УДК 658.26, 004.421, 004.89 http://doi.org/10.15407/pp2024.04.034

А.Ю. Дорошенко, Я.О. Гайдукевич, В.О. Гайдукевич, О.С. Жиренков

КЛІЄНТО-ЦЕНТРИЧНИЙ ТЕХНОЛОГІЧНИЙ СТЕК ДЛЯ ПРОГНОЗУ ПОГОДИ ТА ЯКОСТІ ПОВІТРЯ

У роботі запропоновано багатоетапний процес доставки прогнозних даних на мобільний пристрій кінцевого користувача при метеорологічному прогнозуванні. Початковим завданням є збір історичних даних про погоду та забруднення повітря. Після цього наступні кроки повинні дозволити побудувати інфраструктуру прогнозування: модель вхідних та вихідних параметрів повинна бути визначена для задачі регресії, алгоритм машинного навчання має бути визначений та його гіперпараметри мають бути оптимізовані. Така модель прогнозування має бути збережена у файлі і використана для створення загальнодоступного веб-сервісу, а на мобільному пристрої має бути встановлений додаток, який би запитував прогнозні дані у сервісу та виводив би їх на екран у вигляді багатопараметричного графіка. Запропонована концепція розподіленого програмного рішення охоплює наступні три головні аспекти цієї архітектури: модель машинного навчання, контейнеризований веб-сервіс та додаток інтерфейсу користувача на мобільному пристрої. Бібліотеки Numpy та Pandas мови Python були використані для підготовки набору даних, бібліотека Scikit-learn і алгоритм Histogram gradient boosting були застосовані для створення регресійної моделі. Наступні технології були використані для побудови веб-сервісу: Docker, Kubernetes, FastAPI та BentoML. Для створення мобільної програми використовувалася платформа Google Flutter.
Ключові слова: задача регресії, машинне навчання, прогнозування, контейнер, веб-сервіс, мобільний застосунок, Google Flutter, Bentoml, Mlops, мікросервісна архітектура.

A.Yu. Doroshenko, Y.O. Haidukevych, V.O. Haidukevych, O.S. Zhyrenkov

USER-CENTRIC TECHNOLOGY STACK FOR WEATHER AND AIR POLLUTION FORECASTING

This paper proposes the multi-stage process of delivering the forecast data to end-user mobile device. The initial task here is to collect the historical weather and air pollution data. After that, the following steps should allow to build the forecasting infrastructure: the input-output model should be defined for regression task, the machine learning algorithm should be trained and its hyperparameters should be optimized, this forecasting model should be serialized to file and it should be used to create the publicly available web-service, the mobile device should have the application installed that would be querying the forecast data from the service and would be displaying the multi-parameter chart on the screen.

The proposed concept of user-centric distributed application covers the following three pillars of this architecture: machine learning model, containerized web-service and user interface application on mobile device. The Python-based libraries Numpy and Pandas were used to prepare the dataset, the Scikit-learn library and Histogram gradient boosting algorithm were leveraged to build the machine learning model. Here are the technologies employed to build the web-service: Docker, Kubernetes, FastAPI and BentoML. The Google Flutter platform was used to build the application for mobile devices.
Key words: regression task, machine learning, forecasting, docker container, web service, mobile application, Google Flutter, Bentoml, Mlops, microservices.

Вступ

Задача прогнозування погоди була актуальною у всі часи. Однак у 21-му столітті екологічна складова стала особливо важливою через зміни клімату та військові ризики. Усі погодні характеристики та параметри забруднення повітря взаємопов'язані та впливають один на одного. Як відомо, збільшення парникових газів призводить до глобального потепління. Отже, ефективний підхід до прогнозування повинен враховувати всі доступні параметри в одній моделі. Саме це і створює додаткові труднощі, оскільки більша кількість вхідних параметрів ускладнює модель, і не всі параметри однаково корисні для цілей прогнозування.

У даній роботі запропоновано багатоетапний процес доставки прогнозних да-

них на мобільний пристрій кінцевого користувача. Початковим завданням є збір історичних даних про погоду та забруднення повітря, після чого будується інфраструктура прогнозування: модель вхідних та вихідних параметрів повинна бути визначена для задачі регресії, алгоритм машинного навчання має бути визначений та його гіперпараметри мають бути оптимізовані. Така модель прогнозування зберігається у файлі і використовується для створення загальнодоступного веб-сервісу, а на мобільному пристрої встановлюється додаток, який виконує запит прогнозних даних до цього вебсервісу та виводить їх на екран у вигляді багатопараметричного графіка.

