Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістра

за спеціальністю 121 – Інженерія програмного забезпечення

На тему: Дослідження та реалізація енергоефективності у системах «Розумний будинок»

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.

Студент гр. ЗІПЗ-23м _____ Афонченко А. І.

Керівник кваліфікаційної роботи _____ Саітгареєв Н. Х.

Економіко-організаційна частина _____ _____

Нормоконтроль _____ Саітгареєв Н. Х.

Завідувач кафедри _____ Стрюк А. М.

Кривий Ріг
2024

Криворізький національний університет
Факультет: Інформаційних технологій
Кафедра: Моделювання та програмного забезпечення
Ступінь вищої освіти: магістр
Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедри
_____ А. М. Стрюк
« » _____ 20 ____р.

ЗАВДАННЯ
на кваліфікаційну роботу

Студенту групи ЗІПЗ-23м Афонченку Андрію Ігоровичу
1. На тему: Дослідження та реалізація енергоефективності у системах «Розумний будинок» затверджено наказом по КНУ № 273с від «15» квітня 2024 р.
2. Термін подання студентом закінченої роботи: «6» грудня 2024 р.
3. Вихідні дані по роботі: Розроблюваний програмний комплекс повинен: генерувати синтетичні дані про «Розумний будинок», аналізувати створений набір даних, надавати можливість створювати моделі штучних нейронних мереж для здійснення прогнозування режимів роботи систем нагрівання води, опалення та кондиціонування, виконувати прогнозування режимів роботи цих систем, виконувати оцінку економічної ефективності оптимізації режимів роботи цих систем.
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): Здійснити аналіз бізнес-процесів у галузі, аналіз актуальності, аналіз досліджень інших авторів, сформулювати задачі кваліфікаційної роботи, аналіз вимог до програмного комплексу, розробку програмного комплексу, аналіз результатів роботи програмного комплексу, оцінку економічного ефекту від впровадження програмного комплексу.
5. Перелік ілюстрованого матеріалу: Діаграма опорних точок зору, ієрархія зацікавлених сторін, схема структури набору даних, діаграми варіантів використання, діаграми діяльності, діаграми потоків даних, функціональні схеми, структурна схема програмного комплексу, діаграма класів, зображення архітектури штучних нейронних мереж, зображення графічного інтерфейсу користувача програмного комплексу.

Календарний план

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Написання вступу, аналіз бізнес процесів, формулювання актуальності теми	15.04.2024 – 19.04.2024
2	Аналіз досліджень інших авторів за темою, формування задач кваліфікаційної роботи	22.04.2024 – 26.04.2024
3	Визначення опорних точок зору до вимог, аналіз вимог до програмного комплексу	29.04.2024 – 10.05.2024
4	Розробка структури набору даних та моделювання варіантів використання	13.05.2024 – 24.05.2024
5	Моделювання процесів, потоків даних та функціональності	27.05.2024 – 07.06.2024
6	Моделювання структури та архітектури програмного комплексу	10.06.2024 – 21.06.2024
7	Моделювання архітектури штучних нейронних мереж	24.06.2024 – 05.07.2024
8	Розробка модулів для здійснення генерації набору даних, аналізу набору даних і моделей	08.07.2024 – 19.07.2024
9	Розробка модулів для здійснення обробки набору даних, файлових операцій і математичних операцій	22.07.2024 – 02.08.2024
10	Розробка моделей штучних нейронних мереж та графічного інтерфейсу користувача	05.08.2024 – 16.08.2024
11	Аналіз результатів розробки програмного комплексу: аналіз головного меню, створення набору даних, створення та аналіз моделей, прогнозування даних, оцінка економічної ефективності	19.08.2024 – 27.09.2024
12	Обчислення собівартості розробки програмного комплексу, аналіз інноваційного ефекту	30.09.2024 – 11.10.2024
13	Формулювання висновків кваліфікаційної роботи	14.10.2024 – 25.10.2025
14	Перевірка кваліфікаційної роботи, внесення поправок	28.10.2024 – 22.11.2024
15	Завершення оформлення пояснювальної записки	25.11.2024 – 04.12.2024

Дата видачі завдання: «15» квітня 20 24 р.
Студент _____ Афонченко А. І.
Керівник роботи _____ Саітгареєв Н. Х.

РЕФЕРАТ

РОЗУМНИЙ БУДИНОК, ЕНЕРГЕТИЧНА ЕФЕКТИВНІСТЬ, ЕНЕРГІЯ, ЗБЕРЕЖЕННЯ ЕНЕРГІЇ, МАШИННЕ НАВЧАННЯ, АЛГОРИТМ, ШТУЧНА НЕЙРОННА МЕРЕЖА, МОДЕЛЬ

Пояснювальна записка: 131 сторінок, 9 таблиць, 33 рисунків, 4 додатка, 54 джерела.

Мета: розробка програмного комплексу для здійснення моделювання оптимізації споживання енергії у системах «Розумний будинок» на основі даних про поведінку мешканців і температурні умови.

Об’єкт: методи оптимізації споживання енергії в системах «Розумний будинок».

Предмет: методи підвищення енергетичної ефективності «Розумного будинку» шляхом аналізу поведінки мешканців і температурних умов із застосуванням інтелектуальних систем.

У першому розділі кваліфікаційної роботи виконаний аналіз досліджень інших авторів, аналіз бізнес-процесів у професійній галузі та формулювання актуальності.

У другому розділі кваліфікаційної роботи виконаний аналіз вимог до програмного комплексу та здійснене його проектування.

У третьому розділі кваліфікаційної роботи описана розробка основних складових програмного комплексу.

У четвертому розділі кваліфікаційної роботи описані результати роботи програмного комплексу.

У п’ятому розділі здійснене обчислення інноваційного ефекту від вдосконалення режимів роботи систем нагрівання води, опалення та кондиціонування.

Основні тези кваліфікаційної роботи були представлені на XVII Всеукраїнській науково-практичній WEB конференції аспірантів, студентів та молодих вчених «Комп’ютерні інтелектуальні системи та мережі».

ABSTRACT

SMART HOME, ENERGY EFFICIENCY, ENERGY, ENERGY CONSERVATION, MACHINE LEARNING, ALGORITHM, ARTIFICIAL NEURAL NETWORK, MODEL

Explanatory note: 131 pages, 9 tables, 33 figures, 4 appendices, 54 sources.

Purpose: development of a software package for modeling the optimization of energy consumption in Smart Home systems based on data on the behavior of residents and temperature conditions.

Object: methods for optimizing energy consumption in Smart Home systems.

Subject: methods for increasing the energy efficiency of a Smart Home by analyzing the behavior of residents and temperature conditions using intelligent systems.

The first section of the qualification work analyzes the research of other authors, analyzes business processes in the professional field, and formulates the relevance.

In the second section of the qualification work, an analysis of the requirements for the software complex was performed and its design was carried out.

In the third section of the qualification work, the development of the main components of the software complex is described.

In the fourth section of the qualification work, the results of the software complex are described.

In the fifth section, the calculation of the innovative effect from improving the operating modes of water heating, heating and air conditioning systems is carried out.

The main theses of the qualification work were presented at the XVII All-Ukrainian Scientific and Practical WEB Conference of Postgraduate Students, Students and Young Scientists "Computer Intelligent Systems and Networks".

ЗМІСТ

ВСТУП

На сьогоднішній день проблема енергетичної ефективності систем «Розумний будинок» набуває особливої актуальності у зв’язку із зростаючим попитом на подібні технологічні рішення та обмеженість невідновлюваних джерел енергії. Використання викопного палива, яке залишається основним джерелом енергії, завдає значної шкоди довкіллю, посилюючи проблему кліматичних змін. З огляду на це, виникає необхідність у впровадженні інноваційних рішень, спрямованих на оптимізацію споживання енергії та підвищення екологічної безпеки систем «Розумний будинок».

Світові тенденції зосереджені на інтеграції відновлюваних джерел енергії у системи «Розумний будинок» для зменшення залежності від викопного палива. Активно розвиваються енергозберігаючі технології – такі, як: «розумні» термостати, LED-освітлення та автоматизоване управління споживанням енергії. Методи машинного навчання та системи штучного інтелекту допомагають оптимізувати споживання енергії, зберігаючи комфорт мешканців.

Об’єктом цієї кваліфікаційної роботи є методи оптимізації споживання енергії в системах «Розумний будинок».

Предметом цієї кваліфікаційної роботи є методи підвищення енергетичної ефективності «Розумного будинку» шляхом аналізу поведінки мешканців і температурних умов із застосуванням інтелектуальних систем.

Мета цієї кваліфікаційної роботи полягає у розробці програмного комплексу для здійснення моделювання оптимізації споживання енергії у системах «Розумний будинок» на основі даних про поведінку мешканців і температурні умови.

Для досягнення визначеної мети цієї кваліфікаційної роботи застосовуються такі методи дослідження:

— аналіз літературних джерел;

— аналіз бізнес-процесів у професійній галузі;

— постановка задач кваліфікаційної роботи;

— аналіз вимог до розроблюваного програмного комплексу;

— проектування програмного комплексу;

— моделювання та генерація даних;

— методи машинного навчання для аналізу даних;

— розробка програмного комплексу;

— комп’ютерне моделювання для тестування розробленого програмного комплексу;

— проведення експериментів із використанням розробленого програмного комплексу;

— оцінка економічної ефективності впровадження розробленого програмного комплексу.

