Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

«Затверджую»
в.о. завідуючого кафедри
комп'ютерних систем та робототехніки
_____ к. ф.-м. н., доцент Максим Хруслов
«___» грудня 2024 р.

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «МОДЕЛЬ ПРОГНОЗУВАННЯ ПАРАМЕТРІВ СОЦІАЛЬНО-ЕКОНОМІЧНИХ СИСТЕМ»

Спеціальність 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Галузь знань 17 – Електроніка, автоматизація та електронні комунікації.
Освітня програма «Комп'ютеризовані системи управління та автоматика»

Захищено на засіданні
Екзаменаційної комісії № 44
протокол № __ від __.12.2024 р.
Оцінка _____ / _____
Голова Екзаменаційної комісії
_____ **СКОБ Ю. О.**

Виконав:
Студент групи КУ– 61
БОНДАРЕНКО Костянтин Володимирович _____

Керівник:
к.т.н., доцент кафедри комп'ютерних систем та робототехніки
СТРІЛЕЦЬ Вікторія Євгенівна _____

Рецензент: к.т.н., доцент, в.о. зав. кафедри теоретичної та прикладної інформатики
МЕНЯЙЛОВ Євген Сергійович _____

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до магістерської кваліфікаційної роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел і 6 додатків. Загальний обсяг роботи складає 105 сторінок, із яких 67 сторінок основної частини з 19 рисунками, 5 таблицями, списку використаних джерел із 18 найменувань та 6 додатками.

Метою кваліфікаційної роботи є підвищення точності прогнозування значень економічних показників через створення моделі прогнозування параметрів соціально-економічних систем із використанням сучасних методів глибокого навчання, зокрема рекурентних нейронних мереж типу LSTM (Long Short-Term Memory) та мови програмування R. Основна увага приділяється аналізу та прогнозуванню одновимірних і багатовимірних показників, які подаються у вигляді часових рядів, а саме ціни на нафту марки Brent та офіційні курси долара США і євро до гривні.

Об'єкт дослідження – соціально-економічні системи та процеси, зокрема моніторинг і прогнозування критично важливих економічних показників.

Предмет дослідження – моделі і методи прогнозування показників соціально-економічних систем.

Проблема, яка вирішується в кваліфікаційній роботі полягає у забезпеченні високоточного прогнозування параметрів соціально-економічних систем для підтримки стратегічних рішень у сфері економіки, бізнесу та фінансів.

Область застосування: результати роботи можуть бути використані для прогнозування фінансових показників, таких як валютні курси, ціни на нафту, а також для аналізу інших соціально-економічних часових рядів. Розроблені моделі мають потенціал для інтеграції в автоматизовані системи управління, аналітичні платформи та інструменти фінансового планування.

Ключові слова: задача прогнозування, часові ряди, соціально-економічні системи, LSTM, R, багатовимірні моделі, глибоке навчання, фінансовий аналіз.

ABSTRACT

The explanatory note to the master's qualification thesis consists of an introduction, three chapters, conclusions, a list of references, and four appendices. The total length of the thesis is 105 pages, including 67 pages of the main content with 19 figures, 5 tables, a list of references with 18 entries, and four appendices.

The aim of the qualification work is to improve the accuracy of forecasting economic indicators by developing a model for predicting the parameters of socio-economic systems using modern deep learning methods, particularly Long Short-Term Memory (LSTM) recurrent neural networks, and the R programming language. The main focus is on the analysis and forecasting of univariate and multivariate indicators represented as time series, specifically Brent crude oil prices and official exchange rates of the US dollar and euro relative to the Ukrainian hryvnia.

Object of study: socio-economic systems and processes, particularly the monitoring and forecasting of critically important economic indicators.

Subject of study: models and methods for forecasting socio-economic system indicators.

The problem addressed in the qualification work: ensuring high-accuracy forecasting of the parameters of socio-economic systems to support strategic decision-making in the fields of economics, business, and finance.

Application area: The results of this work can be used for forecasting financial indicators, such as exchange rates and oil prices, as well as for analyzing other socio-economic time series. The developed models have the potential to be integrated into automated management systems, analytical platforms, and financial planning tools.

Keywords: forecasting task, time series, socio-economic systems, LSTM, R, multivariate models, deep learning, financial analysis.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

ARIMA – AutoRegressive Integrated Moving Average
RNN – Recurrent Neural Networks
ANN – Artificial Neural Networks
LSTM – Long Short-Term Memory
AR – авторегресія
MA – метод ковзного середнього
ACF – аналіз автокореляційної функції
PACF – часткової автокореляційної функції
VAR – Vector Autoregression
SARIMA – Seasonal AutoRegressive Integrated Moving Average
ReLU – Rectified Linear Unit
GRU – Gated Recurrent Unit
BPTT – Backpropagation through Time
BiLSTM – Bidirectional Long Short-Term Memory
NN – Neural Networks
MAE – Mean Absolute Error
RMSE – Root Mean Squared Error
MAPE – Mean Absolute Percentage Error
БЧР – Багатовимірні часові ряди
STL – Seasonal Decomposition of Time Series
MLE – Метод максимальної правдоподібності
MICE – Імпутація множинного заповнення
PCA – Аналіз головних компонент
t-SNE – t-Distributed Stochastic Neighbor Embedding
RFE – Recursive Feature Elimination
AutoML – Automated Machine Learning
ts – часовий ряд
USD – офіційна валюта Сполучених Штатів Америки, долар США
EUR – офіційна валюта Європейського Союзу, євро

ВСТУП

У сучасному світі соціально-економічні системи, які охоплюють фінансові, виробничі та інші важливі процеси, характеризуються високою динамічністю та взаємозалежністю. Прогнозування параметрів таких систем є невід’ємною складовою для прийняття стратегічних рішень у сфері економіки, бізнесу та управління ресурсами.

Актуальність роботи. З розвитком технологій та глобалізацією економічних процесів виникає потреба у точному прогнозуванні соціально-економічних параметрів. Волатильність цін на нафту, зміни у валютних курсах та інші показники мають значний вплив на фінансову стабільність та планування як на макро-, так і на мікрорівні. Традиційні методи аналізу, такі як ARIMA, часто не враховують складних нелінійних залежностей та довгострокових взаємозв’язків, що обмежує їхню ефективність.

Сучасні методи глибокого навчання, зокрема рекурентні нейронні мережі типу LSTM (Long Short-Term Memory), дозволяють розв'язувати ці проблеми завдяки здатності обробляти складні часові ряди, враховувати довгострокові залежності та нелінійні взаємодії. У цьому контексті розробка і впровадження таких моделей є актуальними для аналізу соціально-економічних систем.

Метою дослідження є підвищення точності прогнозування значень економічних показників через створення моделей прогнозування на основі методів машинного навчання та нейронних мереж.

Об’єкт дослідження – соціально-економічні системи, що характеризуються параметрами, представленими у вигляді часових рядів. Об’єкт охоплює такі аспекти, як зміни валютних курсів, динаміка цін на ресурси (наприклад, на нафту) та інші макроекономічні індикатори, які впливають на економічну стабільність і прийняття управлінських рішень.

Методи дослідження:

- Глибоке навчання: побудова і навчання рекурентних нейронних мереж типу LSTM.
- Аналіз часових рядів: обробка даних, нормалізація, побудова лагових змінних.
- Математична статистика: оцінка точності моделей за метриками MSE, MAE та RMSE.
- Програмні інструменти: реалізація моделей у середовищі R із використанням бібліотек Keras, TensorFlow, ggplot2 тощо.

Предмет дослідження – моделі та методи прогнозування параметрів соціально-економічних систем на основі аналізу часових рядів.

Завдання дослідження:

1. Провести аналіз існуючих методів прогнозування параметрів соціально-економічних систем.
2. Обґрунтувати вибір моделі LSTM для вирішення задачі прогнозування.
3. Розробити модель прогнозування для одновимірного часового ряду (ціни на нафту марки Brent).
4. Розробити модель прогнозування для багатовимірного часового ряду (валютні курси долара США та євро до гривні).
5. Провести тестування моделей та оцінити їхню ефективність за метриками точності.
6. Надати рекомендації щодо використання моделей у реальних сценаріях та перспективи подальших досліджень.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ ПРОГНОЗУВАННЯ ПОКАЗНИКІВ СОЦІАЛЬНО-ЕКОНОМІЧНИХ СИСТЕМ

1.1. Поняття часових рядів

Часовий ряд — це послідовність даних, зібраних або виміряних у рівномірно розподілені моменти часу. Іншими словами, часовий ряд — це набір спостережень, що фіксуються з певною регулярністю, наприклад, щодня, щомісяця або щорічно. Часові ряди є основою для прогнозування, оскільки на їх основі можна виявляти тренди, сезонні коливання та інші закономірності для майбутнього передбачення параметрів.

У випадку соціально-економічних систем часові ряди можуть включати такі показники, як зміна рівня інфляції, ВВП, безробіття, обсягу продажів, цін на нафту та інші економічні параметри. Ключовою характеристикою таких даних є те, що кожен показник залежить від часу, а зміни можуть мати циклічний або випадковий характер.

Часові ряди зазвичай поділяються на два основні типи:

1. *Стаціонарні часові ряди*: це такі ряди, статистичні властивості яких (середнє, дисперсія) залишаються постійними протягом часу. Такі ряди не мають виражених трендів або сезонних коливань. Прикладом може бути рівномірний потік замовлень на виробництво товарів у стабільних економічних умовах.

2. *Нестаціонарні часові ряди*: для таких рядів характерні зміни в статистичних властивостях у часі, такі як наявність трендів (зростаючих або спадних) чи сезонних коливань. Нестаціонарні ряди часто зустрічаються в економіці, оскільки більшість макроекономічних показників мають тенденцію до зростання або зниження протягом тривалого періоду часу. Наприклад, ціни на нафту можуть мати тренд на зростання або сезонні коливання попиту.

Основна мета аналізу часових рядів – це виявлення закономірностей у даних для побудови моделей, що можуть передбачити майбутні значення. У

прогнозуванні часових рядів особливо важливо враховувати різноманітні типи змін: тренди, сезонність та випадкові коливання. Це дозволяє побудувати більш точні прогнози та адаптувати методи прогнозування до специфіки конкретного ряду.

1.2. Класи методів прогнозування часових рядів

Прогнозування часових рядів є однією з основних задач у сфері аналізу даних, і для його вирішення застосовують різноманітні методи, які можна класифікувати за трьома основними напрямками: статистичні методи, методи машинного навчання та гібридні методи.

1. *Статистичні методи* прогнозування базуються на аналізі минулих даних та їх закономірностей. Ці методи добре підходять для стаціонарних і лінійних часових рядів, коли зміни в даних мають певну стабільність і незначні випадкові коливання [7].

ARIMA (AutoRegressive Integrated Moving Average) – один із найпоширеніших статистичних методів для прогнозування часових рядів. Він дозволяє враховувати автокореляцію між значеннями ряду та згладжувати випадкові зміни для покращення прогнозу. ARIMA добре працює з лінійними та стаціонарними рядами, однак має обмеження щодо прогнозування складних, нелінійних процесів.

Експоненціальне згладжування (Exponential Smoothing) — метод, який також використовує минулі спостереження для прогнозування, але з акцентом на останні дані, які мають більший вплив на прогноз, ніж більш давні. Модель Хольта-Вінтерса, яка розширює експоненціальне згладжування для врахування сезонності, є однією з популярних модифікацій цього методу.

2. *Методи машинного навчання.* З розвитком технологій та збільшенням обсягів даних, машинне навчання стало ефективним інструментом для прогнозування часових рядів, особливо у випадках складних та нелінійних процесів. Методи машинного навчання здатні виявляти приховані

закономірності в даних та ефективно працювати з великими і складними наборами даних.

Штучні нейронні мережі (Artificial Neural Networks, ANN) — ці моделі імітують роботу людського мозку і можуть навчатися з даних для побудови складних нелінійних залежностей. Найбільш популярною архітектурою для роботи з часовими рядами є рекурентні нейронні мережі (Recurrent Neural Networks, RNN), оскільки вони враховують залежності між спостереженнями у часі.

LSTM (Long Short-Term Memory) — спеціальний тип рекурентної нейронної мережі, який був розроблений для вирішення проблеми "забування" в довготривалих залежностях. LSTM моделі добре підходять для прогнозування складних часових рядів, таких як зміна цін на ринках або економічні показники, де важливо зберігати інформацію про тривалі залежності між даними.

3. *Гібридні методи* методи поєднують у собі переваги статистичних методів та машинного навчання. Вони дозволяють покращити точність прогнозів, оскільки використовують сильні сторони обох підходів. Гібридні моделі можуть спочатку застосовувати статистичні методи для виділення основних трендів і сезонних коливань, а потім використовувати нейронні мережі для аналізу складніших, нелінійних компонентів.

ARIMA+LSTM – приклад гібридної моделі, яка спочатку застосовує ARIMA для базового прогнозування, а потім використовує LSTM для вдосконалення прогнозів, враховуючи складніші патерни в даних.

Таким чином, кожен із підходів до прогнозування має свої переваги та недоліки, і вибір методу залежить від характеристик конкретного часового ряду. Статистичні методи ефективні для роботи з лінійними та стаціонарними рядами, в той час як методи машинного навчання і гібридні підходи дозволяють краще моделювати складні й нелінійні процеси.

1.3. Статистичні методи прогнозування

1.3.1. Моделі ARIMA (AutoRegressive Integrated Moving Average)

Модель ARIMA (AutoRegressive Integrated Moving Average) є однією з найпопулярніших статистичних методик для прогнозування часових рядів, що використовує минулі значення та похибки для створення прогнозів [2, 10]. Вона поєднує в собі три компоненти: авторегресію (AR), інтеграцію (I) та метод ковзного середнього (MA). Це дозволяє ARIMA моделі враховувати як автокореляцію між значеннями в часовому ряді, так і усувати нестабільні тренди для покращення прогнозів.

ARIMA моделі широко використовуються для прогнозування часових рядів у багатьох галузях, таких як економіка, фінанси, індустрія, оскільки вони є досить гнучкими й можуть адаптуватися до різних типів даних.

Побудова моделей ARIMA включає кілька етапів.

1. Перевірка на стаціонарність. На початку аналізу часового ряду необхідно визначити, чи є він стаціонарним, тобто, чи не змінюються його статистичні характеристики (середнє значення, дисперсія) протягом часу. Якщо ряд не є стаціонарним, необхідно провести операцію диференціювання (інтеграцію), щоб зробити його стаціонарним. Це дозволяє усунути тренди та сезонні коливання з ряду.

2. Визначення параметрів ARIMA (p, d, q):

- p – порядок авторегресії (кількість минулих значень, що використовуються для прогнозу);
- d – порядок інтеграції (кількість разів, коли ряд підлягає диференціюванню для досягнення стаціонарності);
- q – порядок ковзного середнього (кількість минулих похибок, що враховуються при побудові прогнозу).

Вибір параметрів p, d та q здійснюється на основі аналізу автокореляційної функції (ACF) та часткової автокореляційної функції (PACF), що дозволяє виявити наявність автокореляцій між даними [3].

3. Оцінка моделі. Після визначення параметрів ARIMA моделі її потрібно навчити на наявних даних. Для цього використовується метод максимізації ймовірності, який дозволяє підібрати коефіцієнти моделі, що мінімізують помилку прогнозу.

4. Перевірка якості моделі. Для оцінки якості побудованої моделі проводиться аналіз залишків (різниця між фактичними значеннями та прогнозом). Залишки повинні бути випадковими та не мати автокореляції. Якщо залишки вказують на кореляції, модель потребує додаткового коригування.

5. Побудова прогнозу. На основі отриманої моделі ARIMA можна робити прогнози на майбутні періоди. Прогнозування здійснюється шляхом екстраполяції на основі раніше отриманих залежностей.

ARIMA залишається популярним методом завдяки своїй простоті та ефективності для прогнозування стаціонарних часових рядів. Проте для більш складних і нелінійних процесів варто звертатися до інших методів, таких як нейронні мережі або гібридні моделі.

1.3.2. Алгоритм Хольта-Вінтерса

Модель Хольта-Вінтерса є одним із найбільш поширених методів експоненціального згладжування для прогнозування часових рядів, які мають як тренди, так і сезонність. В основі цього методу лежить принцип, згідно з яким останні спостереження мають більший вплив на прогноз, ніж давніші дані. Це дозволяє моделі оперативно реагувати на нові зміни у часових рядах [10].

Модель Хольта-Вінтерса складається з трьох основних компонентів: згладжування рівня, згладжування тренду та згладжування сезонних коливань. Цей метод підходить для прогнозування як адитивних, так і мультиплікативних часових рядів, залежно від природи сезонності.

Алгоритм Хольта-Вінтерса базується на трьох рівняннях, що відповідають за рівень, тренд і сезонність у часовому ряді:

1. *Згладжування рівня* (overall level smoothing):

$$l_t = \alpha(y_t - S_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}), \quad (1.1)$$

де l_t — згладжений рівень на момент часу t, y_t — фактичне значення часового ряду, S_{t-m} — сезонний компонент з попереднього сезону (де mmm — період сезонності), α — коефіцієнт згладжування рівня (значення від 0 до 1).

2. *Згладжування тренду* (trend smoothing):

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}, \quad (1.2)$$

де b_t — згладжений тренд на момент часу t, β — коефіцієнт згладжування тренду (значення від 0 до 1).

3. *Згладжування сезонності* (seasonal smoothing):

$$S_t = \Upsilon(y_t - l_t) + (1-\Upsilon)S_{t-m}, \quad (1.3)$$

де S_t — сезонний компонент на момент часу t, Υ — коефіцієнт згладжування сезонності (значення від 0 до 1).

4. *Прогнозування* (forecasting). Прогноз для майбутнього періоду hhh розраховується як:

$$\hat{y}_t + h = l_t + hb_t + S_{t+h-m(k+1)}, \quad (1.4)$$

де $\hat{y}_t + h$ — прогнозоване значення для періоду t+h, l_t — рівень на момент часу t, b_t — тренд на момент часу t, $S_{t+h-m(k+1)}$ — сезонний компонент, що коригується на основі попередніх циклів сезонності.

Таким чином, алгоритм Хольта-Вінтерса є потужним інструментом для прогнозування часових рядів із сезонними коливаннями та трендами. Він залишається популярним завдяки простоті й ефективності, особливо для середньострокового прогнозування.

Окрім моделей ARIMA та експоненціального згладжування, існує ряд інших статистичних методів, що використовуються для прогнозування часових рядів. До найбільш популярних належать моделі VAR (Vector Autoregression) та SARIMA (Seasonal ARIMA). Ці методи дозволяють працювати з більш складними часовими рядами, які можуть включати взаємозалежні змінні та сезонні коливання.

1.3.3. Модель VAR (Vector Autoregression)

VAR — це модель векторної авторегресії, яка використовується для прогнозування кількох взаємозалежних часових рядів одночасно. На відміну від ARIMA, яка працює з одним часовим рядом, VAR дозволяє враховувати вплив кількох змінних одна на одну, що особливо корисно в економіці, де різні макроекономічні показники взаємопов'язані.

Основна ідея VAR полягає в тому, що кожна змінна прогнозується на основі попередніх значень як власних спостережень, так і значень інших змінних у системі. Наприклад, для прогнозування ВВП можна використовувати не лише минулі значення ВВП, але й минулі значення таких показників, як інфляція або рівень безробіття.

Алгоритм VAR:

1. Визначають кілька змінних, які можуть впливати одна на одну (наприклад, ВВП, інфляція, зайнятість).

2. Для кожної змінної будують рівняння регресії на основі її попередніх значень та значень інших змінних.

3. Модель одночасно оцінює всі рівняння для передбачення майбутніх значень кожної змінної.

1.3.4. Модель SARIMA (Seasonal ARIMA)

SARIMA — це розширення моделі ARIMA, яке враховує сезонні коливання у часових рядах. Вона поєднує в собі автокореляцію, інтеграцію та метод ковзного середнього з додатковими сезонними компонентами. SARIMA використовується для прогнозування часових рядів, які мають виражену сезонність, тобто повторювані цикли через певні інтервали часу (місяць, квартал, рік тощо) [10].

Як і ARIMA, SARIMA моделі характеризуються трьома параметрами (p, d, q), але додатково враховуються сезонні параметри (P, D, Q, m), де:

- P — порядок сезонної авторегресії,
- D — кількість диференціювань для сезонності,

- Q — порядок сезонного ковзного середнього,
- m — довжина сезону (наприклад, для річної сезонності m = 12).

Алгоритм SARIMA:

1. Спочатку перевіряється, чи має часовий ряд сезонні коливання. Якщо сезонність присутня, її необхідно врахувати при моделюванні.

2. Виконується сезонне диференціювання для усунення сезонних трендів.

3. Визначаються параметри як для базової ARIMA моделі (p, d, q), так і для сезонної частини (P, D, Q, m).

4. Модель оцінюється та використовується для прогнозування.

Таким чином, обидві моделі — VAR та SARIMA — є потужними інструментами для прогнозування часових рядів у різних ситуаціях. VAR підходить для аналізу взаємопов'язаних змінних, тоді як SARIMA дозволяє ефективно прогнозувати дані з сезонністю. Обидві моделі залишаються важливими інструментами в арсеналі статистичних методів прогнозування.

1.4. Моделі на основі машинного навчання

1.4.1. Штучні нейронні мережі (Artificial Neural Networks, ANN)

Штучні нейронні мережі (Artificial Neural Networks, ANN) — це один з найважливіших методів машинного навчання, який застосовується для вирішення складних завдань, включаючи прогнозування часових рядів, розпізнавання образів, класифікацію та багато інших. Нейронні мережі надихаються біологічними процесами роботи мозку, де нейрони взаємодіють між собою, передаючи сигнали. В основі штучної нейронної мережі також лежить ідея передачі інформації між нейронами (штучними елементами), які обробляють вхідні дані та формують вихід [4].

Нейронна мережа складається з трьох основних шарів:

1. Вхідний шар (Input Layer). Це перший шар, через який дані потрапляють у нейронну мережу. Кількість нейронів у цьому шарі відповідає кількості вхідних характеристик або ознак, які необхідно обробити.

Наприклад, якщо ми прогнозуємо часовий ряд на основі минулих даних, кожен нейрон у вхідному шарі відповідає за одне з попередніх значень ряду.

2. Приховані шари (Hidden Layers). Приховані шари складають основну частину нейронної мережі. Це шари, де відбувається обробка вхідної інформації та її перетворення. Кожен нейрон у прихованому шарі отримує вхідні дані від попередніх нейронів, виконує математичну операцію (активаційну функцію) та передає результат до наступного шару. Кількість прихованих шарів і нейронів у кожному шарі може варіюватися в залежності від складності задачі. Мережі, що мають кілька прихованих шарів, називаються глибокими нейронними мережами (Deep Neural Networks, DNN).

3. Вихідний шар (Output Layer). Це останній шар у нейронній мережі, який формує кінцевий результат обробки даних. Вихідний шар може мати один або кілька нейронів, в залежності від характеру задачі. Наприклад, у задачах прогнозування часових рядів вихідний нейрон може представляти значення прогнозу для наступного періоду.

Нейронна мережа працює за принципом поступової передачі інформації через шари з вхідного шару до вихідного [9]. Основні етапи роботи нейронної мережі можна описати так:

1. Пропускання вперед (Forward Propagation). На цьому етапі дані передаються від вхідного шару до вихідного через приховані шари. Кожен нейрон у прихованому шарі обчислює зважену суму вхідних сигналів і передає результат через активаційну функцію. Активаційні функції можуть бути різними, найбільш популярними є сигмоїдна функція, ReLU (Rectified Linear Unit), або гіперболічний тангенс.

Формула роботи нейрона:

$$z = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b, \tag{1.5}$$

де $x_1, x_2, \ldots, x_n$ — вхідні значення (сигнали), $w_1, w_2 \ldots, w_n$— вагові коефіцієнти нейрона, b — зсув (bias), z — результат суми, який передається через активаційну функцію.

2. Активаційна функція. Після обчислення суми вхідних сигналів нейрон використовує активаційну функцію, щоб визначити, чи активується цей нейрон, тобто чи передасть він сигнал далі. Активаційні функції забезпечують нелінійність у нейронній мережі, що дозволяє моделювати складні взаємозалежності між даними.

3. Зворотне поширення помилки (Backpropagation). Після того, як мережа сформувала вихідний сигнал, обчислюється помилка, яка представляє різницю між фактичним і прогнозованим значенням. Зворотне поширення помилки використовується для коригування вагових коефіцієнтів мережі з метою мінімізації цієї помилки. Алгоритм навчання нейронної мережі базується на градієнтному спуску, який поступово змінює ваги, щоб зменшити помилку на кожному кроці.

4. Навчання моделі. Процес навчання нейронної мережі полягає у багаторазовому пропусканні даних через мережу та коригуванні вагових коефіцієнтів на кожному кроці зворотного поширення помилки. Після того, як мережа достатньо навчена, вона може прогнозувати або класифікувати нові дані.

Таким чином, штучні нейронні мережі є потужним інструментом для моделювання складних і нелінійних процесів, особливо в задачах прогнозування часових рядів, що дозволяє отримувати точні й адаптивні прогнози для соціально-економічних систем.