Набір даних для навчання

Як погодні дані, так і показники забруднення повітря для міста Київ були отримані із сайту openweathermap.org. Зокрема, цей сервіс дозволяє отримати багато атмосферних характеристик для довільних GPS координат. Основні колонки цього набору даних показані на рис. 1. Дана таблиця містить погодинні дані та 33’863 записи в цілому, від 25-11-2020 до 05-10-2024.

Для реєстрації напрямку вітру зазвичай використовуються градуси. Але цей формат не є добрим для алгоритмів машинного навчання [1] через розривність представлення між 359° та 0°. Одним із популярних методів вирішення цієї проблеми є використання сінусу та косінусу відповідного кута [2]. Ці колонки були розраховані з використанням алгоритму написаного на Python.

Забруднення повітря для всіх показників вимірюється у мікрограмах на кубічний метр (μg/m^3). Найбільшим забруднювачем є чадний газ через високу концентрацію. Параметри LevelPM2 та LevelPM10 позначають пилове забруднення частинками до 2.5 та до 10 мікрометрів відповідно. Крім того, значення PM10 включає рівень PM2.5. Частинки PM2.5 шкідливі тим, що потрапляють безпосередньо в кров людини. Частинки більше ніж PM10 зазвичай відфільтровуються в дихальних шляхах та не потрапляють в легені.

Якість прогнозу можна суттєво покращити з використанням циклічних параметрів [2]. Очевидно, добові та річні цикли безпосередньо впливають на температуру повітря. Тижневі цикли пов’язані із забрудненням повітря, тому що багато підприємств не працюють на вихідних. Ці вісім параметрів також були згенеровані з використанням алгоритму, реалізованого на Python. Робота з фреймами даних була спрощена бібліотеками pandas та numpy.

Етап машинного навчання

Для навчання регресійної моделі були обрані наступні набори вхідних та вихідних параметрів:

```
input_columns = \
    ['SineDay', 'CosineDay',
    'SineWeek', 'CosineWeek',
    'SineMonth', 'CosineMonth',
    'SineYear', 'CosineYear',
    'Temperature', 'DewPoint',
    'Pressure', 'Humidity',
    'WindSpeed', 'WindSine',
    'WindCosine', 'CloudLevel',
    'LevelCO', 'LevelNO',
    'LevelNO2', 'LevelO3',
    'LevelSO2', 'LevelNH3',
    'LevelPM2', 'LevelPM10',
    'Temperature-M1', 'DewPoint-M1',
    'Pressure-M1', 'Humidity-M1',
    'WindSpeed-M1', 'WindSine-M1',
```

	A	B	C	D	E	F	G	H	N	O	P	Q	R	S	T	U
1	UtcTime	LocalDate	LocalHour	Temperature	DewPoint	Pressure	Humidity	WindSpeed	LevelCO	LevelNO	LevelNO2	LevelO3	LevelSO2	LevelNH3	LevelPM2	LevelPM10
2	1606266000	2020-11-25	3	3.29	1.01	1022	85	4.38	223.64	0	4.24	49.35	3.46	0.46	3.18	5.08
3	1606269600	2020-11-25	4	3.24	1.13	1021	86	4.23	226.97	0	4.33	48.64	3.73	0.47	3.1	4.89
4	1606273200	2020-11-25	5	3.43	1.79	1021	89	4.09	230.31	0	4.8	46.49	4.05	0.45	3.72	5.5
5	1606276800	2020-11-25	6	3.56	1.76	1021	88	0.45	233.65	0.01	5.83	43.63	4.59	0.48	4.49	6.33
6	1606280400	2020-11-25	7	3.76	1.96	1021	88	4.1	243.66	0.02	8.82	39.34	5.25	0.55	5.24	7.22
7	1606284000	2020-11-25	8	3.71	1.75	1021	87	4.44	250.34	0.04	11.48	36.48	5.84	0.59	5.97	8.04
8	1606287600	2020-11-25	9	3.77	1.81	1021	87	4.78	253.68	0.08	11.82	36.84	6.2	0.59	6.7	8.66
33860	1728154800	2024-10-05	22	15.27	14.96	1012	98	0.45	236.99	0	8.4	43.99	3.22	0.46	7.43	8.24
33861	1728158400	2024-10-05	23	15.07	14.76	1011	98	0.45	230.31	0	6.6	45.42	3.01	0.4	8.15	8.8
33862	1728162000	2024-10-06	0	15.07	14.76	1011	98	0.45	226.97	0	3.98	48.64	2.86	0.34	8.6	9.07
33863	1728165600	2024-10-06	1	15.17	14.86	1011	98	4.25	223.64	0	2.72	50.07	2.86	0.32	8.63	8.96
33864	1728169200	2024-10-06	2	15.08	14.77	1010	98	0.45	223.64	0	2.4	50.78	2.92	0.33	8.52	8.82

