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

Навчально-науковий інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___»_____2025 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Методи і моделі інтелектуального аналізу даних для прогнозування ключових показників на ринку електроенергії»

Виконав:
студент IV курсу, групи КА-14
Зеленський Микита Максимович _____

Керівник:
старший викладач кафедри ММСА, PhD
Левенчук Людмила Борисівна _____

Консультант з економічного розділу:
доцент, к.е.н.
Рощина Надія Василівна _____

Консультант з нормконтролю:
ас.каф. ММСА
Канцедал Георгій Олегович _____

Рецензент:
доцент кафедри ШІ, PhD
Гуськова Віра Геннадіївна _____

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

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2025 р.

ЗАВДАННЯ
на дипломну роботу студенту
Зеленському Микиті Максимовичу

1. Тема роботи «Методи і моделі інтелектуального аналізу даних для прогнозування ключових показників на ринку електроенергії», керівник роботи старший викладач кафедри ММСА, PhD Левенчук Л.Б., затверджені наказом по університету від «26 » травня 2025 р. № 1759-с.
2. Термін подання студентом роботи 11.06.2025.
3. Вихідні дані до роботи: данні погодинного надходження в мережу внутрішньодобового сегменту Оператора Ринку розподілу електричної енергії.
4. Зміст роботи: дослідження предметної області, огляд сучасних архітектур, програмна реалізація та аналіз результатів, функціонально-вартісний аналіз програмного продукту.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): актуальність, мета, об’єкт та предмет дослідження; постановка задачі; предметна область; опис набору даних.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В., доцент, к.е.н		

7. Дата видачі завдання 14.04.2025

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми БДР	14.04.2025-20.04.2025	виконано
2	Ознайомлення зі структурою БДР згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	14.04.2025-20.04.2025	виконано
3	Ознайомлення з ДСТУ 3008-95 та стандартами ЄСПД	14.04.2025-27.04.2025	виконано
4	Проведення дослідження за темою БДР під керівництвом керівника	14.04.2025-27.04.2025	виконано
5	Завершення роботи над першим варіантом частини БДР	21.04.2025-11.05.2025	виконано
6	Проведення роботи над експериментальною частиною БДР	05.05.2025-18.05.2025	виконано
7	Проведення роботи над програмним продуктом	05.05.2025-18.05.2025	виконано
8	Проведення роботи над економічною частиною БДР	18.05.2025-31.05.2025	виконано
9	Оформлення БДР та аналіз отриманих результатів	18.05.2025-31.05.2025	виконано

Студент Микита ЗЕЛЕНСЬКИЙ

Керівник Людмила ЛЕВЕНЧУК

РЕФЕРАТ

Дипломна робота: 139 с., 6 табл., 25 рис., 2 додатки, 41 джерел.

ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ, РИНОК ЕЛЕКТРОЕНЕРГІЇ, ANFIS, SARIMAX, LSTM.

Об’єкт дослідження – процес прогнозування ключових показників діяльності на ринку електроенергії.

Предмет дослідження – методи та моделі інтелектуального аналізу даних, що застосовуються для прогнозування часових рядів в енергетичній сфері.

Мета роботи – дослідити, порівняти та реалізувати сучасні методи прогнозування часових рядів. Розробити програмний продукт для прогнозування ключових показників на ринку електроенергії.

Актуальність – вирішення задачі прогнозування в енергетичному секторі для оптимального функціонування ринку, зниження фінансових втрат та підвищення стійкості системи до коливань попиту і пропозиції.

Методи дослідження – моделі машинного навчання, класичні статистичні методи, методи обробки часових рядів.

Результати роботи – зібрано набір даних, створено програмний продукт, що забезпечує аналіз даних, побудову моделей прогнозування, валідацію результатів та візуалізацію прогнозів. Проведено порівняння моделей за технічними та економічними критеріями.

Шляхи подальшого розвитку – розширення функціональності програмного забезпечення за рахунок інтеграції додаткових джерел даних (API, SCADA), впровадження механізмів автоматичної адаптації моделей до змін у даних, створення користувацького графічного інтерфейсу для підвищення зручності використання системи в прикладних задачах енергетики.

ABSTRACT

Diploma thesis: 139 p., 6 tabl., 25 fig., 2 appendices, 41 references.

INTELLIGENT DATA ANALYSIS, TIME SERIES FORECASTING, ELECTRICITY MARKET, ANFIS, SARIMAX, LSTM.

Object of research — the process of forecasting key performance indicators in the electricity market.

Subject of the study — methods and models of intelligent data analysis used for time series forecasting in the energy sector.

Purpose of the study — to investigate, compare, and implement modern methods of time series forecasting. To develop a software product for forecasting key indicators in the electricity market.

Relevance — solving the problem of forecasting in the energy sector to ensure optimal market functioning, reduce financial losses, and increase system resilience to fluctuations in supply and demand.

Research methods — machine learning models, classical statistical methods, time series analysis techniques.

Results — a dataset was collected and a software product was developed that provides data analysis, model construction, result validation, and forecast visualization. The models were compared based on technical and economic criteria.

Ways of further development of the research subject — expanding the functionality of the software by integrating additional data sources (API, SCADA), implementing mechanisms for automatic model adaptation to changing data, and creating a user-friendly graphical interface to enhance usability for practical tasks in the energy domain.

ЗМІСТ

ВСТУП

На сучасному етапі розвитку економіки України ринок електроенергії є одним із ключових секторів, стабільність і ефективність роботи якого мають безпосередній вплив на економічне зростання держави та добробут населення. Особливістю цього ринку є його динамічність і мінливість, зумовлені численними факторами, серед яких – погодні умови, зміни в споживанні та виробництві енергії, а також коливання цін на світових ринках енергоресурсів.

В умовах зростаючої невизначеності, війни та високої волатильності ключових показників електроенергетичного ринку, таких як ціни та обсяги продажу й закупівель, виникає гостра потреба в застосуванні сучасних інтелектуальних технологій для точного прогнозування часових рядів. Прогнозування цих показників дає можливість компаніям-постачальникам електроенергії ефективніше планувати свою діяльність, знижувати економічні ризики, оптимізувати витрати та покращувати якість управлінських рішень.

Використання класичних статистичних методів прогнозування часто є недостатнім для аналізу складних закономірностей і нелінійних взаємозв’язків у великих масивах даних, характерних для енергоринку. Тому актуальним завданням є розробка та впровадження методів і моделей інтелектуального аналізу даних, що дозволяють суттєво підвищити точність прогнозних моделей.

Метою даної дипломної роботи є дослідження та практична реалізація методів і моделей інтелектуального аналізу даних для прогнозування ключових показників ціни та обсягів на ринку електроенергії. Для досягнення поставленої мети планується проаналізувати існуючі підходи до прогнозування часових рядів, визначити найбільш ефективні моделі для умов українського енергетичного ринку, розробити алгоритми прогнозування та перевірити їх на реальних даних.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Мотивація дослідження

Енергетичний ринок відіграє стратегічну роль у розвитку економіки, забезпеченні національної безпеки та формуванні умов для сталого функціонування державних і приватних інституцій. Особливої ваги це питання набуває в умовах постійної трансформації енергетичного сектору, що супроводжується змінами у структурі споживання, генерації, механізмах ціноутворення та регуляторній політиці. Динамічність зовнішніх умов, включаючи вплив геополітичної ситуації, зміну кліматичних параметрів, коливання на міжнародних ринках енергоносіїв і розвиток новітніх технологій, формує складне і нестабільне середовище функціонування ринку електроенергії.

Український ринок електроенергії є складним і багатогранним сектором економіки, що перебуває у процесі постійних трансформацій. Після реформування у 2019 році він перейшов до моделі, яка базується на конкурентних принципах та ринковому ціноутворенні, де діє багаторівнева модель, яка включає ринок "на добу наперед", внутрішньодобовий ринок, балансуючий ринок та ринок допоміжних послуг. Кожен із цих сегментів функціонує відповідно до власних механізмів взаємодії попиту та пропозиції, а ефективність їхньої роботи значною мірою залежить від можливості здійснювати обґрунтовані прогнози основних параметрів [1].

Сучасна практика управління в енергетиці все частіше спирається на дані, отримані з великих масивів інформації, що накопичуються операторами ринку, системними операторами, прогнозними агентствами та регуляторами. Ці дані охоплюють широкий спектр показників: обсяги генерації, споживання,

ринкові ціни, балансові відхилення, погодні умови, структуру виробництва тощо. Ефективне використання цих даних вимагає впровадження методів інтелектуального аналізу, які дозволяють виявляти приховані закономірності, будувати прогностичні моделі та оптимізувати прийняття управлінських рішень.

Потреба у розробці надійних і якісних прогнозних рішень зумовлена не лише економічними чинниками, а й технологічними викликами. Інтеграція відновлюваних джерел енергії, які мають нерегулярний характер генерації, вимагає оперативного реагування на короткострокові зміни попиту та пропозиції. В умовах, коли навіть незначні відхилення можуть призвести до дисбалансу в енергосистемі, наявність ефективного прогнозного інструменту стає критично важливою.

Таким чином, мотивація дослідження полягає у необхідності системного підходу до побудови та оцінювання моделей прогнозування на ринку електроенергії з урахуванням як специфіки самої галузі, так і сучасних вимог до обробки інформації. Очікується, що результати дослідження сприятимуть підвищенню якості прогнозування в енергетичній галузі, забезпечать аналітичну підтримку процесів балансування системи та дозволять більш ефективно адаптуватися до умов динамічного середовища.

1.2 Актуальність теми

У сучасних умовах ринок електроенергії зазнає масштабних трансформацій, зумовлених як глобальними енергетичними трендами, так і внутрішніми чинниками розвитку енергосистеми України. Підвищення частки відновлюваних джерел енергії, децентралізація генерації, розвиток цифрових технологій, інтеграція до європейського енергетичного простору — все це ставить перед галуззю нові виклики. Одним із таких викликів є необхідність

якісного та своєчасного прогнозування ключових параметрів ринку, зокрема цінових індексів, обсягів генерації, споживання, а також балансуючих відхилень у реальному часі [2, 3].

Складність ринку електроенергії полягає в його багаторівневій структурі та високій чутливості до зовнішніх і внутрішніх змін. З одного боку, ринкові механізми потребують об’єктивного й оперативного інформаційного супроводу. З іншого — стрімкий розвиток енергетичних технологій вимагає інтеграції сучасних методів обробки даних і прогнозування в управлінські процеси. У випадку України ці виклики набувають особливого значення через відновлення інфраструктури, потребу в зміцненні енергетичної безпеки та адаптацію до нових європейських норм функціонування ринку [4].

Особливістю українського ринку електроенергії є його динамічність та нестабільність: суттєві коливання обсягів генерації, залежність від погодних умов і сезонності, зміни регуляторних правил та нерівномірне навантаження між регіонами. У зв’язку з цим, ефективне управління енергетичними потоками неможливе без якісного прогнозування, яке дозволяє уникати дисбалансів, забезпечити стале функціонування системи та оптимізувати роботу учасників ринку [5].

Традиційні статистичні методи прогнозування часто не враховують усі складнощі системи, особливо у випадках, коли спостерігається нерегулярна поведінка показників, вплив зовнішніх подій (наприклад, аварій, воєнних дій, ринкових шоків), або коли мають місце різнотипні часові лаги. Тому актуальним є використання сучасних підходів до аналізу даних — методів інтелектуального аналізу, що ґрунтуються на вивченні історичних даних, виявленні прихованих закономірностей, а також адаптації моделей до нових патернів [6].

Крім цього, світова енергетична спільнота впроваджує концепцію Smart Grid, яка передбачає інтеграцію високоточної аналітики, автоматизованого управління навантаженнями та взаємодії з розподіленими джерелами генерації. Основою таких систем є саме прогнозні моделі, які працюють в умовах

великого обсягу даних, високої частоти оновлення інформації та необхідності реагування в реальному часі [7]. Відповідно, створення ефективних моделей прогнозування із застосуванням методів інтелектуального аналізу даних є ключовим елементом цифрової трансформації енергетики.

Актуальність теми дослідження також підтверджується тим, що дані є основним ресурсом для прийняття рішень на всіх рівнях управління ринком — від стратегічного (планування розвитку генерації) до оперативного (розрахунок обсягів купівлі-продажу електроенергії на внутрішньодобовому ринку). Надійне прогнозування дозволяє знижувати ризики, уникати фінансових втрат та покращувати якість планування і регулювання [2, 5].

Таким чином, тема дослідження є вкрай актуальною як для України, так і для світової практики в цілому. Вона поєднує в собі дослідження складних ринкових систем, сучасні технології обробки даних, алгоритмічні рішення та прикладні завдання галузевого рівня. Результати роботи мають потенціал подальшого використання як у дослідницьких цілях, так і для розробки практичних інструментів прогнозування в реальному енергетичному середовищі.

1.3 Мета дослідження

У сучасних умовах розвитку енергетичного сектору надзвичайно важливою є здатність систем прогнозування оперативно та точно оцінювати майбутню поведінку основних показників ринку електроенергії. Це пояснюється не лише технічними, а й економічними, регуляторними та геополітичними викликами, що суттєво впливають на баланс попиту та пропозиції, формування ринкової ціни, структуру генерації та рівень енергетичної безпеки країни. У зв’язку з цим дедалі більшої ваги набуває

необхідність використання сучасних методів інтелектуального аналізу даних у задачах прогнозування, що дозволяє перейти від інтуїтивного управління до обґрунтованих рішень, які спираються на математичне та статистичне підґрунтя.

Метою даного дослідження є розроблення та обґрунтування моделей інтелектуального аналізу даних для підвищення якості прогнозування ключових показників функціонування ринку електроенергії в умовах нестабільності, багатофакторності та інформаційної складності. Така мета визначає не лише науковий, а й прикладний характер дослідження, що спрямоване на вирішення реальних задач управління енергетичними процесами та цифрової трансформації галузі.

Ринок електроенергії України має характерну складність, що проявляється у високій волатильності часових рядів, залежності від сезонних коливань, швидкозмінності регуляторного середовища, інтеграції відновлюваних джерел енергії та значному впливі зовнішніх факторів. За таких умов прогнозування навіть на короткостроковому горизонті вимагає використання гнучких, адаптивних моделей, здатних до самонавчання та інтеграції великого обсягу вхідної інформації з різних джерел. Отже, мета дослідження полягає не лише в теоретичному аналізі сучасних підходів, але й у практичному застосуванні моделей, що базуються на принципах штучного інтелекту, статистичного навчання та аналізу структурованих часових рядів.

Досягнення поставленої мети передбачає формування єдиного методологічного підходу до аналізу часових рядів у сфері енергетики, що базується на поєднанні системного аналізу, математичного моделювання та інформаційних технологій. Здійснення прогнозу в реальному часі, адаптація моделі до змін ринкової ситуації, обробка неповних або зашумлених даних, виявлення прихованих закономірностей у багатовимірних рядах — усі ці задачі є складовими досягнення мети дослідження. Практична реалізація таких підходів дозволяє не лише підвищити ефективність роботи окремих суб’єктів

ринку, але й сприяє формуванню конкурентного, стабільного та відкритого енергетичного середовища відповідно до європейських стандартів.

Таким чином, сформульована мета відповідає актуальним викликам сучасного енергетичного ринку, що перебуває в стані активної трансформації, і визначає теоретичну й прикладну спрямованість подальших етапів дослідження, включаючи побудову моделей, аналіз їхньої ефективності та розробку рекомендацій щодо впровадження у практичну діяльність операторів ринку та інших зацікавлених сторін.

1.4 Постановка задачі

Ринок електроенергії належить до найскладніших об'єктів прикладного аналізу в силу своєї динамічності, нерівномірності розподілу навантажень, багатофакторності впливу та постійної зміни умов функціонування. У сучасному енергетичному середовищі особливої значущості набуває здатність суб’єктів ринку оперативно адаптуватися до зміни показників попиту, генерації, ціноутворення та балансування. При цьому, якість управлінських рішень напряму залежить від точності прогнозної інформації, що стосується ключових ринкових параметрів. В умовах інтеграції відновлюваних джерел енергії, лібералізації торгівлі електроенергією та підвищення залежності ринку від погодних, геополітичних та економічних чинників, прогнозування таких показників стає критично важливою задачею для стабільного функціонування всієї системи.

Незважаючи на наявність широкого спектру моделей, які застосовуються для прогнозування часових рядів в економічних і технічних системах, існуючі підходи часто мають обмежене застосування в умовах ринку електроенергії. По-перше, багато моделей не враховують високий ступінь

стохастичності даних, які формуються під впливом випадкових або короткострокових факторів. По-друге, традиційні методи мають труднощі з обробкою структурно неоднорідних, неповних або нерегулярних даних, які часто зустрічаються в енергетичних базах. По-третє, багато існуючих підходів не мають належної гнучкості в адаптації до нових умов або змін у структурі системи.

Становлення нової парадигми в енергетичній аналітиці, яка передбачає інтеграцію методів машинного навчання, гібридних і нейромережевих підходів, спричинило зростання наукового інтересу до створення моделей прогнозування, здатних забезпечити високу якість і стійкість до змін середовища. Водночас впровадження таких моделей вимагає попереднього аналізу джерел даних, формалізації закономірностей часових рядів, побудови відповідного інструментарію обробки, навчання моделей на історичних даних і подальшої оцінки їх ефективності в реальних умовах функціонування ринку.

Ураховуючи актуальність прогнозування показників ринку електроенергії, постає науково-практична задача, яка полягає у формалізації процесу побудови та адаптації моделей інтелектуального аналізу даних, здатних працювати з різними типами часових рядів, враховувати багатофакторність впливу та забезпечувати достатню якість прогнозу в умовах невизначеності. Для цього необхідно не лише дослідити властивості часових рядів енергетичних показників, але й забезпечити цілісний підхід до їх обробки, що охоплює як етапи попередньої підготовки даних, так і побудову моделей, їх навчання, валідацію та практичне тестування. Важливо також здійснити критичну оцінку результатів прогнозування з використанням відповідних кількісних метрик і визначити межі застосовності запропонованих рішень у межах реального ринку.

Таким чином, задача дослідження полягає в обґрунтуванні підходів до прогнозування ключових показників ринку електроенергії на основі сучасних методів інтелектуального аналізу даних з урахуванням специфіки об'єкта дослідження, особливостей часових рядів, складності структури впливів та

вимог до точності й адаптивності моделей. Постановка задачі охоплює як науковий аналіз існуючих підходів і виявлення їх недоліків, так і розробку нової або адаптованої методології, яка дозволить підвищити ефективність процесів прогнозування в енергетичній галузі.

1.5 Завдання дослідження

Відповідно до сформульованої мети дослідження, яка полягає у розробці моделей інтелектуального аналізу даних для прогнозування ключових показників функціонування ринку електроенергії, постає необхідність у конкретизації наукових та практичних завдань, вирішення яких дозволить досягти запланованого результату. Завдання дослідження формулюються з урахуванням специфіки предметної області, складності структури ринку електроенергії, а також обсягу і типу вхідних даних, що використовуються в моделюванні.

Сучасний ринок електроенергії функціонує як багаторівнева система з високим рівнем взаємозалежності між попитом, пропозицією, погодними умовами, регуляторними рішеннями та інституційною поведінкою учасників ринку. У зв’язку з цим формулювання завдань дослідження має враховувати як теоретичний, так і прикладний рівні аналізу.

Першим і фундаментальним завданням дослідження є систематизація існуючих підходів до прогнозування часових рядів у сфері енергетики та аналіз сучасного стану методів інтелектуального аналізу даних, що застосовуються у світовій практиці. Це завдання дозволяє виявити переваги, недоліки та межі застосовності різних підходів у контексті функціонування енергетичного ринку, зокрема на прикладі української системи електропостачання.

Другим важливим завданням є ідентифікація особливостей та закономірностей часових рядів ключових енергетичних показників, таких як обсяги генерації, споживання, ринкова ціна електроенергії, обсяги балансуючих відхилень тощо. Ці часові ряди, як правило, характеризуються сезонністю, трендами, а також стохастичними впливами, тому їх попередній аналіз дозволяє сформулювати обґрунтовані вимоги до прогнозної моделі.

Наступним завданням є підготовка вхідних даних до моделювання, що включає попередню обробку, очищення, нормалізацію та перетворення інформації у відповідні формати. Особливу увагу слід приділяти заповненню пропущених значень, корекції викидів та забезпеченню часової узгодженості даних, оскільки від якості підготовлених вибірок значною мірою залежить успішність прогнозування.

Четвертим завданням є побудова, апробація та порівняльна оцінка моделей інтелектуального аналізу даних, орієнтованих на прогнозування енергетичних показників. У цьому контексті важливою є не лише якість передбачення, але й інтерпретованість результатів, стабільність моделей до зміни середовища, а також їх здатність до адаптації в умовах невизначеності.

П’ятим завданням дослідження є оцінювання якості прогнозів за допомогою відповідних метрик, що враховують як абсолютні, так і відносні помилки передбачення. На основі результатів порівняння моделей формуються висновки щодо доцільності їх практичного застосування в умовах функціонування енергетичного ринку України.

Завершальне завдання полягає у формуванні рекомендацій щодо впровадження розроблених моделей у практичну діяльність суб’єктів енергетичного ринку, включаючи операторів систем передачі, трейдерів, аналітичні підрозділи регуляторних органів та учасників ринку "на добу наперед" і внутрішньодобового ринку. Такі рекомендації мають сприяти покращенню якості рішень, мінімізації ризиків та оптимізації стратегічного планування.

Таким чином, виконання зазначених завдань є необхідною умовою досягнення загальної мети дослідження та забезпечує цілісність і логічну завершеність дослідницького процесу.

1.6 Об’єкт і предмет дослідження

Для досягнення поставленої мети та реалізації завдань дослідження необхідним є чітке визначення об’єкта та предмета дослідження, оскільки саме ці елементи задають межі наукового аналізу, конкретизують напрям опрацювання теми та слугують методологічною основою для подальших теоретичних і прикладних узагальнень. Правильне формулювання об’єкта й предмета дослідження забезпечує логічну завершеність структури наукової роботи, її цілісність та методичну обґрунтованість.

У межах даної кваліфікаційної роботи об’єктом дослідження виступає ринок електроенергії як складна динамічна система, що характеризується багаторівневою організацією, взаємозалежністю суб’єктів, постійною зміною зовнішніх і внутрішніх параметрів, а також значною інформаційною насиченістю. Як елемент енергетичного сектору, ринок електроенергії функціонує на стику технічних, економічних і регуляторних процесів, де прийняття рішень значною мірою базується на кількісному аналізі часових рядів, які описують зміну основних показників у часі. Особливістю функціонування такого ринку є чутливість до коливань споживання та генерації, до змін погодних умов, політичної ситуації, ринкової лібералізації, а також впровадження інновацій у виробничі та облікові процеси.

Предметом дослідження у межах дипломної роботи є методи та моделі інтелектуального аналізу даних, які застосовуються для прогнозування ключових показників ринку електроенергії на основі часових рядів, з

урахуванням їхньої стохастичної, сезонної та трендової природи. Зокрема, дослідження фокусується на алгоритмах обробки історичних даних, моделюванні поведінки змінних у динаміці, оцінюванні якості прогнозів, а також адаптації відповідних методів до умов українського енергетичного ринку. Предмет дослідження охоплює як загальні методологічні засади інтелектуального аналізу даних, так і прикладні аспекти реалізації моделей прогнозування в інформаційно-аналітичних системах.

Об’єкт дослідження визначає системний рівень вивчення проблематики, що охоплює загальні закономірності функціонування енергетичних ринків, а предмет — конкретизує науковий інтерес на окремому аспекті, який безпосередньо підлягає моделюванню, аналізу та оцінюванню. Відповідно, дослідження зосереджене на вивченні аналітичних характеристик ринкових даних, визначенні ефективних моделей прогнозування та розробці методичних підходів до їх реалізації на практиці.

Таким чином, об’єкт і предмет дослідження формують єдину логічну основу наукового пошуку, що дозволяє не лише описати загальні процеси на ринку електроенергії, а й сформувати інструментарій для їх прогнозування в умовах складної, багатофакторної та нестабільної системи.

1.7 Методи дослідження

Для досягнення поставленої мети та реалізації визначених завдань у межах даної дипломної роботи було застосовано комплекс наукових методів, що включають загальнонаукові, математико-статистичні, обчислювальні та інтелектуальні підходи до аналізу часових рядів. Методологічною основою дослідження став системний підхід, який дозволив розглядати ринок електроенергії як складну адаптивну систему, що формується під впливом

широкого кола взаємопов’язаних економічних, технічних та зовнішніх чинників.

У теоретичній частині дослідження було застосовано методи аналізу, синтезу, узагальнення, порівняння та критичного осмислення результатів попередніх наукових праць. Це дало змогу ідентифікувати сучасні тенденції в галузі прогнозування часових рядів, систематизувати підходи до обробки енергетичних даних і виявити основні обмеження традиційних моделей у контексті їх застосування до динамічних ринкових умов.

Для попереднього аналізу даних та виявлення статистичних характеристик часових рядів було використано інструментарій математичної статистики, зокрема автокореляційний аналіз, перевірку стаціонарності, а також графічну декомпозицію на трендову, сезонну та випадкову компоненти. Ці методи дали змогу обґрунтувати вибір відповідного класу моделей прогнозування.

У рамках практичної реалізації моделювання було обрано три підходи: статистичний, нейромережевий та гібридний. Статистичне прогнозування реалізовано за допомогою моделі SARIMAX, яка дозволяє ураховувати сезонні ефекти, авторегресію, інтегрування, ковзне середнє та зовнішні регресори. Для обробки нелінійних залежностей між спостереженнями застосовано нейромережеву модель LSTM, що належить до класу рекурентних нейронних мереж із механізмом довготривалої пам’яті. Завдяки цій моделі вдалося врахувати довготривалі залежності у часових рядах і покращити якість коротко- та середньострокового прогнозування. Гібридний підхід було реалізовано через модель ANFIS (Adaptive Neuro-Fuzzy Inference System), яка поєднує логіку нечітких висновків і здатність нейронної мережі до навчання. Така модель дозволила створити інтерпретовані правила поведінки часових рядів на основі статистичних закономірностей і адаптивних механізмів навчання.

Оцінювання точності та ефективності побудованих моделей здійснювалося з використанням поширених метрик: середньої абсолютної

похибки (MAE), середньоквадратичної похибки (RMSE), коефіцієнт детермінації (R^2) а також середньої абсолютної відносної похибки (MAPE). Результати аналізу дозволили здійснити обґрунтоване порівняння моделей між собою за якістю прогнозу та адаптивністю до нових даних.

Реалізація моделей, обробка даних, візуалізація результатів і управління робочими середовищами проводилися з використанням інтегрованого середовища JetBrains DataSpell — професійного IDE для роботи з даними, що забезпечує зручне управління структурами проєктів, інтерактивну взаємодію з візуалізаціями та повну підтримку середовища Jupyter Notebook. Усі експерименти виконувалися мовою програмування Python, з використанням бібліотек NumPy, pandas, scikit-learn, statsmodels, TensorFlow, Keras, matplotlib, seaborn та ін.

Таким чином, реалізація зазначених методів дозволила провести якісне й кількісне моделювання динаміки ключових показників ринку електроенергії, забезпечивши багаторівневу оцінку результатів та створення обґрунтованих прогнозних рішень у контексті задач інтелектуального аналізу даних.

1.8 Висновки до розділу 1

У першому розділі дипломної роботи було здійснено комплексний аналіз теоретичних і методологічних засад, що становлять основу дослідження процесів прогнозування в енергетичній сфері з використанням інтелектуальних методів аналізу даних. Розкрито актуальність теми дослідження в контексті динамічних змін, що відбуваються на ринку електроенергії, зокрема з огляду на зростання ролі відновлюваних джерел, децентралізацію генерації, коливання попиту, а також значну залежність

ринку від зовнішніх факторів економічного, політичного та кліматичного характеру.

Постановка задачі дослідження дозволила окреслити ключову наукову проблему — необхідність підвищення якості прогнозування ринкових показників за рахунок впровадження адаптивних моделей, здатних до роботи з великомасштабними, сезонними та стохастично залежними часовими рядами. На основі цього були сформульовані конкретні завдання, які охоплюють як теоретичне вивчення проблематики, так і прикладну реалізацію моделей прогнозування з подальшим оцінюванням їх ефективності.

У межах розділу чітко окреслено об’єкт і предмет дослідження. Як об’єкт виступає ринок електроенергії в його багаторівневій структурі, а як предмет — методи й моделі інтелектуального аналізу даних, застосовувані для прогнозування його ключових показників. Такий підхід дозволяє сфокусувати дослідження на найбільш релевантному та прикладному аспекті сучасної енергетичної аналітики.

Окрему увагу було приділено методам дослідження, серед яких особливо важливими є статистичні підходи, алгоритми машинного навчання та гібридні інтелектуальні моделі. В роботі передбачено використання таких моделей, як SARIMAX, LSTM та ANFIS, що дає змогу поєднати переваги класичних і сучасних підходів до прогнозування з метою досягнення високої точності та адаптивності прогнозів.

Узагальнюючи, можна зробити висновок, що розділ заклав теоретико-методологічне підґрунтя для проведення практичної частини дослідження. Зібрані та систематизовані відомості дозволяють перейти до етапу побудови моделей, підготовки даних і проведення експериментального моделювання, що й буде розкрито у наступних розділах дипломної роботи.

РОЗДІЛ 2 ОГЛЯД СУЧАСНИХ АРХІТЕКТУР

2.1 Основні поняття та властивості часових рядів

Часовий ряд — це впорядкована у часі послідовність спостережень деякого показника, які отримані через рівні або нерівні інтервали часу. У контексті енергетичних ринків часові ряди можуть представляти, зокрема, ціну електроенергії, її обсяги купівлі та продажу, а також відповідні оголошені значення. Основною метою аналізу часових рядів є побудова моделей, що дозволяють адекватно описати внутрішню структуру даних та здійснити їх точне прогнозування [8].

Ключові характеристики часових подані нижче.

1. Стаціонарність — це властивість ряду, за якої його статистичні характеристики (математичне сподівання, дисперсія, автокореляція) залишаються незмінними у часі. Відповідно до слабкої стаціонарності, математичне сподівання ряду повинно бути сталим, дисперсія — постійною, а автокореляція — залежати лише від лагу, а не від абсолютного часу [9]. Для перевірки стаціонарності застосовуються такі статистичні тести, як ADF (Augmented Dickey-Fuller) та KPSS (Kwiatkowski-Phillips-Schmidt-Shin) [10].
2. Тренд — довгострокова зміна середнього рівня часового ряду. Він може бути зростаючим, спадним або нелінійним. Виявлення тренду дозволяє уточнити модель прогнозування. Методи виділення тренду включають ковзне середнє, регресійне згладжування або фільтрацію Ходріка-Прескотта (HP-фільтр) [11].
3. Сезонність — періодичні коливання значень часового ряду, які повторюються через однакові інтервали часу (наприклад, добова або

тижнева циклічність). Сезонність може бути адитивною (вплив не залежить від рівня ряду) або мультиплікативною (вплив залежить від рівня ряду). У випадку електроенергетичного ринку сезонні коливання яскраво виражені, наприклад, внаслідок зміни попиту між робочими та вихідними днями [12].

