НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
“___”_____2025 р.

Дипломна робота
на здобуття ступеня бакалавр

За освітньою програмою “Цифрові технології в енергетиці”

Спеціальності 122 “Комп’ютерні науки”

на тему: “Моделювання впливу рушійних факторів на систему”

Виконала: студентка 4 курсу, групи ТР-15
Тимкова Анастасія Вікторівна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. викладач Ольга БЕСПАЛА___ _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент зав.кафедри ТАЕ, д.т.н., проф. Ольга ЧЕРНОУСЕНКО _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Н.контроль асистент Антон ПАСІЧНЮК _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

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

Київ – 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА
(підпис)
“___” _____2025 р.

ЗАВДАННЯ
на дипломну роботу студенту

Тимковій Анастасії Вікторівні
(прізвище, ім’я, по батькові)

1. Тема роботи “Моделювання впливу рушійних факторів на систему”
Науковий керівник Беспала Ольга Миколаївна, ст. викладач
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “ 02 ” червня 2025 року № 1875-с

2. Термін подання студентом роботи 09.06.2025

3. Вихідні дані до роботи мова програмування Python, фреймворк Flask, СУБД SQLite, шаблонізатор Jinja, середовище розробки Visual Studio Code

4. Перелік питань, які потрібно розробити _____

1) провести аналіз існуючих методів та програмних рішень для моделювання впливу рушійних факторів на систему та прогнозування часових рядів
2) аргументувати вибрані інструменти, алгоритми та технології для реалізації вебсервісу
3) побудувати архітектуру програмного забезпечення та бази даних

4) створити вебсервіс для моделювання впливу рушійних факторів та прогнозування часових рядів

5) представити процес застосування вебсервісу

5. Орієнтований перелік ілюстративного матеріалу схеми, що показують роботу використаних алгоритмів машинного навчання, діаграма архітектури проєкту, діаграми послідовностей функціональних модулів, діаграма прецедентів, скріншоти вхідних даних та роботи користувачів із вебсервісом.

6. Дата видачі завдання 13.09.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вибір теми роботи	02.09.2024-12.09.2024	Виконано
2.	Аналіз методів та засобів розв'язання задачі	14.10.2024-29.11.2024	Виконано
3.	Розробка архітектури та загальної структури системи	16.12.2024-17.01.2025	Виконано
4.	Розробка окремих підсистем	20.01.2025-14.03.2025	Виконано
5.	Програмна реалізація системи	24.03.2025-02.05.2025	Виконано
6.	Оформлення пояснювальної записки	05.05.2025-13.05.2025	Виконано
7.	Захист програмного забезпечення	14.05.2025	Виконано
8.	Передзахист	27.05.2025	Виконано
9.	Захист	16.06.2025-20.06.2025	Виконано

Студент _____ Анастасія ТИМКОВА
(підпис) (прізвище та ініціали)

Керівник _____ Ольга БЕСПАЛА
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота виконана на 52 сторінки, містить 18 рисунків, 1 таблицю, 4 додатки, 24 джерел в переліку посилань.

Мета роботи – створення вебсервісу для моделювання впливу рушійних факторів на систему та прогнозування цільового показника.

Методи та засоби: тест Грейнджера для визначення зв’язків між часовими змінними, алгоритм оцінювання важливості факторів Random Forest, модель нейронної мережі для прогнозування LSTM, мова програмування Python, бібліотеки для аналізу даних pandas і statsmodels, бібліотека машинного навчання scikit-learn, фреймворк глибокого навчання TensorFlow, фреймворк для реалізації серверної частини Flask, бібліотеки візуалізації matplotlib та networkx.

Результати – вебсервіс, який дозволяє виявляти причинно-наслідкові зв’язки між змінними, оцінювати важливість факторів та формувати прогноз цільового показника на основі часових рядів.

Ключові слова: ЧАСОВІ РЯДИ, ПРИЧИННО-НАСЛІДКОВІ ЗВ’ЯЗКИ, ПРОГНОЗУВАННЯ, МАШИННЕ НАВЧАННЯ.

ABSTRACT

The thesis consists of 52 pages, 18 figures, 1 table, 4 appendices, and 24 sources in the list of references.

The purpose of the work is to create a web service for modeling the impact of driving factors on the system and forecasting the target indicator.

Methods and tools: Granger's test for determining the relationships between time variables, Random Forest algorithm for assessing the importance of factors, LSTM neural network model for forecasting, Python programming language, pandas and statsmodels libraries for data analysis, scikit-learn machine learning library, TensorFlow deep learning framework, Flask framework for implementing the server side, matplotlib and networkx visualization libraries.

The results are a web service that allows identifying cause-and-effect relationships between variables, assessing the importance of factors, and generating a forecast of a target indicator based on time series.

Keywords: TIME SERIES, CAUSE-AND-EFFECT RELATIONSHIPS, FORECASTING, MACHINE LEARNING.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ETC (Effort-To-Compress) – міра складності послідовності даних, що оцінює наскільки вона непередбачувана або інформаційно насичена.

MDI (Mean Decrease in Impurity) – метрика зменшення невизначеності дерева.

ARIMA (AutoRegressive Integrated Moving Average) – авторегресійна інтегрована модель ковзного середнього.

SARIMA (Seasonal AutoRegressive Integrated Moving Average) – сезонна авторегресійна інтегрована модель ковзного середнього.

VAR (Vector AutoRegression) – модель векторної авторегресії для аналізу та прогнозування багатозмінних часових рядів.

LSTM (Long Short-Term Memory) – тип рекурентної нейронної мережі, здатний запам’ятовувати довгострокові залежності в послідовних даних завдяки спеціальній структурі комірок пам’яті.

PC (Peter-Clark algorithm) – алгоритм виявлення причинних зв’язків між змінними на основі умовної незалежності в статистичних даних.

FCI (Fast Causal Inference) – метод побудови причинно-наслідкових графів, здатний враховувати приховані змінні та спільні причини.

GES (Greedy Equivalence Search) – алгоритм пошуку структури причинно-наслідкової мережі шляхом жадібного додавання та вилучення зв’язків, що покращують модель.

ADF (Augmented Dickey-Fuller) – статистичний тест для перевірки стаціонарності часового ряду.

AIC (Akaike Information Criterion) – критерій якості моделі.

BIC (Bayesian Information Criterion) – критерій складності моделі.

HTML (HyperText Markup Language) – стандартна мова розмітки для створення та структурування вебсторінок.

CSS (Cascading Style Sheets) – мова стилів, що визначає зовнішній вигляд і оформлення елементів HTML-документа.

СУБД (Система управління базами даних) – програмне забезпечення для створення, зберігання, модифікації та управління базами даних.

HTTP (HyperText Transfer Protocol) – протокол прикладного рівня, що використовується для обміну інформацією між клієнтом і сервером у вебмережі.

DAG (Directed Acyclic Graph) – структура у вигляді орієнтованого ациклічного графа без циклів.

MSE (Mean Squared Error) – метрика якості прогнозу, яка обчислює середнє квадратичне відхилення між передбаченими та фактичними значеннями.

API (Application Programming Interface) – набір функцій і протоколів, що дозволяють одній програмі взаємодіяти з іншою або з окремими компонентами системи.

ВСТУП

Упродовж останніх десятиліть світова спільнота стикається із дедалі складнішими викликами, що потребують глибокого аналізу даних для прийняття ефективних рішень. Стрімкий розвиток інформаційних технологій, інтернету речей та автоматизації процесів призвів до швидкого зростання обсягів даних, які щосекунди генеруються у різних сферах діяльності. Станом на 2024 рік обсяг створених, зібраних та оброблених даних сягнув 149 зетабайт і ця кількість стрімко зростає [1]. Значна частина цих даних має тимчасову структуру або накопичується у процесі довготривалих спостережень. Усе це формує потребу в інструментах, які будуть здатні не лише зберігати великі обсяги даних, а й ефективно їх обробляти, виявляючи приховані залежності, закономірності та тенденції, які складно виявити класичними підходами у складних системах.

Значної актуальності набуває можливість виявляти причинні взаємозв’язки між факторами, щоб глибше зрозуміти механізми функціонування складних систем. На сьогодні, у соціальній, економічній та екологічній сферах, які відзначаються високим рівнем динамічності, недостатньо лише фіксувати зміни. Необхідно також розуміти, які фактори спричиняють їх поведінку. Швидке та ефективне реагування на внутрішні та зовнішні впливи можливе лише за наявності чіткого уявлення про те, які фактори відіграють важливу роль у функціонуванні системи, а які є незначними або взагалі випадковими.

Метою роботи є розробка вебсервісу, який дозволяє моделювати вплив рушійних факторів на систему та здійснювати прогноз цільового показника на основі часових даних.

Для досягнення поставленої мети потрібно виконати такі завдання:

– ознайомитися із існуючими методами моделювання впливу рушійних факторів на систему та моделями для прогнозування часових рядів;

– обрати найбільш доцільні методи та моделі відповідно до специфіки вхідних даних та цілей аналізу;

– розробити моделі у вигляді програмних компонентів із врахуванням вимог до інтеграції в єдину аналітичну систему;

– зібрати, попередньо обробити та підготувати статистичні дані;

– провести тестування розроблених моделей, здійснити оцінку їх точності, надійності та стабільності результатів з використанням відповідних метрик.

У ході виконання дипломної роботи було прийнято участь у міжнародній науково-практичній конференції молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики» з публікацією тез на тему «Моделювання впливу рушійного фактора на екологічні та економічні показники з використанням машинного навчання» [2] (Додатки Б і В). Також було написано статтю на тему «Оптимізація прогнозування шляхом урахування причинного впливу» до журналу Наукові записки Державного університету інформаційно-комунікаційних технологій (Додаток А).

Дипломна робота містить вступ, чотири розділи, висновки, список використаних джерел та додатки. Загальний обсяг становить 52 сторінки. У роботі наведено 18 рисунків, 1 таблицю, 4 додатки та 24 використаних джерел.

Перший розділ містить постановку задачі моделювання впливу рушійних факторів, призначення програмного продукту, огляд методів аналізу причинно-наслідкових зв’язків і засобів прогнозування та існуючих програмних рішень у цій сфері.

У другому розділі розглянуто основні підходи, які використовуються в програмі для виявлення причинних зв’язків, оцінки важливості факторів та побудови прогнозу. Подано процес підготовки вхідних і вихідних даних, необхідних для коректної роботи системи.

Третій розділ містить опис реалізації програмного продукту. Розглянуто використані засоби розробки, структуру архітектури, компоненти клієнтської та серверної частин, базу даних та функціональні модулі. Також оцінено точність реалізованих моделей для підтвердження їх надійності.

Четвертий розділ описує вимоги до використання програми та демонструє його функціональні можливості при взаємодії користувача з інтерфейсом системи.

1 ЗАДАЧА МОДЕЛЮВАННЯ ВПЛИВУ РУШІЙНИХ ФАКТОРІВ НА СИСТЕМУ

Розробка моделі для виявлення впливу рушійних факторів систему та прогнозування цільового показника потребує попереднього аналізу теоретичних та практичних рішень у цій галузі. У цьому розділі визначено задачу моделювання, описано призначення програми та її цінність для користувачів. Розглядаються основні методи аналізу причинних зв’язків у часових рядах та методи прогнозування, які застосовуються залежно від складності даних. Проаналізовано існуючі програмні рішення, схожі за функціональністю, щоб показати переваги розробленої системи порівняно з іншими існуючими рішеннями. Така структура дозволяє сформувати цілісне уявлення про поставлену задачу, визначити засоби її реалізації та місце розробленої системи серед існуючих аналогів.

1.1 Постановка задачі моделювання впливу рушійних факторів

У процесі аналізу багатофакторних систем, де змінні мають часову структуру, виникає потреба не лише у фіксації змін, а й у глибшому розумінні механізмів впливу одних показників на інші. Для цього необхідно виявити змінні, що мають причинний вплив на цільовий показник, оцінити їх важливість для системи та спрогнозувати її подальшу поведінку. Ефективне моделювання можливе лише за умови поєднання класичних статистичних підходів із сучасними методами машинного навчання.

Усе це зумовлює необхідність розробки аналітичного програмного засобу, призначеного для аналізу багатофакторних часових даних, що відображають динаміку змін параметрів у системі, які представлені у вигляді файлів формату .xlsx. Припускається, що спостереження мають часову залежність, тобто кожна змінна змінюється з часом, що створює можливості для виявлення рушійних

факторів системи із часовим зсувом. Це дає підстави для побудови моделі, яка здатна визначати змінні з причинним впливом на цільовий показник, оцінювати їх ступінь впливу та формувати прогноз поведінки цільового фактора в майбутньому.

Для досягнення поставленої мети потрібно проаналізувати існуючі методи виявлення рушійних факторів у системі та моделей прогнозування. Це дозволяє визначити найбільш доцільні підходи для роботи з багатовимірними даними, що змінюються в часі. Обрані методи мають відповідати структурі вхідної інформації та завданням, які ставляться перед системою аналізу.

На основі цього проводиться розробка програмних компонентів, які зможуть працювати у єдиній аналітичній системі. Для забезпечення коректної роботи програмного продукту важливо приділити увагу збору та обробці статистичних даних для їх подальшого аналізу.

Важливо провести тестування створеної системи, щоб переконатися в правильності її роботи та стабільності отриманих результатів. Це дає змогу оцінити, наскільки точними та надійними є побудовані моделі, а також визначити їхню придатність для практичного використання в аналізі часових даних.

Таким чином, поставлена задача полягає у побудові системи, яка здатна аналізувати часові ряди, виявляти причинні зв’язки та формувати прогнози, що в подальшому може бути використано для прийняття рішень у соціально-економічній, екологічній та бізнес-сферах.

1.2 Призначення розробленої системи

Системи, у яких змінні взаємодіють між собою в часі, потребують спеціальних інструментів для аналізу динаміки та прогнозування подальшого розвитку. У таких умовах важливо мати засіб, що дозволяє виявити закономірності між показниками та оцінити характер їх впливу на цільовий результат. Розроблена модель може допомогти будувати ефективні стратегії та приймати обґрунтовані рішення в різних сферах.

Основне призначення розробленої моделі – надати користувачам необхідні програмні засоби для аналізу факторів з метою визначення рушійних впливів та на їх основі пояснення поведінки системи. За часовими статистичними даними модель дає змогу побачити зміну цільової змінної залежно від динаміки інших показників, у який спосіб ті чи інші фактори взаємодіють між собою та як ці взаємодії впливають на загальну поведінку системи. Програма вирішує задачу аналізу багатофакторних залежностей часових даних, моделює зміни параметрів системи залежно від внутрішніх та зовнішніх умов, оцінює ступінь впливу причинних змінних на цільовий показник та формує прогноз поведінки системи у майбутньому.

Потенційними користувачами розробленої моделі можуть бути фахівці різних сфер діяльності, яким потрібні аналітичні підходи для прийняття рішень. В економіці та фінансах система може стати засобом вивчення взаємозв’язків ринкових показників, аналізі факторів, що впливають на доходи, витрати, інфляцію чи інвестиційну активність. У психології та соціології – для виявлення динаміки суспільних процесів і поведінкових патернів. У державному управлінні – для оцінки ефективності реформ, планування бюджету та виявлення зон ризику. Екологічні організації та науково-дослідні установи можуть використовувати систему для відстеження впливу природних чинників на стан довкілля. У бізнесі система допоможе прогнозувати продажі, оцінювати поведінку споживачів та здійснювати оптимізацію маркетингових стратегій.

Розробка подібного програмного продукту відкриває нові можливості для ефективного управління системами різного призначення – від бізнес-процесів до економічних чи екологічних структур. Його універсальність дозволяє адаптуватися до різних сфер, де є потреба в аналізі часових даних, виявленні рушійних факторів впливу та прогнозуванні майбутньої динаміки. Особливої цінності подібні моделі набувають в умовах нестабільності, де ціна помилки може бути високою. У таких умовах аналітичні інструменти стають важливою складовою стратегічного планування.

1.3 Методи виявлення впливу рушійних факторів на систему

У світі зі складними взаємозв’язками між подіями, процесами і явищами важливо не лише встановлювати факт статистичних залежностей, а й розуміти, яка змінна впливає на іншу, у якому напрямі та з якою часовою затримкою. Причинно-наслідкові зв’язки – це взаємозалежності, при яких значення однієї змінної прямо або опосередковано спричиняють наслідок іншої у часовому просторі. Їхнє правильне визначення має ключове значення у наукових дослідження, політичній діяльності, економіці та прогнозуванні, адже саме причинність, а не звичайна статистична залежність, дозволяє передбачити результати впливу [3]. Помилкове трактування кореляційних вимірювань як причинності призводить до хибних висновків, особливо через приховані змінні, непрямі ефекти чи складні нелінійні залежності [4]. Щоб цього уникнути, існує низка спеціалізованих методів аналізу даних, які здатні виявляти причинно-наслідкові зв’язки в часових рядах.

Статистичні методи виявлення причинних взаємозв’язків ґрунтуються на перевірці, чи покращується прогноз однієї змінної за умови врахування інформації про іншу. Основна ідея більшості методів заснована на принципі Норберта Віннера, сформульованому у середині XX століття: якщо часовий ряд X спричиняє часовий ряд Y, то минулі значення змінної X містять інформацію, яка допомагає передбачити поведінку змінної Y порівняно із тим, що міститься в минулих значеннях самого Y [5]. Цей принцип заклав теоретичну основу для ряду математичних підходів до аналізу причинності.

Класичним і одним із найпоширеніших методів такого аналізу є тест Грейнджера, який був запропонований англійським економетристом Клайвом Грейнджером у 1969 році. Метод ґрунтується на авторегресійному моделюванні та перевіряє, чи статистично значущим є покращення прогнозу змінної Y після додавання в модель інформації про лаги змінної X. Якщо таке покращення наявне – вважається, що «X спричиняє Y за Грейнджером» [6]. Цей підхід широко використовується в економіці, фінансах, а також у кліматичних дослідженнях. Метод простий у реалізації та для розуміння, однак має певні обмеження, які

впливають на коректність результатів: стаціонарність даних, чутливість до вибору часових затримок, а також припущення про лінійність зв’язків, що не завжди відповідає реальній природі складних систем [4].

У випадках, коли залежності між змінними є нелінійними, часто застосовують ентропію перенесення – метод, який розроблений на основі теорії інформації. Суть цього методу полягає у вимірюванні кількості інформації, що передається від однієї змінної до іншої з урахуванням часової структури. Якщо минулі значення змінної X дозволяють суттєво зменшити невизначеність щодо майбутніх значень Y, це свідчить про причинний вплив. Метод не вимагає лінійності, стаціонарності чи нормального розподілу даних, що робить його придатним для аналізу складних систем, зокрема, у кліматології, де процеси мають численні непрямі ефекти, та в нейронауках, де залежності між сигналами складні, швидкоплинні та багатошарові. Водночас, метод має певні обмеження: висока обчислювальна складність, чутливість до налаштування параметрів та потреба у значному обсягу даних [7].

Ще одним підходом, що привертає дедалі більшої уваги дослідників, є причинність складності стиснення. На відміну від підходів, що ґрунтуються на регресії чи ентропії, цей метод орієнтується на вимірювання інформаційної складності. Відповідно до цієї методології, якщо змінна X спричиняє змінну Y, то використання інформації про X дозволяє стискати Y більш ефективно. Це оцінюється за допомогою метрики ETC, яка відображає, наскільки зменшується складність часового ряду при врахуванні додаткової змінної. Метод не має строгих вимог до структури даних і може працювати навіть у межах одного й того самого тимчасового блоку, а не послідовно, як це відбувається у класичних методах. Однак він є чутливим до вибору алгоритму стиснення, менш масштабований та складний для розуміння результатів, особливо для користувачів без технічної підготовки [8].

У даній роботі для реалізації моделі визначення причинно-наслідкових зв’язків було обрано тест Грейнджера завдяки простоті реалізації, зрозумілій інтерпретації результатів, а також широкому практичному застосуванню у сфері аналізу часових рядів. Для визначення ступеня впливу виявлених причинних

змінних модель доповнено методом машинного навчання Random Forest із використанням метрики MDI. Поєднання цих підходів забезпечує не лише виявлення напрямку причинного впливу, а й кількісну оцінку значущості змінних, що робить такий підхід придатним для аналізу складних багатофакторних систем.