Рис. 1. Погодинна інформація про погодні умови та забруднення повітря, м. Київ.

```
    'WindCosine-M1', 'CloudLevel-M1',
    'LevelCO-M1', 'LevelNO-M1',
    'LevelNO2-M1', 'LevelO3-M1',
    'LevelSO2-M1','LevelNH3-M1',
    'LevelPM2-M1', 'LevelPM10-M1']

output_columns = \
    ['Temperature-P1', 'DewPoint-P1',
    'Pressure-P1', 'Humidity-P1',
    'WindSpeed-P1', 'WindSine-P1',
    'WindCosine-P1', 'CloudLevel-P1',
    'LevelCO-P1', 'LevelNO-P1',
    'LevelNO2-P1', 'LevelO3-P1',
    'LevelSO2-P1', 'LevelNH3-P1',
    'LevelPM2-P1', 'LevelPM10-P1']
```

Тут суфікс M1 позначає значення показника годину тому, а суфікс P1 – через годину. В подальшому модель даних може бути покращена, і глибина історії прогнозування може бути збільшена. Але дана модель вже дозволяє ітеративне прогнозування на довільний час наперед.

Набір даних був розділений на навчальну та тестову вибірку у пропорції 80% до 20% за допомогою бібліотечної функції train_test_split. Алгоримом регресії було обрано Histogram Gradient Boosting, що є переможцем багатьох комерційних змагань. Слід відзначити швидкість цього алгоритму, що досягається за рахунок квантизації векторів навчальної вибірки. Наприклад, фаза тренування на поточному наборі даних потребувала лише 45 секунд. Натренована модель була збережена у файл у форматах ONNX та Pickle [3–5] для подальшого використання. Точність прогнозування на годину наперед наведена в табл. 1.

Таблиця 1

Точність прогнозування на годину наперед

Параметр	**R2 Score**	**MAE**
Temperature-P1	0.995264	0.436919
DewPoint-P1	0.981279	0.590434
Pressure-P1	0.995751	0.340160
Humidity-P1	0.935305	2.676903
WindSpeed-P1	0.146426	0.915481
WindSine-P1	0.491392	0.299113
WindCosine-P1	0.683517	0.234346
CloudLevel-P1	0.871486	7.293946
LevelCO-P1	0.977555	3.846348
LevelNO-P1	0.858722	0.444098
LevelNO2-P1	0.006143	2.336877
LevelO3-P1	0.974631	2.397414
LevelSO2-P1	0.931484	0.511058
LevelNH3-P1	0.926684	0.151170
LevelPM2-P1	0.969548	0.425212
LevelPM10-P1	0.946920	0.537251

Мікросервісний підхід до архітектури програмного забезпечення

У сучасний період розвитку штучного інтелекту (AI) та машинного навчання (ML) важливим є гарантування швидкої розробки та розгортання програмного забезпечення, яке інтегрує розроблені моделі та доводить їх до кінцевого споживача. Мікросервісна архітектура є важливим підходом, що спрощує цей процес завдяки декомпозиції систем на незалежні компоненти. Цей підхід дозволяє командам розробників працювати над різними частинами системи паралельно, що значно прискорює цикл розробки та впровадження нових функцій [6].

Мікросервісна архітектура базується на принципі декомпозиції системи на слабо пов’язані між собою компоненти, кожен з яких виконує окрему функцію. Це дозволяє забезпечувати:

- **Масштабованість (Scalability):** можливість незалежного горизонтального масштабування компонентів.
- **Гнучкість (Adaptability):** підвищення адаптивності системи, швидкості внесення змін як реакції на зміну бізнес-вимог.
- **Ремонтопридатність (Maintain ability):** простота оновлення або заміни окремих сервісів.

MLOps підходи

MLOps – це практичний підхід, який поєднує машинне навчання (ML) та операційні процеси (Ops), спрямований на автоматизацію та оптимізацію життєвого циклу моделей машинного навчання. Цей підхід дозволяє командам швидше впроваджувати нові алгоритми та моделі, забезпечуючи безперервну інтеграцію і доставку (CI/CD) для проєктів машинного навчання [7].

