**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Чорноморський національний університет**
**імені Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра інтелектуальних інформаційних систем**

**ДОПУЩЕНО ДО ЗАХИСТУ**
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

**КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА**

# ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПРОГНОЗУВАННЯ ЕНЕРГОСПОЖИВАННЯ

Спеціальність 122 «Комп'ютерні науки»

**122 – КРБ – 401.22010315**

***Виконала студентка 4-го курсу, групи 401***
***_____Н. А. Сагіян***
«_21_» червня 2024 р.

***Керівник: канд. пед. наук, доцент***
***_____Н. М. Болюбаш***
«_21_» червня 2024 р.

**Миколаїв – 2024**

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Чорноморський національний університет ім. Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра інтелектуальних інформаційних систем**

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
*(шифр і назва)*
Галузь знань **12 «Інформаційні технології»**
*(шифр і назва)*

**ЗАТВЕРДЖУЮ**

Завідувач кафедри інтелектуальних інформаційних систем, д-р. техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2024 р.

# З А В Д А Н Н Я

## на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Сагіян Неллі Аветіківні.

1.Тема кваліфікаційної роботи «Інтелектуальна система прогнозування енергоспоживання».

Керівник роботи Болюбаш Надія Миколаївна, канд. пед. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2024 р. № 271

2. Строк представлення кваліфікаційної роботи студентом «21» червня 2024 р.

3. Вхідні (початкові) дані до роботи: предметна сфера енергоспоживання, методи та моделі для аналізу та прогнозування споживання електроенергії, а також набір даних із історичними показниками споживання електроенергії для домогосподарств, які обладнані розумними лічильниками.

Очікуваний результат: інтелектуальна система прогнозування енергоспоживання.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– здійснення аналізу глобальних трендів і наявних програмних засобів у сфері енергоспоживання;

– обґрунтування вибору технологій і інструментальних засобів розробки інтелектуальної системи прогнозування енергоспоживання;

– розробка, здійснення програмної реалізації системи прогнозування споживання електроенергії домогосподарствами та дослідження якості прогнозної моделі.

5. Перелік графічного матеріалу: презентація, рисунки, таблиці.

6. Завдання до спеціальної частини: Охорона праці на робочих місцях у ІТ-відділі ПАТ «Миколаївські енергетичні системи».

7. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис |
|---|---|---|
| Спеціальна частина з охорони праці | Алексєєва А. О., канд. техн. наук, доцент кафедри екології | |

Керівник роботи __канд. пед. наук, доцент Болюбаш Н. М._____
*(наук. ступінь, вчене звання, прізвище та ініціали)*

_____
*(підпис)*

Завдання прийнято до виконання ___Сагіян Н. А._____
*(прізвище та ініціали)*

_____
*(підпис)*

Дата видачі завдання «__14__» ___січня__ 2024 р.

**КАЛЕНДАРНИЙ ПЛАН**
**виконання бакалаврської кваліфікаційної роботи**

Тема: Інтелектуальна система прогнозування енергоспоживання

| № | Найменування роботи | Початок | Закінчення | Примітки |
|---|---|---|---|---|
| 1 | Визначення керівника і теми БКР. Подання заяви на затвердження теми БКР | 10.11.2023 | 15.11.2023 | Виконано |
| 2 | Отримання завдання на виконання БКР | 10.01.2024 | 15.01.2024 | Виконано |
| 3 | Складання календарного плану | 16.01.2024 | 29.01.2024 | Виконано |
| 4 | Огляд літератури за темою дослідження. Аналіз сучасних методів прогнозування енергоспоживання | 30.01.2024 | 17.02.2024 | Виконано |
| 5 | Вибір технологій та інструментальних засобів розробки системи | 18.02.2024 | 29.02.2024 | Виконано |
| 6 | Створення дизайну, проєктування та програмна реалізація, тестування | 1.03.2024 | 15.04.2024 | Виконано |
| 7 | Робота над розділами фахової частини БКР | 16.04.2024 | 31.04.2024 | Виконано |
| 8 | Проходження переддипломної практики, збір та аналіз матеріалів, остаточне оформлення розділів фахової частини БКР | 29.04.2024 | 12.05.2024 | Виконано |
| 9 | Розробка спеціальної частини з охорони праці | 13.05.2024 | 25.05.2024 | Виконано |
| 10 | Обговорення отриманих результатів з керівником та попередній захист БКР | 27.05.2024 | 29.05.2024 | Виконано |
| 11 | Корегування роботи за результатами попереднього захисту | 30.05.2024 | 6.06.2024 | Виконано |
| 12 | Другий попередній захист БКР | 10.06.2024 | 10.06.2024 | Виконано |
| 13 | Остаточне оформлення пояснювальної записки та слайдів доповіді до захисту | 11.06.2024 | 12.06.2024 | Виконано |
| 14 | Подання рецензенту та рецензування БКР | 13.06.2024 | 13.06.2024 | Виконано |
| 15 | Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту | 17.06.2024 | 21.06.2024 | Виконано |
| 16 | Захист БКР перед ЕК | 28.06.2024 | 28.06.2024 | Виконано |

Розробив студент _____Сагіян Н. А._____ _____
*(прізвище та ініціали)* *(підпис)*

Керівник _____канд. пед. наук, доцент Болюбаш Н. М___ _____
*(наук. ступінь, вчене звання, прізвище та ініціали)* *(підпис)*

«_29_» _____01_____ 2024 р.

# АНОТАЦІЯ

**кваліфікаційної роботи бакалавра**
**студентки групи 401 ЧНУ ім. Петра Могили**
**Сагіян Неллі Аветіківни**
**Тема: «Інтелектуальна система прогнозування енергоспоживання»**


Дана кваліфікаційна робота бакалавра спрямована на розробку та здійснення програмної реалізації інтелектуальної системи прогнозування електроенергії домогосподарствами. Що є актуальним в умовах динамічної цифровізації усіх сфер суспільства та зростання попиту на електроенергію. Застосування систем прогнозування енергоспоживання забезпечує стабільність у роботі енергопостачаючих підприємств, зменшує витрати та дозволяє ефективніше управляти енергопостачанням.

**Об'єкт роботи** – процес прогнозування енергоспоживання домогосподарствами.

**Предмет роботи** – програмні засоби, методи та алгоритми аналізу і прогнозування енергоспоживання.

**Мета роботи** – підвищення ефективності управління енергопостачанням шляхом розробки системи прогнозування споживання електроенергії із вбудованими методами аналізу історичних показників споживання електроенергії домогосподарствами та побудови прогнозних моделей.

Структура кваліфікаційної роботи включає фахову та спеціальну частину з охорони праці. Фахова частина включає вступ, три розділи, висновки та додатки. У першому розділі розкрито глобальні тренди та наявні програмні платформи у сфері енергоспоживання. У другому розділі обґрунтовано вибір технологій і засобів розробки системи прогнозування споживання електроенергії домогосподарствами. У третьому – описано проєктування та програмну реалізацію інтелектуальної системи прогнозування.

Кваліфікаційна робота містить 66 сторінок (без додатків), 25 рисунків, 25 джерел та 5 додатків.

**Ключові слова**: прогнозування енергоспоживання, модель Prophet, часові ряди, адитивна модель часового ряду.

# ABSTRACT

**for bachelor's qualification work**
**of a student of 401 group at Petro Mohyla Black Sea National University**
**Sahiian Nelli**
**Theme: «Intelligent energy consumption forecasting system»**

This bachelor's qualification work is aimed at the development and implementation of the software implementation of the intelligent system of electricity forecasting by households. What is relevant in the conditions of dynamic digitalization of all spheres of society and the growth of demand for electricity. The use of energy consumption forecasting systems ensures stability in the work of energy supply enterprises, reduces costs and allows more efficient management of energy supply.

**Object of work** – the process of forecasting energy consumption by households.

**Subject of work** – software tools, methods and algorithms for energy consumption analysis and forecasting.

**The purpose of this work** is to improving the efficiency of energy supply management by developing a system for forecasting electricity consumption with built-in methods of analyzing historical indicators of electricity consumption by households and building predictive models.

The structure of the qualification work includes a professional and special part on labor protection. The professional part includes an introduction, three chapters, conclusions and appendices. The first chapter reveals global trends and available software platforms in the field of energy consumption. The second section substantiates the choice of technologies and means of developing a system for forecasting electricity consumption by households. The third section describes the design and software implementation of an intelligent forecasting system.

The qualification work contains 66 pages (without appendices), 18 figures, 25 sources and 5 appendices.



**Keywords**: energy consumption forecasting, Prophet model, time series, additive model of the time series.

# ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ

AR – Autoregressive model

MACD – Moving Average Convergence Divergence

MSE – Mean Squared Error

# ВСТУП

**Актуальність**. Зростаючий глобальний інтерес до енергоефективності та зниження вуглецевого сліду обумовлює підвищення точності прогнозування попиту на елетроенергію домогосподарствами та промисловими підприємствами. Відхилення фактичного споживання електроенергії від спрогнозованого обсягу тягне за собою необхідність для енергогенеруючих підприємств продажу зайвого або закупівлі недостатнього обсягу електроенергії по завідомо невигідним цінам. Тому завдання планування та прогнозування енергоспоживання є досить значущим в електроенергетиці, і підвищення точності цього прогнозування може суттєво знизити витрати на покупку електроенергії підприємством.

В умовах високих темпів цифровізації сфери енергопостачання, перспективним напрямком підвищення ефективності планування є застосування інтелектуальних систем аналізу та прогнозування попиту на електроенергію, які дозволяють прогнозувати обсяги споживання шляхом аналізу історичних даних про споживання електроенергії домогосподарствами протягом певного періоду часу – часових рядів.

Моделювання енергоспоживання є складною задачею, оскільки попит на електроенергію містить багато чинників, які можуть викликати ріст та падіння попиту на електроенергію. Існують різні методи прогнозування енергоспоживання, включаючи нейронні мережі, регресійний аналіз, метод опорних векторів, наївного Байєса, прогнозування з використанням гібридних систем, прогнозну екстраполяцію, кореляційний та регресійний аналізи, прогнозування на основі моделей часових рядів. На вибір методів аналізу та прогнозування впливає стаціонарність та не стаціонарність часового ряду. До методів аналізу часових рядів, які базуються на припущенні про стаціонарність ряду, відносять авторегресійну модель (AR-модель) та її модифікації. Для підвищення ефективності планування у енергопостачаючих підприємствах необхідна система, яка дозволяє автоматизовано здійснювати аналіз споживання електроенергії

домогосподарствами та надає можливість отримувати більш точний прогноз на основі застосування інтелектуальних моделей прогнозування.

Це обумовило **мету роботи**, яка полягає у підвищенні ефективності управління енергопостачанням шляхом розробки системи прогнозування споживання електроенергії із вбудованими методами аналізу історичних показників споживання електроенергії домогосподарствами та побудови прогнозних моделей.

Відповідно до поставленої мети було сформульовано **завдання**:

– здійснити аналіз глобальних трендів і наявних програмних засобів у сфері енергоспоживання;

– обґрунтувати вибір технологій та інструментальних засобів розробки інтелектуальної системи прогнозування енергоспоживання;

– розробити, здійснити програмну реалізацію системи прогнозування споживання електроенергії домогосподарствами та дослідити якість прогнозної моделі.

**Об’єкт роботи** – процес прогнозування енергоспоживання домогосподарствами.

**Предмет роботи** – програмні засоби, методи та алгоритми аналізу і прогнозування енергоспоживання.

**Методологічною основою** дослідження є загальнонаукові та статистично-аналітичні методи, які дозволили комплексно вивчити предмет та об’єкт дослідження, дослідити основні підходи до прогнозування споживання електроенергії домогосподарствами.

**Практичне значення** отриманих результатів полягає в тому, що використання розробленої системи дозволить підвищити ефективність управління енергоспоживанням та забезпечити стабільність у роботі енергопостачаючих підприємств.

**Структура кваліфікаційної роботи.** Відповідно до мети, завдань і предмета дослідження, бакалаврська робота містить основну та спеціальну частини. Основна

частина роботи складається із вступу, трьох розділів, висновку, списку використаних джерел та 5 додатків. Загальний обсяг роботи – 95 сторінок, із них основного тексту основної частини – 66 сторінок, спеціальної – 20 сторінок. Кількість використаних джерел – 25.

# 1 ТЕОРЕТИЧНІ ЗАСАДИ АНАЛІЗУ СФЕРИ ЕНЕРГОСПОЖИВАННЯ

## 1.1 Глобальні тренди та світовий попит на електроенергію

Попит на електроенергію змінюється в залежності від регіону і економічного стану країн. В 2023 році глобальний попит помірно зріс, але прогнозується більш швидке зростання до 2026 року. Попит в розвинених країнах падав через економічні складнощі, тоді як у країнах, що розвиваються, він зростав завдяки широкомасштабній електрифікації та збільшенню промислового виробництва. Світовий попит на електроенергію зріс на 2,2% у 2023 році у 2023 році, що менше, ніж зростання на 2,4%, яке спостерігалося у 2022 році [1]. У той час як Китай, Індія та численні країни Південно-Східної Азії продемонстрували стійке зростання попиту на електроенергію у 2023 році. У 2023 році, у країнах з розвиненою економікою спостерігалося значне зниження попиту на електроенергію через несприятливого макроекономічного середовища та високої інфляції, що призвело до скорочення виробництва та промислового виробництва. Очікується, що світовий попит на електроенергію зростатиме швидшими темпами протягом наступних трьох років, зростаючи в середньому на 3,4% щорічно до 2026 року. Зростання буде зумовлений покращенням економічних перспектив, що сприятиме швидшому зростанню попиту на електроенергію зростання попиту на електроенергію як у розвинених країнах, так і в країнах, що розвиваються.

Споживання електроенергії центрами обробки даних, штучним інтелектом (ШІ) та криптовалют може подвоїтися до 2026 року. Центри обробки даних є важливими драйверами зростання попиту на електроенергію в багатьох регіонах. Після глобального споживання приблизно 460 ТВт/год у 2022 році, загальне споживання електроенергії центрами обробки даних у 2026 році, загальне споживання електроенергії центрами обробки даних може сягнути понад 1 000 ТВт/год. Цей попит приблизно еквівалентний споживанню електроенергії в Японії.

Завдання планування та прогнозування енергоспоживання завжди вважалося важливим для процесів закупівлі та реалізації електроенергії. У зв'язку з реформами в енергетики останніми роками вона стала дуже гостро. Світове споживання електроенергії демонструє високі темпи зростання до 2026 року (рис 1.1). Країни, що розвиваються, є рушіями зростання світового попиту на електроенергію. Світовий попит на електроенергію зріс на відносно скромні 2,2% у 2023 році порівняно з 2,4% у 2022 році. Однак, зростання прискориться до більш високих 3,4% у прогнозі на 2024-2026 роках, при цьому ринки, що розвиваються, як і раніше, домінуватимуть у зростаючому попиті на електроенергію, як це було у 2023 році.

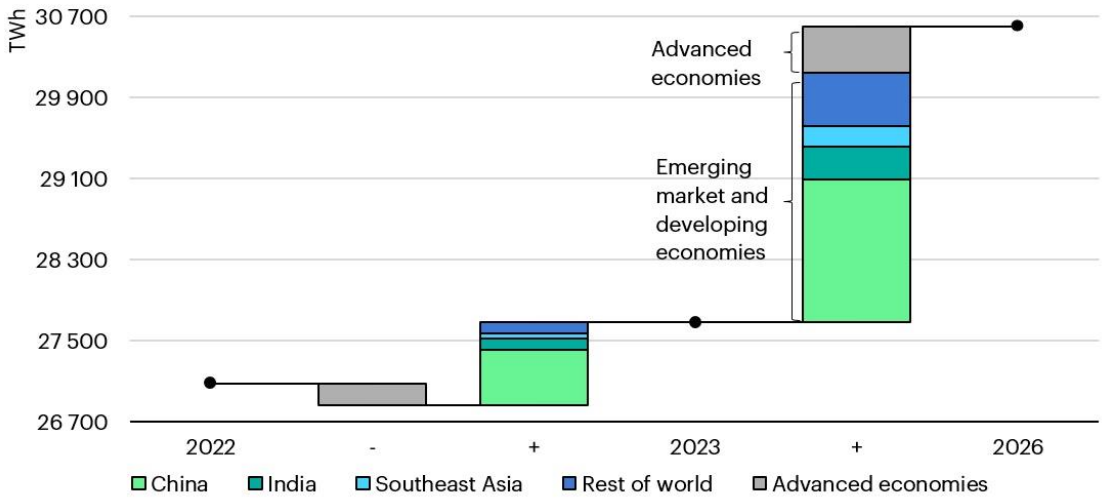



Рисунок 1.1 – Зміна попиту на електроенергію за регіонами, 2022-2026 роки

Далекосяжні наслідки енергетичної кризи продовжували відчуватися протягом усього 2023 року, що супроводжувалося підвищеним рівнем інфляції, високими процентними ставками та відсоткові ставки та важкий борговий тягар чинили негативний тиск на економіку по всьому світу. Утім, країни з ринками, що формуються, фіксували значне зростання попиту на електроенергію. Натомість у більшості розвинених країн спостерігався спад на тлі несприятливого макроекономічного середовища, а також слабкої промисловості та виробничих секторів, незважаючи на триваючу електрифікацію. М'якша погода порівняно з

попереднім роком порівняно з попереднім роком, також чинила знижувальний тиск на споживання електроенергії в деяких регіонах, включаючи США.

Близько 85% додаткового попиту на електроенергію до 2026 року буде забезпечено за межами країн з розвиненою економікою, причому Китай зробить значний внесок, навіть незважаючи на те, що економіка країни зазнає структурних змін. У 2023 році попит на електроенергію в Китаї зріс на 6,4% за рахунок сектору послуг та промисловості. Оскільки очікується, що економічне зростання країни сповільниться і стане менш залежним від важкої промисловості, за прогнозами, темпи зростання попиту на електроенергію в Китаї знизяться до 5,1% у 2024 році, 4,9% у 2025 році та 4,7% у 2026 році. Незважаючи на це, загальний приріст попиту на електроенергію в Китаї до 2026 року складе близько 1 400 ТВт/год, що більше половини поточного річного споживання електроенергії в Європейському Союзі. Споживання електроенергії на душу населення в Китаї вже перевищило показник Європейського Союзу наприкінці 2022 року і продовжить зростати. Швидко зростаюче виробництво сонячних фотоелектричних модулів та електромобілів, а також переробка супутніх матеріалів підтримуватимуть постійне зростання попиту на електроенергію в Китаї в той час, як структура його економіки буде розвиватися.

Китай забезпечує найбільшу частку зростання світового попиту на електроенергію з точки зору обсягу, але Індія демонструє найшвидші темпи зростання до 2026 року серед великих економік. Після збільшення попиту на електроенергію в Індії на 7% у 2023 році очікується, що до 2026 року він зростатиме в середньому на 6% щорічно, підтримуваний високою економічною активністю та збільшенням кількості власників кондиціонерів. Протягом наступних трьох років Індія збільшить попит на електроенергію приблизно до рівня поточного споживання Сполученого Королівства. Хоча відновлювані джерела енергії мають задовольнити майже половину цього зростання попиту, очікується, що третина буде забезпечена за рахунок зростання вугільної генерації. У Південно-Східній Азії спостерігатиметься стійке щорічне зростання попиту на електроенергію в середньому на 5% до 2026 року, що буде зумовлено високою економічною

активністю.

