тощо. Мережа може виявити складні залежності між цими факторами та відвідуваністю, що дозволить здійснювати точні прогнози.

Одним з прикладів досліджень у цій області є робота [16]. У цій статті автори досліджують застосування нейромереж для прогнозування відвідуваності вебсайтів. Вони використовують дані про відвідуваність сайту та інші фактори, щоб навчити нейромережі прогнозувати майбутній трафік. Результати показують, що нейромережевий підхід може бути ефективним для цієї задачі.

Інший приклад – стаття [17]. У цій роботі автори досліджують прогнозування відвідуваності вебсайтів зокрема для електронної комерції [18]. Вони використовують нейромережі для аналізу великих обсягів даних та прогнозування майбутнього трафіку на сайті. Результати показують, що нейромережевий підхід може допомогти покращити точність прогнозування трафіку.

Прогнозування відвідуваності вебсайтів є важливим завданням, яке може бути вирішено різними методами. Статистичні методи, машинне навчання та нейронні мережі – це лише кілька з можливих підходів. Вибір конкретного методу залежить від конкретних вимог та особливостей задачі прогнозування. Для розробки методу прогнозування відвідуваності вебсайтів було обрано саме нейромережевий підхід оскільки за характеристиками є найбільш ефективний та розповсюджений.

1.2 Огляд існуючих програмних рішень для прогнозування відвідуваності сайтів

В сучасному цифровому світі відвідуваність вебсайту визначається не лише якістю його контенту, а й ефективністю маркетингових стратегій та технічною оптимізацією. Прогнозування та аналіз відвідуваності стає важливим завданням для веброзробників та маркетологів з метою підвищення трафіку та конверсії сайту.

У даному розділі проводиться огляд існуючих програмних рішень для прогнозування відвідуваності сайтів. Аналізується спектр інструментів та методів, що застосовуються в цілях прогнозування вебтрафіку. Висвітлюються переваги та недоліки кожного програмного рішення та його обмеження.

Огляд існуючих програмних рішень для прогнозування відвідуваності сайтів дозволить визначити найбільш оптимальний та ефективний підхід до вирішення поставленої задачі.

Щодо готових програмних рішень для прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу, то їх існує досить багато. Найпопулярніший з усіх це вебсервіс під назвою «Google Analytics» [19].

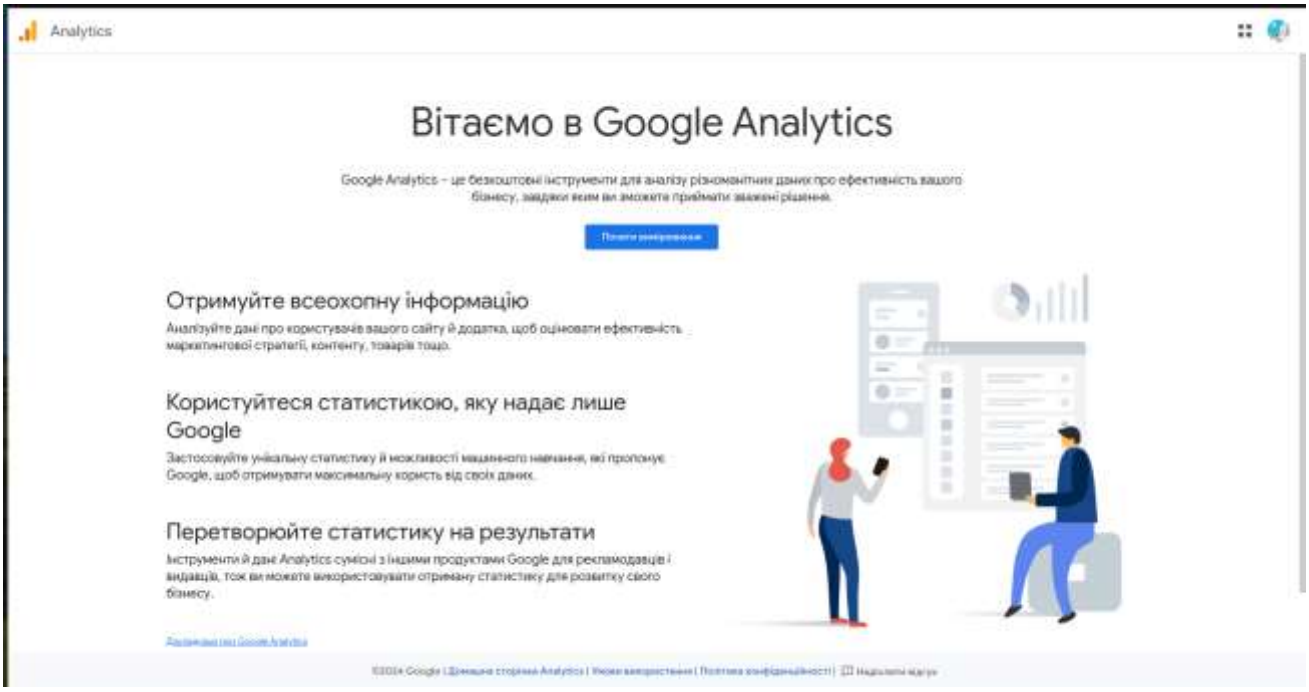


Рисунок 1.1 – Сторінка вітання «Google Analytics»

На сторінці вітання (рисунок 1.1) є єдина кнопка «Почати вимірювання” та інформація про можливості вебсервісу, що є інтуєтивно зрозумілим для користувача. Після переходу за допомогою кнопки, потрібно пройти авторизацію або регістрацію облікового запису в «Google Analytics». Після успішної авторизації є можливість створити новий ресурс (або вибрати вже існуючий) для відстеження. Ресурс може бути власним вебсайтом, мобільним додатком або іншим цифровим активом. Далі вебсервіс пропонує вказати деякі бізнес-деталі, такі як назва бізнесу, категорія бізнесу, часовий пояс тощо. Ці дані допоможуть «Google Analytics» краще зрозуміти бізнес та його контекст. Після цього кроку можна визначити бізнес-цілі власного ресурсу. Це може бути, наприклад, продаж продукту, підписка на розсилку, завантаження файлу тощо. Можна вибрати доступні метрики для вимірювання, щоб оцінити досягнення цих цілей. Після налаштування всіх необхідних параметрів «Google Analytics» почне збирати дані про ресурс. З плином часу можна переглядати дані у зручному для

користувача форматі, а також аналізувати їх, щоб зрозуміти, як користувачі взаємодіють з ресурсом та як можна покращити їх досвід.

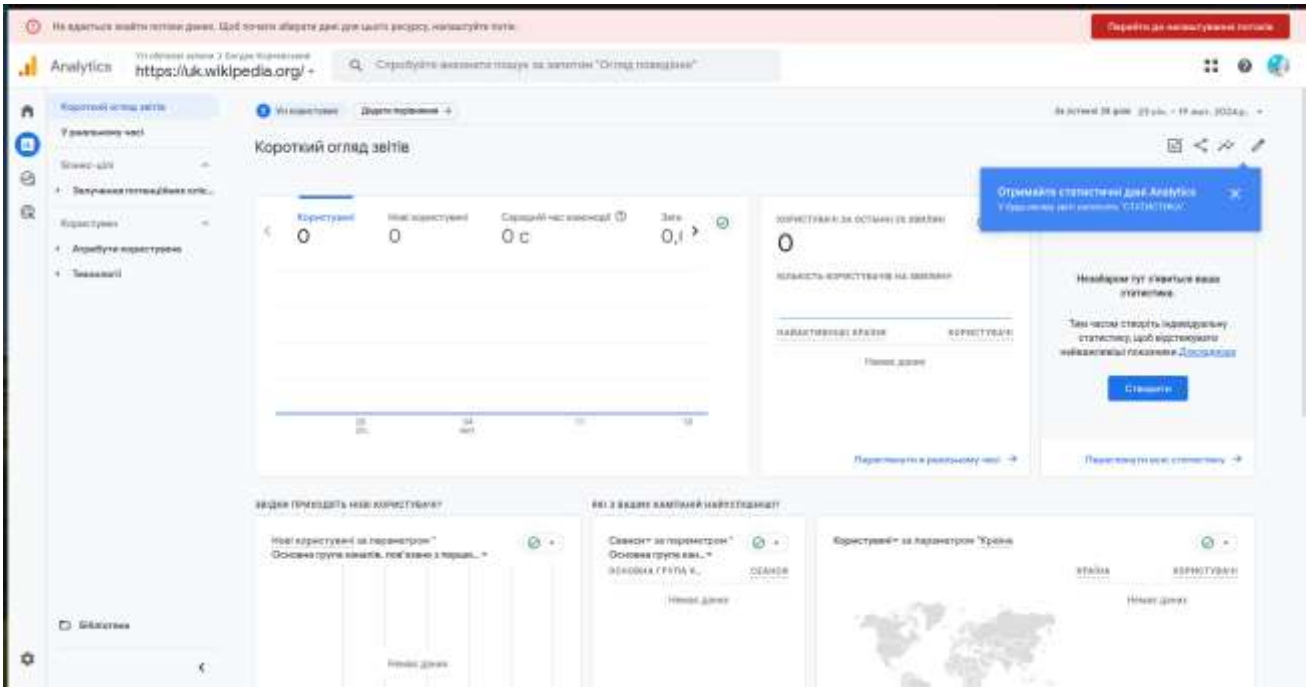


Рисунок 1.2 – Сторінка з показниками та діаграмами «Google Analytics»

Після проходження усіх реєстраційних завдань, користувач може з плином часу користуватися готовою аналітикою вказаного ресурсу і отримувати прогнозуючу інформацію (рисунок 1.2). Працювати з вебсервісом дуже зручно оскільки він має інтуїтивно зрозумілий інтерфейс та багато документації яка описує використання системи.

Наступний вебсервіс, який слід розглянути це «Amazon Forecast» [20]. Сторінка вітань також містить необхідну інформацію про користування вебсервісом, умовами користування, ціни та перехід на наступні кроки. Сторінка виглядає досить стильно і відповідає сучасним стандартам оформлення сайтів.

Після реєстрації та авторизації можна працювати з сервісом для прогнозування відвідуваності і після успішного внесення усіх необхідних даних отримуємо результат, який можна представити у вигляді таблиці (рисунок 1.3) та проводити різноманітні сортування, фільтрації, ділення по групах та багато інших операцій.

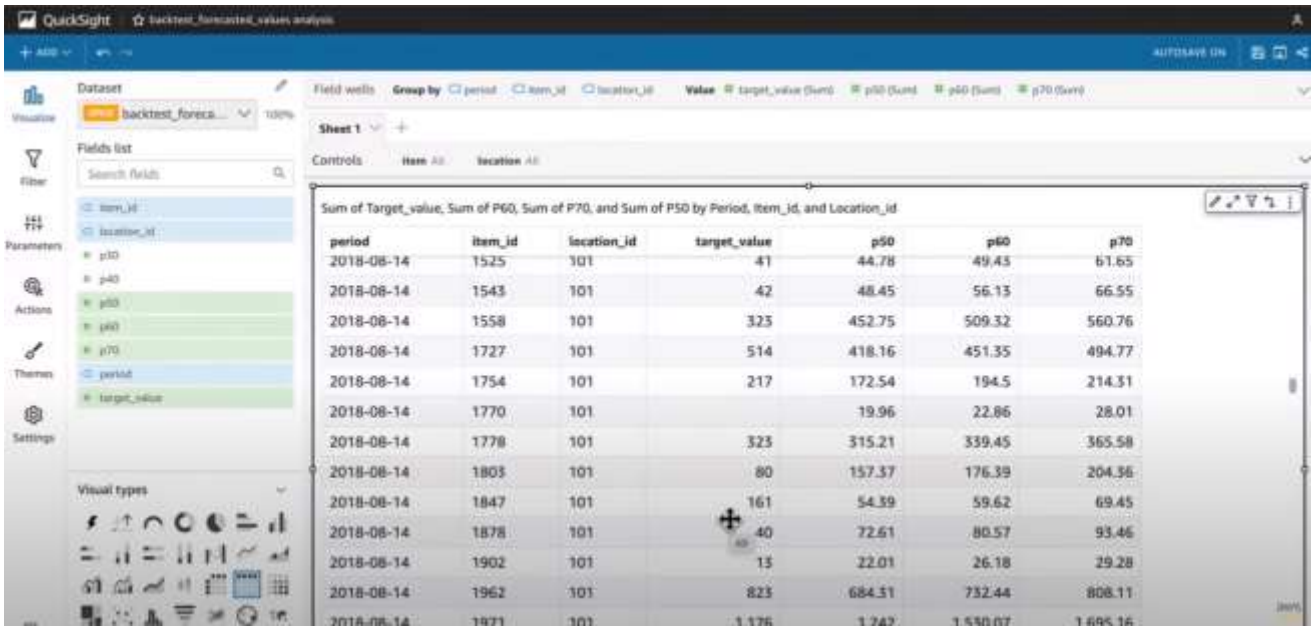


Рисунок 1.3 – Сторінка результатів у вигляді таблиці «Amazon Forecast»

Цільове призначення дещо відрізняється від попереднього вебсайту. «Google Analytics» призначений для відстеження та аналізу вебтрафіку, поведінки користувачів та ефективності маркетингових кампаній [21]. «Amazon Forecast» призначений для прогнозування майбутніх значень часових рядів, таких як продажі, відвідуваність тощо, на основі історичних даних. Також «Google Analytics» надає засоби для аналізу вебтрафіку, включаючи відвідуваність сторінок, джерела трафіку, конверсії тощо. «Amazon Forecast» в свою чергу аналізує часові ряди і створює прогнози майбутніх значень на основі цих даних. Також «Google Analytics» використовує дані про вебтрафік та поведінку користувачів для аналізу та вдосконалення маркетингових стратегій. «Amazon Forecast» використовує історичні дані для створення моделей прогнозування, які можна використовувати для планування запасів, бюджетування [22] тощо. Також не потрібно виключати врахування інтеграцій з іншими сервісами. «Google Analytics» може бути інтегрований з іншими сервісами «Google», такими як «Google Ads» [23], для більш точного аналізу ефективності рекламних кампаній. «Amazon Forecast» може бути інтегрований з

іншими сервісами «Amazon», такими як «Amazon S3» [24], для зберігання та обробки даних перед прогнозуванням.

Для реалізації методів прогнозування відвідуваності сайтів можна використовувати різні бібліотеки та фреймворки залежно від обраного підходу та мови програмування. Деякі з популярних бібліотек це:

1) ML.NET – це бібліотека машинного навчання для .NET, яка містить реалізації різних алгоритмів, включаючи регресію, класифікацію та кластеризацію;
2) Accord.NET – бібліотека також надає інструменти для машинного навчання, включаючи підтримку векторів, регресію, класифікацію та кластеризацію;
3) SharpLearning – бібліотека, яка спрощує використання машинного навчання у C#. Вона містить реалізації різних алгоритмів, таких як випадковий ліс, градієнтний бустінг тощо;
4) Weka – бібліотека для класичних методів машинного навчання для Java;
5) TensorFlow / Keras – бібліотека для нейронних мереж та глибокого навчання для Python.

Отже, після проведення аналізу існуючих програмних рішень для прогнозування відвідуваності сайтів визначено, що конкуренція в даній області досить потужна, вебсервіси конкурентів зручні та чудово пристосовані для користування будь-якими користувачами, для аналізу використовуються сучасні інструменти та засоби.

1.3 Огляд застосування прогнозувань відвідуваності для вебсайтів

Прогнозування відвідуваності вебсайтів є досить складним завданням для розробників. Оскільки потрібно оперувати досить великою кількістю даних, які включають різноманітні найважливіші фактори, які стосуються обраної тематики. Тому слід розглянути саме застосування прогнозувань відвідуваності вебсайтів в уже готових програмних рішеннях.

Одним з найпопулярніших вебсервісів, які реалізують прогнозування відвідуваності вебсайтів є «Similarweb» [25].

«Similarweb» використовує ряд методів та підходів для прогнозування відвідуваності вебсайтів. Сервіс аналізує історичні дані про трафік вебсайту, використовуючи їх для розрахунку тенденцій та патернів відвідуваності. Програма використовує моделі машинного навчання для прогнозування майбутньої відвідуваності на основі історичних даних та інших факторів, таких як сезонність, рекламні кампанії та інші. Також сервіс враховує ключові показники, такі як кількість унікальних відвідувачів, перегляди сторінок (рисунок 1.4), час перебування на сайті та інші, для прогнозування загальної відвідуваності. Сервіс аналізує діяльність конкурентів та інші зовнішні фактори, щоб урахувати їх вплив на прогнозування відвідуваності.

Ці методи дозволяють «Similarweb» надавати користувачам точні та корисні прогнози відвідуваності вебсайтів, що допомагає їм приймати обґрунтовані рішення з планування та оптимізації своєї вебприсутності.

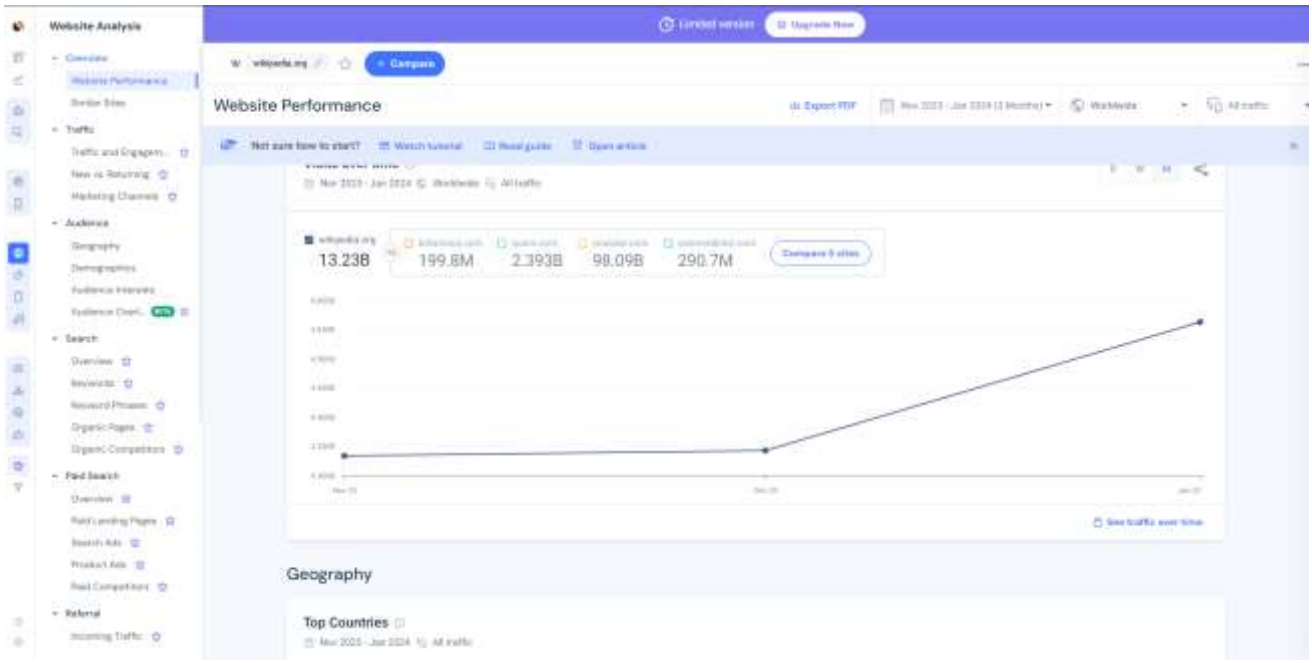


Рисунок 1.4 – Сторінка з різноманітною аналітикою та прогнозами вебсайту «Similarweb»

Наступним сервіс, який слід розглянути – це «Serpstat» [26].

«Serpstat» – це інструмент для аналізу SEO та маркетингу, який надає широкий спектр функцій для вивчення вебтрафіку, ключових слів, конкурентів та інших аспектів вебприсутності. «Serpstat» дозволяє знаходити популярні ключові слова для власного бізнесу (рисунок 1.5) та аналізувати їх ефективність у привертанні трафіку. За допомогою сервісу можна порівнювати власний вебсайт з конкурентами за різними показниками, щоб зрозуміти їх стратегії та підвищити свою конкурентоспроможність.

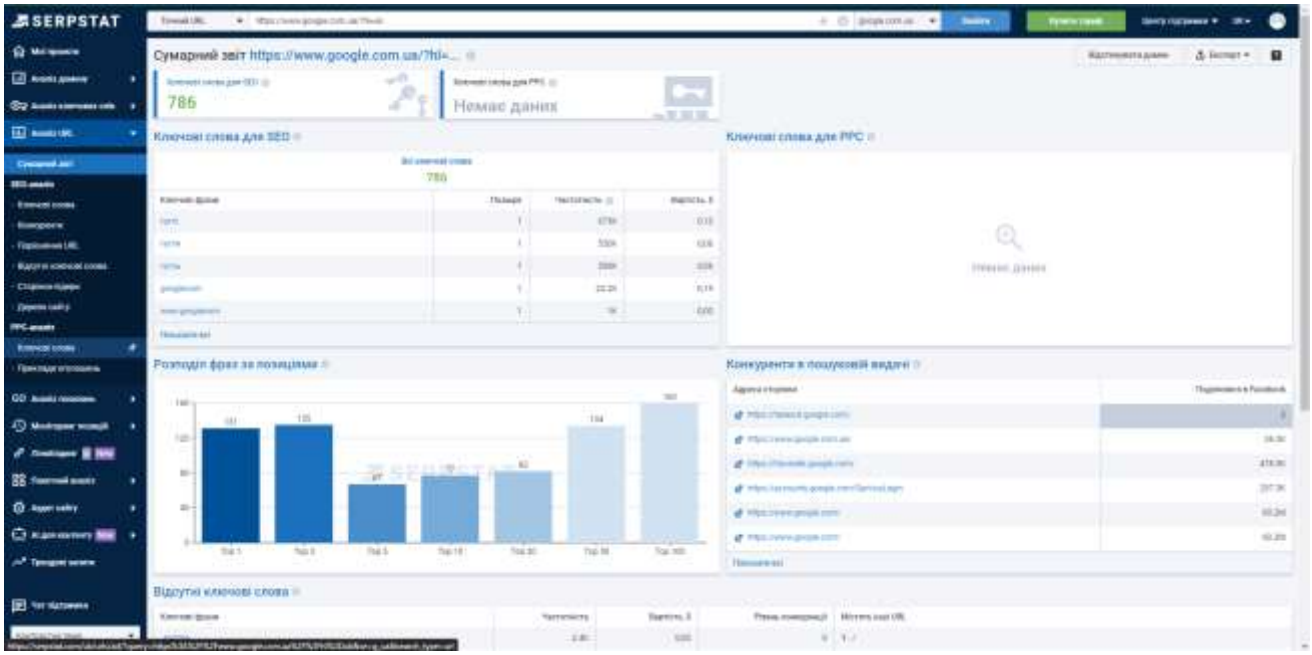


Рисунок 1.5 – Сторінка з різноманітною аналітикою ключових слів та конкурентами вебсайту «Serpstat»

Вебресурс дозволяє знаходити важливі об'єкти для власного бізнесу, такі як ключові слова, конкуренти. «Serpstat» дозволяє відстежувати зміни в рейтингу вебсайту в пошукових системах для ключових слів. Вебсервіс використовує дані про ключові слова та трафік для розрахунку очікуваної відвідуваності вебсайту на основі розташування вашого сайту у пошукових результатах для цих ключових слів. Дослідивши трафік вашого вебсайту порівняно з конкурентами, «Serpstat» надає зрозуміти, як можна покращити власну стратегію та збільшити трафік.

Отже, провівши дослідження, щодо існуючих реалізацій прогнозування відвідуваності вебсайтів, які використовуються в популярних вебсервісах, визначено, що найефективнішим способом прогнозування відвідуваності вебсайтів може бути використання ансамблевих моделей машинного навчання.

Важливо також враховувати, що прогнозування відвідуваності – це складний процес, і точність прогнозів може залежати від якості та кількості вхідних даних, а також від обраної моделі та методів її навчання.

1.4 Мета та завдання

Метою кваліфікаційної роботи бакалавра є покращення ефективності прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу реалізованого у вигляді вебсервісу. Для досягнення зазначеної мети потрібно виконати наступні завдання:

1) провести аналіз предметної області прогнозування відвідуваності вебсайтів; виконати аналіз методів прогнозування за допомогою нейромережевого підходу; виконати аналіз наявних рішень щодо подібних задач;
2) розробити метод прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу;
3) спроєктувати структуру вебресурсу з автоматизованим прогнозуванням відвідуваності вебсайтів за допомогою нейромережевого підходу;
4) спроєктувати структуру бази даних вебресурсу;
5) виконати програмну реалізацію вебресурсу;
6) провести тестування розробленого вебресурсу;
7) дослідити ефективність розробленого методу прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу.

Розділ 2 Метод прогнозування відвідуваності вебсайтів нейромережевими засобами

2.1 Підходи до прогнозування відвідуваності вебсайтів

Архітектура нейронної мережі для прогнозування відвідуваності вебсайтів є критичною складовою в процесі розробки ефективної та точної моделі прогнозування. Нейронні мережі, зокрема глибокі нейронні мережі, здатні автоматично виявляти складні зв’язки з даними та виконувати нелінійні перетворення, що робить їх дуже потужними інструментами для розв’язання завдань прогнозування. Важливо зазначити, що структура нейронної мережі повинна бути підібрана таким чином, щоб уникнути перенавчання (перетренування) чи недонавчання (недооцінки).

У кваліфікаційній роботі було розглянуто декілька підходів, які можуть використовуватись для прогнозування відвідуваності вебсайтів.