У свою чергу це сприяє підвищенню якості моделей та зменшенню часу виходу на ринок нових продуктів, що є критично важливим у сучасному конкурентному середовищі [8].

На рис. 2 наведено діаграму Ейлера-Венна, яка описує логічне місце MLOps практик у комплексі задач, пов'язаних із машинним навчанням.

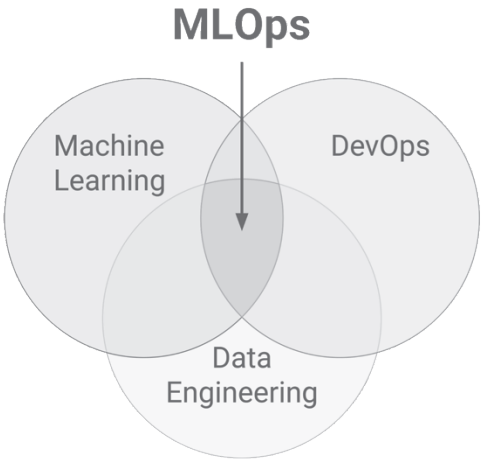

Рис. 2. Місце MLOps серед задач машинного навчання та інженерії програмного забезпечення

Прикладами MLOps фреймворків, які надають подібний інструментарій є:

- **MLFlow** – платформа з відкритим вихідним кодом, яка полегшує управління робочими процесами машинного навчання за допомогою експериментів відстеження, упаковки коду у відтворювані запуски та спільного використання моделей у середовищі співпраці;
- **KubeFlow** – платформа, що спрощує розгортання та управління робочими процесами машинного навчання на Kubernetes, забезпечуючи масштабованість і гнучкість для різноманітних проєктів.
- **BentoML** – платформа, яка дозволяє легко упаковувати та розгортати моделі машинного навчання у вигляді веб-сервісів, що спрощує інтеграцію з існуючими додатками та системами.

Також існує велика кількість комерційних MLOps систем, як-от AWS SageMaker, Microsoft Azure Machine Learning та інші, які не є предметом даного дослідження, але про які необхідно згадати.

У контексті MLOps мікросервіси сприяють:

- **Розподілу завдань:** команди можуть самостійно розробляти, тестувати та розгортати моделі.
- **Контейнеризації:** використання Docker для ізоляції моделей та їх залежностей. Контейнеризація спрощує процес розгортання та повторного використання моделей. Вона також підвищує ефективність управління залежностями, полегшуючи обслуговування.
- **Оркестрації:** автоматизація координації мікросервісів через інструменти, як Apache Airflow [9], Temporal [10], Dagster [11] та інші інструмента оркестрації роботи з даними засновані на формування циклу роботи з даними у вигляді спрямованих ациклічних графів (DAG – Direct Acyclic Graphs).

Запропонована архітектура

Для вирішення задачі побудови клієнто-центричного додатку прогнозування погодних умов з використанням моделей машинного навчання було використано підхід, що передбачає наступні структурні компоненти:

- **Сервіс машинного навчання**. Використано технологію BentoML – MLOps програмне забезпечення для розгортання масштабованих додатків та розміщення моделей будь-якого типу.
- **Серверна частина (Backend)**: FastAPI – фреймворк, призначений для створення швидких та ефективних веб-додатків, які забезпечують REST API, гарантуючи водночас високу продуктивність та простоту інтеграції з іншими сервісами.
- **Оркестратор**: для експерименту було використано короткий python-скрипт, який забезпечує наповнення даних щодо прогнозованих значень погоди. У випадку подальшого розвитку проєкту за рахунок винесення оркестратора в окремий компонент його дуже легко замінити на більш масштабовані системи по оркестрації потоків роботи з даними.