У той час як споживання електроенергії на душу населення в Індії та Південно-Східній Азії стрімко зростає, в Африці воно фактично перебуває в стагнації вже понад три десятиліття. В останні роки споживання на душу населення в Африці навіть знизилося, оскільки населення зростало швидше, ніж забезпечувалося постачання електроенергії, і очікується, що воно відновиться до рівня 2010-2015 років не раніше кінця 2026 року. Тридцять років тому людина в Африці споживала в середньому більше електроенергії, ніж житель Індії чи Південно-Східної Азії. Однак значне зростання попиту та пропозиції на електроенергію в Індії та Південно-Східній Азії в останні десятиліття, яке супроводжувалося бумом економічного розвитку, трансформувало ці регіони вражаючими темпами. Тим часом, споживання електроенергії на душу населення в Африці у 2023 році було вдвічі меншим, ніж в Індії, і на 70% нижчим, ніж у Південно-Східній Азії. Прогноз для Африки на період 2024-26 років передбачає середньорічне зростання загального попиту на електроенергію на 4%, що вдвічі перевищує середні темпи зростання, які спостерігалися у 2017-2023 роках. Дві третини цього зростання попиту буде забезпечено за рахунок розширення використання відновлюваних джерел енергії, а решта – переважно за рахунок природного газу.

Попит на електроенергію в США впав на 1,6% у 2023 році після зростання на 2,6% у 2022 році, але очікується, що він відновиться у прогнозному періоді 2024-26 років. Основною причиною падіння була м'якша погода у 2023 році порівняно з 2022 роком, хоча уповільнення темпів зростання у виробничому секторі також було одним із факторів. Прогнозується помірне зростання попиту на 2,5% у 2024 році за умови повернення до середніх погодних умов. За цим послідує зростання в середньому на 1% у 2025-26 роках, зумовлене електрифікацією та розширенням сектору центрів обробки даних, на який, як очікується, припадатиме понад третина додаткового попиту до 2026 року.

У 2023 році попит на електроенергію в Європейському Союзі знизився

другий рік поспіль, навіть незважаючи на те, що ціни на енергоносії впали з рекордно високих рівнів. Після падіння на 3,1% у 2022 році, падіння попиту в ЄС на 3,2% у річному обчисленні у 2023 році означало, що він впав до рівня, який востаннє спостерігався два десятиліття тому. Як і в 2022 році, основним фактором зниження попиту на електроенергію було слабке споживання в промисловому секторі, оскільки ціни на енергоносії знизилися, але залишилися вище допандемічного рівня. У 2023 році також спостерігалися ознаки деякого стійкого падіння попиту, особливо в енергоємних секторах хімічної промисловості та виробництва первинних металів. Очікується, що споживання електроенергії в ЄС повернеться до рівня 2021 року не раніше 2026 року (рис 1.2).

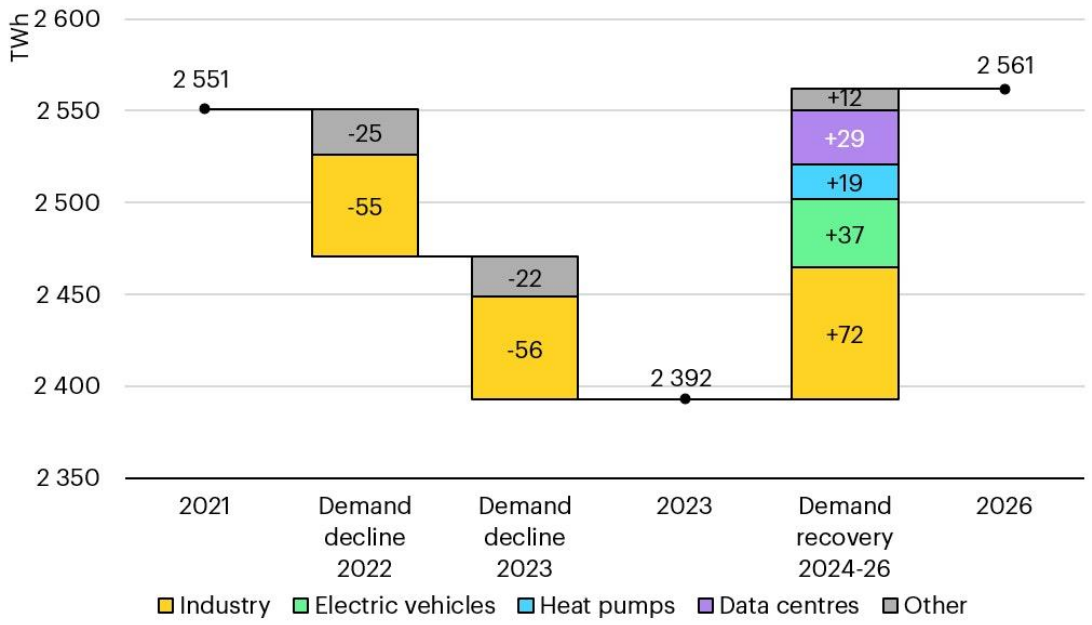



Рисунок 1.2 – Оціночні фактори зміни попиту на електроенергію в Європейському Союзі, 2021-2026 рр.

Попит на електроенергію в промисловому секторі Європейського Союзу впав приблизно на 6% у 2023 році після аналогічного падіння у 2022 році. Якщо

припустити, що промисловий сектор поступово відновлюватиметься в міру зниження цін на енергоносії, зростання попиту на електроенергію в ЄС зросте в середньому на 2,3% у 2024-26 роках. Електромобілі, теплові насоси та центри обробки даних залишатимуться сильними стовпами зростання протягом цього періоду – разом вони забезпечать половину очікуваного приросту загального попиту.

Рекордні обсяги виробництва електроенергії з низьковуглецевих джерел, до яких належать атомна енергетика та відновлювані джерела енергії, такі як сонячна, вітрова та гідроенергетика, мають покрити весь світовий попит протягом наступних трьох років (рис 1.3).

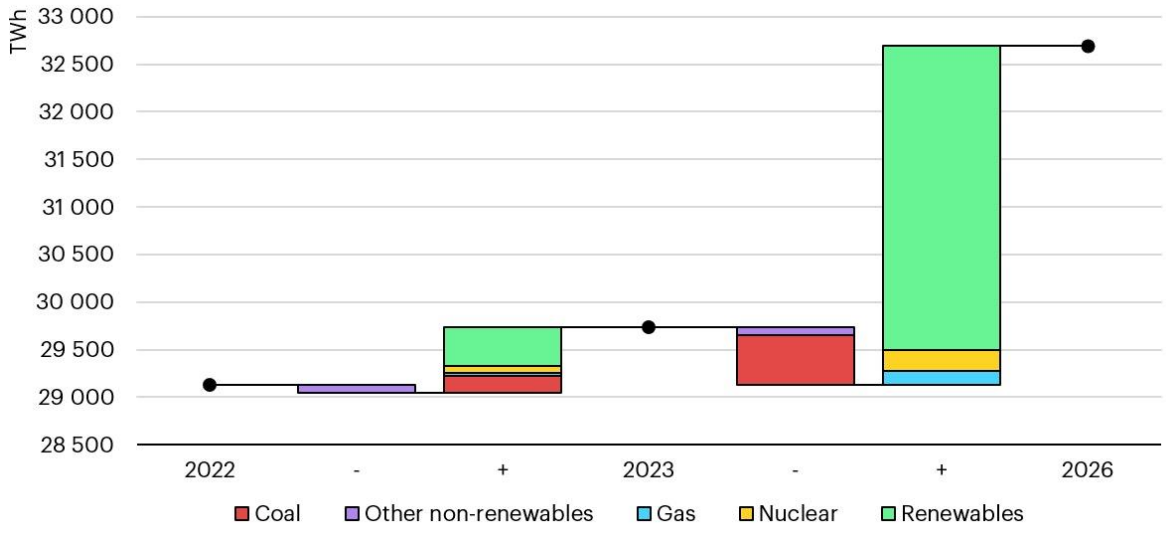



Рисунок 1.3 – Зміни у світовому виробництві електроенергії, 2022-2026 роки

Прогнозується, що до 2026 року на джерела з низьким рівнем викидів, які зменшать роль викопного палива у виробництві електроенергії в усьому світі, припадатиме майже половина світового виробництва електроенергії, порівняно з 39% у 2023 році. Протягом наступних трьох років виробництво електроенергії з низьким рівнем викидів зростатиме вдвічі швидше, ніж у 2018-2023 роках, що є закономірною зміною, враховуючи, що сьогодні енергетичний сектор робить найбільший внесок у глобальні викиди двоокису вуглецю (CO2).

До початку 2025 року відновлювані джерела енергії забезпечуватимуть понад третину загального обсягу виробництва електроенергії у світі, випередивши вугільну енергетику. Прогнозується, що частка ВДЕ у виробництві електроенергії зросте з 30% у 2023 році до 37% у 2026 році, причому це зростання значною мірою підтримуватиметься розширенням використання все більш дешевих сонячних фотоелектричних установок. Протягом цього періоду відновлювані джерела енергії більш ніж компенсують зростання попиту в розвинених країнах, таких як США та Європейський Союз, витісняючи пропозицію викопного палива. У той же час у Китаї очікується, що швидке розширення відновлюваних джерел енергії задовольнить весь додатковий попит на електроенергію, хоча погодні умови та ступінь зниження темпів зростання попиту в країні залишаються ключовими джерелами невизначеності для прогнозу. Значне розширення потужностей відновлюваної енергетики повинно також супроводжуватися прискореними інвестиціями в мережі та гнучкість системи для забезпечення її безперешкодної інтеграції.

Швидке зростання відновлюваної енергетики, підтримане зростанням атомної генерації, витіснить світову вугільну генерацію, яка, за прогнозами, буде скорочуватися в середньому на 1,7% щорічно до 2026 року. Основним фактором, що визначатиме глобальні перспективи, є розвиток тенденцій у Китаї, де виробляється більше половини світового обсягу вугільної генерації. Вугільна генерація в Китаї наразі перебуває на шляху до повільного структурного спаду, зумовленого активним розвитком відновлюваних джерел енергії та зростанням атомної генерації, а також помірним економічним зростанням.

Незважаючи на введення в експлуатацію нових електростанцій для підвищення безпеки енергопостачання, очікується, що коефіцієнт використання вугільних електростанцій в Китаї продовжить падати, оскільки вони використовуються більш гнучко, доповнюючи відновлювані джерела енергії. Тим не менш, на вугільну генерацію в Китаї значною мірою впливатимуть темпи

ребалансування економіки, тенденції в гідроенергетиці та вузькі місця в інтеграції відновлюваних джерел енергії в енергосистему країни.

Одним з ключових трендів в енергетичному секторі є перехід до низьковуглецевої економіки та збільшення частки відновлюваних джерел енергії у світовому енергобалансі. Зростання занепокоєння щодо зміни клімату та необхідність скорочення викидів парникових газів сприяють активному розвитку та впровадженню технологій відновлюваної енергетики, таких як сонячна, вітрова та гідроенергетика. Згідно з прогнозами МЕА, частка відновлюваних джерел у світовому виробництві електроенергії зросте з 26% у 2018 році до 44% у 2040 році [2].

Ще одним важливим трендом є розвиток розподіленої генерації та інтелектуальних енергетичних систем. Поширення малих генеруючих потужностей, таких як сонячні панелі на дахах будинків та мікро турбіни, змінює традиційну модель централізованого виробництва та розподілу електроенергії. Розвиток інтелектуальних мереж (smart grids) та технологій зберігання енергії дозволяє ефективно інтегрувати розподілену генерацію в енергосистему та оптимізувати управління попитом на електроенергію.

Цифрова трансформація енергетичного сектору також набирає обертів. Застосування технологій Інтернету речей (IoT), аналізу великих даних та штучного інтелекту дозволяє підвищити ефективність виробництва, передачі та споживання електроенергії. Розумні лічильники та системи управління енергоспоживанням дають можливість споживачам відстежувати своє енергоспоживання в режимі реального часу та оптимізувати його відповідно до цінових сигналів.

Електрифікація транспорту також є важливим трендом, який впливає на попит на електроенергію. Зростання популярності електромобілів та розвиток інфраструктури зарядних станцій створюють додаткове навантаження на енергосистему, особливо в періоди пікового попиту. Водночас, електромобілі можуть виступати як розподілені системи зберігання енергії та надавати послуги з балансування мережі через технології Vehicle-to-Grid (V2G).

Енергоефективність залишається ключовим напрямком у зниженні попиту на електроенергію та досягненні цілей сталого розвитку. Впровадження енергоефективних технологій та практик у різних секторах економіки, таких як будівництво, промисловість та транспорт, дозволяє зменшити енергоспоживання без шкоди для економічного зростання та якості життя.

Глобальна пандемія COVID-19 мала значний вплив на енергетичний сектор. Скорочення економічної активності та обмеження пересування призвели до тимчасового зниження попиту на електроенергію в багатьох країнах. Однак довгострокові наслідки пандемії для енергетичної галузі ще належить оцінити.

Отже, світовий попит на електроенергію продовжує зростати під впливом глобальних трендів, таких як зростання населення, урбанізація та технологічний прогрес. Водночас, енергетичний сектор зазнає трансформацій, пов'язаних з переходом до низьковуглецевої економіки, розвитком відновлюваних джерел енергії, розподіленої генерації та інтелектуальних енергетичних систем. Ці тенденції створюють нові виклики та можливості для забезпечення надійного, доступного та сталого енергопостачання в майбутньому.

## 1.2 Огляд та аналіз наявних систем прогнозування

В умовах зростаючого попиту на електроенергію та необхідності ефективного управління енергетичними ресурсами, системи прогнозування енергоспоживання стають все більш важливими інструментами для енергетичних компаній, операторів мереж та інших учасників ринку. Точні прогнози дозволяють оптимізувати виробництво, розподіл та споживання електроенергії, знизити витрати та підвищити надійність енергопостачання.

Системи прогнозування енергоспоживання – це інструменти, що використовують аналіз даних та різноманітні моделі, включаючи машинне навчання та штучний інтелект, для передбачення обсягів електроенергії, які будуть спожиті в майбутньому. Ці системи дозволяють ефективніше управляти

енергопостачанням та забезпечити стабільність мережі, а також зменшити витрати та покращити економічну ефективність. Основні функції систем прогнозування енергоспоживання включають:

– прогнозування попиту на електроенергію в різний час;

– аналіз впливу різних факторів на енергоспоживання, таких як погода, сезонність, добові коливання тощо;

– оптимізація виробництва та постачання електроенергії для забезпечення потреб споживачів;

– прогнозування витрат на енергію та розробка стратегій енергоефективності.

Такі системи широко використовуються як у домашніх умовах, так і у промисловості та комерційних підприємствах для забезпечення стійкого та ефективного управління енергією.

На сьогоднішній день існує широкий спектр методів та підходів до прогнозування енергоспоживання, які можна розділити на кілька основних категорій:

1. Традиційні статистичні методи, такі як авторегресійні моделі (AR), моделі ковзного середнього (MA), авторегресійні моделі з ковзним середнім (ARMA) та сезонні ARIMA (SARIMA) [3]. Ці методи базуються на аналізі історичних даних про енергоспоживання та виявленні статистичних закономірностей і трендів. Вони прості в реалізації та інтерпретації, але можуть бути менш ефективними для моделювання складних нелінійних залежностей та врахування зовнішніх факторів.

2. Методи машинного навчання, такі як штучні нейронні мережі (ANN), метод опорних векторів (SVM), дерева рішень та випадкові ліси (Random Forest) [4]. Ці методи здатні моделювати складні нелінійні залежності та враховувати вплив різноманітних факторів, таких як погодні умови, соціально-економічні показники та календарні ефекти. Вони потребують достатньо великих обсягів історичних даних для навчання моделей та можуть бути обчислювально затратними.

3. Гібридні методи, які поєднують переваги статистичних та машинного навчання підходів. Наприклад, комбінація ARIMA та ANN дозволяє враховувати як лінійні, так і нелінійні залежності в даних [5]. Гібридні моделі часто демонструють кращу точність прогнозування порівняно з окремими методами.

4. Методи глибокого навчання, такі як згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та довга короткочасна пам'ять (LSTM) [6]. Ці методи здатні автоматично виявляти складні просторово-часові залежності в даних та показують високу ефективність у задачах прогнозування часових рядів. Вони потребують великих обсягів даних та значних обчислювальних ресурсів для навчання моделей.

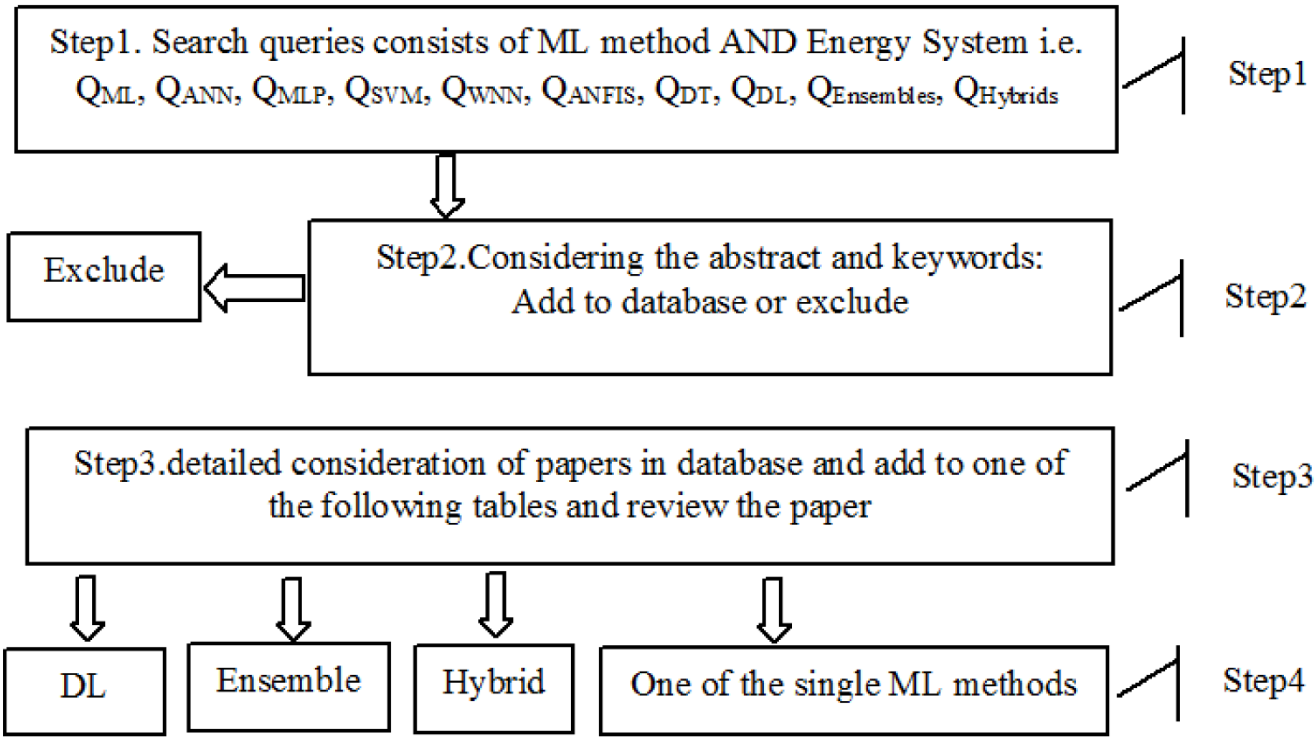



Рисунок 1.4 – Порівняння методів прогнозування енергоспоживання [7]

На рисунку 1.4 наведено порівняння різних методів прогнозування енергоспоживання на прикладі набору даних з погодинним енергоспоживанням в житловому секторі.