1 АНАЛІТИЧНИЙ ОГЛЯД ТА ПОСТАНОВКА ЗАДАЧ

1.1 Аналіз бізнес-процесів у галузі

Бізнес-процеси «Розумного будинку» містять сукупність дій та операцій, спрямованих на забезпечення автоматизації, безпеки, комфорту й енергетичної ефективності в домі за допомогою різноманітних технологічних рішень. Основні бізнес-процеси «Розумного будинку», зазвичай, такі:

— віддалений контроль та моніторинг різних систем будинку [1];

— автоматичне виконання сценаріїв на основі заданих умов [1];

— моніторинг систем безпеки [1];

— оптимізація використання енергії шляхом автоматичного керування, різноманітними системами [1];

— підключення до зовнішніх постачальників послуг для автоматичного відстеження та оплати рахунків [1];

— керування та синхронізація різноманітних «розумних» пристроїв [1];

— виявлення поломок або потреби у технічному обслуговуванні систем [1];

— збір та аналіз даних про використання систем і пристроїв для покращення управління будинком [1];

— забезпечення користувачів консультаціями, підтримкою та можливістю звернутися до служби технічної підтримки [1].

1.2 Формулювання актуальності теми

Актуальність цієї кваліфікаційної роботи зумовлена нагальною необхідністю у забезпеченні високого рівня енергетичної ефективності та екологічної безпеки систем «Розумний будинок». Ця потреба викликана стрімко зростаючим попитом на такі системи в усьому світі й критичним

виснаженням запасів корисних копалин, котрі використовуються як джерела енергії.

На сьогоднішній день левову частку світового виробництва енергії забезпечують невідновлювані викопні види палива, а саме: вугілля (35,67%), природний газ (22,82%) та ядерне паливо (2,15%) [2]. Проте наявні оцінки свідчать про те, що за теперішніх темпів видобутку існуючі світові запаси вугілля можуть повністю вичерпатися приблизно через 114 років, тоді як запаси природного газу будуть виснажені уже через 53 роки [3]. Ці тривожні факти змушують замислитися над необхідністю оптимізації використання енергії.

Крім проблеми виснаження невідновлюваних природних ресурсів, використання викопних видів палива, особливо вугілля, завдає величезної шкоди довкіллю [4]. Спалювання викопного палива призводить до забруднення повітря шкідливими речовинами, що спричиняє розвиток респіраторних захворювань у людей [4]. Також цей процес є однією з головних причин зміни клімату внаслідок глобального потепління та погіршення якості водних ресурсів [4].

Тим часом, ринок систем «Розумний будинок» демонструє стрімке зростання, і, згідно із прогнозами авторитетної компанії Statista, до 2025 року його обсяг сягне 174 мільярдів доларів США [5]. Такий показник яскраво ілюструє масштаби поширення даних систем і підкреслює нагальну потребу у забезпеченні їхньої енергетичної ефективності й екологічності.

Вищезазначені факти переконливо вказують на потребу упровадження інноваційних енергетично ефективних рішень у системи «Розумних будинків» із метою скорочення споживання енергії та мінімізації негативного впливу на довкілля. Підвищення енергетичної ефективності є одним із ключових завдань на шляху до забезпечення сталого розвитку людства та боротьби зі змінами клімату. Оптимізація споживання енергії системами «Розумний будинок» може знизити викиди парникових газів та сприяти досягненню Цілей сталого розвитку, визначених Організацією Об'єднаних Націй [6].

1.3 Аналіз досліджень інших авторів

1.3.1 Аналіз наявних протиріч

На основі аналізу літературних джерел можна виявити деякі протиріччя та проблемні питання, пов’язані з енергетичною ефективністю і системами «Розумний будинок»:

— існує потреба у підвищенні енергетичної ефективності в муніципальному секторі, проте реалізація відповідних проектів стикається з викликами – такими, як: бюджетні обмеження, необхідність залучення зовнішнього фінансування та сучасного програмного забезпечення [10];

— концепція «Розумних будинків» має потенціал для забезпечення енергетичної ефективності і збереження енергії, але існують виклики щодо вибору оптимальних методів і технологічних рішень [11];

— попри наявність досвіду впровадження «Розумних міст» у деяких країнах, в Україні ця концепція ще перебуває на початковому етапі розвитку [12].

1.3.2 Аналіз актуальності теми в дослідженнях інших авторів

Аналіз літературних джерел підтверджує, що актуальність досліджень у галузі енергетичної ефективності систем «Розумний будинок» існує із таких причин:

— енергетична ефективність та збереження енергії є важливими питаннями для економічного та соціального розвитку України, оскільки країна має менше досягнень у цій галузі, порівняно з більш розвиненими країнами [13];

— концепція «Розумних будинків» та «Розумних міст» розглядається як перспективний шлях до підвищення енергетичної ефективності і зменшення впливу на довкілля, але потребує подальших досліджень та розробок [14];

— існує необхідність вдосконалення управління енергією та реалізації проектів з енергетичної ефективності в муніципальному секторі, що може призвести до покращення умов життя, підвищення конкурентоспроможності та зменшення впливу на довкілля [10].

13.3 Аналіз ступеню вивченості проблеми і потреби у подальших дослідженнях

На основі аналізу літературних джерел можна зробити висновок, що проблема енергетичної ефективності і «Розумних будинків» досліджувалась різними авторами, але все ще потребує подальшого розвитку в декількох аспектах:

— існує потреба у подальших дослідженнях та розробках для вирішення викликів та обмежень, пов’язаних з методами забезпечення енергетичної ефективності і збереження енергії в «Розумних будинках» – такими, як: необхідність точного збору та обробки даних, складність проектування систем [11];

— концепція «Розумних міст» в Україні ще перебуває на початковій стадії розвитку, що вимагає додаткових досліджень та адаптації до місцевих умов для ефективного впровадження [12];

— потрібні подальші дослідження для оптимізації вибору та налаштування різних типів штучних нейронних мереж для різних завдань автоматизації та управління в системах «Розумний будинок» [15].

1.3.4 Аналіз методів досліджень інших авторів та їхніх результатів

У проаналізованих літературних джерелах автори використовували різноманітні методи та підходи для дослідження енергетичної ефективності і «Розумних будинків». Далі описані ці методи й отримані результати:

— SWOT-аналіз. Декілька авторів використовували SWOT-аналіз для оцінки ризиків та визначення сильних і слабких сторін, можливостей та

загроз у проектах, пов’язаних з «Розумними будинками» та енергетичною ефективністю [16]. Цей метод допоміг ідентифікувати внутрішні та зовнішні чинники, що впливають на успіх проекту;

— моделювання та симуляція. У дослідженні, присвяченому розробці системи «Розумний будинок» для приватного будинку, автори використовували моделювання електронних схем і системи за допомогою програмного забезпечення Proteus VSM [17]. Це дозволило виявити і виправити проблеми на етапі проектування до фізичної реалізації системи;

— порівняльний аналіз штучних нейронних мереж. В одному з досліджень автори порівнювали різні типи штучних нейронних мереж (прямі, рекурентні, з довгою короткочасною пам’яттю і вентильовані рекурентні мережі) для визначення оптимального вибору в системах «Розумного будинку» для контролю температури, освітлення та безпеки [15]. Результати показали, що різні типи штучних нейронних мереж краще підходять для різних завдань [15];

— вивчення світового досвіду. Деякі автори досліджували досвід впровадження концепції «Розумних міст» в Європейському Союзі, Японії, Сполучених Штатах Америки та Китайській Народній Республіці, аналізуючи їхню актуальність, ефективність та переваги [18]. Це допомогло сформувати комплексне розуміння концепції та визначити перспективи для України;

— аналіз потенціалу збереження енергії. У дослідженнях, присвячених проблемам енергетичної ефективності і збереження енергії в Україні, автори аналізували потенціал збереження енергії в різних секторах (житловий, промисловість, транспорт) і пропонували стратегії для підвищення енергетичної ефективності і скорочення споживання енергії [19];

— огляд методів і технологій. Декілька авторів представили огляд сучасних методів і технологій, спрямованих на забезпечення енергетичної ефективності і збереження енергії в «Розумних будинках» – таких, як: інтелектуальні будівельні системи, системи предиктивного управління та

системи нечіткої логіки. Автори також обговорювали виклики та обмеження цих методів [15].

Ці та інші методи дослідження, використані авторами, дозволили отримати цінні результати і висновки щодо енергетичної ефективності і «Розумних будинків», які можуть слугувати основою для проведення подальших досліджень та розробок у цій галузі.

1.4 Визначення задач кваліфікаційної роботи

Розробка моделі інтелектуальної інформаційної системи керування системами опалення, кондиціонування і нагрівання води у «Розумному будинку» потребує ретельного планування та проектування. Далі будуть окреслені основні задачі та етапи створення моделі, яка забезпечить демонстрацію ефективного керування системами опалення, кондиціонування та нагрівання води у «Розумному будинку».

Аналіз функціональних та нефункціональних вимог до програмного комплексу:

— збір та аналіз вимог до програмного комплексу;

— визначення функціональних та нефункціональних вимог до програмного комплексу.

Моделювання програмного комплексу:

— побудова діаграми опорних точок зору;

— побудова діаграми ієрархії зацікавлених сторін;

— розробка структури набору даних;

— побудова діаграм варіантів використання;

— побудова діаграм діяльності;

— побудова діаграм потоків даних;

— побудова функціональних схем;

— побудова структурної схеми;

— побудова діаграми класів;

— побудова архітектури моделей нейронних мереж.

