НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

“До захисту допущено”
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

“___”_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
“Цифрові технології в енергетиці”
зі спеціальності *122 “Комп’ютерні науки”*

на тему: **Прогнозування споживання електроенергії підприємства засобами машинного навчання**

Виконав:
студент IV курсу, групи ТМ-01

АКІШОВ Андрій Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник: *доцент каф. цифрових технологій в енергетиці*
доцент, к.військ.н. Андрій ОНИСЬКО _____
(посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент: _____
_____
(посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ) (підпис)

Нормоконтролер: *доцент Андрій ДОНЕЦЬ* _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____
(підпис)

Київ – 2024

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

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

«__»_____2024 р.

ЗАВДАННЯ
на дипломну роботу студенту

АКІШОВУ Андрію Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи **Прогнозування споживання електроенергії підприємства засобами машинного навчання**

керівник роботи ***Онисько Андрій Ілліч, к. військ. н., доцент***
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 202__ р. №_____

2. Термін подання роботи студентом **07.06.2024 року**

3. Вихідні дані до роботи мова програмування Python, бібліотеки pandas, numpy, sklearn, matplotlib, customtkinter, редактор коду Jupyter Notebook

4. Перелік питань, які потрібно розробити _____
1) провести аналіз серед наявних рішень прогнозування енергоспоживання
2) пошук та вибір методів прогнозування на основі машинного навчання

3) аргументація вибору методів прогнозування енергоспоживання
4) реалізація прогнозування енергоспоживання на основі обраних методів
5) створення інтерфейсу програмного забезпечення
6) представлення процесу застосування програмного забезпечення

5. Орієнтований перелік ілюстративного матеріалу формули, що обраховують оцінку ефективності роботи алгоритмів; рисунки, що показують залежність даних, демонструють інтерфейс програмного забезпечення, структури моделей.

6. Дата видачі завдання «15» вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі	17-20.04.24	
3.	Розробка архітектури та загальної структури системи	21-25.04.24	
4.	Розробка частин окремих підсистем	25.04-02.05.24	
5.	Програмна реалізація системи	03-07.05.24	
6.	Оформлення пояснювальної записки	08-12.05.24	
7.	Захист програмного продукту	13-17.05.24	
8.	Передзахист	03-06.06.24	
9.	Подання готової роботи на кафедру	07.06.2024	
10.	Захист	17-21.06.2024	

Студент _____ **Андрій АКІШОВ**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ **Андрій ОНИСЬКО**
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота виконана на 53 сторінках, містить 26 ілюстрацій, 11 таблиць, 1 додаток, 19 джерел в переліку посилань.

Мета роботи – розробка програмного забезпечення для прогнозування використання енергоспоживання на підприємстві за допомогою алгоритмів машинного навчання.

Методи та засоби: методи аналізу та прогнозування енергоспоживання, а саме регресійні моделі машинного навчання та метрики для оцінювання результатів їх роботи.

Результат – програмне забезпечення для прогнозування використання енергоспоживання на підприємстві.

Ключові слова: МАШИННЕ НАВЧАННЯ, РЕГРЕСІЙНІ МОДЕЛІ, МЕТРИКИ, ДЕРЕВА ПРИЙНЯТТЯ РІШЕНЬ, ГРАДІЄНТНИЙ СПУСК, ПРОГНОЗУВАННЯ, МЕТОДИ.

ABSTRACT

The thesis consists of 53 pages, includes 26 illustrations, 11 tables, 1 appendix, and references 19 sources.

The aim of the work is to develop software for predicting energy consumption at an enterprise using machine learning algorithms.

Methods and Tools: methods of analysis and prediction of energy consumption, specifically regression models of machine learning and metrics for evaluating their performance.

Result: software for predicting energy consumption at an enterprise.

Keywords: MACHINE LEARNING, REGRESSION MODELS, METRICS, DECISION TREES, GRADIENT DESCENT, FORECASTING, METHODS.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

AANN – адаптивні штучні нейронні мережі

ANN – штучні нейронні мережі

ARIMA – авторегресійне інтегроване ковзне середнє

DT – дерево прийняття рішень

EEDM – ансамблеве емпіричне розкладання мод

GA – генетичний алгоритм

KNN – k найближчих сусідів

MAE – середнє абсолютне відхилення

MAPE – середнє абсолютне відсоткове відхилення

MSE – середньоквадратичне відхилення

NN – нейронні мережі

RF – випадковий ліс

RMSE – кореневе середньоквадратичне відхилення

PL – поліномне ядро

SVR – регресія на основі опорних векторів

ПЗ – програмне забезпечення

ВСТУП

Сучасна промисловість має ключовий вплив на економіку та загальний розвиток будь-якої країни. Від потужностей та ефективності роботи підприємства залежить виробнича спроможність. На це впливає багато факторів, в тому числі й енергоспоживання. Витрати різняться від галузі до галузі, але незмінним залишається бажання якомога ефективнішого використання ресурсів.

Оптимізація енергоспоживання на підприємстві дозволяє заощаджувати та інвестувати збережені ресурси в інші ділянки виробництва. Оскільки енергія використовується не лише для освітлення та роботи обладнання, а й також конвертується у теплову енергію та паливо, її застосування набуває значно ширших масштабів, ніж може здатись на перший погляд. Зважаючи на кількість залучених до виробництва ресурсів та видів енергії, важливим постає контроль викидів, що генерує промисловість.

Прогнозування є потужним інструментом у задачах оптимізації, адже воно дає змогу на основі існуючих реальних даних оцінювати майбутні витрати та побудувати якісніші виробничі процеси, що призведе до значних заощаджень. Це дозволить підприємству підвищити енергоефективність, аналізувати та приймати довготривалі стратегічні рішення, а також краще контролювати негативний вплив на навколишнє середовище шляхом зменшення шкідливих викидів [1].

Актуальність розробки методів оптимізації і прогнозування енергоспоживання з кожним роком стає все більш суттєвою, оскільки шалені темпи індустріалізації генерують підприємства різних масштабів. Особливої уваги цей напрям заслуговує саме сьогодні, в період занепаду українського енергетичного сектору.

Відповідно до Концепції впровадження “Розумних мереж” в Україні до 2035 року було розроблено стратегію із модернізації та розвитку національної електроенергетики [2]. Передумовою розробки цієї Концепції стала потреба у поліпшенні енергоефективності, вирівнювання добового графіку електричного

навантаження енергосистеми та зменшення неефективного використання генеруючих потужностей в період максимальних електричних навантажень енергосистеми.

Як вказано у документі, на сьогодні кожне підприємство може здійснювати заходи на свій розсуд, не зважаючи на відсутність великого досвіду, іноді застосовуючи неефективні рішення та методи. Як наслідок, у результаті відсутності єдиної політики у цій галузі, виникають ризики впровадження та використання неефективних та несистемних заходів. Сукупність всіх цих факторів призводить до надлишкового використання коштів, залучених у формуванні тарифів.

Абстрагуючись від енергетичних питань України, світова спільнота також занепокоїна хаотичним, несистемним використанням скінченних ресурсів. На сьогодні, існує досить багато досліджень та публікацій, які фокусуються на темі енергоспоживання. Більшість із матеріалів пропонують рішення щодо оптимізації використання енергії, застосовуючи найрізноманітніші методи. У цій роботі буде розглянуто декілька потужних досліджень, що оглядають предметну область споживання енергоресурсів та пропонують ефективні алгоритми їх заощадження.

Метою даної роботи є розробка програмного забезпечення для прогнозування використання енергоспоживання на підприємстві за допомогою алгоритмів машинного навчання.

Головним завдання роботи є розробка якісного та зручного інструменту для якомога точнішого прогнозування енергоспоживання за заданими даними користувача. Для цього необхідно провести ґрунтовний аналіз існуючих рішень і розробити рішення, яке буде ідеально виконувати задану задачу.

1 РОЗРОБКА ПРОГНОЗУВАННЯ СПОЖИВАННЯ ЕЛЕКТРОЕНЕРГІЇ НА ПІДПРИЄМСТВІ ЗАСОБАМИ МАШИННОГО НАВЧАННЯ

Розробка програмного забезпечення для прогнозування енергоспоживання є комплексною задачею, що складається із багатьох етапів. Важливо зазначити всі ключові кроки у розробці, а також визначити задачі, які потрібно вирішити для розробки якісного програмного забезпечення. Аби краще зрозуміти предметну область, варто оглянути існуючі рішення, оцінити їх актуальність у контексті поставленої задачі, а також зважати на новизну цих методів.

1.1 Постановка задачі

Енергетичні ресурси мають велике значення у життєздатності будь-якого підприємства. Вміння якісно та ефективно застосовувати наявні ресурси може розкрити потенціал виробництва. Існують і зворотні ситуації, коли наявні всі необхідні засоби та матеріали для виробництва, але невміле, неякісне застосування може занапастити роботу підприємства.

Дана робота пропонує вирішення проблеми неефективного використання енергетичних ресурсів, за допомогою алгоритмів машинного навчання. Машинне навчання є частиною напрямку штучного інтелекту та комп'ютерних наук, що дозволяє машині "навчатись" на попередньому досвіді та приймати власні рішення на основі даних. У контексті задачі прогнозування енергоспоживання алгоритми машинного навчання можуть аналізувати історичні дані та знаходити складні патерни використання енергії. Особливо корисним машинне навчання є тоді, коли зв'язки між даними є нелінійними або їх важко моделювати. Також існують задачі, у яких кількість незалежних змінних досягає таких значень, коли обробка людиною

стає неможливою, навіть за умови використання сучасних інструментів обробки та моделювання даних.

Для розробки програмного забезпечення прогнозування енергоспоживання на підприємстві потрібно визначити завдання, що будуть виконані та висунути вимоги до програмного забезпечення, що розробляється.

1.1.1 Завдання до розв'язання

При розробці програмного забезпечення потрібно розв'язати такі завдання:

- дослідити та порівняти існуючі методології прогнозування енергоспоживання;
- обрати та провести навчання, тестування обраних моделей машинного навчання;
- проаналізувати результати прогнозування та обґрунтувати оптимальну модель для виконання задачі;
- розробити зручний та інтуїтивно зрозумілий інтерфейс для роботи із користувачем;
- провести тестування фінального програмного забезпечення та оцінити його якість розробленого.

1.1.2 Вимоги до програмного забезпечення

Програмне забезпечення повинно відповідати наступним вимогам:

- Точність прогнозування. Прогнозування енергоспоживання є головним об'єктом роботи, тому його точність має ключову цінність даної розробки.
- Швидкодія. ПЗ повинно швидко надавати результати прогнозування, забезпечуючи зручність обробки даних без значних часових витрат.
- Зручний інтерфейс. ПЗ повинно мати зручний та інтуїтивно зрозумілий інтерфейс, який максимально спрощує процес прогнозування даних.
- Багаторазове використання. ПЗ повинно забезпечити можливість багаторазового прогнозування для зручної роботи користувача з різними наборами даних, без потреби перезапуску програму.

1.2 Огляд методів та існуючих рішень прогнозування енергоспоживання

Промисловість відіграє величезну роль у процвітанні країн [3]. Деякі держави забезпечують свій економічний стан виключно за допомогою виробки різного роду продуктів, тому ця сфера людської діяльності приковує до себе стільки уваги. Зацікавленість в оптимізації витрат та максимізації прибутку проявляється не лише на рівні окремих підприємств, а й може бути стратегічним напрямом всієї країни.

У промисловій сфері було проведено немалу кількість досліджень, які присвячені прогнозуванню енергоспоживання. Ці роботи охоплюють як класичні методи передбачення часових рядів, так і новітні методології, які базуються на сучасних технологіях. Застосування цих методів варіюється від житлових до промислових масштабів.

1.2.1 Гібридна модель ARIMA та GA-SVR на основі EEMD розкладу

У 2020 році, Kao Yu-Sheng, Kazumitsu Nawata та Chi-Yo Huang, запропонували гібридну модель для прогнозування енергоспоживання [4]. Емпірично було доведено ефективність запропонованого методу на основі реального споживання енергії у Тайвані.

За результатами дослідження було розроблено фреймворк, який використовував ансамблевий підхід до застосування моделей та методів попередньої обробки даних. До розробленої гібридної моделі входив ансамбль емпіричного розкладання мод (EEMD), авторегресійне інтегроване ковзне середнє (ARIMA), метод опорних векторів (SVR) та генетичний алгоритм (GA).

Тестування моделі проводилось на різних комбінаціях набору технологій, а для оцінки результатів були використані наступні метрики: MAE, MAPE (представлена у відсотках, %.), MSE, RMSE. Результати порівняльного аналізу представлено у таблиці 1.1, що демонструє ефективність запропонованих рішень прогнозування.

Таблиця 1.1 - Результати порівняння комбінацій моделей

Моделі	Тренувальна вибірка				Тестова вибірка			
	MAE	MAPE	MSE	RMSE	MAE	MAPE	MSE	RMSE
ARIMA	1.397	3.704	5.669	2.381	5.305	4.782	32.301	5.683
ARIMA-SVR	1.285	3.467	4.535	2.130	3.877	3.491	18.851	4.342
ARIMA-GA-SVR	1.257	3.408	4.246	2.061	3.320	2.988	14.723	3.837
EEDM-ARIMA	0.499	1.429	0.611	0.781	2.266	2.048	6.856	2.618
EEDM-ARIMA-SVR	0.425	1.351	0.388	0.623	1.547	1.396	4.139	2.034
EEDM-ARIMA-GA-SVR	0.413	1.308	0.381	0.617	1.492	1.346	3.711	1.926

Порівняльний аналіз різних комбінацій застосування методів встановив, що найбільш якісною є композиція EEDM-ARIMA-GA-SVR, оскільки вона дозволяє досягти найменшого відхилення прогнозованого енергоспоживання від фактичного. Це відображають як показники метрик у таблиці 1.1, так і графік на рисунку 1.1.

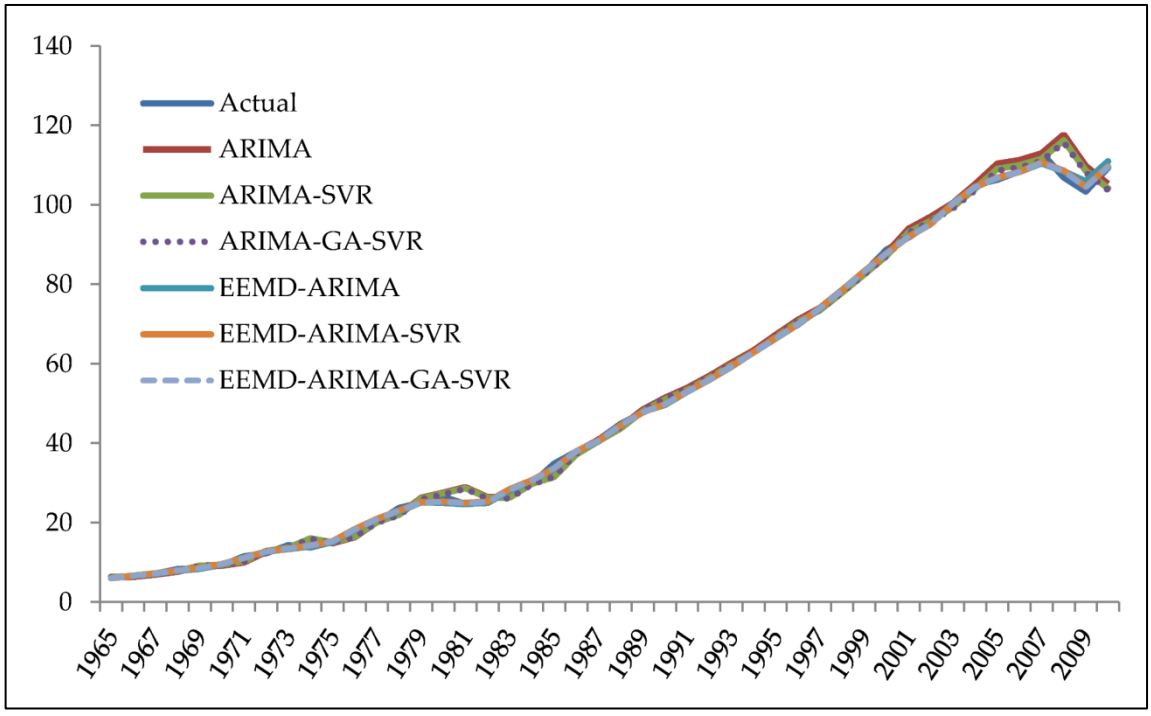


Рисунок 1.1 – Порівняльний графік точності прогнозування моделей

1.2.2 Адитивна модель штучного інтелекту прогнозування енергетичних часових рядів

Ngoc-Son Truong, Ngoc-Tri Ngo та Anh-Duc Pham запропонували прогнозування часових рядів енергетичних даних у будівлях з використанням адитивної моделі штучного інтелекту для підвищення енергоефективності [5]. У даному дослідженні зосереджуються на енергоефективності житлових будинків та розробки інструменту менеджменту, що дозволяє якісно аналізувати та оптимізувати енергоспоживання у будівлях. Ця робота детально описує актуальність теми енергетичної ефективності. Глобальна урбанізація, збільшення населення та загальний розвиток людства мають сильний вплив на екологію, який проявляється через викиди вуглецю, утворення парникового ефекту та глобальне потепління. Помилково вважати, що метою роботи є лише оптимізація використання та контроль енергоспоживання, оскільки автори також зазначили, що розроблене рішення повинно забезпечувати комфортне проживання населення, при цьому споживаючи менше енергії.

Для оцінки моделей використовувався набір даних, який був зібраний з житлового будинку з сонячною фотовольтаїчною системою. Також на вхід моделям подавались дані про погодні умови та інсоляцію, які також впливали на енергоспоживання. Інсоляційні дані — це вимірювання рівня сонячної радіації, що потрапляють на певну площу земної поверхні. У контексті прогнозування енергоспоживання вони можуть використовуватись для розрахунків кліматичних умов.

Дане дослідження фокусується на використанні адаптивних штучних нейронних мереж (AANN), що є покращеною версією штучних нейронних мереж (ANN), оскільки містить градієнтне підсилення, а також підбирає залишки штучних нейронних мереж (ANN) , що були залишені на попередньому етапі роботи. Окрім штучних нейронних мереж (ANN) та адаптивних штучних нейронних мереж (AANN) також були запропоновані інші методи машинного навчання аби порівняти ефективність, такі як лінійна регресія, регресія опорних векторів з

поліномним ядром (SVR-PL) та регресія опорних векторів з радіальною функцією базису (SVR-RBF).

У результаті порівняння обраних моделей, що демонструються у таблиці 1.2, найкращими виявились адаптивні штучні нейронні мережі (AANN), що забезпечують найбільш точне прогнозування на тестовій вибірці. Варто відзначити модель ANN, яка мала досить близькі до AANN результати.