4. Шум — випадкова компонента часового ряду, що не має системної природи. Шум ускладнює побудову прогнозних моделей і часто вимагає попередньої фільтрації чи згладжування. Для боротьби з шумом використовуються методи, що базуються на регуляризації, фільтрах Кальмана або баєсових підходах.

Більшість моделей класичного прогнозування, як-от ARIMA, SARIMA, вимагають стаціонарності вхідного ряду. Оскільки реальні часові ряди часто є нестаціонарними, застосовуються різні методи їх приведення до стаціонарного вигляду.

1. Диференціювання — обчислення різниці між сусідніми значеннями. Перша різниця дозволяє усунути лінійний тренд, друга — параболічний тощо.
2. Сезонне диференціювання — вилучення сезонного компоненту шляхом обчислення різниці між значеннями на певному лагу, як правило, кратному сезонному циклу (наприклад, для добової сезонності).
3. Логарифмування — корисне для стабілізації дисперсії, особливо коли спостерігається експоненційне зростання або спад.
4. Згладжування — застосовується для виявлення основної тенденції ряду. Сюди входять методи експоненційного згладжування, ковзного середнього та інші.
5. Детрендування — вилучення трендової компоненти шляхом регресійного моделювання або фільтрації.

Коректне розуміння структури часових рядів є необхідною передумовою побудови точних моделей прогнозування. Визначення трендів,

сезонності та шуму дозволяє правильно вибрати архітектуру моделі та покращити її прогностичну здатність.

У наступних підрозділах буде здійснено математичну формалізацію моделей SARIMAX, LSTM та ANFIS, які використовувалися у межах цього дослідження.

2.2 Класичні статистичні методи прогнозування

Класичні статистичні методи прогнозування є одними з найдавніших та найбільш досліджених підходів до аналізу часових рядів. Вони базуються на припущеннях про лінійну структуру даних, стаціонарність процесу та наявність автокореляційних залежностей. Серед найбільш відомих методів варто виділити моделі AR, MA, ARMA, ARIMA, SARIMA та їх розширення з урахуванням зовнішніх регресорів — SARIMAX.

Автоматичне прогнозування у рамках цих моделей передбачає аналіз попередніх значень часового ряду та, за необхідності, врахування впливу інших (екзогенних) змінних.

1. Модель AR (autoregressive model) — авторегресійна модель порядку p описується рівнянням:

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

де X_t — значення ряду в момент часу t;

ϕ_i — коефіцієнти авторегресії;

c — константа;

ε_t — білий шум.

2. Модель MA (moving average model) — модель ковзного середнього порядку q:

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

де μ — середнє значення ряду;

θ_i — коефіцієнти ковзного середнього.

3. Модель ARMA (autoregressive moving average) — поєднання AR(p) та MA(q):

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t$$

Модель ARMA застосовується до стаціонарних рядів.

4. Модель ARIMA (autoregressive integrated moving average) — узагальнення ARMA, яке дозволяє працювати з нестаціонарними рядами шляхом диференціювання:

$$\Delta^d X_t = c + \sum_{i=1}^{p} \phi_i \Delta^d X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t,$$

де d — порядок диференціювання, необхідний для досягнення стаціонарності [13].

5. Модель SARIMA (seasonal ARIMA) — доповнення ARIMA сезонною компонентою (P, D, Q, s). Записується як:

$$\Phi_P(L^s)\phi_p(L)\Delta^d\Delta_s^D X_t = \Theta_Q(L^s)\theta_q(L)\varepsilon_t,$$

де Δ_s^D — сезонне диференціювання;

L — оператор лагу;

s — довжина сезонного циклу [14].

6. Модель SARIMAX (SARIMA with exogenous variables) — розширення моделі SARIMA, що враховує зовнішні регресори:

$$Y_t = X_t\beta + SARIMA(p,d,q)(P,D,Q)_s + \varepsilon_t,$$

де $X_t\beta$ — вплив екзогенних змінних, які можуть представляти зовнішні фактори (наприклад, свята, погодні умови, макроекономічні індикатори).

SARIMAX дозволяє враховувати сезонність, тренд та вплив зовнішніх змінних в одному підході. Це робить її особливо корисною в

енергетичних системах, де важливо враховувати як внутрішню часову динаміку, так і зовнішні тригери.

У практичному застосуванні підбір параметрів моделей здійснюється на основі критеріїв інформації, таких як AIC (Akaike Information Criterion), BIC (Bayesian Information Criterion), а також графічного аналізу автокореляційних та часткових автокореляційних функцій (ACF та PACF).

Таким чином, класичні статистичні моделі, зокрема SARIMAX, є потужним інструментом для моделювання лінійних та сезонних структур часових рядів, особливо у випадках, коли обсяг даних є обмеженим, а модель має бути інтерпретованою.

2.3 Нейронні мережі для прогнозування часових рядів

У контексті задач прогнозування часових рядів, нейронні мережі зарекомендували себе як потужні інструменти для моделювання складних, нелінійних, високоволатильних і сезонних процесів. Серед усіх підходів особливу роль відіграють рекурентні нейронні мережі (RNN) та їхні вдосконалення, зокрема архітектура LSTM (Long Short-Term Memory).

Рекурентні нейронні мережі (RNN) — це клас штучних нейронних мереж, спеціально призначений для обробки послідовних даних [15]. Основна ідея полягає в тому, що кожен вихід поточного шару залежить не лише від поточного входу, але й від стану попереднього кроку:

$$h_t = \sigma(Wx_t + Uh_{t-1} + b),$$

де x_t — вхід на момент часу t;

h_t — прихований стан;

W, U — вагові матриці;

σ — функція активації (зазвичай tanh або ReLU).

Хоча RNN добре моделюють часову залежність, вони страждають від проблеми затухаючих або вибухаючих градієнтів, що унеможливлює навчання на довгих послідовностях. Для подолання цих обмежень було запропоновано архітектуру LSTM [16].

Long Short-Term Memory (LSTM) — покращена версія RNN, розроблена спеціально для моделювання довготривалих залежностей [17]. LSTM включає спеціальні компоненти — осередок пам'яті (cell) та керуючі вентилі, які дозволяють контролювати потік інформації.

1. Forget gate:

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

2. Input gate:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \qquad \tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

3. Update осередку пам'яті:

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$$

4. Output gate:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \qquad h_t = o_t * \tanh(c_t)$$

Така структура дозволяє мережі вибірково зберігати або забувати інформацію та підтримувати контекст у довгих послідовностях. Завдяки цьому LSTM є надзвичайно ефективною для прогнозування часових рядів, де мають місце як короткотермінові, так і довгострокові залежності [18].

Застосування LSTM у прогнозуванні часових рядів має наступні переваги [3,5].

1. Моделювання складної нелінійної динаміки.
2. Врахування сезонності та трендів без необхідності явного виділення цих компонент.
3. Адаптація до змін у структурі даних.
4. Висока точність при великому обсязі даних.

Окрім базової архітектури, часто використовуються розширення.

1. Stacked LSTM.
2. Bidirectional LSTM.
3. Seq2Seq LSTM.
4. LSTM з attention-механізмом [18].

Процес навчання LSTM зазвичай базується на оптимізації функції втрат (наприклад, MSE) з використанням методів стохастичного градієнтного спуску (SGD, Adam), а для регуляризації може використовуватись dropout між шарами [17].

LSTM успішно застосовуються для прогнозування обсягів споживання електроенергії, цін на ринку, навантаження в мережах, попиту у торгівлі тощо [19]. Вони здатні опрацьовувати великі масиви даних з високою частотою спостережень і виявляти закономірності, які недоступні для лінійних моделей.

2.4 Адаптивні нейро-нечіткі системи (ANFIS)

Адаптивна нейро-нечітка система виведення (ANFIS — Adaptive Neuro-Fuzzy Inference System) поєднує в собі елементи нечіткої логіки та штучних нейронних мереж, забезпечуючи потужний інструмент для моделювання складних систем, що характеризуються нечіткістю, неповнотою або неточністю інформації. Такий підхід особливо корисний у випадках, коли складно сформулювати чіткі математичні залежності, але можливо використовувати евристичні правила або мовні конструкції.

2.4.1 Теоретичні основи нечіткої логіки

Нечітка логіка була запропонована Лотфі Заде у 1965 році [20] як альтернатива класичній логіці, що оперує лише істинними або хибними твердженнями (0 або 1). Нечітка логіка дозволяє описувати значення, які належать певній множині частково, через функцію належності . В основі нечітких систем лежать нечіткі множини, правила типу "якщо-то", а також операції агрегації та дефазифікації (перетворення нечітких висновків у числове значення).

Основна структура системи нечіткого виведення (FIS — Fuzzy Inference System) включає наступні елементи.

1. базу правил (напр. "Якщо — великий, то — малий").
2. базу даних (функції належності).
3. механізм логічного виведення.
4. механізм дефазифікації (наприклад, метод центру ваги).

Найпоширенішою є модель нечіткого виведення Типу Такагі-Суґено (TS модель) [21], де результатом правила є лінійна або константна функція.

2.4.2 ANFIS механізми та архітектура

ANFIS був вперше запропонований Джан-Ши Роджером у 1993 році [22]. Ця система реалізує нечітку інференцію у вигляді нейронної мережі, яка дозволяє автоматично навчати параметри функцій належності та висновків.

Архітектура ANFIS зазвичай містить п’ять рівнів (шарів).

1. Шар 1 (вхідний — функції належності):

   Кожен нейрон у цьому шарі представляє нечітку множину та обчислює рівень належності входу x до неї:

$$O_1^i = \mu_{A_i}(x), \qquad O_1^j = \mu_{B_j}(y),$$

де μ_{A_i}, μ_{B_j} — функції належності (наприклад, гаусова або трикутна).

2. Шар 2 (правила):

   Нейрони цього шару обчислюють ступінь активації правила через добуток:

$$w_{ij} = \mu_{A_i}(x) * \mu_{B_j}(y)$$

3. Шар 3 (нормалізація):

   Ваги нормалізуються:

$$\overline{w}_{ij} = \frac{w_{ij}}{\sum_k w_k}$$

4. Шар 4 (висновки):

   Нейрони цього шару генерують вихід правил:

$$O_4^{ij} = \overline{w}_{ij} * f_{ij}, \qquad f_{ij} = a_0^{ij} + a_1^{ij}x + a_2^{ij}y$$

5. Шар 5 (агрегація):

   Підсумовується результат усіх правил:

$$y = \sum_{i,j} \overline{w}_{ij} f_{ij}$$

Процес навчання ANFIS виконується в два етапи (гібридний алгоритм).

1. Перший етап — визначення параметрів функцій належності за допомогою градієнтного спуску.
2. Другий етап — оцінка параметрів лінійних функцій наслідків за допомогою методу найменших квадратів (LSE) [23].

Таким чином, навчання ANFIS поєднує в собі переваги адаптивності нейронних мереж та інтерпретованості нечітких систем. На практиці використання ANFIS є ефективним у випадках, коли присутні обмежені обсяги даних, нечітка природа задачі або потреба у прозорості прийняття рішень.

Застосування ANFIS охоплює численні галузі, включаючи енергетику (регулювання навантаження), фінанси (оцінка ризиків), медицину

(діагностика), та багато інших. У задачах прогнозування часових рядів ANFIS дозволяє обробляти вхідні шаблони як нечіткі значення, зберігаючи гнучкість та точність навіть у випадках неповної або спотвореної інформації [24].

2.5 Метрики оцінювання якості прогнозу

Для кількісної оцінки якості моделей прогнозування часових рядів застосовуються різноманітні статистичні метрики, які дозволяють оцінити відхилення прогнозованих значень від фактичних. Метрики мають принципове значення при порівнянні різних моделей між собою, а також для валідації результатів у практичних задачах. Найпоширенішими є: середня квадратична помилка (MSE), середня абсолютна похибка (MAE), корінь середньої квадратичної помилки (RMSE) та коефіцієнт детермінації (R^2).

2.5.1 Середня квадратична помилка (MSE)

Середня квадратична помилка (Mean Squared Error) є однією з базових метрик оцінки прогнозів. Вона обчислює середній квадрат різниць між фактичними та прогнозованими значеннями:

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2,$$

де y_t — фактичне значення;

$\hat{y}_t$ — прогнозоване значення;

n — кількість спостережень.

MSE особливо чутлива до великих відхилень, оскільки квадратична форма штрафує великі помилки сильніше, ніж малі. Ця властивість робить MSE корисною у випадках, коли важливо мінімізувати великі промахи. Недоліком є наявність квадратичної одиниці виміру (наприклад, якщо вихід у кіловатах, MSE буде в кіловатах квадратних) [25].

2.5.2 Середня абсолютна похибка (MAE)

Середня абсолютна похибка (Mean Absolute Error) визначається як середнє абсолютне відхилення прогнозу від фактичного значення:

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

MAE є інтуїтивно зрозумілою та легко інтерпретованою метрикою: вона виражає середню величину помилки в тих самих одиницях, що й прогнозовані дані. MAE не надто чутлива до окремих викидів, що може бути як перевагою (у задачах із шумними даними), так і недоліком (якщо важливо враховувати великі промахи) [26].

2.5.3 Корінь середньої квадратичної помилки (RMSE)

Корінь середньої квадратичної помилки (Root Mean Squared Error) є квадратним коренем із MSE:

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

RMSE часто використовується як основна метрика при навчанні нейронних мереж і в задачах регресії, оскільки вона зберігає одиниці виміру вихідного ряду. RMSE дозволяє краще виявляти великі помилки та підкреслює вплив віддалених спостережень. У випадках із нормально розподіленими помилками RMSE має добре обґрунтоване статистичне тлумачення [27].

2.5.4 Коефіцієнт детермінації (R^2)

Коефіцієнт детермінації (R^2) або коефіцієнт поясненої дисперсії характеризує ступінь узгодженості між прогнозованими та фактичними значеннями. Він обчислюється як:

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{n}(y_t - \bar{y}_t)^2}$$

де $\bar{y}$ — середнє значення фактичного ряду.

$R^2 = 1$ означає ідеальне передбачення, а $R^2 = 0$ свідчить про те, що модель не краща за наївне прогнозування (середнє значення). Значення $R^2 < 0$ вказує на те, що модель гірша за середню. На відміну від MAE та RMSE, R^2 є безрозмірною метрикою. Вона дозволяє зрозуміти, яка частина варіації в даних пояснюється моделлю [28].

Усі перелічені метрики відіграють важливу роль у порівнянні продуктивності моделей. Наприклад, MAE забезпечує прозору інтерпретацію похибки, тоді як RMSE — чутливість до великих відхилень. R^2 дозволяє оцінити загальну ефективність моделі без врахування розмірності, а MSE широко застосовується у процесі оптимізації функції втрат під час навчання.

2.6 Висновки до розділу 2

У другому розділі було розглянуто теоретичну основу та методологічні підходи до прогнозування часових рядів, що є основою для подальшого практичного моделювання в контексті дослідження енергетичного ринку України. Детальний аналіз понять часових рядів дозволив охарактеризувати ключові властивості даних — зокрема стаціонарність, тренд, сезонність і шум — а також окреслити необхідні трансформації для приведення рядів до стану, придатного для моделювання.

Серед класичних підходів було розглянуто моделі ARIMA та SARIMA, а також їх розширення — SARIMAX, яке дозволяє враховувати зовнішні регресори та сезонні компоненти. Ці моделі є основоположними для статистичного прогнозування та мають перевагу в інтерпретованості та стійкості при роботі з обмеженим обсягом даних.

У рамках сучасних інтелектуальних методів прогнозування було розглянуто рекурентні нейронні мережі, з особливим акцентом на архітектурі LSTM, що здатна моделювати як коротко-, так і довгострокові залежності в даних. Також охарактеризовано адаптивні нейро-нечіткі системи (ANFIS), які поєднують переваги нечіткої логіки та нейромереж для побудови прозорих, адаптивних моделей, здатних до обробки нечітких вхідних даних.

Завершальним компонентом теоретичної бази стали метрики оцінювання якості моделей: середня квадратична помилка (MSE), середня абсолютна похибка (MAE), корінь середньої квадратичної помилки (RMSE) та коефіцієнт детермінації (R^2). Вони забезпечують кількісну оцінку ефективності прогнозування та дозволяють здійснювати об’єктивне порівняння між моделями.

Таким чином, розділ закладає теоретичне підґрунтя для реалізації, тестування та аналізу моделей у подальшій практичній частині дослідження, що буде представлена у третьому розділі.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Вибір мови програмування та бібліотек

Для реалізації задачі прогнозування часових рядів у межах дипломного дослідження було обрано мову програмування Python. Основними критеріями вибору стали: підтримка сучасних алгоритмів машинного навчання, наявність засобів для роботи з часовими рядами, активна спільнота користувачів, кросплатформеність та інтеграція з науковими обчислювальними інструментами.

Python є однією з найбільш поширених мов у галузі прикладної аналітики даних, що підтверджується її використанням у наукових дослідженнях, промислових застосуваннях та в рамках більшості платформ машинного навчання. Завдяки підтримці процедурного, об'єктно-орієнтованого та функціонального стилів програмування, мова Python дозволяє реалізовувати як експериментальні прототипи, так і повноцінні програмні рішення.

Таким чином, вибір Python як основної мови програмування обумовлений її відповідністю технічним і функціональним вимогам дослідження.

3.1.1 Використані бібліотеки для моделювання

У межах дипломного дослідження для реалізації моделей прогнозування ключових показників ринку електроенергії було використано низку

програмних бібліотек, що забезпечують побудову статистичних, нейронних та нечітких моделей.

TensorFlow — програмна платформа з відкритим кодом для реалізації моделей машинного та глибинного навчання. Застосовується для побудови рекурентних нейронних мереж, зокрема LSTM (Long Short-Term Memory), які ефективно моделюють часові залежності у даних [29].

statsmodels — бібліотека для побудови та оцінки статистичних моделей. У цьому дослідженні вона використовується для реалізації моделі SARIMAX, яка враховує сезонність та вплив зовнішніх регресорів [30].

scikit-fuzzy — модуль для реалізації нечіткої логіки в середовищі Python. Забезпечує побудову систем нечітких висновків, що дозволяє використовувати лінгвістичні правила у прогнозних задачах [31].

anfis — спеціалізована бібліотека для реалізації адаптивних нейро-нечітких систем (ANFIS). Об’єднує можливості нечіткої логіки та нейронних мереж, дозволяючи створювати моделі з високою гнучкістю до нелінійних залежностей у даних [32].

Зазначені бібліотеки було обрано з огляду на їхню наукову визнаність, підтримку складних архітектур, сумісність із іншими інструментами Python та можливість повної інтеграції в середовище програмування DataSpell.

3.1.2 Інструменти для візуалізації, обробки та обчислень

У процесі реалізації модулів обробки та аналізу даних, а також під час візуалізації результатів прогнозування, було використано низку бібліотек, що є загальновизнаними стандартами в галузі наукового програмування на Python.

pandas — потужна бібліотека для роботи зі структурованими даними, що надає зручний інструментарій для маніпуляцій з табличними наборами даних,

агрегації, обробки пропущених значень, групування та попередньої обробки часових рядів [33].

NumPy — базова бібліотека для наукових обчислень, яка забезпечує підтримку багатовимірних масивів і великий набір математичних функцій. Застосовується для реалізації математичних операцій, обчислення статистичних показників та оптимізації швидкодії програмних модулів [34].

matplotlib — універсальна бібліотека для створення графіків і діаграм, яка дозволяє будувати лінійні, гістограмні, точкові та інші типи графіків для візуального представлення як сирих, так і оброблених даних [35].

seaborn — бібліотека візуалізації на основі matplotlib, яка надає високорівневі функції для побудови статистичних графіків, включно з тепловими картами, графіками розсіювання з регресійними лініями та кореляційними матрицями [36].

Використання зазначених інструментів дозволило ефективно підготувати дані до моделювання, провести їх попередній аналіз та наочно представити результати прогнозування.

3.2 Збір даних

Збір даних для прогнозного моделювання ключових показників ринку електроенергії здійснювався шляхом парсингу таблиць із веб-сайту офіційного джерела — ТОВ "Оператор ринку" [3]. Перед початком збору було отримано дозвіл на використання опублікованих даних у межах дослідницької роботи.

З огляду на специфіку веб-сайту, який містить динамічно завантажуванні HTML-елементи, для парсингу було використано бібліотеку Selenium. Вона забезпечує повноцінну емуляцію дій користувача в браузері та

дозволяє обробляти контент, який не доступний для традиційних інструментів парсингу на кшталт BeautifulSoup. Такий підхід виявився необхідним для коректного збору повної таблиці результатів внутрішньодобового ринку електроенергії [37].

Для цілей даного дослідження обрано саме внутрішньодобовий ринок, оскільки він характеризується високим ступенем цінової і об’ємної волатильності. Його структура менш передбачувана, ніж у випадку ринку "на добу наперед" або балансуючого ринку, що робить його більш складним, але й більш інформативним об'єктом для моделювання і тестування ефективності алгоритмів прогнозування.

Результатом збору є таблиця, яка містить показники за кожну годину доби, перелічені нижче.

1. Година — порядковий номер години доби (від 1 до 24).
2. Ціна, грн/МВт.год — середньозважена фактична ціна електроенергії за відповідну годину.
3. Мінімальна ціна, грн/МВт.год — мінімальна запропонована ціна продажу електроенергії серед усіх поданих заявок;
4. Максимальна ціна, грн/МВт.год — максимальна запропонована ціна продажу електроенергії.
5. Остання ціна, грн/МВт.год — ціна, за якою була здійснена остання операція в межах години.
6. Обсяг продажу, МВт.год — фактичний обсяг електроенергії, проданої на ринку.
7. Обсяг купівлі, МВт.год — фактичний обсяг електроенергії, купленої на ринку.
8. Заявлений обсяг продажу, МВт.год — сумарний обсяг електроенергії, який продавці подали в заявках.
9. Заявлений обсяг купівлі, МВт.год — сумарний обсяг електроенергії, який покупці подали в заявках.

Приклад отриманої таблиці наведено на рисунку 3.1 — 3.2 нижче. Ці дані надалі використовуються для побудови часових рядів та подальшої підготовки до моделювання.

Година	Ціна, грн/МВт.год	Мінімальна ціна, грн/МВт.год	Максимальна ціна, грн/МВт.год	Остання ціна, грн/МВт.год	Обсяг продажу, МВт.год	Обсяг купівлі, МВт.год	Заявлений обсяг продажу, МВт.год	Заявлений обсяг купівлі, МВт.год
1	4989.47	4000.00	5050.00	5049.99	106.0	106.0	189.5	122.2
2	3722.70	2500.00	3801.00	3800.00	75.7	75.7	127.5	96.1
3	2993.82	2955.00	3599.00	2965.00	107.5	107.5	161.9	138.5
4	3776.80	3699.00	3810.33	3750.50	106.4	106.4	183.2	116.0
5	3294.39	2965.00	3600.00	2967.07	81.9	81.9	106.8	448.9
6	3773.38	3699.00	3800.00	3799.97	69.3	69.3	128.0	77.4

Рисунок 3.1 — Таблична структура даних з сайту "Оператор ринку"

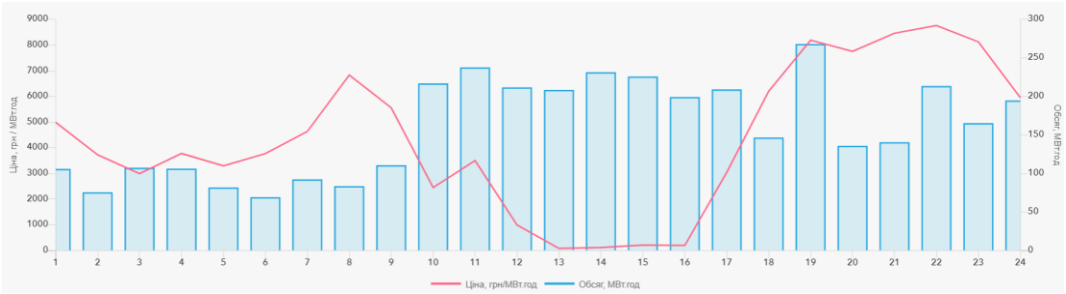


Рисунок 3.2 — Візуалізація ціни та обсягу електроенергії протягом доби на внутрішньодобовому ринку

3.3 Аналіз набору даних

3.3.1 Загальна характеристика даних

Вихідний набір даних містить часові спостереження за внутрішньодобовим ринком електроенергії України, зібрані за період з 2020 по 2025 рік. Дані представлені у вигляді погодинних записів ключових ринкових показників. Кожна колонка таблиці відображає певну економічну або технічну характеристику електроенергетичного ринку.

На рисунку 3.3 нижче зображено динаміку восьми основних показників.

1. Price, UAH/MWh: помітне зростання середнього рівня цін, починаючи з 2023 року, а також висока волатильність у 2024–2025 рр.
2. Minimum price, UAH/MWh та Maximum price, UAH/MWh: демонструють значне розширення діапазону коливань цін, особливо після початку повномасштабної війни в Україні.
3. Last price, UAH/MWh: має схожу структуру до середньої ціни, підтверджуючи загальні тенденції ринку.
4. Sales volume, MWh та Purchase volume, MWh: загальний тренд до зниження з 2021 року, ймовірно зумовлений змінами у структурі генерації та споживання.
5. Declared sales volume, MWh: значна кількість пікових значень, які можуть бути зумовлені поведінкою окремих учасників ринку або технічними збоями у поданні заявок.
6. Declared purchase volume, MWh: варіативність також зберігається, хоча тенденція до зменшення більш виражена.

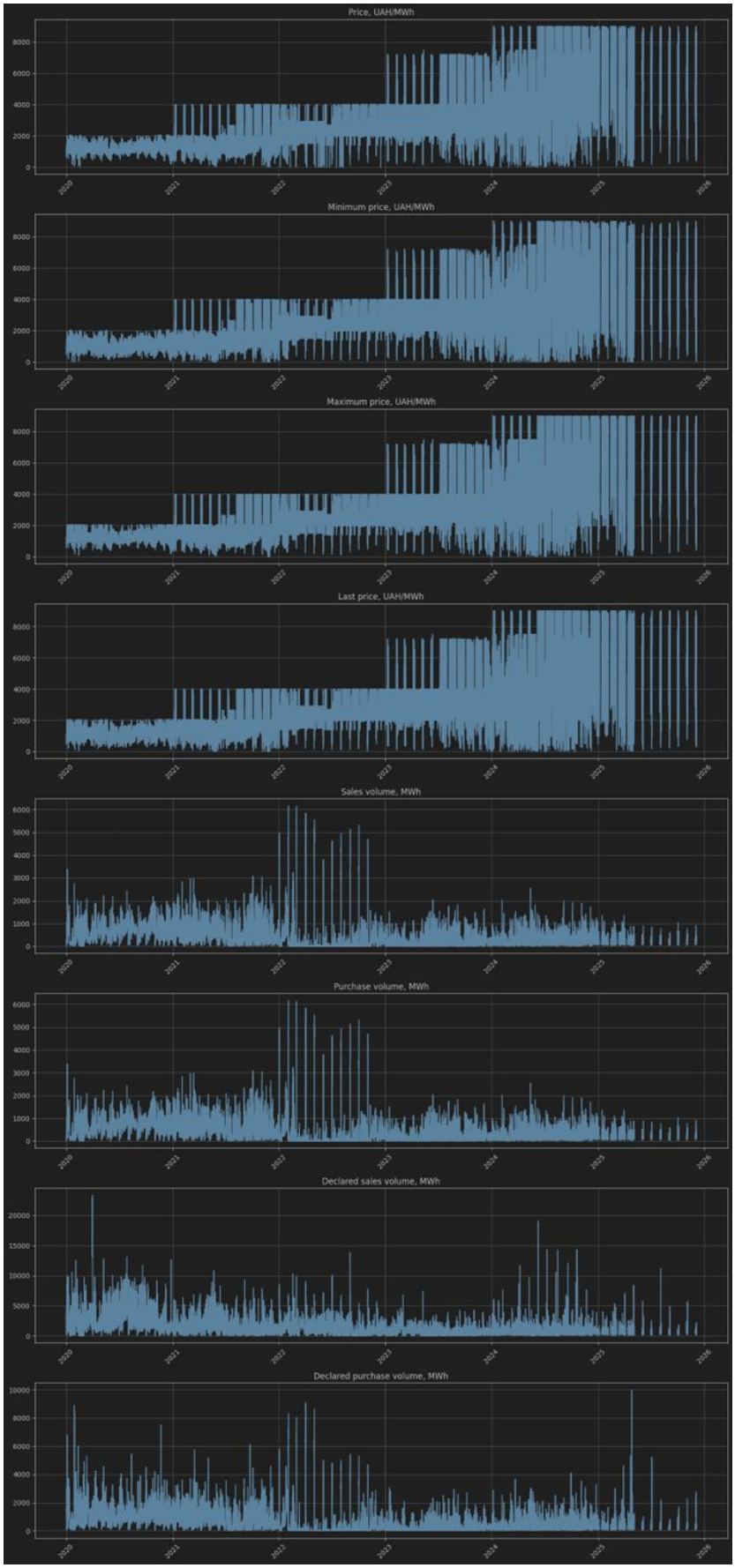


Рисунок 3.3 — Динаміка основних погодинних показників внутрішньодобового ринку електроенергії за 2020 - 2025 роки

Загалом, дані мають високий рівень нестабільності, наявність пікових сплесків, сезонних коливань та потенційних структурних зламів унаслідок зовнішніх подій (наприклад, початок повномасштабного вторгнення в Україну).

З огляду на відсутність явної залежності між окремими змінними в контексті прогнозування (наприклад, між ціною та обсягами), було прийнято рішення розділити набір даних на вісім окремих часових рядів. Кожен з них моделюється та прогнозується незалежно. Це рішення ґрунтується на тому, що всі значення в межах одного запису (години) формуються одночасно та не мають однозначної причинно-наслідкової структури між собою. Такий підхід дозволяє уникнути хибної регресії між одночасними змінними та забезпечити більш коректне навчання моделей прогнозування.

3.3.2 Виявлення аномалій і пропусків

На основі аналізу часових рядів, отриманих із внутрішньодобового ринку, було виявлено наявність пропусків у даних, які спостерігаються на всіх без винятку ключових показниках. Пропущені значення (NaN) позначено червоним кольором на рисунку 3.4 нижче. Особливо велика кількість пропусків зафіксована у 2023 - 2025 роках, що, ймовірно, пов’язано з війною, перебоями в роботі інформаційної системи або неповним заповненням даних з боку учасників ринку. Окремі пропуски також трапляються в попередні роки.

