

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПРОГНОЗУВАННЯ ЗАБРУДНЕННЯ АТМОСФЕРНОГО ПОВІТРЯ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS
FOR FORECASTING THE AIR POLLUTION

Виконав: студент(ка) 2 курсу, групи КНТ-129м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Лукашенко Ю.О.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет Інститут інформатики та радіоелектроніки,
факультет комп'ютерних наук і технологій
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2020 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Лукашенка Юрія Олексійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів
прогнозування забруднення атмосферного повітря.
Research and Software Implementation of Methods for Forecasting the Air Pollution.

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “3” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) грудень 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження.
2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні
рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та
експериментальне дослідження програми. 6. Економіко-організаційна частина.
7. Охорона праці та безпека у надзвичайних ситуаціях.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Льовкін, В.М., доцент		
Організаційно-економічна частина	Пожуєва Т.О., професор		
Охорона праці та цивільна безпека	Коробко О.В., ст. викладач		
Нормоконтроль	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “4” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз та дослідження предметної області	1-2 тиждень	
2	Розробка та затвердження технічного завдання	3 тиждень	Технічне завдання
3	Дослідження методів розв’язання проблеми	4 тиждень	
4	Проектування структури програмного забезпечення	5 тиждень	Структурна схема
5	Розроблення схеми функціонування програмного забезпечення та модулів	6-7 тиждень	Функціональна схема. Модулі
6	Розроблення програмного коду	8 тиждень	Текст програми
7	Тестування та відлагодження програмного забезпечення	9 тиждень	Програма
8	Розробка розділів пояснювальної записки	10-11 тиждень	Пояснювальна записка
9	Розробка графічного матеріалу	12 тиждень	Графічний матеріал
10	Захист магістерської роботи	13 тиждень	

Студент

(підпис) Лукашенко Ю.О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Льовкін В.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
151 с., 9 табл., 51 рис., 3 дод., 53 джерела.

ГІПЕРПАРАМЕТРИ, ДОВГА КОРОТКОСТРОКОВА ПАМ'ЯТЬ, ЗАБРУДНЕННЯ ПОВІТРЯ, ІНДЕКС ЯКОСТІ ПОВІТРЯ, МАШИННЕ НАВЧАННЯ, МІКРОСЕРВІС, ПРОГНОЗУВАННЯ, РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ, ШТУЧНИЙ ІНТЕЛЕКТ, ANGULAR, C#, PYTHON.

Об'єкт дослідження – процес прогнозування забруднення атмосферного повітря.

Предмет дослідження – методи прогнозування забруднення атмосферного повітря.

Мета роботи – розроблення програмного забезпечення для прогнозування забруднення атмосферного повітря на основі досліджених методів заради зменшення ймовірності перебування людини у потенційно небезпечному для здоров'я середовищі.

Матеріали, методи та технічні засоби: рекурентні нейронні мережі типу «довга короткострокова пам'ять» та їхні модифікації з гіперпараметрами, бібліотеки машинного навчання Keras, SkLearn, TensorFlow, мови програмування Python, C#, TypeScript, фреймворки ASP.NET Core, Angular з використанням HTML та CSS, Entity Framework Core, а також Microsoft SQL Server 2019 та інтегровані середовища розробки Visual Studio 2019 та PyCharm.

Результати: розроблено програмну систему, яка складається з програми, що реалізує запропонований метод прогнозування та вебзастосунку.

Висновки: удосконалено алгоритм прогнозування, що дозволяє підвищити точність прогнозування та розроблено комплекс програм.

Галузь використання: прогнозування індексу якості повітря для покращення екологічного становища міста.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 151 pages, 51 figures, 9 tables, 3 appendixes, 53 sources.

AIR POLLUTION, AIR QUALITY INDEX, ANGULAR, ARTIRICIAL INTELLIGENCE, C#, FORECASTING, HYPERPARAMETERS, LONG SHORT-TERM MEMORY, MACHINE LEARNING, MICROSERVICE, PYTHON, RECURRENT NEURAL NETWORK.

The object of study is the process of forecasting air pollution.

The subject of the study is the methods of forecasting air pollution.

The purpose of the work is to develop software for forecasting air pollution based on the studied methods to reduce the likelihood of a person being in a potentially hazardous environment.

Materials, methods, and technical tools: recurrent neural networks like "long short-term memory" and their modifications with hyperparameters, machine learning libraries Keras, SkLearn, TensorFlow, programming languages Python, C#, TypeScript, frameworks ASP.NET Core, Angular with using HTML and CSS, Entity Framework Core, also Microsoft SQL Server 2019 and, integrated development environments Visual Studio 2019 and PyCharm.

Results: a software system is developed, which consists of a program that implements the proposed method of forecasting and web application.

Conclusions: the forecasting algorithm has been improved, which allows to increase the accuracy of forecasting and a set of programs has been developed.

The field of use: forecasting the air quality index to improve the ecological situation of the city.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	10
Вступ.....	11
1 Аналіз проблеми та постановка задач дослідження	13
1.1 Моделювання якості повітряного середовища. Поняття часового ряду ...	13
1.2 Види моделей для прогнозування.....	14
1.2.1 Емпіричні моделі	14
1.2.2 Статистичні моделі.....	15
1.2.3 Нейронні моделі	16
1.3 Принципи моделювання на основі машинного навчання та нейронних мереж	17
1.3.1 Алгоритм дій для задачі машинного навчання	17
1.3.2 Основні складності машинного навчання	18
1.3.3 Логіка нейронної мережі. Будова перцептрону	18
1.4 Аналіз вимог предметної області.....	21
1.4.1 Індекс якості повітря.....	21
1.4.2 Робочий набір даних. План прогнозування.....	22
1.5 Огляд існуючих програмних продуктів, що вирішують аналогічні або схожі задачі	23
1.5.1 Сайт BreezoMeter.....	23
1.5.2 Додаток AirVisual.....	24
1.5.3 Сайти проєкту «The World Air Quality Index».....	25
1.6 Постановка завдання.....	27
1.7 Висновки за розділом 1	28
2 Матеріали та методи.....	29
2.1 Аналіз методів.....	29
2.2 Рекурентні нейронні мережі	30
2.3 Архітектурні особливості мережі LSTM	32
2.3.1 Загальна схема LSTM комірки.....	32

2.3.2 Фільтр забування	33
2.3.3 Фільтр запам'ятовування.....	34
2.3.4 Фільтр актуальних знань	36
2.3.5 Фільтр висновків	37
2.4 Шляхи покращення моделі для прогнозування якості повітря	38
2.4.1 Застосування функцій активації	38
2.4.2 Налаштування гіперпараметрів моделі	39
2.4.3 Додавання шарів та кількості нейронів у шарі	40
2.4.4 Пошук кореляції з трафіком руху автомобілів	42
2.4.5 Передбачення майбутніх значень з використанням look_back	43
2.5 Запропонований метод прогнозування	43
2.5.1 Етап 1. Формування вибірки.....	44
2.5.2 Етап 2. Створення моделі	44
2.5.3 Етап 3. Навчання моделі.....	45
2.5.4 Етап 4. Тестування моделі.....	46
2.5.5 Етап 5. Прогнозування для майбутніх значень.....	46
2.6 Висновки за розділом 2.....	47
3 Проєктування програмного забезпечення	48
3.1 Функціональні вимоги до програмного забезпечення	48
3.2 Структура програмного забезпечення. Мікросервіси	49
3.3 Моделювання сценарію використання програми	50
3.4 Вибір засобів реалізації програмного забезпечення.....	51
3.4.1 Вибір мов програмування	51
3.4.2 Вибір фреймворків	53
3.4.3 Вибір інтегрованих середовищ розробки	54
3.5 Проєктування бази даних	55
3.5.1 Схема бази даних	56
3.5.2 Підхід до взаємодії бази даних та мікросервісів	57
3.6 Вимоги до технічних засобів	57
3.7 Висновки за розділом 3	58

4 Основні рішення щодо реалізації компонентів системи.....	59
4.1 Алгоритм функціонування системи	59
4.2 Модуль обробки даних та прогнозування	60
4.2.1 Компонент попередньої обробки даних	60
4.2.2 Компонент імплементації моделі та прогнозування	61
4.2.3 Компонент збереження результатів	62
4.3 Модуль серверної частини	62
4.3.1 Підпроект AirForecasting.Core	62
4.3.2 Сервіси та програмні інтерфейси	63
4.3.3 Контролери	63
4.4 Модуль графічного інтерфейсу	64
4.4.1 Модулі Angular	64
4.4.2 Компоненти Angular	64
4.4.3 Сервіси Angular	65
4.5 Висновки за розділом 4	66
5 Експлуатація, тестування та експериментальне дослідження програми	67
5.1 Опис застосування програми	67
5.2 Інструкція по запуску та експлуатації програмної системи	67
5.3 Методика роботи з програмою	68
5.3.1 Результати ПЗ для прогнозування.....	68
5.3.2 Взаємодія з вебсервісом	69
5.4 Методика тестування системи	71
5.4.1 Тестування моделі.....	72
5.4.2 Тестування серверної частини	76
5.5 Висновки за розділом 5	76
6 Економіко-організаційна частина.....	78
6.1 Концепція економічного обґрунтування	78
6.2 Планування розробки програмного продукту.....	79
6.3 Визначення витрат на розробку.....	82
6.4 Розрахунок основної заробітної плати.....	82

6.5 Розрахунок додаткової заробітної плати	82
6.6 Відрахування на єдиний соціальний внесок	83
6.7 Визначення витрат на матеріали	83
6.8 Витрати на спеціальне обладнання	84
6.9 Інші прямі витрати	88
6.10 Розрахунок накладних витрат	89
6.11 Розрахунок економічної ефективності програмного продукту	90
7 Охорона праці та безпека у надзвичайних ситуаціях	93
7.1 Аналіз потенційних небезпек	93
7.2 Заходи із забезпечення безпеки	94
7.3 Заходи із забезпечення виробничої санітарії та гігієни праці	97
7.4 Заходи безпеки у надзвичайних ситуаціях	102
7.4.1 Заходи з пожежної безпеки	102
7.4.2 Заходи з цивільного захисту	105
Висновки	109
Перелік джерел посилання	111
Додаток А Технічне завдання	117
Додаток Б Текст програми	124
Додаток В Слайди презентації	124

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – application programming interface;
ARIMA – autoregressive integrated moving average;
LSTM – long short-term memory;
REST – representational state transfer;
RNN – recurrent neural network;
ДКЧП – довга короткочасна пам'ять;
ЄІЯП – Європейський індекс якості повітря;
ССВ – єдиний соціальний внесок;
ІЯП – індекс якості повітря;
МН – машинне навчання;
ПЗ – програмне забезпечення;
РіНР – рятувальні та невідкладні роботи;
РНМ – рекурентна нейронна мережа;
СКБД – система керування базами даних;
ШН – штучний нейрон;
ШНМ – штучні нейронні мережі.

ВСТУП

У сучасному світі значним темпом зростає урбанізація. Більше 50% населення живуть у великих містах. Велике місто функціонує, як велика піч, викидаючи в атмосферу велику кількість тепла, речовин і відходів, що забруднюють повітря. У результаті токсичні гази спричиняють шкідливий вплив на здоров'я людей. Зокрема, проблема повітря низької якості особливо жорстко постає у місті Запоріжжя, що є одним з індустріальних центрів України із значною кількістю заводів, що продукують відходи не тільки у повітря, а й у річки та й загалом у навколишнє середовище.

Розв'язання проблем забруднення має бути комплексним. Якщо постає завдання покращити якість атмосферного повітря у певному місті, то необхідно виконати такі дії:

- розподілити місто на рівномірні райони, в яких встановити станції, які будуть встановлювати якість повітря в реальному часі;
- має бути проведений аналіз хімічного складу повітря у різних за типами районах міста;
- на об'єктах, які є критичними, мають бути встановлені відповідні очисні споруди.

Однак причинами забруднення можуть бути різноманітні фактори, тож їхнім аналізом має займатися ціла низка організацій. Основна мета на початковій стадії – це створення комплексної тривимірної високощільної міської мережі моніторингу для вимірювання як метеорологічних параметрів, так і параметрів хімічного складу атмосфери. Така інтенсивна програма вимірювання метеорологічних і хімічних параметрів забезпечує більш якісне уявлення про розподіли динамічних і хімічних компонентів [1].

Далі, маючи актуальні дані про склад атмосферного повітря, необхідно створити системи прогнозування та оповіщення (раннього попередження). Для цих систем потрібні різні виміри, методи (такі як статистичні та діагностичні) і моделі, включаючи метеорологічні та хімічні моделі [1].

З точки зору програмних систем, маючи набір даних про попередній період часу, можна створити систему прогнозування, мета якої – прогнозувати рівень основних хімічних елементів та сполук в залежності від їх значень у минулому. Маючи таку систему як один з інструментів, державні метеорологічні підрозділи зможуть поширювати сповіщення про поточний та майбутній стан атмосферного повітря у автономному режимі за допомогою мережі Інтернет.

Прогнозування за попередніми інтервалами передбачає роботу з часовим рядом та його аналіз. У даній магістерській роботі розглянуті методи прогнозування з використанням нейронних мереж, комбінування різних вхідних даних та мережевих архітектур для досягнення максимально точного прогнозу рівня забрудненості атмосферного повітря, а також імплементація такої системи.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Моделювання якості повітряного середовища. Поняття часового ряду

В даний час для вивчення властивостей складних систем, в тому числі і при експериментальних дослідженнях, широко використовується підхід, заснований на аналізі сигналів, вироблених системою. Це дуже актуально в тих випадках, коли математично описати досліджуваний процес практично неможливо, але у розпорядженні дослідників є деяка характерна величина або декілька величин, що піддаються дослідженню. Тому аналіз систем, особливо при експериментальних дослідженнях, часто реалізується за допомогою обробки реєстрованих сигналів [2].

Зазвичай такий сигнал називається спостережуваним, а метод дослідження – реконструкцією динамічних систем [2]. Цей розділ теорії динамічних систем називається аналізом часових рядів. Прогнозування майбутніх значень якості атмосферного повітря є також однією з областей, у якій можна застосовувати такий аналіз. Загалом часовим рядом називається масив з N чисел, який представляє собою значення деякої спостережуваної величини з деяким постійним кроком t за часом на основі формули (1.1) [2]:

$$t_i = t_0 + (i - 1) \cdot t; x_i = x(t_i), i = 1, \dots, N, \quad (1.1)$$

де t_i – i -те значення часового інтервалу;

t_0 – початкове значення часового інтервалу;

$x(t_i)$ – функція для спостережуваної величини для t_i ;

x_i – значення i -тої спостережуваної величини;

N – кількість часових інтервалів.

У програмних системах часові ряди частіше за все представлені списками пар «ключ-значення», де ключем є i -тий часовий інтервал, а значенням – значення спостережуваної величини.

1.2 Види моделей для прогнозування

Прогнозування – це визначення майбутніх можливих параметрів об'єкта, явища (наприклад рівня забруднення навколишнього середовища у промисловій області) при заданих вхідних даних (швидкість вітру, його напрямку, потужності викиду джерела тощо) [3].

Прогнози в галузі охорони навколишнього середовища з погляду часового чинника можуть бути:

- короткостроковими – це прогнози на декілька годин, діб. Зокрема, прогноз рівня забруднення атмосферного повітря на конкретних вулицях на наступний день;

- довгостроковими – це прогнози на декілька років або десятиліть [3].

В залежності від поставленої задачі підходи до її рішення будуть кардинально відрізнятися. Розглянемо моделі в контексті моделювання забруднення атмосферного повітря, за допомогою яких можна виконувати його прогнозування.

1.2.1 Емпіричні моделі

Емпіричні моделі мають вигляд досить простих алгебраїчних відношень з яких визначаються шукані величини, наприклад концентрація забруднення. Прикладом моделей даного класу є методика ОНД-86, яка використовується для прогнозування рівня забруднення атмосферного повітря в Україні [3].

Одним з методів емпіричних моделей є метод емпіричної модової декомпозиції, суть якої полягає в розкладанні часового ряду на прості складові – так звані власні характерні форми [4].

Безпосередньо використання емпіричних моделей є достатньо простим підходом, однак вони не враховують фізичний зміст та не можуть дати узагальнення при зміні вхідних даних [3]-[4].

1.2.2 Статистичні моделі

Статистичні моделі – прогнознi моделі, побудовані на базі оброблення результатів спостережень за допомогою статистичних методів. Для побудови цих моделей потрібна значна кількість даних вимірювань, що потребує багато часу на одержання цих даних та їх оброблення [3].

Серед статистичних моделей вирізняються регресійні та авторегресійні моделі. Регресійні моделі бувають такі [5]:

- проста лінійна регресія;
- багатовимірна регресія;
- нелінійна регресія.

Найвідомішою з авторегресійних моделей є модель ARIMA – інтегрована модель авторегресії рухомого середнього [6].

Якщо за допомогою лінійної регресії можна знайти одне прогнозоване значення за моделлю, то авторегресійна модель допомагає знайти декілька таких значень. Зокрема, для лінійної регресії має місце така формула (1.2) [3, 7]:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n, \quad (1.2)$$

де $\hat{y}$ – прогнозоване значення,

θ_0 – константа регресії;

n – кількість регресорів (факторів моделі);

x_i – значення i -того регресору;

θ_j – значення j -того параметру регресії.

Суть авторегресії полягає у використанні результатів попередніх спостережень як вхідних даних для рівняння регресії на наступних кроках. Тому для авторегресії формула (1.2) трансформується у формулу (1.3):

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \dots + \phi_p X_{t-p} + Z_t, \quad (1.3)$$

де X_t – t -тий прогноз моделі;

p – кількість виразів $\phi_1 X_{t-1}$, які корелюють з поточним значенням моделі;

ϕ_j – значення j -того параметру регресії;

X_i – значення i -того прогнозованого значення;

Z_t – величина білого шуму моделі.

Загалом, розглядаючи модель у контексті прогнозування забруднення атмосферного повітря, такі моделі призначені для короткострокових прогнозів на декілька годин або діб.

1.2.3 Нейронні моделі

Нейронні моделі складаються з нейронних мереж. Нейронні мережі – це розділ штучного інтелекту, в якому для обробки сигналів використовуються інструменти, аналогічні явищам, що відбуваються в нейронах живих істот [8].

Найважливіша особливість мережі, яка свідчить про її широкі можливості і величезний потенціал, полягає в паралельній обробці інформації усіма ланками, що дозволяє значно прискорити процес обробки інформації. Крім того, при великому числі міжнейронних з'єднань мережа набуває стійкість до помилок, які виникають на деяких лініях [8].

Інша не менш важлива властивість – здатність до навчання і узагальнення накопичених знань. Нейронна мережа має риси штучного інтелекту. Натренована на обмеженій множині даних мережа здатна узагальнювати отриману інформацію і показувати хороші результати на даних, що не використовувалися при її навчанні [8]-[9].

Тож для імплементації поставленої задачі було обрано моделювання з використанням нейронних мереж, сутність та підходи якого детальніше будуть розглянуті далі.

1.3 Принципи моделювання на основі машинного навчання та нейронних мереж

Машинне навчання (МН) – це галузь дослідження, яка дає комп’ютерам можливість вчитися на даних без явного програмування. У випадку машинного навчання кажуть, що комп’ютерна програма вчиться на досвіді щодо певного завдання і деякого показника ефективності, якщо її ефективність на виконання завдання покращується із досвідом [7].

1.3.1 Алгоритм дій для задачі машинного навчання

Загалом, будь-яка задача, що пов’язана з МН, не залежно від наявності чи відсутності нейронних мереж як інструменту її вирішення, у загальному випадку поділяється на такі етапи [7]:

- сформулювати проблему і подивитися на загальну картину. Необхідно декомпонувати задачу у рамках бізнес-правил, виявити тип задачі та інструменти її вирішення;
- отримати дані. Зрозуміти, які дані потрібні, яким способом їх отримати. Перетворити дані у зручний формат, при цьому їх не змінюючи;
- дослідити дані, щоб отримати статистику. Вивчити атрибути, візуалізувати дані, спробувати знайти кореляцію між атрибутами;
- підготувати дані, щоб краще представити основні шаблони даних алгоритмам машинного навчання. Видалити несуттєві дані, заповнити пропуски у даних, за необхідності нормалізувати дані, виділити тестову вибірку;
- вивчити низку різних моделей та скласти список найкращих з них. Можна почати аналіз з вибірки менших розмірів, тренувати моделі з різних категорій, перевіряючи їхню результативність, точність, похибку;
- точно налаштувати свої моделі та об’єднати їх у остаточне рішення;
- провести тестування. За необхідності внести корективи у моделі;
- підтримувати робочу систему, регулювати моделі за новими даними.

1.3.2 Основні складності машинного навчання

Оскільки головне завдання МН – вибрати алгоритм навчання та навчити його на деяких даних, дві речі, які можуть піти не так, це «поганий алгоритм» та «погані дані» [7].

Проблеми, пов’язані з неякісними даними [7]:

- недостатня кількість навчальних даних. Навіть для дуже простих проблем для навчання необхідні тисячі прикладів;
- нерепрезентативні навчальні дані. Результуюча модель буде мати низьке узагальнення;
- дані низької якості. Очевидно, що якщо ваші навчальні дані сповнені помилок, відхилень та шуму (наприклад, через низьку якість вимірювання), це ускладнить системі виявлення основного шаблону;
- нерелевантні властивості. Вхідні дані можуть містити багато параметрів, з яких деякі можуть бути несуттєвими для фінального результату.

Проблеми, пов’язані з неякісними алгоритмами [7], [10]:

- перенавчання моделі. Хоч для навчальної вибірки кількість похибок буде мінімальною, однак така модель буде погано передбачати значення з тестової вибірки, оскільки надто реагує на шуми та відхилення тренувальних даних. Зокрема, перенавчена модель може спиратися на зв’язки, які насправді не матимуть жодного значення;
- недостатнє навчання. Надто проста модель також може бути не досить потужним інструментом для прогнозування тестових даних.

1.3.3 Логіка нейронної мережі. Будова перцептрону

Нейронна мережа у загальному вигляді складається з нейронів та зв’язків між вхідними сигналами та нейронами, та вихідних сигналів.

Найпростіша модель, так званий, «штучний нейрон» (ШН), складається з одного або більше двійкових входів та одного бінарного виходу [7]. На вхід

подаються сигнали, а нейрон містить власну функцію активації, яка вирішує, яким чином реагувати на отримані сигнали. Наприклад, на рис. 1.1 зображені штучні нейронні мережі (ШНМ), які виконують прості логічні обчислення. У них функцією активації є проста формула, яка дає 1, якщо кількість сигналів більше або дорівнює двом, і 0 у всіх інших випадках.

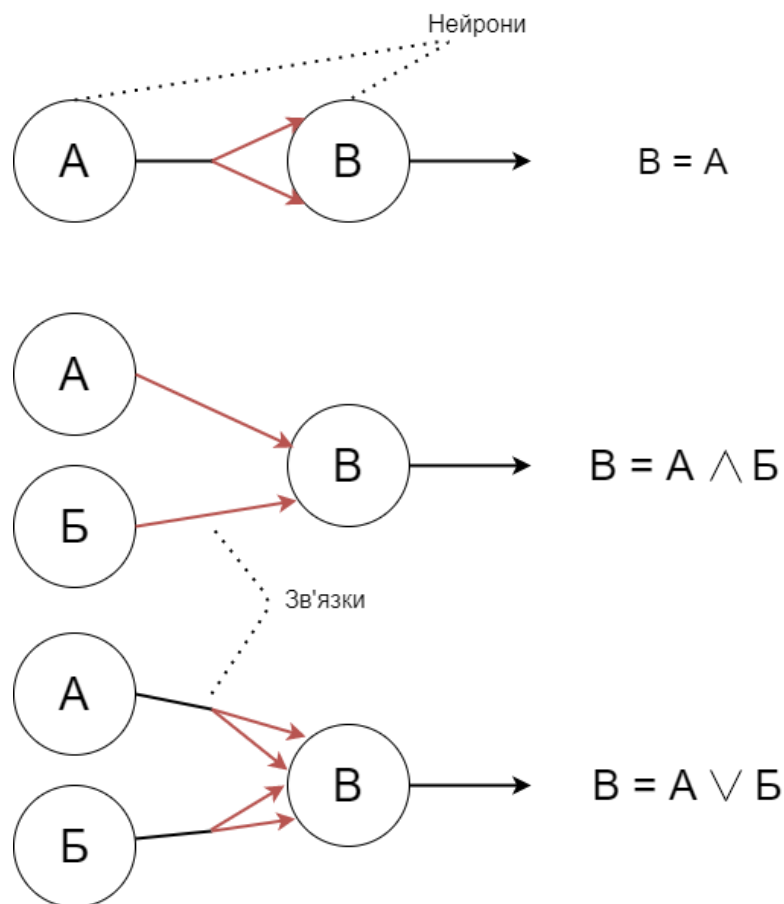


Рисунок 1.1 – ШНМ, що виконують прості логічні обчислення

Одним з базових архітектур ШНМ, є перцептрон. Його сутність полягає у вигляді деякої функції в залежності від суми зважених вхідних сигналів [7]. Одним з часткових випадків формули суми добутків, якщо сума вагів разом складають 1, є формула математичного очікування. Однак для перцептрона задача налаштування моделі, пошук оптимальних вагів і є однією з найважливіших і найскладніших. Розглянемо формулу (1.4) детальніше:

$$y = f\left(\sum_{i=1}^n w_i x_i\right), \quad (1.4)$$

де y – вихідне значення;

f – функція активації;

x_i – значення i -того входу;

w_i – значення i -тої ваги;

n – кількість входів.

Схема перцептрону зображена на рис 1.2.

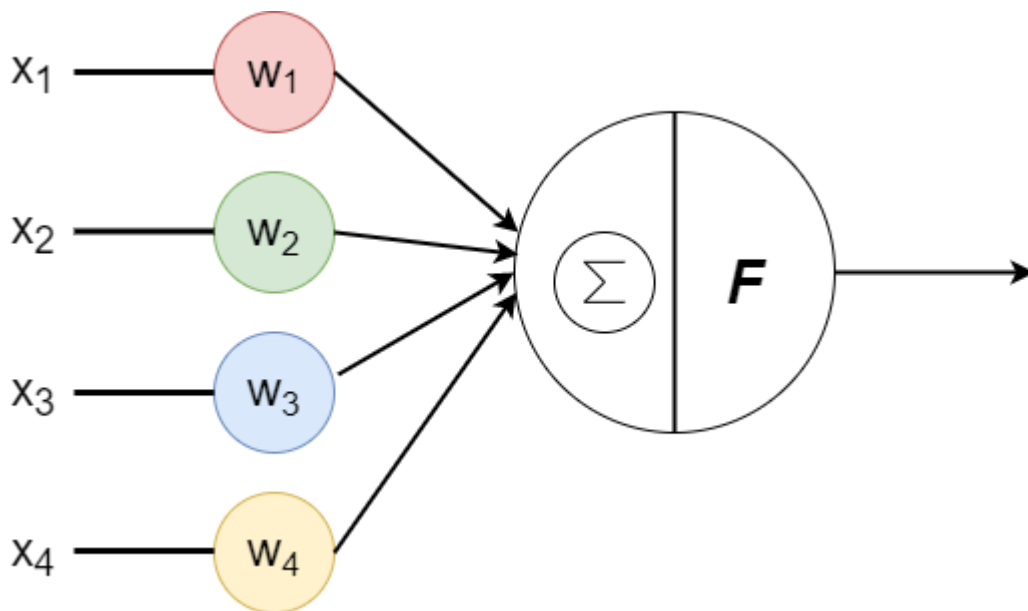


Рисунок 1.2 – Схема перцептрону

В залежності від складності системи, сума може перетворюватися різними функціями активації. Наприклад, однією з найпростіших є функція Гевісайда [7], яка поверне 1, якщо сума більше 0, та 0, якщо менше або дорівнює 0. Загалом ця функція використовується нечасто, оскільки має певні недоліки пов'язані з різким стрибком. Замість неї використовують сигмоїди, лінійні, ReLu, tanh та інші активаційні функції [7]-[11].

Нейрони можуть утворювати нейронний шар. Таких шарів може бути декілька. У такому випадку внутрішній шар називається прихованим, а

нейронна мережа – багатошаровим перцептроном. Наявність декількох прихованих шарів дає змогу віднести навчання з такими типами мереж до глибокого машинного навчання.

1.4 Аналіз вимог предметної області

1.4.1 Індекс якості повітря

Європейський індекс якості повітря (ЄІЯП) відповідає визначенню Європейського агентства з охорони навколишнього середовища (ЕЕА, eea.europa.eu). Індекс розраховується по п'яти основних забруднюючих речовин, які регулюються Європейським законодавством: O_3 (озон), NO_2 (діоксид азоту), SO_2 (діоксид сірки), $PM_{2,5}$ і PM_{10} (зважені частинки діаметром менше 2,5 мікрметрів і 10 мікрметрів відповідно). Для кожної забруднюючої речовини, значення індексу варіюється від 1 (добре) до 5 (дуже погано). ЄІЯП розраховується за різними забруднюючими речовинами окремо відповідно до концентрацій (моментально або в середньому за день, в залежності від забруднюючої речовини): чим вище концентрація, тим вище індекс. ЄІЯП представлений цілим числом, відповідним п'яти діапазонах концентрацій, характерним для кожної забруднюючої речовини (табл. 1.1) [12].

Таблиця 1.1 – Забруднювачі та їх рівні [12]

Забруднювач	Рівень індексу (на основі концентрації забруднюючих речовин в мкг/м ³)				
	1 – дуже гарний	2 – гарний	3 – середній	4 – поганий	5 – дуже поганий
Озон (O_3)	0-80	80-120	120-180	180-240	240-600
Двоокис азоту (NO_2)	0-40	40-100	100-200	200-400	400-1000
Діоксид сірки (SO_2)	0-100	100-200	200-350	350-500	500-1250
Частинки менше 10 мкм (PM_{10})	0-20	20-35	35-50	50-100	100-1200
Частинки менше 2,5 мкм ($PM_{2,5}$)	0-10	10-20	20-25	25-50	50-800

Загальний погодинний ЄІЯП визначається як найбільше значення з п'яти індивідуальних індексів забруднюючих речовин, розрахованих за одну й ту ж годину. Для прикладу, якщо індекси O₃, NO₂, SO₂, PM_{2,5} і PM₁₀ рівні 1, 3, 1, 2, 2 відповідно, середній індекс буде дорівнює трьом. Відповідно, загальний щоденний ЄІЯП – це найбільше значення загального погодинного ЄІЯП за відповідний день [12].

1.4.2 Робочий набір даних. План прогнозування

Для прогнозування якості атмосферного повітря було обрано набір даних з величинами шкідливих домішок на станціях м. Мадрид з 2001 по 2018 роки. Фіксування значень відбувалось погодинно [13]. Серед усіх інших наборів був обраний саме цей, оскільки він був розроблений для зручності використання із використанням загальноприйнятих стандартів та продуктивних форматів. Він представлений у форматах .csv за роками та .h5, які рекомендовані для проведення машинного навчання.

Дані у форматі .csv сформовані дуже зручно для розуміння людиною: у кожному рядку перелічені дата, низка забруднювачів та відповідна станція. Кожний .csv файл містить дані за один рік. Усі дані можна було б об'єднати у один файл, але тоді він був би не оптимізований для пошуку, а також містив би багато повторень. Спеціально для таких цілей був створений формат .h5, де інформація зберігається у ієрархічному вигляді, за рахунок чого дані займають менше місця та швидше оброблюються комп'ютером.

Даний набір даних був сформований розробником Diego Vicente, який у [14] розповідає, яким чином з сирих даних з датчиків його команді вдалось створити датасет, що підлягає прогнозуванню.

Даний набір даних містить усі необхідні для ЄІЯП вимірювання рівня забруднювачів з табл. 1.1, а також низку інших забруднювачів.

Згідно з пп. 1.3.1 та 1.3.2, цей датасет необхідно проаналізувати та покращити для цілей прогнозування.

Отже, оскільки у загальному вигляді необхідно дослідити методи та спрогнозувати рівень забрудненості атмосферного повітря, то необхідно:

- взяти дані з датасету, спрогнозувати значення для кожної станції окремо;
- для основного прогнозу взяти лише ключові забруднювачі з табл. 1.1;
- виявити відсутні дані та/або викиди в даних;
- нормалізувати дані, аби прогнозування виконувалось в одній розмірності;
- оскільки маємо справу з часовим рядом, його необхідно розподілити на тренувальну та тестову вибірки за часом;
- маємо задачу навчання зі вчителем, отже нехай непарні 24 години будуть тим, на основі чого будемо прогнозувати, а парні 24 години, тим, що необхідно буде прогнозувати;
- необхідно проаналізувати, яка з архітектур нейронних мереж матиме найбільш точний результат, та провести експерименти з їхньою конфігурацією;
- дати прогноз для кожного з ключових забруднювачів на два тижні вперед;
- поєднати прогноз ключових забруднювачів та вивести єдине значення ЄІАП на два тижні вперед (погодинно) для кожної станції.

1.5 Огляд існуючих програмних продуктів, що вирішують аналогічні або схожі задачі

1.5.1 Сайт BreezoMeter

Сервіс BreezoMeter надає інформацію про якість повітря в режимі реального часу в межах заданої геолокації [15]. BreezoMeter володіє низкою API, пов'язаних з якістю повітря, однак не прогнозує якість повітря на майбутнє. Рівень забруднення, який показує сервіс, є достатньо точним (рис. 1.3).

