Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти другий (магістерський)

Прогнозування напруги
електричної мережі гідроелектростанції
методами машинного навчання
(тема)

Виконав:
здобувач _2_ року навчання, групи _ПМм-23-1 _
Ляшенко Є.С.
(прізвище, ініціали)
Спеціальність
113 Прикладна математика
(код і повна назва спеціальності)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Прикладна математика
(повна назва освітньої програми)
Керівник доц. Гибкіна Н.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ Сидоров М.В.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту
Кафедра _прикладної математики
Рівень вищої освіти _другий (магістерський)
Спеціальність _113 Прикладна математика
(код і повна назва)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Прикладна математика
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
“ _25_ ” _листопада_ 2024 р.

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _Ляшенку Євгенію Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Прогнозування напруги електричної мережі гідроелектростанції методами машинного навчання
затверджена наказом по університету від 22 листопада 2024 р. № 1223 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 6 січня 2025 р.

3. Вихідні дані до роботи _математична модель рекурентних нейронних мереж

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області
2. Вибір і обґрунтування методу розв’язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Аналіз предметної області
4. Метод чисельного аналізу
5. Результати обчислювального експерименту

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Підбір та вивчення технічної літератури за темою роботи | 25 листопада – 1 грудня 2024 р. | виконано |
| 2 | Вибір та обґрунтування методу | 2 – 8 грудня 2024 р. | виконано |
| 3 | Розробка алгоритму і програми | 9 – 22 грудня 2023 р. | виконано |
| 4 | Проведення аналітичних досліджень та розрахунків | 23 – 29 грудня 2024 р. | виконано |
| 5 | Робота над текстом пояснювальної записки | 30 грудня 2024 р. – 9 січня 2025 р. | виконано |
| 6 | Представлення роботи на рецензію в ЕК | 10 січня 2025 р. | виконано |

Дата видачі завдання _25_ листопада 2024 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Гибкіна Н. В.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка: 84 с., 9 рис., 1 дод., 28 джерел.

НЕЙРОННІ МЕРЕЖІ, ПРОГНОЗУВАННЯ НАПРУГИ, LSTM, GRU, МАШИННЕ НАВЧАННЯ, ЧАСОВІ РЯДИ, РЕКУРЕНТНІ МЕРЕЖІ, СЕЗОННІСТЬ, ТЕХНОЛОГІЇ ПРОГНОЗУВАННЯ.

Об’єкт дослідження – задача прогнозування напруги електричної мережі гідроелектростанції.

Мета роботи – розробка та впровадження ефективної моделі прогнозування напруги в електричній мережі гідроелектростанції з використанням методів машинного навчання, зокрема нейронних мереж LSTM і GRU.

Методи дослідження – методи машинного навчання, зокрема рекурентні нейронні мережі LSTM і GRU.

Дослідження присвячено прогнозуванню напруги в гідроелектричній мережі з використанням сучасних методів машинного навчання, зокрема нейронних мереж. В рамках експерименту реалізовано та порівняно кілька моделей, включно з LSTM, GRU, трансформерами та TCN. Отримані результати демонструють суттєве покращення точності прогнозування порівняно з традиційними лінійними моделями, такими як авторегресія.

Новизна роботи полягає у використанні сучасних нейронних мереж для врахування як короткострокових, так і довгострокових залежностей у даних.

Результати роботи рекомендуються до використання в енергетичній сфері для прогнозування та управління режимами роботи трансформаторів. Значимість роботи полягає у підвищенні точності прогнозування, що зменшує ризики перенапруг, відмов обладнання та підвищує надійність енергомережі.

Висновки роботи підтверджують ефективність використання нейронних мереж у задачах прогнозування. Подальші дослідження можуть бути спрямовані на вдосконалення моделей шляхом розробки гібридних підходів, оптимізації обчислювальних витрат та інтеграції моделей у реальні системи моніторингу.

# ABSTRACT

Introductory note: 84 pages, 9 figures, 1 appendixes, 28 sources.

NEURAL NETWORKS, VOLTAGE FORECASTING, LSTM, GRU, MACHINE LEARNING, TIME SERIES, RECURRENT NETWORKS, SEASONALITY, FORECASTING TECHNOLOGIES.

Object of research – the problem of predicting the voltage of an electrical circuit of a hydroelectric power station.

Purpose of work – development and implementation of an effective model for predicting voltage in the power grid of a hydroelectric power plant using machine learning methods, in particular LSTM and GRU neural networks.

Methods of research – machine learning methods, in particular, recurrent neural networks LSTM and GRU.

The study is devoted to voltage forecasting in a hydroelectric network using modern machine learning methods, in particular neural networks. Several models, including LSTM, GRU, transformers, and TCN, were implemented and compared in the experiment. The results show a significant improvement in forecasting accuracy compared to traditional linear models such as autoregression.

The novelty of the work lies in the use of modern neural networks to take into account both short-term and long-term dependencies in the data.

The results of this work are recommended for use in the energy sector to predict and control transformer operating modes. The significance of the work lies in improving the accuracy of forecasting, which reduces the risks of overvoltage, equipment failures, and increases the reliability of the power grid.

The conclusions of the study confirm the effectiveness of using neural networks in forecasting tasks. Further research could be aimed at improving the models by developing hybrid approaches, optimizing computational costs, and integrating the models into real-world monitoring systems.

# ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

LSTM – Long Short-Term Memory;

GRU – Gated Recurrent Unit;

TCN – Temporal Convolutional Network;

ARMA – AutoRegressive Moving Average;

ARIMA – AutoRegressive Integrated Moving Average;

RNN – Recurrent Neural Network;

SVM – Support Vector Machine;

SVR – Support Vector Regression;

ReLU – Rrectified Linear Unit.

# ВСТУП

**Актуальність теми.** Ефективне прогнозування напруги у гідроенергетичних мережах є критично важливим завданням для забезпечення стабільності енергосистеми та її адаптації до змінних умов роботи. З огляду на зростання обсягів даних та складність взаємозв'язків між метеорологічними та енергетичними показниками, використання традиційних методів прогнозування стає недостатньо ефективним. Застосування сучасних методів машинного навчання, зокрема нейронних мереж, дозволяє враховувати нелінійні залежності та забезпечувати більш точні прогнози. Ця тема є актуальною як для наукових досліджень, так і для практичного застосування в енергетиці, оскільки підвищує ефективність управління електроенергетичною системою та сприяє оптимізації її роботи.

**Мета і завдання кваліфікаційної роботи.** Метою кваліфікаційної роботи є розробка та впровадження ефективної моделі прогнозування напруги в електричній мережі гідроелектростанції з використанням методів машинного навчання, зокрема нейронних мереж LSTM і GRU. Ці моделі дозволять забезпечити більш точний прогноз на основі історичних даних та метеорологічних показників, що, у свою чергу, сприятиме стабільній роботі енергомережі. Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд та аналіз сучасного стану проблеми прогнозування напруги у гідроелектростанціях;

– дослідити теоретичні основи нейронних мереж, зокрема RNN, LSTM і GRU, у контексті машинного навчання;

– провести попередню обробку даних, включаючи очищення, стандартизацію та підготовку до моделювання;

– побудувати модель нейронної мережі для прогнозування напруги в електричній мережі, використовуючи історичні дані про напругу мережі та метеорологічні показники;

– провести тестування та валідацію моделі, оцінити її точність та порівня-

ти результати прогнозів із реальними даними, щоб зробити висновки про ефективність та можливість подальшого застосування моделі для прогнозування напруги на гідроелектростанції.

*Об'єктом дослідження* є задача прогнозування напруги електричної мережі гідроелектростанції.

*Предметом дослідження* є моделі прогнозування напруги на основі нейронних мереж LSTM та GRU, а також їх здатність узагальнювати дані з урахуванням впливу метеорологічних факторів.

**Методи дослідження.** У кваліфікаційній роботі використовуються методи машинного навчання, зокрема рекурентні нейронні мережі LSTM і GRU.

Для обробки даних застосовуються методи попередньої обробки, включаючи нормалізацію, усунення аномалій та заповнення пропусків. Оцінка якості моделей проводиться за допомогою метрик середньоквадратичної помилки (MSE) та методу крос-валідації.

**Публікації.** Результати, отримані у кваліфікаційній роботі, було представлено на 3-тій Міжнародній науково-практичній конференції англійською мовою (м. Харків, 8 листопада 2024 р.) [1].

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

## 1.1 Методи прогнозування напруги електричної мережі

Прогнозування напруги електричної мережі є однією з ключових задач у галузі енергетики, особливо для забезпечення стабільності роботи електромереж та оптимізації роботи енергогенеруючих установок. Оскільки електричні мережі функціонують у режимі реального часу і мають обмежені ресурси, ефективне прогнозування навантаження дозволяє запобігти перенавантаженню системи та збоїв у роботі. Існує кілька основних підходів до вирішення цієї задачі, які відрізняються як за використовуваними методами, так і за точністю прогнозів.

### 1.1.1 Класифікація методів прогнозування

Для прогнозування напруги електричних мереж можна використовувати такі групи методів:

– статистичні методи (базуються на класичних статистичних техніках, таких як регресія, часові ряди, експоненціальне згладжування);

– методи машинного навчання (пропонують більш складні та адаптивні підходи до прогнозування);

– гібридні методи (поєднують елементи статистичних та машинних методів, прагнучи використовувати переваги обох підходів).

Кожен із цих підходів має свої переваги та недоліки, що робить їх застосування доцільним в залежності від специфіки даних, типу навантаження та вимог до точності прогнозування.

### 1.1.2 Статистичні методи

Статистичні методи прогнозування базуються на аналізі часових рядів історичних даних про споживання електроенергії та інші параметри. Найпоширенішими серед них є методи авторегресії, ковзної середньої та їх поєднання. Розглянемо більш детально кожен з цих методів.

Метод авторегресії (AR). Цей метод базується на припущенні, що поточне значення напруги є лінійною комбінацією її попередніх значень. Модель авторегресії має такий вигляд

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \varepsilon_t,$$

де $y_t$ – значення напруги у момент часу $t$;

$c$ – константа;

$\phi_i$ – коефіцієнти авторегресії;

$p$ – порядок моделі;

$\varepsilon_t$ – випадкова похибка.

Метод ковзної середньої (MA). Цей метод враховує вплив випадкових збурень або шумів на попередніх етапах

$$y_t = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i},$$

де $\theta_i$ – параметри моделі;

$\mu$ – середнє значення напруги;

$\varepsilon_t$ – випадкова похибка.

Модель ARMA. Цей метод є поєднанням авторегресії та ковзної середньої

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t,$$

де $y_t$ – значення напруги у момент часу $t$;

$\theta_i$ – параметри моделі;

$c$ – константа;

$\phi_i$ – коефіцієнти авторегресії;

$p$ – порядок моделі;

$\varepsilon_t$ – випадкова похибка.

Ця модель дозволяє одночасно враховувати і попередні значення напруги, і випадкові збурення.

Модель ARIMA. Модифікована версія ARMA, яка використовується для нестаціонарних даних шляхом застосування різницевих операцій

$$\nabla^d y_t = (1 - L)^d y_t,$$

де $L$ – визначається як $Ly_t = y_{t-1}$, тобто він зсуває ряд на один крок назад;

$d$ – порядок диференціювання, що відповідає кількості застосувань операції $(1 - L)$ для досягнення стаціонарності.

Ідея полягає в тому, що кожне значення ряду зсувається на один крок назад. Наприклад, для ряду $y_t$ оператор лагу $L$ створює новий ряд $Ly_t = y_{t-1}$. Це дозволяє виявляти залежності між поточним значенням і попередніми спостереженнями.

### 1.1.3 Методи машинного навчання

Методи машинного навчання забезпечують більшу точність прогнозування у випадках, коли статистичні моделі не можуть адекватно описати складні

залежності в даних. Розглянемо кожен із методів детальніше [2].

Множинна лінійна регресія. Цей метод є одним із найпоширеніших методів у статистичному аналізі, що використовується для прогнозування залежної змінної $y$ на основі кількох незалежних змінних $x_1, x_2, \ldots, x_n$. Він дозволяє моделювати лінійну залежність між змінними, що робить його зручним для оцінки впливу кількох факторів одночасно. Загальна математична формула множинної лінійної регресії має вигляд

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon,$$

де $y$ – залежна змінна (цільовий показник, який ми прогнозуємо);

$\beta_0$ – константа (вільний член), яка відповідає за значення $y$, коли всі незалежні змінні $x_1, x_2, \ldots, x_n$ дорівнюють нулю;

$\beta_i$ – коефіцієнти регресії, які визначають вплив незалежних змінних $x_i$ на залежну змінну $y$;

$x_1, x_2, \ldots, x_n$ – незалежні змінні (предиктори, що пояснюють варіацію залежної змінної);

$\varepsilon$ – випадкова похибка, яка враховує вплив факторів, що не включені у модель.

Метод опорних векторів (SVM). Модель машинного навчання, що шукає гіперплощину, яка максимізує відстань між різними класами даних. Для задач регресії застосовується версія Support Vector Regression (SVR), яка мінімізує відхилення прогнозу від реальних значень напруги [2].

Дерева рішень та випадкові ліси. Ці методи будують прогнози шляхом послідовного розбиття даних на підмножини на основі певних критеріїв. Випадкові ліси складаються з кількох дерев рішень, що підвищує точність прогнозів за рахунок агрегування результатів.

Нейронні мережі. Різні архітектури нейронних мереж, такі як RNN, LSTM та GRU, використовуються для прогнозування залежностей у часових рядах.

Нейронні мережі можуть моделювати складні нелінійні взаємозв’язки між змінними, що робить їх ефективним інструментом для прогнозування напруги.

Різні архітектури, такі як RNN, LSTM та GRU, успішно застосовуються для вирішення задач прогнозування, особливо в тих випадках, коли є довгострокові залежності. RNN забезпечують обробку послідовностей шляхом передачі інформації між елементами через приховані стани, однак їх обмеженнями є проблема зникнення градієнтів при роботі з довгими послідовностями.

Для вирішення цієї проблеми було розроблено архітектури LSTM та GRU, які включають механізми "затворів", що дозволяють зберігати або забувати інформацію на різних етапах обробки даних. Це робить ці моделі особливо ефективними для задач, де попередні значення впливають на прогноз із суттєвою затримкою, наприклад, у задачах прогнозування напруги в енергомережах. LSTM і GRU дозволяють враховувати як короткострокові, так і довгострокові залежності, що є критичним для точного прогнозування.

Окрім рекурентних моделей, значну увагу привернули трансформери, які стали революційною архітектурою для роботи з послідовностями. Трансформери використовують механізм багатоголової уваги, що дозволяє моделі фокусуватися на різних аспектах взаємодії між елементами послідовності одночасно. Це дає змогу враховувати глобальні залежності в даних, що є їхньою головною перевагою над рекурентними мережами. Трансформери також не обмежуються обробкою послідовностей у заданому порядку, що значно прискорює їх навчання та робить їх більш придатними для задач із великими обсягами даних.