Таблиця 1.2 - Результати порівняння моделей

Моделі	Тренувальна вибірка (90%)				Тестова вибірка (10%)			
	MAPE	MAE	RMSE	R	MAPE	MAE	RMSE	R
SVR-PL	26.18	240.69	438.45	0.653	28.60	236.83	430.69	0.622
SVR-RBF	24.13	226.58	412.98	0.698	26.38	225.06	412.53	0.659
Linear Regression	30.74	265.67	409.80	0.586	36.74	275.68	416.64	0.654
M5Rules	11.96	104.26	172.77	0.953	14.20	112.44	213.43	0.921
ANNs	12.86	106.80	170.28	0.955	14.68	114.91	199.64	0.932
AANNs	12.78	106.88	165.15	0.961	14.04	111.98	188.68	0.940

На рисунках 1.2 та 1.3 зображена якість передбачення для моделей регресії опорних векторів з поліномним ядром (SVR-PL) та адаптивних штучних нейронних мереж (AANN). Діагональна лінія вказує на абсолютну відповідність передбаченого та фактичного енергоспоживання.

Як можна бачити з наведених рисунків 1.2 та 1.3, модель регресії опорних векторів з поліномним ядром (SVR-PL) має багато значних відхилень, що вказує на досить посереднє прогнозування енерговитрат. Натомість адаптивні штучні нейронні мережі (AANN) мають значно менше відхилень від діагональної лінії та кращу кучність точок, що відображають високу точність у передбаченнях, які підтверджують метрики.

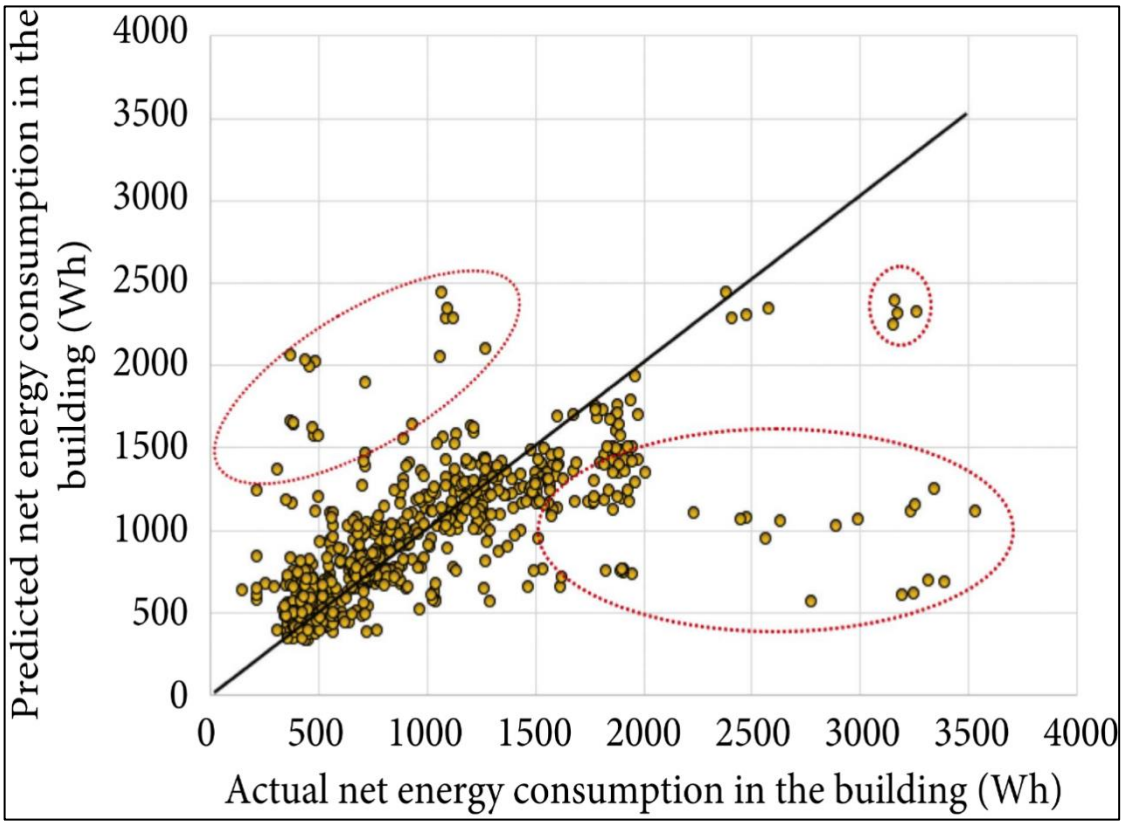


Рисунок 1.2 – Візуальне представлення точності прогнозування моделі SVR-PL

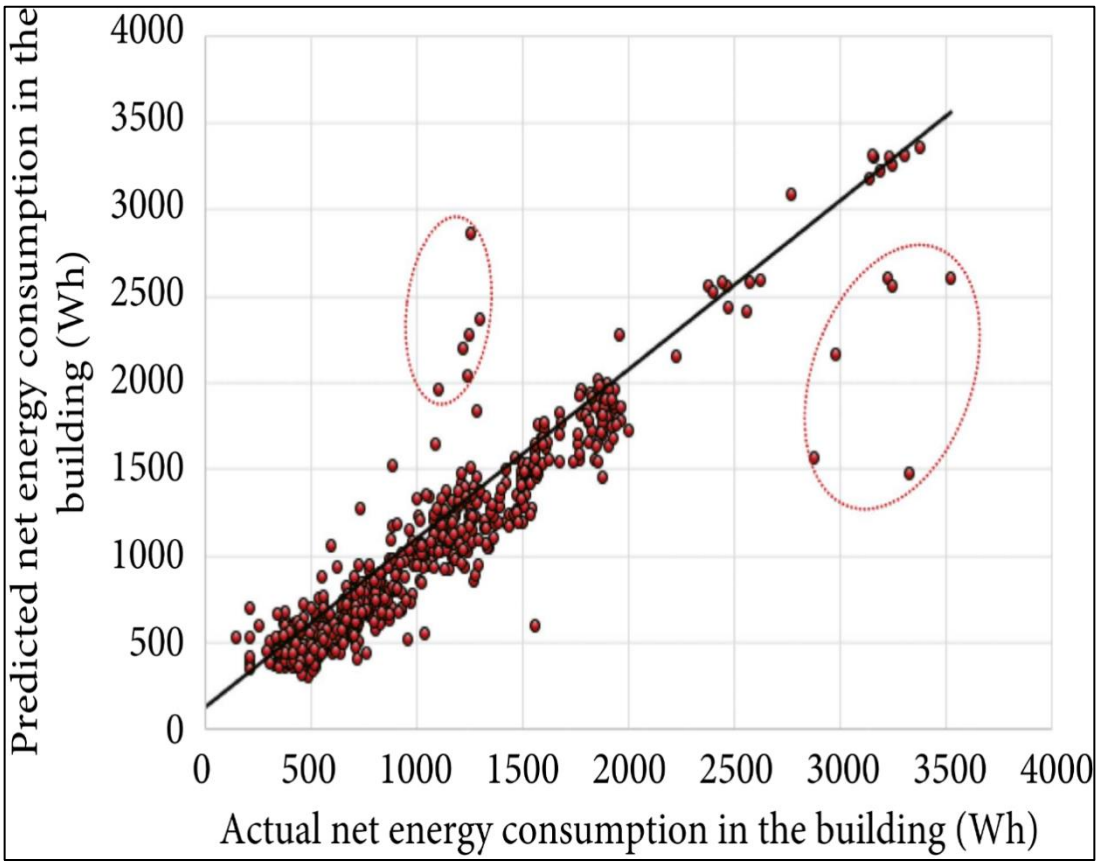


Рисунок 1.3 – Візуальне представлення точності прогнозування моделі AANN

1.3 Огляд програмних засобів

Вибір інструментів для реалізації програмного забезпечення є не менш важливою частиною роботи як вибір методів розв'язання задачі. Оскільки універсальної мови програмування не існує (як і будь-якого іншого сучасного інструменту для вирішення різноманітних задач), варто обирати технології розробки, зважаючи на предмету область та особливості конкретного завдання, що необхідно виконати. Важливо, аби програмні засоби могли якомога полегшити розробку, зробити процес виконання завдання комфортним. Неправильний перелік інструментів може призвести до ресурсних витрат на побічні від завдання виклики, які необхідно вирішити для коректного виконання основної задачі.

У даному розділі буде детально обґрунтовано вибір певного переліку технологій, які будуть використані для задачі прогнозування енергоспоживання.

1.3.1 Мова програмування

Для реалізації програмного забезпечення було обрано мову програмування Python, оскільки вона має широкий інструментарій для роботи із машинним навчанням. Python має високу популярність у сфері наукових досліджень, саме завдяки свій великій екосистемі, яка нараховує безліч готових рішень та реалізацій задач різного роду. Ця мова програмування застосовується як для написання веб-застосунків, так і для створення моделей штучного інтелекту.

Немало важливим є спільнота мови Python, яка, напевно, є найбільшою серед всіх існуючих мов програмування, що дозволяє переймати досвід багатьох спеціалістів у потрібній галузі.

Самої лише мови програмування Python недостатньо для виконання поставленого завдання, адже її справжній потенціал розкривається при роботі із широким списком бібліотек для виконання найрізноманітніших задач. Бібліотеками називають додаткове програмне забезпечення, що містить у собі нові функції та рішення, які вже були заздалегідь реалізованими іншими розробниками

для оптимізації робочого процесу. Надалі, всі бібліотеки, що будуть вказані, належать до екосистеми Python.

1.3.2 Бібліотеки для обробки даних

Обробка даних є важливим аспектом кожної роботи, що фокусується на вирішенні математичних, наукових, та суміжних до цих галузей, задач. Без інструментів взаємодії із даними, важко буде ними маніпулювати та виконувати найпростіші математичні дії. Дана робота не є виключенням, оскільки на основі оброблених та підготовлених вхідних даних відбувається моделювання енергоспоживання для вирішення задачі прогнозування.

У якості інструментів для обробки, очистки та маніпулювання даними було обрано дві найпопулярніші у цій сфері бібліотеки: NumPy та Pandas.

Pandas — це популярна бібліотека мови програмування Python, яка створена для маніпуляції з даними та їх аналізу. Основною її перевагою є зручний функціонал для роботи з табличним типом даних, що має назву DataFrame. Можна привести аналогію з Excel, але Pandas є більш гнучким та зручним інструментом, хоча і потребує більшої кількості попередніх знань порівняно з Excel. Фільтрація, нормалізація, агрегація, очистка, маніпуляція та перетворення — це лише мала частина можливостей Pandas, які гарантовано використовують у роботі із об’ємними даними.

Numpy (Numerical Python) — це бібліотека, основне завдання якої стосується роботи з одновимірними та багатовимірними масивами даних. Дана бібліотека також містить велику кількість математичних функцій для виконання різноманітних операцій. Ще однією перевагою цього інструменту є його швидкість. Це пов’язано із особливостями використання пам’яті пристрою, де відбувається виконання функцій Numpy.

Як результат, бібліотека Numpy є незамінним інструментом при будь-якому аналізі даних, оскільки вона швидка, легка у використанні, має зручний тип даних, який не конфліктує з іншими функціями та бібліотеками мови програмування Python.

1.3.3 Бібліотеки для візуалізації даних

Самі лише цифри мало що можуть пояснити у процесі дослідження. Яким би глибоким та широким не був набір даних, не завжди можна отримати користь з самих лише обрахунків та маніпуляцій із даними. Більш наглядним та інформативним є візуальне представлення інформації, оскільки воно забезпечує структурність та наочність результатів. Більше того, інколи, ніякі методи та моделі не зможуть надати відповідей на поставлені запитання, як це зробить графік. Це універсальний інструмент як в області аналізу даних, так і у візуалізації результатів, що дозволяє їх краще та легше інтерпретувати.

Для візуалізації залежностей між змінними та побудови графіків різного типу була обрана бібліотека matplotlib.

Matplotlib — найбільш популярна бібліотека для візуалізації даних. За допомогою цієї бібліотеки можна будувати графіки різних типів та форм. Простота використання, у поєднанні із можливістю детальної кастомізації всіх частин графіків, робить цей інструмент ідеальним для досліджень, які потребують графічного та візуального опису.

1.3.4 Бібліотеки для машинного навчання

Методи та моделі, що будуть використовуватись для прогнозування енергоспоживання відносять до сфери машинного навчання. Оскільки Python є найпопулярнішим інструментом у контексті розв’язання задач прогнозування, класифікації та кластеризації, він має багато корисних реалізованих методів та моделей машинного навчання. Було обрано декілька бібліотек, що пропонують широкий спектр вбудованих рішень для задач різного типу.

Scikit-learn (Sklearn) — це бібліотека для машинного навчання, яка містить чималий перелік моделей та алгоритмів для задач класифікації, прогнозування, регресії, кластеризації і багато іншого. Є чудовою відправною точкою у вивченні напряму машинного навчання, оскільки бібліотека чудово інтегрується з іншими популярними бібліотеками, а також є досить простою у використанні, що дозволить новачкам без зайвого клопоту працювати з нею.

Також sklearn містить інші корисні функції для користувача, наприклад, розбиття вибірок на тренувальну та тестову, метрики оцінки результатів роботи, кодування вхідних даних у інші формати, налаштування параметрів моделей та вибір важливих ознак для оцінки важливості вхідних даних.

Хоч бібліотека Scikit-learn пропонує досить широкий список реалізованих моделей машинного навчання, але все ж деякі корисні для цього дослідження моделі мають власні бібліотеки, для яких необхідний окремий імпорт та ініціалізація. Додатково, буде використано моделі із бібліотек xgboost, lightgbm та catboost. Використання цих бібліотек зумовлено необхідністю у тестуванні якомога більшої кількості методів, які потенційно можуть дати найкращі результати для задачі прогнозування енергоспоживання. Тому, ці бібліотеки будуть використані лише у дослідженні та пошуку найкращого методу. У фінальному програмному забезпечення залишиться лише 1 метод, який буде найкращим.

1.3.5 Бібліотеки для створення інтерфейсу

Окрім розробки програмного забезпечення, що прогнозує енергоспоживання, потрібно також розробити зручний інтерфейс для користувача. Це забезпечить зручний спосіб тестування та використання програмного забезпечення без зайвого клопоту. Розробка зручного інтерфейсу для користувача не лише сприяє зручності тестування та використання програмного забезпечення, але й підвищує продуктивність користувача, оскільки спрощує взаємодію з програмою та полегшує навігацію у функціоналі.

Для реалізації інтерфейсу програмного забезпечення було обрано CustomTkinter, покращену версію бібліотеки Tkinter, яка дозволяє створювати графічні інтерфейси для програм. Tkinter є простою у застосуванні та допомагає відносно легко розробити інтерфейс будь-якої складності для програм на Python. CustomTkinter значно розширює функціонал класичного Tkinter та надає більше можливостей для кастомізації інтерфейсу. До того ж, деякі графічні рішення у Tkinter є застарілими і виглядають значно гірше за аналогічні функції CustomTkinter.

1.4 Висновки до розділу

У цьому розділі була детально розписана постановка задачі, а також визначенні завдання, які потрібно буде вирішити у процесі розробки програмного забезпечення. Також, було сформовано чіткі вимоги до програмного забезпечення, що розробляється.

Було проведено детальний аналіз існуючих програмних рішень, які використовуються у задачах прогнозування енергоспоживання. У результаті аналізу, було обрано перелік регресійних моделей, які будуть протестовані у даній роботі. На основі обраного методу буде реалізовано кінцеве програмне забезпечення.

Визначено перелік програмних засобів, які будуть використанні при розробці програмного забезпечення. Ці інструменти дозволять реалізувати якісний та надійний програмну реалізацію прогнозування енергоспоживання.

2 МЕТОДИ ТА ЗАСОБИ ПРОГНОЗУВАННЯ ЕНЕРГОСПОЖИВАННЯ

Найважливішим кроком у дослідженні та реалізації програмного забезпечення є вибір методів та засобів розробки. У цій частині роботи буде детально описано вхідні дані для навчання та тестування, а також обрано найкращу модель машинного навчання на основі порівняльного аналізу.

2.1 Опис вхідних даних

Дані були зібрані з малої металургійної промисловості в Південній Кореї, а саме DAEWOO Steel Co. Ltd у місті Куаньян [6]. Ця компанія виробляє кілька видів котушок, сталевих пластин і залізних плит. Інформація про споживання електроенергії цією компанією зберігається в хмарній системі, якою володіє та керує Korea Electric Power Corporation.

Дані містять щоденні, місячні та річні цифри споживання енергії, які розраховуються та відображаються для аналізу. Набір даних, який використовувався у роботі, включає 11 атрибутів і 35040 записів, що охоплюють річний період спостереження з 15-хвилинними інтервалами.

Таблиця 2.1 - Опис вхідних даних

Назва показника	Тип даних	Одиниці вимірювання	Опис
Date	Неперервні	date	Дата та час
Usage_kWh	Неперервні	kWh	Спожита енергія
Lagging_Current_Reactive.Power_kVarh	Неперервні	$kVarh$	Реактивна потужність, коли струм відстає від напруги

Кінець таблиці 2.1

Leading_Current_Reactive_Power_kVarh	Неперервні	$kVarh$	Реактивна потужність, коли струм випереджає від напруги
$CO_2(tCO_2)$	Неперервні	ppm	Викиди вуглекислого газу
Lagging_Current_Power_Factor	Неперервні	%	Коефіцієнт потужності, коли струм відстає від напруги
Leading_Current_Power_Factor	Неперервні	%	Коефіцієнт потужності, коли струм випереджає від напруги
NSM	Неперервні	sec	Кількість секунд від опівночі
WeekStatus	Категорійні	Weekday/Weekend	Тип дня тижня
Day_of_week	Категорійні	Monday… Sunday	День тижня
Load_Type	Категорійні	Light, Medium, Maximum Load	Рівень навантаження

У таблиці 2.1 поверхнево описано змінні, що містяться у початкових даних. Вказано тип кожної змінної та одиниці вимірювання.

2.2 Розвідувальний аналіз даних

Важливим кроком у будь-якому дослідженні є ознайомлення та аналіз даних. Це допоможе детальніше зрозуміти природу та структуру даних, виявити зв’язки між змінними, а також знайти та виправити всі пропущені або некоректні записи у даних.

Розвідувальний аналіз даних (англ. Exploratory Data Analysis, EDA) — це підхід до аналізу даних з метою розуміння та узагальнення їх основних характеристик [7]. Розвідувальний аналіз найчастіше використовує статистичні та графічні методи для опису даних, залежностей, аномалій та викидів. Графічне представлення стає особливо корисним, коли дані мають великий об'єм, оскільки відбувається перетворення числових значень у зрозумілі графічні форми. Це допомагає визначити тенденції, а також формувати гіпотези на основі візуалізацій.