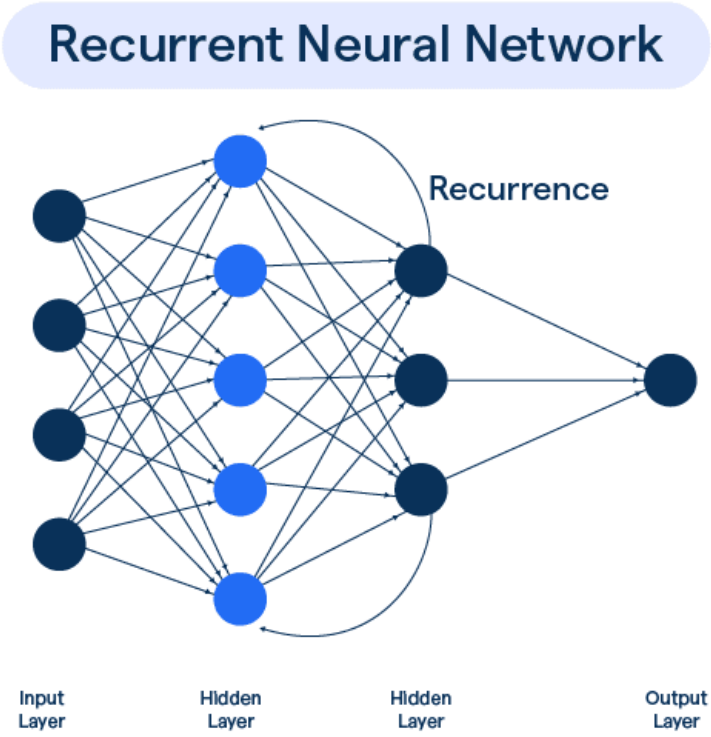


Рисунок 2.1 – Архітектура RNN [27]

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) є ефективним інструментом для роботи з послідовними даними, такими як часові ряди або текстова інформація. Це робить їх ідеальними для аналізу часових рядів відвідуваності вебсайтів та текстового контенту, що міститься на них. RNN організовані в шари (рисунок 2.1), які складаються з взаємопов’язаних вузлів або нейронів. У простій RNN вихід з одного рівня стає входом для того самого рівня, коли дані переміщуються з одного часового кроку до наступного. Ця петля зворотного зв’язку дозволяє RNN зберігати внутрішній стан між часовими кроками, дозволяючи їм запам’ятовувати попередні виходи та вчитися на них. Отже, цей ітеративний процес зворотного зв’язку дає RNN можливість фіксувати тимчасові залежності та розпізнавати закономірності в часі. Зокрема, RNN можуть використовуватися для моделювання темпоральних залежностей між відвідуванням сторінок та аналізу коментарів користувачів. Основними перевагами RNN є їхня здатність до роботи з послідовними даними та автоматичне виявлення залежностей у вхідних даних. Однак варто відзначити, що у них можуть виникати проблеми з вивченням довгих залежностей, що може вплинути на їхню точність, а також управління градієнтами під час навчання, що може зробити процес тривалим та складним.

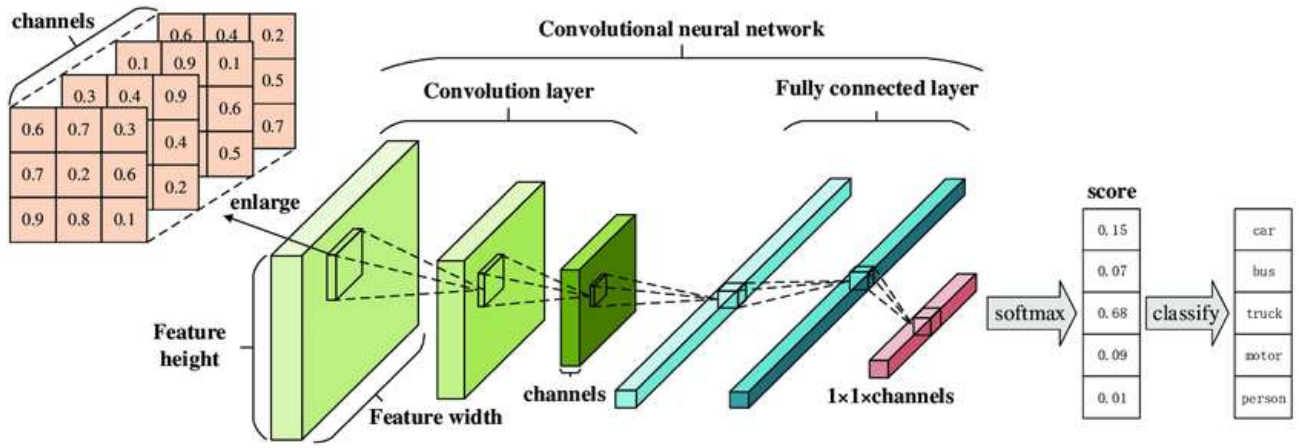


Рисунок 2.2 – Архітектура CNN [28]

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) представляють собою потужний інструмент у сфері аналізу зображень та

послідовностей даних. Традиційна структура CNN (рисунок 2.2) в основному складається з шарів згортки, шарів об’єднання, повністю зв’язаних шарів і деяких функцій активації. Кожне ядро згортки пов'язане з частиною карт функцій. Вхід з’єднаний з усіма вихідними елементами в повністю зв’язаному шарі. У контексті прогнозування відвідуваності вебсайтів, CNN можуть використовуватися для аналізу часових рядів, обробки зображень сторінок вебсайту або аналізу текстової інформації, що міститься на вебсторінках. Головною перевагою CNN є їхня висока швидкодія та здатність автоматично виявляти важливі ознаки у вхідних даних, що дозволяє здійснювати точні прогнози. Однак слід зазначити, що для успішного навчання моделі потрібно мати значну кількість даних, а також налаштувати нейромережу, що може бути складним завданням.

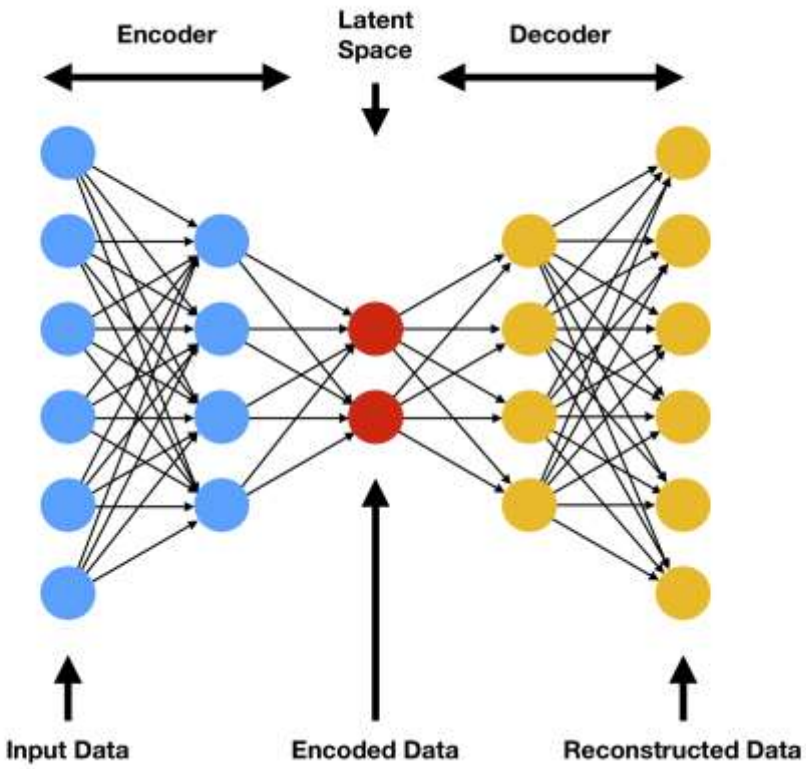


Рисунок 2.3 – Архітектура VAE [29]

Варіативні автоенкодери (Variational Autoencoders, VAE) - це тип нейромереж, які навчаються розпізнавати внутрішню структуру даних та генерувати нові екземпляри цих даних (рисунок 2.3). Вони можуть бути

використані для прогнозування відвідуваності вебсайтів, розпізнавання патернів в користувацькому поведінці та генерації нових даних для аналізу. Перевагами VAE є здатність до автоматичного вивчення внутрішньої структури даних та генерації нових прикладів даних. Однак недоліками можуть бути складність у налаштуванні та вимоги до обчислювальних ресурсів під час навчання.

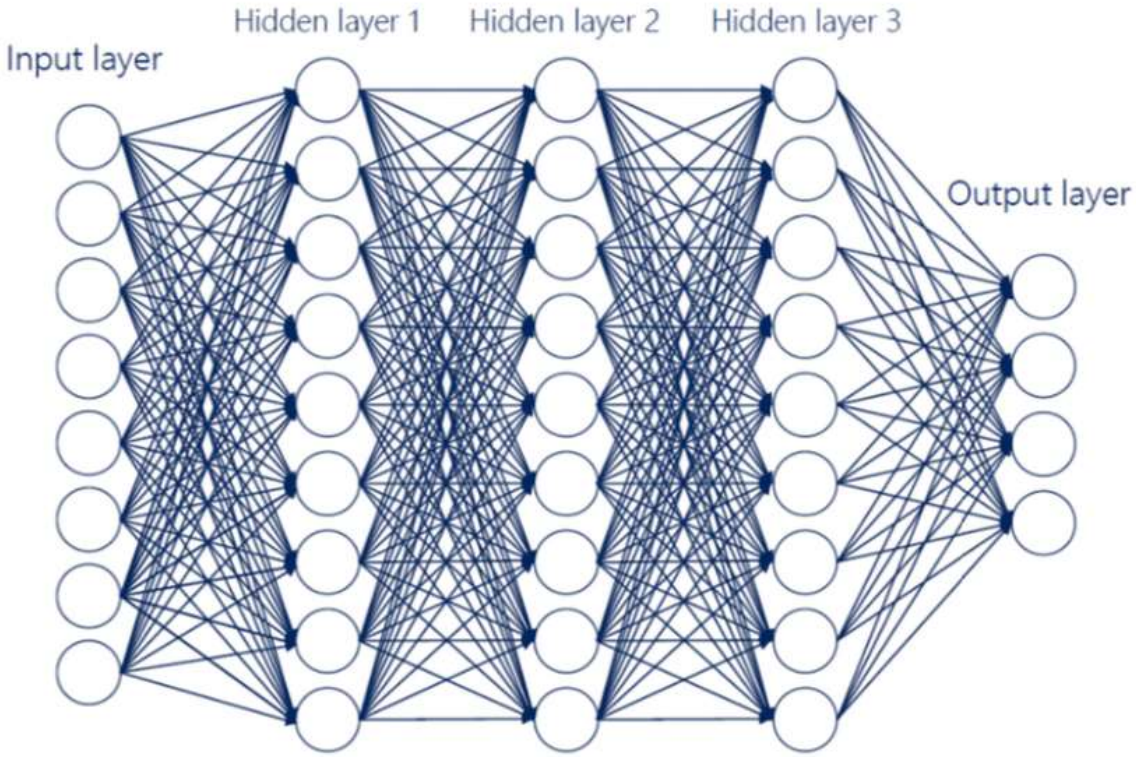


Рисунок 2.4 – Архітектура DNN [30]

Глибокі нейромережі (Deep Neural Networks, DNN) є спеціалізованими моделями, призначеними для використання глибокого навчання у сфері аналізу та прогнозування відвідуваності вебсайтів. Вони характеризуються значною кількістю шарів та параметрів (рисунок 2.4), що дозволяє їм ефективно адаптуватися до різних видів даних та складних залежностей. Перевагами DNN є висока точність прогнозування та здатність працювати з великими обсягами даних. Однак варто врахувати, що вони можуть потребувати складної настройки та високих вимог до обчислювальних ресурсів під час процесу навчання.

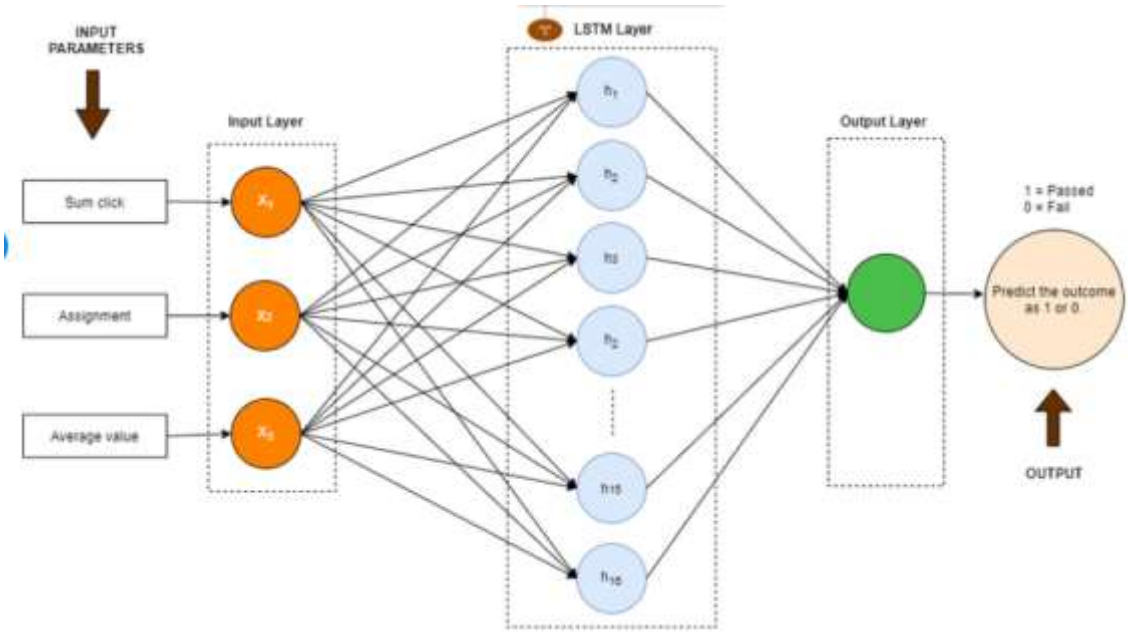


Рисунок 2.5 – Архітектура LSTM [31]

Довготривала короткострокова пам'ять (Long Short-Term Memory, LSTM) є особливим типом рекурентних нейронних мереж (рисунок 2.5), який спеціально розроблений для роботи з послідовними даними та уникнення проблеми зникаючих і вибуваючих градієнтів. Вони відзначаються ефективністю у роботі з послідовностями будь-якої довжини та здатністю моделі враховувати довгострокові залежності у даних. Перевагами LSTM є їхня здатність до роботи з довгими залежностями та висока точність прогнозів. Однак слід зазначити, що вони можуть мати потребу у складній настройці параметрів моделі для досягнення оптимальних результатів.

Проаналізувавши кожен з цих нейромережевих підходів, їхні переваги і недолікіи, для прогнозування відвідуваності вебсайтів було вирішено обрати різновид архітектури рекурентних нейромереж - LSTM. Адже LSTM відзначається ефективністю у роботі з послідовними даними, здатністю до врахування довгострокових залежностей та уникненням проблеми зникаючих і вибуваючих градієнтів, що є важливими аспектами в аналізі вебтрафіку.

2.2 Модель глибокого навчання для прогнозування та основні кроки методу

Для розробки методу для прогнозування відвідуваності вебсайтів було обрано бібліотеку Keras (рисунок 2.6). Keras є високорівневим інтерфейсом для побудови та навчання нейронних мереж. Вона має ряд переваг, які роблять її привабливим вибором для реалізації нейронних мереж у багатьох додатках, включаючи прогнозування відвідуваності вебсайтів. Однією з переваг Keras є його простота використання та зрозумілість. Інтерфейс Keras розроблений таким чином, щоб бути інтуїтивно зрозумілим навіть для початківців у галузі машинного навчання. Він надає зручний API для створення та конфігурування нейронних мереж за допомогою невеликої кількості коду, що робить процес розробки швидким та ефективним.

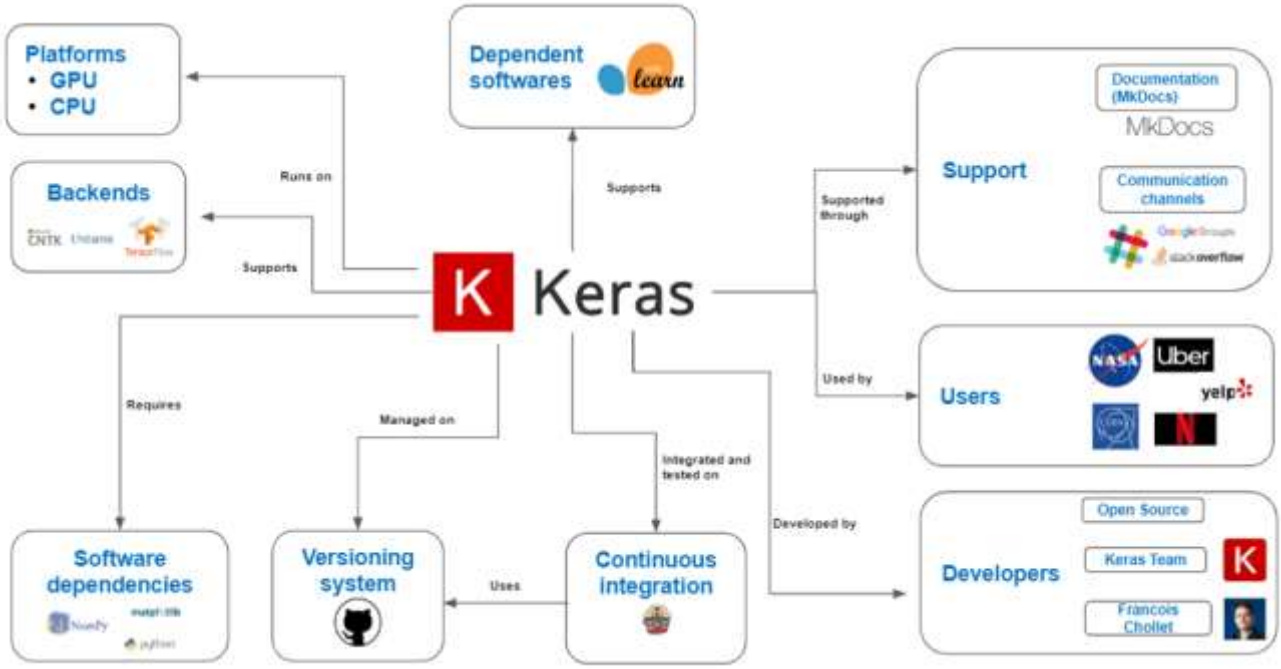


Рисунок 2.6 – Варіанти використання бібліотеки Keras [32]

Крім того, Keras має широкий спектр готових моделей та шаблонів, що дозволяє швидко створювати та навчати нейронні мережі для різних завдань, включаючи класифікацію, регресію, генерацію тексту, обробку зображень тощо.

Ці готові моделі можуть бути використані як основа для подальшої розробки та налаштування моделі для конкретної задачі прогнозування відвідуваності вебсайтів. Keras інтегрується з багатьма іншими потужними бібліотеками та інструментами для розробки та візуалізації нейронних мереж, такими як TensorFlow, PyTorch, Scikit-learn, Matplotlib тощо. Це дозволяє використовувати всі можливості цих інструментів у поєднанні з простотою та зручністю Keras для реалізації складних та потужних моделей. Ще однією важливою перевагою Keras є його гнучкість та масштабованість. Вона дозволяє створювати різноманітні архітектури нейронних мереж, від простих одношарових персептронів до складних глибоких нейронних мереж з багатьма шарами та складними зв'язками. Крім того, Keras підтримує розподілене навчання на багатьох процесорах або графічних процесорах (GPU), що дозволяє прискорити процес навчання нейронних мереж та обробки великих обсягів даних.

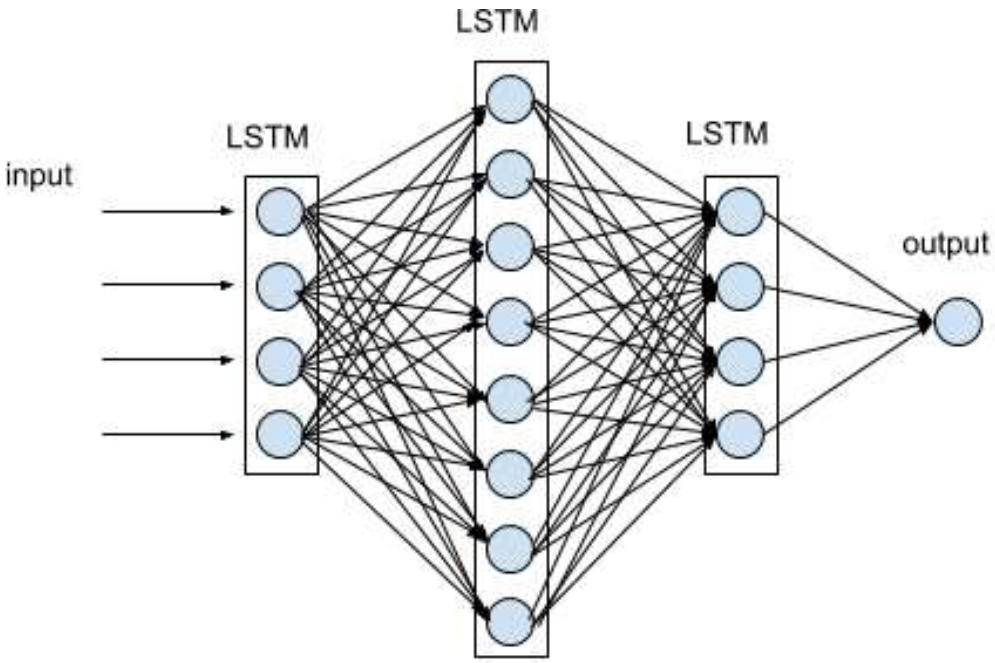


Рисунок 2.7 – Використання бібліотеки Keras з архітектурою LSTM [33]

Для розробки методу прогнозування відвідуваності вебсайтів було обрано бібліотеку Keras з архітектурою LSTM (Long Short-Term Memory) (рисунок 2.7). LSTM є спеціальним типом рекурентних нейронних мереж, який спроектований

для обробки послідовних даних з довготривалими залежностями між елементами послідовності. Це робить їх ефективними для аналізу часових рядів та тексту, а також для прогнозування вебтрафіку та поведінки користувачів.

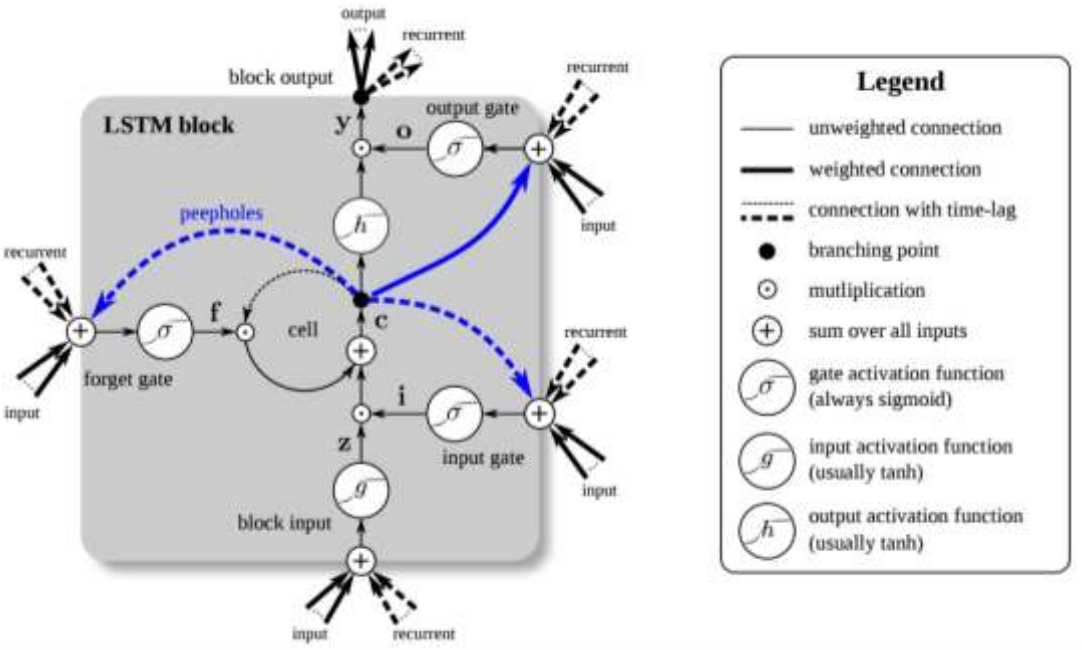


Рисунок 2.8 – Будова клітини LSTM шару [34]

Вхідний LSTM шар приймає послідовний вектор даних з фіксованою довжиною. LSTM шари складаються з множини клітин (рисунок 2.8), кожна з яких зберігає внутрішній стан та має важливі вентиляційні механізми (входу, виходу, забування), які дозволяють регулювати потік інформації. Вихідний шар генерує прогнозовані значення відвідуваності вебсайтів на основі внутрішнього стану LSTM шарів. Використання LSTM дозволяє ефективно моделювати та передбачати складні та довготривалі шаблони у відвідуваності вебсайтів, включаючи сезонні та трендові зміни, а також вплив різних факторів на поведінку користувачів.

Основні кроки запропонованого методу прогнозування відвідуваності вебсайтів зображено на рисунку 2.9.

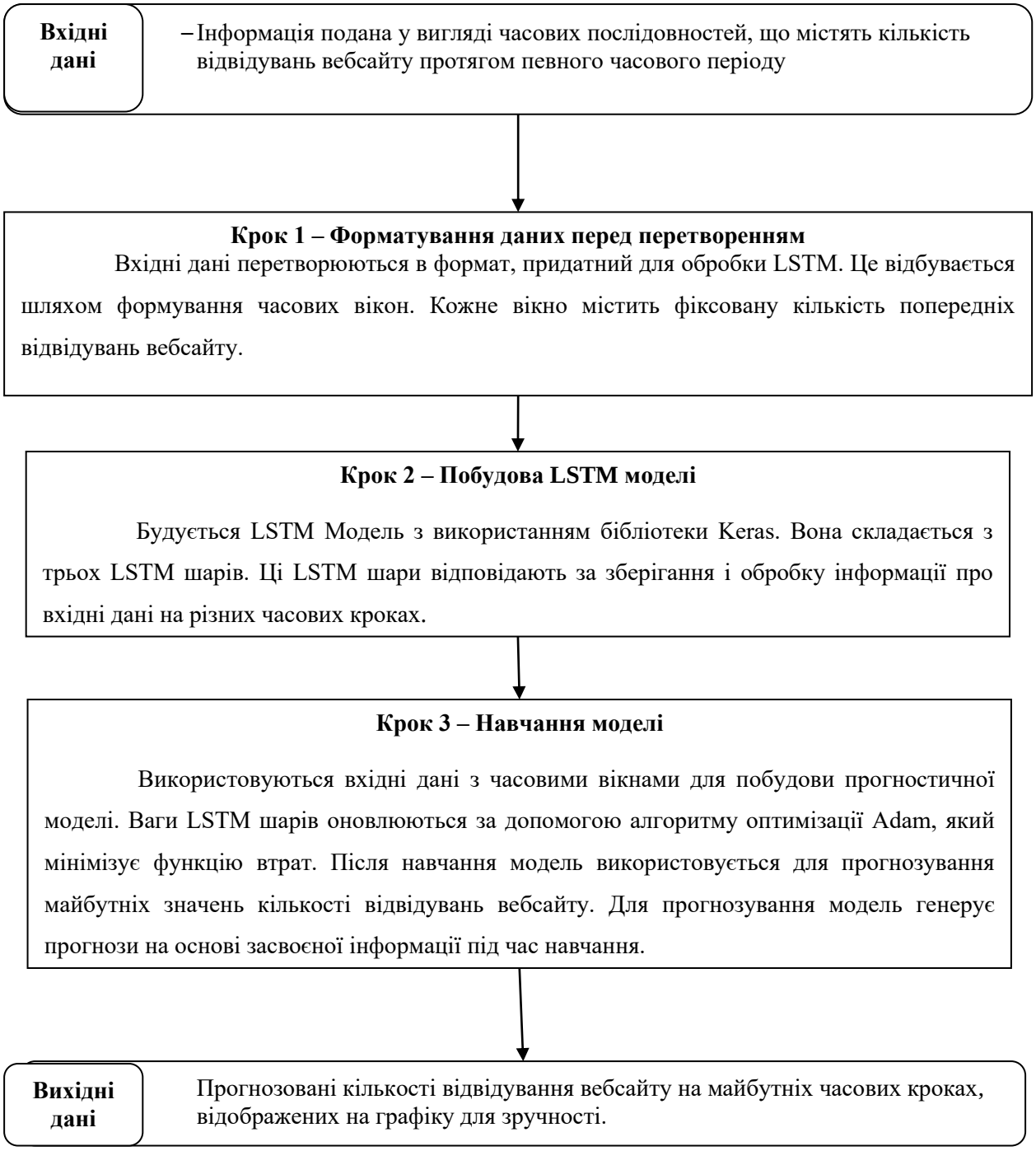


Рисунок 2.9 – Основні кроки методу прогнозування відвідуваності вебсайтів

Однією з ключових переваг використання LSTM є їх здатність до врахування довготривалих залежностей в даних. У контексті прогнозування відвідуваності вебсайтів це означає, що LSTM можуть ефективно моделювати та передбачати складні залежності між відвідуванням сайту на протязі тривалого часу, враховуючи різноманітні фактори, які впливають на цей процес.