Розробка програмного комплексу:

— реалізація програмної моделі за допомогою мови програмування Python згідно із розробленими UML-діаграмами;

— забезпечення модульної архітектуру для спрощення підтримки та розширення програмного комплексу.

Впровадження алгоритмів аналізу даних та ухвалення рішень у програмний комплекс:

— розробка модулю для здійснення генерації набору даних;

— розробка модулю для здійснення аналізу набору даних;

— розробка модулю для здійснення обробки набору даних;

— розробка моделі нейронної мережі для симуляції керування режимом роботи системи нагрівання води;

— розробка моделі нейронної мережі для симуляції керування системами опалення та кондиціонування;

— розробка модулю для здійснення файлових операцій;

— розробка модулю для здійснення математичних операцій;

— розробка графічного інтерфейсу користувача.

Аналіз результатів:

— демонстрація результатів аналізу набору даних;

— демонстрація аналізу точності моделей нейронної мережі для симуляції керування режимом роботи системи нагрівання води;

— демонстрація аналізу точності моделі нейронної мережі для симуляції керування режимом роботи систем опалення та кондиціонування;

— порівняння показників енергетичної ефективності в різних режимах роботи програмного комплексу;

— обчислення собівартості розробки та інноваційного ефекту впровадження програмного комплексу.

2 ФОРМУЛЮВАННЯ ВИМОГ ДО ПРОГРАМНОГО КОМПЛЕКСУ ТА ЙОГО МОДЕЛЮВАННЯ

2.1 Аналіз опорних точок зору до вимог

2.1.1 Аналіз опорних точок зору

Діаграма опорних точок зору є корисним засобом для проектування програмних комплексів, спрямованим на забезпечення всебічного розуміння інформаційної системи з різних точок зору зацікавлених сторін. Діаграма опорних точок зору складається з набору схематичних зображень або макетів, кожен із яких представляє ключовий аспект або сценарій використання інформаційної системи з погляду певної зацікавленої сторони – такої, як: кінцевий користувач, адміністратор, розробник тощо [21].

На рисунку 2.1 показана діаграма опорних точок зору для проектованого програмного комплексу.

Рисунок 2.1 – Діаграма визначених опорних точок зору до вимог

На рисунку 2.1 визначені такі опорні точки зору:

— збереження енергії;
— комфорт проживання;
— надійність інформаційної системи;
— безпека та конфіденційність.

На рисунку 2.1 визначені такі зацікавлені сторони:

— мешканці будинку;

— власники будинку;

— розробники інформаційної системи;

— постачальники енергії;

— екологічні організації.

Діаграма опорних точок зору, показана на рисунку 2.1, була розроблена за допомогою програмного комплексу Microsoft Visio [20].

2.1.2 Аналіз ієрархії зацікавлених сторін

Діаграма ієрархії зацікавлених сторі – це візуальне відображення ієрархічної структури організації, системи або складного об’єкту. Вона використовується для представлення зв’язків між різними рівнями або компонентами, де кожен елемент може мати множинні підпорядковані елементи [21].

На рисунку 2.2 показана діаграма ієрархії зацікавлених сторін відносно усіх визначених опорних точок зору. Визначена така ієрархія зацікавлених сторін відносно усіх визначених опорних точок зору:

— екологічні організації;

— постачальники енергії: фінансовий відділ;

— клієнти: власники будинку: мешканці будинку;

— розробники: керівники: розробники, аналітики, тестувальники.

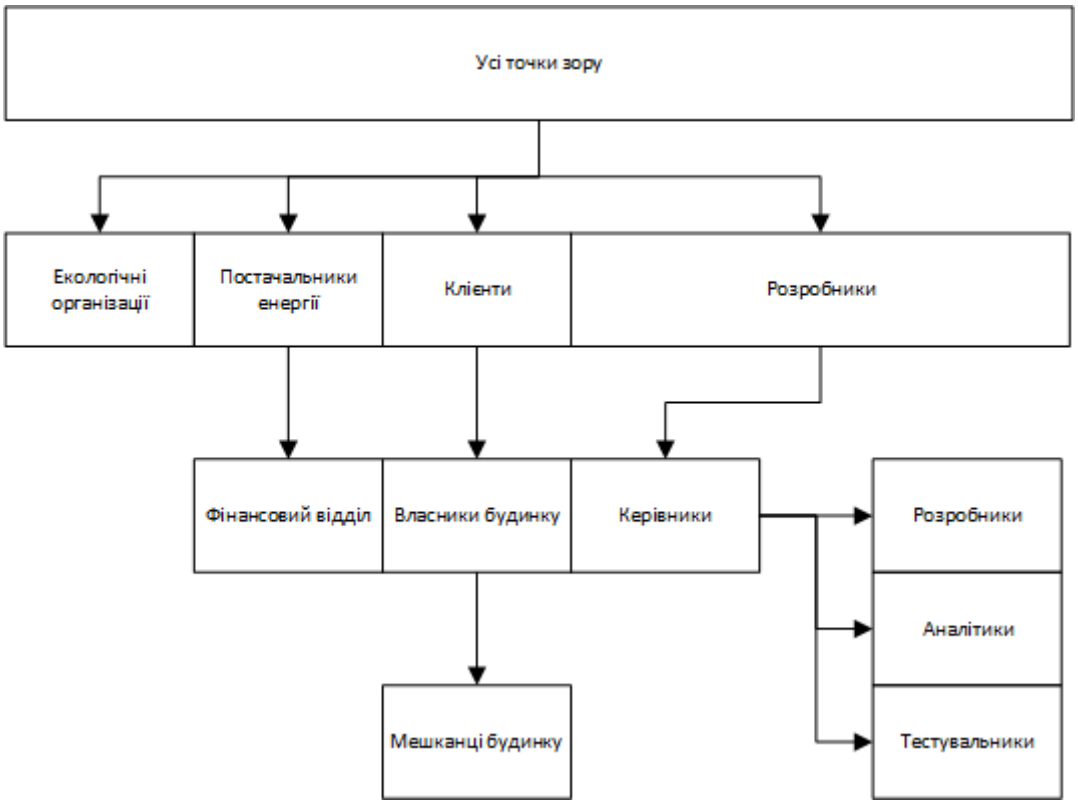


Рисунок 2.2 – Діаграма ієрархії зацікавлених сторін

Діаграма ієрархії зацікавлених сторін відносно усіх визначених опорних точок зору, показана на рисунку 2.2, була розроблена за допомогою програмного комплексу Microsoft Visio [20].

2.2 Аналіз вимог

У цьому підрозділі цієї кваліфікаційної роботи визначені основні вимоги до програмного комплексу, який розробляється для здійснення моделювання інтелектуального керування системами нагрівання води, опалення та кондиціонування у системах «Розумний будинок». Програмний комплекс повинен забезпечувати моделювання ефективної взаємодії з інженерними системами будинку, аналізувати дані про активність мешканців і температуру повітря. Також програмний комплекс повинен моделювати ухвалення рішень для оптимізації роботи опалення, кондиціонування та нагрівання води.

Визначені вимоги охоплюють функціональні та нефункціональні аспекти, технічні деталі, а також критерії тестування програмного комплексу.

Це має забезпечити цілісне розуміння необхідних характеристик програмного комплексу і сприяти його успішній розробці.

Функціональні вимоги до програмного комплексу:

— програмний комплекс повинен генерувати дані: часова мітка, назва кімнати, кількість людей у кімнаті, температура в кімнаті, температура довкілля, використання гарячої води, режим роботи системи нагрівання води (увімкнена або вимкнена), режим роботи системи опалення та кондиціонування (увімкнене опалення, увімкнене кондиціонування, все вимкнене);

— програмний комплекс повинен моделювати отримання даних з датчиків: присутність мешканців у кімнатах, внутрішня температура в кожній кімнаті, зовнішня температура довкілля, споживання гарячої води;

— програмний комплекс повинен аналізувати зібрані дані для виявлення закономірностей поведінки мешканців. Дані для аналізу: періоди присутності та активності мешканців, зміни внутрішньої та зовнішньої температури, інформація про споживання гарячої води, інформація про використання системи нагрівання води, інформація про використання систем опалення та кондиціонування;

— програмний комплекс повинен моделювати ухвалення рішень щодо: ввімкнення або вимкнення опалення та кондиціонування в окремих кімнатах, оптимізації роботи системи нагрівання води;

— програмний комплекс повинен порівнювати витрати і вартість енергії у різних режимах роботи систем нагрівання води, опалення та кондиціонування;

— у програмному комплексі необхідно надати інтуїтивно зрозумілий інтерфейс для моніторингу та керування інформаційною системою.

Нефункціональні вимоги до програмного комплексу:

— програний комплекс повинен моделювати підтримку різних конфігурацій будинків;

— необхідно забезпечити стійку роботу програмного комплексу у разі виникнення помилках;

— у програмному комплексі має бути забезпечена швидка обробка даних та ухвалення рішень у режимі реального часу;

— архітектура програмного комплексу повинна дозволяти легке оновлення та розширення функціональності.

Технічні вимоги до програмного комплексу:

— реалізація програмного комплексу повинна відбуватися за допомогою мови програмування Python із використанням відповідних бібліотек для аналізу даних та розробки інтерфейсу користувача;

— необхідно забезпечити сумісність програмного комплексу з основними операційними системами: Windows, Linux та macOS;

— для програмного комплексу необхідно використовувати надійні системи зберігання даних.