### 1.1.4 Гібридні методи

Гібридні моделі поєднують різні методи для досягнення більш високої точності прогнозування. Наприклад, можна поєднувати статистичні моделі з нейронними мережами, що дозволяє врахувати як лінійні, так і нелінійні взаємозв’язки.

Одним із прикладів є гібридна модель ARIMA + LSTM, де ARIMA використовується для моделювання стаціонарних компонентів часових рядів, а LSTM – для моделювання нелінійних трендів і залежностей.

Інший приклад – це поєднання SVM і RNN, де SVM використовується для класифікації типів навантажень, а RNN моделює їх зміни в часі [3].

### 1.1.5 Вибір методу для прогнозування напруги

Вибір методу прогнозування залежить від природи даних, цілей дослідження та вимог до точності прогнозу. Статистичні моделі добре працюють для лінійних задач і стаціонарних даних, тоді як методи машинного навчання підходять для складніших залежностей. Гібридні моделі забезпечують найвищу точність, оскільки вони можуть враховувати як лінійні, так і нелінійні компоненти прогнозу.

Для задачі прогнозування напруги в електричній мережі гідроелектростанції найбільш ефективними інструментами є нейронні мережі, зокрема RNN, LSTM та GRU, оскільки вони дозволяють моделювати залежності в часових рядах і враховувати вплив попередніх станів системи.

## 1.2 Рекурентні нейронні мережі у задачах прогнозування

### 1.2.1 Основні відомості про нейронні мережі

Нейронні мережі є класом моделей машинного навчання, які засновані на принципах роботи біологічних нейронних систем, зокрема людського мозку. Вони складаються з набору обчислювальних елементів, що імітують функцію біологічних нейронів, з’єднаних між собою у певну структуру, яка утворює штучну нейронну мережу. Такі мережі здатні навчатися на основі даних та мо-

жуть моделювати складні взаємозв'язки між вхідними та вихідними сигналами, що робить їх корисними для вирішення широкого спектра завдань, таких як класифікація, регресія, розпізнавання образів, прогнозування, тощо.

Типова нейронна мережа складається з трьох основних рівнів. Перший рівень, або вхідний шар – це шар, до якого подаються вхідні дані. Кожен вузол (або нейрон) цього шару відповідає окремому входу. В контексті прогнозування напруги електричної мережі такими входами можуть бути дані про поточну напругу, частоту, потужність, історичні показники або ж погодні умови.

Другий рівень складається з прихованих шарів – це посередники між вхідними даними та кінцевим результатом. Їх кількість та конфігурація визначають здатність мережі до навчання складних функцій. У прихованих шарах кожен нейрон застосовує до отриманих вхідних сигналів зважене додавання і функцію активації для врахування нелінійності. Основними функціями активації є сигмоїдна функція, гіперболічний тангенс та функція RELU [4].

Останній рівнем є вихідний шар. Це шар, який генерує кінцевий результат. У випадку регресії для прогнозування значень (наприклад, напруги), вихідний шар може містити один нейрон, що відображає прогнозоване значення.

Нейронні мережі використовують лінійні комбінації вхідних даних та нелінійні функції активації для побудови складних моделей. Для кожного нейрона прихованого шару обчислюється зважена сума його входів

$$z_j = \sum_i w_{ij} x_i + b_j ,$$

де $z_j$ – це лінійна комбінація вагових коефіцієнтів $w_{ij}$ і вхідних сигналів $x_i$;

$b_j$ – зміщення, яке дозволяє мережі краще адаптуватися до складних взаємозв'язків [4].

Після цього до $z_j$ застосовується функція активації, наприклад, ReLU

$$a_j = \max(0, z_j) .$$

Цей процес повторюється для кожного нейрона прихованих шарів, аж до вихідного шару, де формується кінцевий результат мережі.

Процес навчання нейронної мережі полягає в підгонці вагових коефіцієнтів з метою мінімізації функції втрат, яка вимірює різницю між прогнозованими значеннями мережі та фактичними значеннями. Найбільш поширеним алгоритмом навчання є метод зворотного поширення помилки, який базується на градієнтному спуску. В ході цього алгоритму градієнти функції втрат по відношенню до ваг оновлюються в зворотному напрямку від вихідного шару до вхідного.

Функція втрат для регресійних задач зазвичай базується на середньоквадратичній помилці

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

де $L$ – це функція втрат;

$n$ – кількість зразків у навчальній вибірці;

$y_i$ – справжні значення відгуку;

$\hat{y}_i$ – прогнозовані значення відгуку.

Існує кілька типів нейронних мереж, кожен з яких має свої унікальні властивості та застосування, зокрема:

а) нейронні мережі прямого поширення – це найбільш базовий тип мереж, де інформація передається лише в одному напрямку – від вхідного шару до вихідного через приховані шари; така мережа корисна для задач прогнозування або класифікації, однак не здатна ефективно обробляти часові ряди;

б) рекурентні нейронні мережі – мережі, які включають зворотні зв’язки, що дозволяє їм враховувати попередні стани системи; вони особливо корисні для обробки часових рядів, оскільки зберігають пам’ять про попередні етапи; однак базові RNN мають труднощі із запам’ятовуванням довгих послідовностей, через що було розроблено їх удосконалені версії – LSTM та GRU;

в) конволюційні нейронні мережі – широко використовуються для оброб-

ки зображень, але також застосовуються для аналізу часових рядів, оскільки можуть виявляти локальні патерни у даних.

Основними перевагами нейронних мереж є їх здатність до автоматичного виявлення складних взаємозв'язків між вхідними та вихідними даними, а також можливість навчатися на великих обсягах даних без явного програмування правил. Водночас, вони вимагають значних обчислювальних ресурсів для навчання та можуть бути схильними до перенавчання, якщо модель має занадто багато параметрів.

Таким чином, нейронні мережі є дієвим інструментом для моделювання складних фізичних процесів, таких як прогнозування напруги електричної мережі гідроелектростанції, що робить їх доцільними для використання в дослідженні прогнозних систем.

### 1.2.2 Рекурентні нейронні мережі

Рекурентні нейронні мережі представляють собою клас нейронних мереж, які особливо підходять для аналізу часових рядів та послідовних даних. Їх ключова особливість полягає у здатності зберігати інформацію про попередні стани системи, що дозволяє враховувати часові залежності між елементами послідовності. Це робить RNN ефективним інструментом для задач прогнозування, оскільки в таких задачах ключову роль відіграє історія змін вхідних параметрів, наприклад, зміни напруги, частоти або інших фізичних характеристик у електричній мережі.

Основна відмінність RNN від інших типів нейронних мереж полягає в тому, що кожен нейрон у прихованому шарі отримує не тільки вхідні дані з поточного кроку, але й інформацію про попередній стан. Це дозволяє моделі зберігати «пам'ять» про попередні стани і використовувати її для прогнозування наступних значень.

На кожному кроці часу $t$ стан прихованого шару $h_t$ обчислюється на ос-

нові інформації про вектор станів прихованого шару $h_{t-1}$ та вектору вхідних даних $x_t$ на момент часу $t$, а вихід мережі $y_t$ на кожному кроці часу залежить від стану прихованого шару $h_t$.

Незважаючи на переваги рекурентних мереж, вони мають деякі недоліки. Однією з головних проблем базових рекурентних мереж є так звані проблеми «зникання» та «вибуху градієнтів». Це відбувається через те, що значення градієнтів при зворотному поширенні помилки можуть або занадто зменшуватись, або стрімко зростати, що ускладнює ефективне навчання мережі на довгих послідовностях.

Щоб подолати ці проблеми, були запропоновані удосконалені архітектури рекурентних мереж, такі як LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit).

Мережі LSTM відрізняються від базових RNN тим, що вони мають спеціальні блоки пам'яті, які контролюються трьома ключовими компонентами – вхідними, забутими та вихідними вентилями. Ці вентилі регулюють потік інформації в мережі, дозволяючи ефективно зберігати або «забувати» інформацію на тривалих часових інтервалах.

У блоках LSTM для кожного моменту часу $t$ обчислюються кілька вентилів, які контролюють інформаційні потоки. Вхідний вентиль визначає, наскільки нові дані важливі для стану. Вентиль забування вирішує, яку частину попереднього стану $h_{t-1}$ слід забути. Стан пам'яті $C_t$ оновлюється на основі цих вентилів, шляхом зберігання частини попереднього стану і додавання нової інформації [5].

Вихідний вентиль контролює, яку інформацію з блоку пам'яті передавати далі, а вихідний стан $h_t$ формується на основі даних вихідного вентиля та стану пам'яті. Це дозволяє моделі LSTM адаптивно управляти як новими, так і старими даними, забезпечуючи ефективне збереження важливої інформації протягом часу.

Така архітектура дозволяє LSTM ефективно працювати з довготривалими залежностями в часових рядах, що робить її оптимальною для задач прогнозування напруги у електричних мережах.

GRU – це ще один тип рекурентних нейронних мереж, що був розроблений як спрощена версія LSTM, яка при цьому зберігає здатність ефективно працювати з довготривалими залежностями в часових рядах. Основна відмінність між GRU і LSTM полягає в тому, що GRU має менше параметрів та вентилів, що робить її більш компактною та швидкою у навчанні [5].

На відміну від LSTM, GRU має тільки два вентиля: вентиль оновлення та вентиль скидання, які контролюють, як інформація передається між станами. Це спрощує модель і зменшує кількість обчислень, що особливо корисно для задач, де важлива висока швидкість обробки. Вентиль оновлення поєднує у собі вхідний вентиль і вентиль забування LSTM, та контролює як інформацію з попереднього стану, так і нову інформацію. Він визначає, скільки старої інформації слід зберегти, а скільки – оновити новими даними. Вентиль скидання, в свою чергу, контролює, скільки інформації з попереднього стану потрібно забути, що дозволяє моделі більш гнучко адаптуватися до нових вхідних даних.

Кандидат на новий стан обчислюється з урахуванням вхідних даних і модифікованого попереднього стану, що дозволяє мережі враховувати лише релевантні частини старої інформації. Остаточний стан формується як комбінація старого стану та нового кандидата, контрольованих оновлювальним вентилем. Це забезпечує баланс між збереженням старої інформації та інтеграцією нових даних, дозволяючи моделі ефективно навчатися та прогнозувати.

Завдяки меншій кількості вентилів та параметрів GRU часто працює швидше, ніж LSTM, і може бути більш ефективною для певних задач прогнозування часових рядів, зокрема, для задач прогнозування напруги в електричній мережі, забезпечуючи при цьому прийнятну точність.

### 1.2.3 Модель прогнозування напруги за допомогою RNN

Для побудови моделі прогнозування навантаження за допомогою рекурентних нейронних мереж (як LSTM, так і GRU), використовують часові ряди, що

містять історичні дані про зміни напруги, частоти, потужності або інші показники електричної мережі. Модель отримує на вхід послідовність даних за минулі періоди часу і використовує ці дані для прогнозування значень на наступний період.

Під час навчання мережі основною метою є мінімізація функції втрат, яка, як правило, базується на середньоквадратичній помилці

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

де $y_i$ – справжні значення напруги в мережі;

$\hat{y}_i$ – прогнозовані значення моделі.

У процесі навчання ваги мережі коригуються таким чином, щоб прогнозовані значення якомога точніше наближалися до реальних значень напруги. Після завершення навчання модель може бути використана для прогнозування напруги на основі поточних та історичних даних.

Вибір між LSTM та GRU залежить від конкретної задачі та вимог до продуктивності: GRU часто використовуються для задач, де важлива швидкість навчання та обчислень, тоді як LSTM можуть бути більш ефективними для задач із довгими часовими залежностями. Обидві моделі мають високу ефективність у прогнозуванні складних нелінійних взаємозв’язків, які, зокрема, мають місце у задачі прогнозування напруги в електричній мережі гідроелектростанції.

## 1.3 Формальна та змістовна постановка задачі

### 1.3.1 Змістовна постановка задачі

Провівши аналіз предметної області прогнозування напруги на основі нейронних мереж, можемо перейти до формулювання змістовної постановки

задачі з урахуванням отриманих результатів.

Завдання полягає у побудові моделі на основі рекурентних нейронних мереж, зокрема таких архітектур, як LSTM або GRU, які здатні найточніше прогнозувати значення напруги в електричній мережі гідроелектростанції на основі історичних даних. Модель повинна враховувати не лише часові залежності навантаження, а й вплив метеорологічних показників, що дозволить забезпечити стабільну та ефективну роботу енергосистеми.

Набір даних включає погодинні вимірювання метеорологічних показників протягом трьох років: температура повітря, хмарність, швидкість вітру, тиск на п’яти метеостанціях, розташованих поблизу гідроелектростанції, а також історичні дані про напругу в мережі. Оскільки ці дані мають часову структуру та складні нелінійні взаємозв’язки, використання нейронних мереж є доцільним. Архітектури RNN, LSTM та GRU можуть зберігати інформацію про попередні стани системи та використовувати її для прогнозування майбутніх значень.

Перед початком роботи необхідним кроком буде підготовка та попередня обробка даних. Це включає очистку даних від пропусків або шумів, стандартизацію для вирівнювання масштабу змінних, а також розподіл даних на тренувальну, валідаційну та тестову вибірки. Це дозволить моделі навчитись на історичних даних і коректно прогнозувати напругу в реальному часі.

Фінальним результатом задачі буде побудована модель на основі нейронних мереж, здатна передбачати напругу в електричній мережі гідроелектростанції. Модель повинна мінімізувати похибку прогнозу та забезпечувати надійне управління навантаженням в реальних умовах експлуатації мережі.

### 1.3.2 Формальна постановка задачі

Нехай маємо набір даних $X = \{(x_t, y_t)\}$, де $x_t$ – вхідні дані на момент часу $t$, а $Y_{t,t+k}$ – значення напруги в електричній мережі за певний період $k$, почина-

ючи з моменту часу $t$. Вхідні дані $x_t$ складаються з таких компонентів:

– набір $Y_{t-n,t}$ значень напруги у мережі за певний період $n$ до моменту часу $t$: $y_{t-n}$, ..., $y_{t-1}$;

– набір $M_{t-n,t}$ метеорологічних показників, включаючи температуру, швидкість вітру, тиск тощо, за певний період $n$ до моменту часу $t$, тобто у моменти часу $t-n$, ..., $t-1$.

Таким чином,

$$x_t = \{Y_{t-n,t}, M_{t-n,t}\}.$$

Задача полягає в побудові функції прогнозування $f$, яка на основі набору вхідних даних $x_t$ зможе прогнозувати значення напруги $y_{t+k}$ для майбутнього моменту часу $t+k$, де $k$ – це горизонт прогнозування, який може варіюватися від кількох годин до кількох діб. Тобто необхідно знайти функцію

$$y_{t+k} = f(x_t, x_{t-1}, \ldots, x_{t-n}),$$

де $n$ – кількість часових інтервалів, які будуть використані для прогнозу (довжина «вікна» спостережень).