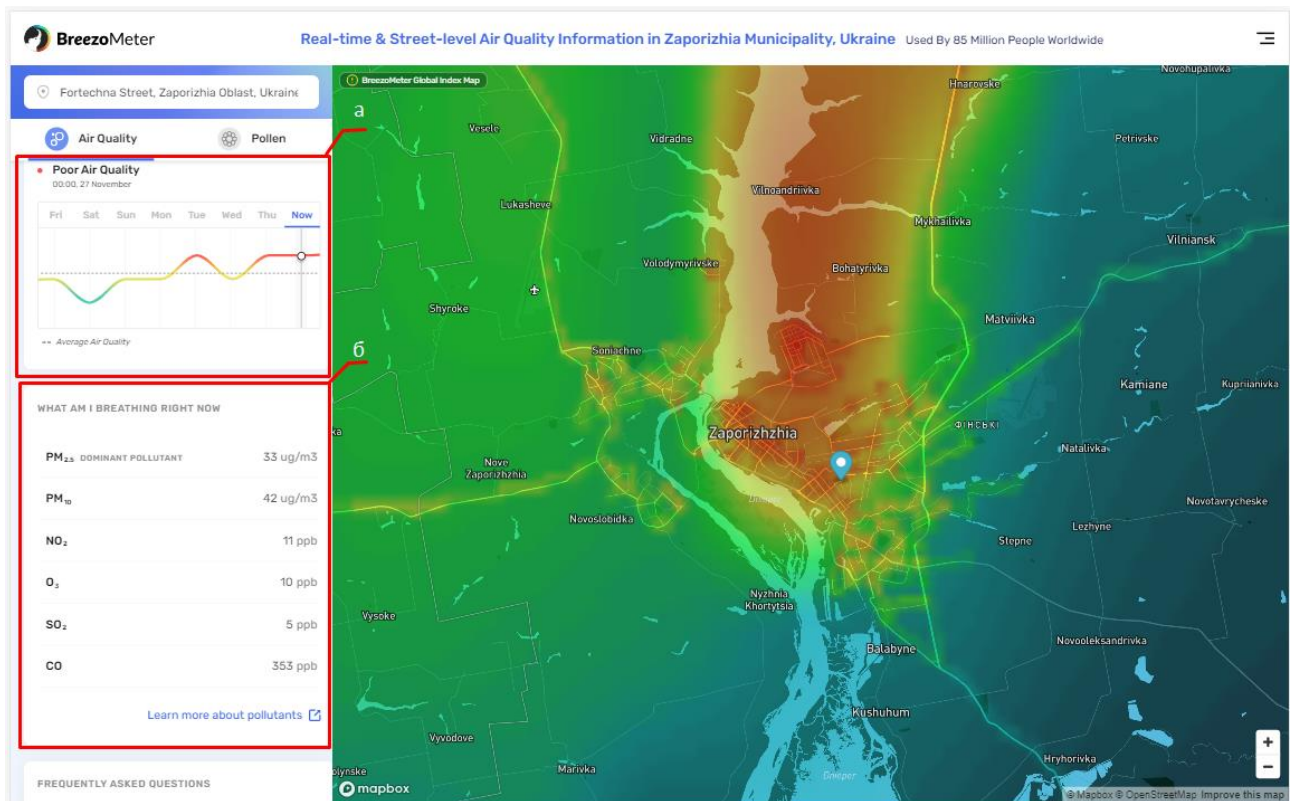


Рисунок 1.3 – Інтерфейс сайту BreezoMeter. Якість повітря у м.Запоріжжя:
а – тижневий графік забрудненості; б – рівень ключових забруднювачів

1.5.2 Додаток AirVisual

Додаток AirVisual забезпечує показники ІЯП для більшості країн, за якими можна стежити та налаштовувати сповіщення для інформування користувача, коли ІЯП занадто високий. Користувач може фільтрувати різні забруднювачі, щоб зрозуміти який забруднювач найбільше впливає на якість повітря. Додаток забезпечує інтерактивну карту, яка допомагає визначити найближчу до користувача станцію, а також додаток надає семиденний прогноз ІЯП (рис. 1.4). Він також дає рекомендації щодо охорони здоров'я, зокрема рекомендує закрити вікна та зменшити фізичні вправи на відкритому повітрі, коли спостерігається наявність шкідливих домішок [16].

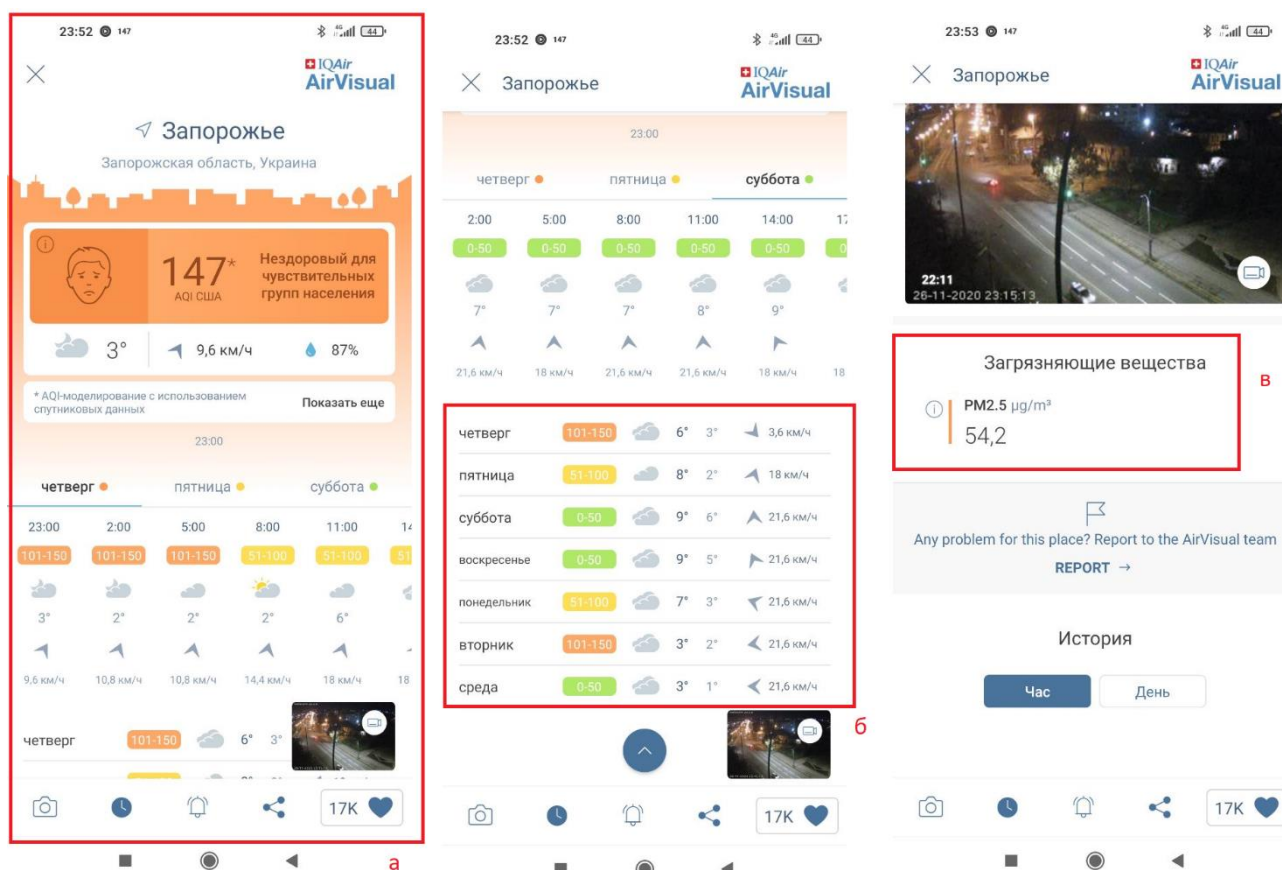


Рисунок 1.4 – Интерфейс додатку AirVisual. Якість повітря у м.Запоріжжя: а – поточне значення забрудненості; б – прогноз на тиждень; в – ключовий забруднювач

1.5.3 Сайти проєкту «The World Air Quality Index»

The World Air Quality Index – це некомерційний проєкт, місія якого – сприяти підвищенню рівня обізнаності громадян щодо забруднення повітря та надавати уніфіковану та загальносвітову інформацію про якість повітря.

Проект забезпечує прозору інформацію про якість повітря для більш ніж 130 країн, охоплюючи понад 30 000 станцій у 2000 великих містах, через ці два веб-сайти: aqicn.org та waqi.info [17].

Сайт waqi.info представляє собою карту світу із актуальними значеннями індексу якості повітря у реальному часі (рис. 1.5).

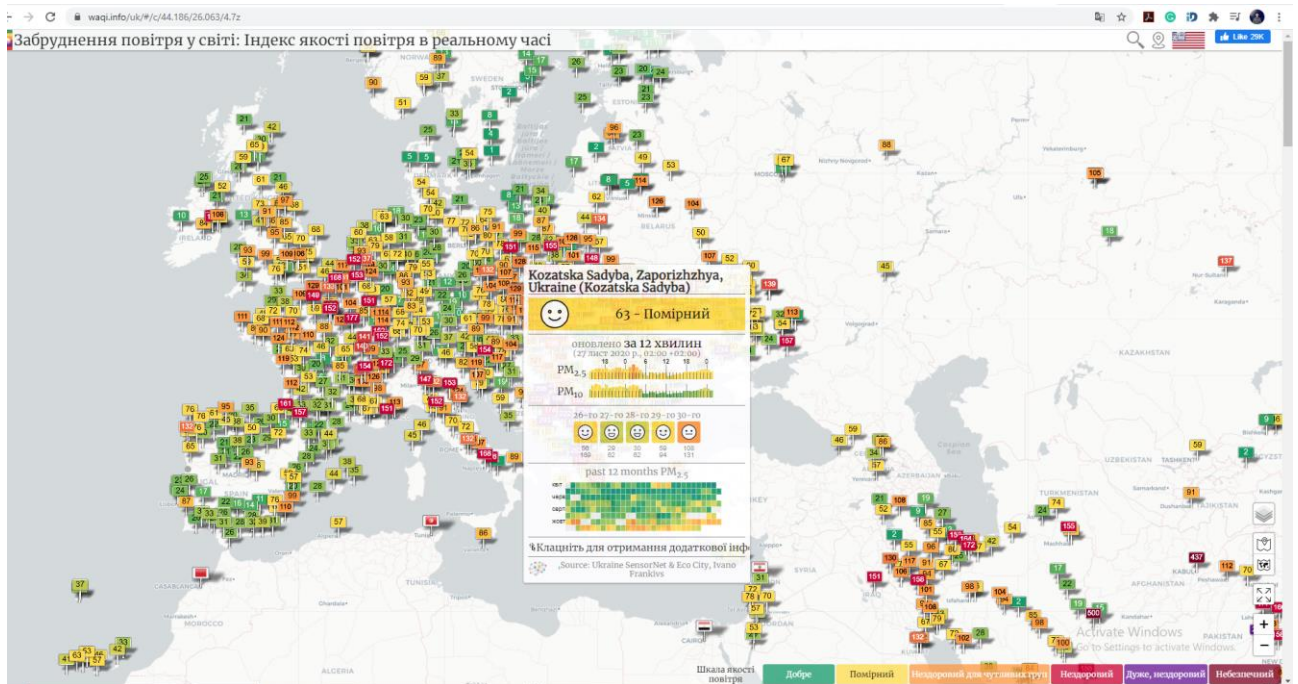


Рисунок 1.5 – Інтерфейс сайту waqi.info: індекс якості повітря у реальному часі

Сайт aqicn.org надає більш детальну інформацію про стан повітря у конкретній геолокації (рис. 1.6).

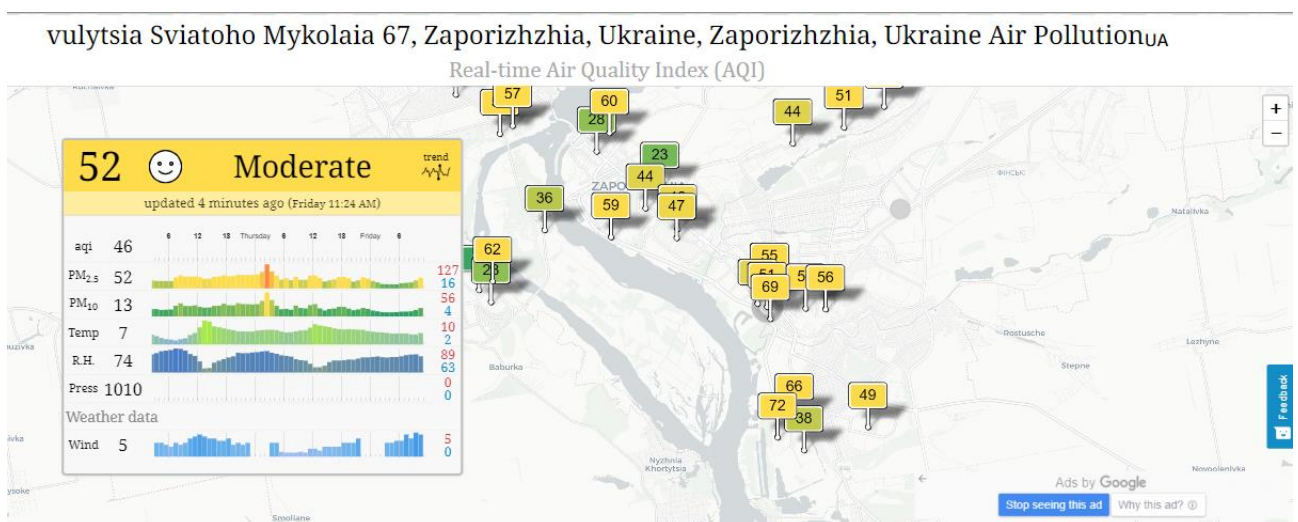


Рисунок 1.6 – Інтерфейс сайту aqicn.org: якість повітря по вул. Святого Миколая, м. Запоріжжя

Крім того, aqicn.org надає прогноз якості повітря на три дні вперед (рис. 1.7).

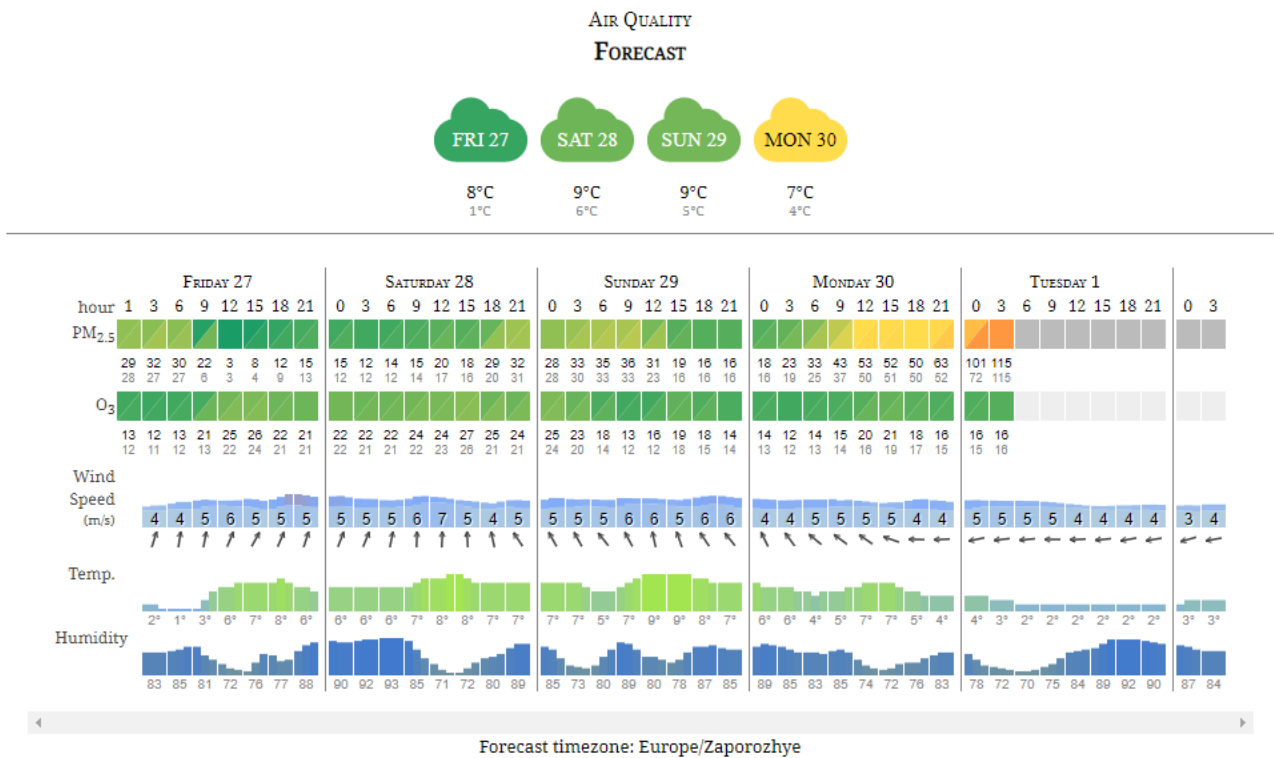


Рисунок 1.7 – Інтерфейс сайту aqicn.org: прогнозування якості повітря

1.6 Постановка завдання

Завдання магістерської роботи полягає у дослідженні та розробці методів прогнозування забруднення атмосферного повітря, шляхом розробки програмної системи. Система повинна будувати модель прогнозування за розробленим методом, давати прогноз на два тижні наперед та візуалізувати цей прогноз за допомогою веб-інструментів.

Для досягнення поставленої мети перед розробником постають такі задачі:

- аналіз програмних аналогів для визначення вимог до розроблюваного програмного забезпечення;
- дослідження методів прогнозування часових рядів шляхом використання рекурентних нейронних мереж типу «Довга короткострокова пам'ять»;
- розробка технічного завдання;
- аналіз та вибір інструментів розробки програмного забезпечення;

- проектування структури програмного забезпечення;
- розробка програмного забезпечення прогнозування забрудненості атмосферного повітря – розроблення ПЗ для прогнозування, бекенду та фронтенду;
- тестування та відлагодження програмного забезпечення;
- економічне обґрунтування доцільності розробки.

1.7 Висновки за розділом 1

У даному розділі було проаналізовано предметну область. В результаті проведеного аналізу запропоновано використовувати ШНМ у якості моделей прогнозування забруднення атмосферного повітря.

Було розглянуто вимоги до предметної області, підходи до проведення машинного навчання, було проведено роботу з набором даних для подальшого прогнозування. Крім того, були розглянуті програмні системи, що виконують схожі задачі.

Зважаючи на усі розглянуті питання, було створено план прогнозування. Загалом, новизна у рамках прогнозування має полягати в пошуку оптимального поєднання архітектур нейронних мереж для того, аби модель була достатньо навченою, але не перенавченою.

2 МАТЕРІАЛИ ТА МЕТОДИ

2.1 Аналіз методів

Поставивши завдання знайти показник ЄІЯП за обраним набором даних, виникає необхідність аналізу способів рішення задачі. Оскільки ЄІЯП є значенням найбільшого з п'яти індивідуальних індексів забруднюючих речовин [12], то відповідно прогноз має здійснюватися окремо за кожним з них, а вже з прогнозованих значень необхідно відібрати найбільше, співставити з таблицею та видати результат індексу ЄІЯП у інтервалі від 1 до 5.

Поставлене завдання є навчанням «із учителем», за умовами якого частину з наданої вибірки необхідно відмітити як «правильну» відповідь, і під час прогнозу спиратися на те, наскільки правильно модель нейронної мережі прогнозує той чи інший показник.

Для аналізу кожного забруднювача необхідно сформувати тренувальні та тестові вибірки, натренувати модель, перевірити її на тестових даних та виявити величину помилок на етапах тренування та тестування.

Оскільки для прогнозування було обрано створення нейронної мережі, то спершу необхідно проаналізувати, до якої з типів відноситься проблема, що розглядається, а отже яка з видів нейронних мереж є основною, базисною для вирішення поставленої задачі.

Прогнозування значень забрудненості атмосферного повітря тісно пов'язане з аналізом часових рядів, як було зазначено і розглянуто у попередньому розділі. Звичайна нейронна мережа будує алгоритм рішення виду $y = f(x)$, де y – вихід нейронної мережі, а x – вхідні дані (ознаки). Часовий ряд – це послідовність x (відомих даних) та y (прогноз), які корелюють між собою, тобто має місце автокореляція, при чому міститься деяка змінна в часі залежність $Z_n = g(Z_1, \dots, Z_{n-2}, Z_{n-1})$. Отже, спрогнозувавши поточне значення y , потрібно якось врахувати це значення для подальших прогнозів. Таким чином з'являється зворотний зв'язок в нейронної мережі і мережа стає рекурентною. Тож загальна функція (2.1) має вигляд:

$$y_n = f(x_n, y_1, \dots, y_{n-2}, y_{n-1}), \quad (2.1)$$

де y_n – n -ний вихід нейронної мережі;

x_n – n -ний вхідний параметр;

y_i – значення вихідного параметру на i -тій ітерації.

Стає зрозумілим те, що звичайних мереж типу «перцептрон» або нейронних мереж прямого поширення недостатньо для опису подібної структури. Серед усіх можливих видів нейронних мереж найкращими для вирішення задач з ітерованими вхідними параметрами є рекурентні нейронні мережі (РНМ), або ж англійською, recurrent neural network (RNN). Також значного поширення набула їхня розширена версія – нейронні мережі типу «довга короткочасна пам'ять» (ДКЧП). Нейронні мережі типу ДКЧП більш відомі за назвою «long short-term memory» (LSTM). Типова структура та особливості RNN та LSTM будуть розглянуті далі.

2.2 Рекурентні нейронні мережі

Рекурентні нейронні мережі – це тип штучних нейронних мереж, розроблених для розпізнавання шаблонів у послідовності даних, незалежно від їхнього роду. Основною відмінністю РНМ від інших архітектур, є так звана «наявність пам'яті». РНМ зберігає попередні значення у своєму стані [18].

Традиційні нейронні мережі не володіють цією властивістю, і в цьому їх головний недолік. РНМ ж навпаки зберігають попередні значення, що може розцінюватися як аналог людських знань і досвіду, оскільки людина робить свої висновки на основі попередніх набутих знань. РНМ є більш динамічними моделями, оскільки вони можуть змінюватися з плином часу таким чином, аби була можливість досягти достатньої точності для поставлених задач.

Фрагмент РНМ (рис. 2.1) A приймає вхідне значення x_i і повертає значення h_i . Наявність зворотного зв'язку дозволяє передавати інформацію від одного кроку мережі до іншого [19]-[20].

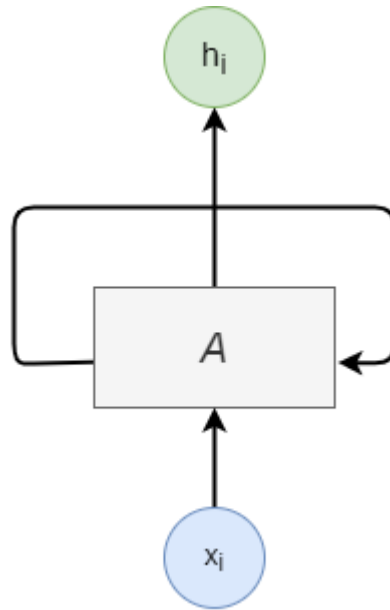


Рисунок 2.1 – Загальний фрагмент РНМ

Наявність циклічної залежності може вносити деяку незрозумілість. Тому для більшої ясності можна розгорнути РНМ за часом, тобто зобразити як кілька копій однієї і тієї ж мережі, кожна з яких передає інформацію подальшій копії (рис. 2.2) [20].

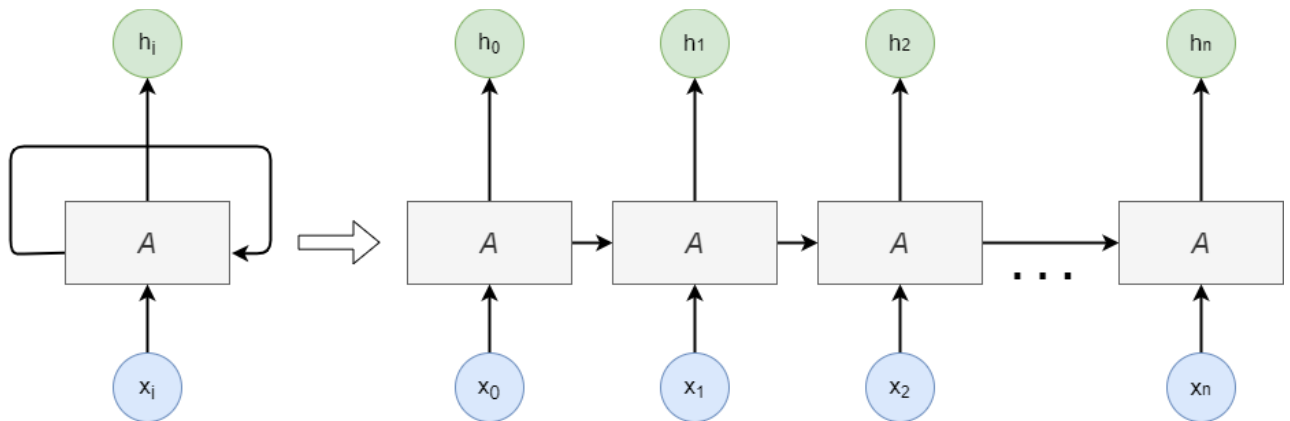


Рисунок 2.2 – Схема РНМ з розгорнутим зворотним зв'язком

Розгорнуте представлення РНМ у вигляді ланцюжка свідчить про тісний зв'язку з послідовностями і списками. Саме тому більшість сходиться на думці, що РНМ – найбільш природна архітектура нейронних мереж для роботи з даними такого типу. Крім того, РНМ підходить для роботи з часовими рядами.

Незважаючи на всі переваги, які дає можливість «запам'ятовувати» інформацію, у РНМ є великий недолік: проблема довгострокових залежностей. Нейрони добре «пам'ятають» недавно отриману інформацію, але не мають можливості надовго зберегти в пам'яті щось, що обробили багато циклів назад, якою б важливою та інформація не була. Для вирішення цієї проблеми використовують окремий вид РНМ – ДКЧП (LSTM). Це особливий різновид архітектури рекурентних нейронних мереж, здатний до навчання довготривалим залежностям [19]-[21].

2.3 Архітектурні особливості мережі LSTM

2.3.1 Загальна схема LSTM комірки

Мережі довгострокової короткострокової пам'яті (LSTM) – це такий тип РНМ, який був спеціально розроблений для виявлення довгострокових залежностей даних, завдяки чому навчання зазвичай було успішним для різних проблем часових рядів. Знаходження цих залежностей зазвичай означає, що вони здатні знаходити різні тенденції в даних і передбачати, якими будуть наступні результати. Для цього вихід LSTM (і загалом РНМ) буде залежати не тільки від його вхідного сигналу, але і від його поточного стану. Щоб переконатися, що цей стан знаходить довгострокові залежності (між навчальними вибірками), нам потрібно імплементувати цю LSTM.

Ці властивості здаються особливо важливими для прогнозування різних рівнів забруднення. Це завдання, яке в основному є прогнозуванням часових рядів, добре підходить для LSTM, і вони зможуть знаходити різні тенденції, які можна прогнозувати від одного дня до іншого. Використання LSTM дасть змогу не лише використовувати останні дані тренувальної вибірки, але й дослідити, як різні довгострокові тенденції можуть вплинути на прогноз [22].

РНМ типу LSTM були представлені Зеппом Хохрайтером і Юргеном Шмидхубером в 1997 році [20], а потім вдосконалені і популярно викладені в

роботах багатьох інших дослідників. Вони чудово вирішують цілий ряд різноманітних завдань і в даний час набули широкого використання.

У ядрі LSTM відбуваються більш складні перетворення, ніж у звичайних РНМ. Розглянемо загальну схему (рис. 2.3), а також кожен перетворення більш детально.

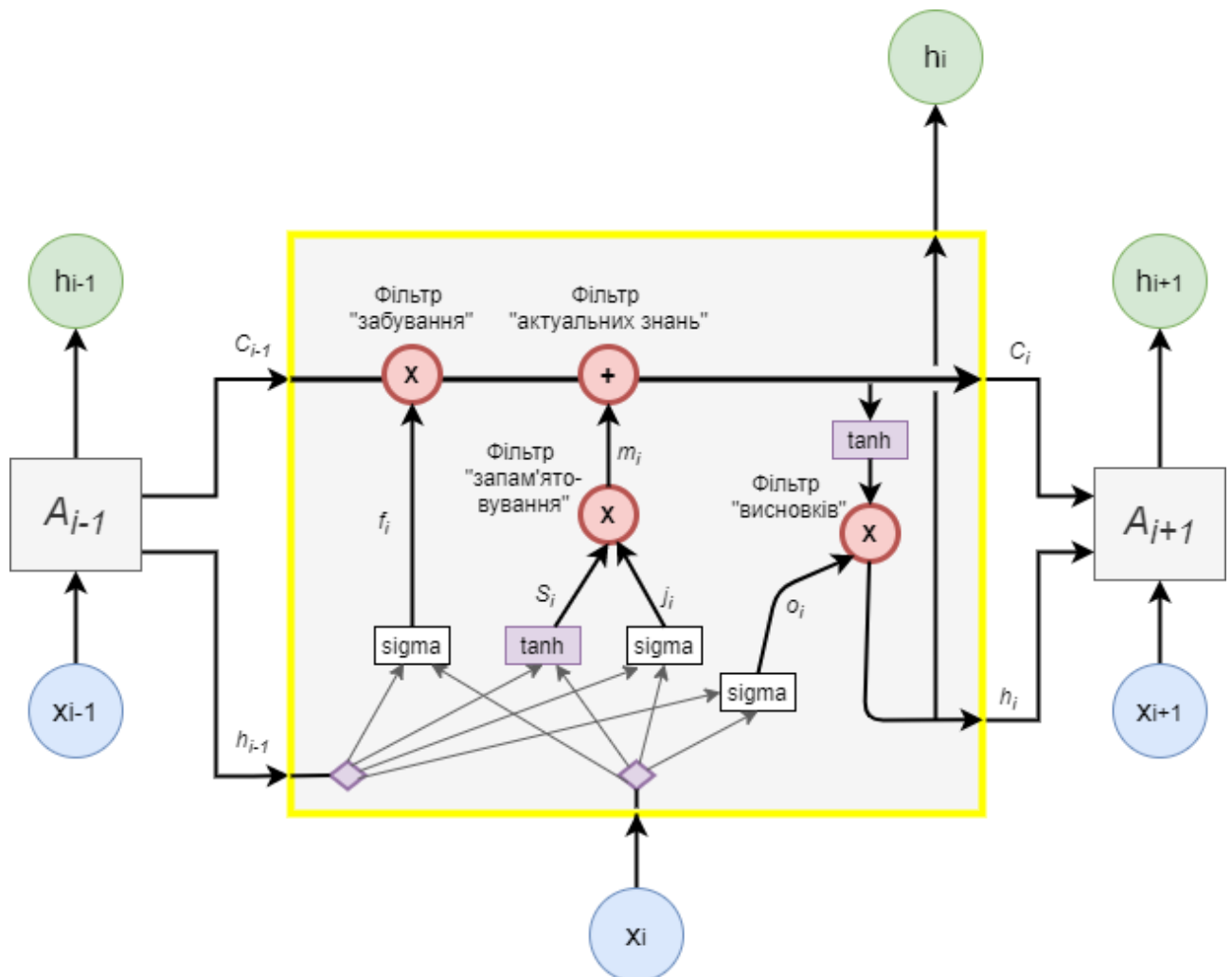


Рисунок 2.3 – Схема взаємозв'язків комірки LSTM

2.3.2 Фільтр забування

Перший крок в LSTM - визначити, яку інформацію можна відкинути зі стану комірки. Це рішення приймає шар із сигмоїдою, що називається «шаром фільтру забування» (forget gate layer) [22]. Його значення можна отримати за допомогою формули (2.2):

$$f_i = \sigma(W_{hf} \cdot h_{i-1} + W_{xf} \cdot x_i + b_f), \quad (2.2)$$

де σ – функція активації значення фільтру забування (сигмоїд);

W_{hf} – ваги виходу попереднього стану мережі для фільтру забування;

h_{i-1} – попередній вихід мережі;

W_{xf} – ваги входу мережі для фільтру забування;

x_i – вхідне значення нейронної мережі;

b_f – зсув для значень фільтру забування.

Результатом функції може бути діапазон значень $[0;1]$, при якому значення «1» означає «повністю зберегти», а «0» – «повністю забути». Шар фільтру забування зображений на рис. 2.4.

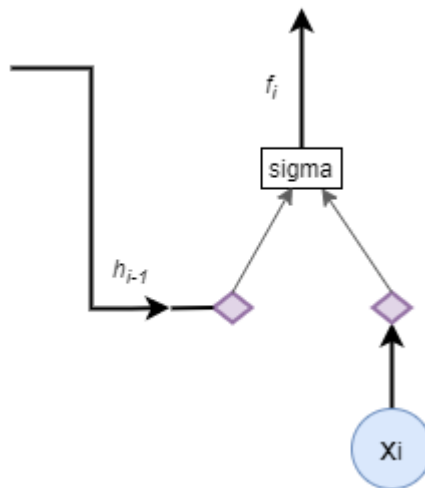


Рисунок 2.4 – Схема шару фільтру забування

2.3.3 Фільтр запам'ятовування

Наступний крок – вирішити, яка нова інформація буде зберігатися в стані комірки. Цей етап складається з двох частин. Спочатку сигмоїдальний шар під назвою «шар фільтру вхідних даних» (input layer gate) визначає, які значення слід оновити. Потім шар гіперболічного тангенсу (tanh) під назвою

«шар зміни пам'яті» будує вектор нових значень-кандидатів, які можна додати в стан комірки [22]. Формула (2.3) для шару фільтру вхідних даних:

$$j_i = \sigma(W_{hj} \cdot h_{i-1} + W_{xj} \cdot x_i + b_j), \quad (2.3)$$

де σ – функція активації значення фільтру вхідних даних (сигмоїд);

W_{hj} – ваги виходу попереднього стану мережі для фільтру вхідних даних;

h_{i-1} – попередній вихід мережі;

W_{xj} – ваги входу мережі для фільтру вхідних даних;

x_i – вхідне значення нейронної мережі;

b_j – зсув для значень фільтру вхідних даних.

Формула (2.4) для шару зміни пам'яті:

$$S_i = \tanh(W_{hs} \cdot h_{i-1} + W_{xs} \cdot x_i + b_s), \quad (2.4)$$

де $\tanh$ – функція активації значення зміни пам'яті (гіперболічний тангенс);

W_{hs} – ваги виходу попереднього стану мережі для зміни пам'яті;

h_{i-1} – попередній вихід мережі;

W_{xs} – ваги входу мережі для зміни пам'яті;

x_i – вхідне значення нейронної мережі;

b_s – зсув для значень змін пам'яті.