Рис. 3 демонструє структуру компонентів системи, що забезпечують інтегра-

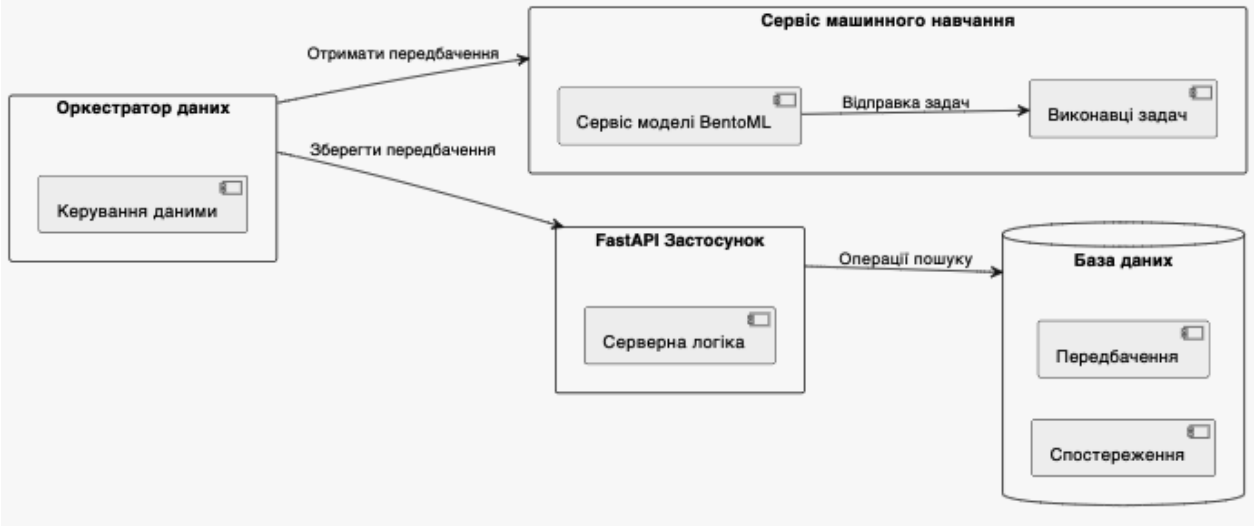

Рис. 3. Запропонована архітектура логіки серверної частини та роботи з моделлю машинного навчання.

цію між різними частинами сервісу прогнозування погоди.

У результаті було отримано набір точок доступу (рис. 4), які дозволяють здійснювати весь комплекс необхідних дій для забезпечення бізнес-логіки.

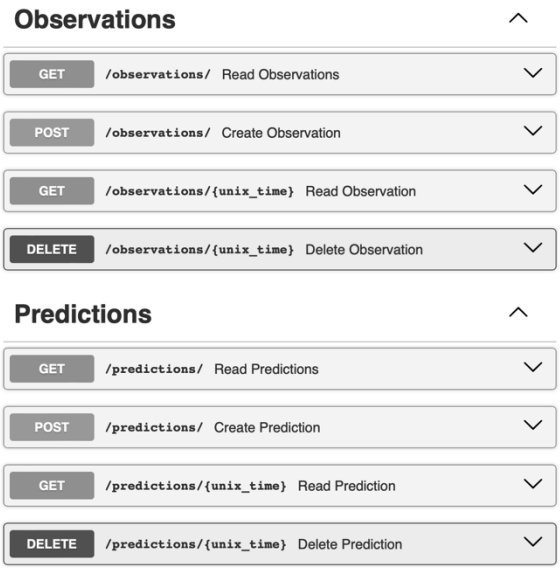

Рис. 4. Набір точок доступу до серверної логіки

Flutter та кросплатформність у створенні інтерактивного екологічного додатку

В основі розробки інтерактивного додатку для моніторингу погоди та якості повітря лежить **Flutter** — сучасний кросплатформний фреймворк, створений компанією Google. Завдяки своїм технічним можливостям Flutter дозволяє створювати додатки з єдиним кодовим базисом, які працюють на Android, iOS, Web та Desktop. Це забезпечує значну економію часу та ресурсів, що є критично важливим у наукових проєктах і стартапах.

Основні переваги Flutter для кросплатформних проєктів:

1. **Єдиний код для всіх платформ**: Flutter дозволяє використовувати одну кодову базу для створення додатку на різних операційних системах, що спрощує його підтримку та оновлення.
2. **Гнучка система компонентів**: Завдяки розгалуженій бібліотеці компонентів (віджетів), включно з унікальними елементами UI, такими як графіки, картографічні компоненти та елементи взаємодії, можна швидко створювати складні інтерфейси.
3. **Швидкість розробки**: Функція "Hot Reload" дає змогу миттєво побачити зміни в інтерфейсі після редагування коду, що суттєво пришвидшує процес розробки.
4. **Продуктивність**: Flutter використовує власне обчислення зображення (без посередників на зразок JavaScript), що забезпечує високу швидкість і плавність роботи додатку навіть на пристроях із низькою продуктивністю.
5. **Широка екосистема бібліотек**: У проєкті використовується ряд поту-

жних бібліотек, які допомагають реалізувати інтерактивність, обробку даних та адаптивний інтерфейс. Наприклад:

I. **Syncfusion Flutter Charts** для візуалізації даних.
II. **fl_chart** та **chart_sparkline** для створення графіків і лінійних графіків (sparkline).
III. **flutter_map** і **latlong2** для інтеграції картографії.