У роботі для прогнозування обрано рекурентні нейронні мережі, зокрема такі архітектури, як LSTM або GRU, оскільки вони дозволяють моделі запам’ятовувати та використовувати інформацію про попередні стани системи, що є ключовим для роботи з часовими рядами.

Модель прогнозування на основі рекурентної нейронної мережі формально подається як послідовність станів

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b_h),$$

де $h_t$ – це прихований стан моделі на момент часу $t$;

$W_h$, $W_x$ – матриці ваг прихованого шару та вхідного шару відповідно;

$b_h$ – вектор зміщень;

$\sigma$ – нелінійна активаційна функція (наприклад, сигмоїда або ReLU).

Вихід моделі на кожному кроці часу визначається як

$$\hat{y}_{t+k} = W_y h_t + b_y,$$

де $W_y$ – матриця ваг вихідного шару;

$b_y$ – зміщення виходу.

Метою є мінімізація функції втрат, яка у задачі прогнозування є середньоквадратичною помилкою (MSE) між прогнозованими значеннями $\hat{y}_{t+k}$ і реальними значеннями $y_{t+k}$

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

де $N$ – кількість спостережень у наборі даних [6].

Таким чином, з математичної точки зору метою є мінімізація похибки прогнозу шляхом навчання параметрів рекурентної нейронної моделі $W_h$, $W_x$, $W_y$, $b_h$, $b_y$, що дозволить отримувати прогнози напруги в електричній мережі на основі історичних даних про напругу мережі та метеорологічні показники.

## 1.4 Постановка задач дослідження

Виходячи з проведеного аналізу предметної області та розглянутих методів прогнозування, ми дійшли висновку, що найдоцільнішим підходом для ви-

рішення поставленої задачі є використання рекурентних нейронних мереж (RNN), зокрема архітектур LSTM та GRU. Ці мережі добре зарекомендували себе у задачах, пов’язаних із прогнозуванням часових рядів, що обумовлює їх вибір як оптимального методу для передбачення напруги в електричній мережі гідроелектростанції.

Отже, метою даної кваліфікаційної роботи є побудова моделі на основі рекурентної нейронної мережі (LSTM або GRU), яка дозволить ефективно прогнозувати значення напруги в мережі гідроелектростанції, спираючись на історичні дані про напругу мережі та метеорологічні показники.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд та аналіз сучасного стану проблеми прогнозування напруги у гідроелектростанціях;

– дослідити теоретичні основи нейронних мереж, зокрема RNN, LSTM і GRU, у контексті машинного навчання;

– провести попередню обробку даних, включаючи очищення, стандартизацію та підготовку до моделювання;

– побудувати модель нейронної мережі для прогнозування напруги в електричній мережі, використовуючи історичні дані про напругу мережі та метеорологічні показники;

– провести тестування та валідацію моделі, оцінити її точність та порівняти результати прогнозів із реальними даними, щоб зробити висновки про ефективність та можливість подальшого застосування моделі для прогнозування напруги на гідроелектростанції.

# 2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ'ЯЗАННЯ

## 2.1 Основи нейронних мереж для прогнозування часових рядів

### 2.1.1 Архітектура рекурентних нейронних мереж

Рекурентні нейронні мережі є спеціалізованим типом нейронних мереж, що дозволяють моделювати послідовні дані, зокрема часові ряди, текст або будь-які інші дані, що мають часову або послідовну структуру. Основна ідея RNN полягає у використанні попередніх значень для обчислення поточного стану, що дозволяє зберігати інформацію про попередні елементи послідовності у внутрішньому стані мережі. Це забезпечує здатність RNN враховувати попередні контексти під час обробки наступних елементів послідовності [7].

Основним елементом RNN є рекурентний шар, що складається з блоків, які виконують повторювані обчислення. Нехай $x_t$ – вхідний вектор у момент часу $t$, а $h_t$ — прихований стан у момент часу $t$. Мережа обчислює прихований стан за допомогою рекурентного виразу

$$h_t = f(W_h h_{t-1} + W_x x_t + b),$$

де $h_t$ – вектор прихованого стану у момент часу $t$;

$W_h$ – матриця ваг для прихованого стану;

$W_x$ – матриця ваг для вхідних даних;

$b$ – вектор зміщень;

$f$ – активаційна функція.

Важливою особливістю є те, що прихований стан $h_t$ залежить як від вхідного вектора $x_t$, так і від попереднього прихованого стану $h_{t-1}$, що дозволяє мережі «пам'ятати» інформацію з попередніх кроків.

На основі прихованого стану $h_t$ обчислюється вихідний вектор $y_t$. Фор-

мула для обчислення виходу може бути записана так

$$y_t = g(W_y h_t + c),$$

де $y_t$ – вихідний вектор у момент часу $t$;

$W_y$ – матриця ваг для перетворення прихованого стану у вихід;

$c$ – вектор зміщень для виходу;

$g$ – активаційна функція вихідного шару.

Навчання RNN здійснюється шляхом мінімізації функції витрат (наприклад, середньоквадратичної помилки) за допомогою методу зворотного поширення похибок (BPTT). У цьому методі мережа оптимізує свої параметри $W_h, W_x, W_y$ і $b$ шляхом обчислення градієнтів на кожному кроці послідовності. Функція витрат для всього часового ряду може бути записана так

$$L = \sum_{t=1}^{T} \ell(y_t, \hat{y}_t),$$

де $L$ – загальна втрата;

$\ell$ – функція витрат для кожного кроку;

$y_t$ – справжнє значення у момент часу $t$;

$\hat{y}_t$ — прогнозоване значення у момент часу $t$;

$T$ – загальна кількість кроків у послідовності.

Таким чином, архітектура RNN забезпечує можливість обробки послідовних даних та врахування довготривалих залежностей. Однак базові RNN можуть мати проблеми з обробкою довгих послідовностей через зникнення або вибух градієнтів під час навчання, що знижує їх ефективність. Для вирішення цих проблем були розроблені більш складні архітектури, такі як LSTM та GRU, які ми розглянемо в наступному підрозділі.

### 2.1.2 Поняття стану та часових залежностей у моделюванні часових рядів

У моделюванні часових рядів важливими аспектами є поняття стану та часових залежностей, які дозволяють відстежувати зміни в часі та робити прогнози на основі попередніх значень. Часові ряди — це послідовності значень, що змінюються з часом і можуть бути залежними як від попередніх значень, так і від інших змінних.

Стан – це вектор або набір значень, що містить всю необхідну інформацію про систему у певний момент часу. У контексті моделювання часових рядів стан $h_t$ містить інформацію про поточний момент часу $t$ і залежить від попередніх значень вхідного сигналу $x_{t-1}, x_{t-2}, \ldots$. Стан можна записати у вигляді функції

$$h_t = f(h_{t-1}, x_t),$$

де $h_t$ – стан у момент часу $t$;

$h_{t-1}$ – стан у попередній момент часу;

$x_t$ – вхідний сигнал або спостереження у момент часу $t$;

$f$ – функція переходу стану, що визначає, як стан змінюється з урахуванням вхідних даних.

Рекурентні нейронні мережі використовують це поняття для моделювання залежностей у часових рядах, де стан передається з кроку на крок, зберігаючи інформацію про попередні значення.

Часові залежності (або автокореляції) виникають, коли значення часового ряду залежать від попередніх значень цього ж ряду. Наприклад, для часового ряду $\{x_t\}$ ми можемо мати таку лінійну модель

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + \varepsilon_t,$$

де $x_t$ – значення часового ряду у момент часу $t$;

$\phi_1, \phi_2, \ldots, \phi_p$ – параметри моделі, що визначають вплив попередніх значень на поточне значення;

$p$ – порядок моделі (кількість попередніх кроків, які враховуються);

$\varepsilon_t$ – випадкова похибка або шум у момент часу $t$.

Ця модель відома як авторегресійна модель (AR) і використовується для опису часових рядів, де поточне значення залежить лінійно від декількох попередніх значень. Нелінійні моделі, такі як RNN, розширюють цю концепцію, дозволяючи враховувати як лінійні, так і нелінійні залежності [7].

У рекурентних нейронних мережах залежності між станами можуть бути як короткочасними, так і довготривалими. Короткочасні залежності означають, що поточний стан $h_t$ залежить від декількох попередніх кроків, тоді як довготривалі залежності враховують вплив значень, що були набагато раніше. Важливо, що традиційні RNN мають обмежену здатність враховувати довготривалі залежності через проблеми з вибухом або зникненням градієнтів під час навчання [7].

Формула змінення станів у RNN, що враховує часові залежності, виглядає наступним чином

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b),$$

де $\sigma$ – активаційна функція, яка може бути нелінійною (наприклад, $\tanh$);

$W_h$ і $W_x$ – вагові матриці для прихованого стану та вхідного сигналу;

$b$ – вектор зміщень, що додається для зсуву значень.

Таким чином, для моделювання часових залежностей використовується комбінація попередніх станів та вхідних сигналів, що дозволяє прогнозувати майбутні значення на основі історії ряду. Важливою особливістю є те, що RNN можуть враховувати нелінійні залежності, що забезпечує їх універсальність у моделюванні складних часових структур.

### 2.1.3 Активаційні функції та метод зворотного розповсюдження похибки

У рекурентних нейронних мережах активаційні функції та алгоритм зворотного розповсюдження похибки через час (BPTT) є ключовими компонентами, які забезпечують можливість навчання і моделювання часових рядів. Активаційні функції допомагають мережі вчитися нелінійним залежностям у даних, тоді як алгоритм BPTT дозволяє оновлювати вагові коефіцієнти мережі з урахуванням послідовності часових станів.

Активаційні функції визначають, як вихід нейрона залежить від вхідного сигналу. У рекурентних нейронних мережах найчастіше використовуються такі функції активації, як сигмоїдна, гіперболічний тангенс та ReLU.

Сигмоїдна функція активації має вигляд:

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Сигмоїдна функція переводить значення $x$ в діапазон від 0 до 1. Вона використовується для прихованих шарів, де потрібно обмежити вихідне значення. Проте сигмоїдні функції страждають від проблеми зникнення градієнта при великих або дуже малих значеннях вхідного сигналу.

Гіперболічний тангенс (tanh) має вигляд:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}.$$

Функція $\tanh(x)$ перетворює вхід $x$ у діапазон від $-1$ до $1$, що дозволяє моделювати як додатні, так і від'ємні значення. Вона зазвичай використовується в RNN завдяки ширшому діапазону значень у порівнянні з сигмоїдою, але також має проблему зникнення градієнта.

Функція активації ReLU має вигляд:

$$\mathrm{ReLU}(x) = \max(0, x).$$

ReLU використовується у багатьох архітектурах нейронних мереж, але для RNN вона може бути менш ефективною, оскільки відсутність від'ємних значень може призвести до нестабільності під час навчання [7].

Алгоритм BPTT є варіацією стандартного методу зворотного розповсюдження похибки, який застосовується для рекурентних нейронних мереж для врахування часових залежностей. У процесі BPTT обчислюються похідні похибки мережі за кожним кроком часу, а потім оновлюються ваги з урахуванням цих похибок.

Загальна похибка послідовності обчислюється за формулою

$$E = \sum_{t=1}^{T} E_t,$$

де $E_t$ – похибка в момент часу $t$;

$T$ – загальна кількість часових кроків у послідовності.

Похибка на кожному кроці $t$ обчислюється як

$$E_t = \frac{1}{2}(y_t - \hat{y}_t)^2,$$

де $y_t$ – реальне значення виходу у момент часу $t$;

$\hat{y}_t$ – прогнозоване значення виходу у момент часу $t$.

Щоб оновити ваги, необхідно обчислити похідну похибки за вагами. Градієнт похибки для вагового коефіцієнта $W$ можна обчислити за формулою:

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}.$$

З урахуванням рекурентної структури мережі градієнт у момент часу $t$ можна розписати через похибку на кроці $t$ та градієнти попередніх станів:

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial W},$$

де $\frac{\partial E_t}{\partial \hat{y}_t}$ – похідна похибки по відношенню до виходу;

$\frac{\partial \hat{y}_t}{\partial h_t}$ – похідна виходу по відношенню до стану;

$\frac{\partial h_t}{\partial W}$ – похідна стану по відношенню до ваг.

Оскільки стан $h_t$ залежить від попередніх станів, похідну необхідно обчислювати рекурсивно, що і є суттю алгоритму BPTT. Це забезпечує навчання мережі з урахуванням часових залежностей, але водночас створює ризики для стабільності мережі через проблему зникнення або вибуху градієнтів, що може вимагати додаткових методів стабілізації, таких як обмеження градієнта.

## 2.2 Архітектури LSTM та GRU

### 2.2.1 Комірки пам’яті рекурентних нейронних мереж

Рекурентні нейронні мережі мають здатність запам’ятовувати інформацію про попередні стани за допомогою циклічних зв’язків. Вони особливо корисні для моделювання часових рядів, де залежність між даними змінюється з часом. Проте стандартна RNN має обмеження – проблема зникання або вибуху градіє-

нта при навчанні тривалих послідовностей. Ці обмеження були подолані завдяки розробці складніших архітектур, таких як LSTM та GRU, які дозволяють зберігати інформацію на тривалих проміжках часу. Розглянемо математичні основи цих моделей [8].

У стандартній RNN кожен елемент послідовності має зв'язок з попереднім, і стан передається далі в часі. Математично, для кожного часу $t$ стан $h_t$ обчислюється як

$$h_t = \phi(W_h x_t + U_h h_{t-1} + b_h),$$

де $x_t$ – вхідні дані на кроці $t$;

$h_{t-1}$ – попередній стан;

$W_h$, $U_h$ – вагові матриці;

$b_h$ – вектор зсуву;

$\phi$ – активаційна функція.

Основна проблема стандартної RNN полягає в тому, що через використання активаційних функцій градієнти можуть ставати дуже малими або дуже великими при зворотному розповсюдженні похибки. Це ускладнює навчання тривалих послідовностей, тому використовують LSTM та GRU.

LSTM має більш складну архітектуру, що включає спеціальні структури, звані вентилями, які дозволяють контролювати, яку інформацію зберігати, яку забувати, а яку передавати далі [9].

GRU є спрощеною версією LSTM. Вона має меншу кількість параметрів, ніж LSTM, і, як наслідок, швидше тренується, зберігаючи при цьому здатність обробляти довготривалі залежності.

Основні відмінності між LSTM і GRU полягають у структурі та кількості вентилів. LSTM має три вентилі, тоді як GRU – два. Це дозволяє зменшити кількість параметрів і зробити модель менш складною. GRU зазвичай швидше тренується і простіша в реалізації, але LSTM може бути ефективнішою при об-

робці дуже довгих залежностей, оскільки її архітектура дозволяє більш гнучке управління інформацією [9].