Вимоги до тестування програмного комплексу: повинна відбуватися перевірка всіх розроблених функцій програмного комплексу на коректність роботи.

2.3 Розробка структури набору даних

Структура набору даних – це організація і спосіб представлення даних у наборі. Вона визначає, як саме дані впорядковані, зберігаються та як до них можна звертатися для здійснення їхнього аналізу [22].

На рисунку 2.3 показана схема структури набору даних, котрий повинен містити дані для здійснення аналізу та ухвалення рішень.

Значення кожного об’єкту в набору даних, показаного на рисунку 2.3:

— часова мітка. Використовується для фіксації дати і часи виконання дії;

— назва кімнати. Вказує на місце виконання дії;

— кількість людей, які перебувають у кімнаті на момент фіксації даних. Використовується для визначення, чи знаходиться хтось у кімнаті;

— температура повітря всередині кімнати. Використовується для аналізу та ухвалення рішень щодо ввімкнення або вимкнення опалення чи кондиціонеру для підтримання комфортного клімату;

— температура повітря зовні будинку. Дозволяє оцінити потребу в опаленні або охолодженні кімнати. Зовнішня температура враховується для збереження енергетичної ефективності;

— кількість використаної гарячої води за певний проміжок часу. Аналіз цього параметру дозволяє визначати пікові періоди використання гарячої води, що допомагає оптимально керувати роботою бойлеру, вмикаючи його тільки у потрібний час;

— режим роботи бойлеру (увімкнений або вимкнений). Використовується для керування бойлером;

— режим опалення і кондиціонування (увімкнений кондиціонер, увімкнене опалення або все вимкнено). Використовується для керування системою опалення та кондиціонування.

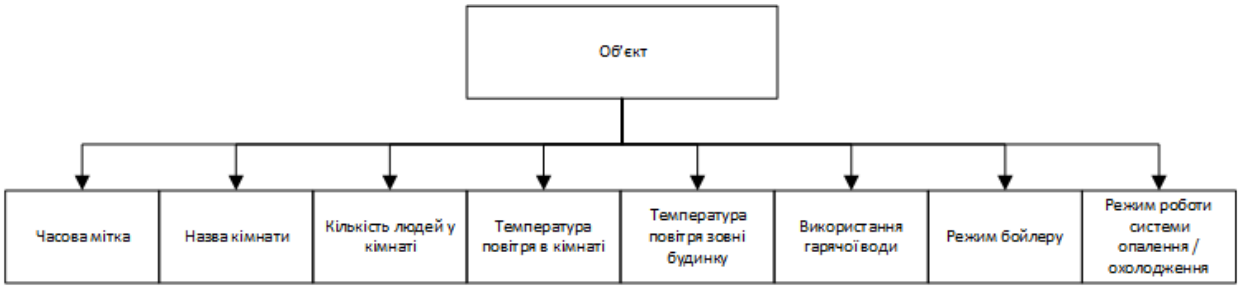


Рисунок 2.3 – Схема структури набору даних

Схема структури набору даних, показана на рисунку 2.3, була розроблена за допомогою програмного комплексу Microsoft Visio [20].

2.4 Моделювання варіантів використання

Діаграма варіантів використання – це графічний спосіб опису функціональності інформаційної системи з точки зору її користувачів. Вона належить до уніфікованої мови моделювання і широко використовується для аналізу вимог і планування розробки інформаційної систем [23].

Основними компонентами діаграми варіантів використання, зазвичай, є:

— актори – це користувачі або інші інформаційні системи, які взаємодіють із поточною інформаційною системою. Актори позначаються у вигляді піктограми людини [23];

— варіанти використання – це функції, які інформаційна система виконує для акторів. Варіанти використання зображуються у вигляді овалів [23];

— зв’язки – зв’язок акторів із варіантами використання: стрілка або проста лінія, яка показує взаємодію [23].

На рисунку 2.4 показана діаграма варіантів використання у процесі генерації набору даних.

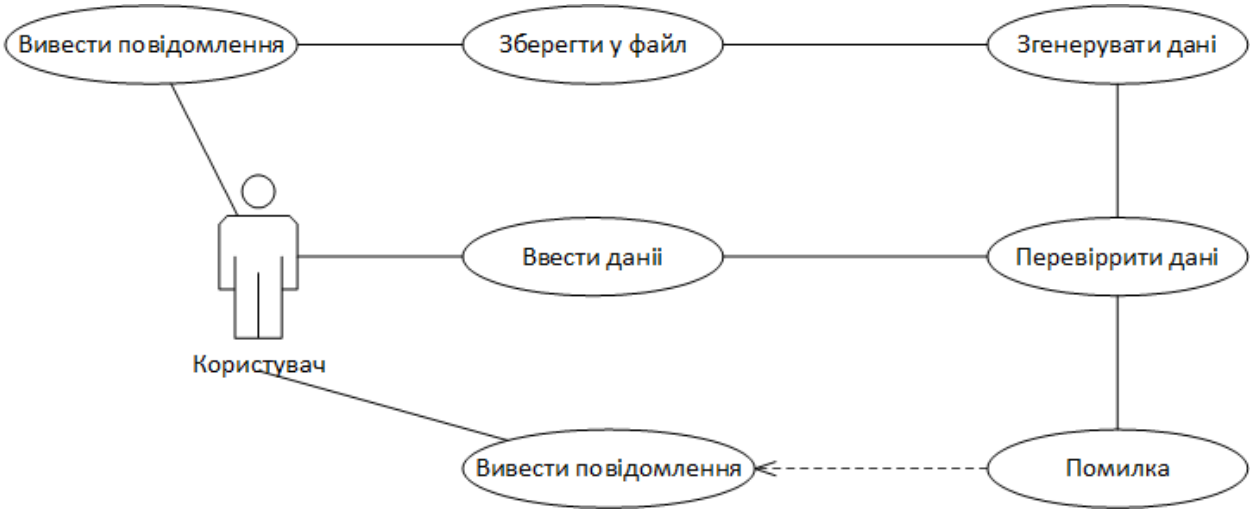


Рисунок 2.4 – Діаграма варіантів використання у процесі генерації набору даних

На рисунку 2.4 визначені такі складові діаграми варіантів використання у процесі генерації набору даних:

— актор «Користувач»;

— варіант використання «Ввести дані»;

— варіант використання «Перевірити дані»;

— варіант використання «Помилка»;

— варіант використання «Вивести повідомлення»;

— варіант використання «Згенерувати дані»;

— варіант використання «Зберегти у файл»;

— варіант використання «Вивести повідомлення».

На рисунку 2.5 показана діаграма варіантів використання у процесі аналізу набору даних.

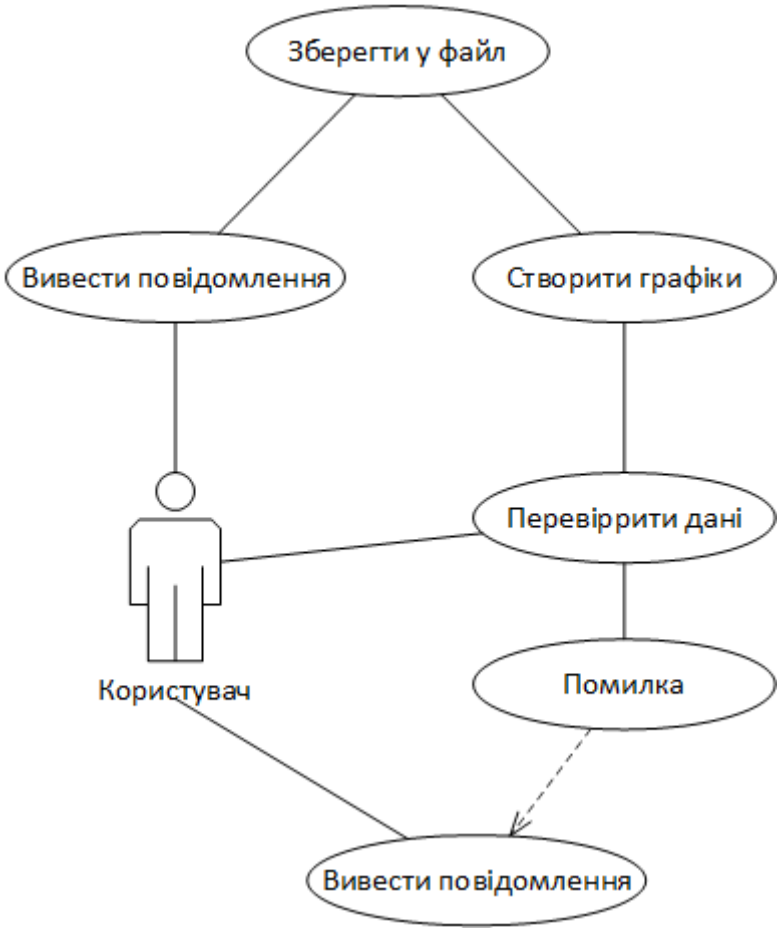


Рисунок 2.5 – Діаграма варіантів використання у процесі аналізу набору даних

На рисунку 2.5 визначені такі складові діаграми варіантів використання у процесі аналізу набору даних:

— актор «Користувач»;

— варіант використання «Перевірити дані»;

— варіант використання «Помилка»;

— варіант використання «Вивести повідомлення»;

— варіант використання «Створити графіки»;

— варіант використання «Зберегти у файл»;

— варіант використання «Вивести повідомлення;

На рисунку 2.6 показана діаграма варіантів використання у процесі створення моделей нейронних мереж.