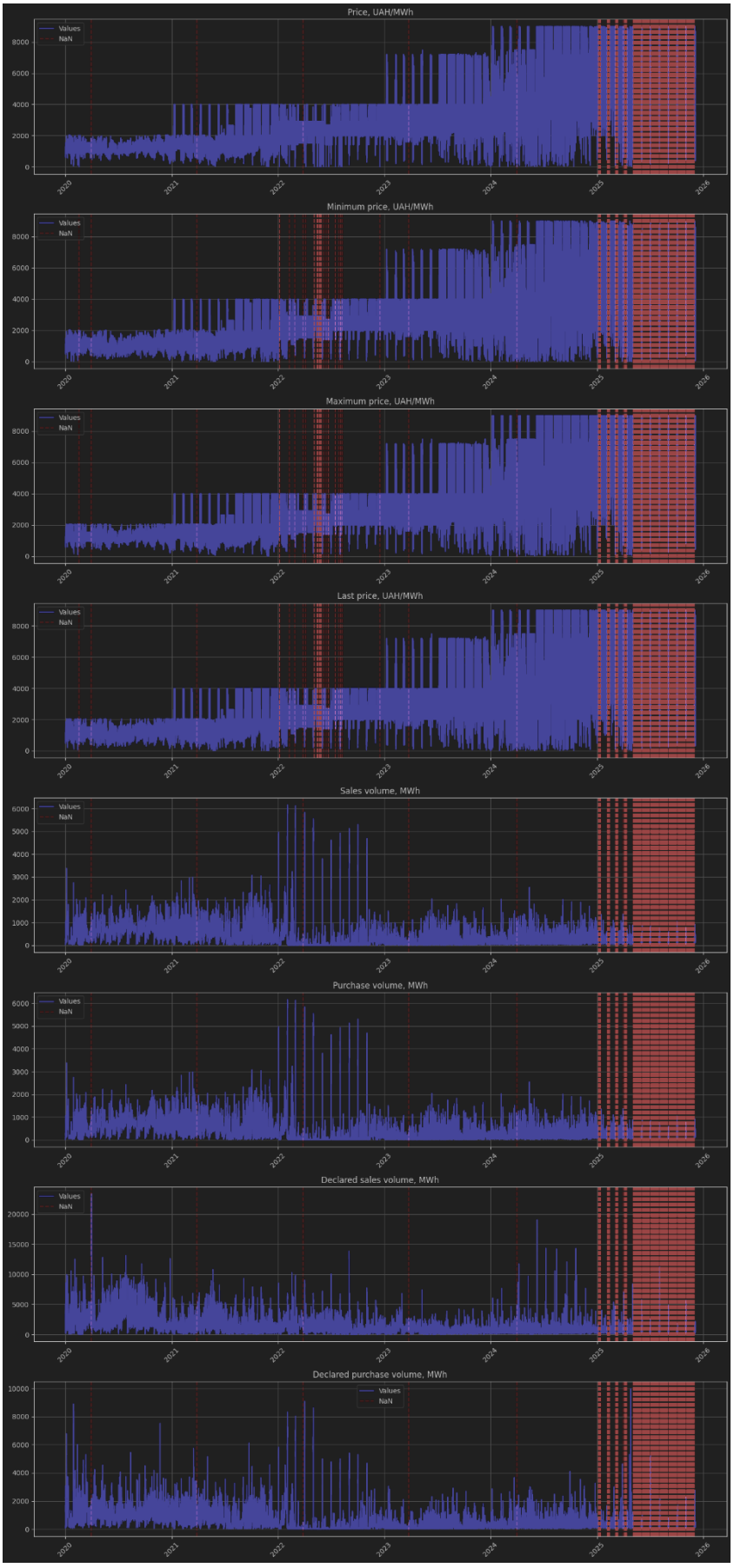


Рисунок 3.4 — Візуалізація пропущених значень (NaN) у часових рядах внутрішньодобового ринку електроенергії за 2020 - 2025 роки

Для обробки пропущених значень було використано стандартну функцію інтерполяції з бібліотеки pandas, яка дозволяє обчислити відсутні значення на основі значень до і після пропуску. Такий підхід забезпечує безперервність часових рядів та запобігає втраті інформації при побудові прогнозних моделей.

Окрім пропусків, було виявлено аномальні коливання значень для ряду показників. Зокрема, на графіках видно різке зростання цінових показників у період 2022–2024 років. Це зростання співпадає з початком повномасштабного вторгнення Росії в Україну (лютий 2022 року), що призвело до значних змін у структурі попиту і пропозиції, дестабілізації генерації, порушення логістичних ланцюгів та зміни режиму ринку.

Інші аномалії, включаючи локальні пікові значення або різке зниження активності, також зафіксовані у періоди святкових та вихідних днів (наприклад, Новий рік, Великдень тощо). У ці дні знижується обсяг споживання, а отже й активність учасників на внутрішньодобовому ринку. Це призводить до нерівномірного заповнення таблиці заявок і різких коливань цін.

Таким чином, наявність пропусків і аномалій у даних є важливим фактором, який необхідно враховувати під час підготовки даних до моделювання. Попередня обробка та коректна обробка аномалій суттєво впливають на точність прогнозних моделей та якість висновків дослідження.

3.3.3 Дослідження стаціонарності, автокореляції та декомпозиція даних

На цьому етапі було проведено аналіз часових рядів з метою визначення їхньої стаціонарності, автокореляційної структури та сезонних компонент. Дослідження базувалося на застосуванні класичних статистичних тестів і методів, рекомендованих у сучасній літературі з аналізу часових рядів [38-41].

Для перевірки стаціонарності використовувалися два методи (див. рис. 3.5): ADF (Augmented Dickey-Fuller) та KPSS (Kwiatkowski-Phillips-Schmidt-Shin).

	ADF p-value	KPSS p-value	ADF Stationary	KPSS Stationary
Price, UAH/MWh	2.125490e-09	0.01	True	False
Minimum price, UAH/MWh	4.058344e-11	0.01	True	False
Maximum price, UAH/MWh	3.529609e-09	0.01	True	False
Last price, UAH/MWh	1.202827e-09	0.01	True	False
Sales volume, MWh	2.583849e-30	0.01	True	False
Purchase volume, MWh	2.583849e-30	0.01	True	False
Declared sales volume, MWh	3.007241e-25	0.01	True	False
Declared purchase volume, MWh	4.392347e-29	0.01	True	False

Рисунок 3.5 — Таблиця результатів тестів ADF та KPSS для перевірки стаціонарності основних часових рядів

Тест ADF перевіряє нульову гіпотезу про наявність одиничного кореня, що вказує на нестаціонарність ряду. Результати тесту для всіх часових рядів показали p-значення менше 0.05, що дозволяє відхилити нульову гіпотезу і стверджувати про наявність стаціонарності у сенсі ADF [38].

Тест KPSS, навпаки, перевіряє гіпотезу про стаціонарність. Результати цього тесту демонструють p-значення менше 0.05, що свідчить про відхилення гіпотези стаціонарності [39].

Наявність суперечливих висновків між ADF і KPSS є типовою ситуацією при аналізі економічних часових рядів і вказує на часткову або слабку стаціонарність. Це вимагає обережного підходу до диференціювання даних та застосування моделей, стійких до таких властивостей.

Для подальшого аналізу було побудовано графіки ACF (автокореляційної функції) та PACF (часткової автокореляційної функції) (див. рис. 3.6 - 3.7).

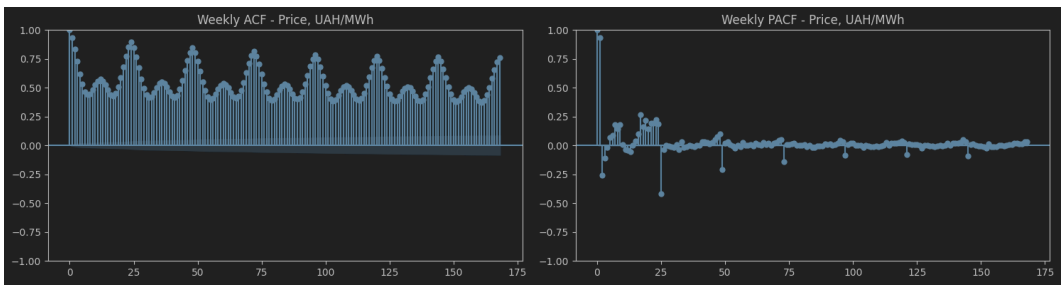


Рисунок 3.6 — ACF та PACF графіки для Price, UAH/MWh

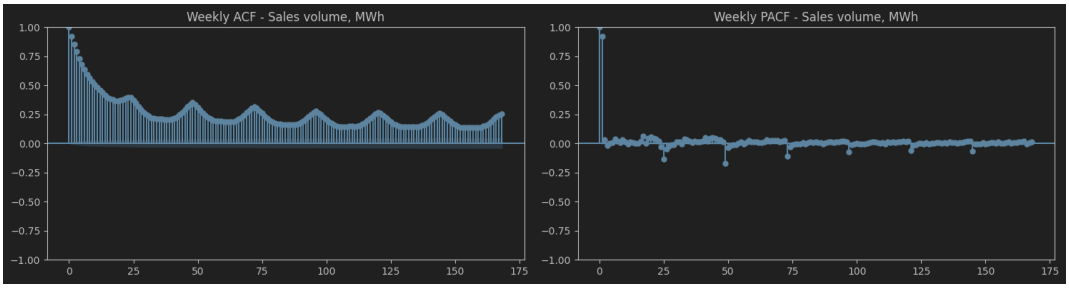


Рисунок 3.7 — ACF та PACF графіки для Sales volume, MWh

На графіку ACF для цін спостерігається чітко виражена сезонна структура з періодом 24 години, що відповідає добовій циклічності електроспоживання.

Графік PACF вказує на сильні залежності між сусідніми значеннями та поступове згасання кореляції на довших лагах.

Ці графіки відіграють ключову роль у виборі порядків p та q при побудові моделей ARIMA, SARIMA або SARIMAX [40].

Окрім цього, було проведено дослідження сезонності за допомогою добових шаблонів, побудованих шляхом агрегування даних за годинами доби (див. рис. 3.8 - 3.9). ANOVA-аналіз продемонстрував p-значення, що дорівнює 0.0, тобто підтверджено наявність суттєвих відмінностей між середніми значеннями в різні години. Для цін спостерігається типовий шаблон з підвищенням у вечірні години та зниженням уночі, що відповідає реальній структурі споживання.

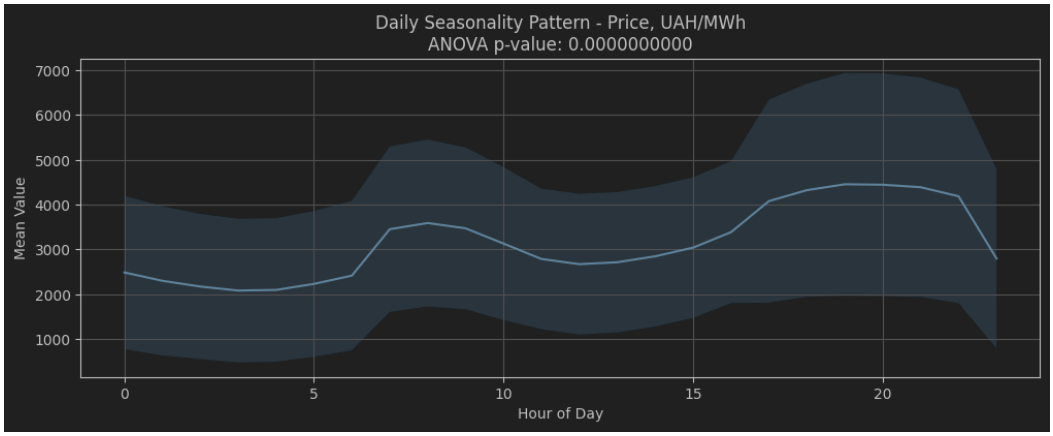


Рисунок 3.8 — Добовий шаблон сезонності для середньої ціни (Price, UAH/MWh)

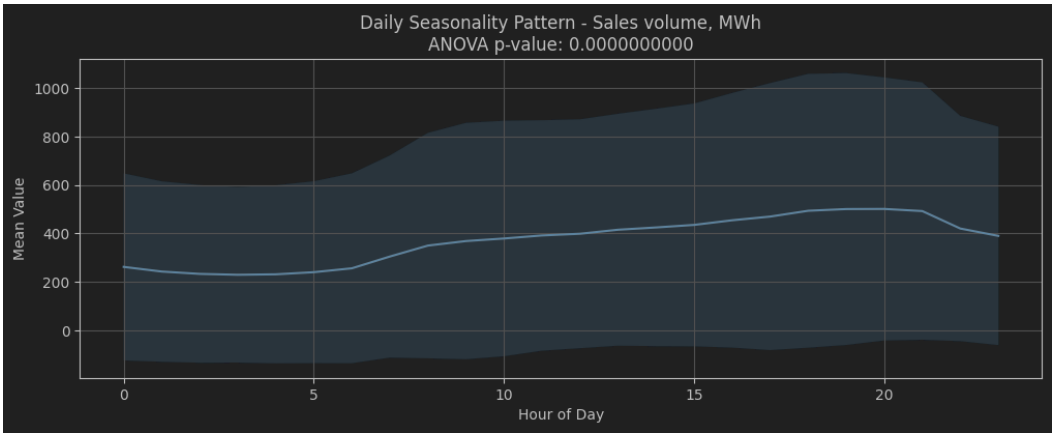


Рисунок 3.9 — Добовий шаблон сезонності для обсягу продажу (Sales volume, MWh)

Для більш глибокого аналізу виконано STL-декомпозицію (Seasonal-Trend decomposition using Loess) часових рядів (див. рис. 3.10 – 3.11).

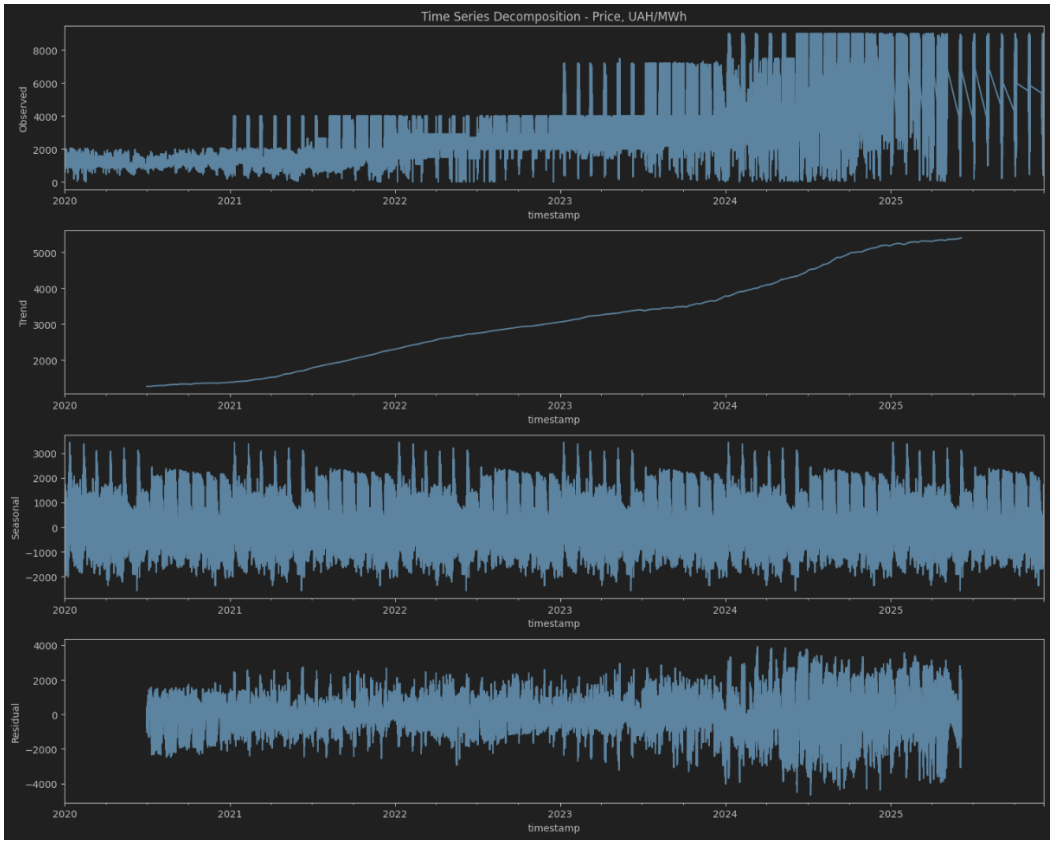


Рисунок 3.10 — STL-декомпозиція часового ряду середньої ціни електроенергії (Price, UAH/MWh)

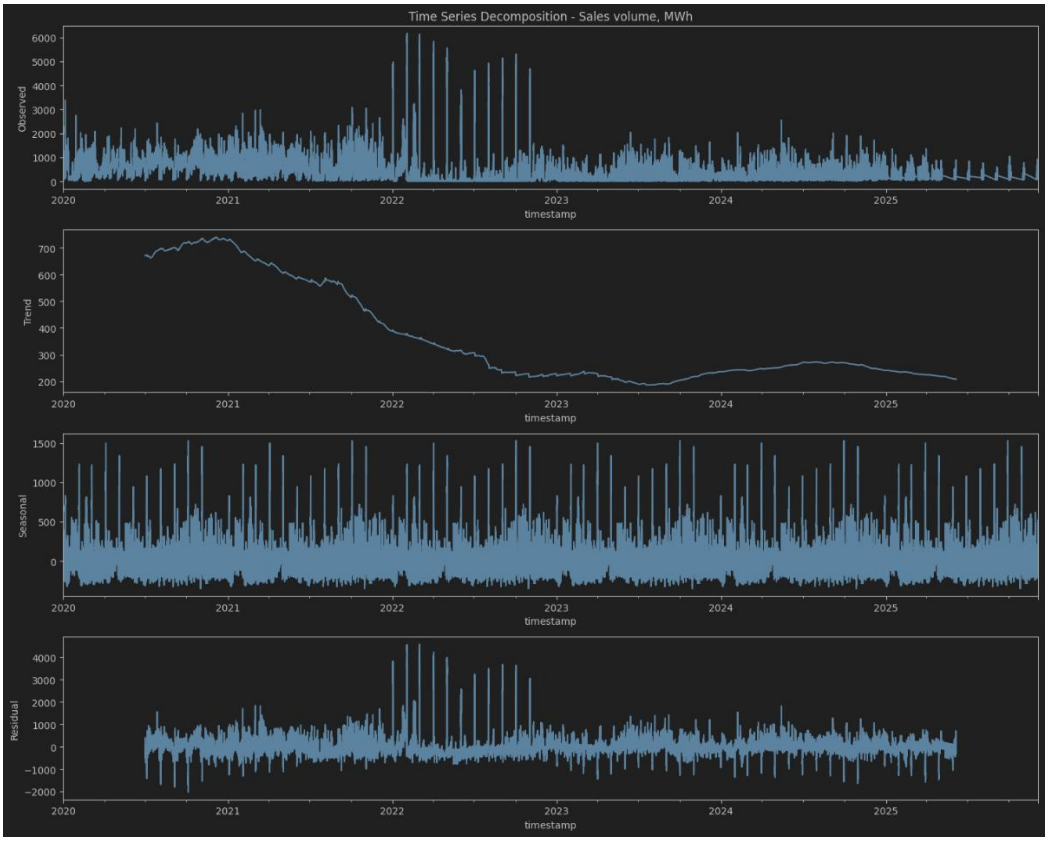


Рисунок 3.11 — STL-декомпозиція часового ряду обсягу продажу електроенергії (Sales volume, MWh)

Компонента тренду для цін демонструє поступове зростання починаючи з 2022 року.

Для обсягів продажів тренд має спадну тенденцію, що може бути пов’язано зі скороченням загального споживання або зміною структури ринку.

Сезонна компонента зберігає регулярність у вигляді добових циклів, а залишкова компонента характеризується високою варіативністю, що вказує на вплив випадкових зовнішніх факторів.

Загалом, результати даного підрозділу вказують на наявність важливих особливостей у структурі часових рядів, які необхідно враховувати при побудові моделей прогнозування: часткова стаціонарність, автокореляційна структура, добова сезонність та довгострокові тренди.

3.4 Створення моделей прогнозування та їх аналіз

Для реалізації задач прогнозування часових рядів ключових показників на ринку електроенергії було обрано низку моделей, що охоплюють як класичні статистичні підходи, так і сучасні методи машинного навчання. Такий підхід дозволяє оцінити ефективність різних типів моделей при роботі з високоволатильними, сезонними та частково нестаціонарними даними внутрішньодобового ринку.

У межах дослідження розглядаються класи моделей, наведені нижче.

1. Статистичні моделі часових рядів — зокрема, SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors). Ця модель дозволяє враховувати сезонні компоненти та потенційні впливи зовнішніх змінних, при цьому зберігаючи інтерпретовану структуру і стійкість до шумів у даних.
2. Нейронні мережі типу LSTM (Long Short-Term Memory) — глибокі рекурентні архітектури, здатні ефективно моделювати довготривалі залежності в часових рядах. Вони є особливо корисними в умовах складної динаміки та великої кількості аномалій.
3. Гібридні моделі ANFIS (Adaptive Neuro-Fuzzy Inference System) — поєднують в собі можливості нечіткої логіки та нейронних мереж. Такі моделі здатні враховувати лінгвістичні правила та адаптивно змінювати параметри на основі навчальних даних.

Окрему увагу приділено варіантам горизонту прогнозування. Для кожного з часових рядів моделі будуть налаштовані на три варіанти прогнозу.

1. Прогноз на один день (24 години).
2. Прогноз на тиждень (168 годин).
3. Прогноз на місяць (720 годин).

Такий підхід дозволяє не лише оцінити точність моделей на різних часових горизонтах, а й виявити залежність ефективності прогнозування від довжини періоду. Усі обрані моделі будуть реалізовані у відповідних бібліотеках Python з подальшою перевіркою якості прогнозів та їх порівнянням за обраними метриками.

Детальний опис кожної з моделей, принципи їхньої роботи, гіперпараметри та архітектура будуть розглянуті в наступних підрозділах цього розділу

3.4.1 Навчання та результати ANFIS

Для реалізації прогнозування часових рядів за допомогою адаптивної нейро-нечіткої моделі ANFIS було використано бібліотеку anfis, яка поєднує принципи нечіткої логіки (FIS — fuzzy inference system) та нейронних мереж. У процесі попереднього налаштування гіперпараметрів було здійснено повномасштабний перебір наступних значень (Grid Search).

1. Кількість нечітких правил (n_rules): тестовано від 20 до 250.
2. Кількість часових кроків (n_steps): перевірено варіанти 24 (добовий патерн), 24*7 (тижневий) та 24*30 (місячний).
3. Кількість епох (epochs): перевірено діапазон від 10 до 100.

Оптимальними гіперпараметрами для даного типу задачі виявилися: epochs = 50, n_rules = 200, n_steps = 168. Така конфігурація забезпечила збалансовану точність при прийнятному часі навчання.

ANFIS тренувалася окремо для кожного з восьми часових рядів і для трьох варіантів горизонтів прогнозування: добовий (24 години), тижневий (168 годин), місячний (720 годин). Навчання здійснювалося на нормалізованих даних із подальшим інверсним перетворенням для оцінки результатів.

Графіки порівняння фактичних і прогнозних значень для обраних змінних (ціна, обсяг продажів) демонструють високу наближеність прогнозу до реальних даних на добовому та тижневому горизонтах (див. рис. 3.12 - 3.13). На місячному горизонті точність помітно знижується, що є очікуваним через кумулятивний ефект помилок та невизначеності.

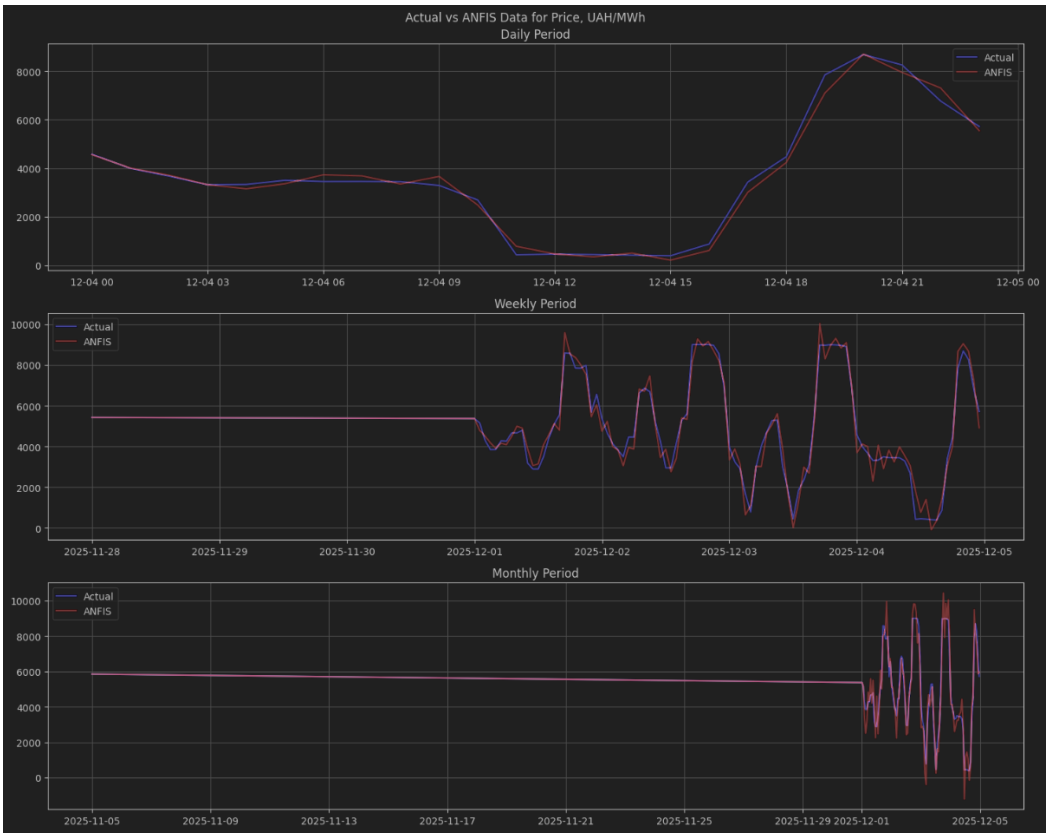


Рисунок 3.12 — Фактичні та прогнозні значення ціни (UAH/MWh), отримані за допомогою ANFIS, для добового, тижневого та місячного горизонтів

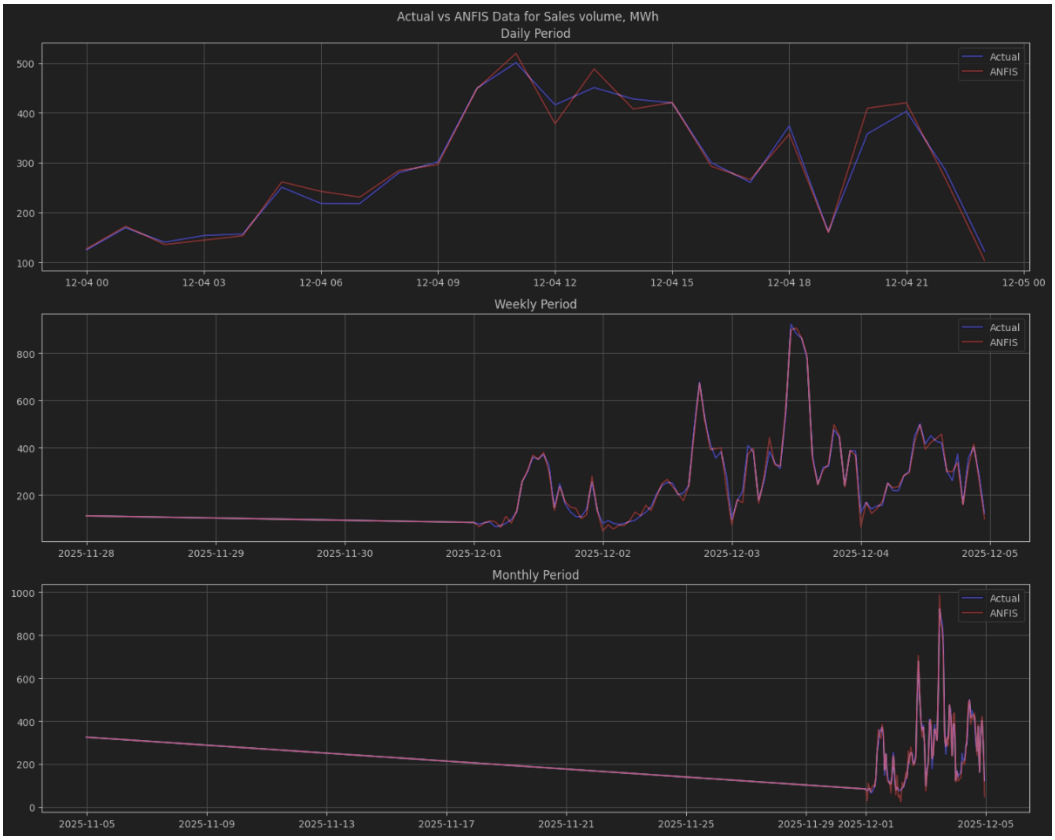


Рисунок 3.13 — Фактичні та прогнозні значення обсягу продажу (MWh), отримані за допомогою ANFIS, для добового, тижневого та місячного горизонтів

Оцінювання якості здійснювалося за такими метриками: MSE (середньоквадратична помилка), RMSE (корінь MSE), MAE (середня абсолютна похибка), R^2 (коефіцієнт детермінації) (див. рис. 3.14).

ANFIS	MSE	RMSE	MAE	R2
Price, UAH/MWh - daily	76475.4109	276.5419	209.1821	0.987
Price, UAH/MWh - weekly	151421.7106	389.1294	238.8468	0.9566
Price, UAH/MWh - monthly	77875.1	279.0611	84.1952	0.9113
Minimum price, UAH/MWh - daily	139604.4536	373.6368	250.802	0.9723
Minimum price, UAH/MWh - weekly	156060.2415	395.0446	242.0861	0.9598
Minimum price, UAH/MWh - monthly	64739.2599	254.4391	73.2886	0.9311
Maximum price, UAH/MWh - daily	128539.0448	358.5234	266.3431	0.9796
Maximum price, UAH/MWh - weekly	238617.7097	488.4851	295.6158	0.9286
Maximum price, UAH/MWh - monthly	75558.1553	274.8784	80.0485	0.9083
Last price, UAH/MWh - daily	82312.6037	286.9017	191.4109	0.9864
Last price, UAH/MWh - weekly	183763.9059	428.6769	257.0019	0.9499
Last price, UAH/MWh - monthly	81045.5809	284.6851	82.8851	0.911
Sales volume, MWh - daily	356.9469	18.893	13.8023	0.974
Sales volume, MWh - weekly	273.5986	16.5408	9.7928	0.9897
Sales volume, MWh - monthly	112.121	10.5887	3.1765	0.9878
Purchase volume, MWh - daily	393.8065	19.8446	13.8731	0.9713
Purchase volume, MWh - weekly	267.9757	16.37	9.8469	0.99
Purchase volume, MWh - monthly	153.7669	12.4003	3.4532	0.9833
Declared sales volume, MWh - daily	4490.6819	67.0126	47.8242	0.9855
Declared sales volume, MWh - weekly	2755.5789	52.4936	30.4923	0.9661
Declared sales volume, MWh - monthly	1133.5864	33.6688	9.305	0.9421
Declared purchase volume, MWh - daily	937.3606	30.6163	22.4284	0.9436
Declared purchase volume, MWh - weekly	520.9687	22.8247	13.5735	0.995
Declared purchase volume, MWh - monthly	187.3756	13.6885	3.9483	0.9933

Рисунок 3.14 — Метрики оцінювання точності прогнозу моделей ANFIS для кожного з часових рядів і горизонтів прогнозування

Згідно з отриманими результатами:

R^2 — значення для більшості серій перевищує 0.95, що свідчить про дуже високу точність моделі. Зокрема:

Ціна (Price, UAH/MWh): $R^2 = 0.987$ (добовий), 0.956 (тижневий), 0.911 (місячний).