Серед наявних систем прогнозування можна виділити описати нижче типи систем.

1. Комерційні програмні продукти, такі як SAS Energy Forecasting, IBM Demand Forecasting for Utilities, Itron Forecasting, та Schneider Electric EcoStruxure Demand Forecasting [8]. Ці системи пропонують широкий набір функцій для збору та обробки даних, побудови моделей прогнозування та візуалізації результатів. Вони призначені для використання енергетичними компаніями та операторами мереж і можуть бути інтегровані з існуючими системами управління енергоспоживанням.

2. Відкриті програмні рішення та бібліотеки, такі як Prophet (Facebook), OpenForecast, Darts та sktime [9]. Ці інструменти надають можливість будувати та налаштовувати моделі прогнозування з використанням різних методів машинного навчання та статистичного аналізу. Вони є безкоштовними та мають активну спільноту розробників і користувачів.

3. Хмарні платформи для аналітики та прогнозування, такі як Amazon Forecast, Google Cloud AI Platform, Microsoft Azure Machine Learning та IBM Watson Studio [10]. Ці платформи надають готові рішення та інструменти для збору, обробки та аналізу даних, а також для побудови та розгортання моделей прогнозування. Вони дозволяють швидко створювати та масштабувати системи прогнозування без необхідності в розгортанні власної інфраструктури.

Є програмні рішення, інтегровані з системами управління енергоспоживанням. Зробимо їх огляд та виявимо їхні можливості, переваги та недоліки.

*Tendril* – компанія, яка надає рішення для прогнозування енергоспоживання, використовуючи аналіз даних та машинне навчання (рис 1.5). Їх платформа дозволяє споживачам та постачальникам ефективно прогнозувати своє енергоспоживання на основі різних факторів, таких як погода, час доби, сезонність тощо. Переваги:

– широкий спектр функціональності, який дозволяє адаптуватися до різних

потреб споживачів та постачальників;

– використання передових технологій аналізу даних та машинного навчання, що дозволяє точно прогнозувати енергоспоживання;

– гнучкість налаштування системи залежно від специфічних вимог клієнта.

Недоліки:

– високі витрати на впровадження та підтримку системи, що може бути обтяжливим для менших компаній або домогосподарств;

– потреба у великій кількості даних для ефективного навчання моделей прогнозування, що може бути складним для нових користувачів.


Рисунок 1.5 – Система Tendril

*Bidgely* – компанія, яка спеціалізується на аналізі енергоспоживання в домашніх умовах (рис 1.6). Вони надають рішення для прогнозування енергоспоживання та рекомендації з енергоефективності, що допомагає споживачам зменшити витрати на електроенергію.

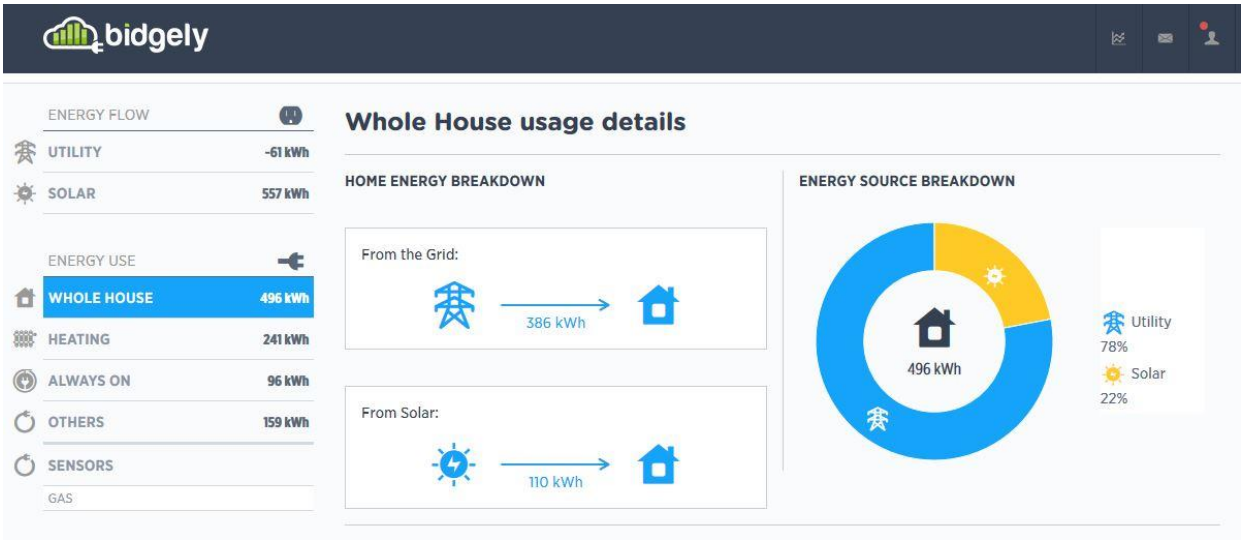



Рисунок 1.6 – Система Bidgely

Переваги:

– легкий використання та інтуїтивний інтерфейс для кінцевих користувачів, що дозволяє швидко розуміти та використовувати дані з прогнозування енергоспоживання;

– персоналізовані рекомендації з енергоефективності, що допомагають споживачам зменшити свої витрати на електроенергію;

– можливість інтеграції з різними типами інтелектуальних лічильників та систем автоматизації будинку.

Недоліки:

– обмежена гнучкість та можливості кастомізації системи, що може не влаштовувати певні види клієнтів зі специфічними потребами;

– можливість перерв у роботі системи при низькій якості сигналу або проблемах з підключенням до мережі.

*AutoGrid* – компанія, яка надає платформу для прогнозування та управління енергоспоживанням (рис 1.7). Їх рішення дозволяють постачальникам електроенергії оптимізувати роботу мережі та забезпечувати надійність постачання електроенергії за допомогою точних прогнозів споживання.


Рисунок 1.7 – Система AutoGrid

Переваги:

– розширені можливості прогнозування енергоспоживання, включаючи прогнозування навантаження та виробництва електроенергії з відновлюваних джерел;

– широкий діапазон інтеграції з іншими системами управління енергією та моніторингу мережі;

– можливості адаптації та масштабування системи відповідно до потреб користувача.

Недоліки:

– складність налаштування та інтеграції системи у вже існуючу інфраструктуру може бути вимагає додаткових зусиль та ресурсів;

– високі витрати на впровадження та підтримку системи, що може бути обтяжливим для деяких організацій або домогосподарств.

Незважаючи на наявність різноманітних систем прогнозування енергоспоживання, існують певні виклики та обмеження, які необхідно враховувати при їх впровадженні та використанні. Розглянемо їх детальніше.

1. *Якість та доступність даних*. Точність прогнозування значною мірою залежить від якості та повноти вхідних даних. Збір, очищення та попередня обробка даних з різних джерел (лічильники, метеорологічні станції, соціально-економічні показники) може бути складним завданням. Крім того, деякі дані можуть бути недоступними або містити пропуски та викиди.

2. *Вибір відповідної моделі та налаштування гіперпараметрів*. Кожен метод прогнозування має свої переваги та обмеження, і вибір найбільш підходящої моделі залежить від специфіки даних та цілей прогнозування. Налаштування гіперпараметрів моделі, таких як кількість нейронів в шарах нейронної мережі або глибина дерева рішень, може суттєво впливати на точність прогнозування та вимагає досвіду та експериментів.

3. *Інтерпретація результатів та оцінка невизначеності*. Деякі методи прогнозування, особливо ті, що базуються на глибокому навчанні, можуть бути складними для інтерпретації та пояснення. Це ускладнює розуміння того, які фактори впливають на прогноз та як модель приймає рішення. Крім того, важливо оцінювати невизначеність прогнозів та надавати довірчі інтервали для прийняття обґрунтованих рішень.

4. *Адаптація до змін та оновлення моделей*. Енергоспоживання може змінюватися з часом під впливом різних факторів, таких як зміни в поведінці споживачів, впровадження нових технологій та зміни в регуляторній політиці. Системи прогнозування повинні бути здатними адаптуватися до цих змін та періодично оновлювати моделі на основі нових даних.

Отже, наявні системи прогнозування енергоспоживання пропонують широкий спектр методів та інструментів для побудови точних та надійних прогнозів. Вибір відповідної системи залежить від специфіки задачі, доступності даних та ресурсів, а також від вимог до точності та інтерпретації результатів. Незалежно від обраного підходу, ефективне впровадження та використання систем прогнозування вимагає комплексного підходу, який враховує якість даних, вибір та налаштування моделей, інтерпретацію результатів та адаптацію до змін

## 1.3 Постановка задачі

Розробка інтелектуальної системи прогнозування енергоспоживання домогосподарств дозволяє надавати точні прогнози, які є критично важливими для успішної діяльності енергопостачаючих підприємств.

**Об’єкт роботи** – процес прогнозування енергоспоживання домогосподарствами.

**Предмет роботи** – програмні засоби, методи та алгоритми аналізу і прогнозування енергоспоживання.

**Мета роботи** – підвищення ефективності управління енергопостачанням шляхом розробки системи прогнозування споживання електроенергії із вбудованими методами аналізу історичних показників споживання електроенергії домогосподарствами та побудови прогнозних моделей.

Для досягнення поставленої мети було поставлено такі **завдання**:

– здійснити аналіз глобальних трендів і наявних програмних засобів у сфері енергоспоживання;

– обґрунтувати вибір технологій та інструментальних засобів розробки інтелектуальної системи прогнозування енергоспоживання;

– розробити, здійснити програмну реалізацію системи прогнозування споживання електроенергії домогосподарствами та дослідити якість прогнозної моделі.

Виконання перерахованих вище завдань потребує:

– визначення вимог до системи прогнозування енергоспоживання, включаючи функціональні та нефункціональні вимоги, а також вимоги до вхідних даних та очікуваних результатів;

– обґрунтування вибору прогнозної моделі для створення системи та інструментальних засобів її розробки;

– збору та підготовки набору даних, який повинен містити історичні дані про енергоспоживання домогосподарств, а також додаткові фактори, які можуть

впливати на споживання електроенергії, такі як погодні умови, час доби, день тижня тощо;

– проведення попередньої обробки та очищення даних, включаючи обробку пропущених значень, дублікатів та масштабування даних у разі потреби;

– розробки вебзастосунку для забезпечення зручної взаємодії користувачів із системою прогнозування, який повинен мати інтуїтивно зрозумілий інтерфейс для введення вхідних даних, відображення результатів прогнозування та візуалізації даних;

– реалізації функціоналу вебзастосунку, включаючи завантаження даних користувачем, передачу даних до моделі прогнозування, отримання результатів прогнозування та їх відображення у зручному для сприйняття вигляді;

– забезпечення належного рівня безпеки та конфіденційності даних користувачів у вебзастосунку.

Ці вимоги є основою для подальшого проєктування та реалізації системи.

## Висновки до розділу 1

В даному розділі кваліфікаційної роботи досліджено зростаючий глобальний попит на електроенергію та загальні тренди у сфері енергетики, аналіз енергоспоживання стає ключовим елементом для розвитку ефективних стратегій енергозбереження та сталого розвитку. У цьому розділі було розглянуто світовий попит на електроенергію, глобальні тренди та розвиток екологічно чистої електроенергії. Глобальні тренди, такі як зростання використання відновлюваних джерел енергії та прагнення до енергоефективності, впливають на структуру та споживчі патерни електроенергії. Крім того, розвиток технологій у сфері виробництва та постачання електроенергії сприяє поширенню екологічно чистих методів та ресурсів.

Аналіз показав, що світовий попит на електроенергію продовжує зростати, особливо в контексті розвитку країн з великою населенням та швидкою

індустріалізацією. Водночас, спостерігаються значні зміни в структурі енергетичного сектору, пов'язані з переходом до низьковуглецевої економіки, збільшенням частки відновлюваних джерел енергії та розвитком розподіленої генерації. Ці тенденції створюють нові виклики для енергетичної галузі та вимагають впровадження інноваційних рішень для оптимізації виробництва, розподілу та споживання електроенергії.

Огляд та аналіз існуючих систем прогнозування енергоспоживання показав, що на сьогоднішній день існує широкий спектр методів та підходів, які можуть бути застосовані для вирішення цієї задачі. Традиційні статистичні методи, такі як авторегресійні моделі та моделі ковзного середнього, є простими в реалізації, але можуть бути недостатньо ефективними для моделювання складних нелінійних залежностей. Методи машинного навчання, такі як штучні нейронні мережі та метод опорних векторів, здатні враховувати більш складні залежності, але потребують великих обсягів даних для навчання. Особливу увагу привертають методи глибокого навчання, які показують високу ефективність у задачах прогнозування часових рядів.

Аналіз наявних систем прогнозування енергоспоживання виявив, що існують як комерційні програмні продукти, так і відкриті рішення та бібліотеки, які можуть бути використані для розробки власних систем. Крім того, хмарні платформи для аналітики та прогнозування надають готові інструменти та сервіси для швидкого розгортання та масштабування систем прогнозування. Однак, незважаючи на наявність різноманітних рішень, залишаються певні виклики, пов'язані з якістю даних, вибором оптимальних моделей, інтерпретацією результатів та адаптацією до змін.

На основі проведеного аналізу було сформульовано постановку задачі для даної кваліфікаційної роботи, яка полягає в розробці інтелектуальної системи прогнозування енергоспоживання домогосподарствами. Визначено основні вимоги до системи, включаючи функціональні та нефункціональні вимоги, а також вимоги до вхідних даних та очікуваних результатів.

Враховуючи ці фактори, важливо розробляти та вдосконалювати системи аналізу енергоспоживання, що дозволить ефективно використовувати електроенергію, зменшити негативний вплив на навколишнє середовище та сприяти сталому розвитку суспільства. Розробка інтелектуальної системи прогнозування енергоспоживання домогосподарствами є актуальною та перспективною задачею, яка має практичне значення для енергетичної галузі та споживачів електроенергії. Реалізація поставлених завдань та досягнення очікуваних результатів дозволить створити ефективний інструмент для прогнозування енергоспоживання, який відповідає сучасним викликам та потребам ринку.

# 2 ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ

## 2.1 Середовище розробки Spider

Вибір відповідного середовища розробки є важливим аспектом у процесі створення програмного забезпечення, оскільки воно значною мірою впливає на ефективність, продуктивність та зручність роботи розробника. Для розробки інтелектуальної системи прогнозування енергоспоживання було обрано середовище розробки Spider.

IDE Spyder – це відкрите інтегроване середовище розробки з відкритим вихідним кодом, спеціально розроблене для мови програмування Python [11]. Spyder був створений для вчених, інженерів та аналітиків даних, що працюють з науковими бібліотеками Python, такими як NumPy, SciPy, Matplotlib, Pandas, IPython, SymPy та Cython. Воно надає повний набір інструментів і функцій для підтримки розробки на Python, включаючи редагування коду, налагодження та управління проектами.

IDE Spyder пропонує багато корисних функцій для розробки на Python:

– багатофункціональний редактор: Spyder має багатофункціональний редактор коду з підсвічуванням синтаксису, автозаповнення коду, навігацією по коду та іншими функціями, які полегшують написання коду на Python;

– інтерактивна консоль: Spider включає інтерактивну консоль Python, яка дозволяє розробникам виконувати фрагменти коду Python і переглядати результати в режимі реального часу;

– інтегроване обчислювальне середовище: Spyder інтегрується з IPython, що дозволяє використовувати інтерактивні блокноти IPython та інші інструменти IPython безпосередньо в IDE;

– повнофункціональний налагоджувач: Spyder має вбудований наладчик для пошуку та виправлення помилок у коді Python, підтримує такі функції, як точки

зупинки, переміщення по коду, перевірка змінних та навігація стеком викликів, що полегшує виявлення та виправлення помилок у програмах на Python;

– переглядач змінних: Spider IDE надає інструмент дослідника змінних, який дозволяє користувачам перевіряти та взаємодіяти зі змінними, масивами та структурами даних у своїх скриптах на Python, ця функція дозволяє переглядати та редагувати значення змінних, які використовуються у коді;

– провідник проєктів: Spider IDE пропонує можливості керування проєктами, дозволяючи розробникам організовувати свої проекти на Python у папки та файли, надає інструмент для навігації по файлах і каталогах проекту, що полегшує роботу з великими кодовими базами;

– інструменти візуалізації: Spyder інтегрується з Matplotlib та іншими бібліотеками візуалізації Python, що дозволяє створювати графіки та інші візуалізації прямо у IDE;

– інтеграція з науковими бібліотеками: Spider IDE користується особливою популярністю серед науковців та дослідників завдяки своїй безшовній інтеграції з науковими бібліотеками, такими як NumPy, SciPy, Pandas та IPython., надає інструменти та функції, спеціально розроблені для аналізу даних, візуалізації та наукових обчислень;

– налаштування та розширюваність, підтримка плагінів: Spyder можна розширювати за допомогою плагінів, що дозволяє додавати нові функції та інструменти до IDE, добре налаштовується, дозволяючи користувачам налаштовувати параметри редактора, комбінації клавіш і теми відповідно до своїх уподобань.

Загалом, Spider IDE – це потужне та зручне середовище розробки для програмістів на Python, особливо для тих, хто працює в науковій чи аналітичній сфері завдяки його інтеграції з науковими бібліотеками Python та іншими особливостями. Добре підходить для наукових обчислень, аналізу даних та дослідницьких проектів.

Таким чином, вибір Spider IDE як середовища розробки для інтелектуальної системи прогнозування енергоспоживання був обґрунтований потужними функціями, зручністю використання, підтримкою мови Python та наукових обчислень і аналізу даних. Це дозволило підвищити ефективність та продуктивність розробки, зменшити кількість помилок та пришвидшити процес створення системи. Для даного проекту Spider IDE виявилася оптимальним вибором, який дозволив успішно реалізувати поставлені задачі та досягти бажаних результатів.

## 2.2 Мова програмування Python

Вибір мови програмування є важливим рішенням при розробці будь-якого програмного забезпечення, оскільки це впливає на швидкість розробки, продуктивність, зручність супроводу та розширюваність системи. Для розробки інтелектуальної системи прогнозування енергоспоживання було обрано мову програмування Python.

Python – це високорівнева, інтерпретована мова програмування загального призначення, яка широко використовується в різних сферах, включаючи веброзробку, аналіз даних, машинне навчання та наукові обчислення [12]. Python має чистий та лаконічний синтаксис, що робить код легким для читання та розуміння, а також сприяє швидкій розробці та зменшенню кількості помилок.

Однією з ключових переваг Python є її чистий і лаконічний синтаксис, який підкреслює читабельність і знижує витрати на підтримку програм. Python використовує відступи для розмежування блоків коду, що дозволяє легко зрозуміти структуру програм з першого погляду. Ця філософія дизайну, яку часто називають «пітонічною», заохочує до створення елегантного та виразного коду, який є одночасно ефективним і простим в обслуговуванні.

Python – це інтерпретована мова, код на Python виконується інтерпретатором під час виконання, а не компілюється в машинний код заздалегідь. Це робить

Python дуже інтерактивним та придатним для швидкого створення прототипів, тестування та налагодження. Крім того, система динамічної типізації Python дозволяє присвоювати змінні без явного зазначення їхнього типу, що забезпечує гнучкість і швидкість розробки.