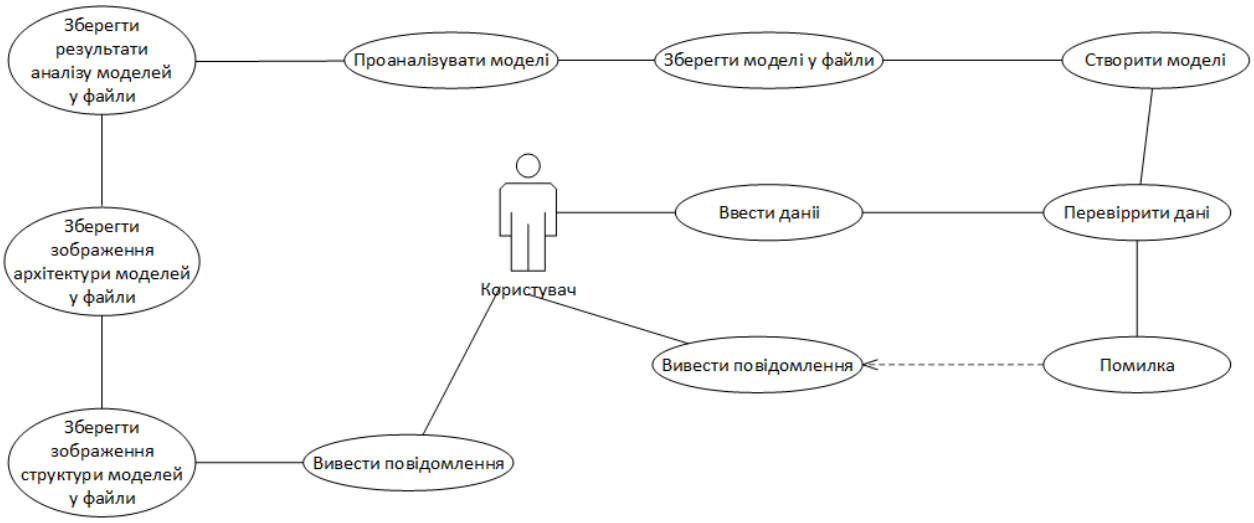


Рисунок 2.6 – Діаграма варіантів використання у процесі створення моделей штучних нейронних мереж

На рисунку 2.6 визначені такі складові діаграми варіантів використання у процесі створення моделей нейронних мереж:

— актор «Користувач»;

— варіант використання «Ввести дані»;

— варіант використання «Перевірити дані»;

— варіант використання «Помилка»;

— варіант використання «Вивести повідомлення»;

— варіант використання «Створити моделі»;

— варіант використання «Зберегти моделі у файли»;

— варіант використання «Проаналізувати моделі»;

— варіант використання «Зберегти результати аналізу моделей у файли»;

— варіант використання «Зберегти зображення архітектури моделей у файли»;

— варіант використання «Зберегти зображення структури моделей у файли»;

— варіант використання «Вивести повідомлення».

На рисунку 2.7 показана діаграма варіантів використання у процесі прогнозування даних.

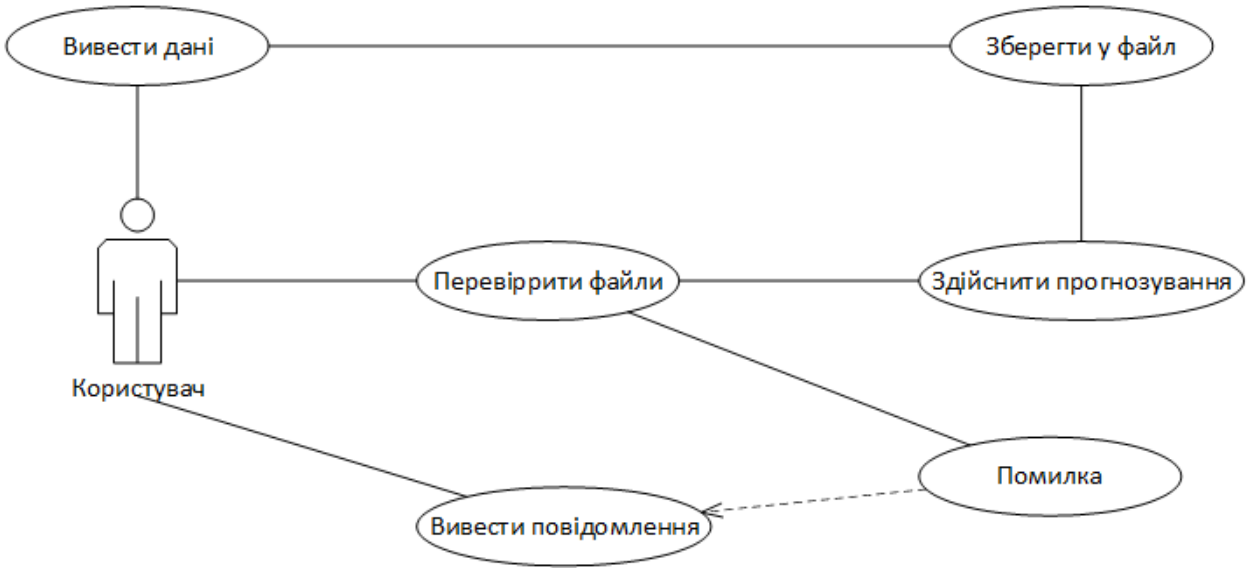


Рисунок 2.7 – Діаграма варіантів використання у процесі прогнозування даних

На рисунку 2.6 визначені такі складові діаграми варіантів використання у процесі прогнозування даних:

— актор «Користувач»;

— варіант використання «Перевірити файли»;

— варіант використання «Помилка»;

— варіант використання «Вивести повідомлення»;

— варіант використання «Прогнозувати дані»;

— варіант використання «Зберегти файл»;

— варіант використання «Вивести дані».

На рисунку 2.8 показана діаграма варіантів використання у процесі оцінки економічної ефективності програмного комплексу.

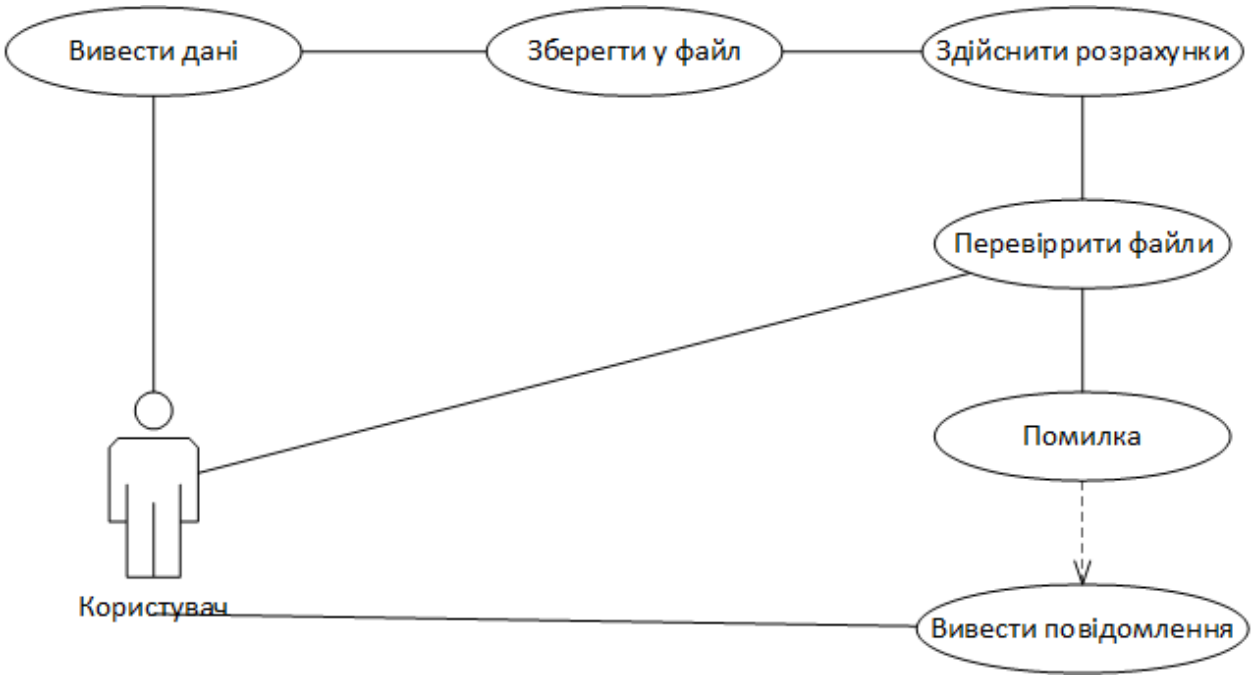


Рисунок 2.8 – Діаграма варіантів використання у процесі оцінки економічної ефективності програмного комплексу

На діаграмі варіантів використання, показані на рисунку 2.8, показані такі складові:

— актор «Користувач»;

— варіант використання «Перевірити файли»;

— варіант використання «Помилка»;

— варіант використання «Вивести повідомлення»;

— варіант використання «Здійснити розрахунки»;

— варіант використання «Зберегти файл»;

— варіант використання «Вивести дані».