У цьому розділі буде надано детальний опис розвідувального аналізу даних, який був проведений з метою кращого їх розуміння. Застосовані статистичні та графічні методи аналізу дозволили виявити ключові особливості вхідного набору даних та надали інформацію про фактори, які найбільше впливають на енергоспоживання підприємства.

2.2.1 Цілісність даних

Аналіз починається із перевірки на наявність пропущених даних, а також коректність їх типу відповідно до опису змінної.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35040 entries, 0 to 35039
Data columns (total 11 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   date                                  35040 non-null  object
 1   Usage_kWh                             35040 non-null  float64
 2   Lagging_Current_Reactive.Power_kVarh  35040 non-null  float64
 3   Leading_Current_Reactive_Power_kVarh  35040 non-null  float64
 4   CO2(tCO2)                             35040 non-null  float64
 5   Lagging_Current_Power_Factor          35040 non-null  float64
 6   Leading_Current_Power_Factor          35040 non-null  float64
 7   NSM                                   35040 non-null  int64
 8   WeekStatus                            35040 non-null  object
 9   Day_of_week                           35040 non-null  object
 10  Load_Type                             35040 non-null  object
dtypes: float64(6), int64(1), object(4)
memory usage: 2.9+ MB
```

Рисунок 2.1 – Опис набору даних за допомогою функції info()

На рисунку 2.1 помітно, що дані не містять пропущених значень. Також варто перейменувати назви колонок та привести їх до одного шаблону.

Таблиця 2.2 – Опис перейменування колонок

Стара назва колонки	Нова назва колонки
Lagging_Current_Reactive.Power_kVarh	Lagging_Reactive_Power_kVarh
Leading_Current_Reactive_Power_kVarh	Leading_Reactive_Power_kVarh
Lagging_Current_Power_Factor	Lagging_Power_Factor
Leading_Current_Power_Factor	Leading_Power_Factor'
$CO_2(tCO_2)$	CO_2

У таблиці 2.2 зазначено нові назви змінних, які покращують якість їх опису.

2.2.2 Статистичні характеристики не категорійних змінних

На рисунку 2.2 описані основні статистичні метрики по кожній числовій змінній набору даних, а саме: кількість записів, середнє значення, стандартне відхилення, мінімальне та максимальне значення, 25-ий, 50-ий та 75-ий перцентилі.

	count	mean	std	min	25%	50%	75%	max
Usage_kWh	35040.0	27.386892	33.444380	0.0	3.20	4.57	51.2375	157.18
Lagging_Reactive_Power_kVarh	35040.0	13.035384	16.306000	0.0	2.30	5.00	22.6400	96.91
Leading_Reactive_Power_kVarh	35040.0	3.870949	7.424463	0.0	0.00	0.00	2.0900	27.76
CO2	35040.0	0.011524	0.016151	0.0	0.00	0.00	0.0200	0.07
Lagging_Power_Factor	35040.0	80.578056	18.921322	0.0	63.32	87.96	99.0225	100.00
Leading_Power_Factor	35040.0	84.367870	30.456535	0.0	99.70	100.00	100.0000	100.00
NSM	35040.0	42750.000000	24940.534317	0.0	21375.00	42750.00	64125.0000	85500.00

Рисунок 2.2 – Статистичний опис набору даних за допомогою функції describe()

Такий поверхневий опис створює перше, ознайомче, враження про дані і є хорошою практикою при розвідувальному аналізі.

2.2.3 Аналіз енергоспоживання

Перш за все, дана робота зосереджена на аналізі енергоспоживання на підприємстві. Важливо встановити, які саме фактори мають найбільший вплив на

споживання енергії. Ці допоможе у майбутньому інтерпретувати результати моделей машинного навчання, а також підтвердити висунуті гіпотези.

Першим кроком у цьому процесі варто визначити, як змінні корелюють між собою, іншими словами, які змінні найбільше впливають одна на одну. Для цього будується кореляційна матриця, яка дозволяє виявити наявність та силу зв'язків між різними змінними. Кореляційна матриця представлена на рисунку 2.3.

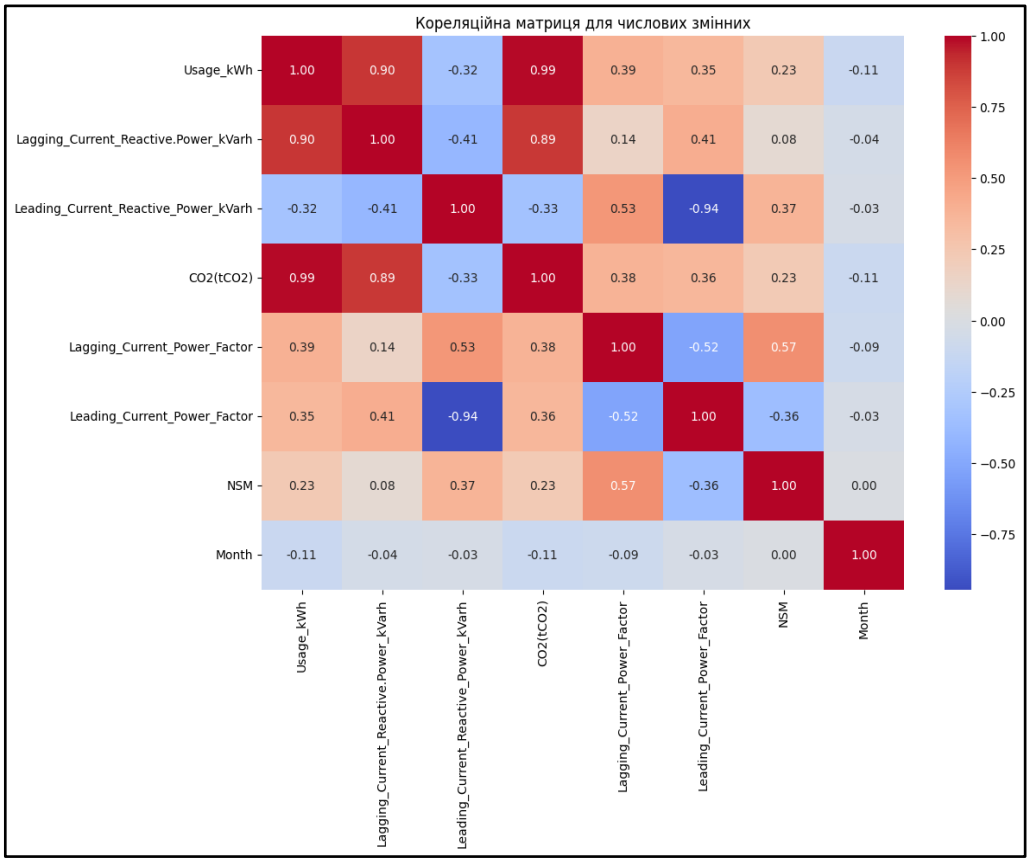


Рисунок 2.3 — Кореляційна матриця

Як можна зазначити, енергоспоживання найбільше корелює зі змінною CO2, що показує викид вуглекислого газу. На основі цього, можна висунути гіпотезу, що це буде найбільш впливова змінна у прогнозуванні енергоспоживання. Це має сенс, оскільки кількість вуглекислого газу, що виробляє підприємство напряму залежить від його виробничих потужностей.

Наступним кроком є оцінка об’ємів енергоспоживання у різні проміжки часу. Це може дати корисну інформацію, що дозволить якісніше розуміти дані. Візьмемо щомісячний, середньотижневий та середньодобовий об’єм споживання.

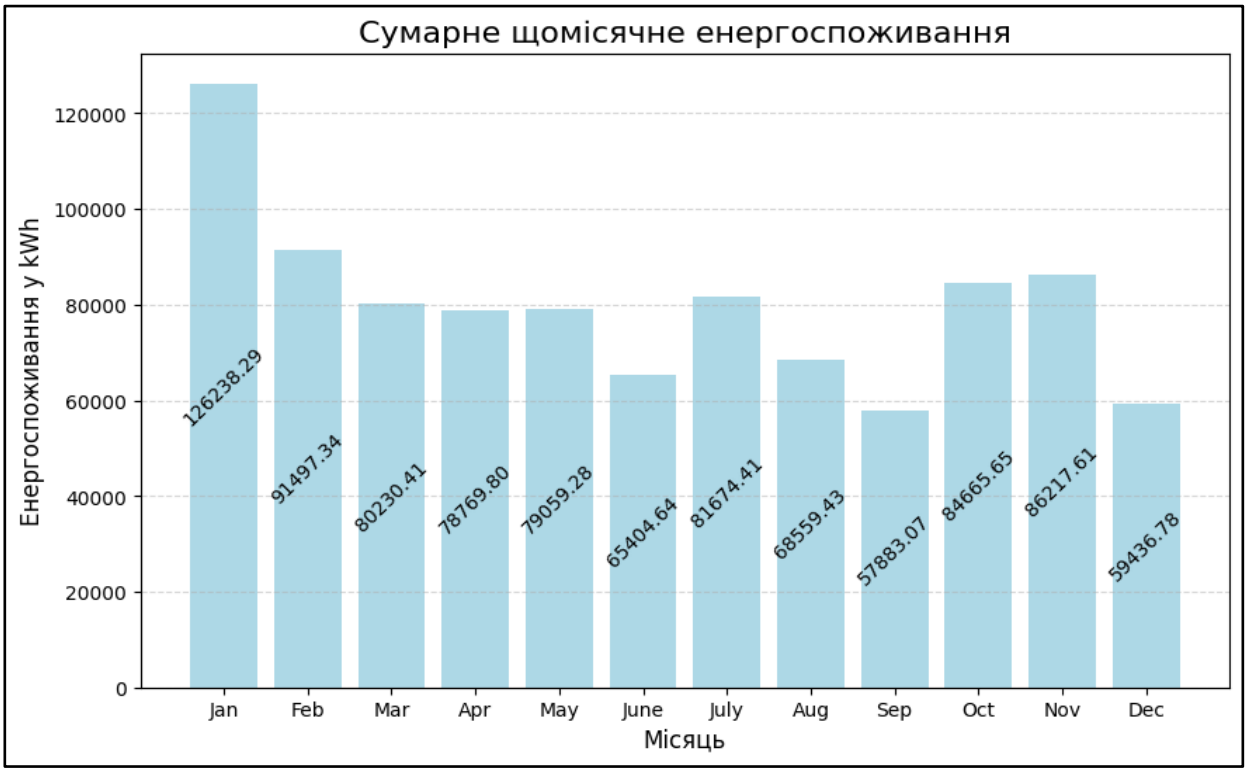


Рисунок 2.4 — Графік щомісячного енергоспоживання

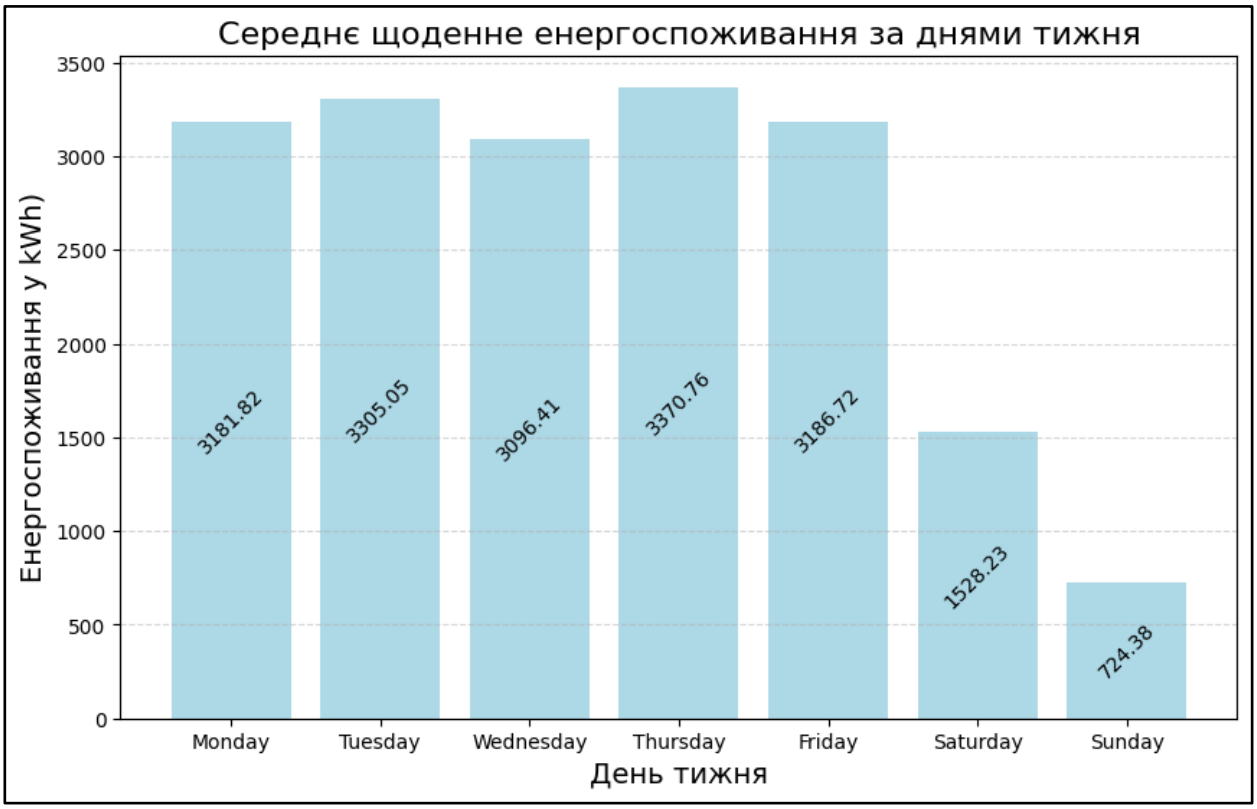


Рисунок 2.5 — Графік середньотижневого енергоспоживання

Рисунок 2.6 — Графік середньодобового енергоспоживання залежно від періоду доби

Проаналізувавши графіки, можемо помітити декілька тенденцій:

1. На рисунку 2.4 зображено графік щомісячного енергоспоживання бачимо, що найбільше споживання відбулось у період січня та лютого. На основі цих даних не можна висунути ніяких гіпотез, оскільки невідомо як погодні умови впливають на це підприємство. Також, вибірка в 1 рік замала, щоб аналізувати та обговорювати щорічні тенденції.

2. На рисунку 2.5 зображено графік, на якому неозброєним оком помітно, що енергоспоживання у вихідні значно нижче ніж у будні дні. На основі цього, можна вважати доцільним збільшення роботи у вихідні, аби більш рівномірно розподілити навантаження на підприємстві. Це вплине на ефективність та збалансованість використання енергії.

3. На рисунку 2.6 зображено графік, на якому помітно, як сильно відрізняється енергоспоживання у нічний період доби порівняно з іншими. Зважаючи на це, можна прийти до таких самих висновків, що і при аналізі

щотижневого енергоспоживання. Впровадження нічних змін та нарощування виробництва у цей період може покращити ефективність використання ресурсів, оскільки їх рівномірний розподіл позитивно вплине на роботу підприємства.

2.3 Огляд регресійних моделей

У даній роботі буде розглянуто декілька підходів до прогнозування енергоспоживання підприємства. Для регресійного прогнозування застосуємо різні моделі декількох класів: лінійні, градієнтні, деревоподібні, ансамблеві та нейронні.

У таблиці 2.3 зазначені моделі та їх належність до певного класу.

Таблиця 2.3 — Класи регресійних моделей

Лінійні	Градієнтні	Деревоподібні	Інші
Ridge	LGBMRegressor	DecisionTree Regressor	BaggingRegressor
LinearRegression	XGBRegressor	RandomForest Regressor	MLPRegressor
PassiveAggressive Regressor	GradientBoosting Regressor	ExtraTrees Regressor	KNeighbors Regressor
ElasticNet	CatBoostRegressor		AdaBoostRegressor
HuberRegressor			
QuantileRegressor			
Lars			
BayesianRidge			
Lasso			
TheilSenRegressor			
TweedieRegressor			

2.3.1 Лінійні регресійні моделі

Лінійна регресійна модель є одним із найбільш вживаних інструментів прогнозування, що застосовується у статистиці та машинному навчанні [8]. Цей метод покриває велику частину задач, із якою можна зіштовхнутись під час прогнозування, оскільки більшість залежностей між параметрами та результатом є лінійно залежними. Лінійна регресія є простою у застосуванні, оскільки можна легко інтерпретувати результати моделі.

Основна форма лінійної регресії:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon, \quad (1)$$

де y — незалежна змінна,

β_0 — вільний член (перетин з віссю O_y),

$\beta_1, \beta_2, \ldots, \beta_n$ — коефіцієнти регресії,

$x_1, x_2, \ldots, x_n$ — незалежні змінні (параметри, фактори),

ε — похибка.

Інші моделі лінійної регресії, такі як Ridge, Lasso, ElasticNet, HuberRegressor, QuantileRegressor, Lars, BayesianRidge, PassiveAggressiveRegressor, TheilSenRegressor та TweedieRegressor пропонують різні підходи до поліпшення класичної лінійної регресії. Основна їх мета - уникнути певних недоліків та обмежень класичної регресії, таких як чутливість до викидів, надмірна складність моделі або недооцінка невизначеності параметрів.

Моделі з регуляризацією, такі як Ridge, Lasso та ElasticNet, додають штраф за розмір коефіцієнтів, що допомагає контролювати перенавчання та поліпшує узагальнювальну здатність. Ridge штрафує суму квадратів коефіцієнтів. Lasso штрафує суму абсолютних значень коефіцієнтів [9]. ElasticNet це комбінація Ridge та Lasso.

Моделі, такі як HuberRegressor та TheilSenRegressor, роблять лінійну регресію більш стійкою до викидів, забезпечуючи більш надійні результати в присутності аномальних даних. На відміну від інших методів лінійної регресії, TheilSenRegressor не робить припущення про розподіл даних [10]. Крім того,

моделі, такі як QuantileRegressor, дозволяють враховувати не тільки середнє значення, але й різні квантилі розподілу залежної змінної, що корисно для оцінки ризиків та роботи з екстремальними значеннями [11]. BayesianRidge використовує апріорне знання про параметри моделі, що допомагає їй уникнути перенавчання та покращити загальну продуктивність. PassiveAggressiveRegressor добре підходить для роботи з потоками даних, завдяки своїй здатності швидко адаптуватися до нових даних.

Узагальнюючи, інші моделі лінійної регресії розширюють функціональні можливості класичної лінійної регресії, надаючи більш гнучкі та стійкі методи для прогнозування та аналізу даних. Вони дозволяють враховувати різні аспекти складності та структуру даних, що робить їх ефективними інструментами для різноманітних задач у сфері аналізу даних та прогнозування.

2.3.2 Градієнтні регресійні моделі

Градієнтні регресії — це моделі, які використовують градієнтні спуски для оптимізації функції витрат (відхилень або похибок). У загальному випадку, градієнтний спуск є чисельним методом знаходження локального мінімума чи максимума заданої функції. У контексті регресійного прогнозування та машинного навчання, мінімумом функції є оптимальний перелік параметрів для задачі, які є відображенням точності та ефективності роботи моделі. Іншими словами, градієнтний спуск допомагає шукати оптимальні параметри моделі, які зменшують різницю між фактичними та прогнозованим значеннями.

Градієнтні методи, такі як LGBMRegressor, XGBRegressor, GradientBoostingRegressor та CatBoostRegressor, є потужними інструментами для задач регресійного аналізу, які використовують ансамбль дерев прийняття рішень. Всі ці методи мають спільний принцип роботи, але й кожен з них має власні переваги та особливості роботи, тому, при виборі бібліотеки та регресії, варто зважати на особливості задачі.

LightGBM (LGBMRegressor) відомий своєю високою швидкістю навчання та ефективністю, завдяки використанню гістограмних методів та паралельних

обчислень [12]. XGBoost (XGBRegressor) відзначається високою точністю та кількістю різних налаштувань, що робить його популярним у багатьох сферах машинного навчання [13]. GradientBoostingRegressor з бібліотеки Sklearn є добре збалансованим підходом, який поєднує швидкість та точність, і має широкий спектр параметрів для налаштування моделі. Нарешті, CatBoostRegressor, спеціалізується на роботі з категоріальними даними та автоматичною обробкою пропущених значень, спрощуючи процес побудови моделі та покращуючи точність прогнозів.

Як результат, градієнтні методи регресії надають потужні та ефективні інструменти для прогнозування, які використовують градієнтний спуск для пошуку оптимальних параметрів моделі та покращення її результатів. Кожне з вказаних методів має свої особливості та переваги, що робить їх популярними серед методів прогнозування у машинному навчанні.

2.3.3 Деревоподібні регресійні моделі

Деревоподібні методи машинного навчання — це клас алгоритмів, які базуються на створенні моделі прийняття рішень у вигляді дерева. Вони включають в себе такі методи: дерева рішень, випадковий ліс, градієнтне підсилення.

Основна ідея полягає в тому, щоб у кожному вузлі дерева робити розбиття на основі певного критерію, які належать задачі. Ці методи використовуються для задач як класифікації, так і регресії.

DecisionTreeRegressor, RandomForestRegressor та ExtraTreesRegressor, використовуються для прогнозування числових значень на основі вхідних ознак [14-16]. DecisionTreeRegressor будує просте дерево рішень для прогнозування, здатне до легкої інтерпретації та роботи з різноманітними типами даних. RandomForestRegressor і ExtraTreesRegressor використовують ансамбль дерев рішень для отримання більш точних прогнозів, дозволяючи обробляти великі обсяги даних та зменшуючи ризик перенавчання. Вони забезпечують високу ефективність навчання з обмеженими даними та допомагають відбирати найбільш важливі ознаки для прогнозування числових значень.

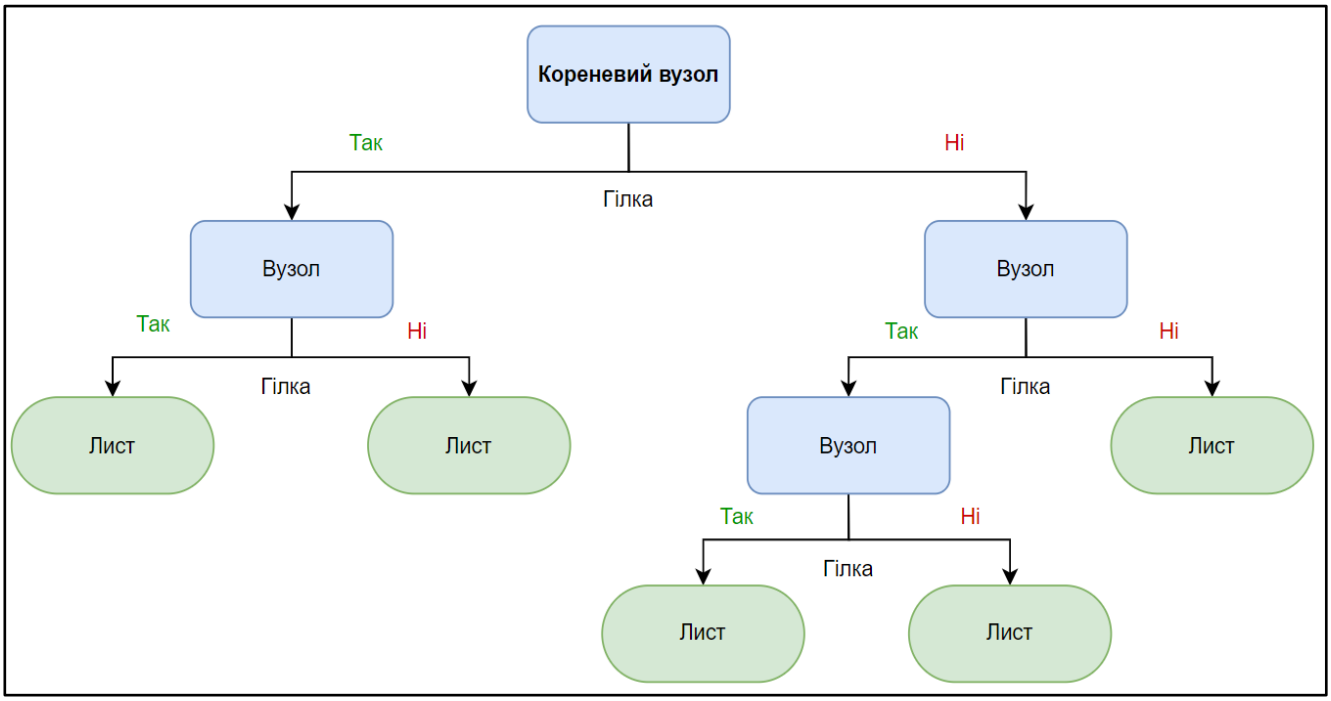


Рисунок 2.7 - Структура дерева прийняття рішень

Характеристики деревоподібних методів машинного навчання:

1. Простота інтерпретації: Дерева рішень мають просту структуру, яка сприяє легкому розумінню принципу прийняття рішень.

