Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»
Навчально-науковий інститут електроенергетики
(інститут)
Електротехнічний факультет
(факультет)
Кафедра кіберфізичних та інформаційно-вимірювальних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню магістра

здобувача вищої освіти Рульов Євгеній Ігорович
(П.І.Б.)

академічної групи 151м-22-1
(шифр)

спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
(код і назва спеціальності)

за освітньо-професійною програмою 151 Автоматизація та комп'ютерно-інтегровані технології
(офіційна назва)

на тему Дослідження та розробка кіберфізичної системи прогнозування потужності фотоелектричної установки
(назва за наказом ректора)

Консультанти	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинг.	інституційною	
Керівник кваліфікаційної роботи	проф. Бубліков А.В.			
Провідний консультант	ас. Зибалов Д.С.			
Теоретичний розділ	проф. Бубліков А.В.			
Дослідження та розробка кібер-фізичної системи	проф. Бубліков А.В			
Експериментальний розділ	проф. Бубліков А.В			
Економічна частина	ст. викл. Яремчук І.О.			
Охорона праці	проф. Чеберячко Ю.І.			

Нормо контролер	ас. Славінський Д.В.			

Дніпро
2023

ЗАТВЕРДЖЕНО:
завідувачем кафедри
кіберфізичних та інформаційно-вимірювальних систем
(повна назва)
_____ Бубліковим А.В.
(підпис) (прізвище, ініціали)

«_____»_____2023 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістра

здобувача вищої освіти Рульов Є.І. **академічної групи** 151м-22-1
(прізвище та ініціали) (шифр)

спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

за освітньо-професійною програмою 151 Автоматизація та комп'ютерно-інтегровані технології
(офіційна назва)

на тему Автоматизація процесів керування сонячною фотоелектричною установкою

затверджену наказом ректора НТУ «Дніпровська політехніка» від _____ № _____.

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	Вступ. Опис технологічного процесу для об'єкта автоматизації. Огляд існуючих систем автоматизації. Науково-технічне завдання, конкретизація предмету та методу дослідження.	16.10.2023
Теоретичний розділ	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу.	23.10.2023
Дослідження та розробка кібер-фізичної системи	Дослідження моделей прогнозування. Аналіз часового ряду та екзогенних параметрів. Пошук залежності даних. Побудова моделей.	30.10.2023
Експериментальний розділ	Впровадження системи в об'єкт керування. Опис технічних вимог. Удосконалення програмного забезпечення.	20.11.2023
Економічна частина	Економічне обґрунтування доцільність витрат на розробку кіберфізичної системи.	27.11.2023
Охорона праці	Розробка організаційно-технічних заходів, щодо реалізації правил безпеки при експлуатації системи.	29.11.2023

Завдання видано _____ проф, Бубліков А.В.
(підпис п.конс.) (прізвище, ініціали)

Дата видачі 01.09.2023

Дата подання до атестаційної комісії

Прийнято до виконання _____ Рульов Є.І.
(підпис здобувача) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить: ____ стор., 81 рис., 4 табл., 4 додат., 17 джерел.

Об’єкт дослідження: Система прогнозування потужності фотоелектричної установки.

Мета: Дослідження та розробка ефективної системи прогнозування потужності сонячної електростанції.

На основі розробленого в дипломі бакалавра прототипу, зібрана необхідна інформація для проводження дослідження. Проведений аналіз ринку, та існуючих систем прогнозування часових рядів. Висунуті теоретичні відомості та потенційно ефективні методи прогнозування відповідно до існуючого об’єкту дослідження.

На базі теоретичної основи розроблені методи прогнозування, результати досліджень та експериментів спрямовані на визначення точності, ефективності і надійності розроблених систем.

В рамках роботи була проведена детальна оцінка ефективності інноваційних технічних досягнень, визначаючи їхні переваги та внесок у загальну функціональність системи. У ході аналізу сучасних методів обробки складних програм відбулася валідація певних гіпотез, що стосуються ефективності впровадження цих методів у конкретному дослідженні та об'єкті. Результати розкривають потенційні області для подальшого удосконалення.

ПРОГНОЗУВАННЯ, ПОТУЖНІСТЬ, НЕЙРОМЕРЕЖІ, CUDA, RNN, LSTM, ARMA, ЧАСОВИЙ РЯД, ФОТОЕЛЕКТРИЧНІ МОДУЛІ,

ABSTRACT

The explanatory note contains: ____ pages, 81 figures, 4 tables, 4 appendices, 17 sources.

The object of the study: System of forecasting the power of a photovoltaic installation.

Purpose: Research and development of an effective system for forecasting the power of a solar power plant.

On the basis of the prototype developed in the bachelor's diploma, the necessary information for conducting the research was collected. An analysis of the market and existing time series forecasting systems was carried out. Theoretical information and potentially effective forecasting methods are put forward in accordance with the existing research object.

Forecasting methods have been developed on the basis of the theoretical basis, the results of research and experiments are aimed at determining the accuracy, efficiency and reliability of the developed systems.

As part of the work, a detailed assessment of the effectiveness of innovative technical achievements was carried out, determining their advantages and contribution to the overall functionality of the system. In the course of the analysis of modern methods of processing complex programs, the validation of certain hypotheses concerning the effectiveness of the implementation of these methods in a specific study and object took place. The results indicate confirmation of some hypotheses and reveal potential areas for further improvement.

FORECASTING, POWER, NEURAL NETWORKS, CUDA, RNN, LSTM, ARMA, TIME SERIES, PHOTOELECTRIC MODULES,

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

СЕС	Сонячна електростанція
ТЕЦ	Теплова електростанція
АЕС	Атомна електростанція
IPxx	Ingress Protection Code
USB	Universal Serial Bus
SPI	Serial Peripheral Interface
UTC	Всесвітній координований час
ARMA	Autoregressive–moving-average models
ARIMA	Autoregressive integrated moving average models
RNN	Recurrent neural network
LSTM	Long short-term memory
CSV	Comma-separated values
SSE	Streaming SIMD Extensions
CUDA	Compute Unified Device Architecture
GPU	Graphics processing unit
CPU	Central processing unit
DDR	Double Data Rate

ВСТУП

Сонячна енергетика стрімко збільшує свої темпи розповсюдження. Якщо у 2019 році ринок фотоенергетики виріс на 44%. То у 2022 на 56%. Фото енергетична галузь продовжує домінувати в нових глобальних потужностях по видобутку електроенергії, зайнявши перше місце серед відновлювальних джерел енергії. За дослідженням SPE (Solar Power Europe) була розкрита наступав економічна ситуація.

Всього за 2022 рік було вироблено 362 ГВт енергії від відновлювальних джерел. З них 239 ГВт, що становить 66% від усіх видів, видобули сонячні електростанції. Це підкреслює зростаюче значення сонячної енергетики у глобальному енергетичному переході, встановлюючи приблизно вдвічі більше потужності, ніж інші відновлювані технології разом узяті.

За останні роки спостерігається загальна тенденція до збільшення потужностей відновлювальних джерел енергії, що обумовлено глобальним енергетичним кризисом і підтримкою з боку політиків. Загальний зріст потужностей склав 56 ГВт, що вище, ніж у попередньому році, і перевищує 306 ГВт у 2021 році на 18%. Збільшення потужностей сонячної енергії в 2022 році компенсувало зменшення щорічних встановлень вітрової енергії: від 95,3 ГВт у 2020 році до 93,6 ГВт у 2021 році та 77,6 ГВт у 2022 році.

Винятковий успіх сонячної енергетики можна пояснити різними чинниками, але ключовим є значне зниження витрат протягом останнього десятиліття, що дозволило їй стати світовим лідером з погляду конкурентоспроможності витрат. Однак, незважаючи на вражаючі темпи зростання сонячної енергетики, майже на 19%, за 3 місяці 2023 року, переважають усі інші технології виробництва електроенергії, частка сонячної енергетики у світовому виробництві електроенергії нині становить лише 4%. Необхідні більш амбітні підходи в усьому світі, щоб використовувати гігантський потенціал енергії набагато швидше, оскільки час у гонці за виконанням Паризької угоди щодо клімату спливає.

Основними проблемами що до масового розповсюдження є:

- Малий коефіцієнт корисної дії. Сучасні фотогальванічні модулі мають близько 20-25% ККД.
- Не ефективне використання сонячного дня, а системи трекінгу занадто дорогі для великих СЕС.
- Велика ціна систем зберігання енергії.
- Залежність від метео умов.
- Складність балансування великих енергосистем, в яких присутні інші, набагато стабільніші джерела енергії. Такі як теплові або атомні електростанції.

Якщо проблеми з малим коефіцієнтом корисної дії та системами зберігання с кожним роком вирішуються, завдяки покращенню існуючих технологій та розробці нових. Ємність, ресурс акумуляторів збільшується, а ціна за кіловат годину зменшується. То інші проблеми більш суттєві. У попередній роботі було висунуте рішення проблеми, що до не ефективного використання сонячного дня, завдяки впровадженню одно осьової системи позиціонування. На основі мережевої енергетичної системи. Завданням цієї роботи є дослідження, аналіз та розробка методу прогнозування потужності сонячної електростанції.

Це дозволить:

- Спрогнозувати вплив екзогенних параметрів на систему.
- Оптимізувати виробництво електроенергії. Прогнозування потужності дозволяє оптимізувати виробництво сонячної електростанції. Відомі прогнози дозволяють виробляти електроенергію в максимальних обсягах та уникати перевиробництва.
- Планувати резервні ресурси. Прогнози потужності допомагають у плануванні використання ресурсів, таких як паливо або запаси, які можуть бути використані відповідно до очікуваного виробництва енергії.
- Зменшити витрати на підтримку. Оптимізувати роботу обладнання, зменшити знос і збільшити ефективність, що може призвести до зниження витрат на підтримку.
- Зменшити навантаження на мережу та спростити її балансування.

- Легше інтегрувати та співпрацювати з іншими видами електростанцій у великій енергосистемі.
- Збільшити надійність енергозабезпечення.

Таким чином робота направлена на вирішення проблеми складності балансування, оптимізації роботи мережевої електростанції. Так як відсутня система зберігання, а отже вплив на стабільність енергомережі багатократно збільшується.

Окрім цього потрібно порівняти найпопулярніші моделі прогнозування, адже необхідне найкраще рішення. Яке зможе забезпечити велику точність але й при цьому мати низьку собівартість впровадження, легкість експлуатації та потребувати оптимальну кількість обчислювальних ресурсів.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Галузь промисловості

Галузь енергетики є ключовим компонентом інфраструктури кожної країни та її соціально-економічного розвитку. Забезпечення надійного та стабільного електропостачання має вирішальне значення для забезпечення потреб населення, функціонування сфери послуг, промисловості, транспорту, комунікацій і багатьох інших аспектів життя. У цьому контексті сонячні електростанції стають важливим чинником в енергетичному ландшафті, надаючи чисту та відновлювальну енергію для задоволення постійно зростаючих потреб суспільства.

Україна, як одна з країн з великим сонячним потенціалом, стала свідком активного розвитку сонячних електростанцій у своїй енергетичній системі. Здійснюючи перехід до використання сонячної енергії, Україна робить крок у напрямку зменшення відчутного впливу на навколишнє середовище і забезпечення сталої енергетичної незалежності.

У 2022 році Україна увійшла в топ-20 країн світу з найбільшим парком сонячної енергії із загальною встановленою потужністю 7,7 ГВт. Великі наземні сонячні електростанції займають значну частку з встановленою потужністю 6,2 ГВт, тоді як установки на комерційних і житлових будівлях досягають 1,5 ГВт. У 2023 році українські компанії планували побудувати потужності 400-500 МВт, однак майже всі проекти були заморожені. Цього року галузь може реалізувати лише кілька проектів комерційного та житлового власного споживання на Заході України. Це викликає питання щодо майбутнього галузі промисловості та його можливих наслідків для енергетичного розподілу Європи.

Але ситуація у світі продовжує стрімко збільшувати масштаб та розвиток галузі. Китай, безперечно, найбільший у світі ринок сонячної енергії, вийшов на новий рівень з річним ринком, близьким до 100 ГВт, що на 72% більше, ніж 54,9 ГВт, встановлених у 2021 році, що вже було рекордним показником. Крім того, позитивна динаміка спостерігалася на численних ринках сонячної енергетики в

усьому світі, причому помітний прогрес був відзначений у Європі, Бразилії та Індії.

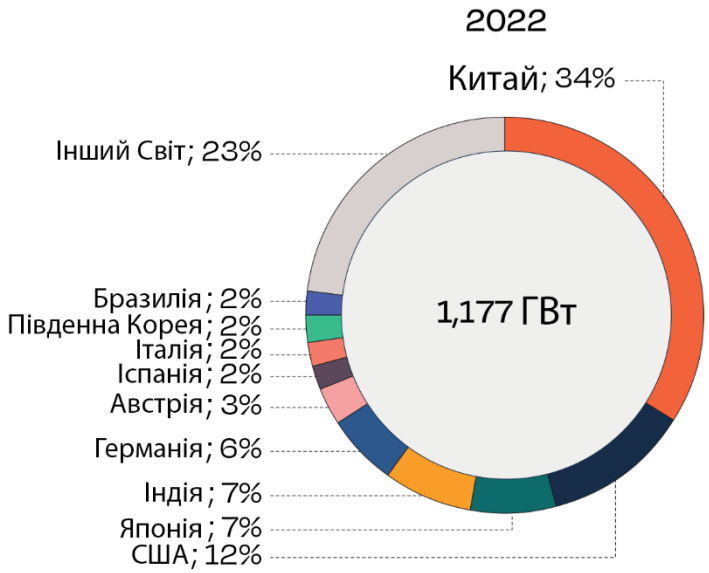


Рисунок 1.1 – Розподіл ринку СЕС

Україна щоб обійти кризис галузі, займається будівництвом та організацією роботи СЕС за кордоном. Так компанія «ДТЕК» зосередилися на будівництві сонячних проектів в Італії, Іспанії та Португалії, а також вітрових електростанцій у Скандинавії.

До початку військово-соціального кризису лідерами галузі були:

- Дніпропетровська область – 290 МВт.
- Одеська область – 240МВт.
- Вінницька область – 230МВт.
- Херсонська область – 100МВт.

Галузь сонячної енергетики під час конфлікту в Україні виявилася важливим і перспективним компонентом енергетичного сектору. Ріст цієї галузі під час війни помітний у приватних господарств. Це обумовлено поганим, а іноді відсутнім енергозабезпеченням. Переваги СЕС залишаються навіть у момент політичної тривоги:

Резерви та потенціал: Україна має значний потенціал для розвитку сонячної енергетики завдяки великій кількості сонячних днів і велику пощадь не займаних земель.

Аварійне живлення: Системи на базі сонячних панелей дозволяють безшумно та не потребуючи палива отримати електроенергію для особливо важливих систем діяльності людини. За наявності певних конструктивних рішень, можуть отримати не погану мобільність.

Економічність та незалежність: Використання сонячної енергії дозволяє зменшити енергозалежність, оскільки сировина для виробництва сонячних панелей є загалом доступною. Це сприяє забезпеченню сталої роботи сонячних електростанцій навіть в умовах конфлікту.

Можливості для господарств: Встановлення сонячних панелей на дахах приватних будинків і підприємств стає доступним рішенням для самопостачання енергією.

Сталість і прозорість: Сонячні електростанції мають низькі витрати на обслуговування і не вимагають значних витрат на логістику та сировину, що робить їх сталим та прозорим джерелом енергії.

У цій ситуації розвиток сонячної енергетики стає важливим фактором для забезпечення сталого постачання електроенергії, зменшення навантаження на існуючі енергетичні системи та сприяє підвищенню надійності енергозабезпечення в Україні.

В умовах дефіциту електроенергії не малу роль відіграє її розумний розподіл та використання. Тому прогнозування потужності СЕС є важливим елементом. Точні прогнози забезпечують ефективне управління споживанням енергії та можливість розробки стратегій для зменшення дефіциту енергії в майбутньому, що робить сонячну енергетику важливим джерелом в розвитку сталої та надійної енергетики. Це дозволяє ефективно використовувати сонячну енергію, зменшити навантаження на існуючі енергетичні джерела, й планувати технічні роботи ТЕЦ, АЕС та інших.

1.2 Технологічний процес

Сонячна електростанція - це технічна споруда, що перетворює енергію сонячної радіації в електричну енергію. Сонячні електростанції складаються з

різних компонентів, які працюють разом. На рисунку 1.2 зображено автономну сонячну електростанцію.

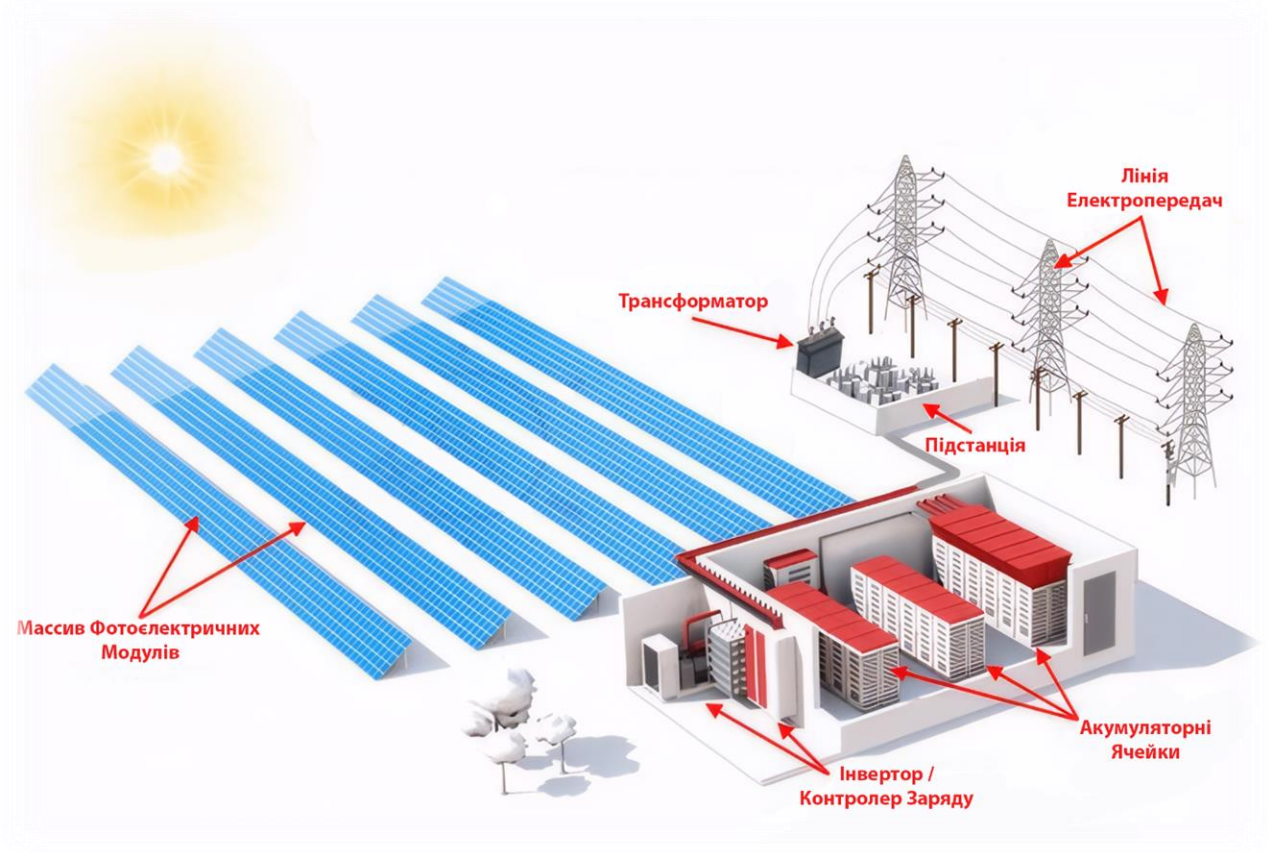


Рисунок 1.2 – Схема технологічного процесу СЕС

Ось основні компоненти сонячної електростанції:

- Фотоелектрична панель є серцем сонячної електростанції. Вона складається з невеликих сонячних елементів, які перетворюють енергію сонячних фотонів на електричну. Кремній зазвичай використовується як напівпровідниковий матеріал у сонячних елементах. Декілька осередків з'єднані послідовно або паралельно, утворюючи модуль, а кілька модулів складають сонячну панель. Фотоелектричні масиви створюються шляхом монтажу кількох панелей разом.
- Інвертор: Вихід фотоелектричної панелі здійснюється у вигляді постійного струму, але для більшості навантажень у мережі енергосистеми потрібен змінний струм. Тому інвертор використовується для перетворення постійного струму сонячних панелей на змінний струм. На електростанціях, підключених до мережі, інвертори оснащені захисними

пристроями та трансформаторами для забезпечення відповідності вихідної напруги та частоти стандартним вимогам.

- Акумулятори використовуються для зберігання надлишкової електричної енергії, що виробляється сонячною електростанцією. Ці накопичувачі енергії мають вирішальне значення задоволення попиту у періоди, коли сонячне світло недоступне.
- Контролер заряду регулює процес заряджання та розряджання акумулятора, щоб запобігти не вірному режиму роботи, що може призвести до пошкодження акумулятора.
- Компонент балансування системи: Цей набір компонентів контролює, захищає та розподіляє енергію всередині системи. Це забезпечує ефективну та безпечну роботу системи, максимізуючи продуктивність та захищаючи інші компоненти сонячної електростанції. Як приклад можна навести блокуючі діоди та стабілізатори напруги.
- Блокуючий діод підключений між батареєю та сонячною панеллю, щоб запобігти зворотному струму від батареї до панелі, який може пошкодити сонячну панель у періоди відсутності сонячного світла.
- Регулятори напруги використовують для підтримки стабільної вихідної напруги від сонячних панелей, особливо при коливаннях доступності сонячного світла. Вони гарантують, що навантаження отримує постійний та прийнятний діапазон потужності.

Перетворення сонячної енергії в електричну енергію розпочинається на фотоелектричних елементах за допомогою процесу, відомого як фотоефект. Коли металева поверхня піддається впливу монохроматичної електромагнітної хвилі з досить короткою довжиною (або, що те саме, вище порогової частоти), падаюче випромінювання поглинається, і опромінювана поверхня випускає електрони. Це явище і є фотоефект. Електрони, які випускаються в цьому процесі, називають фотоелектронами. Кремній є найбільш широко використовуваним напівпровідниковим матеріалом для створення фотоелектричних компонентів.

Той самий процес відбувається, коли світло падає на кристалі кремнію. Якщо інтенсивність падаючого світла досить велика, то достатня кількість фотонів поглинається кристалом, і ці фотони, в свою чергу, збуджують деякі електрони ковалентних зв'язків. Збуджені електрони отримують достатньо енергії, щоб перейти з валентної зони до зони провідності. Оскільки рівень енергії цих електронів знаходиться в зоні провідності, вони виходять з ковалентного зв'язку, залишаючи за собою "дірку" в зв'язку після виходу кожного електрона. Ці електрони називають вільними електронами, які рухаються випадково в структурі кристалу кремнію. Вільні електрони та "дірки" відіграють важливу роль у створенні електроенергії в фотоелектричній комірці. Ці електрони та "дірки", які виникають внаслідок світла, називаються відповідно світло генерованими електронами та "дірками". Проте ці світло генеровані електрони та "дірки" не можуть створити електроенергію у кристалі кремнію самі по собі.

Коли до кремнію додається п'ятивалентна домішка, така як фосфор, чотири валентних електрони кожного атома п'ятивалентного фосфору утворюють ковалентні зв'язки з чотирма сусідніми атомами кремнію, тоді як п'ятий валентний електрон залишається не залученим до жодного ковалентного зв'язку.

Коли до напівпровідникового кристалу додають атоми тривалентної домішки, такі як бор, замість атомів п'ятивалентного фосфору, створюється напівпровідник протилежного типу. У цьому випадку частина атомів кремнію в кристалічній решітці буде замінена атомами бора, іншими словами, атоми бора займуть місця заміщених атомів кремнію у структурі решітки. Коли у елементі створюються різні напівпровідники і він проходить процес металізації й отримується фотоелектричний модуль рисунок 1.3

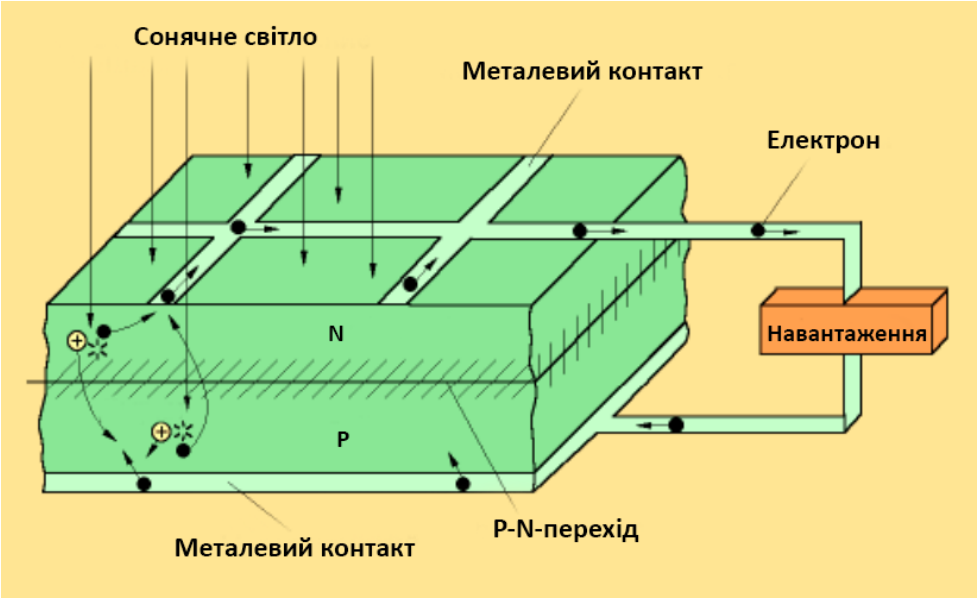


Рисунок 1.3 – Фотоелемент

Коли світло падає на напівпровідник, частина світла поглинається кристалом, і це збуджує валентні електрони, які виходять з ковалентних зв'язків і утворюють вільні електронно-діркові пари. У напівпровіднику n-типу, де є багато вільних електронів, дірки не можуть перейти в область p-типу через електричне поле, і навпаки. В результаті світлові дірки переходять в область p-типу, де вони рекомбінуються з електронами, і утворюється різниця потенціалів між областями. Ця різниця потенціалів створює напругу, яка може бути використана для створення електроенергії.

1.3 Об’єкт дослідження

1.3.1 Загальна характеристика об’єкта управління

Метою бакалаврської роботи було підвищення продуктивності сонячної фотоелектричної установки для генерації електроенергії. Для досягнення цієї мети необхідно, щоб сонячні промені падали на сонячну панель під правильним кутом та протягом якомога довшого часу. Однак, це неможливо без системи відстеження руху сонця, оскільки сонце постійно переміщається, змінюючи свій азимут протягом дня та висоту протягом року. Система, яка відстежує та спрямовує сонячну панель, називається сонячним трекером. Такі трекери можуть бути двох- або одноосьовими, тобто вони можуть відстежувати рух сонця як по

азимуту, так і по висоті, або лише по азимуту. В роботі був обраний саме одноосьовий трекер.

Схематичний зовнішній вигляд системи позиціонування (Рис. 1.4) описує металеву конструкцію з двома опорами, яка встановлюється на поверхню землі. На кінцях опор розміщені підшипники для підтримки металевої вісі каркасу. Каркас призначений для кріплення чотирьох сонячних панелей загальною потужністю 1 280 Вт. Сонячні панелі з’єднані в паралель. Крім панелей, на каркас встановлюється одноосьовий датчик для вимірювання кута нахилу конструкції.

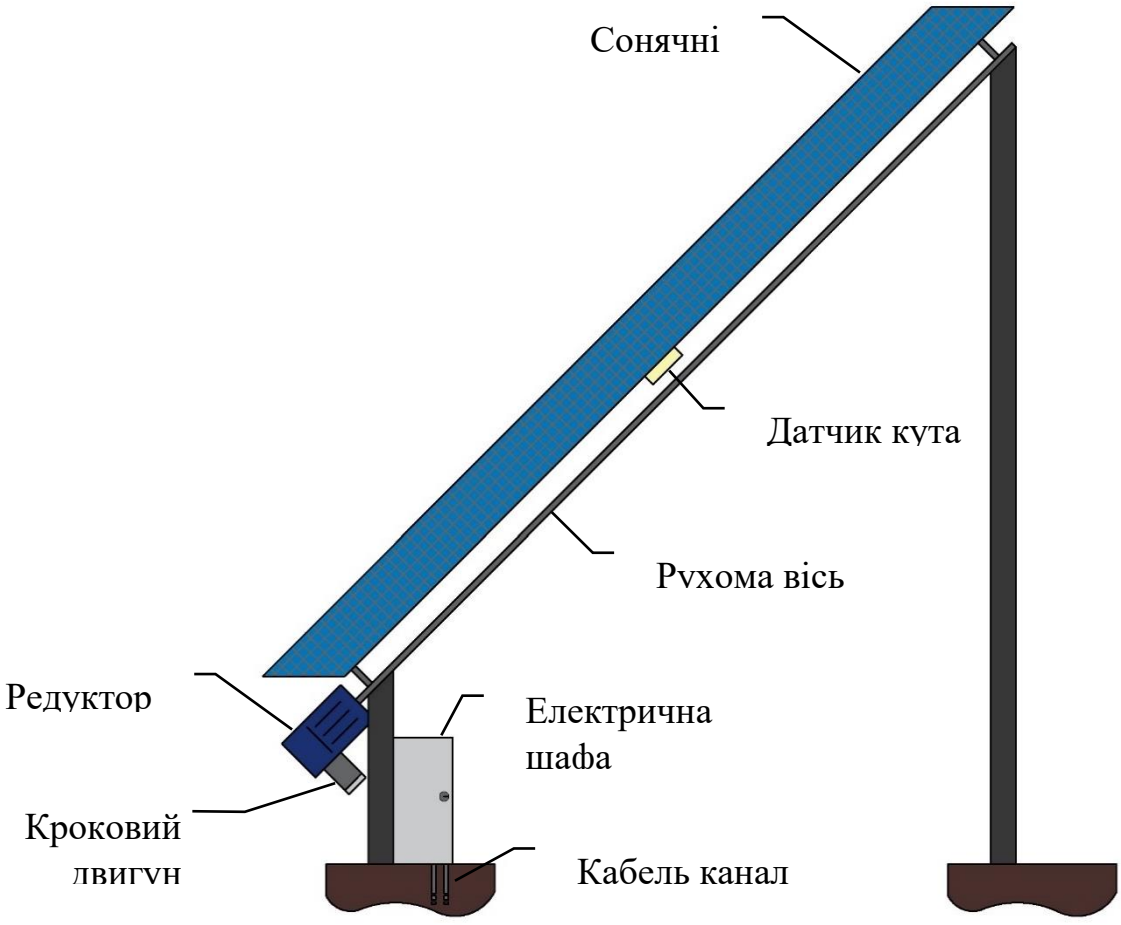


Рисунок 1.4 – Об'єкт управління

Каркас може повертатися вздовж однієї осі в діапазоні від 20° до 160° відносно вертикальної лінії. На нижньому кінці валу встановлено черв'яковий редуктор, до якого підключений кроковий двигун. Металева шафа, що

розташована на нижній балці, має захист не нижче, ніж IP 63, і включає в себе плату управління трекером, датчики напруги та струму, блок живлення, частотний перетворювач та одноплатний комп'ютер. Всі кабелі до установки прокладені через кабельний канал.

Для реалізації дослідницької частини роботи був збудований стенд (Рис. 1.5). Він повністю повторює систему описану раніше, але в меншому масштабі. Складеться з 50 ватної сонячної панелі, характеристики якої наведені у таблиці 1.1. Оскільки сонячний елемент лише один, то використовується одно опорна конструкція, так як вага та парусність об’єкту набагато менша.

Рисунок 1.5 – Об'єкт дослідження

Таблиця 1.1 – Технічні характеристики фотоелектричного модуля

Назва параметру	**Значення**
Номінальний струм	2.78 А
Струм короткого замкнення	2.9 А
Номінальна напруга	18 В
Напруга холостого ходу	22 В
Номінальна потужність	50 Вт
ККД	15.96%

Назва параметру	Значення
Коефіцієнт втрати потужності в залежно від нагрівання фото-модуля	-0.34%
Вага	4.5 Кг
Розміри	540x670x25 мм
Технологія	Моно-кристал

У зв'язку з одноосьовим трекером, нахил металевої конструкції має велике значення. Для його визначення потрібно аналізувати зміну висоти сонця протягом року відносно координат місця розташування конструкції. Для цього були використані координати університету та розроблене власноруч програмне забезпечення. Результати цієї аналітики представлені на графіку (Рис. 1.6), де вісь X представляє день року, а вісь Y - кут сонця. З графіку видно не лише зміну висоти сонця, але й тривалість світлового дня. Світлий колір відображає день, тоді як темний - ніч. На основі цих даних встановлено, що оптимальний нахил конструкції для осінньої пори становить від 20° до 36°. Тому було обрано нахил в 28°.

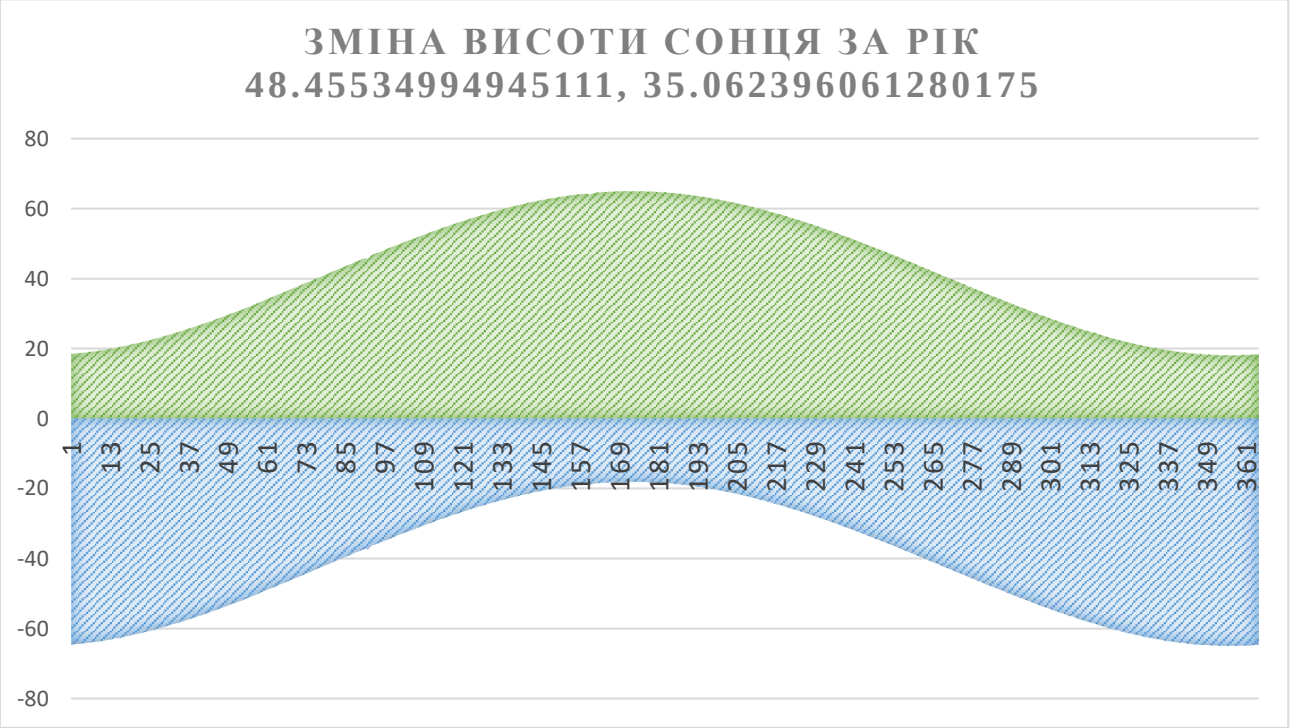


Рисунок 1.6 – Процес зміни висоти сонця

1.3.2 Структура об’єкта управління

Вхідні параметри:

- Орієнтація фотоелектричної установки.
- Напруга.
- Струм.
- Час.

Вихідні параметри:

- Напрямок руху крокового двигуна.
- Кількість кроків.

Рисунок 1.7 відображає взаємодію та структуру вхідних та вихідних параметрів системи.

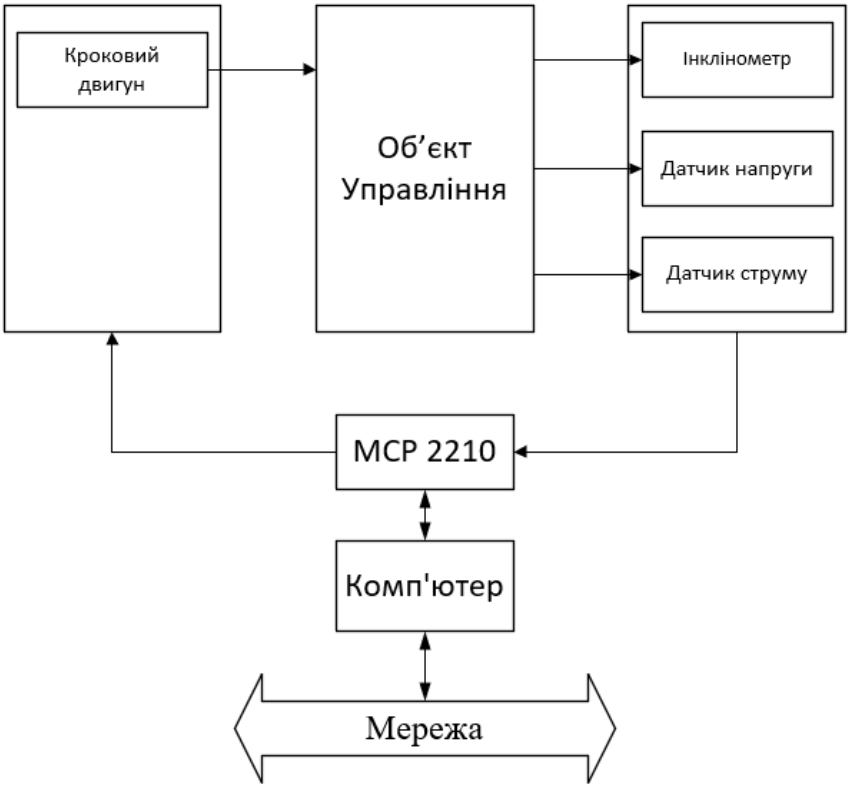


Рисунок 1.7 – Структура об’єкта управління

Дані параметри, після зчитування, проходять початкову обробку та служать основою для створення системи керування, яка базується на MCP 2210.

MCP 2210 - це спеціальний контролер (USB – SPI міст), який використовується для взаємодії з різними сенсорами та пристроями через інтерфейс USB. В даному випадку, MCP 2210 відповідає за зчитування параметрів, керування та виконання важливих функцій у процесі контролю сонячної електростанції. А плата управління зображена на рисунку 1.8.

Для усіх необхідних математичних розрахунків, остаточної обробки даних та формування команд для управління системою використовується одноплатний комп'ютер або любий інший, з USB інтерфейсом та сучасною операційною системою. Комп'ютер підключений до глобальної мережі, що дозволяє точно синхронізувати систему за часом і надавати можливість дистанційного керування та моніторингу системи.

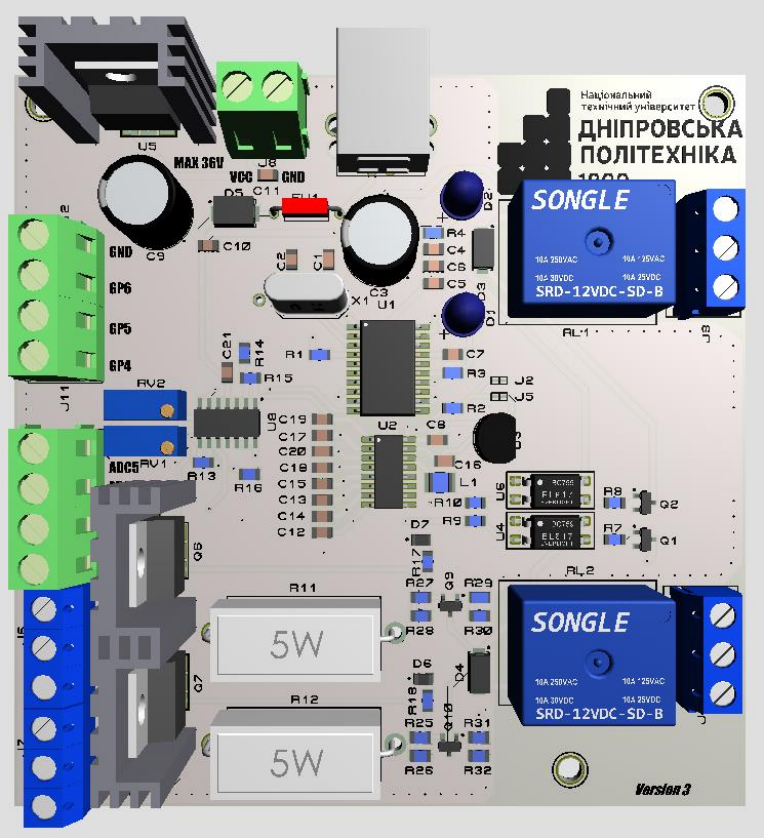


Рисунок 1.8 – Зовнішній вигляд контролера

Оскільки однією з цілей роботи є розробка системи прогнозування потужності, то досить важливим елементом об’єкту є обчислювальна техніка, так як складні математичні моделі та нейромережі потребують багато обчислювальних ресурсів у період прогнозу та ще більше в момент навчання.

1.3.3 Принцип функціонування об’єкта керування

Об'єкт управління функціонує у двох режимах, в залежності від вибору оператора. У ручному режимі оператор має повний контроль над системою і може визначити кут нахилу установки самостійно. Після цього програмне забезпечення автоматично розраховує кількість кроків та напрямок руху для досягнення обраного кута. Крім того, у ручному режимі оператор може переглядати всі параметри системи та отримувати інформацію щодо положення сонця, але ці дані є виключно інформативними.

Автоматичний режим є основним режимом роботи трекеру, де система автоматично керує положенням сонячних панелей, враховуючи координати сонця та налаштовуючи кут нахилу. Якщо висота сонця стає менше 0 градусів, система припиняє слідкування та повертається на схід, очікуючи сходу сонця. Паралельно ведеться моніторинг вихідної потужності. Мінімальна або відсутня потужність може свідчити про хмарність або снігове покриття. В такому випадку система намагається видалити сніг з поверхні панелей, особливо взимку, коли потужність близька до нуля. Швидкість встановлюється на максимальне значення і панелі розміщаються під кутом 20 або 160 градусів відносно вертикальної лінії. Після очищення панелей система повертається до звичайного режиму роботи. Це допомагає знизити витрати енергії. У випадку, якщо сніг не був видалений, система переключається на роботу в режимі з паралельним розташуванням панелей відносно поверхні землі та залишається в цьому стані до заходу сонця. Це важливо для збереження енергії.

Метод визначення положення сонця передбачає обчислення його висоти і азимуту на основі часу та точних географічних координат місця, де встановлена установка. Перший крок полягає у визначенні різниці між істинним часом та середнім сонячним часом, що відоме як рівняння часу за допомогою формули 1.1.

$$T_{\text{рв}} = 9.87 * \sin(2B) - 7.53 * \cos(B) - 1.5 * \sin(B) \quad (1.1)$$

де $B = \frac{360° * (N - 81)}{365}$; N – номер дня року.

Після чого знаходиться сонячний час, формула 1.2 та часовий кут світила 1.3.

$$T_{\text{ic}} = T_{\text{рв}} + T_{UTC} + \alpha \quad (1.2)$$

$$t = T_{\text{ic}} - 720 \quad (1.3)$$

T_{UTC} – це реальний час у форматі UTC, α – довгота.

Маючи ці вихідні дані і враховуючи зміну нахилу сонця відносно екватора, ми можемо розрахувати висоту та азимут. Висота - це кут між напрямком на сонце та математичним горизонтом. Азимут - це кут між лінією перетину математичного горизонту і площиною вертикального кола сонця та прямолінійною лінією. Нижче наведені формули для розрахунку азимуту (1.4) і висоти (1.5) на Рисунку 1.9.

$$A = atan2\,(\cos(\delta) * \sin(t),\ \cos(\delta) * \sin(\varphi) * \cos(t) - \sin(\delta) * \cos(\varphi)) \quad (1.4)$$

$$H = \text{asin}(\sin(\delta) * \sin(\varphi) + \cos(\delta) * \cos(\varphi) * \cos(t)) \quad (1.5)$$

Де φ – широта; t – часовий кут; δ – нахил сонця.

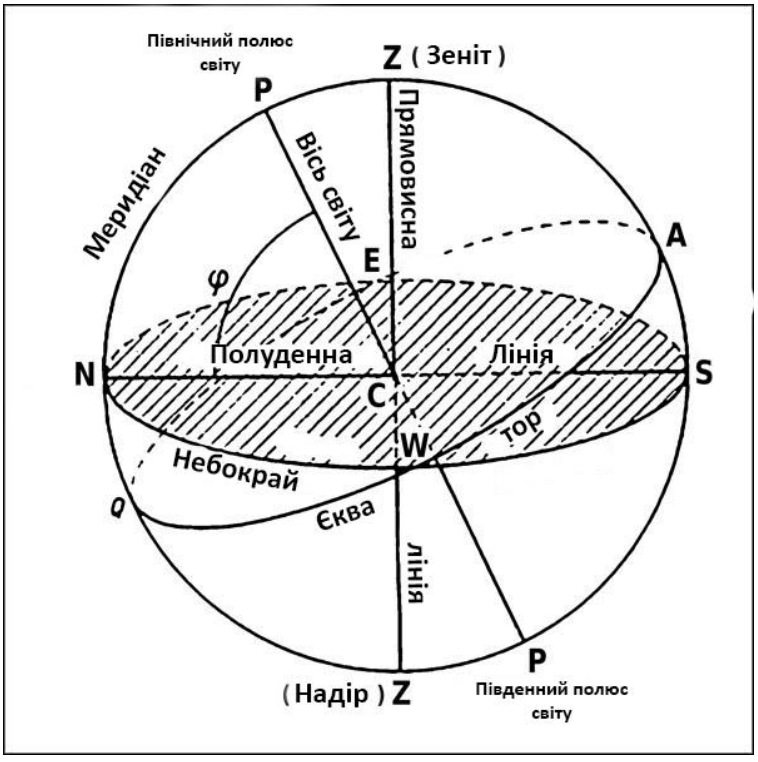


Рисунок 1.9 – Небесна сфера

1.4 Формулювання завдань дослідження

В попередніх пунктах висунутий опис об’єкта керування, який був розроблений в дипломній роботі рівня бакалавр. Це не обхідно для розуміння

принципу побудови та планування дослідження. В якості джерела інформації виступає не уявний об’єкт чи модель, а реальний прототип. Який піддається впливу навколишнього середовища і факторів, які можуть бути не враховані фахівцем під час розробки системи. Це також дозволяє реалізувати повноцінні випробування результатів дослідження, зробити певні висновки щодо дієвості наукової роботи.

Основним завдання дослідження є розробити дієву систему прогнозування потужності. Яка може вирішити суттєві недоліки не тільки галузі сонячних електростанцій, а й інших переривчатих джерел енергії.

Ця задача потребує критичного та комплексного підходу. Адже сфера розробки не є достатньо розкритою. В загально доступних джерелах не існує готового рішення чи методу розробки таких систем. Тому ціль вимагає аналізу, дослідження, пошуку залежностей та оптимальних рішень.

Основна задача одночасно породжує велику кількість науково-технічних завдань, а саме:

- Розробити системи аналізу тимчасових рядів, різного рівня складності, з використання екзогенних складових.
- Дослідити залежності екзогенних факторів на потужність фотоелектричного модуля.
- Розробити метод оцінювання та отримання екзогенних параметрів.
- Розробити модель аналізу тимчасових рядів на основі нейромережі.
- Дослідити рівень складності систем, порівняти їх. Прийняти рішення щодо переваг та недоліків певних систем, та визначити лідера.
- Про аналізувати існуючи апаратні заходи розробки складних математичних систем та комп’ютерних програм.
- На основі існуючих обчислювальних ресурсів, визначити оптимальну систему прогнозування. Що дозволить виявити найкращу модель не тільки в плані точності, а й економічної доцільності.
- Зробити чіткі та змістовні висновки щодо результатів дослідження.

1.5 Висновки за розділом

Розділ дає змогу ознайомитись с об’єктом дослідження, фотоелектричним модулем. Описує стан галузі та дозволяє зрозуміти проблеми та недоліки цього типу об’єктів. Предметом дослідження є процес прогнозування потужності СЕС.

Висунутий опис прототипу його алгоритму роботи та характеристики, які відіграють роль у дослідженні. Поставленні чіткі задачі дослідження.

2 ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Прогнозування параметрів потужності на основі моделей обробки даних

Прогнозування сонячної енергії можна розділити на дві основні категорії. Перша категорія - це пряме прогнозування (Рис. 2.1), яке безпосередньо оцінює фотоелектричну потужність. Друга - це непряме прогнозування (Рис. 2.2), яке базується на прогнозуванні сонячної радіації. Велика кількість дослідників працювала над цим питанням протягом останніх кількох років. Однак, навіть зараз точність прогнозу залишається складною задачею для науковців. Прогнозування сонячної радіації та фотоелектричної енергії - це нелінійна задача, яка залежить від декількох погодних параметрів. Для такої нелінійної системи вибір відповідного методу оцінки параметрів є важливим завданням.

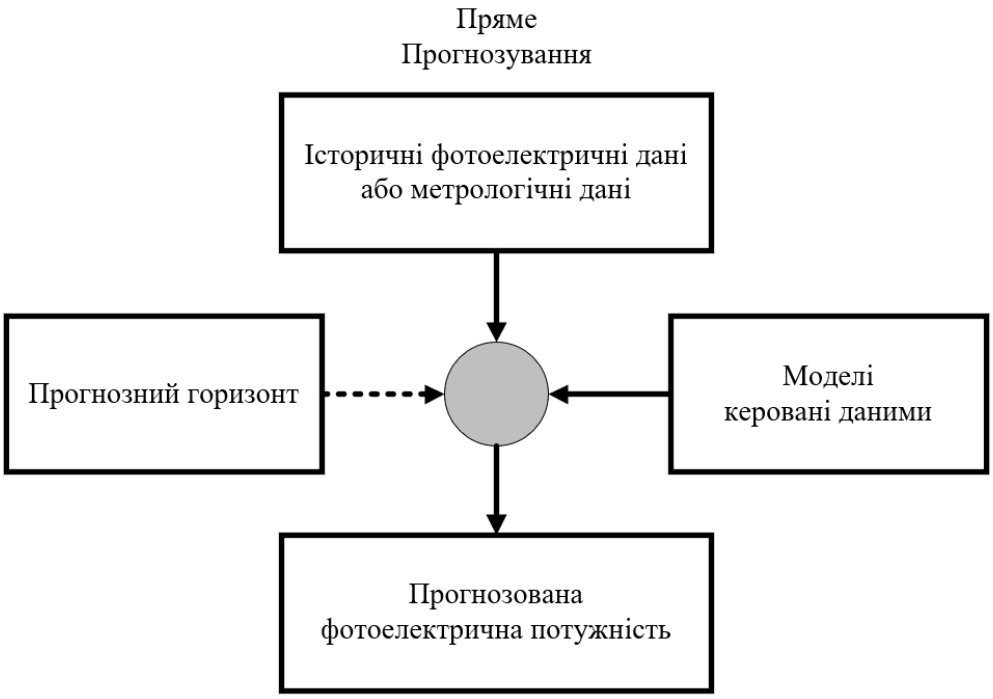


Рисунок 2.1 – Графічне зображення структури прямого прогнозування.