Обидві ці моделі є вдосконаленням стандартної RNN, і вибір між ними залежить від конкретної задачі. У деяких випадках, коли часові ряди мають короткі або середньої довжини залежності, GRU може бути кращим вибором. Якщо ж дані мають складніші та довготривалі залежності, LSTM часто демонструє кращі результати.

### 2.2.2 Функції активації LSTM та механізм «затворів»

У LSTM (Long Short-Term Memory) нейронних мережах ключовими є механізми активації, що забезпечують обробку та збереження інформації протягом тривалих часових проміжків. Для цього використовуються функції активації, такі як сигмоїда і гіперболічний тангенс, а також вентилі (або затвори), що регулюють потік інформації через комірку пам'яті [9].

Сигмоїдальна функція активації $\sigma(x) = \frac{1}{1+e^{-x}}$ приймає на вхід будь-яке дійсне значення $x$ та перетворює його у значення від 0 до 1. Виходячи з цього, основна роль сигмоїдальної функції в LSTM – контролювати, які значення інформації передаються далі, а які блокуються. У LSTM сигмоїда використовується у вентилях входу ($i_t$), забування ($f_t$) і виходу ($o_t$).

Функція активації гіперболічний тангенс $\tanh(x)$ використовується в LSTM для перетворення вхідної інформації та генерації нових станів комірки. Функція $\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$ приймає на вхід будь-яке дійсне значення $x$ та перетворює його у значення від –1 до 1, що дозволяє LSTM масштабувати вхідні дані та значення стану комірки таким чином, щоб вони були пропорційні за величиною [10].

Сигмоїдальна функція і гіперболічний тангенс доповнюють одне одного,

сигмоїда використовуються для «рішення», яку інформацію пропускати, тоді як tanh нормалізує величини інформації, що передається або зберігається.

Активаційні функції відіграють важливу роль у роботі LSTM, оскільки вони контролюють активацію вентилів і, як наслідок, те, які дані зберігаються, оновлюються або видаляються у комірці пам'яті. Сигмоїдальна функція визначає ступінь активації вентилів, а tanh контролює значення стану комірки, що дозволяє LSTM ефективно зберігати і обробляти інформацію протягом тривалих часових періодів [10].

Вентилі у LSTM виконують ключову функцію контролю потоку інформації. Вони дозволяють або блокувати інформацію, або зберігати її в пам'яті протягом тривалих часових проміжків, запобігаючи проблемі зникнення градієнта [11]. Оновлення стану LSTM з використанням вентилів відбувається у такій послідовності:

а) вентиль забування $f_t$ визначає, яку частину інформації з попереднього стану комірки потрібно забути, і розраховується за формулою

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f),$$

де $W_f$ – ваги вентиля забування;

$U_f$ – ваги для попереднього стану;

$b_f$ – зсув;

б) вентиль входу $i_t$ визначає, яку нову інформацію потрібно зберегти в комірці пам'яті

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i),$$

де $W_i$ – ваги вентиля входу;

$U_i$ – ваги для попереднього стану;

$b_i$ – зсув;

в) «кандидат» на новий стан комірки $\tilde{C}_t$ визначає інформацію, яка може бути додана до поточного стану

$$\tilde{C}_t = \tanh(W_C x_t + U_C h_{t-1} + b_C),$$

де $W_C$ – ваги для генерації кандидата на новий стан;

$U_C$ – ваги для стану комірки;

$b_C$ – зсув;

г) оновлення стану комірки $C_t$ відбувається на основі попереднього стану $C_{t-1}$ та нової інформації шляхом об'єднання інформації від вентилів забування і входу для оновлення стану комірки, тобто попередній стан комірки $C_{t-1}$ певним чином зберігається або забувається, а також додається нова інформація

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

д) вентиль виходу $o_t$ контролює, яка частина оновленого стану комірки передається на вихід

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o),$$

де $W_o$ – ваги вентиля виходу;

$U_o$ – ваги для попереднього стану;

$b_o$ – зсув.

Фінальне оновлення вихідного стану $h_t$ визначається за допомогою вентиля виходу і оновленого стану комірки

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

Це забезпечує нормалізацію вихідного стану і контроль потоку інформації на наступні часові кроки.

### 2.2.3 Опис та математична модель GRU як спрощеної версії LSTM

GRU – це архітектура нейронної мережі, яка є спрощеною версією LSTM. Вона була розроблена для вирішення проблеми зникнення градієнта в стандартних рекурентних нейронних мережах (RNN) і для зменшення складності обчислень, що виникає в LSTM через наявність кількох вентилів [12].

Основна ідея GRU полягає в об’єднанні механізмів керування пам’яттю та оновлення стану в один вентиль , що зменшує кількість параметрів та обчислювальних ресурсів, необхідних для навчання моделі, зберігаючи при цьому здатність захоплювати довготривалі залежності [13].

На відміну від LSTM, в GRU використовуються лише два основних вентилі:

– вентиль оновлення $z_t$, який об’єднує у собі вхідний вентиль та вентиль забування LSTM;

– вентиль скидання $r_t$.

Ці вентилі керують тим, як інформація передається крізь нейронну мережу та як зберігається стан.

Вентиль оновлення визначає, яка частина попереднього стану $h_{t-1}$ зберігається і яка частина оновлюється новою інформацією. Він розраховується за допомогою сигмоїдної функції активації

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z),$$

де $W_z$ – матриця ваг для вхідного вектора $x_t$;

$U_z$ – матриця ваг для попереднього стану $h_{t-1}$;

$b_z$ – вектор зсуву;

$\sigma$ – сигмоїдна функція активації.

Сигмоїдна функція дає значення $z_t$ у межах від 0 до 1. Якщо значення близьке до 1, це означає, що велика частина попереднього стану зберігається; якщо близьке до 0, то стан оновлюється новою інформацією.

Вентиль скидання контролює, скільки інформації з попереднього стану використовується для обчислення нового стану

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r),$$

де $W_r$ – матриця ваг для вхідного вектора $x_t$;

$U_r$ – матриця ваг для попереднього стану $h_{t-1}$;

$b_r$ – вектор зсуву.

Цей вентиль дозволяє моделі «забувати» частину минулої інформації, коли вона обчислює новий стан, що корисно для задач, де не всі попередні значення однаково важливі [14].

«Кандидат» у наступний стан $\tilde{h}_t$ – це новий стан, що розраховується з використанням скинутого попереднього стану і поточного вхідного значення

$$\tilde{h}_t = \tanh(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h),$$

де $W_h$ – матриця ваг для вхідного вектора $x_t$;

$U_h$ – матриця ваг для попереднього стану $h_{t-1}$;

$\odot$ – операція поелементного множення;

$b_h$ – вектор зсуву.

Така формула дозволяє враховувати лише важливу інформацію з минулих станів залежно від контексту.

Остаточно новий стан $h_t$ є комбінацією попереднього стану $h_{t-1}$ і нової інформації $\tilde{h}_t$, де ваги для кожного з цих значень визначаються за допомогою вентиля оновлення:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t .$$

Ця формула показує, як вентиль оновлення вирішує, яка частина попереднього стану буде збережена, а яка – оновлена новими даними. Якщо $z_t$ близьке до 1, GRU зберігає значну частину попереднього стану; якщо $z_t$ близьке до 0, модель більшу увагу приділяє новій інформації.

GRU спрощує LSTM, об'єднуючи вентилі входу та виходу в єдиний вентиль оновлення. Це зменшує кількість параметрів і робить навчання більш ефективним. GRU швидше навчається, оскільки має менше вентилів і параметрів, що дозволяє уникнути проблеми перенавчання при невеликій кількості даних.

Головною перевагою GRU є здатність забезпечувати подібний рівень точності та ефективності, як і LSTM, але на основі простішої структури. Це робить GRU привабливою для задач, де є необхідність у запам'ятовуванні довготривалих залежностей у часових рядах, але при обмежених обчислювальних ресурсах або великих наборах даних [15].

## 2.3 Методи навчання нейронних мереж для часових рядів

### 2.3.1 Оптимізація: метод зворотного розповсюдження похибки та функція втрат

Процес оптимізації нейронних мереж, особливо в задачах прогнозування часових рядів, базується на методі зворотного розповсюдження похибки (BPTT) та мінімізації функцій втрат. Для RNN, LSTM та GRU у якості функції

втрат найчастіше використовується середньоквадратична похибка (MSE), яка дозволяє зменшити відмінність між прогнозованими та фактичними значеннями, що покращує якість прогнозів [16].

Спочатку мережа обчислює значення активації кожного нейрона в прихованих шарах на основі вхідних даних. Це значення передається по всій мережі до вихідного шару, де відбувається прогноз. Після отримання прогнозованого значення порівнюємо його з фактичним значенням, і на основі цього обчислюється функція втрат у вигляді середньоквадратичної похибки:

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

де $y_i$ – це справжнє значення;

$\hat{y}_i$ – прогнозоване значення;

$N$ – кількість зразків у вибірці.

MSE обчислює середнє значення квадратів різниць між фактичними та прогнозованими значеннями, що дозволяє визначити великі помилки і оцінити, наскільки далеко прогнозовані значення від реальних [17].

Для мінімізації функції втрат використовується метод градієнтного спуску. Він є ітераційною процедурою, відповідно до якої ваги нейронної мережі оновлюються за правилом

$$w_{ij} = w_{ij} - \eta\frac{\partial \text{MSE}}{\partial w_{ij}},$$

де $w_{ij}$ – це вага, яка з'єднує нейрони;

$\eta$ – швидкість навчання;

$\frac{\partial \text{MSE}}{\partial w_{ij}}$ – часткова похідна функції втрат за відповідною вагою.

Цей вираз визначає напрямок і величину зміни ваги для того, щоб змен-

шити помилку прогнозування. Вибір оптимальної швидкості навчання є критично важливим: якщо вона надто висока, модель може не збігатися, а якщо надто низька – процес навчання буде дуже повільним.

Метод зворотного розповсюдження похибки для задач прогнозування часових рядів (BPTT) враховує часові залежності в даних. BPTT дозволяє враховувати всі кроки попередніх станів моделі. Основна ідея полягає в обчисленні похідних для всіх шарів мережі та оновленні ваг у зворотному порядку, починаючи з вихідного шару і повертаючись до вхідного. В процесі зворотного проходження градієнти обчислюються не лише для поточного часу, але й для всіх попередніх часових кроків, що дозволяє мережі враховувати довгострокові залежності в даних [17].

У випадку зворотного розповсюдження похибки можливе явище зникнення або вибуху градієнта, коли значення градієнтів стають надто малими або великими. Для подолання цього в LSTM та GRU використовуються механізми вентилів, що допомагають контролювати проходження градієнтів через мережу, тим самим забезпечуючи стабільність процесу оптимізації.

Таким чином, процес оптимізації нейронних мереж для часових рядів є складним і багатокомпонентним. Він включає правильний вибір функції втрат, алгоритму оптимізації, а також методів для подолання специфічних проблем, таких як зникнення градієнтів. Усе це робить зворотне розповсюдження похибки та функцію втрат MSE критично важливими елементами для успішного навчання моделей на часових рядах.

### 2.3.2 Крос-валідація для часових рядів

Крос-валідація є фундаментальною технікою для оцінки якості моделі та її здатності узагальнювати дані. Проте у випадку часових рядів традиційні методи крос-валідації, такі як розбиття даних на випадкові фолди, непридатні через часові залежності між спостереженнями. Дані часових рядів мають певну

послідовність, і значення в наступні моменти часу можуть залежати від значень у попередні моменти. Це означає, що випадкове перемішування даних призведе до порушення хронологічного порядку, що, у свою чергу, спотворить оцінку моделі [18].

Для часових рядів використовується спеціалізований підхід до крос-валідації, який враховує часову структуру даних. Цей метод дозволяє розбити дані таким чином, щоб зберегти їх хронологічний порядок, і поступово розширювати навчальну вибірку, наближаючись до реальних умов прогнозування.

Розглянемо математичний опис цього процесу. Нехай маємо часовий ряд

$$X = \{x_1, x_2, \ldots, x_T\},$$

де $x_t$ – значення в момент часу $t$;

$T$ – загальна кількість спостережень.

Ми розділяємо цей часовий ряд на навчальну та тестову вибірки, при цьому навчальна вибірка постійно розширюється на кожному кроці.

На першому кроці вибираємо початковий набір даних для навчання, який містить перші $n$ спостережень $X^1_{\text{train}} = \{x_1, x_2, \ldots, x_n\}$. Модель навчається на цій вибірці, а потім ми тестуємо її на наступному часовому інтервалі, скажімо, на $k$ спостереженнях

$$X^1_{\text{test}} = \{x_{n+1}, x_{n+2}, \ldots, x_{n+k}\}.$$

На наступному кроці навчальна вибірка розширюється на $k$ спостережень, і ми отримуємо нову навчальну вибірку $X^2_{\text{train}} = \{x_1, x_2, \ldots, x_{n+k}\}$. Знову навчаємо модель на цій розширеній вибірці та тестуємо її на наступному часовому інтервалі довжиною $k$. Таким чином, тестова вибірка не перекривається з попередніми тестовими наборами, що забезпечує незалежність оцінок.

Цей процес повторюється до тих пір, поки не буде вичерпано весь часо-

вий ряд. Такий підхід забезпечує адекватну оцінку здатності моделі прогнозувати дані, що раніше не використовувалися для навчання, зберігаючи хронологічний порядок, що дуже важливо для часових рядів [19].

Формально, для кожного кроку $i$, де $i = 1, 2, \ldots, m$, маємо

$$X^i_{\text{train}} = \{x_1, x_2, \ldots, x_{n+(i-1)k}\},$$
$$X^i_{\text{test}} = \{x_{n+(i-1)k+1}, \ldots, x_{n+ik}\},$$

де $m$ – кількість ітерацій, що залежить від розміру даних та обраної довжини тестового інтервалу $k$.

Завдяки такій методології ми можемо отримати набір оцінок помилок для кожної тестової вибірки, які потім усереднюємо для отримання узагальненої оцінки помилки моделі.

Функція втрат, така як середньоквадратична похибка (MSE), обчислюється на кожному кроці для тестових даних. Після кожної ітерації оцінюється середня помилка

$$\text{MSE}_i = \frac{1}{k}\sum_{j=1}^{k}(y_{n+(i-1)k+j} - \hat{y}_{n+(i-1)k+j})^2,$$

де $\hat{y}_{n+(i-1)k+j}$ – прогнозоване значення, отримане від моделі для відповідного тестового інтервалу.

Після всіх ітерацій обчислюється середня помилка по всіх тестових інтервалах

$$\text{MSE}_{\text{average}} = \frac{1}{m}\sum_{i=1}^{m}\text{MSE}_i.$$

Така середня помилка надає уявлення про те, наскільки модель здатна

узагальнювати дані та точно прогнозувати майбутні значення.