Ще однією перевагою використання LSTM є їх здатність до врахування контексту та попередньої інформації. LSTM здатні запам'ятовувати та використовувати інформацію з попередніх кроків в послідовності для прийняття рішення на поточному кроці. Це дозволяє їм ефективно враховувати попередні патерни та тенденції в поведінці користувачів для прогнозування майбутніх відвідувань сайту. Крім того, LSTM мають вбудований механізм керування, який дозволяє регулювати ступінь важливості різних вхідних сигналів в процесі прийняття рішення. Це дозволяє їм адаптуватися до змін в умовах та оточенні та підтримувати високу точність прогнозування навіть у динамічних умовах.

Отже, моделлю глибокого навчання для прогнозування відвідуваності вебсайтів було обрано архітектуру LSTM, реалізовану у бібліотеці Keras. Це зумовлено простотою використання, гнучкістю, масштабованістю та підтримкою активною спільнотою, що є ідеальним вибором для широкого спектру завдань машинного навчання та аналітики даних. За допомогою архітектури LSTM, реалізованої у бібліотеці Keras реалізовано метод для прогнозування відвідуваності вебсайтів.

2.3 Особливості отримання вхідної інформації для прогнозування відвідуваності вебсайтів

Процес отримання вхідної інформації для прогнозування відвідуваності вебсайтів від користувача включає можливість введення даних вручну по днях або місяцях, а також імпорт даних з файлу. Кожен з цих методів має свої переваги та недоліки, і вибір підходу може залежати від специфіки сайту користувача та обсягу даних.

Ручне введення дозволяє користувачу відразу включати всі наявні дані безпосередньо у систему, що може бути корисним, якщо обсяг даних невеликий або якщо дані формуються в реальному часі і не можуть бути автоматично зібрані. Однак ручне введення даних також має свої недоліки. Воно може бути часо- та працезатратним, особливо якщо обсяг даних великий або якщо дані

оновлюються регулярно. Крім того, цей підхід підвищує ризик помилок та неточностей у даних через людський фактор. Наприклад, можливі помилки при введенні кількості відвідувань, що може призвести до неточностей у прогнозуванні та аналізі результатів.

Наступний метод отримання вхідної інформації є імпорт даних з файлу. Цей підхід передбачає завантаження файлу з вже зібраними та структурованими даними у систему прогнозування. Основна перевага цього методу полягає в його ефективності та точності. Великі обсяги даних можуть бути легко завантажені в систему одним махом, що дозволяє економити час та зусилля операторів. Крім того, імпорт даних з файлу може знизити ризик помилок та неточностей, оскільки дані вже попередньо структуровані та перевірені. Однак імпорт даних з файлу може мати свої обмеження. Наприклад, для успішного імпорту файл повинен мати певний формат та структуру даних, що може вимагати певної підготовки даних перед завантаженням. Крім того, цей підхід може бути не дуже практичним для систем, де дані постійно оновлюються або генеруються в реальному часі, оскільки потребує постійного завантаження файлів.

Вибір методу отримання вхідної інформації для прогнозування відвідуваності вебсайтів здійснюється з урахуванням конкретних потреб та обмежень проекту. Іноді комбінація обох підходів може бути найбільш ефективною, дозволяючи операторам вводити дані вручну в реальному часі, а також регулярно імпортувати великі обсяги даних з файлів для аналізу та прогнозування. У будь-якому випадку, користувачу важливо забезпечити точність вхідної інформації для забезпечення якісних результатів прогнозування та аналізу.

У рамках даного проєкту для прогнозування відвідуваності вебсайтів використовуються такі типи даних як дата та кількість відвідувань. Ці дві основні категорії даних є ключовими у вивченні та аналізі активності користувачів на сайті. Дати надають можливість визначити зміни у відвідуваності сайту з плином часу, а також виявити сезонні та інші патерни в поведінці користувачів. З іншого боку, кількість відвідувань є метрикою, яка

вказує на активність на сайті та його популярність серед відвідувачів. Аналізуючи ці дві категорії даних, можна розробити стратегії залучення нових відвідувачів, оптимізувати контент та функціонал сайту для підвищення його привабливості та ефективності. Одним з ключових аспектів використання цих типів даних є їх потужний аналітичний потенціал. Аналізуючи дати відвідування, можна виявити деякі стабільні тенденції у відвідуваності сайту, такі як пікові часи активності у вихідні дні, тощо. Це дозволяє адаптувати стратегії маркетингу та залучення нових користувачів у відповідь на ці тенденції та максимізувати ефективність зусиль. Аналіз кількості відвідувань дозволяє виявити ті сторінки чи розділи сайту, які є найбільш популярними серед користувачів, та спрямовувати ресурси на їх подальший розвиток та оптимізацію.

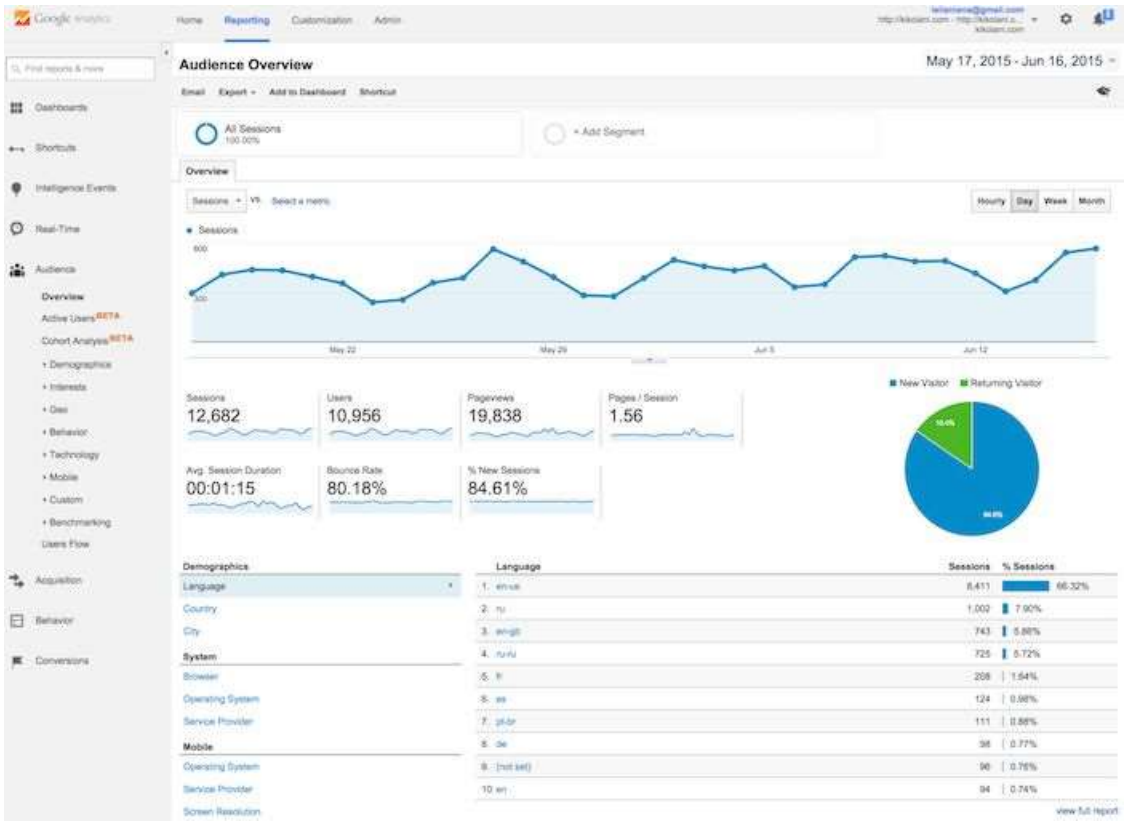


Рисунок 2.10 – Результати роботи Google Analytics

Важливою перевагою цих типів даних є їх доступність та простота обробки. Дати відвідування та кількість відвідувань можуть бути легко зібрані та

збережені за допомогою різноманітних аналітичних інструментів, таких як Google Analytics (рисунок 2.10). Крім того, ці дані зазвичай не вимагають складних технік обробки що робить їх доступними для використання навіть для початківців у галузі аналітики даних.

Однак, важливо враховувати, що обмеженість типів даних також може мати свої недоліки. Наприклад, вони можуть не забезпечити повного зображення про всі аспекти відвідуваності сайту. Деякі важливі фактори, такі як джерело трафіку, конверсія або показники залучення, можуть бути пропущені при використанні лише цих двох типів даних. Тому для отримання повного розуміння поведінки користувачів та ефективного управління сайтом може бути необхідно використовувати додаткові аналітичні інструменти.

Загалом, використання дат та кількості відвідувань для прогнозування відвідуваності вебсайтів є ефективним та економічно обґрунтованим рішенням. Ці дані дозволяють отримувати цінну інформацію про активність користувачів та виявляти тенденції у їх поведінці, що дозволяє розробляти ефективні стратегії маркетингу та управління сайтом.

2.4 Особливості реалізації методу прогнозування відвідуваності у вигляді вебсервісу

Вебсервіс - це програмне забезпечення, яке надає можливість взаємодії з клієнтами чи системами через мережу Інтернет. Він приймає запити, обробляє їх та повертає результати у вигляді відповідей. Вебсервіси можуть використовуватися для різноманітних цілей, включаючи обмін даними, автоматизацію бізнес-процесів та інтеграцію з іншими системами.

Однією з особливостей реалізації методу прогнозування у вигляді вебсервісу є можливість доступу до нього через інтернет з будь-якого пристрою з підключенням до мережі. Це означає, що користувачі можуть отримувати прогнози відвідуваності сайту в реальному часі без необхідності встановлення спеціального програмного забезпечення або обробки даних на своєму пристрої.

Такий підхід робить систему більш доступною та зручною для використання, особливо для тих, хто не має технічних навичок у роботі з аналітичними програмами.

Система може підтримувати різні формати вхідних даних, що дозволяє користувачам використовувати різні джерела даних для прогнозування. Наприклад, вона може приймати дані у форматі JSON, CSV або інші формати, що дозволяє інтегрувати систему з різними джерелами даних, такими як бази даних, дата-фрейми Python, або зовнішні вебсервіси. Ще однією важливою особливістю реалізації методу є його масштабованість. LSTM модель, побудована з використанням Keras, може бути легко масштабована для обробки великих обсягів даних та виконання складних обчислень. Це дозволяє розробникам побудувати потужну систему прогнозування, яка здатна обробляти великі обсяги даних та надавати точні та швидкі прогнози. Також є можливість налаштування параметрів моделі та алгоритму прогнозування в реальному часі. Це дозволяє операторам системи швидко реагувати на зміни у вебтрафіку та виправляти модель для підвищення точності прогнозів. Наприклад, вони можуть змінювати кількість шарів LSTM, розмір пакету (batch size), швидкість навчання (learning rate) та інші параметри для досягнення кращої проективної точності.

На рахунок безпеки, що є ще одною особливістю розробленого вебсервісу, використовується Spring Security. Spring Security - це потужний інструмент для захисту вебзастосунків та мікросервісів, який надає різноманітні механізми аутентифікації та авторизації, а також захист від різноманітних загроз веббезпеки. Однією з ключових особливостей Spring Security є його можливість легко інтегруватися з існуючими програмними рішеннями та стандартами. Він підтримує різні механізми аутентифікації, такі як базова аутентифікація, формування, OAuth, OpenID Connect тощо. Це дозволяє розробникам вибрати найбільш підходящий метод аутентифікації для їхнього вебсервісу в залежності від вимог безпеки та специфіки проекту. Крім того, Spring Security надає широкий набір можливостей для конфігурації прав доступу та авторизації. За допомогою анотацій або конфігураційних файлів розробники можуть визначити,

хто має доступ до конкретних ресурсів чи операцій у вебсервісі. Це дозволяє створювати різні ролі користувачів та надавати їм відповідні права доступу до функціоналу системи. Однією з особливостей Spring Security є його можливість захисту від різноманітних загроз веббезпеки, таких як перехоплення сесій, атаки на переповнення буфера, міжсайтовий скриптінг (XSS), міжсайтовий підтрюмувачний запит (CSRF) та інші. Він надає вбудовану підтримку для захисту від цих атак, а також дозволяє розробникам налаштовувати правила безпеки відповідно до вимог проекту. Spring Security також забезпечує можливість використання шифрування для захисту конфіденційних даних, таких як паролі користувачів чи конфіденційна інформація, що передається між вебсервісом та клієнтом. За допомогою інтеграції з бібліотеками шифрування, такими як BCrypt, розробники можуть забезпечити безпеку та конфіденційність даних в системі. Це дозволяє підтримувати єдиний центр авторизації та ідентифікації (SSO) для додатків у великих організаціях, а також спрощує управління користувачами та їхніми правами. Коротко кажучи, Spring Security - це потужний інструмент для захисту вебсервісів, який надає розробникам широкий набір можливостей для забезпечення безпеки та конфіденційності даних. Він надає зручні інструменти для аутентифікації та авторизації, захист від різноманітних загроз веббезпеки та інтеграцію з існуючими системами безпеки. Таким чином, використання Spring Security допомагає забезпечити високий рівень безпеки та надійності вебсервісу.

Додатковою особливістю є можливість використання технологій управління ресурсами, таких як контейнеризація з Docker або оркестрація з Kubernetes, що дозволяє легко масштабувати та керувати ресурсами вебсервісу в залежності від навантаження. Важливою особливістю є можливість моніторингу та аналізу роботи вебсервісу. Збір та аналіз логів, моніторинг метрик продуктивності та роботи моделі дозволяють операторам системи вчасно виявляти проблеми та вдосконалювати роботу системи. Ці особливості реалізації методу прогнозування у вигляді вебсервісу роблять систему потужним та ефективним інструментом для прогнозування відвідуваності вебсайтів. Вони

дозволяють налаштовувати та масштабувати модель для кращої точності прогнозів, а також забезпечують легку інтеграцію з іншими системами та сервісами.

Узагальнюючи всі перераховані особливості можна сказати, що запропонований метод прогнозування вебсайтів у вигляді вебсервісу забезпечує доступність, безпеку, масштабованість та автоматизацію процесу прогнозування, що робить його привабливим для широкого кола користувачів.

2.5 Проектна архітектура вебсервісу та взаємозв’язок компонентів

Вебсервіс для прогнозування відвідуваності вебсайтів використовує мікросервісну архітектуру, що складається з мікросервісів на Java Spring, Python з бібліотеками Keras LSTM та React JS, а також їхній взаємозв'язок та способи взаємодії. Мікросервісна архітектура є сучасним підходом до розробки програмного забезпечення, який передбачає розбиття системи на невеликі та незалежні компоненти, що називаються мікросервісами. Кожен мікросервіс виконує конкретну функцію.

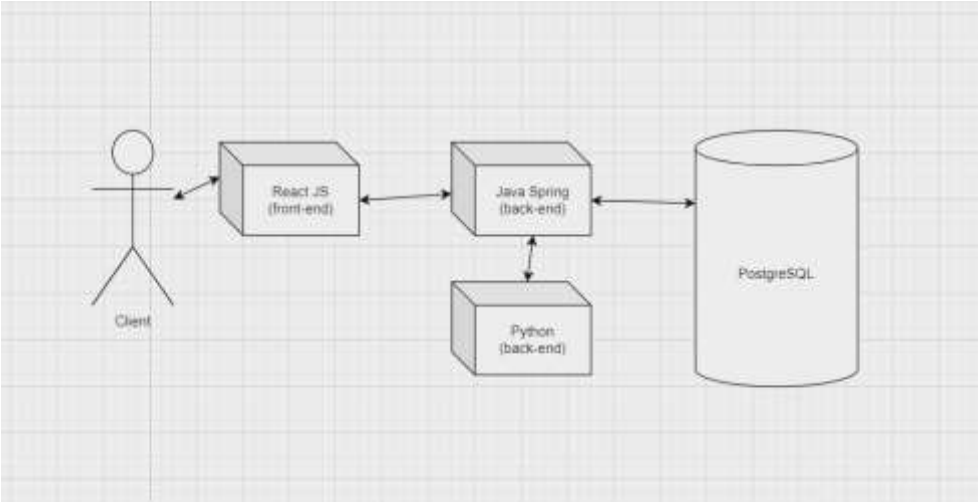


Рисунок 2.11 – Мікросервісна архітектура вебсервісу для прогнозування відвідування вебсайтів.

На рисунку 2.11 видно три мікросервіси з яких складається вебсервіс для прогнозування відвідуваності вебсайтів, де один мікросервіс написаний на React JS і відповідає за фронт-енд частину вебсервісу, а два інших – за бек-енд, а саме мікросервіс на Java Spring та мікросервіс на Python з використанням Keras LSTM для прогнозування.

Загальна взаємодія між цими трьома мікросервісами (рисунок 2.10) є критичним елементом архітектури системи прогнозування відвідуваності вебсайтів. Кожен мікросервіс виконує свої унікальні завдання, а також спільно працює з іншими для забезпечення функціональності вебсервісу. Далі буде детальніше розглянуто кожен аспект взаємодії між ними та вплив цього процесу на загальну ефективність системи. Перший мікросервіс відповідає за прийом та обробку запитів від клієнтів. Він служить точкою входу в систему, де клієнти можуть надсилати свої дані та отримувати відповіді на свої запити. Цей мікросервіс використовує Java Spring та взаємодіє з базою даних PostgreSQL для зберігання та обробки інформації. Після отримання запитів від клієнтів, він пересилає ці дані до другого мікросервісу для проведення аналізу та прогнозування. Другий мікросервіс, реалізований на мові програмування Python та використовуючи бібліотеки Keras LSTM, виконує завдання аналізу та прогнозування відвідуваності вебсайтів. Він отримує дані від першого мікросервісу та використовує їх для побудови моделі прогнозування. Цей процес може включати обробку великої кількості даних та виконання складних аналітичних операцій. Після завершення аналізу він повертає результати до першого мікросервісу для подальшої обробки та використання. Третій мікросервіс відповідає за реалізацію фронтенду системи на React JS. Він взаємодіє з першим мікросервісом для отримання даних та відображення їх користувачам. Після того, як користувач взаємодіє з інтерфейсом, цей мікросервіс може генерувати користувацькі запити та інтерактивні дії, які надсилаються назад до серверної частини системи для подальшої обробки та відповіді на них.

Одним із ключових аспектів взаємозв'язку компонентів є механізми комунікації між мікросервісами. У архітектурі вебсервісу це відбувається за допомогою HTTP протоколу та RESTful API. Кожен мікросервіс має свої власні API точки доступу, через які він надсилає та отримує дані від інших сервісів. Це дозволяє створювати легко розширювані та модульні системи, в яких окремі компоненти можуть бути замінені або модифіковані без зміни інших компонентів.

Крім того, важливою частиною архітектури є забезпечення масштабованості та надійності системи. Для цього можуть використовуватися такі технології, як контейнеризація з Docker та оркестрація з Kubernetes, які дозволяють автоматично керувати та масштабувати окремі компоненти системи в залежності від навантаження.

Мікросервісна архітектура надає багато переваг, особливо у контексті розробки складних та великих програмних проектів. Однією з головних переваг є збільшена гнучкість та модульність. Розбиття системи на невеликі, самодостатні мікросервіси дозволяє розробникам працювати над окремими компонентами незалежно один від одного. Це полегшує тестування та розгортання нового функціоналу, а також зменшує ризик впровадження змін. Замість великої монолітної системи, де весь трафік та навантаження обробляється централізовано, у мікросервісній архітектурі кожен сервіс може бути масштабований та керувати своїми ресурсами незалежно. Це дозволяє забезпечити більшу доступність та швидкодію системи, а також легше виявляти та усувати проблеми в окремих компонентах. Оскільки кожен мікросервіс є незалежним, розробники можуть вибирати найкращі технології для кожного компонента

Перший мікросервіс у архітектурі вебсервісу для прогнозування відвідуваності вебсайтів реалізований на Java Spring та відповідає за обробку запитів від клієнтів, а також взаємодію з базою даних PostgreSQL. Java Spring є популярним фреймворком для розробки вебзастосунків та мікросервісів, який надає широкий набір інструментів та бібліотек для швидкої та ефективної

розробки. Цей мікросервіс отримує дані від клієнтів, обробляє їх та зберігає в базі даних для подальшого використання.

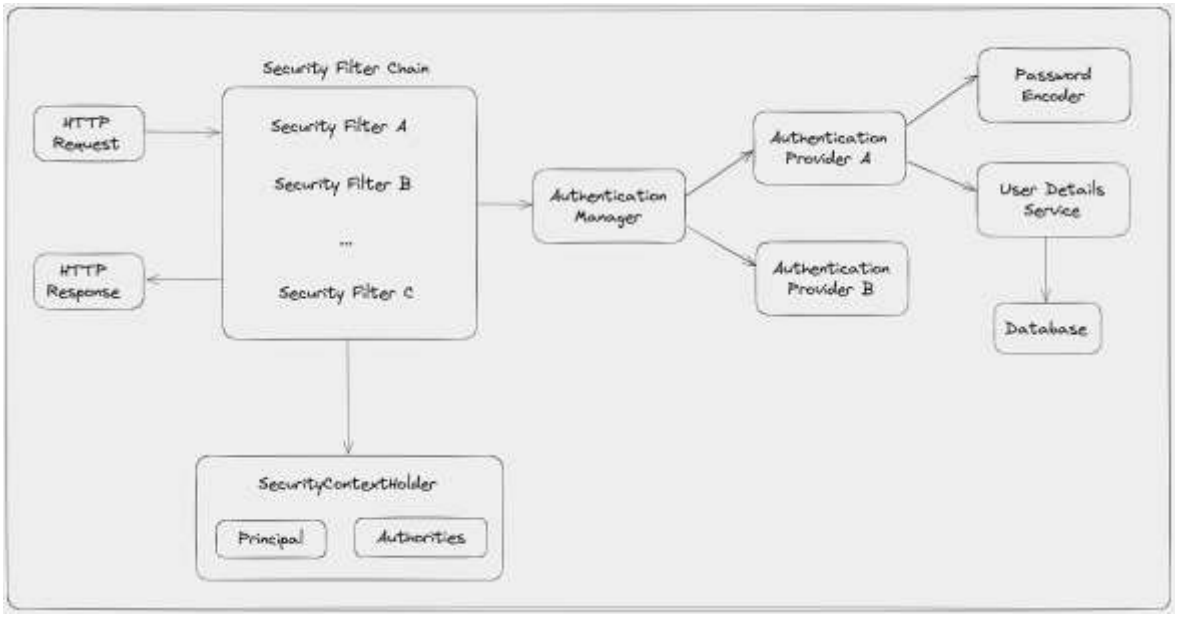


Рисунок 2.12 – Загальна архітектура Spring Security

Також він відповідальний за безпеку, яка забезпечується Spring Security (рисунок 2.12). Коли клієнт надсилає HTTP-запит на сервер, він спочатку потрапляє до Security Filter Chain - це ланцюжок фільтрів, які обробляють HTTP-запити. Вони виконують різні операції, такі як перевірка наявності користувача, перенаправлення на сторінку входу або перевірка прав доступу. Основний компонент, який відповідає за аутентифікацію користувачів це Authentication Manager. Він приймає аутентифікаційні об'єкти і намагається перевірити їхню достовірність. Authentication Manager спирається на Authentication Provider для перевірки аутентифікаційних даних. Authentication Provider може перевірити ідентичність користувача, використовуючи різні джерела, такі як база даних, LDAP або інші системи. Для безпеки паролів, Spring Security рекомендує використовувати кодувальники паролів. Паролі користувачів зберігаються в зашифрованому вигляді, що забезпечує їхню безпеку в разі витоку даних. Authentication Provider може звертатися до User Details Service для отримання додаткової інформації про користувачів. Нарешті, база даних є одним з можливих джерел для зберігання інформації про користувачів та їхніх даних

аутентифікації. Після успішної аутентифікації користувача, його основна інформація, така як ідентифікатор та список ролей, зберігається в SecurityContextHolder. Ця інформація може бути використана для подальшої авторизації та доступу до ресурсів. Взагалі, Spring Security надає гнучкі та потужні інструменти для забезпечення безпеки вебзастосунків, включаючи аутентифікацію, авторизацію та захист від різних видів атак.

Другий мікросервіс реалізований на мові програмування Python та використовує бібліотеки Keras LSTM для прогнозування відвідуваності вебсайтів. Python є популярною мовою програмування в галузі машинного навчання та аналізу даних, а бібліотеки Keras LSTM забезпечують потужні та ефективні інструменти для створення та навчання рекурентних нейронних мереж. Цей мікросервіс приймає дані з першого мікросервісу, виконує аналіз та прогнозування, а потім повертає результати назад для подальшого використання.