Оскільки значення шару зміни пам'яті знаходиться у інтервалі $[-1; 1]$, то такий шар визначає знак, тобто прямою або зворотною є залежність, яку необхідно запам'ятати. Значення шару фільтру вхідних даних знаходиться у інтервалі $[0; 1]$. Отже, значення фільтру запам'ятовування (рис. 2.5) є добуток зазначених шарів, що обчислюється за формулою (2.5):

$$m_i = j_i \cdot S_i, \quad (2.5)$$

де j_i – величина шару фільтру вхідних даних;

S_i – величина шару зміни пам'яті.

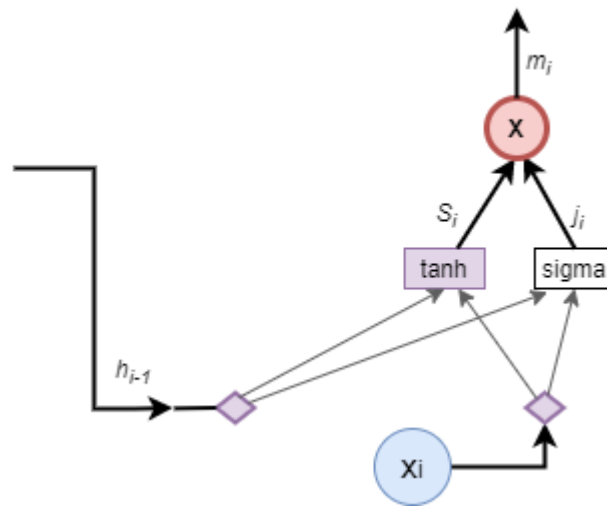


Рисунок 2.5 – Схема шару фільтру запам'ятовування

2.3.4 Фільтр актуальних знань

У фільтрі актуальних знань необхідно «актуалізувати знання», тобто замінити попереднє значення комірки на нове. Це один з центральних елементів усієї мережі LSTM, оскільки представляє собою стан комірки (cell state). Він нагадує конвеєрну стрічку, по якій інформація тече наскрізно, лише змінюючись на лінійних перетворювачах [22]. Необхідно помножити попередній стан на ступінь того, наскільки необхідно забути та додати ступінь того, що необхідно запам'ятати. Формула (2.6) має вигляд:

$$C_i = C_{i-1} \cdot f_i + m_i, \quad (2.6)$$

де C_{i-1} – значення попереднього стану пам'яті мережі;

f_i – значення фільтру забування;

m_i – значення фільтру запам'ятовування.

Схема отримання поточного значення стану пам'яті мережі, тобто фільтру актуальних знань зображено на рис. 2.6.

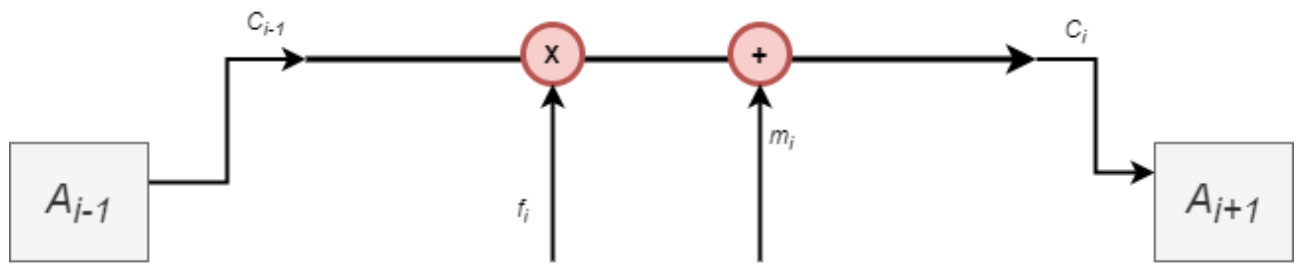


Рисунок 2.6 – Схема шару фільтру актуальних знань

2.3.5 Фільтр висновків

Для отримання вихідного результату LSTM-комірки необхідно поєднати шар фільтру вихідних даних (output gate layer) з поточним станом пам'яті комірки. Розрахунок шару фільтру вихідних даних наведено у формулі (2.7):

$$o_i = \sigma(W_{ho} \cdot h_{i-1} + W_{xo} \cdot x_i + b_o), \quad (2.7)$$

де σ – функція активації значення фільтру вихідних даних (сигмоїд);

W_{ho} – ваги виходу попереднього стану мережі для фільтру вихідних даних;

h_{i-1} – попередній вихід мережі;

W_{xo} – ваги входу мережі для фільтру вихідних даних;

x_i – вхідне значення нейронної мережі;

b_o – зсув для значень фільтру вихідних даних.

Розрахувавши значення фільтру висновків, на виході матимемо результуюче значення комірки, яке з одного боку виводиться з системи як результат обчислень, а з іншого – залишається для подальших комірок. Формула (2.8) для фільтру висновків має вигляд:

$$h_i = o_i \cdot \tanh(C_i), \quad (2.8)$$

де o_i – величина шару фільтру вихідних даних;

$\tanh$ – функція активації значення зміни пам'яті (гіперболічний тангенс);

C_i – величина поточного стану пам'яті комірки.

Схема отримання вихідного значення LSTM-комірки, тобто фільтру висновків зображено на рис. 2.7.

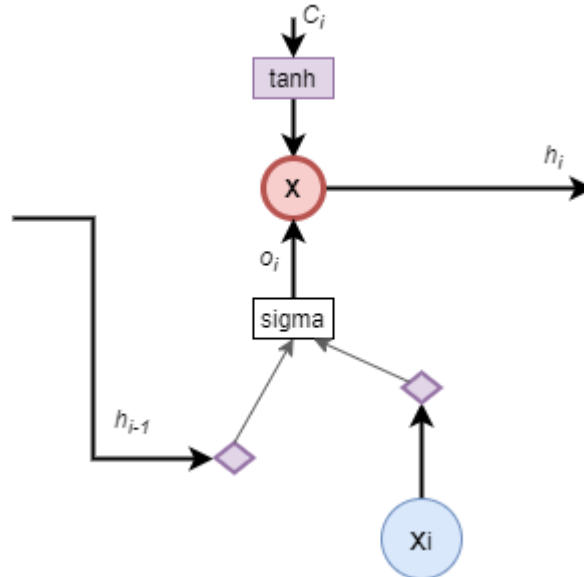


Рисунок 2.7 – Схема шару фільтру висновків

2.4 Шляхи покращення моделі для прогнозування якості повітря

Застосування лише одного шару з одним нейроном, нехай і навіть типу LSTM наврядчи принесе бажану точність прогнозуванню. Звичайно, розширення LSTM-мережі як і в ширину, так і в глибину може покращити якість прогнозування, однак необхідно розуміти й зворотні сторони: велике навантаження на систему, пов'язане зі складністю обчислень, проблеми з перенавчанням вибірки, «вибухові» або «затухаючі» градієнти, «карусель константної помилки» тощо.

2.4.1 Застосування функцій активації

Крім зазначених у попередньому підрозділі функцій активації типу «сигмоїда» (σ), яка перетворює вхідний сигнал у вихідний у інтервалі $[0;1]$, а також гіперболічного тангенсу ($\tanh$), у свою чергу виконує перетворення у

інтервалі $[-1; 1]$, існують й інші функції активації. Зокрема, можна використати «випрямлювач» (ReLU), який повертає 0 для від'ємних значень, та саме число – для додатних. Тобто його область значень знаходиться у межах $[0, +\infty)$. Крім того, для фільтру шару вихідних значень можна застосувати функцію «softmax», яка нормалізує до одиниці суму значень вихідних нейронів. Однак з практичної точки зору, емпіричним шляхом було встановлено, що змінення функцій активації не вносить значних змін у результат вбік його покращення. Що й не дивно, оскільки базова модель LSTM була розроблена і неодноразово успішно протестована на різних реальних задачах.

2.4.2 Налаштування гіперпараметрів моделі

Методи градієнтного спуску та зворотного поширення помилки є одними з основних, які використовуються для навчання нейронної мережі. Під час навчання нейронної мережі ці методи потребують застосовування деяких гіперпараметрів. Їхній сенс в тому, що їхнє значення неможливо визначити наперед. Лише ітеративним процесом можна виявити, комбінація яких з них принесе найкращий результат безпосередньо для того набору даних, над яким виконується аналіз.

Зокрема, вибираючи такий гіперпараметр, як швидкість навчання, необхідно розуміти, з якою швидкістю модель адаптується до її змін. Висока швидкість означає, що модель забуває набуті давно знання, та користується щойно отриманими. Це значно збільшує швидкість прогнозування та потребує меншої кількості обчислювальних ресурсів, однак погіршує точність, особливо для прогнозування часових рядів, де спостерігається сезонність або тренд. З іншого боку, низька швидкість навчання також призводить до неякісного узагальнення щодо отриманих знань у найближчому минулому. Шляхом практичного аналізу величиною навчання було встановлено значення 0.25, тобто згладжування змін на 25%.

2.4.3 Додавання шарів та кількості нейронів у шарі

Незважаючи на усю складність внутрішньої будови комірки LSTM, вочевидь зрозуміло, що одного нейрону не достатньо. Кількість нейронів не має якогось фіксованого значення, тому також є гіперпараметром, оскільки їхня кількість залежить від типу задачі. Загалом, підбір кількості вирішується емпірично, але є декілька порад щодо їхнього вибору.

Спочатку необхідно обчислити середньоквадратичне відхилення (root mean-square error – RMSE) тренувальної та тестової вибірок для кожної епохи з різною кількістю максимальних епох. Це запобігає перенавчанню і дає приблизний діапазон епох для початку. Потім можна повторити метод, але встановивши епохи константними і протестувати модель з різною кількістю нейронів. Важливо, щоб крива випробувань RMSE не повинна бути опуклою, оскільки вона позначає перенавчання. За допомогою такого методу можна налаштувати параметри так, аби отримати хороший компроміс між точністю моделі та її узагальненням [23].

Кількість шарів також відіграє надзвичайно важливу роль у прогнозуванні, але при їхньому виборі також треба уникати проблеми перенавчання. У розробленій системі було протестовано багато комбінацій шарів нейронних мереж, однак найкраще себе показали два з них: комбінація LSTM з вісьмома нейронами, SoftMax з чотирма, та Dense з одним (рис. 2.8), а також LSTM з вісьмома, LSTM з двома та Dense з одним нейроном (рис 2.9). В них спостерігалось найменше значення функції втрат – близько 1% для тренувальної вибірки, та 1.5% – для тестової. Крім того, вхідними даними для мережі можуть бути не лише безпосередньо прогнозовані значення (x_i), а й додаткові (a_i), які можуть коригувати вихід мережі. У розробленій моделі у якості додаткового вхідного параметру було використано величину автомобільного трафіку.

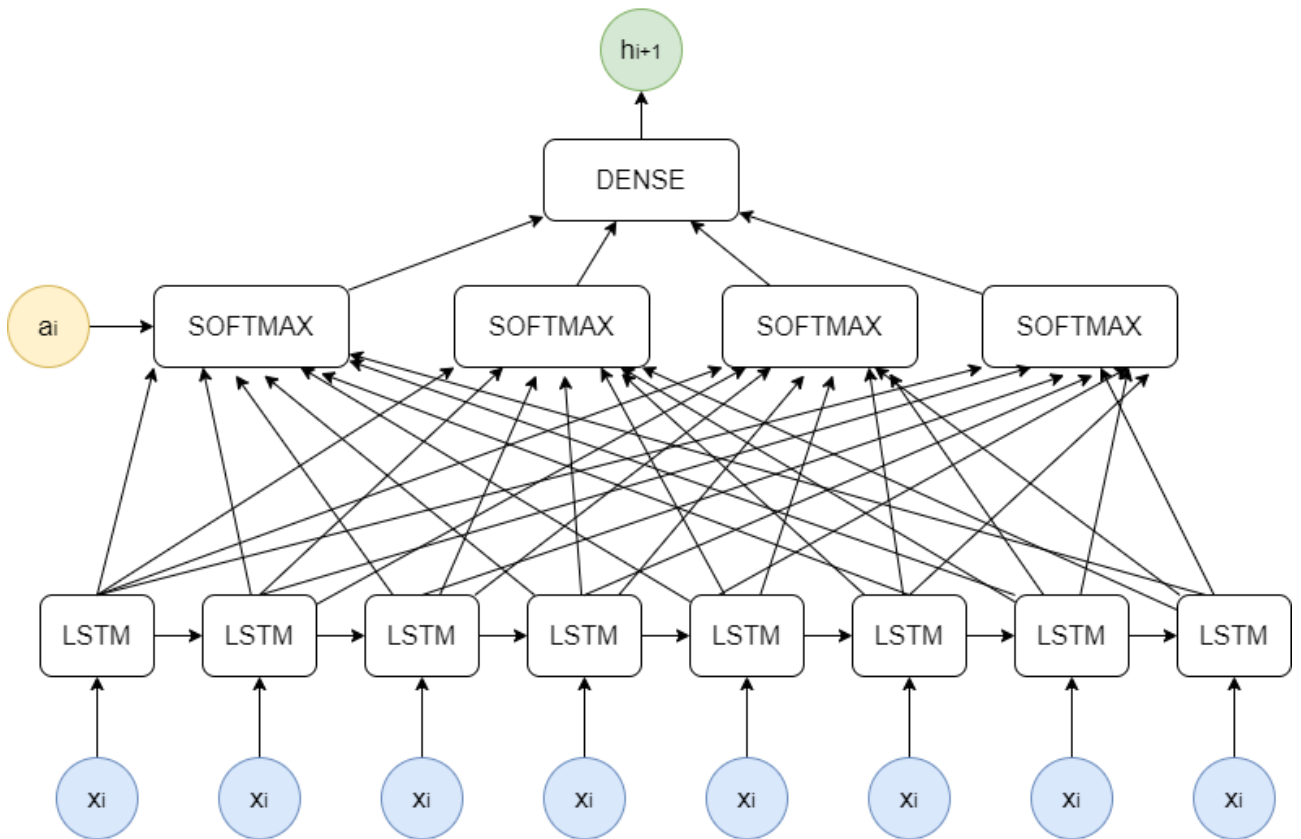


Рисунок 2.8 – Схема нейронної мережі з шарами LSTM, SoftMax та Dense

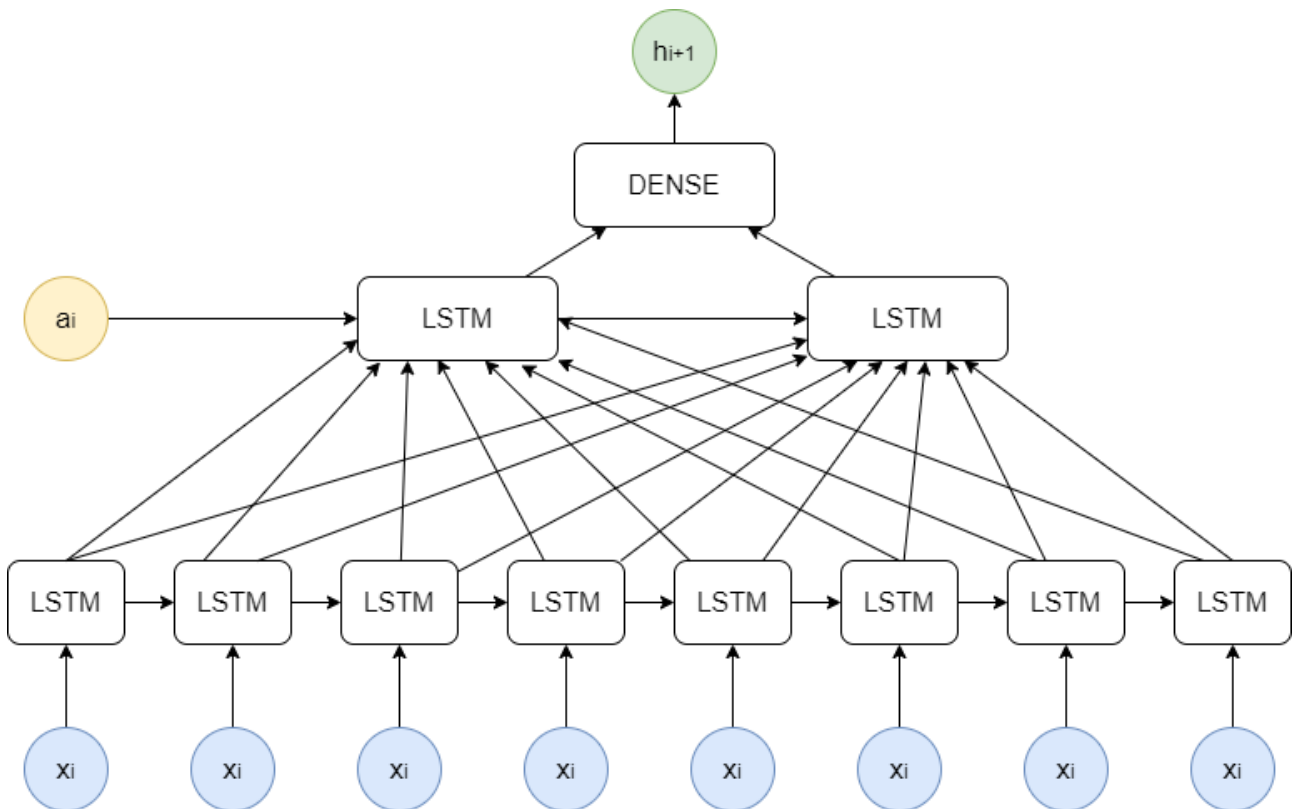


Рисунок 2.9 – Схема нейронної мережі з шарами LSTM, LSTM та Dense

2.4.4 Пошук кореляції з трафіком руху автомобілів

Очевидно, що забруднення атмосферного повітря має свої причини. Це можуть бути викиди шкідливих речовин із заводів або автомобілів. З цього боку аналіз автомобільного трафіку може допомогти більш точно проаналізувати та спрогнозувати рівень забрудненості такими сполуками, як NO_2 , SO_2 та інших. Майже всі забруднювачі, які були зазначені у наборі даних, можуть продукуватися автомобілями у тій чи іншій кількості.

Аналіз даних підтверджує гіпотезу – показники забруднювачів та щільність трафіку насправді показують позитивну кореляцію (рис. 2.10).

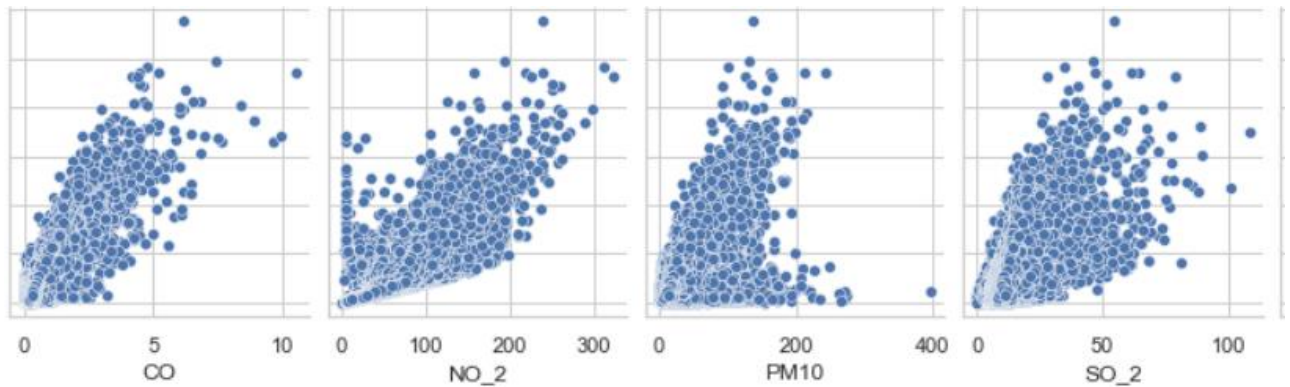


Рисунок 2.10 – Графік кореляції трафіку та забруднювачів CO, NO₂, PM₁₀, SO₂

Модель прогнозування з урахуванням автомобільного трафіку матиме вигляд, зображений на рис. 2.11.

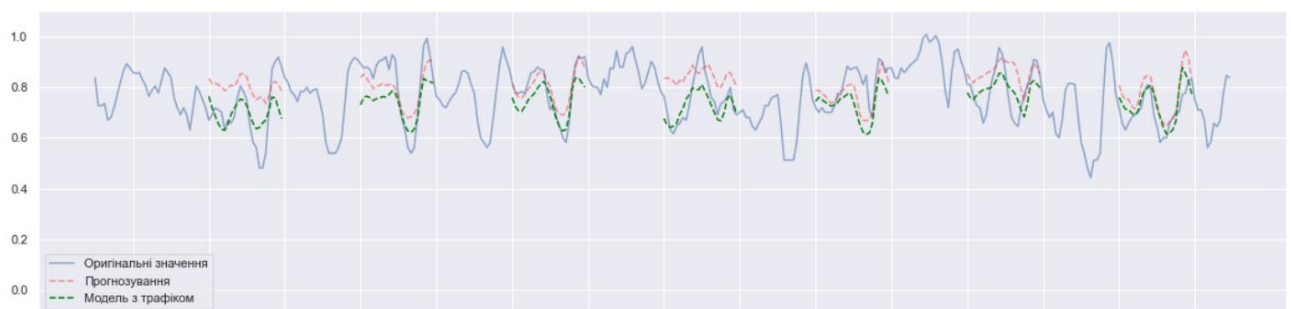


Рисунок 2.11 – Графік прогнозування з урахуванням автомобільного трафіку

2.4.5 Передбачення майбутніх значень з використанням `look_back`

Маючи натреновану модель з непоганими значенням прогнозування, що було перевірено на тестовій вибірці, необхідно розширити модель для передбачення майбутніх значень.

Оскільки ми намагаємося передбачити майбутнє, у прогнозі буде велика кількість невизначеності. Щоб передбачити значення завтрашнього дня, необхідно додати в модель значення останніх n (`look_back`) днів, і тоді можна отримати завтрашнє значення як результат. Щоб отримати значення післязавтра, необхідно подати значення за останні $n-1$ днів разом із значенням завтрашнього дня та вихідним даним моделі після завтрашнього значення [24]. Однак оскільки наступне значення буде використовувати не істинне значення, а прогнозоване, то прогнозування навіть на місяць вперед може бути не досить точним. Однак воно загалом зможе показати загальну тенденцію рівня забрудненості. Крім того, оскільки прогнозування йде на наступні дні, модель прогнозування можна запускати один раз вдень, наприклад опівночі, підтягуючи актуальні, справжні значення з датчиків. У такому випадку прогноз буде уточнюватися. Схема такої мережі дещо змінюється щодо вхідних значень: на перший шар нейронів з LSTM комірками подаються `batch_input_shape` у розмірі тієї кількості днів, скільки нам необхідно для прогнозування. Прогнозовані значення замінюють попередні за схемою черги. Наприклад, якщо прогноз треба здійснити на 30 днів, дані за день, що був 30 днів тому, видаляються, увесь масив зсувається, і у кінець додається прогнозоване значення. На кожній ітерації ця операція повторюється.

2.5 Запропонований метод прогнозування

Зважаючи на зазначені шляхи покращення моделі LSTM, запропонований метод полягає у реалізації таких етапів:

- етап 1. Формування вибірки;

- етап 2. Створення моделі;
- етап 3. Навчання моделі;
- етап 4. Тестування моделі;
- етап 5. Прогнозування для майбутніх значень.

2.5.1 Етап 1. Формування вибірки

Набір даних, для якого виконувалось прогнозування складається з забруднювачів, дані про які збирались напротязі 18 років щогодини у м. Мадрид. У середньому це близько 150 тисяч замірів для кожного забруднювача. Однак деякі зі станцій фіксували дані не весь час, тому необхідно обрати саме такі станції, що мають достатньо даних про головні забруднювачі, а саме про O_3 , NO_2 , SO_2 , $PM_{2,5}$ і PM_{10} , оскільки вони є складовими для прогнозування індексу якості повітря. Прогнозування виконується окремо для кожного забруднювача.

Аби підготувати дані для задачі, треба зробити декілька кроків. Треба розподілити дані на вхід так, аби модель обробляла 24 записи (тобто один повний день) та прогнозувала очікуване значення на наступний день.

Далі необхідно відокремити тренувальну та тестову вибірки. У якості величини розділення було обрано 0.2, тобто 80% значень відносяться до тренувальної вибірки, а 20% – до тестової.

Крім того, необхідно нормалізувати вхідні дані, оскільки більшість методів МН краще працюють з нормалізованими функціями.

2.5.2 Етап 2. Створення моделі

Архітектура розробленої нейронної мережі складається з першого шару нейронів типу LSTM, що взаємодіють із вхідними даними, а також прихованого шару. Запропоновано 8 нейронів на першому шарі, оскільки коливання забруднювачів у межах 3-х годин незначне. Отже на кожен із 8-ми входів

мережі подається середнє арифметичне значення за 3 години. Далі результати обробляються прихованим шаром з 2-ма LSTM-нейронами.

Як було зазначено у п. 2.4.4, набір даних забруднювачів має позитивну кореляцію із величиною автомобільного трафіку, тож було запропоновано використання величини трафіку у обраний день, як уточнюючої характеристики. Величина трафіку впливає на один з нейронів прихованого шару, а отже, й на підсумковий результат.

2.5.3 Етап 3. Навчання моделі

На етапі навчання моделі необхідно підібрати оптимальні функції активації. За рекомендаціями використання LSTM були обрані сигмоїда та гіперболічний тангенс на відповідних фільтрах, як зазначено у пп. 2.3.2-2.3.5.

Щодо способу навчання, оскільки одна епоха занадто велика для комп'ютера, як у нашому випадку, її необхідно розділити на маленькі партії (batches). Також нашу вибірку зручно поділити на дні, тобто по 24 години, а крім того, необхідно зробити так, щоб мережа передбачала усереднене значення наступного дня, то найбільш зручним варіантом є розділити вибірку на партії з 48 елементів – 2 повних дні. З 24-х елементів наступного дня обчислюється медіанне значення, яке необхідне для порівняння з прогнозованим значенням. Отож робочий набір прогнозування для однієї епохи буде складатися приблизно по 3 тисячі ітерацій по 48 елементів у партії.

Епохи – якщо відбулася одна епоха (epoch) – весь набір даних пройшов через нейронну мережу в прямому і зворотному напрямку тільки один раз. Одна епоха призводить до недонавчання, а надлишок епох - до перенавчання [25]. Під час практичного аналізу коду було встановлено, що оптимальною кількістю нейронів є 8 з кількістю епох 8, оскільки при такій конфігурації RMSE тестової вибірки є мінімальним.

Крім того, необхідно налаштувати LSTM таким чином, аби вона передавала стан з однієї партії як початкові значення для іншої. Таким чином,

навіть якщо градієнти будуть схильні до вибуху або затухання, актуальний стан мережі все одно буде доступним. У фреймворку Keras для мови програмування Python це зробити легко за допомогою параметру `stateful=true`.

2.5.4 Етап 4. Тестування моделі

На етапі тестування розробленої моделі необхідно порівняти точність та рівень втрат моделі. Тестування здійснюється за допомогою проведення всіх необхідних епох навчання та прогнозування значень тестової вибірки. Необхідно виконати низку тестувань, змінюючи гіперпараметри моделі. Найкращі значення точності та втрат тестової вибірки будуть вказувати на оптимальність підбору гіперпараметрів моделі.

2.5.5 Етап 5. Прогнозування для майбутніх значень

Коли розроблена модель буде мати достатню точність, її можна використовувати для прогнозування майбутніх значень.

Спочатку визначається кількість днів, яку необхідно спрогнозувати. Як було зазначено у п. 2.4.5, необхідно взяти дані як з вибірки забруднювачів, так і з вибіркою трафіку про n останніх днів, та отримати значення для $(n+1)$ -го дня. Далі для того, щоб спрогнозувати величину забруднювача для $(n+2)$ -го дня, необхідно зсунути вибірку на 1, і використати вже спрогнозоване значення у якості вхідного параметру. Однак необхідно мати на увазі, що прогнозування майбутніх значень пов'язане з невизначеністю, оскільки кожен наступний день прогнозу буде використовувати вже спрогнозовані значення, які є лише узагальненням моделі і можуть бути далекими від стрімких змін у реальності. Для вирішення цієї проблеми у системах реального часу ПЗ для прогнозування один раз на день оновлювало би свій прогноз з урахуванням істинних значень за минулий день.

2.6 Висновки за розділом 2

У даному розділі були проаналізовані методи прогнозування з використанням рекурентних нейронних мереж, розглянуто особливості роботи РНМ, а також глибоко досліджено будову LSTM-комірки.

На основі архітектури нейронних мереж типу LSTM було створено модель прогнозування якості атмосферного повітря, яка була перевірена на тестовій вибірці. Над моделлю був проведений аналіз, були встановлені необхідні функції активації та налаштовані гіперпараметри моделі. Були розроблені різні комбінації шарів, і обрані кращі з них – тобто ті, що дають меншу помилку. Метод роботи був розширений на обробку не тільки вхідних даних про забруднювачі, але й автомобільним трафіком для покращення розуміння предметної області моделлю. Був конкретизований підхід для передбачення майбутніх значень з використанням минулих.

Тож було теоретично та практично обґрунтовано способи та методи прогнозування якості атмосферного повітря за набором даних з часовими рядами.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Функціональні вимоги до програмного забезпечення

Головним завданням програмного забезпечення є прогнозування індексу якості повітря у межах міста за зазначеними станціями. Для реалізації цього завдання система поділяється на окремі блоки – веб, прогнозуюча модель, база даних. Основою задачею є обробка датасету з даними про рівень забруднювачів повітря, створення моделі на основі блоків LSTM та прогнозування майбутніх значень. Прогнозовані значення записуються у базу даних, далі оброблюються бекендом, що надає зручний API. Фронтенд-частина відповідає за графічну візуалізацію даних та використовує API бекенду. Тож функціональні вимоги до програмного забезпечення можна розділити на декілька рівнів. Вимоги до блоку аналізу даних та прогнозування:

- аналіз датасету: вибір релевантних станцій, видалення пропусків в даних, нормалізація даних, формування тренувальної та тестової вибірок;
- імплементація моделі, тренування моделі, підбір гіперпараметрів, тестування моделі;
- візуальне порівняння справжніх значень та прогнозованих для тестової вибірки;
- прогнозування майбутніх значень;
- візуалізація прогнозованих значень;
- збереження прогнозу у базу даних.

Вимоги до бекенду:

- взаємодія з базою даних;
- перетворення даних прогнозів у зручний для візуалізації формат;
- створення сервісу для аналізу найсвіжіших прогнозів, виявлення найбільшого забруднювача, обчислення індексу якості повітря;
- надання API з можливістю давати прогноз для обраної станції на задану кількість днів, регулювати тип прогнозу (оптимістичний / середній / песимістичний).

Вимоги до фронтенду:

- карта міста з позначками станцій та прогнозованої якості повітря;
- детальна інформація про прогноз обраної станції: дата, загальний індекс, найбільший забруднювач, його прогнозоване значення;
- вибір інтервалу прогнозування;
- вибір типу прогнозування.

3.2 Структура програмного забезпечення. Мікросервіси

Розподілення програмного забезпечення на окремі за логікою підпроекти є однією з сутностей мікросервісної архітектури. Мікросервіси – це спосіб розбиття великого програмного забезпечення на слабо пов'язані модулі, які комунікують один з одним за допомогою API [26]. Навпаки, коли усе ПЗ зібрано до купи його називають монолітом. Моноліти складніше піддаються масштабуванню та розробці окремих модулів системи без змін інших. Тож мікросервісна архітектура націлена на зменшення ступеня зачеплення – залежності одного модуля від іншого, та збільшення ступеня зв'язності – сфокусованості підсистем. Переваги використання мікросервісної архітектури:

- мікросервіси простіше масштабувати;
- покращують ізоляцію збою компонентів: великі програми можуть продовжити ефективно працювати, навіть при несправності якогось окремого модуля;
- усувають схильність додатків до одного технологічного стеку, що дозволяє обрати найбільш відповідні інструменти для тої чи іншої задачі, замість універсальних;
- залежності будуть значно простіші, ніж при монолітній архітектурі, до того ж буде набагато простіше відкотити все назад. Чим менше коду в одному додатку, тим легше з ним працювати;
- новим членам команди буде легше розбиратися з окремими модулями, аби зрозуміти їхнє функціональне призначення.

Структурна схема розробленої системи з урахуванням мікросервісного підходу зображена на рис. 3.1.

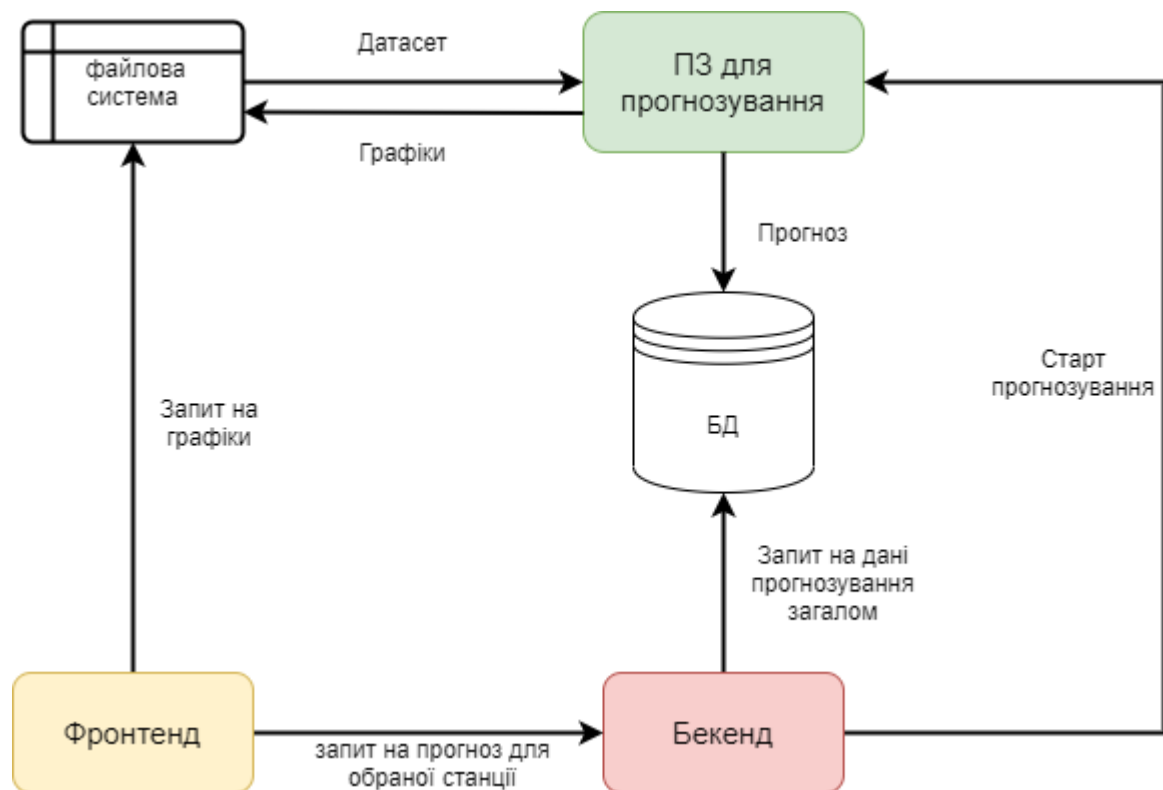


Рисунок 3.1 – Структурна схема ПЗ, що розроблюється

3.3 Моделювання сценарію використання програми

Структурна схема, зазначена на рис. 3.1, складається з логічних частин, поділених на мікросервіси. Однак вона не достатньо описує безпосередньо логіку роботи користувача з програмним забезпеченням. Кінцевий користувач має доступ лише к частині фронтенду, яка візуалізує прогнози на вибраних станціях. Відповідно, прогнозування здійснюється на сервері, дані про результат прогнозування зберігаються у БД.