У доступній науковій літературі запропоновано кілька методів прогнозування, які можна розділити на дві основні категорії: фізичні моделі та моделі, які базуються на даних. Вибір конкретної моделі прогнозування залежить від горизонту прогнозу та регіону, для якого проводиться прогноз.

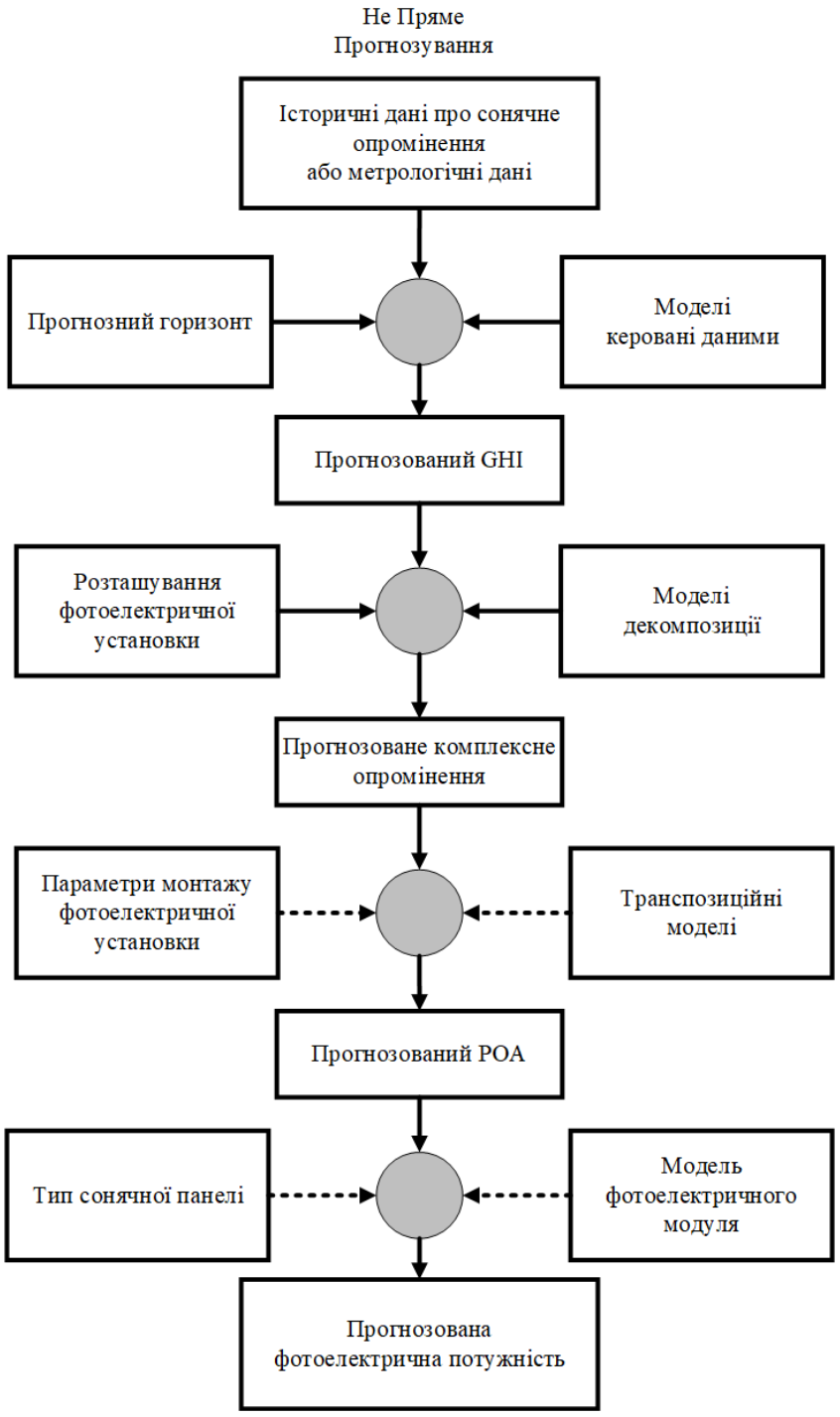


Рисунок 2.2 – Графічне зображення структури не прямого прогнозування.

Пояснення до зображення: GHI в даному контексті є абревіатурою для "Global Horizontal Irradiance" (глобальна горизонтальна радіація). Це параметр, який використовується в сфері сонячної енергії та метеорології для вимірювання сонячної радіації, яка приходить на горизонтальну поверхню Землі. Прогнозований GHI вказує на очікувану кількість сонячної радіації, яка досягне горизонтальної поверхні в певному місці і в певний час. Ця інформація є

важливою для визначення потенційної ефективності сонячних енергетичних систем, а також для прогнозування енергетичних виробництв сонячних станцій і планування сонячних проектів.

POA (Plane of Array) означає кількість сонячного випромінювання, яке досягає сонячної панелі або сонячної батареї. Це вимірювання є важливим для оцінки ефективності та продуктивності систем сонячної енергії, оскільки воно вказує, скільки сонячного світла фактично потрапляє на сонячні панелі, беручи до уваги кут і орієнтацію панелей. Опромінюваність POA зазвичай вимірюється у ватах на квадратний метр (Вт/м²) і допомагає оцінити виробництво енергії сонячною установкою за різних умов, наприклад зміни кута нахилу сонця або затінення від сусідніх об’єктів. Це ключовий фактор у визначенні загальної потужності сонячної фотоелектричної системи.

Один з найпоширеніших методів - це чисельний прогноз погоди, який використовує фізичну модель атмосфери для прогнозування радіації на певний часовий горизонт. Однак існують також моделі, які базуються на внутрішніх даних, і вони використовують інформацію, отриману з вхідних навчальних даних, для прогнозування вихідних параметрів. Ефективність цих методів не є вивченою і великою мірою залежить від якості і доступності навчальних даних.

Припускається гіпотеза, що моделі які базуються на нейронних мережах мають більшу точність прогнозування, адже можуть враховувати неявні залежності системи та самостійно оцінювати їх значимість для прогнозу.

Існують і більш вивчені методи прогнозування потужності. Наприклад метод оцінки хмарності по спеціальним камерам або супутниковим знімкам. В таких системах цифрове зображення майже по піксельно завдяки інфрачервоній камері класифікується на чисте небо, оптично тонкі та оптично щільні хмари. Такий спосіб дозволяє з похибкою в 1% класифікувати ясну погоду та щільні хмари. А тонкі, не щільні хмари з точністю у 60%. Але такий спосіб є коротко строковим, добре вивченим й як покажуть подальші експерименти, хмари та їх щільність далеко не ключовий фактор для оцінення продуктивності модуля. А іноді вони будуть негативно впливати на точність прогнозу.

Системи базовані на супутникових знімках це набагато вагоміший підхід. Який дозволяє прогнозувати на довший термін, до 9-12 годин вперед. Але потребує доступ до таких знімків безпосередньо «тут й зараз», що є неможливим для звичайної людини або малої компанії. Окрім цього необхідна модель прогнозу руху хмар та їх оцінки, що потребує наявності дуже потужних розрахункових систем. І знову таки, цей метод вже добре відомий та досліджений.

2.2 Тимчасовий ряд та його особливості

Завдання дослідження вимагає аналізу даних. Таких як зовнішні стійкі та не стійкі чинники, закономірності у зв’язках певних груп значень, властивості попередніх даних, їх патерни. Найпростіший аналіз та прогноз не можливий без знання попередніх даних об’єкту дослідження. У нашому випадку об’єкт надає можливість зберігати параметри напруги, струму та потужності з урахування часу запису. Такий вид даних називається тимчасовим рядом.

Тимчасовий ряд - це структурований набір даних, де кожне вимірювання виконується і фіксується відповідно до рівних і постійних проміжків часу. У цьому наборі інформації кожен вимір має чітку відмітку відносно часової шкали, що дозволяє візуалізувати та аналізувати як зміни, які відбуваються в часі, так і різниці між різними вимірюваннями у визначені моменти. Відомості у таких рядах можуть бути збережені, проаналізовані, та використані для передбачення, розуміння тенденцій або виявлення залежностей в динаміці даних. На рисунку 2.3 графічно зображено тимчасовий ряд досліджуваного об’єкту. Це записані показники потужності з 30 липня по 4 серпня 2023 року.

Завдяки таким даним є можливість виконати велику кількість аналізів та отримати певні результати на основі часу. А саме:

- Прогнозування.
- Класифікація.
- Сегментація.
- Аналіз збурювачів.

- Описовий аналіз.

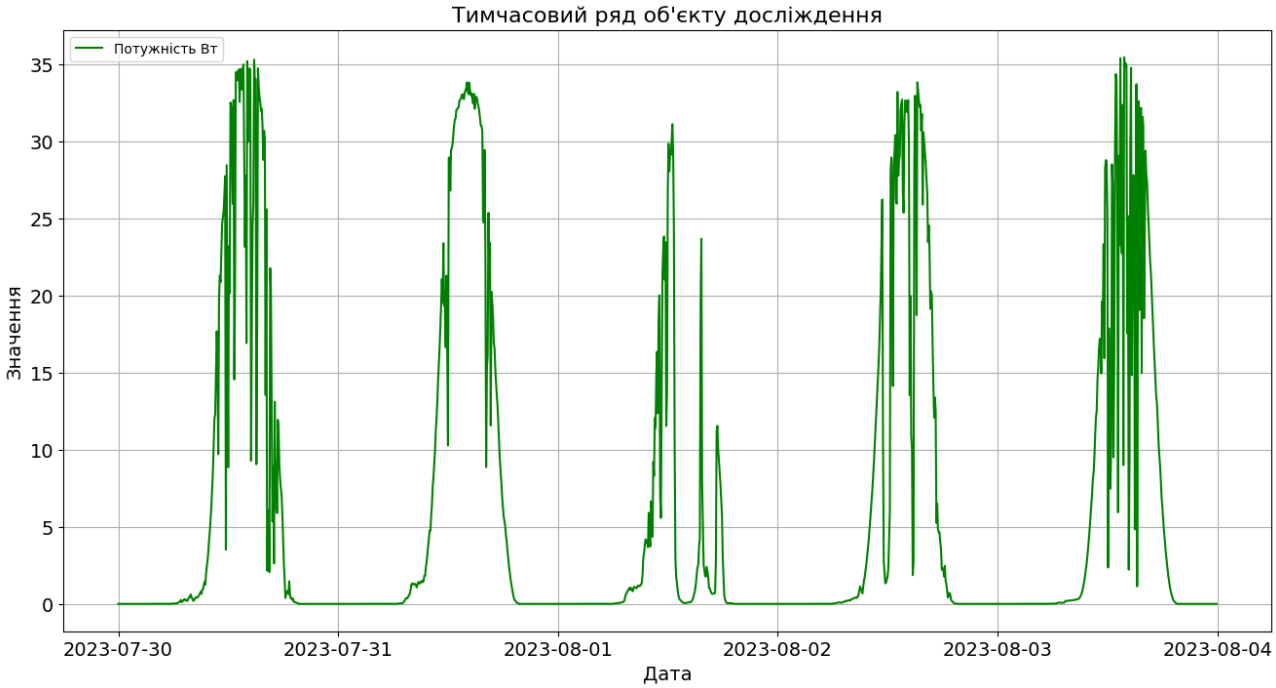


Рисунок 2.3 – Тимчасовий ряд об’єкту дослідження.

У подальшому ми детальніше, та у більш складному виді розглянемо цей часовий ряд. Кожен ряд має компоненти для аналізу, а саме:

- Тенденція (Тренд). Довгостроковий патерн часового ряду, що вказує на загальну тенденцію зростання чи спадання протягом певного періоду часу.
- Сезонність. Регулярні коливання, які повторюються в певні періоди часової лінії, чітко вказують на сильний вплив екзогенних циклічних параметрів.
- Циклічність. Довгострокові коливання які не мають фіксованого періоду.
- Нерегулярність. Непередбачувані відхилення від тренду, сезонності та циклічності в часовому ряді.

В середні ряду данні можуть бути стаціонарними або не стаціонарними. Для стаціонарного ряду притаманна властивість того, що статистичні характеристики ряду (такі як середнє значення і дисперсія) залишаються сталими або не змінюються від часу до часу. Іншими словами, стаціонарний ряд не має

систематичних трендів, сезонних коливань або інших регулярних змін у часовому плані, що полегшує статистичний аналіз та передбачення майбутніх значень.

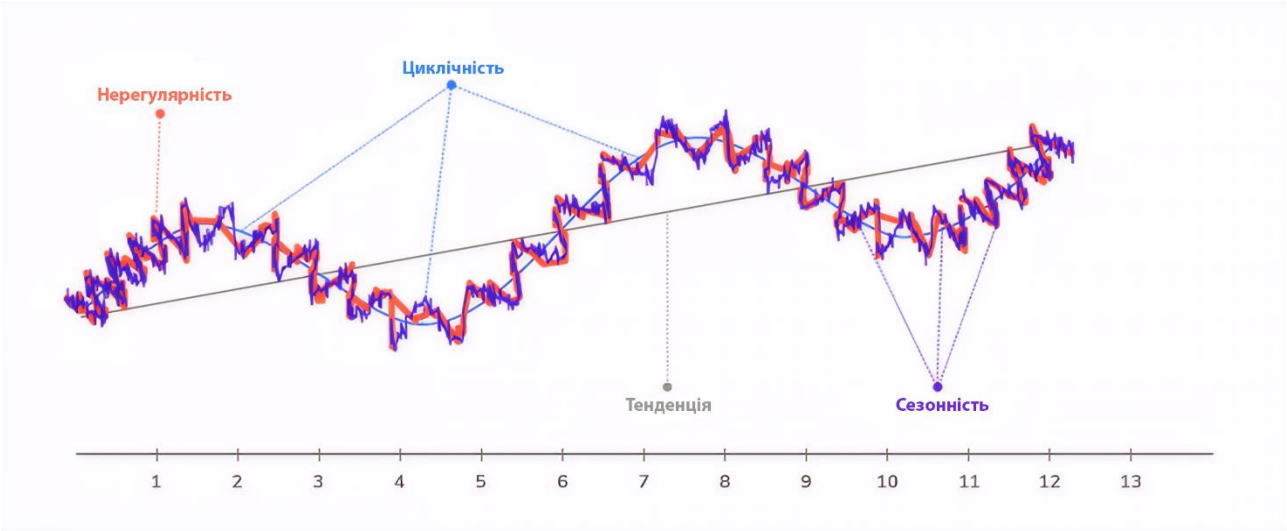


Рисунок 2.4 – Явне зображення компонентів ряду

2.3 Модель Холта-Уінтерса

Прогнозування за допомогою методу Холта-Уінтерса - це спосіб моделювання та передбачення поведінки послідовності значень у часі, тобто часового ряду. Метод Холта-Уінтерса є одним із найпоширеніших підходів до прогнозування часових рядів. Він використовується протягом багатьох десятиліть і досі широко використовується у різних сферах, включаючи моніторинг, де він застосовується для таких завдань, як виявлення аномалій та планування ресурсів.

У свою чергу даний метод не використовувався у сфері прогнозування потужності енергетичних установок. Він має досить легку та швидку форму обчислення. Це важливий фактор для інтеграції. Адже дозволяє розмістити навчання та прогнозування на всіх обчислювальних пристроях. Починаючи з простих 8-бітні мікроконтролерів. У нашому випадку це дозволяє виконувати прогноз на стороні клієнту в браузері чи на стороні серверу, але про це пізніше.

Існує гіпотеза, що ця модель занадто проста, і програє моделі SARIMA, яка є набагато складнішою та комплексною. Обидві моделі не враховують екзогенні параметри, а прогнозують лише опираючись на попередні данні часового ряду.

В подальшій роботі ми перевіримо припущення щодо ефективності моделей та дослідимо, чи мають такі моделі право на життя у системі, яка потенційно сильно залежна від зовнішніх факторів.

Метод Холта-Уинтерса використовує експоненційне згладжування для врахування набору минулих значень та використання їх для прогнозування "типових" значень у поточний та майбутні періоди. Експоненційне згладжування включає в себе використання експоненційно зваженої слизької середньої (EWMA) для "згладжування" часового ряду. Якщо у вас є часові ряди X_t, то можна визначити новий часовий ряд E_t, який представляє собою згладжену версію X_t.

$$E_t = a * X_t + (1 - a) * E_{t-1} \tag{2.1}$$

Так як модель зовсім ігнорує лінійний тренд, то додається згладжування Холта. І рівняння моделі прийме вид 2.2.

$$\begin{cases} E_t = a * X_t + (1 - a) * (E_{t-1} + B_{t-1}) \\ B_t = \beta(E_t - E_{t-1}) + (1 - \beta) * B_{t-1} \end{cases} \tag{2.2}$$

Цього абсолютно не достатньо для складного об’єкту як у нас. Тому необхідно додати сезонну компоненту, для того щоб модель могла враховувати усі чотири компоненти часового ряду. Які були описані вище. Отримаєм повну модель у вигляді формули 2.3.

$$y_{t+h|t} = E_t + hB_t + S_{t+h-m(k+1)}$$

$$\begin{cases} E_t = a(X_t - S_{t-m}) + (1 - a)(E_{t-1} + B_{t-1}) \\ B_t = \beta(E_t - E_{t-1}) + (1 - \beta) * B_{t-1} \\ S_t = \gamma(X_t - E_{t-1} - B_{t-1}) + (1 - \gamma) * S_{t-m} \end{cases} \tag{2.3}$$

Отримана модель включає у себе одне рівняння прогнозування та три рівняння згладжування. Прогнозування залежить від трьох коефіцієнтів a, β, γ, кожен з яких відповідає своєму компоненту ряду: циклічність, тренд, сезонність. Існують дві варіації цього методу, які різняться характером сезонної компоненти.

Адитивний метод вибирається, коли сезонні коливання залишаються приблизно постійними протягом всього ряду, тоді як мультиплікативному методу віддають перевагу, коли сезонні варіації змінюються пропорційно рівню ряду. У випадку адитивного методу сезонна компонента виражається в

абсолютних величинах, що відповідають масштабу спостережуваного ряду, і рівень ряду коригується сезонним відніманням сезонної компоненти. Протягом кожного року сезонна компонента буде практично нульовою. У випадку мультиплікативного методу сезонна компонента виражається у відсотках, і ряд коригується сезонно шляхом ділення на сезонну компоненту. Протягом кожного року сезонна компонента буде становити приблизно m відсотків. Вказана вище модель є адитивною. Оскільки сезонні коливання є постійними, детальніше у 3 розділі.

Повна модель Холта-Уінтерса має недолік для нашої системи, а саме низька стійкість до викидів у часовому ряді. Потужність фотоелектричного модуля це складний набір даних. Буть що, хмара, відблиск об’єкта створює аномалію у часовому ряду. Тому є ідея покращення моделі завдяки методу Брутлага, також відомий як метод "3σ" (три рівня вибірки стандартного відхилення). Це статистичний спосіб для виявлення аномалій в часових рядах або наборах даних. Цей метод допомагає виявити значення, які виходять за межі очікуваних патернів або нормального розподілу даних. Модель Холта-Уінтерса використовується для прогнозування часових рядів, і точність прогнозу може бути порушена наявністю аномалій або викидів у даних.

Включення методу Брутлага в модель Холта-Уінтерса дозволяє:

- Виявляти аномалії: Метод Брутлага допомагає виявляти незвичайні або аномальні значення у часовому ряді, які можуть бути в результаті помилок в вимірюваннях або інших факторів. Виявлення цих аномалій важливо для покращення точності прогнозування та забезпечення надійності моделі.
- Зменшення впливу аномалій на прогноз: Якщо аномалії виявлені, можливо, їх можна врахувати або виключити з аналізу, щоб зменшити їх вплив на результати прогнозування. Це допомагає моделі Холта-Уінтерса краще адаптуватися до звичайних патернів у даних.
- Покращення якості прогнозу: Виявлення та обробка аномалій може сприяти покращенню якості прогнозів, оскільки модель буде менш схильна до впливу надзвичайних або неповторних подій.

Однак важливо враховувати, що метод Брутлага передбачає нормальний розподіл даних і може не бути ефективним в випадках, коли розподіл даних суттєво відрізняється від нормального. Тому важливо адаптувати методи виявлення аномалій до конкретних вимог та властивостей даних, які аналізуються в рамках моделі Холта-Уінтерса.

Не можливо використовувати модель у дослідженні не розуміючи загально підтверджених недоліків:

- Передбачення складних нестаціонарних рядів: Модель Холта-Уінтерса призначена для передбачення часових рядів зі стабільними трендами та сезонними компонентами. Вона може не бути ефективною для нестаціонарних рядів, де тренд та сезонність змінюються в часі.
- Обмежена узагальненість: Модель Холта-Уінтерса підходить для відносно простих часових рядів з регулярною сезонністю. У випадку складних даних, що містять багато шуму або аномалій, ця модель може бути недостатньою.
- Вимоги до даних: Модель Холта-Уінтерса вимагає наявності історичних даних для розрахунку параметрів моделі. Якщо немає достатньої кількості даних, то точність передбачення може бути низькою.
- Наявність змішаних сезонностей: Модель Холта-Уінтерса передбачає лише одну видиму сезонність. Якщо в даних присутні різні рівні сезонності, це може вплинути на точність передбачення.
- Відсутність автоматизації: Модель Холта-Уінтерса вимагає ручного вибору параметрів моделі (наприклад, параметрів згладжування) та підбору їх на підставі даних.
- Залежність від початкових значень: Початкові значення параметрів моделі можуть впливати на точність передбачень. Вибір початкових значень може бути непростим завданням.

Як можна зрозуміти стосовно деяких недоліків ми прийняли рішення вже у цьому розділі. Наприклад додавши метод Брутлага, детальніше продовжмо у розділі з реалізацією цього методу. Також ми бачимо певну кількість наукових завдань які постають для практичного розділу, такі як: порівняння з іншими методами, оцінка впливу зміни параметрів моделі, оцінка впливу аномалій.

2.4 Моделі сімейства ARMA

Моделі сімейства ARMA (авторегресія та ковзне середнє) є важливим інструментом в аналізі та прогнозуванні часових рядів. Їх використання розповсюджується на багато галузей, включаючи фінанси, економіку, метеорологію, та інші, і є критичним для розуміння та передбачення динаміки даних зі змінними у часі.

Моделі ARMA поєднують в собі два основних компонента: авторегресію, також відому, як AR модель, що враховує залежність поточного значення від попередніх значень, та ковзне середнє, MA модель, що описує вплив змінних шоків на поточне значення. Ці моделі дозволяють аналізувати та передбачати залежність між змінними в часі, враховуючи їхню історію та структуру.

ARMA модель не підходить для нашого часового ряду, так як вона не враховує важливі патерни ряду, наприклад сезонність, та є не придатною для нестаціонарного часового ряду. Для вирішення цієї проблеми система була допрацьована і на її основі з’явилась ARIMA. В ній додається компонент I, який вказує на присутність процесу інтеграції в моделі. Це дозволяє вирівнювати не стаціонарні ряди. Завдяки рівням різницювання, в залежності від складності ряду.

Але ця модель як і раніше не враховує сезонність, яка у нашому об’єкті дослідження сильно виражена, й не буде коректно працювати на відміну від моделі Холта-Уінтерса, яка враховує цей аспект даних.

На малюнку 2.5 зображено дерево моделей ARMA. Кожна модель є ускладненою версією попередньої моделі, яка враховує більше залежностей. Червоним кольором вказані моделі які не можуть бути використанні у процесі

прогнозування фотоелектричної потужності. Світло-зеленим кольором відображена модель, яка є мінімальної для прогнозу і саме її необхідно буде у подальшому порівняти с моделлю Холта-Уінтерса. Адже вона враховує усі аспекти попередніх даних у більш складному вигляді, що повинно привести до більшої точності прогнозування. Зеленим кольором відзначена модель, що є кращою за всі попередні, так як враховує зовнішні фактори на вплив даних у ряді.

Ця модель в подальшому буде порівнюватися з нейромережами, які дві системи з урахуванням екзогенних параметрів.

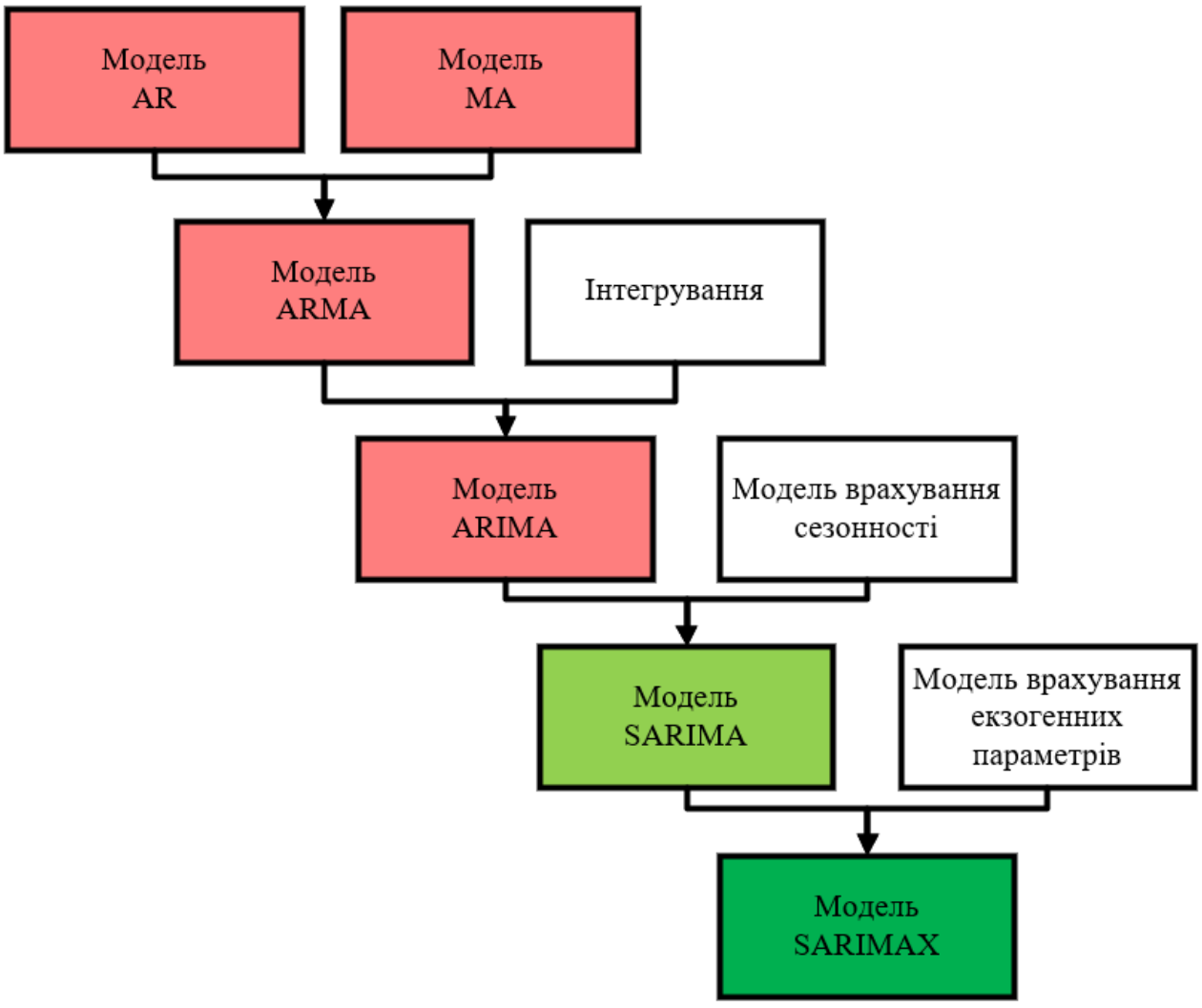


Рисунок 2.5 – Структура ARMA подібних моделей

2.4.1 Модель SARIMA

Модель SARIMA (Seasonal Autoregressive Integrated Moving Average) є розширенням моделі ARIMA і включає в себе сезонну компоненту S, для аналізу

та прогнозування часових рядів. Вивчення SARIMA моделі стає дедалі важливішим у зв'язку з необхідністю аналізу та передбачення подій, що мають явний сезонний характер, такі як температурні коливання, в схід та захід сонця. Сезонна залежність в часових рядах відіграє ключову роль у різних галузях. Це може бути циклічна активність підприємства, щорічні погодні коливання або інші періодичні події. Саме для врахування цих особливостей та здатність адекватно моделювати сезонну залежність була створена модель SARIMA.

Навіть не аналізуючи детально ряд об’єкта дослідження, можна бачити сильну виражену щоденну сезонність, окрім цього існує ще річна сезонність яка пов’язана с циклічною зміною висоти сонця протягом року.

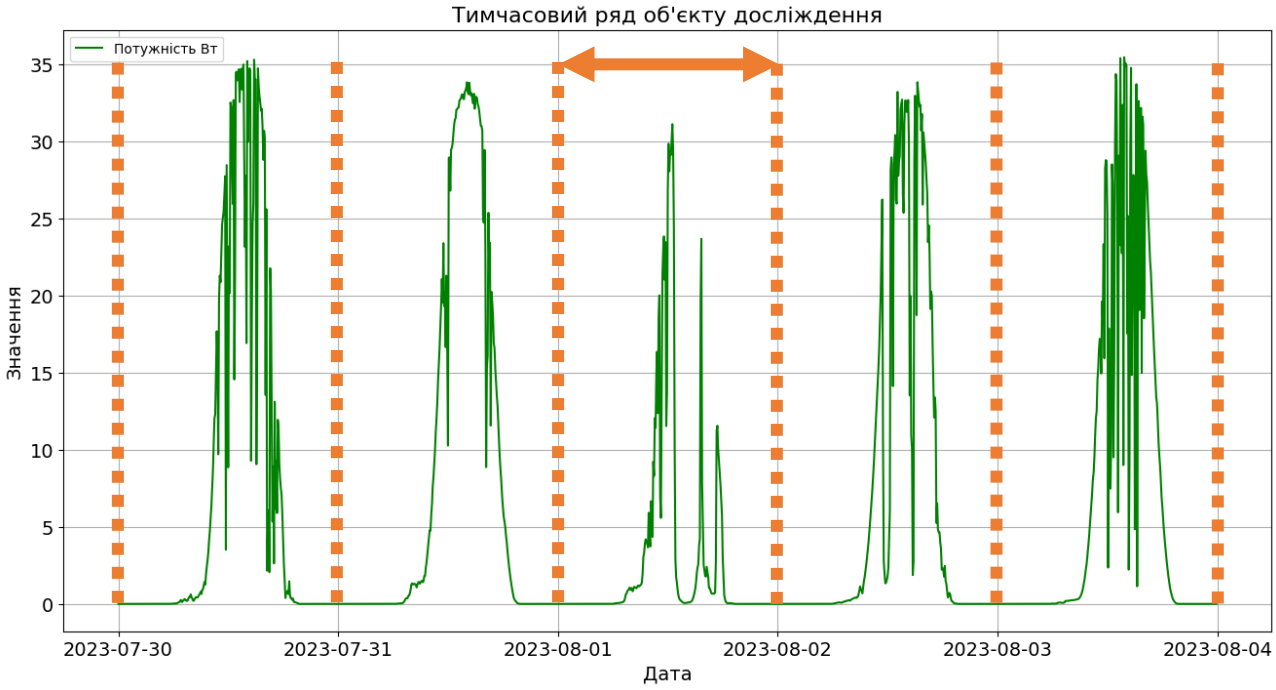


Рисунок 2.6 – Сезонність у даних об’єкта дослідження.

Модель ARIMA мала 3 кофіцієнта для налаштування, кожен з яких відповідає за свою особливість поведінки ряду. В SARIMA таких кофіцієнтів 7.

$$SARIMA = ARIMA\ (p, d, q) + S(P, D, Q)_m$$

В свою чергу модель S э той самою ARIMA але яка використовуэться для прогнозування сезоної компонети. m – це довжина сизону, а кофіцієнти P, D, Q повність аналогічні p, d, q.

Усі методи мають якісь недоліки стосовно досліджувального об’єкта і ми можемо їх припустити зараз стосовно моделі SARIMA, а первірити чи спробувати вирішити вже підчас побудови моделей:

- **Складність в підборі параметрів:** SARIMA має кілька параметрів, включаючи порядок авторегресії, порядок ковзного середнього, порядок інтеграції та сезонні параметри. Підбір оптимальних значень цих параметрів може бути складним завданням і вимагати багато експериментів.
- **Чутливість до даних:** SARIMA може бути чутливою до змін в даних. Якщо часовий ряд має складну структуру або непередбачувані зміни, то модель може давати невірні прогнози.
- **Необхідність стаціонарності:** SARIMA передбачає стаціонарність часового ряду. В ідеальному випадку, часовий ряд повинен бути стаціонарним до застосування моделі SARIMA, і це може вимагати попереднього перетворення даних.
- **Підсумкові прогнози:** SARIMA надає прогнози лише для майбутніх точок в часі, а не для цілого періоду. Це може бути незручним, коли потрібно зробити декілька кроків уперед у часі.
- **Складність інтерпретації:** Визначення сезонних та несезонних параметрів у моделі SARIMA може бути складним завданням, і інтерпретація цих параметрів не завжди є очевидною.
- **Недостатність для деяких видів даних:** SARIMA може бути менше ефективною для деяких видів даних, де є складні взаємозв'язки або несправедлива стаціонарність.

Ці недоліки є лише припущенням і вони вимагають перевірки на практиці.

2.4.2 Екзогенні параметри. Модель SARIMAX

Фотоелектрична установка є досить складним об’єктом для прогнозування, мала інерційність, висока збурюваність, зашумленість попередніх даних, та суттєва залежність від екзогенних параметрів.

Екзогенний фактор - це зовнішній чинник або змінна, яка впливає на систему, процес або явище і походить ззовні цієї системи. Екзогенні фактори є зовнішніми відносно об'єкта або системи, на яку вони впливають. В контексті прогнозування і моделювання, екзогенні фактори представляють собою змінні, які враховуються при аналізі та прогнозуванні явища, але вони не керуються самою системою. Це можуть бути змінні, такі як погодні умови, економічні показники, політичні рішення, технологічні інновації тощо. Екзогенні фактори враховуються в моделях для того, щоб зрозуміти, як вони впливають на систему або процес, та для покращення точності прогнозів.

У контексті фотоелектричних установок для виробництва сонячної енергії, екзогенні фактори можуть включати в себе метеорологічні умови, географічне розташування, економічні зміни, сезонні варіації тощо. Ці фактори впливають на продуктивність і ефективність фотоелектричної установки і повинні бути враховані при розробці моделей прогнозування сонячної енергії. Усього у об’єкта більше 21 екзогенного параметру.

Однак попередні системи не можуть враховувати такий тип інформації. Тому до системи SARIMA був доданий екзогенний регресор й вийшла система SARIMAX, для реалізації прогнозування с підвищеною точністю.

$$SARIMAX = ARIMA\ (p, d, q) + S(P, D, Q)_m + Exog$$

Exog представляє собою набір екзогенних регресорів або екзогенних змінних, які враховуються у моделі. Формули для цих регресорів можуть бути додані до відповідних компонентів AR, I і MA для врахування їх впливу на часовий ряд. Така модель є найскладнішою в сімействі і потребує суттєвих ресурсів для навчання та прогнозу.

Для об’єкту дослідження можуть виникнути наступні труднощі:

- **Велика кількість параметрів:** SARIMAX може мати велику кількість параметрів для авторегресії (AR), інтеграції (I), ковзного середнього (MA) та сезонних компонент. Це може зробити модель складною і вимагати багато часу на налагодження та оцінку параметрів.

- **Потребує великої кількості даних:** SARIMAX вимагає наявності великої кількості часових даних для ефективної оцінки параметрів та стабільних прогнозів.
- **Вимоглива до передприємливості:** Модель SARIMAX потребує вивчення і ретельного аналізу даних для визначення правильних значень параметрів AR, I, MA та сезонних компонент. Це вимагає великих зусиль і експертної економетричної підготовки.
- **Не враховує складні зв'язки:** SARIMAX передбачає лінійні зв'язки між екзогенними регресорами і часовим рядом. Вона не враховує складні нелінійні зв'язки та можливі нестаціонарності в даних.
- **Відсутність автоматичного відбору регресорів:** SARIMAX не має вбудованого автоматичного відбору екзогенних регресорів. Це означає, що вам доведеться вручну вибирати та додавати регресори до моделі.
- **Не враховує загальних зовнішніх факторів:** SARIMAX враховує лише певні екзогенні регресори, включені в модель. Вона не враховує загальні зовнішні фактори, які можуть впливати на часовий ряд, але не включені у модель.
- **Чутливість до викидів:** SARIMAX може бути чутливою до викидів або аномалій у даних, що може призвести до неправильних прогнозів.

2.5 Нейромережі. Архітектура RNN та LSTM

Останнім методом прогнозування який розглядається у дипломній роботі є прогнозування на основі рекурентних нейромереж. Це клас штучних нейромереж, призначений для обробки послідовних даних. Одна з ключових особливостей RNN - це їх здатність до роботи з даними, які мають послідовну структуру, таку як текст, часові ряди, аудіо та багато іншого. Основна ідея полягає в тому, що RNN зберігає певний стан або пам'ять, яка допомагає йому враховувати інформацію з попередніх кроків в обробці нових даних.

Основні компоненти RNN включають наступне та зображені на малюнку 2.7:

- Вхідний шар (Input Layer): Цей шар приймає вхідні дані, такі як послідовності слів у тексті, пікселі зображення або виміри часового ряду.
- Прихований шар (Hidden Layer): Прихований шар містить рекурентні нейрони або "пам'ятні" одиниці, які зберігають попередній стан і враховують його при обробці нових даних.
- Вихідний шар (Output Layer): Цей шар генерує вихідні прогнози або результати на основі обробки даних прихованим шаром.

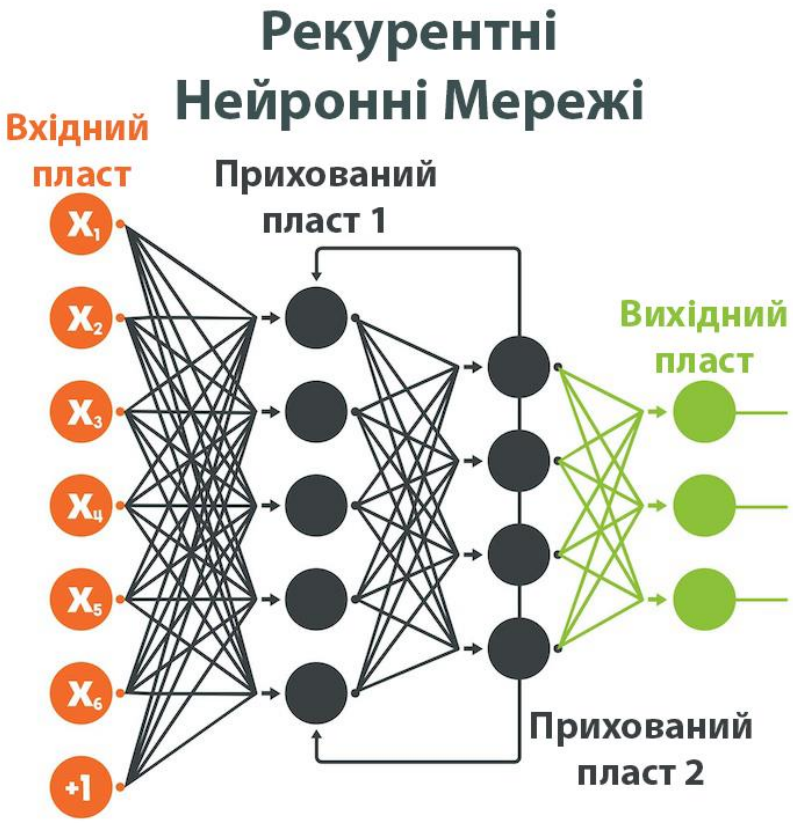


Рисунок 2.7 – Структура RNN архітектури.

Перша основна перевага RNN полягає в їх здатності до обробки даних різної довжини та структури завдяки механізму зворотного зв'язку (рекурентності). Це робить їх корисними в багатьох застосуваннях, включаючи машинний переклад, аналіз тексту, генерацію послідовностей, розпізнавання мови та багато інших завдань обробки послідовних даних. У нашому випадку послідовністю даних виступає потужність сонячної панелі.

Друга перевага суттєвіша, це пошук залежності в даних між екзогенними параметрами та результатами часових ряду. В найгіршому випадку розробник може навіть не оптимізувати мережу. А задати їй усі доступні впливи для прогнозу, й система автоматично зрозуміє який параметр дійсно впливає й коли. Хоча в свою чергу це залежить від складності самої мережі, а саме існує чотири основних параметри які встановлюють складність моделі, та архітектуру допоміжних систем таких як оптимізатори.

Перший параметр, це кількість вхідних нейронів. Їх кількість залежить виключно від кількості вхідних параметрів часового ряду без урахування дати, та екзогенних параметрів без урахування дати. Необхідно привернути увагу до обов’язкової необхідності однакової частоти вибірки, та розміру даних ряду, та даних екзогенних параметрів. У випадку коли це не можливо, необхідно виконувати масштабування та підготовку даних. Наприклад використовуючи інтерполяцію. Також є чітка вимога до форми ряду та його параметрів, наразі ця інформація не доречна і буде розкрита в практичному розділі. Але необхідно сказати про форму. А саме тензор, який візуально зображений на малюнку 2.8.

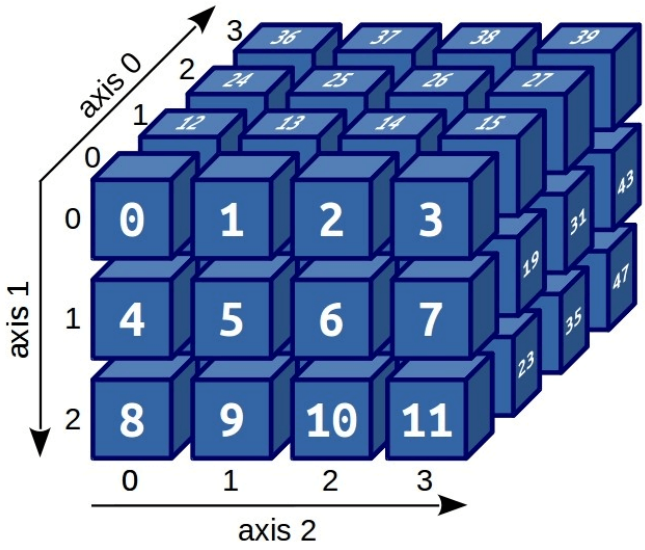


Рисунок 2.8 – Вигляд 3 рангового тензору.

В нейромережах термін "тензор" використовується для позначення даних, які подаються на вхід мережі або виводяться як її вихід. Тензори є основними структурами даних в глибокому навчанні і представляють собою багатовимірні масиви чисел.

Він має ранг який вказує на кількість осей або вимірів, які в ньому містяться. Наприклад скаляр, тобто число, має ранг 0. Вектор, він же одномірний масив чисел ранг 1, а матриця має ранг 2 й так до більших рангів. Розмірність вказує на кількість елементів (комірок) у кожному виміру даних. Тип даних вказує на сам тип комірок, тобто апаратну ємність. Вона може бути FP16, FP32, INT8, INT1 й інші. Це важливі показники, ігнорування яких може привести не тільки до не можливості виконати прогноз на обладнанні, але й створити помилки в ряді тим самим порушивши роботу мережі. Правильний тензор разом з розумінням технологій та сучасних апаратних засобів, допоможе досягти феноменальної швидкості навчання, і ці питання буде розібрано у цій роботі.

Другий параметр, це приховані шари. Його друга назва прихований вектор. Це основна одиниця нейронної мережі архітектури RNN. Їх кількість може бути від 1 (але така модель не має корисності) до нескінченності. Все залежить від ресурсів системи. Чим більше прихованих шарів тим модель складніша й може запам’ятовувати досить великі об’єми даних, аналізувати велику кількість вхідних параметрі та шукати складні закономірності. Ось основні функції прихованого шару як об’єкта нейронної мережі:

- **Зберігання інформації:** Прихований шар містить набір нейронів (або одиниць), кожен з яких має ваги та функцію активації. Ці нейрони виконують обчислення, які допомагають зберігати та відображати інформацію про вхідні дані. За допомогою ваг і активаційних функцій прихований шар може виокремлювати корисні функції або патерни з вхідних даних.
- **Функція активації:** Кожен нейрон в прихованому шарі використовує функцію активації для обчислення вагової суми вхідних даних та подальшого виходу нейрона. Ця функція може впливати на нелінійність обчислень в мережі, дозволяючи вирішувати складні завдання та розпізнавати залежності в даних.
- **Роль у передачі інформації:** Прихований шар служить для передачі інформації між вхідним та вихідним шарами нейронної мережі. Він

виконує обчислення, які дозволяють адаптувати модель до конкретного завдання, навчаючи її розпізнавати важливі ознаки в даних.

- **Регуляризація:** Прихований шар може виконувати функцію регуляризації, допомагаючи запобігти перенавчанню (overfitting) та підвищити загальну здатність мережі.

В останньому пункті сказано про overfitting, тобто перенавчання. Це феномен в машинному навчанні, одна з проблем яка може стати перед розробником. Коли модель навчається тренувальним даними настільки добре, що вона надто точно пристосовується до них і втрачає здатність узагальнювати знання до нових, раніше не бачених даних. Іншими словами, модель стає надто чутливою до варіацій та шуму в тренувальних даних, що може призвести до поганої продуктивності на тестових або реальних даних.

Тому параметр кількості прихованих шарів повинен бути використаний раціонально, адже велика кількість шарів викличе перенавчання. А не тільки створить додаткове навантаження на апаратуру. На рисунку 2.9 показано недостатнє навчання, перенавчання та оптимальне навчання.

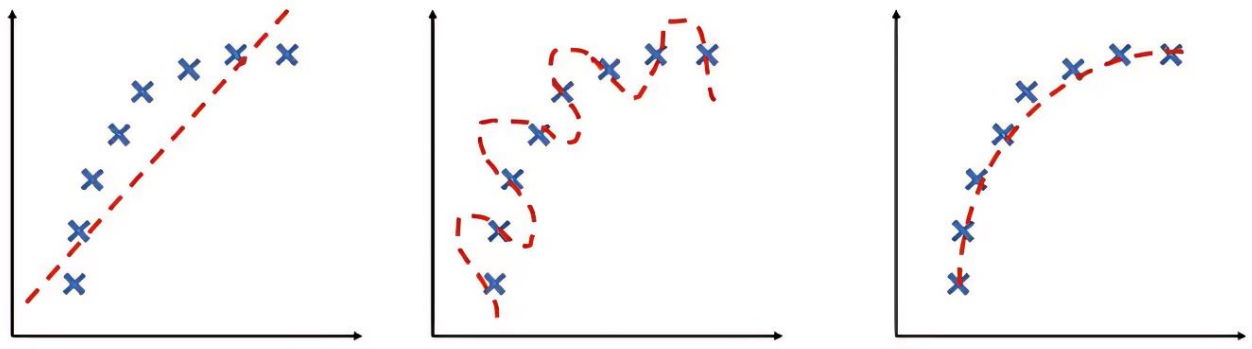


Рисунок 2.9 – Візуальні стани навчання.

Третій параметр це кількість вихідних нейронів, у нашому випадку тут все просто. Скільки параметрів необхідно спрогнозувати стільки й виходів. Прогнозуємо лише потужність тому вихід лише один. Його форма теж у вигляді тензору, але сама структура залежить виключно від вхідних даних.

Останній параметр це кількість епох. Цей параметр відповідає кількості навчаючих проходів. Він впливає на термін навчання, та має ті ж обмеження що й кількість прихованих шарів. Тобто велика кількість проходів призводить до перенавчання.

В дипломній роботі розглядається два виду нейронних мереж. Перша має назву Simple RNN. Це найпростіший вид, структура якого показана на рисунку 2.10.

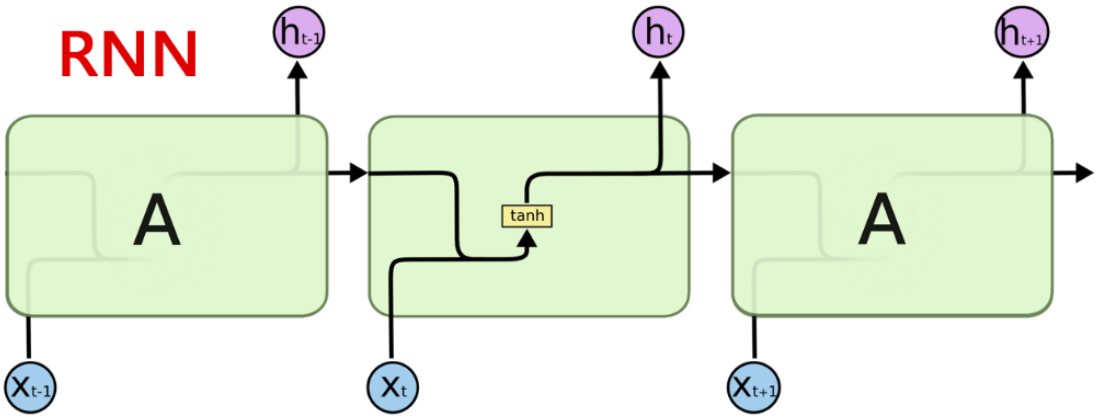


Рисунок 2.10 – Структура Simple RNN.

Другий вид має назву LSTM. Long Short-Term Memory – це ускладнений тип рекурентної нейронної мережі. Основна особливість і різниця від Simple RNN полягає у наявності довгострокової пам’яті. Така можливість дає системі краще запам’ятовувати градієнт в даних, легше моделювати складні взаємозв’язки.

Структурно різниця ще більше, на малюнку 2.11 зображена внутрішня структура комірки. LSTM включає gates (*на малюнку я переклав як шлях / вхід / вихід*) для керування інформаційним потоком в середині мережі. Вони дозволяють LSTM вибирати, що тримати в пам'яті та що викинути. Це робить LSTM більш гнучкою та здатною до більш точного контролю над процесом навчання. Окрім цього gates дозволяють вимкнути обробку зайвої інформації, коли вона не несе корисного семантичного значення.

