НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
___ Наталія АУШЕВА
«___»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра
спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Моделювання забруднення навколишнього середовища на основі байєсівського аналізу та усереднення моделей»

Виконав:
студент IV курсу, групи ТМ-82
Фіголь Богдан Олександрович _____
Керівник:
доктор технічних наук, професор
Сліпченко Володимир Георгійович _____
Рецензент:
Ст. н. с. Інституту проблем реєстрації
Інформації НАН України, к.т.н.
Сенченко Вячеслав Родіонович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2022

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний
Кафедра автоматизації проектування енергетичних процесів і систем
Рівень вищої освіти перший
спеціальність 122 «Комп’ютерні науки»
освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
”___”_____2022р.

ЗАВДАННЯ
на дипломну роботу студенту
Фіголю Богдану Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Моделювання забруднення навколишнього середовища на основі байєсівського аналізу та усереднення моделей.
керівник роботи Сліпченко Володимир Георгійович д.т.н., професор.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”_8_” червня 2022 р.
№ __965-с_____
2. Строк подання студентом роботи _____
3. Вихідні дані до роботи мова програмування – Python, середовище розробки – PyCharm Communiy.

_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) провести аналіз предметної області та знайти рішення для реалізації прогнозування значення показника забруднення. Розробити програмний продукт. Запустити тестування. За результатами робити зробити висновки.
5. Перелік ілюстративного матеріалу
   1. Актуальність роботи.
   2. Мета дипломної рооти.
   3. Використання байєсівських моделей.
   4. Засоби розробки.
   5. Робота користувача з системою.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”___”_____2021 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	23.02.2022	
2.	Вивчення та аналіз задачі	02.03.2022 – 12.03.2022	
3.	Розробка архітектури та загальної структури системи	13.03.2022 – 17.03.2022	
4.	Розробка структур окремих підсистем	18.03.2022 – 29.03.2022	
5.	Програмна реалізація системи	01.04.2022 – 28.05.2022	
6.	Оформлення пояснювальної записки	28.05.2022 – 03.06.2022	
7.	Захист програмного продукту	01.04.2022 – 28.05.2022	
8.	Передзахист	07.06.2022	
9.	Захист	23.06.2022	

Студент _____ Фіголь Б. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Сліпченко В. Г.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою дипломної роботи є розробка системи, що прогнозуватиме значення забруднювача навколишнього середовища на території кожної області України, на основі метеорологічних даних. Створений програмний продукт розроблений на мові Python. Він має інтуїтивно зрозумілий та інтерактивний інтерфейс користувача.

Дипломну роботу виконано на 61 аркушах, вона містить 1 додаток та перелік посилань на використані джерела з 15 найменувань. У роботі наведено 6 формул та 28 рисунків.

Ключові слова: програмний продукт, забруднення, байєсівська модель, прогнозування.

ABSTRACT

The purpose of this thesis is to develop a system that will predict the value of environmental pollutants in each region of Ukraine, based on known meteorological data. The created software product must be developed in Python. It must also have an intuitive and interactive user interface.

This thesis was completed on 61 sheets, it contains 1 appendix and a list of references to the sources used with 15 titles. There are 6 formulas and 28 figures.

Key words: software product, pollution, Bayesian model, forecasting.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ТЕС – теплоенергетична станція;

LSTM (англ. Long Short-Term Memory) – тип рекурентної нейронної мережі;

Bi-LSTM (англ. Bidirectional Long Short-Term Memory) – розширення традиційних LSTM;

GRU (англ. Gatted Recurent Units) – нейронна мережа, призначена для вирішення проблеми зникаючого градієнта;

CNN (англ. Convolutional Neural Network) – нейронна мережа, створена для обробки структурованих масивів даних;

Qt (англ. PyQt) – програмне забезпечення для створення графічного інтерфейсу;

ARIMA (англ. AutoRegressive Integrated Moving Average) – модель статистичного аналізу, що використовує дані часових рядів.

ВСТУП

Основним забруднювачем повітря в Україні є промисловість, яка виробляє майже вдвічі більше шкідливих викидів, ніж автомобільний транспорт, що становить 65 % і 35 % відповідно.

Серед промислових об’єктів основними забруднювачами повітря є ТЕС, забруднення від якої становить близько 29 % усіх шкідливих викидів в атмосферу. Загалом на енергетичну, металургійну та вугільну промисловість припадає 33, 25 та 23 % всіх забруднюючих речовин, що викидаються в атмосферу, відповідно, хімічна та нафтохімічна промисловість – близько 3 %. Найбільша частка викидів знаходиться в Донецько-Дніпровському регіоні, що становить 79 % від загального обсягу викидів в країні [1].

За оцінками експертів, основними забруднювачами повітря в Україні є чорна металургія, теплова, вугільна, нафтогазова та цементна промисловість. Найбільша частка викидів забруднюючих речовин (41,3 % без вуглекислого газу) припадає на виробництво та розподіл електроенергії, газу та води.

Основною метою цієї роботи є розроблення програмного забезпечення, що дозволяє користувачам передбачати розповсюдження викидів шкідливих речовин за наявними метеорологічними даними. Актуальність проекту полягає у браку зручних та доступних баз даних для території України, які можуть бути використані з цією ціллю та у відсутності подібних аналогів програмного продукту українською або російською мовами.

1. ПОСТАНОВКА ЗАДАЧІ МОДЕЛЮВАННЯ ЗАБРУДНЕННЯ НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Метою цього проекту є розроблення програмного продукту, що надає змогу отримати графічне зображення прогнозування наперед викидів шкідливих речовин за наявності метеорологічних даних. Потенційними користувачами можуть бути представники наукової сфери, а також керівництва на рівні країни, області чи району, оскільки наслідки викидів шкідливих речовин впливають на всі сфери життєдіяльності населення окремих районів або ж цілої країни.

Під час розробки програмного продукту використовувалася мова програмування Python для швидкої обробки даних та коректної обробки моделей.

1.1 Аналогічні системи

Наразі існує велика кількість систем, що аналізують забруднення повітря у світі та роблять прогнози на основі метеорологічних даних. Проте переважна більшість з таких ресурсів в Україні є недоступними для звичайних користувачів. Також варто звернути увагу на те, що доступне програмне забезпечення аналізує забруднення повітря у таких країнах, як Китай, Америка, Корея тощо.

Найбільш популярними методами для передбачення показників забруднення є системи, що використовують методи глибокого навчання. Це моделі нейронних мереж, такі як LSTM, Bi-LSTM, GRU, CNN, CNN-LSTM.

Було знайдено декілька аналогічних систем. Перша система [7] – карта світу з індексом забруднення. Червоним кольором позначено найбільш забруднені території, а зеленим – локації з найкращим показником якості повітря. При наведенні курсора на конкретну територію з’являється вікно з календарем індексів забруднень за останній місяць. Друга система [8] визначає якість повітря за умови

введення міста користувачем або його дозволу на отримання локації свого девайсу. Система має вікно для передбачення забруднення на завтра, проте в цьому вікні немає інформації. Третя система [9] – сайт, що містить інтерактивну 3D модель Землі, яку можна обертати. На ній зображено напрямки вітру та індекс якості повітря.

1.2 Структура системи та її реалізація

Метою проекту є розроблення програмного продукту, що надає змогу отримати прогноз якості забруднення в різних областях України. Потенційними користувачами системи можуть бути як звичайні люди, що турбуються про своє здоров’я, так і науковці та представники екологічних організацій.

Програмний продукт повинен мати такий функціонал:

– Можливість введення даних користувачем та вибір області.

– Отримання прогнозу щодо забруднення певної області та його відображення на карті.

Для створення додатку використовується мова Python, яка містить у собі зручні бібліотеки для обробки і аналізу даних, зображення їх на графіках. Також було використано програмне забезпечення Qt для створення графічного інтерфейсу. Систему було реалізовано за допомогою Байєсівського аналізу даних. Дані зберігаються в файлах таблиць. Для кожної області передбачено файл з показниками температури та тиску за 2020 рік, а також спільний файл для всіх областей, у якому містяться забруднювачі.

1.3 Висновки до розділу

У цьому розділі було проведено дослідження наявних систем-аналогів для аналізу забруднення повітря, а також визначено структуру системи, що розроблюється.

2. АНАЛІЗ ПРОБЛЕМИ ЗАБРУДНЕННЯ НАВКОЛИШНЬОГО СЕРЕДОВИЩА В УКРАЇНІ

Проблема оцінки якості повітря та прогнозу її в майбутньому в Україні є дуже актуальною, оскільки рівень забрудненості впливає на життя та здоров'я людей.

У цьому розділі розглянуто проблему забруднення повітря в Україні. Також описано зібрані дані та виконано їх аналіз.

2.1 Наслідки забруднення повітря

На забруднення повітря впливає велика кількість факторів, таких як викиди хімічних речовин з підприємств паливно-енергетичного комплексу, а також підприємств обробної та видобувної промисловості. Також існує кореляція між такими факторами, як тиск, температура повітря та землі, напрямок та швидкість вітру.

Наслідками забрудненого повітря є виникнення різних респіраторних хвороб, серцевих та онкологічних захворювань.

Основними забруднювачами є оксиди вуглецю, азоту, діоксиди сірки, аміак.

2.2 Збір даних

Для створення програмного продукту було зібрано дані про викиди хімічних речовин у повітря за 2020 рік по областях України. Після обробки даних і видалення найменш шкідливих для людини було обрано тверді зважені речовини, діоксид азоту, аміак, діоксид сірки та метан.

Також було знайдено метеорологічні дані, такі як швидкість вітру, температура повітря, опади, напрямок вітру тощо.

Цільовими змінними, які будуть прогнозуватися, є хімічні елементи. За коваріаційною матрицею відсіяно такі ознаки, як швидкість вітру, відносна вологість, хмарність.

2.3.Байєсівський аналіз даних

Байєсівський аналіз даних означає створення методів і моделей для формування імовірнісних висновків стосовно значень вибраних змінних або параметрів на основі статистичних (експериментальних) даних і експертних оцінок [2].

Процедуру Байєсівського аналізу даних виконують за допомогою таких етапів:

1. Побудова ймовірнісної моделі у вигляді спільного розподілу ймовірностей для усіх ознак, що будуть використані для дослідження.

2. Використання набору даних, що задіяні в процесі. Побудова апостеріорного розподілу для визначення цільових змінних.

3. Перебір можливих параметрів та чутливість моделі до них, оцінка якості побудованої моделі за допомогою метрик. Перевірка висновків та розширення моделі за потреби.

Баєсівський статистичний висновок оцінки параметра θ формулюється у термінах тверджень теорії ймовірності.

Для отримання можливості формування імовірнісних висновків стосовно θ за умови наявності спостережень y необхідно будувати модель у вигляді спільного розподілу ймовірностей для θ і y .

Спільний розподіл записується за формулою 2.1 у вигляді добутку двох щільностей:

$$p(\theta, y) = p(\theta)p(y|\theta) \qquad (2.1)$$

де $p(\theta)$ – апріорний розподіл параметра θ;

p(y| θ) – розподіл даних.

За теоремою Баєса апостеріорна щільність обчислюється за формулою 2.2:

$$p(y|\theta) = \frac{p(\theta,y)}{p(y)} = \frac{p(\theta)p(y|\theta)}{p(y)} \quad (2.2),$$

де $p(y) = \sum_{\theta} p(\theta)p(y|\theta)$[2].

У розробці досліджується неперервна величина – отже, формула 2.3 обчислення ймовірності має такий вигляд:

$$p(y|\theta) = \int p(\theta)p(y|\theta)d(\theta) \quad (2.3)$$

Для прогнозу значення змінної *y* необхідно знати її маргінальний розподіл за допомогою формули 2.4*:*

$$p(y) = \int p(y|\theta)d(\theta) \quad (2.4)$$

Після отримання результатів навчання моделей за допомогою змінної *y* можна обчислити оцінку невідомого значення $\tilde{y}$ спостережуваної змінної за допомогою попередніх спостережень.

Для прикладу:

$y = [y_1, \ldots y_n]$ – масив значень цільової змінної.

$\theta = [\mu, \sigma^2]$ – невідома змінна та дисперсія вимірів відповідно.

$\tilde{y}$ – очікуване значення.

Тоді обчислення виконується за допомогою формули 2.5:

$$p(\tilde{y}|y) = \int p(\tilde{y}, \theta|y)d(\theta) \quad (2.5)$$

2.4. Переваги використання байєсівського підходу порівняно з аналогами

ARIMA було одним із стандартних підходів до прогнозування часових рядів; це потужний алгоритм, який широко використовується в багатьох галузях.
Кожен компонент в ARIMA функціонує як параметр зі стандартним записом. Для моделей ARIMA стандартним позначенням буде ARIMA з p, d і q, де цілі значення замінюють параметри, що вказують на тип використовуваної моделі ARIMA. Параметри можна визначити як:
p: кількість спостережень, що перевіряються на серійну кореляцію;
d: ступінь відмінності;
q: порядок ковзного середнього.

Однак існує багато умов: дані повинні бути стаціонарними, може бути кілька тенденцій, незалежні змінні часто відстають, є сезонність — іноді в одних і тих же даних кілька сезонів, має бути значна кількість доступних даних тощо. З цих причин динамічні ряди є складною статистичною технікою для освоєння, а створення точних прогнозів є досить складним.