Діаграми варіантів використання, показані на рисунках 2.4 – 2.8, були розроблені за допомогою програмного комплексу Microsoft Visio [20].

2.5 Моделювання процесів

Діаграма діяльності – це один із видів діаграм в уніфікованій мові моделювання UML, який використовується для моделювання процесів, дій або робочих потоків в інформаційних системах. Вона графічно відображає послідовність дій і потоків управління між ними, що дозволяє зрозуміти, як виконується певний процес або сценарій [24].

Основними елементами діаграми діяльності, зазвичай, є:

— дія – являє собою окрему операцію або крок процесу [24];

— контрольні потоки – стрілки, які вказують послідовність виконання дій [24];

— початкове положення – позначається чорним кругом, показує початок виконання процесу [24];

— кінцеве положення – позначається чорним кругом із кільцем, вказує на завершення процесу [24];

— рішення – ромб, який використовується для розгалуження потоку в залежності від умов [24].

На рисунку 2.9 показана діаграма діяльності у процесі генерації набору даних. Показаний такий процес:

— користувач вказує параметри набору даних і натискає кнопку;

— відбувається перевірка, чи були дані вказані. Якщо ні, виводиться повідомлення про помилку;

— перевіряється, чи всі значення є числами. Якщо ні, виводиться повідомлення про помилку;

— генерується набір даних;

— набір даних зберігається у файл;

— виводиться повідомлення про завершення.

Рисунок 2.9 – Діаграма діяльності у процесі генерації набору даних

На рисунку 2.10 показана діаграма діяльності у процесі аналізу набору даних.

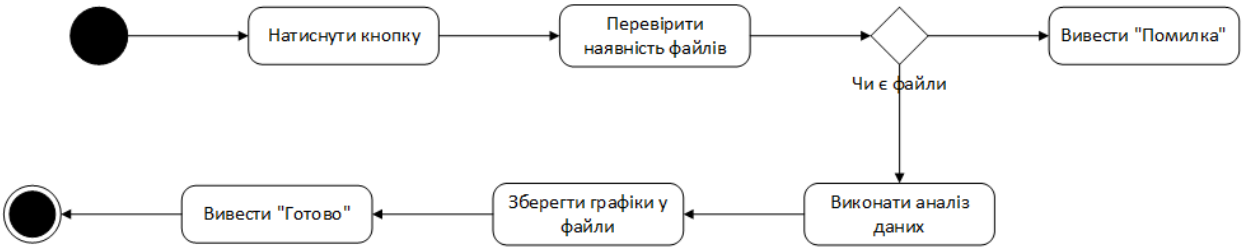


Рисунок 2.10 – Діаграма діяльності у процесі аналізу набору даних

На рисунку 2.10 на діаграмі діяльності показаний такий процес:

— користувач натискає кнопку;

— відбувається перевірка: чи наявний файл набору даних. Якщо ні, виводиться повідомлення про помилку;

— відбувається аналіз даних;

— графіки зберігаються у файли;

— виводиться повідомлення про завершення процесу.

На рисунку 2.11 показана діаграма діяльності у процесі створення моделей штучних нейронних мереж. На діаграмі діяльності показаний такий процес:

— відбувається введення даних і натиснення кнопки;

— відбувається перевірка введених даних. Якщо були введені не всі дані, виводиться повідомлення про помилку. Якщо введені дані не всі є числами, виводиться повідомлення про помилку;

— відбувається створення моделей нейронних мереж;

— відбувається збереження моделей нейронних мереж у файли;

— відбувається аналіз точності моделей і збереження цих даних у файли;

— відбувається збереження зображень архітектури і структури моделей у файли;

— виводиться повідомлення про виконання.

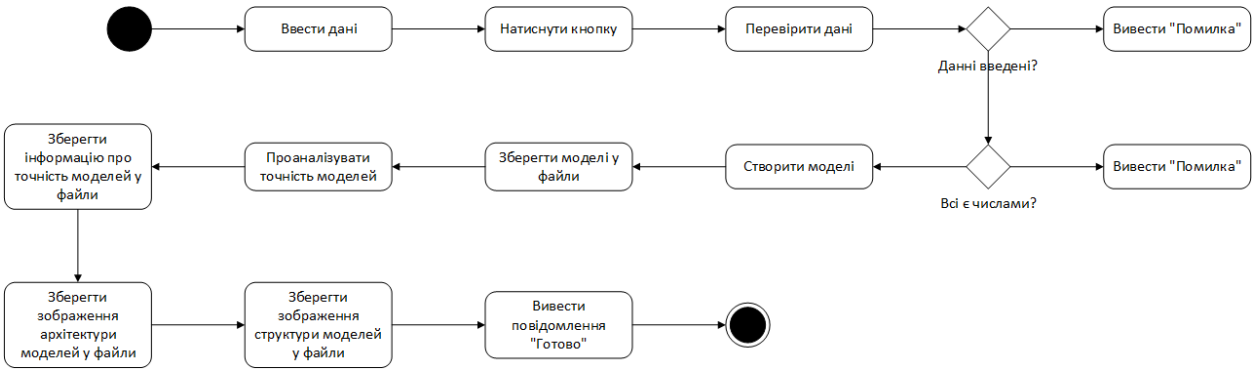


Рисунок 2.11 – Діаграма діяльності у процесі створення моделей штучних нейронних мереж

На рисунку 2.12 показана діаграма діяльності у процесі прогнозування даних. На діаграмі діяльності показаний такий процес:

— відбувається натиснення на кнопку;

— відбувається перевірка наявності файлів. Якщо файли набору даних відсутні, виводиться повідомлення про помилку. Якщо файли моделей відсутні, виводиться повідомлення про помилку;

— відбувається прогнозування даних;

— відбувається виведення на екран прогнозованих даних.

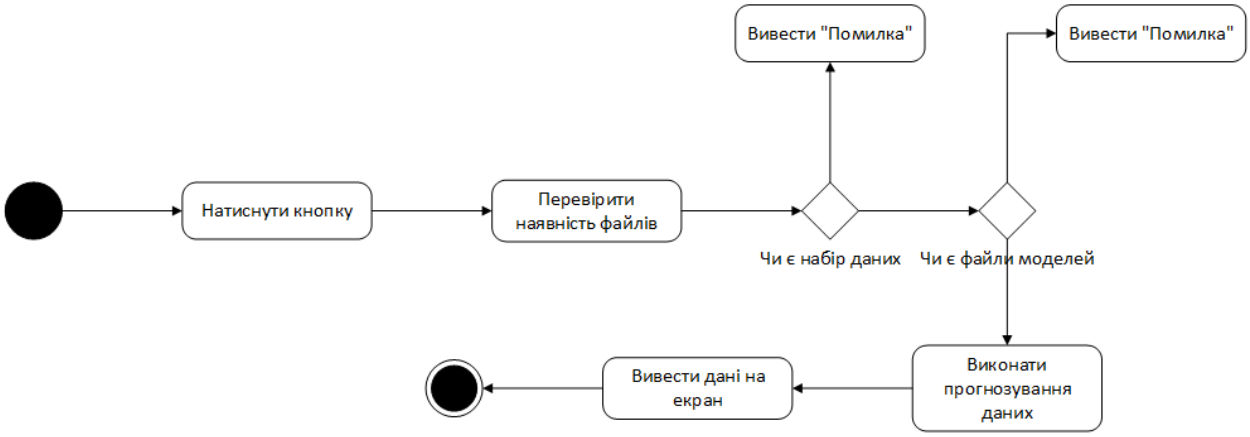


Рисунок 2.12 – Діаграма діяльності у процесі прогнозування даних

На рисунку 2.13 показана діаграма діяльності у процесі оцінки економічної ефективності програмного комплексу.

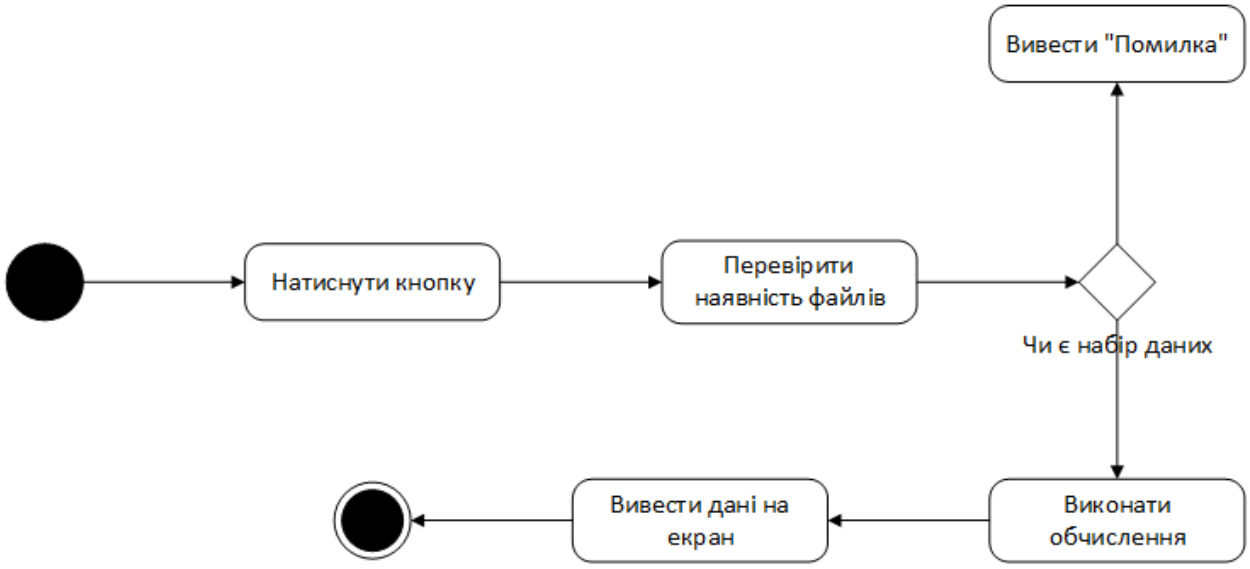


Рисунок 2.13 – Діаграма діяльності у процесі оцінки економічної ефективності програмного комплексу

На рисунку 2.13 визначений такий процес, показаний на діаграмі діяльності у процесі оцінки економічної ефективності програмного комплексу:

— відбувається натиснення на кнопку;

— відбувається перевірка наявності файлів. Якщо файли набору даних відсутні, виводиться повідомлення про помилку;

— відбувається виконання обчислень;

— відбувається виведення даних на екран.