1.4.2. Рекурентні нейронні мережі (Recurrent Neural Networks, RNN)

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) є особливим типом штучних нейронних мереж, які призначені для обробки послідовних даних, таких як часові ряди. Основна особливість RNN полягає в тому, що вони можуть враховувати попередні значення в послідовності при формуванні прогнозів, що робить їх особливо корисними для роботи з даними, які мають часову залежність [8].

У традиційних нейронних мережах кожен вхід розглядається незалежно один від одного. Однак у багатьох задачах, зокрема в прогнозуванні часових рядів, поточне значення може бути тісно пов'язане з попередніми значеннями. RNN розв'язують цю проблему, зберігаючи інформацію про попередні стани мережі у внутрішній пам'яті, що дозволяє їм аналізувати послідовні залежності у часі.

В основі RNN лежить ідея зворотного зв'язку. Кожен нейрон в прихованому шарі RNN передає свій вихід не тільки на наступний шар, але й назад на себе, зберігаючи контекст попередніх кроків. Це дозволяє мережі "пам'ятати" інформацію про попередні входи і використовувати її при прогнозуванні наступних значень.

Класична рекурентна нейронна мережа має таку структуру [8]:

1. Вхідний шар (Input Layer). Кожен елемент вхідного шару відповідає за певний крок у послідовності (наприклад, за одне значення часового ряду на певний момент часу). Ці дані подаються до прихованого шару для подальшої обробки.

2. Прихований шар (Hidden Layer). Основна особливість RNN полягає у збереженні стану прихованого шару на кожному кроці. На кожному кроці часу прихований шар отримує два типи інформації: поточні дані з вхідного шару та значення, збережене з попереднього кроку (пам'ять). Формально це можна виразити так:

$$h_t = f(W_h x_t + U_h h_{t-1} + b_h), \tag{1.6}$$

де h_t — стан прихованого шару на момент часу t, x_t — вхідне значення на момент часу t, h_{t-1} — стан прихованого шару на попередньому кроці t -1 , W_h і U_h— вагові матриці для поточних входів та попередніх станів відповідно, b_h — зміщення (bias), f — активаційна функція (зазвичай це гіперболічний тангенс або сигмоїдна функція).

3. Вихідний шар (Output Layer). Після обробки кожного елемента послідовності прихований шар формує вихідний сигнал, який передається на вихідний шар для формування кінцевого прогнозу. У випадку прогнозування

часових рядів вихідний шар на кожному кроці може формувати прогноз для наступного значення у ряді.

RNN стали популярними для прогнозування часових рядів через їхню здатність "пам'ятати" попередні значення даних і використовувати цю інформацію для прогнозування майбутніх подій [8]. Зокрема, RNN застосовуються для таких задач:

- прогнозування фінансових показників (ціни акцій, валютних курсів тощо);
- прогнозування попиту на товари або послуги;
- прогнозування економічних параметрів, таких як рівень безробіття, ВВП або інфляція.

Проте, класичні RNN мають одну суттєву проблему – вони не здатні ефективно запам'ятовувати інформацію на довгих послідовностях. Це призводить до того, що з часом попередні дані стають менш релевантними, а прогноз може бути неточним.

Класичні RNN мають обмежені можливості щодо зберігання довгострокових залежностей, оскільки сигнали поступово згасають при багаторазовому зворотному поширенні помилки (backpropagation). Це призводить до втрати важливої інформації, яка є актуальною для довготривалих часових рядів. Для вирішення цієї проблеми були розроблені спеціальні варіанти RNN, такі як LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit).

RNN є важливим інструментом для прогнозування часових рядів завдяки своїй здатності враховувати послідовну залежність даних. Проте для задач з довготривалими залежностями рекомендується використовувати розширені варіанти RNN, такі як LSTM або GRU, які здатні ефективніше зберігати інформацію протягом довгого часу.

1.4.3. Модель довгої короткострокової пам'яті (Long Short-Term Memory, LSTM)

Модель Long Short-Term Memory (LSTM) — це різновид рекурентної нейронної мережі (RNN), розроблений для подолання обмежень класичних RNN, зокрема проблеми "затухання градієнтів", яка ускладнює запам'ятовування довгострокових залежностей. LSTM використовується для обробки послідовних даних, таких як часові ряди, і дозволяє зберігати інформацію на тривалих відрізках часу. Це робить модель ефективною для прогнозування складних часових рядів із нелінійними та тривалими залежностями [1].

Основною характеристикою LSTM є наявність спеціальних осередків пам'яті та механізму керування потоком інформації через три головні "ворота" (gates): вхідні ворота, ворота забування та вихідні ворота. Ці ворота контролюють, яку інформацію слід запам'ятати, яку забути і яку використати для формування прогнозу.

Кожен осередок пам'яті в LSTM містить три основні компоненти:

1. Вхідні ворота (Input Gate). Вхідні ворота визначають, яку частину нової інформації з поточного стану потрібно зберегти у пам'яті осередку. Це регулюється за допомогою активаційної функції (зазвичай сигмоїдальної), яка пропускає або блокує частину інформації.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i), \tag{1.7}$$

де i_t — активація вхідних воріт на момент часу t, h_{t-1} — стан прихованого шару на попередньому кроці, x_t — поточне вхідне значення, W_i — вагова матриця для вхідних воріт, b_i — зміщення.

2. Ворота забування (Forget Gate). Ворота забування вирішують, яку частину інформації з попереднього стану потрібно видалити з пам'яті осередку. Якщо певна інформація більше не є актуальною для майбутніх прогнозів, вона може бути забута.

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f), \tag{1.8}$$

де f_t — активація воріт забування, W_f — вагова матриця для воріт забування, b_f— зміщення.

3. Вихідні ворота (Output Gate). Вихідні ворота визначають, яка частина інформації з осередку пам'яті буде використана для формування вихідного сигналу на поточному кроці.

$$O_t = \sigma(W_o * [h_{t-1}, x_t] + b_o), \quad (1.9)$$

де O_t — активація вихідних воріт, W_o — вагова матриця для вихідних воріт, b_o — зміщення.

4. Оновлення стану пам'яті. Після активації воріт забування та вхідних воріт, модель оновлює стан пам'яті:

$$C_t = f_t * C_{t-1} + i_t * \bar{C}_t, \quad (1.10)$$

де C_t — новий стан пам'яті на момент часу t, C_{t-1} — попередній стан пам'яті, $\bar{C}_t$ — нова кандидатура стану пам'яті, сформована на основі вхідних даних і попереднього стану.

5. Вихідний стан. Нарешті, модель формує вихідний стан на основі оновленого стану пам'яті:

$$h_t = o_t * \tanh(C_t). \quad (1.11)$$

Алгоритм роботи LSTM та його переваги у порівнянні з традиційними методами [12]:

1. Пропускання вперед (Forward Propagation). На кожному кроці часу модель отримує нове вхідне значення та оновлює свій внутрішній стан (осередок пам'яті) за допомогою вхідних, воріт забування та вихідних воріт. Пропускання вперед включає обробку інформації через кожен із цих компонентів і формування вихідного сигналу для поточного кроку.

2. Зворотне поширення помилки (Backpropagation through Time, BPTT). Для навчання LSTM модель використовує алгоритм зворотного поширення помилки з урахуванням часу. Це дозволяє коригувати ваги мережі таким чином, щоб мінімізувати різницю між фактичними та прогнозованими значеннями для кожного кроку часу.

3. Навчання моделі. Навчання LSTM виконується на основі великої кількості даних, і для цього часто використовують оптимізаційні алгоритми, такі як градієнтний спуск. Після навчання модель може прогнозувати наступні значення в послідовності даних.

Переваги LSTM у порівнянні з традиційними методами:

1. Запам'ятовування довготривалих залежностей. Одна з основних переваг LSTM перед класичними RNN полягає в здатності зберігати інформацію на довгі проміжки часу. Завдяки структурі пам'яті, LSTM може зберігати важливі дані, навіть якщо вони були введені багато кроків тому.

2. Адаптивність до нелінійних часових рядів. LSTM здатна обробляти складні та нелінійні часові ряди, які часто зустрічаються у соціально-економічних системах. Традиційні статистичні методи, такі як ARIMA, мають обмеження у роботі з нелінійними процесами, тоді як LSTM може моделювати такі взаємозалежності.

3. Захист від проблеми затухання градієнта. На відміну від класичних RNN, LSTM захищена від проблеми затухання градієнта, яка ускладнює навчання мережі на довгих часових послідовностях. Це робить її більш ефективною для прогнозування даних із довготривалими взаємозв'язками.

4. Гнучкість застосування. LSTM широко використовується у фінансових програмах, прогнозуванні попиту, аналізі ринкових трендів та інших задачах, де важливо враховувати минулі значення для формування точних прогнозів.

Отже, LSTM є потужним інструментом для прогнозування часових рядів завдяки своїй здатності враховувати довготривалі залежності, обробляти нелінійні процеси та адаптуватися до складних задач. Це робить її одним із найбільш ефективних методів для прогнозування соціально-економічних параметрів у сучасних умовах.

1.4.4. Gated Recurrent Unit (GRU)

Окрім LSTM, існують й інші типи нейронних мереж, що використовуються для прогнозування часових рядів і обробки послідовних

даних. До них належать GRU (Gated Recurrent Unit) та BiLSTM (Bidirectional Long Short-Term Memory). Обидві ці архітектури є вдосконаленням традиційних рекурентних нейронних мереж і мають певні переваги при вирішенні специфічних завдань.

GRU — це спрощений варіант LSTM, який було розроблено для зменшення складності моделі та підвищення її ефективності. Як і LSTM, GRU використовує механізми керування пам'яттю та ворота, що дозволяють контролювати потік інформації, але на відміну від LSTM, GRU має лише двоє воріт: ворота оновлення (Update Gate) та ворота скидання (Reset Gate) [8].

- Ворота оновлення відповідають за контроль того, яка частина інформації з попередніх станів має зберігатися.
- Ворота скидання вирішують, яку частину попередньої інформації можна "забути" для врахування нових даних.

Формально, GRU визначається такими рівняннями:

1. Активація воріт оновлення:

$$z_t = \sigma(W_z * [h_{t-1}, x_t] + b_z). \quad (1.12)$$

2. Активація воріт скидання:

$$r_t = \sigma(W_r * [h_{t-1}, x_t] + b_r). \quad (1.13)$$

3. Оновлений стан пам'яті:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t, \quad (1.14)$$

де z_t— активація воріт оновлення, r_t — активація воріт скидання, $\tilde{h}_t$ — новий кандидат на стан пам'яті, отриманий на основі поточного стану та входу.

1.4.5. Bidirectional Long Short-Term Memory (BiLSTM)

BiLSTM — це модифікація традиційної LSTM, яка дозволяє моделі навчатися на даних як у прямому, так і у зворотному напрямках. У звичайній LSTM прогноз формується лише на основі попередніх значень послідовності (з минулого в майбутнє). BiLSTM, натомість, використовує дві LSTM-мережі: одна аналізує послідовність вперед, а інша — назад. Це дозволяє моделі

враховувати як попередні значення, так і майбутні, що особливо корисно для завдань, де важливо знати весь контекст даних [14].

Архітектура BiLSTM:

- Мережа складається з двох шарів LSTM: один аналізує дані в хронологічному порядку, інший — у зворотному.
- Обидва шари об'єднуються на виході, що дає змогу отримати більш повну інформацію про часові залежності в обох напрямах.

Порівняння GRU та BiLSTM:

- GRU є більш простою і менш ресурсоємною моделлю порівняно з LSTM і BiLSTM, тому часто використовується для задач з меншими обсягами даних або обмеженими ресурсами. Вона добре підходить для коротких і середньострокових прогнозів.
- BiLSTM забезпечує більш точне прогнозування за рахунок використання інформації як із минулого, так і з майбутнього в послідовності. Ця модель зазвичай використовується в задачах, де важливо мати повний контекст послідовності (наприклад, у завданнях обробки природної мови), проте її складність і вимоги до ресурсів вищі, ніж у GRU.

Таким чином, GRU і BiLSTM є важливими інструментами для роботи з послідовними даними, такими як часові ряди. Вибір між ними залежить від специфіки задачі: для більш простих і швидких рішень підходить GRU, тоді як BiLSTM забезпечує кращу точність для завдань, де важливий повний контекст даних.

1.5. Гібридні моделі

Гібридні моделі для прогнозування часових рядів поєднують традиційні статистичні методи, такі як ARIMA (AutoRegressive Integrated Moving Average), з методами машинного навчання, зокрема нейронними мережами. Такий підхід дозволяє використовувати сильні сторони обох методів: статистичні моделі добре працюють з лінійними і стаціонарними даними, тоді як нейронні мережі здатні моделювати складні нелінійні взаємозалежності.

Основна ідея поєднання ARIMA і нейронних мереж полягає в тому, що ARIMA моделює лінійну частину часових рядів, видаляючи тенденції та сезонні коливання, після чого нейронні мережі використовуються для моделювання залишкової (нелінійної) частини даних. Це дозволяє значно підвищити точність прогнозів, особливо у випадках, коли часові ряди мають складні динаміки, які важко змоделювати лише статистичними методами [4].

Алгоритм побудови гібридної моделі ARIMA+NN [13].

1. *Моделювання лінійних компонентів* за допомогою ARIMA. Спочатку часовий ряд аналізується за допомогою ARIMA, яка моделює лінійну залежність та сезонні компоненти. На цьому етапі отримується базовий прогноз і залишки (різниця між фактичними значеннями і прогнозом ARIMA), які містять нелінійні компоненти.

2. *Моделювання залишкових помилок* за допомогою нейронної мережі. Після видалення лінійних компонентів із часового ряду, залишкові помилки передаються на вхід нейронної мережі (наприклад, LSTM або GRU). Нейронна мережа навчається на цих залишках, що дозволяє їй моделювати складні нелінійні патерни.

3. *Комбінування прогнозів***.** Остаточний прогноз формується шляхом додавання прогнозів, отриманих від ARIMA та нейронної мережі. ARIMA відповідає за лінійні компоненти, а нейронна мережа — за нелінійні.

Гібридні моделі значно покращують якість прогнозів, оскільки поєднують переваги різних підходів. Традиційні методи, такі як ARIMA, ефективні для прогнозування лінійних і стаціонарних часових рядів, але мають обмеження у моделюванні складних процесів із нелінійними взаємодіями [6]. Нейронні мережі ж, зокрема LSTM або GRU, можуть моделювати нелінійності, але для досягнення точних прогнозів вони часто потребують великих обсягів даних і ресурсів для навчання.

Гібридні моделі дозволяють:

- Підвищити точність прогнозів. Поєднання ARIMA і нейронних мереж зменшує похибки, оскільки кожен метод моделює ту частину даних, для якої він найефективніший [9].
- Моделювати складні залежності**.** Нейронні мережі можуть враховувати нелінійні взаємодії та складні закономірності, які ARIMA не здатна відобразити.
- Зменшити ризик оверфітингу**.** ARIMA знижує ризик переобладнання, оскільки спочатку видаляє лінійні компоненти, залишаючи нейронній мережі лише складну частину ряду.

1.6. Методи оцінки якості прогнозування

1.6.1. Метрики оцінки точності прогнозів

Для оцінки точності прогнозування використовують різні метрики, які дозволяють визначити, наскільки передбачені значення відрізняються від реальних. Найбільш поширеними метриками є MAE, RMSE та MAPE [10].

MAE (Mean Absolute Error, середня абсолютна помилка) – це середнє значення абсолютних помилок між прогнозованими та фактичними значеннями. Вона показує, наскільки в середньому прогноз відрізняється від реальних даних без врахування напряму цієї різниці (негативних або позитивних відхилень).

Формула MAE:

$$MAE = \frac{1}{n}\sum_{t=1}^{n} |y_t - \hat{y}_t|, \tag{1.15}$$

де y_t — фактичне значення, $\hat{y}_t$ — прогнозоване значення, n — кількість спостережень.

RMSE (Root Mean Squared Error, корінь середньоквадратичної помилки) є поширеною метрикою, яка враховує квадрати помилок, тому більше карає великі відхилення. Це робить RMSE більш чутливою до великих помилок порівняно з MAE.

Формула RMSE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}. \quad (1.16)$$

MAPE (Mean Absolute Percentage Error, середня абсолютна відносна помилка) вимірює відносну помилку у відсотках і дозволяє порівнювати точність прогнозів у випадках, коли фактичні значення мають різні масштаби. MAPE показує, на скільки відсотків у середньому прогноз відхиляється від фактичних значень.

Формула MAPE:

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right|. \quad (1.17)$$

Переваги та недоліки різних метрик наведені у табл. 1.1.

Таблиця 1.1

Переваги і недоліки метрик оцінки точності

Метрика	Переваги	Недоліки
MAE	є інтуїтивно зрозумілою, оскільки показує середнє відхилення у тих самих одиницях вимірювання, що й самі дані. Це робить її зручною для інтерпретації	не робить різниці між великими та малими помилками, тому вона не враховує значні відхилення
RMSE	карає за великі помилки завдяки квадратичному компоненту. Це корисно у випадках, коли великі помилки є небажаними та потребують особливої уваги	складніше інтерпретувати, оскільки вона не є середнім абсолютним відхиленням, а також вона менш стійка до викидів у даних, які можуть значно вплинути на результат.
MAPE	виражає похибку у відсотках, що дозволяє легко порівнювати точність прогнозів для різних наборів даних із різними масштабами.	не працює коректно, якщо фактичні значення близькі до нуля, оскільки тоді відсоткові відхилення можуть бути надто великими або невизначеними.

1.6.2. Методи перехресної валідації у прогнозуванні часових рядів

Перехресна валідація (cross-validation) — це метод, який дозволяє оцінити стабільність та узагальнюваність моделі на нових даних. Для прогнозування

часових рядів стандартний підхід до перехресної валідації (наприклад, метод "k-fold") не підходить через наявність часової залежності між даними. Тому для часових рядів використовуються специфічні методи перехресної валідації [10].

1. Метод ковзного вікна (Rolling Window). Цей метод полягає в тому, що на кожному кроці модель навчається на попередньому вікні даних і перевіряється на наступному інтервалі. Це дозволяє симулювати прогнозування в реальних умовах, коли модель повинна передбачати майбутні значення на основі історичних даних.

2. Розширюване вікно (Expanding Window). У цьому методі модель навчається на всіх доступних даних до поточного моменту, а тестування проводиться на наступних кількох точках. На кожному кроці вікно навчальних даних розширюється, що дозволяє використовувати більше інформації для тренування моделі.

3. Time Series Split. Цей метод полягає в розбитті часових рядів на кілька неперетинаючих інтервалів для навчання та тестування моделі. Він використовується для оцінки моделі в ситуаціях, коли важливо побачити, як модель поводиться на різних етапах часу.

Переваги перехресної валідації для часових рядів:

- Оцінка стабільності моделі: Перехресна валідація дозволяє оцінити, наскільки добре модель узагальнює прогноз для різних періодів часу.
- Запобігання переобладнанню (overfitting): Використання перехресної валідації допомагає уникнути проблеми оверфітингу, оскільки модель тестується на даних, які вона раніше не бачила.

Таким чином, методи перехресної валідації є критично важливими для точного прогнозування часових рядів, оскільки вони дозволяють оцінити ефективність моделей на різних етапах часу і забезпечують стабільніші прогнози.

Висновки за розділом 1

Прогнозування часових рядів є складною задачею, яка вимагає використання різноманітних підходів, залежно від характеристик даних та вимог до точності. Різні методи прогнозування мають свої переваги та обмеження [5].

Статистичні методи, такі як ARIMA, є ефективними для прогнозування лінійних і стаціонарних часових рядів, де динаміка змін є відносно передбачуваною. Їхня простота та прозорість роблять їх привабливими для багатьох економічних задач. Однак, вони обмежені в моделюванні складних, нелінійних процесів і не завжди здатні врахувати багатофакторні залежності у соціально-економічних системах.

*Моделі машинного навчання***,** такі як LSTM та GRU, забезпечують кращу продуктивність для прогнозування нелінійних часових рядів з довготривалими залежностями. Вони здатні "навчатися" на складних даних і виявляти приховані закономірності, що робить їх більш гнучкими та потужними у порівнянні з традиційними підходами. Проте їхня складність і потреба у великих обсягах даних можуть бути недоліками у деяких задачах [11].

Гібридні моделі, які поєднують статистичні методи та нейронні мережі, дозволяють взяти найкраще з обох світів. Вони моделюють лінійні компоненти за допомогою ARIMA або SARIMA, тоді як нейронні мережі допомагають врахувати нелінійності та складні взаємозв'язки. Такі моделі показують високу точність і є ефективним рішенням для задач з неоднорідною структурою даних.

Вибір методу прогнозування має важливе значення, оскільки соціально-економічні дані часто мають складну структуру з різними типами взаємозалежностей. Для досягнення високої точності прогнозів важливо враховувати специфіку даних:

- Наявність трендів і сезонності**.** Для рядів, які мають яскраво виражену сезонність, добре підходять моделі типу SARIMA або Holt-Winters. Однак, для

даних із складними трендами або з високою волатильністю варто застосовувати нейронні мережі.

- Нелінійність та багатофакторність**.** Для соціально-економічних систем, де важливі нелінійні залежності та врахування багатьох факторів (політичних, демографічних, ринкових тощо), перевагу слід надавати методам машинного навчання, зокрема рекурентним мережам типу LSTM та GRU.
- Довготривала пам'ять**.** У випадках, коли необхідно враховувати взаємозв'язки на великі відрізки часу, класичні RNN можуть виявитися неефективними. LSTM і GRU, завдяки своїй здатності зберігати інформацію на тривалий період, є кращим вибором для таких задач.

Отже, правильний вибір моделі залежить від характеру часових рядів, що аналізуються, і специфічних вимог до прогнозу. Одні моделі краще підходять для простих задач із лінійними тенденціями, тоді як інші є більш придатними для складних і динамічних процесів [15].

РОЗДІЛ 2
МОДЕЛЮВАННЯ БАГАТОВИМІРНИХ ЧАСОВИХ РЯДІВ

2.1. Визначення та класифікація багатовимірних часових рядів

2.1.1. Поняття багатовимірних часових рядів

Багатовимірні часові ряди (БЧР) – це сукупність кількох часових рядів, кожен з яких є функцією часу, але які розглядаються разом через наявність взаємозв'язків між ними. Формально багатовимірний часовий ряд можна описати як $\boldsymbol{X(t) = [x_1(t), x_2(t), \dots, x_d(t)]^T}$, де $\boldsymbol{x_i(t)}$ – значення $\boldsymbol{i}$ - тої змінної у момент часу $\boldsymbol{t}$, а $\boldsymbol{d}$ – кількість змінних.

Особливість багатовимірних часових рядів полягає у тому, що їх аналіз вимагає врахування як індивідуальних змін кожної змінної, так і їх взаємодій між собою. Наприклад, при аналізі макроекономічних показників такі змінні, як рівень інфляції, безробіття та облікова ставка, мають складні взаємозалежності, які необхідно враховувати для точного прогнозування.

Основна відмінність між одномірними та багатовимірними часовими рядами полягає у вимірності простору. В одномірному ряді кожна точка часу представлена одним значенням, тоді як у багатовимірному – вектором значень (рис. 2.1), що може включати десятки чи навіть сотні змінних. Це робить багатовимірні ряди надзвичайно корисними для моделювання складних систем, але водночас ускладнює їх аналіз через такі проблеми, як висока розмірність і шумові компоненти.

Основні приклади застосування БЧР:

1. Економіка та фінанси: аналіз взаємозв'язків між курсами валют, цінами на активи та макроекономічними показниками.

2. Медицина: аналіз взаємодії різних фізіологічних показників, наприклад, пульсу, артеріального тиску та рівня кисню у крові.

3. Енергетика: прогнозування споживання енергії з урахуванням кліматичних змінних, таких як температура та вологість.

4. Кліматологія: аналіз температури, вологості, тиску та інших факторів, які впливають на глобальні зміни клімату.

Особливістю багатовимірних часових рядів є взаємозалежності між змінними, які можуть бути як статичними (наприклад, кореляція між змінними), так і динамічними (наприклад, причинно-наслідкові зв'язки або лагові залежності). Це відкриває можливості для розширеного аналізу даних, але водночас вимагає спеціалізованих методів, таких як векторні авторегресійні моделі (VAR) чи рекурентні нейронні мережі (RNN), для ефективного аналізу [16].