Стандартна бібліотека Python обширна і включає модулі для широкого спектру завдань, від файлового вводу/виводу та роботи з мережею до веброзробки, маніпулювання даними та наукових обчислень. Крім того, екосистема пакунків Python, що підтримується індексом пакунків Python PyPI, надає доступ до тисяч сторонніх бібліотек та фреймворків, які ще більше розширюють можливості Python. Серед відомих бібліотек – NumPy для чисельних обчислень, Pandas для аналізу даних, Matplotlib для візуалізації даних, Django та Flask для веб-розробки, TensorFlow та PyTorch для машинного та глибокого навчання та багато інших.

Однією з головних переваг Python є його багата екосистема бібліотек та фреймворків. Python має величезну кількість бібліотек для різних задач, включаючи аналіз даних, машинне навчання, візуалізацію даних, веброзробку тощо. Це дозволяє швидко та ефективно вирішувати складні задачі без необхідності писати все з нуля.

Для розробки інтелектуальної системи прогнозування енергоспоживання, Python був обраний з перерахованих нижче причин.

1. Бібліотеки для аналізу даних та машинного навчання. Python має багату екосистему бібліотек для аналізу даних та машинного навчання, такі як NumPy, Pandas, Scikit-learn та Keras. Ці бібліотеки надають потужні інструменти для обробки, очищення та аналізу даних, побудови та навчання моделей машинного навчання, а також оцінки їх ефективності. Це дозволяє швидко та ефективно розробляти складні системи аналізу даних та прогнозування.

2. Швидкість розробки. Python має чистий та лаконічний синтаксис, що дозволяє швидко писати код та зменшує кількість помилок. Це особливо важливо для проектів з обмеженими часовими рамками, де швидкість розробки є

критичною. Python дозволяє зосередитися на вирішенні бізнес-задач, а не на технічних деталях реалізації.

3. Інтеграція з іншими технологіями. Python має хорошу інтеграцію з іншими технологіями та системами. Він може легко взаємодіяти з базами даних, веб-сервісами, файловими системами тощо. Це дозволяє будувати комплексні системи, що включають різні компоненти та джерела даних.

4. Веброзробка. Python має потужні фреймворки для веб-розробки, такі як Django та Flask. Ці фреймворки дозволяють швидко та ефективно розробляти веб-додатки та API. У даному проекті, Flask був використаний для розробки веб-інтерфейсу системи прогнозування, що дозволило легко інтегрувати модель прогнозування з веб-додатком.

5. Спільнота та підтримка. Python має велику та активну спільноту розробників, що забезпечує постійну підтримку та розвиток мови. Існує багато ресурсів, таких як документація, навчальні матеріали, форуми та блоги, де можна знайти відповіді на питання та вирішення проблем. Крім того , спільнота активно розробляє нові бібліотеки та інструменти, що розширюють можливості Python.

6. Кросплатформеність. Python є кросплатформенною мовою програмування, що означає, що програми, написані на Python, можуть бути легко перенесені між різними операційними системами, такими як Windows, Linux та macOS. Це дозволяє розробляти системи, які можуть працювати на різних платформах без необхідності внесення значних змін до коду.

Універсальність Python робить її придатною для широкого спектру застосувань, включаючи веб-розробку, програмування настільних графічних інтерфейсів, наукові обчислення, аналіз даних, штучний інтелект, машинне навчання, автоматизацію, написання сценаріїв та багато іншого.

Її переносимість на різні операційні системи та платформи, а також потужна підтримка спільноти та активний розвиток забезпечують її

актуальність та незмінну популярність у постійно мінливому ландшафті мов програмування.

Таким чином, простота, читабельність, універсальність, велика стандартна бібліотека та динамічна екосистема сторонніх пакетів роблять Python найкращим вибором для розробників, які шукають потужну та гнучку мову для вирішення широкого спектру завдань програмування. Зростаюче використання мови у різних галузях та сферах свідчить про її ефективність та незмінну привабливість у світі розробки програмного забезпечення.

Отже, вибір Python як мови програмування для розробки інтелектуальної системи прогнозування енергоспоживання був обґрунтований її багатою екосистемою бібліотек для аналізу даних та машинного навчання, швидкістю розробки, інтеграцією з іншими технологіями, можливостями веб-розробки, підтримкою спільноти та кросплатформенністю. Python дозволив ефективно вирішити поставлені задачі та створити потужну та гнучку систему прогнозування.

### 2.3 Реляційна система керування базами даних SQLite

При розробці інтелектуальної системи прогнозування енергоспоживання, важливим аспектом є застосування СКБД для збереження даних стосовно спожитої електроенергії. Для вирішення цієї задачі було обрано SQLite.

SQLite – це легка, вбудована реляційна система керування базами даних (СКБД), яка широко використовується для розробки програмного забезпечення [13]. Вона відома своєю простотою, надійністю та легкістю інтеграції в застосунки. Ключові особливості SQLite включають:

– самодостатність: бази даних SQLite є автономними і зберігаються у вигляді одного файлу на диску, вони вимагають мінімального налаштування та конфігурації, що робить їх простими у розгортанні та обслуговуванні;

– нульова конфігурація: SQLite не вимагає окремого серверного процесу або адміністрування, програми можуть безпосередньо звертатися до баз даних SQLite, що спрощує розгортання;

– кросплатформеність: SQLite є кросплатформенною і працює на різних операційних системах, включаючи Windows, macOS, Linux і мобільні платформи, такі як iOS і Android;

– сумісність з SQL: SQLite підтримує підмножину мови SQL (мова структурованих запитів), що надає команди для створення, запитів, оновлення та видалення даних. Є сумісною з більшістю функцій SQL, що полегшує роботу з реляційними даними;

– підтримка транзакцій: SQLite підтримує транзакції ACID (Atomicity, Consistency, Isolation, Durability – атомарність, узгодженість, ізоляція, довговічність), забезпечуючи цілісність та узгодженість даних навіть при одночасному доступі та системних збоях;

– малий розмір: SQLite має невелику обсяг і займає мало місця в пам’яті та на диску, що робить її придатною для вбудованих систем, мобільних додатків і середовищ з обмеженими ресурсами;

– універсальність: SQLite є універсальною і використовується в різних застосунках, включаючи десктопні та мобільні застосунки, веброзробку, IoT тощо. Часто використовується як вбудована база даних, але також може слугувати автономною або внутрішньою базою даних.

У контексті розробки інтелектуальної системи прогнозування енергоспоживання, використання SQLite дозволяє користувачам завантажувати історичні дані про енергоспоживання, обирати параметри прогнозування, та отримувати результати прогнозування у вигляді таблиць і графіків. Крім того, SQLite дозволив реалізувати додаткові функції, такі як збереження результатів прогнозування, генерація звітів та можливість порівняння різних сценаріїв прогнозування.

## 2.4 Бібліотеки NumPy, Pandas, Keras, Scikit-learn, Matplotlib

Розробка інтелектуальної системи прогнозування енергоспоживання вимагає використання потужних та ефективних інструментів для обробки, аналізу та моделювання даних. В рамках даного проекту було обрано ряд бібліотек Python, які забезпечують необхідну функціональність та дозволяють вирішувати поставлені задачі з високою продуктивністю та точністю.

Важливою бібліотекою, яка була використана в проєкті, є NumPy. NumPy – це фундаментальна бібліотека для наукових обчислень в Python, яка надає підтримку для роботи з багатовимірними масивами та матрицями, а також набір математичних функцій для ефективної роботи з цими масивами [14]. Основною структурою даних NumPy є ndarray (n-вимірний масив), який є однорідним контейнером для даних одного типу.

Основні можливості NumPy:

– NumPy's ndarray – це потужна структура даних, яка представляє масиви довільної розмірності та дозволяє ефективно зберігати та маніпулювати великими масивами даних;

– NumPy надає широкий спектр математичних функцій для виконання операцій з масивами, включаючи арифметичні операції, тригонометричні функції, логарифми, експоненти і багато іншого;

– NumPy пропонує повний набір функцій лінійної алгебри для маніпуляцій з матрицями, включаючи множення матриць, інверсію, обчислення визначників, розкладання за власними значеннями та розкладання за сингулярними значеннями (SVD);

– NumPy містить функції для генерування випадкових чисел і випадкових вибірок з різних розподілів ймовірностей, таких як рівномірний, нормальний, біноміальний і пуассонівський;

– інтеграція з іншими бібліотеками: NumPy легко інтегрується з іншими науковими обчислювальними бібліотеками Python, формуючи основу наукової екосистеми Python.

NumPy є основою для багатьох інших бібліотек Python, таких як Pandas, Scikit-learn та Keras, і забезпечує високу продуктивність та ефективність обчислень. NumPy надає широкий спектр математичних функцій для роботи з масивами, такі як статистичні функції, лінійна алгебра, перетворення Фур'є та багато іншого.

У контексті розробки інтелектуальної системи прогнозування енергоспоживання, NumPy був використаний для виконання числових операцій та перетворень над даними, а також для підготовки даних для подальшого використання в моделях машинного навчання. NumPy дозволив ефективно працювати з великими обсягами числових даних та забезпечив високу швидкодію обчислень.

Однією з ключових бібліотек, яка була використана в проекті, є Pandas. Pandas – це потужна бібліотека для маніпулювання та аналізу даних, побудована на основі NumPy [14]. Вона надає високорівневі структури даних, такі як DataFrame і Series, призначені для роботи зі структурованими або табличними даними. Pandas спрощує завдання маніпулювання даними, такі як завантаження, очищення, перетворення та аналіз даних, що робить її важливим інструментом для роботи з даними та їх дослідження у Python.

Pandas – це бібліотека Python, яка надає зручні та ефективні структури даних для роботи з табличними даними, такими як часові ряди та числові таблиці. Вона дозволяє легко завантажувати, маніпулювати та аналізувати великі обсяги даних з різних джерел, таких як CSV файли, бази даних та веб-сервіси.

Pandas надає потужні інструменти для очищення та підготовки даних, такі як фільтрація, агрегація, злиття таблиць та перетворення даних. Крім того, Pandas має вбудовані функції для візуалізації даних, що дозволяє швидко створювати графіки та діаграми для дослідження та презентації результатів.

Основні можливості Pandas:

– DataFrame в Pandas: двовимірна розмічена структура даних зі стовпчиками потенційно різних типів даних, надає потужні можливості індексування, нарізки та зміни форми для маніпулювання даними в табличному форматі;

– Pandas' Series: одновимірний розмічений масив, здатний зберігати дані будь-якого типу, являє собою один стовпець або рядок даних у фреймі даних і підтримує багато з тих же операцій, що і масиви NumPy;

– вирівнювання даних: Pandas автоматично вирівнює дані на основі індексів міток, що дозволяє легко маніпулювати даними та виконувати арифметичні операції між різними структурами даних;

– обробка відсутніх даних: Pandas надає функції для виявлення, видалення та заповнення відсутніх значень у наборах даних, що дозволяє користувачам ефективно працювати з неповними або суперечливими даними;

– групування та агрегація: Pandas підтримує потужні операції групування та агрегації, дозволяючи користувачам групувати дані на основі одного або декількох ключів і виконувати обчислення, такі як сума, середнє, медіана, мінімум, максимум тощо, для згрупованих даних.

У контексті розробки інтелектуальної системи прогнозування енергоспоживання, Pandas був використаний для завантаження та обробки історичних даних про енергоспоживання з різних регіонів України. За допомогою Pandas було виконано очищення даних від пропусків та викидів, агрегацію даних за часовими періодами та регіонами, а також створення нових ознак на основі наявної інформації. Pandas дозволив ефективно працювати з великими обсягами даних та підготувати їх для подальшого аналізу та моделювання.

Для розробки системи прогнозування енергоспоживання було використано бібліотеку Keras. Keras є високорівневою бібліотекою Python для роботи з нейронними мережами та глибоким навчанням [15]. Keras як високорівневий API нейронних мереж, написаний на Python, призначений для швидкого експериментування та створення прототипів моделей глибокого навчання. Він

надає зручний інтерфейс для побудови, навчання та розгортання нейронних мереж, з акцентом на простоту, модульність та розширюваність.

Ключові особливості Keras:

– зручний інтерфейс: Keras пропонує простий та інтуїтивно зрозумілий API для визначення архітектури нейронних мереж, що дозволяє користувачам легко створювати та налаштовувати складні моделі з мінімальним кодом;

– модульність: Keras має модульну конструкцію з набором багаторазових будівельних блоків (шарів), які можна легко комбінувати для створення різних типів нейронних мереж, включаючи згорткові мережі, рекурентні мережі та інші;

– гнучкість: Keras підтримує прискорення як на CPU, так і на GPU і легко інтегрується з іншими фреймворками для глибокого навчання, такими як TensorFlow і Theano, дозволяючи користувачам скористатися перевагами апаратного прискорення для швидшого навчання і висновків;

– розширюваність: Keras має гнучку архітектуру, яка дозволяє легко налаштовувати та розширювати його а користувачі можуть створювати власні шари, функції втрат і метрики, підключати їх до існуючих моделей або створювати абсолютно нові компоненти;

– підтримка спільноти: Keras має велику і активну спільноту розробників, дослідників і практиків, які роблять свій внесок у його розвиток, діляться кодом і ресурсами, а також надають підтримку через форуми, навчальні посібники та документацію.

Для оцінки якості моделі прогнозування було використано бібліотеку Scikit-learn [16]. Scikit-learn – це потужна бібліотека Python для машинного навчання, побудована на основі NumPy, SciPy та matplotlib. Вона надає прості та ефективні інструменти для інтелектуального аналізу даних, аналізу даних та завдань машинного навчання, включаючи класифікацію, регресію, кластеризацію, зменшення розмірності та вибір моделі.

Ключові особливості Scikit-learn:

– простий і послідовний API: Scikit-learn пропонує простий і послідовний API, що робить його простим у використанні і вивченні, всі алгоритми реалізовані як класи Python з послідовними інтерфейсами для підбору, прогнозування та оцінки моделей;

– широкий вибір алгоритмів: Scikit-learn надає широкий спектр алгоритмів керованого та некерованого навчання, включаючи лінійні моделі, машини опорних векторів, дерева рішень, випадкові ліси, градієнтний бустінг, k-найближчих сусідів, алгоритми кластеризації та багато іншого;

– оцінка та вибір моделей: Scikit-learn включає інструменти для оцінки та вибору моделей, такі як перехресна перевірка, пошук по сітці та метрики продуктивності (точність, F1-бали тощо), що дозволяє користувачам оцінювати продуктивність своїх моделей та ефективно налаштовувати гіперпараметри;

– вилучення та попередня обробка даних: Scikit-learn пропонує утиліти для вилучення ознак, вибору ознак та попередньої обробки даних, включаючи масштабування, нормалізацію, кодування категорійних змінних, обробку пропущених значень та генерування поліноміальних ознак;

– інтеграція з науковою екосистемою Python: Sikit learn легко інтегрується з іншими бібліотеками наукової екосистеми Python, такими як NumPy, SciPy, Pandas та matplotlib, що дозволяє користувачам використовувати всю потужність Python для аналізу даних та завдань машинного навчання.

Matplotlib – це потужна та гнучка бібліотека для створення графіків на мові програмування Python [17]. Вона дозволяє створювати широкий спектр високоякісних візуалізацій, таких як лінійні графіки, гістограми, діаграми розсіювання, теплові карти та багато інших. Основні характеристики Matplotlib:

– гнучкість та потужність: Matplotlib дозволяє створювати різноманітні типи графіків, від простих лінійних до складних 3D-візуалізацій, можливість детально налаштовувати кожен аспект графіку, включаючи стилі ліній, маркерів, кольорів та написів;

– сумісність: підтримка різних форматів файлів для збереження графіків, таких як PNG, PDF, SVG та інші, інтеграція з багатьма іншими бібліотеками Python, такими як NumPy, Pandas, та SciPy, що полегшує обробку та візуалізацію даних;

– зручність використання: простий у використанні API, який дозволяє швидко створювати базові графіки, підтримка як інтерактивного, так і неінтерактивного режимів роботи, що дозволяє використовувати Matplotlib як в інтерактивних середовищах, таких як Jupyter Notebook, так і в сценаріях, що запускаються автоматично.

До основних компонент Matplotlib відносять:

– Pyplot: основний модуль, який забезпечує високорівневий інтерфейс для створення графіків та функції, які дозволяють швидко і легко створювати стандартні графіки;

– Figure: контейнер для всіх частин візуалізації, містить один або більше об'єктів Axes, використовується для створення та налаштування графічного полотна;

– Axes: основний простір для малювання графіків всередині Figure, кожен Axes об’єкт має свою систему координат і свої параметри, забезпечує методи для додавання елементів до графіку, таких як лінії, точки, легенди, осі тощо;

– Artist: всі видимі частини графіка в Matplotlib є обєктами Artist, це можуть бути лінії, тексти, фігури та інші елементи.

Matplotlib є універсальним інструментом для створення високоякісних графіків у Python, що робить його надзвичайно корисним для дослідників, аналітиків та розробників програмного забезпечення.

Розглянуті бібліотеки відіграють важливу роль у різних аспектах науки про дані, машинного навчання та наукових обчислень у Python, надаючи необхідні інструменти та функції для роботи з даними, побудови моделей та виконання аналізу. Їх універсальність, простота використання та велика документація роблять їх незамінними як для розробників, так і для дослідників та практиків.

Вибір бібліотек Pandas, NumPy, Keras, Scikit-learn та Matplotlib для розробки інтелектуальної системи прогнозування енергоспоживання був обумовлений їх потужними можливостями та ефективністю у вирішенні поставлених задач. Pandas дозволив ефективно працювати з табличними даними, виконувати очищення та підготовку даних. NumPy забезпечив високу продуктивність числових обчислень та операцій над масивами. Scikit-learn дозволив оцінити якість моделі прогнозування та виконати вибір найкращих гіперпараметрів.

Використання цих бібліотек надає можливості для створення потужної та ефективної системи прогнозування енергоспоживання, яка здатна обробляти великі обсяги даних, будувати точні моделі прогнозування та надавати надійні результати. Завдяки гнучкості та розширюваності обраних бібліотек, розроблена система може бути легко адаптована до нових вимог та масштабована для обробки більших обсягів даних та вирішення складніших задач прогнозування.

## 2.5 Розробка GUI за допомогою PyQt5

PyQt5 – набір бібліотек, написаних мовою програмування Python для Qt – крос-платформного фреймворку, що використовується у розробці графічних інтерфейсів користувача GUI [18]. Qt фреймворк є потужним інструментом для кросплатформеної розробки, розроблений для створення застосунків із графічним інтерфейсом користувача. Основною мовою програмування для Qt є C++, проте існує можливість використання мов Python та QML (Qt Meta-Object Language) при розробці.

Переваги використання Qt [19]:

– *кросплатформеність*: Qt дозволяє розробникам створювати програми, які можуть працювати на різних платформах без необхідності зміни вихідного коду. Це суттєво скорочує час розробки та спрощує підтримку програми;

– *висока продуктивність*: Завдяки використанню C++ для розробки, програми на Qt мають високу продуктивність і швидкодію;

– *широкий спектр інструментів*: Qt надає розробникам великий вибір інструментів та бібліотек для різних завдань, що спрощує процес розробки та робить його зручнішим;

– *потужна документація та спільнота*: Qt має велику документацію, а також велику спільноту розробників, які готові допомогти і поділитися досвідом;

– *сумісність з іншими мовами*: Qt підтримує не тільки C++, але й інші мови, такі як Python та QML, що збільшує гнучкість під час розробки.