1.4 Методи прогнозування часових рядів

Прогнозування часових рядів є ключовим інструментом для задач, що потребують передбачення майбутніх значень показників на основі їхньої попередньої динаміки. Це дає змогу завчасно реагувати на зміни у системі, оцінювати ризики та приймати обґрунтовані рішення. Сучасні методи охоплюють велику кількість моделей, що відрізняються за підходами та типом досліджуваних процесів.

Однією з найпоширеніших класичних моделей є ARIMA, яка прогнозує значення змінної на основі лінійної комбінації власних попередніх значень (авторегресія), зглажених похибок попередніх прогнозів (ковзне середнє) та операції диференціювання для забезпечення стаціонарності ряду. Для обробки сезонних коливань застосовується її модифікований варіант SARIMA, який враховує періодичність у даних. Ці методи є добре вивченими та зручні для інтерпретації результатів, однак мають обмеження щодо адаптації до складної динаміки, особливо у нестабільних чи нелінійних процесах [9].

Для моделювання кількох пов’язаних часових рядів використовується модель VAR. Вона заснована на припущенні, що кожна змінна в системі залежить не тільки від власних минулих значень, а від часових затримок інших змінних. VAR враховує взаємозв’язки між показниками, що є важливим для задач, де змінні мають значний вплив одна на одну. Недоліком даної моделі є те, що вона описує лише короткострокову динаміку змін, не враховуючи довгострокові співвідношення. Крім того, модель чутлива до вибору кількості часових затримок

та потребує значного обсягу даних, що підвищує ризик перенавчання та втрати точності прогнозу [10].

На відмінну від класичних методів, сучасні нейронні моделі, зокрема LSTM, здатні працювати з нестаціонарними та нелінійними часовими рядами зі складною структурою. LSTM – рекурентна нейронна мережа, що містить спеціальні комірки пам’яті, які зберігають інформацію про довготривалі залежності в послідовностях [11]. Завдяки цьому модель не просто реагує на останні зміни, а враховує значущі коливання, що траплялися раніше. Вона не потребує попереднього приведення часового ряду до стаціонарного вигляду і здатна навчатися без припущень про лінійність. Недоліками даної моделі є необхідність великої кількості даних для якісного навчання, чутливість до налаштувань гіперпараметрів та вимоги до обчислювальних ресурсів.

З огляду на потребу адаптувати модель до складної динаміки цільового фактора без обмежень лінійності, у даній роботі для прогнозування часових рядів було обрано модель нейронної мережі LSTM. Вона ефективно враховує короткострокові та довгострокові залежності, забезпечуючи гнучкий і точний прогноз навіть у випадках, коли класичні методи є не ефективними. Такий вибір обумовлено здатністю LSTM адаптуватися до нестабільних, нелінійних закономірностей у реальних даних.

1.5 Аналіз існуючих програмних рішень

У сфері аналізу причинно-наслідкових зв’язків і прогнозування часових рядів існує низка програмних продуктів, які вже реалізують подібну функціональність. Серед них можна виділити як готові інтерактивні сервіси, так і численні бібліотеки та фреймворки мов програмування, зокрема для Python та R. Хоча розроблені бібліотеки, наприклад, DoWhy, CasualImpact, CausalLib, надають широку функціональність, вони потребують глибокого розуміння математичних моделей і навичок програмування, що обмежує доступність для широкого кола користувачів

[12]. Особливу уваги привертають готові програмні рішення, які зручні навіть для користувачів без технічної підготовки.

Одним із найвідоміших інструментів є платформа Tetrad, розроблена в Університеті Карнегі-Меллона, яка забезпечує побудову, візуалізацію та аналіз причинно-наслідкових моделей. Ця система пропонує широкий вибір алгоритмів, зокрема PC, FCI та GES, які дозволяють автоматично будувати причинні графи на основі табличних даних. Даний програмний продукт орієнтований на дослідників та передбачає гнучке налаштування параметрів, що забезпечує високу точність та глибину аналізу [13]. Водночас, інтерфейс Tetrad є технічним і потребує розуміння статистичних моделей, що обмежує його використання для непідготовлених користувачів. Крім того, система вимагає попередньої структуризації даних та часто використовується для експериментального моделювання.

Ще однією платформою є Actable AI – сучасна хмарна платформа, для аналізу причинно-наслідкових зв’язків та прогнозування часових рядів, орієнтована на користувачів без досвіду програмування. Використовує методи машинного навчання, аналіз чутливості та сценарного прогнозування. Перевагою цього сервісу є простота використання та швидке отримання результатів. Проте, оскільки користувач не має повного контролю над алгоритмами та параметрами моделі, використання цієї платформи в наукових дослідженнях є обмеженим [14].

На відмінну від розглянутих сервісів, розроблена система поєднує простоту використання з аналітичною гнучкістю. Користувач може легко завантажити власні Excel-файли, задати параметри аналізу й отримати результати у вигляді графіків, таблиць та текстових висновків. Це забезпечує зручну роботу без спеціальних знань у програмуванні та дозволяє адаптувати систему до різних задач.

2 МЕТОДИ ВИЯВЛЕННЯ РУШІЙНИХ ФАКТОРІВ ТА ЇХ ПРОГНОЗУВАННЯ

Теоретична основа обраних методів та їх програмна реалізація є важливим етапом у побудові системи аналізу. На цьому етапі відбувається практична інтеграція алгоритмів, які дозволяють виявляти причинно-наслідкові зв’язки між змінними та здійснювати прогноз на основі даних із часовими залежностями. У цьому розділі буде розглянуто реалізацію тесту Грейнджера для виявлення причинних залежностей між змінними, застосування методу Random Forest із метрикою MDI для оцінки сили впливу факторів та модель LSTM, яка використовується для прогнозування майбутніх значень цільового показника. Також подано опис структури вхідних та вихідних даних, які забезпечують коректну роботу алгоритмів та інтерпретацію результатів користувачу.

2.1 Виявлення впливу рушійних факторів на систему

Тест Грейнджера є одним із найпоширеніших статистичних методів для визначення причинно-наслідкових зв’язків між часовими рядами. Його ідея полягає в тому, що якщо минулі значення змінної X допомагають покращити прогноз майбутніх значень змінної Y, порівняно з моделлю, яка враховує лише власну історію Y, то вважається, що «X спричиняє Y у сенсі Грейнджера».

Для перевірки цієї гіпотези будуються дві регресійні моделі: спрощена та повна. У спрощеній моделі значення Y_t пояснюються лише попередніми значеннями самої змінної Y. Вона має такий вигляд:

$$Y_t = \alpha_0 + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + ... + \alpha_p Y_{t-p} + \varepsilon_t,$$

де α_0 – константа,

$\alpha_1, \alpha_2, ..., \alpha_p$ – ваги впливу попередніх значень змінної Y на її поточне значення,

$Y_{t-1}, ..., Y_{t-p}$ – попередні значення змінної Y,

ε_t – випадкова похибка,

p – кількість часових затримок.

У повній моделі, окрім лагів змінної Y, включаються також минулі значення змінної X:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + ... + \beta_p Y_{t-p} + \gamma_1 X_{t-1} + ... + \gamma_p X_{t-p} + u_t,$$

де β_0 – константа,

β_i – ваги впливу попередніх значень змінної Y на її поточне значення,

$Y_{t-1},...,Y_{t-p}$ – попередні значення змінної Y,

γ_i – ваги впливу попередніх значень змінної X,

$X_{t-1},..., X_{t-p}$ – попередні значення змінної X,

u_t – випадкова похибка.

У разі, якщо хоча б один із коефіцієнтів γ_i є статистично значущим, нульова гіпотеза $H_0 : \gamma_1 = \gamma_2 = ... = \gamma_p = 0$ відхиляється, що свідчить про наявність причинно-наслідкового впливу X на Y [6].

Для розробки моделі виявлення рушійних факторів у системі із застосуванням тесту Грейнджера було використано функцію grangercausalitytests з бібліотеки statsmodels мови програмування Python. Вона дозволяє автоматично побудувати моделі, оцінити і порівняти їх за допомогою F-тесту, який визначає, чи внесення змінної X покращує прогноз змінної Y. Якщо так, вважається, що між змінними існує причинно-наслідковий зв’язок.

Проте для коректної роботи цього методу необхідно попередньо підготувати дані. Насамперед, вони мають бути відсортовані за часом, оскільки часові залежності мають сенс лише тоді, коли значення розташовані у правильній часовій послідовності. Крім того, значення змінних потрібно привести до єдиного масштабу, щоб уникнути ситуації, коли змінні з більшими числовими значеннями домінують у моделі. Для цього було використано метод MinMaxScaler з пакету sklearn.preprocessing, який масштабує значення кожної змінної до інтервалу від 0 до 1.

Ще однією важливою умовою є стаціонарність рядів. Стаціонарним вважається ряд, у якого середнє значення, дисперсія та автокореляція залишаються сталими впродовж часу. Тому для коректної роботи алгоритму для кожної змінної було проведено перевірку на стаціонарність за допомогою ADF-тесту (тест Діккі-Фуллера) [15]. Нульова гіпотеза цього тесту передбачає наявність одиничного кореня, тобто нестаціонарність ряду. Основним показником є p-значення: якщо воно менше за 0,05, нульова гіпотеза спростовується.

Дану перевірку реалізовано з використанням функції adfuller бібліотеки statsmodels.tsa.stattools. Якщо часові ряди є нестаціонарними, то застосовується метод диференціювання, який дозволяє усунути тренд та сезонність. Він обчислюється за формулою:

$$\Delta Y_t = Y_t = Y_{t-1},$$

де Y_t – значення змінної Y в момент часу *t*,

Y_{t-1} – значення цієї ж змінної в попередній момент часу $t-1$.

Диференціювання обрано як базовий підхід через його простоту, ефективність та підтримку у вбудованих бібліотеках Python.

Наступним і дуже важливим етапом є визначення оптимальної кількість часових затримок. Занадто мала кількість може призвести до втрати інформації, а надмірна – до перенавчання. Для цього застосовуються інформаційні критерії. Найчастіше використовуються AIC та BIC [16]. Вони мають такий вигляд:

$$AIC = 2k - 2\ln(L),$$

$$BIC = \ln(n)k - 2\ln(L),$$

де *k* – кількість параметрів у моделі,

L – максимальне значення функції правдоподібності,

n – кількість спостережень у вибірці.

AIC більше орієнтований на зменшення похибки прогнозу та менш суворо карає складні моделі, що дає змогу зберігати більшу кількість змінних у моделі. Натомість BIC надає перевагу простішим моделям, штрафуючи за зростання кількості параметрів. У цій роботі було обрано критерій AIC, оскільки він краще підходить для виявлення інформативних взаємозв’язків між змінними з

урахуванням динаміки системи, особливо у випадках, коли можливі слабкі або приховані впливи. Такий вибір дає змогу не втратити важливої інформації в процесі моделювання та забезпечує кращу адаптацію моделі до складної структури часових даних.

Кількість лагів обирається автоматично за допомогою функції select_order класу VAR бібліотеки statsmodels.tsa.api, яка оцінює модель для різних значень часових затримок і повертає оптимальну кількість за мінімальним значенням AIC.

Після цього проводиться тест Грейнджера: для кожної пари змінних будується повна та спрощена модель, обчислюється значення F-статистики і на основі p-значень робиться висновок про причинність.

Також, слід зазначити низку важливих обмежень [17], які необхідно враховувати для коректності результатів:

– лінійність: тест Грейнджера орієнтований, насамперед, на виявлення лінійних залежностей, які він визначає з високою точністю, оскільки ґрунтується на регресійній моделі. Водночас, за певних умов він може частково виявляти і нелінійні зв’язки, однак робить це з меншою надійністю. У системах зі складною або багаторівневою структурою взаємодій результати тесту слід розглядати з обережністю, оскільки не всі впливи можуть бути коректно зафіксовані**;**

– рівномірна частота спостережень: дані мають бути зібрані через однакові проміжки часу: щодня, щотижня, щомісяця тощо. Якщо інтервали між спостереженнями різні або мають пропуски, модель некоректно оцінює залежності часових затримок. Це може спричинити виявлення неіснуючих зв’язків або ігнорувати реальні**;**

– обсяг вибірки: для проведення тесту потрібна достатня кількість спостережень, особливо при роботі із багатофакторною системою. Чим більше часових затримок враховується – тим більше параметрів треба оцінити. Мала кількість значень може призвести до перенавчання, втрати точності та зниження довіри до висновків.

Завдяки поетапному підходу**,** розроблено модель, що дозволяє на практиці використовувати тест Грейнджера для аналізу причинно-наслідкових зв’язків у

часових рядах. Ретельна підготовка даних, включаючи нормалізацію, приведення до стаціонарності та автоматичний вибір оптимальної кількості часових затримок забезпечує відповідність статистичним вимогам методу та підвищує достовірність результатів. Отримана реалізація може бути використана як основа для подальших етапів аналізу структури системи та виявлення ключових факторів її динаміки.

2.2 Оцінка важливості факторів

У багатофакторних системах важливо не лише встановити факт причинного впливу, а й зрозуміти, наскільки суттєво кожна змінна впливає на результат. Тест Грейнджера дозволяє виявити наявність причинно-наслідкових зв’язків між змінними, проте він не надає інформації про їх важливість для системи. Крім того, метод працює з лінійними залежностями, що обмежує його застосування у складних системах, де часто присутні ймовірні нелінійні взаємозв’язки. Тому, у межах даної роботи, результати тесту Грейнджера доповнюються алгоритмом машинного навчання Random Forest, який оцінює значимість кожного фактору системи. Поєднання обох підходів забезпечує гнучкіший аналіз взаємозв’язків багатофакторних систем.

Модель Random Forest – це ансамблевий метод регресії та класифікації, який поєднує велику кількість дерев рішень, побудованих на підмножинах навчальних даних. Принцип роботи полягає в тому, що кожне дерево формує власне рішення, а остаточний результат отримується шляхом усереднення (для регресії) або голосування (для класифікації) окремих дерев [18]. На рисунку 2.1 зображено, як дані проходять через кілька дерев рішень, після чого їхні результати об’єднуються в один загальний прогноз.

Однією з переваг Random Forest є можливість визначати важливість кожної змінної для системи. У цій роботі застосовано метрику MDI – середнє зменшення домішок. Вона оцінює, наскільки ефективно кожна змінна ділить дані у вузлах дерев, тобто, наскільки вона зменшує розкид значень цільового показника. Чим

частіше та ефективніше змінна використовується для поділу, тим більшою є її важливість [19].

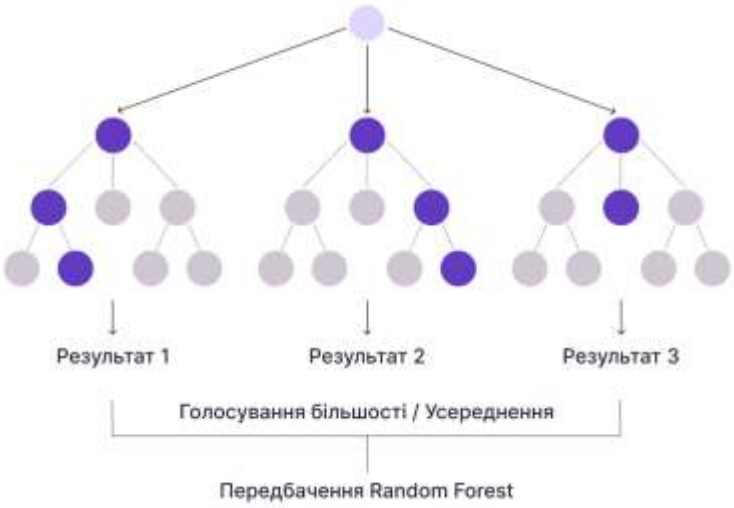


Рисунок 2.1 – Схематичне представлення роботи моделі Random Forest

У розробленій моделі, після виявлення змінних, що мають статистично підтверджений причинний вплив на цільовий показник, формується нова вибірка даних. До неї входять лише виявлені рушійні фактори та залежна змінна. Для підвищення об’єктивності моделювання, вона поділяється на навчальну та тренувальну частини з використанням функції train_test_split бібліотеки sklearn.model_selection, що забезпечує випадковий, але контрольований розподіл спостережень.

На основі навчальної вибірки створюється модель RandomForestRegressor із фіксованим числом дерев і випадковим станом для стабільності результатів при повторних запусках. Після навчання моделі обчислюється числовий показник важливості, який показує, наскільки сильно ця змінна впливає на результат. Значення ваг нормалізуються в межах від 0 до 1, де більший показник відображає вагоміший внесок кожної змінної у формуванні цільового значення.

Завдяки поєднанню класичного статистичного підходу та алгоритму машинного навчання вдається не лише встановити наявність причинних зв’язків, а й оцінити їхню вагу, що має важливе практичне значення для розставлення

пріоритетів та прийняття обґрунтованих рішень в складних системах. Така інтеграція дозволяє підвищити глибину аналізу та адаптувати систему до умов, де існують і лінійні, і потенційно нелінійні залежності.

2.3 Прогнозування часових рядів

Однією із основних задач даного дипломного проєкту є прогнозування майбутніх значень цільового показника системи. Для цього доцільно використовувати сучасні інструменти, які здатні обробляти часові ряди з урахуванням їх динаміки та минулих значень. Одним із таких програмних засобів є модель нейронної мережі LSTM – тип рекурентної нейронної мережі, створений для запам’ятовування довгострокових залежностей у послідовностях даних.

Основна ідея роботи LSTM полягає в тому, що вона використовує спеціальну комірку пам’яті для зберігання накопиченої інформації і дозволяє контролювати, яку частину треба зберігати, оновлювати та передавати [20]. На рисунку 2.2 показано архітектуру одного з блоків прихованого шару моделі LSTM, який виконує обробку даних на певному кроці часу *t*. Він є складовою повноцінного LSTM-шару, що розміщується між вхідним та вихідним шарами моделі. На вхід подаються два вектори: поточне значення ряду X_t та прихований стан з попереднього кроку h_{t-1}. Також передається стан довготривалої пам’яті C_{t-1}, який оновлюється в ході обчислень. До складу цього блоку входять вентилі – забувальний, вхідний та вихідний – які керують потоком інформації всередині мережі.

Першим кроком роботи LSTM-блоку є забувальний вентиль f_t Він визначає, яку частину старої інформації потрібно зберегти, а яку – відкинути. Цей шар розраховується за допомогою сигмоїдної функції активації:

$$f_t = \sigma(W_f \cdot [h_{t-1}, X_t] + b_f),$$

де σ – сигмоїдна функція, яка повертає значення в межах від 0 до 1,

W_f – вагові коефіцієнти,

b_f – вектор зміщення.

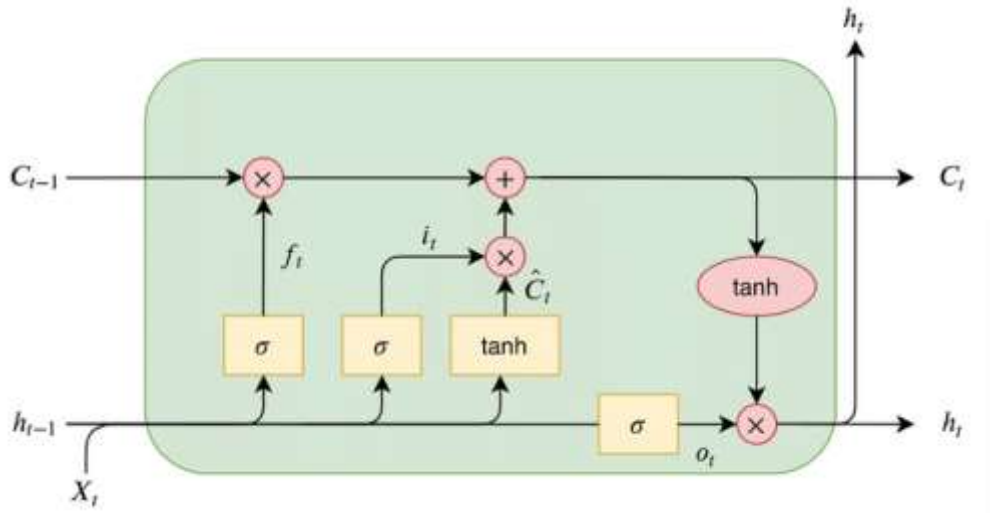


Рисунок 2.2 – Архітектура одного із блоків прихованого шару моделі LSTM

Наступним етапом є побудова вхідного вентиля i_t, який визначає, яка нова інформація повинна потрапити до комірки пам’яті. Для цього формується відповідний вектор значень, який контролює оновлення, також із застосування сигмоїдної функції активації:

$$i_t = \sigma(W_i \cdot [h_{t-1}, X_t] + b_i),$$

де σ – сигмоїдна функція, яка повертає значення в межах від 0 до 1,

W_i – вагові коефіцієнти,

b_i – вектор зміщення.