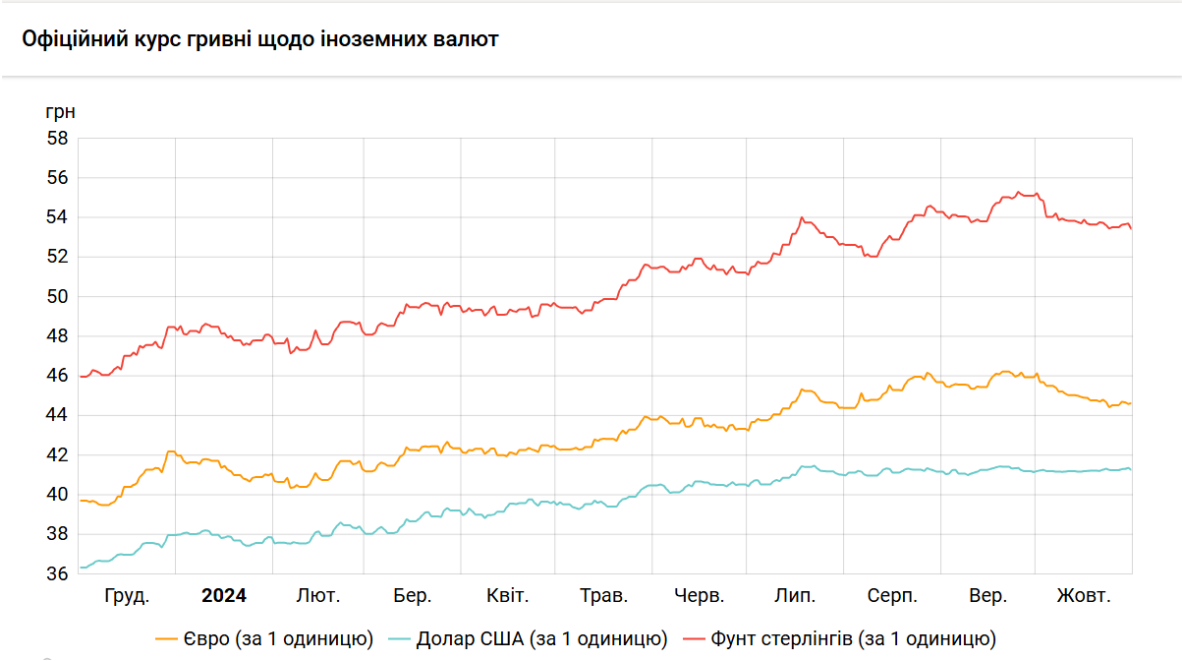

Рисунок 2.1 – Приклад багатовимірного часового ряду

2.1.2. Класифікація багатовимірних часових рядів

Багатовимірні часові ряди (БЧР) мають широку сферу застосувань і велику варіативність структур, що дозволяє класифікувати їх за різними критеріями. Розуміння класифікації необхідне для правильного вибору методів аналізу та моделювання.

Класифікація *за типами даних* [17]:

1. Континуальні (або безперервні) часові ряди містять дані, які приймають будь-які значення в заданому діапазоні. Прикладом є температура

повітря або рівень води у річці. Такі ряди часто описуються за допомогою диференціальних рівнянь або статистичних моделей.

Особливістю роботи з континуальними даними є потреба враховувати їхню високу деталізацію, що може вимагати додаткових обчислювальних ресурсів для аналізу.

2. Дискретні часові ряди включають дані, що змінюються в конкретні моменти часу. Наприклад, обсяги продажів упродовж місяця або щоденний рівень трафіку на вебсайті.

Дискретні дані мають природні інтервали між вимірюваннями, що робить їх більш зручними для аналізу та обробки. Методи прогнозування таких рядів часто базуються на лінійних моделях, таких як ARIMA, або машинному навчанні, наприклад, LSTM.

3. Категоріальні часові ряди. У деяких випадках дані можуть мати категоріальний характер, наприклад, у маркетингу, коли потрібно передбачити зміну вподобань споживачів (високий, середній чи низький попит). Аналіз таких рядів часто потребує використання спеціалізованих алгоритмів класифікації, наприклад, рекурентних нейронних мереж із врахуванням категорій.

Класифікація *за структурою* [18]:

1. Регулярні часові ряди характеризуються однаковими інтервалами між вимірюваннями даних. Наприклад, дані, що записуються кожні 5 хвилин, щодня або щомісяця. Регулярність забезпечує простоту аналізу та дозволяє використовувати широкий спектр алгоритмів, які вимагають однакових інтервалів, наприклад, автокореляційні методи.

2. У нерегулярних часових рядах дані записуються в неоднорідні моменти часу. Прикладами є події у фінансових ринках (покупки чи продажі активів у випадкові моменти). Для обробки таких рядів часто потрібні методи інтерполяції або спеціалізовані алгоритми, такі як Gaussian Processes.

3. Уніфіковані та розподілені часові ряди:

- Уніфіковані ряди представляють дані однієї природи (наприклад, фінансові показники, які вимірюються у валюті).
- Розподілені ряди включають змінні, що мають різну природу (наприклад, температура, вологість і кількість опадів). Аналіз таких рядів може вимагати додаткової нормалізації або уніфікації шкал.

Класифікація багатовимірних часових рядів дозволяє адаптувати методи аналізу та прогнозування до конкретних особливостей даних. Наприклад, регулярність даних дозволяє використовувати прості методи лінійного прогнозування, тоді як нерегулярні ряди можуть вимагати спеціалізованих підходів для інтерполяції або перетворення у регулярний формат.

2.2. Основні характеристики багатовимірних часових рядів

2.2.1. Тренд, сезонність та циклічність. Багатовимірні часові ряди (БЧР) складаються з різних компонентів, які визначають структуру даних та їхні динамічні особливості. Основними компонентами БЧР є тренд, сезонність та циклічність [16]. Аналіз цих компонентів дозволяє виділити базові закономірності та використовувати їх для прогнозування.

Тренд визначає довгострокову тенденцію у часі. У БЧР тренд може бути спільним для всіх змінних або специфічним для кожної змінної. Наприклад, в економічних часових рядах ВВП країни зазвичай зростає з часом, відображаючи економічний розвиток. Аналогічно, у фінансових даних можна спостерігати зростання вартості активів у довгостроковій перспективі.

Методи виділення тренду:

1. Ковзні середні (Moving Average).
2. Поліноміальна регресія для апроксимації довгострокових змін.
3. Фільтри, наприклад, Hodrick-Prescott (HP) фільтр.

У багатовимірних часових рядах тренди можуть взаємодіяти, наприклад, зростання рівня безробіття може корелювати зі спадом у виробничій сфері.

Сезонність відображає повторювані патерни у фіксовані інтервали часу, які викликані специфічними факторами. Наприклад, у сфері енергетики споживання газу зростає взимку, а в роздрібній торгівлі обсяги продажів збільшуються під час свят.

Особливості багатовимірної сезонності:

1. У деяких змінних сезонність може бути синхронною. Наприклад, в екологічних системах температура повітря і рівень води в річках мають сезонну змінність.

2. В інших випадках сезонні патерни можуть бути незалежними між змінними, що створює додаткові виклики для моделювання.

Методи моделювання сезонності:

- Гармонійний аналіз (Fourier Analysis) для виділення періодичних компонентів.
- Декомпозиція часових рядів (Seasonal Decomposition of Time Series, STL).

Циклічність пов’язана з нерегулярними коливаннями, які не мають фіксованого періоду. Ці коливання зазвичай спричинені економічними, соціальними або політичними факторами. Наприклад, економічні рецесії, що тривають кілька років, впливають на безліч змінних, таких як рівень інфляції, зайнятості та виробництва.

Основна відмінність циклічності від сезонності полягає у її неперіодичному характері. Для моделювання циклічних компонентів використовують методи аналізу трендів та економічні моделі, наприклад, модель ДСГЕ (динамічні стохастичні загальні рівноваги).

У багатовимірних часових рядах тренд, сезонність та циклічність змінних можуть впливати один на одного. Наприклад, під час економічної кризи (циклічність) сезонність витрат на свята може послабшати. Взаємозалежності компонентів потребують особливої уваги під час моделювання для забезпечення точності прогнозів.

2.2.2. Випадкові коливання та шум. Багатовимірні часові ряди (БЧР), як і будь-які дані, що відображають динаміку реальних систем, містять випадкові коливання та шум. Ці компоненти є невід'ємною частиною більшості часових рядів і відображають вплив неконтрольованих факторів або помилок вимірювання. Аналіз та обробка випадкових коливань і шуму є важливим етапом підготовки даних до моделювання [16].

Випадкові коливання представляють собою короткострокові зміни в даних, які неможливо пояснити трендом, сезонністю чи циклічністю. Ці зміни можуть бути зумовлені:

1. Випадковими зовнішніми подіями (наприклад, погодні умови, які впливають на сільськогосподарське виробництво).

2. Флуктуаціями, пов'язаними із системними процесами, що не є регулярними.

Для багатовимірних часових рядів випадкові коливання можуть проявлятися у вигляді:

- Незалежних змін, що впливають лише на одну змінну.
- Взаємозалежних коливань, що впливають на кілька змінних одночасно (наприклад, глобальні події, які змінюють кілька макроекономічних показників).

Шум – це компонента даних, яка виникає через похибки вимірювання, помилки в зборі даних або вплив незначних факторів, що не є предметом аналізу. Він може бути:

- Білим шумом – випадковими незалежними відхиленнями, які мають нормальний розподіл.
- Корельованим шумом – коли з часом з'являються залежності між шумовими компонентами.

Для багатовимірних часових рядів шум часто є залежним між змінними, особливо у випадках, коли використовуються сенсори для збору даних (наприклад, датчики у виробничих процесах).

Обробка шуму є важливим етапом підготовки даних до моделювання, оскільки високий рівень шуму може погіршити точність прогнозів. Основні методи:

1. Фільтри згладжування
   - Ковзні середні (Moving Average).
   - Експоненціальне згладжування (Exponential Smoothing).
2. Сигнальні методи
   - Фільтр Калмана (Kalman Filter), який широко використовується для очищення даних у реальному часі.
   - Вейвлет-аналіз для виділення сигналу в багатовимірних даних.
3. Моделі для врахування шуму. У багатьох моделях, таких як LSTM, шум розглядається як частина вхідних даних, і модель вчиться враховувати його вплив.

Хоча шум вважається небажаним компонентом [18], він може містити інформацію про приховані закономірності у складних системах. Наприклад, у фінансових ринках короткострокові коливання можуть сигналізувати про зміну трендів, а в медичних даних – про виникнення нових симптомів (рис. 2.2).

Випадкові коливання, з іншого боку, допомагають оцінити стійкість системи до зовнішніх впливів. Аналіз взаємозв’язків між випадковими коливаннями різних змінних дозволяє будувати більш стійкі прогнози.

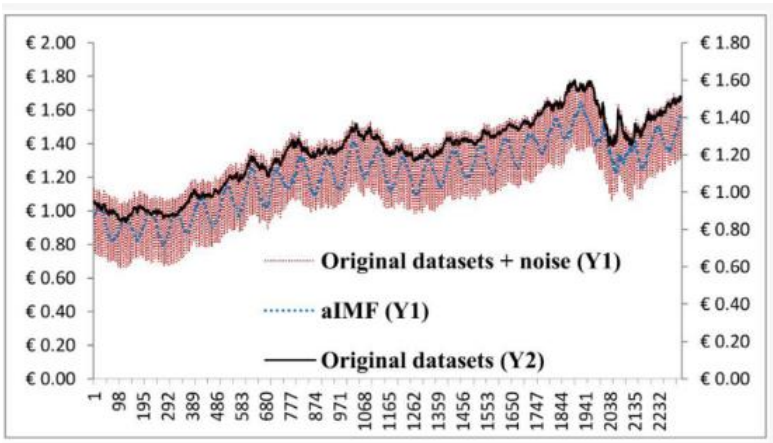

Рисунок 2.2 – Процес видалення шуму з багатовимірного часового ряду

2.3. Особливості роботи з багатовимірними часовими рядами

2.3.1. Відсутні дані та методи їх заповнення

Багатовимірні часові ряди (БЧР) часто мають пропущені значення через різноманітні фактори, такі як збої в записі даних, обмеження сенсорів чи людські помилки. Відсутні дані є одним із ключових викликів у роботі з БЧР, оскільки вони можуть спотворювати результати аналізу та знижувати точність прогнозування [17].

Причини відсутності даних:

1. Технічні збої: переривання у роботі сенсорів або систем запису даних.

2. Нерегулярність у вимірюваннях: окремі змінні можуть реєструватися з різною частотою.

3. Людські фактори: помилки під час збору або обробки даних.

4. Фізичні обмеження: наприклад, недоступність деяких показників у певний момент часу через зовнішні обставини (погодні умови, технічні обмеження).

Пропущені дані можуть викликати такі проблеми [18]:

1. Зменшення точності моделей: деякі методи машинного навчання (наприклад, RNN чи LSTM) вимагають повних наборів даних для ефективного навчання.

2. Спотворення трендів та сезонності: пропущені значення можуть змінити виявлені патерни у часових рядах.

3. Складність у моделюванні взаємозв’язків між змінними: відсутні значення порушують кореляції між змінними, що ускладнює прогнозування.

Для коректної обробки пропущених значень існує кілька підходів [16], які варіюються за складністю та точністю:

1. Прості методи:

- Заповнення середнім значенням: відсутнє значення замінюється середнім для відповідної змінної.
- Лінійна інтерполяція: значення обчислюється на основі лінійного тренду між сусідніми точками.

- Повторення попереднього значення (Forward Fill): використовується останнє доступне значення.

2. Моделі прогнозування:

- ARIMA: використовується для моделювання трендів і заповнення відсутніх значень.
- Рекурентні мережі: здатні враховувати часові залежності між змінними для передбачення пропущених значень.

3. Мультимодальні підходи:

- Метод максимальної правдоподібності (MLE): обчислення пропущених даних через максимізацію ймовірності наявних даних.
- Імпутація множинного заповнення (MICE): використання кількох моделей для заповнення пропущених значень з оцінкою невизначеності.

4. Методи на основі глибокого навчання:

- Автокодувальники (Autoencoders): моделі, які навчаються відновлювати повний часовий ряд із неповних даних.
- Трансформери з увагою: обчислюють пропущені значення на основі вагомості залежностей між змінними та часовими точками.

Таблиця 2.1

Переваги та недоліки методів заповнення

Метод	Переваги	Недоліки
Прості методи	Легкість реалізації, швидкість обчислень	Ігнорування складних взаємозв’язків між змінними
Моделі прогнозування	Враховують часові залежності	Можуть бути чутливими до шуму
Мультимодальні підходи	Висока точність, врахування невизначеності	Вимагають значних обчислювальних ресурсів
Глибокі моделі	Враховують нелінійності та взаємозалежності	Складність навчання, потреба у великій кількості даних

2.3.2. Високорозмірність та проблема "прокляття розмірності"

Багатовимірні часові ряди часто характеризуються високою розмірністю, коли кількість змінних (d) є значно більшою за кількість спостережень (n) [18]. Така ситуація створює серйозні виклики для аналізу та моделювання, відомі під загальною назвою "прокляття розмірності". Високорозмірність даних суттєво впливає на ефективність традиційних та сучасних алгоритмів, ускладнюючи їхнє навчання, прогнозування та інтерпретацію результатів.

Термін "*прокляття розмірності*" (англ. curse of dimensionality) описує проблеми, що виникають із зростанням розмірності даних [17]. Головні аспекти цієї проблеми:

1. Збільшення розмірності простору: зростання кількості змінних призводить до експоненційного збільшення простору можливих значень, що робить дані більш розрідженими.

2. Складність аналізу: зі збільшенням кількості змінних класичні методи аналізу, такі як регресія або кластеризація, стають менш точними через високий рівень шуму.

3. Втрата узагальнюваності: моделі можуть добре підлаштовуватися під навчальні дані, але втрачати здатність узагальнювати на нові дані.

4. Обчислювальна складність: із збільшенням змінних зростають вимоги до пам’яті та часу обчислень.

Наслідки високорозмірності у багатовимірних часових рядах:

1. Перенавчання (overfitting): моделі можуть запам’ятовувати шум замість виявлення загальних закономірностей.

2. Зменшення значущості окремих змінних: зі збільшенням кількості змінних корисність кожної окремої змінної може знижуватися.

3. Проблеми з візуалізацією: у багатовимірному просторі стає важко графічно відобразити взаємозалежності між змінними.

Методи вирішення проблеми "прокляття розмірності":

1. Зменшення розмірності (Dimensionality Reduction):

- Аналіз головних компонент (PCA): виділяє основні компоненти, які пояснюють найбільшу частину варіації даних.
- t-SNE (t-Distributed Stochastic Neighbor Embedding): візуалізаційний метод, що зменшує розмірність для виявлення кластерів.
- Автокодувальники (Autoencoders): нейронні мережі, які навчаються стискати дані у прихований простір меншої розмірності.

2. Відбір ознак (Feature Selection). Вибір найзначущіших змінних для аналізу, наприклад:

- Використання кореляційних коефіцієнтів для визначення залежностей між змінними.
- Метод обертового відбору (Recursive Feature Elimination, RFE).

3. Регуляризація. Додавання штрафів для великих ваг у моделях, таких як Lasso (L1-регуляризація) та Ridge (L2-регуляризація). Це дозволяє зменшити вплив незначущих змінних.

4. Кластеризація змінних. Групування змінних із подібною динамікою для створення агрегованих ознак. Наприклад, об’єднання змінних, що описують макроекономічну ситуацію, у загальний індекс.

5. Застосування сучасних підходів:

- Гібридні моделі. Використання нейронних мереж разом із методами зменшення розмірності. Наприклад, PCA для попереднього зменшення розмірності перед навчанням LSTM.
- Трансформери з механізмом уваги. Трансформери можуть адаптуватися до високорозмірних даних, виділяючи найбільш значущі часові залежності. Це допомагає уникнути перенавчання навіть за великої кількості змінних.

2.3.3. Взаємозалежності між змінними

Однією з ключових характеристик багатовимірних часових рядів (БЧР) є взаємозалежності між змінними. Ці залежності можуть бути лінійними, нелінійними або динамічними, що ускладнює їх аналіз. Виявлення та

моделювання взаємозалежностей є критично важливим завданням для підвищення точності прогнозування і розуміння структури даних [16].

Типи взаємозалежностей:

1. Лінійні взаємозалежності. Найпростіший вид залежностей, які можна визначити за допомогою кореляційного аналізу. Наприклад, рівень доходів і витрат може мати високу позитивну кореляцію.

2. Нелінійні взаємозалежності. Такі залежності є більш складними, і їх неможливо описати за допомогою простих кореляцій. Для виявлення нелінійних взаємозалежностей використовують взаємну інформацію (Mutual Information) або метод парціальних кореляцій.

3. Динамічні взаємозалежності. У БЧР змінні можуть залежати одна від одної у різні часові моменти. Наприклад, зміна облікової ставки впливає на рівень інфляції з певним запізненням.

Методи виявлення взаємозалежностей:

1. Кореляційний аналіз дозволяє оцінити ступінь лінійної залежності між змінними. Основними метриками є коефіцієнт Пірсона для лінійних залежностей та коефіцієнт Спірмена для рангових залежностей.

2. Крос-кореляція використовується для оцінки залежностей між змінними з урахуванням лагів. Це корисно для виявлення запізнених ефектів між змінними.

3. Графові методи і моделі, такі як байєсівські мережі, дозволяють виявляти причинно-наслідкові залежності між змінними у багатовимірному просторі.

4. Методи на основі ентропії. Взаємна інформація оцінює, наскільки одна змінна зменшує невизначеність іншої, що дозволяє виявляти нелінійні залежності.

Вплив взаємозалежностей на моделювання

1. Підвищення точності прогнозів. Використання взаємозалежностей між змінними дозволяє враховувати контекст та взаємний вплив змінних.

Наприклад, у макроекономічних моделях інфляція та рівень безробіття розглядаються разом для точнішого прогнозу.

2. Виявлення причинно-наслідкових зв’язків. Причинно-наслідковий аналіз допомагає зрозуміти, як одна змінна впливає на інші, що є корисним для прийняття управлінських рішень.

3. Зменшення надмірності. У БЧР часто зустрічаються змінні з високою кореляцією, які можуть дублювати інформацію. Виявлення таких змінних дозволяє зменшити розмірність даних без втрати значущої інформації.

Сучасні підходи до моделювання взаємозалежностей:

1. Векторна авторегресія (VAR) дозволяє моделювати залежності між усіма змінними в багатовимірному часовому ряді, враховуючи їхній взаємний вплив.

2. Трансформери з увагою. Механізми уваги у трансформерах дозволяють виявляти найбільш значущі взаємозалежності між змінними та часовими точками.

3. Графові нейронні мережі (Graph Neural Networks). Ці моделі об’єднують графову структуру та глибоке навчання для моделювання складних взаємозв’язків між змінними.

2.4. Перспективи подальших досліджень та застосувань

Багатовимірні часові ряди є важливим інструментом для аналізу складних систем, і розвиток методів їхнього аналізу залишається актуальним завданням сучасної науки. Постійне зростання обсягів даних, розвиток обчислювальних потужностей та вдосконалення алгоритмів машинного навчання створюють нові можливості для подальших досліджень і практичного застосування.

Основні напрями подальших досліджень:

1. Розвиток моделей для роботи з нерегулярними часовими рядами:
   - У багатьох галузях вимірювання проводяться нерегулярно, і розробка моделей, які можуть ефективно працювати з такими даними, є важливим напрямком.

- Використання трансформерів для нерегулярних даних є перспективним підходом.

2. Інтеграція глибокого навчання з класичними методами: поєднання потужності глибоких моделей, таких як LSTM і GRU, з інтерпретованістю класичних підходів (наприклад, VAR) може підвищити точність і зрозумілість прогнозів.

3. Розробка ефективних методів роботи з високорозмірними даними:

- Подальше вдосконалення методів зменшення розмірності (наприклад, автоенкодери) для збереження важливих характеристик багатовимірних даних.
- Дослідження графових нейронних мереж для моделювання залежностей між великою кількістю змінних.

4. Виявлення нелінійних причинно-наслідкових зв’язків. Розробка нових методів, що дозволяють моделювати складні нелінійні залежності між змінними, враховуючи як часові, так і причинно-наслідкові зв’язки.

5. Автоматизація аналізу БЧР. Використання AutoML (Automated Machine Learning) для автоматизації підбору моделей та параметрів у роботі з БЧР.

Виклики подальших досліджень:

1. Обчислювальна складність: зростання кількості змінних і даних вимагає вдосконалення алгоритмів для їхньої швидкої та точної обробки.

2. Проблема інтерпретованості: сучасні методи машинного навчання, такі як трансформери, є потужними, але складно інтерпретованими. Розробка методів, які забезпечують баланс між точністю та інтерпретованістю, є важливим завданням.

3. Робота з шумом: покращення методів очищення даних від шуму для збереження важливих характеристик.

Висновки за розділом 2

Робота з багатовимірними часовими рядами (БЧР) є складною, але важливою задачею, яка має широкий спектр застосувань у різних галузях, таких як економіка, медицина, енергетика та кліматологія. БЧР дозволяють моделювати складні системи, враховуючи як часові залежності кожної змінної, так і взаємозв'язки між ними. Основні аспекти роботи з такими даними можна узагальнити у кількох ключових напрямах.

БЧР складаються з багатьох змінних, які змінюються в часі та мають взаємозалежності. Класифікація БЧР базується на типах даних (континуальні, дискретні, категоріальні) та їхній структурі (регулярні, нерегулярні ряди). Розуміння структури та типу даних є критично важливим для вибору відповідних методів аналізу.

Методи виділення компонентів, такі як декомпозиція рядів, дозволяють краще зрозуміти структуру даних і підготувати їх для моделювання.

Інтеграція сучасних методів у процес роботи з БЧР дозволяє отримувати точніші прогнози, знижувати обчислювальні витрати та покращувати інтерпретацію даних.

Багатовимірні часові ряди є потужним інструментом для аналізу складних систем, але вони вимагають застосування сучасних підходів для подолання викликів, пов'язаних із високою розмірністю, пропущеними даними та нелінійними взаємозалежностями. Розуміння структури БЧР, вибір правильних методів аналізу та інтеграція машинного навчання забезпечують високу якість прогнозів та практичну цінність отриманих результатів.

РОЗДІЛ 3
РОЗРОБКА МОДЕЛЕЙ ПРОГНОЗУВАННЯ ПАРАМЕТРІВ СОЦІАЛЬНО-ЕКОНОМІЧНИХ СИСТЕМ

3.1. Модель для прогнозування параметрів одновимірного часового ряду.

Метою створення моделі для прогнозування параметрів одновимірного часового ряду є забезпечення точного передбачення майбутніх значень на основі історичних даних. У контексті даної роботи модель була розроблена для прогнозування цін на нафту марки Brent, які є важливим індикатором економічної стабільності, ринкових умов та стратегічного планування в різних галузях.

Вибір моделі LSTM (Long Short-Term Memory) зумовлений її здатністю обробляти часові ряди з довгостроковими залежностями та складними нелінійними патернами. Нафта як ресурс піддається значним коливанням, спричиненим економічними, політичними та природними факторами, що ускладнює прогнозування традиційними методами. Тому LSTM, яка забезпечує ефективну роботу з нелінійними процесами, була обрана як основа моделі.

Основними завданнями побудови моделі є:

- виявлення та врахування довгострокових залежностей у даних;
- забезпечення адаптивності до високої волатильності ринку нафти;
- отримання високоточних прогнозів для прийняття обґрунтованих управлінських рішень.

Таким чином, модель LSTM орієнтована на надання якісних інструментів для аналізу та прогнозування одновимірних часових рядів з практичним застосуванням у сфері соціально-економічного моделювання.