Перевагою методу послідовного розбиття є збереження хронологічного порядку, що робить оцінку моделі більш реалістичною для задач прогнозування часових рядів. Недоліком є те, що цей підхід може бути обчислювально складним, оскільки модель тренується кілька разів із поступовим розширенням вибірки. Проте, саме такий підхід забезпечує точну і надійну оцінку продуктивності моделі в умовах часових залежностей, що є критичним для обрання оптимальної моделі прогнозування [20].

## 2.4 Регуляризація нейронних мереж

Регуляризація є важливою методикою в навчанні нейронних мереж, особливо в задачах прогнозування часових рядів, де вона сприяє запобіганню перенавчання моделі. Перенавчання виникає, коли модель добре пристосовується до тренувальних даних, але демонструє погану узагальнюючу здатність на нових даних, що суттєво знижує її ефективність. Регуляційні методи допомагають контролювати складність моделі, обмежуючи її параметри або штучно зменшуючи кількість доступних ознак на кожній ітерації навчання. В контексті регуляризації в нейронних мережах часто застосовують два ключові методи: Dropout і L2-регуляризацію. Обидва методи мають математичне обґрунтування, що дозволяє покращити здатність моделі узагальнювати залежності в даних [21].

Dropout є однією з найпопулярніших і ефективних технік регуляризації для нейронних мереж, включаючи моделі RNN, LSTM і GRU, що використовуються для прогнозування часових рядів. Dropout виконує випадкове «вимикання» нейронів у прихованих шарах під час кожної ітерації навчання, що дозволяє створити ансамбль моделей всередині однієї архітектури. Це зменшує кореляцію між нейронами та запобігає тому, щоб модель надмірно пристосовувалася до певних зразків у тренувальних даних. Формально, для кожного нейрона $i$ в прихованому шарі вводиться випадкова змінна $d_i$, яка приймає значення 0 або

1 із заданою ймовірністю $p$, де $p$ – ймовірність вимкнення нейрона. Формально активація нейрона із застосуванням Dropout визначається наступним чином

$$h_i = d_i \cdot f\left(\sum_j w_{ij} \cdot x_j + b_i\right),$$

де $h_i$ – вихід нейрона $i$ після застосування активаційної функції;

$d_i$ – випадкова змінна, яка визначає, чи буде нейрон активний (якщо $d_i = 1$) або вимкнений (якщо $d_i = 0$);

$f$ – активаційна функція (наприклад, ReLU або $\tanh$), що застосовується до вхідного сигналу;

$w_{ij}$ – вага, яка з'єднує нейрон $i$ з нейроном $j$;

$x_j$ – вхідний сигнал від нейрона $j$;

$b_i$ – зсув нейрона $i$.

Dropout є гарною технікою регуляризації, оскільки змушує модель покладатися не на певні нейрони, а на їхні поєднання, що робить модель більш стійкою до перенавчання і покращує здатність узагальнювати дані. Під час тестування Dropout не застосовується, а значення ваг зменшуються на фактор $p$, щоб компенсувати середню кількість вимкнених нейронів під час тренування. Математично це виражається як

$$\hat{w}_{ij} = p \cdot w_{ij},$$

де $\hat{w}_{ij}$ – вага, що використовується під час тестування;

$p$ – ймовірність активації нейрона під час тренування;

$w_{ij}$ – початкова вага.

Іншим фундаментальним методом регуляризації є L2-регуляризація, яка має на меті зменшити значення ваг шляхом додавання штрафу до функції втрат

[22]. В основі цього методу лежить ідея обмеження значень ваг, що знижує схильність моделі до великого впливу окремих змінних на результат прогнозування. Математично L2-регуляризація додає до функції втрат додатковий штрафний член, пропорційний квадрату норми ваг. Формула загальної функції втрат $L_{\text{total}}$, що включає L2-регуляризацію, виглядає так

$$L_{\text{total}} = L + \frac{\lambda}{2}\sum_{i,j} w_{ij}^2 ,$$

де $L_{\text{total}}$ – загальна функція втрат з урахуванням регуляризації;

$L$ – базова функція втрат, наприклад, середньоквадратична похибка (MSE);

$\lambda$ – коефіцієнт регуляризації, який визначає ступінь штрафу за великі значення ваг;

$w_{ij}$ – вага між нейронами $i$ та $j$.

Коефіцієнт регуляризації $\lambda$ є гіперпараметром, який визначає, наскільки сильно модель буде штрафуватися за великі значення ваг. Чим менший $\lambda$, тим сильніша регуляризація, що може допомогти зменшити складність моделі і запобігти перенавчанню, але якщо $\lambda$ занадто велике, то це може призвести до недонавчання. Похідна загальної функції втрат за вагою $w_{ij}$, яка враховує L2-регуляризацію, виглядає так

$$\frac{\partial L_{\text{total}}}{\partial w_{ij}} = \frac{\partial L}{\partial w_{ij}} + \lambda w_{ij} ,$$

де $\frac{\partial L}{\partial w_{ij}}$ – похідна базової функції втрат за вагою;

$\lambda w_{ij}$ – додатковий член, що виникає через L2-регуляризацію.

Це рівняння показує, як під час оновлення ваг враховуються як стандартні градієнти, так і штраф за великі значення ваг. Завдяки цьому модель на-

вчиться обирати ваги, які одночасно мінімізують функцію втрат і не допускають великих значень, що сприяє кращій узагальнювальній здатності моделі.

Загалом, і Dropout, і L2-регуляризація спрямовані на зменшення перенавчання, але роблять це різними способами. Dropout сприяє зменшенню кореляції між нейронами, що робить модель більш стійкою до перенавчання на конкретних зразках. L2-регуляризація, у свою чергу, обмежує значення ваг, що знижує ймовірність великого впливу окремих ознак на результати моделі. У задачах прогнозування часових рядів, де важливо враховувати і короткотривалі, і довготривалі залежності, правильне поєднання цих двох методів може значно покращити якість прогнозування та здатність моделі до узагальнення.

## Висновки за розділом 2

У даному розділі було детально розглянуто теоретичні основи нейронних мереж для прогнозування часових рядів. Описано ключові архітектури, такі як RNN, LSTM та GRU, що ефективно працюють із послідовними даними, зокрема, з часовими рядами. Проаналізовано принципи функціонування цих мереж, механізми запам’ятовування та обчислення довгострокових залежностей.

Також було розглянуто метод зворотного поширення похибки через час, що лежить в основі навчання рекурентних моделей, та функції втрат, які використовуються для мінімізації похибки прогнозування. Приділено увагу регуляризації нейронних мереж, зокрема, L2-регуляризації та Dropout, які дозволяють зменшити ризик перенавчання. Також розглянуто ефективний підхід до валідації моделей прогнозування часових рядів – крос-валідацію з урахуванням послідовності даних, що забезпечує об’єктивність оцінки.

Таким чином, цей розділ надав теоретичну основу для подальшої практичної реалізації методів прогнозування та аналізу часових рядів.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1 Python як інструмент для розробки

Python є однією з найбільш використовуваною мов програмування для задач машинного навчання, аналізу даних і, зокрема, прогнозування часових рядів. Його популярність зумовлена простотою синтаксису, гнучкістю та широким спектром інструментів, які роблять його зручним для використання як досвідченими розробниками, так і початківцями. Завдяки відкритості мови Python і великій спільноті користувачів, постійно вдосконалюються існуючі інструменти та створюються нові, що полегшують реалізацію складних алгоритмів [15].

Python пропонує численні бібліотеки, які значно спрощують процес розробки нейронних мереж. Однією з ключових є TensorFlow, що дозволяє працювати з багатошаровими нейронними мережами, а також оптимізувати їх за допомогою розподілених обчислень. Важливим доповненням до TensorFlow є Keras – високорівнева бібліотека, яка надає інтуїтивно зрозумілий інтерфейс для побудови моделей. Інша корисна бібліотека, NumPy, спеціалізується на роботі з багатовимірними масивами та матрицями, які є основою для виконання числових обчислень. Наприклад, за допомогою NumPy можна швидко реалізувати операції, такі як множення матриць, що є базовою складовою в роботі нейронних мереж. Для роботи з табличними даними використовується pandas, яка також забезпечує зручну обробку часових рядів, зокрема їх нормалізацію, фільтрацію та групування за часовими індексами. Крім того, для візуалізації результатів аналізу та прогнозів широко застосовуються бібліотеки Matplotlib і Seaborn, які дозволяють створювати як базові, так і складні графіки, що забезпечує краще розуміння даних.

Для початку роботи з Python необхідно налаштувати середовище розробки. Зокрема, Python доступний для завантаження з офіційного сайту, а встановлення додаткових бібліотек здійснюється за допомогою менеджера пакетів pip.

Одним із найзручніших інструментів для написання коду є Jupyter

Notebook – інтерактивне середовище, що дозволяє комбінувати текст, код і результати його виконання. Воно є ідеальним для проведення експериментів і тестування алгоритмів. Альтернативно можна використовувати редактори коду, такі як PyCharm, що забезпечують інтегроване середовище для розробки з підтримкою налагодження, автозаповнення та інших функцій, які підвищують продуктивність роботи.

Типовий алгоритм роботи з Python для реалізації прогнозування часових рядів включає кілька основних етапів. Перш за все, необхідно завантажити та обробити дані. Для цього використовуються бібліотеки pandas і NumPy, які дозволяють здійснювати нормалізацію даних, усунення пропусків та формування вхідних послідовностей для нейронної мережі. Після цього дані поділяються на тренувальний та тестовий набори, що є важливим для перевірки узагальнюючої здатності моделі. Наступним етапом є побудова архітектури моделі, яка може бути реалізована за допомогою Keras. Наприклад, нейронна мережа LSTM створюється шляхом додавання відповідних шарів до послідовної моделі. У кінці здійснюється навчання моделі методом зворотного розповсюдження похибки з використанням функції втрат, такої як середньоквадратична помилка. Останнім етапом є оцінка результатів за допомогою візуалізації, побудови графіків, які показують різницю між реальними та прогнозованими значеннями.

Таким чином, Python є універсальним інструментом для розробки та впровадження моделей машинного навчання, що забезпечує повний цикл розробки від попередньої обробки даних до аналізу результатів. Його гнучкість, багатофункціональність і великий набір бібліотек роблять його незамінним для роботи з часовими рядами.

## 3.2 Обробка та підготовка даних

Обробка даних є ключовим етапом у розробці моделей прогнозування часових рядів. Вона включає очищення, перетворення, нормалізацію та структу-

ризацію даних таким чином, щоб модель могла ефективно навчатися та робити точні прогнози. У цьому розділі детально описується, як підготувати дані для роботи з нейронними мережами.

Першим кроком є завантаження даних. Сирі дані часто мають неоднорідний формат, який необхідно стандартизувати. Для цього потрібно зчитати файл із даними, незалежно від його формату (CSV, Excel тощо), і замінити некоректні символи. Зокрема, десяткові розділювачі замінюються з коми на крапку, щоб забезпечити коректність обчислень. Після цього всі числові стовпці конвертуються у формат із плаваючою точкою, що забезпечує коректність математичних операцій.

Далі, обробка даних передбачає приведення часової ознаки до формату «datetime». Це забезпечує коректність маніпуляцій із датами, наприклад, розрахунок інтервалів або виділення сезонних трендів.

Наступним етапом є перевірка на наявність пропущених значень. Важливо виявити стовпці з відсутніми даними, щоб запобігти помилкам під час роботи моделі. Пропущені дані можуть бути заповнені різними способами, залежно від контексту. Наприклад, використання середнього значення, медіани або спеціальних методів екстраполяції. Після цього необхідно провести попередній аналіз даних, щоб оцінити розподіл числових ознак. Цей аналіз дозволяє виявити потенційні аномалії або невідповідності.

Наступним кроком є нормалізація числових ознак. Оскільки різні стовпці можуть мати значення в різних діапазонах, це може негативно вплинути на навчання. Нормалізація перетворює дані в діапазон $[0,1]$, що прискорює збіжність моделі. Далі формується набір ознак та цільовий показник. Ознаками є вибрані стовпці, які найбільше корелюють із цільовим показником – для розв’язуваної у роботі задачі ним є загальний рівень напруги BQ. Створення таких наборів є важливим, щоб виділити лише релевантну інформацію для моделі.

Для ефективного навчання моделі дані розділяються на тренувальний і тестовий набори. Це дозволяє моделі спочатку навчатися, а потім перевіряти свої прогнози на нових даних. Зазвичай для тестового набору виділяється

20 - 30% даних. У нашому випадку використовується 20% наявних даних.

Таким чином, обробка даних є комплексним процесом, що передбачає кілька етапів: завантаження даних, перевірку на коректність, нормалізацію, формування наборів ознак та цільових значень, а також розділення на тренувальний і тестовий набори. Ці дії є обов'язковими для підготовки даних до машинного навчання, зокрема для побудови моделей нейронних мереж.

## 3.3 Опис реалізованої програми

Розроблена програма для прогнозування напруги в гідроелектричній мережі є універсальним інструментом, що включає в себе алгоритми обробки даних, побудови нейронних мереж та оцінки їхньої точності. Її структура побудована так, щоб забезпечити максимальну ефективність роботи з часовими рядами та відповідати сучасним вимогам до моделей машинного навчання.

На першому етапі роботи програма завантажує дані з текстового файлу формату CSV. Усі вхідні дані проходять етап попередньої обробки. Це включає автоматичну заміну ком для дробових чисел на крапки, що дозволяє коректно конвертувати їх у числовий тип. Усі стовпці, які містять числові дані, переводяться у формат «float», що є стандартним для математичних обчислень. Крім того, програма обробляє колонку з датами, перетворюючи її у формат «datetime», що значно спрощує аналіз часових інтервалів. Особлива увага приділяється перевірці пропущених значень. Програма автоматично обчислює кількість таких пропусків у кожній колонці, і за необхідності вони або заповнюються, або видаляються.

Наступний етап передбачає нормалізацію числових даних, що є критично важливим для стабільної роботи нейронних мереж. У роботі використовується метод мінімаксної нормалізації, який зводить усі значення до діапазону $[0,1]$. Це забезпечує рівномірний масштаб ознак і попереджає проблему домінування показників із великим діапазоном значень.

Далі формуються навчальні й тестові вибірки. Для цього обираються ключові ознаки, що впливають на прогнозування, такі як метеорологічні показники та значення напруги. Вибрані ознаки об'єднуються в набір $X$, а цільовий показник – у вектор $y$. Для перевірки якості моделі дані розподіляються на тренувальну та тестову частини у співвідношенні 80% до 20%. Такий підхід дозволяє уникнути перенавчання та забезпечує об'єктивну оцінку точності.

Створення та навчання моделі реалізовано за допомогою архітектур нейронних мереж LSTM або GRU, залежно від обраної конфігурації. Спочатку ініціалізуються параметри моделі, зокрема кількість шарів, кількість нейронів на кожному шарі та функції активації. Потім вибудовується архітектура мережі, яка дозволяє обробляти послідовні дані. Навчання відбувається шляхом мінімізації функції втрат за допомогою алгоритму оптимізації. Функція втрат визначається як середньоквадратична похибка

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

де $y_i$ – реальні значення;

$\hat{y}_i$ – прогнозовані моделлю значення;

$N$ – кількість спостережень.