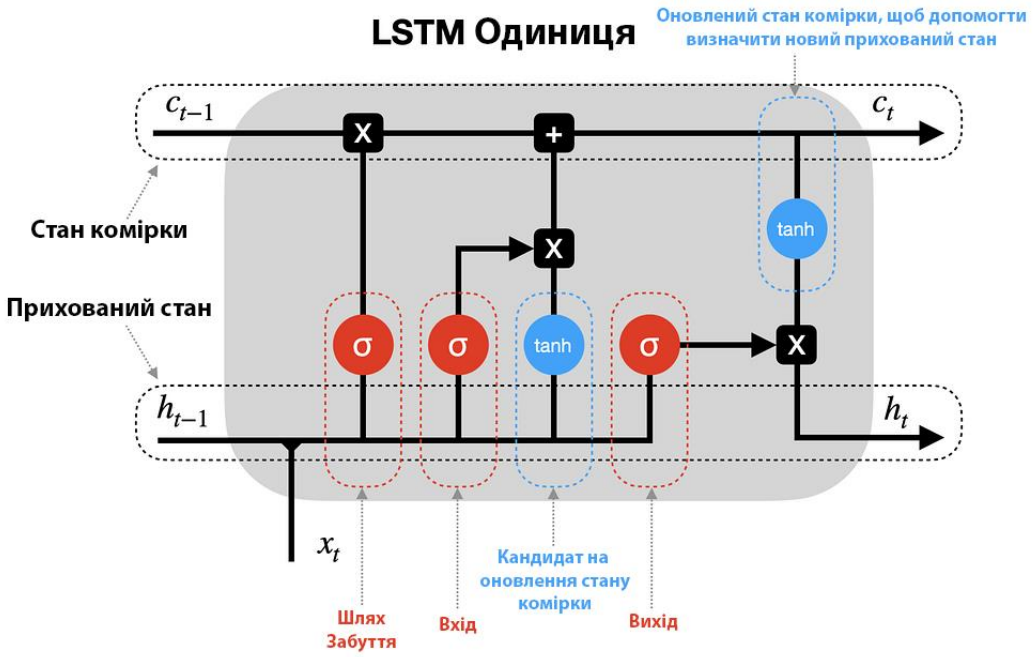


Рисунок 2.11 – Структура комірки LSTM.

2.6 Системи прогнозу та аналізу екзогенних параметрів ERA5, GFS Global,

Однією з проблем в питаннях прогнозування параметрів фотоелектричних модулів є отримання максимально точних екзогенних параметрів. Проблем немає коли зовнішній параметр сталий та стабільний протягам часу. Для нашої системи такими параметрами виступає точне положення сонця. Сонце не може різко змінити свій шлях, воно із року в рік проходить одну й туж саму траєкторію яку можна точно розрахувати використовуючи формули із астрономічної механіки. Ці данні допомагають гарно прогнозувати сезонність, адже є стабільні сезоні параметри.

Проблеми починаються тоді коли необхідно отримати щось менш стабільне. В нашому випадку це параметри погоди, і їх набагато більше ніж постійних параметрів. Погоду не можливо розрахувати за формулами та чітко сказати що буде завтра чи на наступній неділі. Так, ми можемо використовувати вже зареєстровані дані з метеостанцій, які мають найбільшу точність. Й такі данні це гарний матеріал для навчання нейромереж та моделі SARIMAX. Але для прогнозу необхідні данні на період прогнозу.

В дипломному проекті мінімальна голубина прогнозу 1 доба максимальна 7. Звичайні методи побудови погодного прогнозу не є ефективними, та не надають той кількості параметрів, яка нам необхідна. Тому для отримання якісних даних для навчання, ми можемо використати систему ERA.

ERA5 - це абревіатура, що позначає п'яту версію реаналізу European Centre for Medium-Range Weather Forecasts (ECMWF) або Європейського центру з погодних прогнозів середнього терміну. Реаналіз - це обробка та передача погодних даних за минулі періоди з метою підготовки оновленого та докладного обліку кліматологічних обставин.

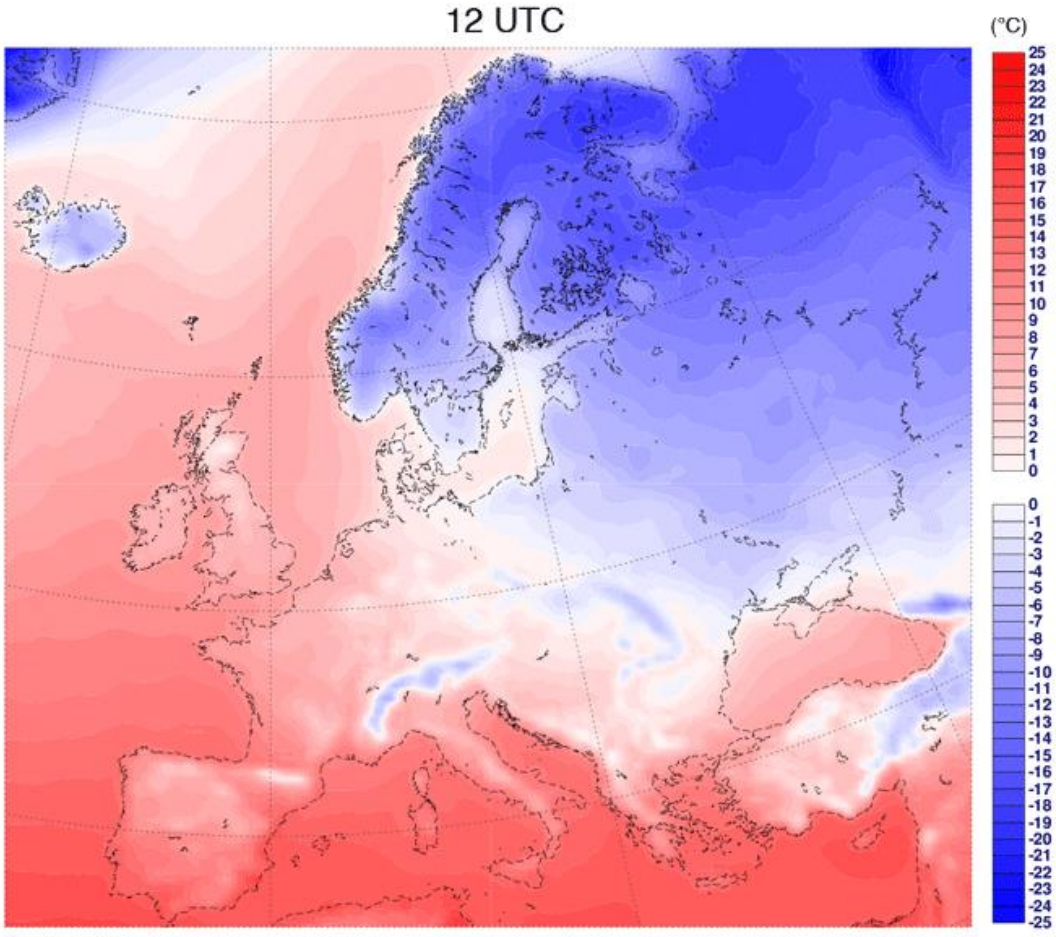


Рисунок 2.12 – Температури повітря ERA.

На відмінок від простих метеостанцій дані які отримується в системі ERA беруться з усіх видів метеорологічних установок Європи. Такі як супутники загального та кліматичного призначення, аеростати, наземні, морські, гірські метеостанції. Морські та повітряні судна.

Рисунок 2.13 – Ілюстрація отримання даних в системі ERA.

Доступність подібних даних є обмеженою, а саме коштує 1500 доларів на місяць. Але для науковців та студентів є можливість отримати доступ до історичних даних з кроком в одну годину. Саме ці данні були використані у подальшому навчанні. Всього їх 19 одиниць, а саме: температура повітря, відносна вологість повітря, точка роси, опади, інтенсивність дощу, інтенсивність снігу, стан погоди у кодовому вигляді, тиск, загальна хмарність, хмарність до 3 км, хмарність 3-6 км, хмарність вище 6 км, еталонне випаровування, дефіцит тиску, короткохвильове сонячне випромінювання, пряме сонячне випромінювання, розсіяна сонячна радіація, земна сонячна радіація, опромінення DNI. Всі ці данні були отримані відносно координат місцевості встановлення об’єкту дослідження.

Але де брати данні на майбутнє? Для цього використовують числові моделі прогнозу. У наше завдання дослідження не входить розробка такої системи, та й це потребує певної команди розробників та науковців. Тому ми будемо використовувати готові системи. Одна з таких це GFS Global.

Global Forecast System - це числова модель погоди, яка розроблена Національною аеронавтичною та космічною адміністрацією NASA та Національним центром для прогнозування середньої тривалості погоди (NCEP)

в США. Ця модель призначена для прогнозу погоди на глобальному рівні та надає прогнози на різний часовий горизонт. Перевагами є:

- **Глобальне покриття:** Модель GFS Global охоплює весь світ і надає прогнози для будь-якого регіону на планеті.
- **Числовий метод:** Модель використовує числові методи для розв'язання рівнянь гідродинаміки, теплообміну, волого обміну та інших фізичних процесів, які відбуваються в атмосфері.
- **Висока роздільна здатність:** GFS Global має високу роздільну здатність, що дозволяє робити прогнози з високою точністю для конкретних регіонів.
- **Часовий ряд даних:** Модель надає прогнози на різний часовий горизонт, включаючи годинні, тривалість уточнення прогнозів, що дозволяє враховувати зміни погодних умов протягом дня та ночі.
- **Інформація для метеорологічних служб:** Прогнози, що надаються моделлю GFS Global, використовуються метеорологічними службами для складання офіційних прогнозів погоди та вирішення питань, пов'язаних з погодними умовами.

І знову таки, всі системи прогнозування є обмеженими для безкоштовного не комерційного використання. У випадку з GFS є декілька тарифних планів які починаються від 150 доларів на місяць, закінчуючи узгодженою ціною с компанією. Але знову таки, для науковців та студентів вищих навчальних закладів є можливість отримати безкоштовний доступ з кроком в 3 години. Це досить великий крок, який може негативно вплинути на прогноз, тому необхідно комбінувати системи «Дешево й сердито». На малюнку 2.14 вказані усі доступні системи для прогнозування.

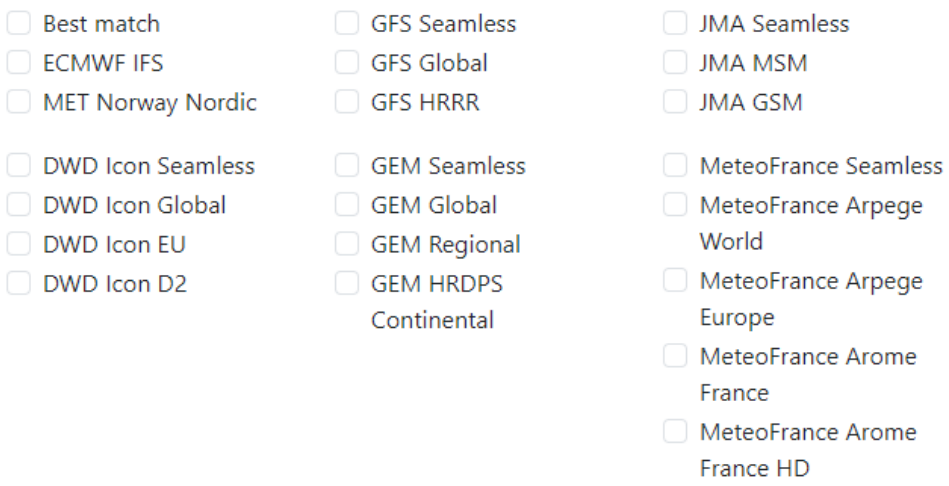


Рисунок 2.14 – Усі можливі моделі з безкоштовними даними.

2.7 Висновки за розділом

У розділі була розкрита теоретична база розв’язуваного завдання дослідження, це необхідно для розуміння практичного розділу. У ході опису теорії були висунуті гіпотези які вимагають перевірки, та описані методи вирішення певних проблем дослідження. Розглянуті обрані моделі прогнозування та викладені припущення, що до проблем реалізації, які можуть спіткати розробника подібних систем. Загальна структура розділу чітко формує шлях вирішення поставленої мети.

Кожна з представлених у цьому розділі моделей представляють фундаментально різні підходи, й як покаже розділ з дослідженням та експериментами, будуть мати різні параметри якості та проблеми в прогнозі.

3 ДОСЛІДЖЕННЯ ТА РОЗРОБКА КІБЕРФІЗИЧНОЇ СИСТЕМИ

3.1 Опис середовища розробки

Для виконання практичної частини дослідницької роботи знадобляться середовища для розробки. Необхідні наступні програмні засоби:

- Visual Studio 2022 с повним пакетом «.NET desktop development»
- Visual Studio Code з звантаженою підтримкою синтаксиста CSV таблиць та Python.
- Python 3.11, з додатками PIP 23.2.1 (*Таб. 3.1*)
- Jupiter Notebook.
- Nvidia CUDA Toolkit 12.3.
- PyTorch 2.1.0 на виконавчій платформі Nvidia CUDA 12.1 чи більш новішій.

Таблиця 3.1 – Основні додатки PIP

Назва бібліотеки	**Версія на якій виконувалось дослідження**
torch	2.1.0+cu121
torchaudio	2.1.0+cu121
torchvision	0.16.0+cu121
sympy	1.12
statsmodels	0.14.0
plotly	5.17.0
numpy	1.26.0
pandas	2.1.1
scikit-learn	1.3.1
scipy	1.11.3
tqdm	4.66.1

3.2 Опис даних та їх підготовка

Об’єкт дослідження був ведений в експлуатацію 31 травня 2023 року, тому на початку проведення дослідницьких робіт маємо данні напруги, струму, та потужності з 31.06.23 по 25.09.23. Запис даних відбувався завдяки програмному забезпеченню на мові програмування C#. Яке дозволяє зберігати данні у TXT файлі (Рис. 3.1), у вільному форматі та з встановленою користувачем частотою.

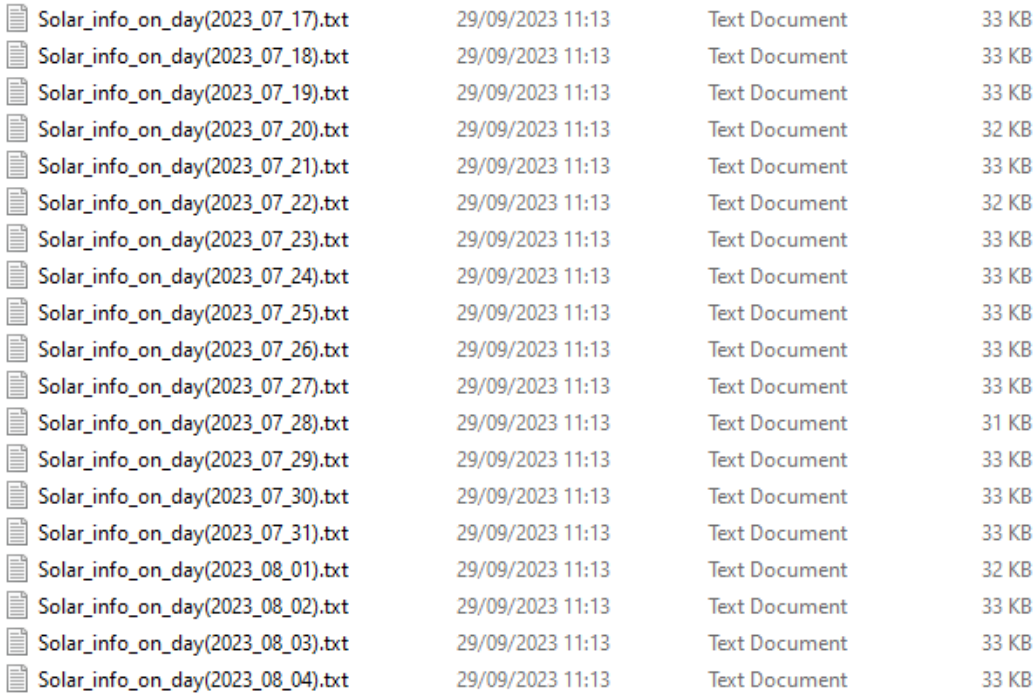


Рисунок 3.1 – Структура зберігання даних об’єктом дослідження.

Сам файл має заголовок, який вказує відповідність стовпчика до певної фізичної величини, та самі дані, с шагом реєстрації в 1 хвилину. Час зберігається у форматі «година.хвилина».

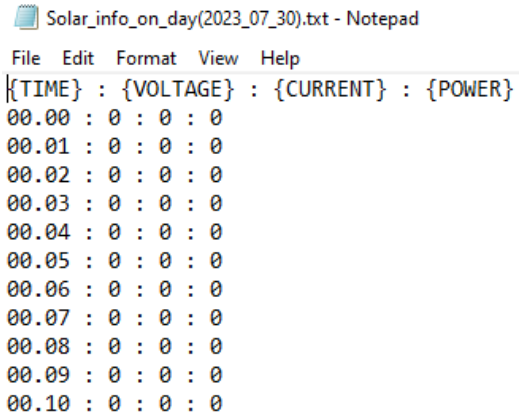


Рисунок 3.2 – Структура даних у файлі.

Хоч самі дані необхідні для дослідження, їх формат не є адаптованим для моделей й подальшої роботи. Необхідно створити повноцінну таблицю в форматі CSV, яка в одному файлі буде містити усі данні стосовно усього періоду та мати час в форматі ISO 8601, тобто наступний «2022-09-27T18:00:00.000». Для цього напишемо консольну програму на мові програмування C#. Яка автоматично зробить все за нас, результатом роботи програми є файл «CombinedFile.CSV», рисунок 3.3. На цьому етапі нам й знадобиться Visual Studio 2022, в подальшому частина із цього алгоритму увійде до самої програми, як її частина. Ці данні вже можемо використовувати для дослідження та побудови моделей без екзогенних параметрів.

	A	B	C	D
1	Date	Voltage	Current	Power
2	2023-05-31T00:00:00	0	0	0
3	2023-05-31T00:01:00	0	0	0
4	2023-05-31T00:02:00	0	0	0
5	2023-05-31T00:03:00	0	0	0
6	2023-05-31T00:04:00	0	0	0
7	2023-05-31T00:05:00	0	0	0
8	2023-05-31T00:06:00	0	0	0
9	2023-05-31T00:07:00	0	0	0
10	2023-05-31T00:08:00	0	0	0
11	2023-05-31T00:09:00	0	0	0
12	2023-05-31T00:10:00	0	0	0
13	2023-05-31T00:11:00	0	0	0
14	2023-05-31T00:12:00	0	0	0
15	2023-05-31T00:13:00	0	0	0
16	2023-05-31T00:14:00	0	0	0
17	2023-05-31T00:15:00	0	0	0

Рисунок 3.3 – Структура даних після їх підготовки.

3.3 Реалізація та навчання моделі Холта-Уінтерса

Для реалізації та дослідження усіх моделей використовується мова програмування Python, та Jupiter Notebook. Це комфортне середовище розробки, яке дозволяє виконувати програмний код блочно, та отримувати результат розрахунку або зображення блоку не виконуючи весь код програми. Після

однократної ініціалізації блоків їх можна виконувати у любому порядку, що дуже корисно для експериментів.

Для початку проініціалізуємо усі необхідні бібліотеки, у блоку нижче зображено усі необхідні бібліотеки для проводження дослідження.

```
# Импорт бібліотек
import sys
import warnings
warnings.filterwarnings('ignore')
from tqdm import tqdm
import pandas as pd
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error
import statsmodels.formula.api as smf
import statsmodels.tsa.api as smt
import statsmodels.api as sm
import scipy.stats as scs
from scipy.optimize import minimize
from statsmodels.tsa.stattools import adfuller
import pmdarima as pm
import matplotlib.pyplot as plt
```

Завантажимо данні до Jupiter Notebook, у змінну «dataset», с параметрами індексування та пошуком даних по даті. Після чого видалимо стовпці «Voltage», «Current», так як вони нам поки що не знадобляться. Встановимо тип даних у комірках float16, це необхідно для економії ресурсів комп'ютера. Все одно, початкові данні мають 2 знаки після зап'ятої, так що поки не має сенсу надавати великі комірки даних. Та розіб'ємо наш великий dataset на 3 маленьких:

- «dataset_for_training» - для навчання моделей.
- «dataset_for_check» - для перевірки моделей.
- «dataset_mini» - для дослідження даних.

На рисунку 3.4 графічно зображено розподіл вхідних даних.

Рисунок 3.4 – Розподіл вхідних даних об'єкту дослідження.

```
dataset = pd.read_csv('C:/Users/llord/OneDrive/Рабочий стол\Diplom
Analitika/CombinedFile.csv', index_col=['Date'], parse_dates=['Date'])
# Видалити всі стовпці крім "Date" та "Power"
dataset = dataset.drop(columns=['Voltage', 'Current'])
dataset = dataset.astype(np.float16)

# Розділ даних
dataset_for_training = dataset['2023-07-01T00:00:00':'2023-08-03T23:59:00']
dataset_for_check = dataset['2023-08-04T00:00:00':'2023-08-10T23:59:00']
dataset_mini = dataset['2023-07-25T00:00:00':'2023-08-03T23:59:00']

dataset.head()
```

Тепер можемо провести візуальну оцінку даних використовуючи dataset_mini. В попередньому розділі вже була мова за сезонність, в наших даних вона сильно виражена, в не залежності від стану погоди чи інших зовнішніх факторів. А пов'язана виключено з настанням світлового дня. Фактор впливу лінійний та легко прогнозуємий простими системами. Тому що схід та захід сонця змінюється пропорційно дню року й модель дуже швидко це побачить з вхідного набору даних для навчання. Данні які зчитуються, можуть мати досить суттєві шуми, які з'являються в результаті дії зовнішніх факторів на систему дослідження. Вони негативно будуть впливати на прогноз, адже якщо система не може співставити екзогенний параметр та дані часового ряду, й знайти закономірність, то для неї це розцінюється як аномалія. В нашому випадку перші дві системи не вміють опрацьовувати зовнішні параметри, тому необхідно буде боротися з цим явищем в даних.

Останній компонент який можемо візуально оцінити це тренд. Він одночасно присутній, алей й в той час відсутній. Чітко видно тренд на зростання в першу половину дня, та на спад у другу половину дня. Але загальний тренд дорівнює нулю. Оскільки метою прогнозу є мінімум доба, можемо стверджувати що тренд відсутній взагалі, адже за період доби дані повертаються в туж точку шкали з якої починали. На малюнку 3.5 виділені аспекти візуальної оцінки часового ряду. Код блоку для відображення графіку зображено нижче.

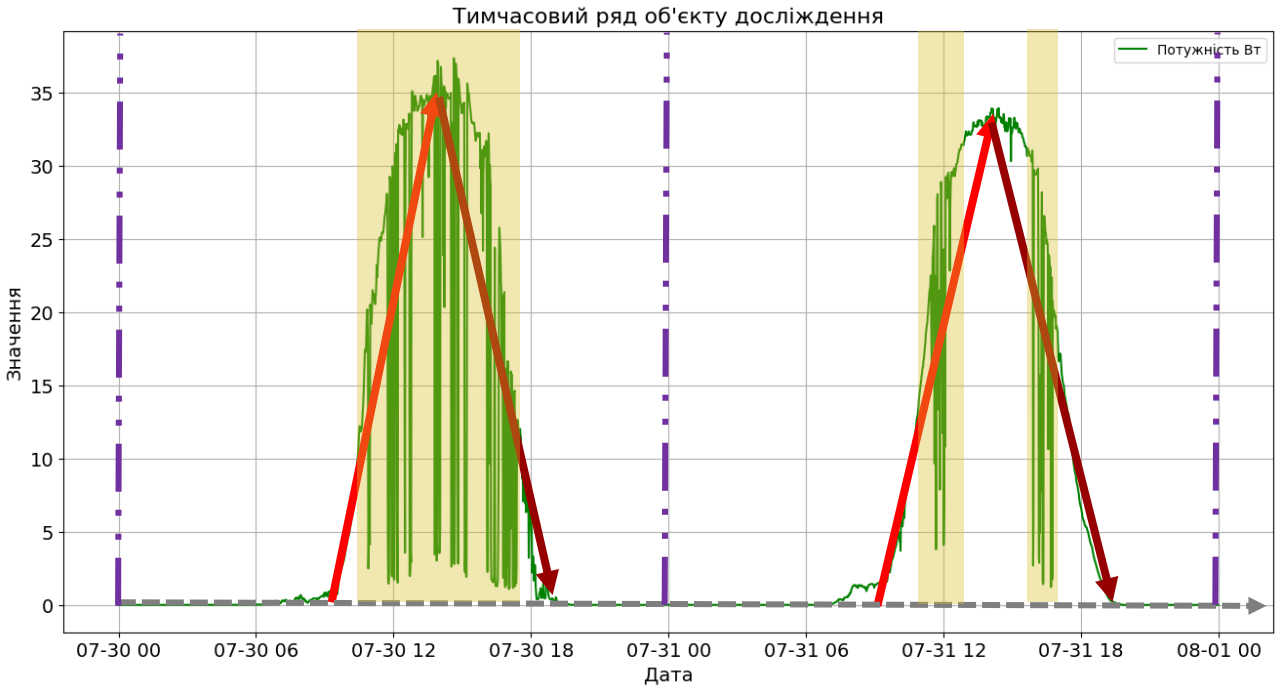


Рисунок 3.5 – Візуальний аналіз часового ряду.

```
plt.figure(figsize=(16, 8))
plt.plot(dataset_mini['2023-07-30T00:00:00':'2023-07-31T23:59:00'],
label='Потужність Вт', color='green')

plt.legend()
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.title('Тимчасовий ряд об\'єкту досліждення', fontsize=16)
plt.xlabel('Дата', fontsize=14)
plt.ylabel('Значення',fontsize=14)
plt.grid(True)

plt.show()
```

Перша ідея яка може стати на думку це використовувати усереднення для згладжування шумів. Спробуємо побудувати ці данні але згладжуючи по 5 хв та по 30 хв. На рисунку 3.6 зображено результат згладжування. З використанням довірчих інтервалів на рівні довіри близької до 95%. Виходячи з даних: при використанні великого вікна втрачається досить суттєва точність вхідного ряду й метод рухливого тренду не є ефективним для точної оцінки потужності; при маленькому розмірі нам вдається знизити рівень викидів приблизно на 43% та не втрачати загальної точності даних.

Потрібно не забувати, що було припущено не ефективність методів прямого усереднення, тому використовуються системи з інтегруванням та методом Брутлага.

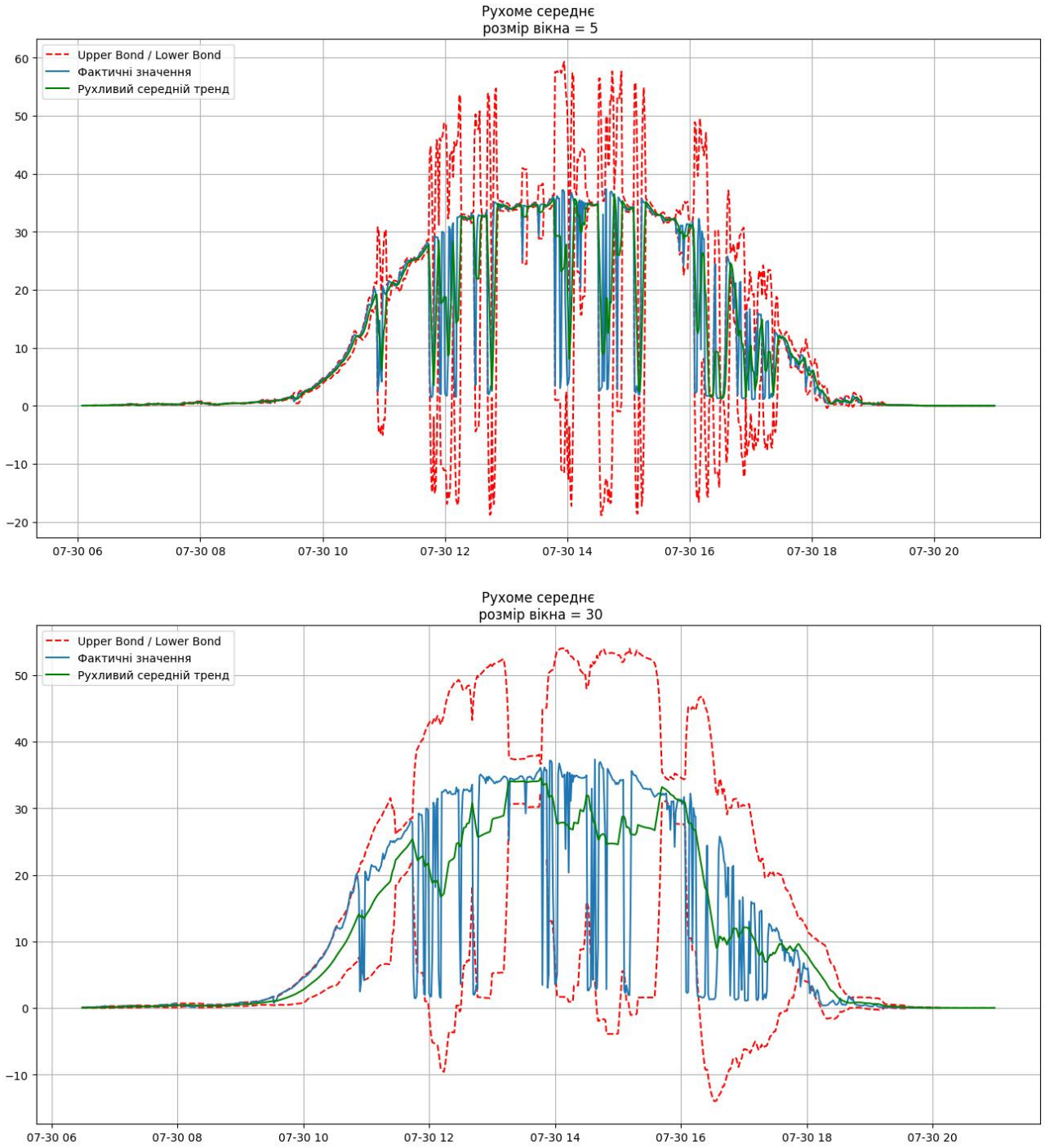


Рисунок 3.6 – Оцінка роботи згладжування.

Оскільки інтегрування є складовою моделі для прогнозу, а метод Брутлага вбудований в модель, то можна розпочати з побудови першої моделі. Виходячи с теоретичного розділу перша частина є експоненційним згладжуванням, це те

що необхідно для боротьби з викидами. Пам’ятаємо модель Холта-Уінтерса має 3 коефіцієнта a, β, γ. Зараз побудуємо перший модуль лише з коефіцієнтом a.

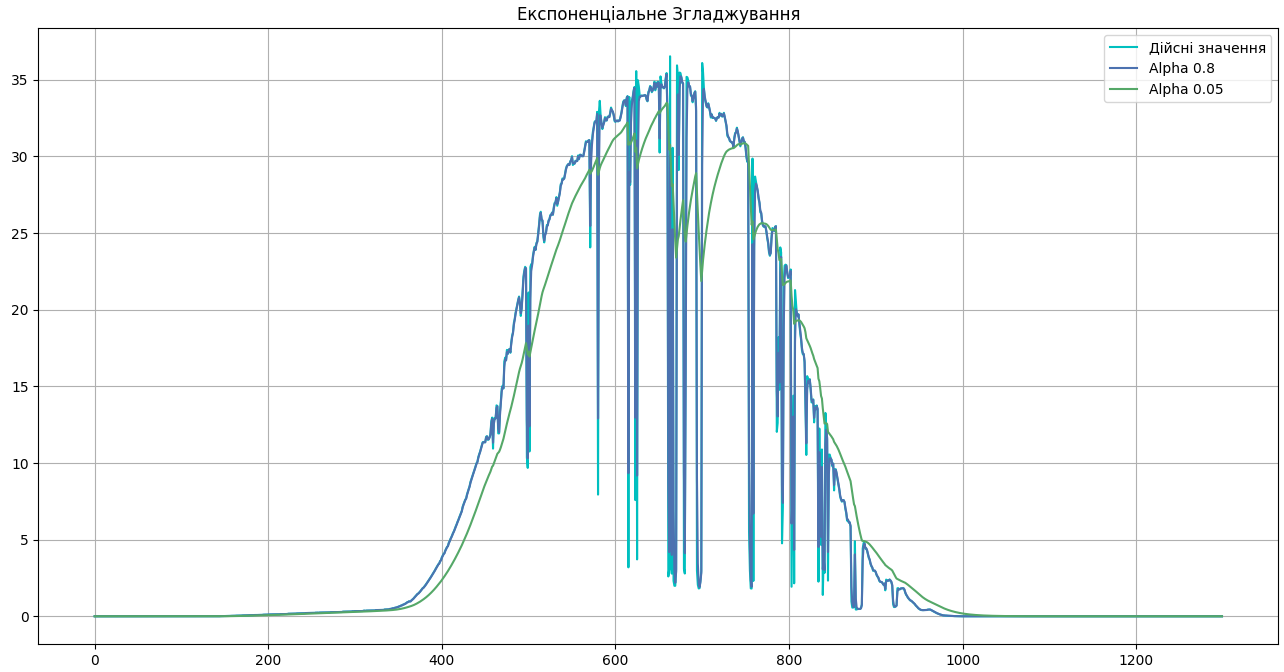


Рисунок 3.7 – Перша частина моделі, вплив коефіцієнту a.

З малюнку 3.7 видно як впливає згладжування та коефіцієнти на ряд даних, велике значення a робить згладжування чутливим до змін в то й час як маленький коефіцієнт навпаки, в подальшому параметр буде підібраний правильно.

Далі пам’ятаємо, що модель не враховує лінійні тренди, і тому додається згладжування Холта, а модель буде приймати вже два коефіцієнти a, β. В нашому випадку лінійний тренд відсутній тому ця частина моделі не є потрібною, але все ж таки вона трохи збільшує точність у місцях зростаючого та спадаючого тренду протягом доби. Високе значення β створює збурювальний вплив який видно на тестовому малюнку 3.8. Мале ж значення коефіцієнту, створює ефект «перерегулювання», це свідчить про відсутність постійного тренду в ряду даних. В теоретичному розділі було сказано що системи без урахування сезонності не підходять для об’єкту. Можемо це дуже легко підтвердити, використовуючи цю неповноцінну модель, та спробуємо спрогнозувати 500 точок на перед.

Видно що адекватного прогнозу не відбувається, і це очікуваний результат.

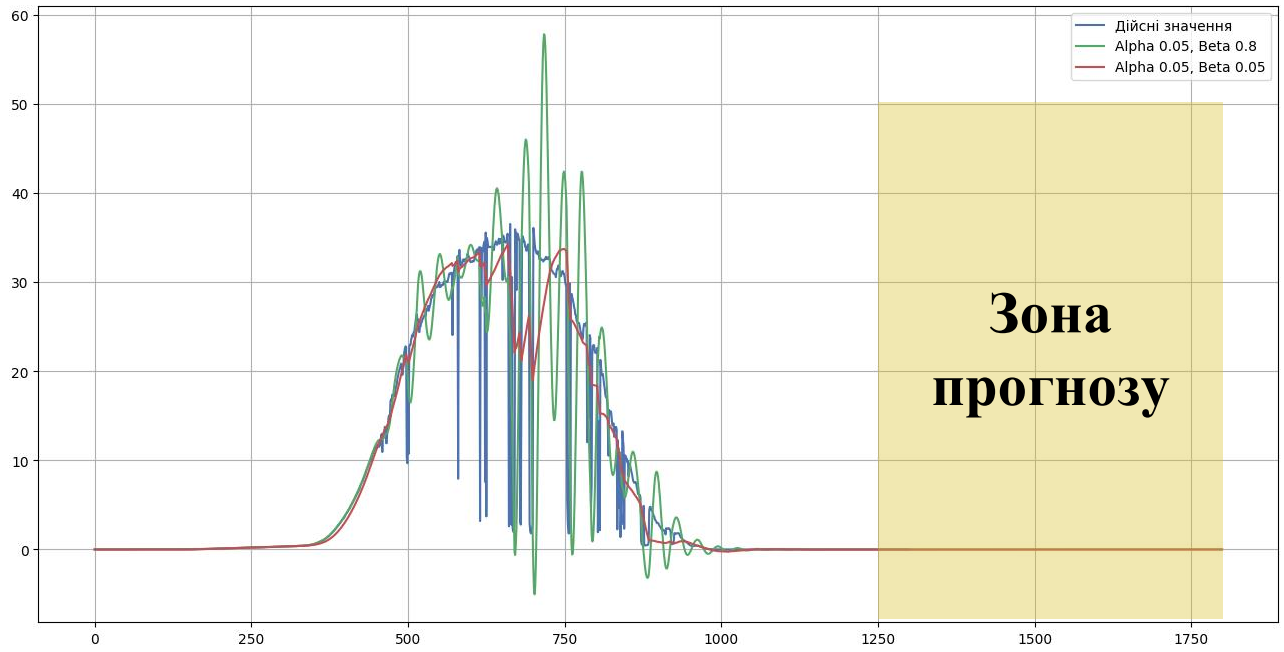


Рисунок 3.8 – Частина моделі з згладжуванням Холта, вплив коефіцієнту β.

Необхідно додати сезонну компоненту, але перед цим потрібно провести дослідження ряду на стаціонарність. Існують різні методи перевірки. У роботі використовується тест ADF. Augmented Dickey-Fuller - це статистичний тест, який використовується для визначення, чи є часовий ряд стаціонарним, тобто чи має ряд сталу структуру в часі. Стаціонарність важлива для багатьох методів аналізу часових рядів і моделювання прогнозів. Тест ADF перевіряє, чи є одиничний корінь в часовому ряді (тобто, чи є ряд нестаціонарним). Якщо результат тесту показує, що одиничного кореня немає, це свідчить про наявність стаціонарності в ряді. У зворотному випадку, якщо є одиничний корінь, це означає, що ряд є нестаціонарним, і його потрібно піддати подальшій обробці або моделюванню для забезпечення стаціонарності.

В одній із бібліотек Python є набор функцій для дослідження, створимо блок та відобразимо дані не тільки в числовому, а й в графічному вигляді. Для дослідження використаємо дані для навчання, всього їх 48960 значень. Результатами тесту будуть наступні:

```
ADF Statistic: -8.282144134474073
p-value: 4.489790264961624e-13
Critical Values: {'1%': -3.430483754184737, '5%': -2.8615991165732115, '10%': -
2.566801465801894}
```

```
# Проведемо тест ADF для визначення стаціонарності
result = sm.tsa.adfuller(dataset_for_training.Power)

# Виведемо результати тесту ADF
print("ADF Statistic:", result[0])
print("p-value:", result[1])
print("Critical Values:", result[4])

# Побудуємо графіки ACF і PACF
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(16, 8))
sm.graphics.tsa.plot_acf(dataset_for_training.Power, lags=48, ax=ax1)
sm.graphics.tsa.plot_pacf(dataset_for_training.Power, lags=48, ax=ax2)
plt.show()
```

Про що свідчать отримані результати тесту ACF:

- ADF Statistic: -8.282144134474073 Включає в себе коефіцієнти регресії, які показують, як змінюються значення ряду після диференціації. Зазвичай, якщо ADF Statistic від'ємне і має велику величину за модулем, це свідчить про стаціонарність ряду. Значення ADF Statisti повинно бути менше за 1%, 5%, 10%-кові критичні значення.
- p-value: 4.489790264961624e-13 - це ймовірність того, що нульова гіпотеза про не стаціонарність ряду є правильною. У нашому випадку, дуже мале значення (порядку e-13) означає, що нульову гіпотезу відхиляють, і ряд вважається стаціонарним. Якщо ж значення більше за 0,05 то ряд не стаціонарний.
- Critical Values: {'1%': -3.430483754184737, '5%': -2.8615991165732115, '10%': -2.566801465801894}. Критичні значення допомагають нам порівнювати ADF Statistic. У нашому випадку, ADF Statistic значно менше, ніж критичні значення для 1%, 5% та 10%. Це також свідчить на користь стаціонарності ряду.

Тепер стосовно графіків (Рис. 3.9) ACF та PACF:

- ACF вона же «Автокореляційна Функція» вказує на кореляцію між значеннями часового ряду та його лагами. Зазвичай, важливим є те, які значення ACF перетнуть нульову лінію (горизонтальну вісь). У

нас є такі значення, при тому вони досить довгі на проміжку лагу в 48 одиниць, це свідчить про впливову сезонно складову, тобто вказує на те що сезонний аспект для моделі прогнозування буде найважливішим. У подальшому буде проведено схожий тест у момент підбору параметрів у SARIMA моделі.

- PACF вона же «Часткова Автокореляційна Функція», про що свідчить: бачимо що перший лаг дорівнює одиниці, другий майже 1 а далі різкий спад вниз, таке явище свідчить про сильну лінійну залежність між значенням часового ряду. Досить багато подальших точок знаходяться в зоні довіри (синій сектор). Вказує на сильну залежність між лагами часового ряду.

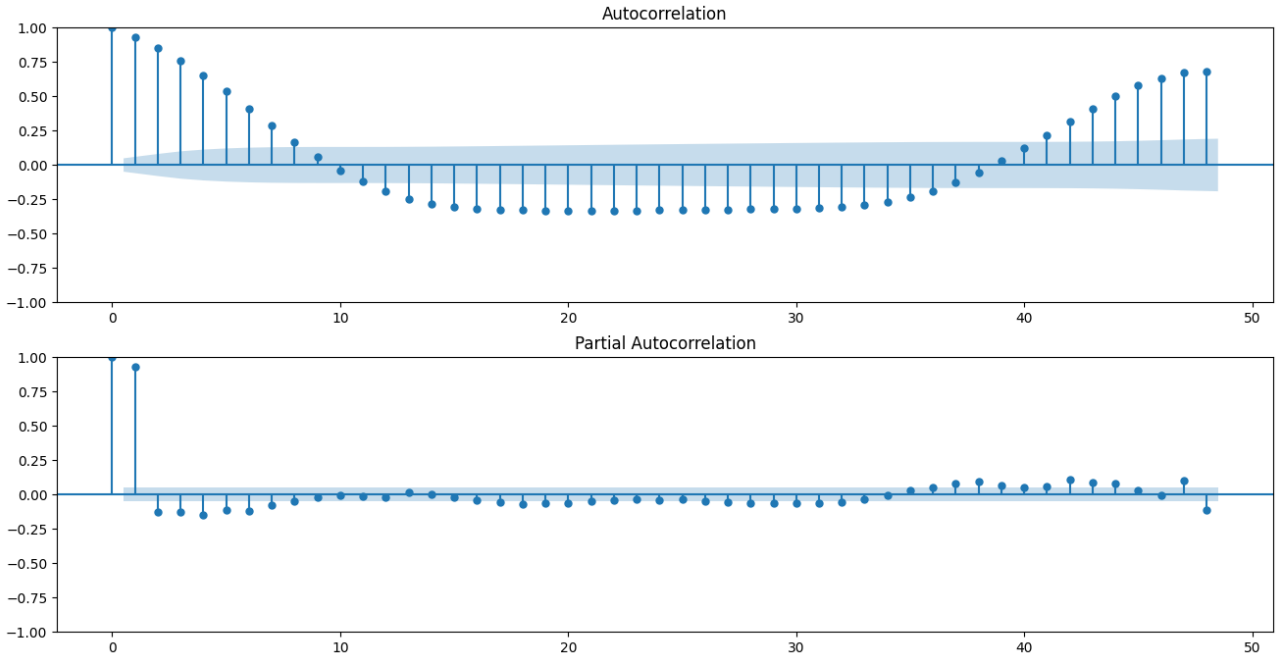


Рисунок 3.9 – Результати ACF та PACF.

Підведемо проміжний підсумок, модель Холта-Уінтерса на вхід отримає данні з сильною лінійною кореляцією, з домінуючою сезонністю, та досить великою кількістю аномалій, відсутнім трендом, з явним свідченням стаціонарності. Потрібно розуміти, що обсяг вивчення часового ряду не закінчений, для моделі SARIMA та SARIMAX необхідно продовжити вивчати ряд, але для першої моделі цієї інформації достатньо особливо коли

використовуємо один із надійніших методів підбору параметрів, а саме «Кросс-валідація» рисунок 3.10.

Рисунок 3.10 – Процес валідації коефіцієнтів.

Валідація неможлива без робочої моделі, додамо останню сезонну компоненту. Оскільки модель займає багато місця, дивіться «Додаток А», саме там приведений код усієї моделі разом з методом Брутлага, у виді блоку Jupiter Notebook. Код проходження крос-валідації та його виклик:

```
# імпорт необхідної бібліотеки
from sklearn.model_selection import TimeSeriesSplit

def timeseriesCVscore(x):
    """
    Функція для оцінки моделі методом Хольта-Вінтерса з використанням крос-валідації на часових рядах.
    Параметри:
    x (list): Вектор параметрів [alpha, beta, gamma].
    Повертає:
    float: Середній квадрат помилки по всіх фолдах крос-валідації.
    """
    # створення порожнього списку для зберігання помилок
    errors = []
```

```
    # отримання часового ряду з даних
    values = data.Power

    # розпакування параметрів alpha, beta, gamma
    alpha, beta, gamma = x

    # створення об'єкта для крос-валідації з 10 фолдами
    tscv = TimeSeriesSplit(n_splits=10)

    # ітерація по фолдах крос-валідації
    for train, test in tscv.split(values):

        # створення та навчання моделі Хольта-Вінтерса на тренувальних даних
        model = HoltWinters(series=values[train], slen=48, alpha=alpha,
beta=beta, gamma=gamma, n_preds=len(test), scaling_factor=1)
        model.triple_exponential_smoothing()

        # отримання прогнозів моделі
        predictions = model.result[-len(test):]
        actual = values[test]

        # розрахунок помилки за допомогою середнього квадрата помилки
        error = mean_squared_error(predictions, actual)
        errors.append(error)

    # повертаємо середній квадрат помилки по всіх фолдах крос-валідації
    return np.mean(np.array(errors))
```

Виклик валідації:

```
%%time
x = [0, 0, 0]

opt = minimize(timeseriesCVscore, x0=x, method="TNC", bounds = ((0, 1), (0, 1),
(0, 1)))

alpha_final, beta_final, gamma_final = opt.x
print(alpha_final, beta_final, gamma_final)
```

Спочатку виконаєм перевірку моделі з випадковими параметрами a, β, γ, та побудовою прогнозу на 3 дні вперед. $a = 0.5, \beta = 0.9, \gamma = 0.2$. З графіку (Рис. 3.11) видно що прогноз відбувається але не вдало, і це зрозуміло. Коефіцієнт β повинен дорівнювати 0 так як відсутній тренд, a мати набагато менше значення адже є зашумленість, γ більше так як сезонність сильно виражена. Дані для навчання не сильно співпадають з прогнозом навчання.

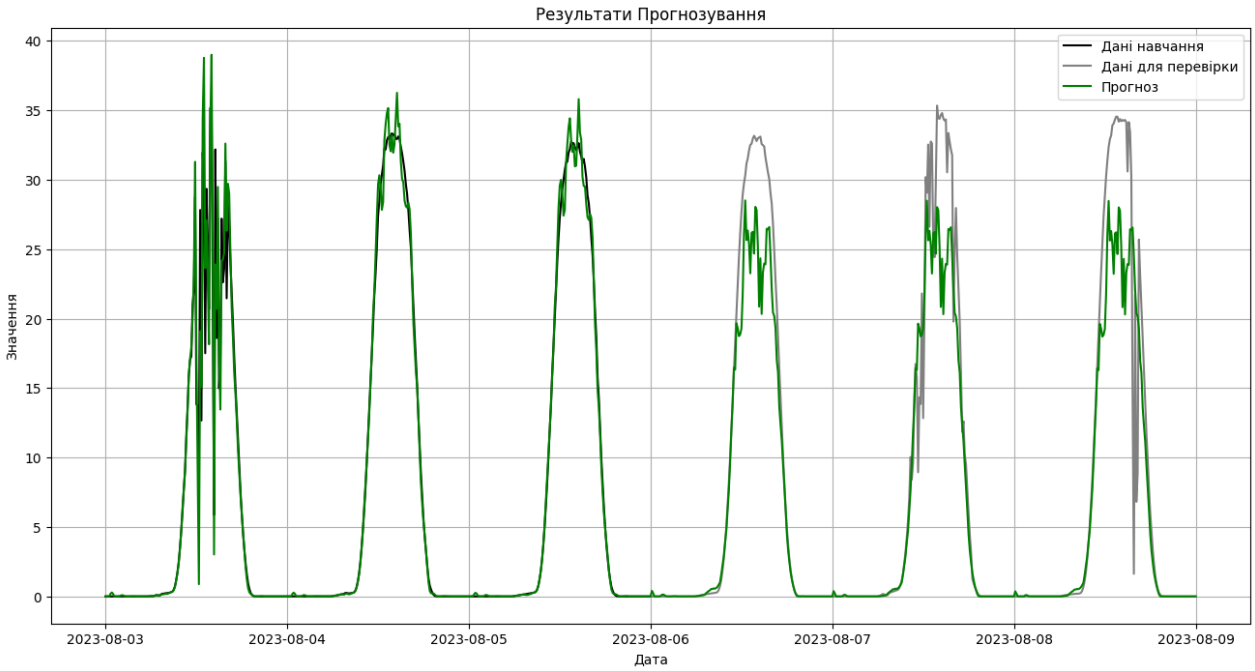


Рисунок 3.11 – Прогноз на 3 доби з невірними коефіцієнтами.

Але модель працює, а отже можемо запусти процес підбору параметрів, у результаті отримаємо:

```
0.005031358366300642 0.0008998829090258176 0.8
CPU times: total: 1min 56s
Wall time: 2min
```

Підбор параметрів займав дві хвилини. В результаті якого були отримані логічні і зрозумілі коефіцієнти, підставимо їх у модель та виконаємо той же прогноз. Результат на рисунку 3.12.