Паралельно з цим формується вектор-кандидат нового стану пам’яті $\hat{C}_t$, що містить потенційні нові значення, які можуть бути включені до довготривалої пам’яті. Він створюється за допомогою тангенс-функції активації, яка застосовується для поєднання попереднього прихованого стану h_{t-1} та поточного входу X_t. Визначається за формулою:

$$\hat{C}_t = \tanh(W_C \cdot [h_{t-1}, X_t] + b_C),$$

де tanh – гіперболічна тангенс-функція, яка стискає значення до діапазону від -1 до 1 і дозволяє зберігати інформацію про напрямок впливу (позитивний чи негативний),

W_C – вагові коефіцієнти,

b_C – вектор зміщення.

Це значення не додається до пам'яті, а проходить додаткову перевірку через вхідний вентиль, який вирішує, яка частина нової інформації є актуальною. Після цього відбувається оновлення стану довготривалої пам'яті C_t, який формується як результат поєднання старої та нової інформації. Формула оновленого стану довготривалої пам'яті має такий вигляд:

$$C_t = i_t \cdot \hat{C}_t + f_t \cdot C_{t-1},$$

де i_t – вхідний вентиль,

$\hat{C}_t$ – вектор-кандидат нового стану пам'яті,

f_t – забувальний вентиль,

C_{t-1} – попередній стан пам'яті.

Завдяки такій структурі пам'ять оновлюється там, де це справді потрібно, що дозволяє мережі ефективно працювати з довгими часовими рядами та уникати накопичення непотрібної інформації.

Після оновлення стану пам'яті активовується вихідний вентиль o_t, який визначає, яка частина нової інформації буде використана для формування виходу блоку. Він обчислюється за формулою:

$$o_t = \sigma(W_o \cdot [h_{t-1}, X_t] + b_o),$$

де σ – сигмоїдна функція, яка повертає значення в межах від 0 до 1,

W_o – вагові коефіцієнти,

b_o – вектор зміщення.

На основі цього значення формується прихований стан h_t, який є виходом поточного кроку та буде входом для наступного:

$$h_t = o_t \cdot \tanh(C_t),$$

де o_t – вихідний вентиль,

C_t – стан комірки пам’яті на кроці *t*.

Таким чином, модель LSTM керує тим, яка інформація зберігається, яка оновлюється, а яка ігнорується, що робить її ефективною у роботі із часовими рядами зі складними залежностями [21].

Для розробки моделі прогнозування у даному проєкті було використано попередньо навчену LSTM-модель, збережену у форматі .h5, яку було додатково донавчено на статистичних даних з найбільш поширених сфер – економіки та фінансів. Це дозволило створити універсальну структуру прогнозування, яка здатна адаптуватися до різних задач без потреби навчати модель з нуля для кожного нового випадку.

У цій роботі модель використовується для прогнозування цільового показника системи на основі його минулих значень. Такий підхід було обрано завдяки швидкому навчанню, простоті реалізації та сумісності із будь-якими наборами вхідних даних незалежно від кількості факторів. Модель здатна адаптуватися до різних типів часових рядів, що робить її універсальним інструментом для короткострокового аналізу. У перспективі передбачається розширення функціоналу системи – зокрема, здійснення багатофакторного прогнозування з урахуванням незалежних змінних, що мають підтверджений причинний вплив із попереднім аналізом. Це дозволить отримати точніші результати та враховувати комплексну динаміку системи.

Програмна реалізація модуля побудована на основі бібліотек TensorFlow, sklearn, NumPy та Matplotlib. Дані попередньо нормалізуються до інтервалу від 0 до 1, за допомогою MinMaxScaler модуля sklearn.preprocessing, що дозволяє узгодити масштаб нових даних із тими, на яких модель була навчена. Виконується покроковий прогноз: останні *n* значень ряду передаються на вхід моделі LSTM і в результаті рекурсивно додається до послідовності, яка використовується для наступного кроку. Процес повторюється до досягнення заданої кількості кроків уперед. Після отримання результатів виконується зворотне масштабування даних

та побудова графіка, що дозволяє користувачам легко візуально оцінити зміну цільового показника та зробити аналітичні висновки.

Застосування LSTM-моделі в межах програмного продукту дає змогу не лише прогнозувати поведінку системи на основі попередніх значень цільового показника, а й забезпечити користувача гнучким, адаптивним інструментом аналітики. Завдяки автоматизованій обробці даних, без потреби в ручному налаштуванні параметрів, модуль забезпечує швидкий результат прогнозу, що підвищує ефективність прийняття рішень. Такий підхід особливо корисний у ситуаціях, коли точність та швидкість оцінки майбутньої динаміки є важливими для планування або реагування на зміни в системі.

2.4 Формування вхідних та вихідних даних

Правильне формування вхідних та вихідних даних є важливим етапом у побудові ефективних систем аналізу. Якість та структура вхідних даних визначають, наскільки достовірними будуть результати. Форма представлення вихідної інформації впливає на зручність інтерпретації та прийняття управлінських рішень.

У проєкті використовується таблична модель даних у форматі .xlsx. Вхідний файл повинен містити щонайменше 3 колонки: дату, впорядковану за часом та із рівномірним часовим інтервалом; незалежну змінну, яка потенційно впливає на систему; та залежну змінну, значення якої є об’єктом аналізу та прогнозування. Також допускається наявність більшої кількості незалежних змінних – у такому разі система автоматично обробляє кожну з них, перевіряючи її можливий причинний вплив на цільовий показник. Це дозволяє проводити повноцінний багатофакторний аналіз, враховуючи взаємодії між численними параметрами (рисунок 2.3).

У рамках дослідження використовувалися як реальні, так і штучно згенеровані дані. Реальні дані було зібрано із відкритих джерел: SaveEcoBot,

Investing.com, Мінфін України та Державна служби статистики України. Вони охоплювали показники економіки, валютного курсу, інфляції, цін на енергоносії тощо. Такі дані дозволяють вивчити реальні залежності у прикладних системах, зокрема у економічній та фінансовій сфері.

	A	B	C	D	E	F	G
1	Дата	Ставка НБУ	Інфляція	Торговий баланс, млн $	Середня заробітна плата	Ціна на нафту	Курс UAH/USD
2	01.05.2001	21	100,4	421	302,96	28,37	5,41
3	01.06.2001	19	100,6	172,4	317,81	26,25	5,4
4	01.07.2001	19	98,3	134,3	327,31	26,35	5,37
5	01.08.2001	17	99,8	-101,8	329,33	27,2	5,35
6	01.09.2001	15	100,4	6,6	326,34	23,43	5,34
7	01.10.2001	15	100,2	95,6	335,75	21,18	5,31

Рисунок 2.3 – Приклад вхідного файлу

Для тестування працездатності алгоритмів і перевірки коректності результатів використовувалися штучні дані, згенеровані засобами Excel. У них заздалегідь були закладені причинно-наслідкові зв’язки між змінними і модель повинна була виявити ці залежності. Результати показали, що алгоритм успішно визначає задачі причинності, що підтверджує його точність і надійність у практичному використанні.

Вихідні результати моделі впливу рушійних факторів на систему представлені у комплексному вигляді, де кожен елемент доповнює інші та формує цілісну картину взаємодій. Користувач отримує спрямований ациклічний граф, який показує встановлені напрямки впливу між змінними та їх важливість для системи. Це дає змогу наочно побачити, які фактори є причинами змін цільового показника та як вони взаємодіють між собою. Одночасно формується текстова інтерпретація, у якій описано встановлені залежності, вказано часову затримку, з якою проявляється вплив, та визначено найвпливовішу змінну. Такий опис робить результати зрозумілими навіть для користувачів без спеціальної підготовки. Також надається таблиця, яка містить інформацію про значущість впливу незалежних змінних на цільовий фактор на кожному кроці часової затримки.

Модель прогнозування цільового показника генерує прогноз на задану кількість періодів уперед на основі попередніх значень. Результати подаються у вигляді графіка, який поєднує частину фактичних даних та прогнозовані значення. Це дозволяє користувачу побачити зміну динаміки показника в часі. Крім того, система також формує текстове пояснення результату, з якого користувач може побачити останню відому величину фактора, діапазон прогнозованих значень та загальна тенденція змін, наприклад, зростання чи спадання.

Таким чином, система забезпечує весь процес аналізу – від завантаження даних до зрозумілого подання результатів аналізу. Вона дозволяє не лише виявляти структуру взаємозв’язків у системі, а й будувати прогнози ключових показників, що має важливе значення для аналітики та ухвалення обґрунтованих рішень.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

У цьому розділі детально описано програмну реалізацію вебсервісу, що дозволяє виявляти причинно-наслідкові зв’язки між змінними в часових рядах та здійснювати прогнозування цільового показника. Представлено вибір засобів розробки, архітектуру системи, структуру клієнтської та серверної частин, реалізацію алгоритмічних модулів, базу даних користувачів та оцінку точності отриманих результатів. Такий поетапний опис дозволяє глибше зрозуміти внутрішню логіку роботи програмного продукту та обґрунтувати прийняті технічні рішення.

3.1 Засоби реалізації

Вибір засобів розробки є важливим етапом у реалізації програмного забезпечення, оскільки від цього залежить ефективність створення, розширюваність системи, зручність підтримки та взаємодії з користувачем. У рамках даного проєкту для побудови клієнтської та серверної частин і взаємодії з базою даних було обрано програмні засоби, які забезпечують простоту реалізації, оптимальне поєднання функціональності та інтеграції з алгоритмами машинного навчання.

3.1.1 Засоби розробки серверної частини

Серверна частина програмного продукту виконує основну логіку взаємодії з даними, обробку запитів, взаємодію з базою даних, виконання аналітичних алгоритмів, маршрутизацію запитів та формування відповідей клієнтській частині. Для реалізації серверної логіки використовуються різні фреймворки та платформи, залежно від вимог до продуктивності, масштабу та зручності розробки. Серед

найпоширеніших рішень – Django, Flask, FastAPI, Node.js, ASP.NET, Spring Boot. Кожен із цих інструментів має свої переваги та обирається відповідно до специфіки проєкту.

У даному проєкті було обрано мову програмування Python разом із фреймворком Flask [22]. Таке поєднання забезпечує просту структуру програми, незначну кількість шаблонного коду та гнучкість при визначення маршрутів. Це дозволило поєднати вебкомпоненти з модулями обробки даних та машинного навчання. Flask безпосередньо взаємодіє з модулями реалізації тесту Грейнджера, побудови моделі Random Forest, прогнозування із застосуванням нейронної мережі LSTM, візуалізації результатів та генерації PDF-звітів. Крім того, із застосуванням Flask-Login розроблено систему авторизації користувачів із підтримкою реєстрації, входу, скидання пароля та захисту доступу до обмежених функцій.

3.1.2 Засоби розробки клієнтської частини

Клієнтська частина вебсервісу відповідає за відображення інтерфейсу, обробку дій користувача, завантаження файлів та вивід результатів аналізу причинно-наслідкових зв’язків та прогнозування. Для цього застосовуються різні технології – від стандартних HTML, CSS, JavaScript до фреймворків React, Vue.js, Angular тощо, які забезпечують гнучке керування станом, маршрутизацію та інтерактивність. Однак, у випадках, коли не потрібна надмірна складність на стороні клієнта, доцільно використовувати класичні вебтехнології.

З урахуванням архітектури застосунку та вимог серверної частини було обрано класичні засоби: HTML, CSS, JavaScript і шаблонізатор Jinja2, який входить до складу фреймворку Flask. Це дозволило забезпечити динамічне формування контенту, просте відображення елементів залежно від типу користувача та повноцінну інтеграцію із серверною логікою. Використання JavaScript дало змогу реалізувати перевірку даних, динамічне оновлення інтерфейсу та передачу запитів на сервер без перезавантаження сторінки. Це зробило роботу з інтерфейсом більш зручною та інтерактивною.

3.1.3 Засоби розробки бази даних

Для зберігання облікових даних користувачів та іншої інформації, пов’язаної з роботою системи, застосовуються різноманітні системи управління базами даних. Серед них можна виділити PostgreSQL та MySQL, які відзначаються високою надійністю та масштабованістю, а також нереляційні рішення: MongoDB для зберігання структурованих документів, Firebase – хмарна платформа з підтримкою реального часу. Вибір конкретної СУБД залежить від масштабів проєкту, вимог до продуктивності, типу даних та умов розгортання.

З урахуванням специфіки даного проєкту було обрано SQLite – легку реляційну базу даних, яка не потребує окремого сервера та легко вбудовується у Python-додатки [23]. Взаємодію із БД реалізовано за допомогою ORM-бібліотеки SQLAlchemy, яка дозволяє визначати таблиці через Python-класи та працювати з даними без написання SQL-запитів. Це спростило реалізацію, підвищило безпеку взаємодії з БД та забезпечило узгодженість структури даних з іншою частиною серверного коду.

3.1.4 Ізоляція програмного середовища

Ізоляція програмного середовища відіграє важливу роль у забезпеченні стабільності роботи системи та уникнення конфліктів між залежностями. Для цього під час розробки було створено окреме віртуальне середовище на основі вбудованого інструменту venv в Python. Його активація дозволяє локально встановлювати бібліотеки без впливу на глобальні налаштування системи та гарантує, що всі модулі, потрібні для роботи системи, працюють у стабільному та передбачуваному середовищі. Ізоляцію програмного середовища було створено за допомогою команди python -m venv venv і активовується залежно від операційної системи: source venv/bin/activate для Unix-подібних систем та venv /Scripts/acrivate для Windows [24]. Такий підхід забезпечив контрольовану розробку та забезпечив

швидке розгортання вебсервісу на інших пристроях без непередбачуваних змін у поведінці коду.

3.2 Архітектура програмного продукту

Організація архітектури програмного продукту є важливим етапом проєктування, оскільки визначає загальну структуру системи, способи взаємодії компонентів і розподіл функціональних можливостей між ними. Чітко визначена архітектура дає змогу досягти модульності, полегшує масштабування функціоналу, спрощує тестування, підтримку та подальший розвиток програмного продукту.

Щоб визначити, які функції має виконувати програмне забезпечення, було побудовано діаграму прецедентів (рисунок 3.1). Вона відображає основні сценарії взаємодії між системою та користувачами і допомагає визначити доступні функції для кожної категорії.

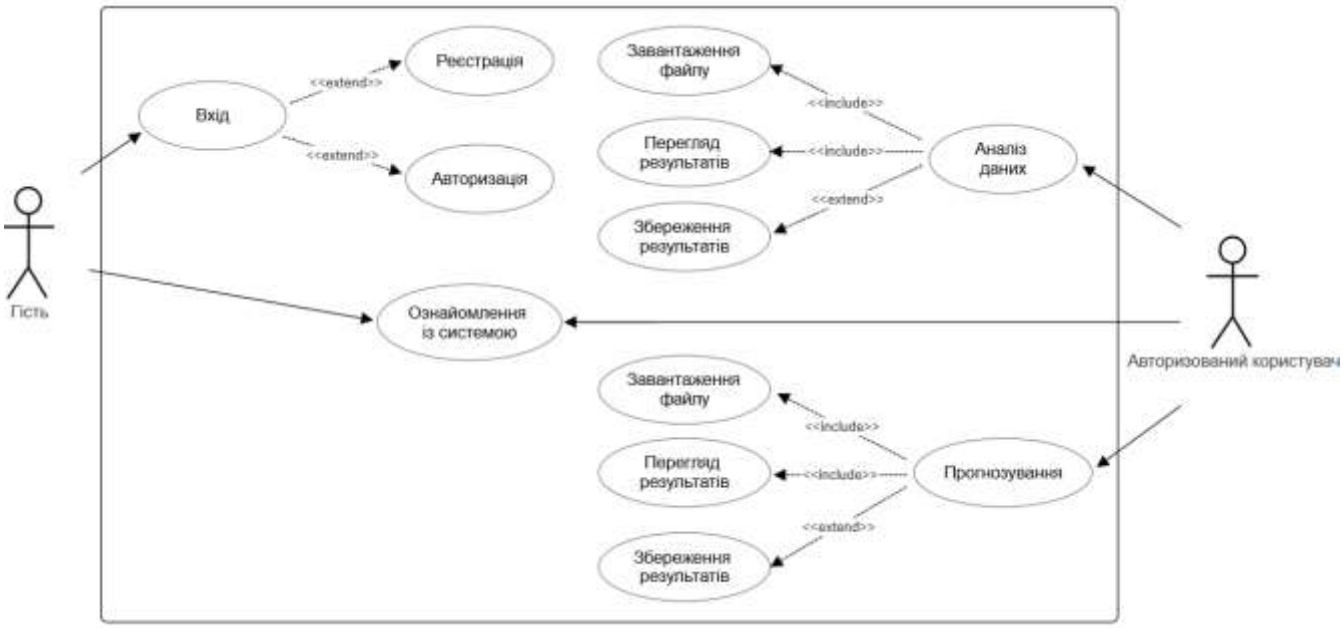


Рисунок 3.1 – Діаграма прецедентів

Система передбачає два типи користувачів – гість та авторизований. Гість має доступ лише до ознайомчого функціоналу: огляд методів, які були використані для розробки функціональних модулів, та перегляд результатів причинно-наслідкових зв’язків екологічної та економічної сфери, які були проведені на реальних статистичних даних. Натомість, авторизованому користувачу відкривається доступ до повного функціоналу вебсервісу. Він може провести аналіз для виявлення рушійних факторів в системі та прогнозування на основі власних зібраних статистичних показників. За бажанням можна зберегти отримані результати у вигляді PDF-звітів для подальшого опрацювання, аналізу динаміки та прийняття обґрунтованих рішень. Діаграма прецедентів є важливою на етапі проєктування, оскільки дозволяє сформувати уявлення про поведінку системи і на її основі побудувати архітектурну модель.

Після аналізу функціональних можливостей програми було обрано трирівневу архітектуру, яка передбачає розділення на три незалежні шари: клієнтський, серверний та бази даних (рисунок 3.2). Таких підхід дозволяє забезпечити масштабованість, покращену безпеку даних та спрощене оновлення окремих компонентів без втручання в інші частини.

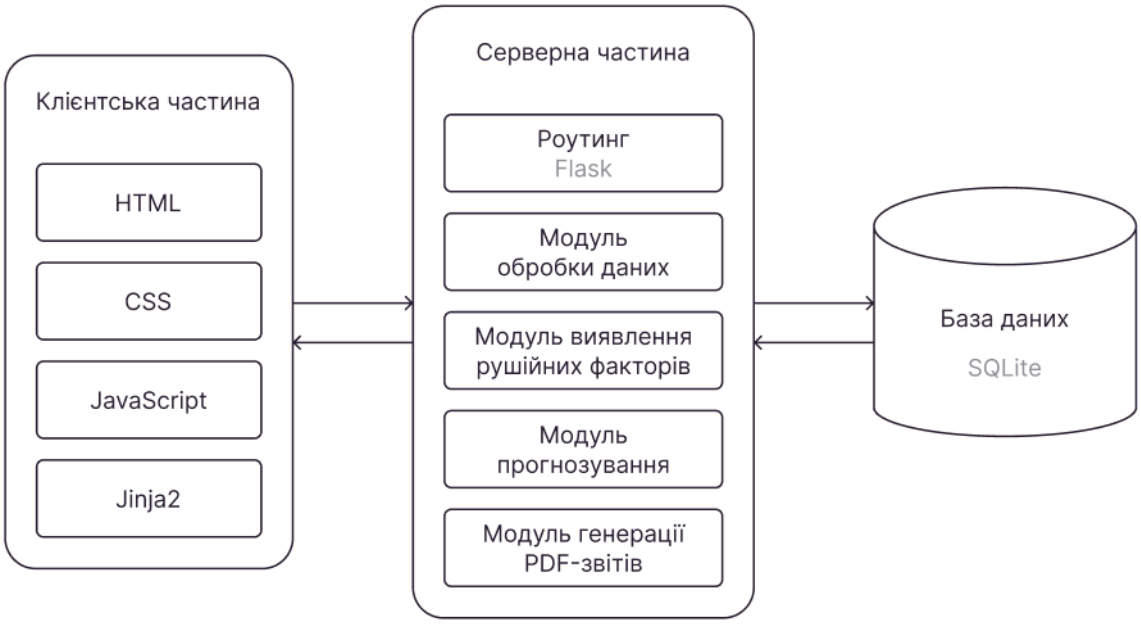


Рисунок 3.2 – Архітектура системи

Клієнтський рівень відповідає за взаємодію з користувачем. Він формує графічний інтерфейс, через який користувач виконує всі дії – реєстрацію, вхід, завантаження файлів, перегляд результатів аналізу та прогнозування. На цьому рівні відображається візуальний контент: графіки, таблиці, пояснювальні тексти. Він адаптований для багатосторінкової навігації, забезпечуючи зручну організацію даних та динамічну зміну контенту залежно від введених параметрів та результатів обчислень. Особлива увага приділена інтерактивності: користувач може завантажити файл і одразу переглянути результат без перезавантаження сторінки.