Використовуючи PyQt5, розробник може створювати додатки набагато швидше, не жертвуючи значною частиною продуктивності C++. PyQt5 має модульну архітектуру, тобто, різні компоненти бібліотеки розділені на окремі модулі. Це дозволяє розробникам імпортувати лише необхідні з них, уникати зайвого навантаження на згадку та процесор. Крім того, модульність полегшує підтримку та розширення бібліотеки, оскільки кожен модуль може бути оновлений або замінений незалежно від інших.

PyQt5 пропонує багатий набір певних віджетів, які можна використовувати для створення інтерфейсу на Python. Віджети – це графічні елементи, такі як кнопки, поля введення, таблиці і т. д., що дозволяють взаємодіяти з програмою. У бібліотеці також реалізована гнучка система макетів, за допомогою якої розробники організовують віджети в інтерфейсі користувача. Макети дозволяють створювати адаптивні інтерфейси, здатні автоматично підлаштовуватись під зміни розмірів вікна або пристрою.

Взаємодія між різними компонентами інтерфейсу користувача в PyQt5 здійснюється за допомогою:

– сигналів – подій, що виникають за певних дій користувача, таких як натискання кнопки або зміна значення поля введення;

– слотів – функцій, які виконуються у відповідь сигнали.

Розробники можуть зв’язувати сигнали і слоти, щоб забезпечити реакцію програми на дії користувача. Цей механізм забезпечує гнучку та ефективну

комунікацію між різними частинами програми. Це дозволяє гнучко та ефективно реагувати на дії користувача та динамічно оновлювати інтерфейс користувача

Бібліотека підтримує схему MVC (модель — представлення — контролер), яка є шаблоном проектування для обробки та відображення даних у інтерфейсі користувача. Вона дозволяє розробникам розділити дані, їх подання та логіку обробки, що спрощує розробку та підтримку складних додатків з великими обсягами даних. Також PyQt5 надає різні класи моделей уявлень, які можна використовувати відповідно до вимог програми.

PyQt5 надає багатий набір готових віджетів, таких як кнопки, поля введення, таблиці, діаграми та інші. Завдяки цьому розробники можуть швидко проектувати різноманітні елементи інтерфейсу користувача. Бібліотека дозволяє також створювати власні віджети, адаптовані під конкретні потреби програми. Це дає можливість створювати унікальні та індивідуальні інтерфейси. Вона також включає PyQt5 Designer, конструктор графічного інтерфейсу користувача. Він значно спрощує створення GUI – розробнику достатньо розташувати готові елементи у відповідних місцях макета. PyQt може генерувати код Python із Qt Designer. Також до конструктора можна додати нові елементи управління графічним інтерфейсом, написані на Python.

Бібліотека пропонує різні механізми для створення гнучких макетів та стилів графічного інтерфейсу на Python. Розробники можуть вибирати з визначених макетів або створювати власні, а також налаштовувати зовнішній вигляд елементів інтерфейсу за допомогою стилів. PyQt5 дозволяє взаємодіяти з іншими процесами або потоками, що корисно для розробки додатків, які потребують асинхронної обробки даних або комунікації зі сторонніми застосунками.

Основні характеристики та переваги PyQt5:

– кросплатформеність: PyQt5 підтримує різні операційні системи, включаючи Windows, macOS, Linux та різні Unix-системи;

– багата бібліотека інтерфейсу користувача: PyQt5 надає широкий набір функцій та інструментів для розробки GUI і створення сучасних та інтерактивних

інтерфейсів: кнопки, мітки, текстові поля, комбіновані поля, таблиці, меню, діалогові вікна, таблиці, діаграми тощо, що надає розробнику багаті можливості для створення різних макетів і стилів інтерфейсу;

– керування розміщенням елементів: PyQt5 надає потужні інструменти для управління розміщенням елементів інтерфейсу в гнучкий та послідовний спосіб, менеджери розміщення, такі як QHBoxLayout, QVBoxLayout, QGridLayout та QFormLayout, допомагають забезпечити адаптивність інтерфейсу до різних розмірів екрану та роздільної здатності;

– простота використання: PyQt5 пропонує простий та інтуїтивно зрозумілий API, який легко освоїти розробникам Python;

– інтеграція з Qt Designer: PyQt5 безперешкодно інтегрується з Qt Designer, візуальним інструментом для створення дизайну інтерфейсу користувача, розробники можуть створювати дизайн інтерфейсу з використанням Qt Designer та завантажувати його в свої програми PyQt5, що полегшує створення складних інтерфейсів;

– PyQt5 добре інтегрується з іншими популярними інструментами та бібліотеками Python, такими як NumPy, SciPy та Matplotlib, це дозволяє розробникам створювати складні та високоефективні програми, використовуючи різні можливості цих інструментів;

– підтримка мультимедіа та графіки: PyQt5 надає підтримку мультимедійних функцій, таких як відтворення аудіо та відео файлів, а також рендерінгу 2D та 3D графіки за допомогою потужного рендерингового движка Qt;

– С має велику документацію та активну спільноту розробників, завжди є підтримка та відповіді на запитання, а також безліч прикладів і посібників, які допоможуть вам швидко розібратися з бібліотекою та вирішити проблеми, що виникають;

– відкритий вихідний код та ліцензія: PyQt5 поширюється під GNU GPL, або комерційною ліцензією, що робить його доступним для використання у різних

проектах, завдяки відкритому коду розробники можуть модифікувати та адаптувати бібліотеку під свої потреби.

Проте PyQt5 має і недоліки. Зокрема, він може бути складним для новачків. Вивчення всіх можливостей та функцій бібліотеки потребує часу та зусиль. Залежність від Qt: PyQt5 є розширенням для Qt-фреймворку, для роботи з бібліотекою необхідно встановити Qt. Також при використанні PyQt5 у комерційних проектах може знадобитися придбання комерційної ліцензії.

Загалом, PyQt5 є потужним інструментом для розробки крос-платформних графічних інтерфейсів мовою Python, що пропонує широкий спектр можливостей, інструментів і ресурсів для спрощення процесу розробки. З його допомогою можна створювати функціональні та привабливі програми, які будуть працювати на різних операційних системах. PyQt5 є оптимальним варіантом для проектування GUI на Python, і, незважаючи на деякі обмеження, є популярним вибором для багатьох розробників.

## 2.6 Бібліотека Prophet

Prophet є бібліотекою для прогнозування часових рядів, розробленою Facebook [20]. Вона надає простий і потужний інтерфейс для прогнозування часових рядів з трендами, сезонністю та святковими ефектами. За допомогою Prophet можна швидко створювати прогнози для різноманітних часових рядів, включаючи споживання електроенергії домогосподарствами.

Prophet дозволяє в автоматичному режимі створювати адитивні моделі часового ряду, які мають високу точність прогнозу [21]:

$$Y(t) = T(t) + S_y(t) + S_w(t) + H(t) + E(t), \qquad (2.1)$$

де $T(t)$ – тренд;

$S_y(t)$ – річна сезонна компонента;

$S_w(t)$ – тижнева сезонна компонента;

$H(t)$ – компонента «святкових» днів;

$E(t)$ – залишкова компонента.

В основі закладеної методології лежить процедура підгонки адитивних регресійних моделей з перерахованими вище компонентами.

Тренд $T(t)$ – моделюється за допомогою кускової лінійної регресії або логістичної функції.

Кусково лінійна регресія враховує наявність суттєвих перепадів у тренді. Для кожного лінійного фрагменту будується рівняння лінійної регресії:

$$T(t) = a_0 + b \cdot t\,, \tag{2.2}$$

де $a_0$ та $b$ – коефіцієнти, які підбираються шляхом підгону.

Формула логістичної функції:

$$T(t) = \frac{C}{1 + \exp(-k(t-b))}, \tag{2.3}$$

де C – пропускна спроможність;

k – швидкість росту;

b – параметр зсуву.

У ході підгонки моделі спрацьовує механізм регуляризації, який дозволяє обрати оптимальну кількість точок суттєвих перепадів у тренді та підібрати описані вище параметри.

Річна сезонність $S_y(t)$ моделюється як ряд Фур'є та відповідає за моделювання періодичних змін, пов'язаних із річними коливаннями.

$$S_y(t) = \sum_{i=1}^{n} \left( a_i \cdot \cos\left(\frac{2\pi n t}{P}\right) + b_i \cdot \sin\left(\frac{2\pi n t}{P}\right) \right), \tag{2.4}$$

де $P$ – період сезонних коливань (для тижневої сезонності 7 днів, для року – 365,25 днів);

$a_i$ та $b_i$ – коефіцієнти, які визначаються шляхом підбору.

Підгонка сезонності вимагає оцінки 2n параметрів $\beta = [a_1, b_1, \ldots, a_n, b_n]$. Це робиться шляхом побудови матриці векторів сезонності для кожного значення $t$ в історичних і майбутніх даних.

Тижнева сезонність $S_w(t)$ відповідає за моделювання періодичних змін, пов'язаних із тижневими коливаннями. Моделюється з використанням індикаторної бінарної змінної, яка відповідає дням тижня:

[*Monday*, *Tuesday*, *Wednesday*, *Thursday*, *Friday*, *Saturday*].

Кожна позиція бінарного вектора може приймати значення 0 і 1 у залежності від дати. Ознака *Sundey*, яка відповідає сьомому дню тижня, не додається, тому що він буде лінійно залежати від інших днів тижня і це буде впливати на модель.

Компонента «святкових» днів $H(t)$ відповідає за аномальні дні, які пов'язані з офіційними святковими та вихідними днями а також іншими днями, під час яких властивості часового ряду можуть суттєво змінитись: спортивними чи культурними подіями, природними явищами тощо. Як і у випадку з днями тижня такі дні представлені в моделі у вигляді індикаторних бінарних змінних.

Залишкова компонента $E(t)$ – містить інформацію, яка не врахована моделлю.

Оцінювання параметрів моделі, що підганяється, виконується з використанням принципів байєсівської статистики або методом знаходження апостеріорного максимуму (MAP), або шляхом повного байєсівського виводу.

Бібліотека Prophet досить зручна, легко кастомізується, має можливість додавання заздалегідь відомих аномальних днів, тому її корисно використовувати для побудови прогнозу енергоспоживання, оскільки воно не є простим у моделюванні – часовий ряд може мати складний тренд. Саме ця модель була використана при побудові прогнозної моделі єнергоспоживання.

## Висновки до розділу 2

У цьому розділі було розглянуто та обґрунтовано вибір технологій та інструментальних засобів розробки системи прогнозування енергоспоживання. Розглянуті технології забезпечують повний цикл розробки від збору і обробки даних до візуалізації результатів і побудови моделей машинного навчання.

Вибір Spyder як середовища розробки забезпечує зручність у написанні та відлагодженні коду завдяки його інтегрованим інструментам та підтримці наукових обчислень. Мова програмування Python обрана через її потужні можливості для аналізу даних. Numpy і Pandas – бібліотеки є основними для роботи з даними. Numpy забезпечує ефективну роботу з багатовимірними масивами і матрицями, тоді як Pandas надає потужні інструменти для маніпуляції та аналізу табличних даних.

Бібліотека Matplotlib використовується для візуалізації даних, що дозволяє створювати різноманітні графіки і діаграми, необхідні для аналізу та презентації результатів. Keras і Scikit-learn – бібліотеки є ключовими для розробки моделей машинного навчання. Keras забезпечує зручний інтерфейс для створення та тренування нейронних мереж, а Scikit-learn надає широкий спектр інструментів для класифікації, регресії, кластеризації та інших методів машинного навчання.

Використання PyQt5 дозволяє створювати графічні інтерфейси користувача для інтелектуальної системи прогнозування, роблячи її зручною та доступною для кінцевих користувачів. Система керування базами даних SQLite використовується для зберігання і управління даними, що забезпечує надійне та ефективне збереження великих обсягів інформації.

Загалом, обрані технології та інструментальні засоби створюють потужну і гнучку платформу для розробки інтелектуальних систем прогнозування, що дозволяє ефективно вирішувати задачі аналізу даних і побудови прогнозних моделей.

# 3 РОЗРОБКА І ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ ЕЛЕКТРОЕНЕРГІЇ ДОМОГОСПОДАРСТВАМИ

## 3.1 Підготовка даних до аналізу

Етап підготовки даних до аналізу є критично важливим для розробки ефективної системи прогнозування енергоспоживання. Якість та повнота вхідних даних безпосередньо впливають на точність та надійність результатів прогнозування. Тому необхідно приділити особливу увагу збору, очищенню, попередній обробці та трансформації даних перед їх використанням для навчання моделі.

Для розробки інтелектуальної системи прогнозування енергоспоживання домогосподарствами на основі моделі Prophet було створено набір даних, який містить історичні дані про споживання електроенергії Набір даних складається з історичних показників споживання електроенергії для 3248 домогосподарств, які обладнані розумними лічильниками. Всі дані були зібрані лише з розумних лічильників, які забезпечують більш точне та автоматизоване зчитування споживання електроенергії. Дані охоплюють період з січня по грудень минулого року і представлені у півгодинних інтервалах. Кожен домогосподарство має різний доступ до історичних даних, залежно від часу їх підключення, починаючи з грудня минулого року і закінчуючи груднем минулого року. Для аналізу обрано домогосподарства, обладнані розумними лічильниками [22].

Оскільки «розумні» лічильники – це інтелектуальні системи обліку, які здатні вимірювати споживання електроенергії в режимі реального часу, вони мають низку переваг порівняно з традиційними лічильниками електроенергії:

– енергопостачальники раніше розпізнають проблеми з обслуговуванням (двосторонній зв’язок);

– оскільки «розумні» лічильники надають дані про споживання в режимі реального часу, споживачі можуть краще розуміти своє енергоспоживання і

можуть контролювати та адаптувати його для зниження витрат і викидів;

– покращується процес виставлення рахунків: оскільки розумні лічильники показують дані про споживання в режимі реального часу, це усуває необхідність щорічного зняття показань лічильників вручну і зменшує кількість запитів на виставлення рахунків;

– розумні лічильники дозволяють постачальникам пропонувати більш гнучкі тарифи, які змінюються залежно від часу доби та попиту;

– розумні лічильники дозволяють операторам розподільчих систем (ОСР) краще розуміти використання мережі і відповідно управляти електроенергією, що призводить до більш надійного енергопостачання.

Після збору даних наступним кроком є їх очищення та попередня обробка [23]. Це включає наступні операції:

– видалення дублікатів та помилкових записів, перевірка цілісності даних та видалення записів з некоректними або відсутніми значеннями;

– обробка пропущених значень, у разі наявності пропусків в даних, застосовуються методи видалення, інтерполяції або заповнення середніми значеннями для відновлення відсутньої інформації;

– перетворення та агрегація даних: оскільки дані про спожиту електроенергію представлені розумними лічильниками, доцільно застосувати повторну вибірку, яка полягає в перетворенні даних часових рядів з однієї частоти на іншу з відповідним коригуванням даних.

*Обробка пропущених значень*. Формула лінійної інтерполяції – найпростіший метод, який використовується для оцінки значення функції між будь-якими двома відомими точками. Рівняння лінійної інтерполяції має вигляд [24]:

$$y = y_1 + (x - x_1)\frac{(y_2 - y_1)}{(x_2 - x_1)}, \quad (3.1)$$

де $x_1$ та $y_1$ – перші координати, $x_2$ та $y_2$ – другі координати;

x – точка для виконання інтерполяції;

y – інтерпольоване значення.

Було проведено аналіз на наявність відсутніх значень (рис. 3.1) та їх кількість і було вирішено використати лінійну інтерполяцію для заповнення відсутніх даних (рис. 3.2).

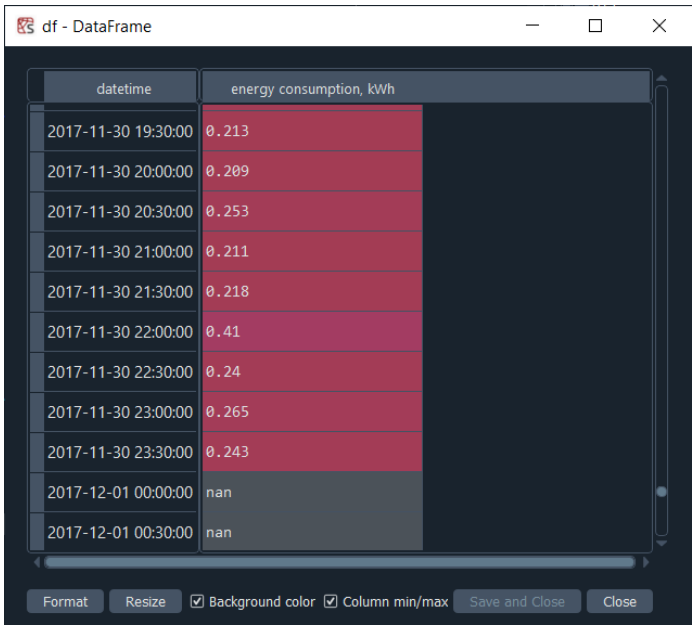



Рисунок 3.1 – Набір даних з відсутніми значеннями

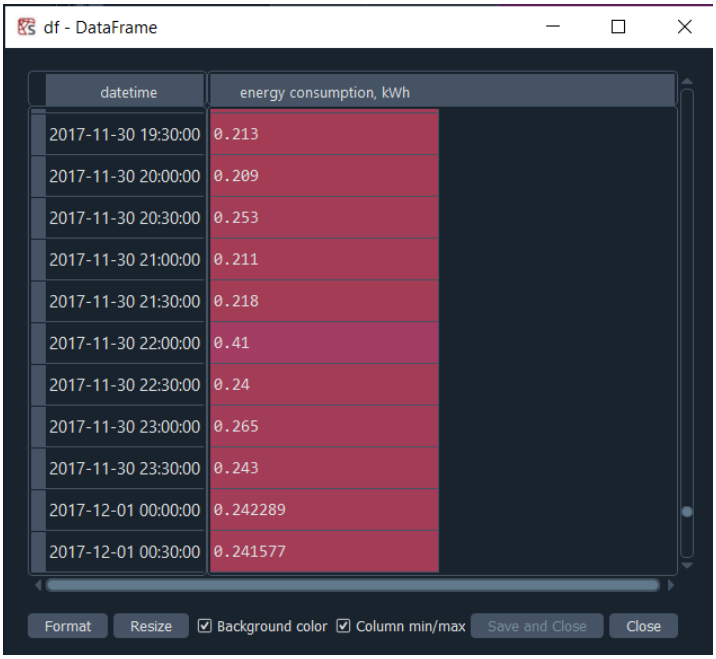



Рисунок 3.2 – Набір даних після заповнення відсутніх значень

*Перетворення та агрегація даних*. Повторна вибірка (resampling) – це фундаментальний метод статистичного аналізу часових рядів, який працює з упорядкованими в часі даними (рис. 3.3). Ця техніка дозволяє змінювати часові інтервали даних, або збільшуючи деталізацію за рахунок збільшення вибірки (upsampling), або зменшуючи її за рахунок зменшення вибірки (downsampling).