Заключний етап роботи програми включає оцінку точності моделі. Для цього використовуються метрики, такі як середньоквадратична похибка та середня абсолютна похибка, які дозволяють об'єктивно оцінити результати прогнозування. Крім того, програма будує графіки, які порівнюють реальні значення з прогнозованими, що дає можливість візуально оцінити якість роботи моделі та виявити можливі недоліки.

Розроблена програма інтегрує всі необхідні етапи аналізу часових рядів, забезпечуючи високу точність і зручність використання. Її структура дозволяє легко адаптуватися до нових задач, додаючи нові модулі або модифікуючи існуючі алгоритми.

## Висновки за розділом 3

У цьому розділі було описано програмну реалізацію прогнозування напруги в гідроелектричній мережі з використанням методів машинного навчання. Було виконано огляд мови програмування Python з точки зору можливостей її застосування для машинного навчання та побудови нейронних мереж. Розглянуто алгоритмічні аспекти побудови нейронних мереж LSTM і GRU, що є основними інструментами прогнозування часових рядів у цьому дослідженні. Детально розглянуто етапи попередньої обробки та підготовки даних, які включають очищення, нормалізацію та формування вибірок для навчання й тестування моделі. Було забезпечено коректне перетворення даних у формати, придатні для аналізу, що є критично важливим етапом для успішного навчання нейронних мереж. Особливу увагу приділено інтеграції всіх компонентів у єдину програму, яка забезпечує автоматизацію всіх етапів – від обробки даних до оцінки точності моделі.

Даний розділ сформував практичну основу для дослідження, надавши повноцінний інструментарій для виконання завдання прогнозування та підтвердив ефективність теоретичних підходів, описаних у попередньому розділі.

# 4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ ТА ЇХ АНАЛІЗ

## 4.1 Опис експериментальних умов

Дані, що використовувалися в експериментах у даній роботі, мають структуру часових рядів і включають значення напруги (BQ), що є цільовим показником, а також набір метеорологічних ознак: температуру (T), хмарність (N), швидкість вітру (V) та тиск (P) з п’яти метеостанцій. Часові інтервали між спостереженнями рівномірні, що забезпечує зручність моделювання часових залежностей.

Набір даних включає кілька тисяч спостережень, охоплюючи період понад три роки, що дозволяє враховувати як довгострокові тренди, так і сезонні коливання. Окремі метеорологічні показники були представлені у вигляді числових значень, які в процесі підготовки даних нормалізувалися до діапазону $[0,1]$, щоб уникнути домінування змінних із великими значеннями. Цільовий показник BQ також був нормалізований, що забезпечило стабільну роботу моделей машинного навчання.

Перед моделюванням та оцінкою різних нейронних мереж для прогнозування напруги було вирішено побудувати кілька ключових графіків для глибшого розуміння структури даних. Ці графіки дозволяють оцінити тренди, залежності між ознаками, сезонність та періодичність. Вибір графіків був обумовлений специфікою часових рядів і необхідністю виявити взаємозв’язки між змінними, що впливають на цільовий показник.

Першим етапом аналізу є графік тренду загальної напруги (BQ) у часі (рис. 4.1). Він дозволяє оцінити, як змінюється напруга протягом усього періоду спостереження. Цей графік важливий для виявлення загальних тенденцій, можливих аномалій і сезонних коливань, які можуть бути критичними для прогнозування. Візуалізація показала, що BQ має чітко виражені коливання, що свідчить про необхідність врахування сезонності в моделі.

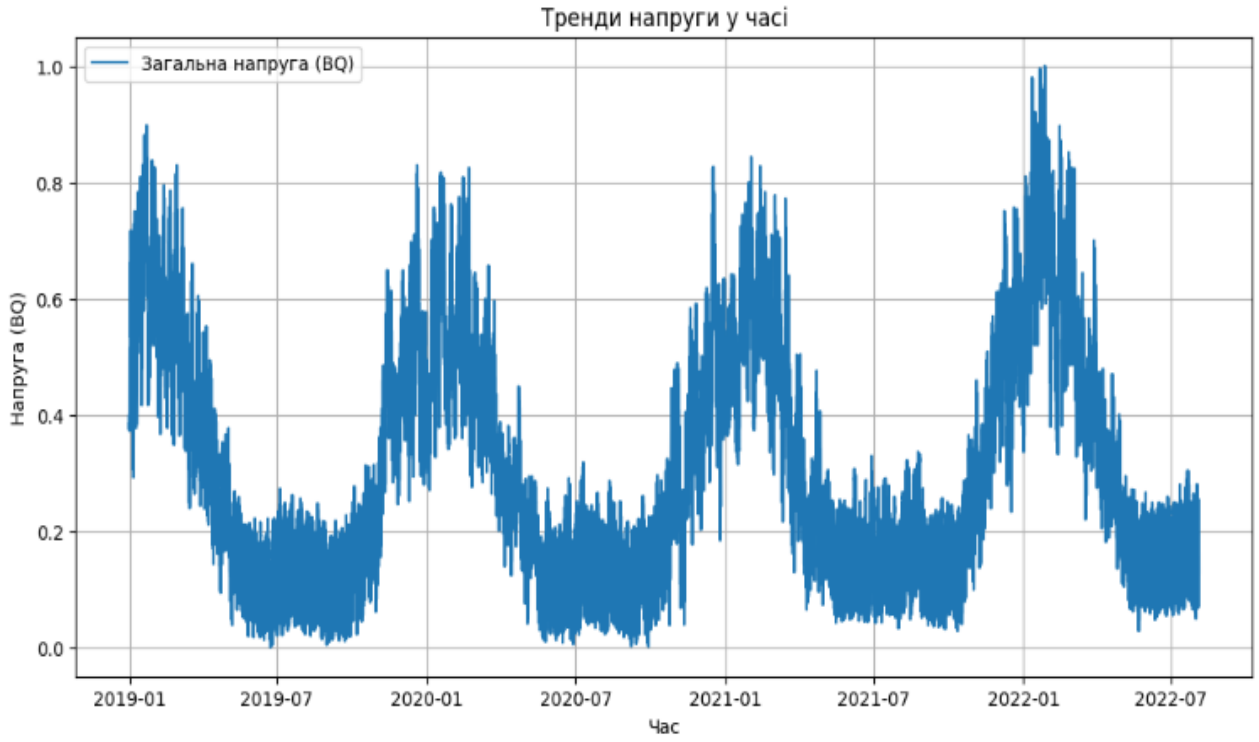


Рисунок 4.1 – Тренди напруги у часі

На другому кроці було побудовано кореляційну матрицю між усіма числовими ознаками, включаючи метеорологічні показники та напругу (рис. 4.2). Ця матриця дозволила виявити, які змінні найбільше впливають на BQ. Висока кореляція між BQ та певними метеорологічними показниками, такими як температура (T), підтвердила доцільність їх використання як ключових ознак у моделі.

Виходячи з цього висновку, додатково було досліджено залежність між температурою (Tyul) та напругою (BQ) за допомогою діаграми розсіювання (рис. 4.3). Цей графік надав можливість оцінити, чи присутній лінійний або нелінійний зв’язок між температурою та напругою. Виявлено, що зміна температури дійсно впливає на напругу, хоча цей вплив може змінюватися залежно від інших факторів.

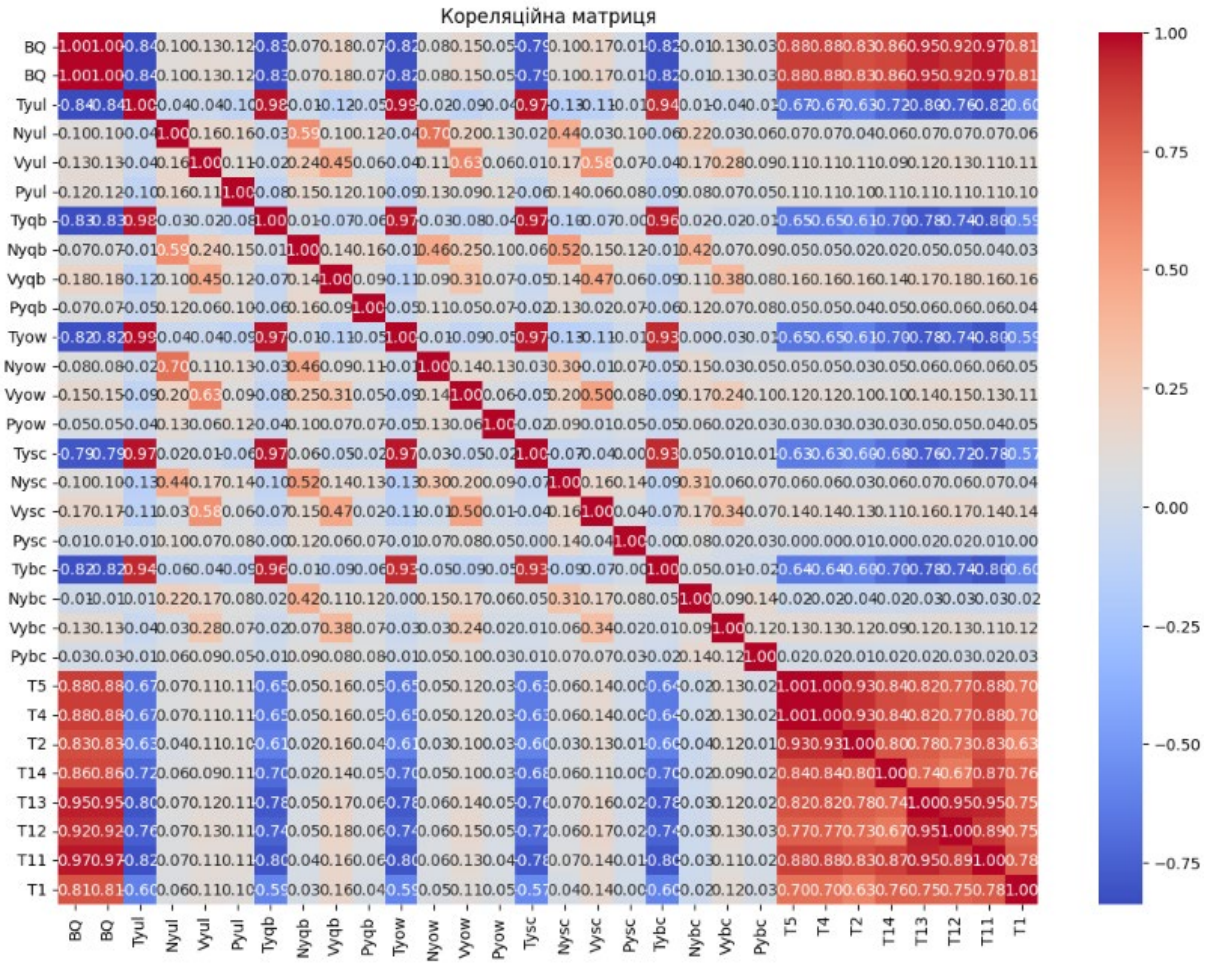


Рисунок 4.2 – Кореляційна матриця

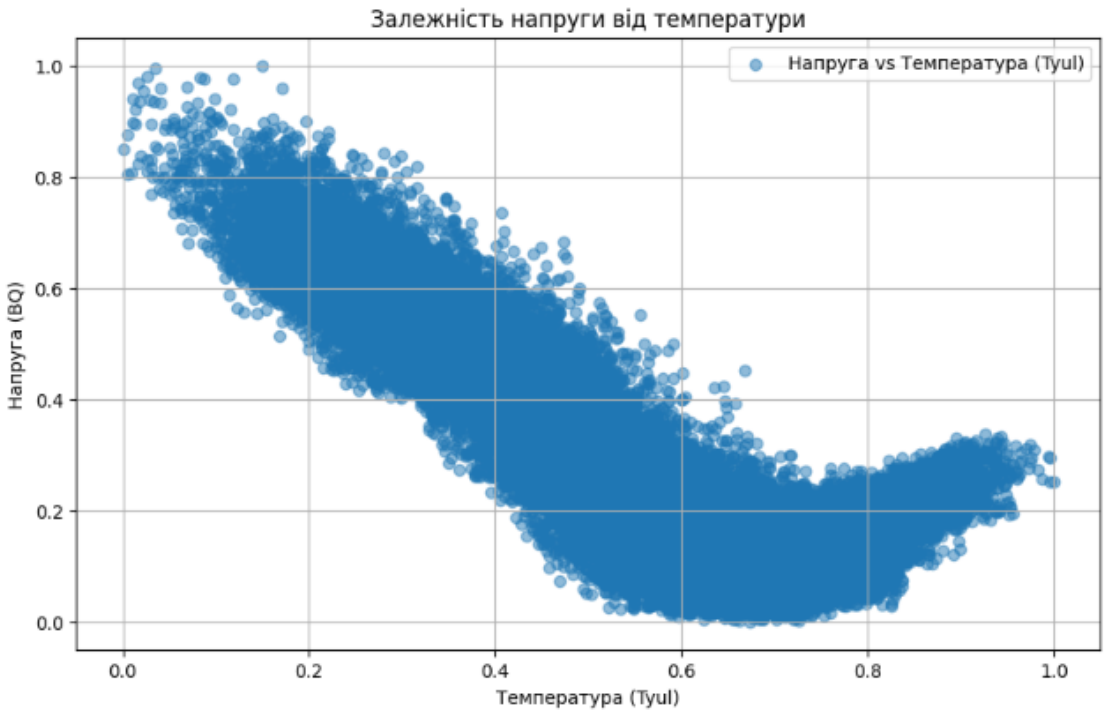


Рисунок 4.3 – Залежність напруги від температури

Щоб оцінити сезонні та довгострокові залежності, було побудовано графік автокореляції напруги (BQ) зображений на рисунку 4.4. Автокореляція показала, що BQ має чітко виражену періодичність, а також залежність від попередніх значень на певних часових зсувах. Ці результати підтвердили необхідність використання моделей, які враховують часові залежності, таких як LSTM або GRU.

Ми побудували графік ковзного середнього, зображений на рисунку 4.5, що дозволив згладити коливання та виявити загальні тренди напруги. Цей підхід є важливим для оцінки базових змін напруги без впливу короткострокових коливань. Графік підтвердив, що BQ має як загальний тренд, так і циклічність.