Серверна частина – ядро застосунку. Вона обробляє запити від клієнта, виконує обчислення, керує сесіями користувачів і забезпечує взаємодію з базою даних. Серверна логіка реалізована з використанням фреймворку Flask, який забезпечує легку маршрутизацію, обробку HTTP-запитів, авторизацію та інтеграцію з модулями машинного навчання. Основними завданнями цього рівня є перевірка даних користувача та авторизація; обробка завантажених файлів і параметрів аналізу; запуск алгоритмів тесту Грейнджера, Random Forest та моделі LSTM; обчислення результатів, формування DAG-графів та побудову графіків; створення динамічних текстових висновків; генерація PDF-звітів. Також він виконує обробку винятків, перевірку форматів даних та контроль доступу до функціональності системи.

Останній рівень архітектури відповідає за збереження інформації про користувачів. Він забезпечує надійне зберігання облікової інформації, зокрема електронних адрес та хешованих паролів. Також виконує роль основи для системи авторизації. Завдяки рівню бази даних система може контролювати доступ до аналітичного функціоналу, розмежовувати права користувачів і забезпечувати безпеку персональних даних. Взаємодія з цим шаром відбувається через серверну частину, яка виступає посередником між клієнтським рівнем і внутрішнім сховищем з боку користувача. Це дозволяє легше підтримувати систему та за потреби розширювати її можливості.

Взаємодія між рівнями відбувається послідовно. Клієнтський рівень надсилає запит до сервера, серверна часина обробляє його, за потреби звертається до бази

даних, проводить аналітичні розрахунки, формує відповідь та повертає її клієнту. Така архітектура забезпечує гнучкість, модульність і дозволяє розширювати систему в майбутньому. Усі рівні є відокремленими, що дає змогу замінити або модифікувати один із компонентів без порушення функціональності решти системи.

3.3. Клієнтська частина

Клієнтська частина вебсервісу реалізована на основі HTML, CSS, JavaScript та шаблонізатора Jinja2, який дозволяє динамічно формувати HTML-код у відповідності до даних, що надходять із серверної частини. Такий підхід забезпечив гнучкість у відображенні контенту, адаптивність до типів користувача і зручність подальшого масштабування інтерфейсу.

Основою клієнтської частини є шаблон base.html, який містить спільні елементи інтерфейсу: header, footer, підключення стилів та скриптів. Усі інші сторінки – index.html, analysis_upload.html, forecast.html, results_generic.html – є дочірніми і наслідують цей шаблон. Це дає змогу централізовано змінювати загальний вигляд і структуру сайту. Наприклад, шаблон index.html містить вступну інформацію про проєкт, переваги причинно-наслідкового аналізу та основні етапи алгоритму. Він також включає заклик до дії – кнопки «Почати аналіз», які, залежно від стану користувача, перенаправляють на сторінку входу або на сторінку завантаження файлу для виявлення впливу рушійних факторів на систему.

Шаблон analysis_upload.html відповідає за інтерфейс проведення аналізу причинно-наслідкових зв’язків. На першому кроці користувач бачить блок з інструкціями щодо формату даних, приклад таблиці, кнопку для завантаження файлу та переходу до наступного етапу. Після вибору файлу інтерфейс динамічно змінюється: відображається назва файлу, поле для введення p-значення, що є рівнем ймовірності, при якому відхилення вважаються значущими, та кнопка «Аналізувати». Після виконання аналізу відображаються результати у вигляді

спрямованого ациклічного графа (DAG-графа), який передається із сервера у форматі base64 для більш чіткого зображення, динамічного текстового опису та таблиці з висновком про причинний вплив незалежних змінних на цільовий показник на кожному часовому кроці.

Весь процес клієнтського блоку контролюється JavaScript-логікою, розміщеною у файлі analysis.js. Цей скрипт забезпечує перевірку формату файлу, відображення помилок у разі порушення вимог, обробку введеного p-значення з перевіркою діапазону значень, надсилання запиту на сервер та відображення відповіді без перезавантаження сторінки. Таблиця результатів будується за допомогою функції generateTableHTML(), яка формує HTML-розмітку на основі отриманого JSON-об’єкта.

Окрема сторінка forecast.html реалізує інтерфейс прогнозування часових рядів за допомогою моделі LSTM. Логіка взаємодії схожа з аналізом причинно-наслідкових зв’язків: користувач завантажує файл, що містить стовпці з датою та цільовим показником, обирає кількість кроків для прогнозу, запускає процес та отримує результати у вигляді графіка та пояснювального тексту. Усі дії обробляються JavaScript-кодом, винесеним у файл forecast.js, який також відповідає за перевірку введених значень, передачу даних, формування запиту, обробку помилок та зворотну реакцію інтерфейсу.

Після завершення аналізу виявлення впливу рушійних факторів на систему та прогнозування користувачу надається можливість зберегти результати у форматі PDF-звіту. Цей функціонал реалізовано для обох сторінок: analysis_upload.html та forecast.html. У першому випадку звіт містить DAG-граф з причинно-наслідковими зв’язками, таблицю зі значеннями впливів та текстовий висновок. У другому – виводяться графік прогнозу, на якому поєднано частину фактичних та майбутні значення, та коротке пояснення динаміки. В обох випадках активовується кнопка «Завантажити результати», яка ініціює збір усіх компонентів – графіки, тексти, таблиці – та передає їх на сервер у вигляді об’єкта FormData. На серверній частині формується PDF-документ, який одразу завантажується на пристрій користувача без потреби додаткового підтвердження. Він містить повноцінний звіт з

урахуванням української мови, структури звіту та візуального оформлення. Така функція спрощує збереження результатів аналізу та дозволяє їх використовувати для подальшої роботи.

У структурі клієнтської частини окрему увагу приділено відображенню результатів досліджень за заздалегідь підготовленими файлами, статистичні дані яких було зібрано на відкритих джерелах. Для цього створено шаблон results_generic.html, який динамічно наповнюється залежно від теми: вплив бойових дій на якість повітря в Україні та вплив інфляції на ставки Національно банку України. Цей шаблон використовує змінні header, paragraphs, image, що передаються з серверної частини. Після завантаження надсилається запит на /analyze_topic, який повертає граф, висновок та таблицю.

Загалом, клієнтська частина є багаторівневою системою, що забезпечує зручну логіку навігації, ефективну взаємодію із серверною частиною та відображення складних результатів аналізу в доступній формі. Поєднання Jinja2 із JavaScript дозволяє динамічно оновлювати вміст сторінок та забезпечує високу інтерактивність вебсервісу без надмірної складності. Це рішення є ефективним для аналітичних систем, що працюють із даними та потребують швидкої реакції на дії користувача.

3.4 Серверна частина

Серверна частина програмного продукту реалізована в єдиному модулі app.py, який є центральним компонентом обробки запитів, взаємодії з клієнтською частиною, обробкою даних і формування результатів аналізу. У ній реалізовано повноцінну систему маршрутизації, яка охоплює всі ключові точки взаємодії з користувачем. Зокрема, маршрути /login та /register обробляють автентифікацію, створення облікового запису й перевірку пароля. Для аналізу даних призначено маршрут /analysis та /analyze, які приймають завантажені файли, p-значення, запускають алгоритми й повертають результати у форматі JSON. Аналогічно,

маршрут /forecast обробляє запити на побудову прогнозу часових рядів, а маршрути /download_pdf та /download_forecast_pdf відповідають за генерацію й передачу звітів у форматі PDF.

Основу структури програми становить екземпляр Flask та конфігурація бази даних через SQLAlchemy. У програмному продукті використовується SQLite як вбудована БД, що не вимагає зовнішнього сервера та зберігає дані про зареєстрованих користувачів. Авторизація реалізована за допомогою Flask-Login, де за перевірку входу відповідає менеджер LoginManager. Користувачі зберігаються як об'єкти класу User, що реалізує інтерфейс UserMixin, а хешування паролів виконується через werkzeug.security.

Важливою частиною логіки є обробка завантажених даних. Сервер приймає Excel-файл, перетворює його у структуру DataFrame за допомогою бібліотеки pandas, проводить попередню обробку та передає ці дані до моделей аналізу. Перевірка коректності введеного p-значення та кількості колонок реалізована всередині маршруту /analyze. У разі помилки система повертає JSON-відповідь із відповідним повідомленням, яке одразу відображається на клієнтській частині, забезпечуючи зручну взаємодію з користувачем та запобігаючи некоректному виконанню аналізу.

Визначення причинно-наслідкових зв’язків виконується за допомогою statsmodels, зокрема, функції grangercausalitytests. Для побудови моделі оцінки важливості факторів використовується RandomForestRegressor пакету sklearn.

Після виконання розрахунків результати виводяться у вигляді таблиці, динамічного текстового висновку та графа. Для кожної причинно-наслідкової пари створюється вершина та напрямлена дуга зі значеннями важливості. Граф зберігається у SVG-форматі у буфері BytesIO, кодується у base64 та повертається до клієнтської частини.

Значну роль відіграє модуль формування PDF-звітів. Для цього використовується бібліотека reportlab, яка дозволяє програмно згенерувати документ у форматі A4 з таблицями, текстом та зображеннями. Зокрема, у звіті виводяться перші 10 рядків завантажених даних, граф причинності або графік

прогнозу та текстове пояснення результатів. Підключення шрифту DejaVuSans.ttf забезпечило коректне відображення української мови у PDF-документі. Збереження зображень відбувається з тимчасовим записом у буфер, після чого генерується і передається у файл користувачу через send_file.

Серверна частина також реалізує повну логіку авторизації. Маршрут /forgot_password дозволяє скинути пароль після перевірки електронної адреси. Усі шаблони формуються з урахуванням стану користувача, а доступ до сторінок аналізу даних та прогнозування обмежено для гостей за допомогою декоратора @login_required.

Таким чином, серверна частина програми виконує роль посередника між інтерфейсом користувача та обробкою даних. Вона обробляє всі запити з клієнтського інтерфейсу, відповідає за маршрутизацію, перевірку цілісності й структури вхідних даних, обробку завантажених файлів, запуск аналітичних моделей, генерацію результатів та створення звітів. Уся логіка виконання запитів чітко структурована, що забезпечує узгоджену взаємодію між модулями системи та швидке реагування на дії користувача. Обрані інструменти дозволяють ефективно реалізувати повноцінний цикл «ввід – аналіз – вивід» у межах єдиної архітектури моделі, зберігаючи масштабованість, надійність і зручність обслуговування.

3.5 Алгоритмічні модулі аналізу та прогнозування

Модульна побудова програмної системи забезпечує її гнучкість, масштабованість та зручність в обслуговуванні. Кожен модуль виконує чітко визначене завдання, що дозволяє спростити розробку та подальше розширення функціоналу. Такий підхід полегшує тестування окремих компонентів і забезпечує логічне розмежування функціональних обов’язків між частинами системи. У реалізованому вебсервісі ключовими є два модулі: модель впливу рушійних факторів на систему та модуль прогнозування. Вони обробляють завантажені дані,

виконують відповідні математичні операції та формують результати. Більш детальний опис роботи кожного з них буде розглянуто нижче.

3.5.1 Модель впливу рушійних факторів на систему

Модуль моделювання впливу рушійних факторів на систему реалізовує послідовність етапів, що дозволяє виявити причинно-наслідкові зв’язки між змінними часових рядів. Основною метою цього компоненту є встановлення, які фактори мають причинний вплив на цільовий показник і яка ступінь їх важливості. На діаграмі послідовності (рисунок 3.3) можна побачити покроковий процес взаємодії між користувачем, сервером, модулями підготовки даних, аналізу та генерації звіту.

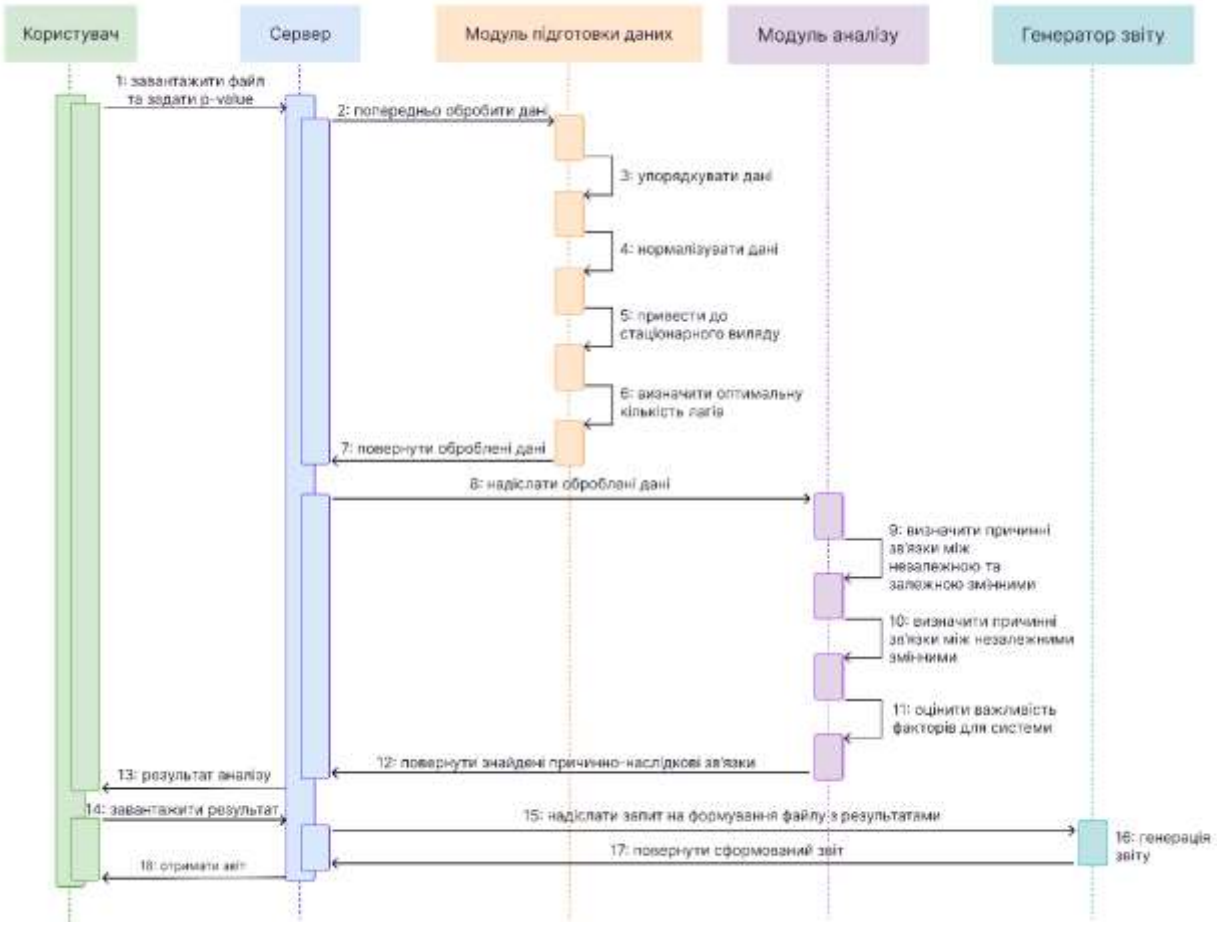


Рисунок 3.3 – Діаграма послідовності виявлення причинно-наслідкових зв’язків

На першому кроці користувач взаємодіє із клієнтською частиною, завантажуючи файл даних та задаючи p-значення, яке є пороговим значенням статистичної значущості. Цей запит надходить на сервер, де активовується логіка обробки. Отримана інформація передається до модуля підготовки даних, який виконує чотири важливі операції: упорядкування даних за датою, нормалізацію, приведення рядів до стаціонарного вигляду та визначення оптимальної кількості часових затримок.

Після завершення підготовки даних модуль повертає результат серверу, який передає їх у модуль аналізу. У цьому компоненті реалізовано три основні підпроцеси: визначення зв’язків між кожною парою незалежної та залежної змінними, оцінка взаємних зв’язків між усіма незалежними змінними та оцінка важливості кожного рушійного фактора для системи. Для виконання цього аналізу застосовується тест Грейнджера у поєднанні з алгоритмом машинного навчання Random Forest (Додаток Г).

Результатом роботи моделі є список усіх виявлених причинно-наслідкових факторів, які надсилаються на сервер, де формується відповідь клієнту, відображаючи динамічний текст, граф та таблицю залежностей. За бажанням користувач може завантажити результати, які формуються генератором звітів та повертаються у зручному форматі.

Модуль впливу рушійних факторів на систему забезпечує глибоку аналітичну обробку вхідних даних, інтегрує класичні статистичні методи та алгоритми машинного навчання для підвищення точності й наочності результатів та відіграє ключову роль у визначенні логіки взаємодій змінних у системі.

3.5.2 Модуль прогнозування

Модуль прогнозування (рисунок 3.4) виконує важливу функцію в системі – він дозволяє користувачу отримати передбачення майбутніх значень цільового показника на основі даних, які потенційно залежать у часі. Це дає змогу виявляти

тенденції, оцінювати можливі зміни в майбутньому та приймати обґрунтовані рішення на основі даних.

Процес прогнозування починається з того, що користувач завантажує файл із статистичними показниками за минулі періоди і вказує, на який проміжок часу потрібно сформувати прогноз. Сервер перевіряє коректність таблиці: структуру, наявність необхідних стовпці та заповнення полів. Далі відбувається підготовка даних. Вона містить впорядкування записів за датою та нормалізацію чисел у межах від 0 до 1.

Після оброки, дані передаються до моделі прогнозування. Вона використовує останні 60 значень із часового ряду як основу для побудови прогнозу. На цьому етапі активовується заздалегідь навчена нейронна мережа типу LSTM, збережена у форматі .h5, яка формує набір майбутніх значень на вказану кількість періодів. Потім відбувається зворотне масштабування, щоб повернути значення у звичний для користувача формат.