Обсяг продажів (Sales volume, MWh): $R^2 = 0.997, 0.993, 0.978$ відповідно.

MAE (mean absolute error) для цінових показників знаходиться на рівні 80-250 грн/МВт.год, що є прийнятним з огляду на масштаби коливань.

RMSE також зростає зі збільшенням горизонту, проте залишається в межах адекватної точності.

Модель продемонструвала стабільну роботу та високу якість при прогнозуванні основних економічних індикаторів у коротко- та середньостроковій перспективі. Найкращі результати досягнуто на добових і тижневих періодах для більшості показників. Прогноз на місячному горизонті

має тенденцію до згладжування, що пов’язано з обмеженою глибиною пам’яті ANFIS-архітектури та накопиченням похибки.

Отримані результати свідчать про ефективність використання ANFIS для задач прогнозування на енергетичному ринку, особливо при локальних прогнозах або задачах з обмеженою тривалістю горизонту.

3.4.2 Навчання та результати LSTM

У межах побудови моделей глибокого навчання для прогнозування часових рядів ключових показників ринку електроенергії була реалізована архітектура LSTM (Long Short-Term Memory). Ця архітектура є однією з найефективніших для обробки послідовних даних завдяки здатності зберігати довготривалу залежність та стійкості до проблеми зникнення градієнта.

Перед початком навчання було проведено Grid Search з підбором перелічених нижче гіперпараметрів.

1. sequence_length: [24×7, 24×14].
2. n_lstm_units: [50, 100, 150].
3. n_lstm_layers: [1, 2, 3, 4].
4. dropout_rate: [0, 0.1, 0.2].
5. learning_rate: [0.01, 0.001, 0.0001].

Найкращі результати були досягнуті при параметрах: sequence_length = 168 (один тиждень), n_lstm_units = 100, n_lstm_layers = 2, dropout_rate = 0, learning_rate = 0.001.

Модель LSTM була навчена окремо для кожного з восьми часових рядів на трьох горизонтах прогнозування: добовий (24 години), тижневий (168 годин), місячний (720 годин). На візуалізаціях (див. рис. 3.15 - 3.16) видно, що модель добре повторює основні закономірності у динаміці змін, зберігаючи чітку відповідність між прогнозними та фактичними значеннями. Добові та

тижневі прогнози демонструють високу якість та синхронність у пікоподібних ділянках. Прогнози на місячному горизонті також відображають базові тренди, хоча деякі піки згладжуються.

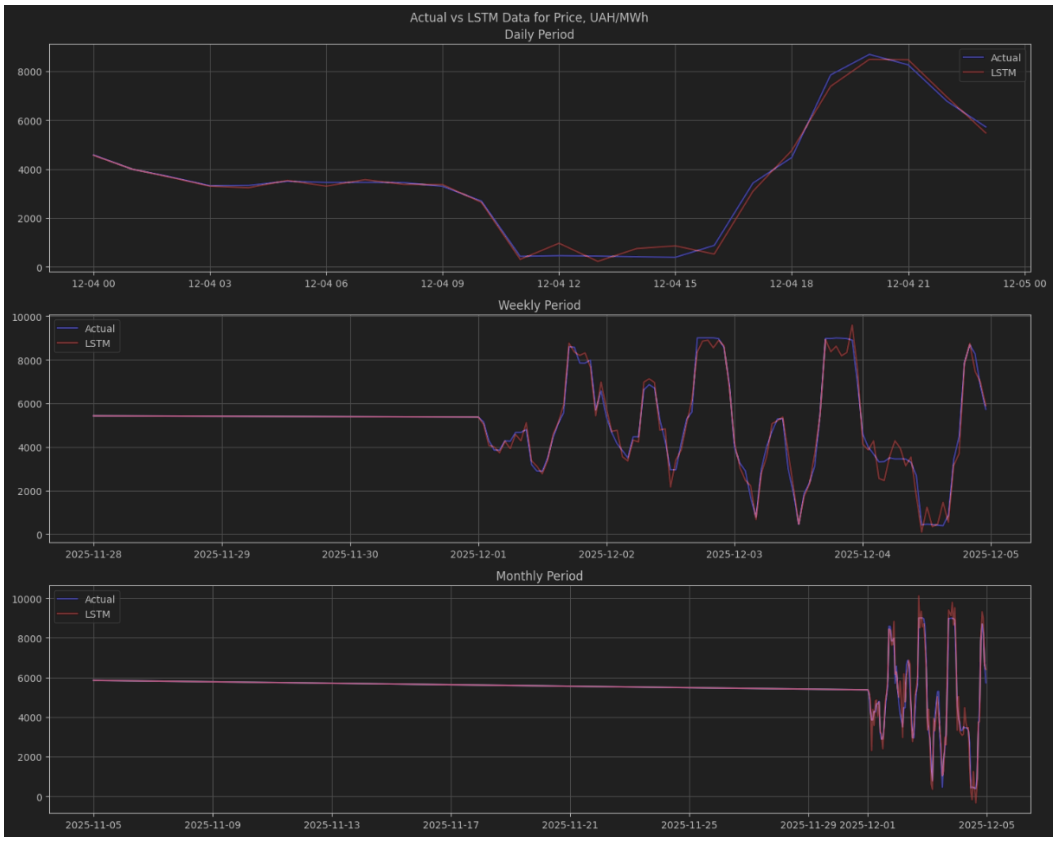


Рисунок 3.15 — Фактичні та прогнозні значення ціни (UAH/MWh), отримані за допомогою LSTM, для добового, тижневого та місячного горизонтів

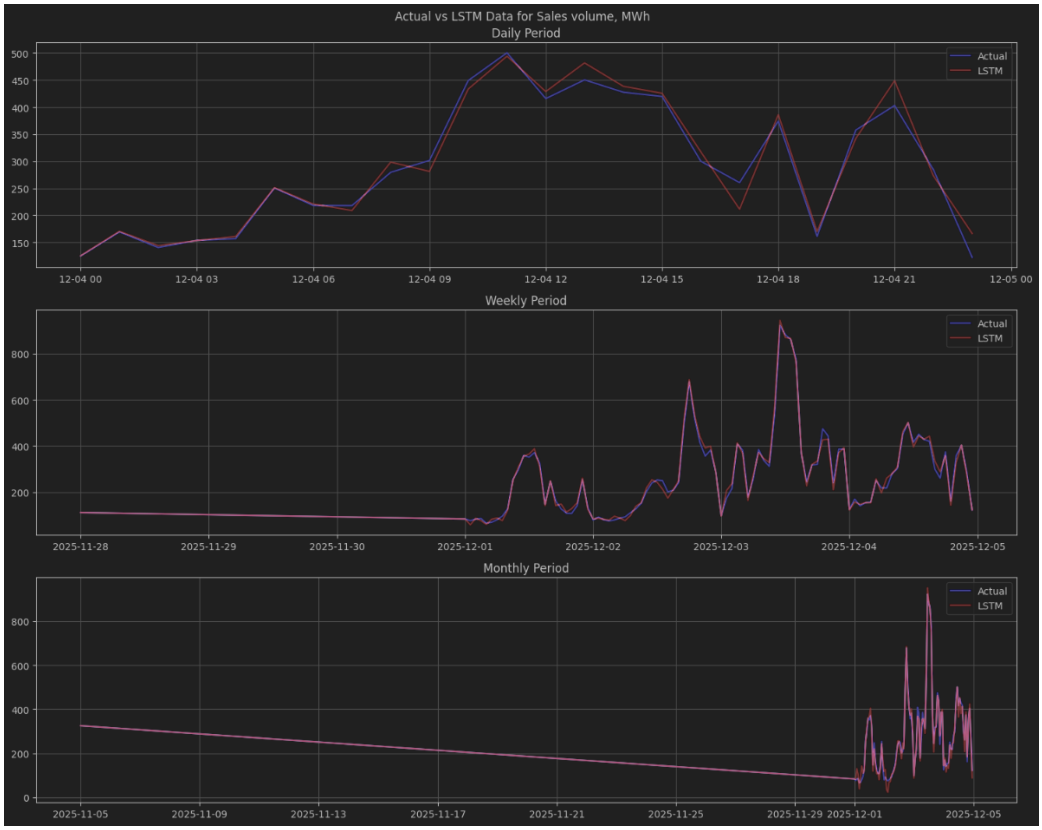


Рисунок 3.16 — Фактичні та прогнозні значення обсягу продажу (MWh), отримані за допомогою LSTM, для добового, тижневого та місячного горизонтів

Оцінювання якості здійснювалося за метриками: MSE (середньоквадратична помилка), RMSE (корінь MSE), MAE (середня абсолютна похибка), R^2 (коефіцієнт детермінації) (див. рис. 3.17). Значення R^2 перевищують 0.98 у більшості прогнозів, що свідчить про високу ефективність моделі.

LSTM	MSE	RMSE	MAE	R2
Price, UAH/MWh - daily	59478.2433	243.8816	190.7141	0.9899
Price, UAH/MWh - weekly	101042.1319	317.8713	192.2553	0.971
Price, UAH/MWh - monthly	56723.0023	238.1659	70.7551	0.9354
Minimum price, UAH/MWh - daily	62473.8556	249.9477	166.6242	0.9876
Minimum price, UAH/MWh - weekly	108724.3592	329.7338	194.872	0.972
Minimum price, UAH/MWh - monthly	39855.7415	199.639	59.5082	0.9576
Maximum price, UAH/MWh - daily	61449.6439	247.8904	175.5946	0.9903
Maximum price, UAH/MWh - weekly	120025.1069	346.4464	208.1277	0.9641
Maximum price, UAH/MWh - monthly	59669.0188	244.2724	69.8671	0.9276
Last price, UAH/MWh - daily	63824.0222	252.6342	179.8226	0.9895
Last price, UAH/MWh - weekly	114778.5281	338.7898	194.8556	0.9687
Last price, UAH/MWh - monthly	41740.1772	204.3041	55.8585	0.9541
Sales volume, MWh - daily	410.0568	20.2499	14.5517	0.9702
Sales volume, MWh - weekly	162.6298	12.7526	7.5893	0.9939
Sales volume, MWh - monthly	83.4912	9.1374	2.7399	0.9909
Purchase volume, MWh - daily	690.9482	26.2859	18.6916	0.9497
Purchase volume, MWh - weekly	167.1954	12.9304	7.6431	0.9937
Purchase volume, MWh - monthly	87.5639	9.3576	2.7187	0.9905
Declared sales volume, MWh - daily	2237.9791	47.3073	34.3353	0.9928
Declared sales volume, MWh - weekly	1614.1652	40.1767	23.3982	0.9802
Declared sales volume, MWh - monthly	688.6796	26.2427	7.3234	0.9648
Declared purchase volume, MWh - daily	525.0472	22.9139	16.0595	0.9684
Declared purchase volume, MWh - weekly	349.508	18.6951	10.5481	0.9967
Declared purchase volume, MWh - monthly	88.4196	9.4032	2.7756	0.9968

Рисунок 3.17 — Метрики оцінювання точності прогнозу моделей LSTM для кожного з часових рядів і горизонтів прогнозування

1. Ціна (Price, UAH/MWh): R^2 = 0.9899 (добовий), 0.9710 (тижневий), 0.9367 (місячний).
2. Обсяг продажів (Sales volume, MWh): R^2 = 0.9970-0.9988.
3. MAE для цінових змінних коливається в межах від 55 до 192 грн/МВт.год залежно від горизонту прогнозу.

У порівнянні з іншими моделями, LSTM показує високу стійкість до складної сезонної структури, а також перевагу при довгостроковому прогнозуванні. Отримані результати підтверджують ефективність застосування LSTM у задачах прогнозування складних економічних часових рядів.

3.4.3 Навчання та результати SARIMAX

У межах дослідження статистичного підходу до прогнозування часових рядів було реалізовано модель SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors). Цей підхід дає змогу враховувати як тренд, так і сезонні компоненти, що особливо актуально для моделювання економічних показників з циклічною природою.

Для побудови моделей були використані параметри (p, d, q) × (P, D, Q, s), що забезпечують як короткотермінову, так і сезонну динаміку. На основі попереднього аналізу автокореляцій, сезонності та стаціонарності було обрано такі параметри: (24, 0, 24) × (24, 2, 24, 168). Ця комбінація дозволила досягти компромісу між адекватністю моделі та її складністю.

Візуалізація прогнозів (див. рис. 3.18 - 3.19) показує, що модель SARIMAX здатна відтворювати загальні коливання і сезонні зміни, особливо на добовому та тижневому горизонтах. На графіках прогнозів для ціни та обсягу продажів видно, що SARIMAX моделює циклічність з деякими відхиленнями на екстремальних значеннях. Прогнози на місячному горизонті характеризуються більшою нестабільністю та амплітудними коливаннями, що знижує точність моделі в довгостроковій перспективі.

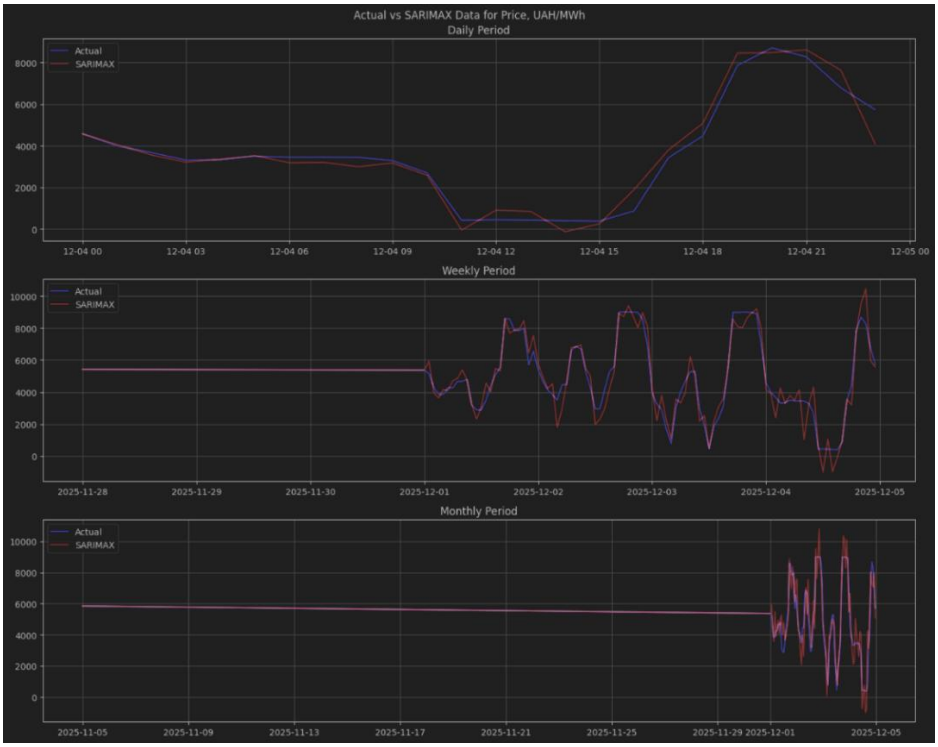


Рисунок 3.18 — Фактичні та прогнозні значення ціни (UAH/MWh), отримані за допомогою SARIMAX, для добового, тижневого та місячного горизонтів

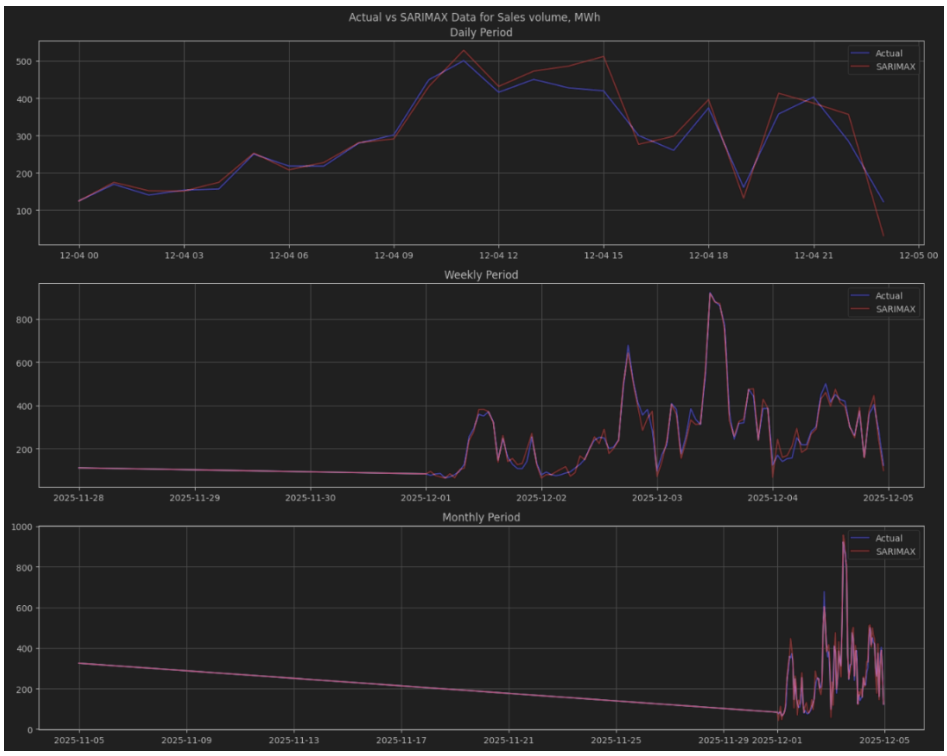


Рисунок 3.19 — Фактичні та прогнозні значення обсягу продажу (MWh), отримані за допомогою SARIMAX, для добового, тижневого та місячного горизонтів

Оцінювання моделі проводилося за стандартними метриками: MSE, RMSE, MAE та R^2. Модель продемонструвала такі результати:

SARIMAX	MSE	RMSE	MAE	R2
Price, UAH/MWh - daily	283318.512	532.2767	383.7267	0.9519
Price, UAH/MWh - weekly	323233.9457	568.5367	324.3918	0.9074
Price, UAH/MWh - monthly	153401.3232	391.6648	117.6648	0.8252
Minimum price, UAH/MWh - daily	159459.8556	399.3242	296.742	0.9683
Minimum price, UAH/MWh - weekly	221149.3457	470.2652	274.2424	0.9431
Minimum price, UAH/MWh - monthly	94330.8217	307.1332	93.0423	0.8996
Maximum price, UAH/MWh - daily	123190.7421	350.9854	274.4554	0.9805
Maximum price, UAH/MWh - weekly	266213.2163	515.9585	297.1086	0.9204
Maximum price, UAH/MWh - monthly	142665.9762	377.7115	113.4753	0.8268
Last price, UAH/MWh - daily	264505.3754	514.3009	381.3035	0.9563
Last price, UAH/MWh - weekly	299998.3552	547.7211	318.5416	0.9182
Last price, UAH/MWh - monthly	152562.755	390.5928	114.8453	0.8324
Sales volume, MWh - daily	1442.4408	37.9795	27.2269	0.895
Sales volume, MWh - weekly	434.7344	20.8503	12.386	0.9837
Sales volume, MWh - monthly	184.0264	13.5656	3.8913	0.98
Purchase volume, MWh - daily	701.0073	26.4765	18.1312	0.949
Purchase volume, MWh - weekly	640.8043	25.3141	15.2494	0.976
Purchase volume, MWh - monthly	190.9591	13.8188	4.0444	0.9792
Declared sales volume, MWh - daily	7964.9823	89.2467	61.3978	0.9743
Declared sales volume, MWh - weekly	6038.5213	77.7079	47.0872	0.9258
Declared sales volume, MWh - monthly	1452.4439	38.1109	11.1977	0.9258
Declared purchase volume, MWh - daily	1522.2159	39.0156	28.163	0.9083
Declared purchase volume, MWh - weekly	1046.6964	32.3527	19.0916	0.99
Declared purchase volume, MWh - monthly	291.6196	17.0769	4.997	0.9895

Рисунок 3.20 — Метрики оцінювання точності прогнозу моделей SARIMAX для кожного з часових рядів і горизонтів прогнозування

Ціна (Price, UAH/MWh): R^2 = 0.9519 (добовий), 0.9074 (тижневий), 0.8523 (місячний).

Обсяг продажів (Sales volume, MWh): R^2 = 0.9804–0.9904.

MAE у цінових показниках перевищує 300 грн/МВт.год для короткострокових прогнозів, а для обсягів залишається в межах 12-27 МВт.год.

Порівняно з нейромережевими моделями LSTM та ANFIS, SARIMAX демонструє помірно нижчу точність, проте зберігає стабільність у коротких інтервалах, особливо для лінійних залежностей. Висновки підтверджують, що SARIMAX може бути ефективним доповненням до інтелектуальних моделей, особливо коли йдеться про інтерпретованість результатів і вплив параметрів сезонності.

3.5 Висновки до розділу 3

У межах третього розділу було реалізовано повний цикл обробки, аналізу та моделювання часових рядів ключових показників ринку електроенергії України. Розпочато з обґрунтування вибору мови програмування та інструментів, серед яких Python показав високу ефективність завдяки великій кількості бібліотек для машинного навчання, статистичного аналізу та візуалізації даних.

Дані було отримано з офіційного сайту Оператора ринку за допомогою бібліотеки Selenium, що дозволило врахувати динамічну природу вебсторінки. Набір охоплював вісім ключових змінних, кожна з яких була досліджена окремо з урахуванням структурних особливостей, аномалій, сезонності, трендів і пропусків. Пропущені значення оброблено методами інтерполяції, а стаціонарність перевірено тестами ADF і KPSS.

У процесі дослідження було реалізовано та протестовано три моделі: LSTM, ANFIS і SARIMAX. Усі моделі навчалися на трьох горизонтах прогнозування — добовому, тижневому та місячному. Найкращі результати з точки зору якості прогнозування ($R^2 > 0.98$) показала модель LSTM, особливо на короткострокових і середньострокових горизонтах. Модель ANFIS також продемонструвала високий рівень якості, проте дещо гірший у порівнянні з LSTM. Статистична модель SARIMAX забезпечила стабільне, хоч і менш якісне прогнозування, особливо на довгострокових інтервалах.

Таким чином, результати експериментального моделювання підтверджують доцільність використання методів інтелектуального аналізу даних для прогнозування показників енергетичного ринку. LSTM слід вважати найефективнішою моделлю для складних часових рядів з високою сезонністю та варіативністю. ANFIS може виступати гнучкою альтернативою, тоді як

SARIMAX — базовим інструментом для інтерпретованих короткострокових прогнозів.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі виконано функціонально-вартісний аналіз програмного продукту, призначеного для прогнозування ключових показників на ринку електроенергії із застосуванням методів інтелектуального аналізу даних. Аналіз спрямований на оцінку доцільності впровадженого функціоналу, визначення рівня витрат на реалізацію кожної з функцій та виявлення найбільш ефективних рішень за критерієм співвідношення функціональності та вартості.

Функціонально-вартісний аналіз (ФВА) є методологією, що передбачає системне дослідження складу функцій програмного продукту, визначення їхньої значущості, оцінювання витрат на реалізацію та техніко-економічне порівняння можливих варіантів. Основна мета застосування ФВА — оптимізація програмного забезпечення шляхом мінімізації витрат при збереженні необхідного рівня функціональних характеристик.

У межах аналізу встановлено перелік функцій, реалізованих у програмному продукті, визначено систему параметрів, проведено їх експертну оцінку, а також виконано порівняння альтернативних варіантів реалізації функціоналу з точки зору якості, вартості та техніко-економічної ефективності. Розрахунок витрат здійснювався з урахуванням трудомісткості реалізації, вартості обчислювальних ресурсів, тривалості розробки та інших релевантних чинників.

На завершальному етапі визначено кращий варіант програмного забезпечення, який забезпечує раціональне поєднання функціональної повноти, точності прогнозування, адаптивності до зовнішніх умов та економічної доцільності реалізації в умовах сучасного енергетичного ринку.

4.1 Постановка задачі проектування

У даному підрозділі сформульовано основні технічні та функціональні вимоги до програмного забезпечення, яке розроблено для прогнозування ключових показників на ринку електроенергії із застосуванням методів інтелектуального аналізу даних. Застосування функціонально-вартісного аналізу (ФВА) дозволяє здійснити структуровану оцінку необхідного функціоналу, визначити доцільність реалізації окремих компонентів системи та обґрунтувати техніко-економічні рішення в процесі розробки.

З урахуванням специфіки задачі прогнозування та характеру вхідних даних, система має забезпечувати обробку великомасштабних часових рядів, адаптацію до зміни зовнішніх факторів, а також збереження точності прогнозів в умовах невизначеності. У зв’язку з цим програмне забезпечення повинно відповідати ряду технічних вимог, які обумовлюють його працездатність, масштабованість та ефективність у реальному середовищі енергетичного ринку.

Основні технічні вимоги до програмного продукту перелічені нижче.

1. Можливість виконання на персональних комп’ютерах або серверних системах із типовими апаратними ресурсами.
2. Підтримка роботи з вхідними даними різного формату, включаючи .csv, .xlsx та API-запити до відкритих джерел.
3. Забезпечення зручного інтерфейсу взаємодії (у вигляді командного рядка, графічного інтерфейсу або інтеграції в Jupyter Notebook).
4. Реалізація ефективних алгоритмів для попередньої обробки даних, побудови моделей, оцінки якості прогнозу та візуалізації результатів.

5. Підтримка гнучкого налаштування параметрів моделей та можливість їх повторного використання без необхідності модифікації коду.
6. Забезпечення швидкої обробки даних із можливістю використання паралельних обчислень при роботі з великими обсягами інформації.
7. Відкритість архітектури програмного продукту для подальшої модифікації, тестування або інтеграції з іншими аналітичними інструментами.

Таким чином, постановка задачі проєктування передбачає реалізацію функціоналу, орієнтованого на практичне використання в умовах нестабільного енергетичного ринку з урахуванням сучасних вимог до продуктивності, адаптивності, масштабованості та економічної доцільності.

4.2 Обґрунтування функцій програмного продукту

Основною функцією програмного продукту (F_0) є реалізація системи прогнозування ключових показників на ринку електроенергії із використанням методів інтелектуального аналізу даних. Ця функція передбачає автоматизацію процесу підготовки даних, побудови моделей, валідації результатів та представлення прогнозної інформації користувачеві у зручній формі.

У контексті функціонально-вартісного аналізу основну функцію F_0 можна декомпозувати на підфункції, що охоплюють ключові етапи розробки програмного забезпечення.

1. F_1: Вибір мови програмування.
2. F_2: Вибір середовища розробки та виконання.
3. F_3: Вибір способу представлення результатів.

Для кожної з підфункцій було визначено декілька можливих варіантів реалізації, які відрізняються за ступенем складності, продуктивності, зручності використання та сумісності з іншими компонентами системи.

Функція F_1.

1. Python — сучасна мова з широким набором бібліотек для аналізу даних та побудови моделей (NumPy, pandas, scikit-learn, TensorFlow).

2. R — мова, орієнтована на статистичні розрахунки та візуалізацію, з великою кількістю спеціалізованих пакетів.

Функція F_2.

1. Google Colaboratory — хмарне середовище, що дозволяє запускати код на GPU/TPU без локального налаштування.

2. DataSpell — локальне інтегроване середовище розробки з розширеною підтримкою .py / .ipynb -проєктів та засобами налагодження.

Функція F_3.

1. Notebook — інтерактивне середовище для покрокового виконання коду з можливістю візуалізації даних.

2. Графічний інтерфейс — самостійно реалізоване візуальне середовище для відображення результатів моделювання у вигляді окремого застосунку або вікна.

Морфологічна карта системи (рис. 4.1), ілюструє всі можливі комбінації реалізації зазначених функцій та їх взаємозв’язки. Така декомпозиція дозволяє здійснити подальший техніко-економічний аналіз кожної з альтернативних конфігурацій програмного продукту з метою вибору оптимального варіанту за критеріями продуктивності, точності, гнучкості та вартості реалізації.

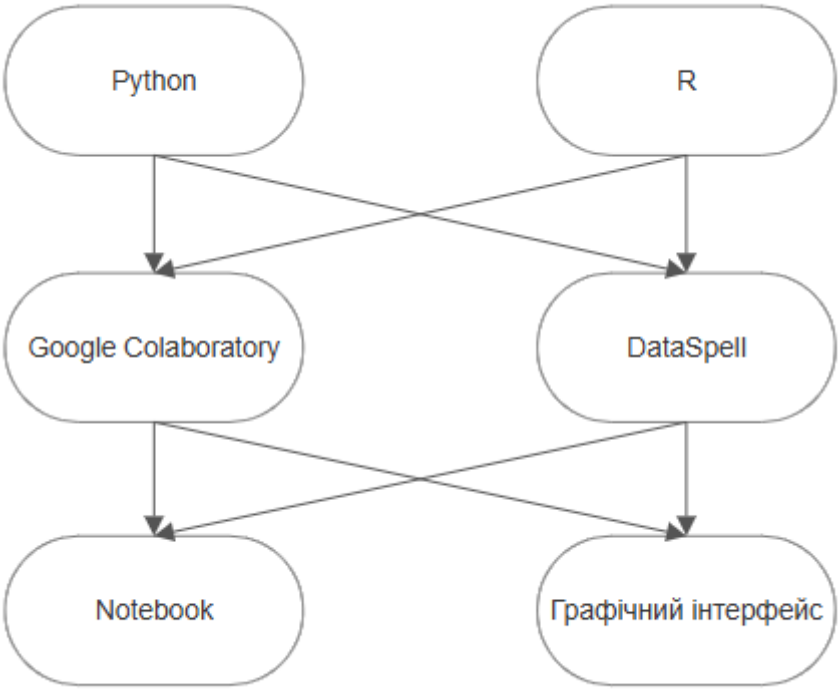


Рисунок 4.1 — Морфологічна карта

Позитивно-негативна матриця наведена в таблиці 4.1.