3.1.1. Опис набору даних

Для побудови моделі було використано відкритий датасет цін на нафту марки Brent, завантажений із платформи Kaggle (Brent Oil Prices Dataset)

[https://www.kaggle.com/datasets/mabusalah/brent-oil-prices]. Датасет містить щоденні ціни на нафту, що забезпечує високу деталізацію та достовірність даних для аналізу й моделювання.

Основні характеристики датасету:

- Структура:
  - дата **(**Date**)**: Стовпець із датами у форматах "%d-%b-%y" (наприклад, "01-Jan-20") та "%b %d, %Y" (наприклад, "Jan 01, 2020");
  - ціна **(**Price**)**: Стовпець із щоденними значеннями ціни на нафту в доларах США.
- Період покриття: Датасет охоплює тривалий період, що дозволяє аналізувати динаміку змін цін на нафту протягом кількох десятиліть.
- Частота даних: Щоденні спостереження забезпечують високу роздільну здатність часових рядів.
- Розмір вибірки: Достатній обсяг даних для тренування та тестування моделі, що забезпечує надійність результатів.

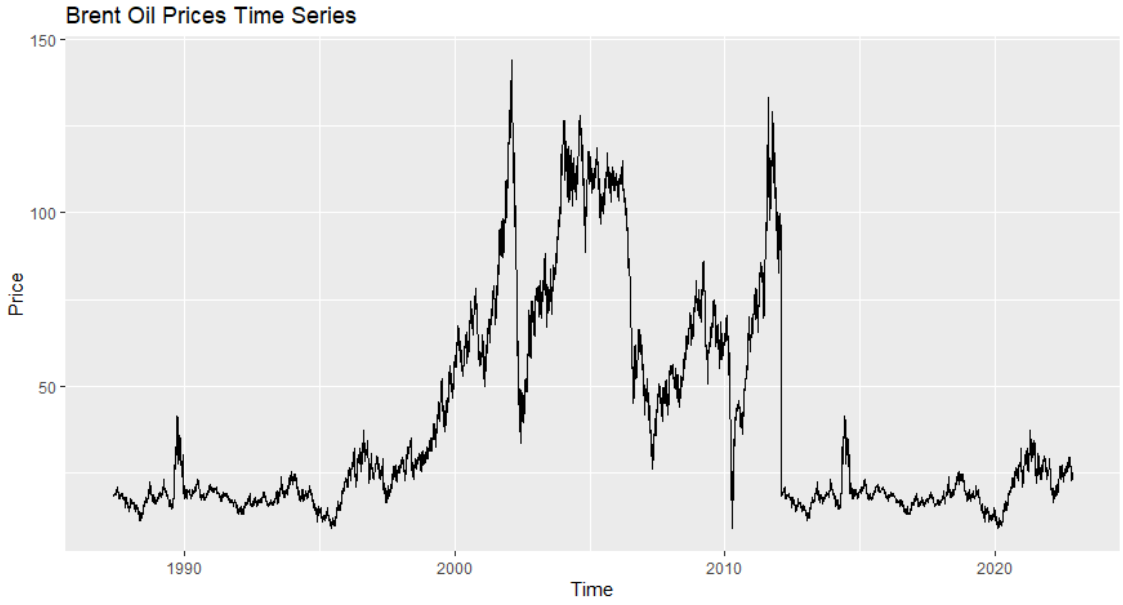

Рисунок 3.1 – Графік часового ряду Brent Oil Prices Dataset

3.1.2. Попередня обробка даних

Попередня обробка даних є ключовим етапом підготовки часових рядів для моделювання, оскільки вона забезпечує їх коректність і придатність для використання в моделі LSTM. Для роботи з датасетом цін на нафту марки Brent були виконані такі кроки:

1. Завантаження даних. Датасет був завантажений з локального файлу у форматі CSV. Для забезпечення подальшої роботи з часовими рядами дані були структуровані у вигляді таблиці з двома основними стовпцями: "Дата" та "Ціна".

2. Обробка дат. Оскільки дати у датасеті були представлені у двох різних форматах ("%d-%b-%y" та "%b %d, %Y"), було застосовано функції з пакету lubridate для обробки цих форматів. Після цього всі дати були конвертовані у стандартний формат Date, що забезпечило послідовність часових рядів.

3. Об’єднання даних. Дані з обох форматів дат були об’єднані в єдиний набір, а потім відсортовані за датою. Це гарантувало, що часовий ряд є хронологічно послідовним.

4. Перевірка на пропуски. Дані були перевірені на наявність пропущених значень. Будь-які пропуски у датасеті були видалені за допомогою функції na.omit, що забезпечило цілісність даних.

5. Перетворення у формат часового ряду. Дані були конвертовані у формат часових рядів (ts) з використанням щоденної частоти (frequency = 365). Це дозволило моделі коректно працювати з сезонними та трендовими компонентами даних.

6. Нормалізація даних. Для підвищення стабільності моделі дані були нормалізовані за допомогою стандартної шкали (Z-score normalization). Це зменшило вплив масштабів вихідних даних і поліпшило навчання моделі LSTM.

7. Розділення на тренувальні та тестові дані. Датасет був поділений на тренувальну (90% датасету) та тестову вибірки (10% датасету). Тренувальна

вибірка використовувалася для навчання моделі, а тестова — для оцінки її точності.

3.1.3. Модель прогнозування LSTM

Модель LSTM (Long Short-Term Memory), розроблена для прогнозування цін на нафту марки Brent, має архітектуру, адаптовану для аналізу одновимірного часового ряду, і налаштовані гіперпараметри, які забезпечують високу точність прогнозування.

Модель побудована за допомогою бібліотек Keras і TensorFlow та має таку *архітектуру*:

1. LSTM-шар:

- Містить 50 блоків пам'яті (units = 50), які дозволяють моделі враховувати довгострокові залежності у часових рядах.
- Вхідний розмір: один часовий крок (lag = 1) і одна ознака, представлена в тривимірному форматі (кількість зразків × кількість часових кроків × кількість ознак).

2. Dense-шар. Щільний (Dense) шар з одним вихідним нейроном для передбачення одного значення.

3. Функція втрат: mean_squared_error для мінімізації середньоквадратичної похибки.

4. Оптимізатор: найкращі результати показав оптимізатор adam, який забезпечує стабільну і швидку збіжність.

Налаштування гіперпараметрів. Під час експериментів було протестовано різні комбінації гіперпараметрів.

1. Кількість блоків (units) LSTM визначає розмір прихованого шару. Більша кількість блоків може дозволити моделі захоплювати більш складні патерни, але також збільшує обчислювальні витрати та ризик перенавчання. Діапазон перевірених значень: 10, 50, 100 блоків.

2. Кількість епох (epochs) визначає, скільки разів модель пройде через весь тренувальний набір даних. Більша кількість епох може поліпшити

точність моделі, але також може призвести до перенавчання. Діапазон перевірених значень: від 1 до 2 епох.

3. Розмір пакета (batch size) визначає кількість прикладів, що використовуються для одного оновлення ваг під час тренування. Менший розмір пакета може забезпечити більш стабільне оновлення ваг, але збільшує час тренування. Діапазон перевірених значень: 1, 10.

4. Оптимізатор (optimizer) визначає метод оновлення ваг моделі під час тренування. Вибір оптимізатора може суттєво вплинути на швидкість збіжності та точність моделі. Перевірені значення: 'adam', 'sgd', 'rmsprop', 'adagrad', 'adadelta', 'adamax', 'nadam', 'ftrl'.

Процес навчання моделі складається з кількох етапів:

1. Підготовка даних:
   - Дані були нормалізовані для зменшення впливу різниці в масштабах.
   - Тренувальна вибірка містила всі дані, крім останніх 10 спостережень, які були виділені для тестування.
   - Дані подавались у модель у вигляді тривимірного масиву.
2. Тренування:
   - Модель тренувалась на тренувальній вибірці з використанням найкращої комбінації гіперпараметрів.
   - Функція fit обчислювала втрати та оновлювала ваги моделі на кожній епосі.
3. Оцінка якості моделі. Точність моделі оцінювалась за допомогою метрик MSE, MAE і RMSE.

Модель реалізована у середовищі R із використанням бібліотек:

- Keras та TensorFlow: Для побудови та навчання моделі.
- Tidyverse та Timetk: Для підготовки даних і роботи з часовими рядами.

Повний скрипт наведено у додатку (Додаток Г) для більш детального ознайомлення.

3.1.4. Результати тестування моделі LSTM

Модель LSTM для прогнозування цін на нафту марки Brent продемонструвала високий рівень точності, що підтверджується отриманими метриками (табл. 3.1).

Таблиця 3.1

Результати найкращої конфігурації

MSE	MAE	RMSE
0.003056	0.055120	0.055283

Найкраща модель, яка отримала RMSE 0.055283, що свідчить про високу точність прогнозів, має таку конфігурацію:

- Units: 50 (кількість блоків у шарі LSTM).
- Epochs: 2 (кількість ітерацій навчання).
- Batch Size: 1 (розмір пакета для обчислення градієнтів).
- Optimizer: adam (оптимізатор, що забезпечив стабільну і швидку збіжність).

Дана конфігурація забезпечила точні прогнози з найменшими похибками, що робить її оптимальною для розв'язання задачі прогнозування одновимірного часового ряду.

Для оцінки продуктивності моделі LSTM та її здатності передбачати зміни цін на нафту марки Brent було створено кілька ключових графіків, які демонструють результати моделі.

Проведено *порівняння фактичних та прогнозованих значень* (Actual vs. Forecasted Values) (рис. 3.2, 3.3). На графіках представлені фактичні значення цін на нафту та значення, передбачені моделлю LSTM. Графік охоплює останні 10 і 100 днів із тестової вибірки. Результати показують високу відповідність прогнозованих значень фактичним, що свідчить про точність моделі.

Рисунок 3.2 – Графік передбачених значень за останні 10 днів вибірки

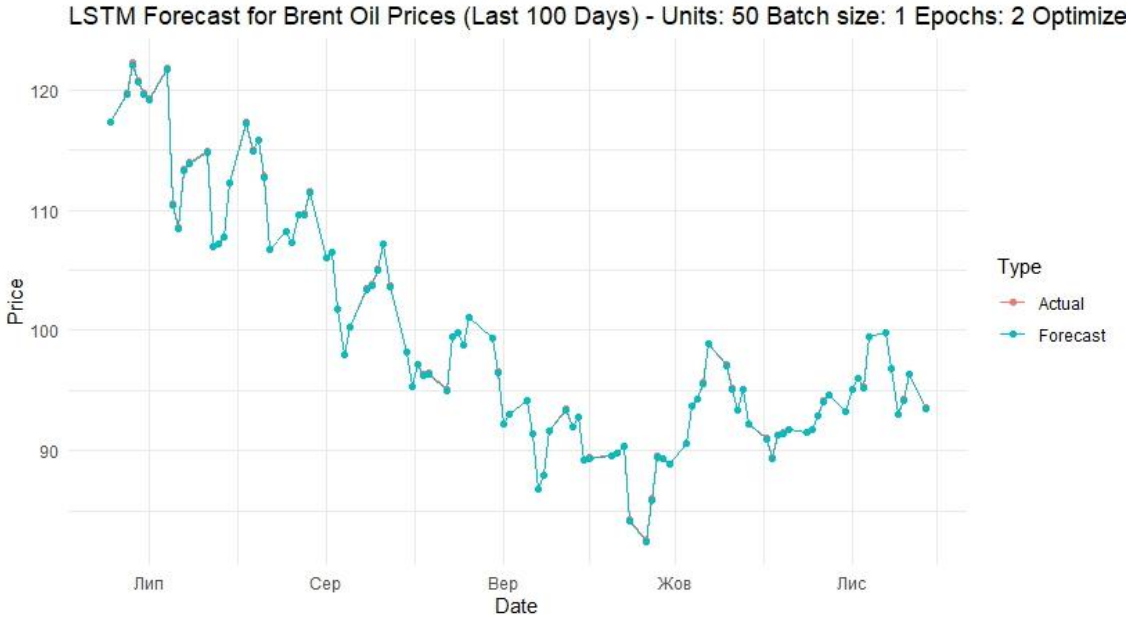

Рисунок 3.3 – Графік передбачених значень за останні 100 днів вибірки

Побудований *прогноз на 100 днів* вперед (Forecast for the Next 100 Days). Графік на рис. 3.4 ілюструє передбачення моделі на 100 днів вперед на основі історичних даних. Він демонструє стабільність прогнозів моделі навіть для довгострокових періодів, підтверджуючи її здатність враховувати довгострокові залежності у часових рядах.

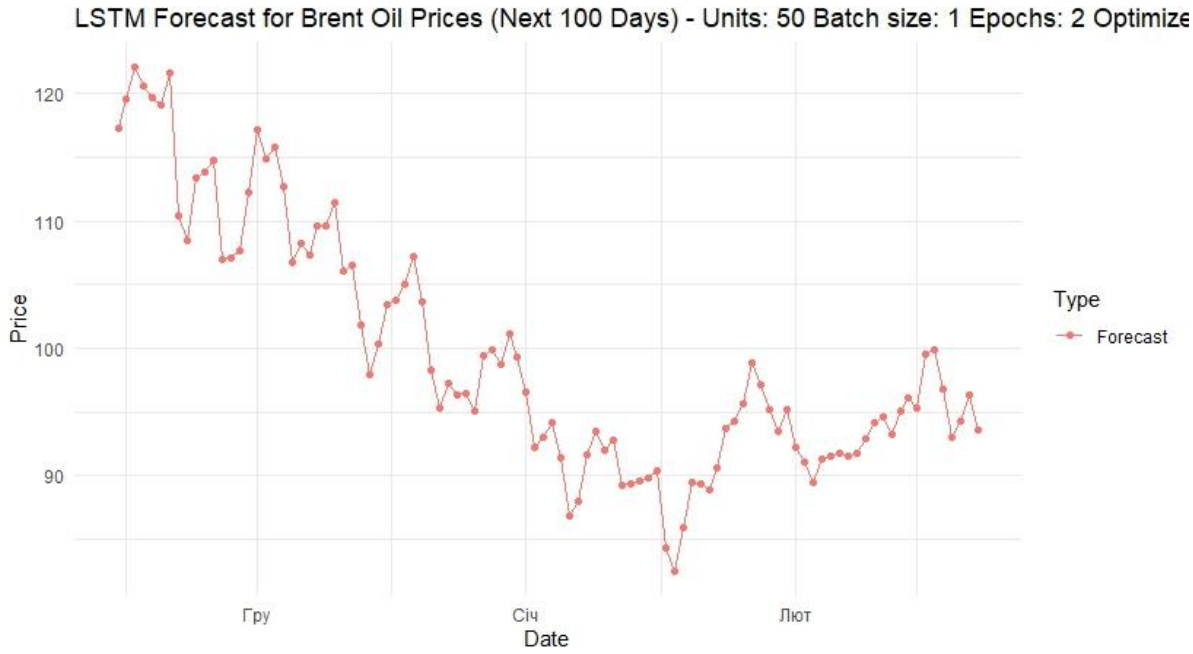

Рисунок 3.4 – Графік передбачених значень на наступні 100 днів

Побудована модель LSTM для прогнозування одновимірного часового ряду, базуючись на даних цін на нафту марки Brent, продемонструвала високу ефективність у вирішенні поставленої задачі. Опираючись на отримані результати, можна підкреслити, що:

- модель LSTM є ефективним інструментом для прогнозування параметрів одновимірного часового ряду;
- використання сучасних методів глибокого навчання дозволяє враховувати складні залежності в даних, що забезпечує значно кращі результати порівняно з традиційними моделями;
- отримані результати можуть бути застосовані не лише для прогнозування цін на нафту, а й для аналізу інших соціально-економічних часових рядів, що підкреслює універсальність моделі.

Таким чином, модель LSTM є надійним і гнучким інструментом для задач прогнозування, який може бути використаний у різних сферах, зокрема для прийняття стратегічних рішень у галузі економіки та фінансів.

3.2. Модель для прогнозування параметрів багатовимірного часового ряду

Метою створення моделі для прогнозування параметрів багатовимірного часового ряду було забезпечення високоточного передбачення офіційного курсу валют долара США та євро відносно гривні. Ця задача є актуальною для фінансового аналізу, прогнозування економічних процесів та прийняття стратегічних рішень у сфері економіки та бізнесу.

Вибір моделі LSTM (Long Short-Term Memory) для цієї задачі обумовлений її здатністю обробляти багатовимірні часові ряди, враховуючи складні залежності між кількома змінними. Врахування таких залежностей, як взаємозв'язок між курсами різних валют, дозволяє покращити точність прогнозу і підвищити адаптивність моделі до змін у ринкових умовах.

Основні завдання, які вирішує ця модель:

- Аналіз взаємозалежностей між курсами валют для покращення якості прогнозу.
- Врахування довгострокових залежностей у багатовимірних часових рядах.
- Забезпечення адаптивності моделі до високої волатильності валютного ринку.
- Надання інструменту для стратегічного планування на основі прогнозів курсу валют.

Таким чином, модель LSTM розроблена для точного прогнозування фінансових показників у складних ринкових умовах, з урахуванням нелінійних і довгострокових залежностей, що робить її надійним інструментом для фінансового аналізу.

3.2.1. Опис набору даних багатовимірного ряду

Для побудови моделі було використано датасет, що включає офіційні курси долара США та євро відносно гривні, зібрані за період із 01.01.1999 до 31.10.2024. Дані надані Національним банком України (https://bank.gov.ua/ua)

та відображають щоденні спостереження, що забезпечує високу деталізацію та точність часових рядів.

Датасет складається з трьох стовпців:

1. Дата (Date): формат дд.мм.рррр, наприклад, "01.01.1999".

2. Курс долара США до гривні (USD): числове значення, що відображає офіційний курс долара США.

3. Курс євро до гривні (EUR): числове значення, що відображає офіційний курс євро.

Особливості даних:

- Частота: Щоденні значення, включаючи будні дні.
- Дані охоплюють значний період (25 років), що включає економічні кризи, стабілізаційні періоди та волатильність валютного ринку, дозволяє аналізувати довгострокові тенденції у курсах валют.
- Деякі дати можуть мати сталий курс (наприклад, на початку війни), що було враховано під час обробки даних.

Для попередньої оцінки динаміки даних було побудовано графік всього часового ряду (рис. 3.4). Графік ілюструє загальну тенденцію зміни валютних курсів, періоди різких коливань і тривалих стабільних значень.

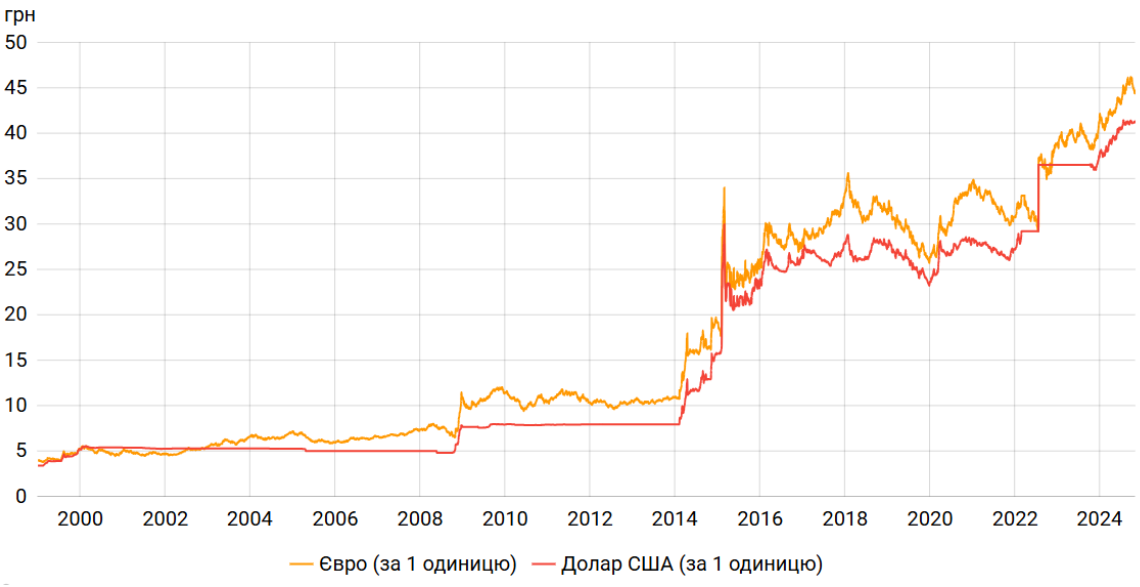

Рисунок 3.5 – Графік офіційного курсу гривні щодо іноземних валют

3.2.2. Попередня обробка даних

Попередня обробка даних є критично важливим етапом підготовки багатовимірного часового ряду для моделювання. Вона забезпечує цілісність і придатність даних для використання в моделі LSTM. На основі наданих скриптів було виконано наступні кроки:

1. Завантаження та обробка даних

- Датасет було завантажено з локального файлу у форматі CSV, використовуючи бібліотеку data.table.
- Стовпець із датами було приведено до формату Date (дд.мм.рррр), використовуючи функцію as.Date.
- Значення курсів долара США (USD) та євро (EUR) були конвертовані у числовий формат за допомогою функції as.numeric.

2. Фільтрація та видалення сталих значень. Для кожного обраного валютного курсу були видалені періоди, де значення курсу залишалося незмінним (наприклад, на початку війни для долара США). Це було реалізовано за допомогою функції filter із порівнянням поточного значення зі значенням попереднього дня.

3. Масштабування даних. Дані були нормалізовані за допомогою функції scale, що забезпечило приведення значень курсів до одного масштабу. Це покращує стабільність роботи нейронної мережі та прискорює її навчання.

4. Формування лагових змінних. Для створення прогнозів з використанням історичних даних було визначено довжину лагу (кількість попередніх значень, що використовуються для прогнозу) — 10. Було створено масиви:

- x_train: багатовимірний масив розмірністю (кількість зразків × довжина лагу × кількість ознак) для тренувальної вибірки;
- y_train: масив із цільовими значеннями для кожного лагу;
- x_test та y_test: аналогічні масиви для тестової вибірки.

5. Розділення на тренувальні та тестові дані. Дані були розділені на тренувальну (80%) та тестову (20%) вибірки. Це забезпечило достатній обсяг даних для навчання та перевірки моделі.

3.2.3. Розробка моделі LSTM для багатовимірного ряду

Модель LSTM, побудована для прогнозування багатовимірного часового ряду (курсів долара США та євро відносно гривні), була розроблена з урахуванням специфіки фінансових даних і адаптована для обробки складних залежностей між валютними курсами.

Архітектура моделі:

1. LSTM-шар:
   - Містить 100 блоків пам’яті (units), що дозволяє враховувати довгострокові залежності в даних.
   - Вхідний розмір: lag = 10 (кількість часових кроків), 1 ознака (валюта).
2. Dropout-шар. Використовувався для запобігання перенавчанню, з параметром dropout = 0.1.
3. Dense-шар - щільний шар з одним вихідним нейроном для передбачення одного значення валютного курсу.
4. Функція втрат: mean_squared_error, що мінімізує середньоквадратичну похибку.
5. Оптимізатор rmsprop, адаптований для роботи з часовими рядами, забезпечує стабільну збіжність моделі.

Для *налаштування* моделі був проведений масштабний перебір гіперпараметрів із такими варіантами:

- Units (блоки пам’яті): 16, 32, 50, 100, 150, 200.
- Epochs (кількість епох): від 1 до 75.
- Batch size (розмір пакета): 8, 16, 32, 64, 128, 256.
- Dropout (шар виключення): 0.1, 0.2, 0.3, 0.4.
- Optimizer (оптимізатор): 'adam', 'sgd', 'rmsprop', 'adagrad', 'adadelta', 'adamax', 'nadam', 'ftrl'.

- Learning rate (швидкість навчання): 0.001, 0.01, 0.1, 0.0005, 0.002.

Масштабний перебір охопив понад 432 000 різних конфігурацій моделей, що дозволило знайти оптимальні налаштування для кожної валюти.

Для побудови та навчання моделі використовувались такі *інструменти*:

- R: основна мова програмування.
- Keras: бібліотека для роботи з нейронними мережами.
- TensorFlow: обчислювальний бекенд для підтримки Keras.
- data.table: для швидкого завантаження й обробки даних.
- ggplot2: для візуалізації результатів.
- lubridate: для роботи з датами.
- dplyr: для маніпуляцій із даними.

Особливості моделі:

- архітектура моделі оптимально адаптована для роботи з фінансовими часовими рядами;
- наявність Dropout-шару забезпечує стійкість моделі до перенавчання навіть за великої кількості епох;
- використання LSTM дозволяє враховувати як внутрішньосерійні залежності (наприклад, курс однієї валюти), так і залежності між кількома валютами.

Повний скрипти побудови моделі і візуалізації графіків наведено у додатках (Додаток Д і Додаток Е відповідно) для більш детального ознайомлення.

3.2.4. Результати тестування моделі

Для оцінки продуктивності побудованої моделі LSTM були протестовані найкращі конфігурації (табл. 3.2) для кожного з часових рядів — курсу долара США та євро до гривні. Модель продемонструвала високу точність у прогнозуванні обох валютних курсів, що підтверджується отриманими метриками (табл. 3.3).