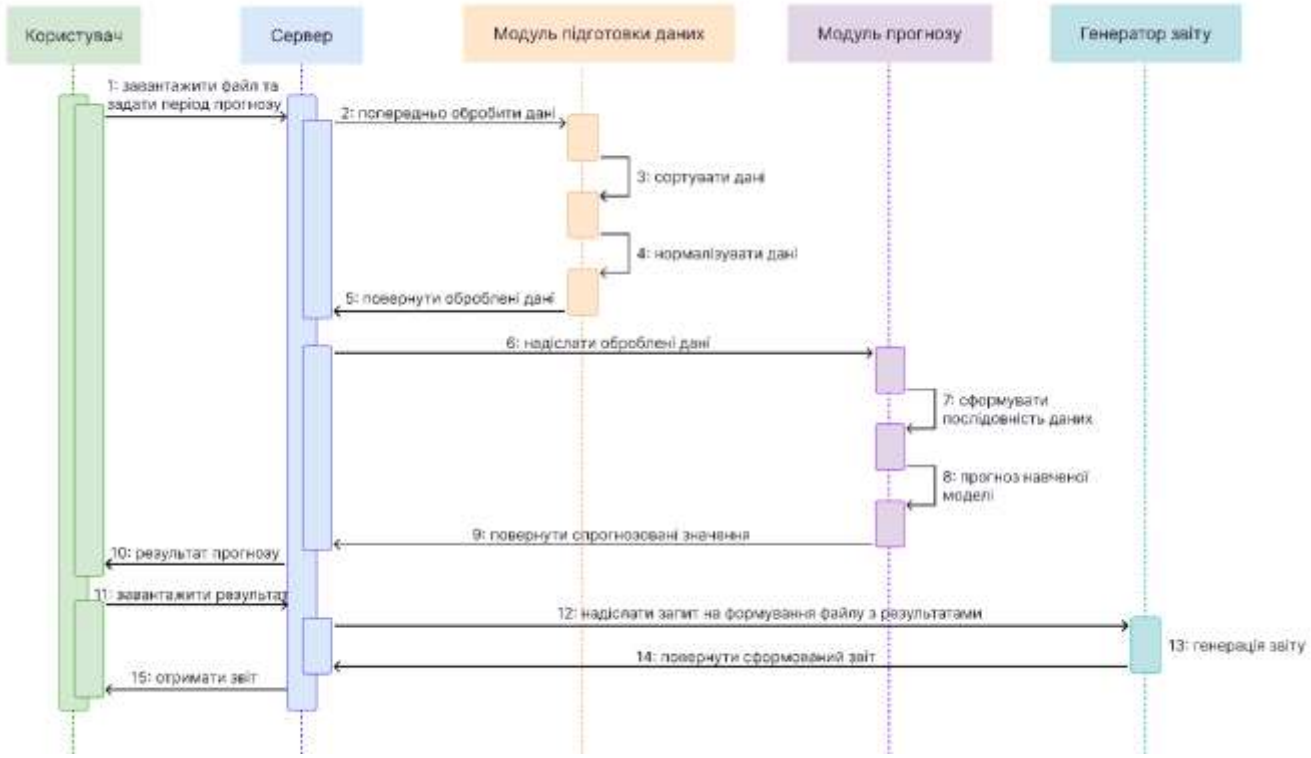


Рисунок 3.4 – Діаграма послідовності модуля прогнозування

Результати прогнозу повертаються на клієнтську частину, де користувач бачить графік , на якому поєднано частину фактичних та прогнозовані значення, а також отримує короткий текстовий висновок, який пояснює основні тенденції.

Крім інтерактивного перегляду, користувач може завантажити прогноз у форматі PDF-звіту. Документ містить графік та текстовий опис результату. Така форма подання зручна для подальшої аналітики, демонстрації або збереження в архів.

У результаті модуль прогнозування забезпечує повний цикл обробки часових рядів – від прийому даних до формування та збереження результатів. Він тісно взаємодіє з іншими модулями системи, що дозволяє автоматизувати процес і забезпечити приємний досвід користувача.

3.6 База даних

База даних містить лише одну таблицю Users (рисунок 3.5), яка виконує єдину, але важливу функцію – зберігання облікових записів зареєстрованих користувачів. Уся взаємодія із сервісом – зокрема, доступ до функцій аналізу та прогнозування – базується на авторизації, тому дана таблиця є обов'язковим компонентом архітектури системи.

	id	email	password
	Filte	Filter...	Filter...
1	1	anastasiatymkova@gmail.com	scrypt:32768:8:1$1I0XUyg7vj9UwQ7Q$bee901f3dab450...
2	2	anna@gmail.com	scrypt:32768:8:1$QPVtbdcstqM7Ynnc$26185bd49847d2...
3	3	alla@gmail.com	pbkdf2:sha256:1000000$kc47TYMOSv1N2jIE$bdcb8479...
4			

Рисунок 3.5 – Структура таблиці зареєстрованих користувачів у базі даних SQLite із прикладами хешованих паролів

Таблиця Users містить три поля:

1) id – ціле число, яке є первинним ключем. Значення цього поля є унікальним для кожного користувача і генерується автоматично під час створення нового запису. Це поле використовується для ідентифікації користувача всередині системи, незалежно від інших параметрів;

2) email – текстове поле, що зберігає електронну адресу користувача. Воно має бути унікальним, оскільки використовується як логін при вході в систему. Перевірка унікальності здійснюється під час реєстрації, щоб уникнути дублювання акаунтів;

3) password – текстове поле, яке містить хешований пароль. У таблиці не зберігається відкритий текстовий вигляд – замість цього при реєстрації або зміні пароля значення проходить хешування за допомогою криптографічного алгоритму scrypt. Це забезпечує належний рівень захисту даних.

Уся взаємодія з таблицею відбувається через відповідні методи серверної частини. Прямий доступ до бази даних відсутній, що дозволяє контролювати всі зміни та запити централізовано. Такий підхід мінімізує ризики пошкодження структури або витоку даних та забезпечує стабільну та безпечну роботу системи.

3.7 Оцінка точності моделей

Для підтвердження надійності та коректної роботи розроблених алгоритмів було проведено комплексну перевірку точності результатів для обох моделей: виявлення впливу рушійних факторів на систему та прогнозування цільового показника. Такий підхід дозволив протестувати стабільність алгоритмів на практиці та оцінити їх здатність до узагальнення результатів на нових наборах даних з різними характеристиками – наприклад, різною кількістю змінних, рівнем шуму або структурою часових рядів.

Особливу увагу було приділено здатності моделей адаптуватися до складної динаміки процесів, відображати реальні, а не випадкові залежності між

параметрами, виявляти прямі та опосередковані зв’язки. Це важливо для подальшого практичного застосування системи – у реальному середовищі дані часто мають нерівномірний розподіл, пропуски або складні взаємодії між змінними.

Перевірка працездатності моделі аналізу рушійних факторів здійснювалася на згенерованих тестових наборах даних, у яких було штучно задано залежності між змінними із часовим зсувом. У результаті застосування тесту Грейнджера модель успішно виявила всі задані причинно-наслідкові зв’язки, що підтверджує її функціональність і точність для виявлення взаємозалежностей у часових рядах.

Для оцінки якості моделі прогнозування було використано метрику MSE. Вона визначає середнє квадратичне відхилення прогнозованих значень від фактичних і обчислюється за формулою:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

де n – кількість точок прогнозу,

y_i – фактичні значення,

$\hat{y}_i$ – прогнозовані значення.

Дану метрику обрано через її простоту, поширеність у задачах регресії та чутливість до великих відхилень, що дозволяє виявляти навіть незначні неточності в прогнозі.

Було проведено порівняння трьох моделей прогнозування: ARIMA, VAR та LSTM. Всі моделі тестувалися на реальних та штучно згенерованих даних. Отримані результати наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння моделей прогнозування за метрикою MSE

Модель прогнозу	MSE на реальних даних	MSE на штучних даних
ARIMA	0,1114	0,1149
VAR	0,1032	0,0982
LSTM	0,0025	0,0027

Результати проведеної перевірки показали, що модель LSTM забезпечує найменші значення середньоквадратичної похибки порівняно з класичними підходами, такими як ARIMA чи векторна авторегресія. Це свідчить про її здатність точно відтворювати динаміку цільового показника навіть у складних багатофакторних умовах, що характерні для реальних систем. Висока точність прогнозування пояснюється гнучкою архітектурою нейронної мережі, яка дозволяє враховувати як короткострокові, так і довгострокові залежності між значеннями у часовому ряду.

Таким чином, здійснена перевірка підтверджує не лише правильність функціональних можливостей розроблених моделей, але й їхню здатність забезпечувати достовірні результати як у виявлення причинно-наслідкових зв’язків, так і у побудові прогнозу. Це підтверджує придатність реалізованої системи для практичного використання в аналітичних задачах, де важливо враховувати складну динаміку та взаємозв’язки між факторами.

4 ВЗАЄМОДІЯ КОРИСТУВАЧА З ПРОГРАМНИМ ПРОДУКТОМ

У цьому розділі наведено вимоги до використання вебсервісу, зокрема описано технічні умови доступу до системи та програмного забезпечення, необхідного для її стабільної роботи. Описано основні етапи взаємодії користувача з програмою, що дозволяє ознайомитися з її функціоналом та послідовністю дій при використанні.

4.1 Системні вимоги

Вебсервіс створено з урахуванням архітектури клієнт-сервер, що дозволяє працювати з ним без встановлення додаткового програмного забезпечення на комп’ютер користувача. Уся основна логіка та обчислення виконуються на сервері, а користувач отримує доступ до функцій системи через вебінтерфейс.

Для коректної роботи достатньо мати персональний комп’ютер зі встановленим веббраузером, який підтримує сучасні вебтехнології.

Наявність інтернет-з’єднання є обов’язковою умовою для взаємодії із системою, оскільки всі запити до сервера виконуються в режимі онлайн. Такий підхід дозволяє користувачу завжди працювати з актуальною версією програми, що зменшує ймовірність помилок та забезпечує стабільну роботу системи.

4.2 Демонстрація функціональних можливостей

Взаємодія користувача із програмною системою починається із головної сторінки (рисунок 4.1), на якій можна ознайомитися із призначенням вебсервісу та

стислим описом його можливостей. Одразу доступна кнопка заклику до дії, яка забезпечує зручний перехід до функціоналу.

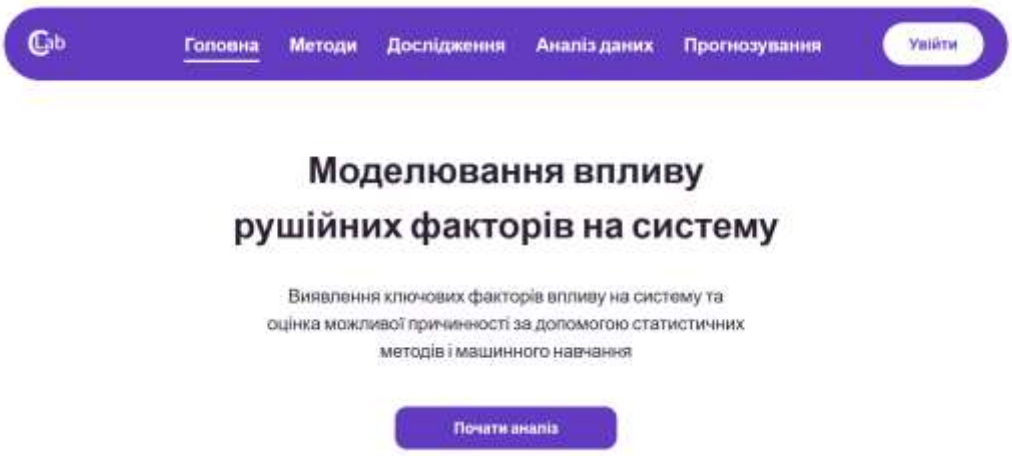


Рисунок 4.1 – Головна сторінка вебсервісу

На сторінці «Дослідження» доступні приклади проведених аналізів у сферах екології та економіки. Ці приклади доступні без потреби входу до системи, що дозволяє швидко ознайомитися з функціональністю платформи.

На рисунку 4.2 подано одне з досліджень, яке ілюструє вплив бойових дій на якість повітря у західному регіоні України.

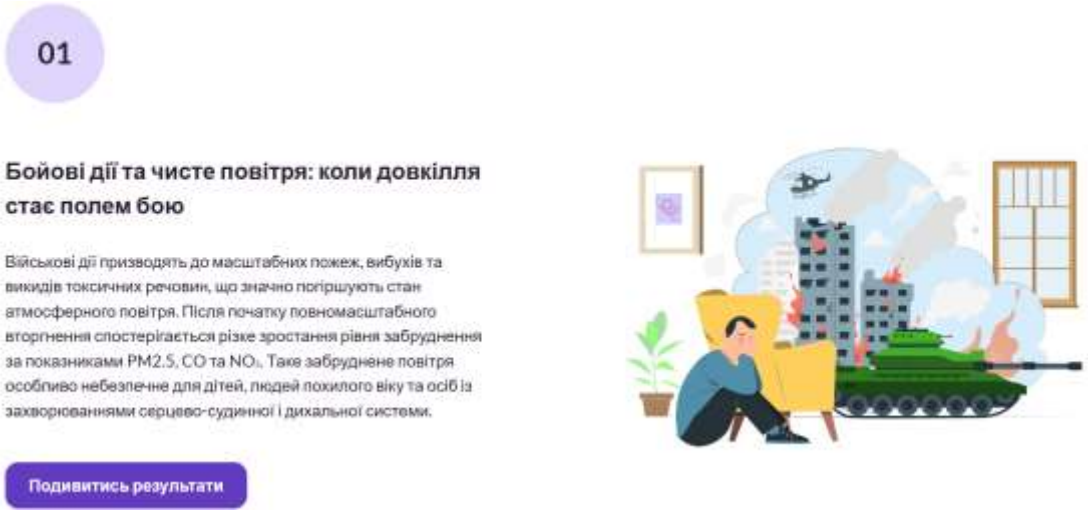


Рисунок 4.2 – Дослідження впливу бойових дій на якість повітря

Натиснувши на кнопку «Подивитися результати», користувач отримує візуалізацію у вигляді спрямованого ациклічного графа, на якому відображено напрямок та оцінку впливу змінної (рисунок 4.3).

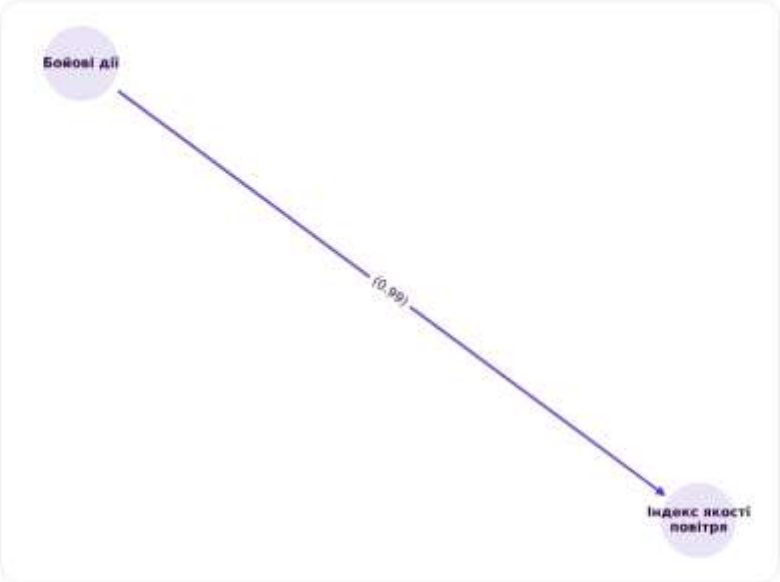


Рисунок 4.3 – Результати аналізу з екологічної сфери

Оцінка становить 0,99 з 1, що свідчить про сильну залежність між цими факторами у межах заданого набору. Проте, важливо враховувати, що модель аналізує лише ті змінні, які явно задані у вхідних даних. Для більш об’єктивної оцінки варто доповнити систему іншими ймовірними забруднювальними факторами, які також впливають на атмосферне повітря. Це дасть змогу відокремити загальний рівень забруднення від того, що безпосередньо пов’язаний із воєнними подіями. Такий підхід дозволить точніше побачити вплив кожного фактору і не перебільшити значення жодного з них.

Часова затримка між початком повномасштабного вторгнення в Україні та виявленим погіршенням якості повітря становить 3-6 періодів, що відповідає

приблизно 1-2 місяці з періодичністю даних у 10 днів у вхідному файлі (рисунок 4.4).

	A	B	C
1	Дата	Бойові дії	Індекс якості повітря
2	18.03.2020	0	1,9
3	28.03.2020	0	11,6
4	07.04.2020	0	9,48
5	17.04.2020	0	1
6	27.04.2020	0	2,6
7	07.05.2020	0	21,48

Рисунок 4.4 – Вхідні дані аналізу впливу бойових дій на якість повітря в Україні

Така затримка свідчить не про миттєвий, а про відтермінований ефект, який з часом накопичується та проявляється у зміні цільового показника. Західна Україна вважається рекреаційною зоною з чистим повітрям, невисоким рівнем промислового забруднення і відносно стабільною екологічною ситуацією. Однак, після початку активних бойових дій ситуація змінилася. Унаслідок масового переселення внутрішньо переміщених осіб суттєво зросло транспортне навантаження, споживання енергії та експлуатація інфраструктури.

Крім того, регулярні обстріли об’єктів критичної інфраструктури, включаючи склади, нафтобази та електростанції, могли спричинити викиди продуктів горіння та хімічних сполук. У сукупності це призвело до поступового зростання концентрації шкідливих речовин у повітрі: тверді частинки PM2.5, оксиди азоту та чадний газ. Саме цей ефект і фіксується аналізом – забруднення не виникає миттєво, а накопичується, досягаючи пікових значень через кілька тижнів після активізації бойових дій.

Для отримання повного доступу до функціональних можливостей вебсервісу користувач повинен авторизуватися в системі (рисунок 4.5). У випадку, якщо облікового запису ще не створено, передбачена зручна форма реєстрації з

обов’язковим введенням електронної адреси та пароля. Для забезпечення безперервного доступу до системи реалізовано функцію відновлення паролю – у разі його втрати користувач може надіслати запит на скидання через електронну пошту, що підвищує зручність та безпеку роботи із застосунком.

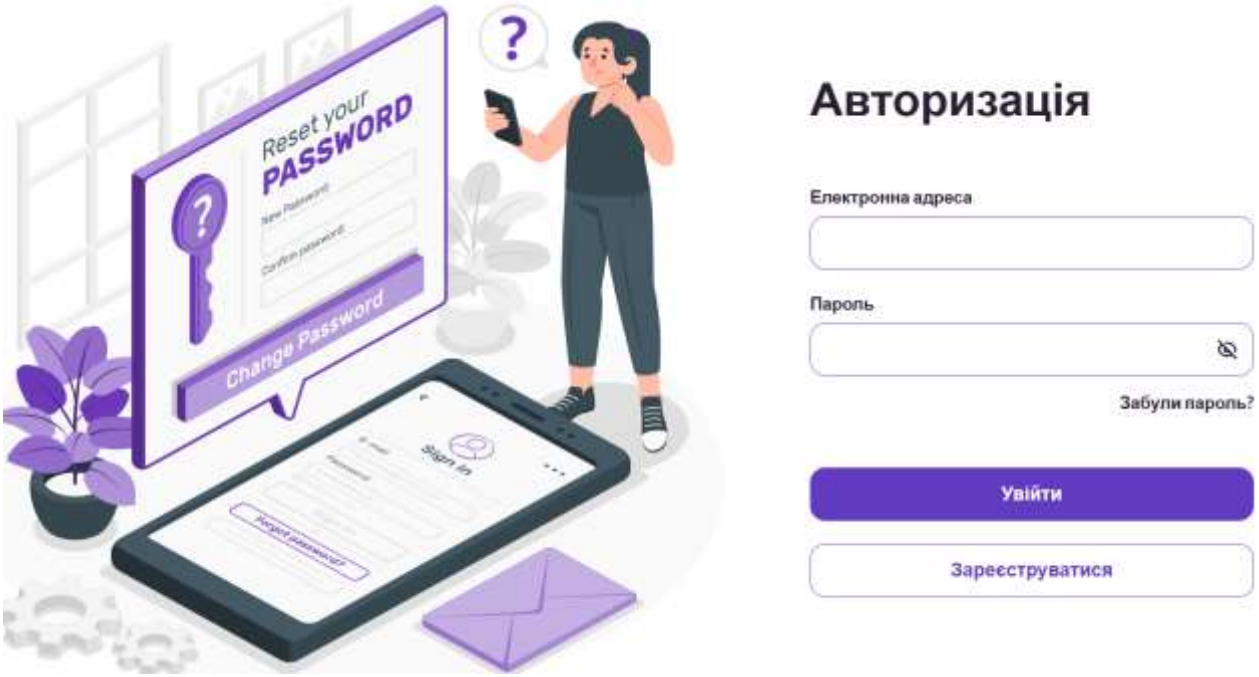


Рисунок 4.5 – Сторінка авторизації користувача з можливістю входу, реєстрації та відновлення паролю

Після входу відкривається доступ до сторінки «Аналіз даних», яка дозволяє виконувати повноцінний причинно-наслідковий аналіз на основі завантажених файлів. Перед початком рекомендовано ознайомитись з інструкцією, де вказано вимоги до файлу, що є важливим для коректного визначення причинності. Зокрема, дані повинні бути у форматі .xlsx і містити щонайменше три колонки: дати спостережень із рівномірним часовим проміжком, фактори впливу та цільовий показник. Користувач завантажує файл та вказує рівень ймовірності, при якому відхилення вважаються значущими (рисунок 4.6). Рекомендоване значення 0,05, що відповідає 5%, оскільки саме на цьому рівні зазвичай проводиться баланс між чутливістю та надійністю. Проте користувач може самостійно обрати інше

значення залежно від потреб дослідження. Варто зазначити, що якщо рівень ймовірності встановити занадто малим, наприклад, 0,01, існує ризик не виявити наявний причинно-наслідковий зв’язок. Натомість, занадто велике значення, наприклад, 0,1 і більше, може призвести до хибних позитивних висновків: тобто виявити зв’язок там, де його насправді немає.