2. Незалежність від нормалізації: Вони працюють як з числовими, так і з категоріальними даними, не потребуючи їхньої нормалізації перед використанням.

3. Ефективність з обмеженими даними: Дерева рішень можуть ефективно працювати з невеликими обсягами даних, що дозволяє їх використання в умовах обмежених ресурсів.

4. Регуляризація для запобігання перенавчанню: Деякі методи автоматично використовують регуляризацію, що допомагає уникнути перенавчання та покращує загальну ефективність моделі.

Деревоподібні методи машинного навчання мають кілька важливих недоліків. Вони схильні до перенавчання, що призводить до поганої узагальнюючої здатності на нових даних. Моделі можуть створювати надмірно специфічні правила, що не відображають загальні закономірності. Дерева рішень чутливі до змін у даних, що робить їх нестабільними. Процес побудови дерев часто не забезпечує глобально оптимальних рішень, а створення великих дерев вимагає

значних обчислювальних ресурсів. Для покращення ефективності і стабільності моделей рекомендується використовувати методи регуляризації, ансамблеві методи, а також обмеження на глибину дерев.

2.3.4 Регресійні моделі інших типів

До інших типів регресійних моделей були віднесені такі моделі як, BaggingRegressor, KNeighborsRegressor, MLPRegressor та AdaBoostRegressor. Їх не можна класифікувати до минулих класів регресійних методів, тому окремо опишемо їх особливості та способи застосування.

BaggingRegressor метод використовує техніку "bagging" для створення ансамблю моделей, що підвищує стійкість моделі до помилок у даних [17]. Принцип роботи полягає у тренуванні декількох моделей на випадкових підмножинах набору даних, після чого відбувається об'єднання їхніх прогнозів, щоб отримати більш точний результат.

До переваг методу BaggingRegressor можна віднести: зниження помилки моделі, покращення стійкості до перенавчання, можливість роботи з великими наборами даних. До недоліків відносять великі обчислювальні витрати, а також непросту інтерпретацію моделі.

Метод KNeighborsRegressor (k найближчих сусідів) використовує алгоритм k найближчих сусідів (kNN) для прогнозування значень. Принцип роботи алгоритму KNeighborsRegressor полягає у пошуку k найближчих сусідів нового прикладу в навчальному наборі даних і використовує їхні значення для прогнозування. Переваги даного методу полягають у простій реалізації, ефективності на великих наборах даних та стійкість до викидів. Недоліками є чутливість до вибору параметру k та зниження точності у випадках низької щільності даних.

Метод MLPRegressor використовує багатошарову штучну нейронну мережу для прогнозування числових значень [18]. MLPRegressor навчається на вхідних даних, коригуючи ваги нейронів мережі, щоб мінімізувати помилку прогнозування. Метод має можливість навчання на складних нелінійних залежностях та високу точність прогнозування на деяких задачах. Водночас, має складну реалізацію та

дуже чутливий до шумів у даних та пропущених значень у даних, до того ж, схильний до перенавчання.

AdaBoostRegressor — це адаптивний метод збільшення стабільності моделі, який навчає послідовність слабких моделей та надає більшу вагу помилковим прикладам, щоб покращити їх класифікацію. Він об'єднує прогнози цих моделей, зважуючи їх відповідно до їх точності. Цей метод є досить схожим на BaggingRegressor, але суттєва відмінність полягає у формуванні ансамблю моделей. Як було зазначено, моделі в ансамблі BaggingRegressor навчаються незалежно одна від одної на випадкових підвибірках набору даних, а прогнози всіх моделей об'єднуються. Тоді як AdaBoostRegressor будує послідовність слабких моделей, кожна з яких намагається коригувати помилки попередньої моделі шляхом зосередження уваги на неправильно класифікованих прикладах.

До переваг методу AdaBoostRegressor відносять точність прогнозування, попередження перенавчання та можливість роботи з великими наборами даних. Недоліки методу полягають у великих обчислювальних витратах та низьку інтерпретованість моделі.

2.4 Критерії оцінювання моделей

Машинне навчання містить безліч алгоритмів, стратегій та рішень для розв’язання різноманітних задач. Навчання моделей є лише відправною точкою у роботі, але справжня цінність полягає у результатах їх застосування. Про успіх чи невдачу можна свідчити лише після оцінки якості та ефективності виконання алгоритму.

Метрики — це міри оцінки якості моделей, які є невід’ємною частиною оцінки результатів навчання та роботи алгоритмів машинного навчання. Важливо зазначити, що метрики можуть відрізнятись від задачі до задачі. Оцінка результатів повинна враховувати специфіку задачі, оскільки одна й та ж метрика може мати

різне значення в різних контекстах. Завдання можуть суттєво відрізнятися, тому оцінювання результатів має бути адаптоване до кожної конкретної задачі.

Таблиця 2.4 — Типи машинного навчання та метрики оцінювання

Тип задачі	Перелік основних метрик
Класифікація	Accuracy, Precision, Recall, F1-score, ROC AUC, Confusion Matrix, Log Loss
Регресія	Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, R-squared, Mean Absolute Percentage Error
Кластеризація	Silhouette Score, Davies–Bouldin Index, Calinski-Harabasz Index, Adjusted Rand Index
Ранжування	Mean Reciprocal Rank, Normalized Discounted Cumulative Gain, Mean Average Precision
Виявлення аномалій	AUC-ROC, PR-AUC, F1 score
Прогнозування часових рядів	Mean Absolute Percentage Error, Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
Генерація даних	Inception score, Fréchet Inception distance, Kernel Inception Distance, Mean Squared Error
Рекомендація	Mean Average Precision, Precision at K (P@K), Recall at K (R@K)

Такі класичні та широковживані метрики як Accuracy, Precision, Recall, Specificity, Sensitivity, F1-score використовуються у задачах класифікації, де результатом є бінарні значення, на відміну від задач прогнозування, де даними є неперервні значення. У цій роботі використовуються регресійні моделі машинного навчання, тому оцінка якості буде відбуватись за допомогою певного переліку метрик, які відображають як добре навчена модель здатна прогнозувати відповідні значення. Для визначення точності прогнозування енергоспоживання на підприємстві використаємо такі метрики: Root Mean Squared Error, Mean Absolute Error, Mean Absolute Percentage Error, R-squared.

2.4.1 Кореневе середньоквадратичне відхилення (RMSE)

Коренева середньоквадратична похибка (англ. root mean square error, RMSE) — це популярна статистична метрика, яка використовується для оцінки точності прогнозування регресійними моделями.

RMSE можна подати у вигляді:

$$RMSE = \sqrt{MSE}, \tag{2}$$

де MSE — середньоквадратична похибка.

Середньоквадратична похибка (англ. mean squared error, MSE) — це середнє значення квадрату різниці між фактичним значенням та передбачуваним.

Середньоквадратична похибка подається у вигляді:

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y})^2 , \tag{3}$$

де n — кількість спостережень у вибірці,

y_i — фактичне значення i-го спостереження, $i = 1 \dots n$,

$\hat{y_i}$ — прогнозоване значення i-го спостереження, $i = 1 \dots n$.

Оскільки результатом віднімання фактичного від передбачуваного значення може бути як додатне, так і від’ємне значення, важливо підносити різницю до квадрату, щоб уникнути ситуацій взаємокомпенсації між позитивними та негативними відхиленнями.

Фактично, RMSE є коренем з MSE. Такий підхід дозволяє якісніше інтерпретувати результати, оскільки середньоквадратична похибка вимірюється у квадратних одиницях, на відміну від кореневої середньоквадратичної похибки, яка має ті ж одиниці вимірювання, що і початкові дані. Це означає, що RMSE надає більш зрозумілу оцінку того, наскільки далеко прогнозовані значення.

Отже, фінальна формула для RMSE подається у вигляді:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y})^2} , \tag{4}$$

Чим менше значення середньоквадратичної похибки (MSE) та кореневої середньоквадратичної похибки (RMSE), тим модель є більш точною у прогнозах.

2.4.2 Середня абсолютна похибка (MAE)

Середня абсолютна похибка (англ. mean absolute error, MAE) — це статистична міра, що оцінює точність моделі, порівнюючи прогнозоване із фактичним значенням. Математично, середня абсолютна похибка є середнім значенням абсолютних відхилень між прогнозованими та фактичними значеннями.

Середня абсолютна похибка подається у вигляді:

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} |y_i - \hat{y}|, \tag{5}$$

де n — кількість спостережень у вибірці,

y_i — фактичне значення i-го спостереження, $i = 1 \dots n$,

$\hat{y_i}$ — прогнозоване значення i-го спостереження, $i = 1 \dots n$.

Середня абсолютна похибка за способом свого обчислення та призначенням нагадує кореневу середньоквадратичну похибку (RMSE). Ці метрики обраховують відхилення у прогнозуванні та повертають значення в одиницях, які відповідають початковим даним.

Основна відмінність між MAE та RMSE в оцінці похибки, оскільки RMSE "штрафує" модель за надто велике відхилення у прогнозуванні шляхом піднесення у квадрат похибки. В той же час, MAE обробляє всі відхилення з однаковою вагою, не зважаючи на величину помилки.

Тому, вибір між MAE та RMSE для оцінки відхилення залежить від типу задачі та особливостей даних. Якщо важливим є величина та діапазон відхилення — варто використовувати кореневе середньоквадратичне відхилення. Однак, якщо більш важливим є точність прогнозування без урахування величини відхилень — середня абсолютна похибка буде найкращим вибором.

2.4.3 Середня абсолютна відсоткова похибка (MAPE)

Середня абсолютна відсоткова похибка (англ. mean absolute percentage error, MAPE) — це міра, що показує середнє відсоткове відхилення між прогнозованими та фактичними значеннями. Дана метрика вимірює точність прогнозування у відсотках і дає уявлення про середню відсоткову похибку моделі.

Середня абсолютна відсоткова похибка подається у наступному вигляді:

$$MAPE = \frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}}{y_i} \right|, \quad (6)$$

де n — кількість спостережень у вибірці,

y_i — фактичне значення i-го спостереження, $i = 1 \dots n$,

$\hat{y_i}$ — прогнозоване значення i-го спостереження, $i = 1 \dots n$.

Іншими словами, середню абсолютну відсоткову похибку (MAPE) можна описати як відношення середнього абсолютного відхилення (MAE) до середнього фактичного значення, усе помножене на 100, щоб виразити результат у відсотках. Чим менше значення MAPE, тим краще точність моделі прогнозування.

2.4.4 Коефіцієнт детермінації R^2

Міра R^2 або R-квадрат — це статистична міра, що використовується у регресійному аналізі і відображає те, наскільки добре модель описує дані [19]. Іншими словами, це чисельний показник, що показує яка частина варіації фактичних значень пояснюється моделлю.

Значення R-квадрату варіюється від 0 до 1. Якщо коефіцієнт детермінації становить 1, це означає, що модель ідеально описує дані. Коефіцієнт детермінації, що прямує до 0, пояснюється низькою точність прогнозування моделі.

На рисунку 2.8 продемонстрований ілюстративний приклад двох різних значень коефіцієнту детермінації, що описує два набори даних та якість їх прогнозування. Синя пряма та точки вздовж неї розташовані досить кучно, що означає якісну модель прогнозування. Червона ж пряма, не так добре описує дані, тому й коефіцієнт детермінації є значно нижчим за синю.

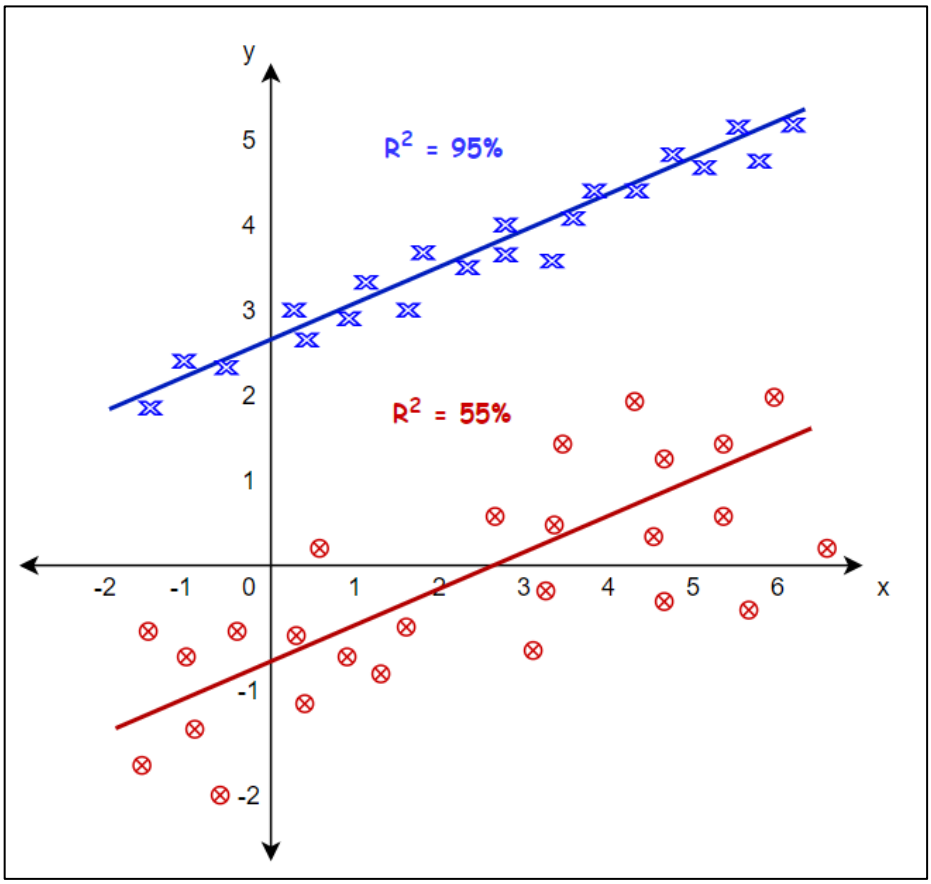


Рисунок 2.8 – Геометричне представлення коефіцієнту детермінації

Коефіцієнт детермінації подається у наступному вигляді:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \quad (7)$$

де n — кількість спостережень у вибірці,

y_i — фактичне значення i-го спостереження, $i = 1 \dots n$,

$\hat{y_i}$ — прогнозоване значення i-го спостереження, $i = 1 \dots n$,

$\bar{y}$ — середнє значення фактичних значень y_i, $i = 1 \dots n$.

Коефіцієнт детермінації є важливим інструментом оцінки у регресійних задачах та орієнтиром покращення моделі.

Проте високий коефіцієнт детермінації R^2 не завжди гарантує ідеальну модель. Для комплексної оцінки необхідно враховувати інші метрики та аналізувати можливі викиди й перенавчання.

2.5 Порівняння регресійних моделей

У цьому розділі будуть описані результати порівняльного аналізу різних класів регресійних моделей. Далі відбудеться вибір оптимальної моделі для використання у програмному забезпеченні для прогнозування енергоспоживання.

2.5.1 Порівняння лінійних регресійних моделей

Після навчання та оцінки лінійних регресійних моделей на тестовому наборі даних отримуємо результати їх роботи, продемонстровані у таблиці 2.5:

Таблиця 2.5 — Порівняння лінійних регресійних моделей

Model	MSE	RMSE	MAE	MAPE	R^2
Ridge	46.11	6.79	4.83	56.78	0.96
LinearRegression	17.79	4.22	2.56	18.38	0.98
Lasso	98.56	9.93	6.98	85.38	0.91
ElasticNet	97.97	9.90	6.96	84.90	0.91
HuberRegressor	196.91	14.03	8.22	40.91	0.83