Таблиця 4.1 — Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	1 — Python	Висока поширеність, активна спільнота, наявність великої кількості бібліотек для машинного навчання, обробки даних та візуалізації, простий синтаксис	Обмежена підтримка апаратного прискорення без сторонніх модулів, повільніше виконання у порівнянні з низькорівневими мовами

Продовження таблиці 4.1

	2 — R	Розвинена екосистема для статистичного аналізу та побудови графіків, підтримка численних пакетів для аналізу часових рядів	Складність у застосуванні для загального програмування, менша кількість універсальних рішень порівняно з Python
F_2	1 — Google Colaboratory	Доступність через браузер, інтеграція з Google Drive, підтримка GPU/TPU, простота розгортання	Обмежена продуктивність при роботі з великими обсягами даних, залежність від інтернет-з'єднання
	2 — DataSpell	Розширені можливості налагодження, автодоповнення коду, підтримка великих проєктів, гнучкість налаштувань, зручний інтерфейс для аналізу даних	Потреба в налаштуванні середовища, залежність від конфігурації локальної машини, відсутність доступу до хмарних обчислень без додаткових інтеграцій
F_3	1 — Notebook	Інтерактивність, зручне документування результатів, підтримка візуалізації в середовищі виконання, можливість поетапного аналізу	Може бути недостатньо ефективним при обробці великих даних, менш гнучке налаштування середовища у порівнянні з IDE

Продовження таблиця 4.1

	2 — Графічний інтерфейс	Простота використання для кінцевих користувачів без знань програмування, інтуїтивно зрозуміле управління	Обмежена функціональність та адаптивність, складність у масштабуванні та модифікації інтерфейсу без доступу до коду

На основі проведеного функціонального порівняння та оцінки переваг і недоліків кожного з варіантів реалізації функцій, наведено в таблиці 4.1, можна зробити висновки щодо їх відповідності поставленим вимогам програмного продукту.

Для функції F_1 (вибір мови програмування) ключовими критеріями є підтримка інтелектуального аналізу даних, наявність широкої бібліотечної екосистеми та можливість інтеграції з іншими системами. З огляду на це, перевага надається варіанту 1 — мові Python, яка забезпечує широку підтримку бібліотек для обробки даних, побудови моделей, візуалізації та гнучкого налаштування. Незважаючи на певні обмеження щодо швидкодії та апаратного прискорення, Python дозволяє реалізувати повноцінну систему прогнозування з перспективою масштабування та подальшої оптимізації.

Щодо функції F_2 (вибір середовища розробки), обидва варіанти — Google Colaboratory (1) та DataSpell (2) — мають чітко виражені переваги залежно від сценарію використання. У випадках, коли важливою є доступність з будь-якого пристрою, простота запуску та інтеграція з хмарними ресурсами, доцільним є використання Google Colaboratory. Водночас при реалізації складних проєктів із необхідністю детального налагодження, роботи з великим кодовим базисом та локального керування залежностями перевагу доцільно надати DataSpell. Таким чином, остаточний вибір залежить від типу задачі та робочого середовища розробника.

У випадку функції F_3 (вибір способу представлення результатів) доцільно обрати варіант 1 — використання інтерактивних ноутбуків, зокрема Jupyter Notebook. Цей підхід дозволяє поєднувати код, візуалізацію, пояснення та результати в єдиному середовищі, що особливо ефективно для демонстрації результатів моделювання, проведення експериментів та оформлення дослідницьких звітів. Варіант 2 (графічний інтерфейс) є актуальним у разі розробки кінцевого продукту для користувачів без технічної підготовки, проте потребує додаткових ресурсів на реалізацію і має нижчу гнучкість у налаштуванні.

Отже, з урахуванням аналізу, найбільш доцільними варіантами реалізації функцій у межах поставленої задачі є наступні.

1. Функція F_1: Python (варіант 1)
2. Функція F_2: залежить від умов використання, допустимими є обидва варіанти 1 і 2
3. Функція F_3: інтерактивне середовище Notebook (варіант 1)

На основі обраних варіантів можна сформувати наступні сценарії реалізації системи.

1. $F_1 1 — F_2 1 — F_3 1$: перевага хмарного середовища, мінімальна потреба у локальній конфігурації, максимальна мобільність.
2. $F_1 1 — F_2 2 — F_3 1$: використання локального IDE для розробки, висока гнучкість та керованість середовищем.

Зазначені конфігурації відповідають сучасним практикам у сфері наукового програмного забезпечення та забезпечують необхідний баланс між функціональністю, гнучкістю та витратами на реалізацію.

4.3 Вибір параметрів

На основі порівняльного аналізу варіантів реалізації функцій програмного продукту, виконаного в попередніх підрозділах, сформовано перелік основних параметрів, які характеризують ефективність, ресурсну вимогливість та технічну доцільність розробленої системи прогнозування. Обрані параметри дозволяють здійснити кількісну оцінку кожного з варіантів реалізації з урахуванням умов експлуатації програмного забезпечення.

До переліку параметрів включено наступні техніко-експлуатаційні показники.

1. X_1 — Швидкодія системи: середній час виконання одного прогнозу при заданому обсязі вхідних даних.
2. X_2 — Споживання оперативної пам’яті: обсяг ОЗП, який використовується під час моделювання та візуалізації результатів.
3. X_3 — Час навчання моделі: повний час, необхідний для навчання однієї конфігурації моделі на типовому наборі даних.
4. X_4 — Обсяг програмного продукту: загальний розмір директорії з усіма файлами програми, необхідними для запуску.

Значення параметрів оцінюються за трибальною шкалою — погано, середньо, добре — що дозволяє виконати подальше нормування результатів і порівняння альтернатив. Границі для кожної оцінки встановлюються з урахуванням технічних можливостей сучасних обчислювальних систем, вимог до продуктивності при експлуатації в реальному середовищі, а також практичних обмежень, пов’язаних із доступними ресурсами.

Узагальнені критерії оцінювання параметрів наведено у таблиці 4.2.

Таблиця 4.2 — Шкала оцінки параметрів програмного забезпечення

Параметр	Умовне позначення	Одиниця виміру	Значення параметра		
			Погано	Середньо	Добре
Швидкість роботи програми	X_1	Операція/мс	0,5	5	20
Обсяг оперативної пам’яті для обчислень	X_2	Мб	8000	5000	1000
Час, що витрачається на навчання моделі	X_3	Сек	10800	3600	1800
Об’єм програмного забезпечення	X_4	Гб	500	150	30

За даними з таблиці 4.2 будуються графічні характеристики параметрів (рисунок 4.2 — рисунок 4.5)

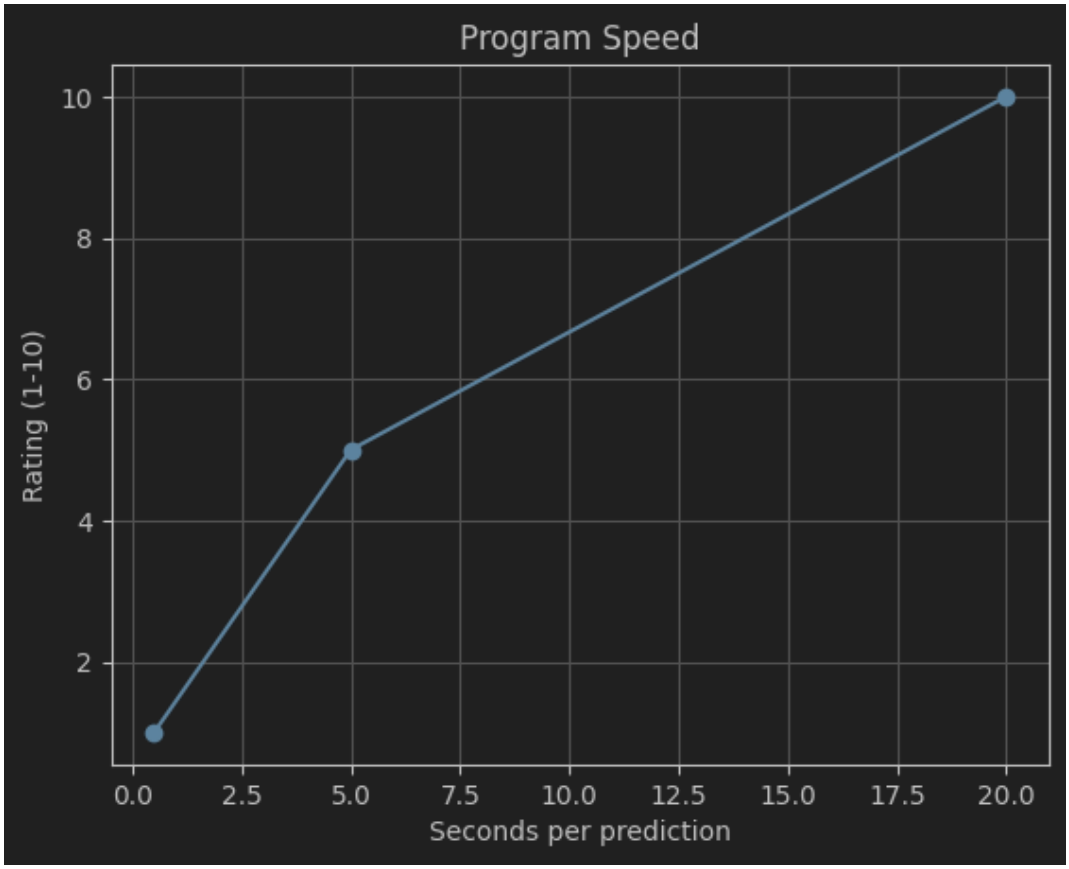


Рисунок 4.2 — X_1, швидкість роботи програми

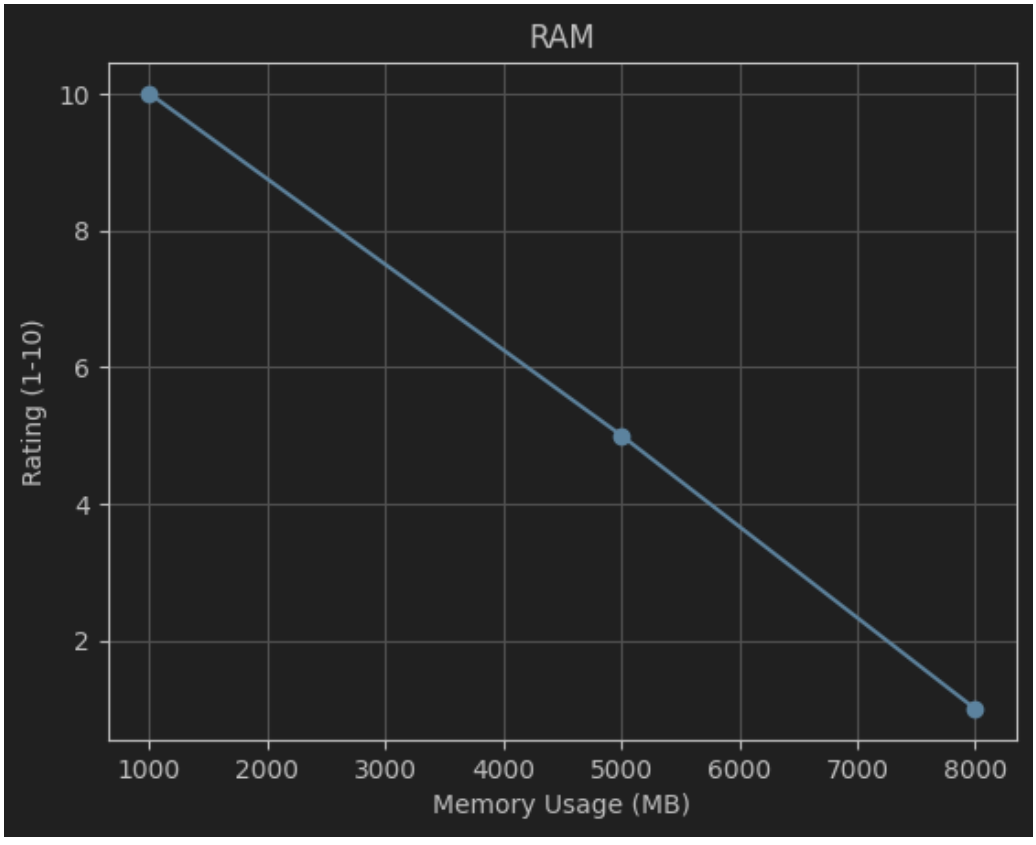


Рисунок 4.3 — X_2, обсяг оперативної пам’яті для обчислень

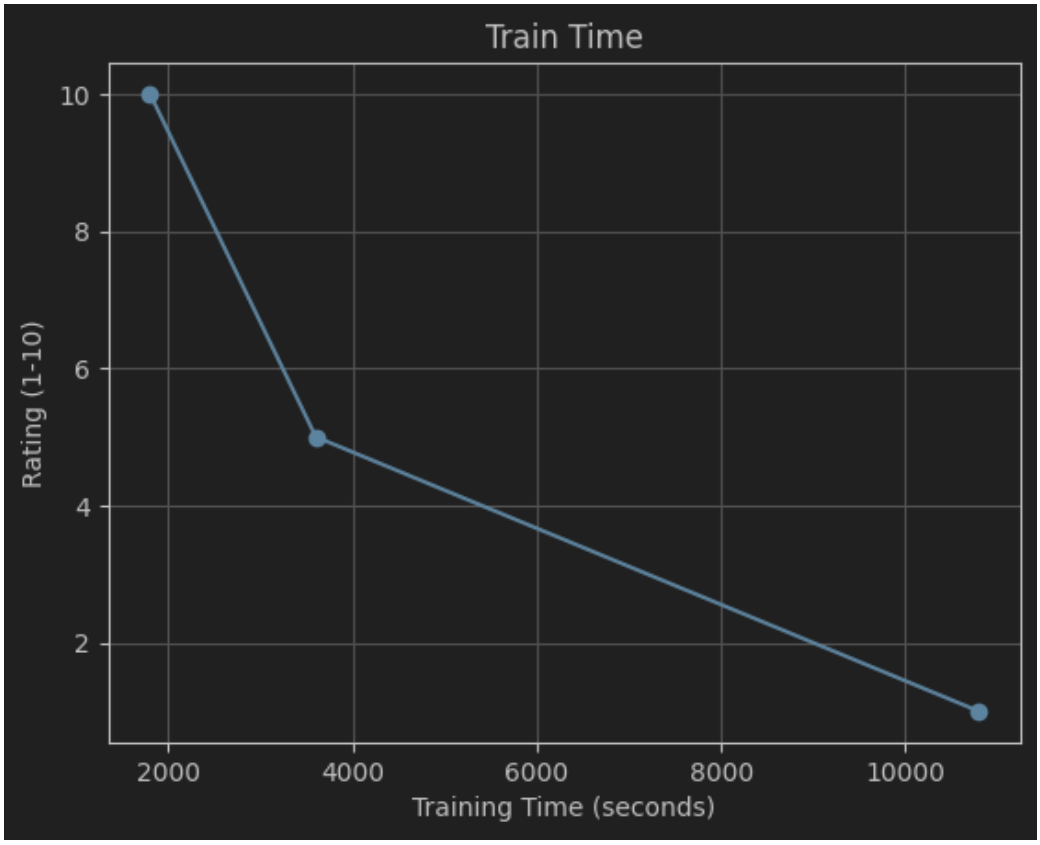


Рисунок 4.4 — X_3, час, що витрачається на навчання моделі

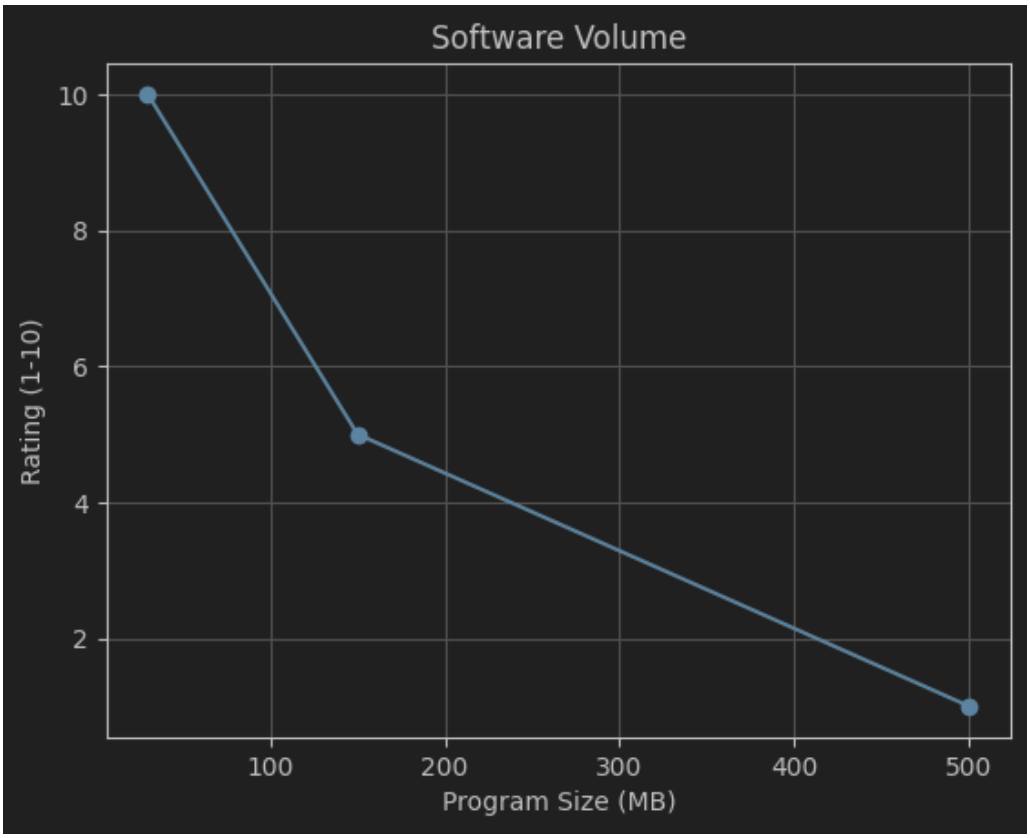


Рисунок 4.5 — X_4, об’єм програмного забезпечення

4.4 Аналіз експертного оцінювання параметрів

Після аналізу ключових параметрів, що характеризують ефективність програмного продукту для прогнозування показників на ринку електроенергії, було здійснено їх експертне оцінювання з метою визначення ступеня впливу кожного з них на загальну якість і доцільність реалізації системи.

Оцінювання проводилось експертною групою у складі 7 фахівців, обізнаних у галузях інтелектуального аналізу даних, прогнозного моделювання та розробки програмного забезпечення. Методом ранжування встановлено відносну важливість кожного з чотирьох параметрів: швидкодії, споживання оперативної пам’яті, часу навчання моделей і загального обсягу програмного продукту.

Оцінювання базується на методі присвоєння рангів, які відображають ступінь важливості відповідного параметра у контексті реалізації точного, ефективного та масштабованого програмного рішення. На основі присвоєних рангів було визначено узагальнені характеристики, необхідні для перевірки достовірності експертних оцінок, зокрема обчислено середнє значення рангу, відхилення та коефіцієнт узгодженості.

Результати ранжування параметрів наведено у таблиці 4.3.

Таблиця 4.3 — Результати експертної оцінки

Позначення	Параметр	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X_1	Швидкість роботи програми	Оп/мс	3	3	4	2	3	4	3	22	4,5	20,25
X_2	Обсяг оперативної пам’яті для обчислень	Мб	2	2	2	3	2	1	2	14	-3,5	12,25
X_3	Час, що витрачається на навчання моделі	Сек	1	1	1	1	1	2	1	8	-9,5	90,25
X_4	Об’єм програмного забезпечення	Гб	4	4	3	4	4	3	4	26	8,5	72,25
	Разом		10	10	10	10	10	10	10	70	0	195

Для перевірки ступеня достовірності експертних оцінок виконаємо обчислення наступних характеристик.

1. Сума рангів кожного параметра та загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij}\,, \qquad \sum R_i = \frac{Nn(n+1)}{2} = \frac{7 \cdot 4 \cdot (4+1)}{2} = 70,$$

де N — кількість експертів;

n — кількість параметрів.

2. Середнє значення суми рангів:

$$T = \frac{1}{n}\sum R_i = \frac{70}{4} = 17{,}5$$

3. Відхилення кожного параметра від середнього значення:

$$\Delta_i = R_i - T$$

Згідно з властивостями методу, сума всіх відхилень повинна дорівнювати нулю.

4. Загальна сума квадратів відхилень:

$$S = \sum \Delta_i^2 = 195$$

5. Коефіцієнт узгодженості Вендта:

$$W = \frac{12 \cdot S}{N^2(n^3 - n)} = \frac{12 \cdot 195}{7^2(4^3 - 4)} = 0{,}796 > W_{\text{кр}} = 0{,}67$$

Оскільки коефіцієнт узгодженості W перевищує нормативне значення, можна зробити висновок про узгодженість експертних оцінок та допустимість їх подальшого використання у процедурі багатокритеріального порівняння.

На основі отриманих результатів ранжування виконується процедура попарного порівняння параметрів, результати якої наведено у таблиці 4.4.

Таблиця 4.4 — Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X_1 і X_2	>	>	>	<	>	>	>	>	1,5
X_1 і X_3	>	>	>	>	>	>	>	>	1,5
X_1 і X_4	<	<	>	<	<	>	<	<	0,5
X_2 і X_3	>	>	>	>	>	<	>	>	1,5

Продовження таблиці 4.4

X_2 і X_4	<	<	<	<	<	<	<	<	0,5
X_3 і X_4	<	<	<	<	<	<	<	<	0,5

На основі експертного оцінювання сформовано матрицю попарного порівняння параметрів, яка відображає відносну перевагу одного параметра над іншим у межах поставленої задачі. Числове значення переваги параметра X_i над параметром X_j, позначене як a_{ij}, визначається за наступним правилом:

$$a_{ij} = \begin{cases} 1.5, & \text{якщо } X_i > X_j \\ 1.0, & \text{якщо } X_i = X_j \\ 0.5, & \text{якщо } X_i < X_j \end{cases}$$

На підставі отриманих оцінок побудовано матрицю $A = \|a_{ij}\|$ яка є основою для подальших обчислень вагомості параметрів. Для кожного параметра обчислюється початковий коефіцієнт вагомості $K_{\text{ві}}$ за формулою:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i},$$

де b_i — сума значень елементів у відповідному рядку матриці:

$$b_i = \sum_{j=1}^{N} a_{ij}$$

Після обчислення початкових значень коефіцієнтів вагомості виконується уточнення їхніх значень методом послідовних наближень. При цьому на кожній наступній ітерації значення вагомості $K_{\text{ві}}$ перераховується за оновленими коефіцієнтами:

$$K'_{\text{ві}} = \frac{b'_i}{\sum_{i=1}^{n} b'_i}$$

$$b'_i = \sum_{j=1}^{N} a_{ij} \cdot b_j$$

Процедура уточнення повторюється до тих пір, поки відносна зміна значень коефіцієнтів вагомості між двома послідовними ітераціями не буде меншою за заданий поріг (у даному випадку — 2%).

Як показано у таблиці 4.5, результати другого етапу уточнення практично не відрізняються від попередніх значень. Відхилення не перевищують 2%, що свідчить про збіжність процедури та можливість використання отриманих коефіцієнтів вагомості для подальшого аналізу.

Таблиця 4.5 — Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер.	
	X_1	X_2	X_2	X_4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X_1	1	1,5	1,5	0,5	4,5	0,28	16,25	0,28	59,125	0,27
X_2	0,5	1	1,5	0,5	3,5	0,22	12,25	0,21	44,875	0,21
X_3	0,5	0,5	1	0,5	2,5	0,16	9,25	0,16	34,125	0,16
X_4	1,5	1,5	1,5	1	5,5	0,34	21,25	0,36	77,875	0,36
Всього					16	1	59	1	216	1

4.6 Аналіз рівня якості варіантів реалізації функцій

У даному підрозділі виконується порівняльний аналіз варіантів реалізації функцій програмного продукту шляхом обчислення узагальненого показника рівня якості для кожної альтернативної конфігурації. Оцінювання базується на значеннях ключових параметрів системи, наведених у попередніх підрозділах, та їх вагомості, визначеної експертним методом.

Для кожної функції (F_1 , F_2 , F_3) обрано конкретні реалізації, що утворюють варіанти повної системи. Для кожного параметра (X_1 - X_4) задано абсолютні значення, які зіставляються зі шкалою оцінювання, поданою у таблиці 4.2, на основі якої визначаються бальні оцінки відповідно до рівня

задоволення технічних вимог. Розрахунок проводиться із використанням вагових коефіцієнтів параметрів, наведених у результатах експертного ранжування.

Коефіцієнт технічного рівня $K_{K(j)}$ для кожного варіанта реалізації функції обчислюється за формулою:

$$K_{K(j)} = \sum_{i=1}^{n} K_{\text{ві}} \cdot B_{ij}$$

де n — кількість оцінюваних параметрів;

$K_{\text{ві}}$ — коефіцієнт вагомості і-го параметра;

B_{ij} — бальна оцінка і-го параметра для j-го варіанта реалізації.

На підставі проведених обчислень формується таблиця 4.6, у якій відображено значення параметрів, їх бальну оцінку, вагомість та відповідні коефіцієнти технічного рівня.

Таблиця 4.6 — Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення	Бальна оцінка	Коефіцієнт вагомості	Коефіцієнт рівня якості
F_1	1 — Python	X_1	18	9	0,27	2,43
F_2	1 — Google Colaboratory	X_2	6500	3	0,21	0,63
	2 — DataSpell	X_3	3600	5	0,16	0,8
F_3	1 — Notebook	X_4	3	10	0,36	3,6

Після визначення коефіцієнтів технічного рівня для кожної функції, обчислюється узагальнений показник якості K_K для кожного варіанта реалізації програмного продукту за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \cdots + K_{\text{ТУ}}[F_{zk}]$$

Використовуючи формулу, розраховано сумарні показники технічного рівня для кожного з варіантів:

$$K_{K1} = 2{,}43 + 0{,}63 + 3.6 = 6{,}66$$

$$K_{K2} = 2{,}43 + 0{,}8 + 3.6 = 6{,}83$$

На основі отриманих результатів встановлено, що варіант із найвищим значенням K_{K2} має кращі технічні характеристики в умовах експлуатації та відповідає сучасним вимогам до якості програмного забезпечення.

4.6 Економічний аналіз варіантів розробки програмного продукту

З метою визначення доцільності реалізації запропонованих варіантів розробки програмного продукту було проведено їх порівняльний економічний аналіз. Розрахунок здійснюється на основі трудомісткості виконання основних етапів робіт, витрат на заробітну плату, відрахувань на соціальні внески, витрат на використання обчислювальних ресурсів (машино-години) та накладних витрат.

Кожен варіант розробки включає два ключові завдання:

1. Розробка проєкту програмного продукту (аналітична та проєктна частина).
2. Розробка програмної оболонки (безпосереднє програмування, тестування, інтеграція).

Перше завдання віднесено до групи А за ступенем новизни, алгоритми — до групи 1 за складністю; друге завдання — до групи Б, з алгоритмами групи 3.

Для обох завдань було використано довідкові норми часу, відповідні поправочні коефіцієнти та коефіцієнти складності. Загальна трудомісткість кожного варіанта обчислюється за формулою:

$$T_O = T_P \cdot K_P \cdot K_{CK} \cdot K_M \cdot K_{CT} \cdot K_{CT.M},$$

де T_P — базова трудомісткість (людино-дні);

K_P — поправочний коефіцієнт залежно від виду інформації;

K_{CK} — коефіцієнт складності інформації;

K_M — коефіцієнт рівня мови програмування;

$K_{CT}, K_{CT.M}$ — коефіцієнти використання стандартних модулів і математичного забезпечення.

На підставі отриманих даних для першого завдання: $T_P = 90$ людино-днів; $K_P = 1.7$; $K_{CT} = 0.8$;

Отже:

$$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4 \text{ людино-дня}$$

Для другого завдання: $T_P = 19$ людино-днів; $K_P = 0.7$; $K_{CT} = 0.75$;

Отже:

$$T_2 = 19 \cdot 0.7 \cdot 0.75 = 9.975 \text{ людино-дня}$$

Загальна трудомісткість кожного варіанта розраховується в людино-годинах (1 людино-день = 8 годин).

1. Варіант 1: $T_1 = (122.4 + 9.975) \cdot 8 = 1059$ годин.
2. Варіант 2: $T_2 = (100 + 9.975) \cdot 8 = 879.8$ годин.

Виконавцями проєкту є: 2 спеціалісти з Data Science з окладом 40000 грн; 1 аналітик з окладом 30000 грн.

Середньогодинна заробітна плата обчислюється за формулою:

$$C_{\text{ч}} = \frac{M}{n \cdot T_m \cdot t}$$

де M — загальний місячний фонд оплати праці;

T_m — кількість робочих днів у місяці (21);

t — тривалість робочого дня (8 годин).

$$C_{\text{ч}} = \frac{(40000 + 40000 + 30000)}{3 \cdot 21 \cdot 8} = 218.25 \text{ грн/год}$$

Витрати на основну заробітну плату з урахуванням додаткових виплат (коефіцієнт $K_{\text{д}} = 1.2$) розраховуються за формулою:

$$\text{С}_{\text{ЗП}} = \text{С}_{\text{ч}} \cdot T_i \cdot K_{\text{д}}$$

1. Варіант 1:

$$C_{\text{ЗП}} = 218.25 \cdot 1059 \cdot 1.2 = 277352.1 \text{ грн}$$

2. Варіант 2:

$$C_{\text{ЗП}} = 218.25 \cdot 879.8 \cdot 1.2 = 230419.62 \text{ грн}$$

Єдиний соціальний внесок ($C_{\text{ВІД}}$), що становить 22%.

1. Варіант 1: $277352.1 \cdot 0.22 = 61017.46$ грн.
2. Варіант 1: $230419.62 \cdot 0.22 = 50692.32$ грн.

Тепер визначимо витрати на оплату однієї машино-години ($C_{\text{М}}$). Витрати на одну ЕОМ для одного фахівця з окладом 40 000 грн; коефіцієнтом зайнятості 0.2; зарплати з урахуванням $K_{\text{д}} = 1.2$: 115200 грн; ЄСВ: $115200 \cdot 0.22 = 25344$ грн.

Амортизація (при 25% вартості 25 000 грн):

$$\text{С}_{\text{А}} = \text{К}_{\text{ТМ}} \cdot \text{К}_{\text{А}} \cdot \text{Ц}_{\text{ПР}} = 1.3 \cdot 0.25 \cdot 25000 = 8125 \text{ грн}$$

де $\text{К}_{\text{ТМ}}$ — коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$\text{К}_{\text{А}}$ — річна норма амортизації;

$\text{Ц}_{\text{ПР}}$ — договірна ціна приладу.

Ремонт:

$$\text{С}_{\text{Р}} = \text{К}_{\text{ТМ}} \cdot \text{К}_{\text{Р}} \cdot \text{Ц}_{\text{ПР}} = 1.3 \cdot 0.08 \cdot 25000 = 2600 \text{ грн}$$

де $\text{К}_{\text{Р}}$ — відсоток витрат на поточні ремонти.