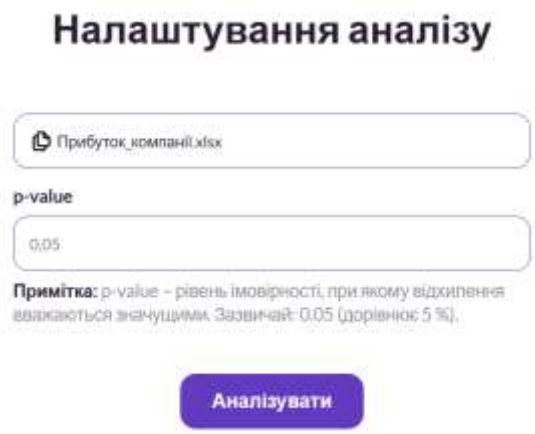


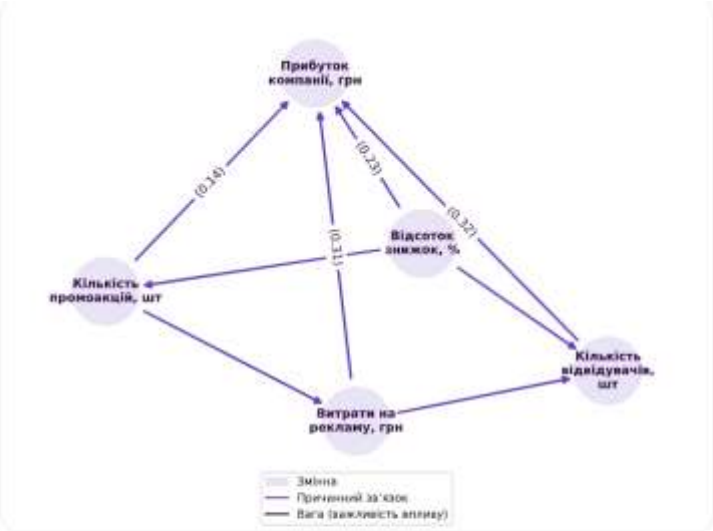
Рисунок 4.6 – Форма налаштування параметрів аналізу з прикріпленням файлу та введенням рівня статистичної значущості

Система обробляє завантажені дані та формує результат у вигляді спрямованого графа, доповненого пояснювальним текстом. На рисунку 4.7 подано приклад аналізу впливу економічних рушійних факторів на прибуток компанії.

За результатами проведеного аналізу рушійними факторами впливу є «Кількість відвідувачів», «Витрати на рекламу», «Відсоток знижок» та «Кількість промоакцій». Найбільший вплив має фактор «Кількість відвідувачів», що є логічним, оскільки саме від покупців залежить дохід. Однак, система дозволяє виявити не лише прямий, а й опосередкований вплив.

Наприклад, «Витрати на рекламу» безпосередньо зменшують прибуток через витрати бюджету компанії. Однак, у довгостроковій перспективі реклама відіграє роль інвестиції в пізнаваності бренду. Чим активніше компанія заявляє про себе на ринку, тим більшою є ймовірність того, що потенційні клієнти звернуться саме до неї. Підвищення рівня довіри, формування позитивного іміджу та постійна

присутність в інформаційному просторі сприяють зростанню кількості відвідувачів, що потенційно підвищує прибуток. Система демонструє, як непрямий вплив одного фактора через інший формує складну динаміку. Крім того, модель виявила часову затримку в межах 1-3 періодів між впливом факторів та зміною прибутку. Це означає, що ефект проявляється не миттєво, а через певний проміжок часу. Такий підхід допомагає краще зрозуміти, як різні дії компанії можуть впливати на результат та як вони взаємопов'язані в часі.



За результатами проведеного аналізу із застосуванням тесту Грейнджера та моделі Random Forest при заданому рівні ймовірності 0,05 (5%), при якому відхилення вважаються значущими, встановлено, що рушійними факторами, які впливають на "Прибуток компанії, грн" є "Витрати на рекламу, грн", "Відсоток знижок, %", "Кількість промоакцій, шт" та "Кількість відвідувачів, шт". Причинно-наслідковий зв'язок спостерігається із затримкою у 1–3 днів, що відповідає короткому періоду. Це вказує на те, що система реагує на зміну факторів протягом цього

Рисунок 4.7 – Результат аналізу впливу рушійних факторів на прибуток компанії

Після завершення аналізу причинно-наслідкових зв'язків користувач може перейти до наступного етапу – побудови прогнозу цільового показника на декілька періодів уперед. Це дозволяє не лише зрозуміти які фактори впливають на систему, а й оцінити очікувану динаміку цільового параметра в майбутньому. Для цього створено вкладку «Прогнозування».

Перед початком роботи користувачу рекомендовано ознайомитися з інструкцією для підготовки даних. Файл має бути у форматі .xlsx і містити щонайменше два стовпці: дату з рівномірними інтервалами спостережень та числове значення цільового показник, який необхідно передбачити. Якщо у файлі більше двох колонок, система автоматично визначить останній стовпець як цільовий. Важливо, щоб був достатній обсяг даних. Чим більше статистичних показників, тим точніше модель розпізнає закономірності та тенденції.

Перед початком прогнозування користувач має вказати кількість періодів, на яку слід сформувати прогноз. На рисунку 4.8 продемонстровано форму для введення цього параметра на прикладі аналізу прибутку компанії. Це дозволяє побачити, як у найближчому часі може змінитися ключовий показник ефективності бізнесу.

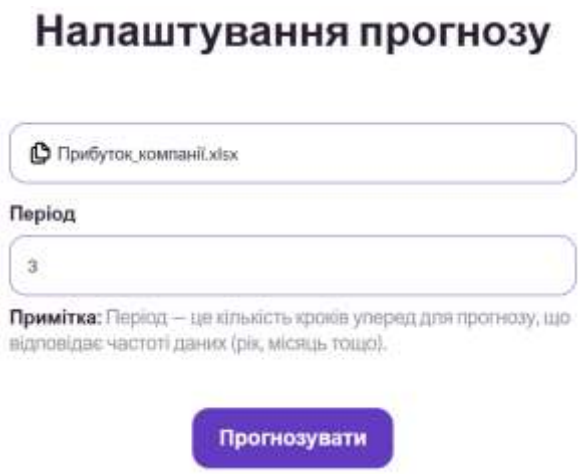
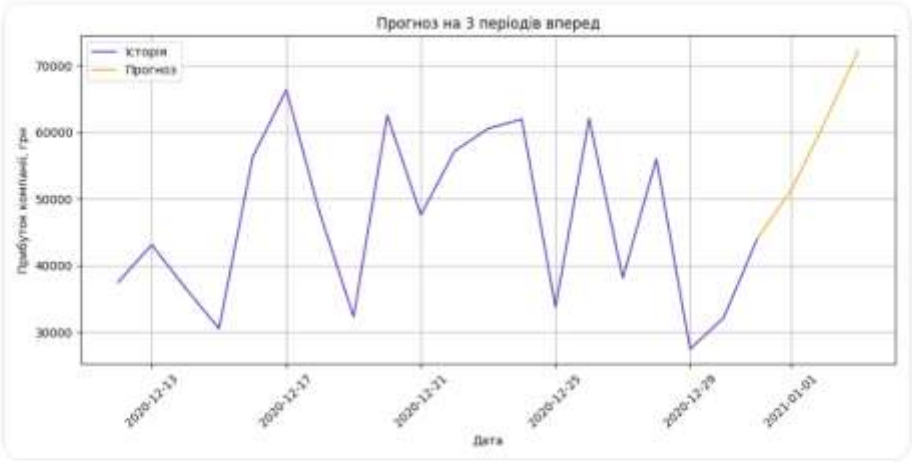


Рисунок 4.8 – Форма для налаштування прогнозування

У результаті виводиться графік із частиною фактичних даних та прогнозом цільового показника (рисунок 4.9). Для зручності сприйняття, історія та прогноз позначені різними кольорами. Це дозволяє швидко зрозуміти як змінюється динаміка та оцінити очікуваний напрям зміни – зростання, спадання або стабільність. Нижче наведено текстове пояснення, де описано характер змін: наскільки сильною є тенденція, у яких межах змінюється прогноз і наскільки він відрізняється від поточного значення.

Останнє зафіксоване значення прибутку становить 44042 грн. За прогнозом моделі LSTM, у найближчі 3 періоди прибуток може зрости до значень у діапазоні від 51256 до 72266 грн. Це означає, що очікується зростання приблизно на 64% у порівнянні з поточним рівнем. Такий прогноз може стати корисним для оцінки фінансової ситуації, щоб, наприклад, заздалегідь спланувати витрати чи прийняти інші важливі рішення.



У результаті прогнозування, виконаного за допомогою моделі нейронної мережі LSTM на основі наданих статистичних даних, сформовано прогноз на 3 періодів уперед. Згідно з графіком, останнє фактичне значення цільового показника становило 44042.00, а прогнозовані значення варіюються в межах від 51256.14 до 72266.47. Упродовж прогнозованого періоду показник демонструє зростання приблизно на 64.1% порівняно з поточним рівнем, що може вказувати на суттєву динаміку у вибраному інтервалі.

Рисунок 4.9 – Результати прогнозування цільового показника на 3 періоди

Оскільки прогноз формується на основі історичних даних і роботи автоматичної моделі, його результати не гарантують 100% точності. На майбутні значення можуть впливати непередбачувані зміни, зовнішні фактори або ситуації, які не враховані у вхідних даних. Саме тому після побудови прогнозу користувач отримує відповідне попередження (рисунок 4.10). Їх варто сприймати як орієнтовані підказки, які допоможуть у прийнятті рішень. Проте перед цим бажано провести власний аналіз або порадитися з фахівцем у відповідній галузі. Прогноз – це орієнтовний сценарій, сформований на основі наявної динаміки, який дає змогу заздалегідь оцінити можливі напрями змін і краще підготуватися до майбутнього.

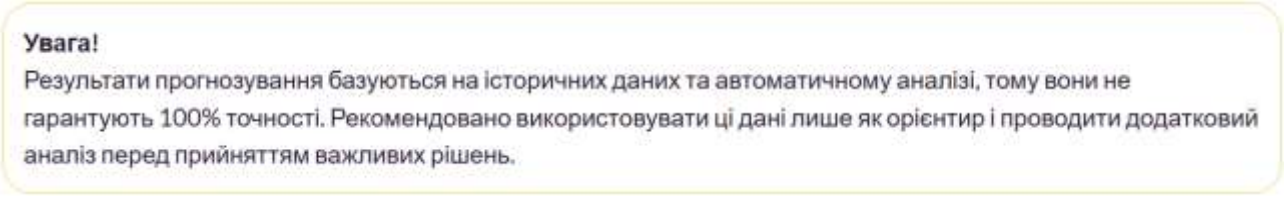


Рисунок 4.10 – Попередження щодо обмежень прогнозування

За бажанням, користувач може завантажити окремі PDF-звіти за результатами проведених аналізів. Їх можна буде використовувати для презентацій, передавати керівництву або колегам, а також додавати до звітної документації або порівнювати з подальшими результатами.

Взаємодія користувача з програмною системою спрямована на отримання корисних даних. Сервіс виконує не лише технічний аналіз причинно-наслідкових зв’язків і прогнозування, а й допомагає користувачу глибше осмислити отримані дані, виявити закономірності та зробити обґрунтовані висновки. Завдяки візуалізації, текстовим поясненням та можливістю збереження результатів у форматі звіту, система підтримує процес прийняття рішень і сприяє підвищенню якості аналітики у різних сферах застосування.

ВИСНОВОК

У результаті виконання дипломної роботи було розроблено вебсервіс, який дозволяє виявляти рушійні фактори в системі на основі часових рядів, оцінювати їх вплив на цільовий показник та формувати прогноз на основі історичних даних.

У процесі роботи було розглянуто сучасні методи аналізу динамічних систем і обґрунтовано вибір трьох основних підходів: тест Грейнджера для виявлення причинно-наслідкових зв’язків, Random Forest – для оцінки важливості факторів та модель LSTM – для прогнозування цільового показника. Поєднання цих методів дозволяє глибоко проаналізувати систему, виявити, що саме на неї впливає, і спрогнозувати її подальшу поведінку.

На основі проведеного аналізу було обґрунтовано використання Python як основної мови програмування, Flask – для серверної частини, SQLite – як бази даних, HTML, CSS, JavaScript і Jinja – для створення інтерфейсу та бібліотек pandas, statsmodels, scikit-learn, matplotlib TensorFlow/Keras – для реалізації функціональних модулів.

Реалізовано програмну систему «Моделювання впливу рушійних факторів на систему», що забезпечує повний цикл взаємодії з користувачем – від завантаження вхідних даних до отримання результату у формі графіків, пояснення та таблиць з можливістю експорту у формат PDF. Система поєднує інструменти статистичного аналізу та прогнозування, перетворюючи складні обчислення на зрозумілі та практичні аналітичні висновки.

Проведене тестування підтвердило коректність роботи програмного продукту. Система стабільно працює з різними наборами даних, коректно обробляє структуру таблиць, визначає статистично значущі зв’язки та будує прогноз із високим рівнем точності. Аналітичні звіти та графіки відображають основні залежності та тренди, що свідчить про правильну реалізацію математичних алгоритмів і надійність реалізованого функціоналу. Результати аналізу представлені в зручному для користувача форматі та можуть бути ефективно використані як інструмент підтримки прийняття рішень у практичній діяльності.

Таким чином, розроблена система об’єднує методи аналізу причинно-наслідкових зв’язків та прогнозування часових рядів у єдиному аналітичному середовищі. Вона дозволяє користувачеві не лише виявити вплив рушійних факторів, а й отримати пояснення результатів у доступному форматі. Завдяки використанню сучасних алгоритмів та чіткій логіці обробки даних, система є універсальним інструментом, який можна застосовувати як у дослідницьких цілях, так і для розв’язання прикладних задач у бізнесі чи управлінні.

У майбутньому можливе розширення функціоналу системи за рахунок впровадження багатофакторного прогнозування з урахуванням зовнішніх змінних, інтеграції з відкритими API для автоматичного оновлення даних, а також хмарного розгортання для забезпечення повного доступу без потреби локального запуску. Це дозволить зробити систему більш гнучкою, масштабованою та зручною у використанні для ширшого кола користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Big data statistics: How much data is there in the world? Rinery: вебсайт. URL: https://rivery.io/blog/big-data-statistics-how-much-data-is-there-in-the-world/ (дата звернення: 12.05.2025).

2. Беспала О.М., Тимкова А.В. Моделювання впливу рушійного фактора на екологічні та економічні показники з використанням машинного навчання. *Сучасні проблеми наукового забезпечення енергетики*: матеріали XXII Міжнар. наук.-практ. конф. молод. вчен. і студ., м. Київ, 22-25 квіт. 2025. Київ: Політехніка, 2025. С. 178-180.

3. Causal Inference Methods: Understanding Cause and Effect Relationships in Data Analysis. Netguru: вебсайт. URL: https://www.netguru.com/blog/causal-inference (дата звернення: 12.05.2025).

4. David Docquier, Giorgia Di Capua, Reik V. Donner, Carlos A. L. Pires, Amélie Simon, and Stéphane Vannitsem A comparison of two causal methods in the context of climate analyses. *Nonlin. Processes Geophys*. 2024. Вип. 1. С. 115-136.

5. Norbert Wiener. The theory of prediction. Modern Mathematics for Engineers. 1956. Вип. 1. С. 125-139.

6. Granger C. W. J. Investigating Causal Relations by Econometric Models and Cross-Spectral Methods. *Econometrica*. 1969. Вип. 37. № 3. С. 424–438.

7. Thomas Schreiber. Measuring information transfer. *Physical review letters*. 2000. Вип. 85. № 2. С. 461-464.

8. Aditi Kathpalia, Nithin Nagaraj. Data-based intervention approach for Complexity-Causality measure. *PeerJ Computer Science*. Вип. 5, 2019. С. 196-216.

9. ARIMA vs SARIMA Model. Geeksforgeeks: вебсайт. URL: https://www.geeksforgeeks.org/arima-vs-sarima-model/ (дата звернення: 14.04.2025).

10. Lütkepohl H. New Introduction to Multiple Time Series Analysis. Berlin: Springer, 2005. 764 с.

11. Hochreiter S., Schmidhuber J. Long Short-Term Memory**.** *Neural Computation*.1997. Вип. 9. № 8. С. 1735–1780.

12. Python Packages for Studying Causal Learning. Non-Brand Data: вебсайт. URL: https://www.nb-data.com/p/python-packages-for-studying-causal (дата звернення: 20.04.2025).

13. Tetrad. CCD Docs: вебсайт. URL: https://bd2kccd.github.io/docs/tetrad/ (дата звернення: 15.04.2025).

14. Actable AI. URL: https://docs.actable.ai/ (дата звернення: 15.04.2025).

15. David A. Dickey, Wayne A. Fuller. Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association*. 1979. Вип. 366. С. 427-431.

16. Arijit Chakrabarti, Jayanta K. Ghosh. AIC, BIC and Recent Advances in Model Selection. *Philosophy of Statistics*. 2011. Вип. 7. С. 583-605.

17. Ali Shojaie, Emily B. Fox. Granger Causality: A Review and Recent Advances. Annual Review of Statistics and Its Application. 2022. Вип. 9. С. 289-319.

18. Breiman L. Random forests. *Machine Learning*. 2001. Вип. 45. С. 5–32.

19. Feature importance evaluation. Scikit-learn: документація. URL: https://scikit-learn.org/stable/modules/ensemble.html#feature-importance-evaluation (дата звернення: 21.04.2025).

20. Sepp Hochreiter, Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*. 1997. Вип. 8. С. 1735-1780.

21. The Ultimate Guide to Building Your Own LSTM Models. ProjectPro: вебсайт. URL: https://www.projectpro.io/article/lstm-model/832 (дата звернення: 21.04.2025).

22. Flask. Palletsprojects: вебсайт. URL: https://flask.palletsprojects.com/en/stable/ (дата звернення: 18.04.2025).

23. About SQLite. SQLite: документація. URL: https://www.sqlite.org/about.html (дата звернення: 23.04.2025).

24. Virtual Environments and Packages. Python: документація. URL: https://docs.python.org/3/tutorial/venv.html (дата звернення: 18.04.2025).

ДОДАТОК А

Апробація
Матеріали статті до журналу «Наукові записки Державного університету інформаційно-комунікаційних технологій»

«Оптимізація прогнозування шляхом урахування причинного впливу»

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ІАТЕ_ЦТЕ_ТР-15

Аркушів 6

Київ — 2025

УДК 004.62
DOI:
О.М. Беспала,
А.В. Тимкова
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

ОПТИМІЗАЦІЯ ПРОГНОЗУВАННЯ ШЛЯХОМ УРАХУВАННЯ ПРИЧИННОГО ВПЛИВУ

OPTIMIZATION OF FORECASTING THROUGH INCORPORATION OF CAUSAL INFLUENCE

Беспала О.М., Тимкова А.В. Оптимізація прогнозування шляхом урахування причинного впливу. У статті досліджено підходи до врахування причинного впливу між змінними у задачах прогнозування. Основна увага приділяється методу причинності Грейнджера для виявлення часових затримок між змінними та їх подальшого застосування в побудові прогнозних моделей. Запропоновано підхід до покращення точності LSTM-моделі шляхом інтеграції аналізу часових затримок. Автори наголошують на важливості врахування лагових взаємозв'язків для формування більш інформативного вхідного простору нейромережі. Експериментальні результати свідчать про підвищення точності прогнозів за умови попереднього виявлення причинно-значущих зв'язків.
Ключові слова: прогнозування, часові ряди, причинність Грейнджера, лаговий вплив, нейронні мережі.