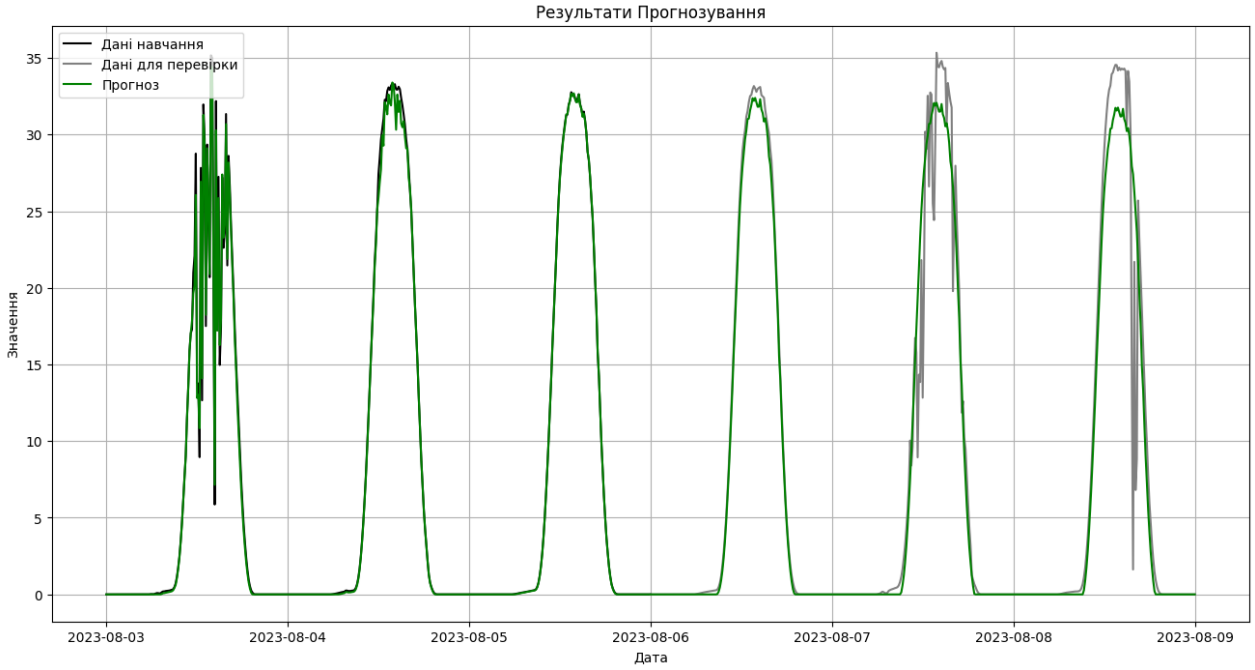


Рисунок 3.12 – Прогноз на 3 доби.

Бачимо результат реального прогнозу та порівняння його з даними об’єкту дослідження. Проміжок графіку обраний спеціально, на ньому добре видно, що маємо 3 сонячні дні, в двох з яких не ідеальна ситуація. Система не здатна передбачити такі викиди під час прогнозування, тому що ці моделі прогнозують виключно з попередніх даних. Також видно явну патернізацію сезону, тобто наступний день прогнозу схожий на попередній. Якщо уважно глянути присутнє затухання амплітуди прогнозу. С точки зору деталізації дані для прогнозу були опосередковані по 10 хвилин. Оскільки точок дуже багато, такі дані погано візуалізувати особливо у вигляді документу, у такому випадку крос-валідація займає більше 40-50 хвилин. Висновки стосовно кожної моделі виконаємо у відповідному підпункту розділу.

3.4 Реалізація та навчання моделі SARIMA

Реалізувати модель SARIMA простіше, адже є готова бібліотека. Оскільки частина дослідження часового ряду була зроблена при розробці попередньої моделі то можемо продовжити, не повторюючись. Спочатку необхідно створити ARIMA модель. Блок нижче автоматично виконає підбір параметрів завдяки кросс-валідації з підтримкою функцій мінімізації.

```
#Standard ARIMA Model
ARIMA_model = pm.auto_arima(dataset_mini['Power'],
                        start_p=1,
                        start_q=1,
                        test='adf',
                        max_p=3, max_q=8,
                        m=1,
                        d=None,
                        seasonal=False,
                        trace=True,
                        error_action='warn',
                        suppress_warnings=True,
                        stepwise=True)
```

Отримаємо наступний результат:

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=15105.270, Time=0.27 sec
...
 ARIMA(2,0,4)(0,0,0)[0] intercept   : AIC=15051.948, Time=2.25 sec
Best model:  ARIMA(1,0,3) intercept
Total fit time: 23.951 seconds
```

Отримали найкращі параметри моделі ARIMA (1, 0, 3) час підбору 24 секунди, з урахуванням того що використовувався dataset_mini. Будування моделі ARIMA необхідно для певного дослідження. В якості оптимізатора використовується AIC. Це статистичний критерій, який потрібен для порівняння моделей, особливо в статистичному моделюванні. Він дозволяє оцінити якість моделі та її адекватність до даних, враховуючи складність моделі. Зазвичай, при порівнянні моделей, вибирається та, яка має найменше значення AIC. Однак важливо пам'ятати, що AIC не є єдиним критерієм для оцінки моделей, і разом з ним можуть використовуватися інші статистичні критерії та методи оцінки моделей. Значення AIC дуже велике у всіх моделях і це зрозуміло, адже немає сезонної компоненти. Але ці дії виконувалися для того, що би перевірити параметри моделі. Побудуємо її, та отримаємо результат навчання.

```
best_p = ARIMA_model.get_params()['order'][0]
best_d = ARIMA_model.get_params()['order'][1]
best_q = ARIMA_model.get_params()['order'][2]
model_summary = ARIMA_model.summary()

print(model_summary)
print(f"Best ARIMA parameters: p={best_p}, d={best_d}, q={best_q}")
```

```
ARIMA Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 2880
Model:               ARIMA(3, 0, 4)   Log Likelihood                4108.053
Date:                Tue, 24 Oct 2023   AIC                          -8198.106
Time:                        13:40:57   BIC                          -8144.416
Sample:                    07-25-2023   HQIC                         -8178.754
                         - 08-03-2023
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0017      0.001      1.246      0.213      -0.001       0.004
ar.L1          1.1537      0.253      4.554      0.000       0.657       1.650
ar.L2          0.3631      0.444      0.818      0.413      -0.507       1.233
ar.L3         -0.5214      0.194     -2.688      0.007      -0.902      -0.141
ma.L1         -0.3382      0.254     -1.333      0.183      -0.835       0.159
ma.L2         -0.5040      0.239     -2.109      0.035      -0.972      -0.036
ma.L3          0.2027      0.028      7.152      0.000       0.147       0.258
ma.L4          0.0394      0.031      1.285      0.199      -0.021       0.099
sigma2         0.0034   2.91e-05    115.714      0.000       0.003       0.003
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):            138759.02
Prob(Q):                              0.97   Prob(JB):                         0.00
Heteroskedasticity (H):               1.26   Skew:                             0.61
```

Тепер можемо перевірити наш ряд та модель у режимі діагностики.

```
ARIMA_model.plot_diagnostics(figsize=(22,8))
plt.show()
```

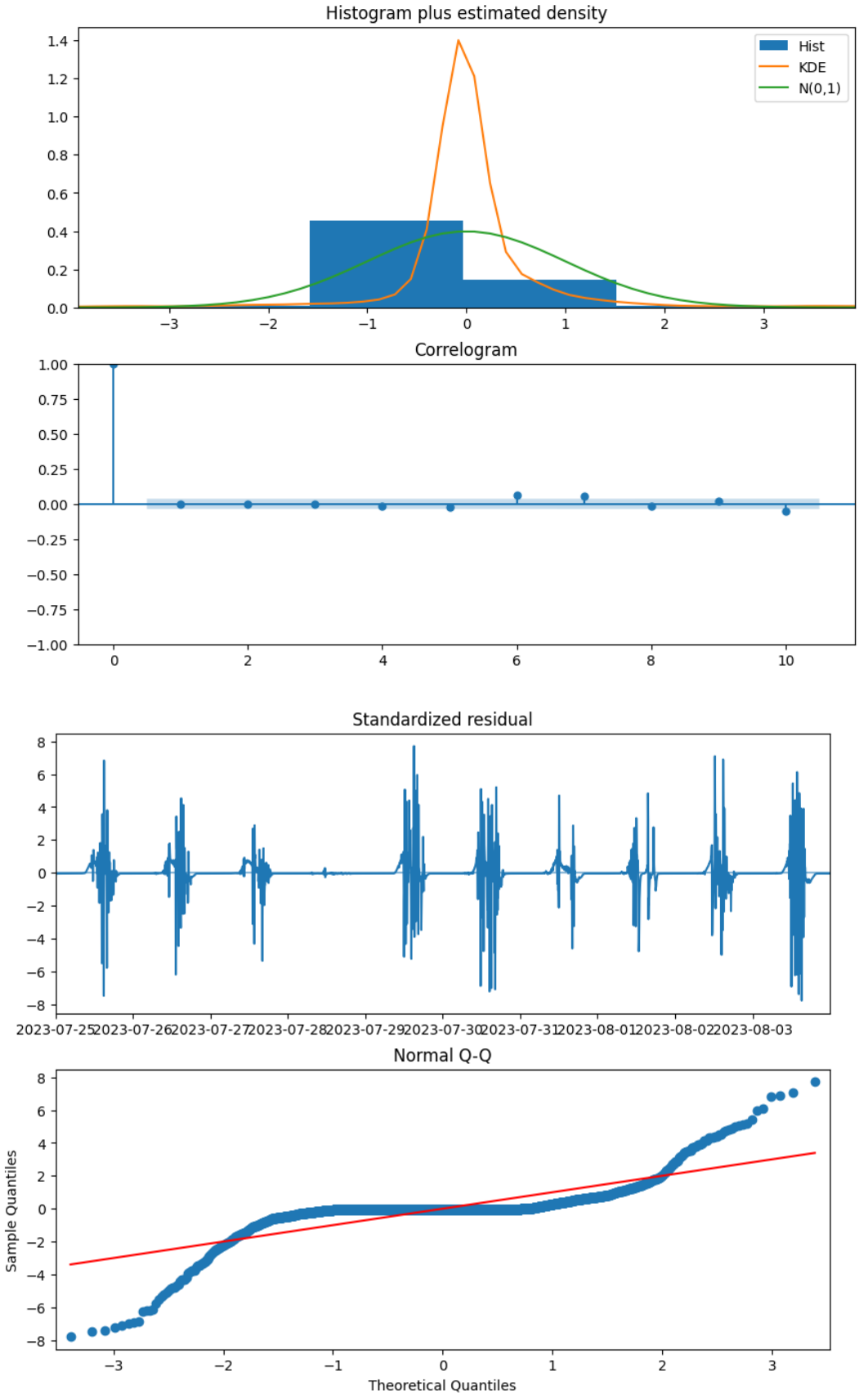


Рисунок 3.13 – Основні діагностичні данні моделі.

Ці графіки допомагають аналізувати, наскільки добре модель підходить для задачі прогнозування та чи є в ній які-небудь систематичні помилки.

«Histogram plus estimated density» або на українській «Гістограма з оціненою щільністю». Вона допомога визначити чи є розподіл залишків подібним до нормального. Якщо розподіл схожий на дзвіночок (bell-shaped), це вказує на те, що залишки майже нормально розподілені, що є важливою умовою для правильної роботи багатьох статистичних методів. Графік дозволяє виявити асиметрії та «тяжкі хвости». Асиметрія може вказувати на те, що є систематичні відхилення від припущень моделі. Хвости можуть вказувати на те, що є великі викиди або аномальні значення в залишках. Графік показує скільки залишків знаходиться біля нуля. Це важливо для підтвердження, що середній залишок дорівнює нулю, як вимагається ARIMA моделлю.

Графік кореляції використовуються с той же цілю, що й раніше. Єдине що можна додати, це те що зазвичай, значення, які виходять за межі інтервалу довіри (синій сектор), вказують на статистично значущу кореляцію.

"Standardized Residual" - це нормалізований залишковий аналіз, який використовується для оцінки адекватності моделі, особливо в контексті аналізу часових рядів, таких як ARIMA. Він допомагає визначити, наскільки великі і важливі залишки (похибки) в моделі та чи вони відповідають передбачуваним значенням. Такий аналіз результатів важливий для перевірки передбачень моделі та виявлення можливих аномалій або невідповідностей. Аналіз стандартизованих залишків допомагає визначити, чи модель правильно враховує основні закономірності в даних. Якщо стандартизовані залишки показують систематичний відхил від нуля, це може вказувати на недооцінку моделі або на наявність додаткових закономірностей, які не враховані в моделі.

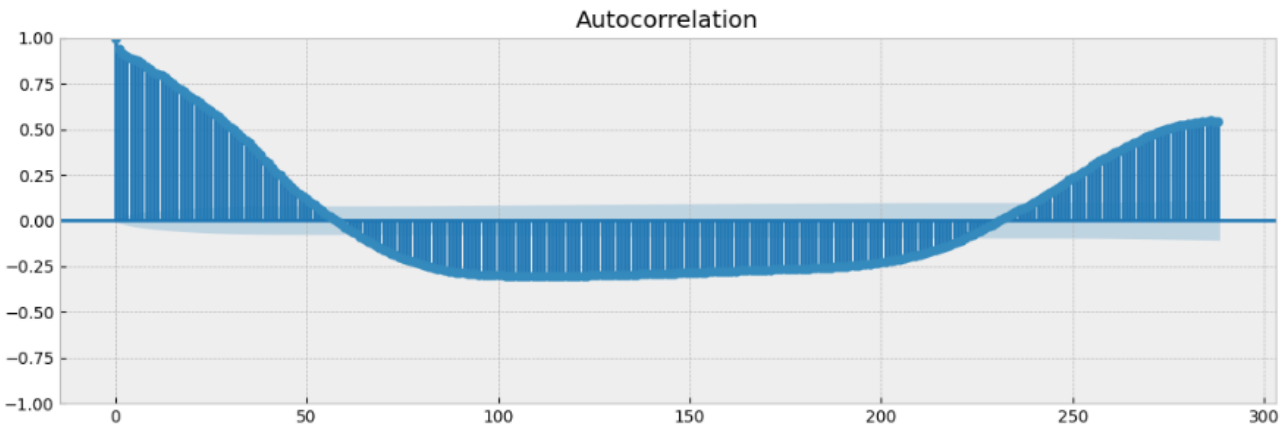
Графік "Normal Q-Q" (квантиль-квантиль) є важливим інструментом для оцінки та визначення нормальності розподілу даних. В контексті аналізу часових рядів, таких як ARIMA, він використовується для перевірки, наскільки відхилення залишків моделі від ідеально нормального розподілу. Основна ідея

графіка "Normal Q-Q" полягає в порівнянні квантилів спостережень з квантилями нормального розподілу.

Виходячи з графіку, відсутній нормальній розподіл, сильно виражена асиметрія, присутні хвости та викиди в даних. Звісно урахування сезонності вирівняє ці графіки, але кардинально картину не змінить. Нам необхідно використати перетворення Бокса-Кокса. Це метод згладжування та стабілізації дисперсії в часових рядах або інших даних. Перетворення може бути корисним для зменшення асиметрії та варіації в даних, щоб покращити прогнозування та моделювання. Перетворення Бокса-Кокса видаляє проблеми, пов'язані з дисперсією та асиметрією, шляхом застосування математичного перетворення до даних. Основна ідея полягає в тому, що вибір оптимального параметра перетворення залежить від даних і може бути знайдений статистичним шляхом.

```
def invboxcox(y,lmbda):
    if lmbda == 0:
        return(np.exp(y))
    else:
        return(np.exp(np.log(lmbda*y+1)/lmbda))

data = dataset_mini.copy()
data['Power'], lmbda = scs.boxcox(data.Power+1)
tsplot(data.Power, lags=288)
print("Оптимальний параметр перетворення Боксу-Коксу: %f" % lmbda)
```

На малюнку 3.14 показано автокореляцію даних до, та після використання перетворення. Оптимальний коефіцієнт -0.825928.



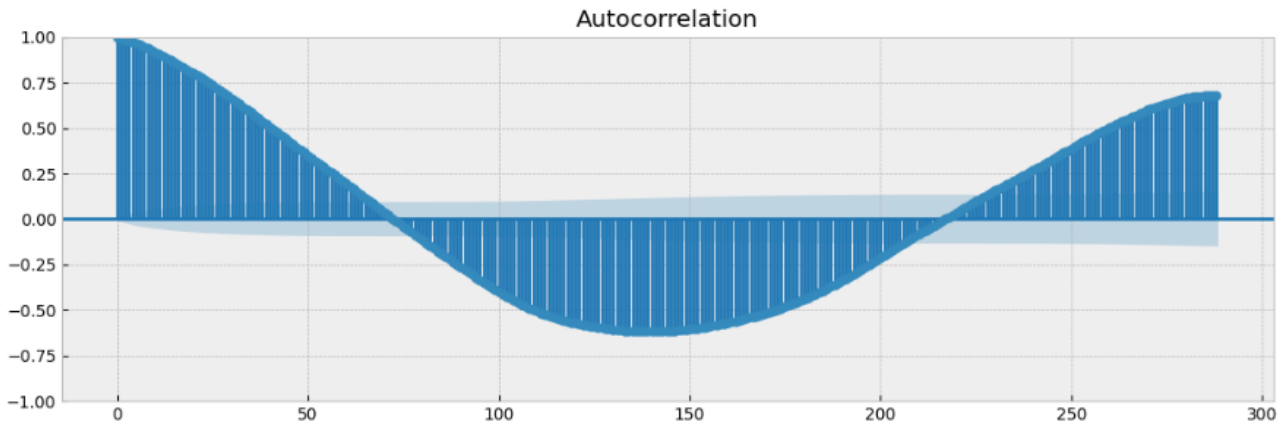


Рисунок 3.14 – Вплив перетворення на дані.

Тепер можем перейти до моделі SARIMA. І знову підбор коефіцієнтів але вже усіх 7, спосіб той самий. Для підбору параметрів заздалегідь виконаємо усереднення з 1440 точок до 144. Це вимушена міра обмеженості апаратних ресурсів. Поговоримо про це детальніше у відповідному розділі.

```
SARIMA_model = pm.auto_arima(dataset_mini['Power'], start_p=2, start_q=1,
                             test='kpss',
                             max_p=3, max_q=3,
                             max_P=3, max_Q=3,
                             m=144,
                             start_P=0,
                             seasonal=True,
                             d=None,
                             D=1,
                             trace=True,
                             error_action='ignore',
                             suppress_warnings=True,
                             stepwise=True)
```

```
Performing stepwise search to minimize aic
 SARIMA(2,0,1)(0,1,1)[144] intercept   : AIC=inf, Time=81.89 sec
 SARIMA(0,0,0)(0,1,0)[144] intercept   : AIC=3050.685, Time=0.54 sec
 SARIMA(1,0,0)(1,1,0)[144] intercept   : AIC=2439.123, Time=14.17 sec
 SARIMA(0,0,1)(0,1,1)[144] intercept   : AIC=inf, Time=56.66 sec
 SARIMA(0,0,0)(0,1,0)[144]             : AIC=3048.848, Time=0.22 sec
 SARIMA(1,0,0)(0,1,0)[144] intercept   : AIC=2555.495, Time=1.41 sec
 SARIMA(1,0,0)(2,1,0)[144] intercept   : AIC=inf, Time=32.05 sec
...
 SARIMA(2,0,1)(1,1,0)[144]             : AIC=2438.945, Time=17.04 sec

Best model:  SARIMA(1,0,0)(1,1,0)[144]
Total fit time: 758.074 seconds
```

Підбор параметрів виконувався 12 хвилин, найкраща модель (1,0,0)(1,1,0) 144. Виконаємо прогноз на 3 дні вперед.

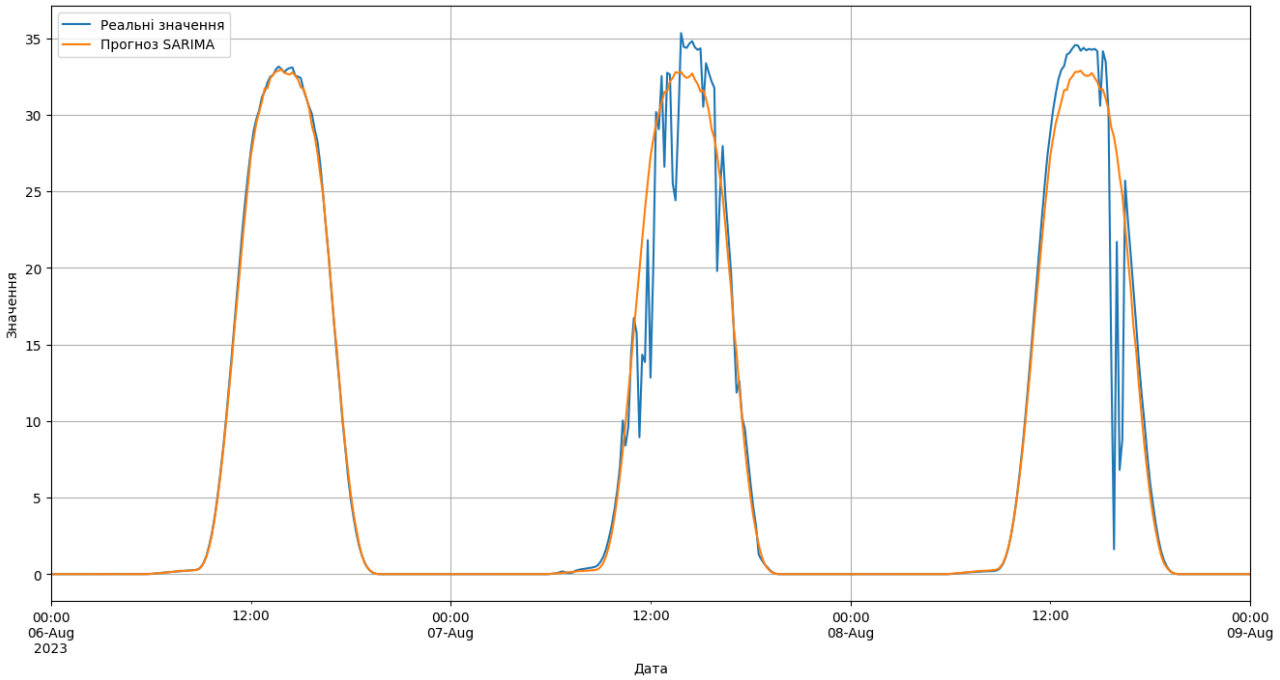


Рисунок 3.15 – Прогноз SARIMA.

Бачимо різницю у системі прогнозування. Відсутня патернізація, а отже можна допустити що цю систему можна використовувати не тільки на день вперед, а й на більші терміни. Хоча в реальності у цьому немає сенсу без екзогенного параметру. У зв’язку з більш складною моделлю, маємо більшу точність прогнозу. Детальніше у висновках.

3.5 Підготовка екзогенних параметрів, пошук залежностей

Для розробки систем з підтримкою екзогенних параметрів необхідно отримати самі параметри та зрозуміти, які з них впливають на систему, а які ні. Як розповідалось у теоретичному розділі для отримання історичних даних використовується ERA5.

Для безкоштовного по годинного отримання даних використовується GitHub проект (https://github.com/open-meteo), та відповідний сайт (Рис. 3.16).

Цей сервіс надає можливість гнучко налаштувати данні, які необхідно отримати відносно певних координат. Детальніше можете ознайомитися самостійно.

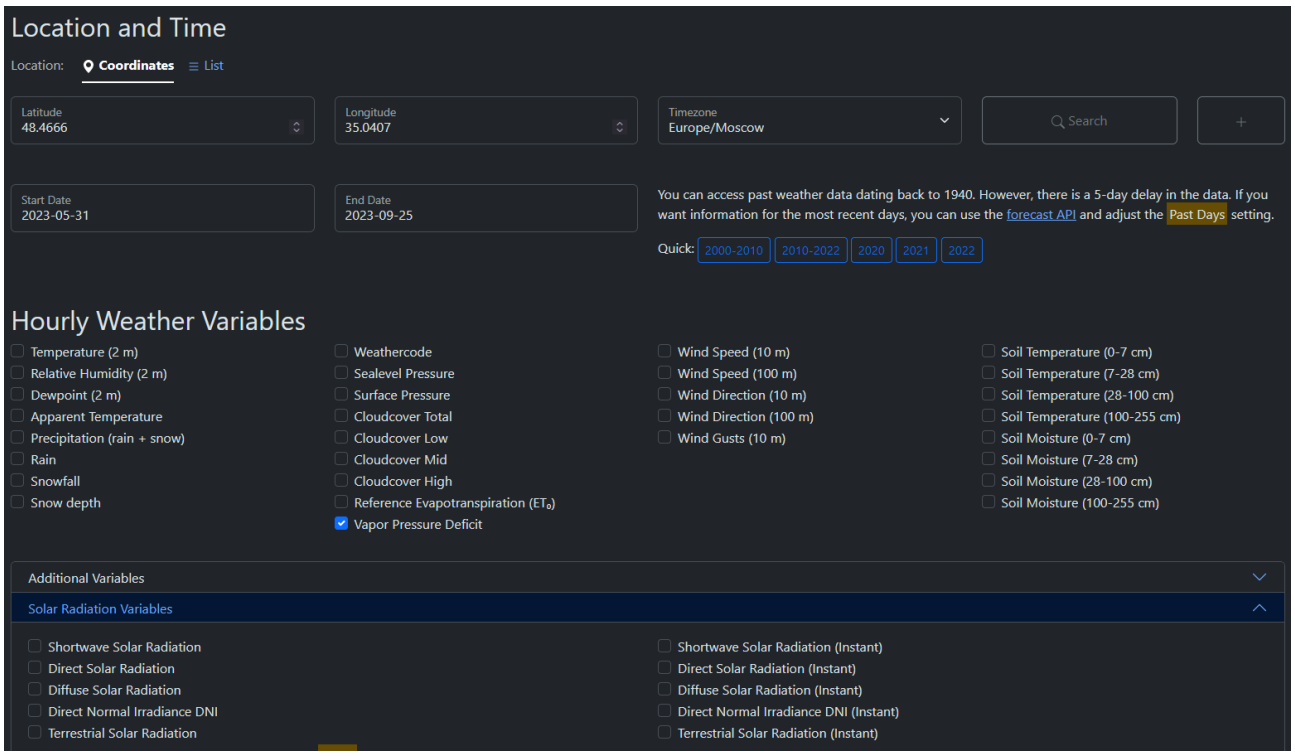


Рисунок 3.16 – Сервіс для отримання погодних даних з 1940 року.

Завантажимо усі данні в період від 31.06.23 по 25.09.23 без урахування явно не потрібних параметрів, таких як волога ґрунту й тому подібне. Отримаємо файл в форматі CSV Рис 3.17.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Date	Sun_azim	Sun_heigh	Cloud	cloudcove	cloudcove	cloudcove	shortwave	direct_rad	diffuse_ra	temperat	relativehu	precipitat	rain	snowfall	weatherc	pressure_	surface_p	direct_no	terrestr
2	2023-05-3	-12.8077	-146.266	100	99	94	73	0	0	0	15.9	84	0.3	0.3	0	51	1012.5	1005.2	0	
3	2023-05-3	-6.41272	-133.72	100	100	100	64	0	0	0	15.6	86	0.5	0.5	0	53	1012.6	1005.3	0	
4	2023-05-3	1.42567	-122.199	100	93	97	68	0	0	0	15.3	88	0.7	0.7	0	53	1012.5	1005.2	0	
5	2023-05-3	10.29538	-111.405	100	80	86	71	0	0	0	15.2	88	0.5	0.5	0	53	1012.4	1005.1	0	
6	2023-05-3	19.83758	-100.895	100	51	77	68	0	0	0	15	90	0.4	0.4	0	51	1012.5	1005.2	0	
7	2023-05-3	29.72446	-90.08	56	0	54	78	0	0	0	14.8	90	0.3	0.3	0	51	1012.8	1005.5	0	6
8	2023-05-3	39.60437	-78.1173	50	0	46	73	19	2	17	14.8	91	0.3	0.3	0	51	1013.3	1006	16	133
9	2023-05-3	48.9958	-63.6825	44	0	37	71	66	12	54	15.1	90	0.3	0.3	0	51	1013.5	1006.2	46.4	34
10	2023-05-3	57.07473	-44.679	37	0	22	80	222	116	106	16.1	84	0.2	0.2	0	51	1013.8	1006.5	277.6	556
11	2023-05-3	62.3582	-18.8917	32	0	11	83	388	233	155	17.3	77	0	0	0	1	1013.8	1006.6	410.4	755
12	2023-05-3	62.99275	11.68769	31	3	5	85	552	381	171	18.7	68	0	0	0	1	1013.9	1006.7	545.9	9
13	2023-05-3	58.68083	39.11416	30	4	0	89	652	502	150	19.9	62	0	0	0	1	1014	1006.8	627.8	1064
14	2023-05-3	51.11172	59.60508	33	6	0	92	723	561	162	20.9	57	0	0	0	1	1013.9	1006.7	647.9	1152
15	2023-05-3	41.94233	74.92004	30	1	0	98	741	554	187	21.7	53	0	0	0	1	1013.3	1006.2	621	1187
16	2023-05-3	32.12802	87.3337	32	3	0	96	729	520	209	22.4	49	0	0	0	1	1013	1005.9	593.2	1166
17	2023-05-3	22.20799	98.3358	30	1	1	95	689	482	207	23	46	0	0	0	1	1013.4	1006.3	587.5	105
18	2023-05-3	12.55272	108.8579	39	1	13	100	513	279	234	23	44	0	0	0	1	1012.8	1005.7	383.5	968
19	2023-05-3	3.489371	119.5376	46	0	28	98	363	169	194	22.9	45	0	0	0	1	1012.8	1005.7	279.8	803
20	2023-05-3	-4.63397	130.8498	51	0	35	100	294	156	138	22.5	50	0	0	0	2	1012.8	1005.7	340.4	6
21	2023-05-3	-11.4182	143.1313	48	0	30	99	173	84	89	21.6	56	0	0	0	1	1012.9	1005.8	279.5	4
22	2023-05-3	-16.4207	156.5195	44	0	23	100	60	20	40	20.2	60	0	0	0	1	1013	1005.8	137.1	188
23	2023-05-3	-19.2186	170.8464	40	0	17	98	4	0	4	18.9	62	0	0	0	1	1013.7	1006.5	0	32
24	2023-05-3	-19.5273	-174.399	35	0	10	98	0	0	0	18.5	68	0	0	0	1	1013.8	1006.6	0	
25	2023-05-3	-17.313	-159.91	34	0	7	98	0	0	0	17.6	69	0	0	0	1	1014.2	1007	0	
26	2023-06-0	-12.6979	-146.352	33	0	8	93	0	0	0	16.8	70	0	0	0	1	1014.3	1007	0	
27	2023-06-0	-6.31522	-133.818	36	0	19	83	0	0	0	16.1	71	0	0	0	1	1014.3	1007	0	
28	2023-06-0	1.512182	-122.306	49	6	34	76	0	0	0	15.6	71	0	0	0	1	1015	1007.7	0	
29	2023-06-0	10.37317	-111.521	61	2	67	63	0	0	0	15.3	70	0	0	0	2	1015	1007.7	0	

Рисунок 3.17 – Приклад отриманих даних.

Окрім зовнішніх даних потрібно отримати деякі внутрішні, такі як положення сонця у період з 31.06.23 по 25.09.23, для цього використовується

програма на мові програмування C#, дивіться «Додаток Б». Отримані данні були додані до основних на рисунку 3.17.

Тепер можна завантажити данні до Jupiter Notebook.

```
data_cloud = pd.read_csv('C:/Users/llord/OneDrive/Рабочий стол\Diplom
Analitika/Cloud.csv', index_col=['Date'], parse_dates=['Date'])
```

Оскільки данні отримано по годино, перетворим їх у данні по 10 хвилин через інтерполяцію. Використовуючи метод Akima.

```
data_cloud = data_cloud.asfreq('10T')
data_cloud = data_cloud.interpolate(method='akima')
data_cloud = data_cloud.astype(np.float16)

data_cloud_for_training = data_cloud['2023-07-01T00:00:00':'2023-08-
06T00:00:00']
data_cloud_for_check = data_cloud['2023-08-06T00:00:00':'2023-08-12T00:00:00']
data_cloud_mini = data_cloud['2023-07-31T00:00:00':'2023-08-06T00:00:00']

data_cloud_for_training = data_cloud_for_training.astype(np.float32)
data_cloud_for_check = data_cloud_for_check.astype(np.float32)
data_cloud_mini = data_cloud_mini.astype(np.float32)
```

У результаті розпаковки даних, бачимо їх кількість, формат, наявність пропусків:

```
DatetimeIndex: 2160 entries, 2023-06-20 00:00:00 to 2023-08-03 23:30:00
Freq: 30T
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Sun_azimuth             2160 non-null   float16
 1   Sun_height              2160 non-null   float16
 2   Cloud                   2160 non-null   float16
 3   cloudcover_low          2160 non-null   float16
 4   cloudcover_mid          2160 non-null   float16
 5   cloudcover_high         2160 non-null   float16
 6   shortwave_radiation     2160 non-null   float16
 7   direct_radiation        2160 non-null   float16
 8   diffuse_radiation       2160 non-null   float16
 9   temperature             2160 non-null   float16
...
 18  terrestrial_radiation   336 non-null    float16
dtypes: float16(19)
memory usage: 15.1 KB
```

Для пошуку залежностей використаймо графічне відображення даних та розрахуємо кореляцію для кожної з величин. Необхідно зазначити, що SARIMAX приймає один параметр. Тому необхідно обрати оптимальний. На

рисунку 3.18 графічно відображено залежність координат сонця та потужності. Потрібно враховувати наявність часового поясу, тому графік потужності візуально здвигаємо по осі часу відповідно до поясу.

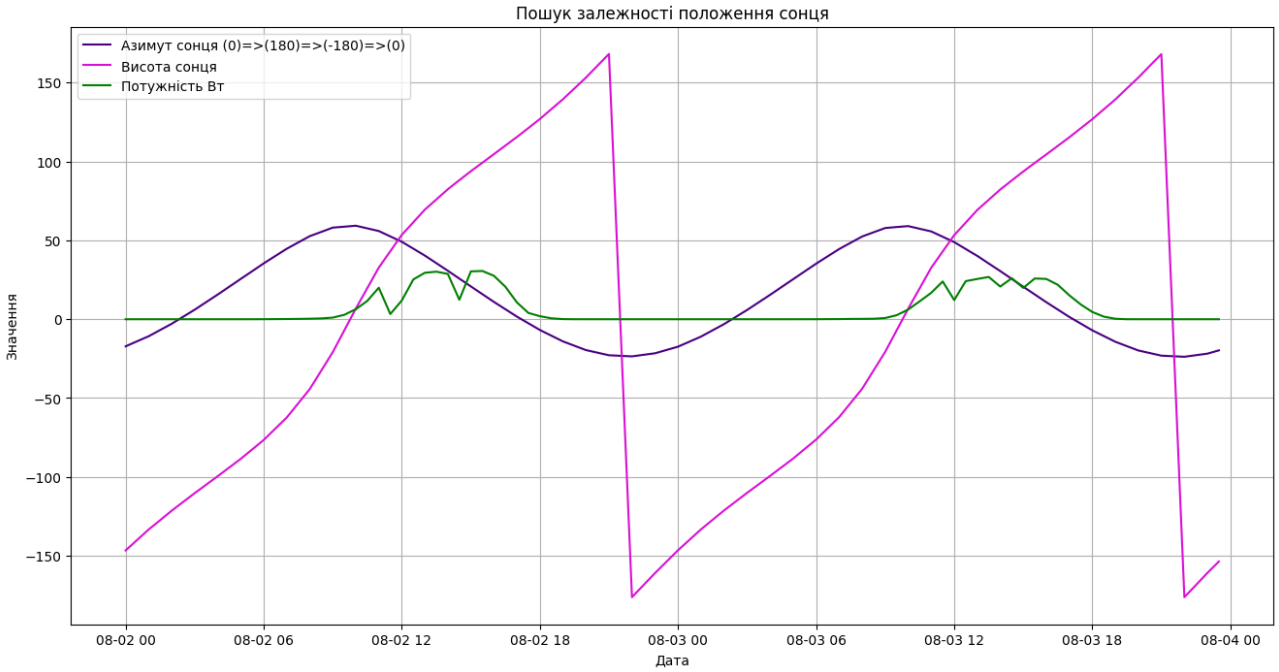


Рисунок 3.18 – Залежність від положення сонця.

Зрозуміло, що залежність присутня, але як говорилось раніше це статичні параметри. Їх використання доцільне в нейромережах, щоб ще трохи збільшити точність у виявленні сезону. Для моделі SARIMAX вони не підходять.

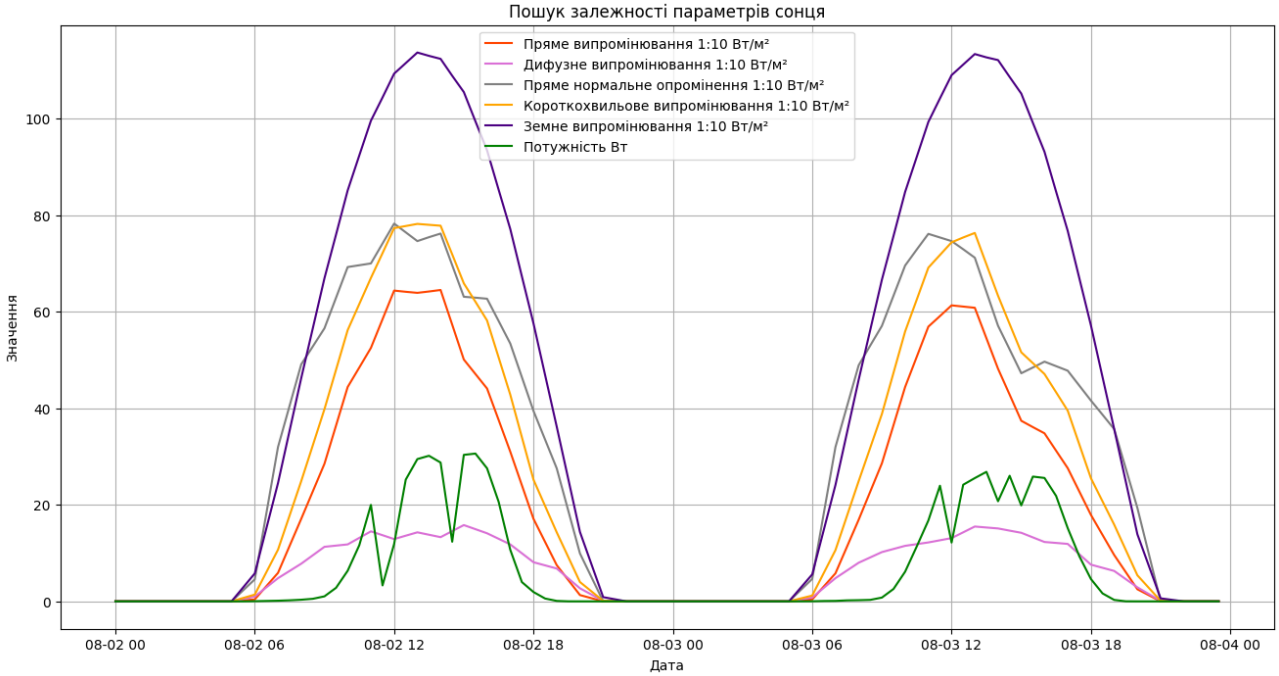


Рисунок 3.19 – Залежність від параметрів сонця.

Рисунок 3.19 показує залежність від параметрів сонячного випромінювання, а малюнок 3.20 від погодних умов.

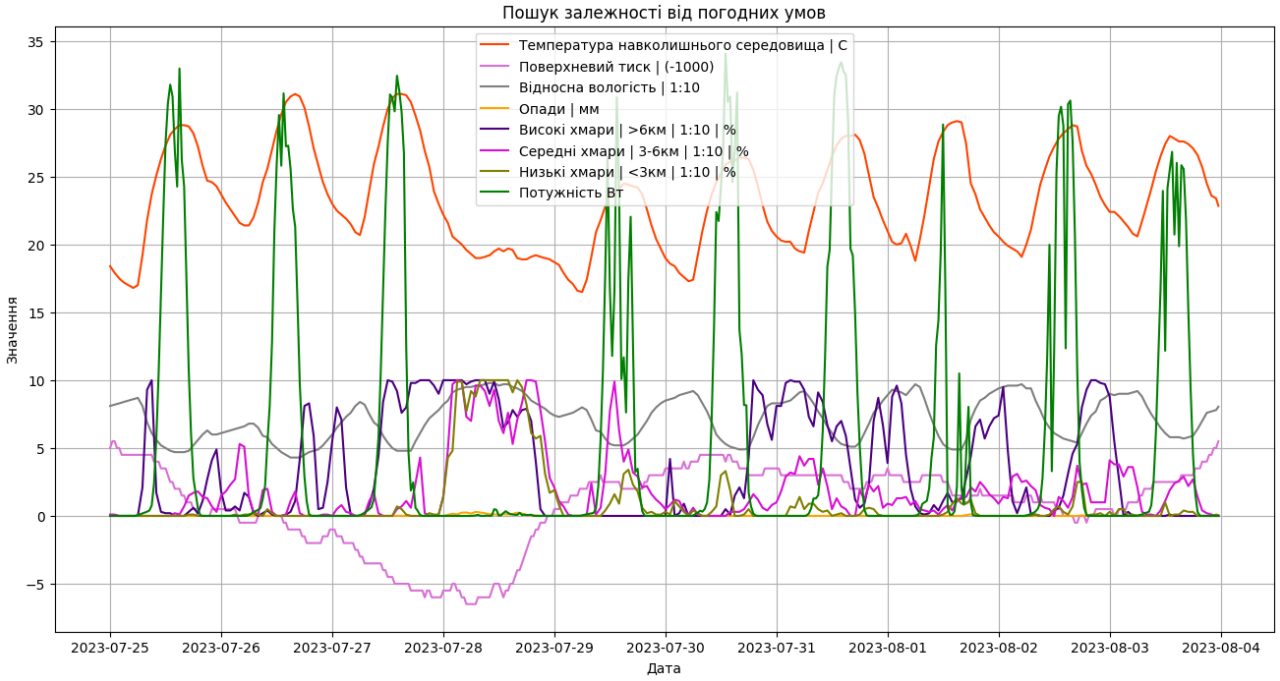


Рисунок 3.20 – Залежність від погодних умов.

Видно що хмарність не завжди прямо впливає на продуктивність видобутку енергії. Найменший вплив у середніх та низьких хмарах. Є певна суттєва залежність від високих хмар, але тут потрібно враховувати нюанс, що між поверхнею землі та шаром високих хмар 6 км. Це великий прошарок для відбиття сонячних променів, через місця відсутності хмарних мас. Для знаходження кореляції використовувався наступний блок коду, а результати відображенні у таблиці 3.2.

```
from scipy.stats.stats import pearsonr
x = dataset_for_training['Power'].values
y = data_for_RNN_for_training['terrestrial_radiation'].values

corr , p = pearsonr(x,y)

print ('Corelation Coefficient =', corr,'\nP-Value =',p)
```

Таблиця 3.2– Кореляція екзогенних параметрів

Назва параметру	**Кореляція**
Sun_azimuth	0.3446262516885366
Sun_height	0.42241813568642644

Назва параметру	Кореляція
Cloud	0.08661388278673368
cloudcover_low	0.027171407455616
cloudcover_mid	0.11070715786535137
cloudcover_high	0.1166856995711352
shortwave_radiation	0.8235891513396931
direct_radiation	0.8127739845210753
diffuse_radiation	0.6565441818819723
temperature	0.561460243788424
relativehumidity	0.6101297997405914
precipitation	0.09043005091413287
rain	0.09043000091416287
snowfall	nan
weathercode_WMO	0.07933589510352172
pressure_msl	0.08136278188958902
surface_pressure	0.09303102263523128
direct_normal_irradiance	0.6670995093988688
terrestrial_radiation	0.7511284839193855

Параметр з найбільшою кореляцією shortwave_radiation будемо використовувати в якості екзогенного параметру для SARIMAX. В той час для нейромереж будемо використовувати усі ці параметри.

3.6 Реалізація та навчання моделі SARIMAX

Реалізація SARIMAX не відрізняється від SARIMA. Досліджувати дані більше не потрібно, а коефіцієнти залишаються не змінними. Додається лише параметр exog, який і є зовнішнім масивом даних.

```
p = 1, d = 0, q = 1, P = 1, D = 1
Q = 0
s = 144

SARIMAX_model = sm.tsa.SARIMAX(dataset_mini, order=(p, d, q), seasonal_order=(P,
D, Q, s), exog = data_cloud_mini.shortwave_radiation )
```

Для побудови моделі необхідно виконати лише дві умови, це однаковий розмір масиву з даними потужності та екзогенним параметром. Та однакова довжина масиву екзогенних параметрів на навчання, та на прогноз. Навчимо модель.

```
SARIMAX_results = SARIMAX_model.fit()
SARIMAX_predictions = SARIMAX_results.get_prediction(start=pd.to_datetime('2023-08-05T00:00:00'), end=pd.to_datetime('2023-08-09T00:00:00'), dynamic=False, full_results=True, exog = data_cloud_for_check.shortwave_radiation['2023-08-04T00:00:00':"2023-08-09T00:00:00"])

SARIMAX_predictions.predicted_mean =
SARIMAX_predictions.predicted_mean.apply(lambda x: max(0, x)).round(5)
```

В результаті отримаєм прогноз на рисунку 3.21, порівняння моделей відбудиться у «Експериментальному Розділі». Точність від моделі SARIMA майже не відрізняється, однак можна зауважити що амплітуда другого дня нижча, так як значення екзогенного параметру випливає на загальне значення прогнозу. Кореляція в моделі досить мала, й це пов’язано з малою роздільною здатністю екзогенного параметру. 24 точки які лінійно інтерполюються у 144.

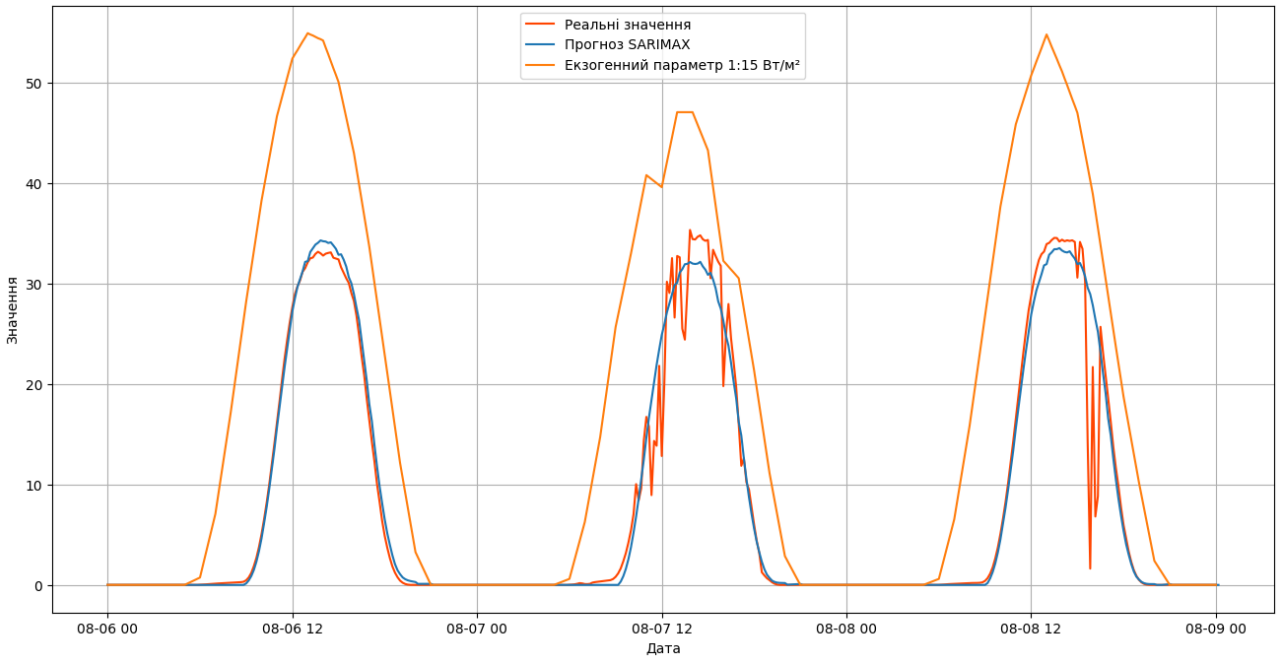


Рисунок 3.21 – Прогноз SARIMAX.

3.7 Реалізація та навчання нейронної мережі

Реалізувати нейромережу не складніше за ARMA-подібні моделі. А в певних моментах простіше. Нема необхідності підбирати коефіцієнти чи досліджувати часовий ряд. Достатньо визначитись з екзогенними параметрами, які впливають на об'єкт дослідження та основними параметрами нейромережі, які описують її складність.

Спочатку необхідно таким же чином завантажити данні у програму, але в цей раз dataset_mini вже не потрібен. Для початку спробуємо побудувати ідеальну модель, яка буде прогнозувати потужність за параметрами струму та напруги. Це одразу якісно покаже вплив параметрів нейромережі на прогноз.

Згідно з теоретичним розділом нам необхідно створити тензор, це робиться наступним блоком програми. У ньому кожен із параметрів для навчання перетворюється у відповідний тензор з типом комірки FP32. Вигляд тензору зображено нижче.

```
# Перетворення датафрейму у тензори.
voltage_tensor = torch.tensor(dataset_for_training['Voltage'].values,
dtype=torch.float32)
current_tensor = torch.tensor(dataset_for_training['Current'].values,
dtype=torch.float32)
power_tensor = torch.tensor(dataset_for_training['Power'].values,
dtype=torch.float32)
print ('Voltage ', voltage_tensor)
print ('Current ',current_tensor)
print ('Power ',power_tensor)
```

```
Voltage  tensor([0., 0., 0.,  ..., 0., 0., 0.])
Current  tensor([0., 0., 0.,  ..., 0., 0., 0.])
Power  tensor([0., 0., 0.,  ..., 0., 0., 0.])
```

Наступним кроком необхідно підготувати данні до об'єднання в один тензор та розбиття на batch він же «міні пакет». Представляє з себе підмножину даних із набору для навчання. І потрібен для оптимізації використання пам'яті, паралелізму розрахунків та стабільності у узагальнюванні.

```
batch_size = 2160
voltage_tensor = voltage_tensor.view(batch_size, -1)
current_tensor = current_tensor.view(batch_size, -1)
power_tensor = power_tensor.view(batch_size, -1)
```

Тепер об’єднуємо у єдиний тензор.

```
combined_input = torch.cat([voltage_tensor.unsqueeze(2),
current_tensor.unsqueeze(2)], dim=2)
combined_input
```

```
tensor([[[0.0000, 0.0000],
         [0.0000, 0.0000],
         [0.0000, 0.0000],
         ...,
...
         [0.0000, 0.0000],
         ...,
         [0.0000, 0.0000],
         [0.0000, 0.0000],
         [0.0000, 0.0000]]])
```

Далі необхідно створити саму нейромережу а точніше її клас, оскільки вона не розробляється з нуля, а використовуємо PyTorch, то зробити це дуже просто. У методі **__init__** конструктор класу визначає архітектуру моделі, передаючи різні параметри. В методі **forward** визначається процес передачі даних через модель.

```
class PowerPredictionRNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(PowerPredictionRNN, self).__init__()

        # RNN шар
        self.rnn = nn.RNN(input_size, hidden_size, batch_first=True)

        # Вихідний лінійний шар
        self.linear = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        # Пропускаємо вхідні дані через RNN
        out, _ = self.rnn(x)

        # Отримуємо вихідні дані з останнього тимчасового кроку та застосовуємо
лінійний шар.
        out = self.linear(out[:, -1, :])
        return out
```

Далі необхідно вказати параметри моделі та створити екземпляр нейромережі. Зрозуміло що на вхід будуть подаватися 2 параметри напруга та струм, а на виході лише один параметр потужність. Кількість схованих шарів поставимо маленьку, для експерименту його впливу.

```
input_size = 2
hidden_size_1 = 8
output_size = 1

model = PowerPredictionRNN(input_size, hidden_size_1, output_size)
```

Тепер перенесемо модель з центрально процесора на графічний з використанням відео буферу, а не оперативної пам’яті.