Електроенергія:

$$\text{С}_{\text{ЕЛ}} = \text{Т}_{\text{ЕФ}} \cdot \text{N}_{\text{С}} \cdot \text{К}_{\text{З}} \cdot \text{Ц}_{\text{ЕН}} = 1240 \cdot 0.2 \cdot 0.8 \cdot 9.43 = 1870.91 \text{ грн}$$

де N_C — середньо-споживча потужність приладу;

$К_З$ — коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ — тариф за 1 КВт-годин електроенергії.

Накладні витрати:

$$С_Н = Ц_{ПР} \cdot 0.67 = 25000 \cdot 0.67 = 16750$$

Загальні експлуатаційні витрати:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_р + С_{ЕЛ} + С_Н$$

$$= 115200 + 25344 + 8125 + 2600 + 1870.91 + 16750$$

$$= 169889.91 \text{ грн}$$

Собівартість машино-години (при 1240 год/рік):

$$С_{М-Г} = \frac{С_{ЕКС}}{Т_{ЕФ}} = \frac{169889.91}{1240} = 137.01 \text{ грн/год}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає наступне.

1. Варіант 1: $С_М = 137.01 \cdot 1059 = 145093.59$ грн.
2. Варіант 2: $С_М = 137.01 \cdot 879.8 = 120541.39$ грн.

Накладні витрати складають 67% від заробітної плати.

$$С_Н = С_{ЗП} \cdot 0.67$$

1. Варіант 1: $С_Н = 277352.1 \cdot 0.67 = 185825.91$ грн.
2. Варіант 2: $С_Н = 230419.62 \cdot 0.67 = 154381.15$ грн.

Сумарна вартість розробки програмного продукту за кожним варіантом визначається за формулою:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

1. Варіант 1:

$$С_{ПП} = 277352.1 + 61017.46 + 145093.59 + 185825.91$$

$$= 669{,}289.06 \text{ грн}$$

2. Варіант 2:

$$С_{ПП} = 230419.62 + 50692.32 + 120541.39 + 154381.15$$
$$= 556{,}034.48 \text{ грн}$$

Таким чином, варіант 2 потребує менших витрат на розробку — 556034,48 грн проти 669289,06 грн для варіанта 1. Водночас, як було встановлено в підрозділі 4.5, варіант 2 також має вищий технічний рівень реалізації. Отже, варіант 2 є оптимальним як з економічної точки зору, так і з позиції технічної доцільності реалізації програмного продукту.

4.7 Вибір найкращого варіанту програмного продукту техніко-економічного рівня

Для здійснення остаточного вибору між варіантами реалізації програмного продукту виконується оцінка його техніко-економічного рівня. Цей показник дозволяє узагальнити технічну якість та економічну доцільність реалізації кожного з варіантів у єдиному інтегральному коефіцієнті. Розрахунок здійснюється за формулою:

$$К_{ТЕР(j)} = \frac{K_{K(j)}}{C_{Ф(j)}}$$

де $K_{K(j)}$ — коефіцієнт технічного рівня для j-го варіанта (розрахований у підрозділі 4.5);

$C_{Ф(j)}$— повна вартість реалізації j-го варіанта (розрахована у підрозділі 4.6).

Підставивши отримані значення:

$$К_{ТЕР1} = \frac{6{,}66}{669{,}289.06} \approx 9{,}95 \cdot 10^{-6}$$

$$К_{ТЕР2} = \frac{6{,}83}{556{,}034.48} \approx 1{,}22 \cdot 10^{-5}$$

Аналіз значень показників свідчить, що варіант 2 має вищий коефіцієнт техніко-економічного рівня. Це означає, що даний варіант забезпечує оптимальне співвідношення технічної якості та економічних витрат при реалізації програмного продукту.

Таким чином, за результатами функціонально-вартісного аналізу та техніко-економічного обґрунтування, найбільш доцільним є варіант 2 реалізації системи прогнозування.

Цей варіант передбачає такі параметри: мова програмування — Python, середовище розробки — DataSpell, формат представлення результатів — Notebook.

Вказана конфігурація забезпечує високий рівень автоматизації, гнучкість налаштування моделей, ефективну інтеграцію з інструментами візуалізації та загалом нижчі витрати на розробку, що робить її найбільш прийнятною для впровадження в реальному середовищі ринку електроенергії.

4.8 Висновки до розділу 4

У цьому розділі було здійснено функціонально-вартісний аналіз програмного продукту, розробленого для прогнозування ключових показників на ринку електроенергії із використанням методів інтелектуального аналізу даних.

Проведено ідентифікацію основних функцій програмного забезпечення, визначено можливі варіанти їх реалізації, а також сформовано систему параметрів для подальшої оцінки технічної ефективності. Шляхом експертного оцінювання визначено вагомість кожного параметра, що дозволило здійснити розрахунок рівня технічної якості для кожної з альтернатив.

У результаті проведеного економічного аналізу були обчислені витрати на реалізацію кожного з варіантів, включаючи трудомісткість, вартість машинного часу та накладні витрати. Узагальнення технічних і економічних показників дозволило отримати інтегральну оцінку техніко-економічного рівня.

За результатами аналізу було обґрунтовано вибір оптимального варіанту реалізації програмного продукту (мова програмування — Python, середовище розробки — DataSpell, формат — Notebook), який забезпечує раціональне співвідношення між технічною ефективністю та вартістю розробки, а також відповідає вимогам практичного застосування у сфері енергетичних прогнозів.

ВИСНОВКИ

У процесі виконання дипломної роботи було досліджено та реалізовано методи і моделі інтелектуального аналізу даних для прогнозування ключових показників на ринку електроенергії. Основна увага приділялася порівнянню ефективності різних підходів до побудови прогнозних моделей, оцінці їх точності та адаптивності до умов нестабільного середовища електроенергетичного ринку.

На основі аналізу предметної області та існуючих дослідницьких підходів було сформовано вимоги до програмного забезпечення, призначеного для вирішення задач прогнозування. Проведено порівняння таких моделей, як ANFIS, SARIMAX та LSTM, що дали змогу оцінити їхню практичну ефективність при роботі з реальними часовими рядами.

Розроблено програмний продукт, який забезпечує обробку даних, навчання моделей, обчислення прогнозних значень та візуалізацію результатів. Реалізація здійснювалась із використанням мови програмування Python та інструментів Jupyter Notebook і середовища DataSpell, що надало гнучкість, масштабованість і зручність для подальшої модифікації.

Отримані результати свідчать про практичну доцільність застосування інтелектуальних методів у задачах прогнозування на енергетичних ринках. Подальший розвиток розробленої системи може включати інтеграцію нових джерел даних, оптимізацію гіперпараметрів моделей, розробку графічного інтерфейсу користувача та впровадження модулів автоматичної адаптації моделей до змін зовнішнього середовища.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Закон України "Про ринок електричної енергії" від 13.04.2017 р. № 2019-VIII, поточна редакція – Редакція від 08.03.2024, підстава – 3577-IX. Відомості Верховної Ради України, 2017, № 27-28. Ст. 312. URL: https://zakon.rada.gov.ua/laws/show/2019-19 (дата звернення 10.04.2024)
2. НКРЕКП. Офіційний сайт Національної комісії, що здійснює державне регулювання у сферах енергетики та комунальних послуг. URL: https://www.nerc.gov.ua (дата звернення: 13.04.2025).
3. Оператор ринку. Публікації ринкових результатів. URL: https://www.oree.com.ua (дата звернення 10.05.2025)
4. НЕК "Укренерго". Офіційний сайт Оператора системи передачі України. URL: https://ua.energy (дата звернення 10.05.2025)
5. ENTSO-E Transparency Platform. Дані щодо європейських енергетичних систем. URL: https://transparency.entsoe.eu (дата звернення 20.05.2025)
6. Weron R. Electricity price forecasting: A review of the state-of-the-art with a look into the future. *International Journal of Forecasting,* 2014. Vol. 30. No. 3 P. 1030–1081.
7. Hyndman R.J., Athanasopoulos G. Forecasting: Principles and Practice. 3rd ed. Australia, Monash University, 2021. 390 p. URL: https://otexts.com/fpp3/ (дата звернення 3.05.2025)
8. Хінін Є. К. Аналіз часових рядів: навч. посіб. К.: КНЕУ, 2013. 340 с.
9. Box G. E. P., Jenkins G. M., Reinsel G. C., Ljung G. M. Time Series Analysis: Forecasting and Control. 5th ed. *Wiley*, 2015.
10. Said, S. E., Dickey, D. A. Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika*, 1984. Vol. 71. P. 599–607.

11. Hodrick, R. J., Prescott, E. C. Postwar U.S. Business Cycles: An Empirical Investigation. *Journal of Money, Credit and Banking*, 1997. Vol. 29, No. 1. C. 1–16.
12. Hyndman R. J., Athanasopoulos G. Forecasting: principles and practice. 3rd ed. *OTexts*, 2021. URL: https://otexts.com/fpp3/ (дата звернення 8.05.2025)
13. Box G. E. P., Jenkins G. M., Reinsel G. C., Ljung G. M. Time Series Analysis: Forecasting and Control. 5th ed., *Wiley*, 2015.
14. Rob J. Hyndman, Anne B. Koehler. Another look at measures of forecast accuracy, *International Journal of Forecasting*, 2006. Vol. 22. P.679-688.
15. Graves A. Supervised Sequence Labelling with Recurrent Neural Networks, *Springer*, 2012.
16. Gers F. A., Schmidhuber J., Cummins F. Learning to forget: Continual prediction with LSTM, *Neural Computation*, 2000.
17. Brownlee J. Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python, *Machine Learning Mastery*, 2018.
18. Hochreiter S., Schmidhuber J. Long short-term memory, *Neural computation*, 1997. Vol. 9, No. 8. P. 1735–1780.
19. Zhang G., Eddy Patuwo B., Hu M. Forecasting with artificial neural networks: The state of the art, *International Journal of Forecasting*, 1998. Vol. 14. P. 35–62.
20. Zadeh L. A. Fuzzy sets, *Information and Control*, 1965. Vol. 8. P. 338–353.
21. Takagi T., Sugeno M. Fuzzy identification of systems and its applications to modeling and control, *IEEE Transactions on Systems, Man, and Cybernetics*, 1985. Vol. 15, No. 1. P. 116–132.
22. Jang J.-S. R. ANFIS: Adaptive-Network-Based Fuzzy Inference System, *IEEE Transactions on Systems, Man and Cybernetics*, 1993. Vol. 23, No. 3. P. 665–685.
23. Wang L. Adaptive Fuzzy Systems and Control: Design and Stability Analysis, *Prentice Hall*, 1994.

24. Abraham A. Neuro-fuzzy systems: State-of-the-art modeling techniques. Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence, *Springer*, 2000. P. 269–276
25. Hyndman R. J., Athanasopoulos G. Forecasting: principles and practice. 3rd edition, *OTexts*, 2021. URL: https://otexts.com/fpp3/. (last access 15.05.2025)
26. Willmott C. J., Matsuura K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, *Climate research*, 2005. Vol. 30, No. 1. P. 79–82.
27. Chai T., Draxler R. R. Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature, *Geoscientific Model Development*, 2014. Vol. 7. P. 1247–1250.
28. Kutner M. H., Nachtsheim C. J., Neter J., Li W. Applied Linear Statistical Models. 5th ed., *McGraw-Hill Irwin*, 2005.
29. Abadi, M., Agarwal, A., Barham, P., et al. (2016). TensorFlow: Large-scale machine learning on heterogeneous systems. URL: https://www.tensorflow.org (last access 08.05.2025)
30. Seabold, S., Perktold, J. Statsmodels: Econometric and statistical modeling with Python. Proceedings of the 9th Python in Science Conference, 2010.
31. scikit-fuzzy documentation. — URL: https://github.com/scikit-fuzzy/scikit-fuzzy (last access 08.05.2025)
32. Baghdadi A., Babovic N., Kloft H. Fuzzy Logic, Neural Network, and Adaptive Neuro-Fuzzy Inference System in Delegation of Standard Concrete Beam Calculations, *Buildings* 2023. December 20. P. 4–13
33. McKinney, W. Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 2010. P. 51–56.
34. Harris, C.R., Millman, K.J., van der Walt, S.J. Array programming with NumPy. *Nature*, 2020. Vol. 585. P. 357–362.
35. Hunter, J.D. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering,* 2009. Vol. 9, No. 3. P. 90–95.

36. Waskom, M. et al. Seaborn: Statistical data visualization, *Journal of Open Source Software*, 2020. Vol. 5, No. 51. P. 3021.
37. Selenium. Selenium WebDriver Documentation. — URL: https://www.selenium.dev/documentation/ (last access 04.05.2025)
38. Said, S.E., Dickey, D.A. Testing for unit roots in autoregressive-moving average models of unknown order, *Biometrika*, 1984. Vol. 71, No.3. P 599–607.
39. Kwiatkowski, D., Phillips, P.C.B., Schmidt, P., Shin, Y. Testing the null hypothesis of stationarity against the alternative of a unit root, *Journal of Econometrics*, 1992. Vol. 54, No. 1–3. P. 159–178.
40. Hyndman, R.J., Athanasopoulos, G. Forecasting: Principles and Practice. 2nd ed. *OTexts*. — URL: https://otexts.com/fpp2/ (last access 16.05.2025)
41. Cleveland, R.B., Cleveland, W.S., McRae, J.E., Terpenning, I. STL: A seasonal-trend decomposition procedure based on Loess, *Journal of Official Statistics*, 1990. Vol. 6, No. 1. P. 3–73.

ДОДАТОК А ЛІСТИНГ КОДУ

Модуль table_scraper.ipynb

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import TimeoutException
from io import StringIO

import datetime
import time
import pandas as pd
#%%
def date_generator(start_date_, end_date_ = datetime.date(2025,5, 10)):

    total_days_ = (end_date_ - start_date_).days + 1

    # generate a list of date-strings
    date_list_ = [
        (start_date +
datetime.timedelta(days=offset)).strftime("%d.%m.%Y")
        for offset in range(total_days_)
    ]

    return date_list_
#%%
# define start and end dates
start_date = datetime.date(2020, 1, 1)
end_date   = datetime.date(2025,5, 10)

date_list = date_generator(start_date, end_date)
#%%
# # generate all months between 01.2020 and 05.2025 inclusive
# dates = []
# year, month = 2020, 1
```

```
        #
        # while (year < 2025) or (year == 2025 and month <= 5):
        #     dates.append(f"{month:02d}.{year}")   # format as "mm.yyyy"
        #     month += 1
        #     if month > 12:
        #         month = 1
        #         year += 1
        #%%
        def search_for_table(wait_, date_):

driver.get("https://www.oree.com.ua/index.php/control/results_mo/IDM")
            tab = wait_.until(EC.presence_of_element_located((
                By.XPATH,
                "//div[@id='trade_res_type']//div[normalize-space() = 'Погодинні
результати на ВДР']"
            )))
            time.sleep(2)
            tab.click()

            wait_.until(EC.element_to_be_clickable((By.ID, "date_pxs")))
            driver.find_element(By.ID, "date_pxs").clear()
            driver.find_element(By.ID, "date_pxs").send_keys(date_)
            time.sleep(1)
            driver.find_element(By.ID, "date_pxs").send_keys(Keys.ENTER)
            driver.find_element(By.TAG_NAME, "body").click()
            time.sleep(1)
        #%%
        def table_is_present(wait_):
            try:
                wait_.until(
                    EC.presence_of_element_located((By.CSS_SELECTOR,
"table.site-table"))
                )
                return True
            except TimeoutException:
                return False
        #%%
        def scrap_table(all_dfs_):
            table_ = pd.read_html(StringIO(driver.page_source),
attrs={"class":"site-table"})
            df_, = table_
            print("-------", date, "-------")
```

```
    df_.insert(0, "Date", date)
    df_.columns = all_dfs_.columns

    if all_dfs_.empty:
        return df_.copy()
    else:
        return pd.concat([all_dfs_, df_], ignore_index=True, sort=False)
#%%
service = Service(executable_path="chromedriver.exe")
driver = webdriver.Chrome(service=service)
driver.get("https://www.oree.com.ua/index.php/control/results_mo/IDM")
# https://www.oree.com.ua/index.php/pricectr  | only price
# https://www.oree.com.ua/index.php/control/results_mo/IDM | all data

# <div class="tab-trade-res active">Погодинні результати на ВДР</div>
all_dfs = pd.DataFrame(columns=['Date',
                                'Hour',
                                'Price, UAH/MWh',
                                'Minimum price, UAH/MWh',
                                'Maximum price, UAH/MWh',
                                'Last price, UAH/MWh',
                                'Sales volume, MWh',
                                'Purchase volume, MWh',
                                'Declared sales volume, MWh',
                                'Declared purchase volume, MWh'])

wait = WebDriverWait(driver, 10)

for date in date_list:

    search_for_table(wait, date)


    while not table_is_present(wait):
        search_for_table(wait, date)
    all_dfs = scrap_table(all_dfs)
    time.sleep(1)

all_dfs
#%%
all_dfs.to_csv('output.csv', index=False)
```

Модуль research.ipynb

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from statsmodels.tsa.stattools import adfuller, kpss
import warnings
from statsmodels.tools.sm_exceptions import InterpolationWarning
warnings.simplefilter('ignore', InterpolationWarning)
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from scipy import stats
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.statespace.sarimax import SARIMAX
import skfuzzy as fuzz
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
import sys
import os
sys.path.append(os.path.abspath('./anfis'))
from anfis.anfis import ANFIS
import itertools

#%%
df = pd.read_csv('output.csv')
df['Hour'] = df['Hour'] - 1
df = df.drop(index=[7175, 16079, 24815, 33551,
42287]).reset_index(drop=True) #
df
#%%
df['Hour'] = df['Hour'].fillna('00').astype(str).str.zfill(2)  # Add
leading zeros if needed
df['timestamp'] = pd.to_datetime(df['Date'].astype(str) + ' ' +
df['Hour'], format='mixed')
df = df.drop(['Date', 'Hour'], axis=1)

cols = ['timestamp'] + [col for col in df.columns if col != 'timestamp']
df = df[cols]
df
```

```
      #%%
      df = df.sort_values(by='timestamp')
      df = df.set_index('timestamp')
      df = df.asfreq('h')
      df
      #%%
      fig, axes = plt.subplots(nrows=len(df.columns), figsize=(15, 4 *
len(df.columns)), squeeze=False)
      axes = axes.flatten()

      for idx, col in enumerate(df.columns):
          axes[idx].plot(df.index, df[col])
          axes[idx].set_title(col)
          axes[idx].tick_params(axis='x', rotation=45)
          axes[idx].grid(True)

      plt.tight_layout()
      plt.show()
      #%%
      fig, axes = plt.subplots(nrows=len(df.columns), figsize=(15, 4 *
len(df.columns)), squeeze=False)
      axes = axes.flatten()

      for idx, col in enumerate(df.columns):

          axes[idx].plot(df.index, df[col], label='Values', color='blue')

          nan_mask = df[col].isna()
          if nan_mask.any():
              for nan_timestamp in df.index[nan_mask]:
                  axes[idx].axvline(x=nan_timestamp, color='red', alpha=0.3,
linestyle='--', label='NaN')

              handles, labels = axes[idx].get_legend_handles_labels()
              unique_labels = dict(zip(labels, handles))
              axes[idx].legend(unique_labels.values(), unique_labels.keys())

          axes[idx].set_title(col)
          axes[idx].tick_params(axis='x', rotation=45)
          axes[idx].grid(True)

      plt.tight_layout()
```

```
plt.show()
#%%
df = df.interpolate(method='time', limit_direction='both')
df
#%%
time_series_dfs = {}
for column in df.columns:
    time_series_dfs[column] = pd.DataFrame(df[column])
#%%
def check_stationarity(series, series_name):
    # ADF Test
    adf_result = adfuller(series.dropna())
    adf_pvalue = adf_result[1]

    # KPSS Test
    kpss_result = kpss(series.dropna())
    kpss_pvalue = kpss_result[1]

    results = pd.DataFrame({
        'ADF p-value': [adf_pvalue],
        'KPSS p-value': [kpss_pvalue],
        'ADF Stationary': [adf_pvalue < 0.05],
        'KPSS Stationary': [kpss_pvalue >= 0.05]
    }, index=[series_name])

    return results


stationarity_results = pd.DataFrame()
for column in df.columns:
    result = check_stationarity(df[column], column)
    stationarity_results = pd.concat([stationarity_results, result])

stationarity_results
#%%
for column in df.columns:
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 4))

    plot_acf(df[column].dropna(), ax=ax1, title=f'Weekly ACF -
{column}', lags=7*24)
    plot_pacf(df[column].dropna(), ax=ax2, title=f'Weekly PACF -
{column}', lags=7*24)
```

```
            plt.tight_layout()
            plt.show()
        #%%
        def check_seasonality(data, column, window=24):  # 24 hours window for
daily seasonality
            seasonal_means = data[column].groupby(data.index.hour).mean()
            seasonal_std = data[column].groupby(data.index.hour).std()

            f_stat, p_value = stats.f_oneway(*[group[1].values for group in
data[column].groupby(data.index.hour)])

            plt.figure(figsize=(12, 4))
            plt.plot(seasonal_means.index, seasonal_means.values)
            plt.fill_between(seasonal_means.index,
                             seasonal_means - seasonal_std,
                             seasonal_means + seasonal_std,
                             alpha=0.2)
            plt.title(f'Daily Seasonality Pattern - {column}\nANOVA p-value:
{p_value:.10f}')
            plt.xlabel('Hour of Day')
            plt.ylabel('Mean Value')
            plt.grid(True)
            plt.show()


        for column in df.columns:
            check_seasonality(df, column)
        #%%
        for column in df.columns:
            # Perform decomposition
            decomposition = seasonal_decompose(df[column], period=24 * 7 * 52)
# Weekly seasonality (24*7 hours)

            # Plot decomposition
            fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(15, 12))

            decomposition.observed.plot(ax=ax1)
            ax1.set_title(f'Time Series Decomposition - {column}')
            ax1.set_ylabel('Observed')

            decomposition.trend.plot(ax=ax2)
```

```
    ax2.set_ylabel('Trend')

    decomposition.seasonal.plot(ax=ax3)
    ax3.set_ylabel('Seasonal')

    decomposition.resid.plot(ax=ax4)
    ax4.set_ylabel('Residual')

    plt.tight_layout()
    plt.show()
#%%
# Store decomposition components in dictionary
decomposition_components = {}
for column in df.columns:
    decomp = seasonal_decompose(df[column], period=24 * 7)
    decomposition_components[column] = {
        'trend': decomp.trend,
        'seasonal': decomp.seasonal,
        'residual': decomp.resid
    }
#%%
# Create splits
splits = {
    'daily': {'test_size': 24},
    'weekly': {'test_size': 168},
    'monthly': {'test_size': 720}
}

train_test_splits = {}

for name, ts_df in time_series_dfs.items():
    train_test_splits[name] = {}

    for split_name, params in splits.items():
        test_size = params['test_size']

        train = ts_df[:-test_size]
        test = ts_df[-test_size:]

        train_test_splits[name][split_name] = {
            'train': train,
            'test': test
```

```
            }
    #%%
    # Visualize the split for each time series
    for column in df.columns:
        for split_name, params in train_test_splits[column].items():
            plt.figure(figsize=(15, 5))
            plt.plot(params['train'].index, params['train'].values,
label='Training', color='blue')
            plt.plot(params['test'].index, params['test'].values,
label='Testing', color='red')
            plt.title(f'{split_name} Train-Test Split for {column}')
            plt.xlabel('Timestamp')
            plt.ylabel('Value')
            plt.legend()
            plt.grid(True)
            plt.xticks(rotation=45)
            plt.tight_layout()
            plt.show()

    #%%
    # Dictionary to store results and predictions
    results = {}
    all_predictions = pd.DataFrame()

    # Define SARIMAX parameters
    order = (24, 0, 24)
    seasonal_order = (24, 2, 24, 168)

    # Dictionary to store evaluation metrics
    metrics_dict = {
        'MSE': [],
        'RMSE': [],
        'MAE': [],
        'R2': [],
        'Series': [],
        'Split': []
    }

    # Train and evaluate models
    for series_name, splits in train_test_splits.items():
        results[series_name] = {}
```

```
for split_name, split_data in splits.items():
    print(f"Training model for {series_name} - {split_name} split")

    train = split_data['train']
    test = split_data['test']

    # Fit SARIMAX model
    model = SARIMAX(train,
                    order=order,
                    seasonal_order=seasonal_order)

    fitted_model = model.fit(disp=False)

    # Make predictions
    predictions = fitted_model.forecast(len(test))

    # Store predictions
    pred_df = pd.DataFrame({
        'timestamp': test.index,
        'actual': test.values.flatten(),
        'predicted': predictions,
        'series': series_name,
        'split': split_name
    })
    all_predictions = pd.concat([all_predictions, pred_df])

    # Calculate metrics
    mse = mean_squared_error(test, predictions)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(test, predictions)
    r2 = r2_score(test, predictions)

    # Store metrics
    metrics_dict['MSE'].append(mse)
    metrics_dict['RMSE'].append(rmse)
    metrics_dict['MAE'].append(mae)
    metrics_dict['R2'].append(r2)
    metrics_dict['Series'].append(series_name)
    metrics_dict['Split'].append(split_name)

    # Visualize results
    plt.figure(figsize=(15, 5))
```

```
            plt.plot(test.index, test.values, label='Actual', color='blue')
            plt.plot(test.index, predictions, label='Predicted',
color='red')
            plt.title(f'SARIMAX Predictions vs Actual Values\n{series_name}
- {split_name} split')
            plt.xlabel('Timestamp')
            plt.ylabel('Value')
            plt.legend()
            plt.grid(True)
            plt.xticks(rotation=45)
            plt.tight_layout()
            plt.show()

    # Create metrics DataFrame
    metrics_df = pd.DataFrame(metrics_dict)
    metrics_df = metrics_df.round(4)
    metrics_df = metrics_df.sort_values(['Series', 'Split'])
    print("\nModel Performance Metrics:")
    display(metrics_df)

    # Save predictions
    all_predictions.to_csv('sarimax_predictions.csv', index=False)
    print("\nPredictions saved to sarimax_predictions.csv")

    #%%
    # Dictionary to store results and predictions
    lstm_results = {}
    all_lstm_predictions = pd.DataFrame()

    # Define parameter grid
    param_grid = {
        'sequence_length': [24 * 7, 24 * 14],  # 1-2 weeks based on
seasonality analysis
        'n_lstm_units': [50, 100],
        'n_lstm_layers': [1, 2],
        'dropout_rate': [0.1, 0.2],
        'learning_rate': [0.001, 0.0001]
    }

    # Dictionary for metrics
    metrics_dict = {
        'MSE': [], 'RMSE': [], 'MAE': [], 'R2': [],
```

```
            'Series': [], 'Split': [], 'Parameters': []
        }

        # Normalize data and create sequences
        scaler = MinMaxScaler()

        for series_name, splits in train_test_splits.items():
            lstm_results[series_name] = {}

            for split_name, split_data in splits.items():
                print(f"\nTraining models for {series_name} - {split_name}
split")

                train = split_data['train']
                test = split_data['test']

                # Scale the data
                train_scaled = scaler.fit_transform(train)
                test_scaled = scaler.transform(test)

                best_score = float('inf')
                best_params = None
                best_predictions = None

                # Grid search
                for seq_len in param_grid['sequence_length']:
                    for n_units in param_grid['n_lstm_units']:
                        for n_layers in param_grid['n_lstm_layers']:
                            for dropout in param_grid['dropout_rate']:
                                for lr in param_grid['learning_rate']:

                                    # Prepare sequences
                                    X_train, y_train = [], []
                                    for i in range(len(train_scaled) - seq_len):
                                        X_train.append(train_scaled[i:(i +
seq_len)])
                                        y_train.append(train_scaled[i +
seq_len])
                                    X_train, y_train = np.array(X_train),
np.array(y_train)

                                    # Build model
```

```
                                    model = Sequential()
                                    for i in range(n_layers):
                                        if i == 0:
                                            model.add(LSTM(n_units,
input_shape=(seq_len, 1), return_sequences=(n_layers > 1)))
                                        elif i == n_layers - 1:
                                            model.add(LSTM(n_units))
                                        else:
                                            model.add(LSTM(n_units,
return_sequences=True))
                                        model.add(Dropout(dropout))
                                    model.add(Dense(1))

                                    # Compile and train

model.compile(optimizer=Adam(learning_rate=lr), loss='mse')
                                    model.fit(X_train, y_train, epochs=50,
batch_size=32, verbose=0)

                                    # Make predictions
                                    X_test, y_test = [], []
                                    for i in range(len(test_scaled) - seq_len):
                                        X_test.append(test_scaled[i:(i +
seq_len)])
                                        y_test.append(test_scaled[i + seq_len])
                                    X_test, y_test = np.array(X_test),
np.array(y_test)

                                    predictions_scaled = model.predict(X_test)
                                    predictions =
scaler.inverse_transform(predictions_scaled)
                                    actual = scaler.inverse_transform(y_test)

                                    # Calculate metrics
                                    mse = mean_squared_error(actual,
predictions)

                                    # Update best model
                                    if mse < best_score:
                                        best_score = mse
                                        best_params = {
                                            'sequence_length': seq_len,
```

```
                                        'n_lstm_units': n_units,
                                        'n_lstm_layers': n_layers,
                                        'dropout_rate': dropout,
                                        'learning_rate': lr
                                    }
                                    best_predictions = predictions

                # Calculate final metrics for best model
                mse = mean_squared_error(actual, best_predictions)
                rmse = np.sqrt(mse)
                mae = mean_absolute_error(actual, best_predictions)
                r2 = r2_score(actual, best_predictions)

                # Store metrics
                metrics_dict['MSE'].append(mse)
                metrics_dict['RMSE'].append(rmse)
                metrics_dict['MAE'].append(mae)
                metrics_dict['R2'].append(r2)
                metrics_dict['Series'].append(series_name)
                metrics_dict['Split'].append(split_name)
                metrics_dict['Parameters'].append(str(best_params))

                # Store predictions
                pred_df = pd.DataFrame({
                    'timestamp': test.index[seq_len:],
                    'actual': actual.flatten(),
                    'predicted': best_predictions.flatten(),
                    'series': series_name,
                    'split': split_name
                })
                all_lstm_predictions = pd.concat([all_lstm_predictions,
pred_df])

                # Visualize results
                plt.figure(figsize=(15, 5))
                plt.plot(test.index[seq_len:], actual, label='Actual',
color='blue')
                plt.plot(test.index[seq_len:], best_predictions,
label='Predicted', color='red')
                plt.title(f'LSTM Predictions vs Actual Values\n{series_name} -
{split_name} split')
                plt.xlabel('Timestamp')
```

```
        plt.ylabel('Value')
        plt.legend()
        plt.grid(True)
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

# Create metrics DataFrame
metrics_df = pd.DataFrame(metrics_dict)
metrics_df = metrics_df.round(4)
metrics_df = metrics_df.sort_values(['Series', 'Split'])
print("\nModel Performance Metrics:")
display(metrics_df)

# Save predictions
all_lstm_predictions.to_csv('lstm_predictions.csv', index=False)
print("\nPredictions saved to lstm_predictions.csv")

#%%
sys.path.append('./anfis')  # Add path to local ANFIS implementation

# Dictionary to store results and predictions
anfis_results = {}
all_anfis_predictions = pd.DataFrame()

# Define parameter grid
param_grid = {
    'n_rules': [20, 50, 100, 150, 200, 250],
    'n_steps': [24, 24 * 7, 24 * 30],
    'epochs': [10, 50, 100]
}

# Dictionary for metrics
metrics_dict = {
    'MSE': [], 'RMSE': [], 'MAE': [], 'R2': [],
    'Series': [], 'Split': [], 'Parameters': []
}

# Scale data
scaler = MinMaxScaler()

for series_name, splits in train_test_splits.items():
```

```
            anfis_results[series_name] = {}

            for split_name, split_data in splits.items():
                print(f"\nTraining models for {series_name} - {split_name}
split")

                train = split_data['train']
                test = split_data['test']

                # Scale data
                train_scaled = scaler.fit_transform(train)
                test_scaled = scaler.transform(test)

                best_score = float('inf')
                best_params = None
                best_predictions = None

                # Grid search
                for params in itertools.product(param_grid['n_rules'],
param_grid['n_steps'], param_grid['epochs']):
                    n_rules, n_steps, epochs = params

                    # Prepare sequences
                    X_train, y_train = [], []
                    for i in range(len(train_scaled) - n_steps):
                        X_train.append(train_scaled[i:(i + n_steps)].flatten())
                        y_train.append(train_scaled[i + n_steps])
                    X_train, y_train = np.array(X_train),
np.array(y_train).reshape(-1, 1)

                    # Create membership functions
                    mf = fuzz.membership.membershipfunction.MemFuncs(X_train)

                    # Create and train ANFIS model
                    model = ANFIS(X_train, y_train, mf)
                    model.trainHybridJangOffLine(epochs=epochs)

                    # Make predictions
                    X_test, y_test = [], []
                    for i in range(len(test_scaled) - n_steps):
                        X_test.append(test_scaled[i:(i + n_steps)].flatten())
                        y_test.append(test_scaled[i + n_steps])
```

```
                X_test, y_test = np.array(X_test), np.array(y_test)

                predictions_scaled = model.predict(X_test)
                predictions = scaler.inverse_transform(predictions_scaled)
                actual = scaler.inverse_transform(y_test.reshape(-1, 1))

                # Calculate metrics
                mse = mean_squared_error(actual, predictions)

                # Update best model
                if mse < best_score:
                    best_score = mse
                    best_params = {'n_rules': n_rules, 'n_steps': n_steps,
'epochs': epochs}
                    best_predictions = predictions

            # Calculate final metrics for best model
            mse = mean_squared_error(actual, best_predictions)
            rmse = np.sqrt(mse)
            mae = mean_absolute_error(actual, best_predictions)
            r2 = r2_score(actual, best_predictions)

            # Store metrics
            metrics_dict['MSE'].append(mse)
            metrics_dict['RMSE'].append(rmse)
            metrics_dict['MAE'].append(mae)
            metrics_dict['R2'].append(r2)
            metrics_dict['Series'].append(series_name)
            metrics_dict['Split'].append(split_name)
            metrics_dict['Parameters'].append(str(best_params))

            # Store predictions
            pred_df = pd.DataFrame({
                'timestamp': test.index[best_params['n_steps']:],
                'actual': actual.flatten(),
                'predicted': best_predictions.flatten(),
                'series': series_name,
                'split': split_name
            })
            all_anfis_predictions = pd.concat([all_anfis_predictions,
pred_df])
```

```
            # Visualize results
            plt.figure(figsize=(15, 5))
            plt.plot(test.index[best_params['n_steps']:], actual,
label='Actual', color='blue')
            plt.plot(test.index[best_params['n_steps']:], best_predictions,
label='Predicted', color='red')
            plt.title(f'ANFIS Predictions vs Actual Values\n{series_name} -
{split_name} split')
            plt.xlabel('Timestamp')
            plt.ylabel('Value')
            plt.legend()
            plt.grid(True)
            plt.xticks(rotation=45)
            plt.tight_layout()
            plt.show()

    # Create metrics DataFrame
    metrics_df = pd.DataFrame(metrics_dict)
    metrics_df = metrics_df.round(4)
    metrics_df = metrics_df.sort_values(['Series', 'Split'])
    print("\nModel Performance Metrics:")
    display(metrics_df)

    # Save predictions
    timestamp = pd.Timestamp.now().strftime('%Y%m%d_%H%M%S')
    filename = f'anfis_predictions_{timestamp}.csv'
    all_anfis_predictions.to_csv(filename, index=False)
    print(f"\nPredictions saved to {filename}")
```