Байєсівський підхід пропонує імовірнісний підхід до часових рядів для зменшення невизначеності та включення «попередньої» інформації. Ці моделі називаються динамічними лінійними моделями або структурними часовими рядами (моделі простору стану). Вони працюють, динамічно вписуючи структурні зміни в часовий ряд, тобто, розвиваючи та оновлюючи параметри моделі з часом з додаванням нової інформації. На відміну від цього, ARIMA оцінює параметри ряду, які залишаються фіксованими, а потім використовує оцінку максимальної правдоподібності для визначення прогнозів часових рядів. Байєсівські методи використовують MCMC для генерування оцінок із розподілів.

2.5. Висновки до розділу

У цьому розділі було проведено аналіз проблеми та описано дані. Також було розглянуто Байєсівський підхід, на основі якого розроблено програмне забезпечення.

3. ЗАСОБИ РОЗРОБКИ

Python – одна з відомих мов програмування високого рівня, розроблена в 1991 році. В основному використовується для веб-розробки на стороні сервера, розробки програмного забезпечення, математики, сценаріїв та штучного інтелекту [11].

Python – це гібридна мова, яка підтримує об’єктно-орієнтоване програмування, імперативне, процедурне та, в обмеженій мірі, функціональне програмування. Він розроблений як продукт з відкритим кодом для більшості поширених платформ (Unix, Windows, Mac OS і в більшості дистрибутивів Linux він є важливою частиною інсталяції). У результаті Python має чудову виразність. Програмний код короткий і легко читається порівняно з іншими мовами.

Важливою особливістю Python є продуктивність з точки зору швидкості запису. Це стосується як найпростіших програм, так і дуже великих. Для простих програм ця особливість проявляється насамперед у стислості позначень. Для великих додатків продуктивність підтримується функціями, які використовуються у великомасштабному програмуванні, такими як підтримка природного простору імен, використання винятків, стандартні інструменти модульного тестування тощо. Висока продуктивність пов'язана з доступністю та простотою використання широкого спектру бібліотечних модулів, що дозволяє легко вирішувати завдання з низки областей.

Як середовище розробки було обрано PyCharm Community. PyCharm – це спеціальне інтегроване середовище розробки Python, що надає широкий спектр важливих інструментів для розробників Python, тісно інтегрованих для створення зручного середовища для продуктивної розробки Python, Інтернету та науки про дані [12].

Також на початкових етапах розробки програмного забезпечення використовувалось таке середовище, як Deepnote. Це середовище надає користувачам можливість компіляції програмного продукту частинами, для зручного відображення помилок під час написання програми.

3.1 Бібліотека NumPy

NumPy, що розшифровується як Numerical Python, – це бібліотека, що складається з багатовимірних об’єктів масиву та набору підпрограм для обробки цих масивів. За допомогою NumPy можна виконувати математичні та логічні операції над масивами[13]. Є базовою бібліотекою для вчених і аналітиків, які працюють з Python. Визначає тип для n-вимірного однорідного масиву (найчастіше чисел) і API для роботи з таким масивом. Майже всі бібліотеки, де з'являються більші матриці або таблиці, або побудовані на NumP, або підтримують numpy.array: від pandas для аналізу даних і matplotlib для графіків, до scips.

3.2 Бібліотека Pandas

На найпримітивнішому рівні об'єкти бібліотеки Pandas можна вважати розширеною версією структурованих масивів бібліотеки NumPy, в яких рядки і стовпці ідентифікуються мітками, а не простими числовими індексами. Бібліотека Pandas надає безліч корисних утиліт, методів і функціональності на додаток до базових структур даних.

Може здатися, що об'єкт Series та одновимірний масив бібліотеки NumPy взаємозамінні. Основна різниця між ними – індекс. У той час як індекс масиву NumPy, що використовується для доступу до значень, – цілісний та описується неявно, індекс об'єкта Series бібліотеки Pandas описується явно та зв'язується зі значеннями DataFrame як узагальнений масив NumPy.

Якщо об'єкт Series – аналог одномірного масиву з гнучкими індексами, об'єкт DataFrame – аналог двовимірного масиву з гнучкими індексами рядків та гнучкими іменами стовпців. Аналогічно тому, що двовимірний масив можна розглядати як упорядковану послідовність вирівняних стовпців, об'єкт DataFrame можна розглядати як упорядковану послідовність вирівняних об'єктів Series. Під

"вирівняними" мається на увазі те, що вони використовують один і той самий індекс [3].

3.3 Бібліотека PyBats

PyBatS – це пакет для моделювання та прогнозування байєсівських часових рядів. Він розроблений, щоб забезпечити швидкий аналіз і гнучкі параметри для налаштування форми моделі, попереднього та прогнозованого періоду. Ядром пакету є клас Dynamic Generalized Linear Model. Підтримувані DGLM: Пуассона, Бернуллі, Нормальний (DLM) і Біноміальний. Ці моделі в основному засновані на байєсівському прогнозуванні та динамічних моделях[15].

3.4 Бібліотека Matplotlib

Matplotlib – найбільш популярна бібліотека, що дозволяє візуалізувати дані за допомогою мови програмування Python. Функціонал цієї бібліотеки дозволяє створювати будь-який необхідний розробнику тип діаграм з гарним рівнем налаштування. З допомогою подібної бібліотеки можлива візуалізація даних у таких форматах як: 2D та 3D, що значно спростовує аналіз даних.

3.5 Бібліотека Folium

Folium – це потужна бібліотека Python, яка допомагає створювати кілька типів карт-листівок. За замовчуванням Folium створює карту в окремому файлі

HTML. Оскільки результати Folium є інтерактивними, ця бібліотека дуже корисна для створення інформаційної панелі [10].

3.6 PyQt5

Qt – це набір кросплатформних бібліотек C++, які реалізують високорівневі API для доступу до багатьох аспектів сучасних настільних і мобільних систем. Сюди входять послуги локації та позиціонування, мультимедіа, підключення NFC та Bluetooth, веб-браузер на основі Chromium, а також традиційна розробка інтерфейсу користувача.

PyQt5 – це повний набір прив’язок Python для Qt v5. Він реалізований у вигляді більш ніж 35 модулів розширення і дозволяє використовувати Python як альтернативну мову розробки програм C++ на всіх підтримуваних платформах, включаючи iOS та Android.

PyQt5 також може бути вбудований в програми на основі C++, щоб дозволити користувачам цих програм налаштовувати або покращувати функціональність цих програм.

3.7 Модуль Pickle

Модуль Pickle реалізує алгоритм серіалізації та десеріалізації об’єктів Python. "Pickling" – процес перетворення об'єкта Python в потік байтів, а "unpickling" – зворотна операція, в результаті якої потік байтів перетворюється в об'єкт Python. Потік байтів дуже легко перетворити у файл, тому модуль pickle широко застосовується для збереження та завантаження складних об’єктів у Python.

3.8 Висновки до розділу

У цьому розділі було описано середовище розробки програмного забезпечення. Також був представлений технологічний стек, що складається з бібліотек, які оброблюють дані, та пакетів, що дозволяють зробити візуалізацію і інтерфейс для користувача.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Структура програмного забезпечення

Для розробки програмного продукту було зібрано дані з 22-х областей України. Зібрані дані щодо кількості викидів за 2020 рік таких шкідливих речовин, як: тверді речовини, NO_2, SO_2, амоній, метан. Також були зібрані і оброблені дані за 2020 рік за такими ознаками, як: тиск і температура повітря, температура землі, опади, швидкість вітру, відносна вологість.

Коваріаційна матриця, продемонстрована на рисунку 4.1.

	CO	NO_2	O_3	PM_{10}	$PM_{2.5}$	SO_2	GST	PRS	RHU	SSD	TEM	WIN
CO	1											
NO_2	0.93**	1										
O_3	−0.78**	−0.89**	1									
PM_{10}	−0.84**	0.89**	−0.74**	1								
$PM_{2.5}$	−0.92**	0.95**	−0.80**	0.93**	1							
SO_2	−0.50*	0.59	−0.53	0.59	0.70	1						
GST	−0.51*	0.57*	− 0.30	0.42*	0.54*	0.58*	1					
PRS	−0.39*	−0.41*	0.28	−0.35*	−0.36*	−0.42*	−0.53*	1				
RHU	0.08	0.01	−0.07	0.14	0.06	−0.21	−0.44*	0.28	1			
SSD	−0.27	−0.34*	0.23	−0.23	−0.44*	−0.58*	−0.48*	0.16	− 0.01	1		
TEM	−0.54*	0.64*	−0.45*	0.40*	0.55*	0.54*	0.88**	−0.51*	−0.40*	−0.39*	1	
WIN	−0.06	−0.01	0.06	−0.10	0.04	−0.30*	0.21	0.17	−0.25	−0.35*	0.12	1

Рисунок 4.1 – Коефіцієнти кореляції між концентраціями забруднювачів повітря та метеорологічними змінними протягом 1 січня 2016 р. – 31 грудня 2016 р. [5].

Досліджуючи коваріаційну матрицю можна з’ясувати, що на речовини найбільший вплив мають температура ,тиск повітря та вологість.

4.2 Структура файлів проекту

Структура проекту що показана на рисунку 4.2. складається з папок Algorythms, що містить файл modelka.

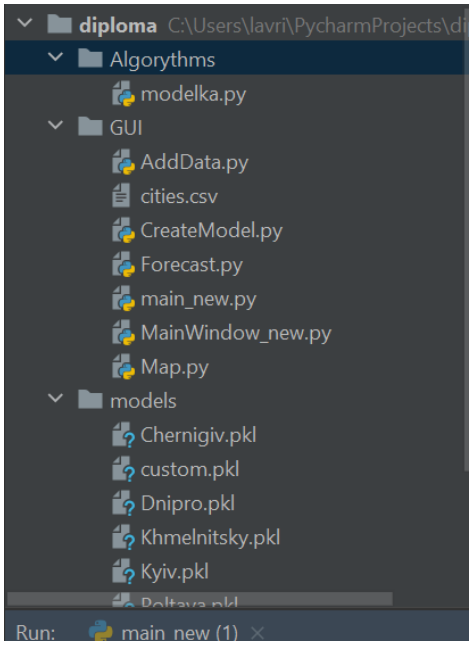

Рисунок 4.2 – Файлова структура

Файлова структура проекту складається з папок Algorythms, що містить файл modelka.Він відповідає за навчання моделі. Папка GUI містить файли AddData, CreateModel, Forecast, MainWindow_new, Map, які створюють графічний інтерфейс, а також файл main_new за допомогою якого запускається програмний продукт. Файл cities.csv – це таблиця, у якій кожен рядок – це назва точки на карті, довгота та широта цієї координати.

Папка models зберігає вже навчені моделі для подальшого їх використання, а також в ній містяться моделі, які створив користувач.

4.3 Підготовка даних та навчання моделі

Для створення програми спочатку виконується обробка даних з джерел. Було обрано Полтавську область для демонстрації роботи програмного продукту. Дані про індекс забруднення та такі ознаки, як температура, тиск та вологість були взяті з електронного джерела [14].

Спочатку необхідно завантажити дані та виконати функцію parse_data_save_eco_bot(). Для того, щоб виконувати навчання моделі, дані мають бути розділені по стовпцях. У файлі Poltava.csv потрібні дані знаходяться в стовпці phenomenon. Їх треба розділити на стовпці “температура”, ”вологість”, ”тиск”, ”забруднювач”. Також існує проблема, що спостереження заносили в таблицю декілька раз на день, тому з цих спостережень береться арифметичне середнє. Після цього потрібно видалити з таблиці ті дані, які потрапили туди випадково чи у невеликій кількості.

Глобальною проблемою самостійного збору даних є існування пустот в даних, тобто пропущених проміжків часу з потрібними показниками. В аналізі часових рядів є методи для того, щоб заповнювати ці проміжки такими значеннями, які підходять до тих, що містяться в даних.

Однією з потужних функцій для заповнення проміжків у часових рядах у бібліотеці pandas є функція resample function. Цей метод дозволяє вказати правило, за яким буде обиратися повторна вибірка.

Ще один спосіб заповнення значень, яких бракує, є forward fill. При такому підході для заповнення відсутнього значення використовується значення, що було перед ним. Наприклад є чотири значення, друге і третє з яких не заповнені. Для них використається те значення, що було перед ними, тобто перше.

Подібним методом є метод backward fill . Цей метод використовує значення, що йдуть після для заповнення відсутніх точок даних.

Останнім розглянутим методом є метод інтерполяції. Дані в ньому змінюються від однієї точки до іншої.

На рисунку 4.3 представлено функцію parse_data_save_eco_bot(), функціонал якої описано вище.

```
def parse_data_save_eco_bot(path, pollution_item):
    usecols = ['phenomenon', 'value', 'logged_at']
    read_func = get_read_func(path)
    data = read_func(path, usecols=usecols)

    data['logged_at'] = pd.to_datetime(data['logged_at'], errors='coerce')
    data['logged_at'] = data['logged_at'].dt.date
    data = data.groupby(['logged_at', 'phenomenon'])['value'].mean().reset_index()
    data['logged_at'] = pd.to_datetime(data['logged_at'], errors='coerce')
    data = data.set_index('logged_at')
    return data
```

Рисунок 4.3 – Функція завантаження даних

Другим етапом роботи з даними є навчання моделі. Це процес, у якому модель машинного навчання (Байєсівська модель) тренується на даних.
Для якісної оцінки цільової змінної виконано аналіз часового ряду та підбір гіперпараметрів. Байєсівська модель з бібліотеки PyBats має такі параметри, як “ntrend”, ”nsamps”, ”seasPeriods”, ”prior_length”, ”deltrend”, ”delregn”, ”delVar”.
Сформовано масив параметрів і за допомогою функції ParameterGrid, реалізація якої показано на рисунку 4.4, створено масив з різних комбінацій для визначення найкращої моделі.

```
    params = {"ntrend": [1, 2], "nsamps": [250, 1000], "seasPeriods": [[10], [12], [14], [16], [18]],
              "prior_length": [22, 24, 26, 30, 34, 36, 38, 40], "deltrend": [0.9, 0.92, 0.94, 0.96, 0.98],
              "delregn": [0.9, 0.92, 0.94, 0.96, 0.98], "delVar": [0.88, 0.9, 0.92, 0.94, 0.96]}

grid_params = ParameterGrid(params)
```

Рисунок 4.4 – Функція ParameterGrid

Фундаментом Баєсівської статистики є те, що усі невизначеності мають бути представлені та виміряні за допомогою ймовірностей. Байєсівська парадигма передбачає прості правила для управління невизначеністю даних, заснованих на законах ймовірностей.