Моделювання логіки роботи та дій, що виконуються у мікросервісах, можна зобразити за допомогою діаграми діяльності, що зображена на рис. 3.2.

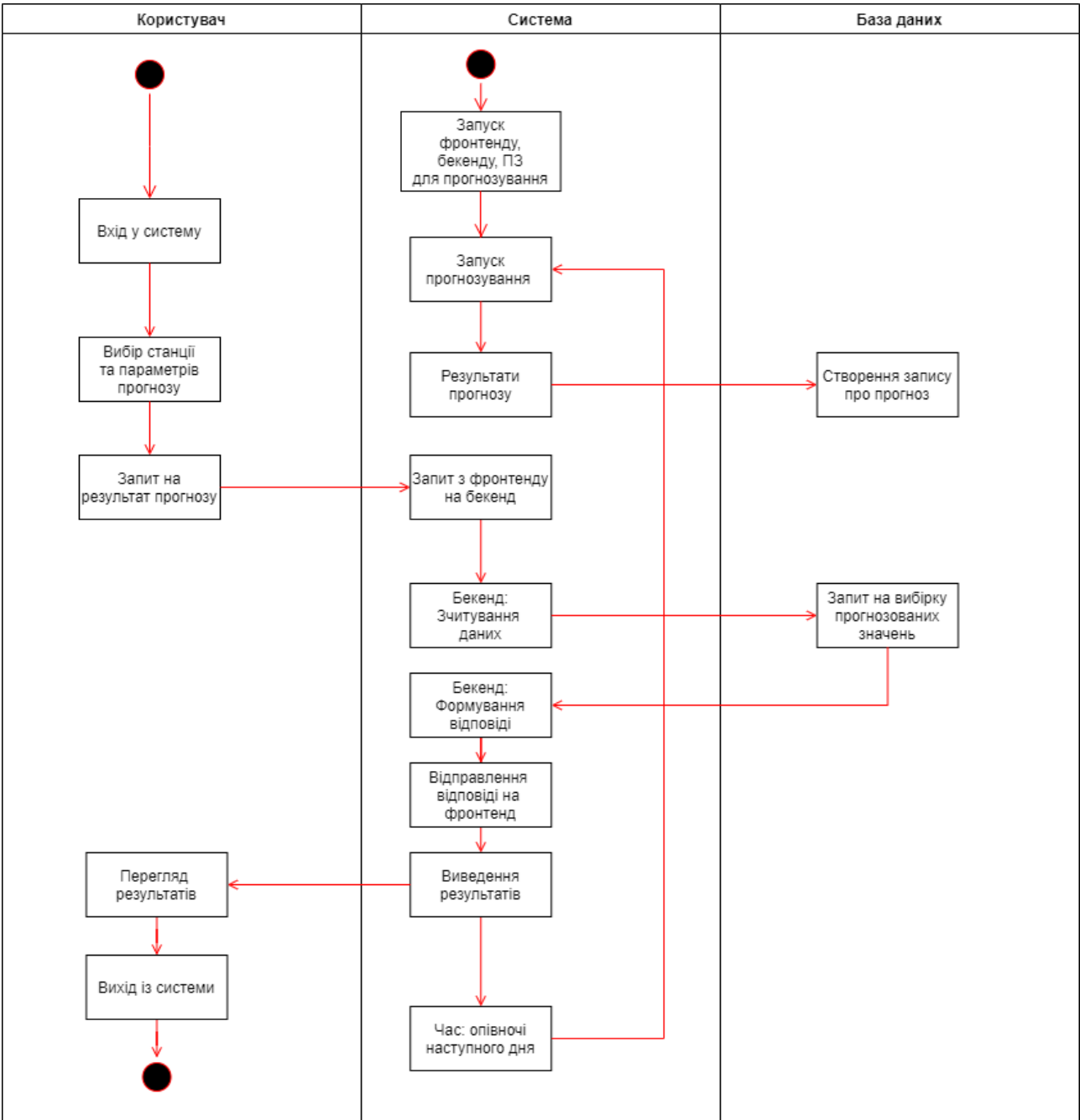


Рисунок 3.2 – Діаграма діяльності в нотації UML 2.5

3.4 Вибір засобів реалізації програмного забезпечення

3.4.1 Вибір мов програмування

Оскільки прогнозування відноситься до типу задач машинного навчання та роботи з нейронними мережами, необхідно обирати з мов, які найбільш для цього створені. Обираючи між Python, R, SPSS, SAS для реалізації ПЗ для прогнозування було обрано мову програмування Python. Вибір був зроблений

саме таким чином, оскільки Python підтримує роботу з необмеженою довжиною чисел, має велику кількість бібліотек, які спеціалізуються на машинному навчанні (Keras, TensorFlow, SkLearn та інші). Порівняння мов програмування, які спеціалізуються на задачах Data Science, наведено у табл. 3.1.

Таблиця 3.1 – Порівняння мов програмування для задач Data Science

Критерії порівняння	Мови програмування			
	Python	R	SAS	SPSS
Стандартні бібліотеки шаблонів	++	+	+	+
Розробка задач машинного навчання	+	+	+	+
Безкоштовність ПЗ	++	++	—	—
Велика спільнота, яка власноруч створює бібліотеки	+	+	—	—
Підтримка drag-drop функціоналу	—	—	+	+
Робота з великими обсягом даних	+	—	+	—
Масштабованість	+	—	+	+

Python – це проста у використанні інтерпретована мова програмування. Це стало одним з найпопулярніших варіантів для Data Science, завдяки спрощенню кривої навчання та корисним бібліотекам. Читабельність коду, яку надає Python, також робить його популярним вибором для Data Science. Оскільки Data Scientist вирішує складні проблеми, отже, ідеально мати мову, яка є легшою для розуміння. Python полегшує користувачеві імплементацію рішень, дотримуючись стандартів необхідних алгоритмів [27].

Python має такі переваги:

- безкоштовний та відкритий код;
- висока читабельність коду;

- автоматичне керування пам'яттю;
- висока продуктивність.

Python підтримує широкий спектр бібліотек. Різні етапи вирішення проблем в Data Science використовують власні бібліотеки. Вирішення проблеми Data Science передбачає попередню обробку, аналіз, візуалізацію, прогнозування та збереження даних. Для виконання цих кроків Python виділив спеціальні бібліотеки, такі як – Pandas, NumPy, Matplotlib, SciPy, scikit-learn тощо. Крім того, вдосконалені бібліотеки Python, такі як Tensorflow та Keras надають інструменти глибокого навчання для вчених з Data Science [27].

Щодо веб-частини програмного забезпечення, було обрано деякі фреймворки, про які буде сказано далі.

Для реалізації бекенду було обрано мову програмування C#, оскільки вона є компільованою та суворо типізованою, реалізує автоматичне очищення пам'яті. Скомпільований код конвертується у байт-код, який інтерпретується за допомогою common language runtime у машинний код. C# є об'єктно-орієнтовним, підтримує мультипоточність та асинхронне програмування, що необхідно для коректної обробки запитів на сервері [28].

Для реалізації фронтенду було обрано HTML 5, CSS 3, а також TypeScript. TypeScript – більш типізована версія JavaScript. Різниця між ними:

- TypeScript відомий як об'єктно-орієнтована мова програмування, тоді як JavaScript – скриптова мова;
- TypeScript надає підтримку модулів, які JavaScript не підтримує;
- TypeScript має змінну типу інтерфейс, а JavaScript не має [29].

Безпосередньо у зв'язці з C#, що є типізованим, використання TypeScript є більш прийнятним та розповсюдженим.

3.4.2 Вибір фреймворків

Для задач прогнозування було обрано Jupyter Notebook. Це така платформа для інтерактивного програмування і створення звітів-записників.

Ключовою сутністю Jupyter-записників є такий записник, де є велика кількість різноманітних віджетів, інтерактивних елементів, які можна змінювати [30]. Писати код у Jupyter можна різними мовами, включаючи Python, запускати його блоками та бачити результат в окремому вікні. Крім того, Jupyter запускається як сервер, тож його можна використовувати у браузері. Jupyter може відображати графіки, які створюються бібліотеками Python, а також зберігати стан змінних, що дуже зручно при роботі з великим обсягом даних.

Для створення бекенду було обрано фреймворк ASP.NET Core 3.0, оскільки він є кросплатформеним, а також на якому ведеться розробка за допомогою мови C#. Наразі планується вихід версії .NET 5, яка поєднає у собі як переваги ASP.NET Framework, так і Core, що дозволить платформі .NET від Microsoft стати уніфікованою.

В ASP.NET Core 3.0 було додано багато шаблонів, аби спростити розробку кодів як серверної частини програми з допомогою .NET Core, так і клієнтської частини за допомогою Angular, React або будь-якої іншої фронтенд-технології [31].

Тож у якості фронтенду було обрано Angular версії 9.0, оскільки .NET Core надає непоганий шаблон зв'язки ASP.NET Core та Angular, передача даних між бекендом та фронтендом здійснюється коректно, Angular надає можливість розбивати роботу з даними на компоненти та модулі.

3.4.3 Вибір інтегрованих середовищ розробки

Для задач прогнозування було обрано інтегроване середовище розробки PyCharm. PyCharm – IDE для мови програмування Python. JetBrains PyCharm виконує інспекцію коду на льоту, автодоповнення, в тому числі ґрунтуючись на інформації, отриманій під час виконання коду, навігацію по коду, забезпечує безліч рефакторингів [32]. PyCharm надає можливість створювати Jupyter-записники і відлагоджувати їх. Серед переваг PyCharm:

- потужний і функціональний редактор коду з підсвічуванням синтаксису, авто-форматуванням і авто-відступами для підтримуваних мов;
- велика кількість інспекцій коду;
- інтеграція з баг / issue-трекерами, такими як JIRA, GitHub, Redmine;
- величезна колекція плагінів, що постійно поповнюється;
- крос-платформеність (Windows, Mac OS X, Linux) [32].

PyCharm надає широкий спектр можливостей, до головних позитивних якостей можна віднести крос-платформеність, підтримку репозиторіїв та фреймворку Jupiter, вбудований сервер та популярність, що надає велику базу шанувальників та підтримку програмного забезпечення розробниками.

Очевидним вибором IDE для обраних фреймворків для бекенду і фронтенду є Microsoft Visual Studio. Visual Studio є одним з сімейства розробок компанії Microsoft, тому взаємодія з обраними технологіями – C#, ASP.NET Core, Angular – є легкою та невимушеною. Visual Studio має: редактор вихідного коду з підтримкою рефакторингу коду, веб-редактор, редактор форм для спрощення створення графічного інтерфейсу, дизайнер класів, дизайнер схеми бази даних.

Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду, додавання нових наборів інструментів (наприклад, для редагування і візуального проєктування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, Team Explorer для роботи з Team Foundation Server) [33].

3.5 Проєктування бази даних

Для розробки БД було використано СКБД Microsoft SQL Server. Схема бази даних предметної області містить 4 таблиці:

- прогнози (таблиця Forecastings);
- станції (таблиця Stations);
- забруднювачі (таблиця Pollutants);
- номер прогнозу (таблиця ForecastingSequenceNumbers).

3.5.1 Схеми бази даних

Основною сутністю предметної області є прогноз, який пов'язаний з таблицями станцій та забруднювачів зв'язками «багато-до-одного». Оскільки для однієї станції можна здійснити багато прогнозів, так само як і для одного забруднювача. Схеми бази даних зображена на рис. 3.3.

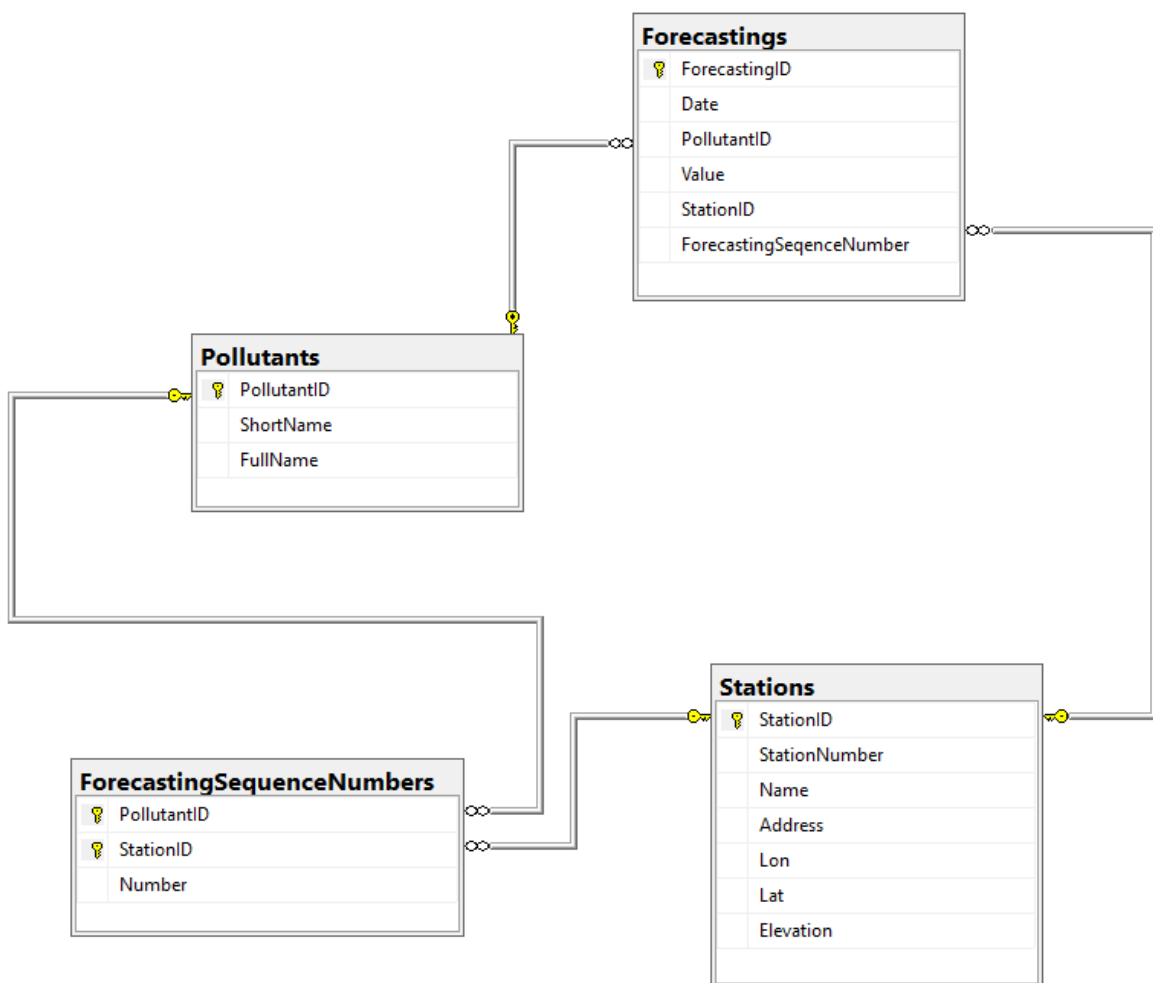


Рисунок 3.3 – Схеми бази даних

3.5.2 Підхід до взаємодії бази даних та мікросервісів

Оскільки у випадку прогнозування та використання мікросервісів база даних виступає центральним коннектором між ПЗ для прогнозування та бекендом, то вирішено було використовувати підхід Database First. Він полягає у створенні схеми бази даних спочатку, а вже потім на її основі генерації коду та зв'язків у бекенд-додатку. Для спрощення взаємодії у бекенді було використано ORM (об'єктно-реляційне відображення) Entity Framework Core.

Entity Framework Core (EF Core) являє собою об'єктно-орієнтовану, легко розширювальну технологію від компанії Microsoft для доступу до даних. Відмінною рисою Entity Framework Core, як технології ORM, є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ можливо створювати різні запити на вибірку об'єктів, в тому числі пов'язаних різними асоціативними зв'язками. А Entity Framework при виконанні запиту трансліює вираження LINQ в вирази, зрозумілі для конкретної СКБД (як правило, в вирази SQL) [34].

Entity Framework дозволяє користуватися реверсивною інженерією, використовуючи як Code First, так і Database First підходи. Для Database First було використано технологію scaffolding – генерацію об'єктів у бекенді за таблицями та зв'язками у базі. Це дозволяє зекономити час та звести до нуля помилки при копіюванні логіки з БД у бекенд.

3.6 Вимоги до технічних засобів

Для нормальної розробки програми, повинні виконуватися наступні умови до апаратної частини:

- наявність IBM-сумісної персональної електронної обчислювальної машини на якій буде працювати додаток;
- процесор із тактовою частотою 2,2 GHz та більше;

- оперативна пам'ять 4 ГБ та більше – оскільки прогнозування є трудомістким процесом;
- вільний простір на жорсткому диску не менше, ніж 7 ГБ;
- Python версії 3.7.9 та вище;
- інтегроване середовище розробки PyCharm актуальної версії;
- інтегроване середовище розробки Visual Studio 2019 та вище;
- Microsoft SQL Server актуальної версії;
- наявність сучасної версії браузера (Google Chrome, Opera, Safari та інші).

3.7 Висновки за розділом 3

У даному розділі було виявлено функціональні вимоги до програмного забезпечення, що розроблюється, була розроблена відповідна структура у вигляді мікросервісів. Було промодельовано сценарій використання програмного забезпечення за допомогою діаграми діяльності. На основі проаналізованих відомостей для розробки програмного забезпечення було обрано мови програмування Python, C#, TypeScript та фреймворки Jupiter, ASP.NET Core, Angular відповідно. Були розглянуті необхідні для ведення розробки інтегровані середовища розробки. Крім того, була спроектована схема бази даних та підходи до взаємодії БД та мікросервісів. Були зазначені вимоги до технічних засобів системи.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Алгоритм функціонування системи

Спираючись на структурну схему та діаграму діяльності, що були розроблені у попередньому розділі, алгоритм функціонування системи полягає у таких кроках:

- обробка даних: вибір релевантних станцій, видалення пропусків в даних, нормалізація даних, формування тренувальної та тестової вибірок;
- імплементація, тренування, тестування моделі;
- прогнозування та візуалізація майбутніх значень;
- збереження прогнозу у базу даних.
- перетворення даних з БД у зручний для візуалізації формат;
- обробка та аналіз прогнозів, обчислення ЄІЯП, виявлення топових забруднювачів та їхньої величини;
- надання прогнозів для обраної станції на задану кількість днів;
- надання карти Мадрида з позначками станцій та прогнозованої якості повітря на них;
- надання детальної інформації про прогноз обраної станції: дата, загальний індекс, найбільший забруднювач, його прогнозоване значення;
- надання можливості вибору інтервалу прогнозування.

Як було зазначено, було обрано мікросервісний підхід до створення архітектури. Тому було реалізовано 3 окремих програмних модуля, кожен з яких виконує поставлені перед ним завдання. Взаємодія між модулями здійснюється за допомогою REST API та SQL-запитів до БД. Бекенд передає запит на старт прогнозування один раз на день, а також SQL-запити на вибірку з актуальними значеннями на вимогу користувача. ПЗ для прогнозування також взаємодіє з БД та зберігає актуальні значення за допомогою SQL-запитів.

Фронтенд взаємодіє з бекендом через REST API, віддаючи запити GET та POST на керуючі блоки коду бекенду – ендпоїнти (англ. endpoint – кінцева точка).

4.2 Модуль обробки даних та прогнозування

Модуль обробки даних написаний мовою програмування Python з використанням фреймворку Jupyter у IDE PyCharm. Логічно модуль можна розділити на препроцессинг (попередню обробку) даних, роботу з моделлю та збереження результатів. Усі логічні блоки розподілені на функції, які викликаються ітеративно та синхронно, оскільки вхідними даними для наступних функцій частіше за все є результати попередніх.

Тож можна сказати, у цьому модулі була застосована парадигма функціонального програмування, оскільки створені функції є за своєю сутністю «чистими» функціями, тобто такими, які виконують тільки те, що повинні без сторонніх ефектів [35].

4.2.1 Компонент попередньої обробки даних

Компонент Preprocessing складається з таких методів:

- `imports()` – імпортує необхідні бібліотеки;
- `getDataBy(station, pollutant)` – зчитує дані про задану станцію та заданий забруднювач із загального набору даних з файлу `madrid.h5`;
- `visualizeSeries(series)` – візуалізація обраного забруднювача для обраної станції. Зокрема, на рис. 4.1 зображений графік зміни рівня озону (O_3) у Мадриді у період з 2001 по 2018 роки;
- `normalizeSeries(series)` – нормалізація даних та видалення пропусків у наборі даних;
- `getTrainSample(series, split_percent)` – формування тренувальної вибірки;
- `getTestSample(series, split_percent)` – формування тестової вибірки.

У результаті маємо підготовлений для прогнозування набір даних. Він буде використаний у подальшому дослідженні.

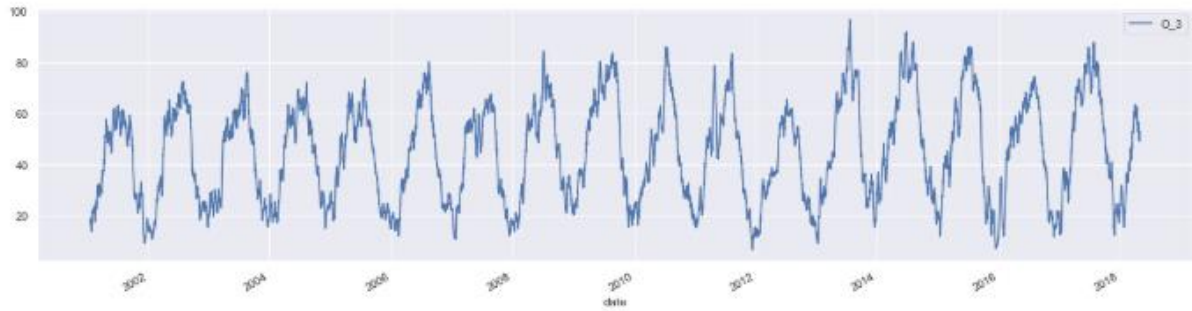


Рисунок 4.1 – Графік зміни рівня озону у м. Мадрид у 2001-2018 рр.

4.2.2 Компонент імплементції моделі та прогнозування

Компонент Forecasting імплементує спроектовану модель та виконує прогнозування. Forecasting містить такі методи:

- `createLSTM(neurons, batch_size, epochs)` – імплементція нейронної мережі зі вказаними гіперпараметрами;
- `trainModel(model, X_train, y_train)` – тренування моделі на основі тренувальної вибірки;
- `validateModel(model, X_test, y_test)` – валідація моделі на основі тестової вибірки;
- `comparing(model, series, X_train, y_train, X_test, y_test)` – візуальне порівняння справжніх значень та прогнозованих для тестової вибірки (рис. 4.2);
- `forecasting(model, series, look_back, num_prediction)` – прогнозування майбутніх значень за попередніми даними;
- `visualizeForecasting(forecast)` – візуалізація прогнозованих значень.

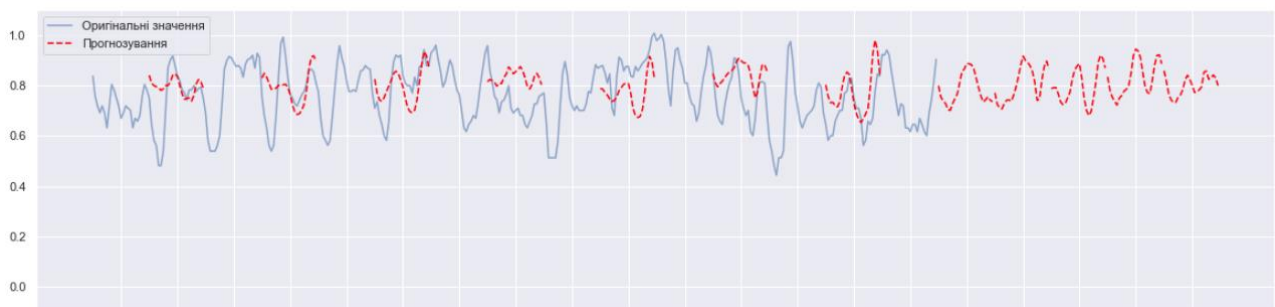


Рисунок 4.2 – Оригінальні та прогнозовані значення

4.2.3 Компонент збереження результатів

Компонент Saving зберігає графіки прогнозування та дані прогнозування у БД. Saving містить такі методи:

- savePlots (plot, stationId, pollutantId, currentNumber) – збереження графіків результатів прогнозування;
- saveToDB(forecasting) – збереження результатів прогнозування у базі даних.

4.3 Модуль серверної частини

Модуль серверної частини, або ж бекенду, написаний мовою програмування C# з використанням фреймворку ASP.NET Core у IDE Visual Studio версії Community. За своєю структурою представляє собою типовий бекенд проєкт типу WebAPI з контроллерами, сервісами, інтерфейсами та шаром доступу до БД.

4.3.1 Підпроєкт AirForecasting.Core

Використовуючи підхід database-first та інструменти ORM-фреймворку Entity Framework Core, були створені моделі, що відповідають сутностям бази даних. Вони мають абсолютно такі ж поля та зв'язки, що й у базі даних. Список C# класів відповідно до таблиць у БД має вигляд:

- Forecasting – список прогнозованих значень зі вказанням дати, станції, забруднювача, та його рівня, номеру прогнозування;
- Station – інформація про розташування станцій;
- Pollutant – інформація про забруднювачі;
- ForecastingSequenceNumber – номер прогнозування для обраного забруднювача для обраної станції.

Схема взаємозв'язків між таблицями, підключення до БД та можливість працювати з таблицями БД як з об'єктами зазначено у класі `AirForecastingContext.cs`.

4.3.2 Сервіси та програмні інтерфейси

Загалом сервіси у модулях дозволяють ще більш декомпонувати предметну область, відокремивши одні логічні частини програми від інших, що дозволяє масштабувати систему з мінімальними змінами інших блоків коду. У системі був створений сервіс `ForecastAnalysisService.cs`, основна задача якого - перетворення сирих даних прогнозів у зручний для візуалізації формат. Методи цього класу:

- `IEnumerable <ForecastViewModel> GetAnalyzedForecast (IEnumerable <Forecasting> forecastings)` – конвертує дані про прогнози у БД в список типу `ForecastViewModel`;
- `int NormalizeByEAQI(int pollutantId, double value)` – нормалізує значення забруднювачі за європейським індексом якості повітря.

Процес використання сервісів виконується за допомогою програмних інтерфейсів та підходу впровадження залежностей (англ. *dependency injection*). Він дозволяє мати список функцій, що будуть використані, незалежно від способу їхньої реалізації. Для цього у системі був створений інтерфейс `IForecastAnalysis`.

4.3.3 Контролери

Створюючи серверну частину, частіше за все створюються контролери, які надають можливість шляхом REST-запитів отримувати необхідну інформацію. У розробленій системі наявні такі контролери:

- `AirForecastController` – надає ендпоїнти для отримання оброблених сервісом даних прогнозування;

- MapController – надання інформації про розташування станцій для відображення їх на карті.

Результати запитів надаються у вигляді тексту у форматі JavaScript Object Notation.

4.4 Модуль графічного інтерфейсу

Модуль графічного інтерфейсу у веб, або ж фронтенду, написаний мовою програмування TypeScript з використанням фреймворку Angular у IDE Visual Studio версії Community. Для розмітки та стилів було використано HTML 5 та CSS 3 відповідно.

4.4.1 Модулі Angular

Архітектура Angular є компонентно-модульною, де модулі відповідають за певний логічний блок системи. У розробленій системі наявні такі модулі:

- app.module – центральний модуль системи, який є контейнером для усіх модулів, компонентів та сервісів системи;
- app-routing.module – модуль з описанням маршрутизації запитів системи.

4.4.2 Компоненти Angular

Компонент Angular є мінімальною функціональною одиницею, в якій закладена логіка системи. Зазвичай компонент складається з файлів типу TypeScript, файлу розмітки HTML та файлу стилів CSS. Крім того, до файлів типу TypeScript можуть бути додані файли з юніт-тестами. У розробленій системі наявні такі компоненти:

- home.component – домашня сторінка системи, яка виступає контейнером і містить усі інші компоненти;

- `day-selector.component` – компонент, який дозволяє обирати кількість днів для прогнозування;
- `map.component` – компонент, який візуалізує карту Мадриду;
- `forecast.component` – компонент, який надає детальну інформацію про прогноз.

4.4.3 Сервіси Angular

У Angular також є сервіси, які зберігають та обробляють інформацію між різними компонентами. У розробленій системі створений `shared-service.ts` сервіс, який оповіщає компонент `forecast`, коли у компоненті `map` була обрана станція, а також витягує актуальне значення днів з `day-selector` компоненту. Оповіщення здійснюється за допомогою використання Observable підходу, тобто коли певний блок коду є спостережуваним іншими компонентами.

Крім того, для відображення карти міста було використано можливості безкоштовного сервісу OpenStreetMap та бібліотеки Leaflet, яка надає зручні інструменти при роботі з картами. Зокрема, карта Мадриду зі вказаними станціями зображена на рис. 4.3.

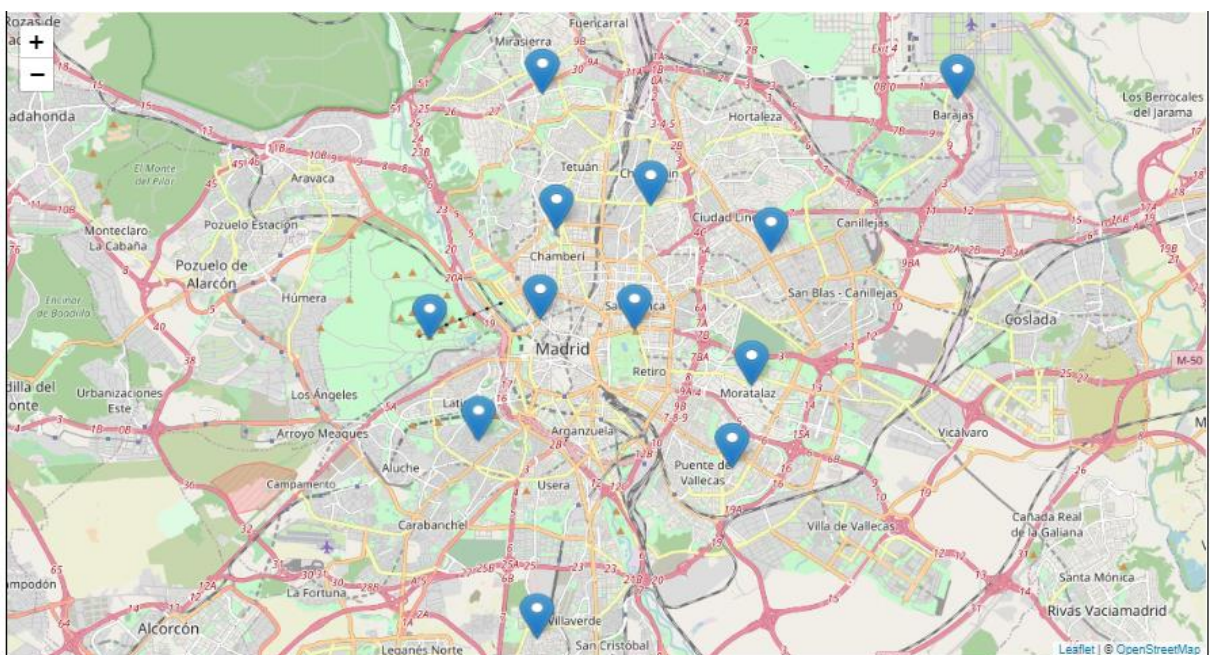


Рисунок 4.3 – Карта Мадриду із зазначеними станціями

4.5 Висновки за розділом 4

У даному розділі було розроблені компоненти системи для прогнозування рівня забрудненості атмосферного повітря. Обрано та реалізовано алгоритм функціонування системи. Розроблено 3 окремих програмних модулів. Розроблено модуль обробки даних та прогнозування з компонентами попередньої обробки даних, імплементації моделі та прогнозування, збереження результатів. Було створено бекенд та фронтенд з відповідними структурними особливостями.

У результаті була створена програмна система, яка дозволить користувачу обирати станції та отримувати прогнозовані значення для заданого часового інтервалу.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

Цей розділ буде присвячено опису розробленої програмної системи. Буде описано інструкцію запуску системи та наведено алгоритм користування. Крім того буде проведено тестування програмної системи та оцінка отриманих результатів.

5.1 Опис застосування програми

Розроблене програмне забезпечення призначене для прогнозування забруднення атмосферного повітря шляхом обробки вхідного набору даних та застосування розробленого методу. Результат прогнозування фіксується у БД, дані про які оброблюються бекендом. Користувач взаємодіє з графічним інтерфейсом фронтенду, обираючи станції та інтервал прогнозування.