```
# Переносимо модель на GPU (якщо є CUDA)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
combined_input = combined_input.to(device)
power_tensor = power_tensor.to(device)
model.to(device)
```

```
PowerPredictionRNN(
  (rnn): RNN(2, 8, batch_first=True)
  (linear): Linear(in_features=8, out_features=1, bias=True)
)
```

У наступному фрагменті коду налаштовуються функція втрат (loss function) та оптимізатор для навчання нейронної мережі. **nn.MSELoss()** представляє собою функцію втрат, яка визначає величину помилки між прогнозованими значеннями і фактичними значеннями за допомогою середньоквадратичного відхилення (MSE). Вона обчислює квадрат відносних похибок для кожного прикладу в навчальному наборі і обчислює середнє значення цих квадратів. MSE є популярною функцією втрат для задач регресії, і завданням її оптимізації є зменшення помилок між прогнозами та фактичними значеннями.

```
criterion = nn.MSELoss()  # Середньоквадратична помилка (MSE) як функція втрат
optimizer = torch.optim.Adam(model.parameters(), lr=0.0001)  # Оптимізатор Adam
```

torch.optim.Adam представляє собою оптимізатор Adam, який використовується для оптимізації параметрів нейронної мережі під час навчання. Його параметри включають:

- model.parameters(): Це передача параметрів моделі для оптимізації. Оптимізатор змінює ці параметри, щоб зменшити функцію втрат.
- lr=0.0001: Це швидкість навчання (learning rate). Вона визначає крок, з яким оптимізатор оновлює параметри моделі після кожної ітерації

навчання. Низьке значення learning rate призводить до повільного навчання, а високе - до швидкого, але може призвести до збоїв. Оптимальне значення learning rate зазвичай залежить від конкретної задачі і може бути налаштоване експериментально.

Можемо почати навчання.

```
num_epochs = 100  # Кількість епох навчання

for epoch in range(num_epochs):
    optimizer.zero_grad()  # Обнуління градієнтів
    outputs = model(combined_input)
    loss = criterion(outputs, power_tensor)
    loss.backward()  # Розрахунок градієнту
    optimizer.step()  # Оновлення параметрів

    print(f'Епоха [{epoch+1}/{num_epochs}], Втрата: {loss.item():.4f}')
```

```
Епоха [1/100], Втрата: 123.9027
...
Епоха [97/100], Втрата: 122.2713
Епоха [98/100], Втрата: 122.2547
Епоха [99/100], Втрата: 122.2381
Епоха [100/100], Втрата: 122.2215
```

Тепер підготуємо тензори для прогнозу, тим же шляхом що й раніше.

```
last_voltage_data = dataset_for_check.Voltage
last_current_data = dataset_for_check.Current

last_voltage_data_tensor = torch.tensor(last_voltage_data.values,
dtype=torch.float32)
last_current_data_tensor = torch.tensor(last_current_data.values,
dtype=torch.float32)

input_data = torch.stack([last_voltage_data_tensor, last_current_data_tensor],
dim=1)
input_data = input_data.to(device)

predicted_power = model(input_data.unsqueeze(1))
predicted_power
```

```
tensor([[0.2141],
        [0.2141],
        ...,
        [0.2141],
        [0.2141],
              [0.2141]], device='cuda:0', grad_fn=<AddmmBackward0>)
```

Отримаємо прогноз, розмір прогнозу залежить від вхідного тензору і не задається іншим шляхом. Прогноз розпаковуємо з тензору у вигляді списку.

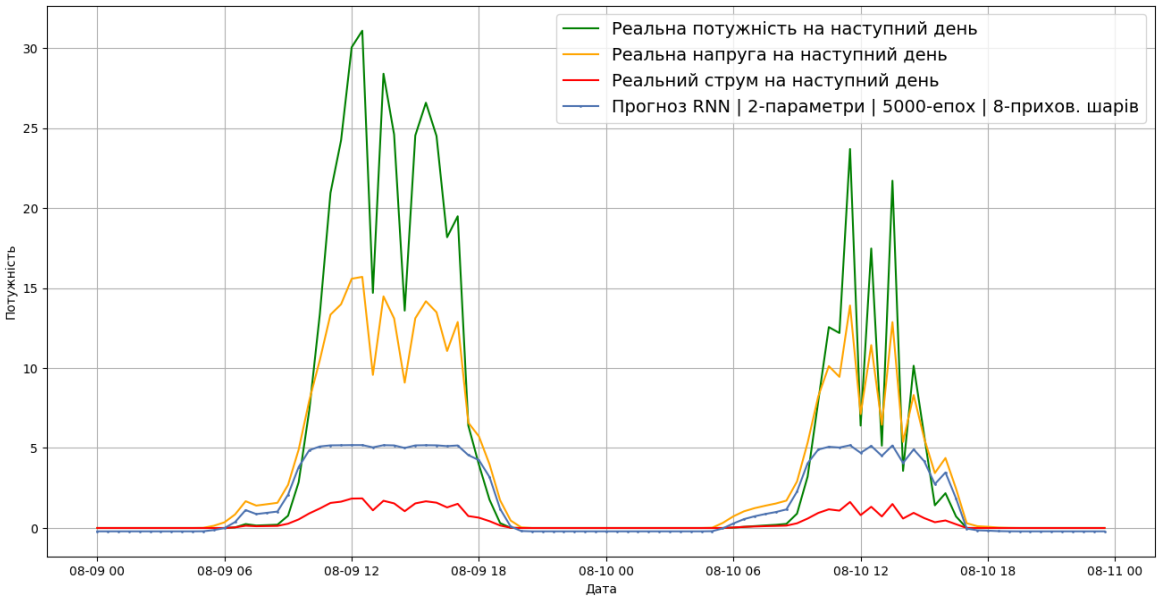
```
predicted_power_values = predicted_power.tolist()
predicted_power_values
```

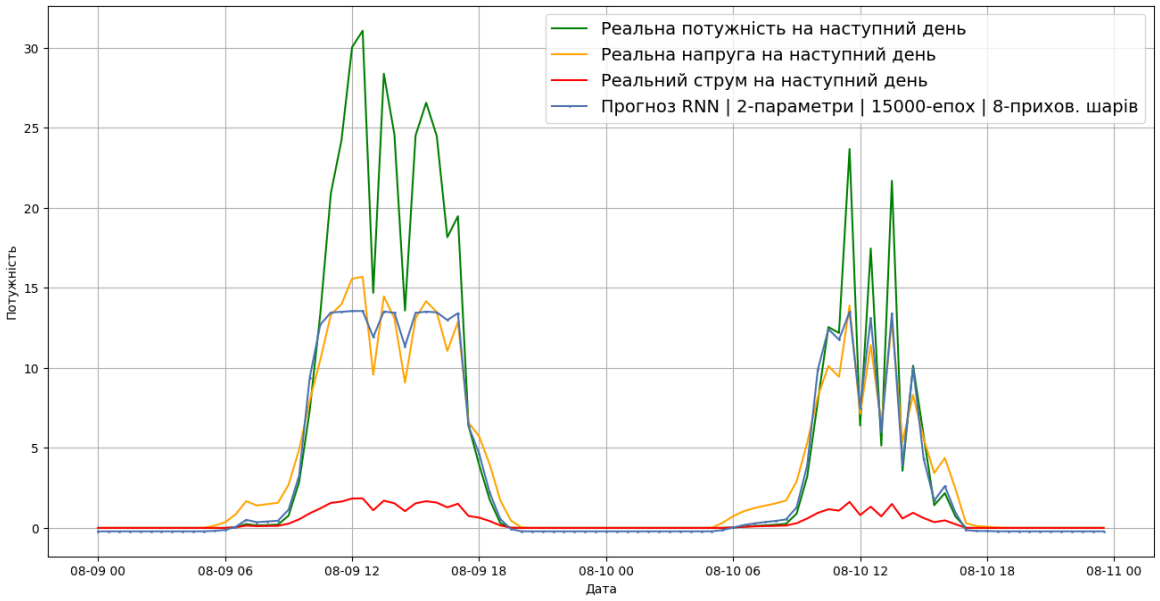
```
[[0.21407155692577362],
 [0.21407155692577362],
...
 [0.21407155692577362],
 [0.21407155692577362],
 [0.21407155692577362],
...]
```

Перетворимо список у дата фрейм та побудуємо графік прогнозу рисунки 3.21 – 3.23.

```
start_date = '2023-08-04T00:00:00'

date_range = pd.date_range(start=start_date,
periods=len(predicted_power_values), freq='30T')

dataset_power_values = pd.DataFrame({'Date': date_range, 'Power':
predicted_power_values})
dataset_power_values['Power'] = dataset_power_values['Power'].apply(lambda x:
x[0])

dataset_power_values.set_index('Date', inplace=True)
```

Рисунок 3.21 – Вплив параметрів RNN.





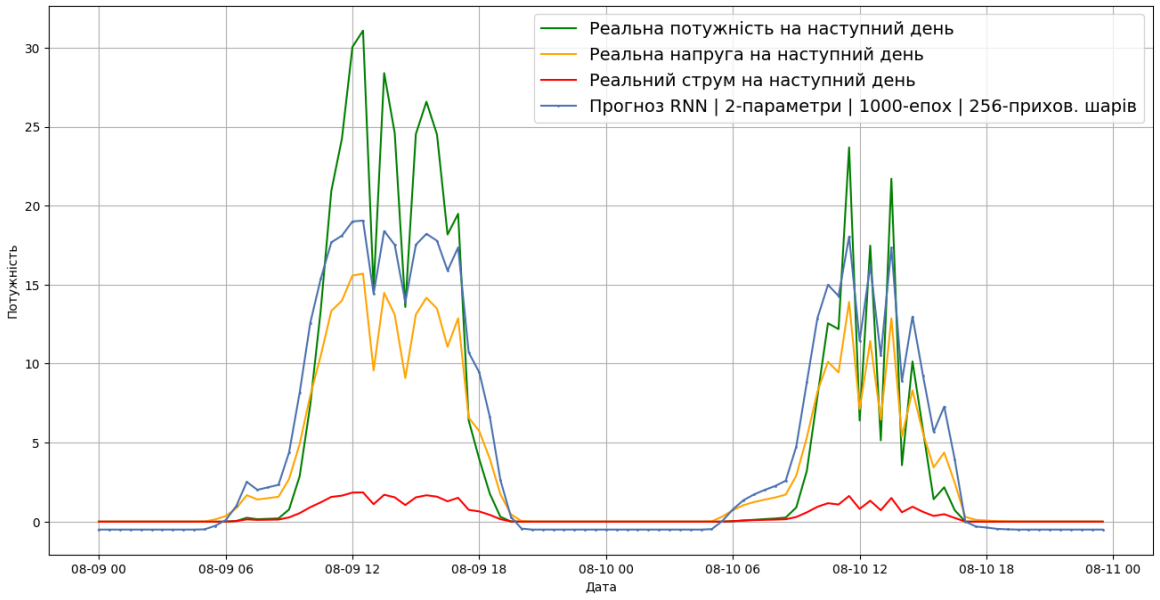


Рисунок 3.22 – Вплив параметрів RNN.

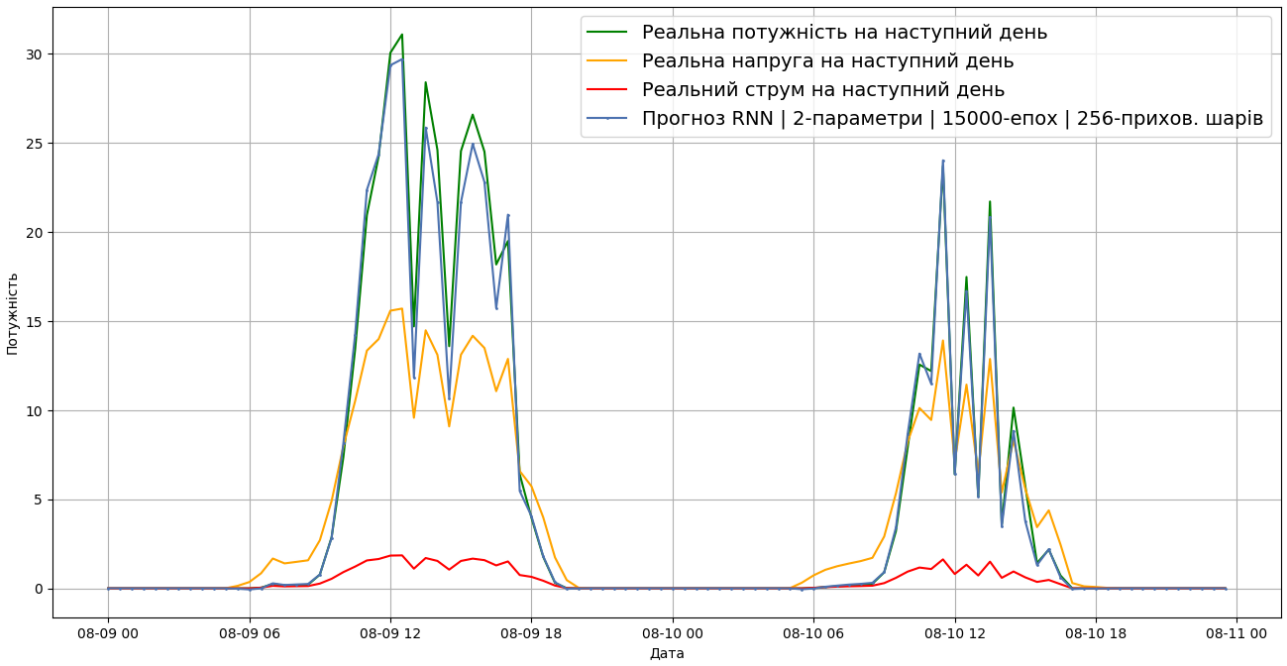


Рисунок 3.23 – Вплив параметрів RNN.

Модель ідеально прогнозує потужність в ідеальних умовах. Вона сама розуміє залежність потужності, струму та напруги. Тепер можемо побудувати нейромережу для об’єкту дослідження. Все відбувається відповідно до описаного вище. Потрібно врахувати дуже важливий момент. **Нейромережа прогнозує виключно по екзогенним параметрам.** Це означає що вона не здатна аналізувати часовий ряд, й шукати у ньому закономірності. Її робота полягає в пошуку залежності між 19 параметрами під час навчання, й на основі цих параметрів будувати майбутні значення. Тому від якості даних напряму залежить й якість прогнозування. Тепер побудуємо два вже якісні прогнози з 24 точками, що відповідає роздільній здатності вхідних параметрів, та 144 точками. На рисунку 3.24 зображеній перший прогноз на 6 днів в перед по даним прогнозу погоди, для реального об’єкту дослідження. Рисунок 3.25 відображає ті ж дані але в меншому розширенні.

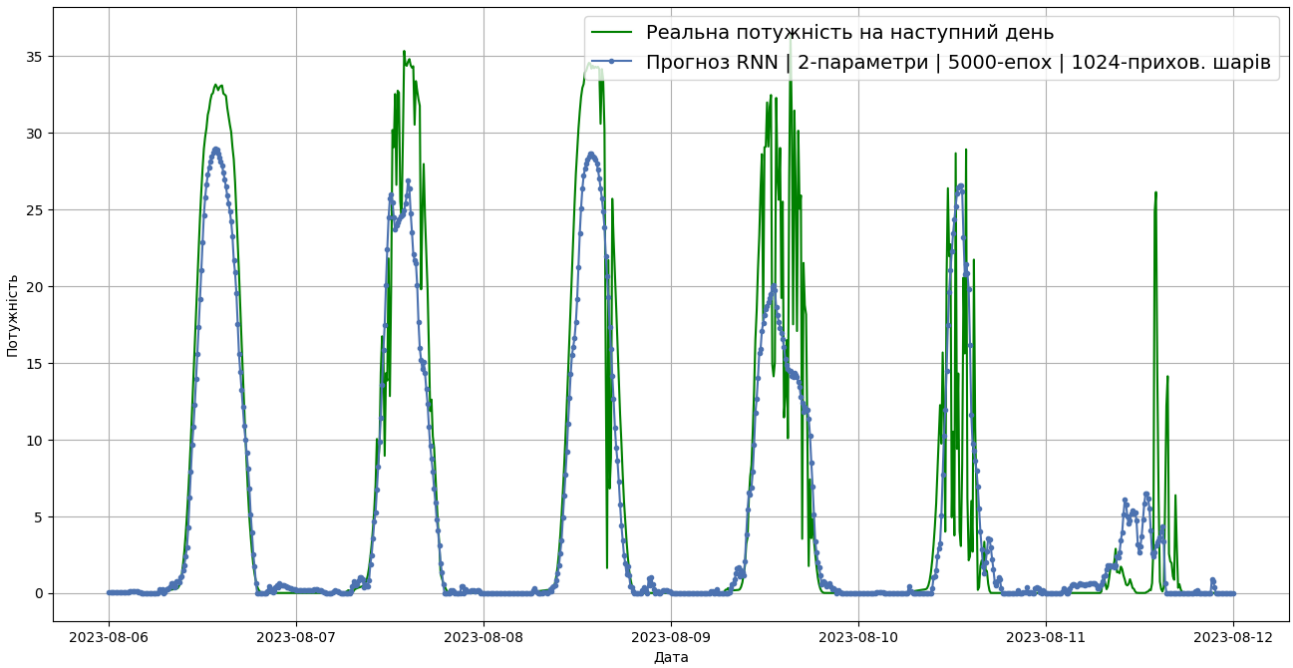


Рисунок 3.24 – Прогноз RNN.

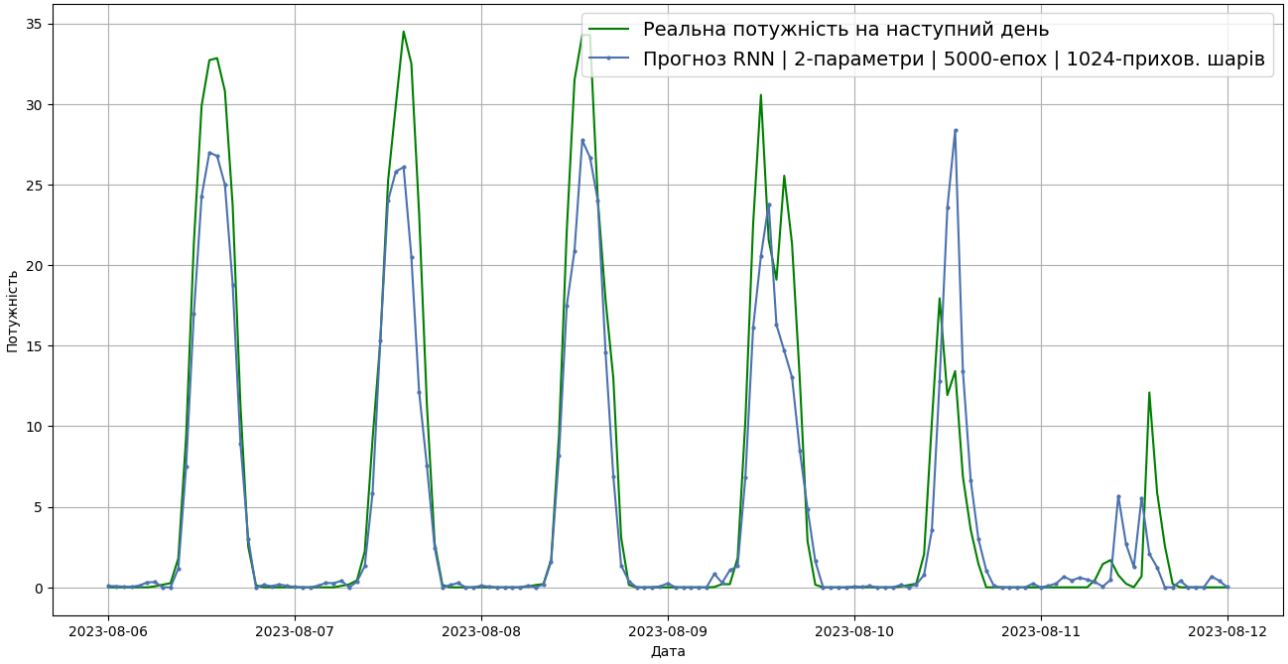


Рисунок 3.25 – Прогноз RNN.

Якщо продовжити навчання то отримаємо ефект «перенавчання», а точність почне знижуватись. Рисунок 3.26 відображає початок процесу перенавчання, нейромережа починає погано реагувати на нові дані виникає збурюваність та хибність тверджень вагів нейронної мережі.

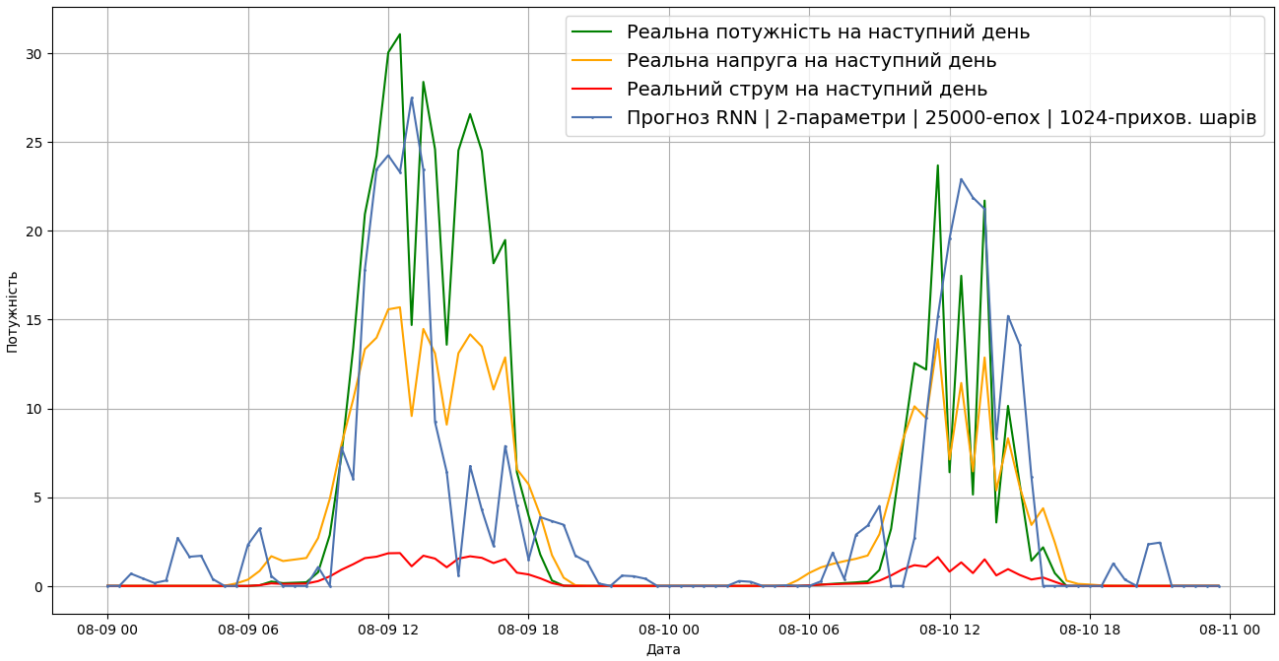


Рисунок 3.26 – Перенавчання RNN моделі.

Також в теоретичному розділі йшла мова про довгострокову пам’ять у нейромережах. Змінимо архітектуру та виконаємо такий же прогноз. Клас такої мережі виглядає наступним чином.

```
class PowerPredictionLSTM(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, dropout_prob):
        super(PowerPredictionLSTM, self).__init__()

        self.lstm = nn.LSTM(input_size, hidden_size, batch_first=True)

        self.dropout = nn.Dropout(p=dropout_prob)

        self.linear = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        out, _ = self.lstm(x)

        out = self.dropout(out)