Таблиця 3.2

Найкращі конфігурації моделей

	USD	EUR
Units	100	
Epochs	44	21
Batch size	8	
Dropout	0.1	
Optimizator	rmsprop	
Learning rate	0.001	

Таблиця 3.3

Результати найкращих конфігурацій моделей

	MSE	MAE	RMSE
USD	0.0844157	0.09913248	0.2905438
EUR	0.08748121	0.1657326	0.2957722

Модель для прогнозування курсу долара (США) продемонструвала високу точність у прогнозуванні, з незначною похибкою (RMSE = 0.29), що є дуже прийнятним результатом для фінансових даних. Її MSE та MAE свідчать про мінімальні відхилення між прогнозованими та фактичними значеннями.

Результати для моделі для прогнозування курсу євро також виявилися високоточними, хоча RMSE (0.296) трохи перевищує показник для долара США. Значення MAE вказує на більш високий середній розмір помилки, що може бути пов’язано з більшою волатильністю курсу євро у порівнянні з доларом.

Для оцінки роботи моделі LSTM було створено кілька графіків, які наочно демонструють результати прогнозування та процес навчання. Ці візуалізації дозволяють глибше зрозуміти продуктивність моделі та якість отриманих прогнозів.

Графік втрат під час навчання (Training Loss Curve) (рис. 3.6, 3.7) відображає зміну функції втрат (loss) та валідаційних втрат (val_loss) на кожній епосі навчання. Графік демонструє, як модель поступово зменшує помилки під час навчання та як добре вона узгоджується з даними валідації.

Рівномірне зменшення втрат свідчить про стабільність процесу навчання та ефективність обраних гіперпараметрів.

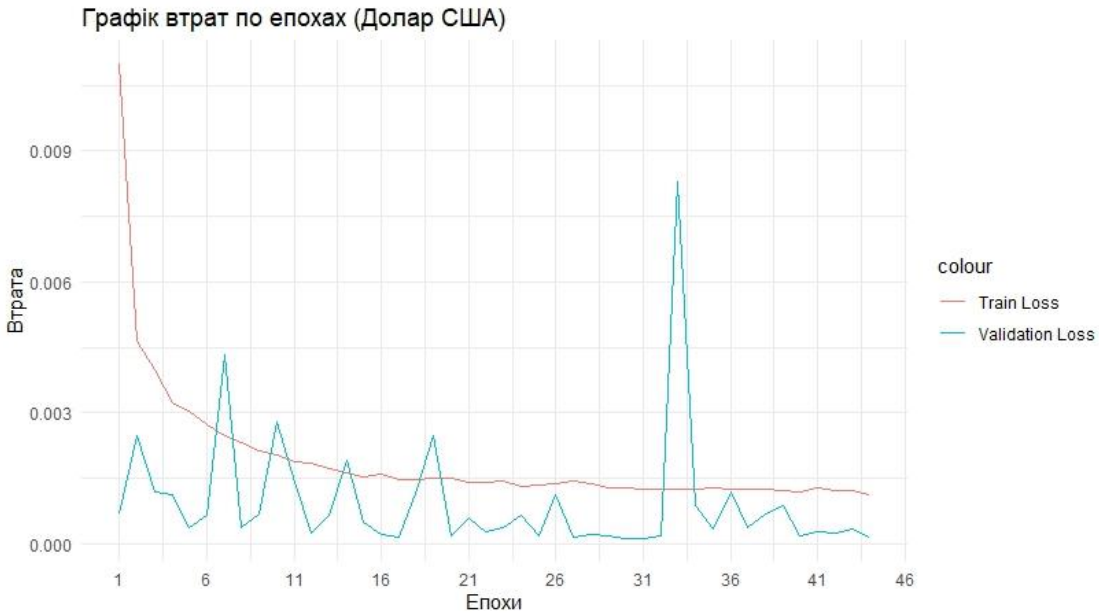

Рисунок 3.5 – Графік втрат під час навчання (USD)

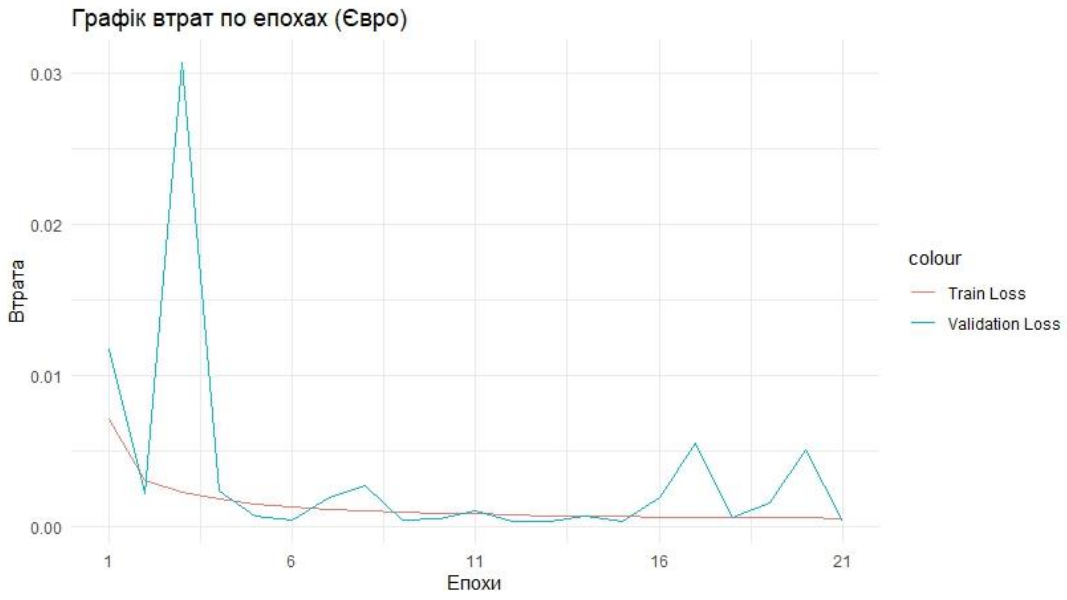

Рисунок 3.6 – Графік втрат під час навчання (EUR)

Порівняння фактичних і прогнозованих значень (Actual vs. Predicted Values) показано на рис. 3.7 – 3.10. Для останніх 10 і 100 днів тестової вибірки побудовано графіки, які ілюструють фактичні значення валютних курсів та

прогнозовані моделлю значення. Графіки на рис. 3.7, 3.8 показують високу точність короткострокового прогнозу. На них лінії фактичних і прогнозованих значень майже збігаються, що вказує на мінімальні відхилення.

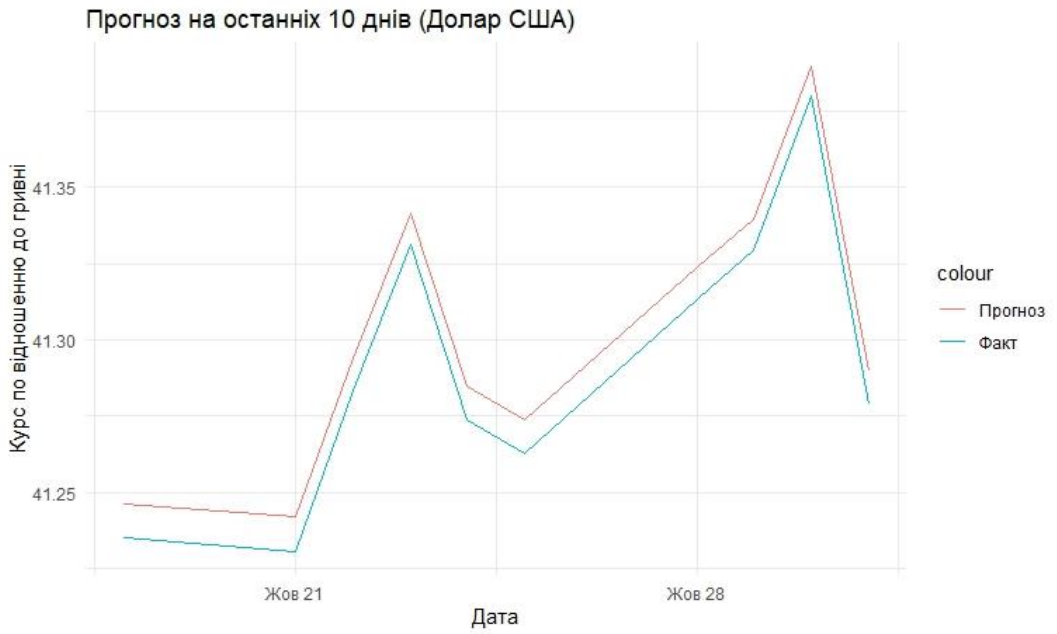

Рисунок 3.7 – Прогноз на останніх 10 днів (USD)

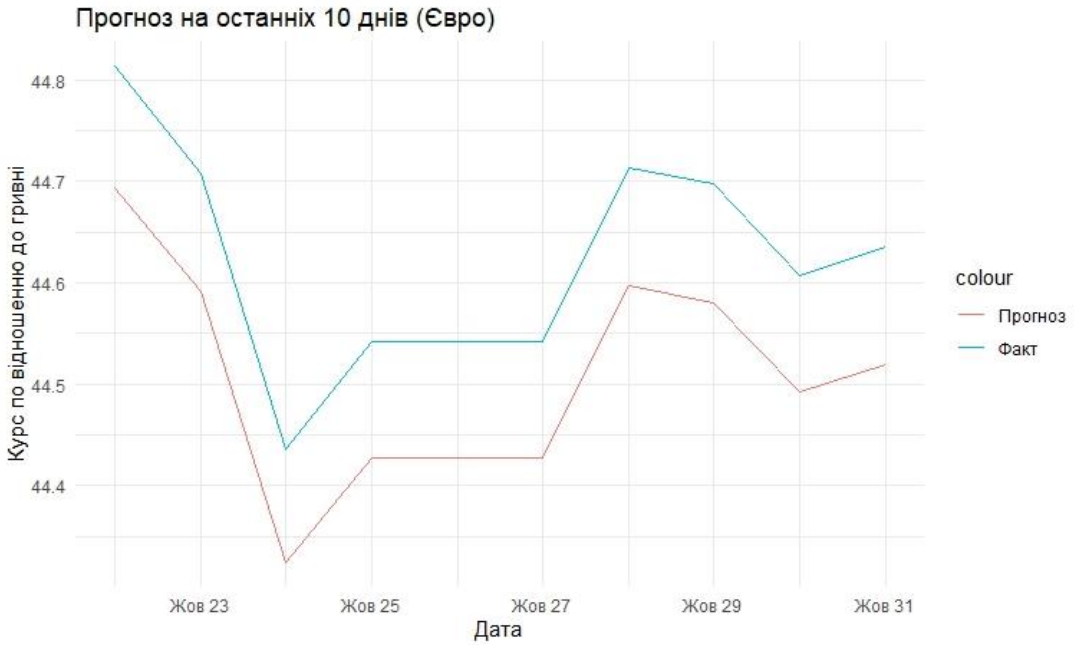

Рисунок 3.8 – Прогноз на останніх 10 днів (EUR)

Графіки на рис. 3.9, 3.10 демонструють стабільність моделі на довших інтервалах. Навіть за умов підвищеної волатильності курсів модель точно передбачає загальну тенденцію.

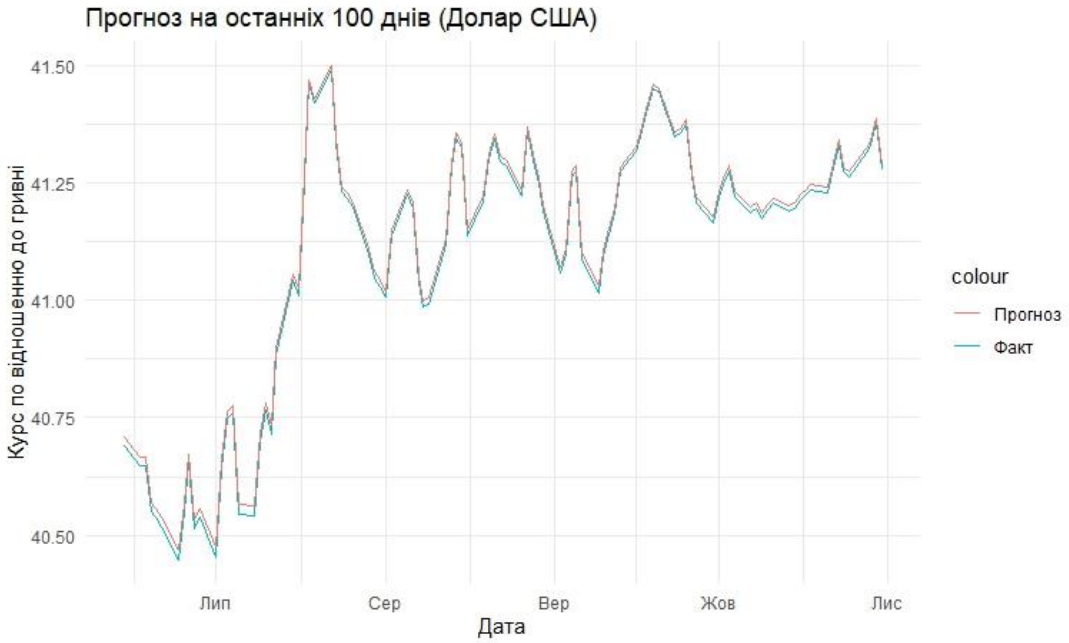

Рисунок 3.9 – Прогноз на останніх 100 днів (USD)

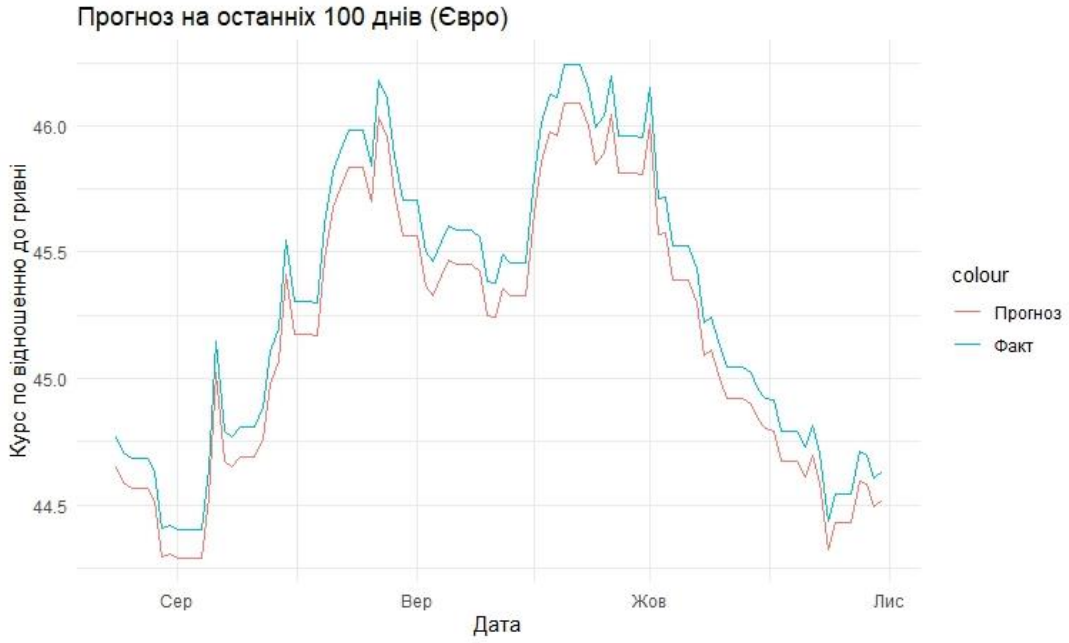

Рисунок 3.10 – Прогноз на останніх 100 днів (EUR)

Графіки *прогнозу на наступні 30 днів* (Forecast for the Next 30 Days) (рис. 3.11, 3.12) ілюструють прогноз курсу валют на майбутні 30 днів. Графік побудований з урахуванням вихідних, під час яких курс залишається сталим. Прогноз відображає тенденцію до плавних змін, характерних для валютних ринків.

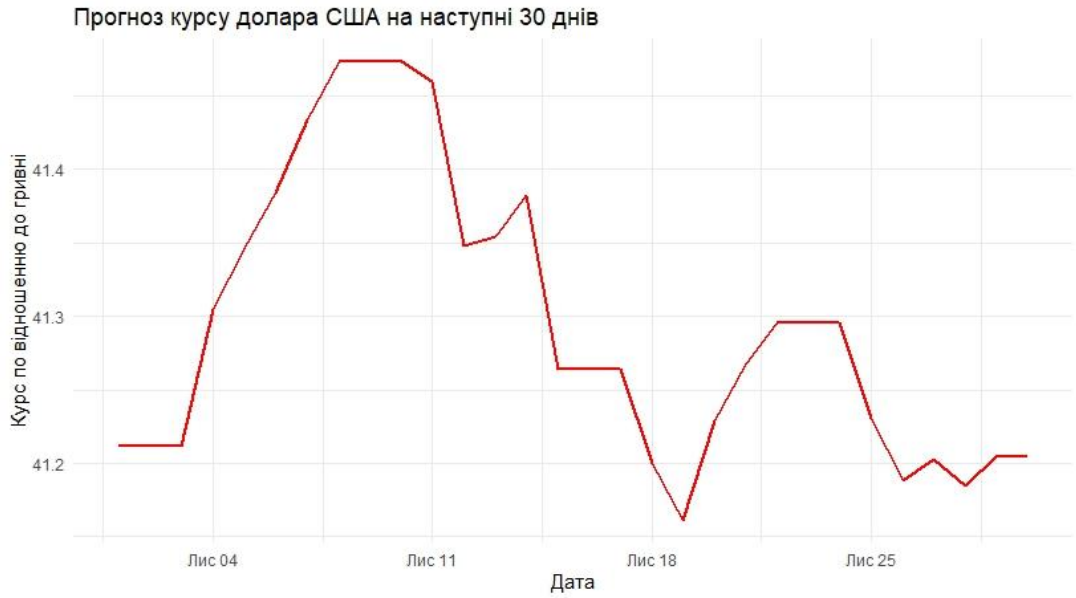

Рисунок 3.11 – Прогноз на наступні 30 днів (USD)

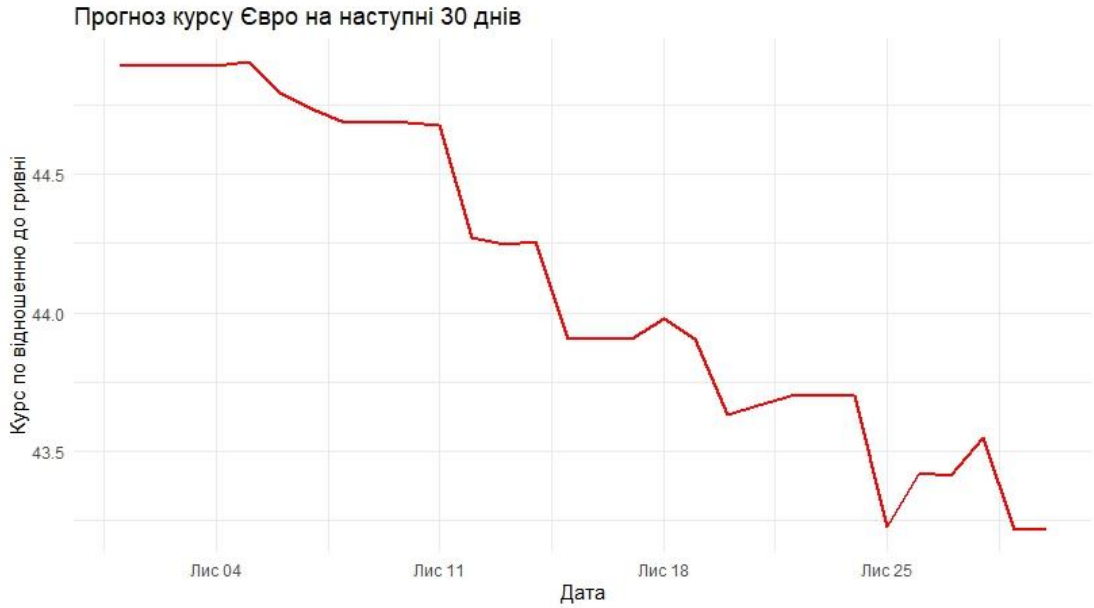

Рисунок 3.12 – Прогноз на наступні 30 днів (EUR)

Графік динаміки курсу за останні 100 днів і прогноз на 30 днів (рис. 3.13, 3.14) об’єднує історичні дані за останні 100 днів і прогноз на наступні 30 днів, дозволяючи оцінити загальну тенденцію.

Рисунок 3.13 – Прогноз на наступні 30 днів та 100 днів до цього (USD)

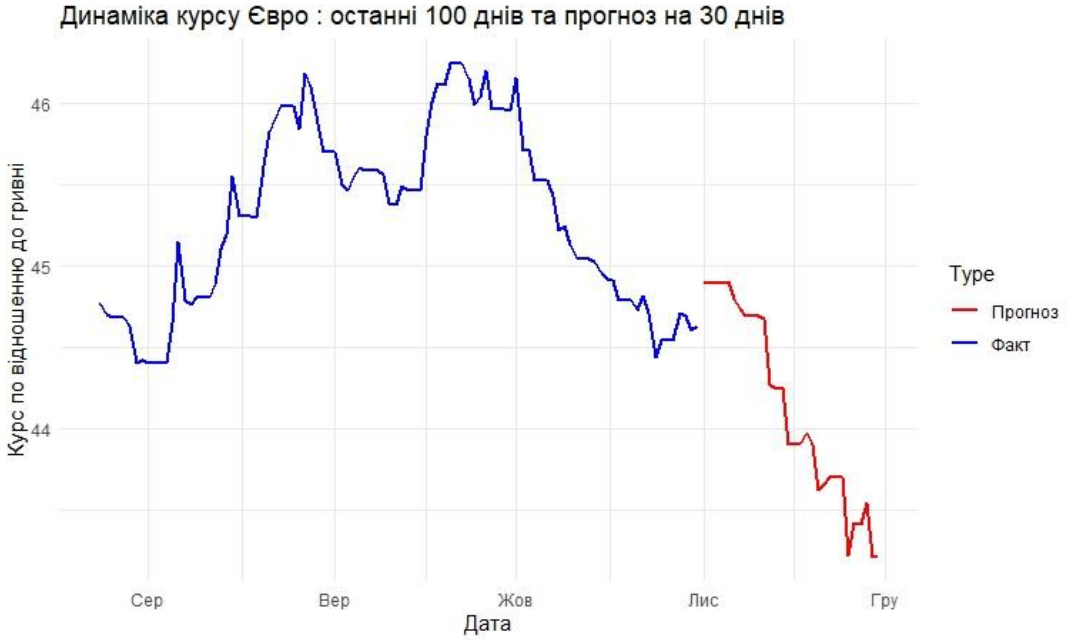

Рисунок 3.14 – Прогноз на наступні 30 днів та 100 днів до цього (EUR)

Для більш тривалого горизонту прогнозування побудовано графік *на 90 днів вперед* (Forecast for the Next 90 Days) (рис. 3.15, 3.16). Він демонструє, як модель адаптується до довгострокових змін і зберігає стабільність навіть у довгих часових інтервалах.