5.2 Інструкція по запуску та експлуатації програмної системи

Програмна система вимагає від кінцевого користувача наявності браузера, оскільки усі обчислення та прогнозування здійснюються на сервері.

Для запуску системи системним адміністратором необхідно виконати декілька кроків, зокрема:

- розгорнути сервер для прогнозування шляхом запуску `jupyter notebook` та запустити команду `pip run forecasting.ipynb` у терміналі PowerShell у папці з проєктом AirForecasting;
- для запуску серверної частини необхідно перейти у проєкт AirForecasting.Backend та запустити команду `dotnet build forecasting.csproj` у терміналі PowerShell. Згенерований серверний застосунок можна запустити за допомогою команди `dotnet run forecasting.exe` У такому випадку за поточною IP-адресою та портом 5000 усі ендпоїнти будуть доступні.

- для запуску клієнтської частини необхідно перейти у проєкт AirForecasting.Frontend та запустити команду «ng build» у терміналі PowerShell. Проєкт запускається за поточною IP-адресою та портом 80, тобто портом за замовченням для фронтенд проєктів;
- для запуску бази даних необхідно встановити Microsoft SQL Server, перейти у проєкт AirForecasting.DB і розгорнути структуру бази. Для цього необхідно виконати команду «sqlpackage.exe publish -db mssql:forecasting».

Після закінчення встановлення необхідних частин системи кінцевому користувачеві можна використовувати систему, ввівши у пошуковому рядку браузера домен, який пов'язаний з IP-адресою, де розгорнута розроблена система.

5.3 Методика роботи з програмою

5.3.1 Результати ПЗ для прогнозування

Сутність ПЗ для прогнозування полягає у перетворенні вхідних даних у дані прогнозування для кожної станції та для кожного з п'яти основних забруднювачів. Перетворені дані зберігаються у БД, а також під час прогнозування формуються графіки, які зберігаються у вигляді рисунків, та можуть бути використані фронтендом. Для прикладу наведемо результати, що генерує система для станції з ідентифікатором 28079016 з назвою Arturo Soria. Спочатку необхідно виявити, які з даних є в наявності (рис. 5.1).

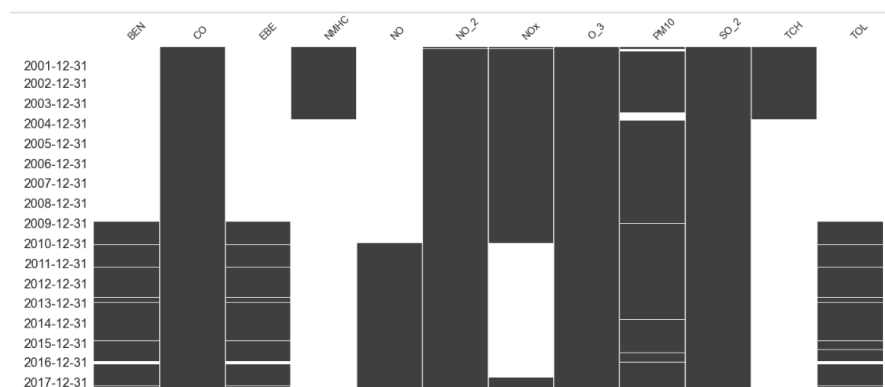


Рисунок 5.1 – Графік пропущених значень забруднювачів

На рис. 5.1 видно, що деякі значення забруднювачів з певного періоду перестали фіксувати, а інші – навпаки, почали фіксуватися з 2009 року.

Виберемо у якості прикладу забруднювач «озон». Використавши розроблений метод прогнозування, отримуємо результат, що візуалізований на рис. 5.2. У якості кількості вхідних нейронів пам'яті було обрано 120 нейронів, що дозволяє виконати прогноз на 120 днів вперед. Було порівняно прогнозування майбутніх значень з використанням розробленої архітектури нейронної мережі з обраними гіперпараметрами, а також ідентичної архітектури, але з урахуванням автомобільного трафіку.

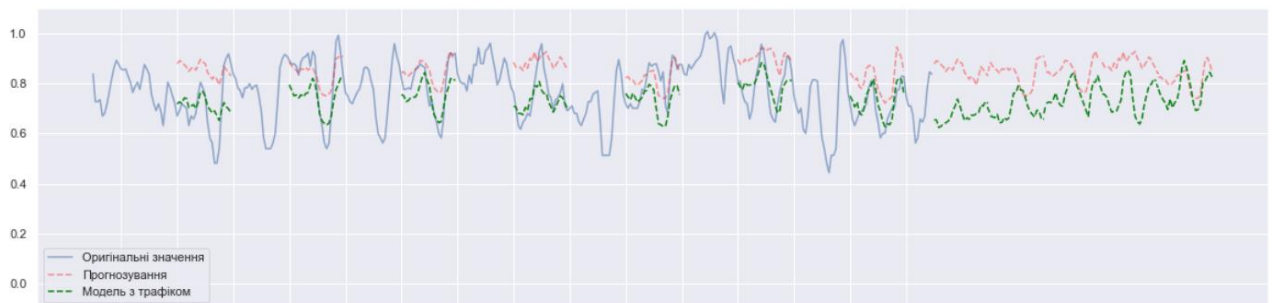


Рисунок 5.2 – Прогнозування майбутніх значень

5.3.2 Взаємодія з вебсервісом

Розроблена програмна система запускається у веббраузері за посиланням, на яке перейшов користувач. Головна сторінка сайту відображає основну мету розробки програмної системи і складається з таких компонентів:

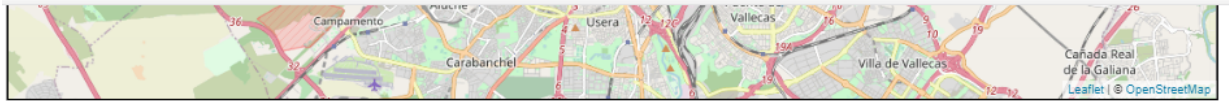
- введення користувачем днів для прогнозування;
- інтерактивна карта Мадриду з позначками станцій;
- таблиця з прогнозованими значеннями забруднювачів, що включає дату, індекс якості повітря, найбільший забруднювач та його значення у мкг/м^3 .

Графічний інтерфейс розробленої системи зображений на рис. 5.3.

Прогнозовані значення забрудненості:

Дата	Індекс якості повітря	Найбільший забруднювач	Значення забуднювача, мкг/м^3
2020-12-03T16:51:15+00:00	1	O_3	51.700828552246094
2020-12-04T16:51:15+00:00	1	O_3	60.258174896240234
2020-12-05T16:51:15+00:00	1	O_3	61.996910095214844
2020-12-06T16:51:15+00:00	1	O_3	63.335819244384766
2020-12-07T16:51:15+00:00	1	O_3	58.21730422973633
2020-12-08T16:51:15+00:00	1	O_3	56.92778396606445
2020-12-09T16:51:15+00:00	1	O_3	64.04347229003906
2020-12-10T19:03:50+00:00	3	NO_2	106.63728332519531
2020-12-11T19:03:50+00:00	2	NO_2	58.091426849365234
2020-12-12T16:51:15+00:00	1	O_3	51.04573059082031

Користувач має змогу обирати станції та кількість днів для прогнозування. Взаємодія із системою відбувається шляхом внесення змін у поле кількості днів прогнозування, а також під час натискання на мітку обраної станції. Наприклад, для станції Farrolilo результат прогнозування зображений на рис. 5.4.



Прогнозування для станції Farolillo

Прогнозовані значення забрудненості:

Дата	Індекс якості повітря	Найбільший забруднювач	Значення забруднювача, мкг/м ³
2020-12-10T19:41:09+00:00	2	NO ₂	58.00605773925781
2020-12-11T19:41:09+00:00	2	NO ₂	48.59748458862305
2020-12-12T19:41:09+00:00	2	NO ₂	58.20423889160156
2020-12-13T19:41:09+00:00	2	NO ₂	65.00354766845703
2020-12-14T19:41:09+00:00	2	NO ₂	74.47877502441406
2020-12-15T19:41:09+00:00	2	NO ₂	82.05707550048828
2020-12-16T19:41:09+00:00	2	NO ₂	93.33690643310547
2020-12-17T19:41:09+00:00	2	NO ₂	92.7013931274414
2020-12-18T19:41:09+00:00	2	NO ₂	78.09093475341797
2020-12-19T19:41:09+00:00	2	NO ₂	51.26122283935547
2020-12-20T19:41:09+00:00	2	NO ₂	50.59099578857422
2020-12-21T19:41:09+00:00	2	NO ₂	46.490692138671875
2020-12-22T19:41:09+00:00	2	NO ₂	63.403289794921875
2020-12-23T19:41:09+00:00	2	NO ₂	73.04400634765625
2020-12-24T19:41:09+00:00	2	NO ₂	86.29544830322266
2020-12-25T19:41:09+00:00	3	NO ₂	102.08235168457031

Рисунок 5.4 – Результат для обраної станції

Крім того, є можливість перевіряти минулі значення забруднювачів та виводити відповідні графіки, які генеруються ПЗ для прогнозування, зокрема як було наведено на рис. 5.2.

5.4 Методика тестування системи

Тестування системи поділено на дві частини: перша – це тестування точності моделі та показника помилки для тестової та тренувальної вибірок, а друга – це тестування серверної частини, а саме відповіді серверу на вебзапит.

5.4.1 Тестування моделі

Тестування моделі було невід’ємним процесом безпосередньо розробки ПЗ для прогнозування. Оскільки розроблений у 2-му розділі метод базується на підборі гіперпараметрів, розширенні вибірки додатковими даними та будові нейронів пам’яті, тобто фінальна модель побудована з використанням емпіричного методу – шляхом підбору оптимальних значень.

Одним з основних інструментів перевірки якості побудованої моделі є використання метрик експертної оцінки (англ. evaluation metrics). Більшість з них реалізовані у бібліотеці Keras.Metrics. Найчастіше для задач прогнозування використовуються метрики середньоквадратичного відхилення (англ. Root mean square error – RMSE) та середнього абсолютного відхилення (англ. Mean absolute error – MAE) [36]. Формула RMSE (5.1) має вигляд:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (E_i - A_i)^2}, \quad (5.1)$$

де n – кількість прогнозованих значень;

E_i – i -те прогнозоване значення;

A_i – i -те актуальне значення.

Формула MAE (5.2) має вигляд:

$$MAE = \frac{1}{n} \sum_{i=1}^n |E_i - A_i|, \quad (5.2)$$

де n – кількість прогнозованих значень;

E_i – i -те прогнозоване значення;

A_i – i -те актуальне значення.

Більше низьке значення цих двох показників відповідає кращому прогнозуванню. Зокрема, у ході експериментів з алгоритмом ARIMA та різними архітектурами нейронних мереж, у тому числі розробленого у другому розділі методу, було проаналізовано та виявлено показники зазначених метрик для різних забруднювачів та різних станцій. Наприклад, порівняння для забруднювача «озон» для станції Arturo Soria зазначено у табл. 5.1.

Таблиця 5.1 – Порівняння RMSE та MAE існуючих та розробленого методу для забруднювача «озон» для станції Arturo Soria

Метод прогнозування	RMSE	MAE
Модель ARIMA	15.33	10.95
Штучні нейронні мережі	19.10	12.87
Рекурентні нейронні мережі	14.95	10.63
Нейронні мережі типу «Довга короткострокова пам'ять»	13.87	9.13
Розроблений метод	12.71	7.22

Оскільки математичне очікування значення озону для даних обраної станції складає 80.24 мкг/м^3 , тоді точність моделі знаходиться на рівні 91%.

Результати розробленого методу можна підтверджувати різними метриками з бібліотеки Keras.Metrics. Загалом, усі метрики задаються під час компіляції моделі (рис. 5.5).

```
model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=[tf.keras.metrics.Accuracy(),
                      tf.keras.metrics.MeanAbsoluteError(),
                      tf.keras.metrics.RootMeanSquaredError()])
```

Рисунок 5.5 – Блок коду для компіляції моделі

Наприклад, метрика Ассигасу дає змогу візуалізувати точність методу зі

змiнами епох (рис. 5.6).

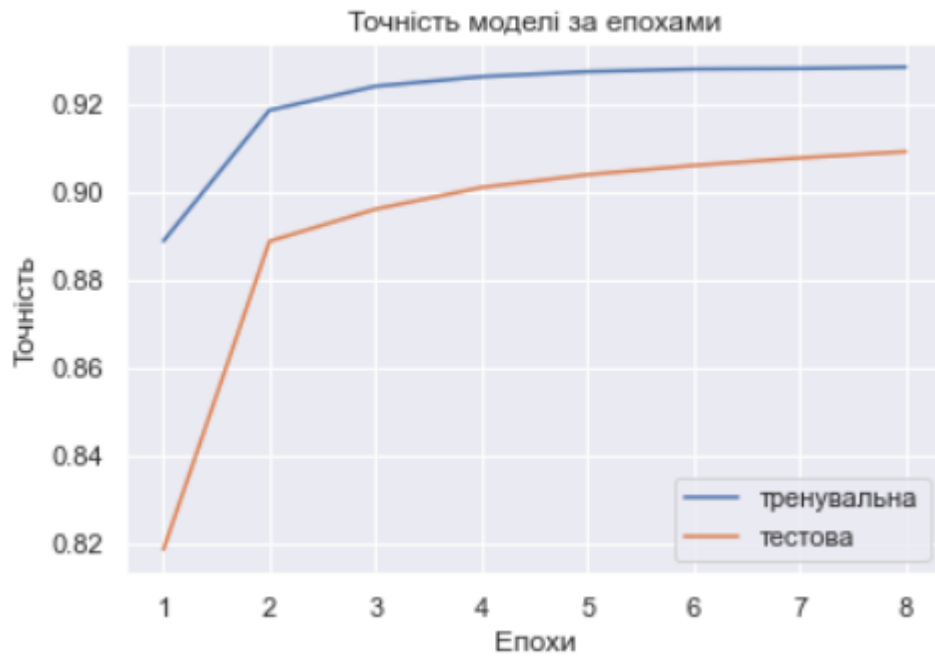


Рисунок 5.6 – Графік точності моделі за розробленим методом

Можна також оцінити графік втрат моделі за розробленими методом після етапу нормалізації (рис. 5.7).

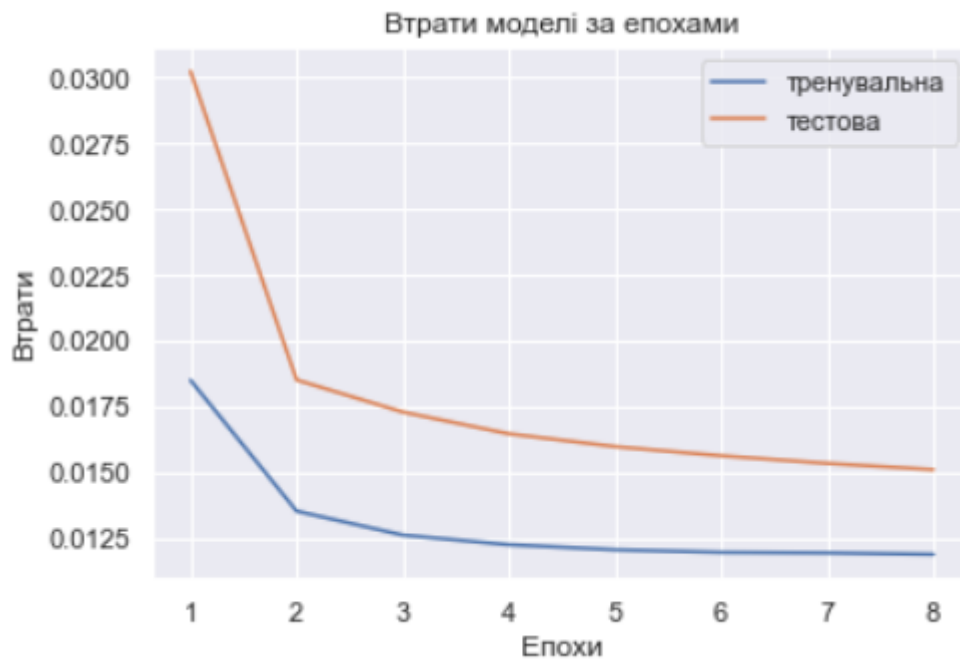


Рисунок 5.7 – Графік втрат моделі за епохами

Загалом, порівнюючи точність існуючих та розробленого методу, можна дійти висновку, що використання трафіку при побудові моделі дійсно дещо покращує точність прогнозування. Зокрема, у табл. 5.2 наведено порівняння точності моделей. Вона була обчислена для тестової вибірки як відсоток значень, спрогнозованих з похибкою не більше 0.5 мкг/м^3 .

Таблиця 5.2 – Порівняння точності прогнозування для тестової вибірки

Метод прогнозування	Точність, %
Модель ARIMA	86.35
Штучні нейронні мережі	83.96
Рекурентні нейронні мережі	86.75
Нейронні мережі типу «Довга короткострокова пам'ять» (LSTM)	88.62
Розроблений метод	90.98

Візуалізація точності прогнозу можна зробити, порівнявши істинні значення, прогноз з використанням звичайних LSTM та розроблений метод LSTM з використанням трафіку (рис. 5.8)

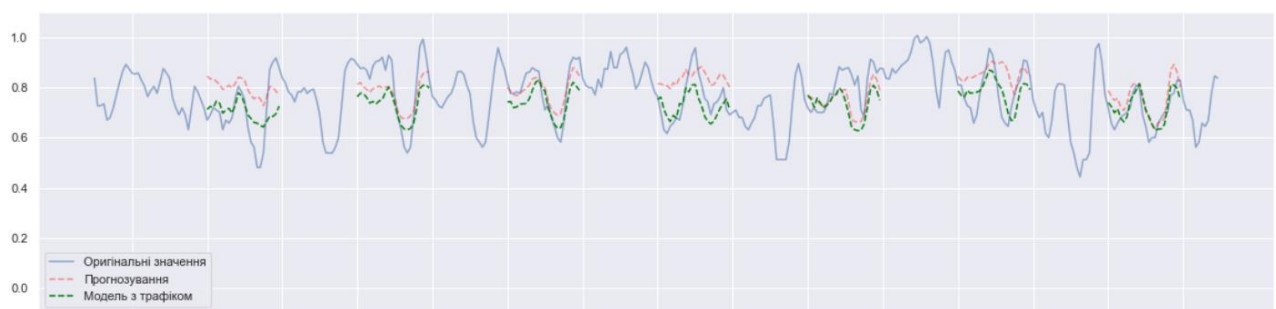


Рисунок 5.8 – Графік прогнозування методами LSTM та розробленого методу

Отже, розроблена модель узагальнює залежності вхідних даних, і придатна для прогнозування майбутніх значень, як було зображено на рис. 5.2. Проте необхідно мати на увазі, що прогнозування майбутніх значень пов'язане

з невизначеністю, оскільки кожен наступний день прогнозу буде використовувати вже спрогнозовані значення, які є лише узагальненням моделі і можуть бути далекими від стрімких змін у реальності. Для вирішення цієї проблеми у системах реального часу ПЗ для прогнозування один раз на день оновлювало би свій прогноз з урахуванням істинних значень за минулий день.

5.4.2 Тестування серверної частини

Тестування серверної частини полягає у перевірці відповідності результатів URL-запитів. Для тестування вебзастосунків з великою кількістю ендпоінтів зазвичай використовують пакет Swagger, однак бекенд розробленої системи містить невелику їх кількість, тож було обрано тестування за допомогою програми Postman. Результат запиту зображено на рис. 5.9.

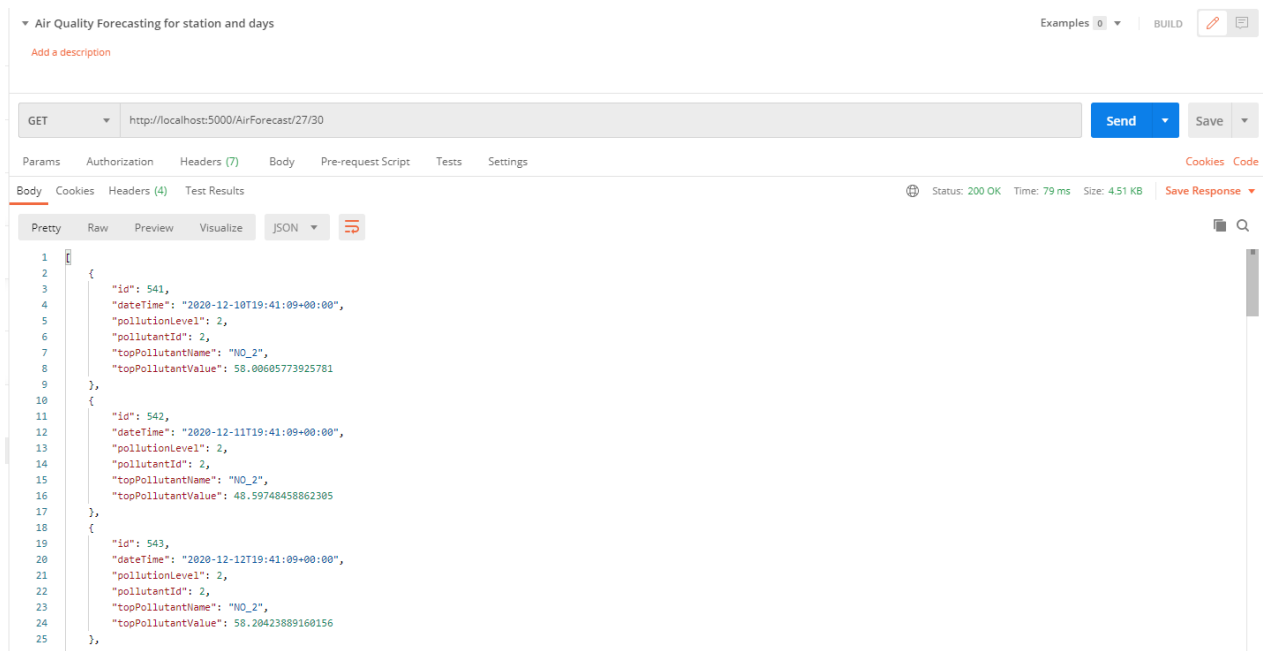


Рисунок 5.9 – Тестування серверної частини з використанням Postman

5.5 Висновки за розділом 5

У даному розділі було розглянуто опис застосування програми, було описано різні сценарії виконання програми, наведено інструкцію розгортання

програми, а також було проведено тестування розробленого методу та серверної частини. Як висновок, розроблене програмне забезпечення цілком відповідає поставленим меті розробки.

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

В цьому розділі приводиться економічне обґрунтування створення програмного забезпечення (ПЗ), оцінка затрат на його створення, розрахунок економічної ефективності та терміну окупності.

6.1 Концепція економічного обґрунтування

Метою даної дипломної роботи є дослідження та програмна реалізація методів прогнозування забруднення атмосферного повітря за допомогою машинного навчання та рекурентних нейронних мереж. Основною метою є розроблення оптимальної архітектури мережі, яка забезпечить точність прогнозів на декілька тижнів вперед. Маючи прогноз забруднення на певний період часу, люди зможуть захиститися і уникнути шкідливого впливу забруднювачів.

Розробка програмного забезпечення, яке дозволило б прогнозувати стан атмосферного повітря, могла би стати одним з важелів для місцевої влади для того, аби вони вирішували цю проблему. Наведемо приклад з містом Запоріжжя. Усім відомо, що це місто є індустріальним центром південно-східної України та містить багато заводів, зокрема промислову зону. Але якості повітря приділяється недостатня увага, через що страждає населення і має значну кількість захворювань, особливо дихальних шляхів.

Нехай у місцевої влади є курс на покращення стану навколишнього середовища міста. Тоді одним з наочних прикладів позитивних змін була би така система, що показувала б якість повітря «до» і «після». У загальному вигляді ідея полягає у тому, щоб встановити станції моніторингу якості повітря у різних районах міста, відображати дані у реальному часі та зберігати дані для подальшого їхнього аналізу та прогнозування. Далі, залучивши засоби масової інформації, необхідно привернути увагу до цієї проблеми. Дані будуть зберігатись, аналізуватись, будуть проведені роботи по покращенню якості

атмосферного повітря. І однією з функцій такої системи буде прогнозування стану повітря. За наявності прогнозів влада або приватні компанії змогли б контролювати стан та оповіщати населення. Така соціальна ініціатива приносила б значний рейтинг, оскільки би ілюструвала наочно певний прогрес у боротьбі з забрудненим повітрям.

Щодо способу прогнозування, оскільки вхідними даними є часовий ряд показників стану повітря, однозначно було обрано глибоке машинне навчання з використанням нейронних мереж. Значний розвиток інформаційних технологій дає змогу емулювати навіть такі складні процеси, як людське мислення. В основу машинного навчання закладені знання з біологічної, математичної та комп'ютерної сфер. Нейронні мережі широко використовуються для вирішення таких важливих завдань, як прогнозування, класифікація, розпізнавання образів, аналіз текстів і керування складними системами. Особливо найбільш вражаючі результати отримано при використанні РНМ.

Практична цінність роботи полягає в наданні корисної для здоров'я людини інформації щодо стану атмосферного повітря та у розроблені методу, який надає достатньо точний прогноз.

Якщо розглядати конкурентів, то можна відзначити додатки «AirVisual» та «BreezoMeter». Однак вони більше орієнтовані на американський ринок, до того ж сконцентровані на поточних значеннях якості повітря та мають невеликий діапазон щодо часу прогнозування.

Оскільки даний проєкт є більш соціальним, ніж комерційним, отримання прибутку очікується за рахунок інвестицій соціально небайдужих бізнесменів, а також шляхом публічного збору коштів за допомогою краудфандингових пожертвувань (наприклад, Kickstarter, Crowdculture, Спільнокошт тощо).

6.2 Планування розробки програмного продукту

Для розрахунку витрат необхідно визначити етапи робіт та їх тривалість. У розробці беруть участь програміст протягом 3 місяців та консультант протягом

0,25 місяця. Розробка ПЗ починається 31 серпня та повинна бути закінчена до 20 листопада 2020 року. Тривалість робіт визначають за формулою (6.1) [37]:

$$T_{\text{ц}} = \frac{Q}{R}, \quad (6.1)$$

де $T_{\text{ц}}$ – тривалість циклу, днів;

Q – трудомісткість, людино-днів;

R – кількість виконавців, людей.

У табл. 6.1 наведена трудомісткість проектування системи за етапами розробки програмного продукту.

Таблиця 6.1 – Етапи розробки продукту

Найменування роботи	Трудомісткість		Виконавці	Тривалість, дні
	люд.-дні	% до підсумку		
1. Аналіз предметної області	12	17,4%	Програміст, консультант	8
2. Узгодження результатів аналізу	2	3%	Програміст консультант	1
3. Визначення вимог до програмного продукту	3	4,3%	Програміст	3
4. Узгодження та документування вимог	4	5,8%	Програміст консультант	2
5. Дослідження методів вирішення задач	9	13%	Програміст	9
6. Проектування структури програми	11	16%	Програміст	11
7. Розробка схеми функціонування програми	4	5,8%	Програміст	4
8. Створення програмного коду	12	17,4%	Програміст	12
9. Тестування та налагодження програмного продукту	7	10%	Програміст, консультант	5
10. Складання програмної документації	5	7,3%	Програміст, консультант	4
Підсумок	69	100		59

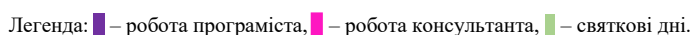


Рисунок 6.1 – Зведений стрічковий графік планування розробки програмного забезпечення

6.3 Визначення витрат на розробку

Для визначення витрат на розробку програми складається калькуляція кошторисної вартості робіт, яка включає наступні статті [37]:

- основна заробітна плата;
- додаткова заробітна плата;
- відрахування на єдиний соціальний внесок;
- матеріали і комплектуючі вироби;
- витрати на спеціальне обладнання та програмне забезпечення;
- накладні витрати.

6.4 Розрахунок основної заробітної плати

Витрати за цією статтею складаються з планового фонду зарплати всіх категорій працівників, зайнятих у розробці програми [37]. Розрахунок зарплати ведеться на підставі даних про трудомісткість, поданих у табл. 6.2.

Таблиця 6.2 – Розрахунок основної заробітної плати

Посада виконавця	Чисельність, чол.	Місячний оклад, грн	Кількість місяців роботи	Сума ЗП, грн
Програміст	1	8000	2,75	22000
Консультант	1	9500	0,5	4750
Підсумок	2			26750

6.5 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати працівників і розраховують за формулою (6.2) [37]:

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 0,1. \quad (6.2)$$

Після підстановки величини основної заробітної плати в дану формулу, отримуємо:

$$ЗП_{\text{дод}} = 26750 \cdot 0,1 = 2675 \text{ грн.}$$

6.6 Відрахування на єдиний соціальний внесок

Відрахування на єдиний соціальний внесок (ЄСВ) становлять 22% і беруться від основної та додаткової заробітної плати. Вони розраховуються за формулою (6.3) [37]:

$$ВТ_{\text{соц}} = (ЗП_{\text{осн}} + ЗП_{\text{дод}}) \cdot 0,22, \quad (6.3)$$

де $ЗП_{\text{осн}}$ – основна заробітна плата (грн.),

$ЗП_{\text{дод}}$ – додаткова заробітна плата (грн.) [37].

В результаті отримуємо:

$$ВТ_{\text{соц}} = (26750 + 2675) \cdot 0,22 = 6473,5 \text{ грн,}$$

Отже, значення ЄСВ становить 6437,50 грн.

6.7 Визначення витрат на матеріали

Витрати на матеріали розраховуються за формулою (6.4) [37]:

$$ВТ_{\text{мат}} = \sum_{i=1}^n (Ц_i \cdot N_i \cdot (1 + K_{\text{т.з}}) - Ц_{io} \cdot N_{io}), \quad (6.4)$$

де $ВТ_{\text{мат}}$ – це витрати на матеріали, покупні напівфабрикати та комплектуючі вироби, грн.;

C_i – ціна i -го найменування матеріалу, напівфабрикату і комплектуючого, грн;

N_i – потреба в i -му матеріалі, напівфабрикаті і комплектуючому, кількість;

$K_{т.з}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

C_{io} – ціна зворотних відходів i -го найменування матеріалу, грн;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих [37].

Витрачені матеріали не містять зворотних відходів, тому $C_{io} = 0$ та $N_{io} = 0$. Зазначимо коефіцієнт транспортних витрат $K_{т.з} = 0.05$. Витрати на матеріали для розробки наведені в табл. 6.3.

Таблиця 6.3 – Витрати на матеріали

Найменування, вид, тип, марка	Одиниця виміру	Ціна за одиницю вим., грн.	Кількість од. вим.	Сума витрат, грн.
Картридж для принтера	шт.	350	1	350
Папір формату А4 для принтера	листів	0,17	500	85
USB Flash накопичувач об'ємом 16Gb	шт.	1	1	140
Всього				575

$$ВТ_{\text{мат}} = (1 + 0,05) \cdot (350 + 85 + 140) = 603,75 \text{ грн.}$$

Сумарна вартість матеріалів становить 603,75 грн.

6.8 Витрати на спеціальне обладнання

У цю статтю витрат входять витрати на придбання, транспортування, монтаж і налагодження обладнання. Також необхідно розрахувати витрати на оплату машинного часу (амортизаційні витрати на обслуговування та вартість спожитої електроенергії). Амортизаційні відрахування розраховуються за формулою (6.5) [37]:

$$A = \Phi_6 \cdot \frac{H_a}{100}, \quad (6.5)$$

де Φ_6 – балансова вартість обчислювальної техніки, грн;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, % [37].

Основною робочою машиною є ноутбук Lenovo IdeaPad L340 у такій конфігурації: Intel Core i7-9750H, 16 ГБ ОЗУ, 512 SSD, nVidia GeForce GTX 1050. Виходячи з даних, наданих виробником обладнання, а так само знятими системними показниками можна припустити, що середнє енергоспоживання ноутбука при подібній проєктній роботі складає 200 Ватт. Початкова ціна робочої машини – 28500 грн. Разом з принтером PIXMA Ink Efficiency вартістю 1800 грн загалом вартість обладнання сягає 30300 грн. Норма амортизації для ноутбуків становить 25%. Тоді отримаємо амортизаційні відрахування:

$$A = 30300 \cdot 0,25 = 7575 \text{ грн.}$$

Амортизаційні відрахування становлять 7 575 грн.

Для розрахунку вартості експлуатації обладнання використаємо формулу (6.6) [37]:

$$C_e = N_n \cdot \Phi_{\text{еф}} \cdot K_{\text{зч}} \cdot K_{\text{зп}} \cdot C_{\text{ел.е}}, \quad (6.6)$$

де N_n – номінальна потужність ЕОМ, кВт;

$\Phi_{\text{еф}}$ – річний ефективний фонд часу роботи ЕОМ, машино-год;

$K_{\text{зч}}$ – середній коефіцієнт завантаження по часу;

$K_{\text{зп}}$ – коефіцієнт завантаження по потужності;

$C_{\text{ел.е}}$ – ціна одного кВт*год електроенергії, грн/(кВт*год) [37].

Річний фонд часу роботи персонального комп'ютера працівника становить 1750 годин. Середні коефіцієнти завантаження за часом та за

потужністю становлять 0.9 та 0.6. Ціна однієї кіловат-години електроенергії становить 2,11 грн. Отримуємо ціну експлуатації обладнання:

$$C_e = 0,2 \cdot 1750 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 398,79 \text{ грн.}$$

Зарплата обслуговуючого персоналу розраховується за формулою (6.7) [37]:

$$ЗП_{\text{обсл}} = \Phi ЗП_p \cdot (1 + K_{\text{відр}}) \cdot \frac{t_{\text{обсл}}}{\Phi_{\text{еф.обсл}}}, \quad (6.7)$$

де $\Phi ЗП_p$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн;

$K_{\text{відр}}$ – коефіцієнт відрахувань на соціальне страхування і до інших фондів;

$t_{\text{обсл}}$ – час протягом року, необхідний на технічне обслуговування ЕОМ, год/рік;

$\Phi_{\text{еф.обсл}}$ – річний ефективний фонд часу обслуговуючого персоналу, год/рік [37].