        out = self.linear(out[:, -1, :])
        return out
```

```
PowerPredictionLSTM(
  (lstm): LSTM(18, 1024, batch_first=True)
  (dropout): Dropout(p=0.3, inplace=False)
  (linear): Linear(in_features=1024, out_features=1, bias=True)
)
```

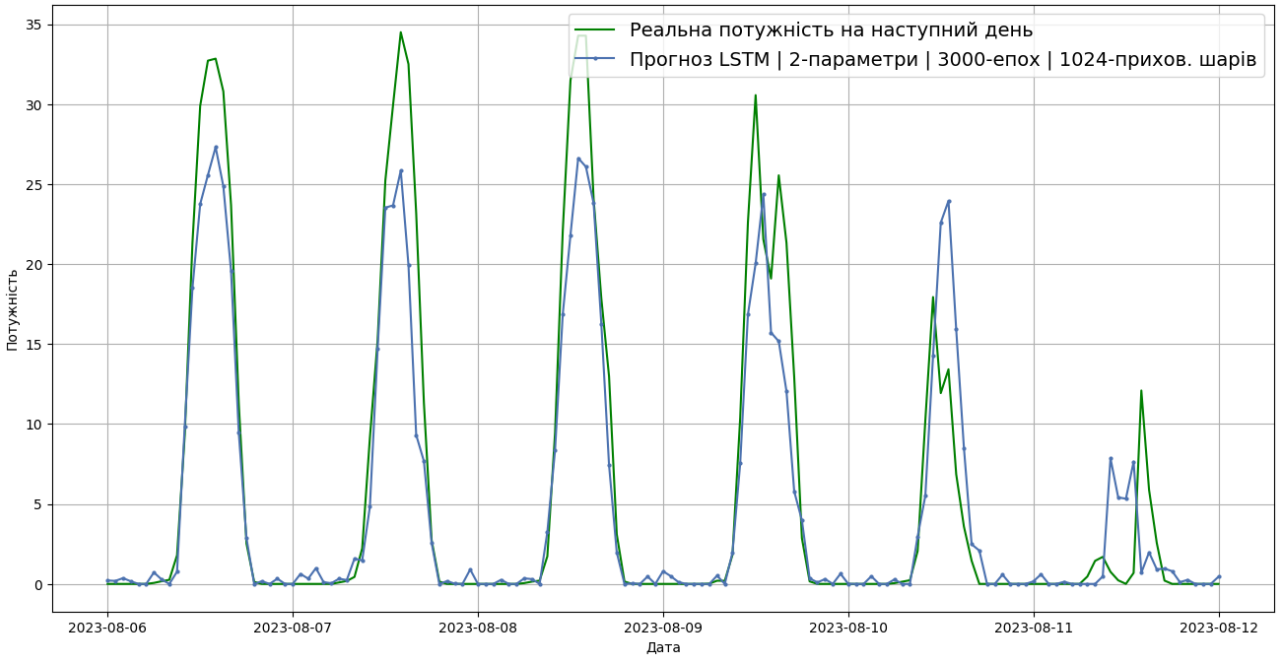


Рисунок 3.27 – Прогноз LSTM моделі.

Різниця у якості майже відсутня. Кількість епох менша, так як така непромережана швидше навчається й отже при той же кількості починається перенавчання.

Можемо також створювати комбіновані системи прогнозування. Тобто об’єднувати різні типи систем наприклад SARIMA + RNN рисунок 3.28. Це дозволяє поєднувати можливості двох моделей. У цьому випадку прогноз SARIMA подається на вхід нейромережі як один із параметрів.

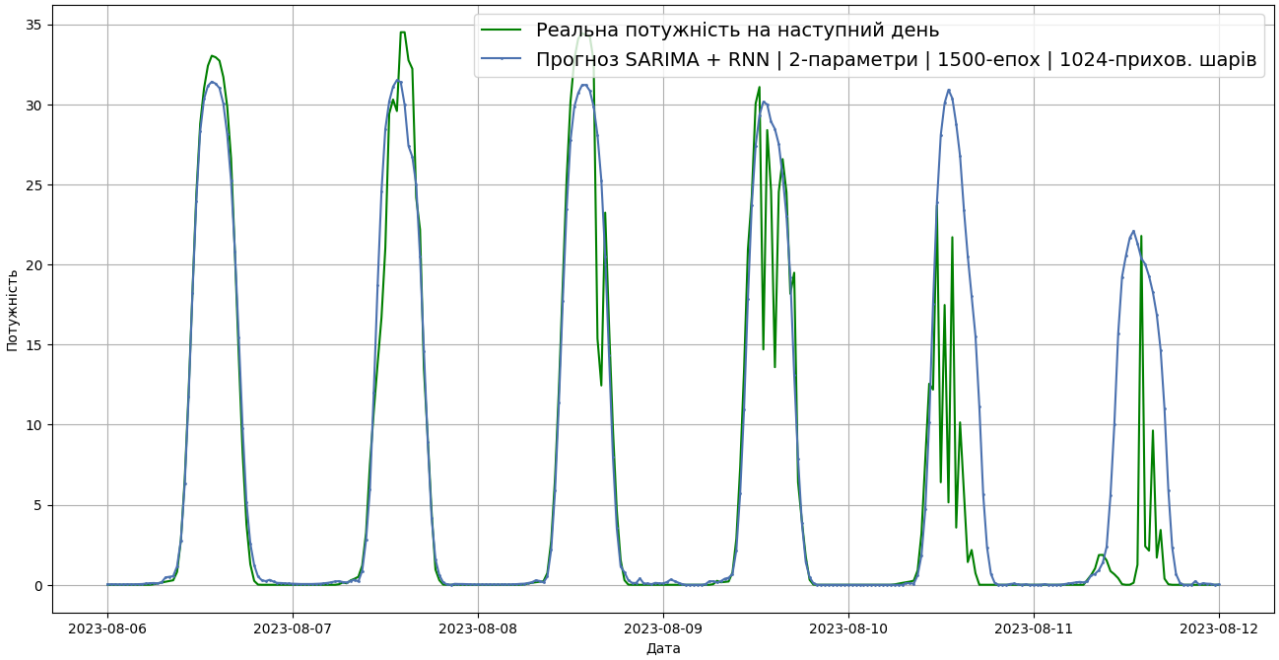


Рисунок 3.28 – Об’єднання SARIMA + RNN моделі.

Була також ідея об’єднати SARIMAX + RNN, але на практиці це працює погано, тобто в якості екзогенного параметру виступає прогноз нейромережі. Але модель SARIMAX не хоче зав’язувати себе на ці дані й тому результат прогнозу знаходиться близько до Холта-Уінтерса.

В рекурентних мережах є властивість Bidirectional. Вона дозволяє досліджувати ряд у два напрямки. Переваги BiLSTM включають здатність захоплювати більше контексту в послідовностях, що призводить до кращої здатності моделі розуміти більш складні залежності в даних. Тому додамо цю можливість до останнього прогнозу.

Й прийшов час шарів, їх існує досить велика кількість але ось опис основних:

- Шар **Embedding** використовується для перетворення категоріальних або дискретних даних (наприклад, слів, символів) у векторне подання безперервних чисел. Це корисно для роботи з текстовими даними, у яких кожне слово чи символ кодується у вигляді вектора. Цей шар допомагає мережі розуміти семантику вхідних даних.
- Шар **Dropout** допомагає боротися з перенавчанням у нейронних мережах. Він випадково відключає деякі нейрони під час навчання, що допомагає узагальнення і підвищує узагальнюючу здатність мережі.
- Шар **Batch Normalization** використовується для нормалізації активацій у часі, що допомагає прискорити збіжність мережі та зробити навчання більш стабільним.
- Шар **Attention** використовується в архітектурах мереж, пов'язаних з обробкою послідовних даних, таких як машинний переклад та обробка природної мови. Цей шар дозволяє моделі фокусуватися на конкретних частинах вхідних даних, які є важливими для конкретного завдання.
- Шар **Convolutional**: У деяких архітектурах RNN можна використовувати згорткові шари, щоб обробляти послідовні дані з урахуванням локальної структури. Це може бути корисним, наприклад, в аналізі часових рядів.

- Шар **Pooling** використовується в згорткових RNN для зменшення розмірності даних шляхом агрегації інформації з набору нейронів. Це дозволяє зменшити обчислювальне навантаження та підвищити інваріантність до невеликих зрушень даних.
- Шар **Recurrent Dropout**: Цей шар аналогічний Dropout, але він застосовується до рекурентних нейронів. Він допомагає боротися з перенавчанням усередині рекурентних верств.
- Шар **TimeDistributed**: Цей шар дозволяє застосувати інший шар (наприклад, Dense) до всіх тимчасових кроків у послідовності даних. Це може бути корисним, коли ви хочете обробляти дані на кожному тимчасовому кроці незалежно.

У нейронних мережах використовувався лінійний шар, який дозволяє виводи часовий ряд, та «дропаут» для захисту від перенавчання моделі LSTM. Але необхідно додати «Attention» для виявлення привілейованих параметрів. Виконаємо прогноз на 12 днів наперед такаю моделлю.

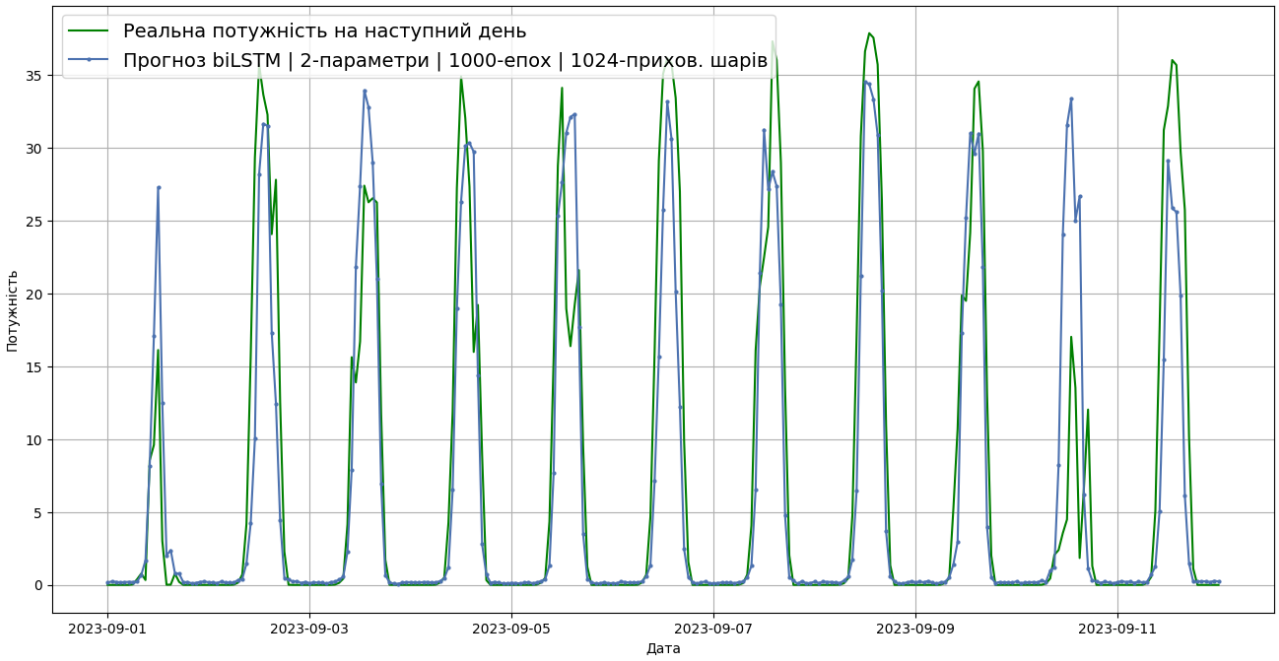


Рисунок 3.29 – Прогноз biLSTM моделі.

3.8 Висновки за розділом

У ході написання розділу було розглянуто побудову усіх моделей. Виконано дослідження часового ряду, зроблені висновки щодо ефективності

моделей. Отримані як підтвердження та й скасування певних гіпотез, які були припущені у теоретичному розділі. Наприклад, що нейромережа буде ефективнішою за SARIMAX. Рисунок 3.30 відображає точність 3-денного прогнозу на досліджуваній виборці. Але на ці дані поки що рано опиратись, необхідне повноцінне тестування яке відбудеться у 4 розділі.

Також були складнощі з використанням деяких параметрів у системах прогнозу під час дослідження у зв'язку з недостатнім апаратними потужностями.

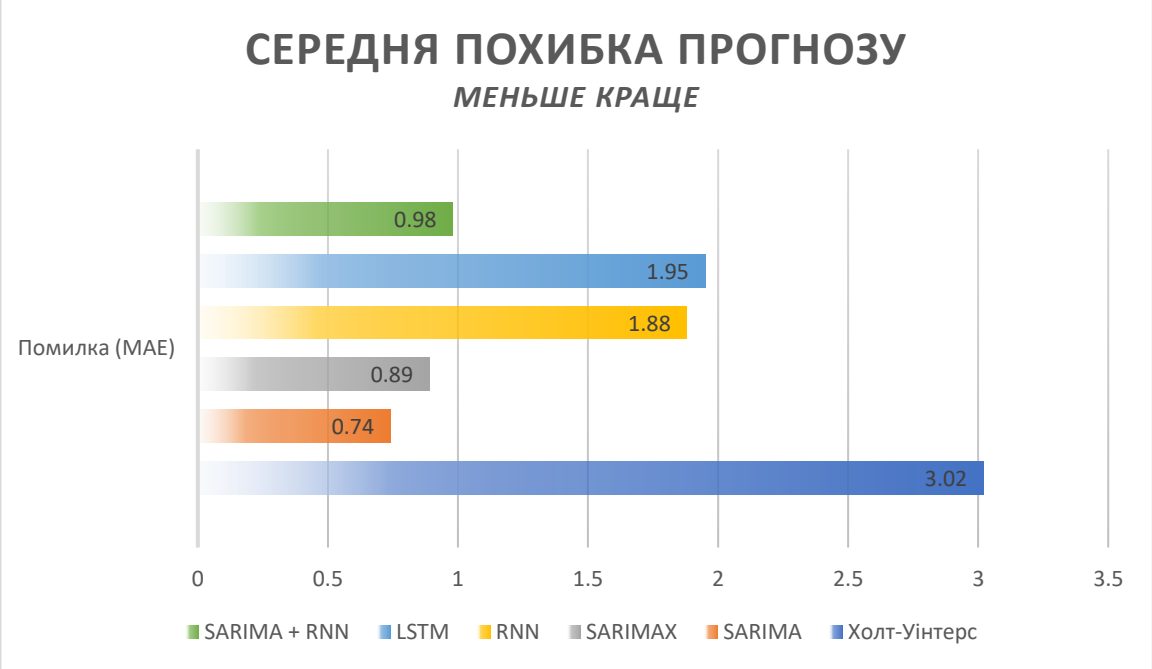


Рисунок 3.30 – Ефективність на досліджуваній виборці.

У цілому можна сказати наступне. Модель Холта-Уінтерса показала себе найгірше, з усіх можливих моделей. Хоча SARIMA має найбільшу точність, вона, що логічно, не здатна адекватно прогнозувати на довгий термін. Окрім цього має досить суттєві потреби до ресурсів. У зв'язку з недостатньою роздільною заданістю екзогенних параметрів додається додаткова похибка, під час навчання, це стосується усіх моделей з екзогенними параметрами. Що призводить до того, що модель не може корелюватися з параметром. Нейромережі дуже залежні від якості отримання погодних даних, але за умови «безкоштовності» кращого не досягти. Необхідно зосередитись на експериментах досліджених і розроблених систем.

4 ЕКСПЕРЕМЕНТАЛЬНИЙ РОЗДІЛ

4.1 Апаратні можливості та ресурси. Труднощі розрахунків.

Використання нейронних мереж та моделей ARMA потребує великої кількості обчислювальних ресурсів. Та вимагає певних апаратних вимог для виконання алгоритмів. На рисунку 4.1 вказаний час виконання прогнозу на однаковий проміжку часу з однаковими даними для навчання. Під час проводження тесту використовувалися усі доступні апаратні можливості для досягнення найбільшої продуктивності.

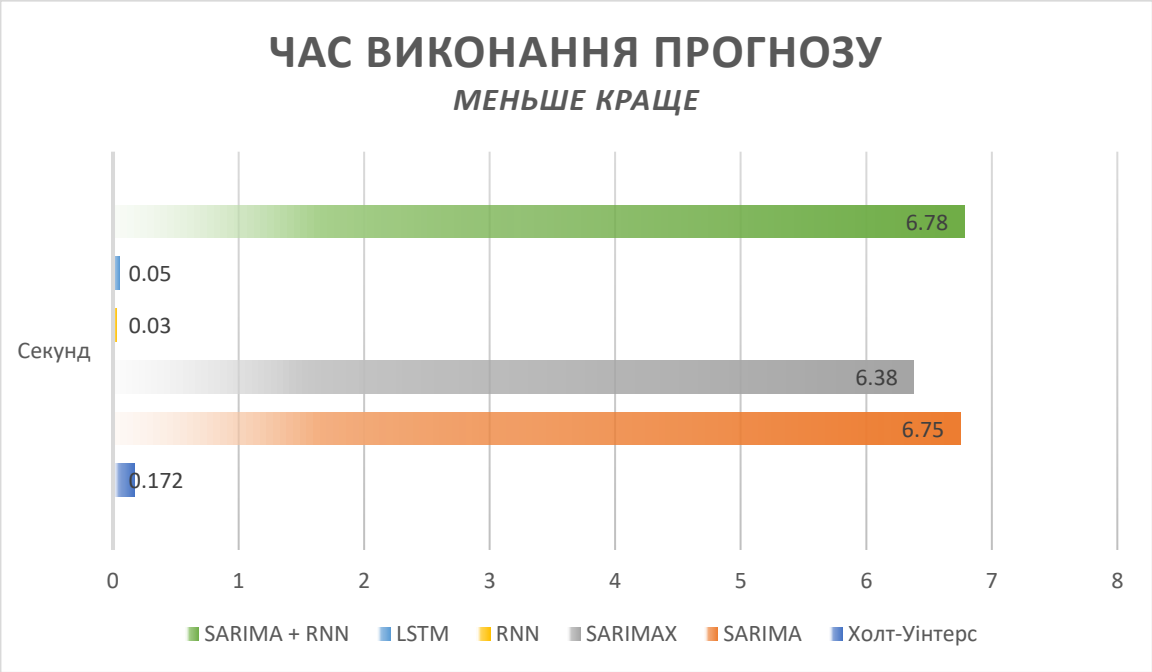


Рисунок 4.1 – Швидкість виконання прогнозу.

Нейромережі мають найбільшу швидкість прогнозу, обходячи модель Холта-Уінтерса. Це все завдяки використанню сучасних технологій обчислення нейронних мереж. На малюнку 4.2 відображена швидкість навчання. Нейромережам не потрібно навчатися при новому наборі даних, тому вони були виключені з тесту. Потрібно врахувати, що загальний час прогнозу для моделей Холта-Уінтерса та SARIMA, SARIMAX - це загальний час навчання плюс час виконання прогнозу. Модель SARIMAX має найдовший термін навчання з усіх моделей. На малюнку 4.3 вказана швидкість однократного навчання нейромережі.

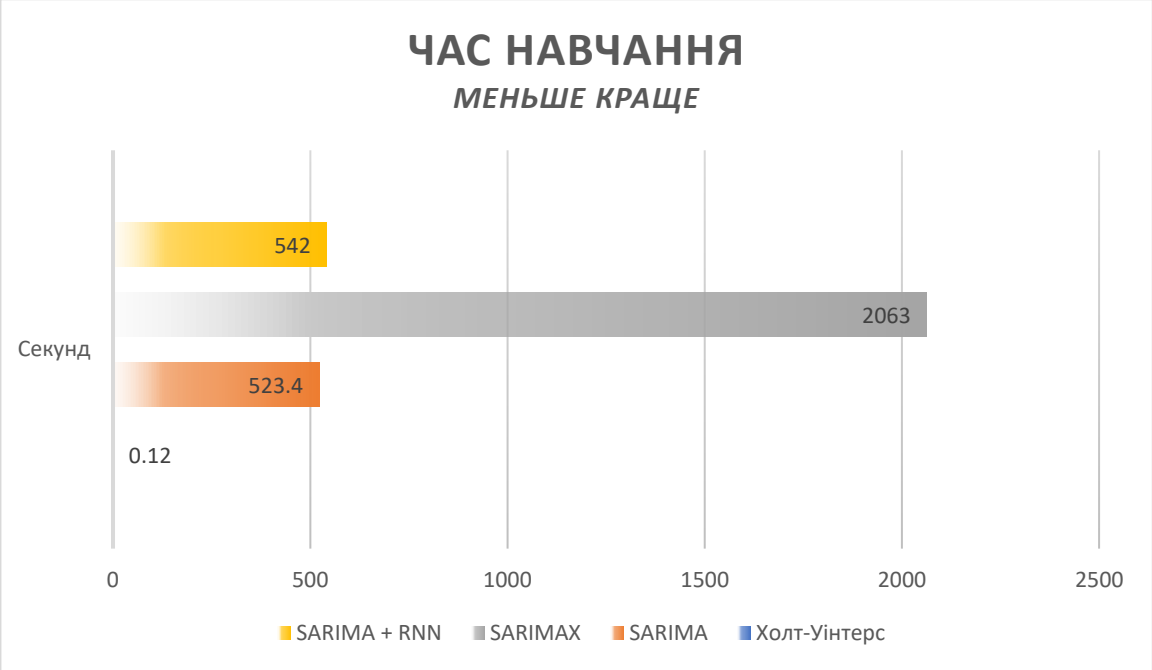


Рисунок 4.2 – Швидкість виконання навчання.

Тест періоду навчання виконувався повністю в однакових умовах: дані, кількість епох, слої та інші параметри.

Рисунок 4.3 – Швидкість виконання навчання.

4.1.1 Опис існуючого апаратного забезпечення об’єкту дослідження.

Об’єкт дослідження побудований на Intel Core 2 Duo E7500, та має 2 гігабайти оперативної пам’яті. Також в системі встановлений графічний прискорювач Nvidia GeForce 7500. Це викликає труднощі, адже графічний прискорювач не може використовуватися для нейромережі за відсутності необхідних технологій, та архітектурних особливостей. Також в системі малий об’єм оперативної пам’яті, що не дає можливості використовувати ARMA моделі. Лише центральний процесор здатний до виконання подібних завдань, завдяки підтримці інструкцій SSE4.

Тому для виконання дослідження та експериментів використовувалася інша система. Саме на цій системі виконувалися усі тести у дипломному проекті. Система має 12 гігабайт оперативної пам’яті, центральний процесор Intel Core I7 3770, та графічний прискорювач Nvidia RTX 2060 Super. Це досить старе залізо (2012 рік) але його достатньо для виконання усіх прогнозів. Для вирішення проблем реального об’єкту можна використовувати спеціалізовані однопалатні системи або виконати модернізацію ПК.

В якості однопалатної системи можна використовувати Nvidia Jetson.

Рисунок 4.4 – Міні комп’ютер Nvidia Jetson Nano Developer Kit V3 .

Jetson Nano, володіє вражаючою продуктивністю для запуску AI додатків. Її відносно компактні розміри, низьке споживання енергії та відносно доступна вартість роблять її одним із найкращих рішень у цьому сегменті. Підтримка NVIDIA JetPack реалізована в цій платформі та включає пакети для підтримки BSP, ОС Linux, бібліотек для NVIDIA CUDA, cuDNN, TensorRT, які використовуються у глибокому навчанні: комп'ютерному баченні, обчисленнях за допомогою графічних ядер, обробці мультимедійних даних і т. д. JetPack SDK використовується у всій лінійці платформ NVIDIA Jetson і повністю сумісний з AI NVIDIA; він призначений для навчання та впровадження програмного забезпечення AI. Це значно знижує поріг для вступу у розробку AI-пристроїв для розробників. В якості центрального процесору виступає Quad-core ARM Cortex-A57, 4 гігабайти оперативної пам’яті та графічний прискорювач на архітектурі Maxwell з 128 ядрами. Така система потужніша за систему на Intel Core 2 Duo E7500.

4.1.2 Технології Nvidia CUDA, Тензорні ядра

Для прискорення нейронних мереж були розробленні спеціальні апаратні системи. В диплому проекті використовується дві такі технології: CUDA та TensorRT.

Рисунок 4.5 – Зображення графічного прискорювача TU106(RTX 2060S).

Технологія CUDA - це архітектурна особливість графічних прискорювачів Nvidia для паралельних розрахунків. Яка дозволяє використовувати GPU у виконанні математичних операцій з великою швидкістю. У чому різниця між

виконанням операції на CPU та GPU? На рисунку 4.6 зображена схема процесору i7 3770 який має 4 фізичних ядра з підтримкою Hyper-Threading. Має 8 мб кешу третього рівня, та підтримує 128 бітну шину даних, що забезпечує 22 Гб/с швидкість обміну з оперативною пам’яттю DDR3.

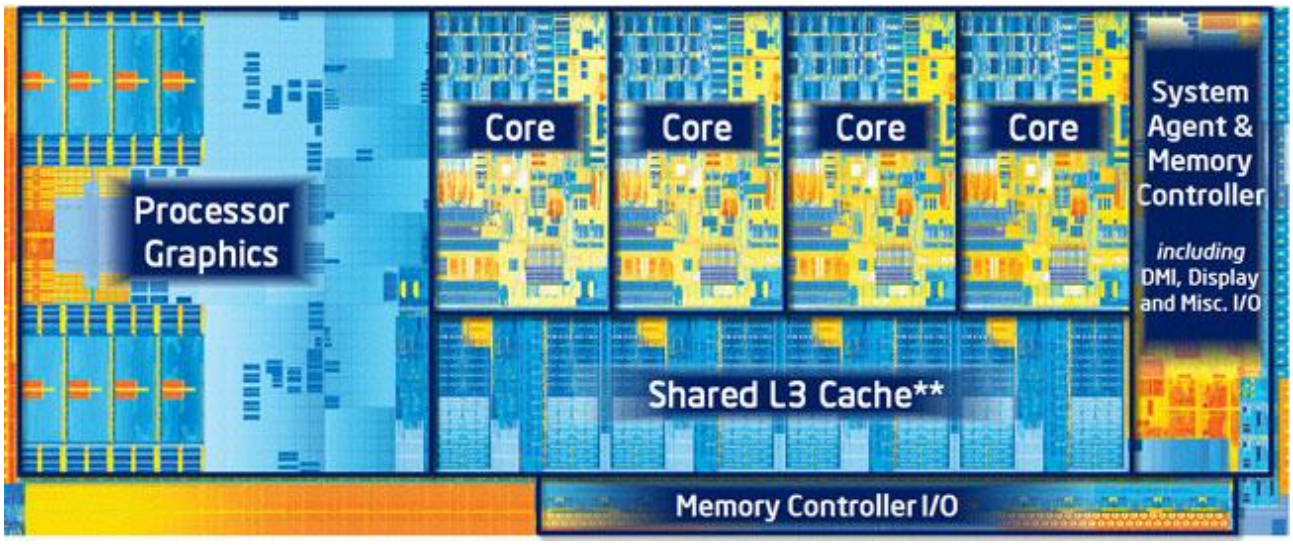


Рисунок 4.6 – Структура мікро архітектури Ivy Bridge.

В той час відеокарти мають набагато більшу розрахункову потужність, у нашому випадку GPU має 2176 ядер CUDA які працюють на частоті 1650 МГц. Кожен SM блок ядра малюнок 4.7., має блоки для цілочисельних операцій та окремий блок для операцій з дробними числами двійної точності. Окрім того, що маємо велику кількість ядер, а саме у 544 рази порівнюючи з CPU, так ще ці операції виконуються асинхронно, тобто кожен блок одночасно може виконувати операції з двома типами числа. Кожен блок має свій кеш обсягом 96 кілобайт, а також вхідний буфер для утримання команд для кожної групи блоку SM. Кожне ядро має доступ до кешів різних рівнів, але при використанні нейромереж велику роль відіграє саме об’єм пам’яті та її швидкість. Сучасні процесори не можуть забезпечити туж швидкість, що й графічні прискорювачи. В нашому випадку маємо 256 бітну шину даних, яка дає можливість виконувати обмін з відео буфером об’ємом 8 гігабайт зі швидкістю 448 Гб/с.

Технологія CUDA дозволяє використовувати ці ресурси саме у розрахунках, а не тільки для роботи з векторами та побудовою зображення. Проведемо дослідження ефективності такого методу розробки, виконаємо

навчання LSTM мережі з використанням лише центрального процесору. Тобто без використання Nvidia CUDA Toolkit 12.3. Потрібно зазначити, що ми виконаємо лише 10 епох адже 4000 першопочаткових епох будуть виконуватися на CPU декілька днів.

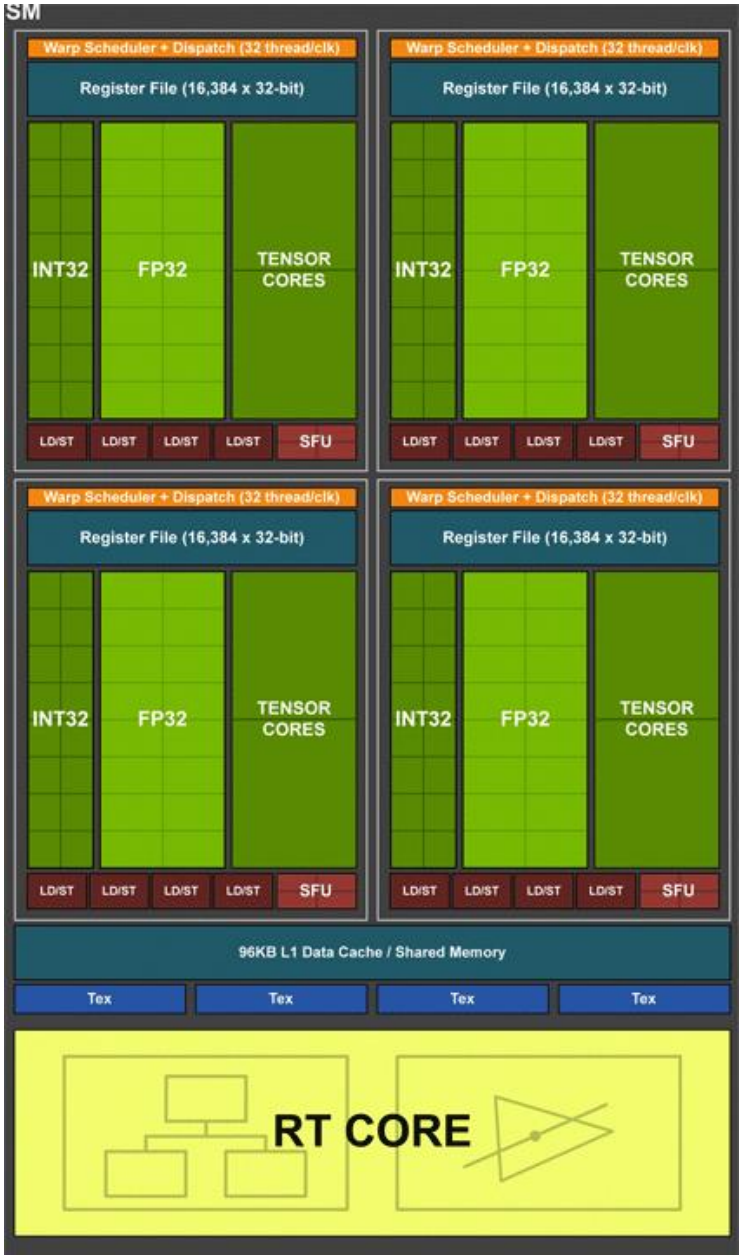


Рисунок 4.7 – Структура блока SM архітектури Turing.

На рисунку 4.8 бачимо домінацію відеокарти у 296 разів. Саме у цьому перевага цієї технології, тепер представимо ситуацію коли у нас відсутня GPU, а для якісного прогнозу нам необхідно 4000 епох навчання. На процесорі це б зайняло 41 годину, в той час відеокарта виконує за 227 секунди. Це не дозволило би виконати дослідження та експерименти. А система на базі Intel E7500 навчить

нейромережу не менше ніж за 123 години, це в найкращому випадку, так як цей процесор не має AVX інструкцій.

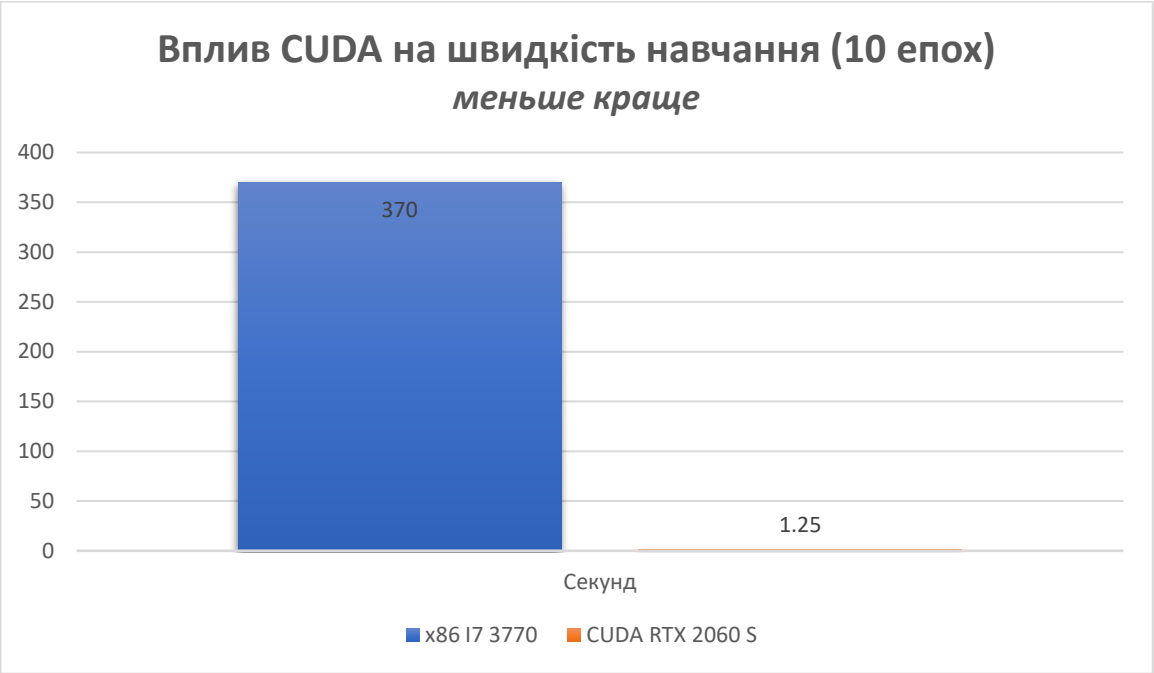


Рисунок 4.8 – Швидкість CPU проти GPU.

Переваги CUDA:

1. Велика продуктивність**:** CUDA дозволяє використовувати потужності GPU для обчислень, що призводить до значного збільшення продуктивності, особливо в обчисленнях з великою кількістю даних.
2. Масова паралелізація**:** CUDA дозволяє паралельно обробляти багато завдань на GPU, що особливо корисно для задач, де обчислення можна поділити на незалежні фрагменти.
3. Широке застосування**:** CUDA використовується в багатьох галузях, включаючи глибоке навчання, обробку зображень, обчислення наукових даних і інші сфери.
4. Зручна розробка**:** Існує велика спільнота розробників, підтримка з боку NVIDIA та доступність безлічі ресурсів для вивчення CUDA.

Недоліки CUDA:

1. Складність вивчення**:** CUDA може виявитися складним для вивчення, особливо для початківців, які раніше не мали досвіду з паралельним програмуванням.

2. Портабельність коду: CUDA-код не завжди є портативним між різними версіями GPU від NVIDIA, що може призвести до проблем при оновленні обладнання.
3. Залежність від апаратної підтримки: обчислення залежать від наявності підтримки CUDA на вашому GPU. Якщо ви не маєте сумісного обладнання, ви не зможете використовувати CUDA.

У цілому технологія CUDA була впроваджена починаючи з архітектури Tesla, чипу G80 у 2007 році. Це відеокарти GeForce 8800 Ultra, GeForce 8800 GTX й новіші. Тому вона досить широко розповсюджена починаючи від мобільних та офісних рішень, закінчуючи ігровими чи професійними.

Але цього почало бракувати, тому починаючи з архітектури Volta для серверів та Turing для ПК було впроваджено нову технологію для нейромереж, тензорні ядра. Причиною заміни CUDA постає у зростаючих потребах складних нейронних мереж. На малюнку 4.7 було також зображено два тензорні ядра у SM блоці, основна особливість цих ядер полягає у виконані операцій з усім тензором. Наприклад необхідно виконати множення матриці 8x8 на таку ж матрицю.

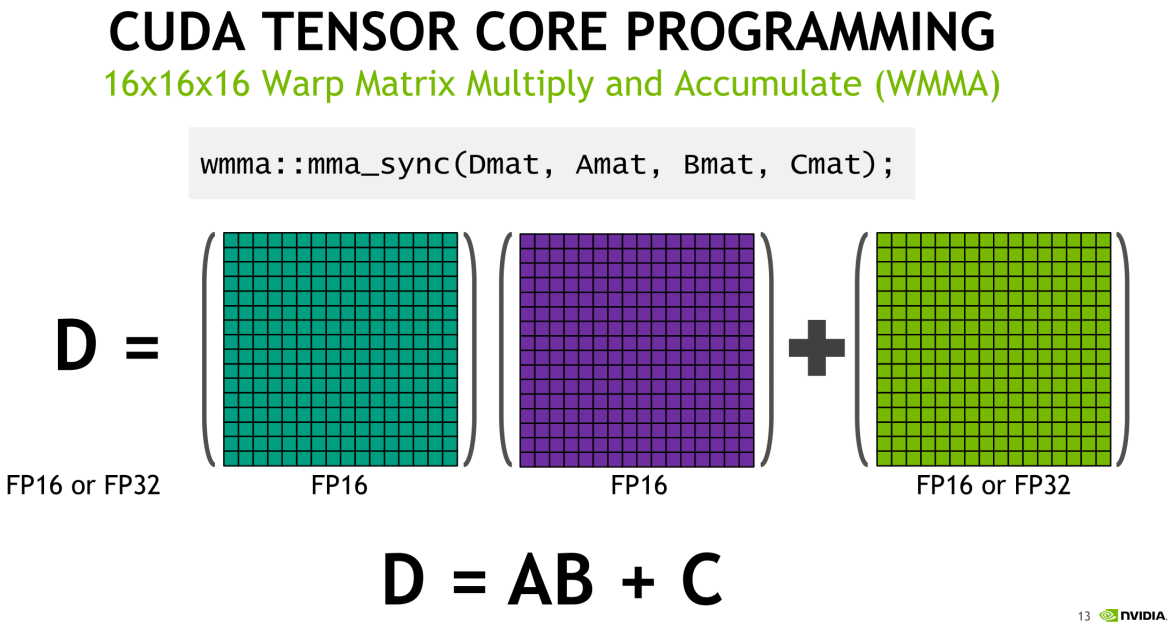


Рисунок 4.9 – Візуальне зображення обчислення тензору.

У випадку з центральним процесором кожен елемент матриці буде окремо перемножуватись на елемент іншої матриці, у випадку CUDA кожен елемент матриці буде перемножуватись у відповідному блоці асинхронно. Але у випадку використання тензорних ядер, матриця не буде розкладатися на елементи, а множення відбувається одразу усіх елементів відповідно до кількості тактів операції. Тобто перемноження двох чисел та перемноження масиву 16x16 буде виконуватись майже за однаковий процесорний час.

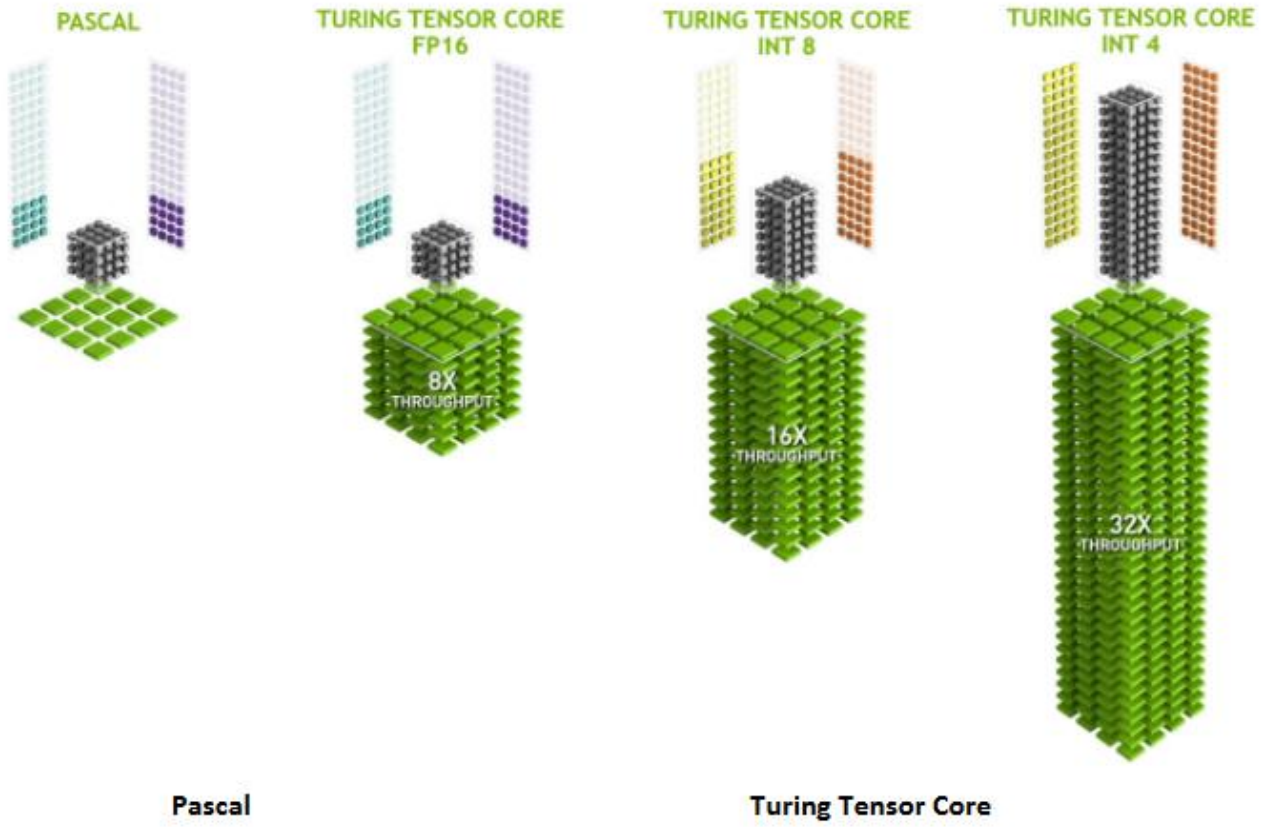


Рисунок 4.10 – Різниця між CUDA та Tensor Core.

Так, є певне обмеження щодо максимально розміру такого масиву, й його розмір залежить від типу даних у блоці. Використання тензорних ядер збільшує швидкість навчання для прогнозу у 3 рази. Тобто загальна домінація над CPU становить 888 разів. Нажаль використовувати Тензорні ядра можна лише на відеокартах серії A, V та RTX. Це архітектури Volta, Turing, Amper, Ada Lovelase. Починаючи з TU106 (2019 рік), RTX 2060 або новіші, для не комерційного ринку. Дана технологія в даний час має досить велику собівартість тому в коді вона не

використовується, лише для експерименту та перевірки її ефективності. Результат швидкості технології на малюнку 4.11.

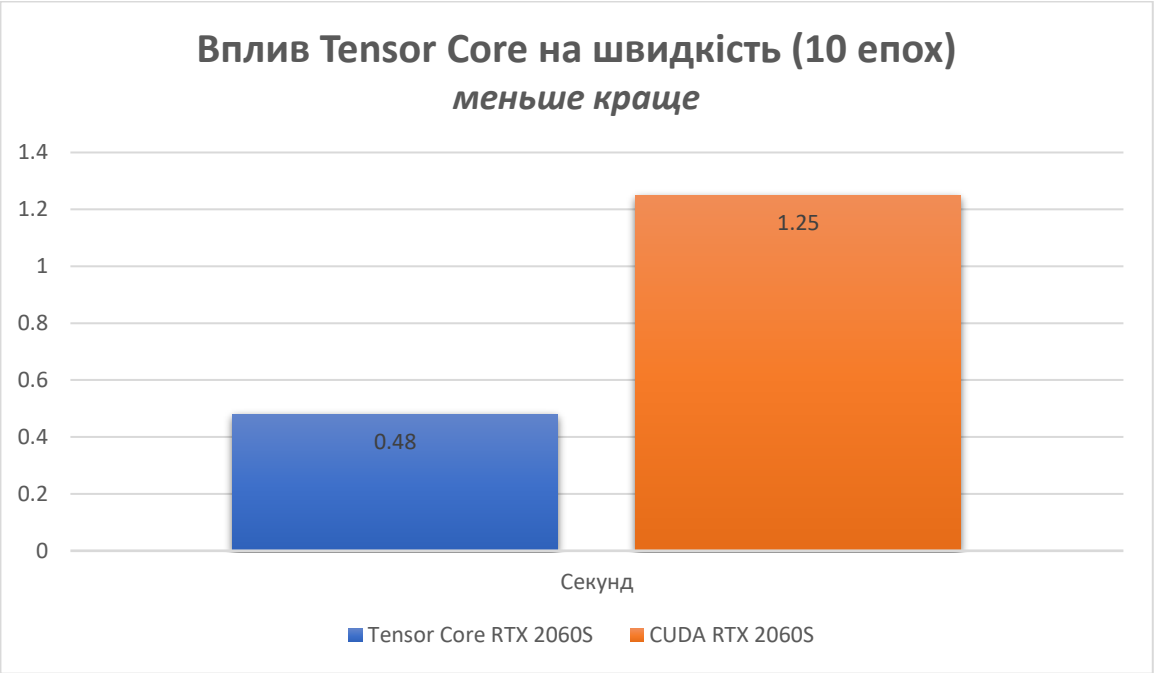


Рисунок 4.11 – Ефективність тензорних ядер у об’єкті дослідження.

4.1.3 Використання набору команд CPU: AVX, AVX2, AVX512.

Центральні процесори теж намагаються не відставати від графічних прискорювачів. Але навіщо розглядати системи на базі CPU, які суттєво програють GPU? Проблема у тому, що ARMA подібні моделі не підтримують ні CUDA для відеокарт Nvidia, ні OpenCL для AMD Radion. Тому використовують лише центральний процесор. У 2008 році компанія Intel представила набор додаткових команд «Advanced Vector Extensions», для поліпшення продуктивності обчислень на процесорах. До процесора додаються додаткові команди, які дозволяють виконувати операції над векторами, що корисно для нейромереж та обчислення зображень тощо. Підтримка цих інструкцій розпочинається у 2011 році, починаючи з процесорів Intel та AMD, це архітектури Sandy Bridge та Bulldozer відповідно. Нажаль вимкнути AVX під час виконання навчання не є можливим, так як це низький рівень взаємодії з процесором, і операційна система не надає можливості керувати AVX. З цього

виходить, що тести які були у попередніх розділах у разі використання CPU автоматично використовували AVX.

Основні нововведення AVX включають:

- **Векторні регістри зміненої довжини**: AVX вводить нові векторні регістри YMM, які мають подвійну довжину порівняно зі старими XMM регістрами. Це дозволяє обробляти більше даних за одну операцію, підвищуючи продуктивність векторних операцій.
- **Покращені арифметичні операції**: AVX додає нові арифметичні операції для векторів, включаючи додавання, віднімання, множення і ділення. Ці операції дозволяють виконувати більше операцій над векторами даних одночасно, що підвищує продуктивність.
- **Покращені операції з плаваючою комою**: AVX додає нові операції з плаваючою комою, включаючи додавання, віднімання, множення і ділення з підтримкою подвійної точності. Це особливо важливо для наукових обчислень і обчислень у галузі ігор.
- **Покращена підтримка для інструкцій завантаження та збереження**: AVX додає нові інструкції для завантаження та збереження даних у векторні регістри, що полегшує обробку великих обсягів даних.
- **Підтримка для фузійних операцій**: AVX підтримує оптимізацію фузії операцій, що дозволяє зменшити кількість обчислень, необхідних для виконання певних операцій.

Для тестування впливу напишемо окрему програму на мові C, а результат швидкості розрахунку зображено на графіку 4.12. За 100% приймемо виконання на сі, без використання розширених команд архітектури x86. З цього виходить наступний висновок, що система на базі E7500 буде ще повільніше, ніж було сказано раніше. Було зазначено, що неменше 123 годин для нейромережі, але враховуючи відсутність AVX у цьому процесорі можемо допустити, що швидкість виконання може бути більшою, ще на 57%. Нажаль виконати експеримент впливу більш сучасних AVX2 та AVX512 не є можливим, адже моє обладнання не підтримує ці інструкції.

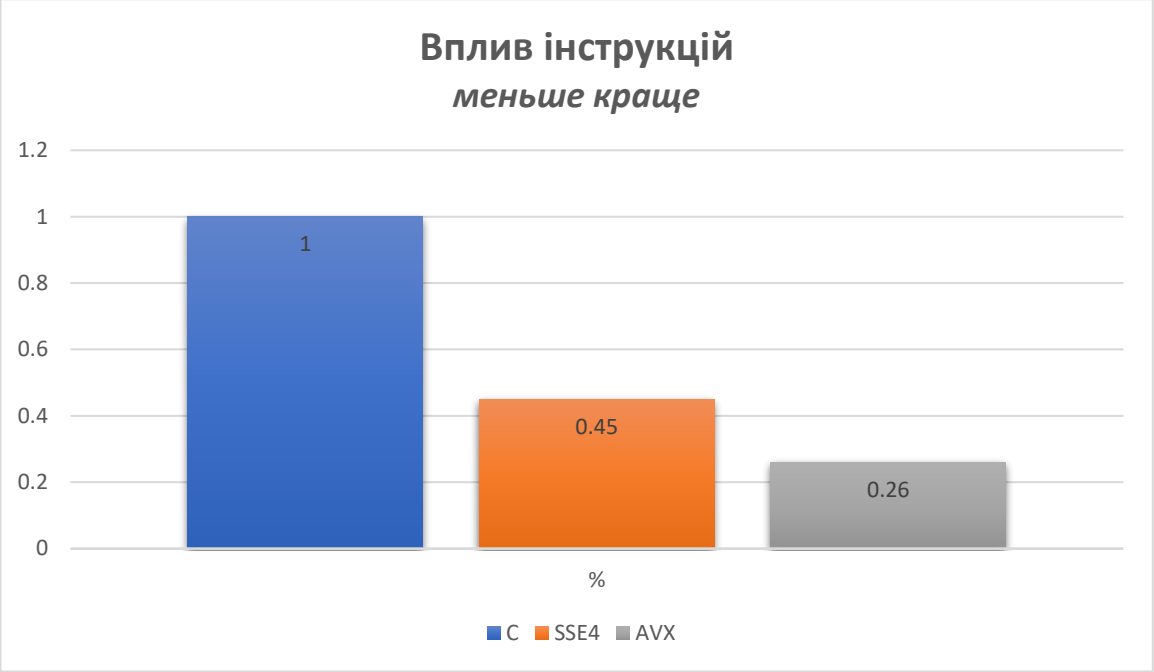


Рисунок 4.12 – Вплив розширених інструкцій на швидкість розрахунків.

Можемо звернутися до технічної документації, що би орієнтовно оцінити ефективність цих методів. На малюнку 4.13 зображено офіційне порівняння швидкості виконання розрахунків. Скоріш за все вони були виконані в ідеальних умовах, але корелює з нашим тестом, а саме SSE4 та AVX майже сходяться 56,79% різниці у ефективності. Тому можна допустити що більш сучасні методи дадуть схожий результат.

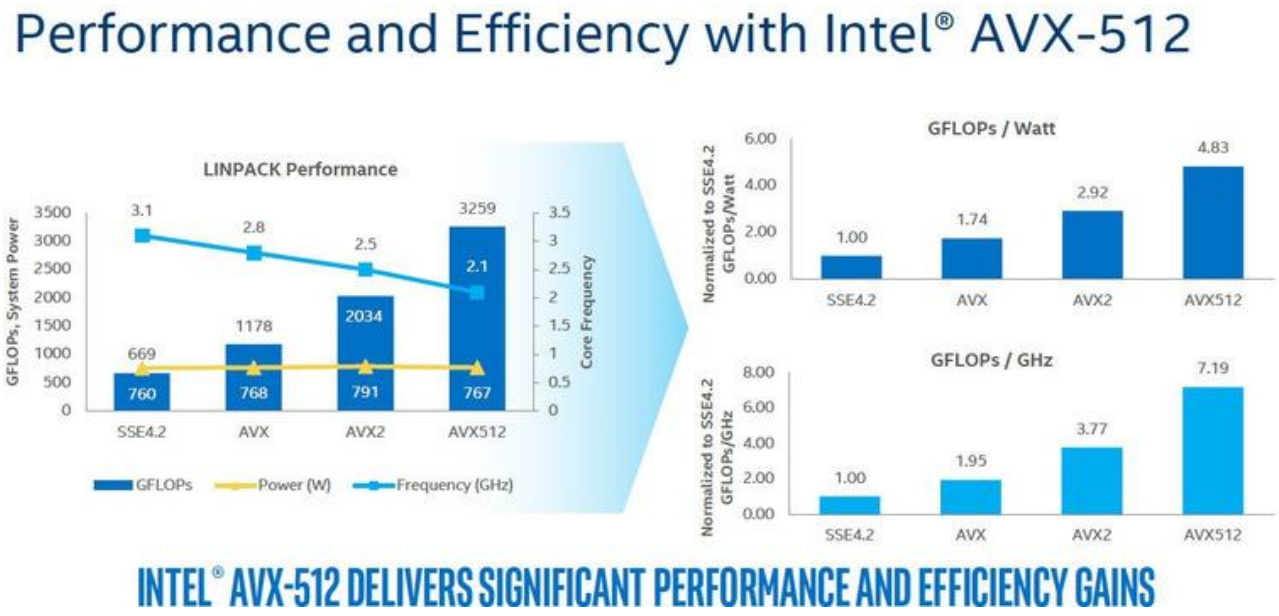


Рисунок 4.13 – Вплив розширених інструкцій на швидкість розрахунків.

Коротко про основні нововведення AVX2 порівнюючи з AVX. Це підтримка FMA операцій, тобто виконання множення та додавання одночасно. Підтримка команд знаходження максимумів та мінімумів, збільшення регістру з 128 біт до 256. Оптимізація циклів. У AVX 512, ще сильніше збільшили регістр до 512 біт, та збільшена кількість регістрів процесора для зберігання проміжних результатів. Підтримка змішаних режимів обчислень. Додавання розширених інструкцій зсуву й розгортання. Та покращена робота з числами двійною точністю.

4.1.4 Ресурсоємність розроблених методів прогнозування

Однією з цілей дипломного проекту є визначення оптимального методу прогнозування. У випадку коли є обмеження обладнання, ми не можемо опиратися лише на точність, а потрібно врахувати й ресурсоємність методів. Усього є 4 параметри: необхідна кількість оперативної пам’яті, необхідна кількість відео пам’яті (якщо є), завантаження GPU (якщо є), завантаження CPU.

Першим буде тестування ОП, маємо однакові умови проводження експерименту, після виконання кожного прогнозу примусово очищується пам’ять. Використання ОП операційною системою та фоновими програмами не враховується. Отримані наступні результати.

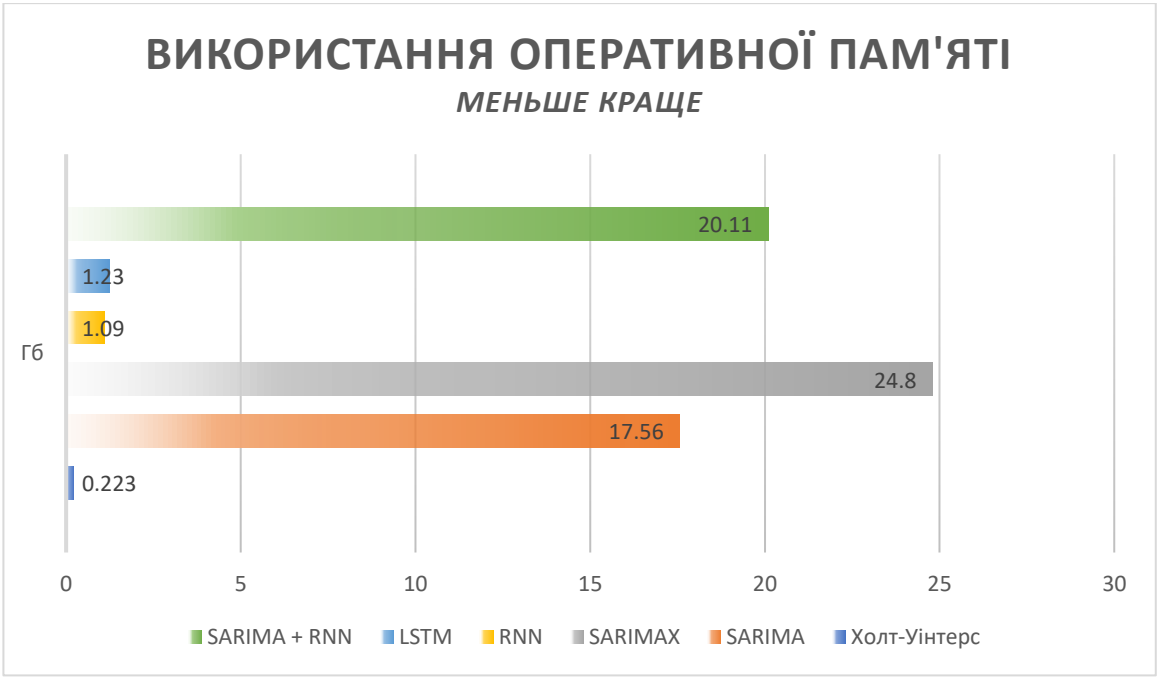


Рисунок 4.14 – Необхідний обсяг ОП для виконання прогнозу.

Системи ARMA використовують дуже великий об’єм пам’яті, у нашому випадку використовується «файл підкачки», тому що, оперативної пам’яті у 12 гігабайт не достатньо. Використання файлу підкачки зменшує швидкість прогнозу. У разі не доступності пам’яті даних, прогноз не відбудеться, а завершиться помилкою. Нейромережі займають малий об’єм ОП, так як використовують відео буфер для зберігання основної інформації. Лідером же виступає модель Холта-Уінтерса, її можна зробити ще більш ефективною. У разі використання меншої роздільною здатності та оптимізації використаних функцій.

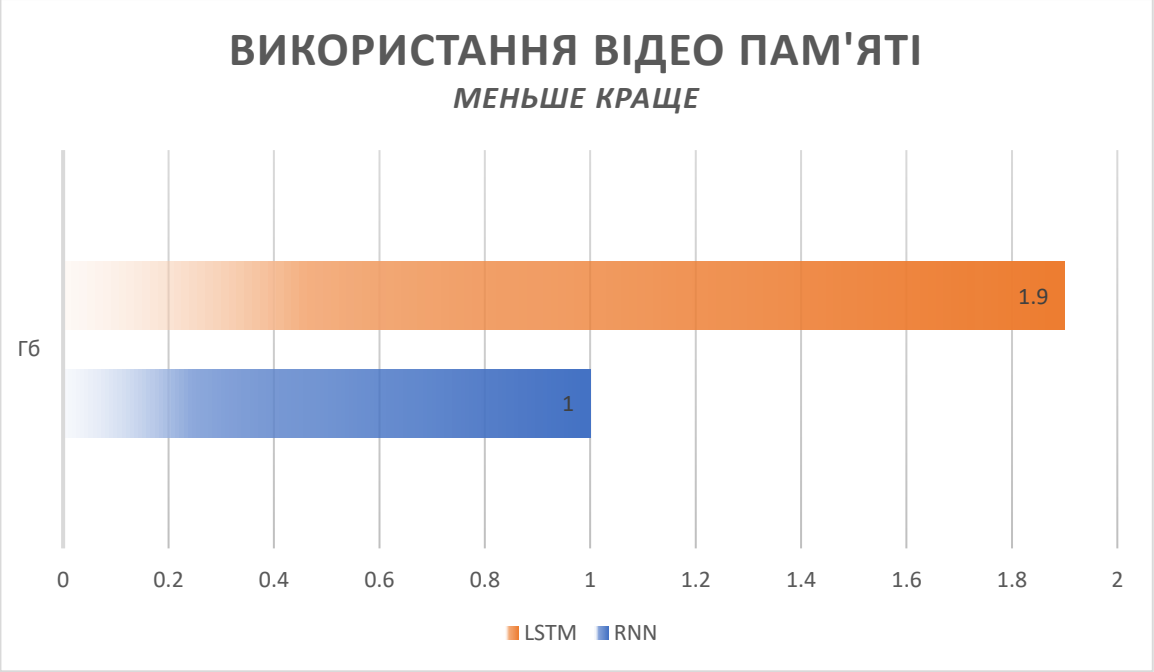


Рисунок 4.15 – Необхідний обсяг відео пам’яті для виконання прогнозу.

Використання центрально процесору зображено на рисунку 4.16. Тести проводились в «безпечному режимі». Графік має 4 групи: час виконання прогнозу, використання центрального процесору у відсотках (7-10% використовує операційна система), кількість задіяних потоків процесору, та частота. Потрібно зауважити, що процесор використовує технологію Turbo Boost, тому частота на кожному з ядрі наступна: 1 Core – 3.9 ГГц, 2 Core – 3.9 ГГц, 3 Core – 3.8 ГГц, 4 Core – 3.7 ГГц. На графіку 4.17 зображено використання GPU нейромережами. Контролер пам’яті у відсотках, вказує на його навантаження. Тобто відображує інтенсивність передачі даних у шині.

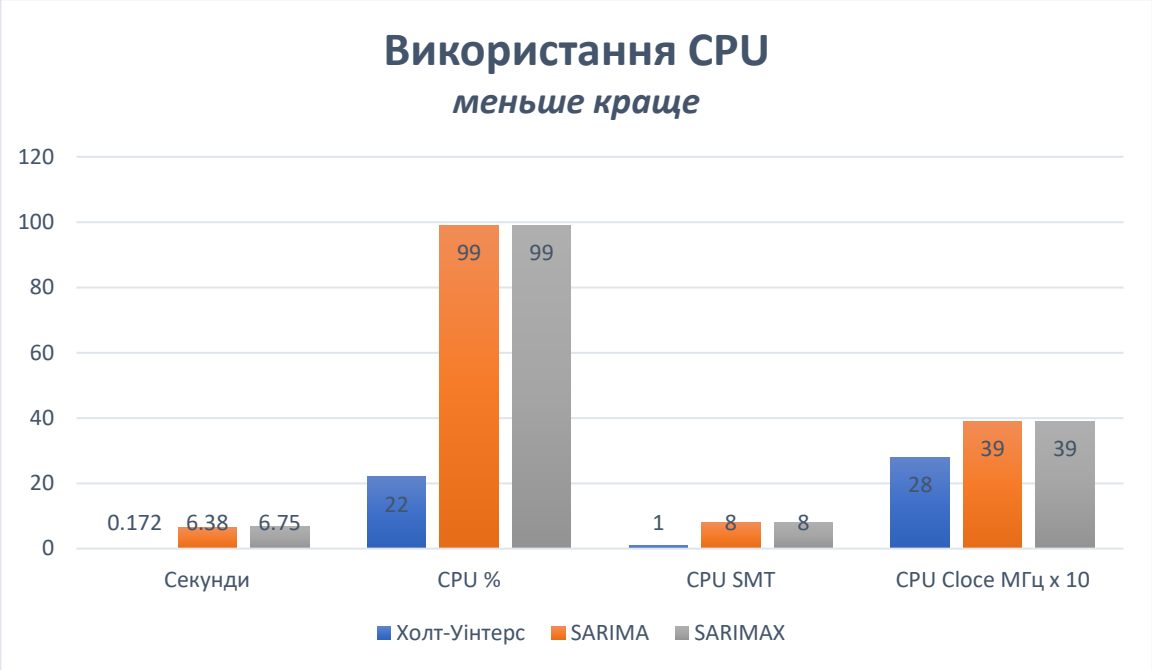


Рисунок 4.16 – Використання процесору відповідно до часу виконання.

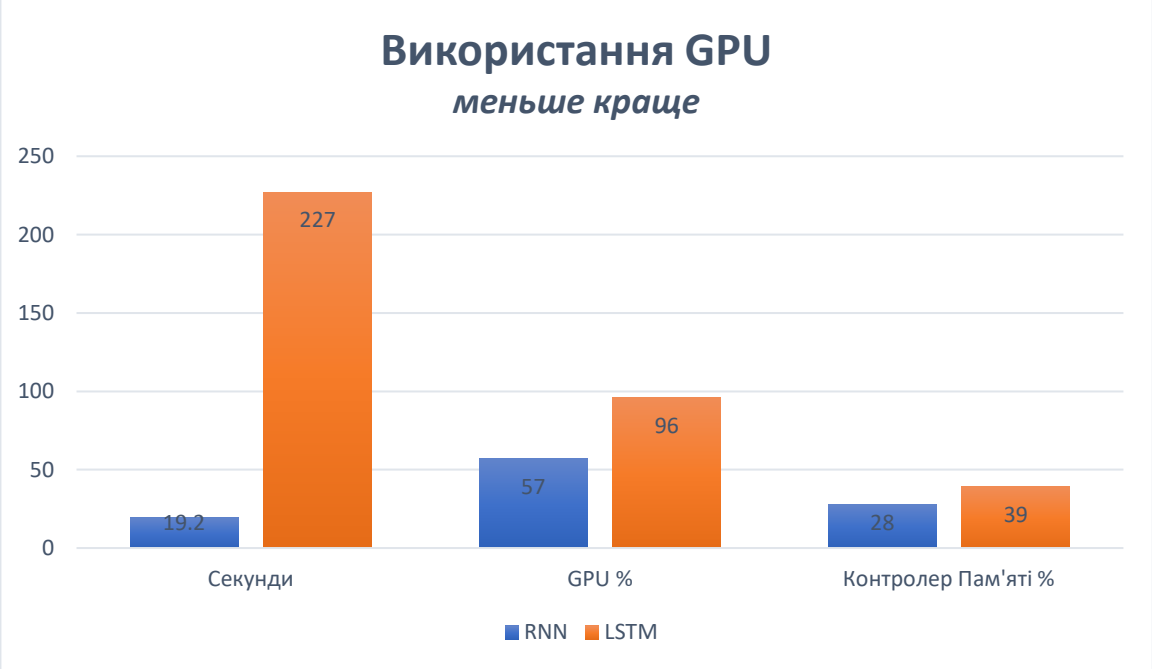


Рисунок 4.17 – Використання GPU відповідно до часу виконання.

4.2 Експеримент з виявлення точності розроблених моделей прогнозування

Для оцінення точності необхідно провести експерименти з прогнозування на різні періоди, та з різними не сподіваними ситуаціями. Спочатку виконаємо експеримент з прогнозом на один день. У часовому ряді після сонячних днів, настає хмарний.

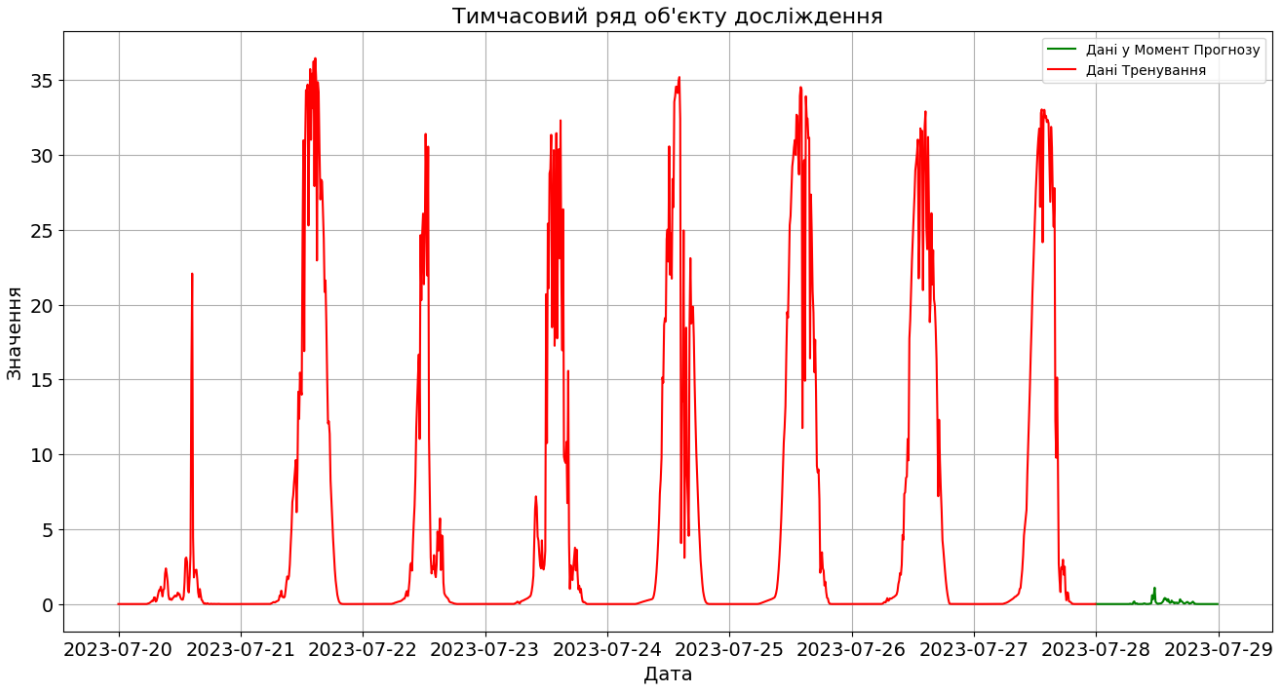


Рисунок 4.18 – Зображення часового ряду експерименту.

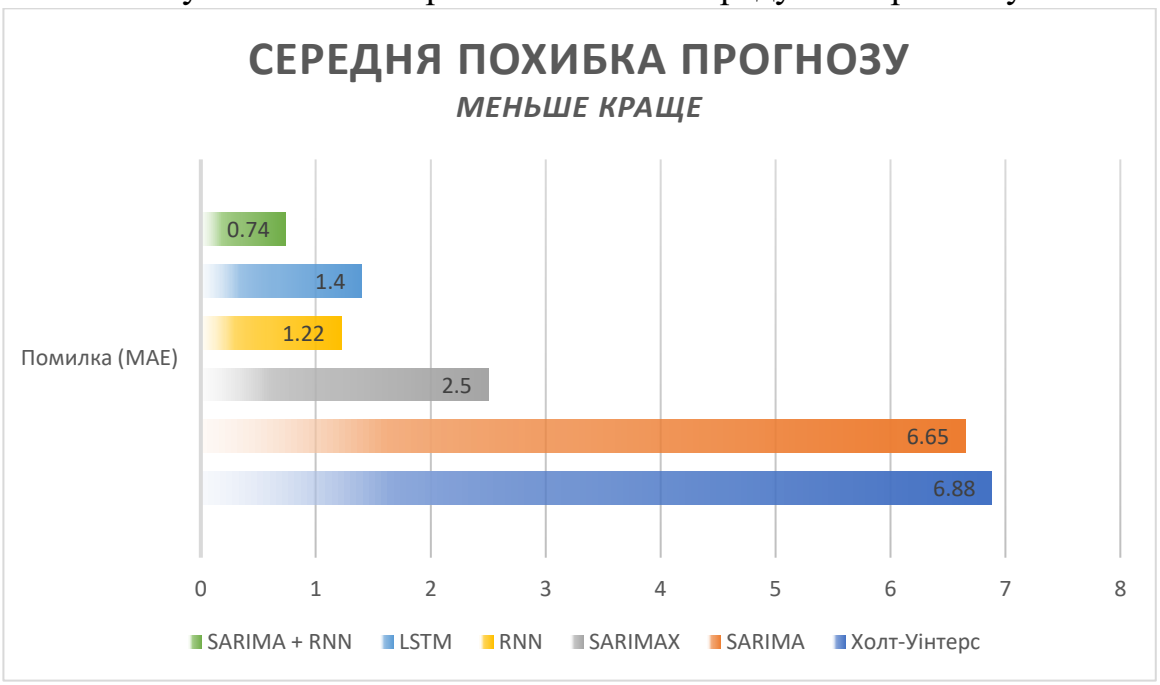


Рисунок 4.19 – Результат експерименту точності.

У третьому розділі, проведений експеримент точності на досліджувальній виборці. Й результати які були отримані повністю відрізняються від тих, що отримані зараз, так як точність залежить від часового ряду, який використовується у виборці. У цьому експерименті перевіряється реакція моделі на несподівану для часового ряду зміну даних. В цілому за графіком 4.19 все стає зрозумілим й логічним. Моделі Холта-Уінтерса та SARIMA мають найгірший

результат, бо не здатні оцінювати зовнішні фактори. Модель SARIMAX має точність у 2,7 рази більшу, завдяки вдало підібраному екзогенному параметру. Інтенсивність сонячного випромінювання працює набагато краще за хмарність. Але точність можна отримати більшу, як би була можливість безкоштовно отримати інформацію не по годино, а наприклад по 10 хвилин. Нейромережі виконали прогноз ще краще у 5,5 разів точніше за прості моделі та у 2 рази за модель SARIMAX. Ці системи здані аналізувати величезну кількість параметрів, що й призводить до кращого результату прогнозу. Модель з довгостроковою пам'яттю має трохи більшу похибку. Скоріш за все це пов'язано з простотою даних, які необхідно прогнозувати. Найкращій результат отримала гібридна модель. Поєднання нейромережі, яка аналізує зовнішні чинники, та SARIMA, яка аналізує властивості часового ряду, створюють потужну модель для прогнозування.

Експеримент з точності на довший період (Рис. 4.20), не й де в супереч минулим тестам. При більш довгому прогнозі лідерство залишається за гібридною системою, а нейромережа з довгостроковою пам'яттю починає обходити RNN, у зв'язку з збільшеною складністю даних.

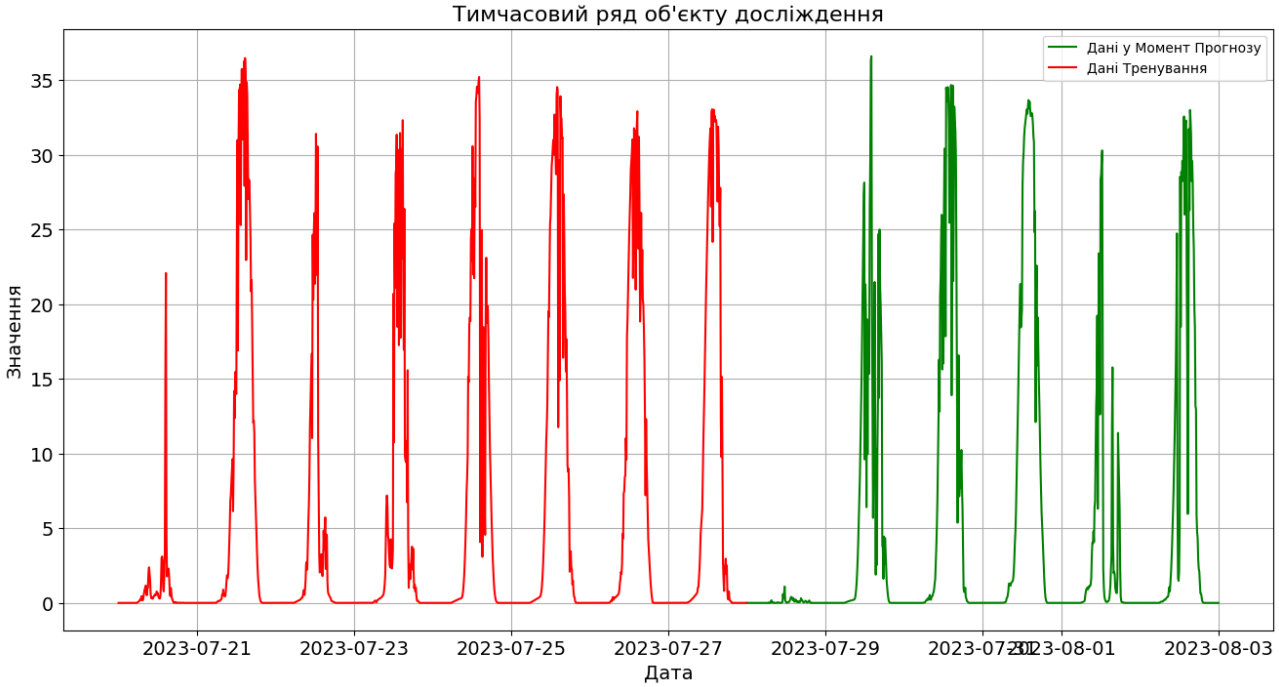


Рисунок 4.20 – Зображення часового ряду експерименту.

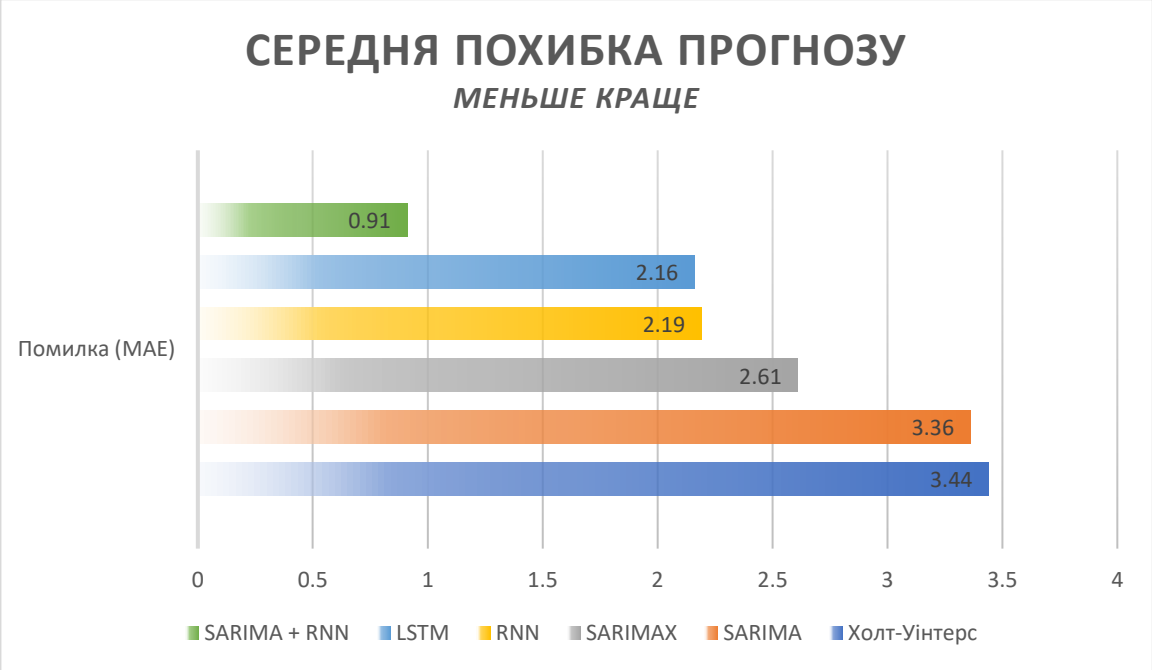


Рисунок 4.21 – Результат експерименту точності.

4.3 Висновки за розділом

У розділі розглянуті технологічні інновації в області комп'ютерних розрахунків. Проведені експерименти з встановлення ефективності подібних технологій у об'єкті дослідження. Проведені експерименти стосовно ресурс ємності методів прогнозування та тести точності прогнозування. Цього достатньо для підведення висновку стосовно усіх моделей прогнозування, які були розглянуті у дипломному проекті.

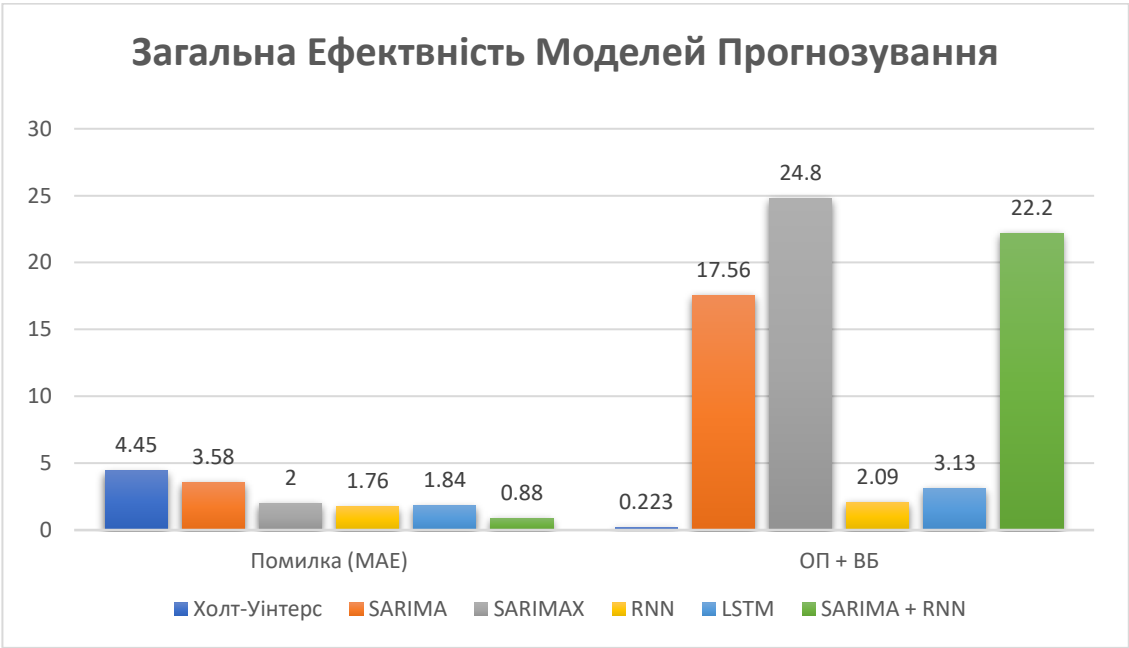


Рисунок 4.22 – Загальний результат моделей.

На рисунку 4.22 зображена загальна похибка моделей в усіх можливих випадках поведінки часового ряду. ОП + ВБ це використання пам’яті відповідною моделлю. А на графіку 4.23 вказана орієнтовна необхідна потужність для виконання прогнозу у FLOPS. Розрахунок відбувався відповідно до результатів дослідження, та технічної документації Intel та Nvidia. Приклад розрахунку для Холта-Уінтерса:

$$GF = SMT * Clock * k * CPL * t \quad (4.1)$$

SMT – кількість задіяних потоків у розрахунку, Clock – частота процесору, k-коефіцієнт продуктивності AVX (I7 3770 = 8 Fl/clock), CPL- завантаженість ALU, t – час операції (у випадку t>1 виконується ділення).

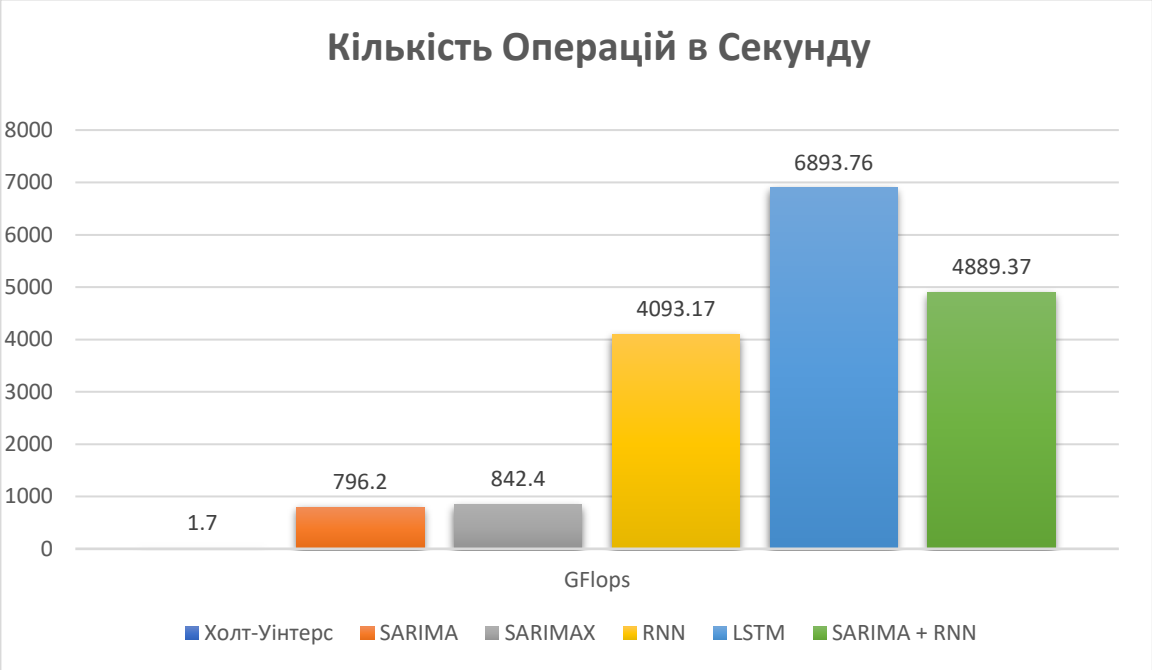


Рисунок 4.23 – Загальна потреба у кількості операцій FP32 в секунду.

- Модель Холта-Уінтерса: найшвидша з усіх моделей, потребує мінімальну кількість ресурсів. Має найгіршу точність, не здатна оцінювати зовнішні фактори. Не ефективна при прогнози на більше ніж один сезон. Має не статичні коефіцієнти, які потрібно перебирати кожен раз коли змінюється часовий ряд. Має постійну трендову складову, навіть якщо в часовому раді загальний тренд відсутній. В цілому дана модель може використовуватися у малопотужних приладах наприклад на ESP 32 чи інших контролерах.

- Модель SARIMA: Показує результат схожий з попередньою моделлю. Не має проблеми повторення сезону, а отже теоретично може використовуватися для прогнозування на більші терміни. Має універсальні коефіцієнти, які є постійними в незалежності від даних у часовому ряді потужності. Правильно оцінює наявність тренду. Потребує дуже багато ресурсів, даючи при цьому мінливі переваги. В цілому система SARIMA має найгірший результат з усіх моделей у відношенні точність/ресурсоємність.
- Модель SARIMAX: Відображає прогноз з більшою точністю, має усі переваги моделі SARIMA. Наявність екзогенного параметра позитивно впливає на прогноз, система може використовуватися на довші терміни прогнозування. Використання ресурсів найбільше з усіх моделей. З урахуванням точності має право на життя, та подальшого використання у об’єкті дослідження.
- RNN: Фактично найкраща модель прогнозування. Поєднує у собі велику точність, швидкість, простоту та надійність. Хоч й потребує багато ресурсів. Компенсує це підтримкою спеціальних технологій. Не потребує повторного навчання при кожному прогнозі. Навчається на кожному результаті, з часом якість буде збільшуватись. Здатна реагувати на нестандарті чинники та знаходити не очевидні залежності самостійно.
- LSTM: Модель, яка має усі переваги RNN, у випадку складання прогнозу на термін від 5 днів стає ефективнішою за інші не комбіновані моделі. Але постає питання про отримання точних даних прогнозу погоди на такий довгий термін. На короткий термін не ефективна, легко перенавчається на такому простому наборі даних. Створена для довших і складніших залежностей. Потребуючи при цьому майже в двічі більше ресурсів GPU.
- SARIMA + RNN: Єдина комбінована система у дипломному проекті, ідея якої виникла під час дослідження, для комбінування основних переваг статистичних систем та нейромереж. Має кращій показник точності. За наявності якісних даних прогнозу, наприклад з використанням

нейромережі Google MetNet. Можна досягти ідеальних результатів прогнозу. Єдиним недоліком є ресурсоємність. Ця система може бути реалізована виключно на потужному обладнання.

Результат дослідження наступний: найкраща модель із досліджуваних, це RNN. За наявності великої кількості ресурсів краще використовувати комбіновану систему прогнозування SARIMA + RNN. Інші моделі не є доцільними у використанні відповідно до об’єкту дослідження.

Розділ також підтвердив гіпотезу ефективності нейромереж у аналізі часових рядів. Заперечив домінацію складних систем ARMA порівнюючи з простою моделлю Холта-Уінтерса, яка фактично являєтеся трійним експоненційним згладжуванням.

5 ЕКОНОМІКА

5.1 Визначення капітальних витрат з впровадження кіберфізичної системи

У випадку коли замовник не має потрібного обладнання для реалізації системи прогнозування, його необхідно закупити. Визначення подібних витрат відбуваються відповідно до:

$$К_{ПКВ} = С_{ОБ} + Д_{ТР} + М_{МН} + К_{ПЗ} \quad (5.1)$$

Де, $К_{ПКВ}$ – витрати, пов'язані із капітальним проектом (грн.); $С_{ОБ}$ – загальна вартість основного та допоміжного обладнання (грн.); $Д_{ТР}$ – витрати на транспортування та заготівлю (грн.); $М_{МН}$ – кошти, витрачені на монтаж і налагодження системи (грн.); $К_{ПЗ}$ – витрати, пов'язані з розробкою програмного забезпечення (грн.).

Вартість усього необхідного обладнання наведена у таблиці 5.1.

Таблиця 5.1 – Вартість обладнання

№	Найменування	Кількість	Вартість	Сума
1	Nvidia Jetson Nano Developer Kit V3	1	9 999 грн	9 999 грн
2	Блок живлення	1	1 400 грн	1 400 грн
3	Металевий корпус	1	650 грн	650 грн
4	Система охолодження	1	499 грн	499 грн
			Разом	12 548 грн

Виходить, що $С_{ОБ} = 12\,548$ грн.

Витрати на транспортно-заготівельні та складські роботи розраховуються як 8% від загальної вартості обладнання.

$$Д_{ТР} = С_{ОБ} * 0.08 \quad (5.2)$$

Вартість транспортно-заготівельних та складських робіт становить частку від загальних витрат:

$$Д_{ТР} = 12\,548 * 0.08 = 1003 \text{ грн}$$

Вартість монтажно-налагоджувальних робіт розглядається як 5% від вартості обладнання.

$$М_{МН} = С_{ОБ} * 0.05 \quad (5.3)$$

Витрати на монтажно-налагоджувальні роботи:

$$М_{МН} = 12\,548 * 0.05 = 627 \text{ грн}$$

5.2 Визначення трудомісткості розробки програмного забезпечення

Вартість розробки програмного забезпечення визначається шляхом оцінки трудовитрат, які включають в себе:

$$t = t_o + t_u + t_a + t_n + t_{OT} + t_g \quad (5.4)$$

Елементи витрат праці визначаються на основі умовної кількості оброблюваних операторів у програмному забезпеченні:

$$Q = q * c * (1 + p) \quad (5.5)$$

де: Q – умовна кількість операторів в програмному забезпеченні;

q – фактична кількість операторів у програмному забезпеченні (за умовчанням, q = 120 для програмного забезпечення);

c – коефіцієнт складності програми (приймається c = 1,75);

p – коефіцієнт корекції програми в процесі обробки (приймається p = 0,3).

Для ПЗ, що розроблялося:

$$Q = 120 * 1.75 * (1 + 0.3) = 273$$

Витрати праці на підготовку та опис завдання в кваліфікаційній роботі становлять 60 людино-годин.

Витрати праці на вивчення опису завдання визначаються з урахуванням уточнення опису та кваліфікації програміста, розглядаючи:

$$t_u = \frac{Q*B}{80*k} \quad (5.6)$$

де:

- B – коефіцієнт збільшення витрат праці (приймається B = 1.5);
- k – коефіцієнт кваліфікації програміста (приймається k = 1.4).

Для ПЗ, що розроблялося:

$$t_u = \frac{273 * 1.5}{80 * 1.4} = 3.7$$

Витрати на розробку кіберфізичної системи визначаються за допомогою:

$$t_a = \frac{Q}{22.5*k} \quad (5.7)$$

Для ПЗ, що розроблялося:

$$t_a = \frac{273}{22.5 * 1.4} = 8.7$$

Витрати праці на налаштування програми розраховуються за допомогою:

$$t_n = \frac{Q}{4*k} \quad (5.8)$$

Для ПЗ, що розроблялося:

$$t_n = \frac{273}{4 * 1.4} = 48.8$$

Витрати праці на підготовку документації по завданню визначаються як:

$$t_g = t_{ДР} + t_{ДО} \quad (5.9)$$

де:

- tДР – трудомісткість підготовки матеріалів до написання;
- tДО – трудомісткість редагування, друку та оформлення документації.

Трудомісткість підготовки матеріалів до написання визначається як:

$$t_{ДР} = \frac{Q}{15*k} \quad (5.10)$$

Для ПЗ, що розроблялося:

$$t_{ДР} = \frac{273}{15 * 1.4} = 13$$

Трудомісткість редагування, друку та оформлення документації визначається як:

$$t_{ДО} = 0.75 * t_{ДР} \quad (5.11)$$

Для ПЗ, що розроблялося:

$$t_{ДО} = 0.75 * 13 = 9.75$$

Для розробленого програмного забезпечення витрати праці на підготовку документації по завданню розраховуються за допомогою:

$$t_g = 13 + 9.75 = 22.75$$

Таким чином трудомісткість розробки ПЗ становить:

$$t = 60 + 3.7 + 8.7 + 48.8 + 22.75 = 144$$

5.3 Витрати на створення програмного забезпечення

Витрати на створення програмного продукту розраховуються як:

$$K_{ПЗ} = З_{ЗП} + З_{МІ} \quad (5.12)$$

Де, $З_{ЗП}$ включає в себе затрати на заробітну плату розробника програмного забезпечення, а $З_{МІ}$ представляє вартість машинного часу, необхідного для налаштування програми (в гривнях).

Оплата праці розробника програмного забезпечення визначається так:

$$З_{ЗП} = t * С_{ПР} \quad (5.13)$$

Де, $С_{ПР}$ - Середня годинна тарифна ставка розробника програмного забезпечення приймається як 150,00 грн. на годину.

Для ПЗ, що розроблялося:

$$З_{ЗП} = 144 * 150 = 21600 \text{ грн}$$

Вартість машинного часу, потрібного для налаштування програми, визначається як:

$$З_{МІ} = t_n * С_{МЧ} \quad (5.14)$$

Де, $С_{МЧ}$ – вартість машинного часу, приймається $С_{МЧ}$ = 22 (грн./год.).

Для ПЗ, що розроблялося:

$$З_{МІ} = 48.8 * 22 = 1073.6 \text{ грн}$$

Вартість розробки програмного забезпечення:

$$K_{ПЗ} = 21600 + 1073.6 = 22673.6 \text{ грн}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k * F_p} \quad (5.15)$$

Число виконавців, позначене як B_k, та місячний фонд робочого часу, обчислено на основі 40-годинного робочого тижня, позначеного як $F_p = 176$ годин.

$$T = \frac{144}{1 * 176} = 0.81 \text{ міс.}$$

Тепер можна остаточно знайти капітальні витрати:

$$К_{ПКВ} = 12\,548 + 1003 + 627 + 22674 = 36\,852 \text{ грн}$$

5.4 Розрахунок експлуатаційних витрат

Експлуатаційні витрати розраховуються як:

$$C_e = C_a + C_З + C_c + C_{РО} + C_{ее} + C_{ІНШ} \quad (5.16)$$

Де, C_e – річні поточні витрати, пов’язані із застосуванням системи керування (грн.); C_a – амортизація основних фондів (грн.); $C_З$ – заробітна плата обслуговуючого персоналу (грн.); C_c – відрахування на соціальні заходи (грн.); $C_{РО}$ – витрати на технічне обслуговування та поточний ремонт обладнання (грн.); $C_{ее}$ – вартість електроенергії, $C_{ІНШ}$ – інші витрати.

Терміни амортизації для різних груп об'єктів основних засобів встановлені відповідно до їхньої класифікації. Розроблене у кваліфікаційній роботі обладнання системи прогнозування відноситься до 4-ої групи (машини та обладнання), і передбачений термін його експлуатації становить 5 років.

$$H_a = \frac{2}{T} * 100\% \quad (5.17)$$

Амортизація основних фондів визначається як:

$$C_a = \frac{ПВ * H_a}{100\%} \quad (5.18)$$

Для розробленої системи:

$$H_a = \frac{2}{5} * 100\% = 40\%$$

$$C_a = \frac{36\,852 * 40\,\%}{100\%} = 14\,741 \text{ грн}$$

Система не потребує нового робітника, тому пропонується додати існуючому спеціалісту надбавку, у розмірі 1 000 гривень для обслуговування подібної системи.

$$C_З = 1000 \text{ грн}, C_c = 0 \text{ грн}$$

Вартість технічного обслуговування та поточного ремонту обладнання та мережі розглядається як 10% від суми капітальних витрат:

$$C_{РО} = 0.1 * К \quad (5.19)$$

Для розробленої системи:

$$C_{РО} = 0.1 * 36\,852 = 3\,685.2 \text{ грн}$$

Система потребує приблизно 200 Вт на годину. Повинна працювати у режимі 24/7. Що становить близько 134 КВт в місяць. Але СЕС може використовувати свою ж енергії для роботи цієї системи, тому справжні витрати нижче. Подібні видатки є виключно проблемою замовника.

$$C_{ee} = 134 * 2.64 = 355 \text{ грн}$$

Від людини не потребується спецодяг, чи додаткові витрати з охорони праці. Тому $C_{ІНШ} = 0$

Таким чином загальні витрати становлять:

$$C_e = 14\,741 + 1000 + 0 + 3\,685.2 + 355 + 0 = 19\,781 \text{ грн}$$

5.5 Маркетингові дослідження ринку збуту розробленого програмного продукту

У сучасному світі фотоелектричні установки виявляють зростаючий інтерес і стають популярним джерелом вироблення електроенергії. Однак для ефективного використання сонячної енергії необхідно точно прогнозувати потужність фотоелектричних установок. У зв'язку з цим, наша компанія розробила програмний продукт, який надає точні та надійні прогнози потужності сонячних установок.

Метою маркетингових досліджень є вивчення та оцінка ринкового попиту на розроблений програмний продукт, визначення ключових сегментів ринку та розробка стратегії збуту для досягнення успіху на ринку.

Аналіз Ринку

1. *Визначення Потенційних Користувачів*

   Основними користувачами програмного продукту є компанії та індивідуальні споживачі, які володіють або виробляють фотоелектричні установки. Особливий акцент робиться на власників сонячних ферм, альтернативних енергетичних підприємств та інженерів, відповідальних за монтаж та обслуговування сонячних панелей.

2. *Аналіз Конкуренції*

Ми провели аналіз конкурентів у сфері програмних продуктів для прогнозування потужності сонячних установок. Оцінка показала, що наш продукт вирізняється точністю та інноваційністю.

Стратегія Збуту

1. *Прямий Збут*

   Забезпечимо можливість придбання програмного продукту напряму через наш веб-сайт. Також розглядається можливість партнерства з виробниками фотоелектричних установок для включення програмного продукту в комплексні рішення.

2. *Співпраця з Інтернаціональними Партнерами*

   Розглядається партнерство з компаніями, які вже працюють у сфері сонячної енергії в різних частинах світу. Це дозволить нам розширити географію збуту та забезпечити відповідь на глобальний попит.

3. *Онлайн-Маркетинг та Реклама*

   Використовуватимемо цифрові маркетингові стратегії, такі як контент-маркетинг, реклама в соціальних мережах і пошукова оптимізація для просування продукту та залучення уваги цільової аудиторії.

Прогноз Розвитку Ринку

Очікується, що ринок фотоелектричних установок буде продовжувати рости, оскільки є зростаючий інтерес та усвідомленість щодо сталості енергії. Наша компанія готова взяти участь у цьому рості, завдяки інноваційному програмному продукту та ефективній стратегії збуту.

5.6 Оцінка економічної ефективності впровадження програмного забезпечення

Ключові показники економічної результативності системи управління, що розробляється:

- Річний економічний ефект:

$$E = \Delta П - \Delta С - \Delta К * Е_{П} > 0 \qquad (5.20)$$

- Економічна ефективність:

$$E_Г = ΔП - ΔС \quad (5.21)$$

- Термін окупності розробки:

$$T_{ОК} = \frac{ΔК}{Е_Г} \quad (5.22)$$

$$Е_П = \frac{N_{КР} - N_{ІНФ}}{100} \quad (5.23)$$

Взято нормативне значення банківської кредитної ставки ($N_{КР}$) у розмірі 25%, із врахуванням інфляції ($N_{ІНФ}$ = 20,0%).

$$Е_П = \frac{25 - 20}{100} = 0.05$$

Система прогнозування не здатна збільшити кількість генеруємої енергії. Тому не впливає на прибутковість. Але це зовсім інша ситуація, подібні системи розробляються не з ціллю збільшити прибутковість, а спростити життя та значно укріпити надійність системи. Виникає складність оцінки економічної ефективності.

Вартість готових систем прогнозування складає від 20 тисяч доларів до 500 тисяч. Обслуговування системи для замовника, обійдеться не менше 3 000 доларів на місяць.

Розробка програмного забезпечення його підтримка та модернізація. Ось головні витрати. Головне джерело прибутку, це продаж системи та щомісячна «підписка». Приймемо ціну за купівлю програмного забезпечення 40 000 доларів, а в якості витрат, робота програміста, маркетолога, юристів, транспортних компаній та ціна обладнання…

- Річний економічний ефект, фактична вартість для замовника:

$$E = 0 - 1\,480\,000 = -1\,480\,000 \text{ грн}$$

- Економічна ефективність:

$$E_Г = 1\,480\,000 - 180\,260 = 1\,299\,740 \text{ грн}$$

- Термін окупності розробки:

$$T_{ОК} = \frac{180\,260}{1\,299\,740} = 0.14 \text{ (року)}$$

5.7 Висновки за розділом

Висновок до економічного розділу вказує на важливі аспекти ефективності системи прогнозування потужності сонячної електростанції. Розглядаючи параметри економічного аналізу, можна зазначити, що запровадження цієї системи є обґрунтованим і виправданим в контексті підвищення продуктивності та надійності фотоелектричних модулів.

Розрахунки затрат на розробку, встановлення та експлуатацію підтверджують фінансову вигідність проекту, особливо при урахуванні факторів, таких як амортизація, технічне обслуговування та інші витрати експлуатації. Вартість системи компенсується покращеною точністю прогнозування, що призводить до оптимізації виробництва та ефективнішого використання сонячної енергії.

Зазначено, що система прогнозування потужності сприяє зменшенню ризиків і надає операторам станції можливість своєчасно реагувати на зміни у виробництві електроенергії. Це не лише підвищує надійність роботи, але й сприяє збільшенню виробництва та зниженню витрат на виробництво електроенергії з сонячних джерел.

Таким чином, на основі економічного аналізу можна зробити висновок, що впровадження системи прогнозування потужності сонячної електростанції є обґрунтованим кроком, спрямованим на оптимізацію функціонування станції та забезпечення стабільного та ефективного виробництва електроенергії.

6 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

6.1 Аналіз небезпечних і шкідливих виробничих чинників сонячної фотоелектричної установки

Проблема у аналізі небезпечних і шкідливих чинників полягає у тому, що розроблена система є повністю автоматичною, та кіберфізичною. Вона не потребує обслуговування, робітників. Тобто це просто алгоритм, метод для прогнозування. Але оскільки диплом вимагає чітку структуру розділів, то проаналізуємо небезпеку фотоелектричних установок.

У даному випадку, установка володіє кількома потенційно небезпечними факторами. Конструкція повністю виготовлена з металу та обладнана шафою з інвертором, в якій присутня напруга 220 вольт. Існує ризик ураження електричним струмом. Кабельний канал, прокладений у землі, перетинає лінії високої напруги. Навіть коли система вимкнена, вона продовжує генерувати електроенергію під впливом сонячного світла і може призвести до:

- Термічних опіки
- Пошкодженню м’язів, нервів і тканин
- Падіння від шоку
- Летальних випадків

Сонячні пеналі генерують небезпечний рівень постійного струму. У разі ураження відбувається постійне скорочування м’язів, що ускладнює розрив контакту з об’єктом, який знаходяться під напругою.

У пристрої для відстеження існують рухомі вузли та механізми, тому невідповідальне обслуговування та непослідовне дотримання правил безпеки може призвести до травм. Конструкція поворотної рами має малу площину профілю, а двигун створює зусилля у 600 кг, що достатньо для заподіяння серйозних ушкоджень при недбалому використанні.

- Переломи або розтягнення
- Колоті ушкодження
- Травми спини, шиї та голови
- Порізи та синці

- Внутрішні травми

Робота на відкритому повітрі протягом тривалого періоду у жаркому та вологому середовищі може спричинити теплові захворювання, дегідратацію та втому. Працівник також може відчувати відблиски від поверхонь фотоелектричних модулів під час роботи протягом дня. Ці відблиски можуть бути як прямими, так і непрямими, і можуть негативно впливати на зір. Це, в свою чергу, може впливати на продуктивність роботи, ускладнювати процес встановлення модулів та погіршувати безпеку монтажу, оскільки встановлення болтів, гайок та іншого обладнання є візуально важливим завданням. Працівник витримує тривалий вплив сонячного випромінювання, яке ще підсилюється відбиттям від сонячних панелей. Негативні наслідки включають ризик захворювань, таких як катаракта, меланома, сонячні опіки та деградація шкіри.

Під час технічного обслуговування може виникнути необхідність у заміні або видаленні фотоелектричного модуля, які характеризуються значною вагою та габаритами, особливо панелі високої потужності, що перевищують 300 Вт. Неправильний підйом може викликати розтягнення та серйозні травми спини, такі як:

- Грижа міжхребцевих дисків
- Розрив ротаторної манжети
- Розтягнення стегон і нижньої частини спини

6.2 Інженерно-технічні заходи з охорони праці

Для проведення обслуговування установки приймаються особи, які не молодше 18 років та мають повну середню освіту разом із професійно-технічною освітою, або повну загальну освіту, а також професійну підготовку на виробництві. Вимог до стажу роботи немає. Працівник повинен бути у відповідному спеціальному одязі та мати необхідне обладнання, зокрема: робочий одяг для сезону роботи, захисні рукавиці для уникнення ураження струмом, а також радіостанцію для зв'язку з оператором у випадку аварійної ситуації.

Заземлення усього обладнання є необхідним не лише для відповідності вимогам охорони праці, але й для забезпечення захисту технічного обладнання та інформаційних каналів від грози та екранування від електромагнітних перешкод. Експлуатація установки без заземлення заборонена. Заземлення установки рекомендується проводити на місці, а в разі неможливості цього застосовується окремий кабель, перетин якого перевищує фазову жилу в 1,5 рази. Усе обладнання встановлюється в металеві герметичні бокси з рівнем захисту не менше IP 65. Захист установки забезпечується за допомогою пристрою захисного відключення, розташованого на виході інвертора та на вході в електричну мережу, в розподільчому пункті.

Використання двох металевих балок конструкції як заземлення заборонено, оскільки їхній супротив перевищує рекомендоване значення в 4 Ом для 220 В. Замість цього передбачено використання окремого заземлюючого контуру. Схема підключення обладнання подана на малюнку 6.1.

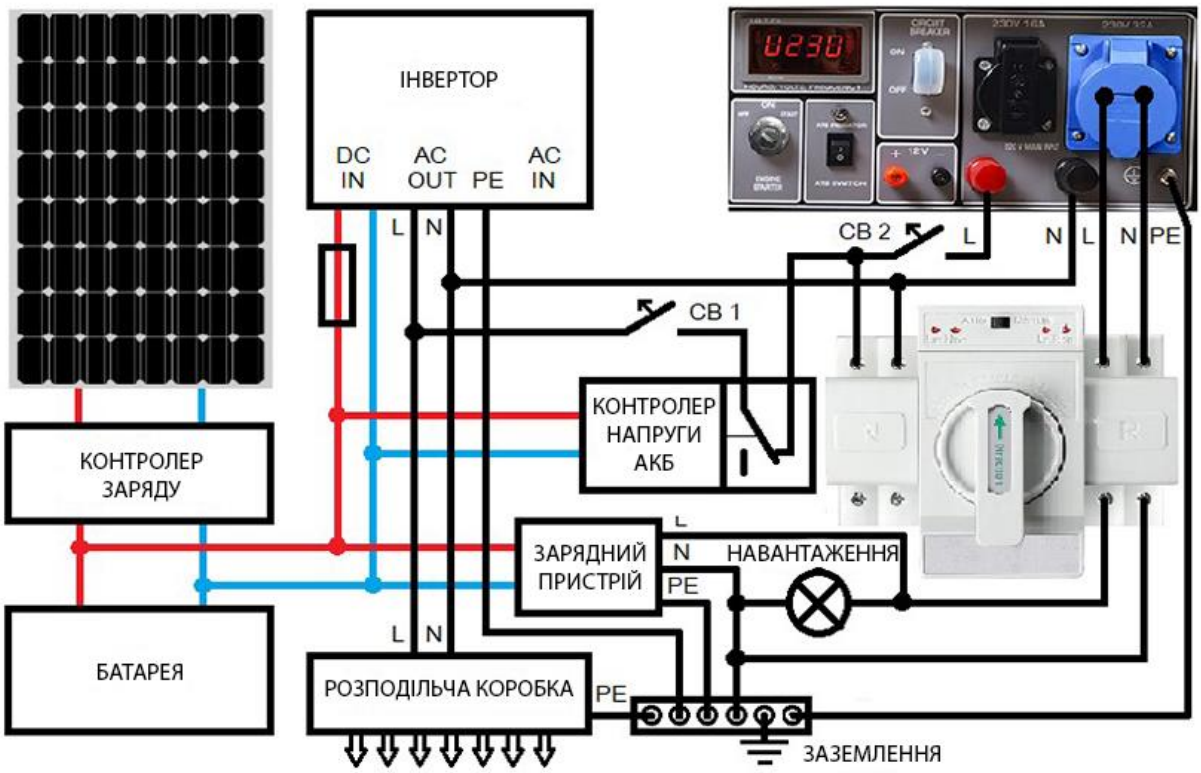


Рисунок 6.1 – Схема підключення обладнання

Рекомендовані параметри для системи заземлення установки включають в себе використання 6 вертикальних заземлювачів діаметром 5 см та довжиною 3

метри кожен. Умови передбачають кліматичну зону 1 та верхній ґрунтовий шар - чорнозем, а нижній - глину. Горизонтальний заземлювач має бути встановлений на глибині 1 метр і мати ширину полозки 5 см, а довжина з'єднань становить 2 метри. Заземлювачі призначені для зниження загального супротиву системи, і в даному випадку очікуваний загальний супротив складає 3,622 Ом.

Проведення електро-технічного обслуговування забороняється під час несприятливих погодних умов, таких як снігопад, дощ, град та інші види опадів.

У процесі виконання робіт з електроустаткування, необхідно накривати сонячну батарею непрозорим листом з метою припинення генерації енергії від сонця. Завжди слід блокувати та відключати джерела постійного та змінного струму. Перед початком робіт слід перевіряти електричні схеми, щоб переконатися, що вони знаходяться у вимкненому стані, і тільки тоді розпочинати виконання завдань.

Перед тим як приступити до роботи з фотоелектричними панелями, використовуйте струмовий затискач або вольтметр, щоб перевірити наявність небезпечної енергії. Особливу увагу слід звертати на інвертори, оскільки вони можуть утримувати заряд навіть після відключення живлення.

Ніколи не від’єднуйте фотоелектричні роз’єми, затискачі, клемники або інше обладнання, яке знаходиться під навантаженням. Носити виключно спец одяг, який відповідає нормам електробезпеки.

Кабельні лінії, які вводяться до шаф, прокладаються у металевому або пластиковому рукаві з використанням сигнальної стрічки та підводяться до шафи у металевій трубі. При використанні броньованого кабелю можна відмовитися від використання рукаву. Весь кабельний матеріал повинен мати ізоляцію, яка не підтримує горіння. На шафі з інвертором передбачено встановлення попереджувальної таблички. Біля вхідної клеми обов'язково встановлюється автоматичний вимикач для захисту від струмів короткого замикання, що працює при робочому напрузі 220 В і струмі спрацювання від короткого замикання 2 А. Наслідки недотримання цих вимог зображені на рисунку 6.2.

Рисунок 6.2 – Наслідкі не дотримань вимог з безпеки

Встановлення вимикача після інвертора не є доцільним, оскільки він вже обладнаний вбудованим автоматичним вимикачем у своєму корпусі.

Використання редукторів відкритого типу без захисного кожуха заборонено. Небезпечні частини рухомої конструкції повинні бути виділені окремим кольором. На найвищій балці обов'язково встановлюється металевий гачок для закріплення захисного поясу. Обслуговування дозволяється тільки після повідомлення оператора, який має обов'язок вимкнути автоматичний режим роботи установки. Якщо роботу виконує черговий інженер, установку слід зупинити заздалегідь.

Перед використанням підйомного обладнання обов'язково проведіть перевірку на цілісність. Якщо виявлено дефекти, позначте їх стрічкою або етикеткою з написом "Не використовувати" або "Технічно не справно". Використовуйте драбини зі скловолокна чи інших діелектричних матеріалів, особливо поблизу джерела живлення. Металеві або алюмінієві сходи можуть

бути небезпечними в областях, де є лінії електропередачі або ведуться електричні роботи.

Встановлювати сходи дозволено лише на сухій рівній землі та на безпечній відстані від ліній електропередачи. Закріплюйте сходи на землі або на верхній частині металоконструкції трекера. Беріться лише за горизонтальні сходинки, а не за вертикальні рейки та дотримуйтесь 3 точок хвату.

Пам'ятайте, ніколи не піднімайте сонячні панелі чи інше обладнання під час підйому по драбині. Завжди використовуйте лебідку або систему підйому для цих робіт.

Перед початком роботи завжди визначайте всі потенційні небезпеки, спрямовані на уникнення спотикання та падінь, що дозволяє мінімізувати ризик отримання травм. Підтримуйте чистоту на робочому місці, постійно переконуючись, що на робочій поверхні відсутні залишки мастила, льоду, води або інших речовин.

Для транспортування та заміни сонячних елементів у випадку необхідності застосовуються візки, вилочні навантажувачі або інші методи, які не вимагають ручної праці.

Використовуйте рукавички для захисту рук та покращення зчеплення. Завжди тримайте сонячні елементи обома руками та використовуйте плавні та рівні рухи. Тримайте вантаж якнайближче до себе. Не скручуйте тіло під час переносу вантажу; зробіть крок в одну чи іншу сторону, щоб повернутися. Не дозволяйте собі замінювати пошкоджені елементи без захисних окулярів, рукавичок та закритого одягу. Під час роботи у сонячну погоду надягайте головний убір та сонцезахисні окуляри.

6.3 Пожежна профілактика

Згідно з нормативним документом ДСТУ Б В.1.1-36:2016, установка відноситься до категорії «Д» відповідно до таблиці 6, оцінюючи вибухо-пожежну та пожежну небезпеку. В ній відсутні вибухонебезпечні речовини, а кількість речовин, які підтримують горіння, мінімізована. Основним джерелом пожежі є вихід з ладу обладнання або перегрів сонячних фотоелектричних

модулів. У сонячних модулях присутня не велика кількість полімерних капсул що оточують фотоелементи, полімерні листи основи, пластикові з’єднання коробки на задній панелі.

Для забезпечення високого рівня пожежної безпеки та уникнення можливості загоряння установки та подальшого розповсюдження вогню через суху траву чи інші горючі матеріали, розроблені профілактичні заходи.

Недопустимо накопичення сміття, заростів чи інших пожежонебезпечних об'єктів на майданчику, де розташована установка. Зона, в якій встановлюється пожежний щит, повинна мати додаткове освітлення для швидкого виявлення при недостатній освітленості.

Необхідно забезпечити наявність заправлених вогнегасників, наповнених контейнерів з піском. Для тушіння установки використовувати порошкові вогнегасники, які встановлюються у пожарному щиті та слід застосувати для гасіння твердих, рідких, газоподібних речовин, електроустановок під напругою до 1000В. Кількість вогнегасників ВП-5 (див. Рис. 6.3) повинна відповідати кількості установок.

Рисунок 6.3 – Вогнегасник ВП-5

Для запобігання розповсюдження вогню в природному середовищі використовується система протипожежного водопостачання низького тиску (див. Рис. 6.4). Джерелами води можуть бути озера або річки, водопровідні системи та протипожежні резервуари.

Рисунок 6.4 – Протипожежне водопостачання.

Для оповіщення про виникнення пожежі використовуються системи сигналізації, які здатні привернути увагу працівників підприємства та сповістити пожежну охорону при спрацьовуванні. Для об'єкта на відкритій місцевості можуть бути застосовані датчики вогню (див. Рис. 6.5).

Рисунок 6.5 – Інфрачервоний датчик вогню

Датчик встановлюється на металевій опорі висотою 20 метрів і призначений для контролю за наявністю полум'я на площі до 200 квадратних метрів. Час реакції такого датчика становить 5 секунд. Біля пожежних щитів також встановлюються порогові ручні сповіщувачі для самостійного виклику пожежників.

ВИСНОВКИ

У ході виконання даного дипломного проекту було проведено комплексне дослідження та розробка кіберфізичної системи прогнозування потужності фотоелектричної установки.

Аналізуючи галузь промисловості, технологічний процес та об’єкт управління, ми визначили ключові аспекти та встановили завдання дослідження.

У теоретичному розділі були розглянуті та детально вивчені різні моделі прогнозування, включаючи тимчасові ряди, модель Холта-Уінтерса, ARMA та нейромережі. На основі цих даних була розроблена кіберфізична система, яка враховує екзогенні параметри та використовує різні методи прогнозування.

Отримані результати дослідження демонструють високий рівень точності і ефективності розроблених методів прогнозування. Це свідчить про їхню здатність надійно прогнозувати потужність фотоелектричної установки в різних умовах.

Експериментальний розділ дозволив провести тестування точності розроблених моделей та дослідити апаратні можливості, використані під час розробки. Результати свідчать про ефективність та точність прогнозування.

Система прогнозування, розроблена на основі вивченої теоретичної бази та інноваційних технічних досягнень, проявила високий рівень стабільності та надійності під час реальних експлуатаційних умов.

У розділі економіки були розглянуті всі аспекти витрат, включаючи капітальні та експлуатаційні витрати, а також проведено маркетингові дослідження та оцінку економічної ефективності.

Охорона праці та безпека у надзвичайних ситуаціях були визначені через аналіз небезпечних чинників та заходів з охорони праці.

Дипломний проект в цілому відповідає поставленим завданням та досягнутий визначеної мети. Розроблена кіберфізична система прогнозування є перспективною для впровадження в сучасних умовах. Результати експериментів та економічні розрахунки свідчать про ефективність та рентабельність

використання розробленої системи. Заходи з охорони праці дозволяють забезпечити безпечну експлуатацію фотоелектричної установки.

Цей проект відкриває широкі можливості для подальших досліджень та розвитку в галузі кіберфізичних систем та прогнозування відновлюваної енергії. У своєму комплексі він представляє інтегроване рішення для оптимізації роботи фотоелектричних установок з використанням передових методів прогнозування та управління.

ПЕРЕЛІК ПОСИЛАНЬ

1. «Pmdarima: ARIMA estimation for Pyhton» Taylor G Smith.
2. Research Article: «Time Series Forecasting» Can Ozdogar.
3. Research Article: «Time Series Forecasting with ARIMA» Brendan Artley.
4. Research Article: «Global Market Outlook For Solar Power 2023-2027» Micharl Schmela, Walburga Hemetsberger, Theresa Cruz.
5. International Journal of Sustainable Engineering «PV power forecasting based on data-driven models» Priya Gupta, Rhythm Singh.
6. CUDA Toolkit Documentation 12.3. (https://docs.nvidia.com/cuda/)
7. Intel Press Release 2013.
8. «Forecasting: Principles and Practice» Rob J Hyndman, George Athanasopoulos (3rd Edition, 2021)
9. «Deep Learning for Time Series Forecasting» N. D. Lewis (2018)
10. «Прогнозування: принципи та практика» Роб Хайндман, Джордж Атанасопулос.
11. Безпека й аварійне реагування на сонячних електростанціях Casey C. Grant, P.E. USA – 2013
12. Визначення категорій приміщень, будинків та зовнішніх установок за вибохопожежною та пожежною небезпекою. ДСТУ Б В.1.1-36:2016 Київ, 2016.
13. Небесна механіка Ю.В. Олександров – Харків «ХНУ», 2006
14. Електронний ресурс: https://otexts.com/fpp2/holt-winters.html
15. Електронний ресурс: https://www.analyticsvidhya.com/blog/2021/08/holt-winters-method-for-time-series-analysis/
16. «The Photoelectric Effect and Its Applications to Solar Cells» Krishiv Bhatia.
17. Електронний ресурс: https://www.sciencedirect.com/topics/physics-and-astronomy/photoelectric-effect

ДОДАТОК А

Метод Холта-Уінтерса Python