Для аналізу комплексних проблем з різними джерелами невизначеності використовують математичні маніпуляції. Після цього закони теорії ймовірностей та математичної статистики застосовуються для висновків різної кількості випадкових величин, часових рядів і пошуку параметрів для моделей.

Вводяться позначення для моделювання серії дійсних величин, що спостерігаються протягом проміжку часу. Цільова змінна позначається Y.

Для створення моделі навчання використовується бібліотека PyBats, що представляє пакет для моделювання та прогнозування баєсівських часових рядів створюємо функцію bayes_forecast, що містить в собі параметри нашої моделі, а також функцію Analysis, що допомагає швидко виконати повний аналіз часових рядів [4].

Функція складається з таких частин:

1. Ініціалізація DGLM за допомогою define_dglm.

2. Оновлення коефіцієнтів моделі на кожному кроці часу за допомогою dglm.update.

3. Прогнозування на кожному кроці часу між forecast_start і forecast_end за допомогою dglm.forecast_marginal або dglm.forecast_path.

4. Повернення потрібного результату, зазначеного в аргументі ret. За замовчуванням повертає модель та зразки прогнозу.

Після виконання аналізу програма робить передбачення. Реалізація функції bayes_forecast показано на рисунку 4.5. Якість зробленого передбачення можна перевірити за допомогою метрик.

У роботі використано метрику MAPE. Ця метрика також називається середнім абсолютним процентним відхиленням — вимірює точність системи прогнозів. Точність вимірюється у відсотках і розраховується за формулою 4.1.

Використання метрики для розробки програмного продукту показано на рисунку 4.6.

$$M = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \quad (4.1)$$

n – кількість підібраних точок;

At – фактичне значення;

Ft – прогнозне значення;

Σ – позначення підсумовування (абсолютне значення підсумовується для кожного прогнозованого моменту часу).

```
def bayes_model(X, y, ntrend=1, nsamps=10000, seasPeriods=[14], seasHarmComponents=[[1, 2]], prior_length=34,
                deltrend=0.96, delregn=0.98, delVar=0.9, delSeas=0.98, rho=0.6):
    k = 1
    dates = y.index
    forecast_start = dates[0]
    forecast_end = dates[-1]
    mod, samples = analysis(Y=y.values, X=X, family='poisson',
                            forecast_start=forecast_start,
                            forecast_end=forecast_end,
                            k=k,
                            ntrend=ntrend,
                            nsamps=nsamps,
                            seasPeriods=seasPeriods,
                            prior_length=prior_length,
                            seasHarmComponents=seasHarmComponents,
                            deltrend=deltrend,
                            delregn=delregn,
                            delVar=delVar,
                            delSeas=delSeas,
                            rho=rho,
                            dates=dates
                            )
```

Рисунок 4.5 – Функція bayes_forecast

```
forecast = median(samples)

mape = MAPE(y[-18:], forecast[-18:]).round(2)

return mod, forecast, mape
```

Рисунок 4.6 – Метрика MAPE

У циклі виконується навчання моделі Byesian Forecasting на зібраних даних і отримуються графіки часових рядів, де жовтою кривою відмічені «реальні» знанчення, а синьою – передбачення.

4.4 Перебір гіперпараметрів моделі

Для того, щоб зрозуміти, яка з моделей виявиться найкращою, існують різні методи перебору параметрів.

Існує такий метод, як перехресна перевірка. Навчальний набір даних розбивається на менший навчальний набір та навчається на ньому. Оцінка проводиться на перевірочному наборі.

Ще один метод ShuffleSplit дозволяє робити випадкові перестановки. За допомогою цього методу перебору гіперпараметрів можна отримати величезну кількість вибірок.

RandomizedSearchCV – метод, що оцінює кількість випадкових комбінацій і вибирає випадкове значення гіперпараметра на кожній ітерації. Його перевагою є ефективне використання обчислювальних ресурсів.

Решітчастий пошук – оцінює всі можливі комбінації значень заданих гіперпараметрів та шукається найкраща комбінація значень цих параметрів. Для створення програмного продукту було використано такий метод пошуку, оскільки він зручний та швидко реалізується.

Проте, цей метод застосовується на відносно невеликій кількості гіперпареметрів. У функції grid_search(), що показана на рисунку 4.7, робиться перебір з 2×2×5×5×5×5×8 параметрів, тобто навчання виконується 20000 разів, що займає до 3 годин часу для даних однієї області.

```python
def grid_search(X, y, params=None):
    if not params:
        params = {"ntrend": [1, 2], "nsamps": [250, 1000], "seasPeriods": [[10], [12], [14], [16], [18]],
                  "prior_length": [22, 24, 26, 30, 34, 36, 38, 40], "deltrend": [0.9, 0.92, 0.94, 0.96, 0.98],
                  "delregn": [0.9, 0.92, 0.94, 0.96, 0.98], "delVar": [0.88, 0.9, 0.92, 0.94, 0.96]}

    grid_params = ParameterGrid(params)

    scores = []
    for i, p in enumerate(grid_params):
        mv_mod, mv_for, mv_mape = bayes_model(X, y, **p)

        scores.append(mv_mape)

    best_model_indx = np.argmin(scores)
    best_model, best_for, best_mape = bayes_model(X, y, **grid_params[best_model_indx])
    return best_model, best_mape
```

Рисунок 4.7 – Решітчастий пошук

Для навчання можна використовувати одновимірні та багатовимірні часові ряди. У багатовимірних часових рядах таргетна змінна залежить від деяких інших вимірів. Кілька величин змінюються з часом. Наприклад, триосьовий акселерометр. Є три прискорення, по одному для кожної осі (x,y,z), і вони змінюються одночасно з часом.

Одновимірні моделі часових рядів – це клас структурних моделей, де намагаються моделювати та прогнозувати дані, використовуючи лише інформацію, що міститься в їхніх власних минулих значеннях (що нагадує авторегресію), наприклад, на основі вимірів температури повітря протягом року.

На рисунках 4.8, 4.9, 4.10 показано процес навчання моделі з використанням багатовимірних часових рядів, тоді як на рисунках 4.11, 4.12, 4.13 – модель навчається за допомогою одновимірних часових рядів. За результатами навчання моделі, було обрано багатовимірні часові ряди, оскільки показники похибки MAPE були кращими з їх використанням.