Місячна заробітна плата обслуговуючого персоналу становить 4723 грн., а річний фонд заробітної плати становить 56676 грн. Річний ефективний фонд робочого часу обслуговуючого ПК робітника становить 1750 годин/рік. На обслуговування одного ПК витрачається по 1 години на місяць, що на рік становить 12 годин. З чого виходить, що:

$$ЗП_{\text{обсл}} = 56676 (1 + 0,22) \cdot \frac{12}{1750} = 474,14 \text{ грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання і складає 909 грн.

Стаття «Інші витрати» приймається рівною 5% від суми всіх попередніх статей витрат на утримання та експлуатацію устаткування [37]. Сума всіх попередніх статей дорівнює 9356,93 грн., 5% від суми становлять 467,85 грн.

Кошторис витрат на утримання та експлуатацію обладнання описано у табл. 6.4.

Таблиця 6.4 – Кошторис витрат на утримання та експлуатацію обладнання

Найменування статей витрат	Сума, грн.
Амортизація обладнання	7575
Експлуатація обладнання (крім витрат на поточний ремонт)	398,79
Заробітна плата основна та додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	474,14
Поточний ремонт обладнання	909
Інші витрати	467,85
Разом	9815,78

Витрати на оплату машинного часу ЕОМ для написання й відлагодження програми вираховується за формулою (6.8) [37]:

$$B_{\text{мч}} = B_{\text{екс}} \cdot t_{\text{мч}}, \quad (6.8)$$

де $B_{\text{мч}}$ – витрати на сплату машинного часу (грн.),

$B_{\text{екс}}$ – експлуатаційні витрати на один час машинного часу ЕОМ,

$t_{\text{мч}}$ – машинний час ЕОМ для написання та відлагодження програмного засобу [37].

Експлуатаційні витрати на одну годину машинного часу використовуваної ЕОМ розраховують діленням суми витрат за кошторисом «Витрати на утримання та експлуатацію обладнання» (табл. 6.5) на річний ефективний фонд часу роботи ЕОМ. Річний ефективний фонд часу роботи ЕОМ дорівнює 1750 годин. В результаті експлуатаційні витрати на одну годину машинного часу рівні:

$$B_{\text{екс}} = \frac{9836,73}{1750} = 5,62 \text{ грн/машино} - \text{год.}$$

ЕОМ експлуатується 59 днів в одну зміну, що становить у сумі 472 години. Таким чином витрати на сплату машинного часу становлять:

$$B_{\text{мч}} = 5,62 \cdot 472 = 2652,64 \text{ грн.}$$

6.9 Інші прямі витрати

До інших прямих витрат включаються витрати на використовуване при розробці системи комерційне ПЗ:

– пайове ПЗ, що використовується постійно при роботі ПК (Windows 10 Professional) – 5 620 грн. без податку на додану вартість (ПДВ);

– цільове ПЗ, яке необхідно для даної конкретної задачі (PyCharm) – 5 500 грн. без ПДВ, витрати за яким обчислюються за формулою (6.9);

– цільове ПЗ, яке необхідно для даної конкретної задачі (VSCode) – 0 грн. без ПДВ, витрати за яким обчислюються за відповідними формулами (6.9) і (6.10):

$$S_{\text{пай.ПО}} = \frac{C_{\text{ПОWindows}} \cdot T_{\text{КТС}}}{\Phi_{\text{еф.КТС}} \cdot T_{\text{с.ПО}}}, \quad (6.9)$$

$$S_{\text{ціль.ПО}} = C_{\text{ПОPyCharm}} + C_{\text{ПОWebStorm}}, \quad (6.10)$$

де $S_{\text{пай.ПО}}$ – витрати на пайове ПЗ при розробці ПЗ з розрахунків, грн;

$S_{\text{ціль.ПО}}$ – витрати на цільове ПЗ, що купується виключно для розробки ПЗ з розрахунків, грн;

$C_{\text{ПОWindows}}$ – ціна за Windows (без ПДВ), грн;

$C_{\text{ПОPyCharm}}$ – ціна за PyCharm (без ПДВ), грн;

T_{KTC} – машинний час КТС, необхідний користувачеві для розробки системи, машино-год/рік;

$\Phi_{\text{еф.КТС}}$ – річний ефективний фонд часу роботи КТС, машино-год/рік;

$T_{\text{с.ПО}}$ – термін служби пайового ПЗ, років.

Тоді можна виконати наступні розрахунки:

$$S_{\text{пай.ПО}} = \frac{5620 \cdot 472}{1750 \cdot 5} = 303,16,$$

$$S_{\text{цїл.ПО}} = 5500 + 0 = 5500,$$

$$S_{\Sigma} = 303,16 + 5500 = 5803,16.$$

6.10 Розрахунок накладних витрат

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т. д.), утримання та експлуатацію будівель [37].

Накладні витрати включаються у вартість розробки програми непрямым шляхом – у відсотках до основної заробітної плати розробників. В даному випадку накладні витрати складають 40% до основної заробітної плати розробників, що становить 10700 грн.

Загальна кошторисна вартість розробки складається з основної та додаткової заробітної плат, відрахування на соціальне страхування, вартості матеріалів, витрат на спеціальне обладнання, накладних та інших прямих витрат, а також податку на додану вартість [37].

Результати визначення витрат на розробку програми у вигляді калькуляції кошторисної вартості робіт наведені в табл. 6.5.

Таким чином, вартість проєкту оцінюється в 66746,46 грн.

Таблиця 6.5 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн.	Питома вага до підсумку, %
Основна заробітна плата	26750	48,1
Додаткова заробітна плата	2675	4,81
ЄСВ	6437,5	11,57
Вартість матеріалів та комплектуючих	603,75	1,09
Витрати на спец. обладнання	2652,64	4,77
Інші прямі витрати	5803,16	10,43
Накладні витрати	10700	19,23
Разом	55622,05	100
ПДВ	11124,41	
Разом (з ПДВ)	66746,46	

6.11 Розрахунок економічної ефективності програмного продукту

Ефективність прикладних науково-дослідних робіт визначається, як відношення річного економічного ефекту від застосування результатів розрахункових досліджень в умовах виробництва і використання капіталовкладень для здійснення втілення їх у виробництво. Ефективність обчислюється за формулою (6.11) [37]:

$$E_{\text{фак}} = \frac{E_{\text{к.еф}}}{K}. \quad (6.11)$$

Зіставляючи фактичну ефективність з нормативною ($E_{\text{н}} = 0,1 \dots 0,15$) робиться висновок про доцільність досліджень. Якщо $E_{\text{фак}} \geq E_{\text{н}}$, то дослідження визнаються економічно ефективними. У іншому випадку виконання дослідження не є доцільним, оскільки економічна вигода від таких досліджень мала або незначна. Економічну ефективність досліджень можливо оцінити за терміном окупності витрат на їх проведення та втілення. Фактичний термін окупності обчислюється за формулою (6.12) [37]:

$$T_{\text{ок}}^{\text{фак}} = \frac{K}{\text{Ек. еф.}}. \quad (6.12)$$

Фактичний термін окупності капіталовкладень зіставляється з нормативним терміном окупності капіталовкладень, прийнятий для промисловості (приблизно 6 років), та на основі порівняння робиться висновок про доцільність досліджень [37].

Для теоретичних досліджень в більшості випадків важко або навіть неможливо розрахувати економічний коефіцієнт, тому доцільно визначити їх техніко-економічну ефективність з урахуванням наступних показників [37]:

- важливість дослідження;
- складності розробки;
- результативності та можливості використання.

Важливість теоретичного дослідження оцінюють за його призначенням:

- вирішення проблемних питань;
- задоволення вимог спеціальної техніки;
- пошук принципово нових конструкцій і технологічних рішень.

Складність виконання роботи визначають порівнянням отриманих результатів даного дослідження з результатами відомих аналогічних досліджень з обліком грошових і трудових витрат на їх проведення [37].

Результативність роботи можна визначити за повнотою рішень поставленого завдання: отриманий результат відповідає плановому, задовільний (часткове рішення) або негативний [37].

Аналіз залежності між цими показниками та витратами на їх досягнення дає можливість кількісної оцінки техніко-економічної ефективності теоретичних робіт за формулою (6.13) [37]:

$$K_{\text{НДР}} = \frac{J^n \cdot R \cdot T}{B_{\text{НДР}} \cdot t_{\text{НДР}}}, \quad (6.13)$$

де $K_{\text{НДР}}$ – рівень ефективності дослідження (коефіцієнт техніко-економічної ефективності роботи);

J – важливість роботи;

n – показник використання результатів роботи;

R – результативність роботи;

T – технічна складність виконання роботи;

$B_{\text{НДР}}$ – витрати на проведення роботи;

$t_{\text{НДР}}$ – час проведення НДР [37].

При $K_{\text{НДР}} \geq 1$ науково-дослідна робота вважається ефективною. Оскільки час на проведення роботи склав 59 днів, це складає 0.16 від одного року. Маємо:

$$K_{\text{НДР}} = \frac{2,5^2 \cdot 1 \cdot 2,5}{66,74 \cdot 0,16} = 1,46.$$

Після розрахунку отримуємо рівень ефективності дослідження вищий за 1, тому можна зробити висновок, що дана дослідницька робота є ефективною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

У сучасному світі розвиток інформаційних технологій зумовлює використання значною кількістю працівників комп'ютерів або іншої обчислювальної техніки як основного засобу праці. Крім того, неочікувана пандемія коронавірусу у 2020 році призвела до перенесення багатьох бізнесів та сфери навчання у дистанційний режим. Тому, працюючи з технікою, необхідно бути усвідомленими щодо потенційних небезпек та уникати їх.

Розробка програмного забезпечення зокрема частіше за все ведеться у офісах, де працівники оточені великою кількістю технічних пристроїв, кожен з яких створює випромінювання, яке може бути шкідливим для людини [38]. Неякісна організація робочого місця, незручне крісло та робочий стіл можуть спричиняти фізичний дискомфорт під час роботи, який у майбутньому може призводити до проблем зі спиною, шиєю, зором та й загалом із здоров'ям людини.

Проводження значного часу з технікою, розумове та емоційне перевантаження, що виникає внаслідок неправильного тайм-менеджменту може призводити до зниження працездатності людини, хронічної стомленості або підвищеного психофізичного навантаження. Результатом такого може стати професійне «вигорання», коли внаслідок надто напруженого графіку людина цілком втрачає інтерес та мотивацію до роботи.

Недостатня освітленість, забруднене повітряне середовище, сторонні шуми та вібрації, випромінювання при невиконанні вимог до використання обладнання та організації робочого місця також можуть призводити до порушення фізичного та психологічного стану працівника [38].

Оскільки технічні засоби працюють зі струмом, не виключена можливість виникання коротких замикань [38]. Вони можуть призводити до виникнення пожеж, при яких за умови відсутності необхідних засобів пожежогасіння можуть бути значні збитки, а також небезпека для життя та

здоров'я працівників. Крім того, у більшості випадків, навіть якщо пожежу вдалось вчасно загасити, така ситуація все одно призводить до несправності обладнання та збитків [38].

Загалом, про наявність потенційних небезпек працівники мають бути усвідомлені. Тому що за умови невідповідності персоналу при появі певних надзвичайних ситуацій може бути неправильне або недостатнє реагування на виникненні ситуації, що буде призводити до небажаних наслідків [38].

7.2 Заходи із забезпечення безпеки

Програмне забезпечення, що розроблюється для прогнозування забруднення атмосферного повітря, передбачає роботу у приміщенні невеликої команди розробників. Розміри приміщення – $8 \cdot 5 \cdot 3 \text{ м}$.

Загалом площа приміщення складає $8 \cdot 5 = 40 \text{ м}^2$, а об'єм – $8 \cdot 5 \cdot 3 = 120 \text{ м}^3$.

Розглядаючи роботу у такому приміщенні, найбільшу небезпеку для розробників створює електричний струм. Тому забезпечення як організаційних, так і технічних заходів щодо роботи технічного обладнання відіграє ключове значення для забезпечення безпеки розробників [38].

Згідно з [39], з метою попередження можливих травмувань, з усіма працівниками, прийнятими на роботу, а також у період роботи проводяться навчання, інструктажі з питань охорони праці (вступний, первинний, повторний, позаплановий, цільовий). Працівники підприємств при прийнятті на роботу і періодично в процесі роботи, а вихованці, учні і студенти під час навчально-виховного процесу повинні проходити навчання і перевірку знань згідно з вимогами цього Типового положення. Допуск до роботи (виконання навчальних практичних завдань) без навчання і перевірки знань з питань охорони праці забороняється. Працівники мають знати, як надавати першу медичну допомогу потерпілим від нещасних випадків, а також бути усвідомлені про правила поведінки під час виникнення аварій, пожеж та правил

електробезпеки. Формою перевірки знань з питань охорони праці працівників є іспит, який проводиться за екзаменаційними білетами у вигляді усного опитування або шляхом тестування на автоекзаменаторі з наступним усним опитуванням [39].

Кожний комп'ютер має власний блок живлення. Для перетворення змінного струму мережі у постійний комп'ютери використовують блоки живлення (5 В, 12 В), потужністю 350 – 1000 Вт. Оскільки усі прилади при підключенні до електричної мережі яка заземляється, використовують систему TN, тому обчислювальні пристрої у приміщенні відносяться до категорії І захисту від ураження електричним струмом. Аби уникнути ураження корпуси всіх електричних пристроїв виготовляються з неструмопровідних матеріалів, а живлення здійснюється спеціальним кабелем [40].

Для забезпечення безпечності у приміщенні всі кабелі сховані в спеціальні коробки, розетки розташовані на висоті одного та 0,3 метри від підлоги, електрична мережа вмикається і вимикається за допомогою пускової апаратури (рубильником). Відповідно до міжнародного стандарту СЕ зазначені міри електробезпеки відповідають вимогам, пропонованим до побутових приладів [40].

Опір захисного заземлення в приміщенні становить 3 Ом (за стандартом – не більше 4 Ом) згідно з "Правилами улаштування електроустановок" (ПУЕ). Крім того, у приміщенні також встановлені автоматичні вимикачі, пристрій заземлення контуру згідно з [40].

Для підготовки робочого місця до роботи, яка вимагає зняття напруги, слід вжити у вказаному порядку таких технічних заходів:

- здійснити потрібні відключення і вжити заходів, що перешкоджають помилковому або самочинному ввімкненню комутаційної апаратури [40];
- вивісити заборонні плакати на приводах ручного і на ключах дистанційного керування комутаційною апаратурою [40];

- перевірити відсутність напруги на струмовідних частинах, які слід заземлити для захисту людей від ураження електричним струмом [40];
- встановити заземлення (ввімкнути заземлювальні ножі, встановити переносні заземлення) [40];
- обгородити, за необхідності, робочі місця або що залишилися під напругою, і вивісити на огороженнях плакати безпеки [40].

Для запобігання ураження людини струмом в розглянутому приміщенні використовується захисне заземлення, окремі розетки по 220 В та мережевий фільтр Xiaomi Mi Power Strip, а також ізолююча – лінолеум підлога.

Оскільки робоче приміщення обладнане персональними комп'ютерами з візуальними дисплейними терміналами (моніторами), то повинні виконуватися ергономічні норми згідно з [41], такі як:

- забезпечення зручності використання за призначенням;
- опанованість обладнання, обслугованість обладнання, гігієнічність обладнання і середовища робочої зони;
- безпечність обладнання;
- композиційна організованість елементів обладнання, відповідність кольорофактурного вирішення обладнання специфіці роботи людини та загальному колірному вирішенню середовища, в якому передбачено експлуатування цього обладнання [41].

Крім того, конструкція виробничого обладнання повинна забезпечувати такі фізичні навантаження на того, хто працює, за яких енергозатрати організму впродовж робочої зміни не перевищували б 1046,7 кДж/год (250 ккал/год). Конструкція виробничого обладнання повинна забезпечувати можливість організації трудового процесу, яка б мінімізувала монотонність праці обмеженням повторень простих трудових дій, їх кількості, а також тривалості безперервного пасивного спостереження за перебігом виробничого процесу або його частини [41].

За правилами людинно-машинної взаємодії, робота людини з технікою має бути безпечною і простою. Яскравість та контрастність екрану має

сприйматися легко, символи на екрані мають бути чіткими, не має бути відблисків, блимання, екрани мають бути чистими [41].

Для роботи у нічний період часу багато програм та інтегрованих середовищ розробки надають дизайн «темної» теми, що дозволяє зменшити шкідливий вплив надмірної яскравості екранів.

7.3 Заходи із забезпечення виробничої санітарії та гігієни праці

Заходи щодо забезпечення виробничої с санітарії та гігієни праці для приміщення, обладнаного ПК з ВДТ були розроблені у відповідності з вимогами [42]-[43].

Щодо показників мікроклімату, оскільки робота виконується розробниками сидячи за постійним місцем роботи, вони мають бути оптимальними. Працівники не отримують значного фізичного напруження за умови дотримання норм і виконують поставлені завдання в незалежності від пори року [42].

За значенням енерговитрат робота з комп'ютером відноситься до легкої категорії, але все ж потребує певної розумової діяльності і потребує в середньому 140-270 ккал/год енерговитрат [43].

Для забезпечення оптимального рівня параметрів повітряного виробничого середовища використовуємо [44]. В такому випадку норми мікроклімату в робочому приміщенні складають:

- у холодний період року: 22-24 °C температури повітря, 40-60% відносної вологості повітря та 0.1 м/с швидкості руху повітря;
- у теплий період року: 23-25 °C температури повітря, 40-60% відносної вологості повітря та 0.1-0.2 м/с швидкості руху повітря.

Для забезпечення оптимальної температури в холодний період року відповідно до [45] в приміщенні передбачено устрій системи водяного опалення.

Відповідно до вимог [45], у теплий період року, навпаки необхідно охолоджувати температуру приміщення, тому для підтримки нормальної температури використовується касетний кондиціонер. Його перевага над звичайним кондиціонером типу «спліт-система» у рівномірному розподілі повітряного потоку по чотирьох напрямках, що дозволяє використовувати всього один кондиціонер касетного типу для охолодження робочого приміщення (при використанні настінних спліт-систем для досягнення аналогічного ефекту довелося б використовувати 2-3 кондиціонери меншої потужності) [45].

Відповідно до вимог [46] освітлення у робочому приміщенні комбіноване (суміщене), оскільки недостатнє природне освітлення доповнюється штучним.

Для забезпечення відносної постійності природного освітлення вікна обладнанні сонцезахисними металевими жалюзі з системою регулювання, в яких коефіцієнт відбиття становить 0,7 – 0,8. Робочі місця розташовані так, аби у поле зору не потрапляли вікна або світлі поверхні світильників [46].

Для робочого приміщення з ПК найбільш прийнятними є люмінесцентні лампи ЛБ (білого світла) і ЛТБ (тепло-білого світла) потужністю 40-80 Вт [46].

Розрахуємо кількість ламп необхідних для освітлення приміщення розміром 8 · 5 · 3м. Скористаємося методом використання світлового потоку.

Площа приміщення обчислюється за формулою (7.1) [46]:

$$S = a \cdot b = 8 \cdot 5 = 40 \text{ м}^2, \quad (7.1)$$

Зазначимо нормовану мінімальну освітленість як $E = 300$ лк.

За вимогами [46] коефіцієнт природної освітленості (КПО) у робочому приміщенні має становити в середньому 1,1% (1,2% – у сонячний день та 1% – у хмарну погоду). Тому коефіцієнт нерівномірності освітлення, що розраховується як відношення середньої освітленості до мінімальної, обчислюється за формулою (7.2) [46]:

$$Z = \frac{E_{\text{ср}}}{E_{\text{мін}}} = \frac{1.1}{1} = 1.1. \quad (7.2)$$

Коефіцієнт відбиття стелі 70% і стін 50% [46]. Для освітлення використовуються люмінесцентні лампи типу ЛТБ в світильниках ЛПО. Коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації приймаємо $K = 1.5$ [46].

Необхідно обчислити коефіцієнт використання світлового потоку η . Його знаходять у таблиці за значенням індексу приміщення. Тож за формулою (7.3), наведеною у [46] знаходимо індекс приміщення:

$$i = \frac{I_n b}{H(I_n + b)} = \frac{8 \cdot 5}{3(8 + 5)} = 1.025. \quad (7.3)$$

При індексі $i = 1,025$ отримаємо коефіцієнт використання світлового потоку 39%. Світильники будемо розміщувати у 2 ряди, тож $N_p = 2$) [46].

За отриманими вище значеннями є можливість розрахувати необхідний світловий потік ламп в кожному ряду за формулою (7.4) [46]:

$$\Phi_p = \frac{E \cdot S \cdot Z \cdot K}{N_p \cdot \eta} = \frac{300 \cdot 40 \cdot 1,1 \cdot 1,5}{2 \cdot 0,39} = 25384 \text{ лм}. \quad (7.4)$$

Якщо в світильнику встановити по дві лампи ЛБ 40-1, потужністю 40 Вт, світловий потік яких $F_{\text{л}} = 3200$ лм. то необхідне число світильників в ряду обчислюється за формулою (7.5) [46]:

$$N_{\text{св}} = \frac{F}{n \cdot F_{\text{л}}} = \frac{25384}{2 \cdot 3200} = 4. \quad (7.5)$$

Таким чином для освітлення приміщення використовуємо 8 світильників, встановлених по 4 у кожному з двох рядів.

Для виключення засвічення екранів дисплеїв прямими світловими потоками світильники загального освітлення будуть розташовуватися збоку від робочого місця, паралельно лінії зору програміста та стіни з вікнами.

На роботу розробників впливає шум, при наявності якого знижується концентрація уваги, з'являється стомленість у зв'язку з нервово-психічною напругою та підвищеними енергетичними витратами [47].

Згідно вимог [47] рівень шуму у робочому приміщенні для комфортної роботи повинен коливатися у межах 35-50 акустичних децибелів (дБА). Згідно із технічним паспортом обладнання (ПК, монітори, принтери, клавіатури (при натисканні), маніпулятори типу «миша» (при натисканні)), що використовується для роботи, рівень шуму складає: 45 дБА. Згідно із ДСН 3.3.6-039-99 нормоване значення рівня шуму більше, ніж фактичне, тож додаткових робіт для усунення надмірного шуму не проводилося. Роботи в даному приміщенні з таким обладнанням відповідають нормативним вимогам [47].

Під час роботи із дисплеєм моніторів, на розробників здійснюється інтенсивне електромагнітне випромінювання, яке може стати причиною професійних захворювань. Тож для роботи використовуються монітори з рідкокристалічними дисплеями.

Організація та обладнання робочого місця в робочому приміщенні має відповідати ергономічним вимогам з урахуванням особливостей трудової діяльності.

У робочу приміщенні можуть працювати одночасно до 4 чоловік. Кожний робітник має власний:

- робочий стіл;
- обчислювальний пристрій, пристрою вводу (клавіатура, миш та інше), виводу (дисплей монітору, принтер, звукова колонка, навушники та інше) та набір периферійних пристроїв (принтер, сканер та інше);

– стілець.

Також кожний робітник може скористатися загальною шафою-купе для одягу та шафою з полицями для інших речей.

Площа на одне робоче місце з ПК складає по 10 м², а об'єм – по 30 м³. Приміщення має природне та штучне освітлення. Робочі місця розташовано так, щоб природне світло падало збоку.

Робочі місця та взаємне розташування всіх його елементів відповідати антропометричним, фізичним і психологічним вимогам.

Робочі стільці мають підйомно-повторні механізми та регулюються по висоті та куту нахилу сидіння і спинки, а також по відстані спинки від переднього краю сидіння. Регулювання кожного параметра незалежне, легко здійснюється та має надійну фіксацію. Екран монітору повинний знаходитися на оптимальній відстані 700 мм.

Ергономічність конструкції відіграє велику роль, це важливо підтримки оптимальної робочої пози розробника. Оскільки робота з комп'ютером характеризується високою напруженістю зорової роботи, значною розумовою напругою і нервово-емоційним навантаженням, а також достатньо великим навантаженням на м'язи рук при роботі з клавіатурою.

Кожна клавіатура має можливість міняти висоту і нахил, всі килимки для миші мають захист від тунельного синдрому, а кожний стілець має підлокітник.

Крім того, робоче приміщення містить кондиціонер, два ряди світильників, а також обладнане зволожувачами повітря.

Загальний план обчислювального центру та робочих місць наведено на рис. 7.1.

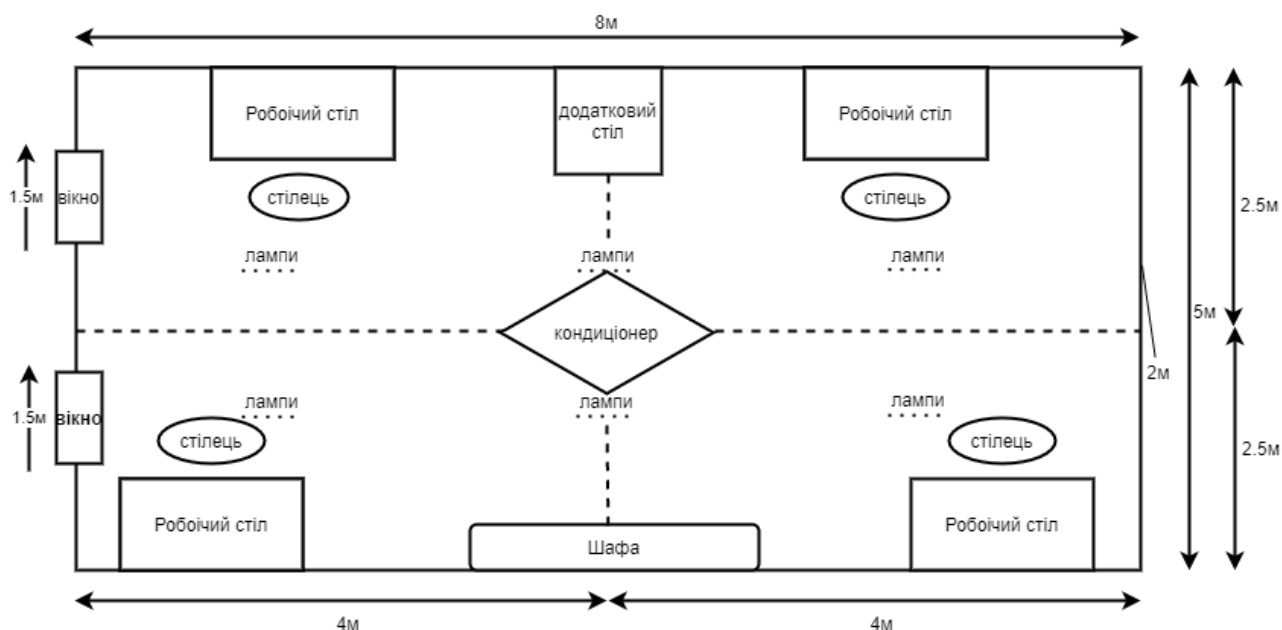


Рисунок 7.1 – Загальна схема робочого приміщення

7.4 Заходи безпеки у надзвичайних ситуаціях

7.4.1 Заходи з пожежної безпеки

Пожежі в робочому приміщенні становлять особливу небезпеку, тому що пов'язані з великими матеріальними втратами. Як відомо, пожежа може виникнути при взаємодії горючих речовин, окислення і джерел запалювання. У приміщеннях ВЦ присутні всі три основні чинники, необхідні для виникнення пожежі [48].

Для відводу теплоти від ПК у обчислювальному центрі постійно діє потужна система кондиціювання. Тому кисень, як окиснювач процесів горіння, є в будь-якій точці робочого приміщення [48].

Ймовірними причинами виникнення пожежі можуть бути несправність електрообладнання (розеток, кабелів), короткі замикання внаслідок виходу з ладу чи експлуатації несправного електроустаткування (ПЕОМ, периферійних пристроїв), порушення правил протипожежної безпеки тощо [48].

Згідно [48] у робочому приміщенні обладнаному ПК з ВДТ можлива пожежа класів – А (горіння твердих речовин) та Е (горіння електроустановок, що перебувають під напругою).

Обчислювальний центр, згідно з [49] відноситься до категорії «П-Па», а клас можливої пожежі визначається як «А».

До засобів гасіння пожежі, призначених для локалізації невеликих загорянь, відносяться пожежні стовбури, внутрішні пожежні водопроводи, вогнегасники, сухий пісок, азбестові ковдри і т. п [49].

Оскільки робоче приміщення, що обладнане ПК з ВДТ має площу 40 м², тому відповідно до вимог [50] для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 2 штук (з розрахунку один вогнегасник с величиною заряду вогнегасної речовини 3 кг. і більше, на 20 м² площі приміщення). Відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м [50].

Усі працівники обчислювального центру під час прийняття на роботу і в процесі праці повинні проходити протипожежний інструктаж, перевірку знань з питань пожежної безпеки. Евакуаційні виходи і проходи утримуються постійно вільними та легкодоступними [51].

Відповідальний за протипожежний стан приміщень обчислювального центру повинен періодично (не рідше одного разу на місяць) перевіряти працездатність пристрою, який забезпечує автоматичне вимкнення системи вентиляції у разі пожежі, а також вогнедимозатримувальних пристроїв [51].

З технічних та організаційних заходів запобігання пожеж в приміщенні (дослідницької лабораторії, конструкторського бюро, тощо) обладнаному ПК з ВДТ, згідно з [51], передбачені наступні протипожежні заходи. На силовому обладнанні, силових та освітлювальних колах, згідно вимог пункту 3.1 «ПУЕ», встановлені захисні пристрої, що вимикають джерело живлення від ділянки електричного кола, у якій виникло коротке замикання.

Для своєчасного попередження пожеж та підвищення оперативності реагування при їх виникненні у приміщенні використовується такий комплекс заходів та правил [51]:

- обов'язковий інструктаж персоналу з питань охорони праці [51];

- наявність системи автоматичної пожежної сигналізації з димовими пожежними оповіщувачами [51];
- заборона залишати без нагляду електричну апаратуру, що використовується [51];
- заборона використання відкритого вогню у приміщенні [51];
- заборона використовувати килими та доріжки із синтетичних матеріалів [51];
- заборона ставити на вікна глухі ґрати (в разі потреби встановлюються ґрати, що розкриваються зсередини) [51];
- наявність шляхів евакуації при виникненні пожежі [51];
- розміщення схеми евакуації людей при пожежі і ознайомлення з нею персоналу [51].

Після закінчення робочого часу всі робочі місця мають бути прибрані, з приміщень необхідно видалити горючі відходи та вимкнути всі струмоприймачі.

При виявленні ознак пожежі працівник повинен [51]:

- негайно повідомити про це службу порятунку за номером телефону – 101 (112);
- вжити заходів щодо евакуації людей та матеріальних цінностей [51];
- вжити заходів до вжити заходів гасіння пожежі з використанням наявних вогнегасників та інших засобів пожежогасіння [51].

Керівник підрозділу при виникненні пожежі зобов'язаний [51]:

- упевнитися що викликані рятувальні служби [51];
- проінформувати про пожежу керівництво та усіх працівників [51];
- відключити подачу електроенергії в приміщення [51];
- організувати евакуацію працівників (за умови загрози життю людей) [51];
- забезпечити дотримання техніки безпеки працівниками, які беруть участь у гасінні пожежі [51];

- організувати зустріч підрозділів Державної пожежної охорони та надати їм допомогу під час локалізації та ліквідації пожежі [51];
- забезпечити безперешкодний доступ до місця виникнення пожежі пожежно-рятувальним підрозділам [51].

7.4.2 Заходи з цивільного захисту

Заходи безпеки при проведенні рятувальних та інших невідкладних робіт (РiНР).

Сучасне виробництво все більше ускладнюється, росте число потенційно небезпечних об'єктів, таких як атомні електростанції, багато джерел іонізуючого випромінювання, велика кількість хімічно небезпечних та вибухонебезпечних, пожежонебезпечних [52].

Тому важливим для керівників на виробництві, чи адміністративній території кваліфіковано вирішувати питання організації і проведення рятувальних та невідкладних робіт при ліквідації наслідків стихійних лих, аварій і катастроф з урахуванням особливостей об'єктів атомної енергетики, хімічнонебезпечних, пожежонебезпечних об'єктів та при повенях [52].

Руйнування і пожежі на об'єктах, пошкодження мереж комунально-енергетичного господарства, можливі зараження викликають необхідність у особового складу формування суворо дотримуватись заходів безпеки та правил поведінки в зонах зараження [53].

Рятувальні та невідкладні роботи (РiНР), їх складність, будуть залежати від виду і характеру осередку ураження. Вони можуть проводитись в дуже складних умовах: в умовах руйнувань, суцільних завалів, масових пожеж; викиду в атмосферу радіоактивних, небезпечних хімічних речовин, а також і біологічних (бактеріальних) засобів [52].

Основна мета РiНР [52]:

- рятувати людей, які опинилися в будь-якому осередку ураження [52];
- надати їм допомогу [52];

- усунути (локалізувати) аварію і ліквідувати пошкодження, які перешкоджають (зважають) проведенню робіт, або загрожують життям людей [52];
- створити умови для проведення відновлювальних робіт [52].

Перед початком робіт в осередках ураження та зонах руйнувань необхідно [53]:

- уважно оглянути зруйновані (пошкоджені) будинки і споруди [53],
- встановити небезпечні та пошкоджені місця [53].

Заборонено без необхідності заходити у зруйновані (пошкоджені) будинки і споруди, які загороджують обвали. Заходити в такі будинки необхідно тільки з найменш небезпечного боку, при цьому уважно прислуховуватись до характерних потріскувань або шороху, які вказують на можливість обрушення пошкоджених конструкцій [53].

Види рятувальних робіт [52]:

- проведення розвідки маршрутів і ділянок наступних робіт [52];
- локалізація і гасіння пожеж на маршрутах висування і ділянках наступних робіт [52];
- пошуки і звільнення людей, які опинилися під завалами в напівзруйнованих будинках, спорудах, загазованих і задимлених, затоплених приміщеннях [52];
- розкриття зруйнованих, або пошкоджених захисних споруд і рятування людей які у них знаходяться [52];
- подача повітря в завалені захисні споруди (якщо пошкоджена фільтровентиляційна система) [52];
- надання першої медичної допомоги ураженим та евакуація їх в лікувальні заклади. Медична допомога потерпілим надається в два етапи: безпосередньо в осередку ураження (перша медична допомога) і в медичних закладах (спеціальна медична допомога) [52];
- вивід населення з небезпечних місць в безпечні, або менше заражені місця [52];

- санітарна обробка людей і знезараження техніки, транспорту, обладнання території, а також взуття, одягу ЗІЗ буде детальніше розглянуто трохи нижче [52].