Практичне застосування Flutter у проєкті:

В проєкті Flutter для створення інтерактивної екосистеми використовується компоненти, які дозволяють:

1. Відображати погодні параметри та індекс якості повітря (AQI) на карті України в реальному часі.
2. Інтегрувати графіки та візуалізації для аналізу змін у параметрах довкілля.
3. Забезпечити доступність додатка як на мобільних пристроях, так і у веббраузерах.

Створення інтерактивної системи моніторингу погоди та якості повітря

Інтеграція сучасних технологій у сферу моніторингу стану довкілля є важливим етапом розвитку екологічної свідомості суспільства. Інтерактивні додатки, що поєднують дані про погоду та якість повітря, не лише забезпечують зручний доступ до інформації, а й сприяють глибшому розумінню впливу екологічних факторів на здоров'я людини. Одним із таких рішень є додаток, розроблений на базі Flutter, який дозволяє в реальному часі оцінювати екологічний стан ключових регіонів України.

Головна мета розробки такого додатку полягає в тому, щоб забезпечити інтерактивний доступ до екологічної інформації через карту, яка відображає погодні умови (температура, вологість, швидкість вітру тощо) та рівень забруднення повітря у вигляді індексу якості повітря (AQI). Візуальна складова, а саме кольорові маркери на карті, є важливим елементом дизайну, що сприяє швидкому сприйняттю інформації. Для кожного з індексів AQI визначено колір, який символізує рівень якості повітря: зелений означає "добре", жовтий — "задовільно", помаранчевий — "помірно", червоний — "погано", а темно-коричневий — "дуже погано". Перевага такого підходу полягає в тому, що кольори дозволяють користувачеві миттєво оцінити ситуацію.

Дані для відображення отримуються з API OpenWeatherMap та власним API, в основі якої лежить модель машинного навчання для прогнозу метеорологічних даних. Сервіс надає вичерпну інформацію про погоду та стан атмосфери у форматі JSON. Приклад показників для міста Київ може включати такі параметри: температура — 18°C, тиск — 1015 гПа, вологість — 60%, швидкість вітру — 5 м/с, а також індекс AQI. Усі ці дані проходять обробку та візуалізацію в реальному часі.

Особливістю додатку є його інтерактивність. Наприклад, натискання на маркер міста відкриває вікно з детальною інформацією, де зазначаються погодні параметри та пояснення щодо рівня AQI. Це дозволяє користувачам не лише сприймати сухі цифри, а й отримувати пояснення щодо їх впливу. Окрім того, вбудовані графіки дають змогу аналізувати зміни певних параметрів протягом доби чи тижня.

Розробка таких додатків має значний потенціал для суспільного блага. З одного боку, вони підвищують екологічну обізнаність населення, а з іншого — можуть бути корисними для спеціалістів: екологів, метеорологів, міських планувальників. Наприклад, дані про якість повітря в конкретному регіоні можуть бути використані для визначення впливу промислових викидів або транспортного навантаження.

Отож, інтерактивний додаток для моніторингу погоди та якості повітря є сучасним інструментом, який сприяє зручному доступу до важливої екологічної інформації. Завдяки об'єднанню актуальних даних, інтерактивних візуалізацій і привабливого дизайну проєкт робить внесок у побудову свідомого ставлення до екології та здоров'я населення.

Візуалізація даних за допомогою інтерактивних графіків

Одним із ключових елементів інтерактивного додатку є графічне представлення прогнозів, яке дозволяє користувачам не лише отримувати поточні дані, а й аналізувати їхні зміни у часі (див. рис. 5). Завдяки інтеграції з API OpenWeatherMap та іншими метеорологічними сервісами, додаток пропонує прогнозування важливих параметрів, таких як температура, вологість, швидкість вітру, атмосферний тиск та рівень забруднення повітря (AQI) на кілька днів наперед.

Особливу увагу приділено візуалізації графіків, які дають змогу чітко простежити динаміку зміни погодних умов та рівня забруднення повітря протягом доби, тижня чи навіть місяця. Графічне подання цих даних не лише робить їх легшими для сприйняття, а й допомагає користувачам передбачати можливі зміни в екологічній ситуації. Наприклад, на графіках можна побачити, як змінюється температура та рівень забруднення повітря в певному регіоні залежно від часу доби або тижня, що дозволяє оцінити ймовірність виникнення шкідливих екологічних умов, таких як високий рівень AQI в певні періоди часу.