```
class HoltWinters:
    """
    Модель Хольта-Винтерса з методом Брутлага для виявлення аномалій

    Параметри:
    - series: вихідний часовий ряд
    - slen: довжина сезону
    - alpha, beta, gamma: коефіцієнти моделі Хольта-Винтерса
    - n_preds: горизонт прогнозування
    - scaling_factor: визначає ширину довірчого інтервалу за методом Брутлага
                      (зазвичай приймає значення від 2 до 3)
    """

    def __init__(self, series, slen, alpha, beta, gamma, n_preds,
scaling_factor):
        self.series = series
        self.slen = slen
        self.alpha = alpha
        self.beta = beta
        self.gamma = gamma
        self.n_preds = n_preds
        self.scaling_factor = scaling_factor

    def initial_trend(self):
        # Обчислення початкового тренду
        sum = 0.0
        for i in range(self.slen):
            sum += float(self.series[i + self.slen] - self.series[i]) /
self.slen
        return sum / self.slen

    def initial_seasonal_components(self):
        # Обчислення початкових сезонних компонентів
        seasonals = {}
        season_averages = []
        n_seasons = int(len(self.series) / self.slen)
        # Обчислення сезонних середніх
        for j in range(n_seasons):
            season_averages.append(sum(self.series[self.slen * j:self.slen * j +
self.slen]) / float(self.slen))
        # Обчислення початкових значень
        for i in range(self.slen):
            sum_of_vals_over_avg = 0.0
            for j in range(n_seasons):
                sum_of_vals_over_avg += self.series[self.slen * j + i] -
season_averages[j]
            seasonals[i] = sum_of_vals_over_avg / n_seasons
        return seasonals
```

```
    def triple_exponential_smoothing(self):
        # Ініціалізація списків для результатів та компонент
        self.result = []
        self.Smooth = []
        self.Season = []
        self.Trend = []
        self.PredictedDeviation = []
        self.UpperBond = []
        self.LowerBond = []

        seasonals = self.initial_seasonal_components()

        for i in range(len(self.series) + self.n_preds):
            if i == 0:
                # Ініціалізація значень компонент на першому кроці
                smooth = self.series[0]
                trend = self.initial_trend()
                self.result.append(self.series[0])
                self.Smooth.append(smooth)
                self.Trend.append(trend)
                self.Season.append(seasonals[i % self.slen])

                self.PredictedDeviation.append(0)

                self.UpperBond.append(self.result[0] +
                                      self.scaling_factor *
                                      self.PredictedDeviation[0])

                self.LowerBond.append(self.result[0] -
                                      self.scaling_factor *
                                      self.PredictedDeviation[0])

                continue
            if i >= len(self.series):
                # Прогнозування
                m = i - len(self.series) + 1
                self.result.append((smooth + m * trend) + seasonals[i %
self.slen])

                # Збільшення невизначеності з кожним кроком прогнозу
                self.PredictedDeviation.append(self.PredictedDeviation[-1] *
1.01)
            else:
                val = self.series[i]
                last_smooth, smooth = smooth, self.alpha * (val - seasonals[i %
self.slen]) + (1 - self.alpha) * (
                                  smooth + trend)
```

```
                trend = self.beta * (smooth - last_smooth) + (1 - self.beta) *
trend
                seasonals[i % self.slen] = self.gamma * (val - smooth) + (1 -
self.gamma) * seasonals[i % self.slen]
                self.result.append(smooth + trend + seasonals[i % self.slen])

                # Обчислення відхилення за алгоритмом Брутлага
                self.PredictedDeviation.append(self.gamma *
np.abs(self.series[i] - self.result[i])
                                               + (1 - self.gamma) *
self.PredictedDeviation[-1])

                self.UpperBond.append(self.result[-1] +
                                      self.scaling_factor *
                                      self.PredictedDeviation[-1])

            self.LowerBond.append(self.result[-1] -
                                      self.scaling_factor *
                                      self.PredictedDeviation[-1])

            self.Smooth.append(smooth)
            self.Trend.append(trend)
            self.Season.append(seasonals[i % self.slen])
```

ДОДАТОК Б

Програма розрахунку положення сонця *C#*

```
const double LONGITUDE = 35.062396061280175;
const double LATITUDE = 48.45534994945111;
double[,] sun_declination_table = {
    {-23.067, -22.983, -22.9, -22.8, -22.7, -22.6, -22.467, -22.35, -22.217, -
22.083, -21.933, -21.783, -21.617, -21.45, -21.267, -21.1, -20.9, -20.7, -20.5, -
20.18, -20.05, -19.867, -19.633, -19.4, -19.167, -18.917, -18.667, -18.417, -18.15,
-17.883, -17.617 },
    {-17.333, -17.05, -16.767, -16.467, -16.167, -15.867, -15.567, -15.25, -14.933,
-14.617, -14.3, -13.967, -13.633, -13.3, -12.967, -12.617, -12.267, -11.917, -
11.567, -11.217, -10.867, -10.5, -10.133, -9.767, -9.4, -9.033, -8.65, -8.283, -
8.05, double.NaN, double.NaN},
    {-7.817, -7.433, -7.05, -6.667, -6.283, -5.9, -5.5, -5.117, -4.733, -4.333, -
3.95, -3.55, -3.167, -2.767, -2.367, -1.983, -1.583, -1.183, -0.8, -0.4, 0.00, 0.4,
0.783, 1.183, 1.583, 1.967, 2.367, 2.75, 3.15, 3.533, 3.917 },
    {4.3, 4.07, 5.083, 5.467, 5.85, 6.217, 6.6, 6.983, 7.35, 7.717, 8.117, 8.467,
8.833, 9.183, 9.55, 9.9, 10.267, 10.616, 10.967, 11.317, 11.65, 12.00, 12.333,
12.667, 13.00, 13.317, 13.633, 13.967, 14.267, 14.583, double.NaN },
    {14.9, 15.2, 15.5, 15.783, 16.083, 16.367, 16.65, 16.917, 17.2, 17.45, 17.717,
17.983, 18.233, 18.483, 18.716, 18.967, 19.183, 19.417, 19.633, 19.85, 20.067,
20.267, 20.467, 20.65, 20.833, 21.017, 21.2, 21.367, 21.517, 21.683, 21.833 },
    {21.967, 22.1, 22.233, 22.367, 22.483, 22.583, 22.7, 22.783, 22.53, 22.883,
23.033, 23.117, 23.183, 23.233, 23.283, 23.333, 23.367, 23.4, 23.417, 23.433,
23.433, 23.433, 23.433, 23.417, 23.4, 23.383, 23.35, 23.317, 23.267, 23.217,
double.NaN },
    {23.15, 23.083, 23.017, 22.93, 22.85, 22.75, 22.65, 22.55, 22.433, 22.317,
22.183, 22.067, 21.917, 21.767, 21.617, 21.467, 21.3, 21.133, 20.97, 20.783, 20.6,
20.4, 20.2, 20.00, 19.783, 19.567, 19.35, 19.133, 18.9, 18.667, 18.417 },
    {18.167, 17.917, 17.667, 17.4, 17.133, 16.867, 16.6, 16.316, 16.033, 15.75,
15.45, 15.167, 14.867, 14.55, 14.25, 13.933, 13.617, 13.3, 12.983, 12.65, 12.317,
11.983, 11.65, 11.317, 10.967, 10.633, 10.283, 9.933, 9.583, 9.217, 8.867 },
    {8.5, 8.15, 7.783, 7.417, 7.05, 6.667, 6.3, 5.933, 5.55, 5.167, 4.8, 4.417,
4.033, 3.65, 3.267, 2.883, 2.5, 2.1, 1.717, 1.333 ,0.95, 0.55, 0.167 , -0.233, -
0.617, -1.00, -1.4, -1.783, -2.167, -2.567, double.NaN },
    {-2.95, -3.333, -3.733, -4.117, -4.5, -4.883, -5.267, -5.65, -6.033, -6.417, -
6.8, -7.167, -7.533, -7.917, -8.3, -8.667, -9.033, -9.4, -9.75, -10.117, -10.483, -
10.833, -11.2, -11.55, -11.9, -12.233, -12.583, -12.917, -13.25, -13.583, -13.917
},
    {-14.233, -14.567, -14.883, -15.017, -15.5, -15.8, -16.1, -16.4, -16.683, -
16.967, -17.083, -17.533, -17.8, -18.067, -18.333, -18.583, -18.833, -19.083, -
19.317, -19.55, -19.783, -20.00, -20.217, -20.433, -20.633, -20.833, -21.017, -
21.2, -21.383, -21.55, double.NaN },
    {-21.717, -21.867, -22.017, -22.167, -22.3, -22.417, -22.533, -22.65, -22.767,
-22.867, -22.95, -23.033, -23.117, -23.183, -23.233, -23.283, -23.333, -23.367, -
23.4, -23.417, -23.433, -23.433, -23.433, -23.433, -23.417, -23.383, -23.35, -
23.317, -23.267, -23.2, -23.133 }
};

double[] azimuth_temp = new double[24];
double[] high_solar_temp = new double[24];

double EOT_time, EOT_variable, true_solar_time, UTC_time, astronomical_time,
high_solar, azimuth_solar;

int day_year;

DateTime current_time = new DateTime(2023,5,31, 0,0, 0);

while (current_time.Year == 2023)
{
    for (int i = 0; i < 24; i++)
    {
```

```
        day_year = current_time.DayOfYear;
        ////////////////////////////////////////////
        UTC_time = (current_time.Hour * 60);
        UTC_time += (current_time.Minute);
        UTC_time += ((double)current_time.Second / 60);
        ////////////////////////////////////////////

        EOT_variable = ((double)(360 * (day_year - 81)) / 365); //44.38356;

        //Нахождение уравнение времени EOT.
        EOT_time = 9.87 * Math.Sin((2 * EOT_variable) * Math.PI / 180) - 7.53 *
Math.Cos(EOT_variable * Math.PI / 180) - 1.5 * Math.Sin(EOT_variable * Math.PI /
180); //Результат минуты.