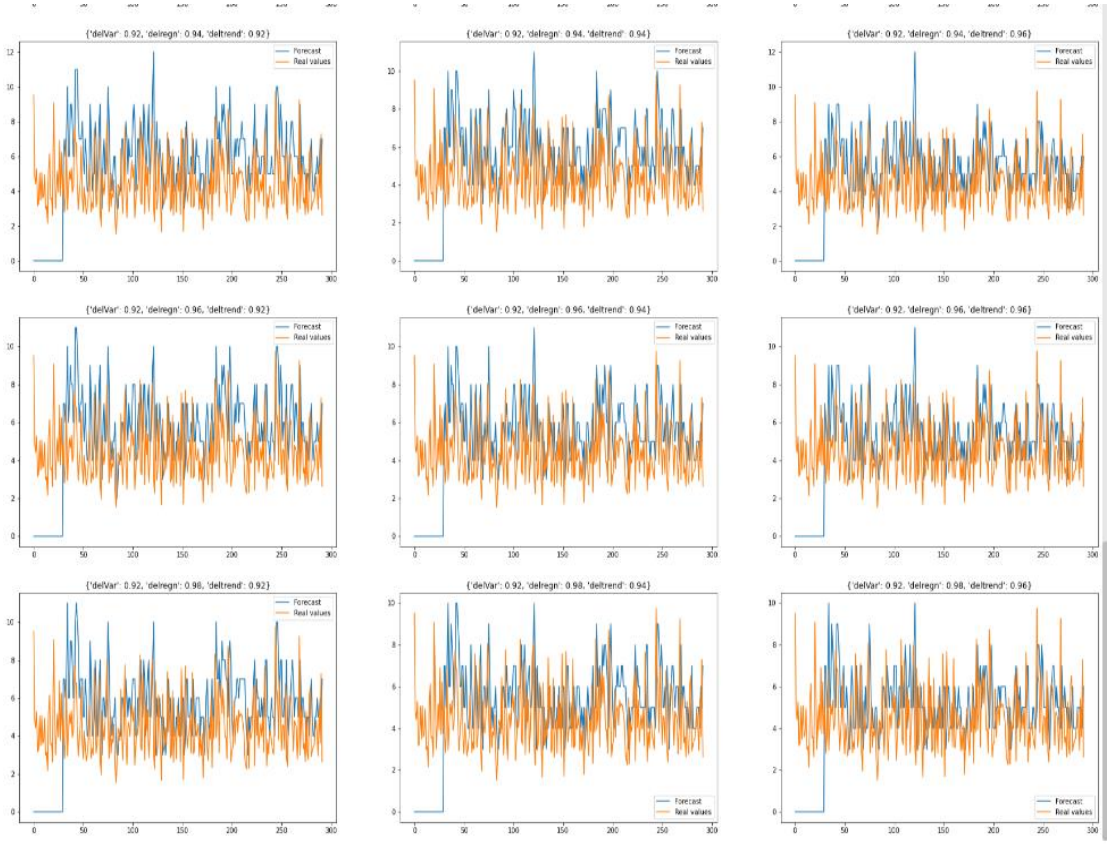

Рисунок 4.8 – Процес навчання моделі з використання багатовимірних рядів

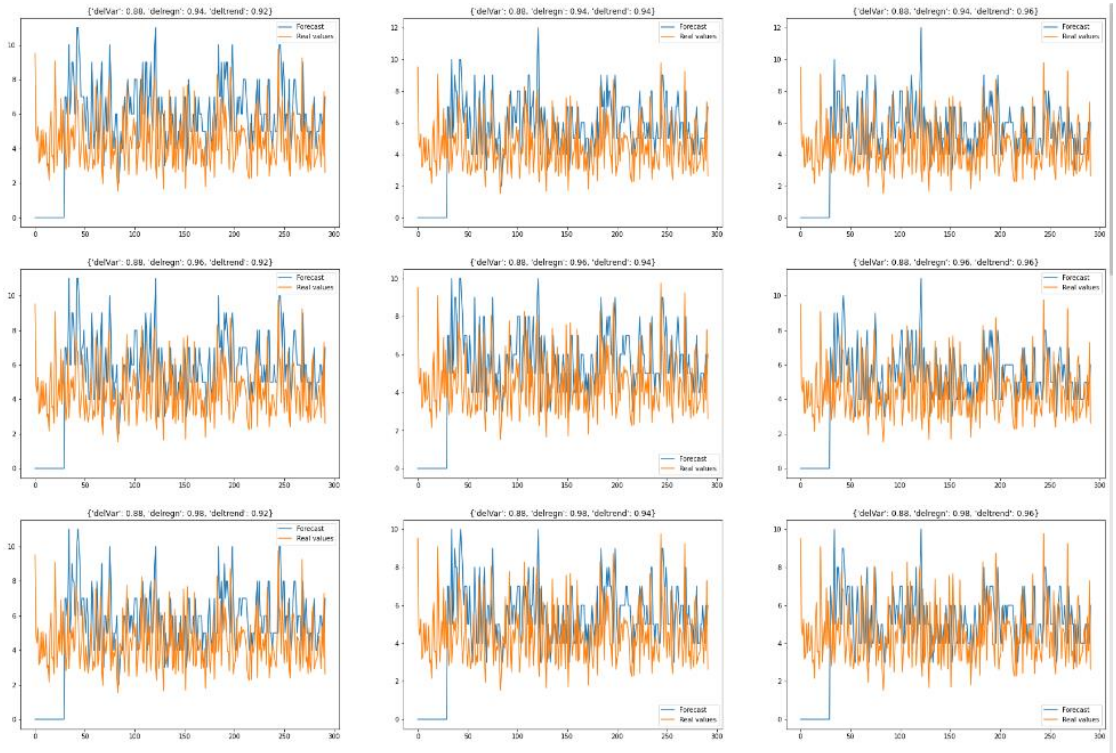

Рисунок 4.9 – Процес навчання моделі з використання багатовимірних рядів

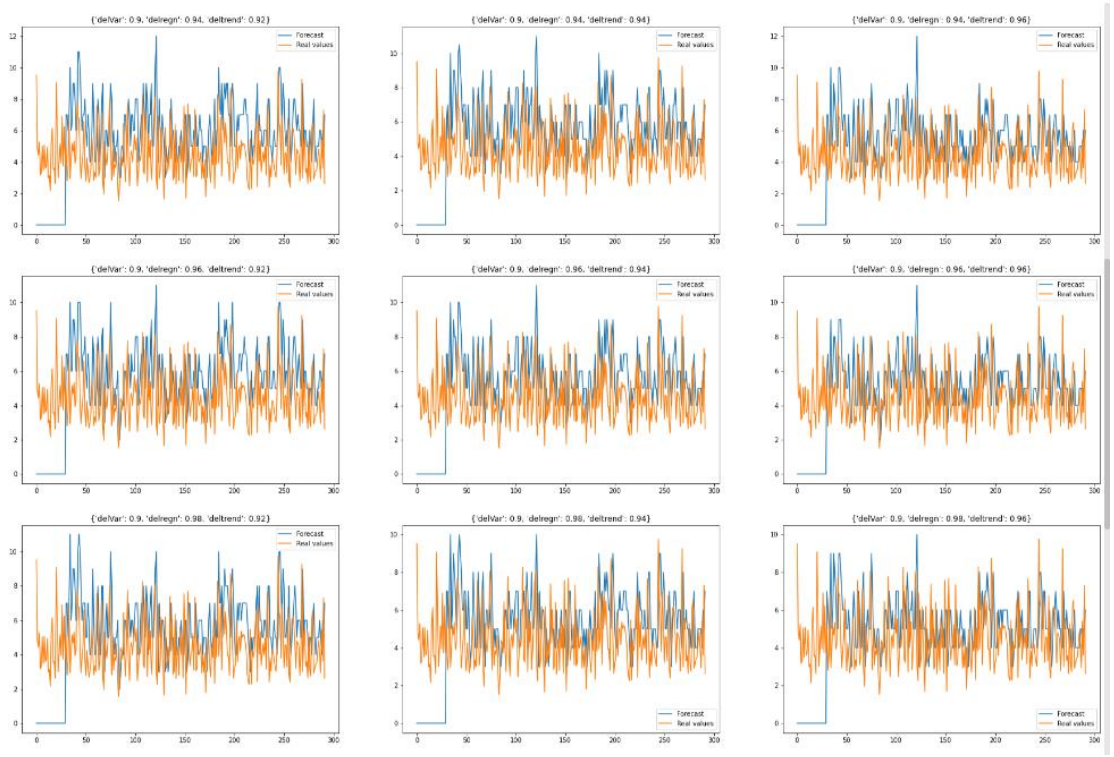

Рисунок 4.10 – Процес навчання моделі з використання багатовимірних рядів

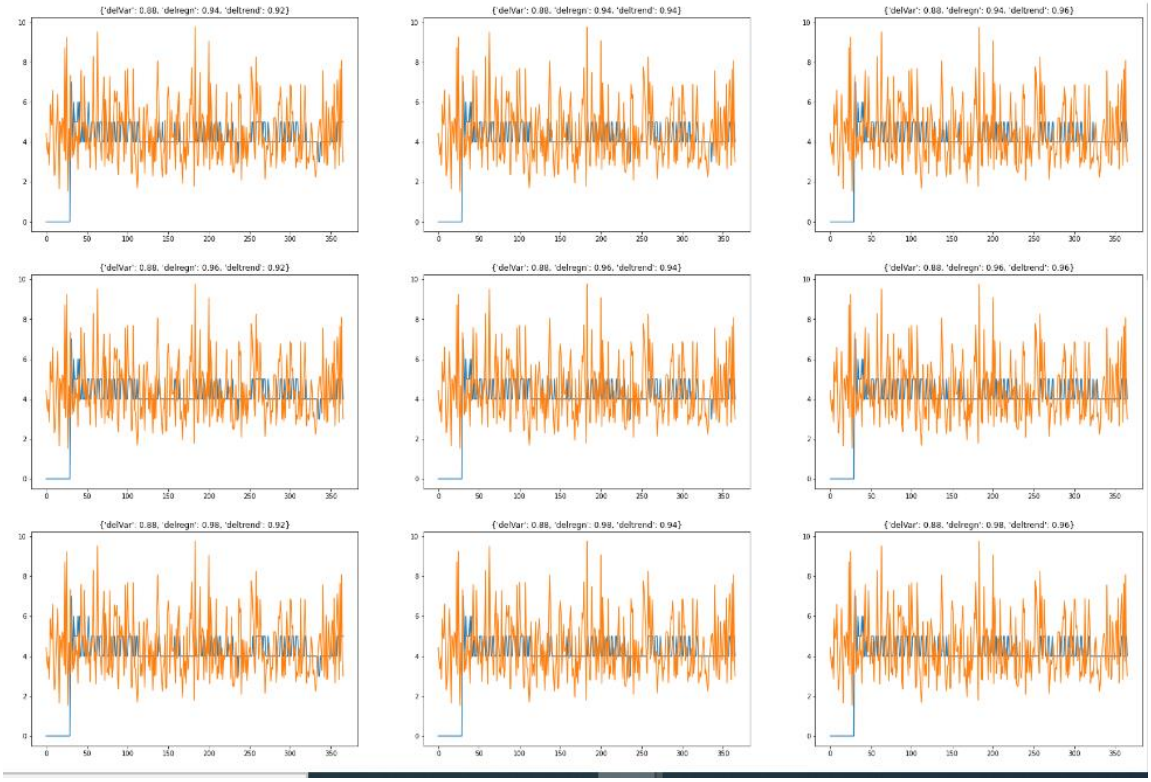

Рисунок 4.11 – Процес навчання моделі з використання одновимірних рядів

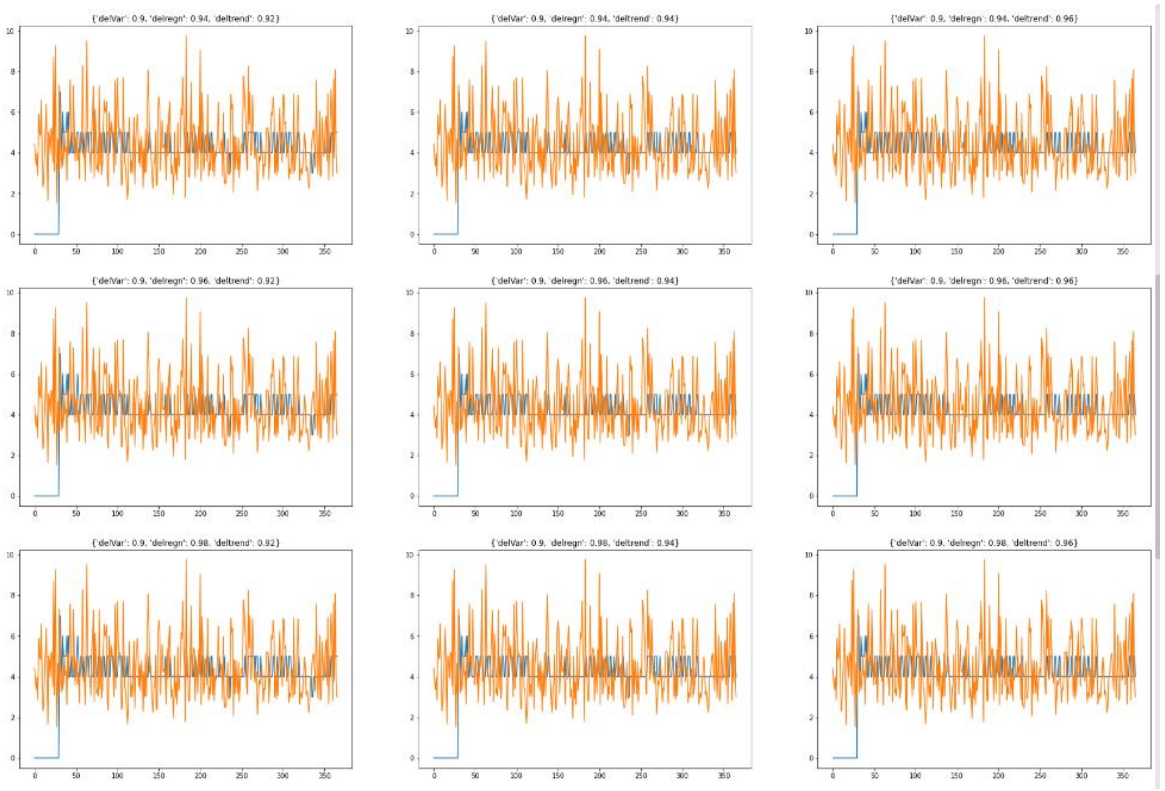

Рисунок 4.12 – Процес навчання моделі з використання одновимірних рядів

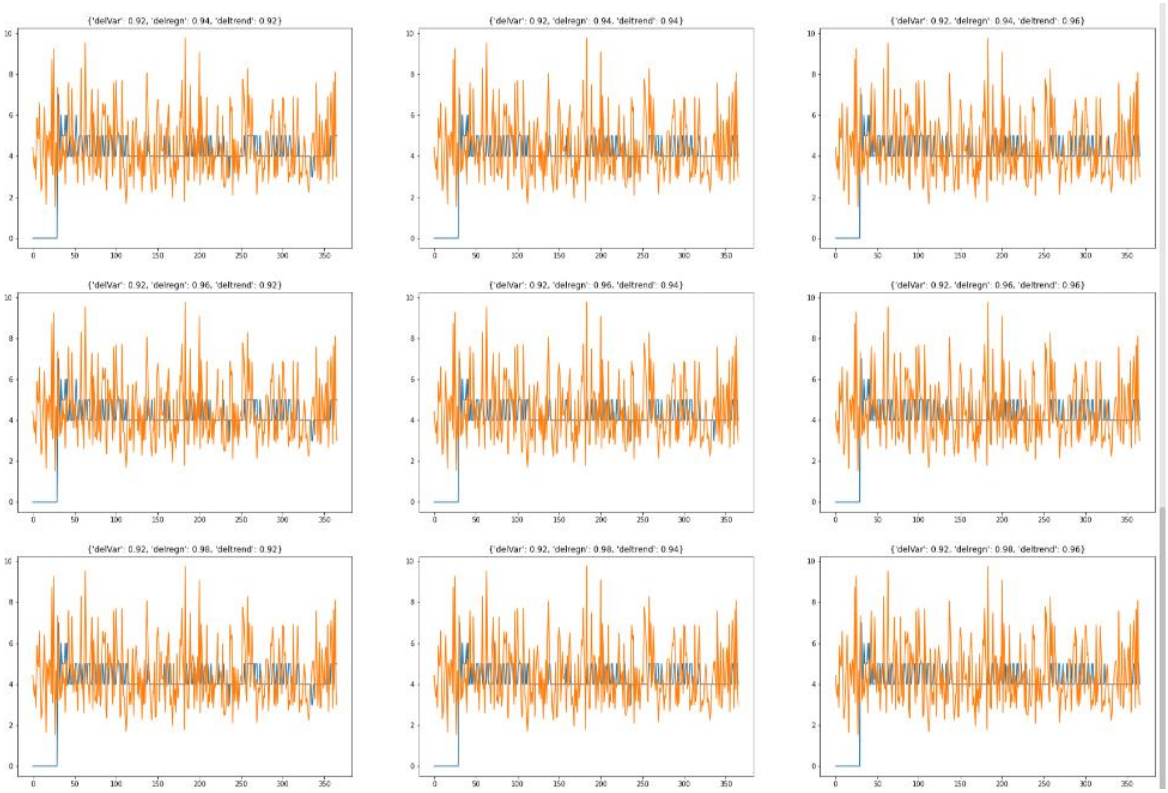

Рисунок 4.13 – Процес навчання моделі з використання одновимірних рядів

Після навчання моделі та перевірки її роботи потрібно її зберегти для майбутнього використання. Тренування моделі кожного разу з самого початку є недоцільним.

Для збереження застосовується функція save_model(), що використовує методи бібліотеки pickle. На рисунку 4.14 показано процес реалізації такої функції.

```
def save_model(model, model_name):
    with open(f'models/{model_name}.pkl', 'wb') as model_file:
        pickle.dump(model, model_file)
```

Рисунок 4.14 – Функція збереження навченої моделі

Для того, щоб зробити прогноз значення забруднювача, викликається функція run_forecast(). Реалізацію цієї функції відображено на рисунку 4.15.

```
def run_forecast(model_name, temperature, humidity, pressure):
    model = restore_model(f'models/{model_name}.pkl')
    return median(model.forecast_marginal(k=1, X=np.array([temperature, humidity, pressure]), nsamps=100000))
```

Рисунок 4.15 – Функція прогнозування значення забруднювача

Ця функція містить такі параметри, як model_name, temperature, humidity, pressure, тобто назву моделі та набір ознак.

4.5 Взаємодія з графічним інтерфейсом

Для взаємодії з графічним інтерфейсом було створено файл main_new.py. У ньому представлений клас MyWindows(), що містить функції для представлення користувачу даних зі збережених моделей, створення нових файлів для навчання, відкриття карти і отримання передбачення значення забруднювача.