Третій мікросервіс відповідає за реалізацію фронтенду на React JS. React JS є потужним інструментом для створення інтерактивних та ефективних користувацьких інтерфейсів вебзастосунків. Цей мікросервіс взаємодіє з першим мікросервісом, надсилаючи запити та отримуючи дані для відображення користувачам. Крім того, він може обробляти користувацькі запити та інтерактивні дії, надсилаючи їх назад на сервер для обробки.

База даних, що використовується для проєкту (рисунок 2.13), складається з кількох таблиць, які забезпечують зберігання та організацію даних, необхідних для прогнозування відвідуваності вебсайтів. Перша таблиця, `nrl_traffic_prediction`, містить інформацію про самі прогнози. У ній зберігається ідентифікатор прогнозу, інформація про вебсайт, якому належить прогноз, дати вивчення та результати прогнозування. Зв'язок між цією таблицею та таблицею `sit_site` забезпечується через поле `site_id`. Таблиця `sit_site` містить дані про вебсайти. Кожен вебсайт має унікальний ідентифікатор, назву, доменне ім'я, ідентифікатор користувача, який є власником цього сайту, а також ідентифікатор типу вебсайту.

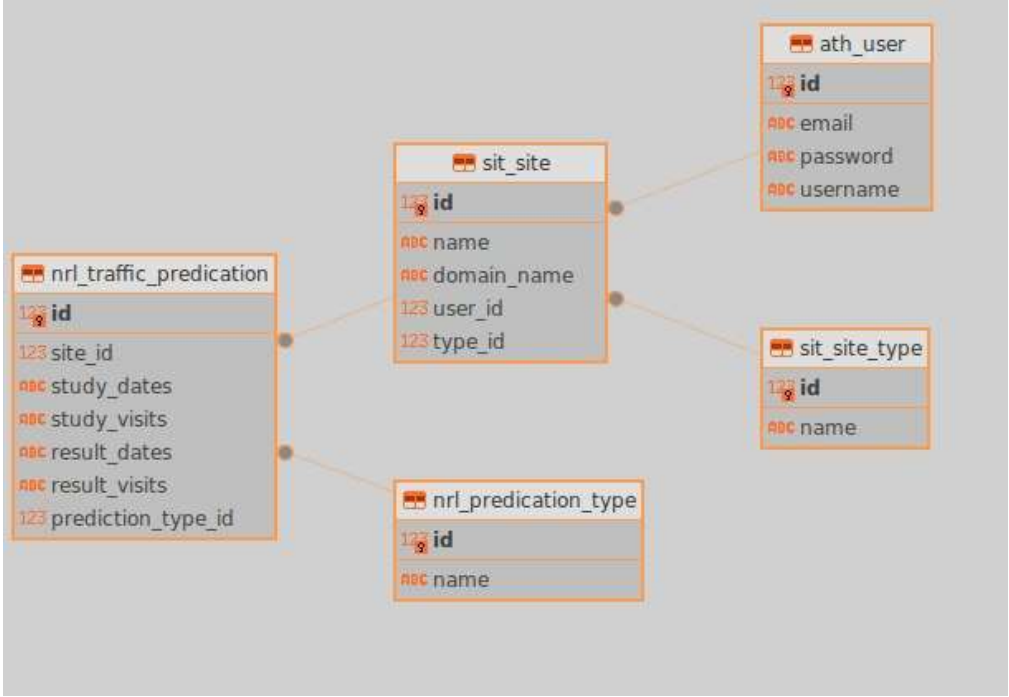


Рисунок 2.13 – База даних проєкту

Зв'язок із таблицею `ath_user` дозволяє встановити власника кожного вебсайту. Таблиця `ath_user` містить дані про користувачів системи. Кожен користувач має унікальний ідентифікатор, електронну адресу, пароль та ім'я. Ці дані дозволяють автентифікувати користувачів та встановлювати їхні права доступу. Таблиця `sit_site_type` містить дані про типи вебсайтів. Кожен тип вебсайту має унікальний ідентифікатор та назву. Ці дані допомагають класифікувати вебсайти та встановлювати специфічні характеристики кожного типу. Нарешті, таблиця `nrl_prediction_type` містить дані про типи прогнозів. Кожен тип прогнозу має унікальний ідентифікатор та назву. Ці дані допомагають встановлювати тип прогнозу, що використовується для кожного конкретного прогнозу в таблиці `nrl_traffic_prediction`.

Усі ці таблиці тісно взаємозв'язані, що дозволяє ефективно організовувати та зберігати дані, необхідні для виконання функцій системи прогнозування відвідуваності вебсайтів. Кожен запис у таблиці має відповідність в інших таблицях, що забезпечує цілісність та консистентність даних.

Важливо підкреслити, що ефективність всієї системи значною мірою залежить від успішної взаємодії між цими мікросервісами. Якщо яка-небудь

частина цього процесу несправна або має помилки, це може призвести до неправильних результатів прогнозування або збоїв у роботі системи в цілому.

Одним із ключових аспектів взаємодії є обмін даними між мікросервісами. Цей процес може включати передачу великих обсягів даних, які необхідно ефективно обробляти та передавати. Кожен мікросервіс може працювати незалежно та обробляти багато запитів одночасно, що збільшує швидкість та продуктивність системи в цілому. Механізми моніторингу та керування дозволяють ефективно відслідковувати та контролювати споживання ресурсів кожним мікросервісом, забезпечуючи оптимальну роботу системи. Були розроблені механізми виявлення та вирішення проблем у взаємодії між мікросервісами, що дозволяють оперативно реагувати на можливі збої та усувати їх, забезпечуючи стабільну роботу системи в будь-який час. Варто врахувати безпеку взаємодії між мікросервісами, яка була впроваджена у систему за допомогою Spring Security. З метою забезпечення конфіденційності та цілісності даних, використовується шифрування даних під час їх передачі між мікросервісами. Також реалізовані механізми автентифікації та авторизації користувачів. Крім того, система обладнана механізмами виявлення та захисту від потенційних атак, що дозволяє забезпечити надійний захист інформації та безпеку взаємодії між мікросервісами.

Підсумовуючи аспекти взаємодії між мікросервісами, система прогнозування відвідуваності вебсайтів забезпечує ефективну та надійну роботу в умовах змінної навантаженості. Оптимізована комунікація між мікросервісами, ефективних методів обробки даних, а також моніторинг та керування взаємодією дозволяють системі працювати з високою продуктивністю. Крім того, Spring Security гарантує захищеність даних та надійність взаємодії між компонентами системи.

2.6 Висновки до розділу 2

У розділі було розглянуто декілька нейромережевих підходів які могли бути використані для прогнозування відвідуваності вебсайтів, а також проаналізовано їхні переваги та недоліки і обрано для розробки Keras LSTM. Було визначено, що цей метод дозволяє ефективно аналізувати історичні дані відвідуваності та робити прогнози їх майбутнього розвитку. Використання архітектури LSTM дозволяє враховувати складні залежності в часових рядках та підходити до моделювання динамічних процесів.

Були розглянуті особливості отримання вхідної інформації від користувача для прогнозування відвідуваності вебсайтів, звернувши увагу на різні типи даних та їхню роль у точності прогнозів.

У підсумку було розглянуто проектну архітектуру системи, яка реалізує метод прогнозування. Вона складається з трьох мікросервісів: фронт-енд на React JS, бек-енд на Java Spring та Python з використанням Keras LSTM, а також БД PostgreSQL. Було визначено, що така архітектура дозволяє забезпечити ефективну комунікацію між компонентами системи і реалізувати комплексну обробку даних для прогнозування відвідуваності вебсайтів.

Розділ 3 Реалізація вебсервісу з методом прогнозування відвідуваності вебсайтів

3.1 Визначення шляхів дослідження та засобів створення програмного забезпечення

Визначення шляхів дослідження та засобів створення програмного забезпечення – найважливіший етап для якісної та ефективної розробки програми. Далі буде розглянуто обґрунтування вибору методу прогнозування трафіку на вебсайті, опис використаних технологій та бібліотек для реалізації вебсервісу, архітектуру програмного забезпечення та інтерфейс користувача. Виділено основні аспекти, що вплинули на вибір конкретного підходу до розробки програми та наведено детальний опис його реалізації.

Для прогнозування трафіку вебсайтів було обрано метод з використанням нейромережі LSTM. Цей метод був обраний через його здатність працювати з послідовними даними, що дозволяє передбачати трафік на основі історичних даних відвідуваності. LSTM нейромережі дозволяють ефективно моделювати складні залежності в даних і показують добрі результати в задачах прогнозування часових рядів.

Порівняно з іншими методами прогнозування, такими як ARIMA чи експоненціальне згладжування, нейромережевий підхід може краще адаптуватися до змін у структурі даних та показувати більш точні результати у випадку нестаціонарних даних. Це особливо важливо для прогнозування трафіку на вебсайті, де зміни в популярності можуть бути дуже різкими і непередбачуваними.

Для реалізації вебсервісу було обрано використати такі технології та бібліотеки:

– Python та бібліотека Keras [35] – використовувалися для побудови та тренування нейромережі LSTM.

– Мікросервісна архітектура – дозволила розділити функціональність сервісу на окремі компоненти, що спрощує розробку та підтримку.

– Postgresql [36] – використовувалася як база даних для зберігання даних про відвідуваність сайтів.

– Spring Security з JWT авторизацією [37] – технологія, яка забезпечила безпеку та авторизацію в мікросервісі на Java.

– Antd компоненти, Formik, Yup [38] – використовувалися для створення користувацького інтерфейсу на Frontend мікросервісі.

Архітектура програмного забезпечення складається з чотирьох основних компонентів:

База даних Postgresql – використовується для зберігання даних про відвідуваність сайтів.

Job для міграцій Liquibase [39] – використовується для оновлення структури бази даних.

Мікросервіс для зв'язку з БД та бізнес логіки на Java – відповідає за взаємодію з базою даних та виконання бізнес-логіки.

Мікросервіс з нейромережею LSTM на Keras для прогнозування – відповідає за прогнозування трафіку на основі навченої нейромережі.

Інтерфейс користувача буде розроблений з урахуванням зручності та інтуїтивності для користувачів. На головній сторінці користувач матиме можливість ввести дані про відвідуваність свого сайту по місяцях або по днях у спеціальне модальне вікно. Після цього система автоматично обробить ці дані та виведе прогноз трафіку на певний період часу. Користувач також зможе переглянути результат своїх прогнозів та відобразити графік по відвідуваності свого сайту.

Таким чином, обґрунтування вибору методу прогнозування та опис використаних засобів для створення програмного забезпечення надає повну картину для розробки вебсервісу для прогнозування трафіку вебсайтів.

3.2 Вибір засобів розробки вебсервісу

Вибір засобів розробки інформаційної системи є досить важливим етапом для швидкості та зручності розробки ПЗ. Тому далі буде описано вибір засобів розробки, які використовувалися при створенні вебсервісу для прогнозування трафіку на вебсайті. При розробці використовувалися такі засоби:

– IntelliJ IDEA [40] – була обрана для розробки мікросервісу на Java через її потужний інтегрований середовище розробки (IDE) та розширені можливості для роботи з Java та фреймворком Spring. Ця IDE забезпечує розширену підтримку автодоповнення коду, аналізу коду, рефакторингу та інших інструментів, що дозволяють підвищити продуктивність розробника та покращити якість коду.

Переваги IntelliJ IDEA порівняно з іншими IDE для Java, такими як Eclipse чи NetBeans, полягають у її швидкодії, багатофункціональності та великій кількості доступних плагінів, що робить її відмінним вибором для серйозного розвитку проектів на Java.

– Visual Studio Code (VS Code) [41] – був використаний для розробки Frontend мікросервісу на React TS та для розробки нейромережі на Python. VS Code є легким, швидким та має широкі можливості розширення за допомогою плагінів, що робить його популярним серед розробників. Для React TS, VS Code надає розширену підтримку TypeScript та React, включаючи автодоповнення коду, перевірку синтаксису та інші корисні функції.

У порівнянні з іншими IDE для роботи з React та Python, такими як Atom чи PyCharm, VS Code відзначається своєю швидкодією та розширеністю, а також безкоштовністю, що робить його привабливим вибором для проектів різної складності.

– DBeaver [42] – був обраний для роботи з базою даних Postgresql через його зручний та потужний інтерфейс. DBeaver надає широкий спектр функціональності для роботи з реляційними базами даних, включаючи

автодоповнення SQL, візуальну роботу з таблицями та зручний інтерфейс для взаємодії з базою даних.

Порівняно з іншими інструментами для роботи з базами даних, такими як pgAdmin чи MySQL Workbench, DBeaver відзначається своєю крос-платформенністю, що дозволяє використовувати його на різних операційних системах, а також багатофункціональністю та підтримкою різних типів баз даних.

– VisualVM [43] – використовувався для визначення memory leaks та оптимізації використання пам'яті в Java мікросервісі. Цей інструмент надає зручний інтерфейс для моніторингу та аналізу пам'яті та роботи JVM-подібних програм. VisualVM надає інформацію про використання пам'яті, потребує пам'яті та інші параметри, що дозволяє виявляти проблеми з витоком пам'яті та ефективно їх вирішувати.

У порівнянні з іншими інструментами для аналізу пам'яті та витоку пам'яті, VisualVM відрізняється своєю простотою використання та широким функціоналом, що дозволяє виявляти й вирішувати проблеми з пам'яттю швидко та ефективно.

Для розгортання та роботи з мікросервісною архітектурою були використані Docker Compose та Docker.

– Docker – це платформа для контейнеризації додатків, яка дозволяє упаковувати програмне забезпечення та його залежності в контейнери для швидкого та надійного розгортання на різних середовищах. Docker забезпечує ізольоване середовище для додатків, що дозволяє запускати їх на будь-якому сервері або в хмарному середовищі без зміни коду.

У порівнянні з іншими конкурентами для контейнеризації, такими як Kubernetes, OpenShift, Amazon ECS та Google Kubernetes Engine, Docker відрізняється своєю простотою використання та широким спектром функцій. Він надає інтуїтивний інтерфейс для створення, управління та моніторингу контейнерів, що дозволяє розробникам швидко розгортати та масштабувати додатки.

Переваги Docker включають:

1. Простота використання – Docker надає інтуїтивний інтерфейс та простий API для роботи з контейнерами, що дозволяє розробникам швидко розгортати додатки.

2. Ізоляція додатків – контейнери Docker забезпечують ізольоване середовище для додатків, що дозволяє запускати їх безпечно та незалежно від інших додатків.

3. Портативність – контейнери можна легко переносити між різними середовищами, що дозволяє розробникам швидко розгортати додатки у різних середовищах.

4. Масштабованість – Docker надає можливості для автоматизації розгортання та масштабування додатків, що дозволяє розробникам швидко реагувати на зміни в навантаженні.

У цілому, Docker залишається однією з найпопулярніших платформ для контейнеризації завдяки своїй простоті використання та широкому спектру функцій, що дозволяє розробникам ефективно працювати з контейнерами.

– Docker Compose [44] – дозволяє визначати та запускати багатоконтейнерні додатки зі складних конфігурацій, що робить розгортання та керування мікросервісами більш простим та зручним.

У порівнянні з іншими засобами віртуалізації та розгортання, Docker та Docker Compose відрізняються своєю простотою використання, швидкістю розгортання та масштабованістю, що робить їх відмінним вибором для розробки мікросервісів.

Отже, вибір засобів розробки був обґрунтований їхньою ефективністю, швидкодією та можливостями, що дозволило успішно реалізувати вебсервіс для прогнозування трафіку на вебсайті з використанням мікросервісної архітектури та нейромережевого підходу.

3.3 Структура та функціональне призначення складових вебсервісу

Розглянемо структуру та функціональне призначення програмних складових системи прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу. Система складається з різних компонентів, які спільно працюють для забезпечення коректної та ефективної роботи сервісу. Розглянемо докладніше кожну з програмних складових системи та їх роль у загальному функціонуванні системи.

Однією з ключових складових системи є база даних Postgresql, яка забезпечує зберігання та організацію даних про відвідуваність вебсайтів. База даних використовується для зберігання історичних даних про кількість відвідувань сайту по місяцях або по днях, які подальше використовуються для навчання нейромережі та прогнозування трафіку.

Для забезпечення стабільності та зручності роботи з базою даних, використовується інструмент для міграцій Liquibase. Він дозволяє автоматизувати процеси зміни схеми бази даних, що дозволяє швидко та безпечно внести зміни до структури даних.

Основна бізнес логіка системи винесена до окремого мікросервісу на Java, який відповідає за взаємодію з базою даних та обробку запитів користувачів. Цей мікросервіс забезпечує роботу з даними та виконання необхідних операцій для прогнозування трафіку на вебсайті.

Для реалізації нейромережевого підходу до прогнозування трафіку на вебсайті використовується окремий мікросервіс, який написаний на мові програмування Python та використовує бібліотеку Keras для побудови та навчання нейромережі типу LSTM (Long Short-Term Memory).

Модель LSTM є типом рекурентної нейромережі, яка добре підходить для аналізу та прогнозування послідовних даних, таких як часові ряди. У випадку системи прогнозування відвідуваності вебсайтів, ця модель використовується для аналізу історичних даних про відвідуваність сайту та прогнозування майбутньої активності користувачів.

Мікросервіс, який містить нейромережеву модель, призначений для прийому запитів від інших компонентів системи, щодо прогнозування трафіку. Після отримання запиту, він застосовує навчення моделі LSTM до вхідних даних (наприклад, історичні дані про відвідуваність сайту) та генерує прогноз для заданого періоду часу.

Отриманий прогноз трафіку передається назад до іншого мікросервісу, який відповідає за відображення результатів користувачеві через інтерфейс. Таким чином, нейромережевий мікросервіс грає ключову роль у системі, забезпечуючи точні та достовірні прогнози трафіку для користувачів.

Для забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, використовується Frontend мікросервіс на React TS. Цей компонент відповідає за відображення результатів прогнозування трафіку та взаємодію з користувачем.

Загальний вигляд архітектури системи з усіма мікросервісами зображений на рисунку 3.1.

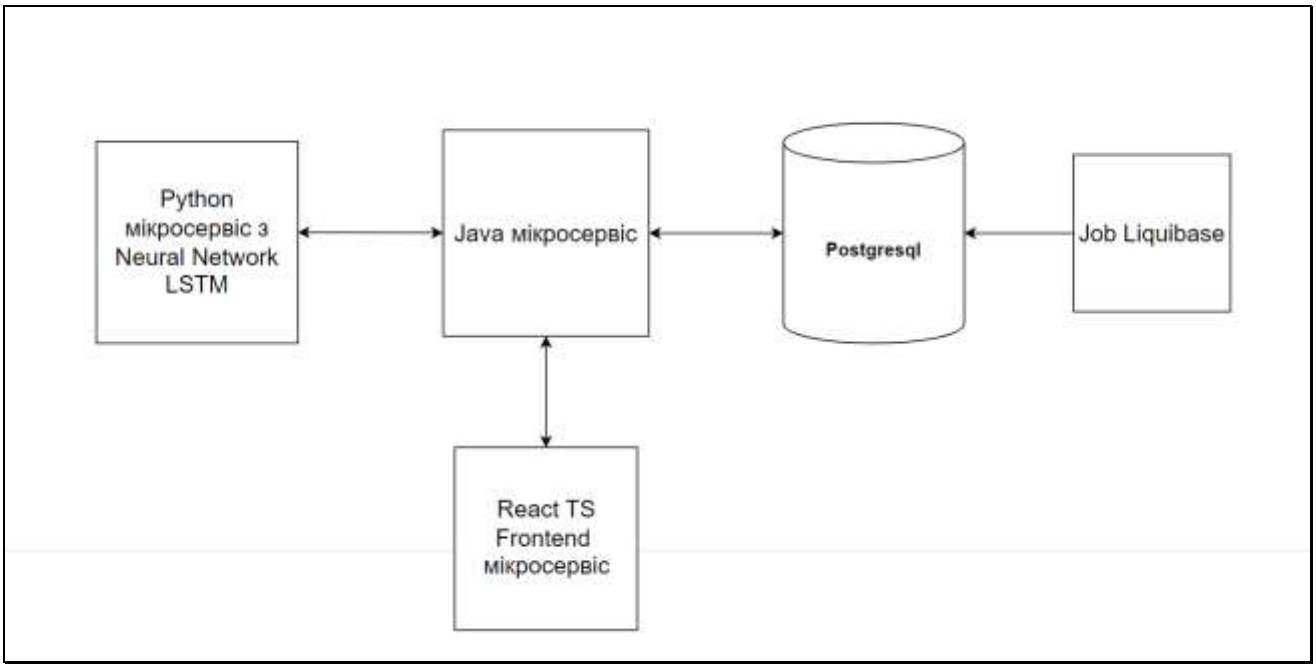


Рисунок 3.1 – Загальний вигляд архітектури системи

У даній системі спілкування між мікросервісами відбувається за допомогою передачі даних у форматі JSON [45] за допомогою об'єктів DTO

(Data Transfer Object) [46]. Кожен мікросервіс може відправляти та отримувати дані у форматі JSON, що дозволяє їм ефективно обмінюватися інформацією.

Наприклад, коли Frontend мікросервіс потребує отримати прогноз трафіку від Python мікросервісу, він може відправити запит на Java мікросервіс, мікросервіс обробить запит та надішле запит до Python мікросервісу, який обробить запит та надішле відповідь у форматі JSON. Дані у форматі JSON будуть містити прогнозовану кількість відвідувачів на вебсайті на певний період часу. Frontend мікросервіс може обробити ці дані та відобразити їх користувачеві.

Використання об'єктів DTO дозволяє стандартизувати структуру даних, які передаються між мікросервісами, що полегшує розробку та обслуговування системи. Крім того, використання JSON дозволяє зменшити розмір переданих даних та забезпечити їх безпеку під час передачі через мережу.

Для документування API та спілкування між мікросервісами використовується Swagger [47]. Swagger дозволяє описати структуру та функціонал API за допомогою спеціального файлу у форматі YAML або JSON, який потім може бути використаний для генерації документації, автоматичної генерації клієнтських бібліотек або для тестування API.

Використання Swagger для документування API у системі допоможе підтримувати чітку та зрозумілу документацію, спростить тестування та інтеграцію з вашим API, а також забезпечить більшу стабільність та надійність системи.

У системі є багато компонентів які є життєво важливими. Одним з таких компонентів є бібліотека Spring Security. Клас WebSecurityConfig (рисунок 3.2) – є основною реалізацією конфігурацій бібліотеки для авторизації.

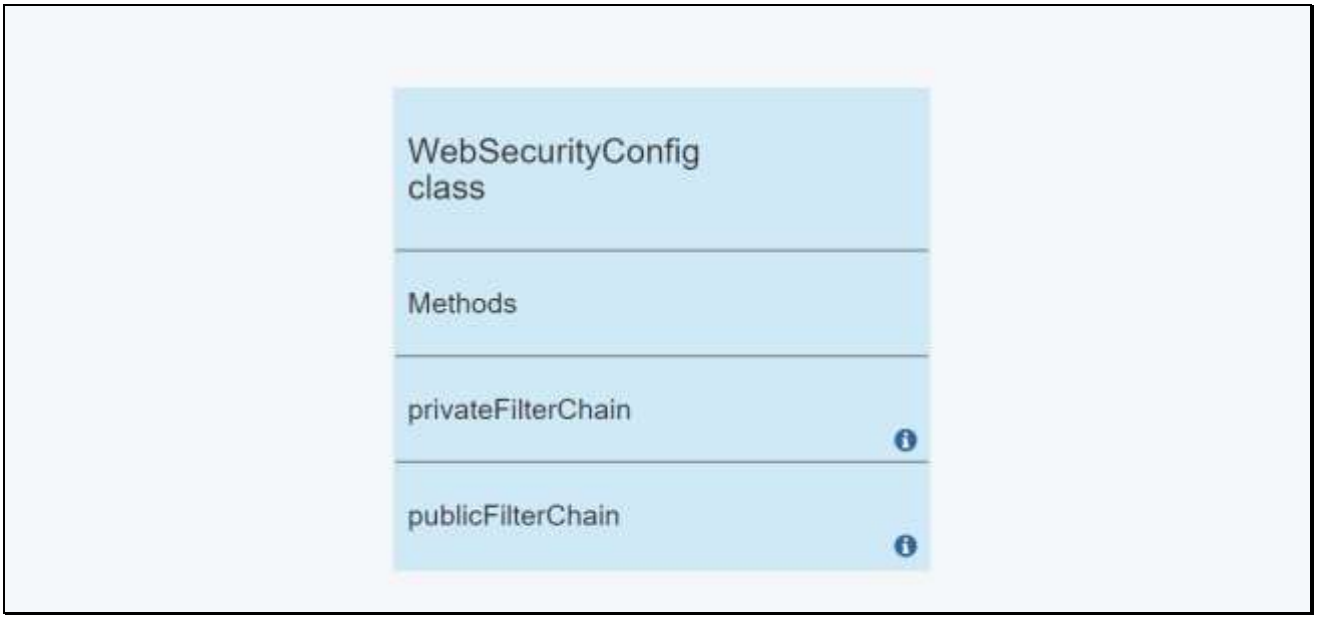


Рисунок 3.2 – Клас-реалізація конфігурацій авторизації за допомогою Spring Security

Цей клас містить конфігураційні налаштування для двох основних фільтрів безпеки.

Перший фільтр publicFilterChain призначений для обробки запитів, які не потребують авторизації, тому всі вони дозволяються.

Другий фільтр privateFilterChain відповідає за обробку запитів, які потребують авторизації. Для цього фільтр налаштовується для перехоплення таких запитів і виконання необхідної перевірки авторизації перед їх обробкою.

Основні процеси, які відбуваються в класі WebSecurityConfig, включають налаштування правил авторизації для різних типів запитів, використання фільтрів безпеки для перехоплення та обробки запитів, а також налаштування роботи з токенами доступу та перевірка їх валідності.

Загалом, клас WebSecurityConfig є важливим компонентом системи, який забезпечує безпеку та авторизацію вебзастосунку і грає ключову роль у забезпеченні захищеності та надійності системи.

Таким чином, система складається з різних програмних складових, які спільно працюють для забезпечення коректної та ефективної роботи сервісу прогнозування трафіку на вебсайті. Кожна з компонентів виконує свою функцію та спільно з іншими допомагає досягти поставлених цілей системи.

3.4 Особливості реалізації програмних складових вебсервісу

У даному розділі дипломної роботи розглядаються особливості реалізації програмних складових системи прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу. Програмна система складається з різних компонентів, які спільно працюють для забезпечення коректної та ефективної роботи сервісу. У цьому розділі будуть розглянуті найважливіші аспекти програмної реалізації, зокрема алгоритми компонентів, особливості їхньої роботи та спілкування між ними. Буде детально описано структуру та функціональне призначення кожної програмної складової системи, а також процеси, які відбуваються в їх взаємодії для забезпечення високої якості прогнозування трафіку на вебсайті. Також будуть надані результати виконання алгоритмів та скріни інтерфейсу для кращого розуміння реалізації системи.