        //Расщёт истинного астрономического времени.
        astronomical_time = UTC_time + (LONGITUDE * 4) + EOT_time;

        int hour = Convert.ToInt32(Math.Floor(astronomical_time / 60));
        int minute = Convert.ToInt32(Math.Floor(astronomical_time - (hour * 60)));
        int second = Convert.ToInt32(Math.Floor((astronomical_time - (hour * 60 +
minute)) * 60));

        //Расщёт истинного времение светила.
        true_solar_time = astronomical_time - (12 * 60);

        hour = Convert.ToInt32(Math.Floor(true_solar_time / 60));
        minute = Convert.ToInt32(Math.Floor(true_solar_time - (hour * 60)));
        second = Convert.ToInt32(Math.Floor((true_solar_time - (hour * 60 +
minute)) * 60));

        //Расщёт высоты солнца.
        high_solar = Math.Asin(Math.Sin(sun_declination_table[(current_time.Month -
1), (current_time.Day - 1)] * Math.PI / 180) * Math.Sin(LATITUDE * Math.PI / 180) +
Math.Cos(sun_declination_table[(current_time.Month - 1), (current_time.Day - 1)] *
Math.PI / 180) * Math.Cos(LATITUDE * Math.PI / 180) * Math.Cos((true_solar_time /
4) * Math.PI / 180));
        high_solar = high_solar * 180 / Math.PI;
        //Расщёт азимута солнца.
        azimuth_solar =
Math.Atan2(Math.Cos(sun_declination_table[(current_time.Month - 1),
(current_time.Day - 1)] * Math.PI / 180) * Math.Sin((true_solar_time / 4) * Math.PI
/ 180), Math.Cos(sun_declination_table[(current_time.Month - 1), (current_time.Day
- 1)] * Math.PI / 180) * Math.Sin(LATITUDE * Math.PI / 180) *
Math.Cos((true_solar_time / 4) * Math.PI / 180) -
Math.Sin(sun_declination_table[(current_time.Month - 1), (current_time.Day - 1)] *
Math.PI / 180) * Math.Cos(LATITUDE * Math.PI / 180));
        azimuth_solar = azimuth_solar * 180 / Math.PI;

        azimuth_temp[i] = azimuth_solar;
        high_solar_temp[i] = high_solar;

        current_time = current_time.AddMinutes(60);
        Console.WriteLine(current_time.ToString());
        File.AppendAllText("high_solar_data.txt", azimuth_solar.ToString() + "\n");
        File.AppendAllText("azimuth_solar_data.txt", high_solar.ToString() + "\n");

    }
    //File.AppendAllText("high_solar_data_MIN.txt",
high_solar_temp.Min().ToString() + "\n");
    //File.AppendAllText("high_solar_data_MAX.txt",
high_solar_temp.Max().ToString() + "\n");
}
```

ДОДАТОК В

Загальний блокнот Jupiter-Notebook дослідження аналітичних моделей Python.

```
# Імпорт бібліотек
import sys
import warnings
warnings.filterwarnings('ignore')
from tqdm import tqdm
import pandas as pd
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error
import statsmodels.formula.api as smf
import statsmodels.tsa.api as smt
import statsmodels.api as sm
import scipy.stats as scs
from scipy.optimize import minimize
from statsmodels.tsa.stattools import adfuller
import pmdarima as pm
import matplotlib.pyplot as plt
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
time_start_training = '2023-07-20T00:00:00'
time_end_training = '2023-07-27T23:59:00'
time_end_check = '2023-08-03T00:00:00'
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
data_cloud = pd.read_csv('C:/Users/llord/OneDrive/Рабочий стол\Diplom
Analitika/Cloud.csv', index_col=['Date'], parse_dates=['Date'])
data_cloud = data_cloud.asfreq('10T')
data_cloud = data_cloud.interpolate(method='akima')
data_cloud = data_cloud.astype(np.float16)
print(data_cloud.info())
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
data_cloud_for_training = data_cloud[time_start_training : time_end_training]
data_cloud_for_check = data_cloud[time_end_training : time_end_check]
data_cloud_mini = data_cloud['2023-07-31T00:00:00':'2023-08-06T00:00:00']

data_cloud_for_training = data_cloud_for_training.astype(np.float32)
data_cloud_for_check = data_cloud_for_check.astype(np.float32)
data_cloud_mini = data_cloud_mini.astype(np.float32)
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
dataset = pd.read_csv('C:/Users/llord/OneDrive/Рабочий стол\Diplom
Analitika/CombinedFile.csv', index_col=['Date'], parse_dates=['Date'])
# Видалити всі стовпці крім "Date" та "Power"
dataset = dataset.drop(columns=['Voltage', 'Current'])
dataset = dataset.resample('10T').mean().astype(np.float32)
#dataset = dataset.astype(np.float16)
# Розділ даних
dataset_for_training = dataset[time_start_training : time_end_training]
```

```
dataset_for_check = dataset[time_end_training : time_end_check]
dataset_mini = dataset['2023-07-31T00:00:00':'2023-08-06T00:00:00']

dataset.head()
```

//

```
plt.figure(figsize=(16, 8))
plt.plot(dataset_for_check, label='Дані у Момент Прогнозу', color='green')
plt.plot(dataset_for_training['2023-07-20T00:00:00':'2023-07-27T23:59:00'],
label='Дані Тренування', color='red')
plt.legend()
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.title('Тимчасовий ряд об\'єкту дослідження', fontsize=16)
plt.xlabel('Дата', fontsize=14)
plt.ylabel('Значення',fontsize=14)
plt.grid(True)

plt.show()
```

//

```
#data_cloud.to_csv('test.csv', index=False)


plt.figure(figsize=(22, 8))
# Построение графика для второго списка (красного цвета)
plt.plot(data_cloud_mini.direct_radiation / 10, label='Пряме випромінювання 1:10
Вт/м²', color='#FF4500')
plt.plot(data_cloud_mini.diffuse_radiation / 10, label='Дифузне випромінювання 1:10
Вт/м²', color='#DA70D6')
# Построение графика для первого списка (синего цвета)
plt.plot(data_cloud_mini.Cloud, label='Хмарність %', color='grey')

plt.plot(data_cloud_mini.shortwave_radiation / 10, label='Короткохвильове
випромінювання 1:10 Вт/м²', color='orange')

# Построение графика для первого списка (синего цвета)
plt.plot(dataset_mini, label='Потужність Вт', color='green')

# Добавление легенды
plt.legend()

# Добавление заголовка и подписей к осям
plt.title('Пошук залежності показників')
plt.xlabel('Дата')
plt.ylabel('Значення')
plt.grid(True)
# Отображение графика
plt.show()
```

//

```
from scipy.stats.stats import pearsonr
x = dataset_mini['Power'].values
y = data_cloud_mini['diffuse_radiation'].values

corr , p = pearsonr(x,y)

# cloudcover_low Corelation = -0.12783508807768468  cloudcover_mid Corelation = -
0.18148663967608775 cloudcover_high = -0.17687691027806088 shortwave_radiation =
0.8595054374981541 direct_radiation = 0.8542415806346382 diffuse_radiation =
0.7448852243010198
print ('Corelation Coefficient =', corr,'\nP-Value =',p)
```

//

```
#data_cloud.to_csv('test.csv', index=False)
data_cloud_for_training = data_cloud['2023-07-01T00:00:00':'2023-08-03T23:59:00']
data_cloud_for_check = data_cloud['2023-08-04T00:00:00':'2023-08-10T23:59:00']
data_cloud_mini = data_cloud['2023-07-25T00:00:00':'2023-08-03T23:59:00']

plt.figure(figsize=(22, 8))
# Построение графика для второго списка (красного цвета)
plt.plot(data_cloud_for_training['2023-08-01T00:00:00':].Cloud, label='Хмарність у
період навчання (Не повна)', color='black')

# Построение графика для первого списка (синего цвета)
plt.plot(data_cloud_for_check, label='Хмарність у період прогнозування',
color='grey')
plt.plot(data_cloud_for_check.cloudcover_low, label='Хмарність у період навчання
(Не повна)', color='yellow')

plt.plot(dataset_for_training['2023-08-01T00:00:00':], label='Потужність у період
навчання (Не повна)', color='green')

plt.plot(dataset_for_check, label='Потужність у період прогнозування',
color='green')

# Добавление легенды
plt.legend()

# Добавление заголовка и подписей к осям
plt.title('Хмарність')
plt.xlabel('Дата')
plt.ylabel('Значення %')
plt.grid(True)
# Отображение графика
plt.show()
```

//

```
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import plotly.graph_objs as go
init_notebook_mode(connected=True)
```

```
def plotly_df(df, title = ''):
    data = []

    for column in df.columns:
        trace = go.Scatter(
            x = df.index,
            y = df[column],
            mode = 'lines',
            name = column
        )
        data.append(trace)

    layout = dict(title = title)
    fig = dict(data = data, layout = layout)
    #plot(fig, show_link=False, filename='C:/Users/llord/OneDrive/Рабочий
стол/Diplom Analitika/Data_In.html')

plotly_df(dataset_for_training, title = "Напруга")
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
def plotMovingAverage(series, n):

    rolling_mean = series.rolling(window=n).mean()

    # При желании, можно строить и доверительные интервалы для сглаженных значений
    rolling_std =  series.rolling(window=n).std()
    upper_bond = rolling_mean+1.96*rolling_std
    lower_bond = rolling_mean-1.96*rolling_std

    plt.figure(figsize=(16,8))
    plt.title("Рухоме середнє\n розмір вікна = {}".format(n))


    plt.plot(upper_bond, "r--", label="Upper Bond / Lower Bond")
    plt.plot(lower_bond, "r--")
    plt.plot(series[n:], label="Фактичні значення")
    plt.plot(rolling_mean, "g", label="Рухливий середній тренд")
    plt.legend(loc="upper left")
    plt.grid(True)
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
plotMovingAverage(dataset_mini['2023-07-30T06:00:00':'2023-07-30T20:59:00'], 5)
plotMovingAverage(dataset_mini['2023-07-30T06:00:00':'2023-07-30T20:59:00'], 30)
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
def exponential_smoothing(series, alpha, n_preds):
    result = [series[0]]
    for n in range(1, len(series)):
        result.append(alpha * series[n] + (1 - alpha) * result[n-1])
```

```
    last_value = result[-1]

    for i in range(n_preds):
        next_value = alpha * last_value + (1 - alpha) * result[-1]
        result.append(next_value)
        last_value = next_value

    return result
```

//

```
with plt.style.context('seaborn-v0_8-deep'):
    plt.figure(figsize=(16, 8))
    plt.plot(dataset_mini.Power.values[200:1500], "c", label = "Дійсні значення")
    for alpha in [0.8, 0.05]:
        plt.plot(exponential_smoothing(dataset_mini.Power.values[200:1500], alpha,
n_preds = 0), label="Alpha {}".format(alpha))
    plt.legend(loc="best")
    plt.axis('tight')
    plt.title("Експоненціальне Згладжування")
    plt.grid(True)
```

//

```
def double_exponential_smoothing(series, alpha, beta, n_preds = 500):
    result = [series[0]]
    level, trend = series[0], series[1] - series[0]

    for n in range(1, len(series) + n_preds):
        if n >= len(series):
            value = result[-1]
        else:
            value = series[n]

        last_level, level = level, alpha * value + (1 - alpha) * (level + trend)
        trend = beta * (level - last_level) + (1 - beta) * trend
        result.append(level + trend)

    return result
```

//

```
with plt.style.context('seaborn-v0_8-deep'):
    plt.figure(figsize=(16, 8))
    plt.plot(dataset_mini.Power.values[200:1500], label = "Дійсні значення")
    alpha = 0.05
    for beta in [0.8, 0.05]:
            plt.plot(double_exponential_smoothing(dataset_mini.Power[200:1500],
alpha, beta), label="Alpha {}, Beta {}".format(alpha, beta))
    plt.legend(loc="best")
    plt.axis('tight')
    plt.grid(True)
```

//

#Вбудуйте додаток А.

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
from statsmodels.tsa.stattools import acf, pacf

# Проведемо тест ADF для визначення стаціонарності
result = sm.tsa.adfuller(dataset_for_training.Power)
# Обчисліть ACF та PACF для ряду
lags = 48  # Кількість лагів, для яких ви хочете обчислити ACF та PACF
acf_values = acf(dataset_for_training.Power, nlags=lags)
pacf_values = pacf(dataset_for_training.Power, nlags=lags)

# Отримайте значення ACF та PACF для аналізу
print("ACF Values:", acf_values)
print("PACF Values:", pacf_values)
# Виведемо результати тесту ADF
print("ADF Statistic:", result[0])
print("p-value:", result[1])
print("Critical Values:", result[4])

# Побудуємо графіки ACF і PACF
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(16, 8))
sm.graphics.tsa.plot_acf(dataset_for_training.Power, lags=48, ax=ax1)
sm.graphics.tsa.plot_pacf(dataset_for_training.Power, lags=48, ax=ax2)
plt.show()
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
%%time
# Создание экземпляра класса с необходимыми параметрами
model = HoltWinters(series=dataset_for_training.Power, slen=144,
alpha=0.0014981751671583332, beta=0.021980265844491387, gamma=0.9, n_preds=144*6,
scaling_factor = 1.96) #1502

# Вызов метода для выполнения тройного экспоненциального сглаживания и получения
прогнозов
model.triple_exponential_smoothing()

# Получение результатов прогноза и других компонент модели
results = model.result
smoothing = model.Smooth
trend = model.Trend
seasonal = model.Season
upper_bond = model.UpperBond
lower_bond = model.LowerBond

print(results)
print("Длина списка results:", len(results))

print("Длина списка dataset_mini.Power:", len(dataset_mini.Power))
```

```
start_date = '2023-07-01T00:00:00'

date_range = pd.date_range(start=time_start_training, periods=len(results),
freq='10T')

Holt_Winters_Forecast = pd.DataFrame({'Date': date_range, 'Power': results})

Holt_Winters_Forecast.set_index('Date', inplace=True)

Holt_Winters_Forecast = Holt_Winters_Forecast['2023-07-28T00:00:00':]
Holt_Winters_Forecast['Power'] = Holt_Winters_Forecast['Power'].apply(lambda x:
max(0, x)).round(5)

print(Holt_Winters_Forecast)
```

//

```
dataset_for_training_end = dataset_for_training['2023-07-25T00:00:00':]

plt.figure(figsize=(16, 8))
# Построение графика для второго списка (красного цвета)
plt.plot(dataset_for_training_end, label='Дані навчання', color='black')

# Построение графика для первого списка (синего цвета)
plt.plot(dataset_for_check, label='Дані для перевірки', color='grey')

# Построение графика для первого списка (синего цвета)
plt.plot(Holt_Winters_Forecast, label='Прогноз', color='green')


# Добавление легенды
plt.legend()

# Добавление заголовка и подписей к осям
plt.title('Результати Прогнозування')
plt.xlabel('Дата')
plt.ylabel('Значення')
plt.grid(True)
# Отображение графика
plt.show()
```

//

```
def adjust_list_length(lst, numb):
    # Проверяем, делится ли длина списка на numb
    if len(lst) % numb != 0:
        # Если не делится нацело, вычисляем количество лишних строк
        extra_rows = len(lst) % numb
        # Удаляем лишние строки с конца списка
        lst = lst[:-extra_rows]
    return lst
```

//

```
data = dataset_for_training # отложим часть данных для тестирования
data = adjust_list_length(data, 5)
print(len(data))
```

//

```
from sklearn.model_selection import TimeSeriesSplit

def timeseriesCVscore(x):
    # вектор ошибок
    errors = []

    values = data.Power
    alpha, beta, gamma = x

    # задаём число фолдов для кросс-валидации
    tscv = TimeSeriesSplit(n_splits=10)

    # идем по фолдам
    for train, test in tscv.split(values):

        model = HoltWinters(series=values[train], slen = 144, alpha=alpha,
beta=beta, gamma=gamma, n_preds=len(test), scaling_factor = 1.96)
        model.triple_exponential_smoothing()

        predictions = model.result[-len(test):]
        actual = values[test]
        error = mean_squared_error(predictions, actual)
        errors.append(error)

    # Возвращаем средний квадрат ошибки по вектору ошибок
    return np.mean(np.array(errors))
```

//

```
%%time
# инициализируем значения параметров
x = [0, 0, 0]

# Минимизируем функцию потерь с ограничениями на параметры
opt = minimize(timeseriesCVscore, x0=x, method="TNC", bounds = ((0, 1), (0, 1), (0,
1)))

#dataset_mini
# Из оптимизатора берем оптимальное значение параметров
alpha_final, beta_final, gamma_final = opt.x
print(alpha_final, beta_final, gamma_final)
```

//

```
def tsplot(y, lags=None, figsize=(22, 8), style='bmh'):
    if not isinstance(y, pd.Series):
        y = pd.Series(y)
```

```
    with plt.style.context(style):
        fig = plt.figure(figsize=figsize)
        layout = (2, 2)
        ts_ax = plt.subplot2grid(layout, (0, 0), colspan=2)
        acf_ax = plt.subplot2grid(layout, (1, 0))
        pacf_ax = plt.subplot2grid(layout, (1, 1))

        y.plot(ax=ts_ax)
        ts_ax.set_title('Time Series Analysis Plots')
        smt.graphics.plot_acf(y, lags=lags, ax=acf_ax, alpha=0.5)
        smt.graphics.plot_pacf(y, lags=lags, ax=pacf_ax, alpha=0.5)

        print("Критерий Дики-Фуллера: p=%f" % sm.tsa.stattools.adfuller(y)[1])

        plt.tight_layout()
    return

tsplot(dataset_mini.Power, lags=24)
```

//

```
def invboxcox(y,lmbda):
    if lmbda == 0:
        return(np.exp(y))
    else:
        return(np.exp(np.log(lmbda*y+1)/lmbda))

data = dataset_mini.copy()
data['Power'], lmbda = scs.boxcox(data.Power+1)
tsplot(data.Power, lags=24)
print("Оптимальний параметр перетворення Боксу-Коксу: %f" % lmbda)
```

//

```
#Standard ARIMA Model
ARIMA_model = pm.auto_arima(data['Power'],
                      start_p=1,
                      start_q=1,
                      test='adf', # use adftest to find optimal 'd'
                      max_p=3, max_q=8, # maximum p and q
                      m=1, # frequency of series (if m==1, seasonal is set to FALSE
automatically)
                      d=None,# let model determine 'd'
                      seasonal=False, # No Seasonality for standard ARIMA
                      trace=True, #logs
                      error_action='warn', #shows errors ('ignore' silences these)
                      suppress_warnings=True,
                      stepwise=True)
```

//

```
best_p = ARIMA_model.get_params()['order'][0]
best_d = ARIMA_model.get_params()['order'][1]
```

```
best_q = ARIMA_model.get_params()['order'][2]
model_summary = ARIMA_model.summary()

print(model_summary)
print(f"Best ARIMA parameters: p={best_p}, d={best_d}, q={best_q}")
```

//

```
ARIMA_model.plot_diagnostics(figsize=(22,8))
plt.show()
```

//

```
mport statsmodels.api as sm

p = 1  # Параметр p (AR)
d = 0  # Параметр d (порядок интегрирования)
q = 1  # Параметр q (MA)
P = 1  # Сезонный параметр P (SAR)
D = 1 # Сезонный параметр D (SAR, порядок интегрирования)
Q = 0  # Сезонный параметр Q (SAR, MA)
s = 144  # Период сезонности (например, 12 для ежемесячных данных)

# объект SARIMAX с указанными параметрами
SARIMA_model = sm.tsa.SARIMAX(dataset_for_training, order=(p, d, q),
seasonal_order=(P, D, Q, s))
```

//

```
%%time
results = SARIMA_model.fit()
```

//

```
%%time
SARIMA_predictions = results.get_prediction(start=pd.to_datetime('2023-07-
28T00:00:00'), end=pd.to_datetime('2023-08-03T00:00:00'), dynamic=False,
full_results=True)
predictions_ci = SARIMA_predictions.conf_int()
```

//

```
ax = dataset_for_check['2023-07-28T00:00:00':'2023-08-
03T00:00:00'].Power.plot(label='Реальні значення', figsize=(16, 8))
#dataset_for_training['2023-08-04T00:00:00':'2023-08-
05T00:00:00'].Power.plot(ax=ax, label='Вилучений день')
SARIMA_predictions.predicted_mean.plot(ax=ax, label='Прогноз SARIMA')
#Holt_Winters_Forecast.Power.plot(ax=ax, label='Прогноз Холта Вінтерса')

#ax.fill_between(predictions_ci.index, predictions_ci.iloc[:, 0],
predictions_ci.iloc[:, 1], color='pink', alpha=0.1)
plt.xlabel('Дата')
plt.ylabel('Значення')

plt.legend()
```

```
plt.grid(True)
plt.show()
```

//

```
p = 1  # Параметр p (AR)
d = 0  # Параметр d (порядок интегрирования)
q = 1  # Параметр q (MA)
P = 1  # Сезонный параметр P (SAR)
D = 1 # Сезонный параметр D (SAR, порядок интегрирования)
Q = 0  # Сезонный параметр Q (SAR, MA)
s = 144  # Период сезонности (например, 12 для ежемесячных данных)

# Создайте объект SARIMAX с указанными параметрами
SARIMAX_model = sm.tsa.SARIMAX(dataset_for_training, order=(p, d, q),
seasonal_order=(P, D, Q, s), exog = data_cloud_for_training.shortwave_radiation)
```

//

```
%%time
SARIMAX_results = SARIMAX_model.fit()
```

//

```
%%time
SARIMAX_predictions =
SARIMAX_results.get_prediction(start=pd.to_datetime(time_end_training),
end=pd.to_datetime(time_end_check), dynamic=False, full_results=True, exog =
data_cloud_for_check.shortwave_radiation[time_end_training: time_end_check])

SARIMAX_predictions.predicted_mean =
SARIMAX_predictions.predicted_mean.apply(lambda x: max(0, x)).round(5)
```

//

```
plt.figure(figsize=(16, 8))

plt.plot(dataset_for_check.Power, label='Реальні значення', color='#FF4500')
plt.plot(SARIMAX_predictions.predicted_mean, label='Прогноз SARIMAX')
plt.plot(data_cloud_for_check["2023-08-06T00:00:00":"2023-08-
09T00:00:00"].shortwave_radiation / 15 , label='Екзогенний параметр 1:15 Вт/м²')
#plt.plot(SARIMA_predictions.predicted_mean, label='Прогноз SARIMA')

plt.xlabel('Дата')
plt.ylabel('Значення')

plt.legend()
plt.grid(True)
plt.show()
```

//

```
predicted_values = Holt_Winters_Forecast.Power.values
actual_values = dataset_for_check['2023-07-28T00:00:00':'2023-08-
02T23:59:00'].Power.values
```

```
# Рассчитайте среднюю абсолютную ошибку (MAE)
mae = np.mean(np.abs(actual_values - predicted_values))

print(f"Середня Абсолютна Помилка (MAE): {mae:.2f}")

max_value = np.max(actual_values)
error = mae * 100 / max_value
print(f"Похибка: {error:.2f} %")
```

//

```
predicted_values = SARIMA_predictions.predicted_mean.values
actual_values = dataset_for_check['2023-07-28T00:00:00':'2023-08-
03T23:59:00'].Power.values

# Рассчитайте среднюю абсолютную ошибку (MAE)
mae = np.mean(np.abs(actual_values - predicted_values))

print(f"Середня Абсолютна Помилка (MAE): {mae:.2f}")
max_value = np.max(actual_values)
error = mae * 100 / max_value
print(f"Похибка: {error:.2f} %")
```

//

```
predicted_values = SARIMAX_predictions.predicted_mean.values
actual_values = dataset_for_check[time_end_training:time_end_check].Power.values

# Рассчитайте среднюю абсолютную ошибку (MAE)
mae = np.mean(np.abs(actual_values - predicted_values))

print(f"Середня Абсолютна Помилка (MAE): {mae:.2f}")
max_value = np.max(actual_values)
error = mae * 100 / max_value
print(f"Похибка: {error:.2f} %")
```

ДОДАТОК Г

Загальний блокнот Jupiter-Notebook дослідження нейромережі V2 Python.

```
import sys
import warnings
warnings.filterwarnings('ignore')
from tqdm import tqdm

import pandas as pd
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error

import statsmodels.formula.api as smf
import statsmodels.tsa.api as smt
import statsmodels.api as sm
import scipy.stats as scs
from scipy.optimize import minimize
from statsmodels.tsa.stattools import adfuller
import matplotlib.pyplot as plt

import torch
import torch.nn as nn
import torch.optim as optim
```

//

```
data_for_RNN = pd.read_csv('D:\Универ\Диплом Магистра\Diplom
Analitika\Data_for_RNN.csv', index_col=['Date'], parse_dates=['Date'])
data_for_RNN.head()
```

//

```
time_start_training = '2023-07-20T00:00:00'
time_end_training = '2023-07-27T23:59:00'
time_end_check = '2023-08-03T00:00:00'
```

//

```
data_for_RNN = data_for_RNN.asfreq('60T')
data_for_RNN = data_for_RNN.interpolate(method='akima')

data_for_RNN_for_training = data_for_RNN['2023-07-01T00:00:00': time_end_training]
data_for_RNN_for_check = data_for_RNN[time_end_training : time_end_check]

data_for_RNN_for_training = data_for_RNN_for_training.astype(np.float16)
data_for_RNN_for_check = data_for_RNN_for_check.astype(np.float16)

print ('data_cloud_for_training')
print(data_for_RNN_for_training.info())
print ('data_cloud_for_check')
print(data_for_RNN_for_check.info())
```

//

```
dataset = pd.read_csv('D:\Универ\Диплом Магистра\Diplom
Analitika\CombinedFile.csv', index_col=['Date'], parse_dates=['Date'])

dataset = dataset.resample('60T').mean().astype(np.float16)

# Розділ даних
dataset_for_training = dataset['2023-07-01T00:00:00': time_end_training]
dataset_for_check = dataset[time_end_training : time_end_check]

print ('dataset_for_training')
print(dataset_for_training.info())
print ('dataset_for_check')
print(dataset_for_check.info())
```

//

```
#data_cloud.to_csv('test.csv', index=False)
time_start ='2023-08-02T00:00:00'

plt.figure(figsize=(16, 8))

# Построение графика для первого списка (синего цвета)
plt.plot(data_for_RNN_for_training.Sun_azimuth[time_start:'2023-08-03T23:59:00'],
label='Азимут сонця (0)=>(180)=>(-180)=>(0)', color='#4B0082')

plt.plot(data_for_RNN_for_training.Sun_height[time_start:'2023-08-03T23:59:00'],
label='Висота сонця', color='#DA10D6')

# Построение графика для первого списка (синего цвета)
plt.plot(dataset_for_training.Power[time_start:'2023-08-03T23:59:00'],
label='Потужність Вт', color='green')

# Добавление легенды
plt.legend()

# Добавление заголовка и подписей к осям
plt.title('Пошук залежності положення сонця')
plt.xlabel('Дата')
plt.ylabel('Значення')
plt.grid(True)
# Отображение графика
plt.show()
```

//

```
#data_cloud.to_csv('test.csv', index=False)
time_start ='2023-08-02T00:00:00'

plt.figure(figsize=(16, 8))
# Построение графика для второго списка (красного цвета)
plt.plot(data_for_RNN_for_training.direct_radiation[time_start:'2023-08-
03T23:59:00'] / 10, label='Пряме випромінювання 1:10 Вт/м²', color='#FF4500')
```

```
plt.plot(data_for_RNN_for_training.diffuse_radiation[time_start:'2023-08-
03T23:59:00'] / 10, label='Дифузне випромінювання 1:10 Вт/м²', color='#DA70D6')
# Построение графика для первого списка (синего цвета)
plt.plot(data_for_RNN_for_training.direct_normal_irradiance[time_start:'2023-08-
03T23:59:00'] / 10, label='Пряме нормальне опромінення 1:10 Вт/м²', color='grey')

plt.plot(data_for_RNN_for_training.shortwave_radiation[time_start:'2023-08-
03T23:59:00'] / 10, label='Короткохвильове випромінювання 1:10 Вт/м²',
color='orange')

# Построение графика для первого списка (синего цвета)
plt.plot(data_for_RNN_for_training.terrestrial_radiation[time_start:'2023-08-
03T23:59:00'] / 10, label='Земне випромінювання 1:10 Вт/м²', color='#4B0082')

#plt.plot(data_for_RNN_for_training.Sun_height[time_start:'2023-08-03T23:59:00'],
label='Висота сонця', color='#DA10D6')

# Построение графика для первого списка (синего цвета)
plt.plot(dataset_for_training.Power[time_start:'2023-08-03T23:59:00'],
label='Потужність Вт', color='green')

# Добавление легенды
plt.legend()

# Добавление заголовка и подписей к осям
plt.title('Пошук залежності параметрів сонця')
plt.xlabel('Дата')
plt.ylabel('Значення')
plt.grid(True)
# Отображение графика
plt.show()
```

//

```
#data_cloud.to_csv('test.csv', index=False)
time_start ='2023-07-25T00:00:00'

plt.figure(figsize=(16, 8))
# Построение графика для второго списка (красного цвета)
plt.plot(data_for_RNN_for_training.temperature[time_start:'2023-08-03T23:59:00'],
label='Температура навколишнього середовища | C', color='#FF4500')
plt.plot(data_for_RNN_for_training.surface_pressure[time_start:'2023-08-
03T23:59:00'] - 1000, label='Поверхневий тиск | (-1000)', color='#DA70D6')
# Построение графика для первого списка (синего цвета)
plt.plot(data_for_RNN_for_training.relativehumidity[time_start:'2023-08-
03T23:59:00'] / 10, label='Відносна вологість | 1:10', color='grey')

plt.plot(data_for_RNN_for_training.precipitation[time_start:'2023-08-03T23:59:00']
/ 10, label='Опади | мм', color='orange')

# Построение графика для первого списка (синего цвета)
```

```
plt.plot(data_for_RNN_for_training.cloudcover_high[time_start:'2023-08-
03T23:59:00'] / 10, label='Високі хмари | >6км | 1:10 | %', color='#4B0082')

plt.plot(data_for_RNN_for_training.cloudcover_mid[time_start:'2023-08-03T23:59:00']
/ 10, label='Середні хмари | 3-6км | 1:10 | %', color='#DA10D6')
plt.plot(data_for_RNN_for_training.cloudcover_low[time_start:'2023-08-03T23:59:00']
/ 10, label='Низькі хмари | <3км | 1:10 | %', color='#808000')

plt.plot(data_for_RNN_for_training.vapor_pressure_deficit[time_start:'2023-08-
03T23:59:00'], label='Дефіцит тиску пари', color='blue')
# Построение графика для первого списка (синего цвета)
plt.plot(dataset_for_training.Power[time_start:'2023-08-03T23:59:00'],
label='Потужність Вт', color='green')

# Добавление легенды
plt.legend()

# Добавление заголовка и подписей к осям
plt.title('Пошук залежності від погодних умов')
plt.xlabel('Дата')
plt.ylabel('Значення')
plt.grid(True)
# Отображение графика
plt.show()
```

//

```
Sun_azimuth_tensor = torch.tensor(data_for_RNN_for_training['Sun_azimuth'].values,
dtype=torch.float32)
Sun_height_tensor = torch.tensor(data_for_RNN_for_training['Sun_height'].values,
dtype=torch.float32)
Cloud_tensor = torch.tensor(data_for_RNN_for_training['Cloud'].values,
dtype=torch.float32)
cloudcover_low_tensor =
torch.tensor(data_for_RNN_for_training['cloudcover_low'].values,
dtype=torch.float32)
cloudcover_mid_tensor =
torch.tensor(data_for_RNN_for_training['cloudcover_mid'].values,
dtype=torch.float32)
cloudcover_high_tensor =
torch.tensor(data_for_RNN_for_training['cloudcover_high'].values,
dtype=torch.float32)
shortwave_radiation_tensor =
torch.tensor(data_for_RNN_for_training['shortwave_radiation'].values,
dtype=torch.float32)
direct_radiation_tensor =
torch.tensor(data_for_RNN_for_training['direct_radiation'].values,
dtype=torch.float32)
diffuse_radiation_tensor =
torch.tensor(data_for_RNN_for_training['diffuse_radiation'].values,
dtype=torch.float32)
```

```
temperature_tensor = torch.tensor(data_for_RNN_for_training['temperature'].values,
dtype=torch.float32)
relativehumidity_tensor =
torch.tensor(data_for_RNN_for_training['relativehumidity'].values,
dtype=torch.float32)
precipitation_tensor =
torch.tensor(data_for_RNN_for_training['precipitation'].values,
dtype=torch.float32)
rain_tensor = torch.tensor(data_for_RNN_for_training['rain'].values,
dtype=torch.float32)
snowfall_tensor = torch.tensor(data_for_RNN_for_training['snowfall'].values,
dtype=torch.float32)
weathercode_WMO_tensor =
torch.tensor(data_for_RNN_for_training['weathercode_WMO'].values,
dtype=torch.float32)
pressure_msl_tensor =
torch.tensor(data_for_RNN_for_training['pressure_msl'].values, dtype=torch.float32)
surface_pressure_tensor =
torch.tensor(data_for_RNN_for_training['surface_pressure'].values,
dtype=torch.float32)
direct_normal_irradiance_tensor =
torch.tensor(data_for_RNN_for_training['direct_normal_irradiance'].values,
dtype=torch.float32)
terrestrial_radiation_tensor =
torch.tensor(data_for_RNN_for_training['terrestrial_radiation'].values,
dtype=torch.float32)

power_tensor = torch.tensor(dataset_for_training['Power'].values,
dtype=torch.float32)

print ('Sun_azimuth ', Sun_azimuth_tensor)
print ('Sun_height ', Sun_height_tensor)
print ('shortwave_radiation ', shortwave_radiation_tensor)
print ('direct_radiation ', direct_radiation_tensor)
print ('diffuse_radiation ', diffuse_radiation_tensor)

print ('Power ', power_tensor)
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

batch_size = 648
# Изменяем форму наших данных перед объединением
Sun_azimuth_tensor = Sun_azimuth_tensor.view(batch_size, -1)
Sun_height_tensor = Sun_height_tensor.view(batch_size, -1)
Cloud_tensor = Cloud_tensor.view(batch_size, -1)
cloudcover_low_tensor = cloudcover_low_tensor.view(batch_size, -1)
cloudcover_mid_tensor = cloudcover_mid_tensor.view(batch_size, -1)
cloudcover_high_tensor = cloudcover_high_tensor.view(batch_size, -1)
shortwave_radiation_tensor = shortwave_radiation_tensor.view(batch_size, -1)
direct_radiation_tensor = direct_radiation_tensor.view(batch_size, -1)
diffuse_radiation_tensor = diffuse_radiation_tensor.view(batch_size, -1)
temperature_tensor = temperature_tensor.view(batch_size, -1)
```

```
relativehumidity_tensor = relativehumidity_tensor.view(batch_size, -1)
rain_tensor = rain_tensor.view(batch_size, -1)
snowfall_tensor = snowfall_tensor.view(batch_size, -1)
weathercode_WMO_tensor = weathercode_WMO_tensor.view(batch_size, -1)
pressure_msl_tensor = pressure_msl_tensor.view(batch_size, -1)
surface_pressure_tensor = surface_pressure_tensor.view(batch_size, -1)
direct_normal_irradiance_tensor = direct_normal_irradiance_tensor.view(batch_size,
-1)
terrestrial_radiation_tensor = terrestrial_radiation_tensor.view(batch_size, -1)

power_tensor = power_tensor.view(batch_size, -1)
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
combined_input = torch.cat([Sun_azimuth_tensor.unsqueeze(2),
Sun_height_tensor.unsqueeze(2), Cloud_tensor.unsqueeze(2),
cloudcover_low_tensor.unsqueeze(2), cloudcover_mid_tensor.unsqueeze(2),
cloudcover_high_tensor.unsqueeze(2),shortwave_radiation_tensor.unsqueeze(2),direct_
radiation_tensor.unsqueeze(2),diffuse_radiation_tensor.unsqueeze(2),temperature_ten
sor.unsqueeze(2),relativehumidity_tensor.unsqueeze(2),rain_tensor.unsqueeze(2),snow
fall_tensor.unsqueeze(2),weathercode_WMO_tensor.unsqueeze(2),pressure_msl_tensor.un
squeeze(2),surface_pressure_tensor.unsqueeze(2),direct_normal_irradiance_tensor.uns
queeze(2),terrestrial_radiation_tensor.unsqueeze(2)], dim=2)
combined_input
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
class PowerPredictionRNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(PowerPredictionRNN, self).__init__()

        # RNN слой
        self.rnn = nn.RNN(input_size, hidden_size, batch_first=True)

        # Выходной линейный слой
        self.linear = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        # Пропустите входные данные через RNN
        out, _ = self.rnn(x)

        # Получите выходные данные с последнего временного шага и примените
линейный слой
        out = self.linear(out[:, -1, :])
        return out
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
class PowerPredictionAttentionLSTM(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, dropout_prob,
num_heads):
        super(PowerPredictionAttentionLSTM, self).__init__()
```

```
        # LSTM слой
        self.lstm = nn.LSTM(input_size, hidden_size, batch_first=True,
bidirectional=True)

        # Многоголовый механизм внимания
        self.attention = nn.MultiheadAttention(hidden_size * 2, num_heads)

        # Слой Dropout
        self.dropout = nn.Dropout(p=dropout_prob)

        # Выходной линейный слой
        self.linear = nn.Linear(hidden_size * 2, output_size)

    def forward(self, x):
        # Пропустите входные данные через BiLSTM
        out, _ = self.lstm(x)

        # Примените Dropout
        out = self.dropout(out)

        # Передайте выход LSTM через многоголовый механизм внимания
        out, _ = self.attention(out, out, out)

        # Получите выходные данные с последнего временного шага и примените
линейный слой
        out = self.linear(out[:, -1, :])
        return out
```

//

```
class PowerPredictionLSTM(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, dropout_prob):
        super(PowerPredictionLSTM, self).__init__()

        # LSTM слой
        self.lstm = nn.LSTM(input_size, hidden_size, batch_first=True)

        # Слой Dropout
        self.dropout = nn.Dropout(p=dropout_prob)

        # Выходной линейный слой
        self.linear = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        # Пропустите входные данные через LSTM
        out, _ = self.lstm(x)

        # Примените Dropout
        out = self.dropout(out)

        # Получите выходные данные с последнего временного шага и примените
линейный слой
```

```
        out = self.linear(out[:, -1, :])
        return out
```

//

```
class PowerPredictionComplexLSTM(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, num_layers):
        super(PowerPredictionComplexLSTM, self).__init__()

        # LSTM слои
        self.lstm1 = nn.LSTM(input_size, hidden_size, batch_first=True,
num_layers=num_layers)
        self.lstm2 = nn.LSTM(hidden_size, hidden_size, batch_first=True,
num_layers=num_layers)


        # Выходные линейные слои
        self.linear1 = nn.Linear(hidden_size, hidden_size)
        self.linear2 = nn.Linear(hidden_size, hidden_size)

    def forward(self, x):
        # Пропустите входные данные через первый LSTM слой
        out, _ = self.lstm1(x)

        # Пропустите данные через второй LSTM слой
        out, _ = self.lstm2(out)


        # Получите выходные данные с последнего временного шага и примените
линейные слои
        out = self.linear1(out[:, -1, :])
        out = self.linear2(out)

        return out
```

//

```
input_size = 18  # Входные признаки: напряжение и ток
hidden_size_1 = 1024  # Размер скрытого состояния RNN
hidden_size_2 = 256  # Размер скрытого состояния RNN
hidden_size_3 = 64  # Размер скрытого состояния RNN
output_size = 1  # Выход: прогноз мощности
dropout_prob = 0.3
num_heads = 1

# Создайте экземпляр модели
model = PowerPredictionRNN(input_size, hidden_size_1, output_size)
#model = PowerPredictionAttentionLSTM(input_size, hidden_size_1, output_size,
dropout_prob, num_heads)
#model = PowerPredictionLSTM(input_size, hidden_size_1, output_size, dropout_prob)
#model = PowerPredictionComplexLSTM(input_size, hidden_size_1, output_size,
num_layers = 2)
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Переносим модель на GPU (если доступен CUDA)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
combined_input = combined_input.to(device)
power_tensor = power_tensor.to(device)
model.to(device)

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

criterion = nn.MSELoss()  # Среднеквадратичная ошибка (MSE) как функция потерь
optimizer = torch.optim.Adam(model.parameters(), lr=0.0001)  # Оптимизатор Adam

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

%%time
num_epochs = 4000  # Количество эпох обучения

for epoch in range(num_epochs):
    optimizer.zero_grad()  # Обнулие градиентов
    outputs = model(combined_input)  # Передай данные через модель, не забуть
исправить ошибку валидации
    loss = criterion(outputs, power_tensor)  # Вычисляем функцию потерь
    loss.backward()  # Рассчитай градиенты
    optimizer.step()  # Обновляем параметры модели

    print(f'Епоха [{epoch+1}/{num_epochs}], Втрата: {loss.item():.4f}')

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

%%time
last_Sun_azimuth = data_for_RNN_for_check.Sun_azimuth
last_Sun_height = data_for_RNN_for_check.Sun_height
last_Cloud = data_for_RNN_for_check.Cloud
last_cloudcover_low = data_for_RNN_for_check.cloudcover_low
last_cloudcover_mid = data_for_RNN_for_check.cloudcover_mid
last_cloudcover_high = data_for_RNN_for_check.cloudcover_high
last_shortwave_radiation = data_for_RNN_for_check.shortwave_radiation
last_direct_radiation = data_for_RNN_for_check.direct_radiation
last_diffuse_radiation = data_for_RNN_for_check.diffuse_radiation

last_temperature = data_for_RNN_for_check.temperature
last_relativehumidity = data_for_RNN_for_check.relativehumidity
last_rain = data_for_RNN_for_check.rain
last_snowfall = data_for_RNN_for_check.snowfall
last_weathercode_WMO = data_for_RNN_for_check.weathercode_WMO
last_pressure_msl = data_for_RNN_for_check.pressure_msl
last_surface_pressure = data_for_RNN_for_check.surface_pressure
last_direct_normal_irradiance = data_for_RNN_for_check.direct_normal_irradiance
last_terrestrial_radiation = data_for_RNN_for_check.terrestrial_radiation

# Преобразуйте их в тензоры PyTorch
last_Sun_azimuth_tensor = torch.tensor(last_Sun_azimuth.values,
dtype=torch.float32)
```

```
last_Sun_height_tensor = torch.tensor(last_Sun_height.values, dtype=torch.float32)
last_Cloud_tensor = torch.tensor(last_Cloud.values, dtype=torch.float32)
last_cloudcover_low_tensor = torch.tensor(last_cloudcover_low.values,
dtype=torch.float32)
last_cloudcover_mid_tensor = torch.tensor(last_cloudcover_mid.values,
dtype=torch.float32)
last_cloudcover_high_tensor = torch.tensor(last_cloudcover_high.values,
dtype=torch.float32)
last_shortwave_radiation_tensor = torch.tensor(last_shortwave_radiation.values,
dtype=torch.float32)
last_direct_radiation_tensor = torch.tensor(last_direct_radiation.values,
dtype=torch.float32)
last_diffuse_radiation_tensor = torch.tensor(last_diffuse_radiation.values,
dtype=torch.float32)

last_temperature_tensor = torch.tensor(last_temperature.values,
dtype=torch.float32)
last_relativehumidity_tensor = torch.tensor(last_relativehumidity.values,
dtype=torch.float32)
last_rain_tensor = torch.tensor(last_rain.values, dtype=torch.float32)
last_snowfall_tensor = torch.tensor(last_snowfall.values, dtype=torch.float32)
last_weathercode_WMO_tensor = torch.tensor(last_weathercode_WMO.values,
dtype=torch.float32)
last_pressure_msl_tensor = torch.tensor(last_pressure_msl.values,
dtype=torch.float32)
last_surface_pressure_tensor = torch.tensor(last_surface_pressure.values,
dtype=torch.float32)
last_direct_normal_irradiance_tensor =
torch.tensor(last_direct_normal_irradiance.values, dtype=torch.float32)
last_terrestrial_radiation_tensor = torch.tensor(last_terrestrial_radiation.values,
dtype=torch.float32)

input_data = torch.stack([last_Sun_azimuth_tensor, last_Sun_height_tensor,
last_Cloud_tensor, last_cloudcover_low_tensor, last_cloudcover_mid_tensor,
last_cloudcover_high_tensor, last_shortwave_radiation_tensor,
last_direct_radiation_tensor
,last_diffuse_radiation_tensor,last_temperature_tensor,
last_relativehumidity_tensor,
last_rain_tensor,last_snowfall_tensor,last_weathercode_WMO_tensor,last_pressure_msl
_tensor,last_surface_pressure_tensor,last_direct_normal_irradiance_tensor,last_terr
estrial_radiation_tensor], dim=1)  # Форма [interval_length, input_size]
input_data = input_data.to(device)

# Используем модель для предсказания мощности на основе входных данных
predicted_power = model(input_data.unsqueeze(1))  # Размер батча 1
predicted_power
```

//

```
# Получаем прогноз мощности
predicted_power_values = predicted_power.tolist()
predicted_power_values
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

start_date = time_end_training

# Создаём диапазон дат с заданным периодом
date_range = pd.date_range(start=start_date, periods=len(predicted_power_values),
freq='60T')

dataset_power_values = pd.DataFrame({'Date': date_range, 'Power':
predicted_power_values})

dataset_power_values['Power'] = dataset_power_values['Power'].apply(lambda x: x[0])
#dataset_power_values['Power'] = dataset_power_values['Power'].apply(lambda x:
x[0])

dataset_power_values.set_index('Date', inplace=True)
dataset_power_values['Power'] = dataset_power_values['Power'].apply(lambda x: 0 if
x < 0 else x)
dataset_power_values
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

from matplotlib.font_manager import FontProperties

with plt.style.context('seaborn-v0_8-deep'):
    plt.figure(figsize=(16, 8))
    # Створення текстового об'єкта з вказаним розміром шрифту
    font = FontProperties()
    font.set_size(14)  # Задаємо розмір шрифту
    plt.plot(dataset_for_check.Power, label='Реальна потужність на наступний
день',color='green')
    #plt.plot(dataset_for_check.Voltage['2023-08-06T00:00:00':'2023-08-
09T00:00:00'], label='Реальна напруга на наступний день',color='orange')
    #plt.plot(dataset_for_check.Current['2023-08-06T00:00:00':'2023-08-
09T00:00:00'], label='Реальний струм на наступний день',color='red')
    plt.plot(dataset_power_values.Power, marker='o', markersize=2, label=f'Прогноз
biLSTM | 2-параметри | {num_epochs}-епох | 1024-прихов. шарів')
    plt.xlabel('Дата')
    plt.ylabel('Потужність')
    plt.grid(True)
    plt.legend(prop=font)
    plt.show()
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

predicted_values = dataset_power_values[time_end_training :
time_end_check].Power.values
actual_values = dataset_for_check[time_end_training : time_end_check].Power.values

# Рассчитайте среднюю абсолютную ошибку (MAE)
mae = np.mean(np.abs(actual_values - predicted_values))

print(f"Середня Абсолютна Помилка (MAE): {mae:.2f}")
```

```
max_value = np.max(actual_values)
error = mae * 100 / max_value
print(f"Похибка: {error:.2f} %")
```

//

```
from sklearn.metrics import mean_squared_error
import math

# Замените эти два датафрейма вашими данными
predicted_values = dataset_power_values['2023-08-06T00:00:00':'2023-08-
09T00:00:00'].Power.values
actual_values = dataset_for_check["2023-08-06T00:00:00":"2023-08-
09T00:10:00"].Power.values

# Рассчитайте RMSE
rmse = math.sqrt(mean_squared_error(actual_values, predicted_values))

print(f"Корінь середньоквадратичної похибки (RMSE): {rmse:.2f}")
```

ВІДГУК КОНСУЛЬТАНТІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

ВІДГУК

на тему «Дослідження та розробка кіберфізичної системи прогнозування потужності фотоелектричної установки», виконаної студентом групи 151м-22-1 Рульовим Є.І.

Керівник кваліфікаційної роботи

(посада, вчене звання, ступінь) (підпис) (ініціали, прізвище)

РЕЦЕНЗІЯ

на тему «Дослідження та розробка кіберфізичної системи прогнозування потужності фотоелектричної установки», виконаної студентом групи 151м-22-1 Рульовим Є.І.

Завдання і зміст кваліфікаційної роботи ступеню магістра відповідає основній меті – перевірці знань та ступеню підготовки здобувача вищої освіти за спеціальністю “151 Автоматизація та комп’ютерно-інтегровані технології”. Оформлення пояснювальної записки та графічних матеріалів кваліфікаційної роботи виконано відповідно до вимог стандартів та методичних рекомендацій повністю.

Актуальність проведення дослідження та розробки кіберфізичної системи обумовлена потребою вдосконалення сучасних методів виробництва та управління енергетичними ресурсами. Реалізація цього проекту має великий потенціал у підвищенні ефективності використання сонячної енергії. При успішній імплементації кіберфізичної системи, можливо досягти оптимального рівня виробництва, що призведе до підвищення конкурентоспроможності та стійкості енергетичної інфраструктури.

Повнота та глибина вирішення поставлених завдань в кваліфікаційній роботі висока.

В рамках кваліфікаційної роботи виконано аналіз часових рядів, проведений якісний опис сучасних технологій, виконані експерименти з моделями прогнозування часових рядів, перевірка ресурсоємності розроблених моделей, отримані як підтвердження так й спростування певних гіпотез, які були висунуті у кваліфікаційній роботі.

В цілому кваліфікаційна робота ступеню магістра заслуговує оцінки “___” балів при відповідному захисті, а здобувач Рульов Є.І. присвоєння кваліфікації “магістр” за спеціальністю “151 Автоматизація та комп’ютерно-інтегровані технології”.

Рецензент, _____

___.12.2023