Функція forecast_page() створена для того, щоб отримувати значення прогнозу при натисканні кнопки за допомогою функції get_data(). Створену функцію показано на рисунку 4.16.

```
def forecast_page(self):
    self.ui = Ui_Forecast_Window()
    self.ui.setupUi(self)
    self.ui.comboBox_UAReg.addItems(get_models())
    self.ui.pushButton_Result.clicked.connect(self.get_data)
    self.ui.pushButton_Back.clicked.connect(self.mainwindow_page)
```

Рисунок 4.16 – Функція видачі значення прогнозу

Метод get_data() приймає дані, що вводяться користувачем та передає їх у функцію run_forecast(). Розроблений метод показано на рисунку 4.17. Після цього створюється об’єкт класу Map. Цей клас використовує бібліотеку folium для виведення у вікно інтерактивної карти. На карті можна побачити відмітку, на якій зображено значення прогнозу.

```
def get_data(self):
    area = self.ui.comboBox_UAReg.currentText()
    temp = self.ui.lineEdit_AirTem.text()
    hum = self.ui.lineEdit_AirHum.text()
    pr = self.ui.lineEdit_AirPr.text()
    forecastic = run_forecast(str(area), float(temp), float(hum), float(pr))
    print(forecastic)
    loop = QEventLoop()
    QTimer.singleShot(1000, loop.quit)
    loop.exec_()
    map = Map_Window()
    map.my_map(forecastic, area)
    map.show()
    loop = QEventLoop()
    QTimer.singleShot(90000, loop.quit)
    loop.exec_()
    return forecastic
```

Рисунок 4.17 – Метод прийому даних, що вводить користувач

Функція addvalue_page(), що показана на рисунку 4.18, відповідає за додавання нових даних в моделі, створеній користувачем. Вона викликає функціє get_models(), яка при встановленні булевої змінної only_custom = True повертає список моделей, створених користувачем.

```
def addvalue_page(self):
    self.ui = Ui_AddData_Window()

    self.ui.setupUi(self)
    self.ui.pushButton_Back.clicked.connect(self.mainwindow_page)
    self.ui.comboBox_select_model.addItems(get_models(only_custom=True))
    self.ui.pushButton_create_model.clicked.connect(self.add_data)
```

Рисунок 4.18 – Функція додання нових даних до моделі

Після цього викликається метод add_data(), що приймає такі параметри, як model_name, path, pollution_item та передає їх у функцію get_update_data(). Цей метод показано на рисунку 4.19.

```
def add_data(self):
    model_name = self.ui.comboBox_select_model.currentText()
    path = self.ui.lineEdit_path_file.text()
    pollution_item = self.ui.lineEdit_name_pollution.text()
    get_update_data(str(model_name), str(path), str(pollution_item))
```

Рисунок 4.19 – Метод прийому нових даних

Метод get_update_data(), розроблення якого показано на рисунку 4.20, оновлює дані й приймає такі аргументи, як ім'я моделі, шлях до неї та забруднювач.

```
def get_update_data(model_name, path, pollution_item, custom=False):
    model = restore_model(f'models/{model_name}.pkl')
    if custom:
        y, X = parse_data(path, pollution_item)
    else:
        pass
    model = update_model(X, y.values, model)
    save_model(model, model_name)
```

Рисунок 4.20 – Метод оновлення даних

Метод create_model(), що показаний на рисунку 4.21, приймає довготу та широту, яку вводить користувач для того, щоб створити нову координату на карті. Також він створює та тренує модель, щоб використовувати її в майбутньому.

```python
def create_model(path, pollution_item, longtitude, latitude, model_name, from_eco_bot, sep=','):
    append_list_as_row('cities.csv', [model_name, latitude, longtitude, 1])
    if from_eco_bot:
        meteo_data, pollution_data = parse_data_save_eco_bot(path, pollution_item)
    else:
        pollution_data, meteo_data = parse_data(path, pollution_item, sep=sep)
    best_model, best_mape = grid_search(meteo_data, pollution_data)
    save_model(best_model, model_name)
```

Рисунок 4.21 – Метод створення нової координати на карті

У програмі є дві функції parse_data та parse_data_save_eco_bot залежно від того, чи взяті дані конкретно з сайту [14] . Після цього викликається решітчастий пошук для перебору гіперпараметрів нової моделі та пошуку найкращої. Результат зберігається за допомогою методу save_model().

4.6 Висновки до розділу

У цьому розділі розповідається про реалізацію програмного продукту на основі отриманих даних та моделей.

5. РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

5.1 Системні вимоги

Для використання додатку має бути встановлене інтегроване середовище розробки для мови програмування Python та завантажений набір бібліотек: NumPy, Pandas, Folium, Matplotlib, PyQt5, PyBats, Pickle.

5.2 Процес роботи з системою

Для початку роботи з програмним продуктом потрібно запустити файл main_new.

Під час запуску додатку відкривається віконний інтерфейс, що зображено на рисунку 5.1. Це меню, в якому є три опції: Forecast, Create a model, Add value.

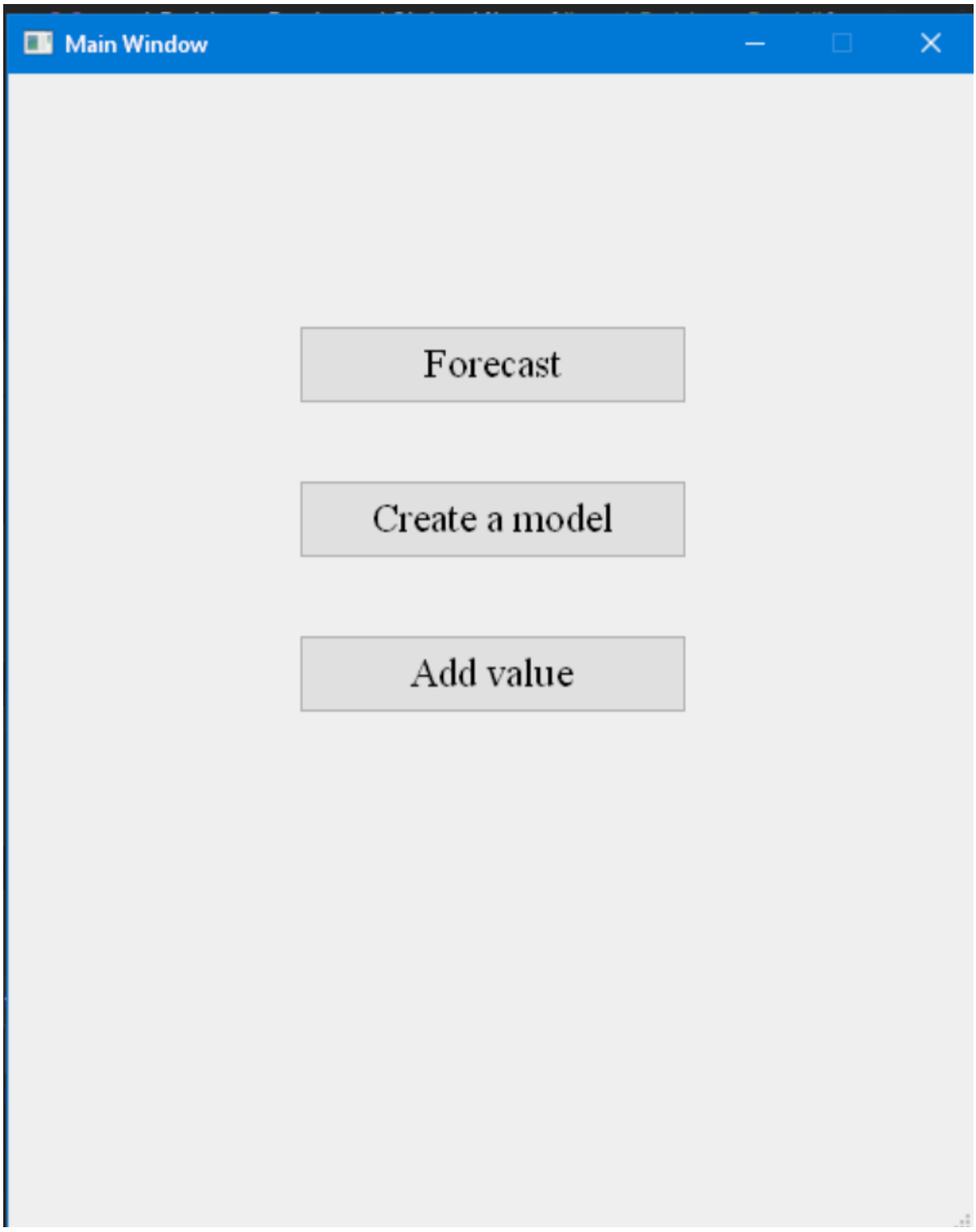

Рисунок 5.1 – Віконний інтерфейс програмного продукту

Якщо натиснути кнопку Forecast, то відкриється вікно(що зображено на рисунку 5.2), в якому треба ввести дані температури повітря, тиску та вологості, а також вибрати область, у якій проводиться дослідження.

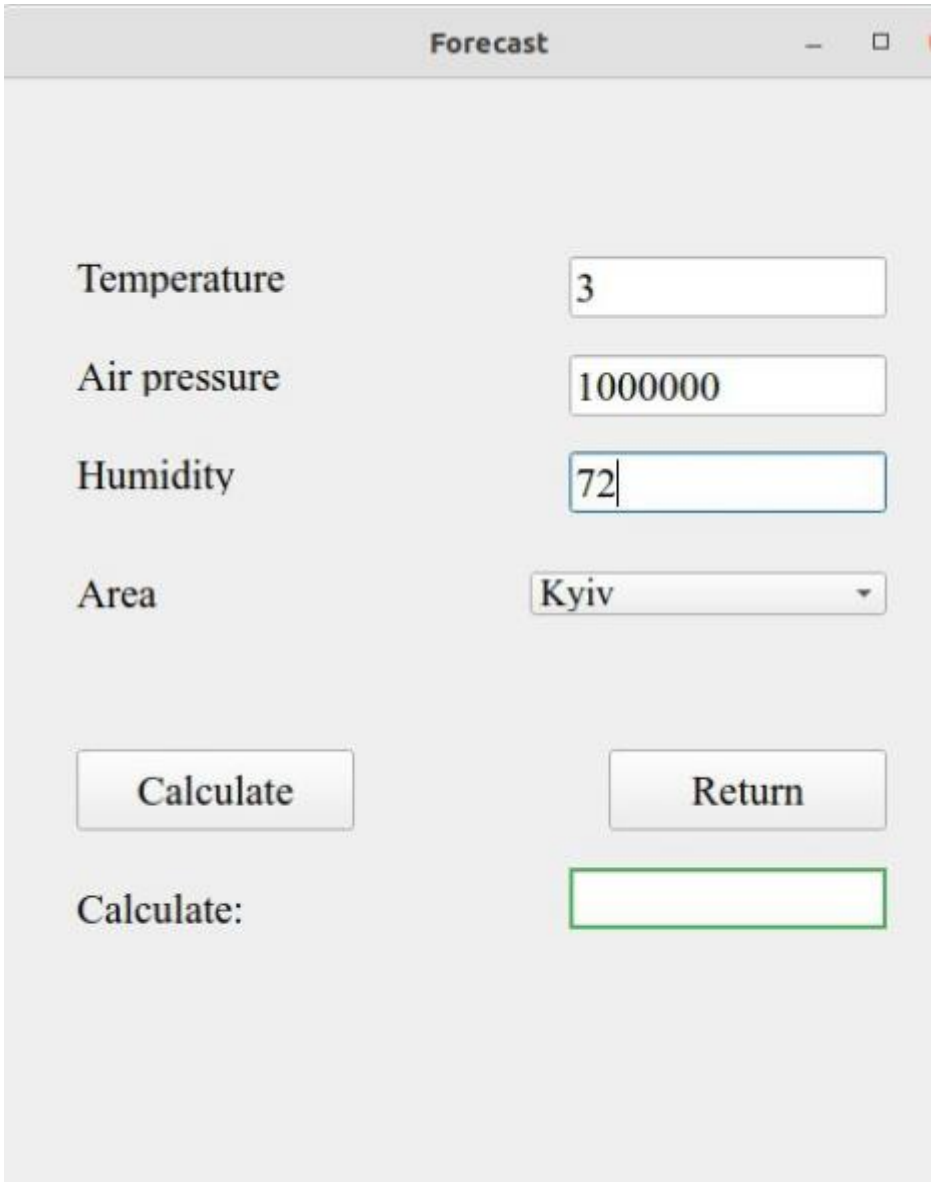

Рисунок 5.2 – Меню Forecast

Після цього відкривається карта, як на рисунку 5.3, на якій зображений прогноз, та область, яку ввели.