Рис. 5. Візуалізація показників AQI та метеорологічних даних у реальному часі

Графіки прогнозів можуть бути представлені як лінійні діаграми (рис. 6), що з'єднують значення певних параметрів на різних часових інтервалах, так і як гістограми, що дозволяють більш наочно відображати порівняння між різними параметрами, як-от, рівнем забруднення на різних локаціях. Це дає змогу швидко зрозуміти, де саме і коли рівень забруднення може бути найбільш небезпечним для здоров'я людей.

Завдяки інтерактивним графікам, користувач може взаємодіяти з даними, обираючи різні періоди часу для аналізу та отримуючи відображення зміни погодних умов чи AQI за вибраний інтервал. Це дозволяє не тільки спостерігати за поточною ситуацією, а й робити прогнози на майбутнє, враховуючи сезонні чи погодні тенденції. Важливою перевагою є також можливість адаптувати графіки під різні типи даних, що робить додаток корисним як для простих користувачів, так і для професіоналів, таких як екологи та метеорологи, які можуть детальніше аналізувати прогнози для конкретних регіонів.

Таким чином, графічні прогнози у додатку не тільки сприяють кращому розумінню екологічної ситуації, а й допомагають в ухваленні обґрунтованих рішень щодо захисту здоров'я та навколишнього середовища.

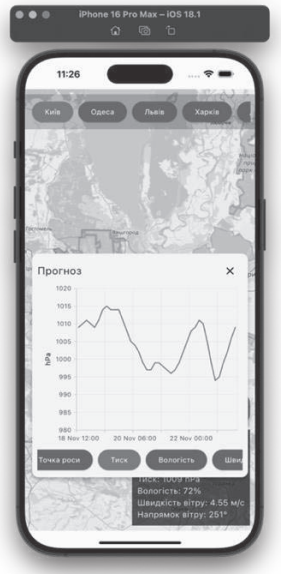

Рис. 6. Графічний прогноз метеорологічних даних на основі машинного навчання

Бібліотека **Syncfusion Flutter Charts** є потужним інструментом для візуалізації великої кількості даних у вигляді графіків, що допомагає здійснювати глибокий аналіз й ухвалювати обґрунтовані

рішення. Вона дозволяє інтегруватися з різноманітними джерелами даних, зокрема, через API, що робить її ідеальним вибором для прогнозування і моніторингу таких параметрів, як погодні умови, рівень забруднення повітря, чи навіть для аналізу економічних показників.

Такі інструменти використовуються в дослідженнях, де необхідно інтерактивно демонструвати дані з часом, виявляти тренди та аномалії, що є важливими у плануванні міської екологічної політики, прогнозуванні погоди та багатьох інших галузях науки та технологій.

У представленому коді (рис. 7), використовується компонент **SfCartesianChart** для створення лінійного графіку, який відображає дані про зміну певного параметра, наприклад, температури чи рівня забруднення повітря. Графік інтерактивний і дає можливість користувачу взаємодіяти з даними для кращого розуміння та прогнозування.

Це здійснюється за допомогою таких параметрів:

- **tooltipBehavior**: дозволяє показувати підказки, які з'являються при наведенні на точку графіка, що покращує взаємодію з даними;
- **primaryXAxis** та **primaryYAxis**: визначають осі графіка. Ось X використовує категоріальну шкалу для часу, де кожна мітка є датою та часом. За допомогою **AxisLabelFormatter** можна налаштувати відображення дат у зручному форматі (dd MMM HH:mm), що полегшує читання графіка;
- **LineSeries**: це основний тип ряду для лінійних графіків, який приймає дані про час та значення параметрів (наприклад, температура чи AQI). Важливим є те, що параметри даних вказуються як **Map<String, dynamic>**, що дає гнучкість у роботі з різними типами даних.