У реалізації нейромережевого підходу для прогнозування відвідуваності вебсайтів використовується мікросервіс на Python з використанням бібліотеки Keras для побудови та навчання нейромережі LSTM. Цей компонент системи відповідає за прогнозування трафіку на основі навченої моделі та передачу результатів до користувача через інтерфейс.

Для початку, дані про дати та відвідуваність конвертуються до зручного формату для обробки. Потім дані підготовлюються для навчання моделі LSTM. Для цього використовується функція prepare_data, яка розділяє вхідні дані на частини, що використовуються для входу та виходу моделі.

Наступним кроком є побудова та навчання моделі LSTM. У цьому випадку модель має один шар LSTM та один підключений шар Dense. Вона компілюється з використанням оптимізатора Adam і функції втрат MSE (Mean Squared Error) [48].

Після навчання моделі виконується прогнозування відвідуваності на наступні дні. Для цього використовується функція predict_next_days, яка

використовує навчену модель для прогнозування значень трафіку на кожен з наступних днів.

Отримані прогнози відображаються у форматі дати та відвідуваності. Перед поверненням результатів, відвідуваність виправляється, щоб уникнути від'ємних значень, що може бути неможливим у реальних умовах.

Цей підхід дозволяє системі точно прогнозувати трафік на вебсайті на наступні дні, що дозволяє користувачам ефективніше планувати свою роботу та ресурси.

На сторінці реєстрації сайту (рисунок 3.3) зображено форму для створення нового облікового запису.

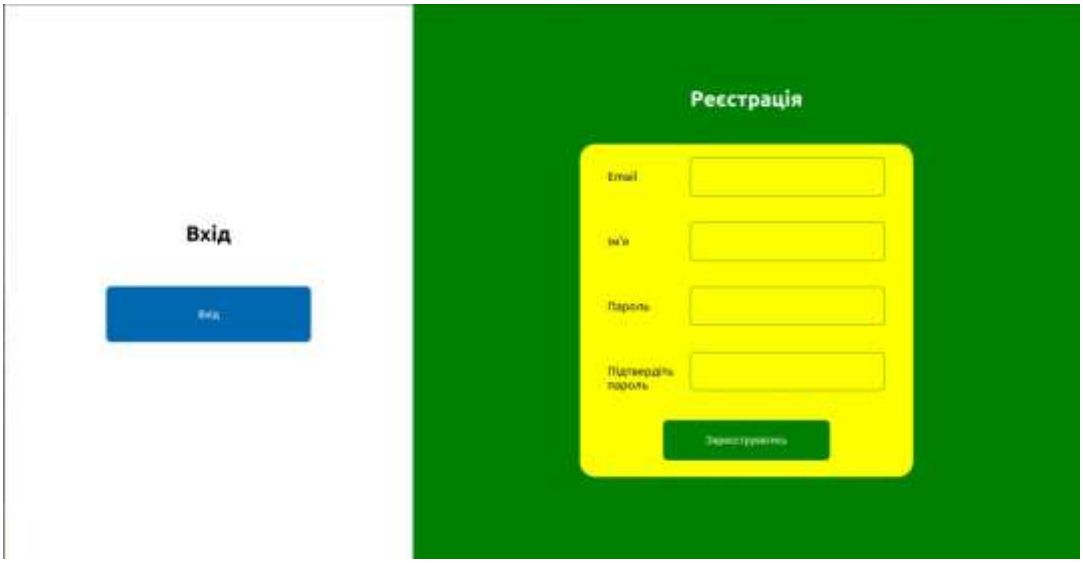


Рисунок 3.3 – Сторінка реєстрації

Форма містить поля для введення електронної пошти (Email), імені користувача (Username), паролю (Password) та підтвердження паролю (Confirm Password). Крім того, на формі присутня кнопка "Зареєструватись", при натисканні на яку дані, введені в поля, відправляються на сервер для обробки.

Щоб забезпечити відповідну перевірку та валідацію введених даних, використовуються бібліотеки Formik та Yup. Formik відповідає за керування станом форми та обробку подій, таких як введення даних та натискання кнопки,

а Yup визначає схему валідації для кожного поля форми, забезпечуючи коректну обробку даних перед їхнім відправленням.

Ця форма є важливим елементом сайту, оскільки вона дозволяє користувачам створювати особисті облікові записи і отримувати доступ до ресурсів сайту, що вимагають авторизації. Вона також відображається у дизайні сайту як проста, зрозуміла та зручна для використання.

На сторінці входу (рисунок 3.4) на сайт використовується форма для введення даних для авторизації.

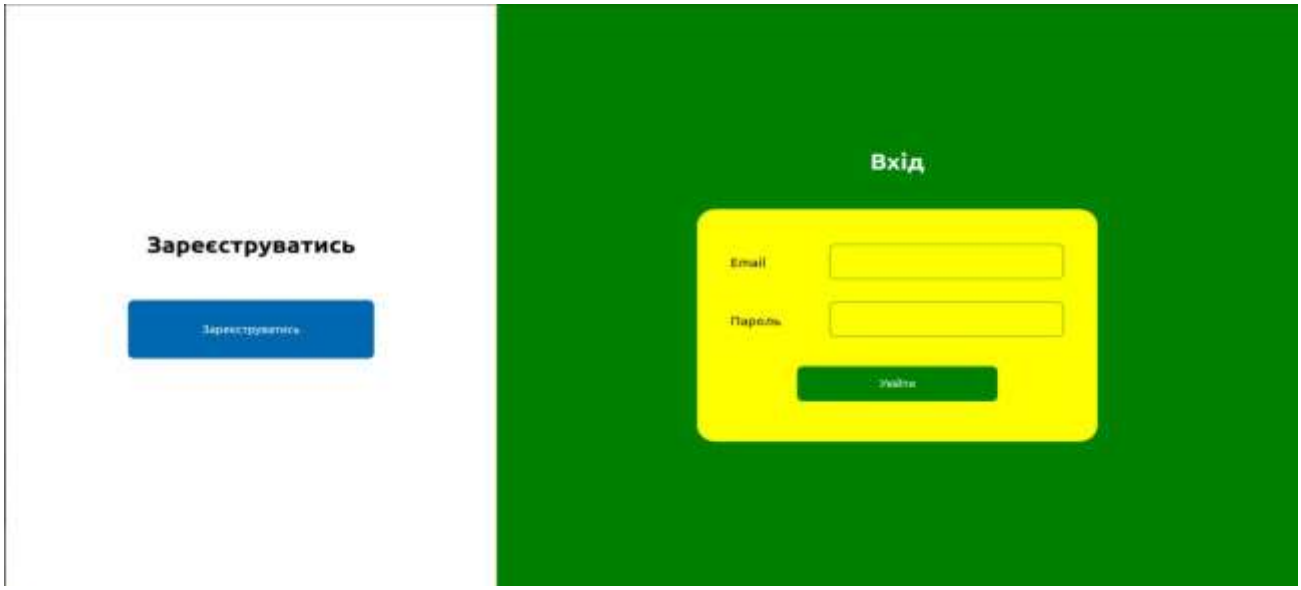


Рисунок 3.4 – Сторінка входу

Форма містить поля для введення електронної пошти (Email) та паролю (Password), а також кнопки для переходу на сторінку реєстрації та для входу на сайт. Використання бібліотек Formik та Yup дозволяє забезпечити коректну валідацію введених даних перед їхнім відправленням на сервер.

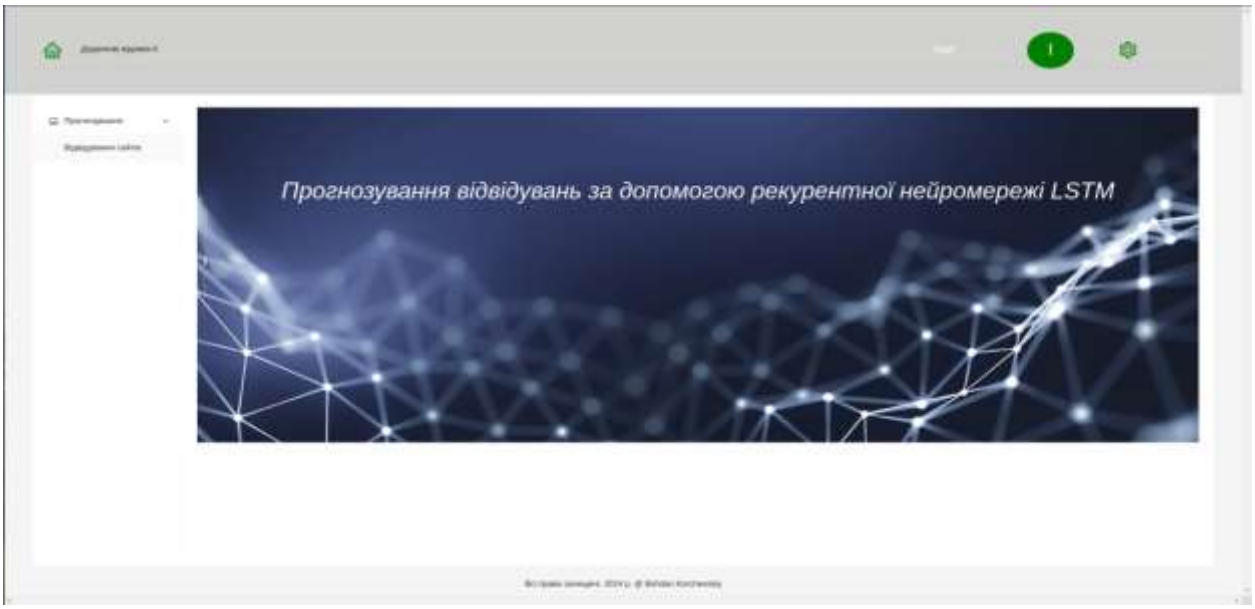


Рисунок 3.5 – Головна сторінка вебсервісу

На рисунку 3.5 зображено головну сторінку вебсервісу для прогнозування відвідуваності вебсайтів. На ній можна помітити слайдер, вкладку “Відвідування сайтів” при натисканні на яку користувач може прогнозувати відвідуванність сайтів, також є кнопка яка переходить до налаштування сайту, присутній аватар та ім’я користувача.

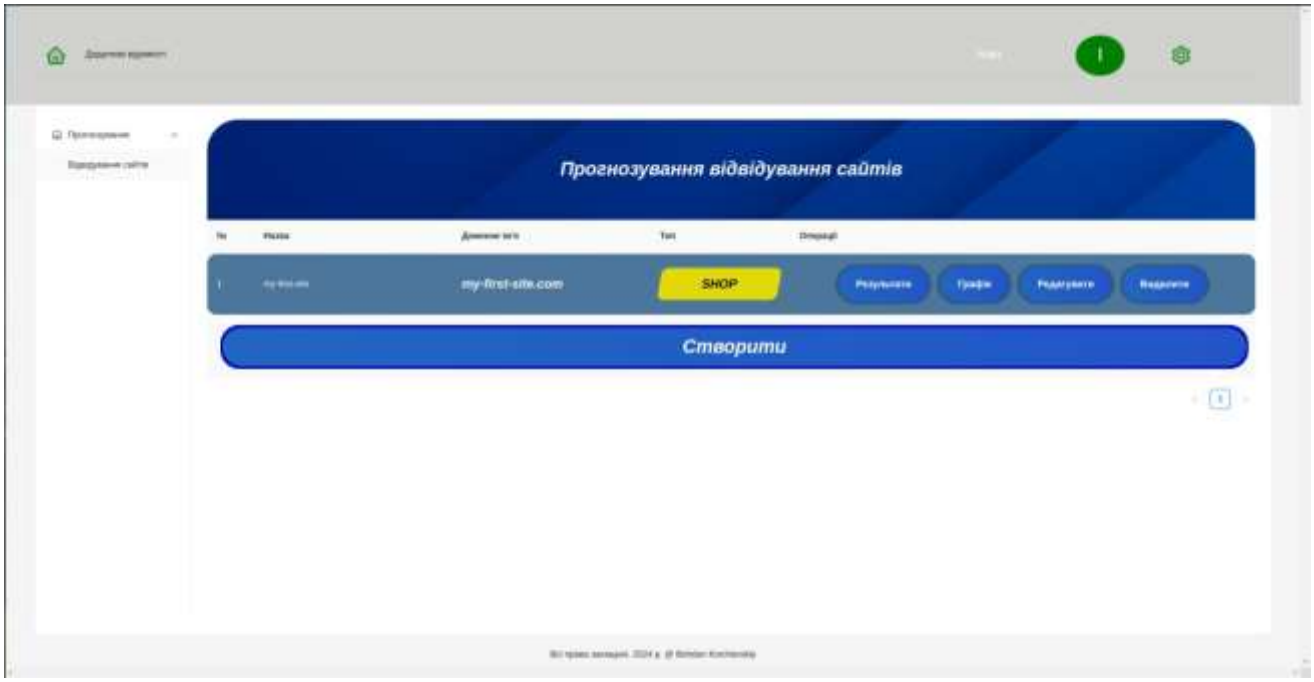


Рисунок 3.6 – Сторінка прогнозування відвідуваності вебсайтів

На сторінці прогнозувань вебсайтів користувача використовується Ant Design (Antd) таблиця для відображення набору сайтів. Antd таблиця є потужним інструментом для візуалізації даних, який дозволяє зручно відображати інформацію у вигляді табличного представлення.

Таблиця має можливість сортування, фільтрації та пошуку даних, що дозволяє користувачеві легко знаходити необхідну інформацію серед великого обсягу даних. Кожен рядок таблиці представляє окремий сайт, а кожен стовпчик містить різні параметри сайту, такі як назва, доменне ім’я, тип та операції.

Завдяки Antd таблиці користувач може зручно переглядати та аналізувати інформацію про свої вебсайти, що дозволяє йому приймати обґрунтовані рішення щодо їхньої діяльності та розвитку. Кожен користувач може натиснути кнопку створити і після цього з’явиться модальне вікно (рисунок 3.7).

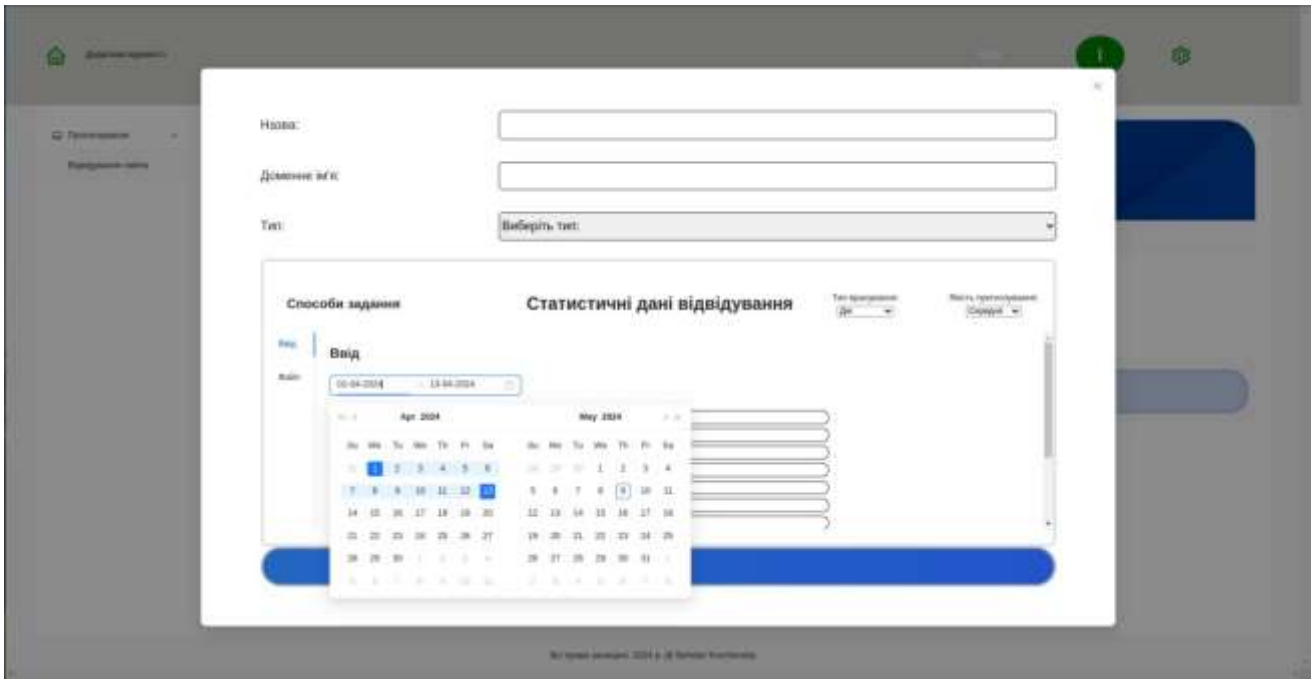


Рисунок 3.7 – Додавання сайту з введенням даних для прогнозування вручну.

Модальне вікно містить поля: назва сайту, доменне ім’я, тип. Далі користувач має обрати спосіб прогнозування – по днях або ж по місяцях. Також можна обрати якість прогнозування, що впливає на кількість епох яку буде проходити нейромережа для навчання.

Ant Design надає можливість легко створювати та керувати модальними вікнами, що значно полегшує взаємодію користувачів з додатком. Інтерфейс модальних вікон дозволяє зручно відображати додаткову інформацію або отримувати від користувачів додаткові дані, не перегружаючи основний інтерфейс. Вони також дозволяють зосередитись на конкретному завданні або операції, забезпечуючи великий комфорт у використанні.

Після вибору періоду за який користувач хоче вказати статистичні дані відвідуваності відображаються поля для вказання цих даних (рисунок 3.8).

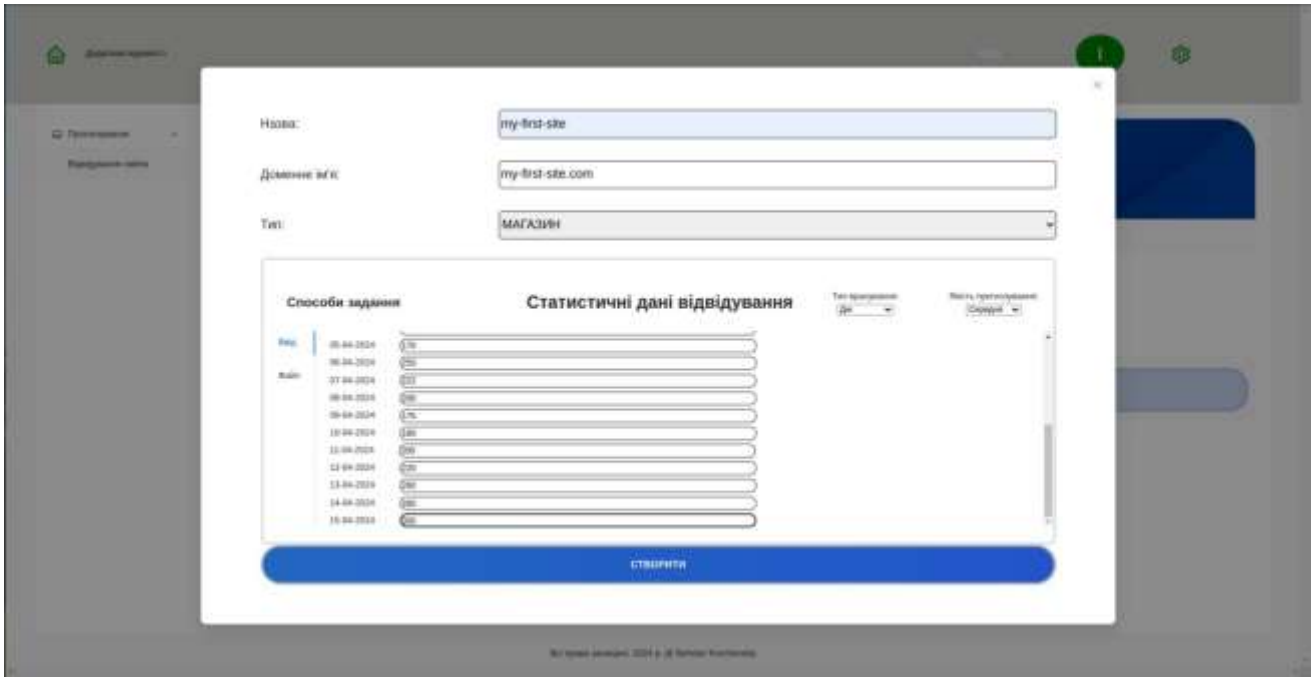


Рисунок 3.8 – Додавання сайту з введенням даних для прогнозування вручну з відображенням полів для кожної дати.

Також користувач має змогу вводити данні для прогнозування відвідуваності вебсайту вручну (рисунок 3.7) або ж імпортувавши файл з даними (рисунок 3.9).

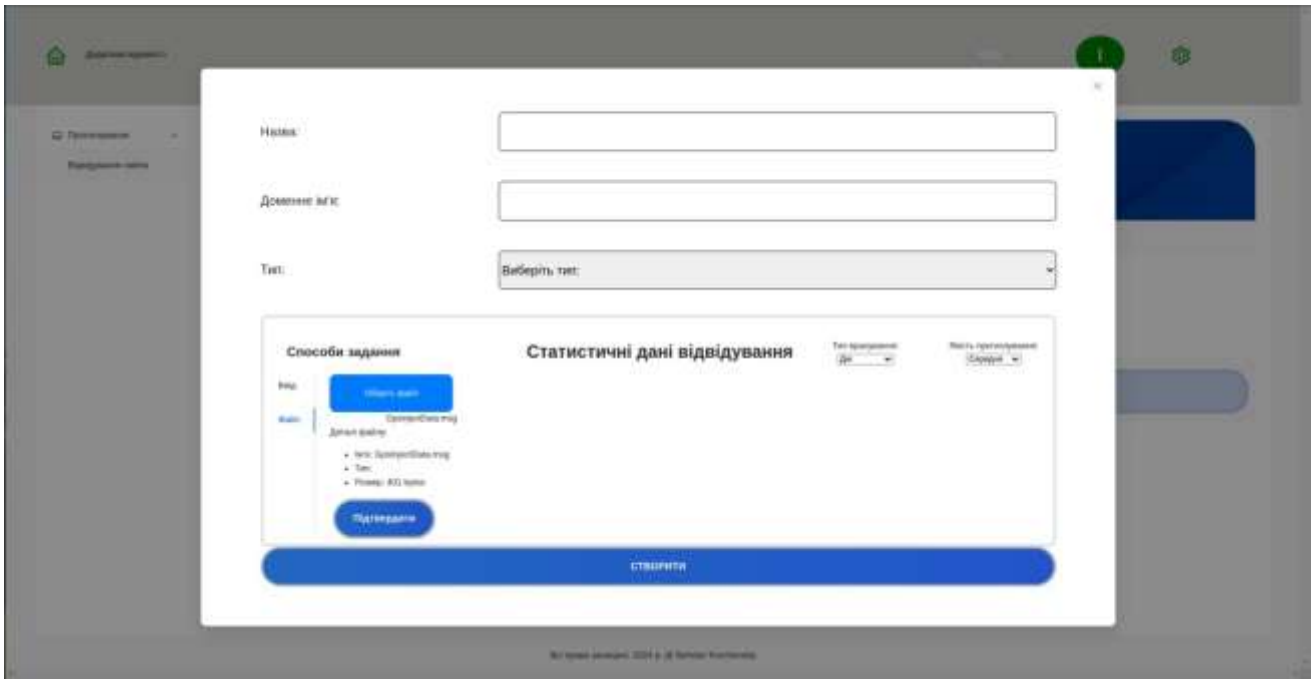


Рисунок 3.9 – Додавання сайту з введенням даних для прогнозування за допомогою імпорту файлу.

Після створення користувач повинен зачекати декілька секунд оскільки нейромережа буде навчатись та робити прогнози. Коли операція завершується, користувач може натиснути кнопку “Графік” та отримати графік як на рисунку 3.10.

Рисунок 3.10 – Графік з прогнозованою відвідуваністю сайту.

На рисунку 3.10 відображено графік з прогнозованою відвідуваністю сайту за введеними користувачем даними. Жовта лінія на графіку – позначає дані введені користувачем, а синя – прогнозовані дані.

На графіку є декілька кнопок, які допомагають масштабувати позначення, рухатись, завантажувати картинку в різних форматах та повертатись до початкового стану.

Також після додавання користувачем власного сайту для прогнозування, на сайті присутня таблиця з результатами на відповідній сторінці (рисунок 3.11).

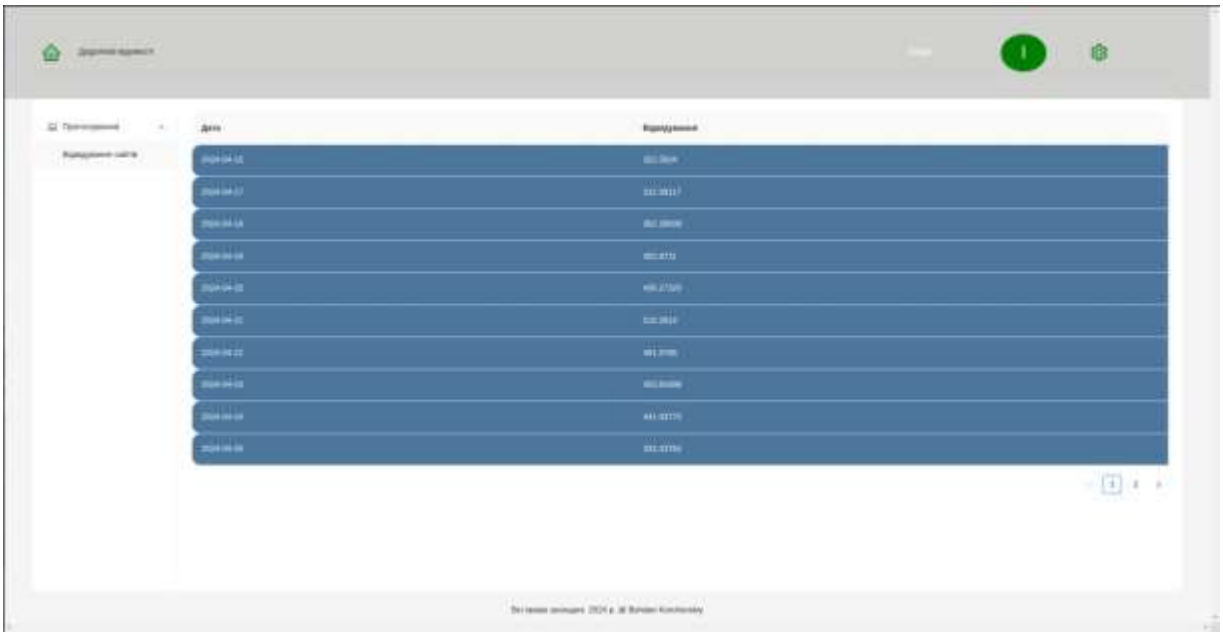


Рисунок 3.11 – Сторінка з результатами прогнозування відвідуваності сайту.