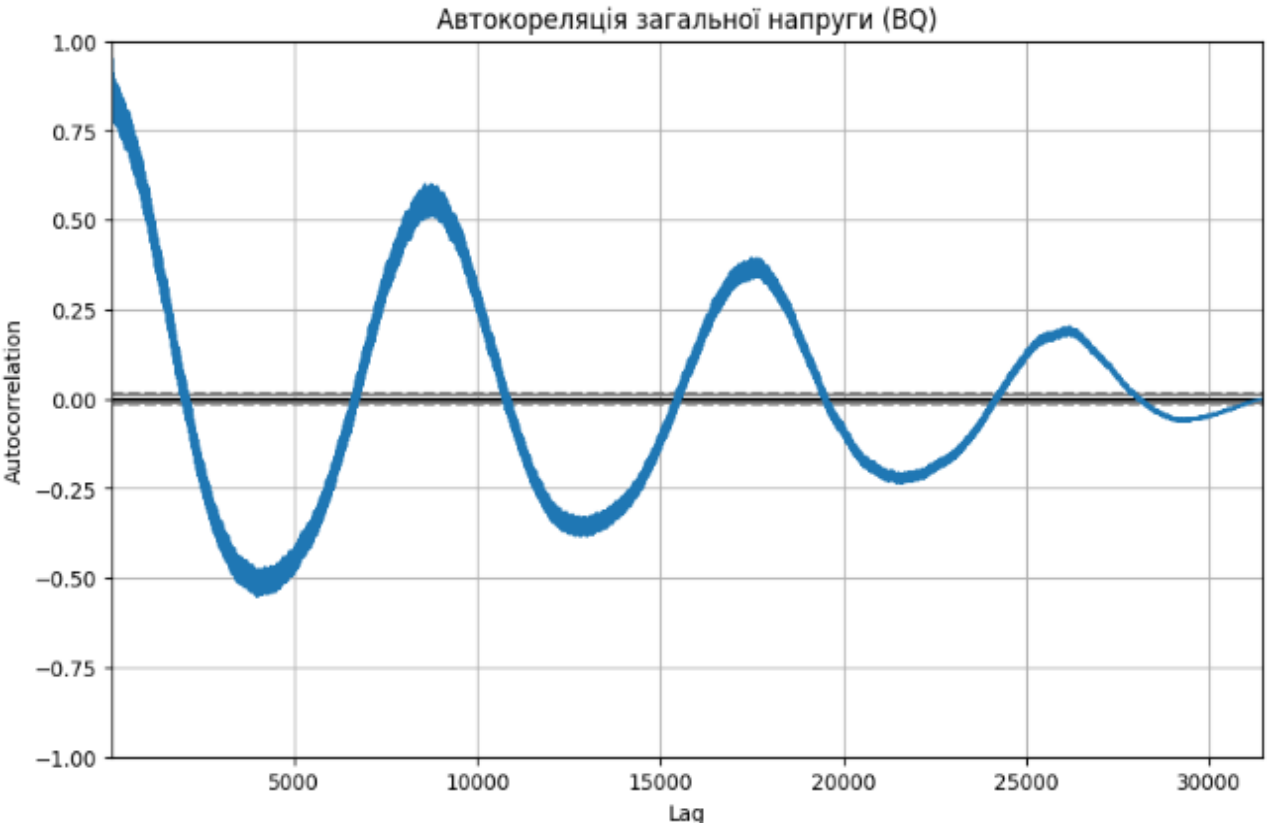


Рисунок 4.4 – Автокореляція загальної напруги


Рисунок 4.5 – Графік ковзного середнього

Окрім цього було проведено аналіз залежності напруги від часу доби за допомогою графіку розподілу середньої напруги по годинах доби. Ця візуалізація дозволила визначити, чи є пікові значення в певний час доби. Виявлено, що напруга може змінюватися залежно від часу, що свідчить про також і добову сезонність (рис. 4.6).


Рисунок 4.6 – Розподіл напруги залежно від години доби

Після побудови цих графіків було сформульовано висновки про ключові фактори, що впливають на напругу. На основі виявлених закономірностей і залежностей було обрано нейронні мережі LSTM та GRU. Додатково було розглянуто моделі на основі Transformer та TCN, що дозволить підтвердити або спростувати висновки щодо доцільності застосування рекурентних нейронних мереж для прогнозування часових залежностей напруги. Кожна модель враховує особливості даних, які були виявлені під час візуалізації. У наступних етапах ці моделі були оцінені за точністю прогнозування та стабільністю роботи.

## 4.2 Обґрунтування вибору моделі

Для оцінки ефективності різних архітектур нейронних мереж було проведено серію обчислювальних експериментів. Усі моделі були навчені на одному і тому ж наборі даних, який містив часові ряди напруги (BQ) та метеорологічні показники, зокрема температуру, хмарність, швидкість вітру та тиск, отримані з п’яти метеостанцій. Дані попередньо були розділені на тренувальну вибірку (80%) та тестову вибірку (20%), що забезпечило об’єктивність оцінки моделей і дозволило перевірити їхню здатність до узагальнення.

Під час навчання нейронних мереж, зображеного на рисунку 4.7, використовувались однакові гіперпараметри, щоб забезпечити коректне порівняння. Зокрема, моделі навчались протягом 50 епох із розміром батчу 32, що дозволило збалансувати обчислювальну складність та ефективність навчання. Як алгоритм оптимізації було обрано метод Adam, який демонструє стабільну роботу для задач прогнозування. Для оцінки похибки використовувалась середньоквадратична похибка (MSE), яка є стандартним критерієм у регресійних задачах. У межах експерименту було реалізовано та протестовано чотири архітектури нейронних мереж: LSTM, GRU, Transformer і TCN.

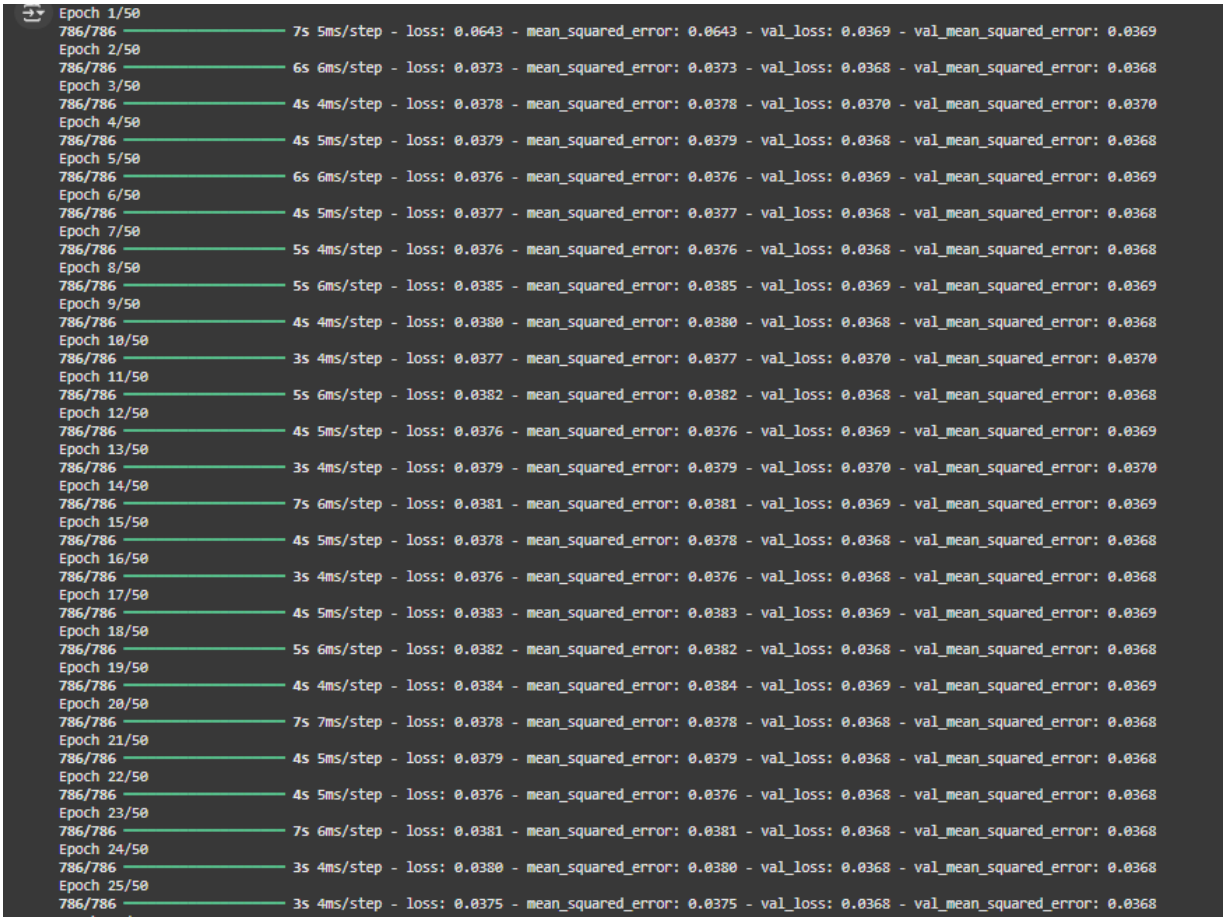


Рисунок 4.7 – Процес навчання нейронних мереж

LSTM була обрана через її здатність ефективно обробляти довгострокові залежності в часових рядах, що є важливим для аналізу даних напруги у гідроелектричній мережі. Аналіз трендів напруги (BQ) у часі та графік автокореляції показали, що в даних присутні чіткі періодичні коливання та довгострокові тренди. Це свідчить про те, що значення напруги залежить від попередніх спостережень на великих часових інтервалах.

LSTM ідеально підходить для такої задачі, оскільки її архітектура з використанням комірок пам’яті дозволяє зберігати інформацію про важливі події в даних навіть на віддалених кроках. Використання двох шарів із 50 нейронами кожен дає змогу моделі виявляти як короткострокові, так і довгострокові залежності, що було підтверджено сезонними трендами, виявленими під час візуалізації.

Використання GRU, як спрощеної альтернативи LSTM, обумовлюється здатністю моделі ефективно працювати з часовими рядами, але з меншою об-

числювальною складністю. Графік автокореляції показав, що в даних присутні довгострокові залежності, однак вони не настільки виражені, як сезонні коливання. Це робить GRU доцільним вибором для задачі, де швидкість навчання і оптимізація ресурсів мають важливе значення.

Два шари з 50 нейронами дозволяють моделі виявляти взаємозв'язки між змінними на різних рівнях складності. Завдяки меншій кількості параметрів, GRU є ефективнішою для обробки великих наборів даних, таких як метеорологічні показники й напруга. Вибір цієї моделі також обґрунтований її здатністю уникати перенавчання за рахунок оптимізованої архітектури.

Для порівняння якості прогнозування за допомогою LSTM та GRU було додатково було розглянуто моделі на основі Transformer та TCN. Трансформер був обраний через його здатність враховувати глобальні залежності між значеннями в часових рядах. Графіки кореляційної матриці виявили, що напруга BQ залежить від багатьох змінних, таких як температура та хмарність, які впливають на напругу не тільки в короткостроковій перспективі, але й у більш широкому часовому діапазоні.

Механізм багатоголової уваги в трансформері дозволяє моделі фокусуватися на різних аспектах даних одночасно, що є критично важливим для врахування складних взаємозв'язків. Використання трансформера особливо виправдане для аналізу напруги, де взаємодія між змінними має нелінійний характер. Крім того, трансформер демонструє хороші результати в задачах прогнозування завдяки можливості ефективно обробляти послідовності будь-якої довжини.

TCN була обрана через її здатність ефективно обробляти локальні залежності в часових рядах. Графік залежності напруги від температури (Tyul) показав, що напруга суттєво змінюється залежно від метеорологічних умов у короткі часові інтервали. Це свідчить про те, що локальні залежності мають значний вплив на цільовий показник.

TCN використовує каузальні згорткові шари, які дозволяють обробляти лише попередні дані для прогнозування. Завдяки цьому, модель забезпечує високу ефективність при аналізі короткострокових взаємозв'язків у даних. Три

шари з різними коефіцієнтами дилатації дозволяють TCN адаптуватися до різних масштабів залежностей, що робить її хорошим вибором для задач із вираженою сезонністю та впливом метеорологічних факторів.

Вибір цих чотирьох архітектур обґрунтований різними характерними особливостями даних. LSTM і GRU були обрані для моделювання довгострокових залежностей, виявлених у трендах і автокореляції, тоді як Transformer враховує складні взаємозв'язки між змінними на глобальному рівні. TCN, зі свого боку, ефективно аналізує короткострокові взаємозв'язки, що є важливим для задач із локальними сезонними впливами. Разом ці моделі дозволяють повноцінно оцінити залежності та закономірності в даних, щоб зробити точний і надійний прогноз.

## 4.3 Результати обчислювального експерименту

### 4.3.1 Аналіз результатів прогнозування за допомогою нейронних мереж

В процесі обчислювального експерименту було протестовано чотири моделі нейронних мереж: LSTM, GRU, Transformer і TCN. Кожна з моделей навчалася на тренувальних даних, після чого її продуктивність була перевірена на тестовому наборі. Для оцінки використовувалася метрика середньоквадратичної похибки (MSE), яка є стандартною для задач регресії. На рисунку 4.8 наведено результати тестової похибки для кожної моделі

```
197/197 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 1.1657e-04 - mean_squared_error: 1.1657e-04
197/197 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 1.0141e-04 - mean_squared_error: 1.0141e-04
197/197 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0368 - mean_squared_error: 0.0368
197/197 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 2.5010e-04 - mean_squared_error: 2.5010e-04
LSTM Test Loss: 0.00011382919183233753
GRU Test Loss: 0.0001041150899254717
Transformer Test Loss: 0.0368121899664402
TCN Test Loss: 0.0002594463003333658
```

Рисунок 4.8 – Результати обчислювального експерименту

Результати показують, що LSTM та GRU продемонстрували найкращі показники точності серед усіх моделей. Модель LSTM змогла успішно врахувати як короткострокові, так і довгострокові залежності в часових рядах. Низька тестова похибка свідчить про те, що LSTM здатна ефективно моделювати навіть складні закономірності в даних. GRU, яка є спрощеною версією LSTM і має аналогічну архітектуру з двома шарами по 50 нейронів, показала ще кращий результат з меншою тестовою похибкою. Це підтверджує, що GRU може бути більш ефективною при роботі з задачами, де важлива оптимізація обчислювальних ресурсів, оскільки її архітектура має меншу кількість параметрів.

Трансформерна модель показала значно вищу тестову похибку, хоча трансформери часто демонструють високу продуктивність у задачах з великими наборами даних і складними взаємозв’язками, у цьому випадку їх продуктивність виявилася гіршою. Це може бути пов’язано з відносно обмеженим обсягом даних та специфікою часових рядів, де локальні й короткострокові залежності є більш критичними. Крім того, трансформери зазвичай потребують додаткової тонкої настройки гіперпараметрів для досягнення високої точності.

TCN добре враховує локальні залежності в даних і показала прийнятну точність, однак поступилася LSTM і GRU. Менш виражена здатність TCN до моделювання довгострокових залежностей могла стати причиною її слабшої продуктивності в порівнянні з рекурентними мережами.