QuantileRegressor	128.92	11.35	7.07	54.71	0.89
Lars	17.81	4.22	2.57	18.55	0.98
BayesianRidge	17.79	4.22	2.56	18.39	0.98
PassiveAggressiveRegressor	263.38	16.23	9.60	44.37	0.77
TheilSenRegressor	18.32	4.28	2.54	16.69	0.98
TweedieRegressor	141.68	11.90	8.81	107.69	0.88

У результаті, не отримуємо однозначну найкращу модель, оскільки метрики не вказують на єдиний регресійний метод. Маємо найкращу BayesianRidge за метриками середньоквадратимим відхиленням (MSE) та кореневим середньоквадратимим відхиленням (RMSE). Якщо зважати на метрики середньої

абсолютної похибки (MAE) та середньої абсолютної відсоткової похибки (MAPE), то TheilSenRegressor має найкращі показники точності.

Оскільки у задачі прогнозування енергоспоживання важливішим є абсолютна точність, тому MAE є оптимальним орієнтиром. Ця метрика менш чутлива до великих відхилень та краще пояснює точність моделі, якщо порівнювати з RMSE.

Підсумовуючи, лінійна регресійна модель TheilSenRegressor найкраще впоралась із прогнозуванням енергоспоживання на тестовому наборі даних.

2.5.2 Порівняння градієнтних регресійних моделей

Після навчання та оцінки градієнтних регресійних моделей на тестовому наборі даних отримуємо результати їх роботи, продемонстровані у таблиці 2.6.

Таблиця 2.6 — Порівняння градієнтних регресійних моделей

Model	MSE	RMSE	MAE	MAPE	R^2
LGBMRegressor	2.81	1.68	0.84	6.24	1.00
XGBRegressor	1.09	1.05	0.50	2.66	1.00
GradientBoostingRegressor	8.18	2.86	1.72	10.00	0.99
CatBoostRegressor	1.08	1.04	0.50	3.49	1.00

Аналогічно до результатів лінійних регресійних моделей, отримуємо 2 моделі, які конкурують у ключових метриках. CatBoostRegressor показує найкраще себе у 4 із 5 метрик, а XGBRegressor лише у 3 з 5. Тому, надаємо перевагу саме CatBoostRegressor.

2.5.3 Порівняння деревоподібних регресійних моделей

Після навчання та оцінки деревоподібних регресійних моделей на тестовому наборі даних отримуємо результати їх роботи, продемонстровані у таблиці 2.7:

Таблиця 2.7 — Порівняння деревоподібних регресійних моделей

Model	MSE	RMSE	MAE	MAPE	R^2
DecisionTreeRegressor	2.37	1.54	0.56	2.03	1.00
RandomForestRegressor	1.12	1.06	0.36	1.39	1.00
ExtraTreesRegressor	1.00	1.00	0.37	1.33	1.00

Серед деревоподібних регресійних моделей однозначно найкращою є модель ExtraTreesRegressor.

2.5.4 Порівняння регресійних моделей іншого типу

Після навчання та оцінки BaggingRegressor, KNeighborsRegressor, MLPRegressor та AdaBoostRegressor регресійних моделей на тестовому наборі даних, отримуємо результати їх роботи, продемонстровані у таблиці 2.8:

Таблиця 2.8 — Порівняння регресійних моделей інших типів

Model	MSE	RMSE	MAE	MAPE	R^2
BaggingRegressor	1.31	1.14	0.41	1.55	1.00
KNeighborsRegressor	25.50	5.05	2.31	8.57	0.98
MLPRegressor	127.31	11.28	7.69	72.52	0.89
AdaBoostRegressor	40.42	6.36	5.82	115.71	0.96

У підсумку, однозначно найкращою серед представлених моделей є BaggingRegressor. Саме ця модель забезпечила найкращі результати точності серед регресійних моделей KNeighborsRegressor, MLPRegressor та AdaBoostRegressor. Варто також зазначити, який великий відрив у BaggingRegressor по метриках.

2.5.5 Порівняння найкращих регресійних моделей свого класу

У підсумку, отримуємо 4 моделі, що показали найкращі результати у своєму класі регресійних моделей. Серед цих методів прогнозування оберемо найбільш

точний. У таблиці 2.9 наведений фінальний перелік регресійних моделей із результатами прогнозування на тестовому наборі даних.

Таблиця 2.9 — Порівняння найкращих регресійних моделей

Model	MSE	RMSE	MAE	MAPE	R^2
TheilSenRegressor	18.32	4.28	2.54	16.69	0.98
CatBoostRegressor	1.08	1.04	0.50	3.49	1.00
ExtraTreesRegressor	1.00	1.00	0.37	1.33	1.00
BaggingRegressor	1.31	1.14	0.41	1.55	1.00

Отже, у фінальному програмному забезпечені будемо використовувати модель ExtraTreesRegressor, оскільки вона забезпечує найкращу точність у ключових метриках, що робить її оптимальним методом для задачі прогнозування енергоспоживання.

2.6 Висновки до розділу

У цьому розділі було детально описано регресійні методи, які використовувались для прогнозування енергоспоживання. Провівши порівняльний аналіз, було визначено найкращу модель, яка забезпечує найбільшу точність передбачення. Також, було описано метрики, за якими відбувалась оцінка якості моделі.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

У даному розділі буде описано структуру та функціонал розробленого програмного забезпечення. Буде наведений перелік всіх розроблених програм, які використовували у підготовці фінального продукту.

3.1 Структура програмного забезпечення

Розроблене програмне забезпечення має 2 окремих програми, які мають різний функціонал, але є однаково корисними для прогнозування енергоспоживання на підприємстві. Також, у процесі було розроблено кілька допоміжних програм, які використовувались для попередньої обробки даних, розвідувального аналізу та тестування моделей прогнозування. Хоч ці програми не входять до фінального програмного забезпечення для прогнозування енергоспоживання на підприємстві, але вони були необхідними для реалізації якісного програмного рішення.

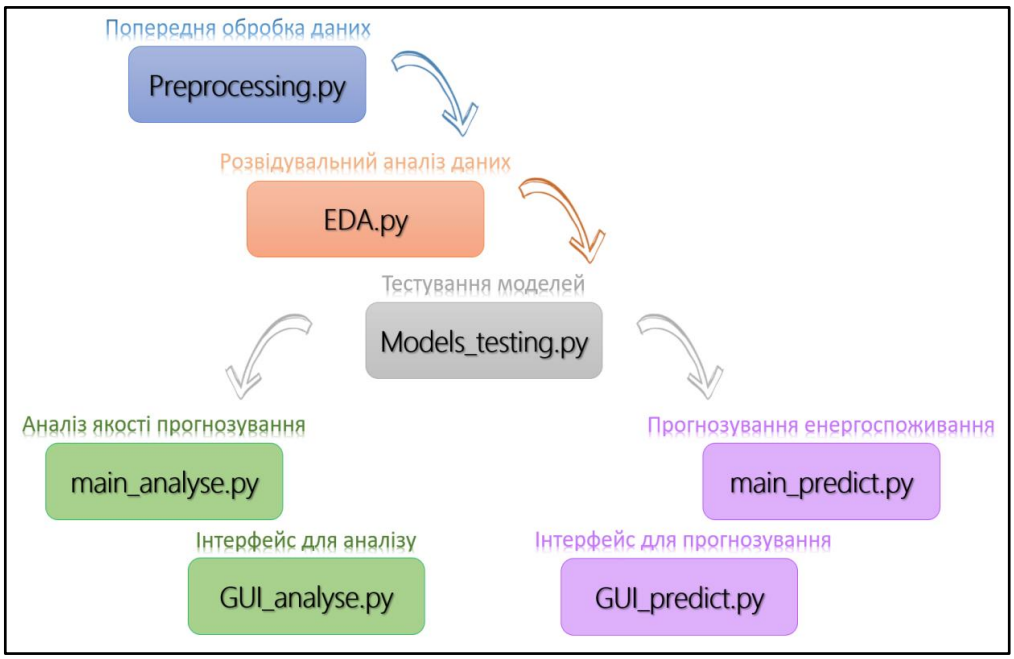


Рисунок 3.1 — Ієрархія процесу створення програмного забезпечення

Ця структура розробки є послідовною, оскільки кожен етап неможливий без попереднього. На початку обробляються вхідні дані, приводяться до одного формату, формуються нові, потенційно важливі змінні. Після цього етапу відбувається розвідувальний аналіз даних, який детально пояснює природу та особливості даних, а також відображає зв'язки між змінними. Найважливішим етапом розробки програмного забезпечення є тестування регресійних моделей, що забезпечує вибір найкращої моделі для задачі прогнозування.

Послідовність та умовний зв'язок виконання файлів Preprocessing.py, EDA.py та Models_testing.py описані на рисунку 3.2.

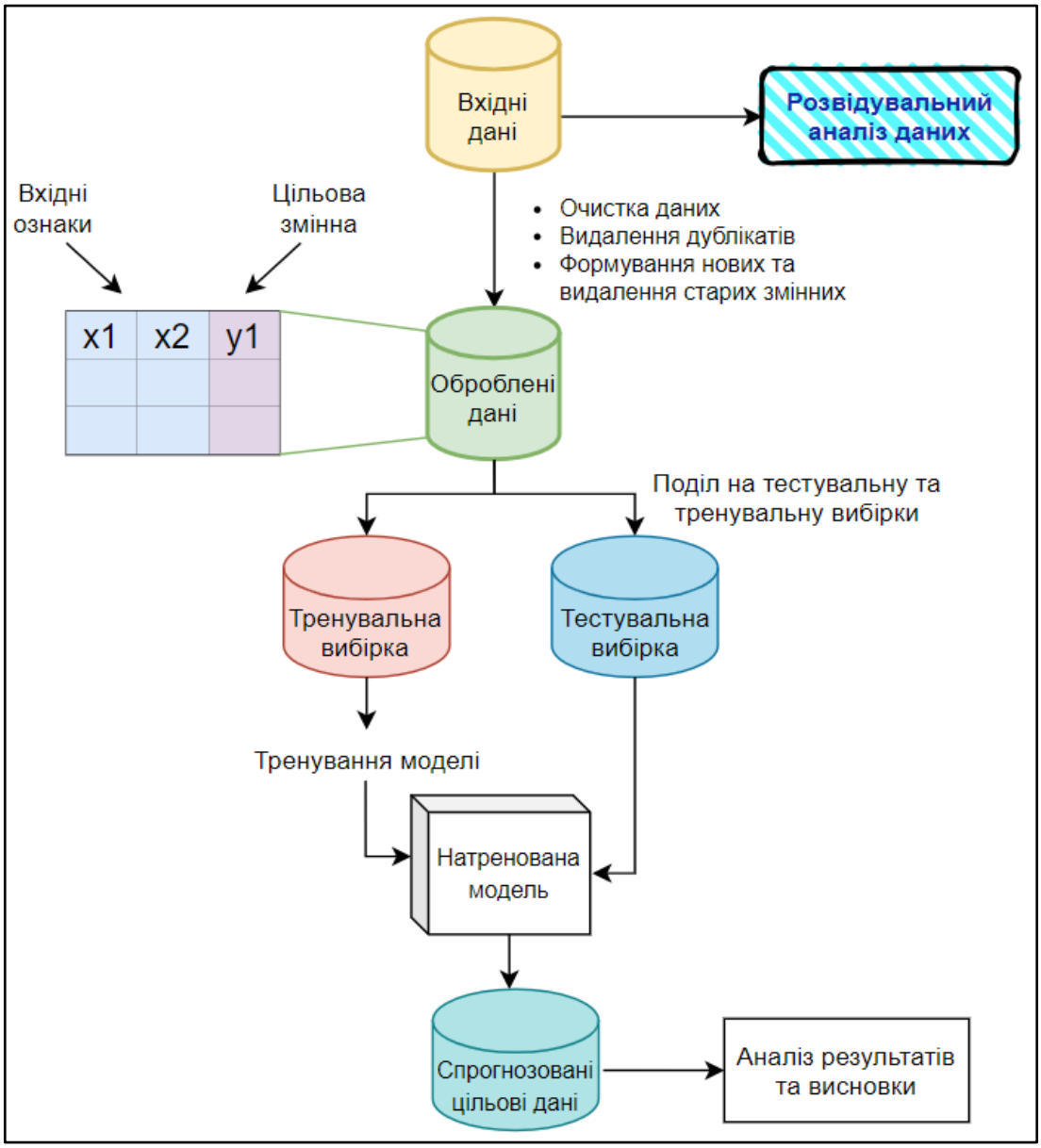


Рисунок 3.2 — Ілюстративна архітектура прогнозування енергоспоживання

Після розробки алгоритму, відбулась розробка програмного забезпечення, що дозволяє користувачеві зручно прогнозувати енергоспоживання на власному наборі даних.

Було вирішено поділити програмне забезпечення на 2 окремих програми у зв'язку із різним функціоналом. Розглядається 2 випадки користування програмою:

1. Користувач вже має набір даних, який містить споживання енергії та інші записи, які містять інформацію про час, тип навантаження підприємства, викид вуглекислого газу та інші показники.

2. Користувач має лише набір даних без фактичного значення енергоспоживання, а лише записи про час, тип навантаження підприємства, викид вуглекислого газу та інші показники.

Такий поділ забезпечує роздільне використання програм відповідно до задачі, яку потрібно вирішити користувачеві. Оскільки дані на вхід можуть бути різними, а моделі можуть працювати краще або гірше в залежності від даних, тому програма аналізу (GUI_analyse.py) допоможе користувачу з'ясувати і наглядно продемонструвати наскільки добре розроблене програмне забезпечення прогнозує енергоспоживання на заданому наборі даних.

Програма прогнозування (GUI_predict.py) працює у реальних умовах, коли користувач має на вхід певні дані роботи підприємства, та хоче спрогнозувати енергоспоживання. Її функціонал є спрощеним, у порівнянні із GUI_analyse.py, оскільки відсутня інформація про фактичне використання, тому неможливо побудувати графіки, навести таблиці із метриками або фактичне порівняння.

3.2 Опис програмного забезпечення

У цьому розділі буде детально описана структури та призначення кожної з розроблених програм. Крім того, розглянеться послідовність виконання конкретних функцій, процесів та методів, які використовуються у кожній програмі.

Визначаються ключові функції та методи, а також описується порядок виконання дій для досягнення конкретних цілей.

3.2.1 Опис файлу Preprocessing.py

Файл Preprocessing.py призначений для попередньої обробки даних, приведення всіх даних до єдиного вигляду та створення додаткових змінних для подальшої роботи вже з оновленими даними.

Структура файлу Preprocessing.py:

1. Імпорт необхідних бібліотек;
2. Завантаження початкового набору даних у форматі .csv;
3. Приведення зчитаних даних у табличний тип DataFrame;
4. Парсинг колонки, що містить дату, коли були записані відповідні дані у створення нових колонок;
5. Утворення змінних, що містять інформацію про місяць, день тижня та період дня (ранок, день, вечір, ніч);
6. Категоризація даних. Переведення категорійних даних (таких як день тижня, місяць, тип навантаження) у числовий;
7. Збереження оброблених даних у формат .csv для подальшого використання.

3.2.2 Опис файлу EDA.py

Файл EDA.py призначений для розвідувального аналізу даних. У ньому відбувається статистична оцінка даних, побудова графіків та пошук залежностей на їх основі.

Структура файлуEDA.py:

1. Імпорт необхідних бібліотек;
2. Завантаження обробленого набору даних у форматі .csv;
3. Приведення зчитаних даних у табличний тип DataFrame;
4. Огляд даних на наявність пропущених та не коректних значень;
5. Перейменування колонок;

6. Побудова кореляційної матриці;
7. Побудова графіків для подальшого аналізу.

3.2.3 Опис файлу Models_testing.py

Файл Models_testing.py призначений для тестування різних регресійних моделей та оцінки точності кожної з них. За результатом проведення тестування було обрану найкращу модель серед запропонованих.

Структура файлу Models_testing.py:

1. Імпорт необхідних бібліотек;
2. Завантаження обробленого набору даних у форматі .csv;
3. Приведення зчитаних даних у табличний тип DataFrame;
4. Розділення даних на тренувальну та тестові вибірки;
5. Ініціалізація регресійних моделей;
6. Навчання моделей;
7. Оцінювання якості моделей за допомогою регресійних метрик;
8. Вивід таблиці результатів якості роботи моделей.

3.2.4 Опис файлу GUI_analyse.py та main_analyse.py

Файли GUI_analyse.py та main_analyse.py є фінальним програмним забезпеченням, у яких реалізована взаємодія із користувачем. Файл GUI_analyse.py є головним, адже у ньому написаний інтерфейс програмного забезпечення, який викликає функції та методи із файлу main_analyse.py. Комбінація цих файлів надає зручний інструмент для аналізу якості та точності прогнозування енергоспоживання на основі вбудованого методу машинного навчання. Також, там наявні функції, що будують графіки та матрицю похибок. Ці методи дозволяють користувачеві наглядно перевірити як добре працює прогнозування.

Структура файлу GUI_analyse.py:

1. Імпорт необхідних бібліотек;
2. Встановлення теми та виду інтерфейсу, стилізація кнопок;
3. Створення вікна для роботи з користувачем;

4. Визначення функцій, які виконуються залежно від дій користувача.

5. Виведення на екран результати виконання функцій з файлу main_analyse.py;

6. Повторна робота користувача із програмою до її закриття.

3.2.5 Опис файлу GUI_predict.py та main_predict.py

Файли GUI_predict.py та main_predict.py є фінальним програмним забезпеченням у яких реалізована взаємодія із користувачем. Файл GUI_predict.py є головним, адже у ньому написаний інтерфейс програмного забезпечення, який викликає функцію прогнозування енергоспоживання із файлу main_predict.py. Комбінація цих файлів надає інструмент для точного прогнозування енергоспоживання на основі вбудованого методу машинного навчання.

Структура файлу GUI_predict.py:

1. Імпорт необхідних бібліотек;

2. Встановлення теми та виду інтерфейсу, стилізація кнопок;

3. Створення вікна для роботи з користувачем;

4. Визначення функцій, які виконуються залежно від дій користувача.

5. Виведення на екран результати виконання функцій з файлу main_analyse.py;

6. Повторна робота користувача із програмою до її закриття.

3.3 Висновки до розділу

У даному розділі було детально описано структуру реалізованого програмного забезпечення. Також було описано функціонал кожної із розроблених програм. Представлено архітектуру деяких файлів та ієрархію розроблених файлів.

4 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Цей розділ містить інформацію про інсталяцію програмного забезпечення, необхідні вимоги до обчислювальної техніки та демонструє функціонал розробленого програмного забезпечення через взаємодію із користувачем.

4.1 Інсталяція програмного забезпечення та відповідні вимоги до обчислювальної техніки

Для використання розробленого забезпечення користувач повинен мати встановлену мову програмування Python та наступні бібліотеки: pandas, numpy, sklearn, matplotlib, customtkinter. Програма не потребує підключення до інтернету та будь-яких додаткових налаштувань. Від користувача не вимагається високих характеристик обчислювальної техніки, на якій буде використовуватись програмне забезпечення. Програма є досить добре оптимізованою, але може довше працювати при прогнозуванні на великих вхідних даних.