Діаграми діяльності, показані на рисунках 2.9 – 2.13, були розроблені за допомогою програмного комплексу Microsoft Visio [20].

2.6 Моделювання потоків даних

Діаграма потоків даних – це графічне представлення «потоку» (руху) даних в інформаційній системі. Вона є одним із засобів візуалізації процесів, що відбуваються в інформаційних системах. Діаграма потоків даних дозволяє зображати процеси та зовнішні сутності, які взаємодіють з інформаційною системою, а також потоки даних, що проходять між ними [25].

Основними компонентами діаграми потоків даних, зазвичай, є:

— процес – дія або трансформація, яка виконується над даними. Процес зображується прямокутником з назвою, що описує його функцію [25];

— зовнішня сутність – об’єкт поза інформаційною системою, який генерує чи споживає дані. Сутність зображується прямокутником або іншою фігурою [25];

— потік даних – пересування даних або інформації від однієї частини інформаційної системи до іншої. Потік зображується стрілкою з назвою, що описує дані [25];

— сховище даних – місце для зберігання даних, наприклад, файли або бази даних. Сховище зображується відкритим прямокутником [25].

На рисунку 2.14 показана діаграма потоків даних у процесі генерації набору даних. Основні компоненти діаграми потоків даних:

— процес «Вказати параметри»;

— потік даних «Параметри»;

— процес «Валідувати дані»;

— потік даних «Параметри»;

— процес «Згенерувати дані»;

— потік даних «Згенеровані дані»;

— процес «Зберегти у файл»;

— потік даних «Згенеровані дані»;

— сховище «Файл».

Рисунок 2.14 – Діаграма потоків даних у процесі генерації набору даних

На рисунку 2.15 показана діаграма потоків даних у процесі аналізу даних.

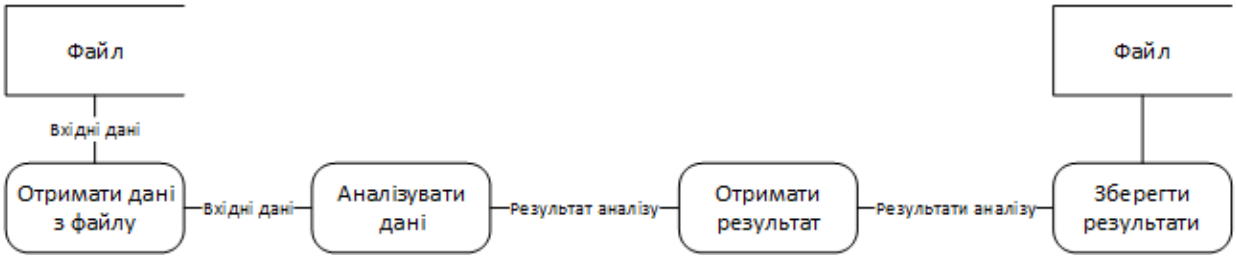


Рисунок 2.15 – Діаграма потоків даних у процесі аналізу набору даних

Основні компоненти діаграми потоків даних у процесі аналізу набору даних, показаної на рисунку 2.15:

— сховище «Файл»;

— процес «Отримати вхідні дані з файлу»;

— потік даних «Вхідні дані»;

— потік даних «Вхідні дані»;

— процес «Аналізувати дані»;

— потік даних «Результати аналізу»;

— процес «Отримати результати»;

— потік даних «Результати аналізу»;

— процес «Зберегти результати»;

— Сховище «Файл».

На рисунку 6.16 показана діаграма потоків даних у процесі створення моделей штучних нейронних мереж.

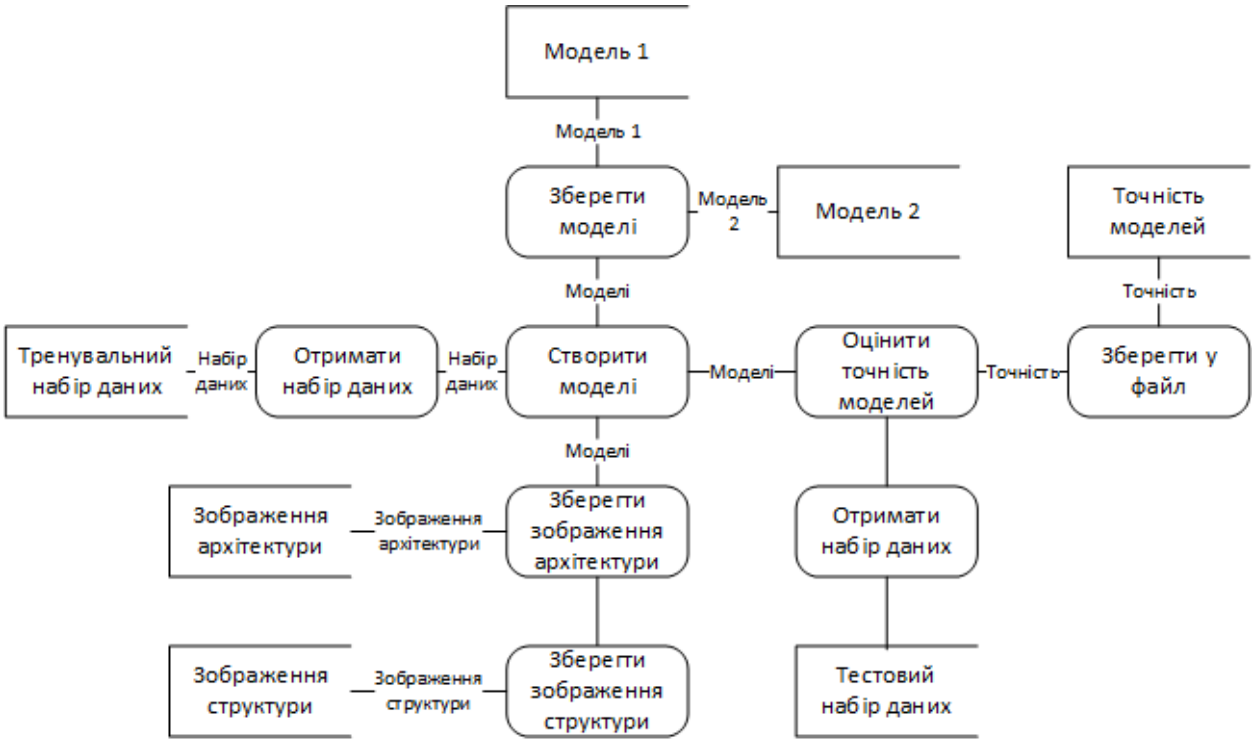


Рисунок 2.16 – Діаграма потоків даних у процесі створення моделей штучних нейронних мереж

Основні компоненти діаграми потоків даних у процесі створення моделей нейронних мереж, показаної на рисунку 2.16:

— сховище «Тренувальний набір даних»;

— потік даних «Набір даних»;

— процес «Отримати набір даних»;

— процес «Створити моделі»;

— потік даних «Моделі»;

— процес «Зберегти моделі»;

— сховище «Модель 1»;

— сховище «Модель 2»;

— сховище «Тестовий набір даних»;

— процес «Оцінити точність моделей»;

— потік даних «Точність моделей»;

— процес «Зберегти у файл точність моделей»;

— сховище «Точність моделей».

На рисунку 2.17 показана діаграма потоків даних у процесі прогнозування даних.

Рисунок 2.17 – Діаграма потоків даних у процесі прогнозування даних

Основні компоненти діаграми потоків даних у процесі прогнозування даних, показаної на рисунку 2.17:

— сховище «Тестовий набір даних»;

— потік даних «Набір даних»;

— процес «Отримати набір даних»;

— процес «Прогнозування даних»;

— потік даних «Прогнозовані дані»;

— процес «Вивести прогнозовані дані»;

— процес «Зберегти прогнозовані дані»;

— сховище «Прогнозовані дані».

На рисунку 2.18 показана діаграма потоків даних у процесі оцінки економічної ефективності програмного комплексу. Основні компоненти діаграми потоків даних:

— сховище «Тестовий набір даних»;

— сховище «Звичайний набір даних»;

— потік даних «Набір даних»;

— процес «Отримати набір даних»;

— процес «Обчислення ефективності»;

— потік даних «Обчислені дані»;

— процес «Вивести обчислені дані»;

— процес «Зберегти обчислені дані»;

— сховище «Обчислені дані».