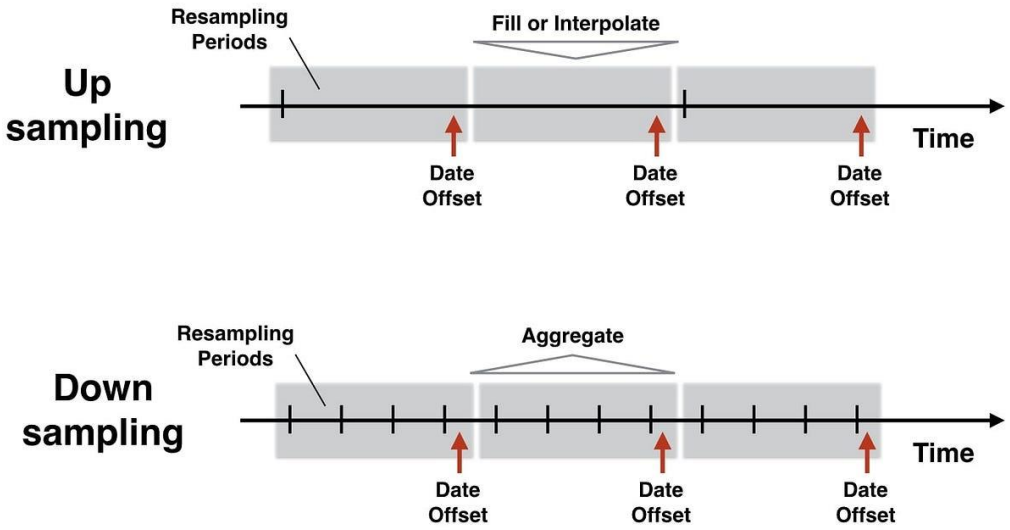



Рисунок 3.3 – Технології upsampling та downsampling

Ретроспективна періодичність, або ресемплінг, є процесом зміни частоти часового ряду шляхом введення нових точок даних [25]. Зменшення частоти часового ряду, навпаки, призводить до зменшення кількості точок даних. Наприклад, добовий часовий ряд може бути зменшений до тижневого, використовуючи середнє значення для кожного тижня. Серед методів агрегації можна вибрати середнє, максимальне, мінімальне значення або іншу функцію, яка відображатиме агреговані значення даних.

Було використано тихнологію resempling, що призвело до зменшення частоти відбору даних з 30-хвилинного інтервалу до добового (рис. 3.4, рис. 3.5). Цей процес включав у себе зменшення точок даних між кожною парою послідовних часових маркерів. Після цього застосовувався метод агрегації шляхом сумування спожитих даних за добу.

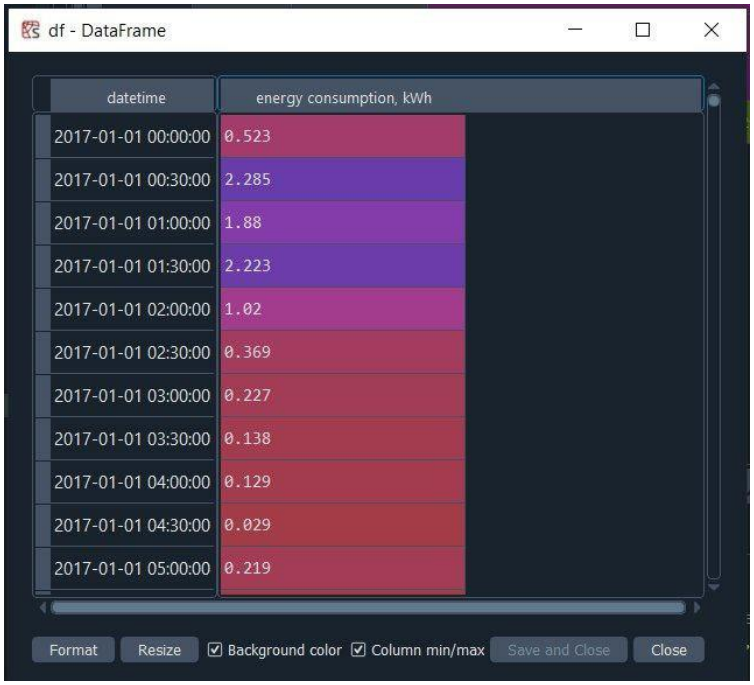



Рисунок 3.4 – До використання технології resempling

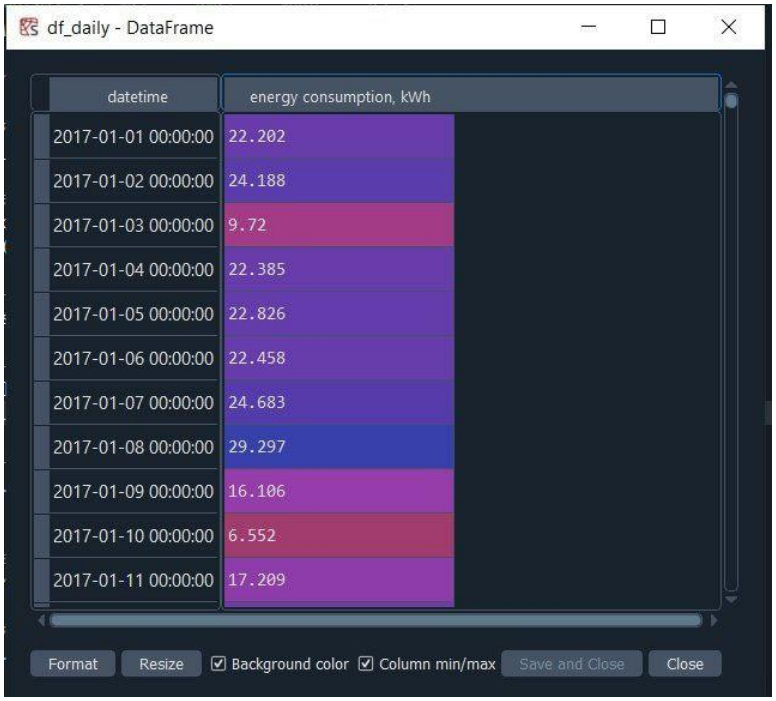



Рисунок 3.5 – Результат використання технології resempling

Після попередньої обробки вхідних даних вони зберігаються у csv-файлі для подальшого використання та аналізу.

### 3.2 Розробка БД

Для системи прогнозування енергоспоживання було використано СКБД SQLite. SQLite зберігає усю базу даних у одному файлі, не використовує парадигму клієнт-сервер. Це дає можливість для розробки декстопного застосунку на Python зі зберіганням інформації в СКБД SQLite локально. Для цього потрібно:

– розробити структуру БД;

– створити БД та таблиці в ній;

– наповнити таблиці даними;

– створити зв’язки;

– перевірити, як це працює.

ERD-діаграма бази даних, яка містить дані про користувачів системи, клієнтів, лічильників та енергоспоживання, зображена на рисунку 3.6.

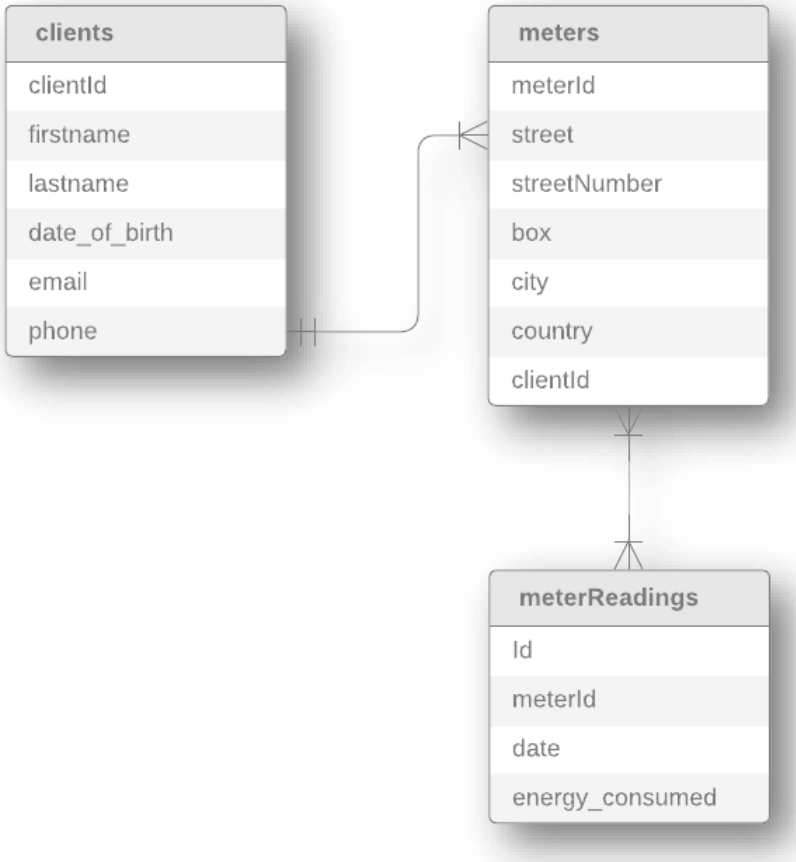



Рисунок 3.6 – Діаграма ERD «сутність-зв’язок»

За роботу з SQLite у Python відповідає стандартна бібліотека sqlite3, тому вводять команду для її підключення та для створення файлу бази даних за допомогою методу connect() (рис. 3.7). Зручність бібліотеки в тому, що достатньо вказати ім’я файлу, а далі буде наступне:

– якщо файлу немає, програма створить порожню базу даних з таким ім’ям;

– якщо вказаний файл є, то програма підключиться до нього і працюватиме з ним.

```
import sqlite3
conn = sqlite3.connect("db.db")
```

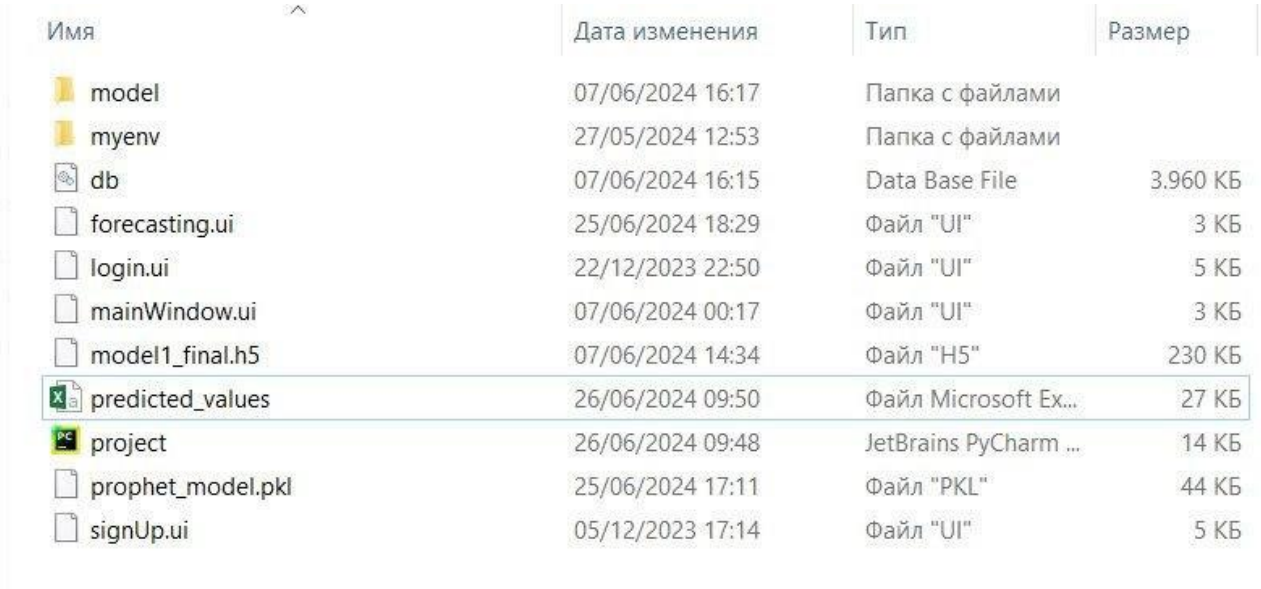
Рисунок 3.7 – Створення файлу бази даних

Далі створюють таблиці бази даних із вказівкою ключових полів, які будуть зберігатися у файлі БД. Інформація про спожиту користувачами електроенергію, яка була попередньо оброблена, вноситься до даної БД з csv-файлу, який містить ці дані.

### 3.3 Програмна розробка системи

Програмна реалізація здійснювалася у середовищі розробки Spider з використанням мови програмування Python та спеціальних бібліотек. Структура папок і файли проєкту відображено на рисунку 3.8.

***Клас MainWindow*** (додаток А). Цей клас є основним вікном застосунку, яке забезпечує відображення та керування даними клієнтів і лічильників, а також взаємодію з базою даних SQLite. Забезпечує основний інтерфейс для роботи з даними клієнтів та лічильників, включаючи функції для прогнозування та побудови графіків. Спадкує від QMainWindow, що дозволяє створювати головне вікно застосунку.



| Имя | Дата изменения | Тип | Размер |
|---|---|---|---|
| model | 07/06/2024 16:17 | Папка с файлами | |
| myenv | 27/05/2024 12:53 | Папка с файлами | |
| db | 07/06/2024 16:15 | Data Base File | 3.960 КБ |
| forecasting.ui | 25/06/2024 18:29 | Файл "UI" | 3 КБ |
| login.ui | 22/12/2023 22:50 | Файл "UI" | 5 КБ |
| mainWindow.ui | 07/06/2024 00:17 | Файл "UI" | 3 КБ |
| model1_final.h5 | 07/06/2024 14:34 | Файл "H5" | 230 КБ |
| predicted_values | 26/06/2024 09:50 | Файл Microsoft Ex... | 27 КБ |
| project | 26/06/2024 09:48 | JetBrains PyCharm ... | 14 КБ |
| prophet_model.pkl | 25/06/2024 17:11 | Файл "PKL" | 44 КБ |
| signUp.ui | 05/12/2023 17:14 | Файл "UI" | 5 КБ |

Рисунок 3.8 – Структура файлів та папок проєкту

Методи класу MainWindow:

– __init__:ініціалізує головне вікно, завантажує інтерфейс з файлу та налаштовує елементи управління;

– runForecast: відкриває вікно прогнозу для вибраного лічильника;

– loadData: завантажує дані клієнтів та лічильників з бази даних та відображає їх у таблиці;

– on_selection_changed: обробляє зміну вибору у таблиці та відображає дані споживання енергії для вибраного лічильника;

– plotEnergyConsumed: будує графік споживання енергії для вибраного лічильника.

Основні функції класу MainWindow:

– ініціалізація: завантажує графічний інтерфейс з файлу mainWindow.ui, налаштовує політику прокручування горизонтального скроллбару для таблиці, завантажує дані клієнтів та лічильників з бази даних;

– прогнозування: метод runForecast відкриває вікно прогнозу для вибраного лічильника, використовуючи клас ForecastWindow;

– завантаження даних: метод loadData виконує запит до бази даних для отримання даних клієнтів та лічильників, відображає отримані дані у таблиці;

– обробка вибору у таблиці: метод on_selection_changed визначає вибраний рядок у таблиці та викликає метод для побудови графіку споживання енергії для вибраного лічильника;

– побудова графіку: метод plotEnergyConsumed виконує запит до бази даних для отримання даних споживання енергії для вибраного лічильника, використовує бібліотеку matplotlib для побудови графіку споживання енергії та відображає його.

Клас MainWindow забезпечує функціональність відображення даних клієнтів, лічильників та графіків споживання енергії, а також прогнозування споживання, використовуючи базу даних SQLite та бібліотеки PyQt5, pyqtgraph та matplotlib.

***Клас LoginScreen*** (додаток Б). Забезпечує інтерфейс для входу користувачів у застосунок (рис. 3.9). Спадкує від QDialog, що дозволяє створювати діалогове вікно.

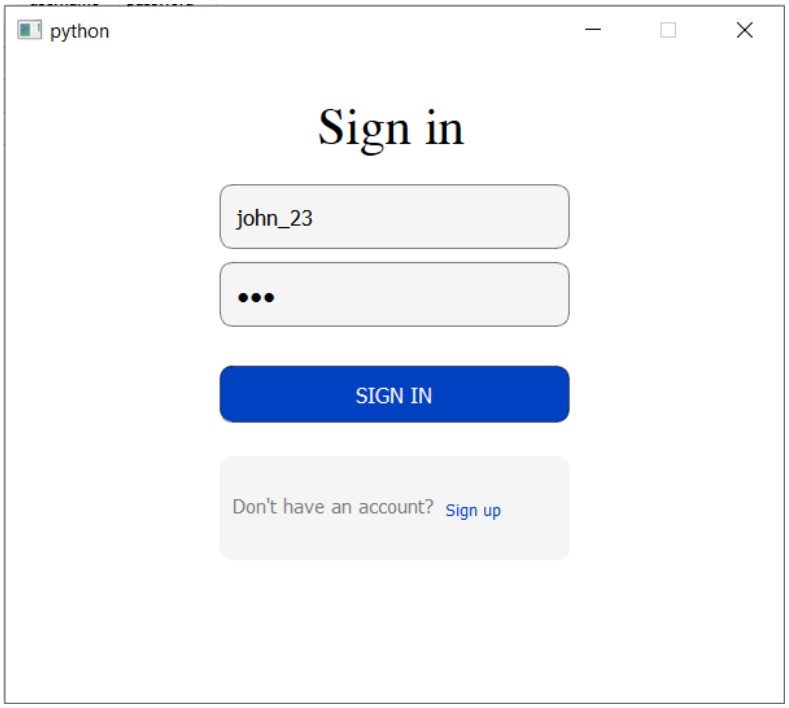



Рисунок 3.9 – Сторінка автентифікації

Методи класу LoginScreen:

– __init__: ініціалізує вікно входу, налаштовує елементи інтерфейсу та зв'язує кнопки з відповідними методами;

– getUserId: виконує запит до бази даних для отримання ідентифікатора користувача за введеними ім'ям та паролем;

– loginFunction: виконує перевірку облікових даних користувача, входить в систему за умови успішної автентифікації, або відображає повідомлення про помилку;

– goToMainWindow: Перемикає інтерфейс на головне вікно додатку після успішного входу;

– goToSignUpScreen: Перемикає інтерфейс на екран реєстрації для нових користувачів.

Основні функції класу LoginScreen:

– ініціалізація: завантажує графічний інтерфейс з файлу, налаштовує поля для введення імені користувача та пароля, зв'язує кнопки з методами для обробки подій;

– взаємодія з базою даних: перевіряє наявність користувача з введеними даними в базі даних, повертає ідентифікатор користувача, якщо він існує;

– авторизація: перевіряє правильність введених облікових даних, переходить до головного вікна додатку у разі успішного входу;

– перемикання між екранами: забезпечує можливість переходу до екрану реєстрації нових користувачів.

Таким чином, клас LoginScreen є важливим компонентом застосунку, який відповідає за процес автентифікації користувачів та забезпечує зручний інтерфейс для входу та реєстрації.

***Клас SignUpScreen*** (додаток В). Клас SignUpScreen є частиною графічного інтерфейсу користувача (GUI) для додатку, який забезпечує функціональність реєстрації нових користувачів (рис. 3.10). Клас розроблений за допомогою бібліотеки PyQt5 та взаємодіє з базою даних SQLite для збереження нових

облікових записів. Забезпечує інтерфейс для реєстрації нових користувачів в додаток. Спадкує від QDialog, що дозволяє створювати діалогове вікно.