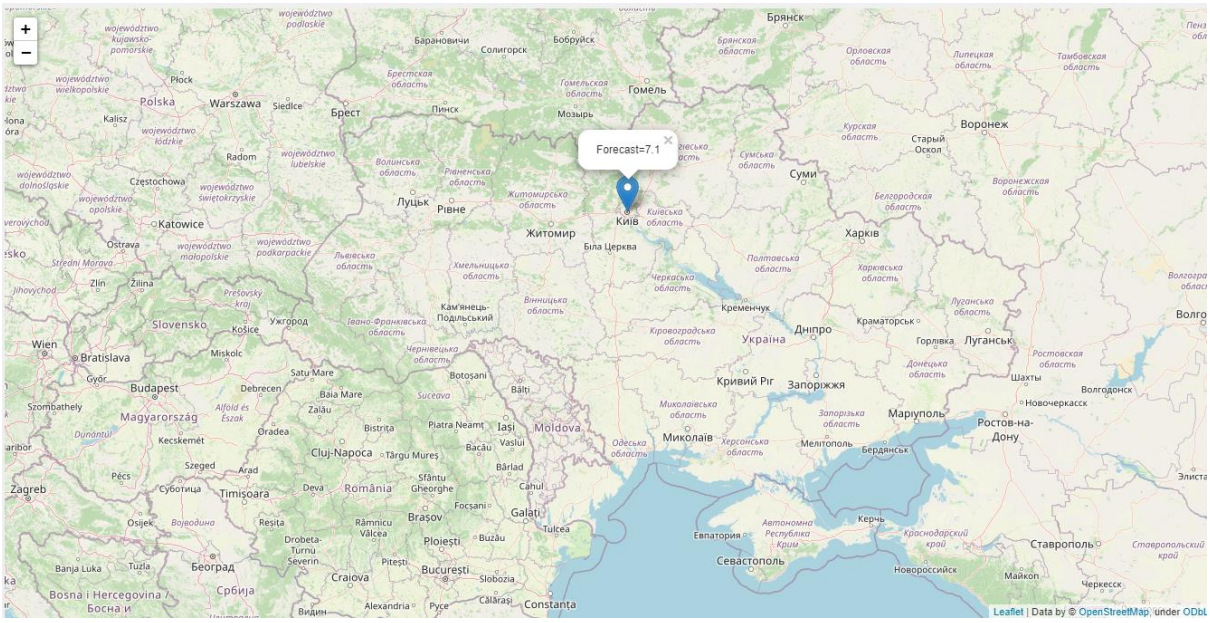

Рисунок 5.3 – Карта з передбаченням викидів у заданій області

Кнопка Create model створює нову модель. Відкривається вікно, що зображено на рисунку 5.4, в якому потрібно ввести ім’я нової моделі, шлях до файлу, який буде оброблятися, довготу та широту, відмітити у полі галочку, якщо дані користувача взяті з сайту [14].

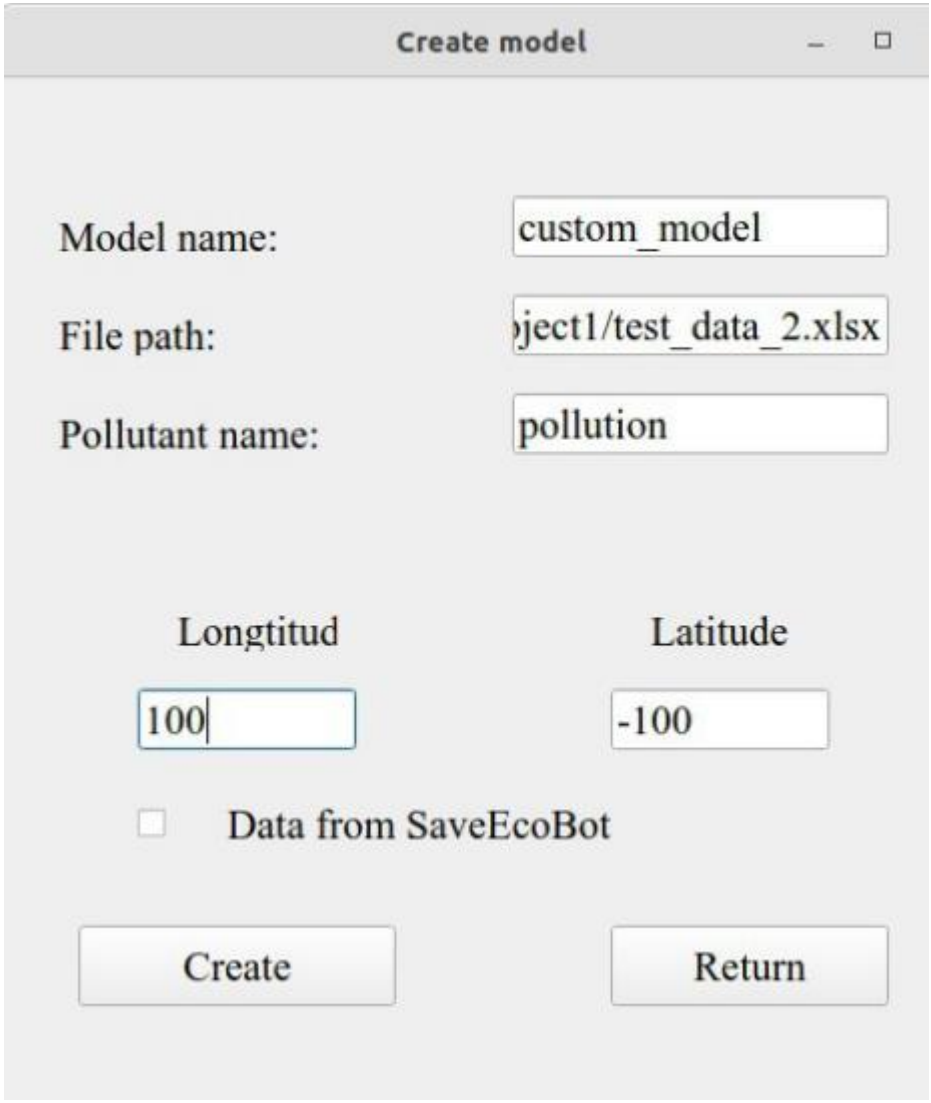

Рисунок 5.4 – Меню створення нової моделі

Після натискання клавіші Create у папці models створиться модель з назвою custom_model. На рисунку 5.5 показано створену користувачем модель.

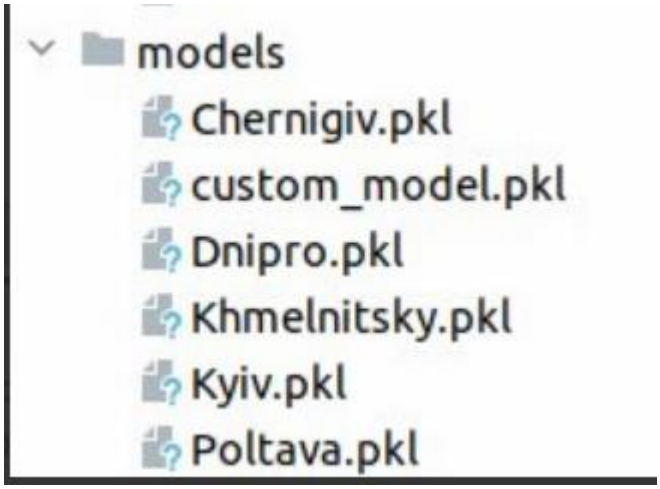

Рисунок 5.5 – Додана модель

У файлі cities.csv з’являється новий рядок з доданою моделлю, що показано на рисунку 5.6.

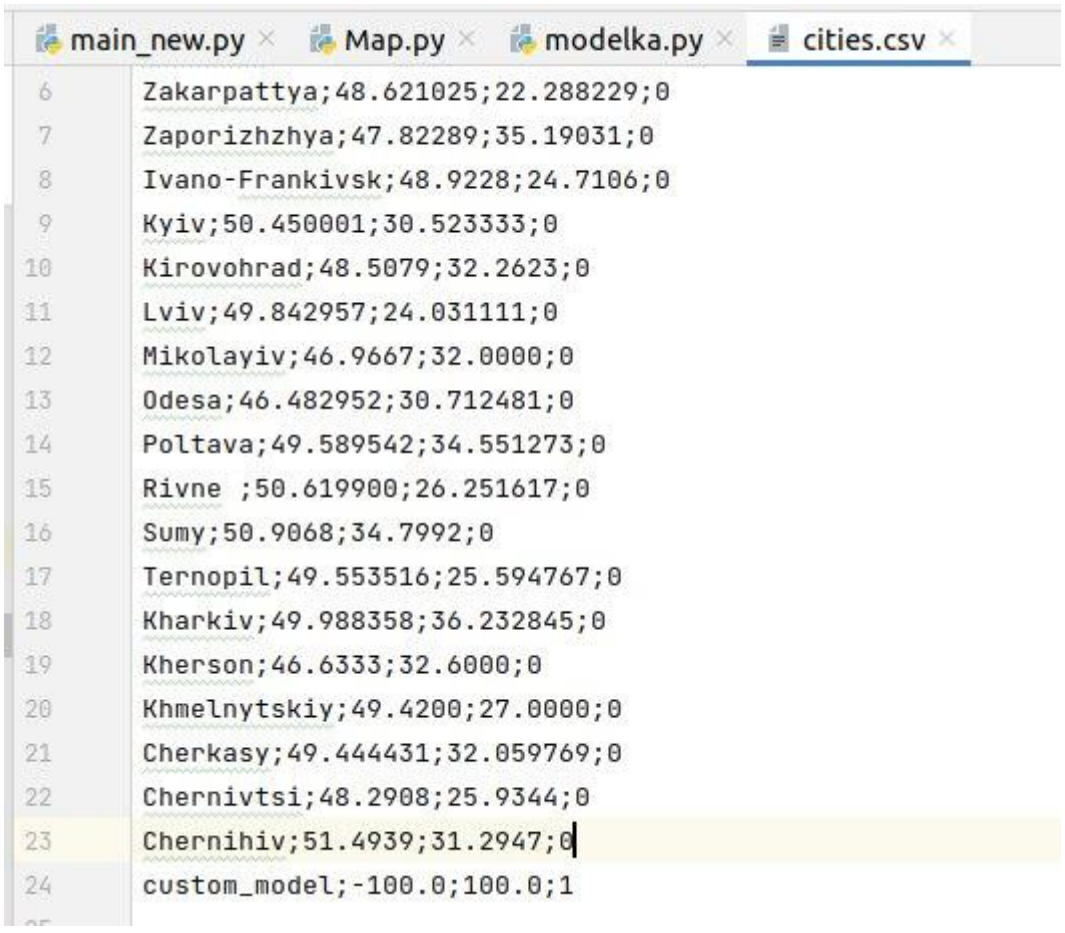

Рисунок 5.6 – Доданий рядок в файлі cities.csv

У додатку передбачена кнопка Add Value, що дає можливість поповнити новими даними модель, яка була додана користувачем. Відкривається вікно, що зображено на рисунку 5.7, в якому необхідно обрати файл, в якому знаходяться нові дані для моделі.

Рисунок 5.7 – Меню додання нових даних до моделі

5.3 Висновки до розділу

Цей розділ містить інформацію стосовно вимог для правильної роботи програмного продукту. Також описано процес взаємодії користувача з комп’ютерним додатком та наведено графічні приклади користування.

ВИСНОВКИ

У ході виконання роботи був розроблений програмний продукт, що визначає кількість забруднювача у повітрі за такими вхідними даними, як тиск та температура. Для розробки використовувалось середовище PyCharm, Deepnote та такі бібліотеки: NumPy, Pandas, PyBats, Matplotlib, PyQt5, Folium.

Необхідність такого додатку була доведена за допомогою проблематики забруднення повітря та браку програми, що дозволяє у вільному доступі спостерігати стан в Україні та робити прогнози.

Таким чином, додаток дозволяє користувачу визначити показники різних хімічних елементів у повітрі відповідно до заданої області й переглянути на карті результати роботи програми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Aar quality in Ukraine. 2021. URL: https://www.iqair.com/ukraine (дата звернення: 15.05.22).
2. Бідюк П. І., Демківський Є. О., Бідюк О. П. Аналіз даних з використанням байєсівських моделей. *Наукові вісті Національного технічного університету України "Київський політехнічний інститут"*. 2012. № 1. С. 40-54. URL: http://nbuv.gov.ua/UJRN/NVKPI_2012_1_7 (дата звернення: 15.05.22).
3. Jake Vander Plas Python Data Science Handbook Essentail Tools For Working With Data. 2018. 576 с.
4. West Mike Bayesian forecasting and dynamic models / Mike West, Jeff Harrisson. *Library of Congress Cataloging – in – Publication Data*. 1997. 637 с.
5. Ding, W., Leung, Y., Zhang, J., & Fung, T. (2021). A hierarchical Bayesian model for the analysis of space-time air pollutant concentrations and an application to air pollution analysis in Northern China. *Stochastic Environmental Research and Risk*. 2021.
6. Forecasting with Bayesian Dynamic Generalized Linear Models in Python. URL: https://towardsdatascience.com/forecasting-with-bayesian-dynamic-generalized-linear-models-in-python-865587fbaf90 (дата звернення: 15.05.22).
7. World's Air Pollution: Real-time Air Quality Index. URL: https://waqi.info/#/c/38.1/50.11/3z (дата звернення: 15.05.22).
8. Air Now. URL: https://www.airnow.gov (дата звернення: 15.05.22).
9. World Air Quality. URL: https://www.iqair.com/earth (дата звернення: 15.05.22).
10. What is Folium? URL: https://www.dominodatalab.com/data-science-dictionary/folium (дата звернення: 15.05.22).
11. 12 BEST Python IDE. URL: https://www.softwaretestinghelp.com/python-ide-code-editors (дата звернення: 15.05.22).
12. PyCharm. URL: https://www.jetbrains.com/help/pycharm/quick-start-guide.html (дата звернення: 15.05.22).

13. What is Numpy in Python. URL: https://www.mygreatlearning.com/blog/python-numpy-tutorial (дата звернення: 15.05.22).
14. SaveEcoBot. URL: https://www.saveecobot.com (дата звернення: 15.05.22).
15. PyBATS. URL: https://github.com/lavinei/pybats/tree/master/ (дата звернення 15.05.22).