Bespala O.M., Tymkova A.V. Optimization of forecasting through incorporation of causal influence. The article explores approaches to improving forecasting accuracy by incorporating causal influence between variables with time delays. The study uses Granger causality to detect temporal lags and adjust input variables accordingly. A method is proposed to enhance LSTM-model performance by integrating lag analysis into the training pipeline. Experimental evaluation shows that accounting for different lag times across variables significantly reduces forecasting error. The results confirm that causal-lag modeling improves robustness and reflects real-world dynamics in time series forecasting. The authors emphasize the importance of considering lagged interrelationships to form a more informative input space for the neural network. Experimental results indicate an improvement in forecasting accuracy when causally significant relationships are identified in advance.
Keywords: forecasting, time series, Granger causality, lag effect, prediction model, neural network.

Вступ. Одним із головних факторів вивчення та моніторингу складних систем є ефективний аналіз на основі даних спостережень. Багато таких інформаційних систем включають часові дані. Наприклад, на основі часових даних фінансові інформаційні системи (ІС) проводять аналіз фінансових ринків, прогнозують ціни акцій, валют і товарів, проводять обробку історичних даних фондового ринку [1]; енергетичні ІС, в тому числі і в режимі реального часу, здійснюють моніторинг та управління електромережами, аналізують їх навантаження та прогнозування споживання електроенергії, а IoT-платформи для "розумних" лічильників відстежують споживання електроенергії, води, газу [2, 3]; метеорологічні системи збирають і обробляють супутникові дані великих розмірностей [4, 5]; медичні ІС здійснюють моніторинг пацієнтів у реальному часі та аналіз серцевого ритму, тиску, температури, а з використанням бази даних епідеміологічного моніторингу проводиться аналіз і прогноз поширення хвороб; в області соціальних мереж та веб-аналітика особлива потреба зростає в аналізі динаміки пошукових запитів, переглядів та поведінки користувачів, а також популярності трендів та новин [6] тощо.

Оскільки спостерігається збільшення об'єму зібраних даних, в тому числі і часових, піднімається проблема вдосконалення методів використання часової інформації. У свою чергу, для більш ефективного аналізу системи виникає необхідність розуміння взаємодії між її компонентами та їхнього впливу як між собою, так і на загальну поведінку системи. Це критично важливо для прийняття рішень, оптимізації роботи системи, прогнозування її розвитку та управління ризиками.

Постановка завдання. Завдання полягає у вдосконаленні прогнозування за допомогою врахування часових лагів між змінними, що впливають на цільову функцію. Використовується метод причинності Грейнджера для виявлення та визначення цих лагів, що дозволяє з'ясувати, як зміни в одних змінних впливають на інші в певний момент часу. Після виявлення лагів, їх слід застосувати для зміщення та коригування змінних з метою підвищення точності прогнозу.

Аналіз останніх досліджень і публікацій. В огляді літератури представлено різні методи моделювання причинно-наслідкових зв'язків [7], які можна поділити на методи, що ґрунтуються на знаннях та на основі даних [8, 9]. Методи, що ґрунтуються на побудові моделі на основі знань, можуть мати перевагу в точності, проте вимагають глибокого розуміння процесу, заснованих на досвіді та фізичних моделях систем [10]. Оскільки сучасні системні архітектури стають складнішими, а кількість компонентів значно збільшується, процес формулювання точної моделі на основі досвіду та знань предметної області стає дуже складним, що означає, що більшість методів, заснованих на знаннях, мають деякі обмеження при застосуванні у великому масштабі. Саме тому, використання доступних даних для побудови моделей шляхом аналізу причинно-наслідкових зв'язків між змінними процесу стає більш актуальною задачею [11, 12], яка потребує системного підходу з можливістю працювати з часовими даними та обробляти одночасно великі об'єми інформації з часовою структурою.

Мета роботи. Метою роботи є вдосконалення методу прогнозування за рахунок застосування методу Грейнджера для виявлення часових лагів між змінними, що впливають на цільову функцію. Задача полягає в тому, щоб знайти ці лаги, скоригувати змінні на їх основі та покращити точність прогностичної моделі, враховуючи причинно-наслідкові зв'язки між змінними.

Виклад основного матеріалу дослідження. У задачах прогнозування в контексті машинного навчання мета полягає у створенні моделі, яка здатна передбачити значення цільової змінної на основі наданих вхідних даних. Задача прогнозування є типовою для багатьох дисциплін, таких як економіка, фінанси, медицина, а також природничі науки, де часто необхідно передбачити значення, які залежать від багатьох змінних.

Нехай у нас є набір спостережень, де кожне спостереження складається з вектору вхідних змінних $x^i \in \mathbb{R}^m$, що представляють ознаки, і відповідного значення цільової змінної $y^i \in \mathbb{R}$. Таким чином, ми маємо набір даних:

$$\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}, \quad (1)$$

де n — кількість спостережень. Задача прогнозування полягає у знаходженні функції $f(x, w)$, яка мапує вхідні дані x на прогнозоване значення y, де w — вектор параметрів, які визначають модель.

Математично це можна записати як:

$$\hat{y} = f(x, w), \quad (2)$$

де $f(x, w)$ – функція, яка залежить від вхідного вектору x та параметрів w, w – вектор параметрів, які потрібно навчити (ваги).

При цьому важливо, щоб ця функція була якомога точнішою для всіх спостережень, що відображає залежність між змінними. Для знаходження оптимальних значень параметрів w необхідно мінімізувати функцію втрат, яка вимірює точність моделі. Одним із найбільш використовуваних підходів є мінімізація середньоквадратичної помилки (MSE), яка визначається як середнє значення квадратів відмінностей між фактичними значеннями y_i та прогнозами $\hat{y}_i$:

$$L(w) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f\left(x_i, w\right)\right)^2 \quad (3)$$

Функція втрат виражає відхилення прогнозів від реальних значень і служить основою для оцінки ефективності моделі. Метою оптимізації є знаходження таких значень параметрів $w*$, які мінімізують функцію втрат:

$$w* = \arg\min_m L(w) \quad (4)$$

Наприклад модель знайшла ваги, але відбулась затримка впливу на цільову змінну від одного з параметрів *x*. Це може значно погіршити точність прогнозування моделі. У такому випадку потрібно додатково відслідковувати зміщення в часі, та коригувати зміщення ваг *w*.

Задача прогнозування в машинному навчанні є надзвичайно важливою для численних застосувань, але у випадку, коли вплив вхідних змінних (наприклад, витрат на рекламу) на цільову змінну (наприклад, обсяг продажів) має різну віддаленість у часі (лаг), традиційні моделі, як лінійна регресія або інші моделі без врахування часу, можуть погано працювати. Це призводить до змінених результатів прогнозування та погіршення точності моделі.

Проблема різної віддаленості впливу змінних. У багатьох реальних задачах, як, наприклад, прогнозування обсягів продажів на основі витрат на рекламу, різні рекламні кампанії можуть мати різні часові лаги впливу. Це означає, що не всі змінні впливають на цільову змінну негайно, і їхній ефект може бути відстрочений на певний період часу. Якщо є кілька змінних, що мають різний час впливу на результат, то математична модель без врахування лагів може виглядати так:

$$y_i = f\left(x_{i,1}, x_{i,2}, ..., x_{i,m}, w\right) + \varepsilon_i \quad (5)$$

де y_i – це цільова змінна (наприклад, обсяг продажів), $x_{i,j}$ – це вхідні змінні, що представляють різні фактори (наприклад, витрати на рекламу в поточному місяці, витрати на рекламу в попередньому місяці), *w* – вектор параметрів моделі, а ε_i – це похибка.

Змішування різних часових лагів в одній моделі може привести до перехресних кореляцій між змінними, що може погіршити здатність моделі правильно оцінювати залежність між змінними.

Вплив різних лагів на функцію втрат. Коли ми маємо кілька змінних з різними лагами, традиційна модель може мати змогу адаптуватися лише до певного виду впливу. Для кожного лагу потрібно розглядати окрему залежність, що можна записати як:

$$y_i = w_0 + w_1 x_{i,1} + w_2 x_{i,2} + ... + w_m x_{i,m} + \varepsilon_i, \quad (6)$$

де кожен $x_{i,j}$ – це значення змінної, враховуючи її лаг. Традиційні методи оптимізації, які враховують всі змінні однаково, можуть привести до зміщення в оцінці параметрів *w*, якщо лаги мають різні часи впливу.

Функція втрат (наприклад, середньоквадратична помилка) в даному випадку виглядатиме так:

$$L(w) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \left(w_0 + w_1 x_{i,1} + w_2 x_{i,2} + ... + w_m x_{i,m}\right)\right)^2 \quad (7)$$

Проте, якщо зміщення в часі (лаги) не враховані коректно, то точність прогнозу зменшується, оскільки різні лаги не будуть коректно враховані в моделі.

Метод Грейнджера для визначення лагів причинного впливу. Одним із способів вирішення цієї проблеми є використання методу причинності Грейнджера для виявлення і коригування лагів в часі між вхідними змінними та цільовою змінною [13]. Метод причинності Грейнджера дозволяє встановити, чи може змінна x_t передбачати значення змінної y_t в майбутньому на основі попередніх значень цих змінних.

Тест на причинність Грейнджера визначається як перевірка на наявність статистично значущої кореляції між ланцюговими значеннями двох часових рядів. Для двох змінних x_t та y_t тест Грейнджера виглядає так:

$$y_t = \alpha + \sum_{i=1}^{p} \beta_i y_{t-i} + \sum_{j=1}^{q} \gamma_i x_{t-j} + \varepsilon_t \,, \tag{8}$$

де p і q — це максимальні лаги для y і x відповідно, α — константа, β_i та y_i – коефіцієнти, що визначають вплив минулих значень y і x на поточне значення y_t.

Якщо $\gamma_i \neq 0$, то змінна x_t має причинний вплив на y_t в часі з лагом j.

Якщо $\gamma_i = 0$ для всіх j, то змінна x_t не має причинного впливу на y_t.

Інтеграція цієї ідеї в модель прогнозування може відбуватись шляхом введення множини лагів для кожної змінної x_t. Таким чином, ми можемо створити модель, яка враховує різні лаги для різних змінних. Математична форма цієї моделі виглядатиме так:

$$y_t = w_0 + \sum_{i=1}^{p} w_i x_{t-i} + \varepsilon_t \,, \tag{9}$$

де x_{t-i} – це значення змінних на попередніх кроках часу (лаги), а w_i – ваги, які визначають вплив цих значень на прогноз.

У межах тестування запропонованого підходу було використано архітектуру спеціального типу нейронної мережі з довготривалою короткочасною пам’яттю (LSTM, Long Short-Term Memory), яка є однією з найефективніших моделей для обробки послідовних даних та часових рядів. Початкове навчання моделі проводилось на синтетичних даних, де всі змінні мали однакову часову віддаленість впливу на цільову змінну. У такому випадку модель демонструвала високу точність прогнозування, що підтверджувалося низьким значенням функції втрат (наприклад, середньоквадратичної помилки MSE).

Після цього було згенеровано новий набір даних, де ступінь часової віддаленості (лагу) між змінними та цільовою змінною був неоднаковим. Зокрема, деякі вхідні змінні починали впливати на результат із затримкою у кілька кроків, у той час як інші – миттєво. У цьому сценарії якість прогнозування значно погіршилася, що проявилося у збільшенні функції втрат $L(w)$, незважаючи на наявність тієї ж самої архітектури LSTM. Це свідчить про те, що базова модель не враховує асиметрію лагових впливів між змінними.

Щоб вирішити цю проблему, до моделі було інтегровано модуль попереднього аналізу лагових впливів, який базується на методі причинності Грейнджера. Цей модуль дозволяє автоматично виявляти, з яким лагом кожна змінна x_j має причинний вплив на цільову змінну y, шляхом статистичного тестування гіпотез для кожної пари змінна-цільова:

H0: x_j не причинно впливає на y з лагом l

H1: x_j причинно впливає на y з лагом l

Для цього було побудовано множину лагових моделей:

$$y_t = \sum_{j=1}^{m} \sum_{l=1}^{L} a_{j,l} x_{j,t-l} + \varepsilon_t \,, \tag{10}$$

та проведено тестування значущості коефіцієнтів $a_{j,l}$. Якщо певна змінна x_j виявилася значущою з лагом l, то вона включалась у модель LSTM як окремий часовий зріз із відповідним зсувом.

Для тестування запропонованого підходу було здійснено генерацію даних:

$x(t)$ – витрати на рекламу в день t,

$u(t)$ – кількість відвідувачів у день t,

вплив реклами має затримку (лаг) на певний період, тобто $x(t)$ впливає на $u(t+1)$.

Після інтеграції модуля аналізу лагів модель була повторно протестована на тому ж наборі даних із затриманими впливами. У результаті було досягнуто суттєвого покращення якості прогнозу, що проявилося у зниженні функції втрат $L(w*)$ у порівнянні з початковою LSTM-моделлю. На рисунку 1, 2 показано результати тестування моделей. В таблиці 1 продемонстровано відповідні показниками оцінки якості прогнозованих моделей.

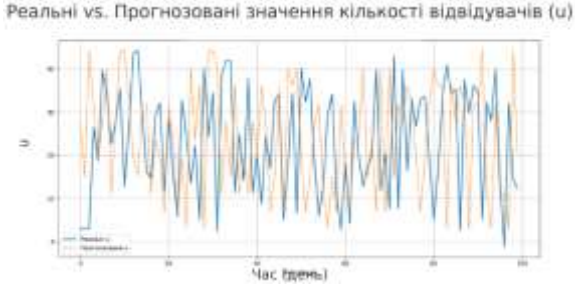


Рисунок 1 – Прогнозована модель 1 без врахування лагів

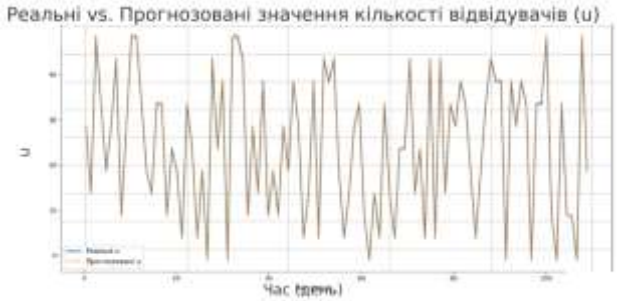


Рисунок 2 – Прогнозована модель 2 з урахуванням лагів

Таблиця 1. Результатів оцінки якості прогнозованих моделей

	MAE $Y = x(t)$	MAE $Y = x(t+i)$
Модель 1	0.1908	15.84
Модель 2	0.1901	0.1900

Таким чином, включення попереднього лагового аналізу дозволяє адаптувати модель до структурної часової неоднорідності вхідних змінних і підвищує її робастність до зміщення впливу.

Висновки і пропозиції. Використання методу Грейнджера для виявлення часових лагів дозволяє більш точно моделювати вплив змінних на цільову функцію, що значно підвищує прогностичну здатність моделей. Зміщення змінних із врахуванням виявлених лагів дає змогу знизити помилки прогнозування, оскільки враховуються затримки в їх взаємодії. Причинно-наслідковий підхід, поєднаний з аналізом лагів, дозволяє точніше оцінювати динаміку змін і здійснювати прогнози, що відображають реальні процеси. Покращення точності прогнозування за рахунок виявлення лагів є важливим кроком у вдосконаленні моделей, особливо в задачах, де часові аспекти мають критичне значення для результатів.

Список використаної літератури

1. Habibnia A., Etesami J., Kiyavash N. Modeling Systemic Risk: A Time-Varying Nonparametric Causal Inference Framework. SSRN. 2024. URL: https://doi.org/10.2139/ssrn.4684230 (дата звернення: 12.05.2025).

2. Research on condition operation monitoring of power system based on supervisory control and data acquisition model / B. Li et al. *Alexandria Engineering Journal*. 2024. Vol. 99. P. 326–334. URL: https://doi.org/10.1016/j.aej.2024.05.027 (дата звернення: 11.05.2025).

3. Open-Source Internet of Things-Based Supervisory Control and Data Acquisition System for Photovoltaic Monitoring and Control Using HTTP and TCP/IP Protocols / W. Khalid et al. *Energies*. 2024. Vol. 17, no. 16. P. 4083. URL: https://doi.org/10.3390/en17164083 (дата звернення: 11.05.2025).

4. Lopes F. M., Dutra E., Boussetta S. Evaluation of Daily Temperature Extremes in the ECMWF Operational Weather Forecasts and ERA5 Reanalysis. *Atmosphere*. 2024. Vol. 15, no. 1. P. 93. URL: https://doi.org/10.3390/atmos15010093 (дата звернення: 11.05.2025).

5. Siddiqi A. Value Analytics for Earth Observing Systems. *IGARSS 2024 - 2024 IEEE International Geoscience and Remote Sensing Symposium*, Athens, Greece, 7–12 July 2024. 2024. P. 3864–3867. URL: https://doi.org/10.1109/igarss53475.2024.10642411 (дата звернення: 11.05.2025).

6. Liao S.-h., Widowati R., Lee C.-Y. Data mining analytics investigation on TikTok users' behaviors: social media app development. *Library Hi Tech*. 2022. URL: https://doi.org/10.1108/lht-08-2022-0368 (дата звернення: 11.05.2025).

7. Chiang L. H., Braatz R. D. Process monitoring using causal map and multivariate statistics: fault detection and identification. *Chemometrics and Intelligent Laboratory Systems*. 2003. Vol. 65, no. 2. P. 159–178. URL: https://doi.org/10.1016/s0169-7439(02)00140-5 (дата звернення: 11.05.2025).

8. Nadim K., Ragab A., Ouali M.-S. Data-driven dynamic causality analysis of industrial systems using interpretable machine learning and process mining. *Journal of Intelligent Manufacturing*. 2022. URL: https://doi.org/10.1007/s10845-021-01903-y (дата звернення: 11.05.2025).

9. Беспала О.М. Інструментарій причинно-наслідкового висновку: огляд та перспективи. *Control Systems and Computers*. 2020. № 5 (289). С. 52–63. URL: https://doi.org/10.15407/csc.2020.05.052 (дата звернення: 12.05.2025).

10. Deep understanding in industrial processes by complementing human expertise with interpretable patterns of machine learning / A. Ragab et al. *Expert Systems with Applications*. 2019. Vol. 122. P. 388–405. URL: https://doi.org/10.1016/j.eswa.2019.01.011 (дата звернення: 11.05.2025).

11. Беспала О. М., Отрох С. І.. Ружинський В. Г.. Моделювання спрямованого ациклічного графа для причинного висновку. *Наукові записки Державного університету телекомунікацій.* 2023. № 2 (1) С. 87-95 DOI: 10.31673/2518-7678.2023.020202

12. Беспала О. М. Метод пошуку та оцінки впливу причинно-наслідкових зв'язків в системах прийняття рішень. *Вісник НТУ "ХПІ". Серія: Інформатика та моделювання.* 2020. № 2 (4). С. 59 – 72.

13. Granger C. W. J. Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica*. 1969. Vol. 37, no. 3. P. 424. URL: https://doi.org/10.2307/1912791 (дата звернення: 11.05.2025).

Автори статті

Беспала Ольга – старший викладач, кафедра цифрових технологій в енергетиці, Навчально-науковий інститут атомної та теплової енергетики, НТУУ "КПІ ім. Ігоря Сікорського", Київ, Україна, olya327@ukr.net; orcid.org: 0000-0003-3285-2585

Тимкова Анастасія – студентка на здобуття ступеня бакалавра, кафедра цифрових технологій в енергетиці, Навчально-науковий інститут атомної та теплової енергетики, НТУУ "КПІ ім. Ігоря Сікорського", Київ, Україна, anastasiatymkova@gmail.com; orcid.org: 0009-0000-7619-3281.

Authors of the article

Bespala Olha - Senior Lecturer, Department of Digital Technologies in Energy, Educational and Research Institute of Nuclear and Thermal Energy, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine, olya327@ukr.net; orcid.org: 0000-0003-3285-2585

Tymkova Anastasiia - Bachelor's degree student, Department of Digital Technologies in Energy, Educational and Research Institute of Nuclear and Thermal Energy, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine, anastasiatymkova@gmail.com; orcid.org: 0009-0000-7619-3281.