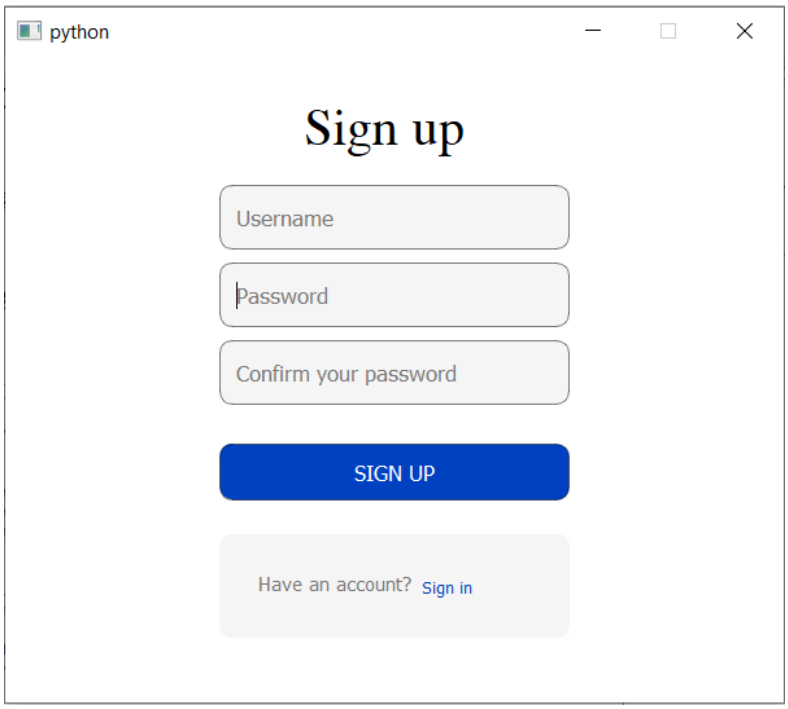



Рисунок 3.10 – Сторінка реєстрації

Методи класу SignUpScreen:

– __init__: ініціалізує вікно реєстрації, налаштовує елементи інтерфейсу та зв'язує кнопку реєстрації з відповідним методом;

– signUpFunction: виконує перевірку введених даних та реєструє нового користувача у базі даних;

– showDialog: відображає діалогове вікно з повідомленням про успішну реєстрацію;

– handleMessageBoxButton: обробляє натискання кнопки в діалоговому вікні;

– handleFinished: обробляє завершення діалогового вікна;

– goToLoginWindow: перемикає інтерфейс на екран входу після успішної реєстрації.

Основні функції класу SignUpScreen:

– ініціалізація: завантажує графічний інтерфейс з файлу, налаштовує поля для введення імені користувача та пароля, включаючи підтвердження пароля, зв'язує кнопку реєстрації з методом для обробки подій;

– реєстрація користувача: перевіряє правильність та заповненість введених даних, порівнює пароль та його підтвердження, зберігає новий обліковий запис у базі даних;

– відображення повідомлень: відображає діалогове вікно з повідомленням про успішну реєстрацію, обробляє натискання кнопки в діалоговому вікні та переключає інтерфейс на екран входу;

– перемикання між екранами: забезпечує можливість переходу до екрану входу після успішної реєстрації.

Таким чином, клас SignUpScreen є важливим компонентом застосунку, який відповідає за процес реєстрації нових користувачів та забезпечує зручний інтерфейс для створення нових облікових записів.

Для побудови прогнозної моделі створено ***клас EnergyPrediction*** (додаток Г). Оскільки для побудови прогнозу обрано модель Prophet, перевіряти дані отриманого часового ряду на стаціонарність та виявляти його структуру немає потреби, так як модель автоматизовано виконує ці операції, здійснюючи декомпозицію часового ряду [21]. Ця модель вимагає специфічного формату вхідних даних, до якого вони були перетворені (рис. 3.11), у вигляді таблиці, де стовпець ds – містить дату, а стовпець y – значення енергоспоживання з використанням функції prepare_data_for_prophet класу EnergyPrediction (рис. 3.12). Функція повертає підготовлений датафрейм для подальшого використання у прогнозуванні спожитої електроенергії.

Функція predict використовує бібліотеку Prophet для прогнозування майбутніх значень спожитої електроенергії. Спершу модель Prophet ініціалізується та навчається на переданих даних, які мають відповідний формат (з колонками "ds" та "y"). Після навчання моделі створюється датафрейм з майбутніми датами на 30 днів уперед. Модель пророкує значення для цього періоду, а отримані прогнози

візуалізуються за допомогою функції plot_plotly, що створює графік з передбаченими значеннями.

| | DS | Y |
|---|---|---|
| 0 | 2023-12-10 | 9.590761 |
| 1 | 2023-12-11 | 28.519590 |
| 2 | 2023-12-12 | 18.183677 |
| 3 | 2023-12-13 | 18.072467 |
| 4 | 2023-12-14 | 17.893572 |

Рисунок 3.11 – Формат вхідних даних для Prophet

```
class EnergyPrediction:
    def __init__(self, data=None):
        self.data = data
        self.model = None

    def prepare_data_for_prophet(self):
        df_prophet = self.data.reset_index().rename(columns={"datetime": "ds", "energy
        return df_prophet

    def load_model(self, model_path):
        self.model = load(model_path)

    def predict(self, periods):
        future = self.model.make_future_dataframe(periods=periods)
        prediction = self.model.predict(future)
        return prediction
```

Рисунок 3.12 – Функції класу EnergyPrediction

Функція evaluate_model обчислює три основні метрики помилки для оцінки точності прогнозів моделі: коефіцієнт детермінації $R^2$ – показує ступінь апроксимації емпіричних даних побудованою моделлю, MSE враховує квадрати різниць, що робить її чутливою до великих помилок, MAPE виражає помилку у відсотках від істинних значень (рис. 3.13).

```
def evaluate_model(self, true_data, prediction):
    true_values = true_data['energy consumption, kWh'].values
    predicted_values = prediction['yhat'][:len(true_values)].values

    mse = mean_squared_error(true_values, predicted_values)
    rmse = np.sqrt(mse)
    mape = mean_absolute_percentage_error(true_values, predicted_values)
    r2 = r2_score(true_values, predicted_values)

    metrics = pd.DataFrame({
        "Metric": ["MSE", "RMSE", "MAPE", "R2"],
        "Value": [mse, rmse, mape, r2]
    })
    return metrics
```

Рисунок 3.13 – Функція evaluate_model

***Клас ForecastWindow*** виконує роль графічного інтерфейсу для відображення прогнозів щодо енергоспоживання наступного місяця на основі аналізу даних з бази даних (додаток Д). Основні завдання цього класу включають ініціалізацію GUI, завантаження даних з бази даних, використання класу EnergyPrediction для прогнозування, відображення результатів прогнозування у відповідних елементах інтерфейсу та управління взаємодією з користувачем через кнопки та інші елементи інтерфейсу.

## 3.4 Інтерфейс системи. Аналіз та прогнозування споживання електроенергії

Головне вікно розробленого застосунку зображено на рисунку 3.14. Для отримання прогнозу споживання електроенергії необхідно у таблиці, яка міститься посередині вікна, обрати клієнта, клацнувши лівою кнопкою миші відповідний рядок таблиці та натиснути кнопку Forecast. Після цього буде створено об’єкт «курсор» для виконання SQL-запитів і операцій з базою даних. Курсор є механізмом, який обирає id розумного лічильника відповідного клієнта та отримує усі дані цього лічильника з бази даних для подальшої їх обробки та аналізу:

```
c = conn.cursor()
 query = 'SELECT date, energy_consumed FROM
energyConsumption WHERE meterId = ?'
c.execute(query, (self.meterId,))
        rows = c.fetchall()
```

Завдяки цьому у вікні буде відкрито таблицю, яка містить прогнозні значення енергоспоживання для обраного домогосподарства (рис. 3.15). Натиснувши кнопку Bask to main screen, буде здійснено перехід до початкової сторінки.

Для перегляду прогнозу у графічному вигляді необхідно натиснути кнопку Plot graphs. Буде відображено у графічному вигляді прогноз споживання електроенергії для обраного домогосподарства (рис. 3.16). Дані агреговані до щоденного рівня за допомогою resample('D').sum(), що показує сумарне споживання енергії за кожен день.

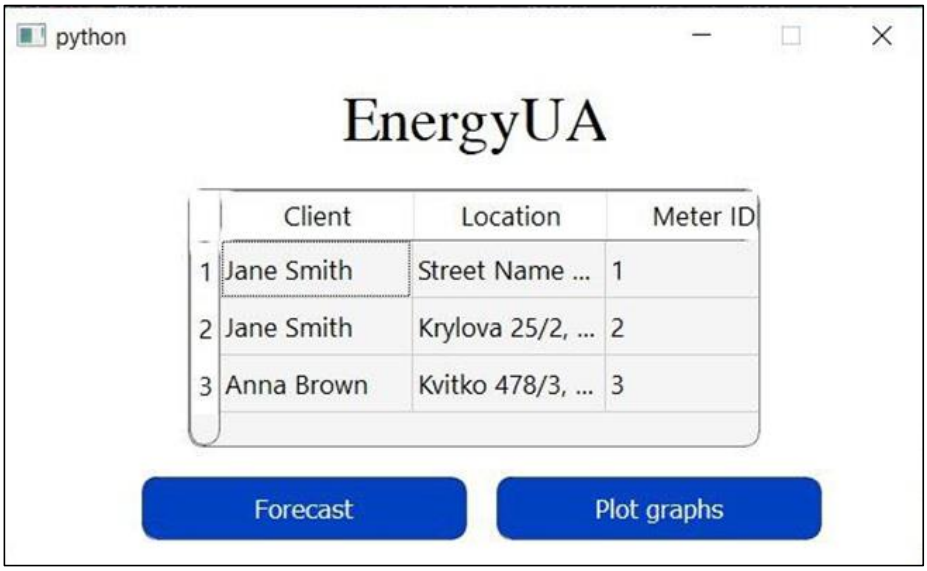



Рисунок 3.14 – Головне вікно застосунку

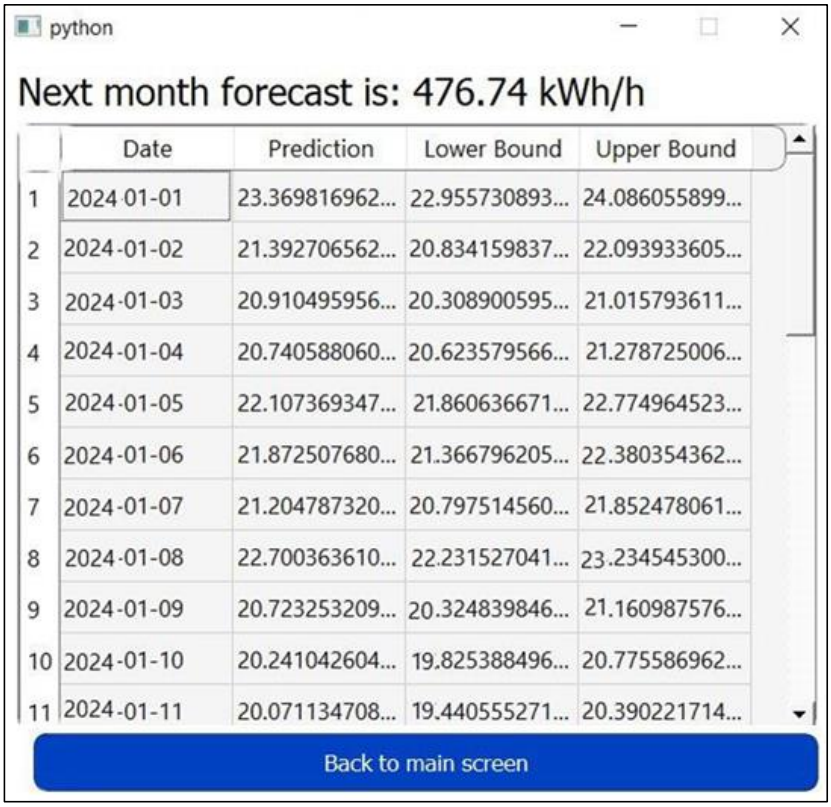



Рисунок 3.15 – Відображення прогнозу у табличному вигляді

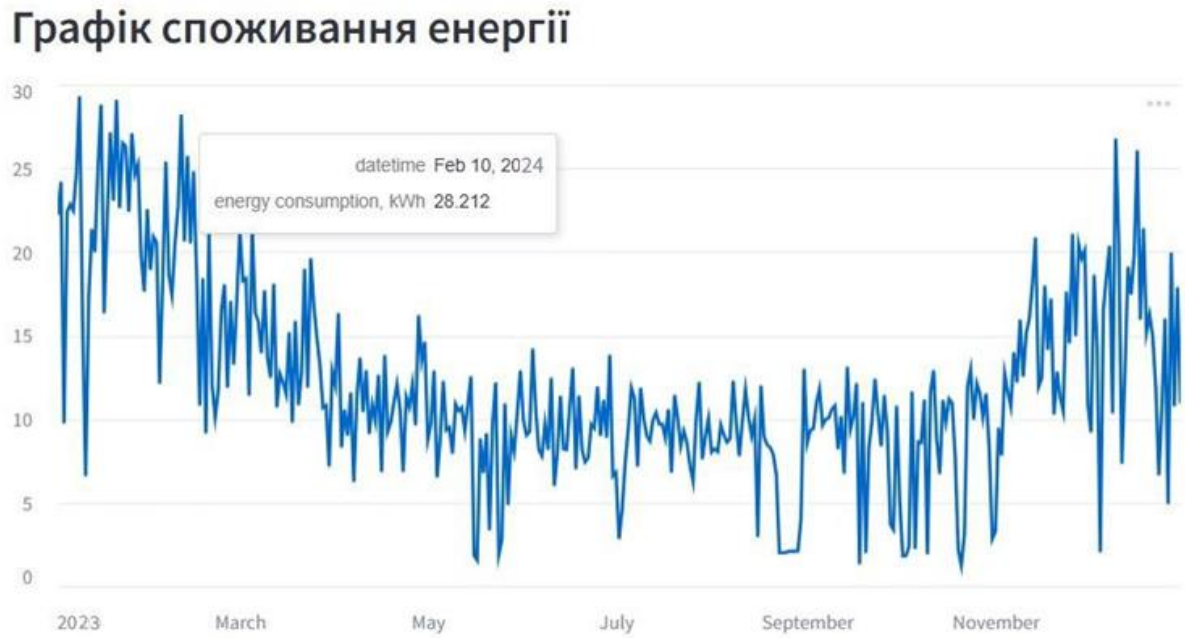



Рисунок 3.16 – Графік споживання електроенергії обраного домогосподарства

Нижче у вікні відображено графік прогнозної моделі (рис. 3.17). Для побудови прогнозної моделі набір даних клієнта розбивається на навчаючу та тестову множини (80% та 20%). Підгонка моделі здійснюється на даних навчаючої множини, тестова множина використовується для оцінки якості побудованої моделі.

Прогнозна модель була створена із використанням параметрів, прийнятий у Prophet по замовчуванню. Отримані результати подаються у вигляді графіка цін на електроенергію з реальними значеннями та розрахованими за побудованою моделлю (рис. 3.17).

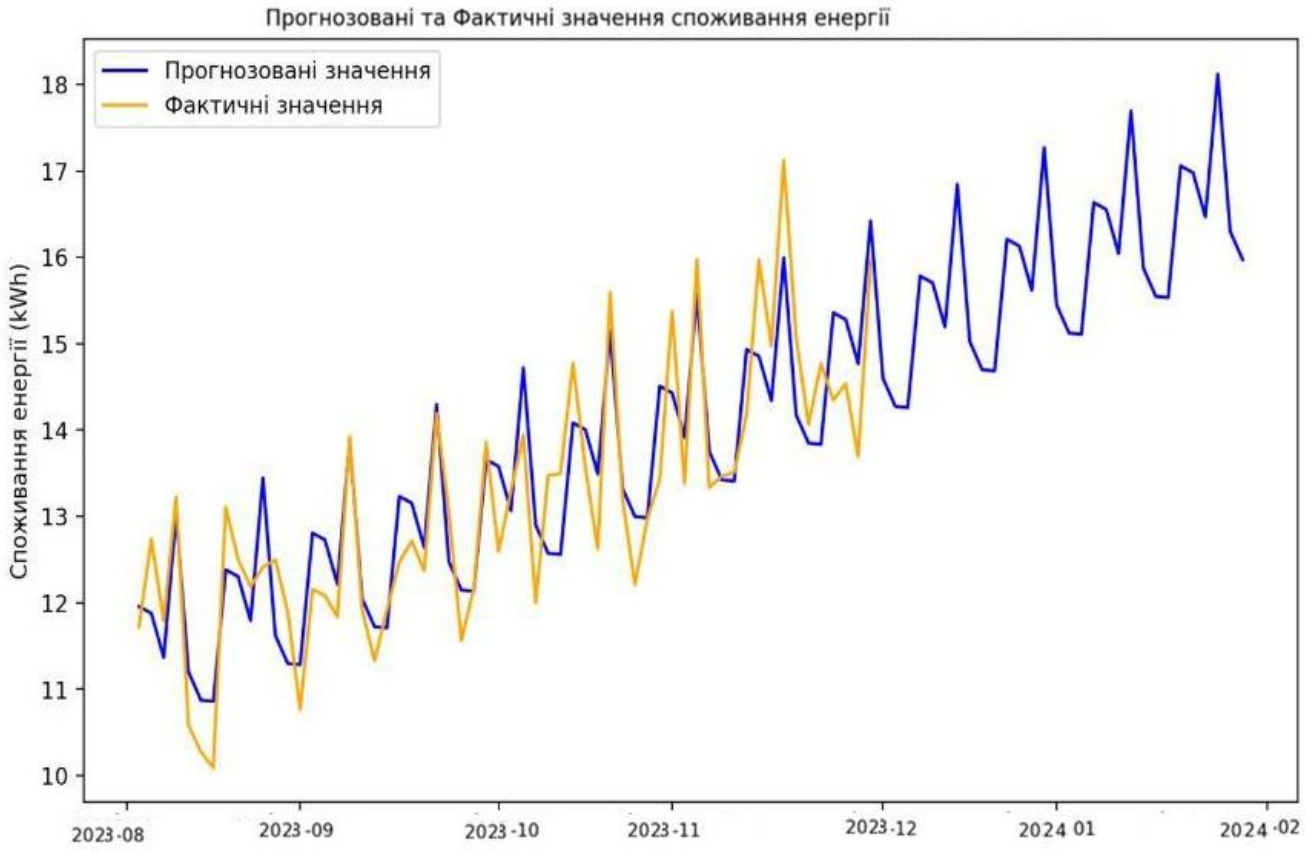



Рисунок 3.17 – Прогноз енергоспоживання

Для оцінки якості та точності прогнозу було використано такі метрики:

– коефіцієнт детермінації $R^2$, який дозволяє оцінити ступінь апроксимації побудованою моделлю зміни спожитої електроенергії:

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{n}(y_t - \bar{y}_t)^2}; \quad (3.2)$$

– середньоквадратичну помилку MSE (англ. Mean Squared Error):

$$MSE = \frac{1}{n}\sum_{t}^{n}(y_t - \hat{y}_t)^2; \quad (3.3)$$

– корінь квадратний із середньоквадратичної помилки (англ. Root Mean Squared Error):

$$RMSE = \sqrt{\frac{1}{n}\sum_{t}^{n}(y_t - \hat{y}_t)^2}; \quad (3.4)$$

– середню абсолютну похибку у відсотках MAPE (англ. Mean Absolute Percentage Error):

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right|, \quad (3.5)$$

де $\hat{y}_t$ та $y_t$ – прогнозоване та фактичне значення рівнів ряду.