Графік навчання і валідації для всіх моделей, зображений на рисунку 4.9, демонструє, що LSTM і GRU мали стабільний процес навчання без суттєвого перенавчання, оскільки тренувальна та валідаційна похибки залишались близькими. У той же час, трансформер і TCN мали дещо вищі валідаційні похибки, що вказує на їхню менш ефективну узагальнюючу здатність для даних цієї задачі.

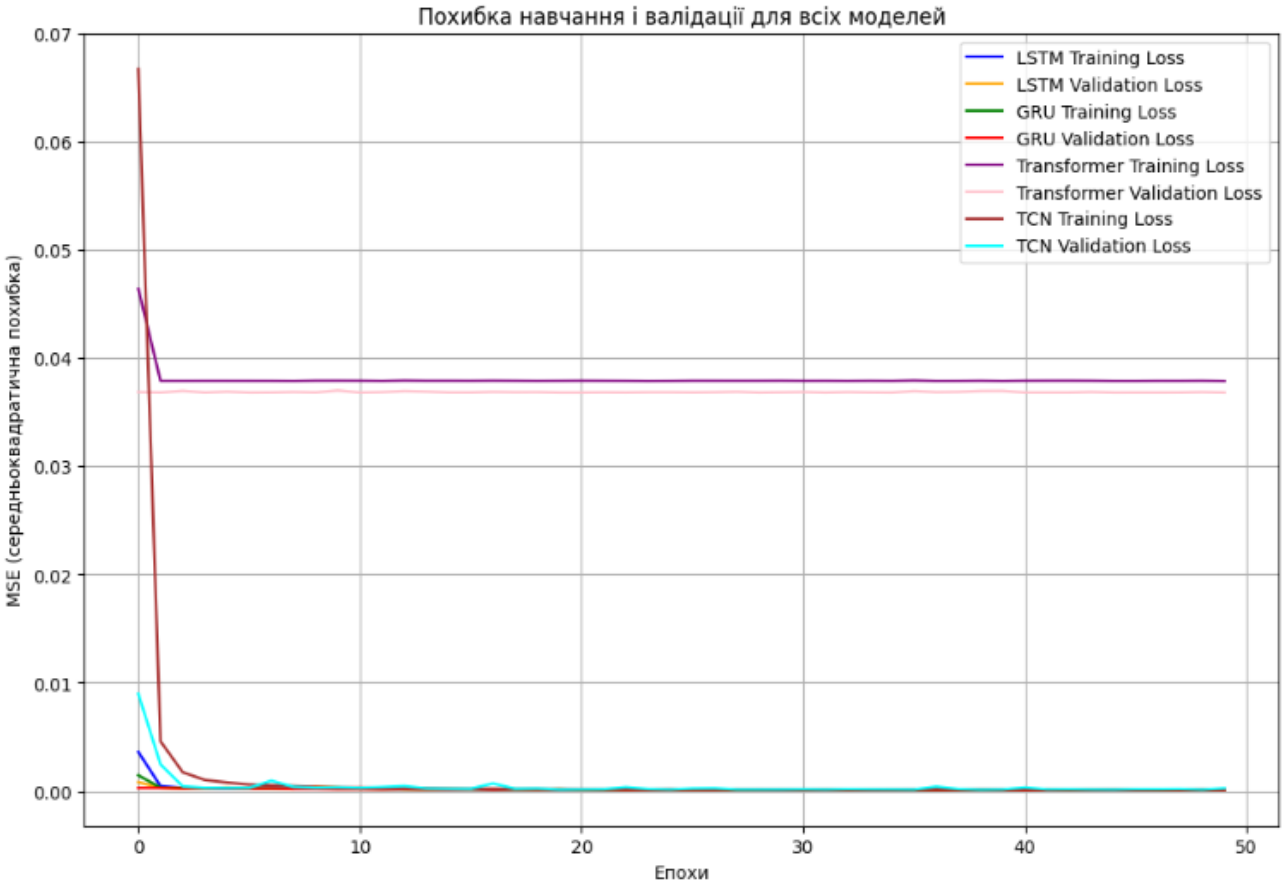


Рисунок 4.9 – Похибка навчання і валідації для всіх моделей

### 4.3.2 Порівняння результатів нейромережевих та статистичних методів прогнозування

Порівняння результатів цього дослідження з попередньою бакалаврською роботою демонструє суттєве покращення точності прогнозування напруги у гідроелектричній мережі. У попередньому дослідженні використовувалися традиційні лінійні моделі, зокрема авторегресійна модель, яка враховувала сезонні залежності через параметри step = 24 та lags = 3. Оцінка точності цих моделей показала середньоквадратичну похибку (MSE) на рівні приблизно 3, що, хоча і є прийнятним результатом для базових моделей, не забезпечує достатньої точності для прогнозування напруги в реальних умовах роботи мережі.

У даній роботі завдяки використанню сучасних технологій прогнозування, а саме, нейронних мереж, таких як LSTM та GRU, вдалося суттєво знизити

похибку. Найкращі моделі показали тестову похибку MSE менше одної десятитисячної, що означає покращення точності прогнозу у десятки тисяч разів у порівнянні з попередньою моделлю. Такий стрибок у продуктивності пояснюється здатністю нейронних мереж враховувати як довгострокові, так і короткострокові залежності в даних, обробляти складні нелінійні взаємозв'язки між змінними та адаптуватися до різних масштабів сезонних змін.

Важливим аспектом цього покращення є використання метеорологічних показників, таких як температура, хмарність, швидкість вітру та тиск, як додаткових ознак у моделі. Тоді як у попередній роботі побудовані авторегресійні моделі не змогли ефективно врахувати інформацію про метеорологічні показники для підвищення якості прогнозування напруги, і основний акцент робився на сезонності самого ряду напруги, в цьому дослідженні нейронні мережі гарно враховують зовнішні фактори, які впливають на формування трендів та короткострокових коливань. Це дозволило досягти більш точного та узагальнюючого прогнозу.

Також суттєвий внесок у покращення зробила здатність нейронних мереж працювати з великим обсягом даних. У той час як авторегресійна модель обмежується невеликим числом лагів, LSTM та GRU можуть зберігати інформацію про значно довші часові інтервали, що є критично важливим для аналізу залежностей у гідроелектричних мережах.

Зниження похибки у прогнозуванні напруги має значний вплив на практичне застосування. Якщо попередні результати могли лише частково допомогти у прийнятті рішень щодо роботи трансформаторів, то розроблені моделі нейронних мереж забезпечують високу точність прогнозів, що дозволяє значно підвищити стабільність роботи мережі. Це, у свою чергу, сприяє зменшенню ризиків перевантажень або відмов обладнання, що є критично важливим для гідроелектростанцій.

Підсумовуючи результати аналізу виконаних досліджень, можна стверджувати, що перехід від статистичних лінійних моделей до сучасних комп’ютерних методів прогнозування за допомогою нейронних мереж дозволив

зробити суттєвий крок уперед у задачі прогнозування напруги об’єктів критичної інфраструктури. Отримані результати підтверджують, що використання методів машинного навчання дозволяє досягти значно вищої точності та забезпечує більш ефективне управління енергомережами. У перспективі це відкриває можливість для подальшої інтеграції таких моделей у реальні системи моніторингу та управління.

## Висновки за розділом 4

На основі отриманих результатів можна зробити висновок, що LSTM і GRU є найкращими моделями для задачі прогнозування напруги у гідроелектричній мережі. Їхня здатність ефективно обробляти як короткострокові, так і довгострокові залежності дозволила досягти найнижчої тестової похибки серед усіх моделей. GRU, показавши найнижчий результат похибки, може вважатися оптимальним вибором завдяки своїй ефективності та меншій обчислювальній складності. LSTM, своєю чергою, є надійним варіантом для ситуацій, де важлива додаткова стабільність у моделюванні складних залежностей. Моделі Transformer і TCN, хоча й мають свої переваги, поступаються за точністю та стабільністю LSTM і GRU у контексті даної задачі.

# ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розглянуто задачу прогнозування напруги гідроелектричної мережі за допомогою методів машинного навчання, зокрема нейронних мереж. Основна мета дослідження полягала у створенні моделі, здатної з високою точністю прогнозувати значення напруги на основі історичних даних та метеорологічних показників, що сприятиме забезпеченню стабільної роботи енергомережі. Для досягнення цієї мети було поставлено та виконано кілька завдань.

По-перше, було проведено аналіз предметної області, в ході якого детально досліджено природу задачі та особливості часових рядів. Було визначено, що напруга у гідроелектричних мережах залежить від сезонних факторів, таких як температура, хмарність, швидкість вітру та тиск. Це обґрунтувало необхідність використання моделей, здатних враховувати як короткострокові, так і довгострокові залежності.

По-друге, було реалізовано повний процес підготовки даних. Проведено очищення, нормалізацію та формування часових вікон для обробки моделей. Завдяки попередній візуалізації та аналізу даних були виявлені ключові закономірності, такі як тренди, сезонність та автокореляція, що стали основою для вибору архітектур нейронних мереж.

По-третє, в рамках роботи було обрано та реалізовано моделі LSTM та GRU. Для порівняльної оцінки їх ефективності експерименти додатково було виконано за допомогою моделей Transformer та TCN. Кожна модель тестувалась на однакових даних із застосуванням єдиних критеріїв оцінки. За результатами обчислювальних експериментів найкращі показники точності продемонстрували LSTM і GRU, зокрема GRU досягла найнижчої тестової похибки. Це свідчить про її здатність ефективно обробляти часові ряди та робити точні прогнози при відносно низьких обчислювальних витратах. Водночас LSTM також показала високу ефективність і може бути рекомендована для задач, що вимагають моделювання складніших довгострокових залежностей.

Створена модель нейронної мережі є важливим кроком до автоматизації прогнозування напруги в гідроелектричних мережах. Її впровадження здатне значно підвищити ефективність управління енергомережами, забезпечити більш стабільну роботу трансформаторів і знизити ризики перенапруги або відмови в мережі. Завдяки високій точності прогнозів, запропонована модель може бути інтегрована в системи моніторингу та управління енергомережами, що сприятиме оптимізації їхньої роботи.

Проте ця робота відкриває нові перспективи для подальших досліджень. У майбутньому можна дослідити використання більш складних трансформерних архітектур, які можуть бути ефективнішими за умови більших обсягів даних. Крім того, можна вивчити гібридні підходи, які поєднують переваги LSTM, GRU та TCN, або впровадити індивідуальні архітектури для різних типів сезонності чи часових інтервалів. Іншим перспективним напрямком є інтеграція моделей машинного навчання з методами оптимізації для автоматичного управління режимами роботи гідроелектростанцій у реальному часі.

Отже, проведене дослідження дозволило розв’язати поставлену задачу прогнозування напруги гідроелектричної мережі та надало основу для створення ефективної системи прогнозування. Результати роботи можуть бути застосовані як для оптимізації роботи гідроелектростанцій, так і для розробки більш складних моделей, орієнтованих на різноманітні завдання енергетичної сфери.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ляшенко Є. С. Прогнозування напруги електричної мережі гідроелектростанції методами машинного навчання. *ІІІ Міжнародна науково-практична конференція англійською мовою*: зб. матеріалів форуму (м. Харків, 08 листопада 2024 р.). Т. 7. Харків : ХНПУ, 2024. С. 227.

2. Жлуктенко В. І., Наконечний С. І., Савіна С. С. Теорія ймовірностей і математична статистика. У 2 ч. Ч. ІІ. Математична статистика. Київ : КНЕУ, 2001. 336 с.

3. Турчин В. М. Теорія ймовірностей і математична статистика. Дніпропетровськ : ІМА-прес, 2014. 556 с.

4. Медведєв М. Г., Пащенко І. О. Теорія ймовірностей та математична статистика. Київ : Ліра-К, 2015. 536 с.

5. Олійник А. О., Субботін С. О., Олійник О. О. Інтелектуальний аналіз даних. Запоріжжя : ЗНТУ, 2012. 278 с.

6. Огірко О. І., Галайко Н. В. Теорія ймовірностей та математична статистика. Львів : ЛьвДУВС, 2017. 292 с.

7. Bruce P., Bruce A., Gedeck P. Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python. Springfield : O'Reilly Media, 2020. 360 p.

8. Vargas L. G., Saaty T. L. Decision Making with the Analytic Network Process: Economic, Political, Social and Technological Applications with Benefits, Opportunities, Costs and Risks. London : Springer, 2006. 248 p.

9. Rawlings J. O. Applied Regression Analysis: A Research Tool. 2nd ed. New York : Springer, 1998. 657 p.

10. Azen S. P., Afifi A. A. Statistical Analysis: a Computer Oriented Approach. Elsevier Science & Technology Books, 2014. 328 p.

11. Metcalfe A. V., Cowpertwait P. S. P. Introductory Time Series with R. Springer, 2009. 272 p.

12. Raschka S., Mirjalili V. Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-learn, and Tensorflow, 2nd Edition. Birmingham :

Packt Publishing, 2017. 622 p.

13. Müller A. C., Guido S. Introduction to Machine Learning with Python: A Guide for Data Scientists. Springfield : O'Reilly Media, Incorporated, 2018. 398 p.

14. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. Springfield : O'Reilly Media, 2019. 856 p.

15. Chollet F. Deep Learning with Python. Shelter Island, New York : Manning Publications Co., 2017. 384 p.

16. Chollet F. Deep Learning with Python. 2nd ed. Shelter Island, New York : Manning Publications Co., 2021. 504 p.

17. Ravichandiran S. Hands-On Reinforcement Learning with Python. Birmingham : Packt Publishing, 2019. 761 p.

18. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge, MA : MIT Press, 2016. 800 p.

19. Jason Brownlee, Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras. URL: https://machinelearningmastery.com/time-seriesprediction-lstm-recurrent-neural-networks-python-keras/ (дата звернення: 16.09.2024).

20. Schmidhuber J., Wierstra D., Gomez F. Evolution: Hybrid Neuroevolution. *Optimal Linear Search for Sequence Learning in Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 2005. P. 466−477.

21. Goodman D, Brette. R. Brian: a simulator for spiking neural networks in Python. *Front, Neuroinform* 2, 2008. P. 5.

22. Rai P., Rai K. Comparison of Stock Prediction Using Different Neural Network Types. *International Journal of Advanced Engineering & Application.* 2011. №1. P. 157−160.

23. Project Jupyter. URL: https://jupyter.org/ (дата звернення: 02.12.2024).

24. NumPy. URL: https://numpy.org/ (дата звернення: 02.12.2024).

25. Pandas – Python Data Analysis Library. URL: https://pandas.pydata.org/ (дата звернення: 02.12.2024).

26. Matplotlib – Visualization with Python. URL: https://matplotlib.org/ (дата звернення: 02.12.2024).

27. Scikit-learn: machine learning in Python – scikit-learn 1.2.2 documentation. URL: https://scikit-learn.org/stable/ (дата звернення: 02.12.2024).

28. Seaborn: statistical data visualization – seaborn 0.12.2 documentation URL: https://seaborn.pydata.org/ (дата звернення: 02.12.2024).