ДОДАТОК Б

Апробація
Матеріали XXII Міжнародної науково-практичної конференції молодих вчених і студентів «Сучасні проблеми наукового забезпечення енергетики»

«Моделювання впливу рушійного фактору на екологічні та економічні показники з використанням машинного навчання»

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ІАТЕ_ЦТЕ_ТР-15

Аркушів 3

Київ — 2025

УДК 004.6

[1] Бакалаврант 4 курсу Тимкова А.В.
[1] Старший викладач Беспала О.М.
https://scholar.google.com.ua/citations?user=gLUrLHUAAAAJ&hl=uk

[1] КПІ ім. Ігоря Сікорського

МОДЕЛЮВАННЯ ВПЛИВУ РУШІЙНОГО ФАКТОРУ НА ЕКОЛОГІЧНІ ТА ЕКОНОМІЧНІ ПОКАЗНИКИ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

Постановка проблеми та її актуальність. В Україні внаслідок багаторічного домінування ресурсо- та енергоємних галузей, сировинної орієнтації експорту та концентрації виробництва у промислових регіонах сформувалася неефективна економічна та екологічно небезпечна структура управління. До цих проблем додався вплив бойових дій, що руйнують інфраструктуру, виробничі потужності і природні екосистеми, що спричиняє додаткові економічні ризики, забруднене довкілля та деградацію природних ландшафтів [1].

Водночас стратегічне бачення сталого розвитку України, викладене у проєкті довгострокового розвитку країни, базується на політичних, економічних, соціальних, екологічних, моральних і культурних цінностях та передбачає проведення структурних реформ, екологічно невиснажливе зростання і створення сприятливих умов для господарської діяльності [2].

Досягнення сталого розвитку вимагає комплексного аналізу взаємодії ключових показників, що впливають на соціально-економічні, енергетичні та екологічні процеси, для визначення масштабу впливів і формування ефективних управлінських рішень. Сучасні методи машинного навчання, здатні аналізувати великі обсяги даних і моделювати багатофакторні залежності, відкривають можливості для точної оцінки впливу змінних, зокрема у часових тенденціях, що дозволяє виявити довготривалі взаємозв’язки.

Поєднання традиційних статистичних підходів із сучасними алгоритмами машинного навчання дозволить глибше зрозуміти механізми впливу та сприятиме формуванню обґрунтованої політики, що адаптує стратегії розвитку до сучасних викликів.

Аналіз останніх досліджень. Методи перевірки причинно-наслідкового зв’язку в часових рядах уперше були запропоновані Грейнджером. Подальші розробки підкреслюють необхідність забезпечення стаціонарності даних для коректного оцінювання моделей. У сфері машинного навчання алгоритми Random Forest та XGBoost [3] ефективно виявляють складні залежності завдяки ансамблевим підходам. Сучасні дослідження інтегрують причинно-наслідковий аналіз із методами машинного навчання для оцінки економічних та екологічних процесів, зокрема впливу бойових дій та зовнішніх шоків на макроекономічні показники. Адаптивні моделі дозволяють оперативно виявити зміни у динаміці показників, що свідчить про актуальність розробки гнучких аналітичних інструментів.

Формулювання мети. Метою дослідження є моделювання та прогнозування впливу рушійних факторів на систему, що спостерігаються за певний період часу.

Основна частина. Для досягнення поставленої мети, з відкритих електронних ресурсів України було зібрано вхідні набори статистичних даних, які містять спостережувані значення за певні періоди часу [4].

За результатами сучасних досліджень, бойові дії виступають як важливий чинник, що негативно впливає на екологічну систему, зокрема на якість повітря в Україні [5]. Руйнування інфраструктури, промислових об’єктів та транспортних мереж, що супроводжується масштабними пожежами, спричиняє збільшення обсягів шкідливих викидів, при цьому вплив проявляється з певним часовим лагом через поступове накопичення забруднюючих речовин. Паралельно, зростання інфляції слугує критичним економічним чинником, що формує монетарну політику, зокрема через регулювання ставок Національного Банку України, реакція на зміну яких

також має часові затримки. Для достовірного встановлення причинно-наслідкових зв'язків у даних системах необхідно застосовувати методи аналізу часових рядів, що дозволяють врахувати як затримки в екологічних, так і в економічних показниках, а також інтегрувати вплив додаткових зовнішніх чинників.

Одним із ефективних методів визначення того, чи має певний рушійний фактор причинний вплив на досліджуваний параметр, є тест Грейнджера. Його основна ідея полягає в оцінці того, наскільки добре минулі значення одного ряду (потенційної причини) допомагають прогнозувати значення іншого ряду (досліджуваного параметра) понад інформацію, яку дають його власні попередні значення. Суть методу полягає в порівнянні двох моделей: першою, де досліджуваний параметр залежить лише від своїх минулих значень, та другою, де до моделі додається можливий вплив іншого ряду, що дозволяє виявити додаткову предиктивну силу. Якщо при додаванні потенційного рушійного фактора (наприклад, інфляції) до моделі результату виявляються статистично значущі коефіцієнти, це вказує на те, що цей фактор має причинний вплив на досліджуваний параметр. Це означає, що зміни у значенні інфляції можуть передувати змінам у ставках Національного Банку України або інших економічних показниках.

Для коректного застосування методу всі часові ряди мають бути стаціонарними, тобто їх статистичні властивості не повинні змінюватися з часом. Тому перед застосуванням тесту Грейнджера кожен з рядів перевіряється на стаціонарність за допомогою критерію Дікі-Фуллера, і у разі необхідності, ряди приводяться до стаціонарного вигляду через різницювання, тощо.

Для покращення результатів було прийнято рішення застосувати попередньо виявлені впливи рушійного фактору, зокрема вплив інфляції на ставки Національного Банку України. Після цього було застосовано метод Random Forest для прогнозування значень. Використання цієї комбінації призвело до покращення результатів, зокрема, спостерігалося значне зменшення середньоквадратичної помилки порівняно з іншими методами. Це свідчить про підвищену точність прогнозування та ефективність вибраної стратегії, що включає інтеграцію впливу зовнішніх факторів і використання потужних алгоритмів машинного навчання.

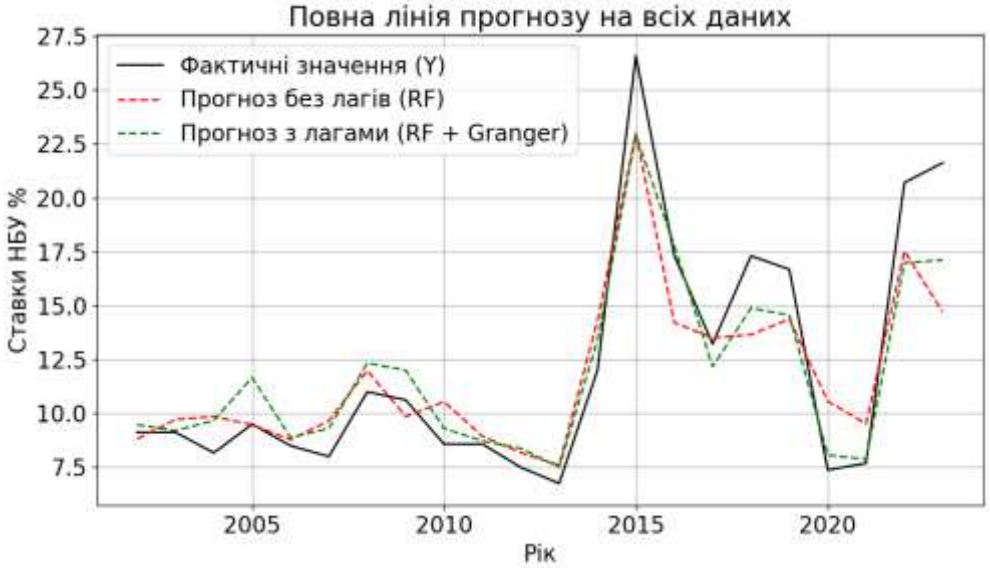


Рисунок 1 – Порівняння фактичних та прогнозованих ставок Національного Банку України

Під час застосування тесту Грейнджера було визначено оптимальну кількість лагів, за якої випереджувальний вплив інфляції на ставки Національного Банку України виявився статистично значущим (p-значення < 0,05). Це дозволяє стверджувати про наявність причинно-наслідкового зв'язку між інфляційними процесами та регулюванням ставок центрального банку. На рисунку 1 наведено порівняння фактичних значень ставок з прогнозами, побудованими за допомогою методу Random Forest: без урахувань часових зсувів та з урахуванням часових лагів, визначеним тестом Грейнджера. Спостерігається, що врахування часових лагів підвищує точність прогнозу, що додатково підтверджує випереджувальний характер інфляційних чинників у формуванні монетарної політики.

Одержані результати узгоджуються з висновками попередніх досліджень, які підтверджують важливість урахування часових зсувів для підвищення ефективності прогнозування. Водночас, при застосуванні аналогічного аналізу для оцінювання впливу бойових дій на якість повітря, було виявлено причинний зв'язок, проте його сила виявилася менш вираженою. Це свідчить про те, що окрім військової активності, на стан атмосфери впливають й інші істотні чинники. Отже, модель потребує подальшого уточнення та розширення переліку змінних аби підвищити точність оцінювання й прогнозування.

Висновки. Використання методу Грейнджера та алгоритмів машинного навчання доводить високу результативність під час аналізу часових рядів, що описують як екологічні так і економічні процеси. Тест Грейнджера допомагає визначити статистично значущі випереджувальні впливи між змінними, уточнюючи розуміння послідовності подій у динаміці даних. Моделі машинного навчання дають змогу ефективно працювати з великими обсягами інформації та виявляти складні закономірності, включно з часовими затримками. Результати таких досліджень мають практичне значення для ухвалення управлінських рішень у сферах екології та макроекономіки, адже точні прогнози дають змогу своєчасно вживати заходів щодо стабілізації економічної ситуації та покращення стану довкілля. Подальший розвиток у цій сфері передбачає розширення переліку чинників, інтеграцію додаткових джерел даних, застосування ще складніших моделей та міждисциплінарний підхід із залученням фахівців різних галузей. Такий комплексний аналіз дозволить підвищити адекватність прогнозів, гнучкість обраних стратегій і загальну ефективність регуляторних заходів.

Перелік посилань:

1. Olga T. Impact of hostilities on the environment of the northern region of Ukraine // Results - OpenURL Connection – EBSCO [Електронний ресурс]. – Режим доступу: https://openurl.ebsco.com/EPDB%3Agcd%3A9%3A19258409/detailv2?sid=ebsco%3Aplink%3Ascholar&id=ebsco%3Agcd%3A181837872&crl=c&link_origin=scholar.google.com.ua (дата звернення: 28.02.2025).

2. **Про стратегію сталого розвитку України до 2030 року: Закон України від 07.08.2018 № 9015** [Електронний ресурс]. – Режим доступу: https://ips.ligazakon.net/document/JH6YF00A?an=332 (дата звернення: 28.02.2025).

3. Wohlwend B. Decision Tree, Random Forest, and XGBoost: An Exploration into the Heart of Machine Learning // Medium [Електронний ресурс]. – Режим доступу: https://medium.com/@brandon93.w/decision-tree-random-forest-and-xgboost-an-exploration-into-the-heart-of-machine-learning-90dc212f4948 (дата звернення: 24.02.2025).

4. Облікова ставка НБУ (1992-2025) // Ставки, індекси, тарифи [Електронний ресурс]. – Режим доступу: https://index.minfin.com.ua/ua/banks/nbu/refinance/ (дата звернення: 26.02.2025).

5. Liu B., Qi Z., Gao L. Enhanced Air Quality Prediction through Spatio-temporal Feature Sxtraction and Fusion: A Self-tuning Hybrid Approach with GCN and GRU // Water, Air, & Soil Pollution. – 2024. – Vol. 235, № 8 [Електронний ресурс]. – Режим доступу: https://link.springer.com/article/10.1007/s11270-024-07346-4 (дата звернення: 26.02.2025).

ДОДАТОК В

Апробація
Диплом учасника XXII Міжнародної науково-практичної конференції молодих вчених і студентів «Сучасні проблеми наукового забезпечення енергетики»

«Моделювання впливу рушійного фактору на екологічні та економічні показники з використанням машинного навчання»

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ІАТЕ_ЦТЕ_ТР-15

Аркушів 1

Київ — 2025

ДИПЛОМ

І ступеня

ЗА АКТИВНУ УЧАСТЬ У ***XXII МІЖНАРОДНІЙ НАУКОВО-ПРАКТИЧНІЙ КОНФЕРЕНЦІЇ МОЛОДИХ ВЧЕНИХ ТА СТУДЕНТІВ «СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ»***

нагороджується бакалаврантка гр. ТР-15

кафедри ЦТЕ

ТИМКОВА Анастасія Вікторівна

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» Навчально-науковий інститут атомної та теплової енергетики

Директор ННІАТЕ **Євген ПИСЬМЕННИЙ**

2025

ДОДАТОК Г

Фрагменти коду основних складових застосунку

«Моделювання впливу рушійних факторів на систему»

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ІАТЕ_ЦТЕ_ТР-15

Аркушів 3

Київ — 2025

Програмні засоби:

– мова програмування – Python;

– обробка табличних даних, робота з датами – Pandas;

– нормалізація числових значень – MinMaxScaler зі Sklearn;

– тест на стаціонарність – ADF-тест зі Statsmodels;

– виявлення причинно-наслідкових зв’язків – тест Грейнджера зі Statsmodels;

– побудова векторної авторегресії – VAR зі Statsmodels;

– алгоритм прогнозування – Random Forest зі Sklearn;

– розділення вибірки - train_test_split зі Sklearn;

– обчислення з масивами даних – NunPy;

– візуалізація результатів – Matplotlib;

– нейронна мережа для прогнозування – LSTM із TensorFlow.keras.models.

Реалізація модуля визначення причинно-наслідоквих зв’язків

```
data = pd.read_excel(file_path)
data = data.sort_values(by=data.columns[0], ascending=True).reset_index(drop=True)

all_columns = data.columns.tolist()
feature_cols = all_columns[1:-1]
target_col = all_columns[-1]

scaler = MinMaxScaler()
data[feature_cols + [target_col]] = scaler.fit_transform(data[feature_cols + [target_col]])

def make_stationary(series, name, max_diff=5):
    diff_order = 0
    current_series = series.copy()
    while diff_order < max_diff:
        result = adfuller(current_series.dropna())
        p_value = result[1]
        print(f"\nADF для {name} (різниціювання {diff_order}): p-value = {p_value:.4f}")
        if p_value < alpha:
            print(f"{name} — стаціонарна після {diff_order} різниціювань.")
            return current_series, diff_order
        diff_order += 1
        current_series = current_series.diff()
    print(f"{name} — не вдалося стаціонаризувати за {max_diff} кроків.")
    return current_series, diff_order

for col in feature_cols + [target_col]:
    data[col + '_stationary'], _ = make_stationary(data[col], col)

data.dropna(inplace=True)
stationary_cols = [col + '_stationary' for col in feature_cols + [target_col]]

time_series = data[stationary_cols]
n_obs = len(time_series)
```

```
n_vars = len(stationary_cols)
max_lags_allowed = min(10, n_obs - n_vars - 1)

model_var = VAR(time_series)
try:
    lag_selection = model_var.select_order(maxlags=max_lags_allowed)
    best_lag_candidate = lag_selection.selected_orders['aic']
    best_lag = best_lag_candidate if best_lag_candidate and best_lag_candidate > 0 else 1
    print(f"\nОптимальна затримка за AIC: {best_lag}")
except Exception as e:
    print(f"Помилка при визначенні оптимального лага за AIC: {e}")
    best_lag = 1

significant_predictors = []
causal_details = {}
causal_results = []

print(f"\n=== Тест Грейнджера (лаги 1 до {best_lag}) ===")

for predictor in feature_cols:
    pair_data = data[[target_col + '_stationary', predictor + '_stationary']]
    try:
        results = grangercausalitytests(pair_data, maxlag=best_lag, verbose=False)
        predictor_is_causal = False
        sig_lags = []
        for lag in range(1, best_lag + 1):
            p_value = results[lag][0]['ssr_ftest'][1]
            print(f"{predictor} → {target_col}, lag={lag}, p-value = {p_value:.4f}")
            if p_value < alpha:
                predictor_is_causal = True
                sig_lags.append((lag, p_value))
        if predictor_is_causal:
            significant_predictors.append(predictor)
            causal_details[(predictor, target_col)] = sig_lags
    except Exception as e:
        print(f"{predictor} пропущено через помилку: {e}")

significant_predictors = list(set(significant_predictors))

if len(feature_cols) >= 2:
    print(f"\n=== Тест Грейнджера між незалежними змінними ===")
    for cause in feature_cols:
        for effect in feature_cols:
            if cause != effect:
                pair_data = data[[effect + '_stationary', cause + '_stationary']]
                try:
                    results = grangercausalitytests(pair_data, maxlag=best_lag, verbose=False)
                    cause_is_causal = False
                    sig_lags = []
                    for lag in range(1, best_lag + 1):
                        p_value = results[lag][0]['ssr_ftest'][1]
                        print(f"{cause} → {effect}, lag={lag}, p-value = {p_value:.4f}")
                        if p_value < alpha:
                            cause_is_causal = True
                            sig_lags.append((lag, p_value))
                    if cause_is_causal:
                        causal_details[(cause, effect)] = sig_lags
                except Exception as e:
                    print(f"{cause} → {effect} пропущено через помилку: {e}")
else:
    print("\nНедостатньо незалежних змінних для перевірки причинності між ними.")


if significant_predictors:
    X = data[significant_predictors] if len(significant_predictors) > 1 else data[[significant_predictors[0]]]
    y = data[target_col]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
    importances = model.feature_importances_ * 0.99

    for feat, imp in zip(significant_predictors, importances):
        print(f"{feat} → {target_col}: важливість = {imp:.4f}")
        causal_results.append((feat, target_col, imp))
```

Реалізація модуля прогнозування цільового показника

```
df = pd.read_excel(file_path)
df.columns = df.columns.astype(str)
df = df.rename(columns={df.columns[0]: 'datetime'})
df['datetime'] = pd.to_datetime(df['datetime'])
df = df.set_index('datetime')

target_col = df.columns[-1]
print(f"Цільова змінна: {target_col}")

scaler = MinMaxScaler(feature_range=(0, 1))
y_scaled = scaler.fit_transform(df[[target_col]]).flatten()

n_input = min(10, len(y_scaled) - 1)
X, y = [], []
for i in range(len(y_scaled) - n_input):
    X.append(y_scaled[i : i + n_input])
    y.append(y_scaled[i + n_input])
X = np.array(X).reshape(-1, n_input, 1)
y = np.array(y)

model = load_model(r'LSTM_Models\lstm_univariate.h5')

preds = model.predict(X).flatten()

plt.figure(figsize=(12, 5))
plt.plot(y, 'o-', label='Actual')
plt.plot(preds, 'x--', label='Predicted')
plt.title('Actual vs Predicted (scaled)')
plt.xlabel('Sample Index')
plt.ylabel('Scaled Value')
plt.legend()
plt.tight_layout()
plt.show()

def future_forecast(series, seq_len, model, steps):
    temp = series.tolist()[-seq_len:]
    out = []
    for _ in range(steps):
        x = np.array(temp[-seq_len:]).reshape(1, seq_len, 1)
        yhat = model.predict(x, verbose=0)[0, 0]
        temp.append(yhat)
        out.append(yhat)
    return out

while True:
    try:
        forecast_steps = int(input("Введіть кількість кроків прогнозу наперед: "))
        if forecast_steps > 0:
            break
        else:
            print("Кількість кроків має бути більше 0.")
    except ValueError:
        print("Будь ласка, введіть ціле число.")

forecast = future_forecast(y_scaled, n_input, model, forecast_steps)

plt.figure(figsize=(12, 5))
plt.plot(range(len(y_scaled)), y_scaled, label='History')
plt.plot(range(len(y_scaled), len(y_scaled) + forecast_steps),
         forecast, label='Forecast')
plt.title(f'Forecast for Next {forecast_steps} Steps (scaled)')
plt.xlabel('Time Step')
plt.ylabel('Scaled Value')
plt.legend()
plt.tight_layout()
plt.show()
```