4.2 Демонстрація функціоналу та представлення сценаріїв роботи

Розглянемо взаємодію користувача з програмою для аналізу якості прогнозування (GUI_analyse.py).

На рисунку 4.1 зображено початкове вікно розробленого програмного забезпечення, у якому буде відбуватись взаємодія із користувачем.

Для початку роботи програми потрібно натиснути на кнопку “Завантажити CSV файл”, що зображена на цьому ж рисунку 4.1, після чого відкриється вікно провідника, де користувач може обрати потрібний йому файл.

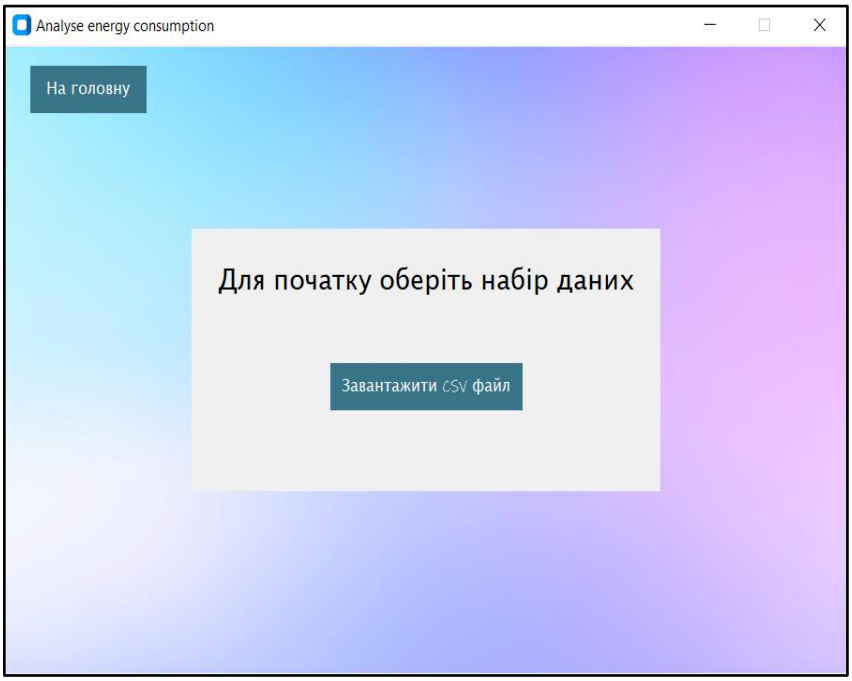


Рисунок 4.1 — Початкове вікно програми

Після успішного зчитування .csv файлу, користувача автоматично буде перенаправлено на нову сторінку, що зображена на рисунку 4.4. У вікні програми наявні 3 кнопки: “На головну”, “Показати вміст CSV файлу”, “Аналізувати дані”. Кнопка “На головну” повертає користувача на початкову сторінку застосунку.

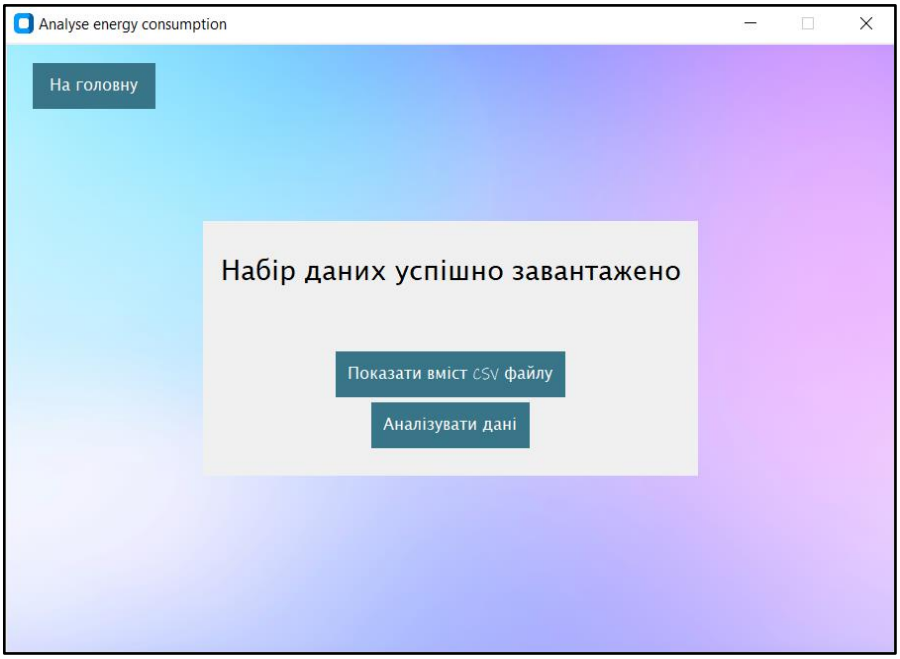


Рисунок 4.2 — Програмне вікно з повідомленням про успішне завантаження набору даних та функціональних кнопок

При натисканні на кнопку “Показати вміст CSV файлу” відкривається додаткове вікно, яке зображене на рисунку 4.3, у якому користувач може ознайомитись з даними, що були завантажені раніше.

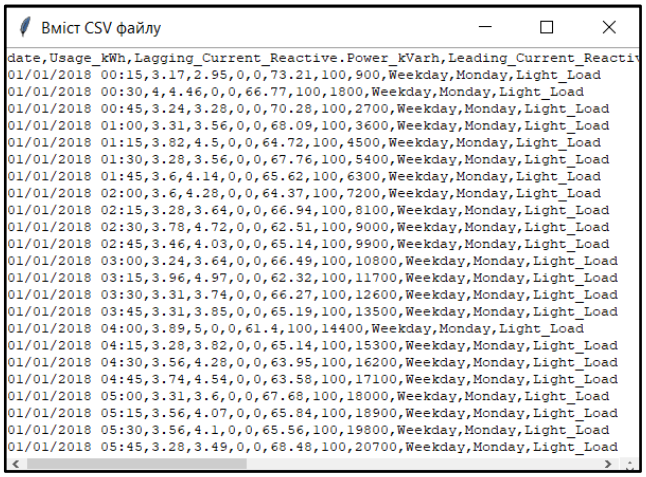


Рисунок 4.3 — Демонстрація вмісту файлу

При натисканні на кнопку “Аналізувати дані” відбувається виконання основного алгоритму, а саме прогнозування енергоспоживання, побудова графіків та обчислення точності. Після виконання алгоритму, користувач може оцінити якість прогнозування у новому вікні, яке зображено на рисунку 4.4.

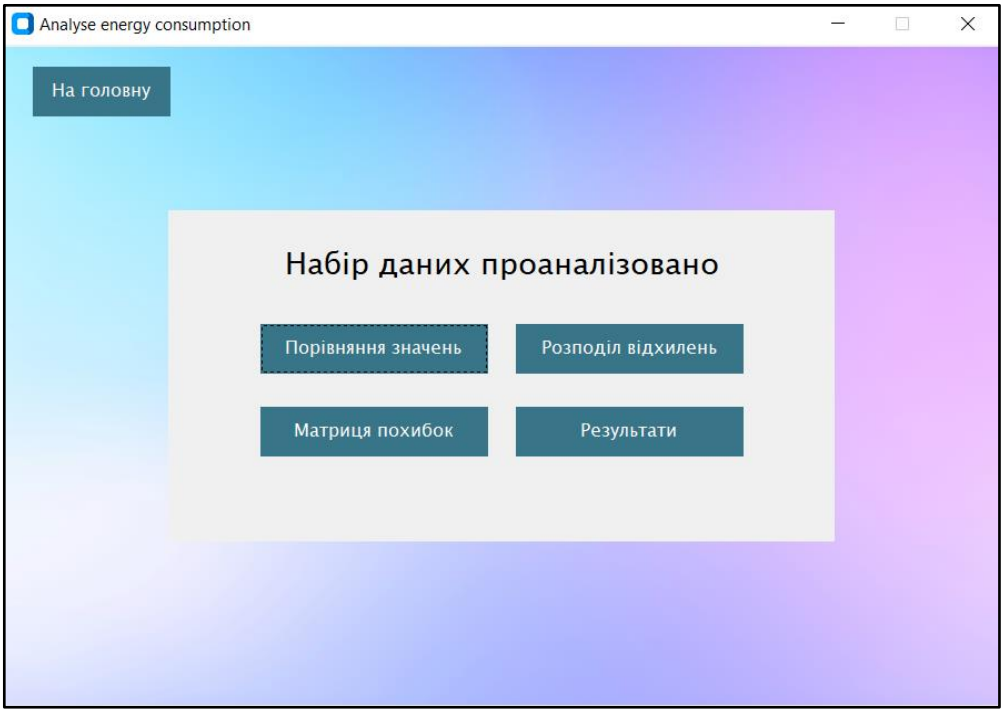


Рисунок 4.4 — Програмне вікно із переліком інструментів аналізу прогнозування

На рисунку 4.4 зображено 4 кнопки, що характеризують якість прогнозування енергоспоживання. Кнопка “Порівняння значень” відкриває графік, що зображений на рисунку 4.5, де показано, як прогнозовані дані збігаються з фактичними. Діагональна лінія на графіку відображає точне співпадіння між фактичними і прогнозованими значеннями.

Кнопка “Розподіл відхилень” відкриває графік 4.6, який показує розподіл помилок моделі. Чим “гостріший” графік навколо нуля, тим точніша модель.

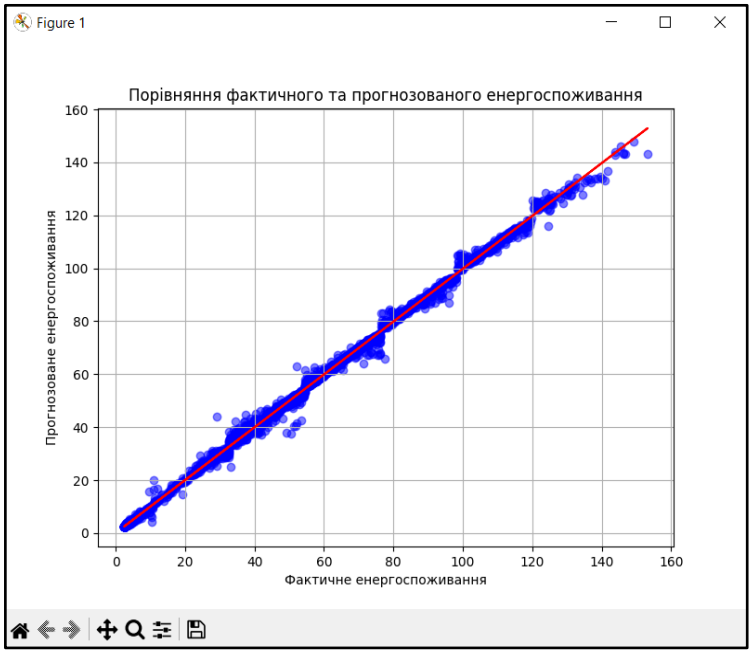


Рисунок 4.5 — Вивід графіка порівняння прогнозованого і фактичного значення

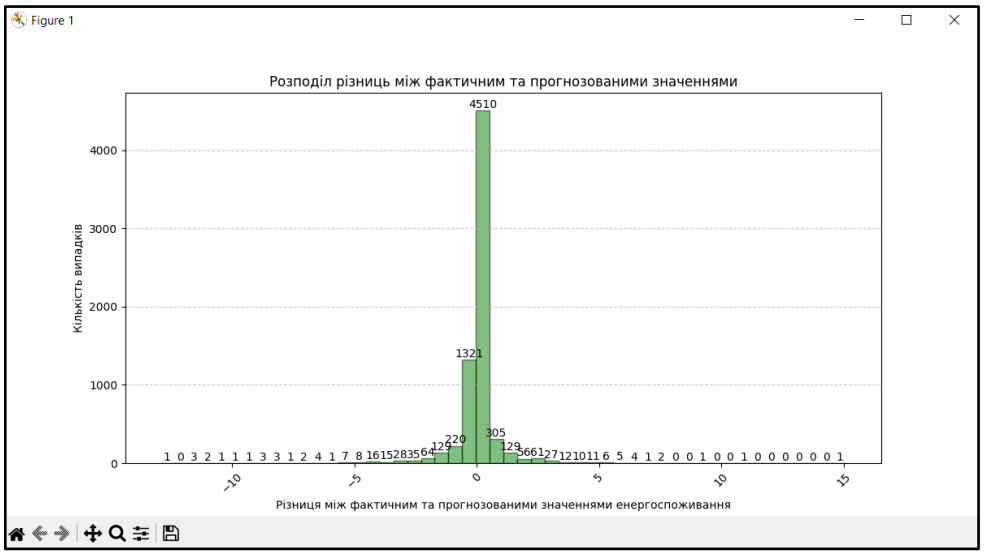
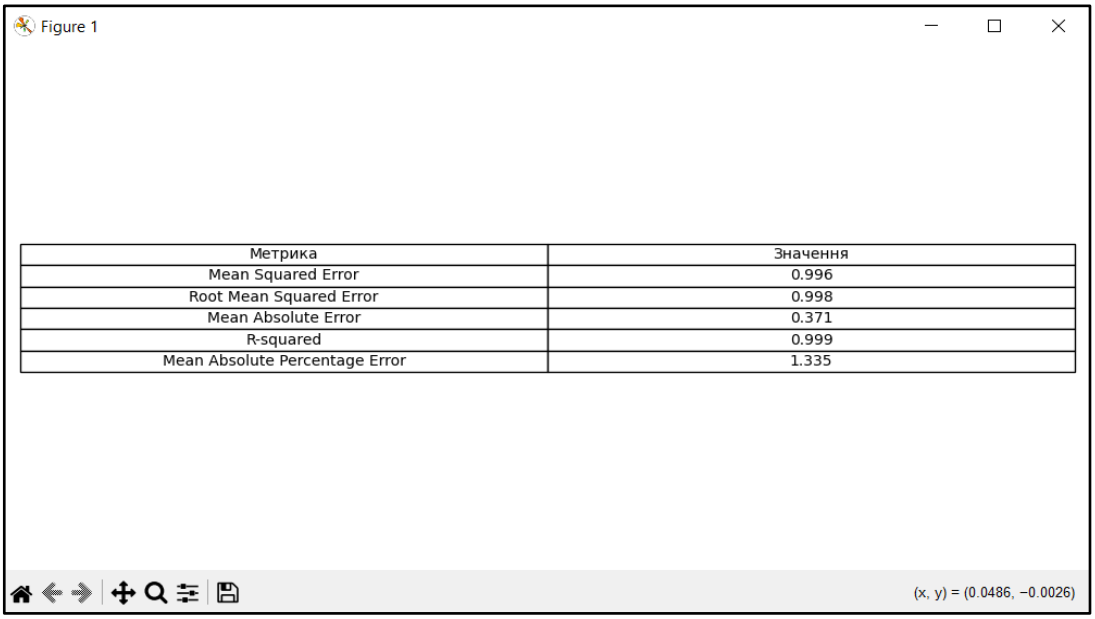


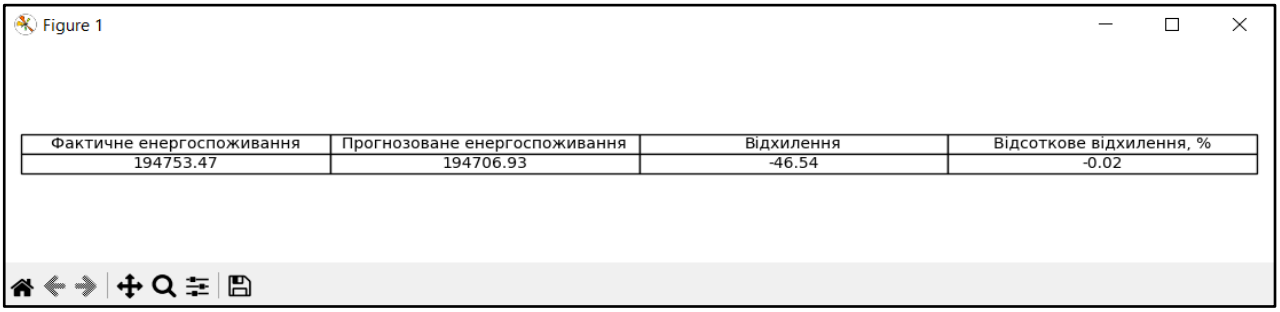
Рисунок 4.6 — Вивід графіка розподілу відхилень

Також, на рисунку 4.4 наявні кнопки “Матриця похибок” та “Результати”. Результати виконання цих кнопок зображені на рисунку 4.7 та рисунку 4.8 відповідно. Рисунок 4.7 відображає таблицю із метриками, які оцінюють точність моделі та дають уявлення про величину та характер помилок. Рисунок 4.8 відображає таблицю з результатами прогнозування енергоспоживання, де також вказується фактичне значення та абсолютну похибку між значеннями.



Метрика	Значення
Mean Squared Error	0.996
Root Mean Squared Error	0.998
Mean Absolute Error	0.371
R-squared	0.999
Mean Absolute Percentage Error	1.335

Рисунок 4.7 — Вивід таблиці “Матриця похибок” із ключовими метриками для оцінки точності



Фактичне енергоспоживання	Прогнозоване енергоспоживання	Відхилення	Відсоткове відхилення, %
194753.47	194706.93	-46.54	-0.02

Рисунок 4.8 — Вивід таблиці “Результат” з результатами прогнозування

Розглянемо взаємодію користувача з програмою для прогнозування енергоспоживання (GUI_predict.py). На рисунку 4.9 зображено початкове вікно розробленого програмного забезпечення, у якому буде відбуватись взаємодія з користувачем.

Для початку роботи програми потрібно натиснути на кнопку “Завантажити CSV файл”, що зображена на рисунку 4.9, після чого відкриється вікно провідника, де користувач може обрати потрібний йому файл.

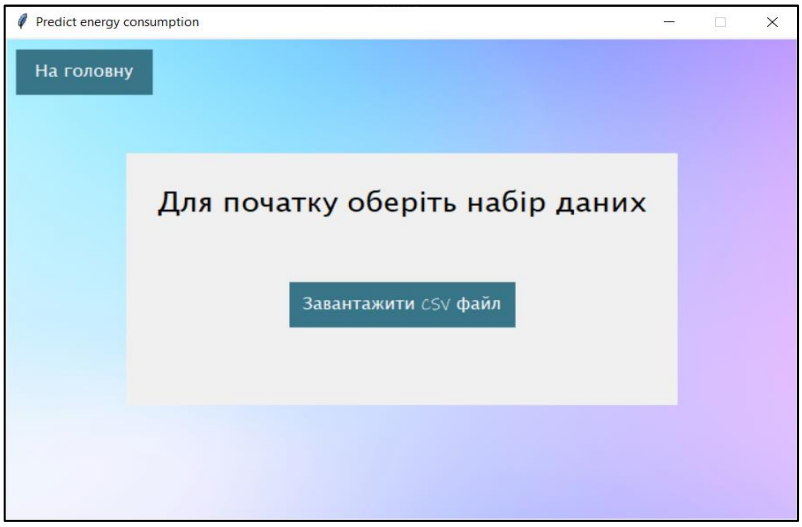


Рисунок 4.9 — Початкове вікно програми

Після успішного зчитування .csv файлу, користувача автоматично буде перенаправлено на нову сторінку, що зображена на рисунку 4.10. У вікні програми наявні 3 кнопки: “На головну”, “Показати вміст CSV файлу”, “Аналізувати дані”. Кнопка “На головну” повертає користувача на початкову сторінку застосунку.

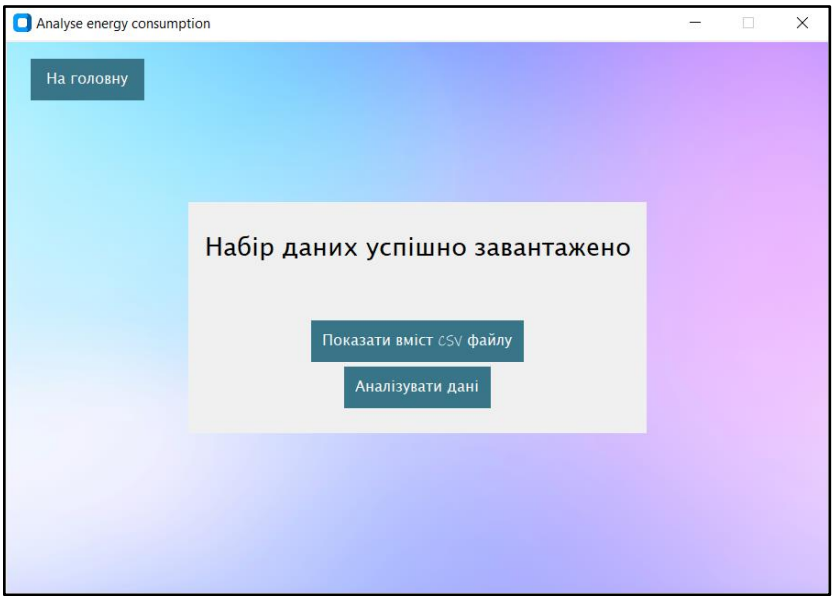


Рисунок 4.10 — Програмне вікно з повідомленням про успішне завантаження набору даних та функціональних кнопок

При натисканні на кнопку “Показати вміст CSV файлу” відкривається додаткове вікно, яке зображене на рисунку 4.11, у якому користувач може ознайомитись з даними, що були завантажені раніше.

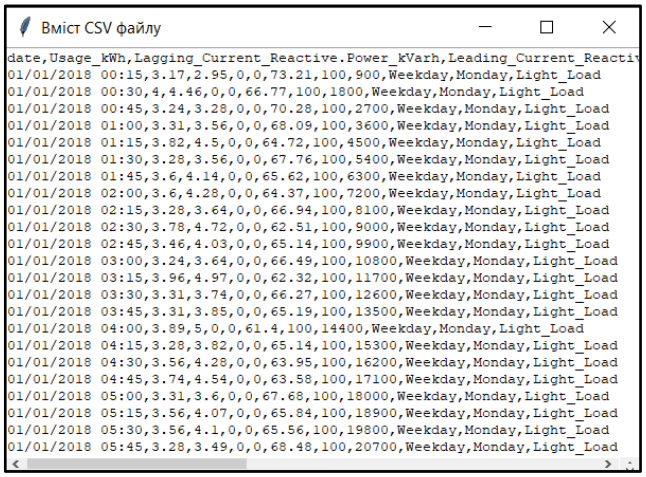


Рисунок 4.11 — Демонстрація вмісту файлу

При натисканні на кнопку “Аналізувати дані” відбувається виконання основного алгоритму, а саме прогнозування енергоспоживання. Після виконання алгоритму, користувачу у новому вікні, яке зображене на рисунку 4.14, може ознайомитись із результатом прогнозування, на основі наданого ним набору даних.

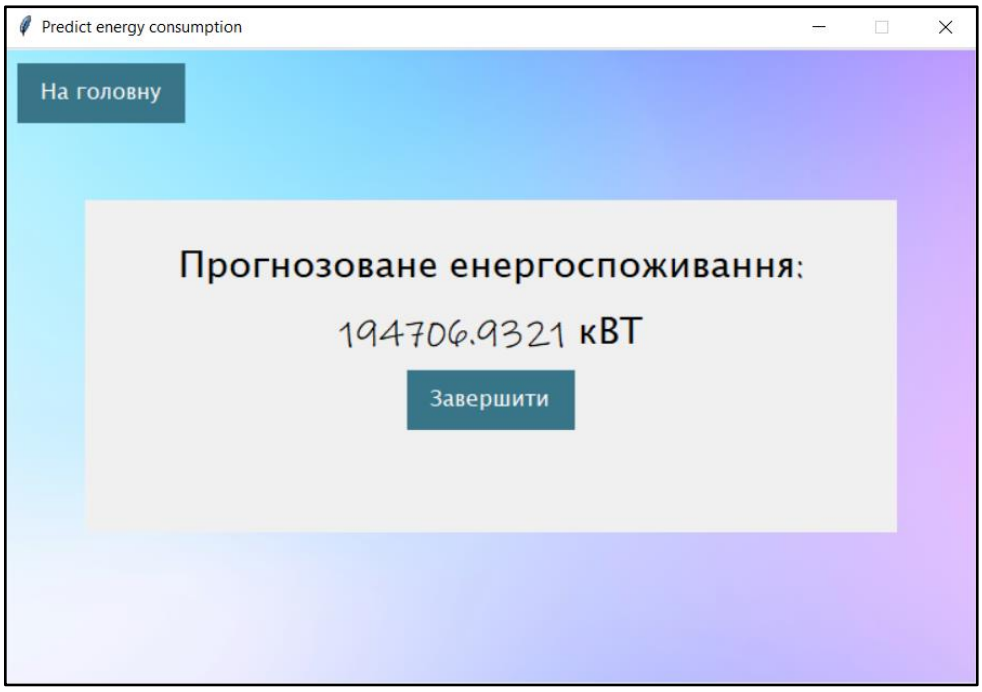


Рисунок 4.12 — Вивід результату прогнозування на екран

4.3 Висновки до розділу

У цьому розділі за допомогою рисунків та коментарів було описано та продемонстровано роботу користувача із розробленим програмним забезпеченням. У результаті, повністю описано весь шлях роботи користувача, що дозволяє користуватись програмним забезпеченням зі зручністю та легкістю. Також було описано технічні вимоги, зазначений повний перелік необхідного встановленого програмного забезпечення для коректної роботи програми.

ВИСНОВКИ

У процесі виконання дипломної роботи було вирішено всі поставлені задачі, отримавши точні результати прогнозування енергоспоживання на підприємстві, використовуючи методи машинного навчання.

Мету дипломної роботи було виконано в повному обсязі. Вдалось реалізувати програмне забезпечення, що з високою точністю прогнозує енергоспоживання на підприємстві. Таких результатів було досягнуто завдяки якісному підбору інструментів та методів розробки.

1. На основі аналізу підходів, методів та алгоритмів розв'язання задачі прогнозування енергоспоживання обґрунтовано використання ExtraTreesRegressor та сукупності метрик, таких як MSE, RMSE, MAE, MAPE та коефіцієнту детермінації, що дозволили забезпечити вражаючі результати точності прогнозування.

2. На основі аналізу програмних засобів обґрунтовано використання мови програмування Python, як найбільш популярної мови завдяки простоті реалізації програм найрізноманітнішого типу. Також, ключовим фактором у виборі мови програмування є її спільнота та екосистема готових реалізацій. У випадку Python, існує багато бібліотек із реалізованими алгоритмами машинного навчання, що значно спростило дослідження і дозволило протестувати найрізноманітніші методи та моделі. Також були використані оптимальні бібліотеки для задач обробки даних, візуалізацій та розробки користувацького інтерфейсу.

3. Було розроблено програмне забезпечення з 2 файлів, які мають зручний та функціональний інтерфейс, що дозволяє швидко прогнозувати та аналізувати енергоспоживання підприємства на основі вхідних даних.

4. На основі тестування доведено коректність роботи розробленого програмного забезпечення, оскільки також було реалізовано функції для перевірки точності, коректності відпрацьовування алгоритму. Ці вбудовані функції

забезпечують користувачу необмежене тестування програмного забезпечення на нових даних.

5. У даної програмної реалізації можлива велика кількість потенційних вдосконалень. Можливі імплементації нових методів прогнозування і їх порівняння. Це допоможе програмному забезпеченню стати більш гнучким у виборі моделі, оскільки різні моделі можуть по різному добре прогнозувати на різних даних. Ще одне вдосконалення пов’язане із реалізацією прогнозування у режимі реального часу. Для цього потрібно буде під’єднати програмне забезпечення безпосередньо до системи підприємства. Такий підхід забезпечить швидкість прийняття рішень та якісніше управління енергоспоживанням.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. B. Sizirici, Y. Fseha, C. S. Cho, I. Yildiz, and Y. J. Byon, “A Review of Carbon Footprint Reduction in. Construction Industry, from Design to Operation,” Materials vol. 14, no. 20, p. 6094, 2021, https://doi.org/10.3390/ma14206094.

2. Про схвалення Концепції впровадження “розумних мереж” в Україні до... : Розпорядження Кабінету Міністрів України; Концепція, План, Заходи від 14.10.2022 № 908-р. https://zakon.rada.gov.ua/go/908-2022-%D1%80

3. Elfaki, Khalid Eltayeb, Rossanto Dwi Handoyo, and Kabiru Hannafi Ibrahim. 2021. "The Impact of Industrialization, Trade Openness, Financial Development, and Energy Consumption on Economic Growth in Indonesia" Economies 9, no. 4: 174. https://doi.org/10.3390/economies9040174

4. Kao, Yu-Sheng, Kazumitsu Nawata, and Chi-Yo Huang. 2020. "Predicting Primary Energy Consumption Using Hybrid ARIMA and GA-SVR Based on EEMD Decomposition" Mathematics 8, no. 10: 1722. https://doi.org/10.3390/math8101722

5. Ngoc-Son Truong, Ngoc-Tri Ngo, Anh-Duc Pham, "Forecasting Time-Series Energy Data in Buildings Using an Additive Artificial Intelligence Model for Improving Energy Efficiency", Computational Intelligence and Neuroscience, vol. 2021, Article ID 6028573, 12 pages, 2021. https://doi.org/10.1155/2021/6028573

6. V E,Sathishkumar, Shin,Changsun, and Cho,Yongyun. (2023). Steel Industry Energy Consumption. UCI Machine Learning Repository. https://doi.org/10.24432/C52G8C.

7. Komorowski, Matthieu & Marshall, Dominic & Salciccioli, Justin & Crutain, Yves. (2016). Exploratory Data Analysis. 10.1007/978-3-319-43742-2_15.

8. Maulud, Dastan & Mohsin Abdulazeez, Adnan. (2020). A Review on Linear Regression Comprehensive in Machine Learning. Journal of Applied Science and Technology Trends. 1. 140-147. 10.38094/jastt1457.

9. Safi, Samir & Alsheryani, Mouza & Alrashdi, Maitha & Suleiman, Rawan & Abdalla, Zainab. (2023). Optimizing Linear Regression Models with Lasso and Ridge Regression: A Study on UAE Financial Behavior during COVID-19. Migration Letters. 20. 139-153. 10.59670/ml.v20i6.3468.

10. Erilli, Alp. (2021). Use of Trimean in Theil-Sen Regression Analysis. Bulletin of Economic Theory and Analysis. 6. 15-26. 10.25229/beta.827053.

11. Xu, Mengfan. (2023). Quantile Regression Model and Its Application Research. Academic Journal of Science and Technology. 8. 172-176. 10.54097/vt1qpm59.

12. S, Purna & Divya, G.. (2022). A Light Gradient Boosting Machine Regression Model for Prediction of Agriculture Insurance Cost over Linear Regression. 10.3233/APC220027.

13. Shahani, Niaz & Zheng, Xigui & Liu, Cancan & Ul Hassan, Fawad & Li, Peng. (2021). Developing an XGBoost Regression Model for Predicting Young's Modulus of Intact Sedimentary Rocks for the Stability of Surface and Subsurface Structures. Frontiers in Earth Science. 9. 1-13. 10.3389/feart.2021.761990.

14. Kocarık Gacar, Burcu & Deveci Kocakoç, Ipek. (2020). Regression Analyses or Decision Trees?. Celal Bayar Üniversitesi Sosyal Bilimler Dergisi. 18. 251-260. 10.18026/cbayarsos.796172.

15. Doz, Daniel, Mara Cotič, and Darjo Felda. 2023. "Random Forest Regression in Predicting Students' Achievements and Fuzzy Grades" Mathematics 11, no. 19: 4129. https://doi.org/10.3390/math11194129.

16. Hameed, Mohammed & Khalid, Mohamed & Khaleel, Faidhalrahman & Al-Ansari, Nadhir. (2021). An Extra Tree Regression Model for Discharge Coefficient Prediction: Novel, Practical Applications in the Hydraulic Sector and Future Research Directions. Mathematical Problems in Engineering. 2021. 1-19. 10.1155/2021/7001710.

17. Tiwari, Durgesh & Agarwal, Sudhir & Singh, Saurabh. (2023). Bagging Regressor with XGBoost Regressor as Base Estimator for Predictions in Indian Power-Energy Trading Market.

18. Qin, Yongfei & Li, Chao & Shi, Xia & Wang, Weigang. (2022). MLP-Based Regression Prediction Model For Compound Bioactivity. Frontiers in Bioengineering and Biotechnology. 10. 946329. 10.3389/fbioe.2022.946329.

19. Chicco, Davide & Warrens, Matthijs & Jurman, Giuseppe. (2021). The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. PeerJ Computer Science. 7. e623. 10.7717/peerj-cs.623.

ДОДАТОК А

Прогнозування споживання електроенергії підприємства засобами машинного навчання

Текст програми

УКР.НТУУ “КПІ імені Ігоря Сікорського”

Аркушів 9

Київ – 2024

Реалізація прогнозування споживання електроенергії підприємства засобами машинного навчання.

Програмні засоби:

– мова програмування – Python;

– застосування моделей машинного навчання – бібліотека sklearn;

– побудова графіків – бібліотека matplotlib;

– виконання математичних/логічних операцій з великими масивами даних – бібліотека NumPy;

– обробка наборів даних – бібліотека Pandas;

– створення інтерфейсу – CustomTkinter.

Файл GUI_analyse.py