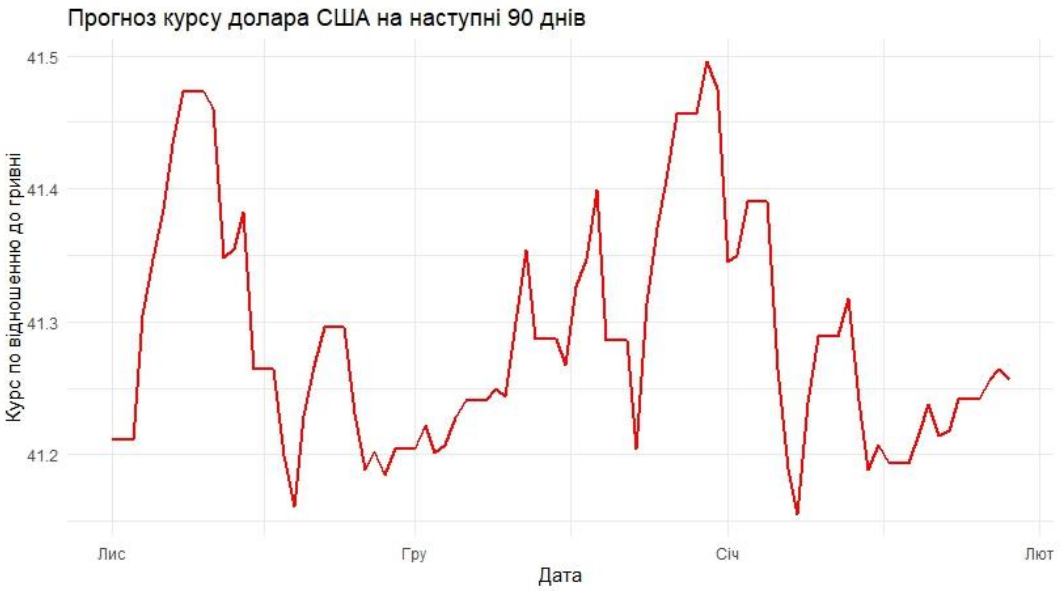

Рисунок 3.15 – Прогноз на наступні 90 днів (USD)

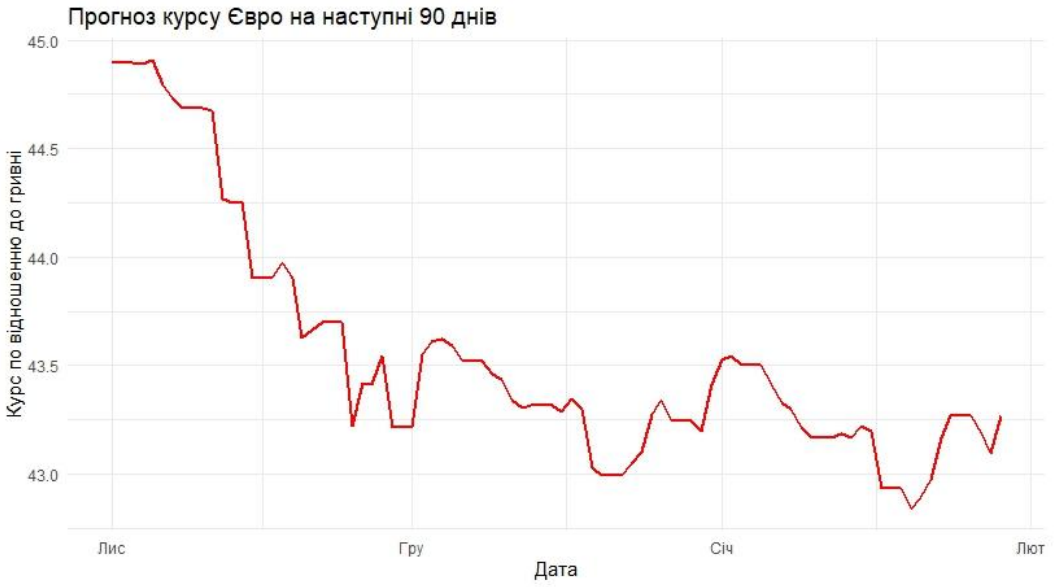

Рисунок 3.16 – Прогноз на наступні 90 днів (EUR)

Розроблена модель LSTM для прогнозування багатовимірного часового ряду продемонструвала високу ефективність у задачах прогнозування

офіційного курсу долара США та євро до гривні. На основі отриманих результатів можна зробити такі висновки:

1. Модель є ефективною. Модель успішно врахувала як короткострокові, так і довгострокові залежності між валютними курсами, що забезпечило точність прогнозів навіть за умов волатильності ринку. Метрики точності (MSE, MAE, RMSE) підтвердили стабільність і надійність моделі для обох валют. Наприклад, RMSE для долара США склала 0.29, що є дуже низьким показником для фінансових часових рядів.

2. Завдяки масштабному перебору гіперпараметрів (432 000 конфігурацій) вдалося знайти оптимальні налаштування для кожного з часових рядів, що дозволило підвищити продуктивність моделі. Найкращі конфігурації для USD та EUR включали однакову кількість блоків пам’яті (100), але різну кількість епох навчання (44 для USD та 21 для EUR), що свідчить про адаптивність моделі до різних наборів даних.

3. Побудовані прогнози є якісними. Короткострокові прогнози (на 10 днів) відзначаються високою точністю, з майже ідеальною відповідністю фактичним значенням. Довгострокові прогнози (на 30 та 90 днів) показали здатність моделі ефективно прогнозувати загальні тенденції змін валютних курсів, що робить її придатною для стратегічного фінансового планування.

4. Модель може бути використана для прогнозування інших багатовимірних фінансових часових рядів завдяки її універсальності та адаптивності. Отримані результати підкреслюють доцільність застосування LSTM для вирішення задач фінансового аналізу, зокрема для прогнозування валютних курсів, оцінки ризиків і побудови економічних стратегій.

Модель LSTM довела свою ефективність для багатовимірного прогнозування фінансових показників, забезпечивши точність, стабільність та адаптивність результатів. Отримані висновки відкривають перспективи для подальшого вдосконалення моделі, зокрема для врахування більш складних залежностей або використання більш широких наборів даних. Це підкреслює

практичну значущість розробленого підходу та його застосовність у реальних фінансових сценаріях.

3.3. Рекомендації до моделей

На основі виконаної роботи з побудови моделі LSTM для прогнозування багатовимірного часового ряду і отриманих результатів, можна сформулювати наступні рекомендації щодо вдосконалення моделі та напрями для майбутніх досліджень.

1. Розширення набору даних:

- Додаткові змінні: Включення інших економічних показників (наприклад, рівня інфляції, облікової ставки, обсягів експорту та імпорту) може допомогти врахувати зовнішні впливи на валютні курси та покращити точність прогнозів.
- Збільшення обсягу даних: Розширення історичних даних за рахунок збору інформації з інших джерел, зокрема міжнародних, дозволить моделі краще розпізнавати довгострокові тенденції.

2. Покращення архітектури моделі:

- Додавання шарів LSTM: Використання декількох LSTM-шарів із різними розмірами блоків пам’яті може підвищити здатність моделі обробляти складніші залежності.
- Дослідження інших типів нейронних мереж: Комбінація LSTM із Convolutional Neural Networks (CNN) може бути корисною для автоматичного виявлення особливостей у даних.
- Тестування інших функцій активації: Замість стандартної tanh та sigmoid можна дослідити такі функції, як ReLU або Leaky ReLU, щоб покращити здатність моделі працювати з нелінійними залежностями.

3. Оптимізація навчання:

- Удосконалення алгоритмів оптимізації: Дослідження ефективності сучасних оптимізаторів, таких як AdamW або Lookahead, може забезпечити швидшу і стабільнішу збіжність.

- Автоматизація пошуку гіперпараметрів: Застосування методів автоматичного підбору, таких як Grid Search або Bayesian Optimization, може зменшити час налаштування моделі.

4. Адаптація до реальних умов:

- Моделювання сезонності: Використання методів для обліку сезонних ефектів, таких як інтеграція з Seasonal Decomposition of Time Series (STL), дозволить покращити прогнози для даних із яскраво вираженою сезонністю.
- Облік зовнішніх подій: Інтеграція економічних новин або політичних подій як додаткових вхідних змінних може підвищити точність прогнозів.

5. Покращення оцінки результатів:

- Використання більш комплексних метрик, таких як Mean Absolute Percentage Error (MAPE) або Symmetric Mean Absolute Percentage Error (SMAPE), для кращої інтерпретації результатів.
- Проведення додаткових тестів на стійкість моделі до різних періодів, зокрема кризових.

6. Майбутні напрямки досліджень:

- Застосування трансформерів: Використання сучасних архітектур, таких як трансформери (Transformers), може значно покращити якість прогнозів завдяки їхній здатності враховувати залежності на різних часових масштабах.
- Реалізація ансамблів моделей: Об’єднання кількох моделей, таких як LSTM, ARIMA та Gradient Boosting, може створити ансамбль, здатний комбінувати переваги різних підходів.
- Аналіз взаємодій між валютами: Дослідження глибших взаємозв’язків між валютними парами дозволить краще зрозуміти поведінку ринку.

Висновки за розділом 3

Було побудовано дві моделі прогнозування параметрів соціально-економічних систем: для одновимірного та багатовимірного часових рядів.

Виконаний аналіз та результати моделювання дозволяють сформулювати ряд висновків.

Обидві моделі, побудовані на основі нейронних мереж LSTM, показали високу точність прогнозування завдяки здатності враховувати довгострокові залежності та нелінійні патерни в даних.

Завдяки масштабному перебору гіперпараметрів було знайдено оптимальні конфігурації для кожної моделі, що дозволило досягти високої продуктивності.

Побудовані графіки підтвердили здатність моделей передбачати як короткострокові, так і довгострокові зміни. Особливо корисними виявились прогнози на 30 та 90 днів вперед, які ілюструють стабільність і надійність моделей у довгострокових періодах.

Розроблені моделі можуть бути інтегровані в системи аналітики та управління, наприклад, для прогнозування фінансових показників, оцінки ризиків або стратегічного планування.

Моделі є універсальними і можуть бути адаптовані для прогнозування інших соціально-економічних показників, таких як рівень інфляції або обсяг експорту.

Моделі залежні від якості даних і можуть потребувати доопрацювання для врахування зовнішніх факторів, таких як політичні події або макроекономічні зміни.

Подальший розвиток моделей може включати використання сучасних архітектур, таких як трансформери, або поєднання декількох моделей у єдиний ансамбль.

Можна підьвердити, що методи глибокого навчання, зокрема нейронні мережі LSTM, є ефективним інструментом для прогнозування соціально-економічних параметрів. Розроблені моделі забезпечують високу точність, гнучкість і адаптивність до різних умов, що робить їх надійним рішенням для аналізу та прогнозування в сучасних динамічних умовах.

ВИСНОВКИ

У кваліфікаційній роботі виконано розробку ефективної моделі прогнозування, яка враховує складні залежності та динамічність соціально-економічних показників. Проведений аналіз підтвердив актуальність використання сучасних методів глибокого навчання, зокрема нейронних мереж типу LSTM, для задач прогнозування часових рядів, таких як ціни на нафту та валютні курси.

Основні результати роботи:

1. Теоретичний аналіз методів прогнозування: Розглянуто сучасні підходи до аналізу та прогнозування часових рядів, зокрема статистичні методи, моделі машинного навчання та гібридні підходи. Визначено переваги і обмеження кожного методу для застосування в соціально-економічних системах.

2. Розробка моделей прогнозування: Запропоновані моделі для одновимірного (ціна нафти марки Brent) та багатовимірного (валютні курси USD та EUR) часових рядів. Оптимізація гіперпараметрів моделей дозволила досягти високої точності прогнозів.

3. Експериментальна оцінка: Тестування моделей продемонструвало їх ефективність за метриками MSE, MAE, RMSE, що свідчить про високу якість прогнозів та доцільність використання моделей у реальних сценаріях.

4. Практична значущість: Результати роботи можуть бути інтегровані в автоматизовані системи управління, фінансові аналітичні платформи та інструменти стратегічного планування.

Запропонована методологія та розроблені моделі можуть бути ефективними інструментами для прогнозування соціально-економічних показників. Подальші дослідження можуть бути спрямовані на розширення застосування моделей на інші типи часових рядів, врахування нових даних та адаптацію до змінюваних умов економічного середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735 (дата звернення 12.10.2024)

2. Box, G. E., Jenkins, G. M., & Reinsel, G. C. (2015). Time Series Analysis: Forecasting and Control (5th ed.). Wiley. https://doi.org/10.1002/9781118619193 (дата звернення 12.10.2024)

3. Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. Journal of Statistical Software, 27(3), 1-22. https://doi.org/10.18637/jss.v027.i03 (дата звернення 12.10.2024)

4. Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159-175. https://doi.org/10.1016/S0925-2312(01)00702-0 (дата звернення 12.10.2024)

5. Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. PLoS ONE, 13(3), e0194889. https://doi.org/10.1371/journal.pone.0194889 (дата звернення 12.10.2024)

6. Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. International Journal of Forecasting, 36(1), 75-85. https://doi.org/10.1016/j.ijforecast.2019.03.017 (дата звернення 12.10.2024)

7. Lawrence, M. J., Goodwin, P., O'Connor, M., & Önkal, D. (2006). Judgmental forecasting: A review of progress over the last 25 years. International Journal of Forecasting, *22*(3), 493-518. https://doi.org/10.1016/j.ijforecast.2006.03.007 (дата звернення 12.10.2024)

8. Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. Neural Computation, *12*(10), 2451-2471. https://doi.org/10.1162/089976600300015015 (дата звернення 12.10.2024)

9. Zhou, Z. H. (2012). Ensemble methods: Foundations and algorithms. Chapman and Hall/CRC. https://doi.org/10.1201/b12207 (дата звернення 12.10.2024)

10. Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice (2nd ed.). OTexts. https://otexts.com/fpp3/ (дата звернення 12.10.2024)

11. Siami-Namini, S., Tavakoli, N., & Siami Namin, A. (2019). A comparison of ARIMA and LSTM in forecasting time series. IEEE International Conference on Machine Learning and Applications (ICMLA), 1394-1401. https://doi.org/10.1109/ICMLA.2018.00227 (дата звернення 12.10.2024)

12. Bouktif, S., Fiaz, A., Ouni, A., & Serhani, M. A. (2018). Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. Energies, 11(7), 1636. https://doi.org/10.3390/en11071636 (дата звернення 12.10.2024)

13. Nelson, D. M., Pereira, A. C. M., & de Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. Proceedings of the International Joint Conference on Neural Networks *(*IJCNN*)*, 1419-1426. https://doi.org/10.1109/IJCNN.2017.7966019 (дата звернення 12.10.2024)

14. Shahid, F., Zameer, A., & Muneeb, M. (2020). Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM using sequential and statistical data. Chaos, Solitons & Fractals, *140*, 110212. https://doi.org/10.1016/j.chaos.2020.110212 (дата звернення 12.10.2024)

15. Dozdar Mahdi Ahmed, Masoud Muhammed Hassan and Ramadhan J. Mstafa (2022). A Review on Deep Sequential Models for Forecasting Time Series Data. https://doi.org/10.1155/2022/6596397 (дата звернення 12.10.2024)

16. Lai, G., Chang, W.-C., Yang, Y., & Liu, H. (2018). Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. https://arxiv.org/pdf/1703.07015 (дата звернення 14.11.2024)

17. ScienceDirect. (n.d.). Multivariate Time Series. Retrieved from https://www.sciencedirect.com/topics/computer-science/multivariate-time-series (дата звернення 14.11.2024)

18. Bansal, P., Deshpande, P., & Sarawagi, S. (2021). Missing Value Imputation on Multidimensional Time Series. https://arxiv.org/pdf/2103.01600 (дата звернення 14.11.2024)

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Магістр**
Галузь знань: 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність: 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітня програма «Комп'ютеризовані системи управління та автоматика»

ЗАТВЕРДЖУЮ
в.о. завідувача комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«12» вересня 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Бондаренка Костянтина Володимировича
(прізвище, ім'я, по батькові студента)

1. Тема роботи **Модель прогнозування параметрів соціально-економічних систем**

керівник роботи Стрілець Вікторія Євгенівна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету ***№ 4101-5/3657 від 12 листопада 2024 року***

2. Строк подання студентом роботи ***30 листопада 2024 року***

3. Перелік питань, які потрібно розробити
1. Аналіз відомих підходів і методів до розв'язання задачі прогнозування станів динамічних систем.
2. Розробка моделей прогнозування параметрів динамічних соціально-економічних систем.
3. Дослідження ефективності та якості моделей прогнозування параметрів соціально-економічних систем.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Проведення літературного огляду з проблематики моделей прогнозування характеристик динамічних систем	05.09.2024 — 15.09.2024
2	Огляд моделей прогнозування характеристик динамічних систем та алгоритми їх застосування	16.09.2024 — 30.09.2024
3	Пошук даних для тестування моделей динамічних систем	01.10.2024 — 10.10.2024
4	Розробка моделей прогнозування характеристик динамічних систем у соціально-економічній сфері	11.10.2024 — 31.10.2024
5	Аналіз результатів тестування та визначення ефективності використання моделей прогнозування характеристик динамічних систем	01.11.2024 — 10.11.2024
6	Розробка рекомендацій щодо застосування моделей прогнозування характеристик динамічних систем у соціально-економічній сфері	11.11.2024 — 18.11.2024
7	Оформлення пояснювальної записки	19.09.2024 — 23.10.2024
8	Оформлення звіту за результатами преддипломної практики	24.11.2024 - 30.11.2024
9	Представлення кваліфікаційної роботи керівнику та рецензенту	24.11.2024 - 30.11.2024

5. Дата видачі завдання ***05 вересня 2024року.***

Студент К.В. Бондаренко _____
ініціали, прізвище підпис

Керівник роботи В.Є. Стрілець _____
ініціали, прізвище підпис

Додаток Б

Технічне завдання

на розробку програмного виробу «Модель прогнозування параметрів соціально-економічних систем»

Назва розділу	Назва і зміст підрозділу
1. Введення	1.1. Назва програмного виробу: Модель прогнозування параметрів соціально-економічних систем. 1.2. Галузь застосування: прогнозування економічних показників для підтримки стратегічного планування.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 174 – «Автоматизація, комп'ютерно-інтегровані технології та робототехніка». 2.2. Завдання на кваліфікаційну роботу магістра затверджено наказом №4101-5/3657 від 12.11.2024 р.
3.Призначення розробки	3.1. Мета розробки програмного виробу: розробка та тестування моделі прогнозування на основі LSTM для підвищення точності прогнозування економічних показників. 3.2. Призначення програмного виробу: створену модель можна використовувати для прогнозування параметрів, таких як ціни на нафту та курси валют, що сприяє прийняттю ефективних управлінських рішень. 3.3. Вихідні дані для розробки: реальні економічні часові ряди, отримані з відкритих джерел, включаючи офіційні курси валют (USD, EUR) та ціни на нафту марки Brent.

<table>
<tr><td>4. Технічні вимоги до програмного виробу</td><td colspan="2">4.1. Вимоги до функціональних характеристик: прогнозування економічних показників із використанням методів глибокого навчання.
4.2. Вимоги до надійності: забезпечення стабільності роботи моделі при використанні різних конфігурацій даних.
4.3. Вимоги до умов експлуатації: наявність встановленого середовища R, бібліотек keras, TensorFlow, ggplot2.
4.4. Вимоги до складу і параметрів технічних засобів: комп’ютер із процесором не нижче Intel Core i5 та оперативною пам’яттю 8 ГБ.
4.5. Вимоги до інформаційної та програмної сумісності: ОС Windows 10 або вище, підтримка середовища R.
4.6. Вимоги до маркування та упаковки: не вимагаються.
4.7. Вимоги до транспортування і зберігання: не вимагаються.</td></tr>
<tr><td>5. Вимоги до програмної документації.</td><td colspan="2">Документація до виробу включає:
1. Технічне завдання (Додаток Б до пояснювальної записки).
2. Програму і методику випробувань (Додаток В до пояснювальної записки).
3. Опис моделі (розділ 3 пояснювальної записки).
4. Код програми (Додатки Г, Д, Е до пояснювальної записки).</td></tr>
<tr><td>6. Техніко економічні показники</td><td colspan="2">Якість моделі прогнозування оцінюється за метриками MSE, RMSE та MAE.</td></tr>
<tr><td rowspan="2">7. Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>01.05.2023 – 01.09.2024

01.09.2024 – 15.09.2024

15.09.2024 – 01.10.2024</td><td>1. Аналіз наукової літератури з методів прогнозування економічних показників.
2. Підготовка даних і формування набору для навчання.

3. Розробка та тестування моделі прогнозування одновимірного часового ряду.</td></tr>
</table>

	01.10.2024 – 15.10.2024 15.10.2024 – 15.11.2024 15.11.2024 – 28.11.2024	4. Розробка та тестування моделі прогнозування багатовимірного часового ряду. 5. Аналіз отриманих результатів, визначення оптимальних конфігурацій моделі. 6. Підготовка рекомендацій для практичного використання моделі.
8. Порядок контролю і приймання	Загальні вимоги до приймання розробленого програмного виробу: 1. Перевірка роботи моделі на тестових даних. 2. Випробування моделі відповідно до програми випробувань. 3. Захист розробленого виробу на засіданні атестаційної комісії. 4. Пояснювальну записку надати у друкованому та електронному форматах.	

Виконавець:
студент групи КУ-61
Бондаренко К.В.

_____

Замовник:
кандидат технічних наук
Стрілець В.Є.

_____

Додаток В

Програма і методика випробувань програмного виробу

«Модель прогнозування параметрів соціально-економічних систем»

1. Об'єкт випробувань

1. Назва програмного виробу: «Модель прогнозування параметрів соціально-економічних систем».
2. Галузь застосування: Програмна реалізація методів прогнозування часових рядів для соціально-економічних систем.

2. Мета випробувань

Перевірка відповідності функціональності розробленої моделі прогнозування параметрів соціально-економічних систем заявленим можливостям у технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

3.1 Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

3.2 Місце і тривалість випробувань

Приймальні випробування проводяться на базі комп'ютерного класу кафедри або приватного приміщення в період роботи атестаційної комісії.

3.3 Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

3.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна задовольняти наступним вимогам:

- Працювати у середовищі R із використанням бібліотек Keras, TensorFlow, ggplot2.
- Забезпечувати точність прогнозування відповідно до метрик MAE, MSE, RMSE.
- Підтримувати обробку як одновимірних, так і багатовимірних часових рядів.
- Передбачати стабільну роботу під час тестування різних конфігурацій даних.
- Бути документованою з урахуванням вимог ЕСПД.

5. Вимоги до програмної документації

До програмної документації належать:

- Технічне завдання на розробку програми (Додаток Б до пояснювальної записки).
- Програма і методика випробувань (Додаток В).
- Код програмного виробу (Додаток Г).
- Результати тестування моделей (розділ 3 пояснювальної записки).

6. Засоби і порядок випробувань

6.1. Засоби випробувань

Для проведення випробувань необхідний комп’ютер із встановленим середовищем R і бібліотеками Keras, TensorFlow, ggplot2, а також економічні дані у форматі CSV (BrentOilPrices.csv, USD_EUR.csv).

6.2. Порядок проведення випробувань

Випробування проходять у два етапи:

– Підготовчий етап (перевірка середовища виконання, завантаження даних, перевірка відповідності документації).

– Виконавчий етап (тестування моделей із одновимірними та багатовимірними часовими рядами, оцінка точності прогнозів).

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1. Перевірку наявності та правильності встановлення необхідних бібліотек (Keras, TensorFlow, ggplot2) у середовищі R.

2. Перевірку відповідності середовища виконання технічним вимогам (ОС, апаратні характеристики).

3. Завантаження та коректність обробки вхідних даних із використанням наданих CSV-файлів.

4. Перевірку відповідності документації програмного виробу технічному завданню.

5. Аналіз структури вхідних даних та перевірку наявності пропущених значень, некоректних записів чи сталих компонентів.

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1. Тестування моделі для одновимірного часового ряду:
   - Перевірка роботи моделі з даними про ціни на нафту марки Brent.
   - Оцінка точності прогнозів за метриками MAE, MSE, RMSE.
   - Візуалізація результатів прогнозування та аналіз графіків.
2. Тестування моделі для багатовимірного часового ряду:
   - Перевірка роботи моделі з даними про валютні курси USD та EUR.
   - Оцінка взаємозв’язків між змінними у багатовимірних рядах.
   - Аналіз точності прогнозів та побудова графіків результатів.

3. Тестування стабільності моделі при зміні конфігурацій гіперпараметрів (units, batch size, epochs, dropout).
4. Аналіз впливу вибору гіперпараметрів на точність моделі.

Тести випробувань програмного виробу

1. Перевірка середовища виконання та підготовки даних

Мета тесту: Перевірити відповідність середовища виконання програмного виробу технічним вимогам і правильність обробки вхідних даних.

Вхідні дані:

– Файл BrentOilPrices.csv для одновимірного часового ряду.

– Файл USD_EUR.csv для багатовимірного часового ряду.

Кроки тестування:

1. Завантажити бібліотеки Keras, TensorFlow, ggplot2 у середовищі R.

2. Завантажити вхідні файли та перевірити їхню коректність (формат дати, числові значення, відсутність пропущених даних).

3. Виконати нормалізацію даних за допомогою функції scale().

4. Перевірити створення лагових змінних і поділу даних на тренувальну та тестову вибірки.

Очікуваний результат:

– Усі бібліотеки коректно завантажені.

– Вхідні дані без пропущених значень, мають правильний формат.

– Лагові змінні створені, дані успішно розділені на тренувальні та тестові вибірки.

```
Версії встановлених бібліотек:
> libraries_info <- data.frame(
+   Library = c("keras", "tensorflow", "ggplot2", "data.table"),
+   Version = c(
+     packageVersion("keras"),
+     packageVersion("tensorflow"),
+     packageVersion("ggplot2"),
+     packageVersion("data.table")
+   )
+ )
> print(libraries_info)
     Library Version
1      keras  2.13.0
2 tensorflow  2.15.0
3    ggplot2   3.5.1
4 data.table  1.15.4
```

Рисунок В.1 – Перевірка версій встановлених бібліотек

```
> # Створення таблиці перевірок
> checks <- data.frame(
+   Check = c(
+     "Чи є пропущені значення в даних",
+     "Чи всі дати мають правильний формат",
+     "Чи всі ціни мають правильний формат"
+   ),
+   Status = c(
+     ifelse(sum(is.na(oil_data)) == 0, "Пройдено", "Не пройдено"),
+     ifelse(!any(is.na(oil_data$Date)), "Пройдено", "Не пройдено"),
+     ifelse(!any(is.na(oil_data$Price)), "Пройдено", "Не пройдено")
+   )
+ )
> cat("Результати перевірок:\n")
Результати перевірок:
> print(checks)
                                  Check      Status
1     Чи є пропущені значення в даних Не пройдено
2 Чи всі дати мають правильний формат Не пройдено
3 Чи всі ціни мають правильний формат    Пройдено
```

Рисунок В.2 – Перевірка пропусків і формату завантажених даних

```
> # Таблиця результатів підготовки даних
> data_checks <- data.frame(
+   Check = c(
+     "Чи дані успішно розділені на тренувальні вибірки",
+     "Чи дані успішно розділені на тестові вибірки",
+     "Чи лагові змінні створені успішно"
+   ),
+   Status = c(
+     ifelse(!any(is.na(x_train)) & !any(is.na(y_train)), "Пройдено", "Не пройдено"),
+     ifelse(!any(is.na(x_test)) & !any(is.na(y_test)), "Пройдено", "Не пройдено"),
+     ifelse(dim(x_train)[1] == length(y_train), "Пройдено", "Не пройдено")
+   )
+ )
> cat("Результати перевірок даних:\n")
Результати перевірок даних:
> print(data_checks)
                                        Check   Status
1 Чи дані успішно розділені на тренувальні вибірки Пройдено
2     Чи дані успішно розділені на тестові вибірки Пройдено
3                Чи лагові змінні створені успішно Пройдено
```

Рисунок В.3 – Перевірка змінних і розподілу тестової і тренувальної вибірки

2. Тестування моделі для одновимірного часового ряду

Мета тесту: Перевірити роботу моделі LSTM для прогнозування цін на нафту марки Brent.