ДОДАТОК А

Моделювання забруднення навколишнього середовища на основі байєсівського аналізу та усереднення моделей

Текст програмного модуля

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ82

Аркушів 22

Київ 2022

modelka.py

```
import os
import pandas as pd
import numpy as np
import datetime
import pickle
import pybats
from pybats.loss_functions import MAPE
from pybats.analysis import analysis
from pybats.point_forecast import median
from sklearn.model_selection import ParameterGrid


from csv import writer
def append_list_as_row(file_name, list_of_elem):
    with open(file_name, 'a+', newline='') as write_obj:
        csv_writer = writer(write_obj, delimiter=';')
        csv_writer.writerow(list_of_elem)

def save_model(model, model_name):
    with open(f'models/{model_name}.pkl', 'wb') as model_file:
        pickle.dump(model, model_file)

def restore_model(model_path):
    with open(model_path, 'rb') as model_file:
        model = pickle.load(model_file)
    return model

def run_forecast(model_name, temperature, humidity, pressure):
    model = restore_model(f'models/{model_name}.pkl')
    return median(model.forecast_marginal(k=1, X=np.array([temperature, humidity, pressure]),
nsamps=100000))


def bayes_model(X, y, ntrend=1, nsamps=10000, seasPeriods=[14], seasHarmComponents=[[1, 2]],
prior_length=34,
                deltrend=0.96, delregn=0.98, delVar=0.9, delSeas=0.98, rho=0.6):
    k = 1
    dates = y.index
    forecast_start = dates[0]
    forecast_end = dates[-1]
    mod, samples = analysis(Y=y.values, X=X, family='poisson',
                            forecast_start=forecast_start,
                            forecast_end=forecast_end,
                            k=k,
                            ntrend=ntrend,
                            nsamps=nsamps,
                            seasPeriods=seasPeriods,
                            prior_length=prior_length,
                            seasHarmComponents=seasHarmComponents,
                            deltrend=deltrend,
```

```
                    delregn=delregn,
                    delVar=delVar,
                    delSeas=delSeas,
                    rho=rho,
                    dates=dates
                    )
    forecast = median(samples)

    mape = MAPE(y[-18:], forecast[-18:]).round(2)

    return mod, forecast, mape


def grid_search(X, y, params=None):
    if not params:
        params = {"ntrend": [1, 2], "nsamps": [250, 1000], "seasPeriods": [[10], [12], [14], [16], [18]],
                "prior_length": [22, 24, 26, 30, 34, 36, 38, 40], "deltrend": [0.9, 0.92, 0.94, 0.96, 0.98],
                "delregn": [0.9, 0.92, 0.94, 0.96, 0.98], "delVar": [0.88, 0.9, 0.92, 0.94, 0.96]}

    grid_params = ParameterGrid(params)

    scores = []
    for i, p in enumerate(grid_params):
        mv_mod, mv_for, mv_mape = bayes_model(X, y, **p)

        scores.append(mv_mape)

    best_model_indx = np.argmin(scores)
    best_model, best_for, best_mape = bayes_model(X, y, **grid_params[best_model_indx])
    return best_model, best_mape


def get_read_func(path):
    file_extension = path.rstrip().split('.')[-1]
    excel_extensions = ['xls', 'xlsx', 'xlsm', 'xlsb', 'odf', 'ods', 'odt']
    if file_extension == 'csv':
        return pd.read_csv
    elif file_extension in excel_extensions:
        return pd.read_excel
    else:
        raise Exception(f'Unsupported file extension: {file_extension}.')


def parse_data_save_eco_bot(path, pollution_item):
    usecols = ['phenomenon', 'value', 'logged_at']
    read_func = get_read_func(path)
    data = read_func(path, usecols=usecols)

    data['logged_at'] = pd.to_datetime(data['logged_at'], errors='coerce')
    data['logged_at'] = data['logged_at'].dt.date
    data = data.groupby(['logged_at', 'phenomenon'])['value'].mean().reset_index()
```

```
    data['logged_at'] = pd.to_datetime(data['logged_at'], errors='coerce')
    data = data.set_index('logged_at')
    return data


def split_data(data, pollution_item):
    phenomenon_mask = (data['phenomenon'] == pollution_item)
    pollution_data = fill_timegaps(data.loc[phenomenon_mask]['value'])
    temp = fill_timegaps(data['value'].loc[(data['phenomenon'] == 'temperature')]).to_list()
    humidity = fill_timegaps(data['value'].loc[(data['phenomenon'] == 'humidity')]).to_list()
    pressure_pa = fill_timegaps(data['value'].loc[(data['phenomenon'] == 'pressure_pa')]).to_list()
    meteo_data = np.array([temp, humidity, pressure_pa]).T
    return meteo_data, pollution_data


def get_pollution(data, pollution_item):
    phenomenon_mask = (data['phenomenon'] == pollution_item)
    return data.loc[phenomenon_mask]


def fill_timegaps(series, rule='1D'):
    return series.resample(rule).mean().interpolate()


def parse_data(path, pollution_item, sep=','):
    usecols = ['date', 'temperature', 'pressure', 'humidity', pollution_item]
    read_func = get_read_func(path)
    data = read_func(path, usecols=usecols)
    data['date'] = pd.to_datetime(data['date'], errors='coerce')
    data = data.set_index('date')
    pollution_data = fill_timegaps(data[pollution_item])
    temp = fill_timegaps(data['temperature']).to_list()
    pressure = fill_timegaps(data['pressure']).to_list()
    humidity = fill_timegaps(data['humidity']).to_list()
    meteo_data = np.array([temp, humidity, pressure]).T
    return pollution_data, meteo_data


def create_model(path, pollution_item, longtitude, latitude, model_name, from_eco_bot, sep=','):
    append_list_as_row('../GUI/cities.csv', [model_name, latitude, longtitude, 1])
    if from_eco_bot:
        meteo_data, pollution_data = parse_data_save_eco_bot(path, pollution_item)
    else:
        pollution_data, meteo_data = parse_data(path, pollution_item, sep=sep)
    best_model, best_mape = grid_search(meteo_data, pollution_data)
    save_model(best_model, model_name)

def get_models(only_custom=False):
    models = pd.read_csv('../GUI/cities.csv', usecols=['cities', 'created'], sep=';')
    if only_custom:
        models = models.loc[models['created'] == 1]
```

```
    return models['cities'].to_list()


def update_model(X, y, model):
    for observation, value in zip(X, y):
        model.update(y=value, X=observation)
    return model

def get_update_data(model_name, path, pollution_item, custom=False):
    model = restore_model(f'models/{model_name}.pkl')
    if custom:
        y, X = parse_data(path, pollution_item)
    else:
        pass
    model = update_model(X, y.values, model)
    save_model(model, model_name)
Forecast.py
from Map import *

class Ui_Forecast_Window(object):
    def setupUi(self, Forecast_Window):
        Forecast_Window.setObjectName("Forecast_Window")
        Forecast_Window.resize(500, 600)
        Forecast_Window.setMaximumSize(QtCore.QSize(500, 600))
        Forecast_Window.setToolTipDuration(0)
        Forecast_Window.setStyleSheet("font: 16pt \"Times New Roman\";\n"
"color:black;")
        self.centralwidget = QtWidgets.QWidget(Forecast_Window)
        self.centralwidget.setObjectName("centralwidget")
        self.label_AirPr = QtWidgets.QLabel(self.centralwidget)
        self.label_AirPr.setGeometry(QtCore.QRect(40, 90, 181, 21))
        self.label_AirPr.setStyleSheet("\n"
"font: 16pt \"Times New Roman\";\n"
"")
        self.label_AirPr.setObjectName("label_AirPr")
        self.label_AirTem = QtWidgets.QLabel(self.centralwidget)
        self.label_AirTem.setGeometry(QtCore.QRect(40, 140, 171, 21))
        self.label_AirTem.setStyleSheet("font: 16pt \"Times New Roman\";\n"
"")
        self.label_AirTem.setObjectName("label_AirTem")
        self.lineEdit_AirTem = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_AirTem.setGeometry(QtCore.QRect(290, 90, 161, 31))
        self.lineEdit_AirTem.setStyleSheet("font: 16pt \"Times New Roman\";\n"
"")
        self.lineEdit_AirTem.setObjectName("lineEdit_AirTem")
        self.lineEdit_AirPr = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_AirPr.setGeometry(QtCore.QRect(290, 140, 161, 31))
        self.lineEdit_AirPr.setStyleSheet("font: 16pt \"Times New Roman\";\n"
"")
        self.lineEdit_AirPr.setObjectName("lineEdit_AirPr")
        self.label_UAReg = QtWidgets.QLabel(self.centralwidget)
```

```
        self.label_UAReg.setGeometry(QtCore.QRect(40, 250, 81, 21))
        self.label_UAReg.setStyleSheet("font: 16pt \"Times New Roman\";\n"
"")
        self.label_UAReg.setObjectName("label_UAReg")
        self.comboBox_UAReg = QtWidgets.QComboBox(self.centralwidget)
        self.comboBox_UAReg.setGeometry(QtCore.QRect(270, 250, 181, 22))
        self.comboBox_UAReg.setStyleSheet("font: 14pt \"Times New Roman\";\n"
"")
        self.comboBox_UAReg.setObjectName("comboBox_UAReg")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        # self.comboBox_UAReg.addItem("")
        self.pushButton_Result = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_Result.setGeometry(QtCore.QRect(40, 340, 141, 41))
        self.pushButton_Result.setStyleSheet(".button {\n"
"  display: inline-block;\n"
"  padding: 15px 25px;\n"
"  font-size: 24px;\n"
"  cursor: pointer;\n"
"  text-align: center;\n"
"  text-decoration: none;\n"
"  outline: none;\n"
"  color: #fff;\n"
"  background-color: #4CAF50;\n"
"  border: none;\n"
"  border-radius: 15px;\n"
"  box-shadow: 0 9px #999;\n"
"}\n"
"\n"
".button:hover {background-color: #3e8e41}\n"
"\n"
".button:active {\n"
```

```
"  background-color: #3e8e41;\n"
"  box-shadow: 0 5px #666;\n"
"  transform: translateY(4px);\n"
"}")
        self.pushButton_Result.setObjectName("pushButton_Result")
        self.label_Result = QtWidgets.QLabel(self.centralwidget)
        self.label_Result.setGeometry(QtCore.QRect(40, 410, 101, 21))
        self.label_Result.setStyleSheet("font: 16pt \"Times New Roman\";\n"
"color:black;\n"
"")
        # self.pushButton_Result.clicked.connect(self.window2)
        self.label_Result.setObjectName("label_Result")
        self.label_DisplayRes = QtWidgets.QLabel(self.centralwidget)
        self.label_DisplayRes.setGeometry(QtCore.QRect(290, 400, 161, 31))
        self.label_DisplayRes.setStyleSheet("font: 16pt \"Times New Roman\";\n"
"background-color:white;\n"
"color:Red;\n"
"border: 2px solid #4CAF50;")
        self.label_DisplayRes.setText("")
        self.label_DisplayRes.setObjectName("label_DisplayRes")
        self.label_AirHum = QtWidgets.QLabel(self.centralwidget)
        self.label_AirHum.setGeometry(QtCore.QRect(40, 190, 111, 21))
        self.label_AirHum.setStyleSheet("font: 16pt \"Times New Roman\";\n"
"")
        self.label_AirHum.setObjectName("label_AirHum")
        self.lineEdit_AirHum = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_AirHum.setGeometry(QtCore.QRect(290, 189, 161, 31))
        self.lineEdit_AirHum.setStyleSheet("font: 16pt \"Times New Roman\";\n"
"")
        self.lineEdit_AirHum.setObjectName("lineEdit_AirHum")
        self.pushButton_Back = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_Back.setGeometry(QtCore.QRect(310, 340, 141, 41))
        self.pushButton_Back.setStyleSheet(".button {\n"
"  display: inline-block;\n"
"  padding: 15px 25px;\n"
"  font-size: 24px;\n"
"  cursor: pointer;\n"
"  text-align: center;\n"
"  text-decoration: none;\n"
"  outline: none;\n"
"  color: #fff;\n"
"  background-color: #4CAF50;\n"
"  border: none;\n"
"  border-radius: 15px;\n"
"  box-shadow: 0 9px #999;\n"
"}\n"
"\n"
".button:hover {background-color: #3e8e41}\n"
"\n"
".button:active {\n"
"  background-color: #3e8e41;\n"
```

```
"  box-shadow: 0 5px #666;\n"
"  transform: translateY(4px);\n"
"}")
        self.pushButton_Back.setObjectName("pushButton_Back")
        Forecast_Window.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(Forecast_Window)
        self.statusbar.setObjectName("statusbar")
        Forecast_Window.setStatusBar(self.statusbar)

        self.retranslateUi(Forecast_Window)
        QtCore.QMetaObject.connectSlotsByName(Forecast_Window)

    def retranslateUi(self, Forecast_Window):
        _translate = QtCore.QCoreApplication.translate
        Forecast_Window.setWindowTitle(_translate("Forecast_Window", "Forecast"))
        self.label_AirPr.setText(_translate("Forecast_Window", "Temperature"))
        self.label_AirTem.setText(_translate("Forecast_Window", "Air pressure"))
        self.label_UAReg.setText(_translate("Forecast_Window", "Area"))
        # self.comboBox_UAReg.setItemText(0, _translate("Forecast_Window", "Вінницька"))
        # self.comboBox_UAReg.setItemText(1, _translate("Forecast_Window", "Волинська"))
        # self.comboBox_UAReg.setItemText(2, _translate("Forecast_Window", "Дніпропетровська"))
        # self.comboBox_UAReg.setItemText(3, _translate("Forecast_Window", "Житомирська"))
        # self.comboBox_UAReg.setItemText(4, _translate("Forecast_Window", "Закарпатська"))
        # self.comboBox_UAReg.setItemText(5, _translate("Forecast_Window", "Запорізька"))
        # self.comboBox_UAReg.setItemText(6, _translate("Forecast_Window", "Івано-Франківська"))
        # self.comboBox_UAReg.setItemText(7, _translate("Forecast_Window", "Київська"))
        # self.comboBox_UAReg.setItemText(8, _translate("Forecast_Window", "Кіровоградська"))
        # self.comboBox_UAReg.setItemText(9, _translate("Forecast_Window", "Львівська"))
        # self.comboBox_UAReg.setItemText(10, _translate("Forecast_Window", "Миколаївська"))
        # self.comboBox_UAReg.setItemText(11, _translate("Forecast_Window", "Одеська"))
        # self.comboBox_UAReg.setItemText(12, _translate("Forecast_Window", "Полтавська"))
        # self.comboBox_UAReg.setItemText(13, _translate("Forecast_Window", "Рівненська"))
        # self.comboBox_UAReg.setItemText(14, _translate("Forecast_Window", "Сумська"))
        # self.comboBox_UAReg.setItemText(15, _translate("Forecast_Window", "Тернопільська"))
        # self.comboBox_UAReg.setItemText(16, _translate("Forecast_Window", "Харківська"))
        # self.comboBox_UAReg.setItemText(17, _translate("Forecast_Window", "Херсонська"))
        # self.comboBox_UAReg.setItemText(18, _translate("Forecast_Window", "Хмельницька"))
        # self.comboBox_UAReg.setItemText(19, _translate("Forecast_Window", "Черкаська"))
        # self.comboBox_UAReg.setItemText(20, _translate("Forecast_Window", "Чернівецька"))
        # self.comboBox_UAReg.setItemText(21, _translate("Forecast_Window", "Чернігівська"))

        self.pushButton_Result.setText(_translate("Forecast_Window", "Calculate"))
        self.label_Result.setText(_translate("Forecast_Window", "Calculate:"))
        self.label_AirHum.setText(_translate("Forecast_Window", "Humidity"))
        self.pushButton_Back.setText(_translate("Forecast_Window", "Return"))

    def window2(self):
        self.w = MyWindows2()
        # self.w.show()
        # self.hide()
```

```
class MyWindows2(QtWidgets.QMainWindow):
    def __init__(self):
        super().__init__()
        # App = QtWidgets.QApplication(sys.argv)
        # window = Window()
        # window.show()
        # sys.exit(App.exec())
        self.map = Map_Window()
        self.map.show()
        # sys.exit(App.exec())
```