Інші невідкладні роботи – це роботи по локалізації або ліквідації наслідків аварій, катастроф, стихійних лих і при використанні сучасної зброї масового ураження. До них відносяться [52]:

- термінове відновлення автомобільних, залізничних доріг; пророблення колонних шляхів по завалах і розчищення проїздів в завалах і на заражених ділянках [52];

- локалізація (ліквідація) аварій на технологічних і КЕМ, короткотермінове відновлення пошкоджених, зруйнованих ділянок з метою забезпечення рятувальних робіт [52];

- підсилення або зруйнування конструкцій, які загрожують обвалом, або перешкоджають проведенню рятувальних робіт і безпечному руху. Зруйнування конструкцій і укріплення їх проводиться за допомогою засобів механізації [52];

- ремонт пошкоджених захисних споруд для вторинного використання при виникненні аварії, чи катастроф, або після використання ворогом зброї масового ураження [52].

При виконанні робіт на висоті необхідно користуватись страхувальними засобами, місця (ділянки) огорожуються і позначаються спеціальними знаками. Організовується надійна страховка рятувальників, які рятують людей у зруйнованих чи пошкоджених будинках (спорудах) і завалах. Не дозволяється проведення робіт у завалах одиночним рятувальникам (бійцям) [53].

Для роботи на електромережах призначають підготовлених для цього людей. Всі роботи здійснюються після вимкнення електроенергії. Забороняється торкатися електричних дротів та з'єднаних з ними металевих конструкцій [53].

При роботі на водопровідних, каналізаційних та газових мережах особовий склад повинен бути забезпечений ізолюючими протигазами.

Наявність газу визначають газоаналізаторами, або іншими спеціальними приладами. Дозволяється працювати у фільтруючих протигазах з додатковими патронами (або спеціальними фільтруючими коробками) і користуватись інструментом із кольорового металу або обмідненим. Забороняється курити, користуватись сірниками та інструментом, який при роботі з ним іскрить, поблизу загазованих ділянок [53].

Слід суворо дотримуватись заходів пожежної безпеки. Не можна гасити палаючі метали (магній, натрій) чи електричні пристрої, що знаходяться під напругою, резервуари з бензином, гасом, матеріали, які зберігаються разом з негашеним вапном та карбідом кальцію, водою, а використовувати тільки вогнегасники [53].

При проведенні РіНР вночі організовують освітлення ділянки робіт та під'їзних шляхів, умовними освітлювальними знаками позначаються небезпечні місця для руху транспорту, переходи, котловани та інше [53].

Встановлюються відповідні режими радіоактивного захисту, які передбачають максимально допустиму тривалість перебування в зоні радіоактивного зараження (на ділянках робіт), термін перебування у захисних спорудах, а також тривалість проїзду з району розташування формувань осередок ураження і назад [53].

Суворе дотримання всього комплексу заходів з забезпечення безпеки особового складу при проведенні РіНР дозволить зберегти їх працездатність, обмежити втрату людей та своєчасно виконати поставлені їм завдання [53].

ВИСНОВКИ

З метою зменшення ймовірності перебування людини у потенційно небезпечному для здоров'я середовищі було досліджено проблему прогнозування забрудненості атмосферного повітря, і як результат було розроблено відповідне програмне забезпечення. З огляду на виконану роботу можна зробити ряд висновків:

- розглянуто вимоги до предметної області та підходи до проведення машинного навчання, запропоновано використання нейронних мереж у якості моделей прогнозування, а також було проведено роботу з набором даних для подальшого прогнозування. Крім того, були розглянуті програмні системи, що виконують схожі задачі;

- проаналізовані методи прогнозування з використанням PHM та LSTM, було створено модель прогнозування, яка була перевірена на тестовій вибірці. Над моделлю був проведений аналіз з відбором гіперпараметрів, комбінації шарів та розроблено метод прогнозування майбутніх значень з моделлю нейронних мереж, що використовує автомобільний трафік у якості уточнюючих даних;

- виявлено функціональні вимоги, була розроблена відповідна структура у вигляді мікросервісів. Розглянуто інструменти для реалізації системи та обрано мови програмування Python, C#, TypeScript та фреймворки Jupiter, ASP.NET Core, Angular відповідно. Крім того, була спроектована схема бази даних та підходи до взаємодії БД та мікросервісів;

- обрано та реалізовано алгоритм функціонування системи. Розроблено модулі прогнозування, а також серверну і клієнтську частини;

- протестовано розроблену систему та зроблено порівняння розробленого методу зі вже існуючими.

Розроблене програмне забезпечення реалізує такі функції:

- аналіз датасету;

- імплементація моделі, тренування моделі, підбір гіперпараметрів, тестування моделі;
- прогнозування майбутніх значень та їхня візуалізація;
- збереження прогнозу у базу даних;
- перетворення даних прогнозів у зручний для візуалізації формат;
- створення сервісу для аналізу прогнозів, виявлення найбільшого забруднювача, обчислення індексу якості повітря;
- надання карти міста з позначками станцій та прогнозованої якості повітря;
- детальна інформація про прогноз обраної станції: дата, загальний індекс, найбільший забруднювач, його прогнозоване значення;
- вибір інтервалу прогнозування.

Наукова новизна роботи полягає в запропонованому методі прогнозування забруднення атмосферного повітря, який ґрунтується на гіпотезі про зв'язок величини основних забруднювачів з інтенсивністю автомобільного трафіку, що дозволяє покращити точність прогнозу у порівнянні з використанням звичайних рекурентних нейронних мереж.

Оскільки розроблене програмне забезпечення є вебзастосунком, його можна запускати на будь-яких пристроях, на яких наявний браузер.

Програмне забезпечення придатне до розширення своєї функціональності, а саме:

- взаємодії зі станціями міста для отримання актуального значення забрудненості;
- більш дискретний прогноз – прогнозувати не на цілий день, а погодинно або частіше.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Изменения в городской окружающей среде, обусловленные урбанизацией, и связанные с этим требования к обслуживанию [Электрон. ресурс]. – Режим доступа : <https://meteo.gov.ua/ua/34421/zmi/articles/read/23>.
2. Лоскутов, А. Ю. Анализ временных рядов. Курс лекций. Физический факультет МГУ [Электрон. ресурс] – Режим доступа : http://chaos.phys.msu.ru/loskutov/PDF/Lectures_time_series_analysis.pdf.
3. Біляєв, М.М. Моделювання і прогнозування стану довкілля: підручник для студентів вищих навчальних закладів [Текст] / М. М. Біляєв, В.В. Біляєва, П.С Кіріченко. – Кривий Ріг : Вид. Р.А. Козлов, 2016. – 207 с.
4. Applications of Hilbert-Huang transform to non-stationary financial time series analysis [Text] / N. E. Huang, M. Wu, W. Qu, S. R. Long, S. S. P. Shen. // Applied Stochastic Models in Business and Industry. – 2003. – Vol. 19, № 3. – Pp. 245–268.
5. Обзор моделей прогнозирования временных рядов [Электрон. ресурс]. – Режим доступа : <https://habr.com/ru/post/180409/>.
6. Deconstructing ARIMA [Electronic resource]. – Access mode : <https://www.kaggle.com/nholloway/deconstructing-arma#ARIMA>.
7. Geron, A. Hands-On Machine Learning with Scikit-Learn and TensorFlow [Text] : Concepts, Tools, and Techniques to Build Intelligent Systems / Aurellen Geron. – Sebastopol, CA, USA : O'Reilly Media Inc., 2017. – 819 p.
8. Солдатов, О.П. Применение нейронных сетей для решения задач прогнозирования [Электрон. ресурс] / О.П. Солдатов, В.В. Семенов. – Режим доступа : <http://elibrary.lt/resursai/Uzsienio%20leidiniai/MFTI/2006/136.pdf>.
9. Осовский, С. Нейронные сети для обработки информации [Текст] / С. Осовский. – Пер. с польского И.Д. Рудинского. – М. : Финансы и статистика, 2002. – 344 с.
10. Tetko, I. V. Neural Network Studies. I. Comparison of Overfitting and Overtraining [Text] / Igor V. Tetko, David J. Livingstone, Alexander I. Luik. – Kyiv,

Ukraine : Biomedical Department, Institute of Bioorganic and Petroleum Chemistry, 1995. – Pp. 826-833.

11. Функции активации нейросети [Электрон. ресурс]. – Режим доступа : <https://neurohive.io/ru/osnovy-data-science/activation-functions/>.

12. Индекс качества воздуха [Электрон. ресурс]. – Режим доступа : <https://ru.euronews.com/weather/copernicus-air-quality-index>.

13. Air Quality in Madrid (2001-2018) [Electronic resource]. – Access mode : <https://www.kaggle.com/decide-soluciones/air-quality-madrid>.

14. Open Data, or Opacity through Transparency [Electronic resource]. – Access mode : <https://diego.codes/post/open-data/>.

15. Air Quality Today. Global air quality map: the most accurate air quality data in 93 countries [Electronic resource]. – Access mode : <https://breezometer.com/air-quality-map>.

16. AirVisual | Air Quality & Pollution Forecast PM2.5 AQI [Electronic resource]. – Access mode : <https://www.amazon.com/IQAir-Air-Quality-AirVisual/dp/B07BRKM6SL>.

17. About the World Air Quality Index project [Electronic resource]. – Access mode : <https://aqicn.org/contact/>.

18. Korbicz, J. Neuro–fuzzy networks and their application to fault detection of dynamical systems [Text] / J. Korbicz, M. Kowa // Engineering Applications of Artificial Intelligence archive. – New York : Pergamon Press, Inc. – Vol. 20. – Issue 5. – 2007. – Pp. 609–617.

19. Understanding LSTM Networks [Electronic resource]. – Access mode : <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

20. LSTM – сети долгой краткосрочной памяти [Электрон. ресурс]. – Режим доступа : <https://habr.com/ru/company/wunderfund/blog/331310/>.

21. Шумнов, П. А. Информатика и системы управления. Умный аранжировщик [Текст] / П. А. Шумнов, Н. К. Завриев // Научно-образовательные соревнования молодых исследователей «Шаг в будущее, Москва» и «Шаг в будущее, космонавтика». Сборник лучших работ. – М. :

2019. – С. 257-258.

22. Particle levels prediction using stateful LSTM networks [Electronic resource]. – Access mode : <https://www.kaggle.com/diegovicente/particle-levels-prediction-using-lstm/>.

23. Optimum number of epochs and neurons for an LSTM network [Electronic resource]. – Access mode : <https://stats.stackexchange.com/questions/286713/optimum-number-of-epochs-and-neurons-for-an-lstm-network>.

24. Predicting Sequential Data using LSTM [Electronic resource] : An Introduction. – Access mode : <https://towardsdatascience.com/time-series-forecasting-with-recurrent-neural-networks-74674e289816>.

25. Эпоха, батч, итерация – в чем различия? [Электрон. ресурс]. – Режим доступа : <https://neurohive.io/ru/osnovy-data-science/jepoha-razmer-batcha-iteracija/>.

26. Микросервисная архитектура [Электрон. ресурс] : плюсы и минусы. – Режим доступа : <https://javarush.ru/groups/posts/2015-mikroservisnaja-arhhitektura-pljusih-i-minusih>.

27. Top 6 Data Science Programming Languages for 2019 [Electronic resource] – Access mode : <https://medium.com/datadriveninvestor/top-6-data-science-programming-languages-for-2019-39ba1b6819a8>.

28. Краткий обзор языка C# [Электрон. ресурс]. – Режим доступа : <https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/>.

29. Difference between TypeScript and JavaScript [Electronic resource]. – Access mode : <https://www.geeksforgeeks.org/difference-between-typescript-and-javascript/>.

30. JupyterHub, или как управлять сотнями пользователей Python. Лекция Яндекса [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/company/yandex/blog/353546/>.

31. Прайс, Марк Дж. C# 7 и .NET Core. Кроссплатформенная разработка для профессионалов. 3-е изд. [Текст] / Марк Дж. Прайс. – СПб. :

Питер, 2018. – 640 с.

32. JetBrains PyCharm [Electronic resource]. – Access mode : <https://www.softkey.ua/ua/catalog/programming/jetbrains-pycharm/>.

33. Visual Studio. Лучшие в своем классе средства для разработчиков [Электрон. ресурс]. – Режим доступа : <https://visualstudio.microsoft.com/ru/>.

34. Введение в Entity Framework Core [Электрон. ресурс]. – Режим доступа : <https://metanit.com/sharp/entityframeworkcore/1.1.php>.

35. Martin, R. C. Clean code [Text] : a handbook of agile software craftsmanship / Robert C. Martin. – Westford, Massachusetts : Pearson Education, Inc., 2015. – 413 p.

36. Iskandaryan, D. Air Quality Prediction in Smart Cities Using Machine Learning Technologies Based on Sensor Data: A Review [Text] / Ditsuhi Iskandaryan, Francisco Ramos, Sergio Trilles. – Spain : Institute of New Imaging Technologies, 2020. – 32 p.

37. Остапенко, В.В. Методичні вказівки по економічному обґрунтуванню дипломних проєктів для студентів спеціальностей 7.080403 «Програмне забезпечення автоматизованих систем», 7.080402 «Інформаційні технології проєктування» і 7.091501 «Комп'ютерні та інтелектуальні системи і мережі» всіх форм навчання [Текст] / В.В. Остапенко, Е.М. Касьян, Є.М. Сердюк. – Запоріжжя : ЗНТУ, 2015. – 22 с.

38. Коробко, О.В. Методичні вказівки до виконання розділу «Охорона праці та безпека у надзвичайних ситуаціях» у дипломних проєктах (роботах) студентів–магістрів спеціальностей 121 Інженерія програмного забезпечення (спеціалізація: інженерія програмного забезпечення), 121 Інженерія програмного забезпечення (спеціалізація: програмне забезпечення систем), 122 Комп'ютерні науки (спеціалізація: інформаційні технології проєктування), 122 Комп'ютерні науки (спеціалізація: системи штучного інтелекту), 123 Комп'ютерна інженерія (спеціалізація: комп'ютерні системи та мережі), 123 Комп'ютерна інженерія (спеціалізація: спеціалізовані комп'ютерні системи), 124 Системний аналіз (спеціалізація: інтелектуальні технології та прийняття

рішень в складних системах) усіх форм навчання [Текст] / О.В. Коробко. – Запоріжжя: ЗНТУ, 2018 – 27 с.

39. Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці [Текст] : НПАОП 0.00-4.12-05. – На заміну ДНАОП 0.00-4.12-99, ДНАОП 0.00-8.01-93 ; чинний від 2005-02-26. – К. : Держнагляд охорони праці України, 2005. – 53 с.

40. Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом [Текст] : ДСТУ Б В.2.5-82:2016. – На заміну ДБН В.2.5-27-2006 ; чинний від 2017-04-01. – К. : ДП «УкрНДНЦ», 2016. – 109 с.

41. Дизайн і ергономіка. Обладнання виробниче. Загальні вимоги дизайну та ергономіки [Текст] : ДСТУ 7234:2011. – На заміну ГОСТ 12.2.049-80; чинний від 2011-08-01. – К. : Держспоживстандарт України, 2011. – 11 с.

42. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу [Електрон. ресурс] : Державні санітарні норми та правила від 30.05.2014 р. № z0472-14. Редакція від: 30.05.2014.. – Режим доступу : <http://zakon2.rada.gov.ua/laws/show/z0472-14>.

43. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Текст] : ДСанПіН 3.3.2.007-98. – Чинний від 1998-12-10. – К. : МОЗ України, 1998. – 121 с.

44. Санітарні норми мікроклімату виробничих приміщень [Текст] : ДСН 3.3.6.042-99. – Чинний від 1999-12-01. – К. : МОЗ України, 1999. – 106 с.

45. Опалення, вентиляція та кондиціонування [Текст] : ДБН В.2.5-67:2013. – Чинний від 2014-01-01. – К. : Мінрегіонбуд України, 2013. – 149 с.

46. Природне і штучне освітлення [Текст] : ДБН В.2.5-28-2018. – Чинний з 2019-03-01. – К. : Мінрегіон України, 2018. – 133 с.

47. Санітарні норми виробничого шуму, ультразвуку та інфразвуку [Текст] : ДСН 3.3.6.037-99. – Чинний від 1999-12-01. – К. : МОЗ України, 1999.

–128 с.

48. Класифікація пожеж (EN 2:1992; EN 2:1992/A1:2004, IDT) [Текст] : ДСТУ EN 2:2014. – На заміну ГОСТ 27331-87 ; чинний з 01.01.2016. – К. : Мінекономрозвитку України, 2014. – 7 с.

49. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою [Текст] : ДСТУ Б В.1.1-36:2016. – На заміну НАПБ Б.03.002-2007 ; чинний від 2017-01-01. – К. : Мінрегіонбуд України, 2016. – 66 с.

50. Правила експлуатації та типових норм належності вогнегасників [Текст] – Чинний від 2018-02-23. – К. : МВС України, 2018. – 23 с.

51. Правила пожежної безпеки в Україні [Текст] : НАПБ А.01.001-14. – На заміну НАПБ А.01.001-04 ; чинний від 2014-12-30. – К. : МВС України, 2014. – 47 с.

52. Основи цивільного захисту: Навч. посібник [Текст] / В.О. Васійчук, В.Є Гончарук, С.І. Качан, С.М. Мохняк. – Л. : Вид. Національного університету "Львівська політехніка", 2010. – 417с.

53. Заходи безпеки при проведенні рятувальних та інших невідкладних робіт [Електрон. ресурс]. – Режим доступу : <https://disted.edu.vn.ua/courses/learn/1520>.

ДОДАТОК А
Технічне завдання

A.1 Вступ

A.1.1 Назва розроблюваного програмного забезпечення

Назва розроблюваного програмного забезпечення: «AirForecasting».

A.1.2 Призначення та область застосування

Розроблюване програмне забезпечення повинне надавати можливості прогнозування забруднення атмосферного повітря. Область застосування – екологія та сфера охорони здоров'я.

A.2 Підстави для розробки

Підставою для розробки є завдання на дипломну роботу на тему «Дослідження та програмна реалізація методів прогнозування забруднення атмосферного повітря» згідно з наказом № 301 від 3 листопада 2020 року у Національному університеті «Запорізька політехніка».

Актуальність розробки програмного забезпечення полягає у підвищенні обізнаності людей про стан якості атмосферного повітря та у зверненні уваги на проблему забрудненості, оскільки чисте повітря є одним з основних складових здоров'я людини. Сенс такої системи полягає у прогнозуванні основних хімічних елементів та сполук в залежності від їх значень у минулому.

A.3 Призначення розробки

Маючи таку систему як один з інструментів, державні метеорологічні підрозділи зможуть поширювати сповіщення про поточний та майбутній стан атмосферного повітря у автономному режимі за допомогою мережі Інтернет.

Розроблюване програмне забезпечення орієнтоване на точність прогнозування індексу якості повітря, що досягається використанням

рекурентних нейронних мереж типу LSTM з розробкою модифікованого методу.

Розроблений програмний продукт має надавати можливість прогнозувати рівень забрудненості за основними забруднювачами на декілька тижнів вперед на станціях, обчислювати європейський індекс якості повітря, зберігати прогнозовані значення та надавати можливість користувачу обирати станції та кількість днів для прогнозування.

A.4 Вимоги до програмного забезпечення

A.4.1 Вимоги до функціональних характеристик

Головним завданням програмного забезпечення є прогнозування індексу якості повітря у межах міста за зазначеними станціями. Для реалізації цього завдання система поділяється на окремі блоки – веб, прогнозуюча модель, база даних. Основою задачею є обробка датасету з даними про рівень забруднювачів повітря, створення моделі на основі блоків LSTM та прогнозування майбутніх значень. Прогнозовані значення записуються у базу даних, далі оброблюються бекендом, що надає зручний API. Фронтенд-частина відповідає за графічну візуалізацію даних та використовує API бекенду. Тож функціональні вимоги до програмного забезпечення можна розділити на декілька рівнів. Вимоги до блоку аналізу даних та прогнозування:

- аналіз датасету: вибір релевантних станцій, видалення пропусків в даних, нормалізація даних, формування тренувальної та тестової вибірок;
- імплементація моделі, тренування моделі, підбір гіперпараметрів, тестування моделі;
- візуальне порівняння справжніх значень та прогнозованих для тестової вибірки;
- прогнозування майбутніх значень;
- візуалізація прогнозованих значень;
- збереження прогнозу у базу даних.

Вимоги до бекенду:

- взаємодія з базою даних;
- перетворення даних прогнозів у зручний для візуалізації формат;
- створення сервісу для аналізу найсвіжіших прогнозів, виявлення найбільшого забруднювача, обчислення індексу якості повітря;
- надання API з можливістю давати прогноз для обраної станції на задану кількість днів, регулювати тип прогнозу (оптимістичний / середній / песимістичний).

Вимоги до фронтенду:

- карта міста з позначками станцій та прогнозованої якості повітря;
- детальна інформація про прогноз обраної станції: дата, загальний індекс, найбільший забруднювач, його прогнозоване значення;
- вибір інтервалу прогнозування;
- вибір типу прогнозування.

Вхідними даними програми є набір даних з величинами забруднювачів по станціях Мадриду у період з 2001 по 2018 рр.

Графічний інтерфейс повинен забезпечити просту й ефективну взаємодію користувача з розробленим програмним забезпеченням.

А.4.2 Вимоги до надійності

Робота користувача з програмним продуктом повинна спрощуватися шляхом інтуїтивно зрозумілого інтерфейсу та системних повідомлень у разі помилок.

Безперебійний доступ до програмного продукту має забезпечуватися коректним розгортанням модулів програмного забезпечення, що виконується системним адміністратором. Бекенд, фронтенд, база даних та ПЗ для прогнозування мають бути запущені за інструкціями.

A.4.3 Вимоги до складу та параметрів технічних засобів

Для відсутності проблем функціонування системи на комп'ютері користувача повинні виконуватися такі умови до апаратної та програмної частини:

- наявність IBM-сумісної персональної електронної обчислювальної машини на якій буде працювати додаток;
- процесор із тактовою частотою 2,2 GHz та більше;
- оперативна пам'ять 4 ГБ та більше – оскільки прогнозування є трудомістким процесом;
- вільний простір на жорсткому диску не менше, ніж 7 ГБ;
- Python версії 3.7.9 та вище;
- інтегроване середовище розробки PyCharm актуальної версії;
- інтегроване середовище розробки Visual Studio 2019 та вище;
- Microsoft SQL Server актуальної версії;
- наявність сучасної версії браузера (Google Chrome, Opera, Safari та інші).

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- текст програми.

A.6 Стадії і етапи розробки

A.6.1 Стадії розробки

Розробка повинна бути проведена за такими стадіями:

- розробка технічного завдання;
- робоче проєктування;
- реалізація;

- впровадження.

A.6.2 Етапи розробки

Стадія розробки технічного завдання полягає у створенні даного технічного завдання, його погодження та затвердження.

Стадія робочого проєктування повинна містити у собі аналіз вимог, вибір інструментарію розробки, розробку методу з використанням нейронних мереж.

Стадія реалізації повинна складатися з безпосередньо розробки ПЗ для прогнозування з імплементацією розробленого методу, реалізації всіх поставлених функціональних вимог, розробки програмної документації та тестування програми.

Стадія впровадження має бути стадією підготовки та передачі програми.

A.6.3 Зміст робіт за етапами

На етапі розробки технічного завдання повинні бути виконані перераховані нижче роботи:

- постановка задачі;
- визначення і уточнення вимог до технічних засобів;
- формування функціональних вимог до програми;
- визначення стадій, етапів і термінів розробки програми та документації на неї;
- узгодження і затвердження технічного завдання.

На етапі аналізу вимог необхідно проаналізувати вимоги та окреслити межі розроблюваного програмного забезпечення.

На етапі розробки методу необхідно розібрати будову обраної нейронної мережі та розширити її шляхом внесення додаткових даних. Результати аналізу та проєктування методу необхідно зафіксувати у 2-му розділі пояснювальної записки.

На етапі вибору інструментарію розробки необхідно проаналізувати доступні інструменти та обрати з них ті, що найкраще підходять до розробки. Результати вибору необхідно зафіксувати у 3-му розділі пояснювальної записки.

На етапі розробки програми повинна бути виконана робота з програмування і налагодженні програми, імплементації розробленого методу.

На етапі розробки програмної документації повинна бути виконана розробка програмних документів відповідно до вимог до складу документації.

На етапі впровадження програми необхідно узгодити методики тестових експериментів, провести їх та скоректувати програму за необхідності, і підготувати програмне забезпечення та документацію до приймання.

ДОДАТОК Б
Текст програми

Б.1 Блок коду «Preprocessing»

```

### Region Preprocessing

### imports
def imports():
    import numpy as np
    import pandas as pd
    import tensorflow as tf

    import matplotlib.pyplot as plt
    import seaborn as sns
    import missingno as msno
    import plotly.graph_objects as go

    from keras.models import Sequential
    from keras.layers import Dense, LSTM, TimeDistributed
    from keras.optimizers import RMSprop

    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import MinMaxScaler
    from keras.preprocessing.sequence import TimeseriesGenerator

    import pyodbc
    from datetime import datetime

    sns.set()
    %matplotlib inline

### get Data
def getDataBy(station, pollutant):

    with pd.HDFStore('../data/madrid.h5') as data:
        df = data[station]

    df = df.sort_index()
    df.info()

    msno.matrix(df, freq='Y')

    series = df[[currentPollutant]]
    return series

### visualize
def visualizeSeries(series):
    fig, ax = plt.subplots(figsize=(20, 5))

    (series.interpolate(method='time')
     .rolling(window=24*15).mean()
     .plot(ax=ax))

def normalizeSeries(series):
    ### normalize
    max = series.max(axis = 0)[0]
    series /= max

    series = series[(series.index.year > startYear) & (series.index.hour %
hoursPoint == 0)]

    series.rolling(window=10).mean().plot()

### drop N/A values

```

```

series = series.dropna()

return series

def getTrainSample(series, split_percent):
    current_pollutant_data = series[currentPollutant].values
    current_pollutant_data = current_pollutant_data.reshape((-1,1))

    split = int(split_percent*len(current_pollutant_data))

    current_pollutant_train = current_pollutant_data[:split]
    series_train = series.index[:split]
    print(len(current_pollutant_train))

    X_train = series.values[:split, :offset]
    y_train = series.values[:split, offset:]

    return X_train, y_train

def getTestSample(series, split_percent):
    current_pollutant_data = series[currentPollutant].values
    current_pollutant_data = current_pollutant_data.reshape((-1,1))

    split = int(split_percent*len(current_pollutant_data))

    current_pollutant_test = current_pollutant_data[split:]
    series_test = series.index[split:]
    print(len(current_pollutant_test))

    X_test = series.values[split:, :offset]
    y_test = series.values[split:, offset:]

    return X_test, y_test

```

Б.2 Блок коду «Forecasting»

```

### create LSTM
def create_lstm(offset, neurons=(2,1), batch_size=32, lr=0.005, n_features=1):
    model = Sequential()

    # First LSTM layer
    model.add(LSTM(
        neurons[0], return_sequences=True, stateful=True,
        batch_input_shape=(batch_size, offset, n_features))
    )

    # Second LSTM layer if defined
    if neurons[1]:
        model.add(LSTM(
            neurons[1], return_sequences=True, stateful=True,
            batch_input_shape=(batch_size, offset, n_features))
        )

    # TimeDistributed layer to generate all the timesteps
    model.add(TimeDistributed(Dense(1)))

    #optimizer = RMSprop(lr=lr)
    #model.compile(loss='mean_squared_error', optimizer=optimizer)

```

```

metric = tf.keras.metrics.MeanAbsoluteError()

model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=[tf.keras.metrics.Accuracy(),
                      tf.keras.metrics.MeanAbsoluteError(),
                      tf.keras.metrics.RootMeanSquaredError()])

return model

### train
def train_model(model, X_train, y_train, epochs=8):
    mse = list()
    mae = list()

    for i in range(epochs):
        if i % 1 == 0:
            print('Epoch {:02d}/{:}...'.format(i + 1, epochs), end=' ')

        log = model.fit(
            X_train, y_train,
            epochs=1, batch_size=32,
            verbose=0, shuffle=False
        )

        mse.append(log.history['loss'][-1])
        print('loss: {:.4f}'.format(mse[-1]), end='')

        metric = tf.keras.metrics.MeanAbsoluteError().name
        mae.append(log.history[metric][-1])
        print('mae: {:.4f}'.format(mae[-1]))

        model.reset_states()

    return model, mse

### validation
def validate_model(model, X_test, y_test):
    preds = model.predict(X_test)
    model.reset_states()
    return np.mean((y_test - preds) ** 2)

### comparing
def comparing(model, series, X_train, y_train, X_test, y_test):
    offset = 24
    model = create_lstm(offset)
    model, mse_train = train_model(model, X_train, y_train)

    model = create_lstm(offset)
    model, mse_test = train_model(model, X_test, y_test)
    plt.plot(mse_train)
    plt.plot(mse_test)
    plt.title('Втрати моделі за епохами')
    plt.xlabel('Епохи')
    plt.xticks(np.arange(0, 8), np.arange(1, 9))
    plt.ylabel('Втрати')
    plt.legend(['тренувальна', 'тестова'], loc='upper right')
    plt.show()

    plt.plot(model['accuracy_train'])
    plt.plot(model['accuracy_test'])
    plt.title('Точність моделі за епохами')
    plt.xlabel('Епохи')

```

```

plt.xticks([0,1,2,3,4,5,6,7], [1,2,3,4,5,6,7,8])
plt.xticks(np.arange(0,8), np.arange(1,9))
plt.ylabel('Точність')
plt.legend(['тренувальна', 'тестова'], loc='lower right')
plt.show()

model.fit(
X_train, y_train,
epochs=1, batch_size=32,
verbose=1, shuffle=False
)
model.reset_states()

preds = model.predict(X_test)

eval_train = model.evaluate(X_train, y_train)
eval_test = model.evaluate(X_test, y_test)

fig, ax = plt.subplots(figsize=(20, 5))

start = X_test.shape[0] - 42
interval = X_test[start+3:start+33:2, :, 0].reshape(-1, 1)

truth, = plt.plot(np.arange(24*15), interval, alpha=0.6)

old_preds = list()

for point in range(1, 15, 2):
    prediction = np.squeeze(preds[start + point*2])
    pred, = plt.plot(point * offset + np.arange(offset), prediction,
                     linestyle='--', color='red')
    old_preds.append(prediction)
    print(point)

for point in range(15, 20, 1):
    prediction = np.squeeze(preds[start + point*2])
    pred, = plt.plot(point * offset + np.arange(offset), prediction,
                     linestyle='--', color='red')
    old_preds.append(prediction)
    print(point)

plt.legend(
    [truth, pred],
    ['Оригінальні значення', 'Прогнозування']
)
ax.set_ylim([-1, 1.1])
ax.set_xticks(12 + np.arange(20) * offset)
_ = ax.set_xticklabels([])

### forecast
def forecasting(model, series, look_back, num_prediction):
    prediction_list = series[-look_back:]

    for _ in range(num_prediction):
        x = prediction_list[-look_back:]
        x = x.reshape((1, look_back, 1))
        out = model.predict(x)[0][0]
        prediction_list = np.append(prediction_list, out)
    prediction_list = prediction_list[look_back-1:]

    return prediction_list

### visualize
def visualizeForecasting(forecast):

```

```

    trace1 = go.Scatter(
        x = series.index,
        y = series,
        mode = 'lines',
        name = 'Дані'
    )
    trace2 = go.Scatter(
        x = forecast_dates,
        y = forecast,
        mode = 'lines',
        name = 'Прогнозування'
    )
    layout = go.Layout(
        title = "Станція: " + currentStation + " Забруднювач: " +
currentPollutant,
        xaxis = {'title' : "Дата"},
        yaxis = {'title' : "Рівень забруднення"}
    )
#%%

```

Б.3 Блок коду «Saving»

```

#%% Region Saving
#%% save plots
def savePlots (plot, name):
    plot.write_image("../images/{0}.png".format(name))

#%% save to db
def saveToDB(forecasting):

    connection = pyodbc.connect('DRIVER={SQL Server Native Client
11.0};Server=DESKTOP-KM5JMS4;Database=AirForecasting;Trusted_Connection=yes;')
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM Stations")
    stations = cursor.fetchall()
    stationId = 0

    for station in stations:
        if(station[1] == currentStation):
            stationId = station[0]

    cursor.execute("SELECT * FROM Pollutants")
    pollutants = cursor.fetchall()
    pollutantId = 0

    for pollutant in pollutants:
        if(pollutant[1] == currentPollutant):
            pollutantId = pollutant[0]

    cursor.execute("SELECT * FROM ForecastingSequenceNumbers")
    forecastingSequenceNumbers = cursor.fetchall()
    isExist = False
    currentNumber = 0
    for row in forecastingSequenceNumbers:
        if(row[0] == pollutantId and row[1] == stationId):
            isExist = True
            currentNumber = row[2]
            break
    currentNumber += 1
    if(isExist):

```

```

        updateStr = "UPDATE ForecastingSequenceNumbers SET Number = {0} WHERE
PollutantID={1} AND StationID={2}".format(currentNumber, pollutantId, stationId)
        cursor.execute(updateStr)
    else:
        inserts = "INSERT INTO ForecastingSequenceNumbers VALUES ({0}, {1},
{2})".format(pollutantId, stationId, currentNumber)
        cursor.execute(insertStr)
        cursor.commit()

    def forecast_dates(num_forecasting):
        last_date = datetime.today()
        prediction_dates = pd.date_range(last_date,
periods=num_forecasting+1).tolist()
        return prediction_dates

    num_forecasting = num_prediction
    dates = forecast_dates(num_forecasting)
    for date in dates:
        print(date)

    for i in range(0, num_forecasting):
        datestr = dates[i].strftime("%Y-%m-%d %H:%M:%S")
        insertStr = "INSERT INTO Forecastings VALUES ('{0}', {1}, {2}, {3},
{4})".format(datestr, pollutantId, forecast[i], stationId, currentNumber)
        cursor.execute(insertStr)
        cursor.commit()
    cursor.close()