Вхідні дані: Підготовлений часовий ряд цін на нафту.

Кроки тестування:

1. Налаштувати модель із конфігурацією: units = 50, epochs = 1, batch_size = 10, optimizer = 'adam'.

2. Навчити модель на тренувальних даних.

3. Виконати прогноз на тестових даних.

4. Розрахувати метрики MAE, MSE, RMSE для оцінки точності.

5. Побудувати графік фактичних та прогнозованих значень.

Очікуваний результат:

– Модель успішно навчається та прогнозує дані.

– Метрики точності знаходяться в межах прийнятних значень.

– Графік демонструє відповідність прогнозу фактичним даним.

```
Best Model: Units: 50 Batch size: 10 Epochs: 1 Optimizer: adam
> cat("MSE:", results[[best_model_name]]$mse, "\n")
MSE: 0.8681222
> cat("MAE:", results[[best_model_name]]$mae, "\n")
MAE: 0.9294543
> cat("RMSE:", results[[best_model_name]]$rmse, "\n")
RMSE: 0.9317308
```

Рисунок В.4 – Метрики точності заданої конфігурації моделі

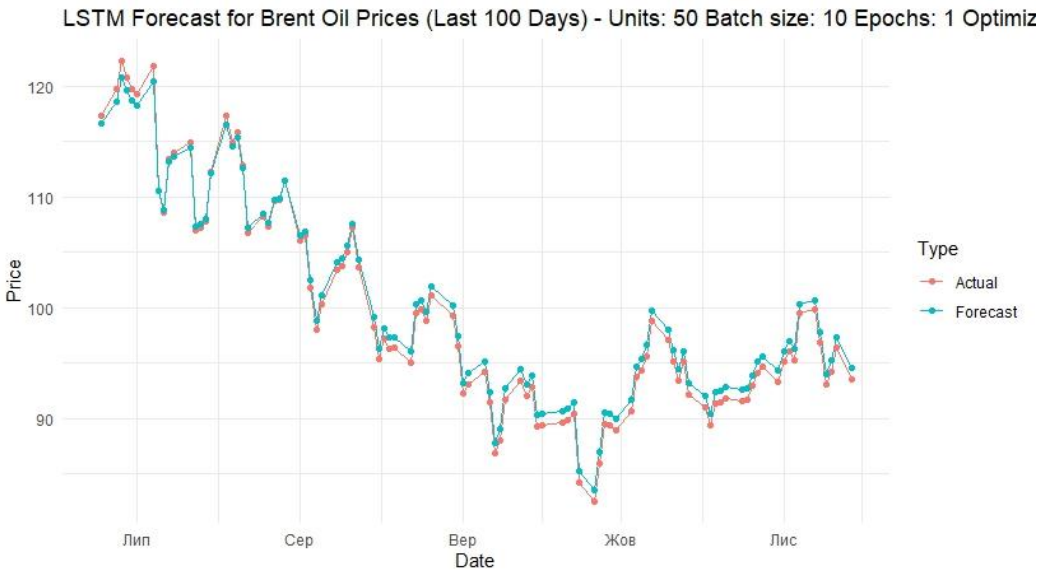

Рисунок В.5 – Графік прогнозу останніх 100 днів вибірки для тестової моделі

3. Тестування моделі для багатовимірного часового ряду

Мета тесту: Перевірити роботу моделі LSTM для прогнозування валютних курсів USD та EUR.

Вхідні дані: Підготовлений багатовимірний часовий ряд валютних курсів.

Кроки тестування:

1. Налаштувати модель із конфігурацією: units = 16, epochs = 75, batch_size = 8, dropout = 0.1, optimizer = ' adam'.

2. Навчити модель на тренувальних даних.

3. Виконати прогноз на тестових даних.

4. Побудувати графік фактичних і прогнозованих значень для USD та EUR.

Очікуваний результат:

– Модель успішно обробляє багатовимірні дані.

– Будується графік прогнозу за 100 останніх днів тестової вибірки для обох змінних.

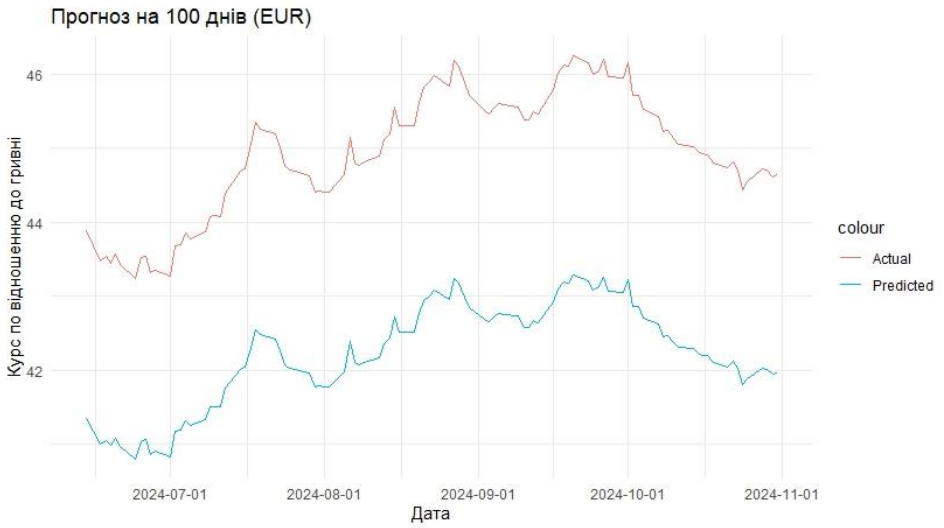

Рисунок В.6 – Графік прогнозу останніх 100 днів вибірки для тестової моделі для курсу євро

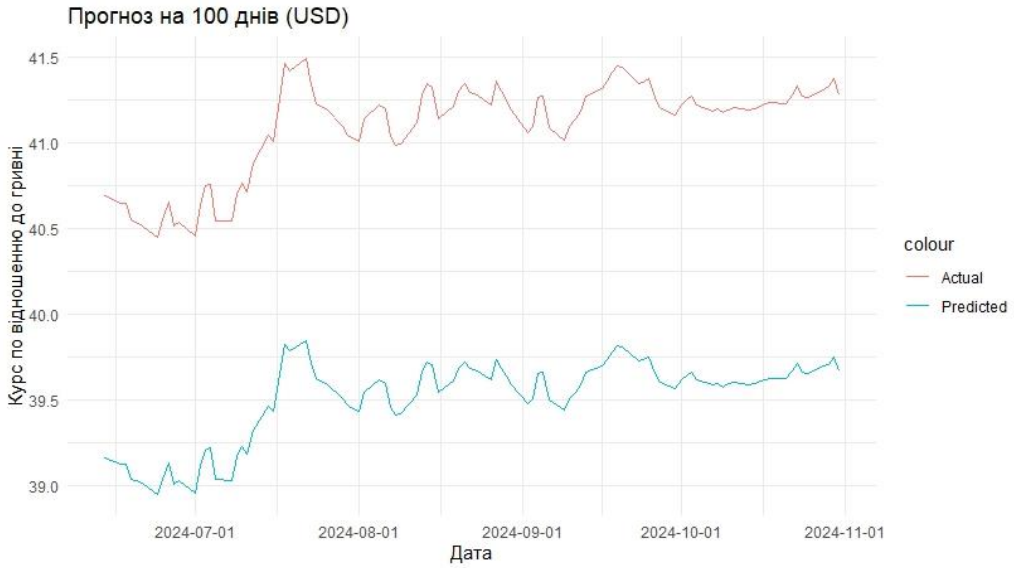

Рисунок В.6 – Графік прогнозу останніх 100 днів вибірки для тестової моделі для курсу долара США

4. Тестування стабільності моделей

Мета тесту: Оцінити стабільність моделей при зміні гіперпараметрів.

Вхідні дані: Часові ряди Brent, USD, EUR.

Кроки тестування:

1. Запустити модель із з такими наборами конфігурацій для змінної USD:

– units = 50, 150

– epochs = 1:5,

– batch_size = 16, 32,

– dropout = 0.2, 0.3

2. Розрахувати метрики MAE, MSE, RMSE для кожної конфігурації.

3. Вибрати оптимальну модель і вивести її метрики MAE, MSE, RMSE.

4. Побудувати графік прогнозу на 30 днів уперед для найкращої моделі.

Очікуваний результат:

– Всі конфігурації успішно тестуються.

– Точність моделей залежить від налаштувань гіперпараметрів, але оптимальна модель вибрана коректно.

– Графік побудований успішно для найкращої моделі з тестових конфігурацій.

Таблиця В.1

Результати метрики всіх конфігурації

units	epochs	batch_size	dropout	optimizer	learning_rate	mse	mae	rmse
150	5	16	0.3	rmsprop	0.001	0,258	0,308	0,508
150	3	16	0.2	rmsprop	0.001	0,487	0,514	0,698
150	2	32	0.3	rmsprop	0.001	0,644	0,618	0,802
150	5	32	0.3	rmsprop	0.001	0,737	0,702	0,859
150	2	32	0.2	rmsprop	0.001	0,941	0,743	0,970
150	3	16	0.3	rmsprop	0.001	1,027	0,720	1,013
150	5	32	0.2	rmsprop	0.001	1,345	0,750	1,160
150	5	16	0.2	rmsprop	0.001	1,383	1,132	1,176
150	1	32	0.2	rmsprop	0.001	1,742	0,940	1,320
150	4	16	0.2	rmsprop	0.001	1,946	1,097	1,395
150	1	16	0.2	rmsprop	0.001	1,948	0,953	1,396
50	3	32	0.2	rmsprop	0.001	10,281	2,059	3,206
50	4	32	0.2	rmsprop	0.001	12,098	2,749	3,478
50	5	32	0.3	rmsprop	0.001	12,238	2,791	3,498
50	1	16	0.2	rmsprop	0.001	12,626	2,441	3,553
50	1	32	0.2	rmsprop	0.001	12,830	2,349	3,582
50	1	16	0.3	rmsprop	0.001	15,442	2,782	3,930
50	1	32	0.3	rmsprop	0.001	17,566	3,310	4,191
50	2	16	0.2	rmsprop	0.001	2,018	1,156	1,421
150	2	16	0.2	rmsprop	0.001	2,121	0,959	1,456
50	5	16	0.2	rmsprop	0.001	2,379	1,012	1,542
150	2	16	0.3	rmsprop	0.001	3,213	1,444	1,792

50	4	16	0.3	rmsprop	0.001	3,545	1,279	1,883
150	3	32	0.3	rmsprop	0.001	3,548	1,524	1,884
50	3	16	0.3	rmsprop	0.001	4,064	1,359	2,016
50	2	16	0.3	rmsprop	0.001	4,759	1,446	2,181
150	4	32	0.2	rmsprop	0.001	4,862	1,876	2,205
150	4	16	0.3	rmsprop	0.001	5,584	2,016	2,363
150	4	32	0.3	rmsprop	0.001	5,731	2,078	2,394
50	4	16	0.2	rmsprop	0.001	6,008	1,910	2,451
150	1	32	0.3	rmsprop	0.001	6,113	1,914	2,472
50	3	32	0.3	rmsprop	0.001	6,204	1,606	2,491
50	3	16	0.2	rmsprop	0.001	6,246	1,739	2,499
150	1	16	0.3	rmsprop	0.001	6,410	2,064	2,532
50	2	32	0.2	rmsprop	0.001	6,671	1,909	2,583
150	3	32	0.2	rmsprop	0.001	6,678	2,209	2,584
50	5	16	0.3	rmsprop	0.001	6,820	2,058	2,611
50	5	32	0.2	rmsprop	0.001	7,652	1,863	2,766
50	2	32	0.3	rmsprop	0.001	8,020	1,877	2,832
50	4	32	0.3	rmsprop	0.001	9,051	2,382	3,009

```
Найкращі параметри:
> print(best_params)
$units
[1] 150

$epochs
[1] 5

$batch_size
[1] 16

$dropout
[1] 0.3

$optimizer
[1] "rmsprop"

$learning_rate
[1] 0.001

> cat("MSE:", best_mse, "\n")
MSE: 0.2583192
> cat("MAE:", best_mae, "\n")
MAE: 0.3082385
> cat("RMSE:", best_rmse, "\n")
RMSE: 0.5082511
```

Рисунок В.7 – Вибір найкращої конфігурації і її метрики

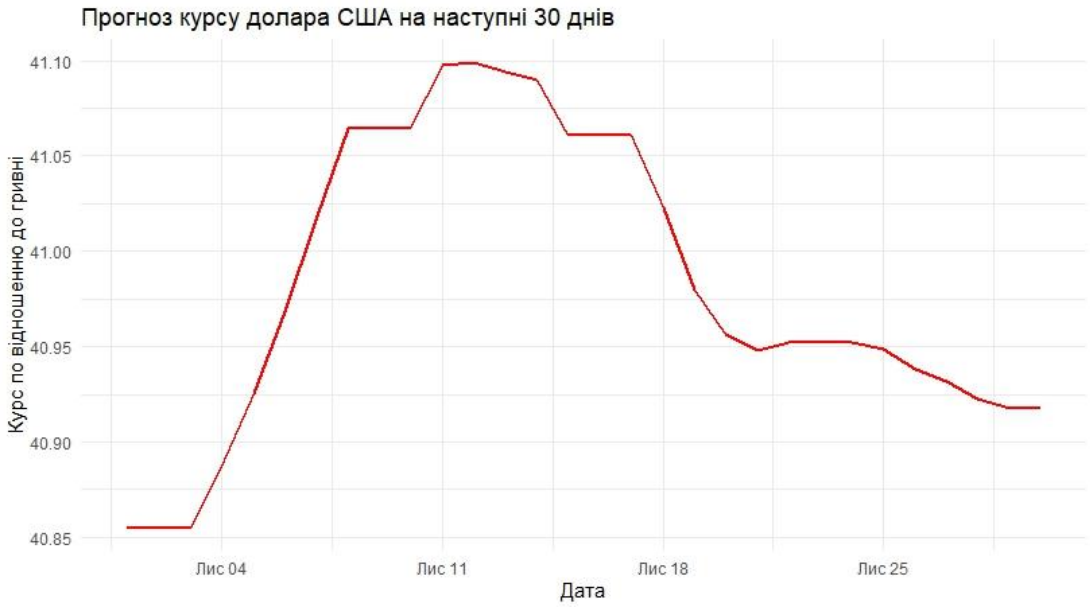

Рисунок В.8 – Графік прогнозу на 30 наступних днів для найкращої моделі серед тестових конфігурацій.

Висновки: всі випробування пройдені успішно, оскільки кожен з тестів показав очікуванні результати.

Виконавець:

студент групи КУ-61, Бондаренко К. В. _____

Додаток Г

Скрипт на R для побудови та оцінки моделі LSTM для датасету Brent Oil Prices

```
# Завантаження бібліотек
library(keras)
library(tensorflow)
library(tidyverse)
library(timetk)

# Завантаження датасету з локального файлу
data <- read.csv("C:/Users/admin/Desktop/Education/2 семестр/Інтелектуальні комп’ютерно-інтегровані технології управління виробничими процесами/Курсова робота/Датасет/archive/BrentOilPrices.csv", stringsAsFactors = FALSE)

# Обробка дат у форматі "%d-%b-%y"
data1 <- data[grepl("-", data$Date), ]
data1$Date <- as.Date(parse_date_time(data1$Date, orders = "dmy"))

# Обробка дат у форматі "%b %d, %Y"
data2 <- data[grepl(",", data$Date), ]
data2$Date <- as.Date(parse_date_time(data2$Date, orders = "b d, Y"))

# Об'єднання оброблених даних
data_combined <- rbind(data1, data2)

# Сортування даних за датою
data_combined <- data_combined %>%
  arrange(Date)

# Перевірка на пропуски
data_combined <- na.omit(data_combined)

# Перетворення даних у формат, який підходить для LSTM
data_combined <- data_combined %>%
  mutate(Date = as.Date(Date)) %>%
  tk_tbl()

# Перетворюємо дані в часовий ряд
start_year <- as.numeric(format(data_combined$Date[1], "%Y"))
start_day <- as.numeric(format(data_combined$Date[1], "%j"))
end_year <- as.numeric(format(data_combined$Date[nrow(data_combined)], "%Y"))
end_day <- as.numeric(format(data_combined$Date[nrow(data_combined)], "%j"))
oil_prices <- ts(data_combined$Price, frequency = 365, start = c(start_year, start_day), end = c(end_year, end_day))

# Візуалізуємо часовий ряд
# autoplot(oil_prices) + ggtitle("Brent Oil Prices Time Series") +
xlab("Time") + ylab("Price")

# Перетворення у часовий ряд
oil_ts <- data_combined %>%
  tk_ts(start = c(year(min(data_combined$Date)), month(min(data_combined$Date))), frequency = 365)

# Підготовка даних для LSTM
data_scaled <- scale(oil_ts)
train_data <- data_scaled[1:(length(data_scaled) - 10)]
test_data <- data_scaled[(length(data_scaled) - 9):length(data_scaled)]

x_train <- array(data = train_data, dim = c(length(train_data), 1, 1))
y_train <- train_data
```

```
# Функція для побудови і навчання моделі LSTM
build_and_train_model <- function(units, epochs, batch_size, optimizer,
x_train, y_train) {
  model <- keras_model_sequential() %>%
    layer_lstm(units = units, input_shape = c(1, 1)) %>%
    layer_dense(units = 1)

  model %>% compile(
    loss = 'mean_squared_error',
    optimizer = optimizer
  )

  history <- model %>% fit(
    x_train, y_train,
    epochs = epochs,
    batch_size = batch_size,
    verbose = 0
  )

  list(model = model, history = history)
}

# Можливі параметри для моделей
units_list <- c(10, 50, 100)
epochs_list <- 1:2
batch_size_list <- c(1, 10)
optimizers_list <- c('adam', 'sgd', 'rmsprop', 'adagrad', 'adadelta',
'adamax', 'nadam', 'ftrl')

results <- list()

# Побудова і навчання моделей з різними параметрами
for (units in units_list) {
  for (batch_size in batch_size_list) {
    for (epochs in epochs_list) {
      for (optimizer in optimizers_list) {
        result <- build_and_train_model(units, epochs, batch_size, optimizer,
x_train, y_train)
        model <- result$model
        history <- result$history

        # Прогнозування
        x_test <- array(data = test_data, dim = c(length(test_data), 1, 1))
        predicted_stock_price <- model %>% predict(x_test)
        predicted_stock_price <- predicted_stock_price * sd(oil_ts) +
mean(oil_ts)

        mse <- mean((predicted_stock_price - tail(data_combined$Price,
10))^2)
        mae <- mean(abs(predicted_stock_price - tail(data_combined$Price,
10)))
        rmse <- sqrt(mse)

        results[[paste("Units:", units, "Batch size:", batch_size, "Epochs:",
epochs, "Optimizer:", optimizer)]] <- list(
          model = model,
          history = history,
          mse = mse,
          mae = mae,
          rmse = rmse,
          predicted_stock_price = predicted_stock_price
        )
      }
```

```
    }
  }
}

# Таблиця з результатами
results_df <- data.frame(
  Model = names(results),
  MSE = sapply(results, function(x) x$mse),
  MAE = sapply(results, function(x) x$mae),
  RMSE = sapply(results, function(x) x$rmse)
)

print(results_df)

# Функція для побудови графіків прогнозів
plot_forecasts <- function(model_name, predicted_stock_price) {
  predicted_data <- data.frame(Date = tail(data_combined$Date, 10), Price =
predicted_stock_price, Type = "Forecast")
  actual_data <- data.frame(Date = tail(data_combined$Date, 10), Price =
tail(data_combined$Price, 10), Type = "Actual")
  plot_data <- rbind(actual_data, predicted_data)

  ggplot(plot_data, aes(x = Date, y = Price, color = Type)) +
    geom_line() +
    geom_point() +
    xlab("Date") + ylab("Price") +
    ggtitle(paste("LSTM Forecast for Brent Oil Prices (Last 10 Days) -",
model_name)) +
    theme_minimal()
}

# Встановлення української локалі
Sys.setlocale("LC_TIME", "uk_UA.UTF-8")

# Побудова графіків для всіх моделей
for (model_name in names(results)) {
  predicted_stock_price <- results[[model_name]]$predicted_stock_price
  print(plot_forecasts(model_name, predicted_stock_price))
}

# Вибір найкращої моделі за MSE
best_model_name <- names(results)[which.min(results_df$MSE)]
best_model <- results[[best_model_name]]$model
best_history <- results[[best_model_name]]$history

# Прогноз на 100 днів назад
forecast_horizon <- 100
train_data_100 <- data_scaled[1:(length(data_scaled) - forecast_horizon)]
test_data_100 <- data_scaled[(length(data_scaled) - forecast_horizon +
1):length(data_scaled)]

x_test_100 <- array(data = test_data_100, dim = c(length(test_data_100), 1,
1))

future_predictions_100 <- best_model %>% predict(x_test_100)
future_predictions_100 <- future_predictions_100 * sd(oil_ts) + mean(oil_ts)

future_dates_100 <- tail(data_combined$Date, forecast_horizon)
future_plot_data_100 <- data.frame(Date = future_dates_100, Price =
future_predictions_100, Type = "Forecast")

actual_data_100 <- data.frame(Date = future_dates_100, Price =
tail(data_combined$Price, forecast_horizon), Type = "Actual")
```

```
plot_data_100 <- rbind(actual_data_100, future_plot_data_100)

ggplot(plot_data_100, aes(x = Date, y = Price, color = Type)) +
  geom_line() +
  geom_point() +
  xlab("Date") + ylab("Price") +
  ggtitle(paste("LSTM Forecast for Brent Oil Prices (Last 100 Days) -",
best_model_name)) +
  theme_minimal()

# Виведення значень MSE, MAE, RMSE для найкращої моделі
cat("Best Model:", best_model_name, "\n")
cat("MSE:", results[[best_model_name]]$mse, "\n")
cat("MAE:", results[[best_model_name]]$mae, "\n")
cat("RMSE:", results[[best_model_name]]$rmse, "\n")

# Прогноз на 100 днів вперед
future_horizon <- 100
last_known_data <- data_scaled[(length(data_scaled) - forecast_horizon +
1):length(data_scaled)]
future_predictions_forward <- numeric(future_horizon)

for (i in 1:future_horizon) {
  input_data <- array(last_known_data[(i):(i + forecast_horizon - 1)], dim =
c(forecast_horizon, 1, 1))
  future_predictions_forward[i] <- best_model %>% predict(input_data)
  last_known_data <- c(last_known_data, future_predictions_forward[i])
}

future_predictions_forward <- future_predictions_forward * sd(oil_ts) +
mean(oil_ts)

future_dates_forward <- seq(max(data_combined$Date) + 1, by = "day",
length.out = future_horizon)
future_plot_data_forward <- data.frame(Date = future_dates_forward, Price =
future_predictions_forward, Type = "Forecast")

ggplot(future_plot_data_forward, aes(x = Date, y = Price, color = Type)) +
  geom_line() +
  geom_point() +
  xlab("Date") + ylab("Price") +
  ggtitle(paste("LSTM Forecast for Brent Oil Prices (Next 100 Days) -",
best_model_name)) +
  theme_minimal()

# Побудова графіку втрат під час навчання
loss_data <- data.frame(
  Epoch = 1:length(best_history$metrics$loss),
  Loss = best_history$metrics$loss
)

ggplot(loss_data, aes(x = Epoch, y = Loss)) +
  geom_line() +
  geom_point() +
  xlab("Epoch") + ylab("Loss") +
  ggtitle(paste("Training Loss Curve -", best_model_name)) +
  theme_minimal()

# Гістограма помилок прогнозу
prediction_errors <- tail(data_combined$Price, forecast_horizon) -
future_predictions_100
```

```
ggplot(data.frame(Errors = prediction_errors), aes(x = Errors)) +
  geom_histogram(binwidth = 1, fill = "blue", alpha = 0.7) +
  xlab("Prediction Error") + ylab("Frequency") +
  ggtitle(paste("Prediction Error Histogram -", best_model_name)) +
  theme_minimal()

# Графік автокореляцій залишків
acf_resid <- acf(prediction_errors, plot = FALSE)

acf_data <- data.frame(
  Lag = acf_resid$lag,
  ACF = acf_resid$acf
)

ggplot(acf_data, aes(x = Lag, y = ACF)) +
  geom_bar(stat = "identity", fill = "blue", alpha = 0.7) +
  geom_hline(yintercept = c(-1.96/sqrt(length(prediction_errors)),
1.96/sqrt(length(prediction_errors))), linetype = "dashed", color = "red") +
  xlab("Lag") + ylab("ACF") +
  ggtitle(paste("ACF of Prediction Errors -", best_model_name)) +
  theme_minimal()

# Прогноз на 100 днів вперед + 100 днів до цього
future_horizon <- 100
last_known_data <- data_scaled[(length(data_scaled) - forecast_horizon +
1):length(data_scaled)]
future_predictions_forward <- numeric(future_horizon)

for (i in 1:future_horizon) {
  input_data <- array(last_known_data[(i):(i + forecast_horizon - 1)], dim =
c(forecast_horizon, 1, 1))
  future_predictions_forward[i] <- best_model %>% predict(input_data)
  last_known_data <- c(last_known_data, future_predictions_forward[i])
}

future_predictions_forward <- future_predictions_forward * sd(oil_ts) +
mean(oil_ts)

future_dates_forward <- seq(from = max(data_combined$Date) + 1, by = 1,
length.out = future_horizon)
future_plot_data_forward <- data.frame(Date = future_dates_forward, Price =
future_predictions_forward, Type = "Forecast")

plot_data_forward <- rbind(data_combined %>% tail(forecast_horizon) %>%
select(Date, Price) %>% mutate(Type = "Actual"), future_plot_data_forward)

# Графік прогнозу на 100 днів вперед + 100 днів до цього
ggplot(plot_data_forward, aes(x = Date, y = Price, color = Type)) +
  geom_line() +
  geom_point() +
  xlab("Date") + ylab("Price") +
  ggtitle(paste("LSTM Forecast for Brent Oil Prices (Next 100 Days) -",
best_model_name)) +
  theme_minimal()
```