Після побудови графіку з прогнозом використовується функція evalute_model, яка розраховує описані вище метрики (рис. 3.18), використовуючи наведені формули. Отримані значення свідчать про високу якість та точність прогнозу (рис. 3.19).

```python
def evaluate_model(self, true_data, prediction):
    true_values = true_data['energy consumption, kWh'].values
    predicted_values = prediction['yhat'][:len(true_values)].values

    mse = mean_squared_error(true_values, predicted_values)
    rmse = np.sqrt(mse)
    mape = mean_absolute_percentage_error(true_values, predicted_values)
    r2 = r2_score(true_values, predicted_values)

    metrics = pd.DataFrame({
        "Metric": ["MSE", "RMSE", "MAPE", "R2"],
        "Value": [mse, rmse, mape, r2]
    })
    return metrics
```

Рисунок 3.18 – Функція evalute_model

Оцінка якості прогнозу

| | Metric | Value |
|---|---|---|
| 0 | MSE | 2.4672 |
| 1 | RMSE | 1.5707 |
| 2 | MAPE | 0.0706 |
| 3 | R2 | 0.9364 |

Рисунок 3.19 – Оцінка якості прогнозу

**Висновки до розділу 3**

У третьому розділі процес розробки та програмну реалізацію інтелектуальної системип рогнозування енергоспоживання домогосподарств. Система інтегрує інструменти для збору, обробки даних, побудови моделей машинного навчання та візуалізації результатів прогнозування. Проведено дослідження якості побудованих прогнозних моделей. Використані різні метрики для оцінки точності прогнозування, що дозволило ідентифікувати найбільш ефективні методи та алгоритми для конкретних завдань.

Таким чином, розроблена система забезпечує автоматизований аналіз та прогнозування споживання електроенергії домогосподарствами, що підвищує ефективність управління енергопостачанням.

# ВИСНОВКИ

Проведене дослідження дозволяє зробити наступні висновки.

Проведено глибокий аналіз глобальних тенденцій і наявних програмних засобів у сфері енергоспоживання. Це дозволило визначити ключові вимоги та підходи до розробки ефективної системи прогнозування.

Досліджено та впроваджено методи підготовки даних для прогнозування енергоспоживання, зокрема обробка історичних даних про споживання електроенергії домогосподарствами. Використано методи очищення даних, нормалізації та формування часових рядів.

Обґрунтовано вибір технологій та інструментальних засобів для розробки системи. Використання Python та його бібліотек, таких як numpy, pandas, matplotlib, keras, scikit-learn, дозволило створити потужну основу для аналізу даних та побудови прогнозних моделей. PyQt5 забезпечив створення зручного графічного інтерфейсу користувача, а SQLite – надійне зберігання даних.

Було розроблено та здійснено програмну реалізацію системи прогнозування споживання електроенергії домогосподарствами. Система інтегрує інструменти для збору, обробки даних, побудови моделей машинного навчання та візуалізації результатів прогнозування.

Проведено дослідження якості побудованих прогнозних моделей. Використані різні метрики для оцінки точності прогнозування, що дозволило ідентифікувати найбільш ефективні методи та алгоритми для конкретних завдань.

Таким чином, розроблена система забезпечує автоматизований аналіз та прогнозування споживання електроенергії домогосподарствами, що підвищує ефективність управління енергопостачанням. Використання інтелектуальних моделей прогнозування сприяє зниженню витрат на закупівлю електроенергії та забезпечує стабільність роботи енергопостачаючих підприємств.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. World Energy Outlook 2019, International Energy Agency. URL: https://www.iea.org/reports/world-energy-outlook-2019 (дата звернення: 01.05.2024).

2. Renewables 2020, International Energy Agency. URL: https://www.iea.org/reports/renewables-2020 (дата звернення: 01.05.2024).

3. Deb C., Zhang F., Yang J., Lee S. & Shah K. A review on time series forecasting techniques for building energy consumption. Renewable and Sustainable Energy Reviews. Vol. 74, 2017. P. 902-924.

4. Ahmad A., Hassan M., Abdullah M. A review on applications of ANN and SVM for building electrical energy consumption forecasting. Renewable and Sustainable Energy Reviews. Vol. 33, 2014/ P. 102-109.

5. Nti I., Teimeh M., Nyarko-Boateng O. & Adekoya, A. Electricity load forecasting: a systematic review. Journal of Electrical Systems and Information Technology. Vol. 7(1), 2020. P. 1-19.

6. Андрусенко Ю.О. Аналіз основних моделей прогнозування часових рядів. Збірник наукових праць Харківського національного університету Повітряних сил. 2020. № 3(65). С. 91-96.

7. Different types of Time-series Forecasting Models. Data Analytics : website. URL: https://vitalflux.com/different-types-of-time-series-forecasting-models/ (дата звернення: 19.05.2024).

8. Top Energy Forecasting Software. Predictive Analytics Today : website. URL: https://www.predictiveanalyticstoday.com/top-energy-forecasting-software/ (дата звернення: 9.03.2024)

9. 10 Open-Source Tools for Time Series Forecasting with Python : website. URL: https://www.maartengrootendorst.com/blog/open-source/ (дата звернення: 19.03.2024)

10. Energy Demand Forecasting in the Cloud : website. URL: https://www.geospatialmedia.net/article/energy-demand-forecasting-in-the-cloud (дата звернення: 19.03.2024)

11. Spyder IDE : вебсайт. URL: https://www.spyder-ide.org/ (дата звернення: 15.04.2024).

12. Підручник з Python – Документація : вебсайт. URL: https://docs.python.org/uk/3/tutorial/index.html (дата звернення: 01.05.2024)

13. Основи роботи з SQLite : вебсайт. URL: https://proglib.io/p/samouchitel-po-python-dlya-nachinayushchih-chast-22-osnovy-raboty-s-sqlite-2023-06-15 (дата звернення: 15.04.2024)

14. NumPy & Pandas : вебсайт. URL: https://numpy.org/, https://pandas.pydata.org/ (дата звернення: 15.06.2024).

15. Порівняння найкращих інструментів машинного навчання TENSORFLOW, KERAS, SCIKIT-LEARN I PYTORCH : вебсайт. URL: https://www.zfort.com.ua/blog/porivnyannya-naikrashikh-instrumentiv-mashinnogo-navchannya-tensorflow-keras (дата звернення: 2.05.2024).

16. Найпопулярніші бібліотеки Python : вебсайт. URL: https://foxminded.ua/ru/biblioteki-python/ (дата звернення: 12.05.2024).

17. Matplotlib & Seaborn : вебсайт. URL: https://matplotlib.org/ (дата звернення: 15.06.2024).

18. PyQt5 : вебсайт. URL: https://www.tutorialspoint.com/pyqt5/index.htm (дата звернення: 7.05.2024)/

19. Qt Group : вебсайт. URL: https://www.qt.io/download-dev (дата звернення: 15.04.2024).

20. Forecasting at Scale : website. URL: https://peerj.com/preprints/3190.pdf (дата звернення: 16.05.2024).

21. Prophet – Automatic Forecasting Procedure : website. URL: https://pypi.org/project/fbprophet/ (дата звернення: 16.05.2024).

22. Що таке розумні лічильники : вебсайт. URL: https://global.linyang.com/uk/news/what-are-smart-meters/ (дата звернення: 28.03.2024)

23. Box G., Jenkins G., Reinsel G. & Ljung G. (2015). Time series analysis: forecasting and control. John Wiley & Sons.

24. Прогнозування та аналіз часових рядів : методичні вказівки / укл. Юрченко М. Є. Чернігів : ЧНТУ, 2018. 78 с.

25. Долгіх А.О., Байбуз О.Г. Аналіз методів, моделей та програмних засобів прогнозування часових рядів. Відкриті інформаційні та комп'ютерні інтегровані технології. 2018. №79. С. 74-85.

# ДОДАТОК А

## Код класу MainWindow

```
class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        loadUi("mainWindow.ui", self)
        self.tableWidget.setHorizontalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
        self.loadData()
        self.forecastButton.clicked.connect(lambda: self.runForecast(self.mId))

    def runForecast(self, mId):
        forecast = ForecastWindow(mId)
        forecast.mId = mId  # Set mId attribute after loading the UI
        widget.addWidget(forecast)
        widget.setCurrentIndex(widget.currentIndex() + 1)

    def loadData(self):
        conn = sqlite3.connect('db.db')
        c = conn.cursor()

        c.execute("""
            SELECT clients.firstname || ' ' || clients.lastname AS client,
                meters.street || ' ' || meters.streetNumber || '/' || meters.box || ', ' || meters.city || ', ' ||
meters.country AS location,
                meters.meterId
            FROM meters
            INNER JOIN clients ON meters.clientId = clients.clientId
        """)
        rows = c.fetchall()
        for row_number, row_data in enumerate(rows):
            self.tableWidget.insertRow(row_number)
            for column_number, data in enumerate(row_data):
                item = QTableWidgetItem(str(data))
                self.tableWidget.setItem(row_number, column_number, item)
        self.tableWidget.itemSelectionChanged.connect(self.on_selection_changed)

    def on_selection_changed(self):
        selected_items = self.tableWidget.selectedItems()
        if selected_items:
            row = selected_items[0].row()
            meter_id_item = self.tableWidget.item(row, 2)
            meter_id = meter_id_item.text() if meter_id_item else None
            if meter_id:
                self.mId = meter_id
```

# ДОДАТОК Б

## Код класу LoginScreen

```
class LoginScreen(QDialog):
    def __init__(self, parent=None, stacked_widget=None):
        super(LoginScreen, self).__init__(parent)
        self.stacked_widget = stacked_widget
        loadUi("login.ui", self)
        self.passwordLineEdit.setEchoMode(QLineEdit.Password)
        self.usernameLineEdit.setPlaceholderText("Username")
        self.passwordLineEdit.setPlaceholderText("Password")
        self.toSignUpScreenButton.clicked.connect(self.goToSignUpScreen)
        self.loginButton.clicked.connect(self.loginFunction)

    def getUserId(self, username, password):
        username = self.usernameLineEdit.text()
        password = self.passwordLineEdit.text()

        conn = sqlite3.connect("db.db")
        c = conn.cursor()
        query = 'SELECT userId FROM users WHERE username = ? AND password = ?'
        c.execute(query, (username, password))

        userId = c.fetchone()
        conn.close()
        return userId[0] if userId is not None else None

    def loginFunction(self):
        username = self.usernameLineEdit.text()
        password = self.passwordLineEdit.text()

        conn = sqlite3.connect("db.db")
        c = conn.cursor()

        query = 'SELECT password FROM users WHERE username = ?'
        c.execute(query, (username,))

        row = c.fetchone()

        if row is not None:
            passwordFromDB = row[0]
            if passwordFromDB == password:
                uId = self.getUserId(username, password)
                self.goToMainWindow()
            else:
                self.errorLabel.setText("Incorrect password.")
        else:
            self.errorLabel.setText("User not found.")
```

```
    def goToMainWindow(self):
        self.main_window = MainWindow()
        self.stacked_widget.addWidget(self.main_window)
        self.stacked_widget.setCurrentWidget(self.main_window)

    def goToSignUpScreen(self):
        sign_up = SignUpScreen(stacked_widget=self.stacked_widget)
        self.stacked_widget.addWidget(sign_up)
        self.stacked_widget.setCurrentWidget(sign_up)

class SignUpScreen(QDialog):
    def __init__(self, parent=None, stacked_widget=None):
        super(SignUpScreen, self).__init__(parent)
        self.stacked_widget = stacked_widget
        loadUi("signUp.ui", self)
        self.usernameInputField.setPlaceholderText("Username")
        self.passwordInputField.setPlaceholderText("Password")
        self.passwordConfirmationLineEdit.setPlaceholderText("Confirm your password")
        self.passwordInputField.setEchoMode(QLineEdit.Password)
        self.passwordConfirmationLineEdit.setEchoMode(QLineEdit.Password)
        self.signUpButton.clicked.connect(self.signUpFunction)

    def signUpFunction(self):
        user = self.usernameInputField.text()
        password = self.passwordInputField.text()
        passwordConfirmation = self.passwordConfirmationLineEdit.text()

        if len(user) == 0 or len(password) == 0 or len(passwordConfirmation) == 0:
            self.errorLabel.setText("Please fill in all fields.")
        elif password != passwordConfirmation:
            self.errorLabel.setText("Passwords don't match.")
        elif len(password) < 6:
            self.errorLabel.setText("Password should be at least 6 characters.")
        else:
            conn = sqlite3.connect('db.db')
            c = conn.cursor()
            userInfo = [user, password]
            c.execute('INSERT INTO users (username, password) VALUES (?, ?)', userInfo)

            conn.commit()
            conn.close()
            self.showDialog()

    def showDialog(self):
        dialog = QMessageBox(self)
        dialog.setWindowTitle('SUCCEDED')
        dialog.setText(f'Successfully registered!')
        dialog.setIcon(QMessageBox.Information)
        dialog.setStandardButtons(QMessageBox.Ok)
```

```
    dialog.finished.connect(self.handleFinished)
    dialog.buttonClicked.connect(self.handleMessageBoxButton)
    dialog.exec_()

def handleMessageBoxButton(self,  button):
    if button.text()  == 'Ok':
        self.goToLoginWindow()

def handleFinished(self):
    self.goToLoginWindow()

def goToLoginWindow(self):
    login = LoginScreen(stacked_widget=self.stacked_widget)
    self.stacked_widget.addWidget(login)
    self.stacked_widget.setCurrentWidget(login)
```

# ДОДАТОК В

## Код класу SignUpScreen

```
class SignUpScreen(QDialog):
    def __init__(self, parent=None, stacked_widget=None):
        super(SignUpScreen, self).__init__(parent)
        self.stacked_widget = stacked_widget
        loadUi("signUp.ui", self)
        self.usernameInputField.setPlaceholderText("Username")
        self.passwordInputField.setPlaceholderText("Password")
        self.passwordConfirmationLineEdit.setPlaceholderText("Confirm your password")
        self.passwordInputField.setEchoMode(QLineEdit.Password)
        self.passwordConfirmationLineEdit.setEchoMode(QLineEdit.Password)
        self.signUpButton.clicked.connect(self.signUpFunction)

    def signUpFunction(self):
        user = self.usernameInputField.text()
        password = self.passwordInputField.text()
        passwordConfirmation = self.passwordConfirmationLineEdit.text()

        if len(user) == 0 or len(password) == 0 or len(passwordConfirmation) == 0:
            self.errorLabel.setText("Please fill in all fields.")
        elif password != passwordConfirmation:
            self.errorLabel.setText("Passwords don't match.")
        elif len(password) < 6:
            self.errorLabel.setText("Password should be at least 6 characters.")
        else:
            conn = sqlite3.connect('db.db')
            c = conn.cursor()
            userInfo = [user, password]
            c.execute('INSERT INTO users (username, password) VALUES (?, ?)', userInfo)

            conn.commit()
            conn.close()
            self.showDialog()

    def showDialog(self):
        dialog = QMessageBox(self)
        dialog.setWindowTitle('SUCCEDED')
        dialog.setText(f'Successfully registered!')
        dialog.setIcon(QMessageBox.Information)
        dialog.setStandardButtons(QMessageBox.Ok)

        dialog.finished.connect(self.handleFinished)
        dialog.buttonClicked.connect(self.handleMessageBoxButton)
        dialog.exec_()

    def handleMessageBoxButton(self, button):
```

```
    if button.text()  == 'Ok':
        self.goToLoginWindow()

def handleFinished(self):
    self.goToLoginWindow()

def goToLoginWindow(self):
    login = LoginScreen(stacked_widget=self.stacked_widget)
    self.stacked_widget.addWidget(login)
    self.stacked_widget.setCurrentWidget(login)
```

# ДОДАТОК Г

## Код класу EnergyPrediction

```
class EnergyPrediction:
    def __init__(self, data=None):
        self.data = data
        self.model = None

    def prepare_data_for_prophet(self):
        df_prophet = self.data.reset_index().rename(columns={"datetime": "ds", "energy consumption, kWh": "y"})
        return df_prophet

    def load_model(self, model_path):
        self.model = load(model_path)

    def predict(self, periods):
        future = self.model.make_future_dataframe(periods=periods)
        prediction = self.model.predict(future)
        return prediction
```

# ДОДАТОК Д

## Код класу ForecastWindow

```
class ForecastWindow(QDialog):
    def __init__(self, mId):
        super(ForecastWindow, self).__init__()
        loadUi("forecasting.ui", self)
        self.meterId = mId

        self.layout = QVBoxLayout(self)

        self.forecastLabel = self.findChild(QLabel, 'forecastLabel')
        if self.forecastLabel:
            self.layout.addWidget(self.forecastLabel)

        self.tableWidget = self.findChild(QTableWidget, 'tableWidget')
        if self.tableWidget:
            self.layout.addWidget(self.tableWidget)

        self.backToMainScreen = self.findChild(QPushButton, 'backToMainScreen')
        if self.backToMainScreen:
            self.backToMainScreen.clicked.connect(self.goBackToMain)
            self.layout.addWidget(self.backToMainScreen)

        self.loadData()

    def loadData(self):
        conn = sqlite3.connect("db.db")
        c = conn.cursor()

        query = 'SELECT date, energy_consumed FROM energyConsumption WHERE meterId = ?'
        c.execute(query, (self.meterId,))
        rows = c.fetchall()

        df = pd.DataFrame(rows, columns=['datetime', 'energy consumption, kWh'])
        df = df.set_index('datetime')
        df.index = pd.to_datetime(df.index, format="%Y-%m-%d %H:%M:%S")

        if df.isnull().sum().any():
            df.interpolate(method='linear', inplace=True)

        df_daily = df.resample('D').sum()

        energy_predictor = EnergyPrediction(data=df_daily)
        energy_predictor.load_model(r'prophet_model.pkl')

        periods = 30
        prediction = energy_predictor.predict(periods)
```

```
        sum_forecast = prediction['yhat'][-periods:].sum()
        self.forecastLabel.setText(f"Next month forecast is: {sum_forecast:.2f} kWh/h")
        self.display_forecast(prediction.head(30))

        prediction[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].to_csv("predicted_values.csv", index=False)

    def display_forecast(self, prediction):
        self.tableWidget.clearContents()
        self.tableWidget.setRowCount(len(prediction))
        self.tableWidget.setColumnCount(4)
        self.tableWidget.setHorizontalHeaderLabels(['Date', 'Prediction', 'Lower Bound', 'Upper Bound'])

        for i, row in prediction.iterrows():
            self.tableWidget.setItem(i, 0, QTableWidgetItem(row['ds'].strftime('%Y-%m-%d')))
            self.tableWidget.setItem(i, 1, QTableWidgetItem(str(row['yhat'])))
            self.tableWidget.setItem(i, 2, QTableWidgetItem(str(row['yhat_lower'])))
            self.tableWidget.setItem(i, 3, QTableWidgetItem(str(row['yhat_upper'])))

    def goBackToMain(self):
        main_window = self.parent().findChild(MainWindow)
        if main_window:
            self.parent().setCurrentWidget(main_window)
```