Інші можливості сайту, досить стандартні – отримання власної інформації в кабінеті користувача (рисунок 3.12), можливість зміни кольору для аватара.

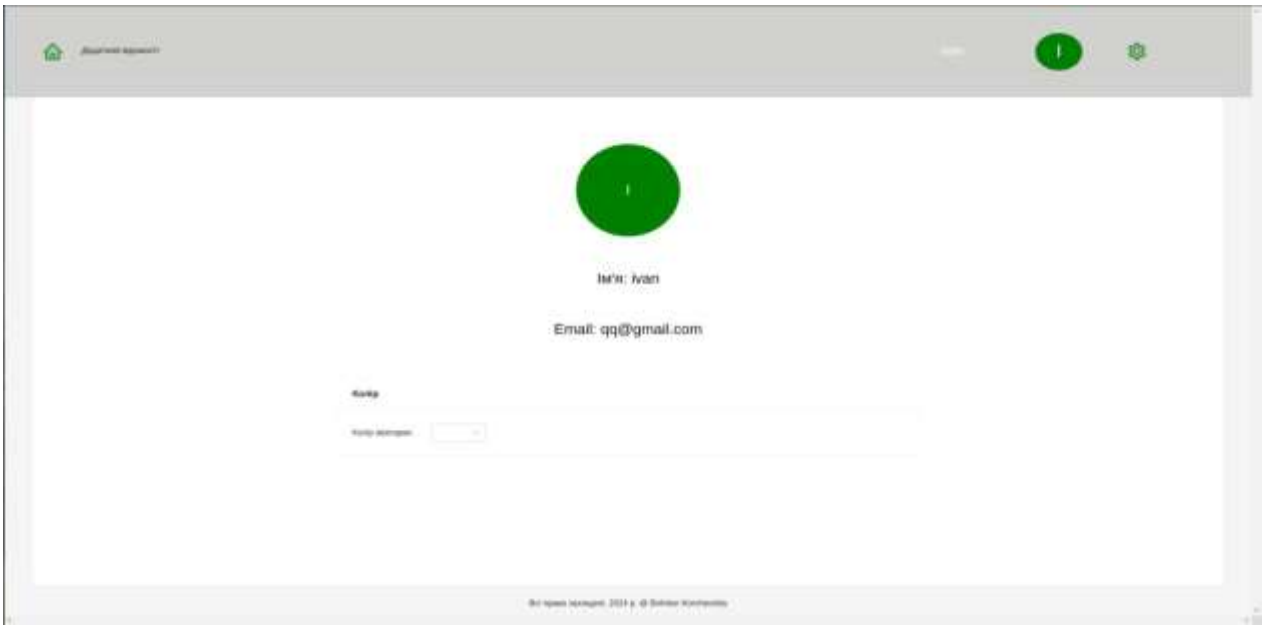


Рисунок 3.12 – Особистий кабінет користувача.

Також в правому верхньому куті сайту є іконка налаштувань, натиснувши на неї користувач має можливість перейти на сторінку налаштувань та змінити тему сайту, видалити аккаунт або змінити ім’я.

Рисунок 3.13 – Сторінка налаштувань.

.

Отже, у даному розділі було розглянуто важливі аспекти програмної реалізації системи прогнозування відвідуваності вебсайтів з використанням

нейромережевого підходу. Були розглянуті алгоритми компонентів, їхні особливості та спілкування між ними, структура та функціональне призначення кожної складової системи, а також процеси взаємодії для забезпечення якісного прогнозування трафіку на сайті.

Окрім того, результати виконання алгоритмів та скріни інтерфейсу дозволили краще розуміти реалізацію системи. Використання мікросервісу на Python з бібліотекою Keras для навчання нейромережі LSTM показало ефективність прогнозування трафіку та передачу результатів користувачам через зручний інтерфейс.

3.5 Тестування інформаційної системи та вимоги до розгортання

Під час тестування функціоналу сайту було проведено ряд тестів для перевірки його працездатності та відповідності вимогам. Використовувався підхід створення тест кейсів для переконання в надійності та ефективності роботи сайту.

Перший тест-кейс (таблиця 3.1) націлений на перевірку правильності реєстрації користувача

Таблиця 3.1 – Тест-кейс TS0001

Тест-кейс ID: TS0001	**Пріоритет:** 1	**Створено:** 15.04.2024, Корчевський Б.В.
Назва: Перевірка реєстрації користувача **Вхідні дані:** Дані про користувача		
Кроки	**Очікуваний результат**	
1. Перейти на ст. входу 2. Натиснути кнопку “Реєстрація” 3. Вказати коректні дані для реєстрації 4. Натиснути кнопку “Зареєструватись”	Коректне заповнення даних у БД Postgresql та перехід на сторінку авторизації	

Наступний тест-кейс робить перевірку на коректне додавання сайту користувача в таблицю.

Таблиця 3.2 – Тест-кейс TS0002

Тест-кейс ID: TS0002	**Пріоритет:** 1	**Створено:** 15.04.2024, Корчевський Б.В.
Назва: Додавання сайту користувача в таблицю **Вхідні дані:** Дані про сайт та його відвідуваність		
Кроки	**Очікуваний результат**	
1. Відкрити будь-яку сторінку сайту 2. Отримати перенаправлення на сторінку входу 3. Вказати дані для входу 4. Натиснути кнопку “Вхід” 5. Натиснути на посилання “Відвідування сайтів” 6. Натиснути кнопку “Створити” 7. Заповнити форму 8. Натиснути кнопку “Створити”	Коректне заповнення даних у БД Postgresql та оновлення даних у UI таблиці	

Вебсервіс для передбачення відвідуваності вебсайтів складається з декількох Docker-контейнерів. Для збирання та запуску цих контейнерів потрібно встановити Docker та Docker-compose на ваш комп'ютер. Потім перейти до папки з файлом docker-compose вашого проєкту, відкрити термінал і виконайти команду docker-compose up.

Після запуску цієї команди ваш вебсайт буде запущений локально, і ви зможете перевірити статус контейнерів за допомогою команди docker ps. Ця команда показує зовнішні та внутрішні порти контейнерів.

Для доступу до UI вебсервісу, потрібно знайти порт Frontend мікросервісу та відкрити URL у браузері, використовуючи localhost.

Після проведення тестування можна зробити висновок, що сайт працює стабільно, відповідає вимогам та може ефективно виконувати свої функції. Всі виявлені під час тестування проблеми були виправлені і сайт готовий до впровадження.

3.6 Результати досліджень

У цьому розділі представлені результати досліджень ефективності використання нейромережі LSTM для прогнозування трафіку вебсайтів. Представлені дані включають результати прогнозування на основі історичних даних відвідуваності, порівняння різних підходів та оцінку точності моделі. Висновки дослідження допоможуть зрозуміти переваги та обмеження нейромережевого підходу у прогнозуванні трафіку на вебсайті.

Для оцінки результатів дослідження ефективності методу прогнозування трафіку на вебсайті з використанням нейромережі LSTM маємо результуючий графік (рисунок 3.14) на якому чудово видно ефективність застосування.

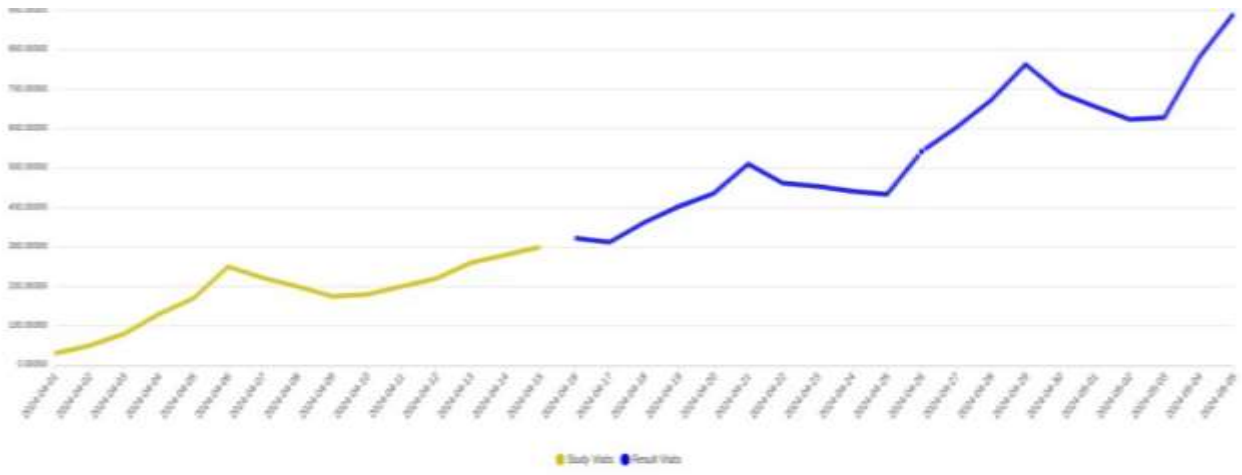


Рисунок 3.14 – Результуючий графік прогнозування відвідуваності.

На графіку жовтим кольором позначено дані які прийшли від користувача на

певний період. Синім кольором позначено передбачення, які були визначені нейромережею LSTM. Наглядно видно, що графік правдиво передбачає зростання відвідуваності на сайті беручи до уваги вказані дані.

Також можна вказати для LSTM нейромережі лінійний набір даних як на рисунку 3.15.

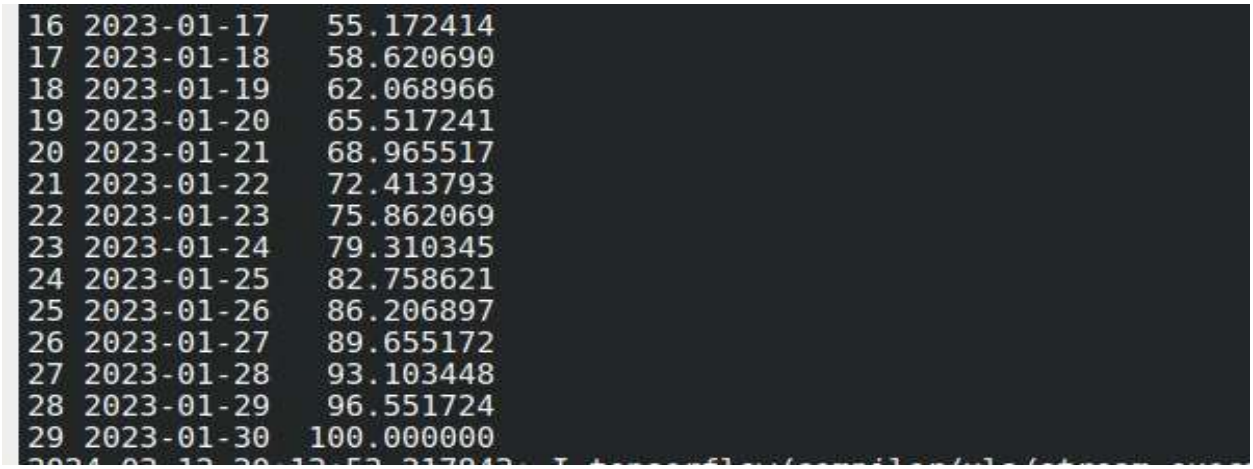


Рисунок 3.15 – Лінійний набір даних.

Отримавши результати (рисунок 3.16), можна зазначити, що нейромережа якісно навчається та робить правдиві передбачення.

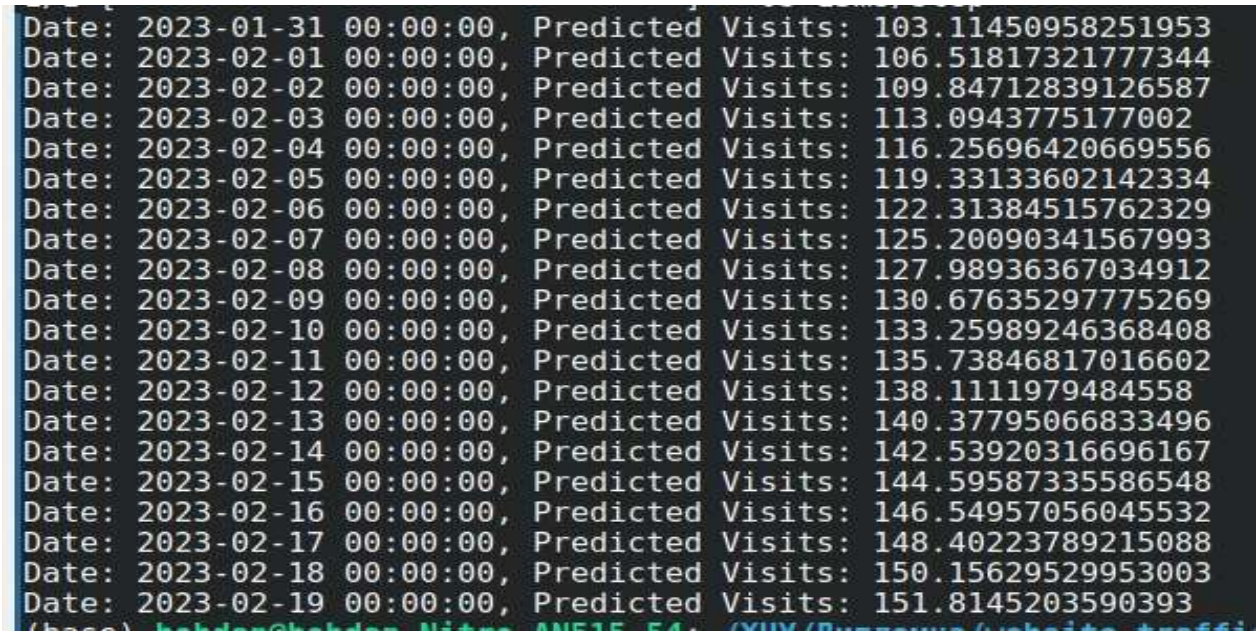


Рисунок 3.16 – Результати прогнозування для лінійного набору даних.

Отже, отримавши результати досліджень, можна зробити висновок, що нейромережа LSTM чудово виконує поставлену мету визначення майбутньої кількості відвідувачів на сайті.

3.7 Висновки до розділу 3

Після завершення експериментального дослідження методу та програмної реалізації інформаційної системи можна зробити висновки, що експериментальні результати підтвердили ефективність обраного методу прогнозування трафіку на вебсайті з використанням нейромережі LSTM.

Метод дозволив досягти високої точності прогнозів, особливо в умовах змінної популярності сайту.

Результати експерименту показали, що нейромережа LSTM демонструє значно кращі результати прогнозування у порівнянні з іншими методами. Модель на основі LSTM показала стабільність у прогнозуванні трафіку навіть при змінах у структурі даних та різких коливаннях відвідуваності сайту.

Отримані результати свідчать про практичну застосовність розробленої системи для прогнозування трафіку на реальному вебсайті з метою покращення управління його ресурсами.

Для подальшого вдосконалення системи рекомендується провести додаткові дослідження з метою підвищення точності прогнозування та оптимізації ресурсних витрат.

Отже, експериментальне дослідження підтвердило ефективність обраного методу та продемонструвало його потенціал для практичного застосування у сфері прогнозування трафіку на вебсайті.

Загальні висновки

Метою кваліфікаційної роботи бакалавра є покращення ефективності прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу, реалізованого у вигляді вебсервісу.

Для досягнення поставленої мети було використано метод з використанням нейромережі LSTM. Цей метод був обраний через його здатність працювати з послідовними даними, що дозволяє передбачати трафік на основі історичних даних відвідуваності. LSTM нейромережі дозволяють ефективно моделювати складні залежності в даних і показують добрі результати в задачах прогнозування часових рядів.

Також було спроєктовано структуру вебресурсу та бази даних для цього вебресурсу.

Для реалізації нейромережевого підходу було розроблено вебсервіс, який використовується для прогнозування трафіку на вебсайті. Цей сервіс може бути використаний власниками вебсайтів для покращення управління ресурсами та планування маркетингових кампаній.

Також було проведено тестування розробленого вебсервісу і було отримано позитивні результати.

Досліджено ефективність розробленого методу прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу, що підтвердило його придатність для використання у реальних умовах.

Усі завдання було успішно виконано, що свідчить про досягнення поставленої мети кваліфікаційної роботи бакалавра.

Щодо перспектив впровадження розробленого методу та програмної системи, можна відзначити їх потенційну значимість для вебсервісів, що спрямовані на прогнозування трафіку. Розроблений метод на основі нейромережі LSTM дозволяє отримувати точні прогнози відвідуваності, що є критичним для вебсайтів, особливо у сферах електронної комерції, медіа та реклами.

Щодо можливостей та шляхів вдосконалення програмного продукту, можна вказати на такі напрями:

Подальше дослідження і покращення нейромережевого підходу. Можливе вдосконалення архітектури нейромережі, використання більш складних моделей для ще точніших прогнозів.

– Розширення функціональності вебсервісу. Додаткові можливості для аналізу трафіку, включаючи прогнозування популярності окремих матеріалів або категорій на сайті.

– Підтримка різних типів вхідних даних. Розширення можливостей моделі для обробки та аналізу інших видів даних, наприклад, відвідуваність мобільних пристроїв, різні джерела трафіку тощо.

– Інтеграція з іншими системами аналітики. Можливість експорту даних або інтеграція з іншими платформами для більш широкого аналізу трафіку та покращення стратегій розвитку вебсайту.

Ці напрями розвитку можуть сприяти подальшому покращенню ефективності та універсальності розробленого методу та програмної системи.

Загалом, розроблений метод прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу може стати потужним інструментом для вебаналітики, допомагаючи вебресурсам удосконалювати свої стратегії контенту та маркетингу на основі точних та достовірних прогнозів.

Перелік посилань

1. Wikipedia. Статистика. URL: https://uk.wikipedia.org/wiki/Статистика
2. Wikipedia. Консолідована інформація. URL: https://uk.wikipedia.org/wiki/Консолідована_інформація
3. Wikipedia. Аналіз часових рядів. URL: https://uk.wikipedia.org/wiki/Аналіз_часових_рядів
4. Wikipedia. Мащинне навчання. URL: https://uk.wikipedia.org/wiki/Машинне_навчання
5. Wikipedia. Random Forest. URL: https://uk.wikipedia.org/wiki/Random_forest
6. Kaggle. Gradient Boosting: теорія та пакети. URL: https://www.kaggle.com/code/nadiia789/gradient-boosting
7. Wikipedia. Штучна нейронна мережа. URL: https://uk.wikipedia.org/wiki/Штучна_нейронна_мережа
8. Wikipedia. Нейрон. URL: https://uk.wikipedia.org/wiki/Нейрон
9. Wikipedia. Метод зворотного поширення помилки. URL: https://uk.wikipedia.org/wiki/Метод_зворотного_поширення_помилки
10. Wikipedia. Перцептрон. URL: https://uk.wikipedia.org/wiki/Перцептрон
11. Wikipedia. Рекурентна нейронна мережа. URL: https://uk.wikipedia.org/wiki/Рекурентна_нейронна_мережа
12. Evergreens. Згорткова нейронна мережа – просте пояснення CNN та її застосування. URL: https://evergreens.com.ua/ua/articles/cnn.html
13. Wikipedia. Глибоке навчання. URL: https://uk.wikipedia.org/wiki/Глибоке_навчання
14. Wikipedia. Модулярні нейронні мережі. URL: https://uk.wikipedia.org/wiki/Модулярні_нейронні_мережі
15. Unite. Що таке автокодер?. URL: https://www.unite.ai/uk/what-is-an-autoencoder/

16. Luis de-la-Fuente-Valentín, Juan M. Corchado. mdpi.com. Web Traffic Time Series Forecasting Using LSTM Neural Networks with Distributed Asynchronous Training. URL: https://www.mdpi.com/2227-7390/9/4/421
17. Shuaiqiang Wang, Jianchao Zeng, and Xin Zhao. ar5iv. Sales Demand Forecast in E-commerce using a Long Short-Term Memory Neural Network Methodology URL: https://ar5iv.labs.arxiv.org/html/1901.04028
18. Wikipedia. Електронна комерція. URL: https://uk.wikipedia.org/wiki/Електронна_комерція
19. Google Analytics. Google Analytics. URL: https://analytics.google.com/
20. Amazon Forecast. Amazon Forecast. URL: https://aws.amazon.com/forecast/
21. Wikipedia. Маркетингова служба. URL: https://uk.wikipedia.org/wiki/Маркетингова_служба
22. Wikipedia. Бюджетування. URL: https://uk.wikipedia.org/wiki/Бюджетування
23. Google Ads. Google Ads. URL: https://ads.google.com/intl/uk_ua/home/
24. Amazon S3. Amazon S3. URL: https://aws.amazon.com/s3/
25. Similarweb. Similarweb. URL: https://pro.similarweb.com/
26. Serpstat. Serpstat. URL: https://serpstat.com/
27. RNN. RNN. URL: https://botpenguin.com/glossary/recurrent-neural-network
28. CNN. CNN. URL: https://www.researchgate.net/figure/Architecture-of-a-Convolutional-Neural-Network-CNN-The-traditional-CNN-structure-is_fig1_330106889
29. Variational Autoencoders. Variational Autoencoders are Beautiful. URL: https://www.compthree.com/blog/autoencoder/
30. DNN. DNN. URL: https://www.researchgate.net/figure/Deep-Neural-Network-DNN-example_fig2_341037496
31. Researchgate. LSTM Model. URL: https://www.researchgate.net/profile/Edi-

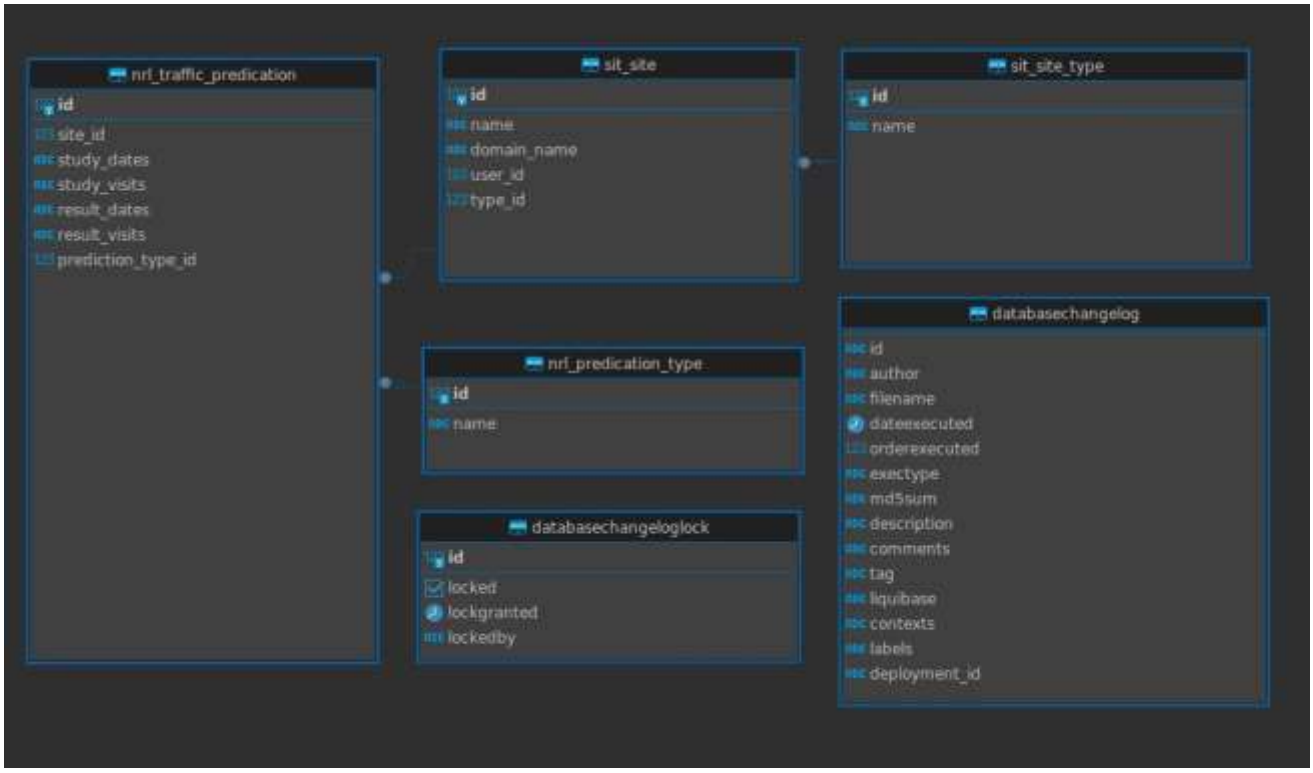
Ismanto/publication/375521544/figure/fig3/AS:11431281204087882@1699584353491/The-architecture-of-the-designed-LSTM-model.ppm

32. Keras. Keras. URL: https://se.ewi.tudelft.nl/desosa2019/chapters/keras/images/keras/context_view.PNG
33. Sciencedirect. Deep learning. URL: https://www.sciencedirect.com/science/article/abs/pii/S1874490720302615
34. LSTM block. LSTM block. URL: https://d3i71xaburhd42.cloudfront.net/a7976c2bacfbb194ddbe7fd10c2e50a545cf4081/2-Figure1-1.png
35. Keras. Keras. URL: https://uk.wikipedia.org/wiki/Keras
36. PostgreSQL. PostgreSQL. URL: https://uk.wikipedia.org/wiki/PostgreSQL
37. Spring Security. Spring Security. URL: https://ru.wikipedia.org/wiki/Spring_Security
38. Yup. Yup. URL: https://www.npmjs.com/package/yup
39. Liquibase. Liquibase. URL: https://docs.liquibase.com/home.html
40. IntelliJ_IDEA. IntelliJ_IDEA. URL: https://uk.wikipedia.org/wiki/IntelliJ_IDEA
41. Visual_Studio_Code. Visual_Studio_Code. URL:https://uk.wikipedia.org/wiki/VisualStudioCode
42. DBeaver. DBeaver. URL: https://uk.wikipedia.org/wiki/DBeaver
43. visualvm. Visual VM URL: https://visualvm.github.io/
44. Docker-compose. Docker-compose. URL: https://itedu.center/ua/blog/devops/docker-compose-tutorial/
45. JSON. JSON. URL: https://uk.wikipedia.org/wiki/JSON
46. DTO. DTO. URL: https://uk.wikipedia.org/wiki/DTO
47. OpenAPI. OpenAPI. URL: https://uk.wikipedia.org/wiki/OpenAPI
48. MSE. MSE. URL: https://uk.wikipedia.org/wiki/MSE