```
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
import customtkinter
import pandas as pd
from PIL import Image, ImageTk
import csv
from main_analyse import main, show_graphic1, show_graphic2, show_error,
result_table

customtkinter.set_appearance_mode("System")
customtkinter.set_default_color_theme("blue")

root = customtkinter.CTk()
root.geometry("720x480")
root.title("Analyse energy consumption")
root.resizable(False, False)

bg = Image.open("grad2.png")
background_image = ImageTk.PhotoImage(bg)

canvas = tk.Canvas(root, width=720, height=480)
canvas.pack(fill="both", expand=True)
canvas.create_image(0, 0, image=background_image, anchor="nw")

button_style = ttk.Style()
button_style.theme_use('alt')
button_style.configure('Custom.TButton', font=('Ink Free', 12),
foreground='white', background='#397588', padding=10, borderwidth=0,
relief="flat")
button_style.map('Custom.TButton', background=[('active', '#2e596f')],
foreground=[('active', 'white')])

file_path1 = None
file_path2 = None
```

```
result, prediction, values, X_test = None, None, None, None

def load_csv_action1():
    global file_path1
    file_path1 = filedialog.askopenfilename(title="Оберіть перший CSV файл",
filetypes=(("CSV files", "*.csv"), ("All files", "*.*")))
    if file_path1:
        frame1.place_forget()
        frame2.place(relx=0.5, rely=0.5, anchor='center')

def load_csv_action2():
    global file_path2
    file_path2 = filedialog.askopenfilename(title="Оберіть другий CSV файл",
filetypes=(("CSV files", "*.csv"), ("All files", "*.*")))

def show_csv_content(file_path):
    if not file_path:
        messagebox.showerror("Помилка", "Не вибрано файл для відображення")
        return
    content = []
    try:
        with open(file_path, newline='', encoding='utf-8') as csvfile:
            csvreader = csv.reader(csvfile)
            content.extend(csvreader)
    except Exception as e:
        messagebox.showerror("Помилка", f"Не вдалося прочитати файл: {e}")
        return

    new_window = tk.Toplevel(root)
    new_window.title("Вміст CSV файлу")
    new_window.geometry("600x400")
    text_area = tk.Text(new_window, wrap='none')
    text_area.pack(expand=True, fill='both')
    scrollbar_y = tk.Scrollbar(new_window, orient='vertical',
command=text_area.yview)
    scrollbar_y.pack(side='right', fill='y')
    text_area.config(yscrollcommand=scrollbar_y.set)
    scrollbar_x = tk.Scrollbar(new_window, orient='horizontal',
command=text_area.xview)
    scrollbar_x.pack(side='bottom', fill='x')
    text_area.config(xscrollcommand=scrollbar_x.set)
    for row in content:
        text_area.insert('end', ','.join(row) + '\n')

def show_frame3():
    global result, prediction, values, X_test
    result, prediction, values, X_test = main(pd.read_csv(file_path1))
    frame2.place_forget()
    frame3.place(relx=0.5, rely=0.5, anchor='center')

def show_frame4():
    frame3.place_forget()
    frame4.place(relx=0.5, rely=0.5, anchor='center')

def show_frame5():
    frame4.place_forget()
    frame5.place(relx=0.5, rely=0.5, anchor='center')

def start_point():
    frame1.place(relx=0.5, rely=0.5, anchor='center')
    frame2.place_forget()
    frame3.place_forget()
    frame4.place_forget()
    frame5.place_forget()
```