У цьому випадку дані про час і значення параметрів зберігаються у форматі словника (chartData), де для кожної точки графіка визначено, коли було отримано значення і яке саме значення параметра для цього часу.

```
return SfCartesianChart(
  tooltipBehavior: _tooltipBehavior,
  primaryXAxis: CategoryAxis(
    axisLabelFormatter: (AxisLabelRenderDetails details) {
      DateTime date = DateTime.parse(formattedString: details.text);
      return ChartAxisLabel(
          text: DateFormat(newPattern: 'dd MMM HH:mm').format(date: date), textStyle: details.textStyle);
    },
  ), // CategoryAxis
  primaryYAxis: NumericAxis(
    title: AxisTitle(
      text: yAxisLabel,
      textStyle: const TextStyle(color: ■Colors.black, fontSize: 14),
    ), // AxisTitle
  ), // NumericAxis
  series: <CartesianSeries<dynamic, dynamic>>[
    LineSeries<Map<String, dynamic>, String>(
      name: selectedParam,
      dataSource: chartData,
      xValueMapper: (Map<String, dynamic> data, int _) => data['time'],
      yValueMapper: (Map<String, dynamic> data, int _) => data[selectedParam],
      color: ■Colors.green,
    ), // LineSeries
  ], // <CartesianSeries>[]
); // SfCartesianChart
}
```

Рис. 7. Прогнозування погодних умов за допомогою бібліотеки Syncfusion Flutter Charts

Висновки

В роботі розроблена концепція розподіленого програмного рішення, орієнтованого на користувача, що охоплює наступні три головні аспекти запропонованої архітектури: модель машинного навчання, контейнеризований веб-сервіс та додаток інтерфейсу користувача на мобільному пристрої. Бібліотеки Numpy та Pandas мови Python були використані для підготовки набору даних, а бібліотека Scikit-learn і алгоритм Histogram gradient boosting були застосовані для створення регресійної моделі. Для побудови веб-сервісу були використані технології: Docker, Kubernetes, FastAPI та BentoML. Для створення мобільної програми використовувалася платформа Google Flutter.

References

1. C. M. Bishop, Pattern recognition and machine learning, Springer, 2006. [cited 04.03.2024]. https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf
2. A. Van Wyk, Encoding Cyclical Features for Deep Learning. [cited 04.03.2024]. https://www.kaggle.com/code/avanwyk/encoding-cyclical-features-for-deep-learning
3. Scikit-learn: Machine Learning in Python. [accessed 04.03.2024]. https://scikit-learn.org/stable/
4. Feature selection with scikit-learn library. [cited 04.03.2024]. https://scikit-learn.org/stable/modules/feature_selection.html
5. X. Dupre, O. Grisel, Accelerate and simplify Scikit-learn model inference with ONNX Runtime. [cited 04.03.2024]. https://cloudblogs.microsoft.com/opensource/2020/12/17/accelerate-simplify-scikit-learn-model-inference-onnx-runtime/
6. Ribeiro, J. L., et al. "A Microservice Based Architecture Topology for Machine Learning Deployment."IEEE International Smart Cities Conference, Oct. 2019, https://doi.org/10.1109/ISC246665.2019.9071708.
7. Kreuzberger, Dominik, et al.Machine Learning Operations MLOps Overview Definition and Architecture.
8. Kim, Chorwon, et al.A Microservice-Based MLOps Platform for Efficient Development of AI Services in an Edge-Cloud Environment. Oct. 2023, https://doi.org/10.1109/ictc58733.2023.10392296.
9. Apache Airflow use cases [cited 20.11.2024] https://airflow.apache.org/use-cases/
10. Temporal IO use cases [cited 20.11.2024] https://temporal.io/in-use
11. Dagster use cases [cited 20.11.2024] https://docs.dagster.io/getting-started

Одержано: 29.11.2024
Внутрішня рецензія отримана: 04.12.2024
Зовнішня рецензія отримана: 06.12.2024

***Про авторів*:**

[1,2]*Дорошенко Анатолій Юхимович*,
доктор фізико-математичних наук,
професор, завідувач відділу
http://orcid.org/0000-0002-8435-1451.

[1]*Гайдукевич Ярослав Олегович*,
аспірант.
http://orcid.org/0000-0002-6300-1778.

[1]*Гайдукевич Владислав Олегович*,
аспірант.
http://orcid.org/0000-0002-0614-6778.

[1]*Жиренков Олексій Сергійович*,
аспірант.
http://orcid.org/0009-0007-3124-1359.

Місце роботи авторів:

[1] Інститут програмних систем
НАН України,
тел. +38-044-526-60-33
E-mail: a-y-doroshenko@ukr.net,
yarmcfly@gmail.com,
gaidukevichvlad@gmail.com,
ozhyrenkov@gmail.com

[2] НТУ України «Київський політехнічний інститут імені Ігоря Сікорського»,
факультет інформатики та обчислювальної техніки,
тел. +38-044-204-86-10.