ДОДАТКИ

Додаток А

Структура бази даних вебсервісу прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу



Додаток Б

Розгорнута структура класів вебсервісу прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу

Додаток В

Програмні коди

Лістинг AuthController.java:

```
package com.predictor.bebase.security.api.controller;

import com.predictor.bebase.security.api.dto.AuthRequestDto;
import com.predictor.bebase.security.api.dto.AuthResponseDto;
import com.predictor.bebase.security.api.dto.UserDto;
import com.predictor.bebase.security.service.AuthService;
import com.predictor.bebase.security.service.UserService;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.web.bind.annotation.*;

import static com.predictor.bebase.security.transformer.dto.AuthResponseDtoTransformer.toAuthResponseDto;
import static com.predictor.bebase.security.transformer.dto.UserDtoTransformer.toUserDto;
import static com.predictor.bebase.security.transformer.model.AuthRequestTransformer.toAuthRequest;
import static com.predictor.bebase.security.transformer.model.UserTransformer.toUser;

@Slf4j
@CrossOrigin
@RestController
@RequestMapping("/public/auth/")
@RequiredArgsConstructor
public class AuthController {

    private final UserService userService;
    private final AuthService authService;

    @PostMapping("register")
    public UserDto register(@RequestBody UserDto dto) {
        log.info("Registering user: %s".formatted(dto.toString()));
        return toUserDto(userService.create(toUser(dto)));
    }

    @PostMapping("login")
    public AuthResponseDto login(@RequestBody AuthRequestDto authRequestDto) {
        log.info("Sign in by: %s".formatted(authRequestDto));
        return toAuthResponseDto(authService.authorizeAccount(toAuthRequest(authRequestDto)));
    }
}
```

Лістинг AuthenticationContext.java:

```
package com.predictor.bebase.security.checkpoint.service.model;

import com.predictor.bebase.security.api.dto.UserDto;
import com.predictor.bebase.security.service.model.User;
import org.springframework.security.authentication.AuthenticationServiceException;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import static com.predictor.bebase.security.transformer.dto.UserDtoTransformer.toUserDto;
public class AuthenticationContext {

    public static UserDto auth() {
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        if (auth == null) {
            throw new AuthenticationServiceException("Authentication failed!");
        }
        return toUserDto( (User) auth.getPrincipal());
    }
}
```

Лістинг AuthenticationConverterFilter.java:

```
package com.predictor.bebase.security.checkpoint.filter;

import static com.predictor.bebase.security.checkpoint.filter.util.FilterLogFormatUtil.getFormatedTitle;
import static com.predictor.bebase.security.transformer.model.UserTransformer.toUser;
import static org.springframework.security.authentication.UsernamePasswordAuthenticationToken.authenticated;
import static org.springframework.security.core.context.SecurityContextHolder.getContext;

import java.io.IOException;


import org.springframework.security.core.Authentication;
import org.springframework.security.oauth2.jwt.Jwt;
import org.springframework.web.filter.OncePerRequestFilter;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
```

```
import jakarta.servlet.http.HttpServletResponse;
import lombok.extern.slf4j.Slf4j;

@Slf4j
public class AuthenticationConverterFilter extends OncePerRequestFilter {

    private final static String FILTER_LOG_TITLE = getFormatedTitle(AuthenticationConverterFilter.class.getSimpleName());

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
            throws ServletException, IOException {

        log.info(FILTER_LOG_TITLE);

        Authentication oldAuthentication = getContext().getAuthentication();
        getContext().setAuthentication(authenticated(toUser((Jwt) oldAuthentication.getPrincipal()),
                oldAuthentication.getCredentials(), oldAuthentication.getAuthorities()));

        filterChain.doFilter(request, response);
    }

    public static AuthenticationConverterFilter authenticationConverterFilter() {
        return new AuthenticationConverterFilter();
    }
}
```

Лістинг AuthRequest.java:

```
package com.predictor.bebase.security.service.model;

import lombok.*;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class AuthRequest {
    private String email;
    private String password;
}
```

Лістинг WebSecurityConfig.java:

```
package com.predictor.bebase.security.checkpoint.config;

import com.predictor.bebase.security.service.model.RsaPublicKey;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.annotation.Order;
import org.springframework.scheduling.annotation.EnableAsync;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.oauth2.server.resource.web.DefaultBearerTokenResolver;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.access.intercept.AuthorizationFilter;
import org.springframework.security.web.savedrequest.HttpSessionRequestCache;

import java.security.interfaces.RSAPublicKey;

import static
com.predictor.bebase.security.checkpoint.authentication.ImplAuthenticationEntryPoint.implAuthenticationEntryPoint;
import static com.predictor.bebase.security.checkpoint.config.BaseSecurityDslConfig.baseSecurityDslConfig;
import static com.predictor.bebase.security.checkpoint.filter.AuthenticationConverterFilter.authenticationConverterFilter;
import static org.springframework.security.oauth2.jwt.JwtDecoders.fromIssuerLocation;
import static org.springframework.security.oauth2.jwt.NimbusJwtDecoder.withPublicKey;

@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class WebSecurityConfig {

    @Value("${oath2.issuer.url}")
    private String issuerUrl;
    private final RsaPublicKey rsaPublicKey;

    @Bean
    @Order(1)
    public SecurityFilterChain publicFilterChain(HttpSecurity http) throws Exception {

        return http.securityMatcher("/**")
                .apply(baseSecurityDslConfig()).and()
                .authorizeHttpRequests(auth -> auth.requestMatchers("/**").permitAll())
                .build();
    }
```

```
    @Bean
    @Order(0)
    public SecurityFilterChain privateFilterChain(HttpSecurity http) throws Exception {

        return http.securityMatcher("/private/**")
                .apply(baseSecurityDslConfig()).and()
                .requestCache(cache -> cache.requestCache(httpSessionRequestCache()))
                .authorizeHttpRequests(auth -> auth.requestMatchers("/private/**").authenticated())
                .oauth2ResourceServer(oauth2 -> oauth2.authenticationEntryPoint(implAuthenticationEntryPoint())
                        .bearerTokenResolver(defaultBearerTokenResolver())
                        .jwt(jwt -> jwt.decoder(withPublicKey(rsaPublicKey.getRsaPublicKey()).build())))
                .addFilterAfter(authenticationConverterFilter(), AuthorizationFilter.class)
                .build();
    }

    private static HttpSessionRequestCache httpSessionRequestCache() {
        return new HttpSessionRequestCache();
    }

    private static DefaultBearerTokenResolver defaultBearerTokenResolver() {
        return new DefaultBearerTokenResolver();
    }
}
```

Лістинг SiteController.java:

```
package com.predictor.bebase.site.api.controller;

import com.predictor.bebase.neural_network.client.dto.PredictResponseDto;
import com.predictor.bebase.neural_network.client.dto.UserFactorsDto;
import com.predictor.bebase.neural_network.service.NeuralNetworkService;
import com.predictor.bebase.postgresql.exception.SqlConnectionException;
import com.predictor.bebase.security.checkpoint.service.model.AuthenticationContext;
import com.predictor.bebase.site.api.dto.SiteDto;
import com.predictor.bebase.site.api.dto.SiteDtoWithFactors;
import com.predictor.bebase.site.api.dto.search_criteria.FilterByDto;
import com.predictor.bebase.site.api.dto.search_criteria.HiddenFieldsDto;
import com.predictor.bebase.site.api.dto.search_criteria.SearchSiteCriteriaDto;
import com.predictor.bebase.site.service.SiteService;
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;

import java.util.List;

import static com.predictor.bebase.site.transformer.SearchSiteCriteriaDtoTransformer.toSearchSiteCriteriaDto;

@CrossOrigin
@RestController
@RequestMapping
@RequiredArgsConstructor
public class SiteController {

    private final SiteService siteService;

    @GetMapping("/private/site/all")
    public Page<SiteDto> findAll(SearchSiteCriteriaDto criteriaDto) {
        var auth = AuthenticationContext.auth();
        return siteService.getAll(toSearchSiteCriteriaDto(auth.getId(), criteriaDto));
    }

    @GetMapping("/private/site/result/{shopId}")
    public PredictResponseDto getResults(@PathVariable Integer shopId) {
        var auth = AuthenticationContext.auth();
        return siteService.getResult(shopId);
    }
    @PostMapping("/private/site/create")
    public SiteDto create(@RequestBody SiteDtoWithFactors siteDtoWithFactors) {
        var auth = AuthenticationContext.auth();
        siteDtoWithFactors.getSiteDto().setUserId(auth.getId());
        return siteService.create(siteDtoWithFactors);
    }

    @PutMapping("/private/site/update/{siteId}")
    public SiteDto update(@PathVariable Integer siteId, @RequestBody SiteDtoWithFactors siteDtoWithFactors) {
        var auth = AuthenticationContext.auth();
        siteDtoWithFactors.getSiteDto().setUserId(auth.getId());
        siteDtoWithFactors.getSiteDto().setId(siteId);
        return siteService.update(siteDtoWithFactors);
    }

    @DeleteMapping("/private/site/delete/{siteId}")
    public void delete(@PathVariable Integer siteId) {
        var auth = AuthenticationContext.auth();
        siteService.delete(SiteDto.builder().id(siteId).userId(auth.getId()).build());
    }
```

```
}
```

Лістинг UserRepository.java:

```
package com.predictor.bebase.security.repository;

import com.predictor.bebase.security.api.dto.UserDto;
import com.predictor.bebase.security.repository.entity.UserEntity;
import com.predictor.bebase.security.service.model.User;
import database.generated.tables.AthUser;
import lombok.RequiredArgsConstructor;
import org.jooq.DSLContext;
import org.springframework.stereotype.Repository;
import static com.predictor.bebase.security.transformer.model.UserTransformer.toUser;
import static com.predictor.bebase.security.transformer.entity.UserEntityTransformer.toUserEntity;

@Repository
@RequiredArgsConstructor
public class UserRepository {
    private final DSLContext context;

    public User create(User user) {
        var userEntity = toUserEntity(user);
        var au = AthUser.ATH_USER;

        return toUser(context.insertInto(au)
                .set(au.EMAIL, userEntity.getEmail())
                .set(au.PASSWORD, userEntity.getPassword())
                .set(au.USERNAME, userEntity.getUsername())
                .returningResult(au.ID, au.USERNAME, au.PASSWORD, au.EMAIL)
                .fetchOneInto(UserEntity.class));
    }

    public UserEntity getUserByEmail(String email) {
        var au = AthUser.ATH_USER;
        return context.select().from(au)
                .where(au.EMAIL.eq(email))
                .fetchOneInto(UserEntity.class);
    }
}
```

Лістинг SearchSiteCriteriaDtoTransformer.java:

```
package com.predictor.bebase.site.transformer;

import com.predictor.bebase.base.api.dto.BasePaginationDto;
import com.predictor.bebase.security.api.dto.AuthResponseDto;
import com.predictor.bebase.security.api.dto.AuthTokenDto;
import com.predictor.bebase.security.service.model.AuthResponse;
import com.predictor.bebase.site.api.dto.SiteDto;
import com.predictor.bebase.site.api.dto.search_criteria.FilterByDto;
import com.predictor.bebase.site.api.dto.search_criteria.HiddenFieldsDto;
import com.predictor.bebase.site.api.dto.search_criteria.SearchSiteCriteriaDto;

import static com.predictor.bebase.security.transformer.dto.UserDtoTransformer.toUserDto;

public class SearchSiteCriteriaDtoTransformer {

    public static SearchSiteCriteriaDto toSearchSiteCriteriaDto(Integer userId, SearchSiteCriteriaDto dto) {
        var fb = dto.getFilterByDto();
        var hf = dto.getHiddenFieldsDto();
        var bp = dto.getBasePaginationDto();

        return SearchSiteCriteriaDto.builder().hiddenFieldsDto(HiddenFieldsDto.builder().userId(userId).build())
                .filterByDto(FilterByDto.builder().domain(fb.getDomain()).name(fb.getName()).siteType(fb.getSiteType())
         .build()).basePaginationDto(BasePaginationDto.builder().page(bp.getPage()).size(bp.getSize()).build()).build();

    }
}
```

Лістинг SiteService.java:

```
package com.predictor.bebase.site.service;

import com.predictor.bebase.neural_network.client.dto.PredictResponseDto;
import com.predictor.bebase.neural_network.repository.PredictionsRepository;
import com.predictor.bebase.neural_network.service.NeuralNetworkService;
import com.predictor.bebase.site.api.dto.SiteDto;
import com.predictor.bebase.site.api.dto.SiteDtoWithFactors;
import com.predictor.bebase.site.api.dto.search_criteria.SearchSiteCriteriaDto;
import com.predictor.bebase.site.repository.SiteRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Service;

import java.util.List;
```

```
import static com.predictor.bebase.neural_network.converter.MarkConverter.toMark;

@Service
@RequiredArgsConstructor
public class SiteService {
    private final SiteRepository siteRepository;
    private final NeuralNetworkService neuralNetworkService;
    private final PredictionsRepository predictionsRepository;

    public Page<SiteDto> getAll(SearchSiteCriteriaDto searchSiteCriteriaDto) {
        return siteRepository.getAll(searchSiteCriteriaDto);
    }

    public PredictResponseDto getResult(Integer shopId) {
        return predictionsRepository.getResults(shopId);
    }

    //public List<Double> getHistory(Integer userId, Integer siteId) {
       // return siteRepository.getHistory(userId, siteId);
   // }

   public SiteDto create(SiteDtoWithFactors siteDtoWithFactors) {
       PredictResponseDto predictionResult = neuralNetworkService.predict(siteDtoWithFactors);
        var createdSite = siteRepository.create(siteDtoWithFactors.getSiteDto());
        predictionsRepository.create(siteDtoWithFactors.getUserFactorsDto().getDates(),
siteDtoWithFactors.getUserFactorsDto().getVisits(), predictionResult.getDates(), predictionResult.getVisits(),
                createdSite.getId(), siteDtoWithFactors.getUserFactorsDto().getPredictionType());
        return createdSite;
    }

    public SiteDto update(SiteDtoWithFactors siteDtoWithFactors) {
        PredictResponseDto predictionResult = neuralNetworkService.predict(siteDtoWithFactors);
        var updatedSite = siteRepository.update(siteDtoWithFactors.getSiteDto());
        predictionsRepository.create(siteDtoWithFactors.getUserFactorsDto().getDates(),
siteDtoWithFactors.getUserFactorsDto().getVisits(), predictionResult.getDates(), predictionResult.getVisits(),
                updatedSite.getId(), siteDtoWithFactors.getUserFactorsDto().getPredictionType());
        return updatedSite;
    }

    public void delete(SiteDto siteDto) {
         siteRepository.delete(siteDto);
    }
}
```

Лістинг NeuralNetworkClient.java:

```
package com.predictor.bebase.neural_network.client;


import com.predictor.bebase.neural_network.client.dto.PredictResponseDto;
import com.predictor.bebase.neural_network.client.dto.UserFactorsDto;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.http.MediaType;
import org.springframework.stereotype.Component;
import org.springframework.web.reactive.function.BodyInserters;
import org.springframework.web.reactive.function.client.WebClient;
import reactor.core.publisher.Mono;

import java.time.Duration;
import java.util.Arrays;

@Component
public class NeuralNetworkClient {

    private final WebClient webClient;
    private static final Logger logger = LoggerFactory.getLogger(NeuralNetworkClient.class);
    public NeuralNetworkClient(WebClient.Builder webClientBuilder) {
        this.webClient = webClientBuilder.baseUrl("http://localhost:5000/").build();
    }

    public PredictResponseDto predict(UserFactorsDto userFactorsDto) {

        String endpointPart = userFactorsDto.getPredictionType().toString().equals("D") ? "/predict-by-dates" : "/predict-by-
months";

         return webClient.post()
                 .uri(endpointPart)
                 .contentType(MediaType.APPLICATION_JSON)
                 .body(BodyInserters.fromValue(userFactorsDto))
                 .retrieve()
                 .bodyToMono(PredictResponseDto.class)
                 .block();
    }

}
```

Лістинг neural_network.py:

```
from flask import Flask, request, jsonify
import pandas as pd
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.model_selection import train_test_split

app = Flask(__name__)

@app.route('/predict-by-dates', methods=['POST'])
def predict():
    data = request.json
    dates = pd.to_datetime(data['dates'])
    visits = np.array(data['visits'])
    forecast_period = data['forecastPeriod']
    epochs_number = data['epochsNumber']
    visits = visits.astype(float)

    def prepare_data(data, time_steps):
        X, y = [], []
        for i in range(len(data)-time_steps):
            X.append(data[i:i+time_steps])
            y.append(data[i+time_steps])
        return np.array(X), np.array(y)

    time_steps = 5
    X, y = prepare_data(visits, time_steps)

    model = Sequential()
    model.add(LSTM(units=50, activation='relu', input_shape=(time_steps, 1)))
    model.add(Dense(units=1))
    model.compile(optimizer='adam', loss='mean_squared_error')
    X = np.reshape(X, (X.shape[0], X.shape[1], 1))
    model.fit(X, y, epochs=epochs_number, batch_size=1)

    def predict_next_days(model, initial_data, time_steps, num_days):
        data = initial_data[-time_steps:].reshape(-1, 1)
        predictions = []
        for _ in range(num_days):
            prediction = model.predict(data[-time_steps:].reshape(1, time_steps, 1))
            predictions.append(prediction[0, 0])
            data = np.append(data, prediction, axis=0)
        return predictions

    num_days = forecast_period
    predicted_visits = predict_next_days(model, visits, time_steps, num_days)
    # Перетворення масиву дат в об'єкт datetime64
    predicted_dates = pd.date_range(start=dates.max() + pd.Timedelta(days=1), periods=num_days)

    predicted_visits = np.maximum(predicted_visits, 0)

    response_data = {'dates': predicted_dates.strftime('%Y-%m-%d').tolist(), 'visits': [str(visit) for visit in
predicted_visits]}
    return jsonify(response_data)

    #**************************************
    #***************MONTHS*****************
    #**************************************

@app.route('/predict-by-months', methods=['POST'])
def predict_by_months():
    data = request.json
    months = pd.to_datetime(data['dates']).to_period('M')
    visits = np.array(data['visits'])
    epochs_number = data['epochsNumber']
    forecast_period = data['forecastPeriod']
    visits = visits.astype(float)

    def prepare_data(data, time_steps):
        X, y = [], []
        for i in range(len(data)-time_steps):
            X.append(data[i:i+time_steps])
            y.append(data[i+time_steps])
        return np.array(X), np.array(y)

    time_steps = 5
    X, y = prepare_data(visits, time_steps)

    model = Sequential()
    model.add(LSTM(units=50, activation='relu', input_shape=(time_steps, 1)))
    model.add(Dense(units=1))
    model.compile(optimizer='adam', loss='mean_squared_error')
    X = np.reshape(X, (X.shape[0], X.shape[1], 1))
    model.fit(X, y, epochs=epochs_number, batch_size=1)

    def predict_next_months(model, initial_data, time_steps, num_months):
```

```
        data = initial_data[-time_steps:].reshape(-1, 1)
        predictions = []
        for _ in range(num_months):
            prediction = model.predict(data[-time_steps:].reshape(1, time_steps, 1))
            predictions.append(prediction[0, 0])
            data = np.append(data, prediction, axis=0)[1:]
        return predictions

    num_months = forecast_period
    future_months = (months[-1].to_timestamp() + pd.DateOffset(months=1)).to_period('M')
    future_months_range = pd.period_range(start=future_months, periods=num_months, freq='M')
    predicted_visits = predict_next_months(model, visits, time_steps, num_months)

    predicted_visits = np.maximum(predicted_visits, 0)

    response_data = {'dates': future_months_range.strftime('%Y-%m').tolist(), 'visits': [str(visit) for visit in
predicted_visits]}
    return jsonify(response_data)

if __name__ == '__main__':
    app.run(debug=True)
```

Лістинг JwtProperties.java:

```
package com.predictor.bebase.security.checkpoint.props;


import lombok.Data;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;

@Data
@ConfigurationProperties(prefix = "jwt")
@Component
public class JwtProperties {
    private Token token;

    @Data
    public static class Token {
        private String privateKey;
        private String publicKey;
        private String bearer;
        private Validity validity;
        private Issuer issuer;

        @Data
        public static class Validity {
            private Long time;
            private Long refreshTime;
        }

        @Data
        public static class Issuer {
            private String url;
            private String jwksUrl;
            private String authorizationUrl;
        }
    }
}
```

Лістинг PostgresqlConfig.java:

```
package com.predictor.bebase.postgresql.config;

import com.predictor.bebase.postgresql.exception.SqlConnectionException;
import com.predictor.bebase.postgresql.props.PostgresqlProperties;
import lombok.RequiredArgsConstructor;
import org.jooq.DSLContext;
import org.jooq.SQLDialect;
import org.jooq.impl.DSL;
import org.springframework.context.annotation.Bean;
import org.springframework.stereotype.Component;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

@Component
@RequiredArgsConstructor
public class PostgresqlConfig {

    private final PostgresqlProperties props;

    @Bean("postgresqlDslContext")
    public DSLContext dslContext(){
        try{
            Connection conn = DriverManager.getConnection(props.getUrl(), props.getUsername(), props.getPassword());
            return DSL.using(conn, SQLDialect.POSTGRES);
        } catch (SQLException ex){
            throw new SqlConnectionException();
```

```
        }
    }
}
```

Лістинг UserTransformer.java:

```
package com.predictor.bebase.security.transformer.model;

import com.predictor.bebase.security.api.dto.UserDto;
import com.predictor.bebase.security.repository.entity.UserEntity;
import com.predictor.bebase.security.service.model.User;
import org.springframework.security.oauth2.jwt.Jwt;

import static java.lang.Math.toIntExact;

public class UserTransformer {
    public static User toUser(UserDto dto) {
        return
User.builder().id(dto.getId()).email(dto.getEmail()).password(dto.getPassword()).username(dto.getUsername()).build();
    }

    public static User toUser(UserEntity userEntity) {
        return
User.builder().id(userEntity.getId()).email(userEntity.getEmail()).password(userEntity.getPassword()).username(userEntity.getUs
ername()).build();
    }

    public static User toUser(Jwt jwt) {
        return toUser(toIntExact(jwt.getClaim("user_id")), jwt.getSubject());
    }

    public static User toUser(Integer id, String email) {
        return User.builder().id(id).email(email).build();
    }

}
```

Лістинг SearchSiteCriteriaDto.java:

```
package com.predictor.bebase.site.api.dto.search_criteria;

import com.predictor.bebase.base.api.dto.BasePaginationDto;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder(toBuilder = true)
@NoArgsConstructor
@AllArgsConstructor
public class SearchSiteCriteriaDto {
    @Builder.Default
    private HiddenFieldsDto hiddenFieldsDto = new HiddenFieldsDto();
    @Builder.Default
    private FilterByDto filterByDto = new FilterByDto();
    @Builder.Default
    private BasePaginationDto basePaginationDto = new BasePaginationDto();
}
```

Лістинг `SignInForm`.tsx:

```
//Antd
import { Button, Modal, Table } from 'antd';
import { Field, Form, Formik } from 'formik';
//Yup
import * as Yup from 'yup';
import { ISignIn } from './interfaces/ISignIn';
import { useAuth } from '../../hooks/useAuth';


export const SignInForm = () => {

    const { signIn } = useAuth();

    const validationSchema = Yup.object().shape({
        password: Yup.string()
            .min(2, 'Too short password.')
            .max(50, 'Too long password.')
            .required('Password is required.'),
        email: Yup.string()
            .email("Email is not valid.")
            .required('Email is required')
    });

    const initialValues = (): ISignIn => {
        return {
            email: '',
            password: ''
```

```
        }
    }

    const onSubmit = (signInFromForm: ISignIn, /*{ resetForm }*/) => {
        signIn(signInFromForm);
    }

    return (

        <Formik
            enableReinitialize={true}
            initialValues={initialValues()}
            validationSchema={validationSchema}
            onSubmit={onSubmit}
            >
            {({ errors, touched }) => (
                <Form className='auth-main-form'>
                    <div className='auth-main-form-field-block'>
                        <label htmlFor={'email'}>Email </label>
                        <Field id={'email'} className={'auth-main-form-field-block__field'} name={'email'} />
                    </div>
                    <div className='auth-main-form__error'>{errors.email && touched.email ? errors.email : null}</div>
                    <div className='auth-main-form-field-block'>
                        <label htmlFor={'password'}>Пароль </label>
                        <Field id={'password'} className={'auth-main-form-field-block__field'} name={'password'} />
                    </div>
                    <div className='auth-main-form__error'>{errors.password && touched.password ? errors.password : null}</div>
                    <Button className='auth-main-form__button'
                        htmlType='submit' type="primary" size={'middle'}>Увійти</Button>
                </Form>
            )}
        </Formik>
    )
}
```

Лістинг SignUp.tsx:

```
//React
import React from 'react';
import './style/sign-in-up.scss'
import { SignInForm } from './SignInForm';
import { Button } from 'antd';
import { SignUpForm } from './SignUpForm';
import { Link } from 'react-router-dom';

export const SignUp = () => {

    return <div className='auth'>
        <div className='auth-bridge'>
            <div className='auth-bridge__title'>Вхід</div>
            <Link className='auth-bridge__link' to="/sign-in">
                Вхід
            </Link>
        </div>
        <div className='auth-main'>
            <div className='auth-main__title'>Реєстрація</div>
            <SignUpForm />
        </div>
    </div>
}
```