Додаток Д

Скрипт на R для побудови та оцінки моделі LSTM для прогнозування курсів долара США і євро по відношенню до гривні

```
# Завантаження необхідних бібліотек
library(data.table)
library(dplyr)
library(keras)
library(tensorflow)
library(ggplot2)
library(lubridate)

# Початок вимірювання часу
start_time <- Sys.time()

# Змінна валюти
currency <- "EUR"  # Можливі значення: "USD", "EUR"

# Фіксація випадкових значень для стабільності результатів
set.seed(42)
tensorflow::set_random_seed(42)
Sys.setenv("TF_CUDNN_DETERMINISTIC" = "1")
Sys.setenv("TF_DETERMINISTIC_OPS" = "1")

# Крок 1: Завантаження та обробка даних
file_path <- "C:/Users/admin/Desktop/Education/3 семестр/Дипломна робота/3
розділ_Модель/USD_EUR.csv"
data <- fread(file_path, sep = ";", header = TRUE, fill = TRUE,
stringsAsFactors = FALSE, encoding = "UTF-8")
data[[1]] <- as.Date(data[[1]], format = "%d.%m.%Y")
data[[2]] <- as.numeric(data[[2]])
data[[3]] <- as.numeric(data[[3]])

# Вибір стовбця + видалення сталого значення
if (currency == "USD") {
  data <- data %>% arrange(Date) %>% na.omit() %>% select(Date, USD)
    # Видалення періодів із фіксованим курсом
  data <- data %>% filter(USD != lag(USD, default = first(USD)))
} else if (currency == "EUR") {
  data <- data %>% arrange(Date) %>% na.omit() %>% select(Date, EUR)
  # Видалення періодів із фіксованим курсом
  data <- data %>% filter(EUR != lag(EUR, default = first(EUR)))
}

# Масштабування даних
scaled_data <- scale(data[[2]])
lag <- 10
train_size <- round(0.8 * length(scaled_data))
train_data <- scaled_data[1:train_size]
test_data <- scaled_data[(train_size + 1):length(scaled_data)]

# Підготовка даних
x_train <- array(NA, dim = c(length(train_data) - lag, lag, 1))
y_train <- array(NA, dim = c(length(train_data) - lag))
for (i in 1:(length(train_data) - lag)) {
  x_train[i,,1] <- train_data[i:(i + lag - 1)]
  y_train[i] <- train_data[i + lag]
}

x_test <- array(NA, dim = c(length(test_data) - lag, lag, 1))
y_test <- array(NA, dim = c(length(test_data) - lag))
for (i in 1:(length(test_data) - lag)) {
```

```
  x_test[i,,1] <- test_data[i:(i + lag - 1)]
  y_test[i] <- test_data[i + lag]
}

# Функція для побудови моделі
build_model <- function(units, dropout, optimizer, learning_rate) {
  if (optimizer == 'adam') {
    opt <- optimizer_adam(learning_rate = learning_rate)
  } else if (optimizer == 'rmsprop') {
    opt <- optimizer_rmsprop(learning_rate = learning_rate)
  } else {
    opt <- optimizer
  }

  model <- keras_model_sequential() %>%
    layer_lstm(units = units, input_shape = c(lag, 1)) %>%
    layer_dropout(rate = dropout) %>%
    layer_dense(units = 1)

  model %>% compile(
    loss = 'mean_squared_error',
    optimizer = opt
  )

  return(model)
}

# Функція для навчання моделі та оцінки
train_and_evaluate <- function(units, epochs, batch_size, dropout, optimizer,
learning_rate) {
  model <- build_model(units, dropout, optimizer, learning_rate)
  history <- model %>% fit(
    x_train, y_train,
    epochs = epochs,
    batch_size = batch_size,
    validation_split = 0.2,
    verbose = 0
  )

  predictions <- model %>% predict(x_test)
  predictions <- predictions * attr(scaled_data, 'scaled:scale') +
attr(scaled_data, 'scaled:center')

  actual_test <- data[[2]][(length(data[[2]]) - length(predictions) +
1):length(data[[2]])]
  mse <- mean((actual_test - predictions)^2)
  mae <- mean(abs(actual_test - predictions))
  rmse <- sqrt(mse)

  return(list(model = model, mse = mse, mae = mae, rmse = rmse, history =
history))
}

# Функція для збереження моделі та додаткових даних
save_model_with_metadata <- function(model, history, params, filename) {
  # Збереження моделі
  save_model_hdf5(model, paste0(filename, ".h5"))

  # Збереження додаткових даних (параметри і історія навчання)
  metadata <- list(
    best_params = params,
    best_history = history
  )
```

```
  saveRDS(metadata, paste0(filename, ".rds"))
}

# Функція для завантаження моделі та додаткових даних
load_model_with_metadata <- function(filename) {
  # Завантаження моделі
  model <- load_model_hdf5(paste0(filename, ".h5"))

  # Завантаження додаткових даних
  metadata <- readRDS(paste0(filename, ".rds"))

  return(list(model = model, metadata = metadata))
}

# Назва файлу моделі
if (currency == "USD") {
  model_filename <- "best_model_USD"
} else if (currency == "EUR") {
  model_filename <- "best_model_EUR"
}

# Налаштування параметрів для перебору (432 000 різних моделей)
units_list <- c(100)           # c(16, 32, 50, 100, 150, 200)
epochs_list <- c(1:50)         # c(1:75)
batch_size_list <- c(8)        # c(8, 16, 32, 64, 128, 256)
dropout_list <- c(0.1)         # c(0.1, 0.2, 0.3, 0.4)
optimizer_list <- c('rmsprop') # c('adam', 'sgd', 'rmsprop', 'adagrad',
'adadelta', 'adamax', 'nadam', 'ftrl')
learning_rate_list <- c(0.001) # c(0.001, 0.01, 0.1, 0.0005, 0.002)

results_df <- data.frame()
best_mse <- Inf
best_mae <- Inf
best_rmse <- Inf
best_history <- NULL
best_model <- NULL
best_params <- NULL

# Розрахунок ітерацій
total_combinations <- length(units_list) * length(epochs_list) *
length(batch_size_list) * length(dropout_list) * length(optimizer_list) *
length(learning_rate_list)
current_combination <- 0
last_iteration_time <- start_time

# Гіперпараметричний пошук
for (units in units_list) {
  for (epochs in epochs_list) {
    for (batch_size in batch_size_list) {
      for (dropout in dropout_list) {
        for (optimizer in optimizer_list) {
          for (learning_rate in learning_rate_list) {
            current_combination <- current_combination + 1

            # Поточна інформація
            cat(sprintf("\nЗараз %d итерація з %d, які будуть виконані.\n",
current_combination, total_combinations))
            cat(sprintf("Поточна конфігурація: units=%d, epochs=%d,
batch_size=%d, dropout=%.1f, optimizer=%s, learning_rate=%.4f\n",
                        units, epochs, batch_size, dropout, optimizer,
learning_rate))
```

```
            result <- train_and_evaluate(units, epochs, batch_size, dropout,
optimizer, learning_rate)
            mse <- result$mse
            mae <- result$mae
            rmse <- result$rmse

            if (mse < best_mse) {
              best_mse <- mse
              best_mae <- mae
              best_rmse <- rmse
              best_params <- list(units = units, epochs = epochs, batch_size
= batch_size,
                                  dropout = dropout, optimizer = optimizer,
learning_rate = learning_rate)
              best_model <- result$model
              best_history <- result$history
              # Збереження найкращої моделі разом з історією та параметрами
              save_model_with_metadata(best_model, best_history, best_params,
model_filename)
            }

             # Час завершення ітерації
            current_time <- Sys.time()
            iteration_time <- as.numeric(difftime(current_time,
last_iteration_time, units = "secs"))
            last_iteration_time <- current_time

            # Перетворення часу в звичний формат
            minutes <- floor(iteration_time / 60)
            seconds <- round(iteration_time %% 60)

            cat("Скрипт запущено:", format(start_time, "%d-%m-%Y %H:%M:%S"),
"\n")
            cat("Час завершення поточної ітерації:", format(current_time,
"%d-%m-%Y %H:%M:%S"), "\n")
            cat(sprintf("Час виконання минулої ітерації: %d хвилин %d
секунд\n", minutes, seconds))

            results_df <- rbind(results_df, data.frame(
              units, epochs, batch_size, dropout, optimizer, learning_rate,
mse, mae, rmse
            ))
          }
        }
      }
    }
  }
}

# Функція для створення підпису на основі найкращої конфігурації
get_model_config <- function(params) {
  return(paste0(
    "units: ", params$units,
    ", epochs: ", params$epochs,
    ", batch_size: ", params$batch_size,
    ", dropout: ", params$dropout,
    ", optimizer: ", params$optimizer,
    ", learning_rate: ", params$learning_rate,

  ))
}

get_model_config <- function(params) {
```

```
  if (is.null(params$units)) params$units <- "N/A"
  if (is.null(params$epochs)) params$epochs <- "N/A"
  if (is.null(params$batch_size)) params$batch_size <- "N/A"
  if (is.null(params$dropout)) params$dropout <- "N/A"
  if (is.null(params$optimizer)) params$optimizer <- "N/A"
  if (is.null(params$learning_rate)) params$learning_rate <- "N/A"

  return(paste0(
    "units: ", params$units,
    ", epochs: ", params$epochs,
    ", batch_size: ", params$batch_size,
    ", dropout: ", params$dropout,
    ", optimizer: ", params$optimizer,
    ", learning_rate: ", params$learning_rate
  ))
}

# Завантаження моделі та метаданих
loaded_data <- load_model_with_metadata(model_filename)
best_model <- loaded_data$model
metadata <- loaded_data$metadata

# Отримання параметрів і історії
best_params <- metadata$best_params
best_history <- metadata$best_history

# Створення конфігурації для графіків
best_model_config <- get_model_config(best_params)


# Збереження результатів у CSV
write.csv(results_df, "results.csv", row.names = FALSE)
cat("Результати збережено у файл 'results.csv'\n")

# Встановлення української локалі для відображення дати
Sys.setlocale("LC_TIME", "uk_UA.UTF-8")

# Отримання конфігурації найкращої моделі
if (!is.null(best_params) && !is.null(best_params$units) &&
    !is.null(best_params$epochs) && !is.null(best_params$batch_size) &&
    !is.null(best_params$dropout) && !is.null(best_params$optimizer) &&
    !is.null(best_params$learning_rate)) {

  best_model_config <- get_model_config(best_params)
} else {
  best_model_config <- "Найкраща модель не знайдена"
}

# Прогнозування для 10 та 100 днів
x_test_10 <- array(data = test_data[(length(test_data) -
9):length(test_data)], dim = c(10, lag, 1))
predictions_10 <- best_model %>% predict(x_test_10)
predictions_10 <- predictions_10 * attr(scaled_data, 'scaled:scale') +
attr(scaled_data, 'scaled:center')

x_test_100 <- array(data = test_data[(length(test_data) -
99):length(test_data)], dim = c(100, lag, 1))
predictions_100 <- best_model %>% predict(x_test_100)
predictions_100 <- predictions_100 * attr(scaled_data, 'scaled:scale') +
attr(scaled_data, 'scaled:center')

# Візуалізація для 10 днів
actual_10 <- data[[2]][(length(data[[2]]) - 9):length(data[[2]])]
```

```
results_10 <- data.frame(Date = data$Date[(length(data$Date) -
9):length(data$Date)], Actual = actual_10, Predicted = predictions_10)

ggplot(results_10, aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Actual")) +
  geom_line(aes(y = Predicted, color = "Predicted")) +
  labs(
    title = paste("Прогноз на останніх 10 днів (", currency, ")", sep = ""),
    x = "Дата",
    y = "Курс по відношенню до гривні"
  ) +
  scale_x_date(date_labels = "%Y-%m-%d") +
  theme_minimal()

# Візуалізація для 100 днів
actual_100 <- data[[2]][(length(data[[2]]) - 99):length(data[[2]])]
results_100 <- data.frame(Date = data$Date[(length(data$Date) -
99):length(data$Date)], Actual = actual_100, Predicted = predictions_100)

ggplot(results_100, aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Actual")) +
  geom_line(aes(y = Predicted, color = "Predicted")) +
  labs(
    title = paste("Прогноз на 100 днів (", currency, ")", sep = ""),
    x = "Дата",
    y = "Курс по відношенню до гривні"
  ) +
  scale_x_date(date_labels = "%Y-%m-%d") +
  theme_minimal()

# Виведення метрик найкращої моделі
cat("Найкращі параметри:\n")
print(best_params)
cat("MSE:", best_mse, "\n")
cat("MAE:", best_mae, "\n")
cat("RMSE:", best_rmse, "\n")

# Додавання графіку похибки по епохах для найкращої моделі
if (!is.null(best_history)) {
  loss_df <- data.frame(
    epoch = 1:length(best_history$metrics$loss),
    loss = best_history$metrics$loss,
    val_loss = best_history$metrics$val_loss
  )

  ggplot(loss_df, aes(x = epoch)) +
    geom_line(aes(y = loss, color = "Train Loss")) +
    geom_line(aes(y = val_loss, color = "Validation Loss")) +
    labs(title = "Графік втрат по епохах", x = "Епохи", y = "Втрата") +
    scale_x_continuous(breaks = seq(1, max(epochs_list), by = 5)) +
    theme_minimal()
}

# Виведення часу виконання
end_time <- Sys.time()
execution_time <- end_time - start_time

# Форматування часу виконання
hours <- floor(as.numeric(execution_time, "secs") / 3600)
minutes <- floor((as.numeric(execution_time, "secs") %% 3600) / 60)
seconds <- round(as.numeric(execution_time, "secs") %% 60)
cat(sprintf("Час виконання: %d годин %d хвилин %d секунд\n", hours, minutes,
seconds))
```

Додаток Е

Скрипт на R для візуалізації графіків моделі LSTM для прогнозування курсів долара США і євро по відношенню до гривні

```r
# Завантаження необхідних бібліотек
library(data.table)
library(dplyr)
library(keras)
library(tensorflow)
library(ggplot2)
library(lubridate)

# Початок вимірювання часу
start_time <- Sys.time()

# Фіксація випадкових значень для стабільності результатів
set.seed(42)
tensorflow::set_random_seed(42)

# Змінна валюти
currency <- "EUR"  # Можливі значення: "USD", "EUR"

# Завантаження та обробка даних
file_path <- "C:/Users/admin/Desktop/Education/3 семестр/Дипломна робота/3 розділ_Модель/USD_EUR.csv"
data <- fread(file_path, sep = ";", header = TRUE, stringsAsFactors = FALSE, encoding = "UTF-8")
data[[1]] <- as.Date(data[[1]], format = "%d.%m.%Y")
data[[2]] <- as.numeric(data[[2]])
data[[3]] <- as.numeric(data[[3]])

# Вибір стовбця + видалення сталого значення
if (currency == "USD") {
  data <- data %>% arrange(Date) %>% na.omit() %>% select(Date, USD)
  data <- data %>% filter(USD != lag(USD, default = first(USD)))
} else if (currency == "EUR") {
  data <- data %>% arrange(Date) %>% na.omit() %>% select(Date, EUR)
}

# Масштабування даних
scaled_data <- scale(data[[2]])
lag <- 10
train_size <- round(0.8 * length(scaled_data))
test_data <- scaled_data[(train_size + 1):length(scaled_data)]

# Вибір стовбця + видалення сталого значення
if (currency == "USD") {
  # Завантаження найкращої моделі
  best_model <- load_model_hdf5("best_model_USD.h5")
  # Завантаження метрик найкращої моделі
  best_params <- readRDS("best_model_USD.rds")$best_params
  best_history <- readRDS("best_model_USD.rds")$best_history
} else if (currency == "EUR") {
  # Завантаження найкращої моделі
  best_model <- load_model_hdf5("best_model_EUR.h5")
  # Завантаження метрик найкращої моделі
  best_params <- readRDS("best_model_EUR.rds")$best_params
  best_history <- readRDS("best_model_EUR.rds")$best_history
}

# Прогнозування для 10 та 100 днів
```

```
x_test_10 <- array(data = test_data[(length(test_data) -
9):length(test_data)], dim = c(10, lag, 1))
predictions_10 <- best_model %>% predict(x_test_10)
predictions_10 <- predictions_10 * attr(scaled_data, 'scaled:scale') +
attr(scaled_data, 'scaled:center')

x_test_100 <- array(data = test_data[(length(test_data) -
99):length(test_data)], dim = c(100, lag, 1))
predictions_100 <- best_model %>% predict(x_test_100)
predictions_100 <- predictions_100 * attr(scaled_data, 'scaled:scale') +
attr(scaled_data, 'scaled:center')

# Візуалізація прогнозу на останніх 10 днів
actual_10 <- data[[2]][(length(data[[2]]) - 9):length(data[[2]])]
results_10 <- data.frame(Date = data$Date[(length(data$Date) -
9):length(data$Date)], Actual = actual_10, Predicted = predictions_10)

# Встановлення української локалі
Sys.setlocale("LC_TIME", "uk_UA.UTF-8")

# Формування заголовка
currency_text <- ifelse(currency == "EUR", "(Євро)", "(Долар США)")

ggplot(results_10, aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Факт")) +
  geom_line(aes(y = Predicted, color = "Прогноз")) +
  labs(
    title = paste("Прогноз на останніх 10 днів", currency_text),
    x = "Дата",
    y = "Курс по відношенню до гривні"
  ) +
  theme_minimal()

# Візуалізація прогнозу на останніх 100 днів
actual_100 <- data[[2]][(length(data[[2]]) - 99):length(data[[2]])]
results_100 <- data.frame(Date = data$Date[(length(data$Date) -
99):length(data$Date)], Actual = actual_100, Predicted = predictions_100)

ggplot(results_100, aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Факт")) +
  geom_line(aes(y = Predicted, color = "Прогноз")) +
  labs(
    title = paste("Прогноз на останніх 100 днів", currency_text),
    x = "Дата",
    y = "Курс по відношенню до гривні"
  ) +
  theme_minimal()

# Виведення метрик найкращої моделі
cat("Найкращі параметри:\n")
print(best_params)

# Графік похибки по епохах для найкращої моделі
if (!is.null(best_history)) {
  loss_df <- data.frame(
    epoch = 1:length(best_history$metrics$loss),
    loss = best_history$metrics$loss,
    val_loss = best_history$metrics$val_loss
  )

  ggplot(loss_df, aes(x = epoch)) +
    geom_line(aes(y = loss, color = "Train Loss")) +
    geom_line(aes(y = val_loss, color = "Validation Loss")) +
```

```
    labs(title = paste("Графік втрат по епохах", currency_text), x = "Епохи",
y = "Втрата") +
    scale_x_continuous(breaks = seq(1, 60, by = 5)) +
    theme_minimal()
}

past_days <- ifelse(currency == "EUR", 30, 45)

# Прогнозування на 30 днів з урахуванням вихідних
last_100 <- tail(test_data, past_days)
window <- array(data = last_100, dim = c(1, lag, 1))
forecast_30 <- numeric(30)
forecast_dates <- seq.Date(from = max(data$Date) + 1, by = "day", length.out
= 30)
last_workday_value <- NULL

for (i in 1:30) {
  current_day <- weekdays(forecast_dates[i], abbreviate = FALSE)
  if (current_day %in% c("Saturday", "Sunday", "субота", "неділя")) {
    if (!is.null(last_workday_value)) {
      forecast_30[i] <- last_workday_value
    }
  } else {
    pred <- best_model %>% predict(window)
    forecast_30[i] <- pred * attr(scaled_data, 'scaled:scale') +
attr(scaled_data, 'scaled:center')
    if (current_day %in% c("Friday", "п'ятниця")) {
      last_workday_value <- forecast_30[i]
    }
    last_100 <- c(last_100[-1], pred)
    window <- array(data = last_100, dim = c(1, lag, 1))
  }
}

# Формування заголовка
forecast_text <- ifelse(currency == "EUR", "Євро", "долара США")

# Графік прогнозу на 30 днів вперед
forecast_results <- data.frame(Date = forecast_dates, Predicted =
forecast_30)

ggplot(forecast_results, aes(x = Date, y = Predicted)) +
  geom_line(color = "red", size = 1) +
  labs(
    title = paste("Прогноз курсу", forecast_text, "на наступні 30 днів"),
    x = "Дата",
    y = "Курс по відношенню до гривні"
  ) +
  theme_minimal()

# Виведення прогнозу на 30 днів
cat("Прогноз на наступні 30 днів:\n")
print(forecast_results)

# Графік розширення даних (100 днів + 30 днів)
combined_dates <- c(tail(data$Date, 100), forecast_results$Date)
combined_values <- c(tail(data[[2]], 100), forecast_results$Predicted)
labels <- c(rep("Факт", 100), rep("Прогноз", 30))

combined_data <- data.frame(Date = combined_dates, Value = combined_values,
Type = labels)

ggplot(combined_data, aes(x = Date, y = Value, color = Type)) +
```

```
  geom_line(size = 1) +
  labs(
    title = paste("Динаміка курсу", forecast_text, ": останні 100 днів та
прогноз на 30 днів"),
    x = "Дата",
    y = "Курс по відношенню до гривні"
  ) +
  scale_color_manual(values = c("Факт" = "blue", "Прогноз" = "red")) +
  theme_minimal()

# Прогнозування на 90 днів з урахуванням вихідних
last_100 <- tail(test_data, past_days)
window <- array(data = last_100, dim = c(1, lag, 1))
forecast_30 <- numeric(90)
forecast_dates <- seq.Date(from = max(data$Date) + 1, by = "day", length.out
= 90)
last_workday_value <- NULL

for (i in 1:90) {
  current_day <- weekdays(forecast_dates[i], abbreviate = FALSE)
  if (current_day %in% c("Saturday", "Sunday", "субота", "неділя")) {
    if (!is.null(last_workday_value)) {
      forecast_30[i] <- last_workday_value
    }
  } else {
    pred <- best_model %>% predict(window)
    forecast_30[i] <- pred * attr(scaled_data, 'scaled:scale') +
attr(scaled_data, 'scaled:center')
    if (current_day %in% c("Friday", "п'ятниця")) {
      last_workday_value <- forecast_30[i]
    }
    last_100 <- c(last_100[-1], pred)
    window <- array(data = last_100, dim = c(1, lag, 1))
  }
}

# Графік прогнозу на 90 днів вперед
forecast_results <- data.frame(Date = forecast_dates, Predicted =
forecast_30)

ggplot(forecast_results, aes(x = Date, y = Predicted)) +
  geom_line(color = "red", size = 1) +
  labs(
    title = paste("Прогноз курсу", forecast_text, "на наступні 90 днів"),
    x = "Дата",
    y = "Курс по відношенню до гривні"
  ) +
  theme_minimal()

# Виведення часу виконання
end_time <- Sys.time()
execution_time <- end_time - start_time
cat(sprintf("Час виконання: %f секунд\n", execution_time))
```