```
frame1 = tk.Frame(root, bg='white', bd=0)
canvas_in_frame1 = tk.Canvas(frame1, width=500, height=250, highlightthickness=0)
canvas_in_frame1.pack(fill="both", expand=True)
canvas_in_frame1.create_text(250, 50, text="Для початку оберіть набір даних",
font=('Ink Free', 20), fill="black")
button1_1 = ttk.Button(frame1, text="Завантажити CSV файл",
command=load_csv_action1, style='Custom.TButton')
canvas_in_frame1.create_window(250, 150, window=button1_1)

frame2 = tk.Frame(root, bg='white', bd=0)
canvas_in_frame2 = tk.Canvas(frame2, width=500, height=250, highlightthickness=0)
canvas_in_frame2.pack(fill="both", expand=True)
canvas_in_frame2.create_text(250, 50, text="Набір даних успішно завантажено",
font=('Ink Free', 20), fill="black")
button2_1 = ttk.Button(frame2, text="Показати вміст CSV файлу", command=lambda:
show_csv_content(file_path1), style='Custom.TButton')
canvas_in_frame2.create_window(250, 150, window=button2_1)
button2_3 = ttk.Button(frame2, text="Аналізувати дані", command=show_frame3,
style='Custom.TButton')
canvas_in_frame2.create_window(250, 200, window=button2_3)
frame2.place_forget()

frame3 = tk.Frame(root, bg='white', bd=0)
canvas_in_frame3 = tk.Canvas(frame3, width=600, height=300, highlightthickness=0)
canvas_in_frame3.pack(fill="both", expand=True)
canvas_in_frame3.create_text(300, 50, text="Набір даних проаналізовано",
font=('Ink Free', 20), fill="black")
button3_2 = ttk.Button(frame3, text="Порівняння значень", style='Custom.TButton',
command=lambda: show_graphic1(values, prediction), width=20)
canvas_in_frame3.create_window(185, 125, window=button3_2)
button3_3 = ttk.Button(frame3, text="Розподіл відхилень", style='Custom.TButton',
command=lambda: show_graphic2(values, prediction), width=20)
canvas_in_frame3.create_window(415, 125, window=button3_3)
button3_4 = ttk.Button(frame3, text="Матриця похибок", style='Custom.TButton',
command=lambda: show_error(values, prediction, X_test), width=20)
canvas_in_frame3.create_window(185, 200, window=button3_4)
button3_5 = ttk.Button(frame3, text="Результати", style='Custom.TButton',
command=lambda: result_table(result), width=20)
canvas_in_frame3.create_window(415, 200, window=button3_5)
frame3.place_forget()

frame4 = tk.Frame(root, bg='white', bd=0)
canvas_in_frame4 = tk.Canvas(frame4, width=500, height=300, highlightthickness=0)
canvas_in_frame4.pack(fill="both", expand=True)
canvas_in_frame4.create_text(250, 50, text="Завантажте новий набір даних",
font=('Ink Free', 20), fill="black")
button4_1 = ttk.Button(frame4, text="Завантажити CSV файл",
command=load_csv_action2, style='Custom.TButton')
canvas_in_frame4.create_window(250, 150, window=button4_1)
button4_2 = ttk.Button(frame4, text="Показати вміст CSV файлу", command=lambda:
show_csv_content(file_path2), style='Custom.TButton')
canvas_in_frame4.create_window(250, 200, window=button4_2)
button4_3 = ttk.Button(frame4, text="Продовжити", command=show_frame5,
style='Custom.TButton')
canvas_in_frame4.create_window(250, 250, window=button4_3)
frame4.place_forget()

frame5 = tk.Frame(root, bg='white', bd=0)
canvas_in_frame5 = tk.Canvas(frame5, width=600, height=250, highlightthickness=0)
canvas_in_frame5.pack(fill="both", expand=True)
canvas_in_frame5.create_text(300, 50, text="Прогноз використаної електроенергії
становить:", font=('Ink Free', 20), fill="black")
canvas_in_frame5.create_text(300, 100, text="326,6 кВТ", font=('Ink Free', 20),
```

```
fill="black")
button5_1 = ttk.Button(frame5, text="Завершити", command=root.quit,
style='Custom.TButton')
canvas_in_frame5.create_window(300, 150, window=button5_1)
frame5.place_forget()

button_home = ttk.Button(root, text="На головну", command=start_point,
style='Custom.TButton')
button_home.place(relx=0.1, rely=0.07, anchor='center')

start_point()
root.mainloop()
```

Файл main_analyse.py

```
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import ExtraTreesRegressor

def day_period(date_time_str):
    hour = datetime.strptime(date_time_str, '%d/%m/%Y %H:%M').hour
    periods = ["night", "morning", "afternoon", "evening"]
    return periods[hour // 6]

def model_train(df):
    X = df.drop(['Usage_kWh', 'date'], axis=1)
    y = df['Usage_kWh']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
    model = ExtraTreesRegressor(random_state=42)
    model.fit(X_train, y_train)
    return model, X_test, y_test

def mean_absolute_percentage_error(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

def show_graphic1(y_test, y_pred):
    plt.figure(figsize=(8, 6))
    plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
    plt.xlabel('Фактичне енергоспоживання')
    plt.ylabel('Прогнозоване енергоспоживання')
    plt.title('Порівняння фактичного та прогнозованого енергоспоживання')
    plt.grid(True)
    z = np.polyfit(y_test, y_pred, 1)
    p = np.poly1d(z)
    plt.plot(y_test, p(y_test), color='red')
    plt.show()

def show_graphic2(y_test, y_pred):
    diff = y_pred - y_test
    plt.figure(figsize=(12, 6))
    hist, bins, _ = plt.hist(diff, bins=50, color='green', alpha=0.5,
edgecolor='black')
    bin_centers = 0.5 * (bins[1:] + bins[:-1])
    for count, x in zip(hist, bin_centers):
        plt.text(x, count + 0.5, f'{int(count)}', ha='center', va='bottom')
    plt.xlabel('Різниця між фактичним та прогнозованими значеннями
```

```
енергоспоживання')
    plt.ylabel('Кількість випадків')
    plt.title('Розподіл різниць між фактичним та прогнозованими значеннями')
    plt.xticks(rotation=45)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()

def show_error(y_test, y_pred, X_test):
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    mae = np.mean(np.abs(y_test - y_pred))
    r2 = r2_score(y_test, y_pred)
    mape = mean_absolute_percentage_error(y_test, y_pred)

    mse = round(mse, 3)
    rmse = round(rmse, 3)
    mae = round(mae, 3)
    r2 = round(r2, 3)
    mape = round(mape, 3)

    metrics_df = pd.DataFrame({
        'Метрика': ['Mean Squared Error', 'Root Mean Squared Error', 'Mean
Absolute Error', 'R-squared',
                        'Mean Absolute Percentage Error'],
        'Значення': [mse, rmse, mae, r2, mape]
    })
    plt.rcParams["figure.figsize"] = [10, 5]
    plt.rcParams["figure.autolayout"] = True
    fig, axs = plt.subplots(1, 1)
    axs.axis('tight')
    axs.axis('off')
    the_table = axs.table(cellText=metrics_df.values,
colLabels=metrics_df.columns, loc='center', colLoc='center',
                          cellLoc='center')
    the_table.auto_set_font_size(False)
    the_table.set_fontsize(10)
    plt.show()

def main(df):
    df['Month'] = pd.to_datetime(df['date'], format='%d/%m/%Y %H:%M').dt.month
    df['DayPeriod'] = df['date'].apply(day_period)
    column_to_categorize = ['Day_of_week', 'Load_Type', 'DayPeriod', 'WeekStatus']
    for i in column_to_categorize:
        df[i] = df[i].astype('category').cat.codes
    model, X_test, y_test = model_train(df)
    y_pred = model.predict(X_test)
    electricity_sum_predict = round(y_pred.sum(), 2)
    electricity_sum_real = round(y_test.sum(), 2)
    data = {'Фактичне енергоспоживання': [electricity_sum_real], 'Прогнозоване
енергоспоживання': [electricity_sum_predict]}
    res = pd.DataFrame(data)
    res['Відхилення'] = res['Прогнозоване енергоспоживання'] - res['Фактичне
енергоспоживання']
    res['Відсоткове відхилення, %'] = (res['Відхилення'] / res['Фактичне
енергоспоживання']) * 100
    return res, y_pred, y_test, X_test

def result_table(data):
    data1 = data.round(2)
    plt.rcParams["figure.figsize"] = [10, 2]
    plt.rcParams["figure.autolayout"] = True
    fig, axs = plt.subplots(1, 1)
    axs.axis('tight')
    axs.axis('off')
```

```
    the_table = axs.table(cellText=data1.values, colLabels=data1.columns,
loc='center', colLoc='center', cellLoc='center')
    the_table.auto_set_font_size(False)
    the_table.set_fontsize(10)
    plt.show()
```

Файл GUI_predict.py

```
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
from PIL import Image, ImageTk
import csv
import pandas as pd
from main_predict import main

root = tk.Tk()
root.geometry("720x480")
root.title("Predict energy consumption")
root.resizable(False, False)

bg = Image.open("grad2.png")
background_image = ImageTk.PhotoImage(bg)

canvas = tk.Canvas(root, width=720, height=480)
canvas.pack(fill="both", expand=True)
canvas.create_image(0, 0, image=background_image, anchor="nw")

button_style = ttk.Style()
button_style.theme_use('alt')
button_style.configure('Custom.TButton',
                       font=('Ink Free', 12),
                       foreground='white',
                       background='#397588',
                       padding=10,
                       borderwidth=0,
                       relief="flat")

button_style.map('Custom.TButton',
                 background=[('active', '#2e596f')],
                 foreground=[('active', 'white')])

file_path1 = None
file_path2 = None
y_pred = None

def load_csv_action1():
    global file_path1
    file_path1 = filedialog.askopenfilename(
        title="Оберіть CSV файл",
        filetypes=(("CSV files", "*.csv"), ("All files", "*.*"))
    )
    if file_path1:
        try:
            frame1.place_forget()
            frame2.place(relx=0.5, rely=0.5, anchor='center')
        except Exception as e:
            messagebox.showerror("Помилка", f"Не вдалося завантажити файл: {e}")

def show_csv_content1():
    show_csv_content(file_path1)
```

```
def show_csv_content(file_path):
    if not file_path:
        messagebox.showerror("Помилка", "Не вибрано файл для відображення")
        return
    content = []
    try:
        with open(file_path, newline='', encoding='utf-8') as csvfile:
            csvreader = csv.reader(csvfile)
            for row in csvreader:
                content.append(row)
    except Exception as e:
        messagebox.showerror("Помилка", f"Не вдалося прочитати файл: {e}")
        return
    new_window = tk.Toplevel(root)
    new_window.title("Вміст CSV файлу")
    new_window.geometry("600x400")
    text_area = tk.Text(new_window, wrap='none')
    text_area.pack(expand=True, fill='both')
    scrollbar_y = tk.Scrollbar(new_window, orient='vertical',
command=text_area.yview)
    scrollbar_y.pack(side='right', fill='y')
    text_area.config(yscrollcommand=scrollbar_y.set)
    scrollbar_x = tk.Scrollbar(new_window, orient='horizontal',
command=text_area.xview)
    scrollbar_x.pack(side='bottom', fill='x')
    text_area.config(xscrollcommand=scrollbar_x.set)
    for row in content:
        text_area.insert('end', ','.join(row) + '\n')

def show_csv_content2():
    show_csv_content(file_path2)

def load_csv_action2():
    global file_path2
    file_path2 = filedialog.askopenfilename(
        title="Оберіть інший CSV файл",
        filetypes=(("CSV files", "*.csv"), ("All files", "*.*"))
    )

def show_frame5():
    global y_pred
    y_pred = main(pd.read_csv(file_path1))
    canvas_in_frame5.itemconfig(prediction_text, text=str(y_pred)+" кВТ")
    frame4.place_forget()
    frame5.place(relx=0.5, rely=0.5, anchor='center')

def start_point():
    frame1.place(relx=0.5, rely=0.5, anchor='center')
    frame2.place_forget()
    frame4.place_forget()
    frame5.place_forget()

frame1 = tk.Frame(root, bg='white', bd=0)
canvas_in_frame1 = tk.Canvas(frame1, width=500, height=250, highlightthickness=0)
canvas_in_frame1.pack(fill="both", expand=True)
canvas_in_frame1.create_text(250, 50, text="Для початку оберіть набір даних",
font=('Ink Free', 20), fill="black")
button1_1 = ttk.Button(frame1, text="Завантажити CSV файл",
command=load_csv_action1, style='Custom.TButton')
canvas_in_frame1.create_window(250, 150, window=button1_1)

frame2 = tk.Frame(root, bg='white', bd=0)
canvas_in_frame2 = tk.Canvas(frame2, width=500, height=250, highlightthickness=0)
canvas_in_frame2.pack(fill="both", expand=True)
```

```
canvas_in_frame2.create_text(250, 50, text="Набір даних успішно завантажено",
font=('Ink Free', 20), fill="black")
button2_1 = ttk.Button(frame2, text="Показати вміст CSV файлу",
command=show_csv_content1, style='Custom.TButton')
canvas_in_frame2.create_window(250, 150, window=button2_1)
button2_3 = ttk.Button(frame2, text="Аналізувати дані", command=show_frame5,
style='Custom.TButton')
canvas_in_frame2.create_window(250, 200, window=button2_3)
frame2.place_forget()

frame4 = tk.Frame(root, bg='white', bd=0)
canvas_in_frame4 = tk.Canvas(frame4, width=500, height=300, highlightthickness=0)
canvas_in_frame4.pack(fill="both", expand=True)
canvas_in_frame4.create_text(250, 50, text="Завантажте новий набір даних",
font=('Ink Free', 20), fill="black")
button4_1 = ttk.Button(frame4, text="Завантажити CSV файл",
command=load_csv_action2, style='Custom.TButton')
canvas_in_frame4.create_window(250, 150, window=button4_1)
button4_2 = ttk.Button(frame4, text="Показати вміст CSV файлу",
command=show_csv_content2, style='Custom.TButton')
canvas_in_frame4.create_window(250, 200, window=button4_2)
button4_3 = ttk.Button(frame4, text="Продовжити", command=show_frame5,
style='Custom.TButton')
canvas_in_frame4.create_window(250, 250, window=button4_3)
frame4.place_forget()

frame5 = tk.Frame(root, bg='white', bd=0)
canvas_in_frame5 = tk.Canvas(frame5, width=600, height=250, highlightthickness=0)
canvas_in_frame5.pack(fill="both", expand=True)
canvas_in_frame5.create_text(300, 50, text="Прогнозоване енергоспоживання:",
font=('Ink Free', 20), fill="black")
prediction_text = canvas_in_frame5.create_text(300, 100, text=str(y_pred)+" kWh",
font=('Ink Free', 20), fill="black")
button5_1 = ttk.Button(frame5, text="Завершити", command=root.quit,
style='Custom.TButton')
canvas_in_frame5.create_window(300, 150, window=button5_1)
frame5.place_forget()

button_home = ttk.Button(root, text="На головну", command=start_point,
style='Custom.TButton')
button_home.place(relx=0.1, rely=0.07, anchor='center')

start_point()

root.mainloop()
```

Файл main_predict.py

```
import pandas as pd
from datetime import datetime
from sklearn.ensemble import ExtraTreesRegressor


def day_period(date_time_str):
    date_time_obj = datetime.strptime(date_time_str, '%d/%m/%Y %H:%M')
    hour = date_time_obj.hour

    periods = ["night", "morning", "day", "evening"]

    return periods[hour // 6]


def preprocess_data(df):
```

```
    df['Month'] = pd.to_datetime(df['date'], format='%d/%m/%Y %H:%M').dt.month
    df['DayPeriod'] = df['date'].apply(day_period)

    columns_to_categorize = ['Day_of_week', 'Load_Type', 'DayPeriod',
'WeekStatus']
    for col in columns_to_categorize:
        df[col] = df[col].astype('category').cat.codes

    return df


def model_train(df):
    X = df.drop(['Usage_kWh', 'date'], axis=1)
    y = df['Usage_kWh']

    model = ExtraTreesRegressor(random_state=42)
    model.fit(X, y)

    return model


def main(df):
    df_train = pd.read_csv('\your train data path\')
    df_train = preprocess_data(df_train)

    model = model_train(df_train)

    df = preprocess_data(df)
    df = df.drop(['date'], axis=1)
    y_pred = model.predict(df).sum()

    return y_pred
```