Лістинг SitePredictionMainTable.tsx:

```
import React, { useEffect, useState } from 'react';
import { Button, Progress, Spin, Table } from 'antd';
import { ColumnsType } from 'antd/lib/table';
import { ISiteModalControl } from './interface/ISiteModalControl';
import { useDispatch, useSelector } from "react-redux";
import { RootState } from '../../redux/store';
import { useSites } from '../../hooks/useSites';
import { SiteOperation } from './enum/SiteOperation';
import { setCurrentSite } from '../../redux/slices/site.slice';
import { ISite } from './interface/ISite';
import { ISiteModalProps } from './props/ISiteModalProps';
import { PredictionModal } from './PredictionModal';
import './style/table-style.scss'
import { useHistory } from 'react-router-dom';

export const SitePredictionMainTable = () => {

    const [modalControl, setModalControl] = useState<ISiteModalControl>({isOpen:false, mod: null});
    const sites = useSelector((state: RootState) => state.site.sites)
    const isRequestInProgress = useSelector((state: RootState) => state.requestProgressSlice.isRequestInProgress)
    const dispatch = useDispatch();
    const history = useHistory();

    const { getHistory ,getAllSites, deleteSite, getResults} = useSites();

    const showModal = (operation : SiteOperation) => {
        setModalControl({isOpen: true, mod: operation});
```

```
    }

    const onResults = (site: ISite) => {
        dispatch(setCurrentSite(site));
        getResults(site.id);
        history.push(`/results`);
    }


    const onHistory = (site: ISite) => {
        dispatch(setCurrentSite(site));
        getResults(site.id);
        history.push(`/history`);
    }

    const modalProps: ISiteModalProps = {
        modalControl: modalControl,
        setModalControl : setModalControl
    }

    const columns: ColumnsType<any> = [
        {
            title: "№",
            dataIndex: "№",
            key: "№",
            width: '4%',
            render: (text, record, index) => index + 1,
        },
        {
            title: "Назва",
            dataIndex: "name",
            key: "name",
            width: "17%"
        },
        {
            title: "Доменне ім'я",
            dataIndex: "domain",
            key: "domain",
            width: "17%",
            render: (text) => <div style={{color: 'white', fontSize:'20px', fontStyle:'italic',
            fontWeight: 'bold'}}>{text}</div>,
        },
        {
            title: "Тип",
            dataIndex: "siteType",
            key: "siteType",
            width: "12%",
            render: (text) => <div style={{borderRadius:'10px'}} className='ant-table-cell_site-type'>{text}</div>,
        },
        {
            title: "Операції",
            dataIndex: "actions",
            key: "actions",
            width: "40%",
            render: (_: any, site: ISite) => {
                return <SitePredictionMainTableActionBlock site={site}/>
              }
        },

    ];

    const SitePredictionMainTableActionBlock = (props : any) => {

        return (
            <div className='table-row-cell-operations'>
                <button className="button-77" role="button" disabled={modalControl.isOpen}
                            onClick={() => { onResults(props.site) }}>Результати  </button>
                <button className="button-77" role="button" disabled={modalControl.isOpen}
                            onClick={() => { onHistory(props.site) }}>Графік   </button>
                <button className="button-77" role="button" disabled={modalControl.isOpen}
                            onClick={() => { dispatch(setCurrentSite(props.site)); showModal(SiteOperation.EDIT) }}>Редагувати
</button>
                <button className="button-77" role="button" disabled={modalControl.isOpen}
                            onClick={() => deleteSite(props.site)}>Видалити</button>
            </div>
        );
    };

    const SitePredictionMainTableFooter = () => {
        return <Button className='button-77' disabled={modalControl.isOpen}
                       style={{ borderRadius: '30px 30px 30px 30px', minHeight: "70px", fontSize: '30px', fontStyle: 'italic',
                         height: '100%', width: '100%', background: 'blue', color: 'white'}}
                       onClick={() => showModal(SiteOperation.CREATE)}>Створити</Button>;
    };

    useEffect(() => {
        getAllSites();
    }, []);

    return (
```

```
        <>
        <Table columns={columns}
                dataSource={sites}
                footer={SitePredictionMainTableFooter}
                pagination={{ pageSize: 4 }}
            />
        <PredictionModal {...modalProps}/>
        {isRequestInProgress ?
        <Spin tip="Завантаження" size="large">
            <div className="content" />
        </Spin> : null}
        </>

    );
};
```

Додаток Г

Презентаційний матеріал

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Метод прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу

Виконав:
студент 4 курсу, групи КН-20-1
Богдан **Корчевський**
Керівник:
зав. кафедри КН, д.т.н., професор
Олександр **Бармак**

Актуальність

У сучасному інтернет-просторі відвідуваність вебсайтів визначає їх успішність та вплив на аудиторію. Розвиток технологій та величезна кількість інформації роблять конкуренцію серед вебресурсів жорсткою. Тому актуальним стає питання розробки ефективного методу прогнозування відвідуваності вебсайтів з метою підвищення їх ефективності та конкурентоспроможності.

Мета і задачі роботи

Метою кваліфікаційної роботи бакалавра є покращення ефективності прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу реалізованого у вигляді вебсервісу.

Завдання кваліфікаційної роботи бакалавра:

- провести аналіз предметної області прогнозування відвідуваності вебсайтів; виконати аналіз методів прогнозування за допомогою нейромережевого підходу; виконати аналіз наявних рішень щодо подібних задач;
- розробити метод прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу;
- спроєктувати структуру вебресурсу з автоматизованим прогнозуванням відвідуваності вебсайтів за допомогою нейромережевого підходу;
- спроєктувати структуру бази даних вебресурсу;
- виконати програмну реалізацію вебресурсу;
- провести тестування розробленого вебресурсу;
- дослідити ефективність розробленого методу прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу.

Підходи до прогнозування відвідуваності вебсайтів

Прогнозування відвідуваності вебсайтів є важливим завданням для власників і адміністраторів сайтів, а також для маркетологів і аналітиків. Існує кілька підходів до прогнозування відвідуваності вебсайтів, кожен з яких має свої переваги і недоліки.

Ось деякі з них:

- **Статистичні моделі**: Використання статистичних моделей, таких як ARIMA (авторегресійне інтегроване ковзне середнє) або Exponential Smoothing (експоненційне згладжування), для аналізу і прогнозування часових рядів відвідуваності вебсайту.
- **Машинне навчання**: Використання алгоритмів машинного навчання, таких як лінійна регресія, дерева рішень, ансамблеві методи (наприклад, Random Forest або Gradient Boosting), нейронні мережі тощо для прогнозування відвідуваності вебсайту на основі різноманітних факторів.
- **Аналіз часових рядів**: Використання аналізу часових рядів для виявлення циклів, трендів та сезонності відвідуваності вебсайту, що допомагає у прогнозуванні майбутніх значень.

Кожен з цих підходів має свої переваги і може бути використаний в залежності від конкретних потреб і можливостей власника вебсайту.

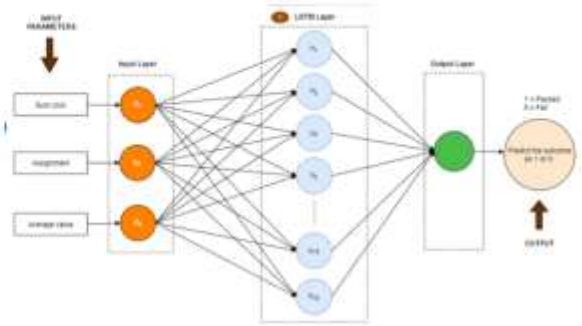
Переваги та недоліки нейромереж різної архітектури

	Перцептрон	RNN	CNN
Переваги	Простота реалізації, добре підходить для завдань бінарної класифікації та лінійних задач.	Здатність враховувати послідовності даних, широко використовується в задачах обробки природної мови та часових рядів.	Ефективність у роботі зі зображеннями, вміння виявляти просторові шаблони.
Недоліки	Обмежена здатність до навчання складних нелінійних залежностей, потребує великої кількості даних для навчання в складних задачах.	Проблема зникнення та вибуху градієнту, обмежена пам'ять для довгих послідовностей.	Потребують багато даних для навчання, можуть бути складні у використанні для інших типів даних.

Продовження таблиці переваг та недоліків нейромереж різної архітектури

	DNN	Модульні нейронні мережі	Автокодери
Переваги	Здатність моделювати складні залежності, висока точність у багатьох задачах.	Зручність у побудові та розширенні, можливість використання готових модулів для побудови складних архітектур.	Використовуються для зменшення розмірності даних та відновлення даних, ефективні для роботи з великими об'ємами даних.
Недоліки	Потребують великої обчислювальної потужності та об'єму даних для навчання, можуть страждати від перенавчання.	Можуть виникати проблеми зі сумісністю модулів, потребують уважного проектування.	Можуть втрачати деяку інформацію під час зменшення розмірності даних, вимагають уважного налаштування гіперпараметрів.

LSTM NEURAL NETWORK



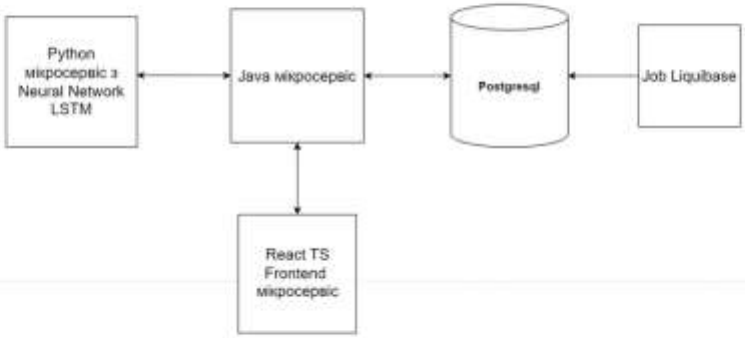
Довготривала короткострокова пам'ять (Long Short-Term Memory, LSTM) є особливим типом рекурентних нейронних мереж, який спеціально розроблений для роботи з послідовними даними та уникнення проблеми зникаючих і вибуваючих градієнтів. Вони відзначаються ефективністю у роботі з послідовностями будь-якої довжини та здатністю моделі враховувати довгострокові залежності у даних. Перевагами LSTM є їхня здатність до роботи з довгими залежностями та висока точність прогнозів. Однак слід зазначити, що вони можуть мати потребу у складній настройці параметрів моделі для досягнення оптимальних результатів.

Інструменти для розробки

Для розробки вебсервісу було використано мову програмування Java, Python, Type Script, фреймворки Spring boot, Flask, React, систему керування базами даних Postgresql, систему керування змінами схеми бази даних Liquibase.

Діаграма класів

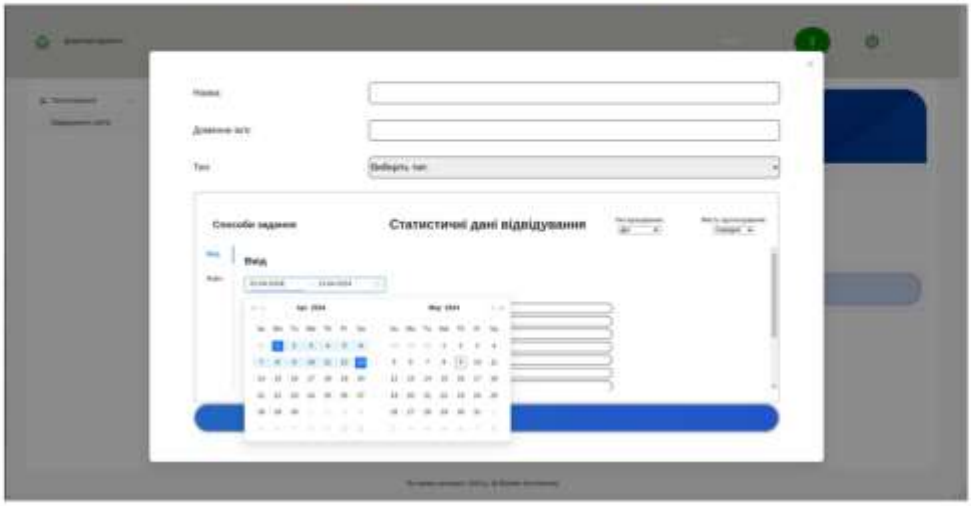
Загальна архітектура системи



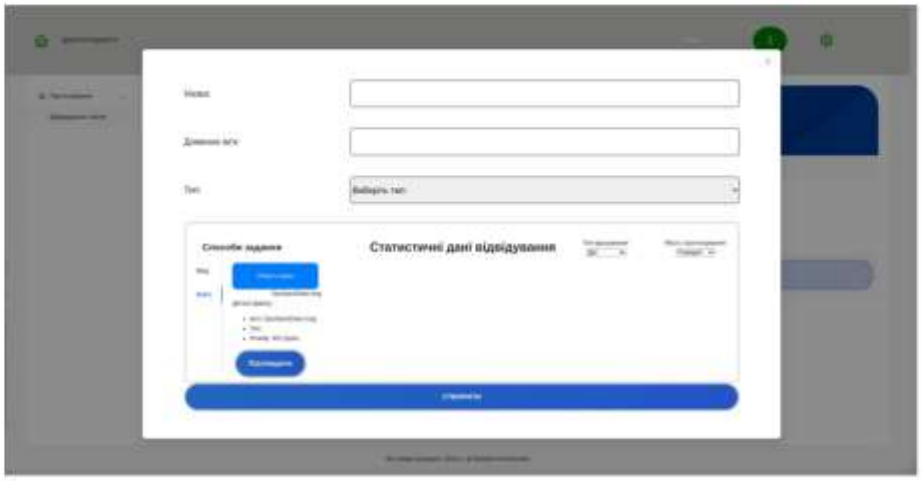
Сторінка входу

Сторінка прогнозування

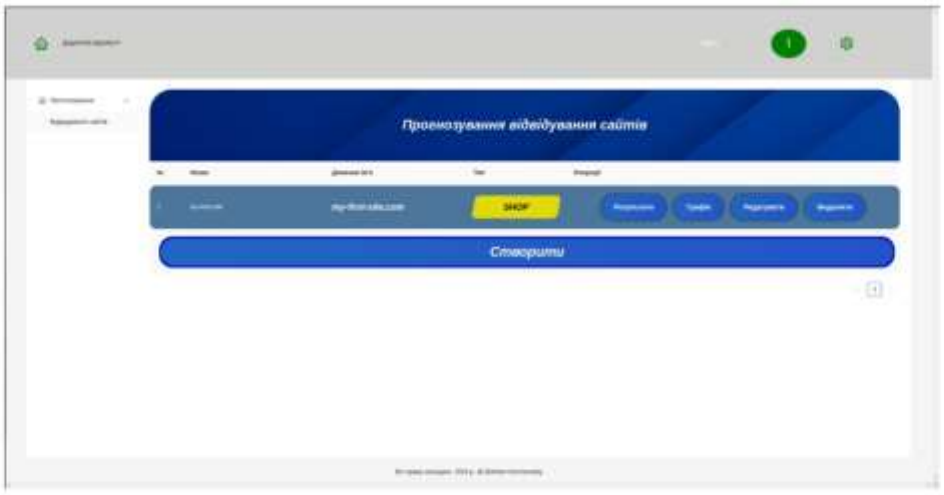
Модальне вікно операції створення сайту для прогнозування відвідування



Можливість завантаження даних через файл

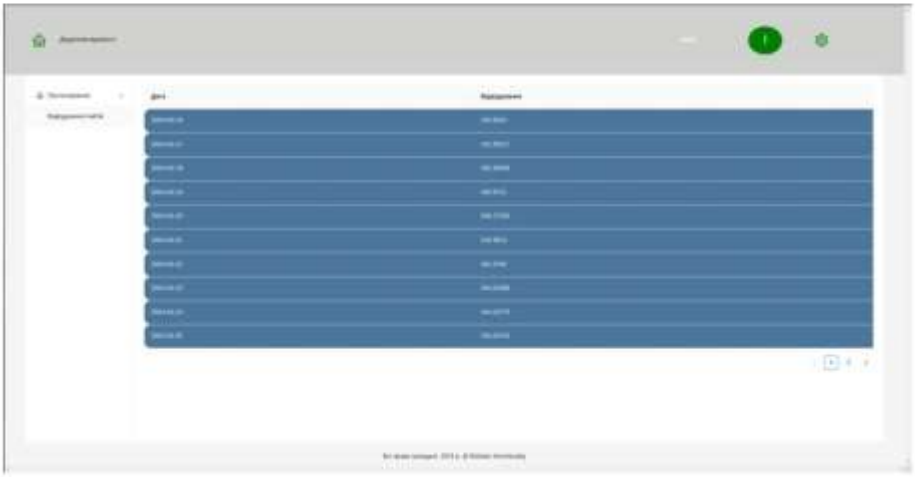


Оновлена сторінка прогнозування після додавання сайту

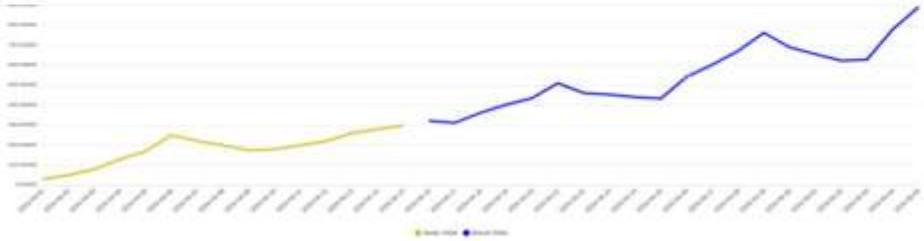


Сторінка з графіком результату

Сторінка з результатами

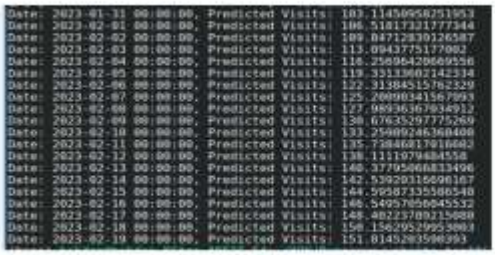


Результати



- не лінійні дані

16	2023-01-17	55.172414
17	2023-01-18	58.620690
18	2023-01-19	62.068966
19	2023-01-20	65.517241
20	2023-01-21	68.965517
21	2023-01-22	72.413793
22	2023-01-23	75.862069
23	2023-01-24	79.310345
24	2023-01-25	82.758621
25	2023-01-26	86.206897
26	2023-01-27	89.655172
27	2023-01-28	93.103448
28	2023-01-29	96.551724
29	2023-01-30	100.000000



- лінійні дані

Висновки

Метою кваліфікаційної роботи бакалавра є покращення ефективності прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу, реалізованого у вигляді вебсервісу.

Для досягнення поставленої мети було використано метод з використанням нейромережі LSTM. Цей метод був обраний через його здатність працювати з послідовними даними, що дозволяє передбачати трафік на основі історичних даних відвідуваності. LSTM нейромережі дозволяють ефективно моделювати складні залежності в даних і показують добрі результати в задачах прогнозування часових рядів.

Також було спроєктовано структуру вебресурсу та бази даних для цього вебресурсу.

Для реалізації нейромережевого підходу було розроблено вебсервіс, який використовується для прогнозування трафіку на вебсайті. Цей сервіс може бути використаний власниками вебсайтів для покращення управління ресурсами та планування маркетингових кампаній.

Усі завдання було успішно виконано, що свідчить про досягнення поставленої мети кваліфікаційної роботи бакалавра.

ДЯКУЮ ЗА УВАГУ!

Wed Jun 12 12:15:24 EEST 2024, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. Помилок в документах: 17%

ID: 129945 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА на тему Метод прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу Додано в БД: 2024-06-12 Автора: Богдан Корчевський Керівники: Олександр БАРМАК Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	78525	1194	3703 (5%)	60 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КН

ID перевірки:
1016352209

Дата перевірки:
12.06.2024 20:51:09 MSK

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
12.06.2024 22:01:57 MSK

ID користувача:
100005671

Назва документа: КНс-21-1 Корчевський_ЗАПИСКА

Кількість сторінок: 72 Кількість слів: 12232 Кількість символів: 99715 Розмір файлу: 2.98 MB ID файлу: 1016156042

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

6.81%
Схожість

Найбільша схожість: 2.62% з джерелом з Бібліотеки (ID файлу: 1016145563)

4.86% Джерела з Інтернету | 587 ..... Сторінка 74

4.28% Джерела з Бібліотеки | 157 ..... Сторінка 78

0% Цитат

Включення цитат вимкнене

Включення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 1

Підозріле форматування | 11 сторінок

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Метод прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу*

Автор: *студент групи КН-20-1 Богдан КОРЧЕВСЬКИЙ*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *д.т.н., проф. Олександр БАРМАК*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі Корчевського Богдана Володимировича, не є плагіатом, оскільки: запозичення розміщені в розділі огляду існуючих підходів, не описують безпосередньо авторську роботу і не стосуються її результатів; усі запозичення фрагментарні; до запозичень входять фрагменти що не мають авторства і містять поширені конструкції; серед запозичень знаходяться загальновідомі терміни та скорочення.

Обсяг запозичень, визначений системами виявлення збігів/ ідентичності/схожості, складає:

- за системою Anti-Plagiarism – 2%;

- за системою Unicheck – 6.81 %.

Керівник роботи — *Олександр БАРМАК*

Гарант ОП — *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН — *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-20-1 Корчевський Богдан Володимирович*
за темою: *Метод прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу*

1. Актуальність обраної теми

Обрана тема є актуальним завданням у сучасному інтернет-просторі, де успішність та вплив вебсайтів на аудиторію значною мірою залежать від їх відвідуваності. Швидкий розвиток технологій і великий обсяг інформації створюють жорстку конкуренцію серед вебресурсів. Тому важливо розробити ефективний метод прогнозування відвідуваності вебсайтів, щоб підвищити їх ефективність та конкурентоспроможність.

2. Повнота розкриття мети та завдань роботи

В роботі детально описані основні аспекти, що стосуються прогнозування відвідуваності вебсайтів за допомогою нейромережевих підходів. Розглянуто актуальні методи і підходи, проаналізовано їх ефективність. Завдання роботи охоплюють весь цикл дослідження: від аналізу існуючих методів до розробки, тестування та впровадження запропонованої моделі. Такий підхід забезпечує комплексне та всебічне розкриття теми. Тому можна зробити висновок, що мета та завдання роботи розкриті в повній мірі.

3. Зміст кожного розділу роботи

Розділ 1: Аналіз методів, засобів та підходів до прогнозування відвідуваності вебсайтів

У цьому розділі проведено всебічний аналіз існуючих методів прогнозування відвідуваності вебсайтів. Розглянуто різні підходи, такі як статистичні методи, машинне навчання та нейромережеві підходи. Проведено порівняльний аналіз їх ефективності, переваг і недоліків, що дозволяє визначити найбільш підходящі методи для подальшого дослідження.

Розділ 2: Метод прогнозування відвідуваності вебсайтів нейромережевими засобами

Ррозділ присвячений розробці методу прогнозування відвідуваності вебсайтів із використанням нейромережевих підходів. Описано структуру та архітектуру нейронної мережі, вибір параметрів та алгоритмів навчання. Наведено обґрунтування вибору моделі та детально розглянуто процес її налаштування для досягнення максимальної точності прогнозування.

Розділ 3: Реалізація вебсервісу з методом прогнозування відвідуваності вебсайтів

У розділі описано процес реалізації вебсервісу, який використовує розроблений метод прогнозування. Детально розглянуто архітектуру системи, її компоненти та взаємодію між ними. Представлено результати тестування вебсервісу та оцінку його продуктивності. Описано процес розгортання та налаштування сервісу для забезпечення стабільної та ефективної роботи.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблений вебсервіс надає власникам вебсайтів інструмент для прогнозування відвідуваності, що сприяє більш ефективному плануванню та управлінню. Застосування розробленої системи може значно знизити витрати на маркетинг та підвищити ефективність рекламних кампаній. Інформаційна система демонструє високу ефективність у прогнозуванні відвідуваності вебсайтів завдяки використанню сучасних нейромережевих технологій.

5. Якість оформлення кваліфікаційної роботи бакалавра

Кваліфікаційна робота бакалавра оформлена відповідно до встановлених вимог та стандартів. Текст роботи структурований, поділений на логічно послідовні розділи, що сприяє легкому сприйняттю матеріалу. Використані таблиці, діаграми та ілюстрації допомагають наочно представити результати дослідження та підвищують зрозумілість викладеного матеріалу.

6. Недоліки кваліфікаційної роботи бакалавра

Хоча кваліфікаційна робота бакалавра демонструє високу якість виконання та ґрунтовність дослідження, вона не позбавлена певних недоліків. В роботі бракує детального опису можливих обмежень та потенційних ризиків, пов'язаних з використанням запропонованого методу в реальних умовах. Крім того, тестування розробленої системи проводилось на обмеженій вибірці даних, що може вплинути на загальну валідність результатів. Також у роботі можна було б більше уваги приділити питанням оптимізації продуктивності та ресурсомісткості запропонованого рішення.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка ***«відмінно»****.*

Рецензент Бердникова І. П.

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-20-1 Корчевського Богдана Володимировича*

за темою *Метод прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу*

1. Актуальність теми

У сучасному інтернет-просторі відвідуваність вебсайтів визначає їх успішність та вплив на аудиторію. Розвиток технологій та величезна кількість інформації роблять конкуренцію серед вебресурсів жорсткою. Тому актуальним стає питання розробки ефективного методу прогнозування відвідуваності вебсайтів з метою підвищення їх ефективності та конкурентоспроможності.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

За стандартом, а саме описом предметної області, об'єктами вивчення та діяльності є математичні, інформаційні, імітаційні моделі реальних явищ, об'єктів, систем і процесів та методи і технології отримання, зберігання, обробки, передачі та використання інформації. Метою роботи саме є покращення ефективності прогнозування відвідуваності вебсайтів за допомогою нейромережевого підходу у вигляді вебсервісу. При вирішенні поставленої задачі використано математичні моделі, методи та алгоритми розв'язання теоретичних і прикладних задач, що виникають при розробці вказаного методу. Тому результати виконання кваліфікаційної роботи бакалавра відповідають стандарту бакалавра спеціальності 122 – Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

При роботі над кваліфікаційною роботою бакалавра Корчевський Богдан Володимирович проявив себе кваліфікованим фахівцем та дисциплінованим студентом, вчасно виконуючи поставлені етапи дослідження. Як в процесі написання пояснювальної записки, так і при розробці прикладного програмного забезпечення проявив достатні для одержання успішного результату компетентності та результати навчання. Опанував професійні скіли за напрямком «Комп'ютерні науки» та достатньо значний софт скіл .

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Одержані в роботі результати є наслідком особистої діяльності студента, який самостійно виконував всі поставлені задачі.

5. Ступінь оволодіння методами дослідження

При реалізації кваліфікаційної роботи показав достатній рівень компетентностей та володіння необхідними інструментами та обладнанням, методами, методиками та технологіями предметної області комп'ютерних наук.

6. Повнота та якість розкриття теми роботи

Тема роботи в повній мірі обґрунтована й розкрита, проведено аналіз актуальності та відомих досліджень в межах обраної теми, поставлені завдання, які у роботі виконані, та розроблено програмне забезпечення для валідації та верифікації запропонованого методу.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблений у роботі метод та його програмна реалізація може бути використана власниками вебсайтів для покращення управління ресурсами та планування маркетингових кампаній.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

*Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «**відмінно**».*

Керівник _____ *д.т.н., проф. зав. каф. КН Олександр БАРМАК*