```

Б.4 Блок коду «Main»

```

### main block
currentStation = '28079016'
currentPollutant = 'O_3'
split_percent = 0.80
offset = 24
look_back = 60
num_prediction = 60

imports()
series = getDataBy(currentStation, currentPollutant)
visualizeSeries(series)
series = normalizeSeries(series)

X_train, y_train = getTrainSample(series, split_percent)
X_test, y_test = getTestSample(series, split_percent)

model = create_lstm(offset)
train_model(model, X_train, y_train)

validate_model(model, X_test, y_test)

comparing(model, series, X_train, y_train, X_test, y_test)

forecasting = forecasting(model, series, look_back, num_prediction)
plot, name = visualizeSeries(forecasting)

savePlots(plot, name)
saveToDB(forecasting)

```

Б.5 Текст файла Program.cs

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Hosting;

namespace AirForecasting
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(web Builder =>
                {
                    webBuilder.UseStartup<Startup>();
                })
        }
    }
}
```

Б.6 Текст файла Startup.cs

```
using AirForecasting.Core.Models;
using AirForecasting.Interfaces;
using AirForecasting.Services;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.SpaServices.AngularCli;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace AirForecasting
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc();
            var connection =
@"Server=localhost;Database=AirForecasting;Trusted_Connection=True";
            services.AddDbContext<AirForecastingContext>(options =>
options.UseSqlServer(connection));

            services.AddScoped<IForecastAnalysis, ForecastAnalysisService>();
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Error");
                // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
                app.UseHsts();
            }

            app.UseStaticFiles();
            app.UseSpaStaticFiles();

            app.UseRouting();

            app.UseAuthorization();

            app.MapControllerRoute(
                name: "default",
                pattern: "{controller}/{action}/{id?}");
        }
    }
}
```

```

{
    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller}/{action=Index}/{id?}");
    });

    app.UseSpa(spa =>
    {
        spa.Options.SourcePath = "ClientApp";
    });
}
}
}

```

Б.7 Текст файла AirForecastingContext.cs

```

using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;

namespace AirForecasting.Core.Models
{
    public partial class AirForecastingContext : DbContext
    {
        public AirForecastingContext()
        {
        }

        public AirForecastingContext(DbContextOptions<AirForecastingContext> options)
            : base(options)
        {
        }

        public virtual DbSet<Forecasting> Forecastings { get; set; }
        public virtual DbSet<ForecastingSequenceNumber> ForecastingSequenceNumbers { get;
set; }
        public virtual DbSet<Pollutant> Pollutants { get; set; }
        public virtual DbSet<Station> Stations { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Forecasting>(entity =>
            {
                entity.Property(e => e.ForecastingId).HasColumnName("ForecastingID");

                entity.Property(e => e.PollutantId).HasColumnName("PollutantID");

                entity.Property(e => e.StationId).HasColumnName("StationID");

                entity.HasOne(d => d.Pollutant)
                    .WithMany(p => p.Forecastings)
                    .HasForeignKey(d => d.PollutantId)
                    .OnDelete(DeleteBehavior.ClientSetNull)
                    .HasConstraintName("FK_Forecastings_PollutantID");
            });
        }
    }
}

```

```

        entity.HasOne(d => d.Station)
            .WithMany(p => p.Forecastings)
            .HasForeignKey(d => d.StationId)
            .OnDelete(DeleteBehavior.ClientSetNull)
            .HasConstraintName("FK_Forecastings_StationID");
    });

    modelBuilder.Entity<ForecastingSequenceNumber>(entity =>
    {
        entity.HasKey(e => new { e.PollutantId, e.StationId });

        entity.Property(e => e.PollutantId).HasColumnName("PollutantID");

        entity.Property(e => e.StationId).HasColumnName("StationID");

        entity.HasOne(d => d.Pollutant)
            .WithMany(p => p.ForecastingSequenceNumbers)
            .HasForeignKey(d => d.PollutantId)
            .OnDelete(DeleteBehavior.ClientSetNull)
            .HasConstraintName("FK_ForecastingSequenceNumbers_PollutantID");

        entity.HasOne(d => d.Station)
            .WithMany(p => p.ForecastingSequenceNumbers)
            .HasForeignKey(d => d.StationId)
            .OnDelete(DeleteBehavior.ClientSetNull)
            .HasConstraintName("FK_ForecastingSequenceNumbers_StationID");
    });

    modelBuilder.Entity<Pollutant>(entity =>
    {
        entity.Property(e => e.PollutantId).HasColumnName("PollutantID");

        entity.Property(e => e.ShortName)
            .IsRequired()
            .HasMaxLength(5);
    });

    modelBuilder.Entity<Station>(entity =>
    {
        entity.Property(e => e.StationId).HasColumnName("StationID");

        entity.Property(e => e.Address).IsRequired();

        entity.Property(e => e.Name).IsRequired();

        entity.Property(e => e.StationNumber)
            .IsRequired()
            .HasMaxLength(8);
    });

    OnModelCreatingPartial(modelBuilder);
}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}

```

Б.8 Текст файла Forecasting.cs

```

using System;
using System.Collections.Generic;

namespace AirForecasting.Core.Models

```

```

{
    public partial class Forecasting
    {
        public int ForecastingId { get; set; }
        public DateTimeOffset Date { get; set; }
        public int PollutantId { get; set; }
        public double Value { get; set; }
        public int StationId { get; set; }
        public int ForecastingSequenceNumber { get; set; }

        public virtual Pollutant Pollutant { get; set; }
        public virtual Station Station { get; set; }
    }
}

```

Б.9 Текст файла ForecastingSequenceNumbers.cs

```

using System;
using System.Collections.Generic;

namespace AirForecasting.Core.Models
{
    public partial class ForecastingSequenceNumber
    {
        public int PollutantId { get; set; }
        public int StationId { get; set; }
        public int Number { get; set; }

        public virtual Pollutant Pollutant { get; set; }
        public virtual Station Station { get; set; }
    }
}

```

Б.10 Текст файла Pollutant.cs

```

using System;
using System.Collections.Generic;

namespace AirForecasting.Core.Models
{
    public partial class Pollutant
    {
        public Pollutant()
        {
            ForecastingSequenceNumbers = new HashSet<ForecastingSequenceNumber>();
            Forecastings = new HashSet<Forecasting>();
        }

        public int PollutantId { get; set; }
        public string ShortName { get; set; }
        public string FullName { get; set; }

        public virtual ICollection<ForecastingSequenceNumber> ForecastingSequenceNumbers {
get; set; }
        public virtual ICollection<Forecasting> Forecastings { get; set; }
    }
}

```

Б.11 Текст файла Station.cs

```
using System;
using System.Collections.Generic;

namespace AirForecasting.Core.Models
{
    public partial class Station
    {
        public Station()
        {
            ForecastingSequenceNumbers = new HashSet<ForecastingSequenceNumber>();
            Forecastings = new HashSet<Forecasting>();
        }

        public int StationId { get; set; }
        public string StationNumber { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
        public double Lon { get; set; }
        public double Lat { get; set; }
        public int? Elevation { get; set; }

        public virtual ICollection<ForecastingSequenceNumber> ForecastingSequenceNumbers {
            get; set; }
        public virtual ICollection<Forecasting> Forecastings { get; set; }
    }
}
```

Б.12 Текст файла AirForecastController.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using AirForecasting.Core.Models;
using AirForecasting.Interfaces;
using AirForecasting.ViewModels;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace AirForecasting.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class AirForecastController : Controller
    {
        private readonly ILogger<AirForecastController> _logger;
        private readonly AirForecastingContext _context;
        private readonly IForecastAnalysis _forecastAnalysisService;

        public AirForecastController(ILogger<AirForecastController> logger,
            AirForecastingContext context, IForecastAnalysis forecastAnalysisService)
        {
            _logger = logger;
            _context = context;
            _forecastAnalysisService = forecastAnalysisService;
        }
    }
}
```

```

[HttpGet("{stationId}/{days}")]
public async Task<ActionResult<IEnumerable<ForecastViewModel>>>
GetForecastingById(int stationId, int days)
{
    var sequenceNumbers = _context.ForecastingSequenceNumbers.AsNoTracking()
        .Where(x => x.StationId == stationId).ToList();

    var dbForecastingSelected = new List<Forecasting>();

    var dbForecastingsFull = _context.Forecastings.Where(x => x.StationId ==
stationId).ToList();

    foreach (var sequenceNumber in sequenceNumbers)
    {
        var currentForecastings = dbForecastingsFull
            .Where(x => x.ForecastingSequenceNumber == sequenceNumber.Number &&
x.PollutantId == sequenceNumber.PollutantId)
            .ToList();

        dbForecastingSelected.AddRange(currentForecastings);
    }

    var forecast =
_forecastAnalysisService.GetAnalyzedForecast(dbForecastingSelected).Take(days);

    return Ok(forecast);
}
}
}

```

Б.13 Текст файла ForecastAnalysisService.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using AirForecasting.Core.Models;
using AirForecasting.Interfaces;
using AirForecasting.ViewModels;

namespace AirForecasting.Services
{
    public class ForecastAnalysisService : IForecastAnalysis
    {
        private readonly AirForecastingContext _context;

        public ForecastAnalysisService(AirForecastingContext context)
        {
            _context = context;
        }

        public IEnumerable<ForecastViewModel> GetAnalyzedForecast(IEnumerable<Forecasting>
forecastings)
        {
            var analyzed = new List<ForecastViewModel>();

            var uniqueDates = forecastings.Select(x => x.Date.Date).Distinct();

            foreach (var unique in uniqueDates)
            {
                var forecastingsByDate = forecastings.Where(x => x.Date.Date == unique);

```

```

var preAnalyzed = new List<ForecastViewModel>();

foreach (var forecasting in forecastingsByDate)
{
    preAnalyzed.Add(new ForecastViewModel()
    {
        Id = forecasting.ForecastingId,
        DateTime = forecasting.Date,
        PollutantId = forecasting.PollutantId,
        PollutionLevel = NormalizeByEAQI(forecasting.PollutantId,
forecasting.Value),
        TopPollutantValue = forecasting.Value
    });
}

var currentForecast = preAnalyzed.OrderByDescending(x =>
x.PollutionLevel).FirstOrDefault();

currentForecast.TopPollutantName = _context.Pollutants
.FirstOrDefault(x => x.PollutantId == currentForecast.PollutantId)
?.ShortName;

analyzed.Add(currentForecast);
}

return analyzed;
}

// European air-quality-index
private int NormalizeByEAQI(int pollutantId, double value)
{
    switch (pollutantId)
    {
        //O3
        case 1:
            if (value < 80) return 1;
            else if (value < 120) return 2;
            else if (value < 180) return 3;
            else if (value < 240) return 4;
            else return 5;
        //NO2
        case 2:
            if (value < 40) return 1;
            else if (value < 100) return 2;
            else if (value < 200) return 3;
            else if (value < 400) return 4;
            else return 5;
        //SO2
        case 3:
            if (value < 100) return 1;
            else if (value < 200) return 2;
            else if (value < 350) return 3;
            else if (value < 500) return 4;
            else return 5;
        //PM10
        case 4:
            if (value < 20) return 1;
            else if (value < 35) return 2;
            else if (value < 50) return 3;
            else if (value < 100) return 4;
            else return 5;
        //PM2.5
        case 5:
            if (value < 10) return 1;
            else if (value < 20) return 2;
    }
}

```

```

        else if (value < 25) return 3;
        else if (value < 50) return 4;
        else return 5;
    }

    return 0;
}
}
}

```

Б.14 Текст файлу ForecastViewModel.cs

```

using System;

namespace AirForecasting.ViewModels
{
    public class ForecastViewModel
    {
        public int Id { get; set; }

        public DateTimeOffset DateTime { get; set; }

        public int PollutionLevel { get; set; }

        public int PollutantId { get; set; }

        public string TopPollutantName { get; set; }

        public double TopPollutantValue { get; set; }
    }
}

```

Б.15 Текст файлу set-days.component.html

```

<div>
  <form [formGroup]="checkoutForm" (ngSubmit)="onSubmit(checkoutForm.value)">
    <div class="">
      <label for="days" class="m-1">
        Введіть кількість днів:
      </label>
      <input id="name" type="number" formControlName="days" class="m-1" />
    </div>
    <button class="button" type="submit" class="btn btn-primary">Спрогнозувати</button>
  </form>
</div>

```

Б.16 Текст файлу set-days.component.ts

```

import { Component } from '@angular/core';
import { FormBuilder } from '@angular/forms';
import { environment } from "../../environments/environment";
import { HttpClient } from '@angular/common/http';
import { SharedService } from "../shared-service";

```

```

@Component({
  selector: 'app-counter-component',
  templateUrl: './counter.component.html'
})
export class CounterComponent {
  public currentCount = 0;
  checkoutForm;
  public forecasts: Forecasting[];

  constructor(private http: HttpClient,
    private formBuilder: FormBuilder,
    private sharedService: SharedService) {
    this.checkoutForm = this.formBuilder.group({
      days: 10
    });
  }

  public incrementCounter() {
    this.currentCount++;
  }

  public onSubmit(data) {
    console.log(data);

    this.http.get<Forecasting[]>(`${environment.apiUrl}AirForecast/${data.station}/${data.days}`)
      .subscribe(result => {
        this.sharedService.newForecasts(result);
        console.log(result);
      }, error => console.error(error));
  }
}

interface Forecasting {
  id: number;
  dateTime: Date;
  pollutionLevel: number;
  pollutantId: number;
  topPollutantName: string;
  topPollutantValue: number;
}

```

Б.17 Текст файлу forecast.component.html

```

<h1 id="tableLabel">Прогнозування для станції {{forecast.stationName}}</h1>

<p>Прогнозовані значення забрудненості:</p>

<p *ngIf="!forecasts"><em>Завантаження...</em></p>

<table class='table table-striped' aria-labelledby="tableLabel" *ngIf="forecasts">
  <thead>
    <tr>
      <th>Дата</th>
      <th>Індекс якості повітря</th>
      <th>Найбільший забруднювач</th>
      <th>Значення забуднювача, мкг/м^3</th>
    </tr>
  </thead>
  <tbody>

```

```

<tr *ngFor="let forecast of forecasts" [ngClass]="(forecast.pollutionLevel == 1)? 'table-
success' : (forecast.pollutionLevel == 2)? 'table-warning' : (forecast.pollutionLevel == 3)?
'table-danger' : (forecast.pollutionLevel == 4)? 'table-dark' : (forecast.pollutionLevel ==
5)? 'table-dark' : 'table-default'">
  <td>{{ forecast.dateTime }}</td>
  <td>{{ forecast.pollutionLevel }}</td>
  <td>{{ forecast.topPollutantName }}</td>
  <td>{{ forecast.topPollutantValue }}</td>
</tr>
</tbody>
</table>

```

Б.18 Текст файла forecast.component.ts

```

import { Component, NgModule, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { environment } from "../../environments/environment";
import { SharedService } from "../../shared-service";

@Component({
  selector: 'app-forecast',
  templateUrl: './forecast.component.html',
})
export class ForecastComponent {

  public forecasts: Forecasting[];

  constructor(private http: HttpClient, private sharedService: SharedService) {

    this.sharedService.forecasts$.subscribe(
      forecast => {
        this.forecasts = forecast;
        console.log(this.forecasts);
      });
  }
}

interface Forecasting {
  id: number;
  dateTime: Date;
  pollutionLevel: number;
  pollutantId: number;
  topPollutantName: string;
  topPollutantValue: number;
}

```

Б.19 Текст файла app-routing.module.ts

```

import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from '../home/home.component';
import { ForecastComponent } from '../forecast/forecast.component';

const routes: Routes = [
  {
    path: '',
    component: HomeComponent,
  },

```

```

    {
      path: 'forecast',
      component: ForecastComponent,
    },
  ],
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

Б.20 Текст файла shared.service.ts

```

import { Injectable } from '@angular/core';
import { Subject } from 'rxjs';

@Injectable()
export class SharedService {

  private forecastSource = new Subject<Forecasting[]>();

  public forecasts$ = this.forecastSource.asObservable();

  public newForecasts(forecasting: Forecasting[]) {
    this.forecastSource.next(forecasting);
  }
}

interface Forecasting {
  id: number;
  dateTime: Date;
  pollutionLevel: number;
  pollutantId: number;
  topPollutantName: string;
  topPollutantValue: number;
}

```

Б.21 Текст файла map.component.css

```

.map {
  height: 70%;
  width: 100%;
}

.map-container {
  height: 600px;
  width: 100%;
}

.map-frame {
  border: 2px solid black;
  height: 100%;
}

#map {
  height: 100%;
}

```

Б.22 Текст файла map.component.html

```
<div class="map-container">
  <div class="map-frame">
    <div id="map"></div>
  </div>
</div>
```

Б.23 Текст файла map.component.ts

```
import { Component, OnInit } from '@angular/core';
import * as L from 'leaflet';

@Component({
  selector: 'app-map',
  templateUrl: './map.component.html',
  styleUrls: ['./map.component.css']
})
export class MapComponent implements OnInit {
  latitude: number = 40.42;
  longitude: number = -3.69;
  iconWidth: number = 33;
  iconHeight: number = 44;
  zoom = 12;
  stations: any;
  map: any;

  constructor(private sharedService: SharedService) {
    this.sharedService.forecasts$.subscribe(stations => {
      this.stations = stations;
    });
  }

  ngOnInit() {
    this.map = L.map('map', {
      center: [this.latitude, this.longitude],
      zoom: this.zoom
    });

    const tiles = L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
      maxZoom: 19,
      attribution: '&copy; <a
href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'
    });

    tiles.addTo(this.map);

    for (var i = 0; i < len(stations); i++) {
      var marker = L.marker([stations[i].latitude, stations[i].longitude]).addTo(this.map);

      var icon = marker.options.icon;
      icon.options.iconSize = [30, 40];

      marker.setIcon(icon);
      marker.on('click', this.onMarkerClick);
    }
  }
}
```

ДОДАТОК В
Слайди презентації

Міністерство освіти та науки України
Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

Дипломна кваліфікаційна робота магістра на тему:

Дослідження та програмна реалізація методів прогнозування забруднення атмосферного повітря

Підготував: Лукашенко Ю.О
ст.гр.КНТ-129м.

Керівник:

Доцент, к.т.н Льовкін В. М.

2020 р.

Рисунок В.1 – Титульний слайд презентації

2

Об'єкт, предмет, мета розробки

- ▶ Об'єктом дослідження є процес прогнозування забруднення атмосферного повітря.
- ▶ Предметом дослідження є методи прогнозування забруднення атмосферного повітря.
- ▶ Метою роботи є розроблення програмного забезпечення для прогнозування забруднення атмосферного повітря на основі досліджених методів заради зменшення ймовірності перебування людини у потенційно небезпечному для здоров'я середовищі.

Рисунок В.2 – Слайд №2

3

Завдання роботи

- Аналіз програмних аналогів для визначення вимог до розроблюваного програмного забезпечення;
- дослідження методів прогнозування часових рядів шляхом використання рекурентних нейронних мереж типу «Довга короткострокова пам'ять»;
- аналіз та вибір інструментів розробки програмного забезпечення;
- проектування програмної системи;
- реалізація програмного забезпечення;
- тестування програмного забезпечення.

Рисунок В.3 – Слайд №3

4

Індекс якості повітря

Забруднювач	Рівень індексу (на основі концентрації забруднюючих речовин в мкг/м3)				
	1 - дуже гарний	2 - гарний	3 - середній	4 - поганий	5 - дуже поганий
Озон (O ₃)	0-80	80-120	120-180	180-240	240-600
Двоокис азоту (NO ₂)	0-40	40-100	100-200	200-400	400-1000
Діоксид сірки (SO ₂)	0-100	100-200	200-350	350-500	500-1250
Частинки менше 10 мкм (PM ₁₀)	0-20	20-35	35-50	50-100	100-1200
Частинки менше 2,5 мкм (PM _{2,5})	0-10	10-20	20-25	25-50	50-800

Рисунок В.4 – Слайд №4

5

Розроблений метод прогнозування

- Формування вибірки. Обрати станції та забруднювачі. Відокремити тренувальну та тестову вибірки. Обробляти 24 вхідних значення та прогнозувати значення на наступний день;
- створення моделі. Реалізація архітектури LSTM з 8-ма нейронами на 1-му шарі та 2-ма – у прихованому шарі. Використання величини трафіку як уточнюючої характеристики;
- навчання моделі – підбір гіперпараметрів: 8 епох, 25% швидкість навчання, сигмоїда та гіперболічний тангенс як функції активації;
- тестування моделі – порівняння точності та втрат моделі;
- прогнозування майбутніх значень. Використання побудованої моделі, прогнозування значень $(n+1)$ -го дня за n днями, $(n+2)$ -го дня зі зсувом початкових значень на 1 та використанням прогнозованого значення у якості вхідного параметру.

Рисунок В.5 – Слайд №5

6

Архітектура моделі за створеним методом

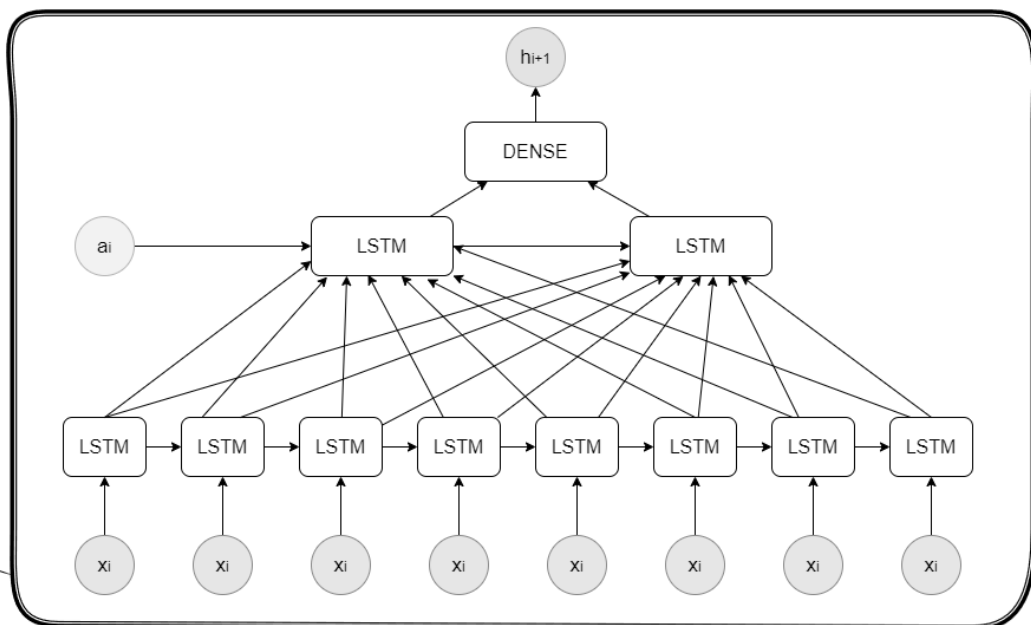


Рисунок В.6 – Слайд №6

Функціональні вимоги до програмного забезпечення

- Аналіз датасету;
- імплементація моделі, її тренування та тестування;
- прогнозування майбутніх значень;
- збереження прогнозу у базу даних;
- створення модулю для аналізу прогнозів, виявлення найбільшого забруднювача, обчислення індексу якості повітря;
- надання API з можливістю давати прогноз для обраної станції на задану кількість днів;
- надання карти міста з позначками станцій;
- надання детальної інформації про прогноз обраної станції: дата, загальний індекс, найбільший забруднювач, його прогнозоване значення;
- надання вибору інтервалу прогнозування.

Рисунок В.7 – Слайд №7

Структурна схема

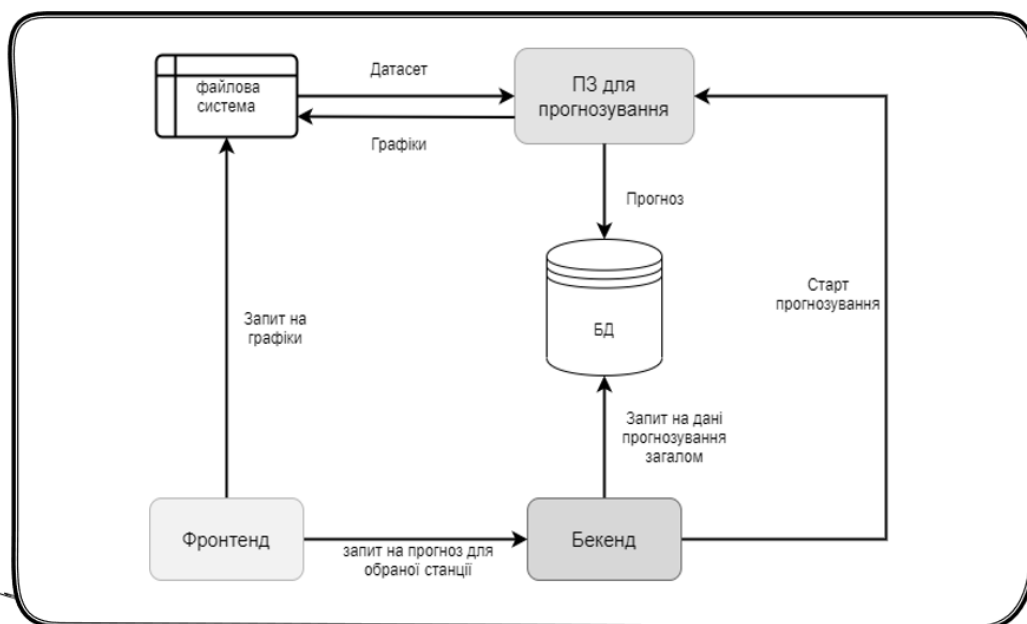


Рисунок В.8 – Слайд №8

9

Діаграма діяльності

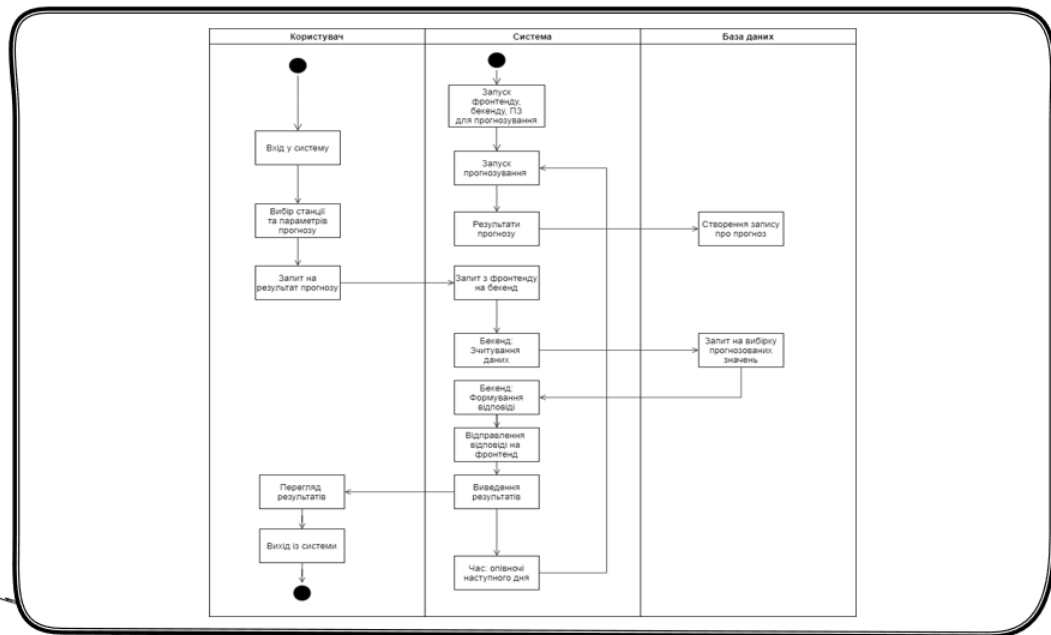


Рисунок В.9 – Слайд №9

10

Схема бази даних

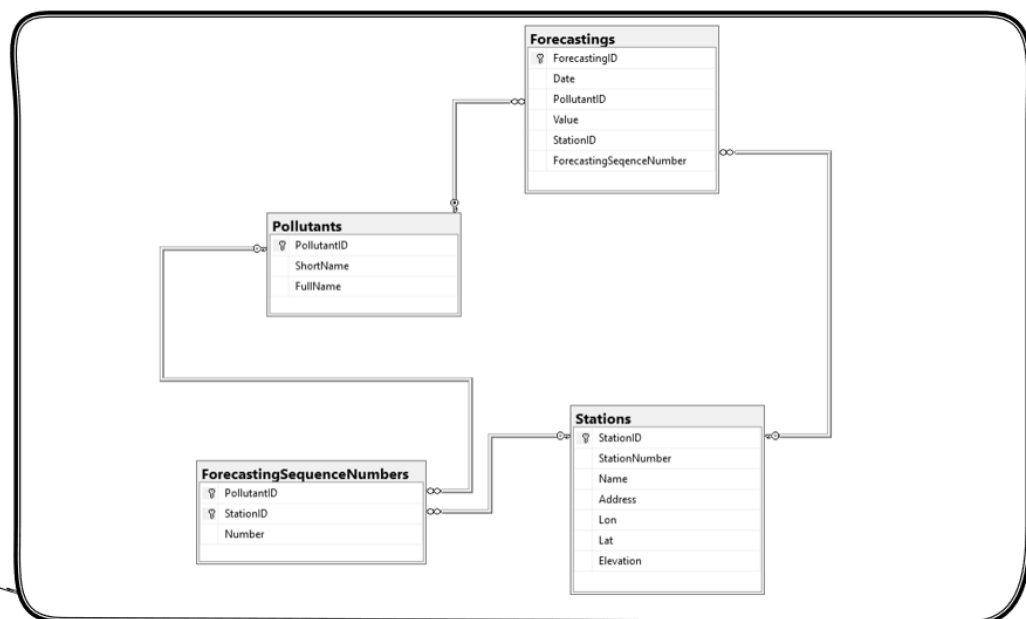


Рисунок В.10 – Слайд №10

Порівняння мов програмування для задач аналізу даних

11

Критерії порівняння	Мови програмування			
	Python	R	SAS	SPSS
Стандартні бібліотеки шаблонів	++	+	+	+
Розробка задач машинного навчання	+	+	+	+
Безкоштовність ПЗ	++	++	-	-
Велика спільнота, яка власноруч створює бібліотеки	+	+	-	-
Підтримка drag-drop функціоналу	-	-	+	+
Робота з великими обсягом даних	+	-	+	-
Масштабованість	+	-	+	+

Рисунок В.11 – Слайд №11

Графічний інтерфейс

12



Рисунок В.12 – Слайд №12

13

Порівняння прогнозів на основі LSTM-моделі та розробленого методу

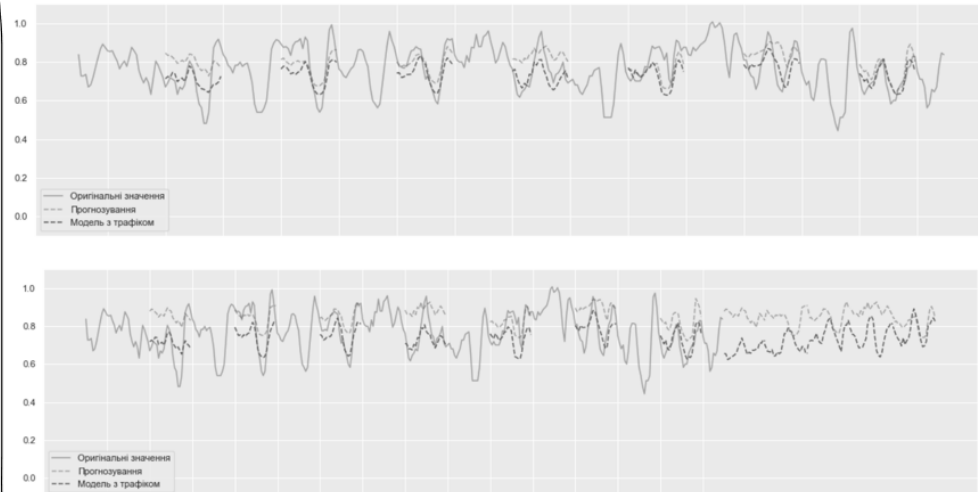


Рисунок В.13 – Слайд №13

14

Показники розробленого методу

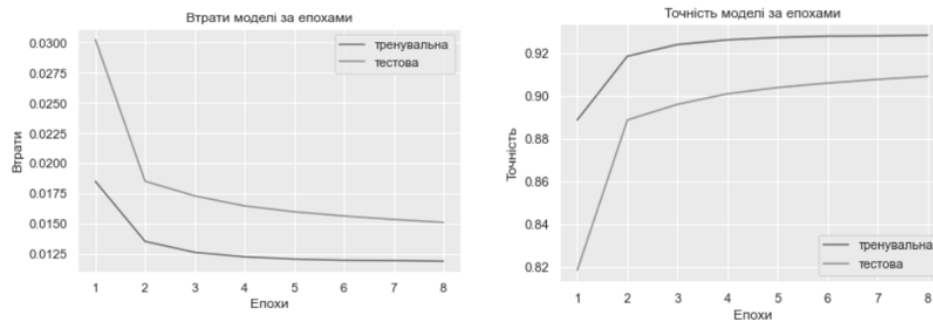


Рисунок В.14 – Слайд №14

Порівняння методів

Метод прогнозування	RMSE	MAE	Точність, %
Модель ARIMA	15.33	10.95	86.35
Штучні нейронні мережі	19.10	12.87	83.96
Рекурентні нейронні мережі	14.95	10.63	86.75
Нейронні мережі типу «Довга короткострокова пам'ять»	13.87	9.13	88.62
Розроблений метод	12.71	7.22	90.98

Рисунок В.15 – Слайд №15

Висновки

- ▶ На основі проаналізованих аналогів було сформовано основні функціональні вимоги до розроблюваної програми;
- ▶ був розроблений метод з використанням моделі LSTM з урахуванням автомобільного трафіку;
- ▶ була повністю реалізована функціональність системи на основі сформованих вимог;
- ▶ розроблений програмний продукт був успішно протестований, були наведені показники розробленого методу для порівняння точності;
- ▶ отримані результати говорять про те, що розроблене програмне забезпечення цілком відповідає поставленій меті розробки.

Рисунок В.16 – Слайд №16