CreateModel.py

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Creat_Model_Window(object):
    def setupUi(self, Creat_Model_Window):
        Creat_Model_Window.setObjectName("Creat_Model_Window")
        Creat_Model_Window.setWindowModality(QtCore.Qt.NonModal)
        Creat_Model_Window.setEnabled(True)
        Creat_Model_Window.resize(500, 600)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(Creat_Model_Window.sizePolicy().hasHeightForWidth())
        Creat_Model_Window.setSizePolicy(sizePolicy)
        Creat_Model_Window.setMaximumSize(QtCore.QSize(500, 600))
        Creat_Model_Window.setStyleSheet("font: 16pt \"Times New Roman\";\n"
"color: black;")
        self.centralwidget = QtWidgets.QWidget(Creat_Model_Window)
        self.centralwidget.setObjectName("centralwidget")
        self.label_model_name = QtWidgets.QLabel(self.centralwidget)
        self.label_model_name.setGeometry(QtCore.QRect(30, 70, 121, 21))
        self.label_model_name.setObjectName("label_model_name")
        self.label_path_file = QtWidgets.QLabel(self.centralwidget)
        self.label_path_file.setGeometry(QtCore.QRect(30, 120, 151, 21))
        self.label_path_file.setObjectName("label_path_file")
        self.label_name_pollution = QtWidgets.QLabel(self.centralwidget)
        self.label_name_pollution.setGeometry(QtCore.QRect(30, 170, 191, 21))
        self.label_name_pollution.setObjectName("label_name_pollution")
        self.label_longitute = QtWidgets.QLabel(self.centralwidget)
        self.label_longitute.setGeometry(QtCore.QRect(90, 270, 81, 21))
        self.label_longitute.setObjectName("label_longitute")
        self.label_latitude = QtWidgets.QLabel(self.centralwidget)
        self.label_latitude.setEnabled(True)
        self.label_latitude.setGeometry(QtCore.QRect(330, 270, 81, 21))
        self.label_latitude.setObjectName("label_latitude")
        self.lineEdit_model_name = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_model_name.setGeometry(QtCore.QRect(260, 60, 191, 31))
        self.lineEdit_model_name.setObjectName("lineEdit_model_name")
        self.lineEdit_path_file = QtWidgets.QLineEdit(self.centralwidget)
```

```
        self.lineEdit_path_file.setGeometry(QtCore.QRect(260, 110, 191, 31))
        self.lineEdit_path_file.setObjectName("lineEdit_path_file")
        self.lineEdit_name_pollution = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_name_pollution.setGeometry(QtCore.QRect(260, 160, 191, 31))
        self.lineEdit_name_pollution.setObjectName("lineEdit_name_pollution")
        self.lineEdit_longitute = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_longitute.setGeometry(QtCore.QRect(70, 310, 111, 31))
        self.lineEdit_longitute.setObjectName("lineEdit_longitute")
        self.lineEdit_latitude = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_latitude.setGeometry(QtCore.QRect(310, 310, 111, 31))
        self.lineEdit_latitude.setObjectName("lineEdit_latitude")
        self.pushButton_create_model = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_create_model.setGeometry(QtCore.QRect(40, 430, 161, 41))
        self.pushButton_create_model.setStyleSheet(".button {\n"
"  display: inline-block;\n"
"  padding: 15px 25px;\n"
"  font-size: 24px;\n"
"  cursor: pointer;\n"
"  text-align: center;\n"
"  text-decoration: none;\n"
"  outline: none;\n"
"  color: #fff;\n"
"  background-color: #4CAF50;\n"
"  border: none;\n"
"  border-radius: 15px;\n"
"  box-shadow: 0 9px #999;\n"
"}\n"
"\n"
".button:hover {background-color: #3e8e41}\n"
"\n"
".button:active {\n"
"  background-color: #3e8e41;\n"
"  box-shadow: 0 5px #666;\n"
"  transform: translateY(4px);\n"
"}\n"
"")
        self.pushButton_create_model.setObjectName("pushButton_create_model")
        self.checkBox = QtWidgets.QCheckBox(self.centralwidget)
        self.checkBox.setGeometry(QtCore.QRect(70, 370, 351, 17))
        self.checkBox.setStyleSheet("width:19px;\n"
"    height:19px;")
        self.checkBox.setIconSize(QtCore.QSize(16, 16))
        self.checkBox.setObjectName("checkBox")
        self.pushButton_Back = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_Back.setGeometry(QtCore.QRect(310, 430, 141, 41))
        self.pushButton_Back.setStyleSheet(".button {\n"
"  display: inline-block;\n"
"  padding: 15px 25px;\n"
"  font-size: 24px;\n"
"  cursor: pointer;\n"
"  text-align: center;\n"
```

```
"  text-decoration: none;\n"
"  outline: none;\n"
"  color: #fff;\n"
"  background-color: #4CAF50;\n"
"  border: none;\n"
"  border-radius: 15px;\n"
"  box-shadow: 0 9px #999;\n"
"}\n"
"\n"
".button:hover {background-color: #3e8e41}\n"
"\n"
".button:active {\n"
"  background-color: #3e8e41;\n"
"  box-shadow: 0 5px #666;\n"
"  transform: translateY(4px);\n"
"}")
        self.pushButton_Back.setObjectName("pushButton_Back")
        Creat_Model_Window.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(Creat_Model_Window)
        self.statusbar.setObjectName("statusbar")
        Creat_Model_Window.setStatusBar(self.statusbar)

        self.retranslateUi(Creat_Model_Window)
        QtCore.QMetaObject.connectSlotsByName(Creat_Model_Window)

    def retranslateUi(self, Creat_Model_Window):
        _translate = QtCore.QCoreApplication.translate
        Creat_Model_Window.setWindowTitle(_translate("Creat_Model_Window", "Create model"))
        self.label_model_name.setText(_translate("Creat_Model_Window", "Model name:"))
        self.label_path_file.setText(_translate("Creat_Model_Window", "File path:"))
        self.label_name_pollution.setText(_translate("Creat_Model_Window", "Pollutant name:"))
        self.label_longitute.setText(_translate("Creat_Model_Window", "Longtitude"))
        self.label_latitude.setText(_translate("Creat_Model_Window", "Latitude"))
        self.pushButton_create_model.setText(_translate("Creat_Model_Window", "Create"))
        self.checkBox.setText(_translate("Creat_Model_Window", "     Data from SaveEcoBot"))
        self.pushButton_Back.setText(_translate("Creat_Model_Window", "Return"))
```

AddData.py

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_AddData_Window(object):
    def setupUi(self, AddData_Window):
        AddData_Window.setObjectName("AddData_Window")
        AddData_Window.resize(500, 600)
        AddData_Window.setMaximumSize(QtCore.QSize(500, 600))
        AddData_Window.setStyleSheet("font: 16pt \"Times New Roman\";\n"
"color: black;")
        self.centralwidget = QtWidgets.QWidget(AddData_Window)
        self.centralwidget.setObjectName("centralwidget")
        self.label_select_model = QtWidgets.QLabel(self.centralwidget)
        self.label_select_model.setGeometry(QtCore.QRect(40, 100, 121, 21))
```

```
        self.label_select_model.setObjectName("label_select_model")
        self.label_path_file = QtWidgets.QLabel(self.centralwidget)
        self.label_path_file.setGeometry(QtCore.QRect(40, 150, 151, 21))
        self.label_path_file.setObjectName("label_path_file")
        self.comboBox_select_model = QtWidgets.QComboBox(self.centralwidget)
        self.comboBox_select_model.setGeometry(QtCore.QRect(260, 90, 191, 31))
        self.comboBox_select_model.setObjectName("comboBox_select_model")
        self.lineEdit_path_file = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_path_file.setGeometry(QtCore.QRect(260, 140, 191, 31))
        self.lineEdit_path_file.setObjectName("lineEdit_path_file")
        self.label_name_pollution = QtWidgets.QLabel(self.centralwidget)
        self.label_name_pollution.setGeometry(QtCore.QRect(40, 200, 191, 21))
        self.label_name_pollution.setObjectName("label_name_pollution")
        self.lineEdit_name_pollution = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_name_pollution.setGeometry(QtCore.QRect(260, 190, 191, 31))
        self.lineEdit_name_pollution.setObjectName("lineEdit_name_pollution")
        self.pushButton_create_model = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_create_model.setGeometry(QtCore.QRect(40, 350, 151, 41))
        self.pushButton_create_model.setStyleSheet(".button {\n"
"  display: inline-block;\n"
"  padding: 15px 25px;\n"
"  font-size: 24px;\n"
"  cursor: pointer;\n"
"  text-align: center;\n"
"  text-decoration: none;\n"
"  outline: none;\n"
"  color: #fff;\n"
"  background-color: #4CAF50;\n"
"  border: none;\n"
"  border-radius: 15px;\n"
"  box-shadow: 0 9px #999;\n"
"}\n"
"\n"
".button:hover {background-color: #3e8e41}\n"
"\n"
".button:active {\n"
"  background-color: #3e8e41;\n"
"  box-shadow: 0 5px #666;\n"
"  transform: translateY(4px);\n"
"}\n"
"")
        self.pushButton_create_model.setObjectName("pushButton_create_model")
        self.pushButton_Back = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_Back.setGeometry(QtCore.QRect(310, 350, 141, 41))
        self.pushButton_Back.setStyleSheet(".button {\n"
"  display: inline-block;\n"
"  padding: 15px 25px;\n"
"  font-size: 24px;\n"
"  cursor: pointer;\n"
"  text-align: center;\n"
"  text-decoration: none;\n"
```

```
"  outline: none;\n"
"  color: #fff;\n"
"  background-color: #4CAF50;\n"
"  border: none;\n"
"  border-radius: 15px;\n"
"  box-shadow: 0 9px #999;\n"
"}\n"
"\n"
".button:hover {background-color: #3e8e41}\n"
"\n"
".button:active {\n"
"  background-color: #3e8e41;\n"
"  box-shadow: 0 5px #666;\n"
"  transform: translateY(4px);\n"
"}")
        self.pushButton_Back.setObjectName("pushButton_Back")
        AddData_Window.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(AddData_Window)
        self.statusbar.setObjectName("statusbar")
        AddData_Window.setStatusBar(self.statusbar)

        self.retranslateUi(AddData_Window)
        QtCore.QMetaObject.connectSlotsByName(AddData_Window)

    def retranslateUi(self, AddData_Window):
        _translate = QtCore.QCoreApplication.translate
        AddData_Window.setWindowTitle(_translate("AddData_Window", "Add data"))
        self.label_select_model.setText(_translate("AddData_Window", "Choosing model:"))
        self.label_path_file.setText(_translate("AddData_Window", "File path:"))
        self.label_name_pollution.setText(_translate("AddData_Window", "Pollutant name:"))
        self.pushButton_create_model.setText(_translate("AddData_Window", "Add"))
        self.pushButton_Back.setText(_translate("AddData_Window", "Return"))
```