Модуль anfis.py

```
    # -*- coding: utf-8 -*-
    """
    Created on Thu Apr 03 07:30:34 2014

    @author: tim.meggs
    """
    import itertools
    import numpy as np
    from .membership import mfDerivs
    import copy
```

```
      class ANFIS:
          """Class to implement an Adaptive Network Fuzzy Inference System:
ANFIS"

          Attributes:
              X
              Y
              XLen
              memClass
              memFuncs
              memFuncsByVariable
              rules
              consequents
              errors
              memFuncsHomo
              trainingType


          """

          def __init__(self, X, Y, memFunction):
              self.X = np.array(copy.copy(X))
              self.Y = np.array(copy.copy(Y))
              self.XLen = len(self.X)
              self.memClass = copy.deepcopy(memFunction)
              self.memFuncs = self.memClass.MFList
              self.memFuncsByVariable = [[x for x in
range(len(self.memFuncs[z]))] for z in range(len(self.memFuncs))]
              self.rules =
np.array(list(itertools.product(*self.memFuncsByVariable)))
              self.consequents = np.empty(self.Y.ndim * len(self.rules) *
(self.X.shape[1] + 1))
              self.consequents.fill(0)
              self.errors = np.empty(0)
              self.memFuncsHomo = all(len(i)==len(self.memFuncsByVariable[0])
for i in self.memFuncsByVariable)
              self.trainingType = 'Not trained yet'

          def LSE(self, A, B, initialGamma = 1000.):
              coeffMat = A
              rhsMat = B
              S = np.eye(coeffMat.shape[1])*initialGamma
```

```
            x = np.zeros((coeffMat.shape[1],1)) # need to correct for multi-
dim B
            for i in range(len(coeffMat[:,0])):
                a = coeffMat[i,:]
                b = np.array(rhsMat[i])
                S = S -
(np.array(np.dot(np.dot(np.dot(S,np.matrix(a).transpose()),np.matrix(a)),S)))
/(1+(np.dot(np.dot(S,a),a)))
                x = x +
(np.dot(S,np.dot(np.matrix(a).transpose(),(np.matrix(b)-
np.dot(np.matrix(a),x)))))
            return x

        def trainHybridJangOffLine(self, epochs=5, tolerance=1e-5,
initialGamma=1000, k=0.01):

            self.trainingType = 'trainHybridJangOffLine'
            convergence = False
            epoch = 1

            while (epoch < epochs) and (convergence is not True):

                #layer four: forward pass
                [layerFour, wSum, w] = forwardHalfPass(self, self.X)

                #layer five: least squares estimate
                layerFive =
np.array(self.LSE(layerFour,self.Y,initialGamma))
                self.consequents = layerFive
                layerFive = np.dot(layerFour,layerFive)

                #error
                error = np.sum((self.Y-layerFive.T)**2)
                print('current error: '+ str(error))
                average_error = np.average(np.absolute(self.Y-layerFive.T))
                self.errors = np.append(self.errors,error)

                if len(self.errors) != 0:
                    if self.errors[len(self.errors)-1] < tolerance:
                        convergence = True

                # back propagation
```

```
                if convergence is not True:
                    cols = range(len(self.X[0,:]))
                    dE_dAlpha = list(backprop(self, colX, cols, wSum, w,
layerFive) for colX in range(self.X.shape[1]))


                if len(self.errors) >= 4:
                    if (self.errors[-4] > self.errors[-3] > self.errors[-2]
> self.errors[-1]):
                        k = k * 1.1

                if len(self.errors) >= 5:
                    if (self.errors[-1] < self.errors[-2]) and
(self.errors[-3] < self.errors[-2]) and (self.errors[-3] < self.errors[-4])
and (self.errors[-5] > self.errors[-4]):
                        k = k * 0.9

                ## handling of variables with a different number of MFs
                t = []
                for x in range(len(dE_dAlpha)):
                    for y in range(len(dE_dAlpha[x])):
                        for z in range(len(dE_dAlpha[x][y])):
                            t.append(dE_dAlpha[x][y][z])

                eta = k / np.abs(np.sum(t))

                if(np.isinf(eta)):
                    eta = k

                ## handling of variables with a different number of MFs
                dAlpha = copy.deepcopy(dE_dAlpha)
                if not(self.memFuncsHomo):
                    for x in range(len(dE_dAlpha)):
                        for y in range(len(dE_dAlpha[x])):
                            for z in range(len(dE_dAlpha[x][y])):
                                dAlpha[x][y][z] = -eta * dE_dAlpha[x][y][z]
                else:
                    dAlpha = -eta * np.array(dE_dAlpha)


                for varsWithMemFuncs in range(len(self.memFuncs)):
```

```
                    for MFs in
range(len(self.memFuncsByVariable[varsWithMemFuncs])):
                        paramList =
sorted(self.memFuncs[varsWithMemFuncs][MFs][1])
                        for param in range(len(paramList)):

self.memFuncs[varsWithMemFuncs][MFs][1][paramList[param]] =
self.memFuncs[varsWithMemFuncs][MFs][1][paramList[param]] +
dAlpha[varsWithMemFuncs][MFs][param]
                epoch = epoch + 1


            self.fittedValues = predict(self,self.X)
            self.residuals = self.Y - self.fittedValues[:,0]

            return self.fittedValues


        def plotErrors(self):
            if self.trainingType == 'Not trained yet':
                print(self.trainingType)
            else:
                import matplotlib.pyplot as plt
                plt.plot(range(len(self.errors)),self.errors,'ro',
label='errors')
                plt.ylabel('error')
                plt.xlabel('epoch')
                plt.show()

        def plotMF(self, x, inputVar):
            import matplotlib.pyplot as plt
            from skfuzzy import gaussmf, gbellmf, sigmf

            for mf in range(len(self.memFuncs[inputVar])):
                if self.memFuncs[inputVar][mf][0] == 'gaussmf':
                    y = gaussmf(x,**self.memClass.MFList[inputVar][mf][1])
                elif self.memFuncs[inputVar][mf][0] == 'gbellmf':
                    y = gbellmf(x,**self.memClass.MFList[inputVar][mf][1])
                elif self.memFuncs[inputVar][mf][0] == 'sigmf':
                    y = sigmf(x,**self.memClass.MFList[inputVar][mf][1])

                plt.plot(x,y,'r')
```

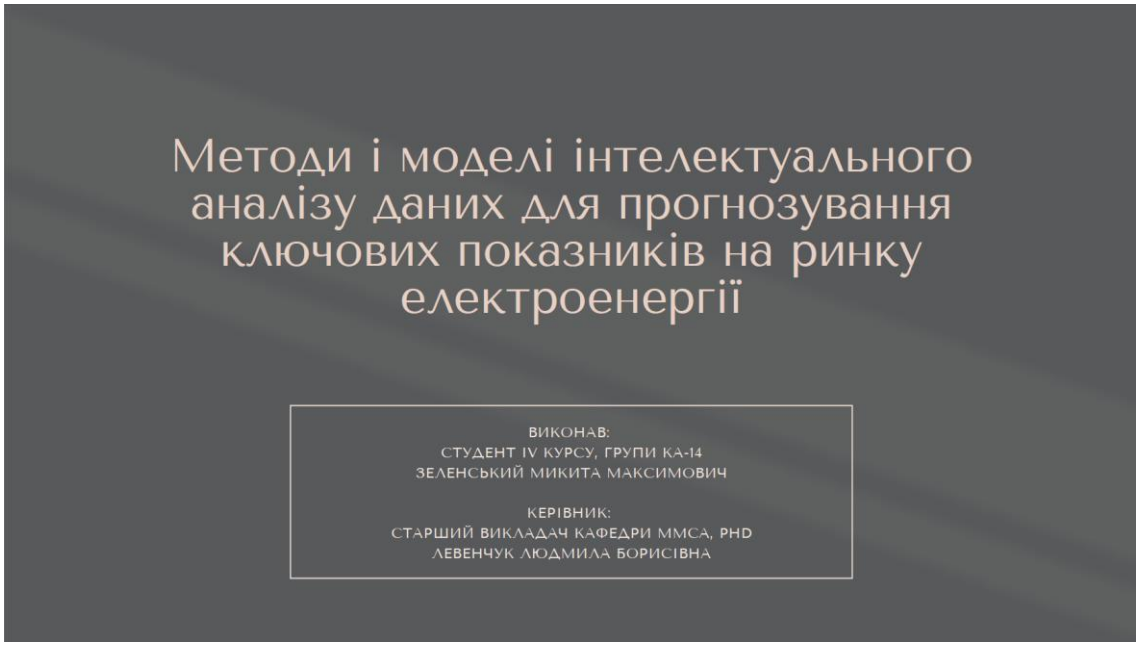
```
            plt.show()

        def plotResults(self):
            if self.trainingType == 'Not trained yet':
                print(self.trainingType)
            else:
                import matplotlib.pyplot as plt

plt.plot(range(len(self.fittedValues)),self.fittedValues,'r',
label='trained')
                plt.plot(range(len(self.Y)),self.Y,'b', label='original')
                plt.legend(loc='upper left')
                plt.show()



    def forwardHalfPass(ANFISObj, Xs):
        layerFour = np.empty(0,)
        wSum = []

        for pattern in range(len(Xs[:,0])):
            #layer one
            layerOne = ANFISObj.memClass.evaluateMF(Xs[pattern,:])

            #layer two
            miAlloc = [[layerOne[x][ANFISObj.rules[row][x]] for x in
range(len(ANFISObj.rules[0]))] for row in range(len(ANFISObj.rules))]
            layerTwo = np.array([np.product(x) for x in miAlloc]).T
            if pattern == 0:
                w = layerTwo
            else:
                w = np.vstack((w,layerTwo))

            #layer three
            wSum.append(np.sum(layerTwo))
            if pattern == 0:
                wNormalized = layerTwo/wSum[pattern]
            else:
                wNormalized =
np.vstack((wNormalized,layerTwo/wSum[pattern]))
```

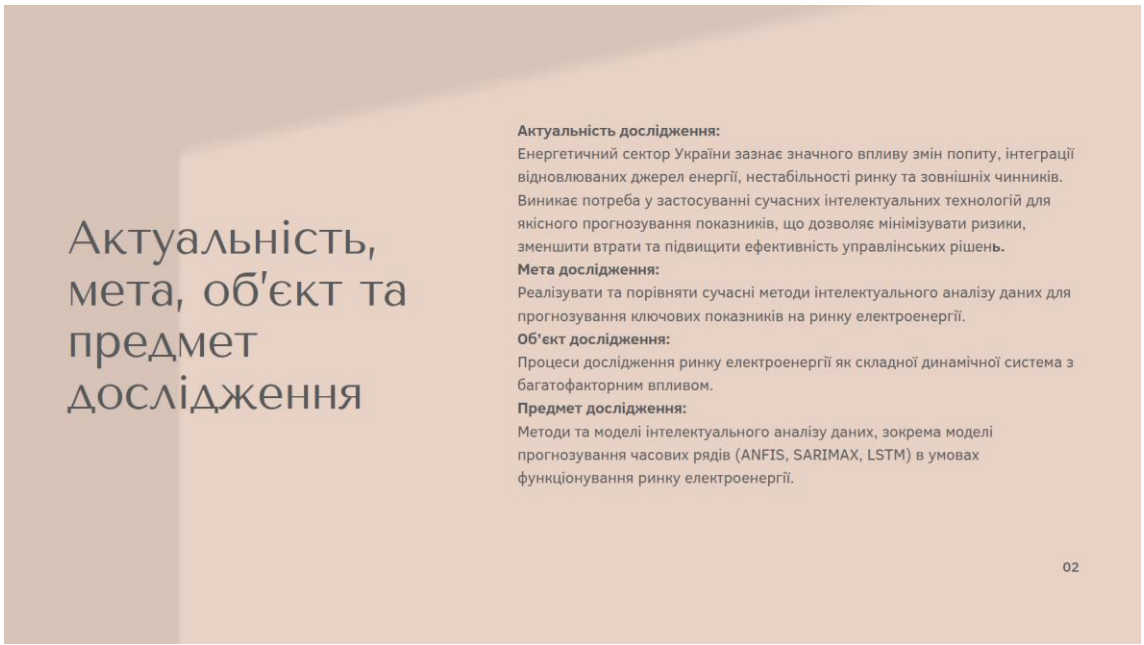
```
            #prep for layer four (bit of a hack)
            layerThree = layerTwo/wSum[pattern]
            rowHolder = np.concatenate([x*np.append(Xs[pattern,:],1) for x
in layerThree])
            layerFour = np.append(layerFour,rowHolder)

        w = w.T
        wNormalized = wNormalized.T

        layerFour = np.array(np.array_split(layerFour,pattern + 1))

        return layerFour, wSum, w


    def backprop(ANFISObj, columnX, columns, theWSum, theW, theLayerFive):

        paramGrp = [0]* len(ANFISObj.memFuncs[columnX])
        for MF in range(len(ANFISObj.memFuncs[columnX])):

            parameters = np.empty(len(ANFISObj.memFuncs[columnX][MF][1]))
            timesThru = 0
            for alpha in sorted(ANFISObj.memFuncs[columnX][MF][1].keys()):

                bucket3 = np.empty(len(ANFISObj.X))
                for rowX in range(len(ANFISObj.X)):
                    varToTest = ANFISObj.X[rowX,columnX]
                    tmpRow = np.empty(len(ANFISObj.memFuncs))
                    tmpRow.fill(varToTest)

                    bucket2 = np.empty(ANFISObj.Y.ndim)
                    for colY in range(ANFISObj.Y.ndim):

                        rulesWithAlpha =
np.array(np.where(ANFISObj.rules[:,columnX]==MF))[0]
                        adjCols = np.delete(columns,columnX)

                        senSit =
mfDerivs.partial_dMF(ANFISObj.X[rowX,columnX],ANFISObj.memFuncs[columnX][MF],
alpha)
                        # produces d_ruleOutput/d_parameterWithinMF
```

```
                        dW_dAplha = senSit *
np.array([np.prod([ANFISObj.memClass.evaluateMF(tmpRow)[c][ANFISObj.rules[r][
c]] for c in adjCols]) for r in rulesWithAlpha])

                        bucket1 = np.empty(len(ANFISObj.rules[:,0]))
                        for consequent in range(len(ANFISObj.rules[:,0])):
                            fConsequent =
np.dot(np.append(ANFISObj.X[rowX,:],1.),ANFISObj.consequents[((ANFISObj.X.sha
pe[1] + 1) * consequent):(((ANFISObj.X.shape[1] + 1) * consequent) +
(ANFISObj.X.shape[1] + 1)),colY])
                            acum = 0
                            if consequent in rulesWithAlpha:
                                acum =
dW_dAplha[np.where(rulesWithAlpha==consequent)] * theWSum[rowX]

                            acum = acum - theW[consequent,rowX] *
np.sum(dW_dAplha)
                            acum = acum / theWSum[rowX]**2
                            bucket1[consequent] = fConsequent * acum

                        sum1 = np.sum(bucket1)

                        if ANFISObj.Y.ndim == 1:
                            bucket2[colY] = sum1 * (ANFISObj.Y[rowX]-
theLayerFive[rowX,colY])*(-2)
                        else:
                            bucket2[colY] = sum1 * (ANFISObj.Y[rowX,colY]-
theLayerFive[rowX,colY])*(-2)

                    sum2 = np.sum(bucket2)
                    bucket3[rowX] = sum2

                sum3 = np.sum(bucket3)
                parameters[timesThru] = sum3
                timesThru = timesThru + 1

            paramGrp[MF] = parameters

        return paramGrp


    def predict(ANFISObj, varsToTest):
```

```
    [layerFour, wSum, w] = forwardHalfPass(ANFISObj, varsToTest)

    #layer five
    layerFive = np.dot(layerFour,ANFISObj.consequents)

    return layerFive


if __name__ == "__main__":
    print("I am main!")
```

ДОДАТОК Б ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДОПОВІДІ





Постановка задачі

Загальна постановка

У межах дипломної роботи необхідно розробити інтелектуальну систему прогнозування ключових показників на ринку електроенергії, яка:

- враховує властивості часових рядів (сезонність, тренди, аномалії);
- є здатною адаптуватися до змін у структурі даних;
- забезпечує високу якість і стабільність результатів у реальних умовах функціонування ринку;
- має потенціал для масштабування та інтеграції з іншими системами.

03

Постановка задачі

Конкретизовані задачі дослідження

1. Проаналізувати сучасні підходи до прогнозування часових рядів в енергетичній сфері.
2. Здійснити підготовку та обробку вхідних даних.
3. Реалізувати та навчити моделі прогнозування:
   - ANFIS (нейро-нечітка система),
   - SARIMAX (статистична модель),
   - LSTM (нейронна мережа довгої короткочасної пам'яті).
4. Провести порівняльний аналіз якості моделей за метриками (MAE, RMSE, R^2).
5. Розробити програмний продукт на базі Python для реалізації моделей.
6. Виконати функціонально-вартісний аналіз варіантів реалізації ПЗ.
7. Здійснити техніко-економічне обґрунтування вибору найкращого варіанту реалізації.

04

Предметна область

Енергетичний ринок України функціонує за моделлю роздрібної конкуренції та включає такі сегменти:

- Ринок двосторонніх договорів
- Ринок «на добу наперед» (РДН)
- Внутрішньодобовий ринок (ВДР)
- Балансуючий ринок
- Допоміжні послуги

Що таке ВДР?

Внутрішньодобовий ринок (ВДР) — це сегмент оптового ринку електроенергії, в якому відбувається купівля-продаж електроенергії в межах операційної доби, з метою оперативного балансування між прогнозованим і фактичним графіками споживання та генерації.

Основні характеристики ВДР

Дозволяє коригувати обсяги постачання електроенергії у відповідь на зміну попиту або пропозиції;
Торги тривають до 1 години до початку кожного розрахункового періоду (графік 24/7);
Забезпечує гнучкість і точність у відповідь на короткострокові зміни (наприклад, зниження генерації з ВДЕ або аварійні ситуації).

05

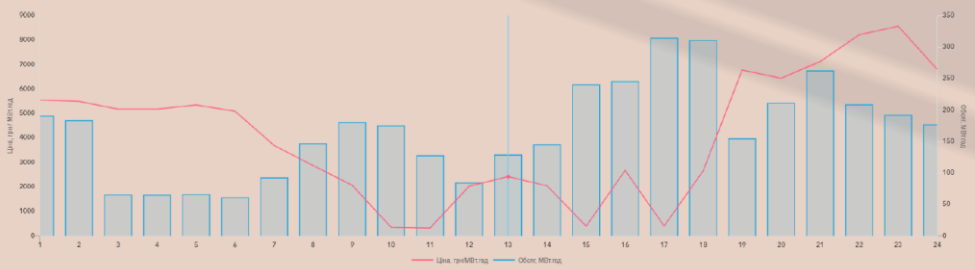
Відмінність ВДР від інших ринків

Ознака	РДН	ВДР	Балансуючий ринок
Горизонт торгів	За 1 день наперед	В межах поточної доби	Постфактум
Частота оновлення	Раз на добу	Протягом доби щогодини	Після доби
Мета	Стратегічне планування	Тактична корекція	Компенсація дисбалансів
Учасники	Всі ліцензовані учасники	Учасники з гнучкими ресурсами	Виробники/споживачі з відхиленнями
Актуальність прогнозів	Середня	**Найвища**	Фактичне відображення подій

- Умови ВДР вимагають високоточних короткострокових прогнозів із часовою дискретністю до 1 години.
- Навіть невелике відхилення прогнозу може спричинити фінансові втрати через небаланси.
- Прогнозування на ВДР є особливо складним через високу волатильність, непередбачувані погодні умови та зростаючу долю ВДЕ (відновлюваних джерел енергії).

06

Парсинг даних



Джерело даних

- Дані отримано шляхом парсингу з офіційного сайту oree.com.ua — розділ Внутрішньодобовий ринок (ВДР). Дозвіл на парсинг було оримано у офіційних представників компанії
- Оскільки таблиці на сайті формуються динамічно за допомогою JavaScript, було використано бібліотеку Selenium (Python) для автоматизації завантаження сторінок і витягу HTML-елементів.

07

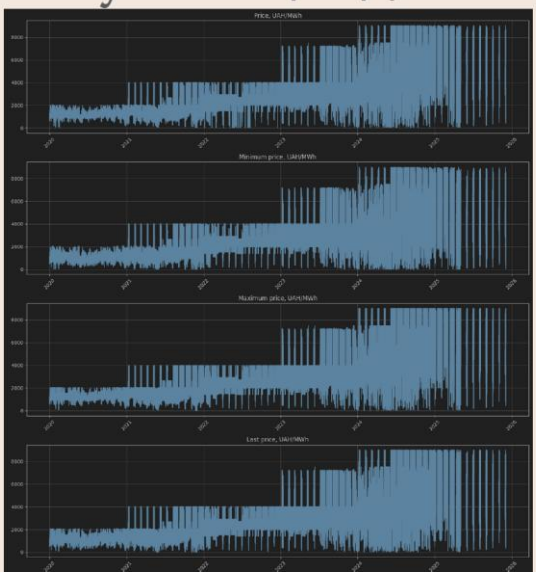
Опис набору даних

	timestamp	Price, …	Minimum price…	Maximum price…	Last price…	Sales volum…	Purchase v…	Declared…	Declared purchase …
None Count	0	None Count	0	None Count	75	None Count	0	None Count	0	None Count	0	None Count	0	None Count	0
Count	46962	Count	46962	Count	46887	Count	46887	Count	46962	Count	46962	Count	46962	Count	46962
Mean	2022-09-08 10:30:25…	Mean	2896.1	Mean	2892.4	Mean	2985.6	Mean	2843.9	Mean	392.2	Mean	392.2	Mean	1476.8	Mean	566.1
Min	2020-01-01 00:00:00	Min	0.1	Min	10.1	Min	10.0	Min	10.1	Min	0.0	Min	0.0	Min	0.0	Min	0.0
5th pctl	2020-04-07 23:33:00	5th pctl	707.4	5th pctl	520.1	5th pctl	763.7	5th pctl	453.0	5th pctl	11.4	5th pctl	11.4	5th pctl	193.2	5th pctl	12.1
25th p…	2021-05-04 06:35:00	25th pctl	1452.2	25th pctl	1259.1	25th pctl	1516.9	25th pctl	1406.0	25th pctl	87.0	25th pctl	87.0	25th pctl	472.6	25th pctl	186.7
50th p…	2022-09-08 11:30:00	50th pctl (Median)	2488.4	50th pctl (Median)	2048.2	50th pctl (Median)	2650.6	50th pctl (Median)	2321.0	50th pctl (Median)	222.2	50th pctl (Median)	222.2	50th pctl (Median)	907.0	50th pctl (Median)	361.1
75th p…	2024-01-07 16:45:00	75th pctl	4000.0	75th pctl	3900.0	75th pctl	4000.0	75th pctl	4000.0	75th pctl	546.4	75th pctl	549.4	75th pctl	1848.1	75th pctl	709.5
95th p…	2025-02-16 01:57:00	95th pctl	7138.4	95th pctl	6833.0	95th pctl	7200.0	95th pctl	6900.0	95th pctl	1211.3	95th pctl	1251.3	95th pctl	6491.3	max	9986.3
Max	2025-12-04 23:00:00	Max	9000.0	Max	9000.0	Max	9010.0	Max	9000.0	Max	6167.1	Max	6167.1	Max	25550.8	Std. Deviation	720.2
Std. Deviation		Std. Deviation	1985.0	Std. Deviation	1890.7	Std. Deviation	2010.6	Std. Deviation	1969.2	Std. Deviation	499.9	Std. Deviation	493.9	Std. Deviation	1863.8		
	46962 Unique values																
0	2020-01-01 00:00:00	693.61	637.0	720.0	637.0	33.2	33.2	2456.8	80.2
1	2020-01-01 01:00:00	774.42	650.0	959.12	655.0	39.4	39.4	3052.8	88.7
2	2020-01-01 02:00:00	759.66	650.0	959.12	655.0	36.8	36.8	3291.7	85.9
3	2020-01-01 03:00:00	707.43	650.0	740.0	650.0	26.9	26.9	3772.3	74.2
4	2020-01-01 04:00:00	708.68	650.0	740.0	740.0	26.6	26.6	3926.8	74.6
5	2020-01-01 05:00:00	710.95	650.0	720.0	700.0	49.4	49.4	3808.1	97.4
6	2020-01-01 06:00:00	712.36	650.0	720.0	720.0	43.2	43.2	3403.1	104.2
7	2020-01-01 07:00:00	692.79	650.0	650.0	650.0	34.0	34.0	2335.7	102.8
8	2020-01-01 08:00:00	976.59	995.0	995.0	995.0	116.9	116.9	1758.6	136.9
9	2020-01-01 09:00:00	1020.63	995.0	995.0	995.0	85.6	85.6	1389.7	111.6
10	2020-01-01 10:00:00	1020.1	995.0	995.0	995.0	85.8	85.8	1057.7	111.6
11	2020-01-01 11:00:00	1020.89	995.0	995.0	995.0	85.5	85.5	913.2	111.6
12	2020-01-01 12:00:00	1016.49	995.0	910.0	995.0	70.7	70.7	702.5	99.1
13	2020-01-01 13:00:00	963.11	799.0	920.0	799.0	44.7	44.7	609.1	85.0
14	2020-01-01 14:00:00	1045.64	873.0	998.0	870.0	56.5	56.5	349.3	85.5

- timestamp — мітка часу з точністю до години
- Price — погодинна ціна на ВДР (грн/МВт·год)
- Minimum price, Maximum price, Last price — статистичні межі ціни
- Sales volume, Purchase volume — фактичні обсяги купівлі-продажу (МВт·год)
- Declared sales, Declared purchases — заявлені обсяги купівлі та продажу

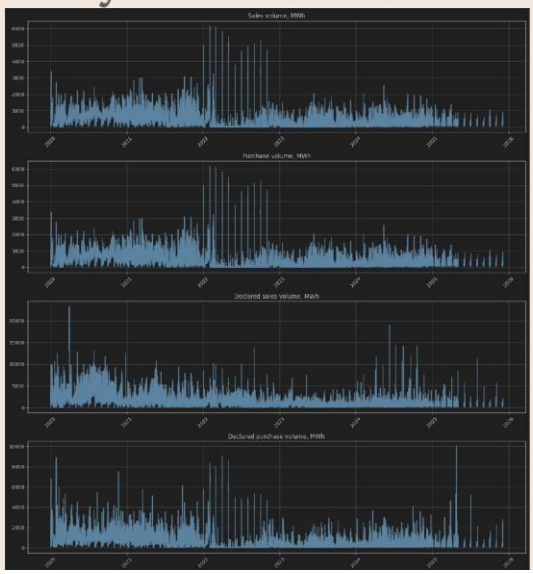
08

Візуалізація даних



09

Візуалізація даних



10

Передобробка даних та розбиття Train/Test

Заповнення пропусків

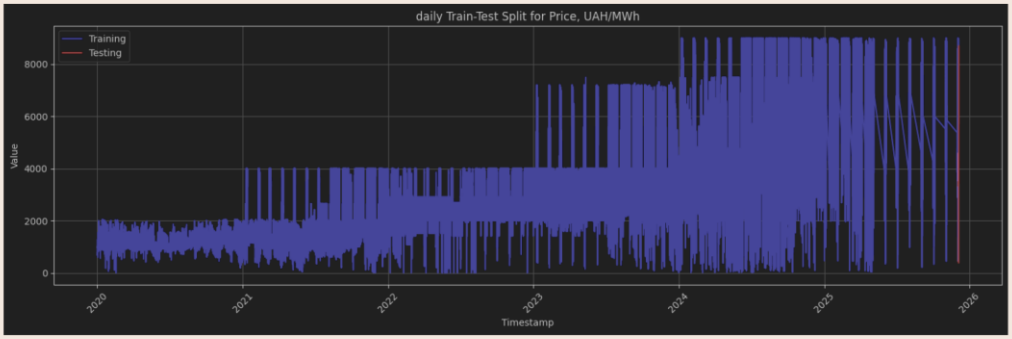
- Пропуски у часових рядах заповнювались методом лінійної інтерполяції (pandas.DataFrame.interpolate()), що дозволяє зберегти часову структуру даних.
- Інтерполяція застосовувалась окремо до кожного числового стовпця (ціна, обсяги тощо).

Розбиття даних

- Використано кілька стратегій Train/Test Split залежно від сценарію прогнозування:
  - Прогноз на 1 день вперед
  - Прогноз на 1 тиждень
  - Прогноз на 1 місяць
- Усі розбиття виконано хронологічно — з дотриманням послідовності часового ряду без перемішування даних.
- Тестова вибірка завжди містила останні значення в часовому ряді (як видно з графіків на наступних слайдах), а тренувальна — всю попередню історію.

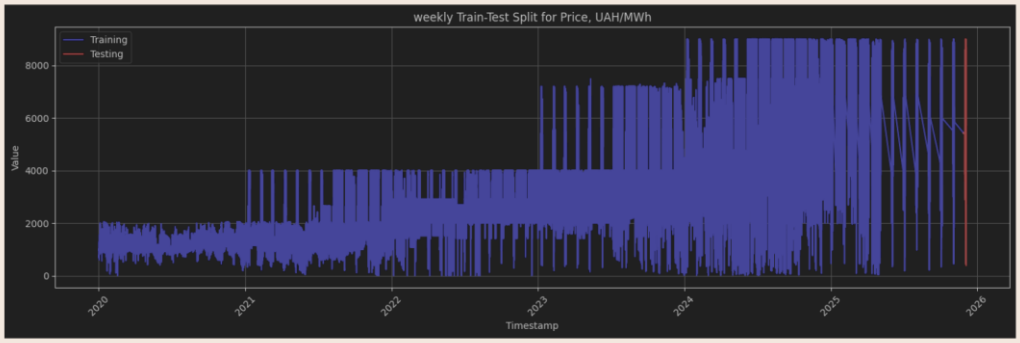
11

Розбиття Test = "day" для "Ціни"



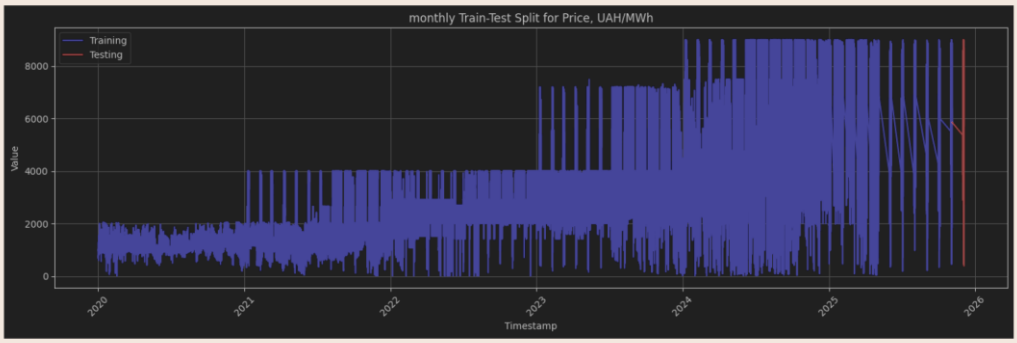
12

Розбиття Test = “week" для “Ціни"



13

Розбиття Test = “month" для “Ціни"



14

Дослідження стаціонарності, автокореляції та декомпозиція даних

Дослідження стаціонарності часових рядів

- ADF-тесту (Augmented Dickey-Fuller) - Перевіряє гіпотезу наявності одиничного кореня (нестійкості)
- KPSS-тесту (Kwiatkowski-Phillips-Schmidt-Shin) - Перевіряє гіпотезу стаціонарності відносно середнього

Автокореляційний аналіз (ACF/PACF)

- ACF (автокореляційна функція) — показує кореляцію з лагами
- PACF (часткова автокореляція) — виявляє лаги з незалежним впливом

Декомпозиція часових рядів

- Trend — загальний напрямок змін
- Seasonal — періодичність (добова, тижнева)
- Residual — залишкова варіація

Отримані результати враховувались при побудові моделей SARIMAX (визначення параметрів p, d, q, P, D, Q) та при формуванні вікон для LSTM/ANFIS.

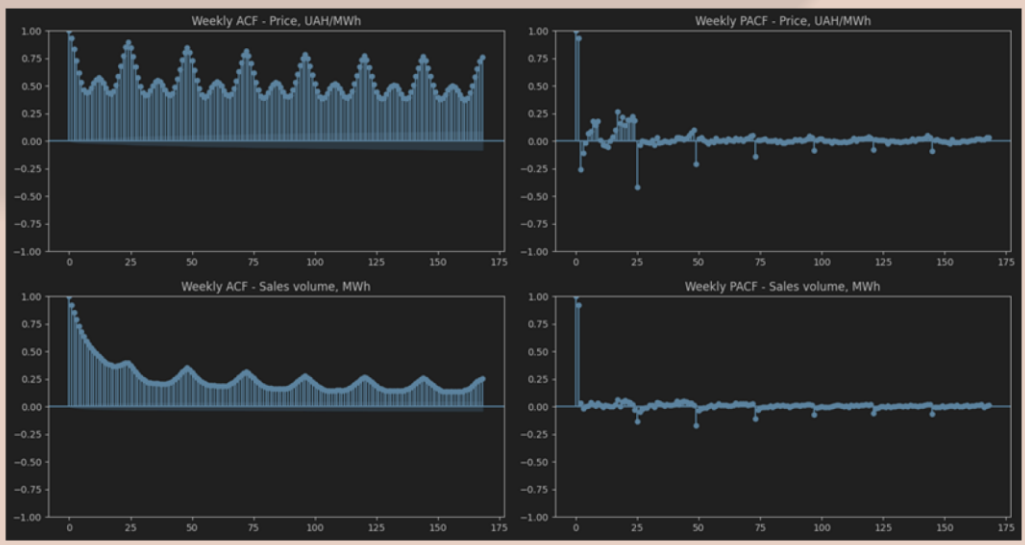
15

Результати тестів на стаціонарність

	ADF p-value	KPSS p-value	ADF Stationary	KPSS Stationary
Price, UAH/MWh	2.125490e-09	0.01	True	False
Minimum price, UAH/MWh	4.058344e-11	0.01	True	False
Maximum price, UAH/MWh	3.529609e-09	0.01	True	False
Last price, UAH/MWh	1.202827e-09	0.01	True	False
Sales volume, MWh	2.583849e-30	0.01	True	False
Purchase volume, MWh	2.583849e-30	0.01	True	False
Declared sales volume, MWh	3.007241e-25	0.01	True	False
Declared purchase volume, MWh	4.392347e-29	0.01	True	False

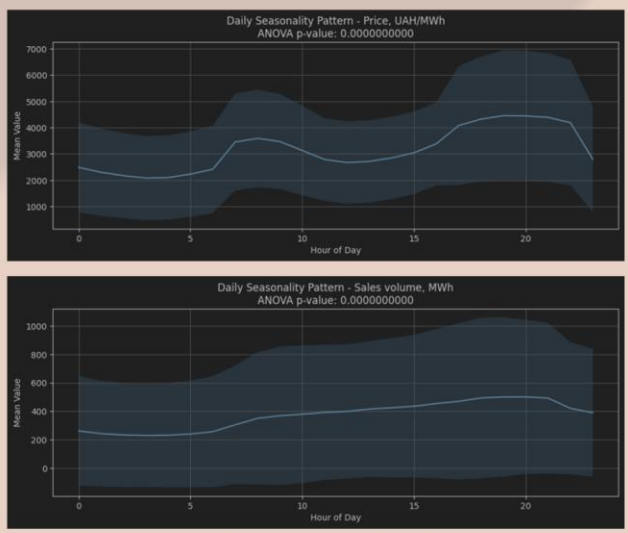
16

ACF/PACF для Ціни та Обсягу



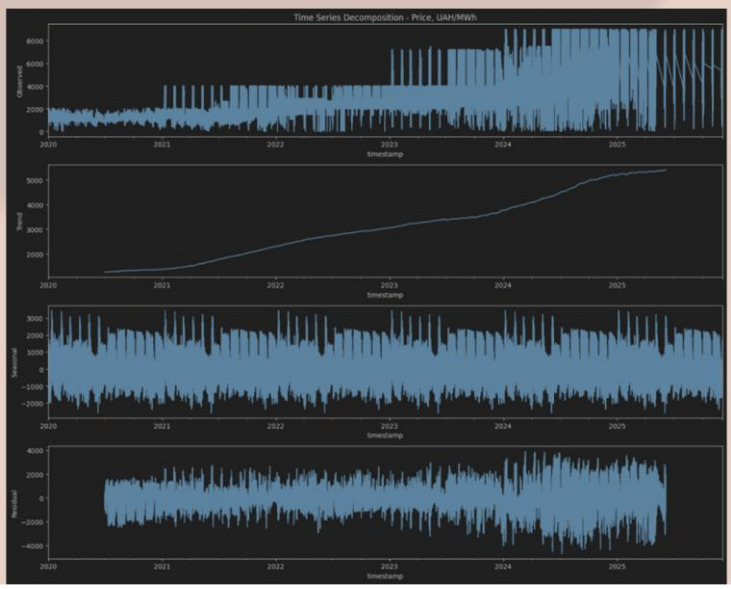
17

Щоденний патерн сезонності



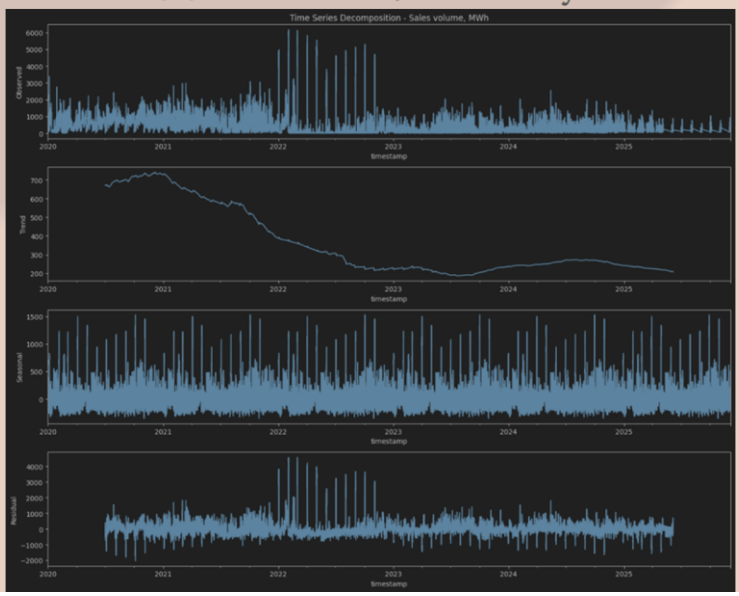
18

Декомпозиція Ціни



19

Декомпозиція Обсягу



20

Використані моделі прогнозування

У дипломній роботі досліджено та реалізовано три підходи до прогнозування часових рядів:

1. SARIMAX
2. LSTM (Long Short-Term Memory)
3. ANFIS (Adaptive Neuro-Fuzzy Inference System)

Кожна з моделей обрана з огляду на її здатність відображати різні характеристики часових рядів — сезонність, нелінійність, стохастичність, нечіткість.

Модель SARIMAX

- Гіперпараметри моделі:
  - order = (24, 0, 24)
  - seasonal_order = (24, 2, 24, 168)
- Реалізація:
  - Для кожної цільової змінної та кожного варіанту розбиття (1 день / 1 тиждень / 1 місяць) пбудова окремої SARIMAX-моделі
- Переваги:
  - Врахування сезонності та високочастотних компонент
- Обмеження:
  - Обмежена здатність до виявлення складних нелінійностей у даних

Модель LSTM

- Передобробка:
  - Масштабування даних за допомогою MinMaxScaler
  - Побудова послідовностей зі змінною довжиною (24×7, 24×14)
- grid_search:
  - n_lstm_units: 50, 100
  - n_lstm_layers: 1, 2
  - dropout_rate: 0.1, 0.2
  - learning_rate: 0.001, 0.0001
- Архітектура:
  - Стековані LSTM-шари з Dropout
  - Вихідний Dense-шар
- Навчання:
- 50 enox, batch size = 32

Модель ANFIS

- Передобробка:
  - Масштабування даних (MinMaxScaler)
  - Формування вікон фіксованого розміру (n_steps)
- grid_search:
  - n_rules: 20, 50, 100, 150, 200, 250
  - n_steps: 24 (1 день), 168 (1 тиждень), 720 (1 місяць)
  - epochs: 10, 50, 100
- Особливості:
  - Побудова нечітких множин для кожної ознаки
  - Навчання через trainHybridJangOffLine

21

Метрики оцінювання якості моделей і прогнозів

MSE

Середнє квадратичне відхилення між фактичними та прогнозованими значеннями.

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2$$

RMSE

Корінь із MSE — має ті ж одиниці, що й прогнозована змінна.

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

MAE

Середнє абсолютне відхилення між реальними та прогнозованими значеннями.

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

R^2

Відображає частку дисперсії, пояснену моделлю.

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{n}(y_t - \bar{y}_t)^2}$$

Всі метрики розраховувались окремо для кожної моделі (SARIMAX, LSTM, ANFIS) та для кожного варіанту розбиття: на 1 день, 1 тиждень, 1 місяць.

22

Технології прогнозування

Загальні інструменти

- Python 3.10 — основна мова програмування
- Jupyter Notebook / DataSpell — середовища для інтерактивної розробки, тестування та візуалізації

Оцінка якості моделей

- sklearn.metrics — метрики MSE, MAE, RMSE, R^2
- MinMaxScaler (з sklearn.preprocessing) — масштабування ознак для нейромереж

Моделі та алгоритми

- SARIMAX:
  - Бібліотека statsmodels.tsa.statespace.sarimax
  - Побудова сезонних ARIMA-моделей із підтримкою зовнішніх регресорів
- LSTM:
  - Фреймворк TensorFlow + Keras
  - Побудова багатошарових рекурентних нейронних мереж
- ANFIS:
  - Локальна бібліотека anfis (від twmeggs, модифікована)
  - Реалізація адаптивної нечіткої системи виведення

Обробка та аналіз даних

- pandas, numpy — маніпуляції з табличними даними та числові операції
- matplotlib, seaborn — побудова графіків, візуалізація результатів

Дані отримані через автоматизований парсинг (Selenium) з [oree.com.ua], з подальшою обробкою в pandas.

23

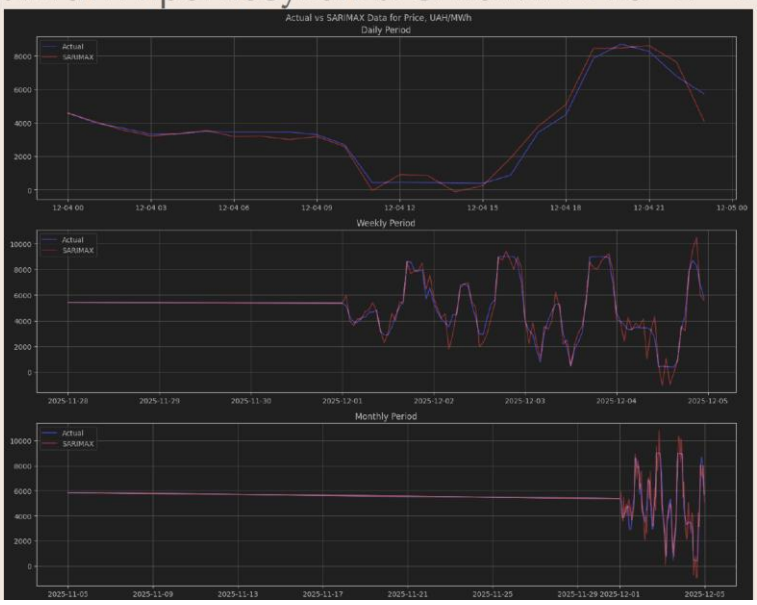
Результати прогнозування моделей

Модель SARIMAX

SARIMAX	MSE	RMSE	MAE	R2
Price, UAH/MWh - daily	283318.512	532.2767	383.7267	0.9519
Price, UAH/MWh - weekly	323233.9457	568.5367	324.3918	0.9074
Price, UAH/MWh - monthly	153401.3232	391.6648	117.6648	0.8252
Minimum price, UAH/MWh - daily	159459.8556	399.3242	296.742	0.9683
Minimum price, UAH/MWh - weekly	221149.3457	470.2652	274.2424	0.9431
Minimum price, UAH/MWh - monthly	94330.8217	307.1332	93.0423	0.8996
Maximum price, UAH/MWh - daily	123190.7421	350.9854	274.4554	0.9805
Maximum price, UAH/MWh - weekly	266213.2163	515.9585	297.1086	0.9204
Maximum price, UAH/MWh - monthly	142665.9762	377.7115	113.4753	0.8268
Last price, UAH/MWh - daily	264505.3754	514.3009	381.3035	0.9563
Last price, UAH/MWh - weekly	299998.3552	547.7211	318.5416	0.9182
Last price, UAH/MWh - monthly	152562.755	390.5928	114.8453	0.8324
Sales volume, MWh - daily	1442.4408	37.9795	27.2269	0.895
Sales volume, MWh - weekly	434.7344	20.8503	12.386	0.9837
Sales volume, MWh - monthly	184.0264	13.5656	3.8913	0.98
Purchase volume, MWh - daily	701.0073	26.4765	18.1312	0.949
Purchase volume, MWh - weekly	640.8043	25.3141	15.2494	0.976
Purchase volume, MWh - monthly	190.9591	13.8188	4.0646	0.9792
Declared sales volume, MWh - daily	7964.9823	89.2467	61.3978	0.9743
Declared sales volume, MWh - weekly	6038.5213	77.7079	47.0872	0.9258
Declared sales volume, MWh - monthly	1452.4439	38.1109	11.1977	0.9258
Declared purchase volume, MWh - daily	1522.2159	39.0156	28.163	0.9083
Declared purchase volume, MWh - weekly	1046.6964	32.3527	19.0916	0.99
Declared purchase volume, MWh - monthly	291.6196	17.0769	4.997	0.9895

24

Результати прогнозування SARIMAX на ЧР "Ціна"



25

Результати прогнозування SARIMAX на ЧР "Обсяг"

26

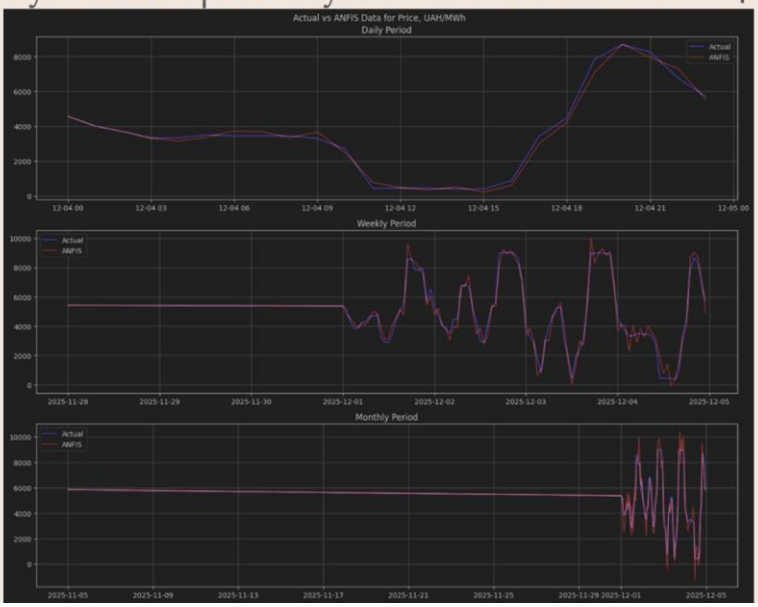
Результати прогнозування моделей

Модель ANFIS

ANFIS	MSE	RMSE	MAE	R2
Price, UAH/MWh - daily	76475.4109	276.5419	209.1821	0.987
Price, UAH/MWh - weekly	151421.7106	389.1294	238.8468	0.9566
Price, UAH/MWh - monthly	77875.1	279.0611	84.1952	0.9113
Minimum price, UAH/MWh - daily	139604.4536	373.6368	250.802	0.9723
Minimum price, UAH/MWh - weekly	156060.2415	395.0446	242.0861	0.9598
Minimum price, UAH/MWh - monthly	64739.2599	254.4391	73.2886	0.9311
Maximum price, UAH/MWh - daily	128539.0448	358.5234	266.3431	0.9796
Maximum price, UAH/MWh - weekly	238617.7097	488.4851	295.6158	0.9286
Maximum price, UAH/MWh - monthly	75558.1553	274.8784	80.0485	0.9083
Last price, UAH/MWh - daily	82312.6037	286.9017	191.4109	0.9864
Last price, UAH/MWh - weekly	183763.9059	428.6769	257.0019	0.9499
Last price, UAH/MWh - monthly	81045.5809	284.6851	82.8851	0.911
Sales volume, MWh - daily	356.9469	18.893	13.8023	0.974
Sales volume, MWh - weekly	273.5986	16.5408	9.7928	0.9897
Sales volume, MWh - monthly	112.121	10.5887	3.1765	0.9878
Purchase volume, MWh - daily	393.8065	19.8446	13.8731	0.9713
Purchase volume, MWh - weekly	267.9757	16.37	9.8469	0.99
Purchase volume, MWh - monthly	153.7669	12.4003	3.4532	0.9833
Declared sales volume, MWh - daily	4490.6819	67.0126	47.8242	0.9855
Declared sales volume, MWh - weekly	2755.5789	52.4936	30.4923	0.9661
Declared sales volume, MWh - monthly	1133.5864	33.6688	9.305	0.9421
Declared purchase volume, MWh - daily	937.3606	30.6163	22.4284	0.9436
Declared purchase volume, MWh - weekly	520.9687	22.8247	13.5735	0.995
Declared purchase volume, MWh - monthly	187.3756	13.6885	3.9483	0.9933

27

Результати прогнозування ANFIS на ЧР "Ціна"



28

Результати прогнозування ANFIS на ЧР "Обсяг"

29

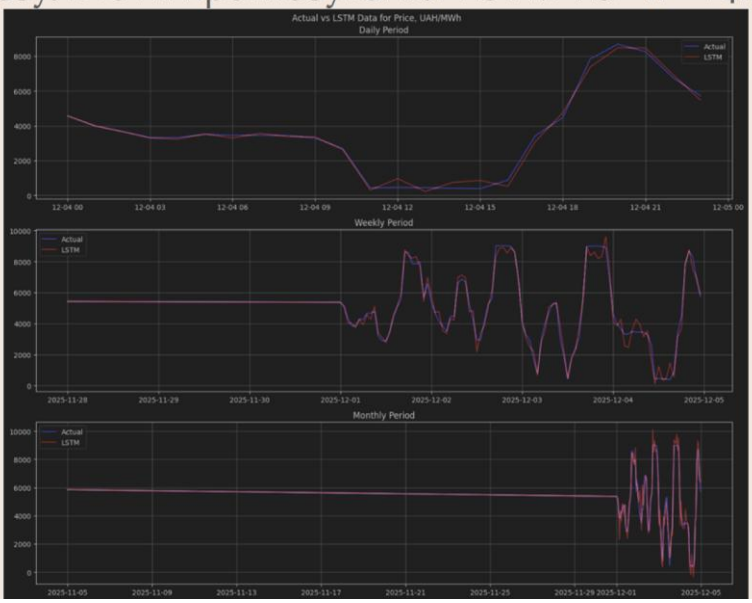
Результати прогнозування моделей

Модель LSTM

LSTM	MSE	RMSE	MAE	R2
Price, UAH/MWh - daily	59478.2433	243.8816	190.7141	0.9899
Price, UAH/MWh - weekly	101042.1319	317.8713	192.2553	0.971
Price, UAH/MWh - monthly	56723.0023	238.1659	70.7551	0.9354
Minimum price, UAH/MWh - daily	62473.8556	249.9477	166.6242	0.9876
Minimum price, UAH/MWh - weekly	108724.3592	329.7338	194.872	0.972
Minimum price, UAH/MWh - monthly	39855.7415	199.639	59.5082	0.9576
Maximum price, UAH/MWh - daily	61449.6439	247.8904	175.5946	0.9903
Maximum price, UAH/MWh - weekly	120025.1069	346.4464	208.1277	0.9641
Maximum price, UAH/MWh - monthly	59669.0188	244.2724	69.8671	0.9276
Last price, UAH/MWh - daily	63824.0222	252.6342	179.8226	0.9895
Last price, UAH/MWh - weekly	114778.5281	338.7898	194.8556	0.9687
Last price, UAH/MWh - monthly	41740.1772	204.3041	55.8585	0.9541
Sales volume, MWh - daily	410.0568	20.2499	14.5517	0.9702
Sales volume, MWh - weekly	162.6298	12.7526	7.5893	0.9939
Sales volume, MWh - monthly	83.4912	9.1374	2.7399	0.9909
Purchase volume, MWh - daily	690.9482	26.2859	18.6916	0.9497
Purchase volume, MWh - weekly	167.1954	12.9304	7.6431	0.9937
Purchase volume, MWh - monthly	87.5639	9.3576	2.7187	0.9905
Declared sales volume, MWh - daily	2237.9791	47.3073	34.3353	0.9928
Declared sales volume, MWh - weekly	1614.1652	40.1767	23.3982	0.9802
Declared sales volume, MWh - monthly	688.6796	26.2427	7.3234	0.9648
Declared purchase volume, MWh - daily	525.0472	22.9139	16.0595	0.9684
Declared purchase volume, MWh - weekly	349.508	18.6951	10.5481	0.9967
Declared purchase volume, MWh - monthly	88.4196	9.4032	2.7756	0.9968

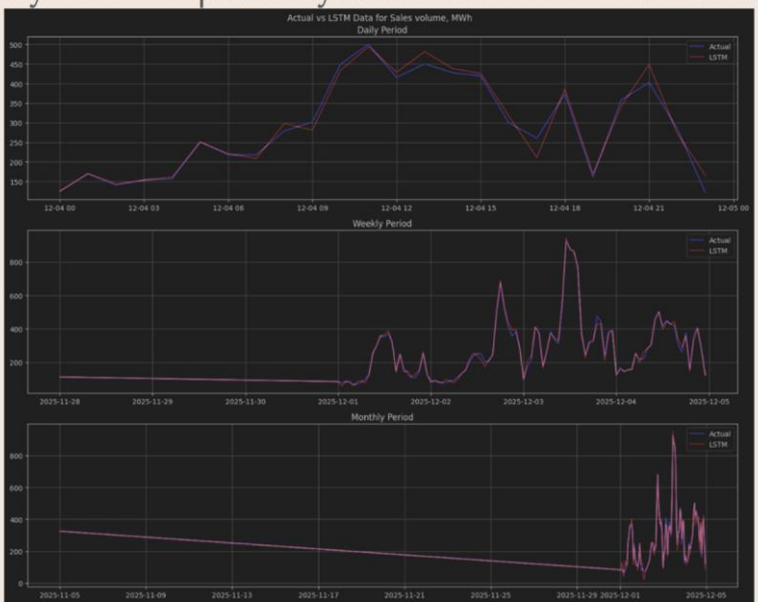
30

Результати прогнозування LSTM на ЧР "Ціна"



31

Результати прогнозування LSTM на ЧР "Обсяг"



32

Висновки по результатах прогнозування

- LSTM показала найкращу якість для короткострокових прогнозів (добових та тижневих).
- SARIMAX добре працює на добових прогнозах, але якість падає для довших інтервалів (місячних).
- ANFIS демонструє хороші результати для обсягів продажу та мінімальних цін, поєднуючи адаптивність та інтерпретованість.
- Якість моделей знижується при прогнозуванні на місячний період через більшу невизначеність даних.
- Рекомендовано розглядати комбінування моделей для підвищення якості прогнозів.

33

Висновки

- Проведено детальне вивчення та аналіз предметної області ринку електроенергії, що дозволило врахувати ключові особливості та фактори, які впливають на динаміку показників.
- Виконано комплексну підготовку даних: очищення, інтерполяція пропусків, розбиття на тренувальну та тестову вибірки для різних часових інтервалів (день, тиждень, місяць).
- Проведено детальний аналіз часових рядів із виявленням сезонності, трендів та аномалій, що сприяло вибору ефективних моделей прогнозування.
- Розроблено та протестовано три підходи до прогнозування ключових показників ринку електроенергії: SARIMAX, LSTM та ANFIS.
- Найвища якість прогнозування досягнута моделлю LSTM, особливо для короткострокових інтервалів.
- Кожна модель має свої сильні сторони: SARIMAX – інтерпретованість, ANFIS – баланс між якістю та прозорістю, LSTM – здатність виявляти складні закономірності.
- Якість прогнозів знижується при збільшенні горизонту прогнозування через зростання невизначеності та складності ринку.
- Результати дослідження можуть бути використані для підвищення ефективності прийняття рішень на ринку електроенергії.
- Подальші напрямки роботи: комбінування моделей і врахування додаткових факторів для покращення якості прогнозування.

34


Дякую за увагу!