Рисунок 2.18 – Діаграма потоків даних у процесі оцінки економічної ефективності програмного комплексу

Діаграми потоків даних, показані на рисунках 2.14 – 2.18, були розроблені за допомогою програмного комплексу Microsoft Visio [20].

2.7 Моделювання функціональності

Функціональна схема – це графічне зображення, яке показує функціональну структуру інформаційної системи, а також взаємозв’язки між окремими функціональними блоками [26].

Основними елементами функціональної схеми, зазвичай, є:

— функціональний блок – графічний елемент, котрий відображає окрему функцію або операцію інформаційної системи. Він зображуються у вигляді прямокутнику [26];

— лінія зв’язку – лінія, котра з’єднує функціональні блоки і показує передачу даних або сигналів між ними [26].

На рисунку 2.19 показана функціональна схема у процесі генерації набору даних.

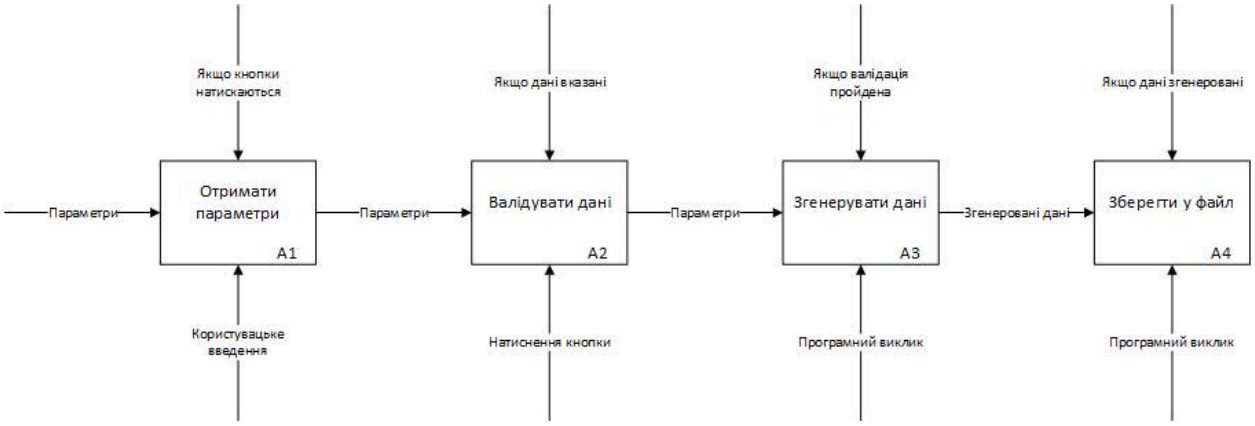


Рисунок 2.19 – Функціональна схема у процесі генерації набору даних

Основні функції у процесі генерації набору даних:

— отримати параметри;

— валідувати дані;

— згенерувати дані;

— зберегти у файл.

На рисунку 2.20 показана функціональна схема у процесі аналізу набору даних.

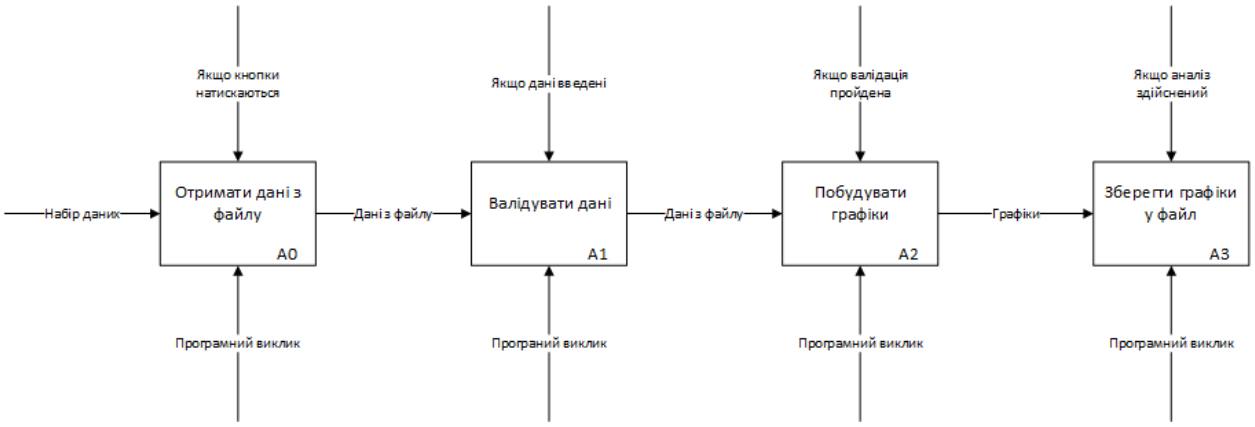


Рисунок 2.20 – Функціональна схема у процесі аналізу набору даних

Основні функції у процесі аналізу набору даних:

— отримати дані з файлу;

— валідувати дані;

— побудувати графіки;

— зберегти графіки у файл.

На рисунку 2.21 показана функціональна схема у процесі створення моделей штучних нейронних мереж.

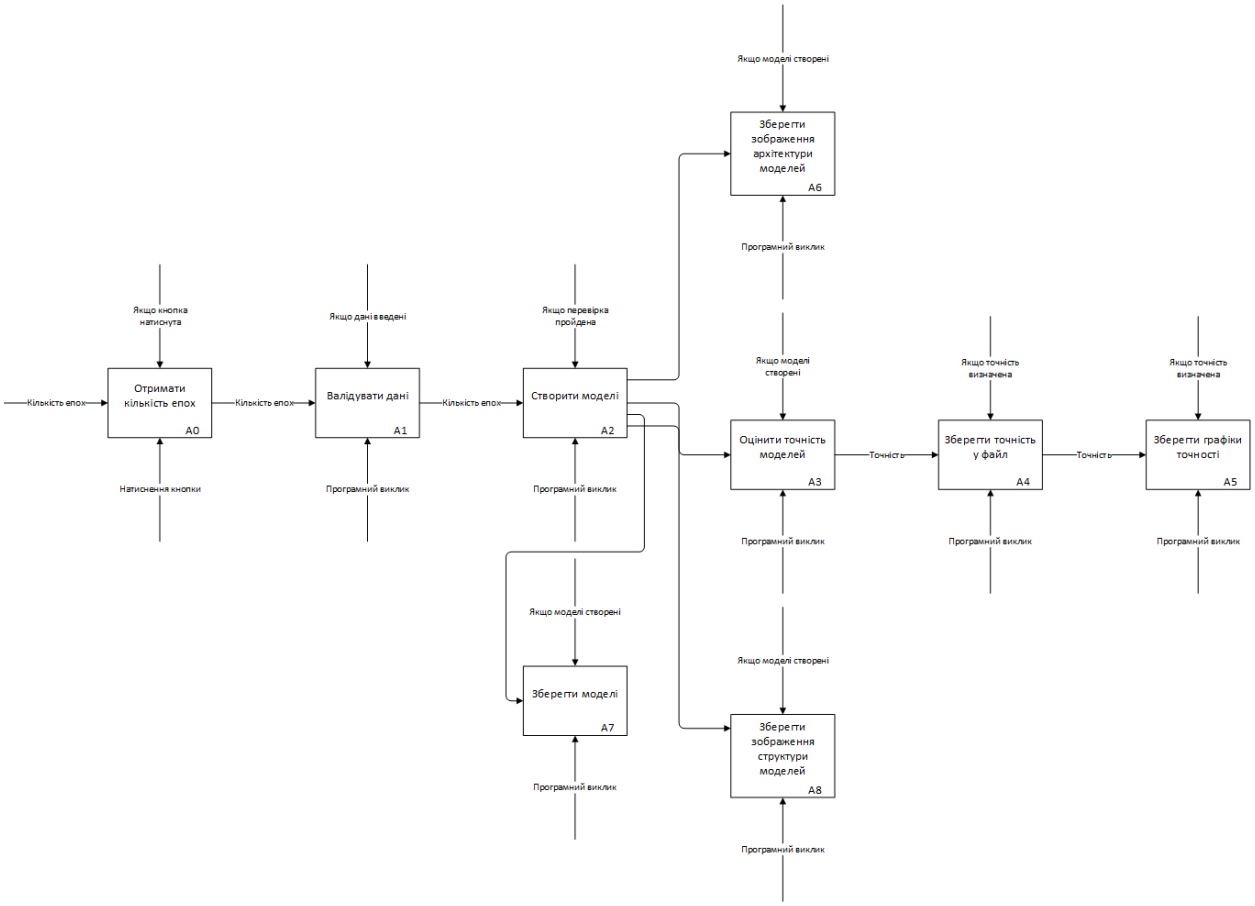


Рисунок 2.21 – Функціональна схема у процесі створення моделей штучних нейронних мереж

Основні функції у процесі створення моделей штучних нейронних мереж:

— отримати кількість епох;

— валідувати дані;

— створити моделі;

— оцінити точність моделей;

— зберегти точність у файл;

— зберегти графіки точності;

— зберегти зображення архітектури моделей;

— зберегти моделі;

— зберегти зображення структури моделей.

На рисунку 2.22 показана функціональна схема у процесі прогнозування даних.

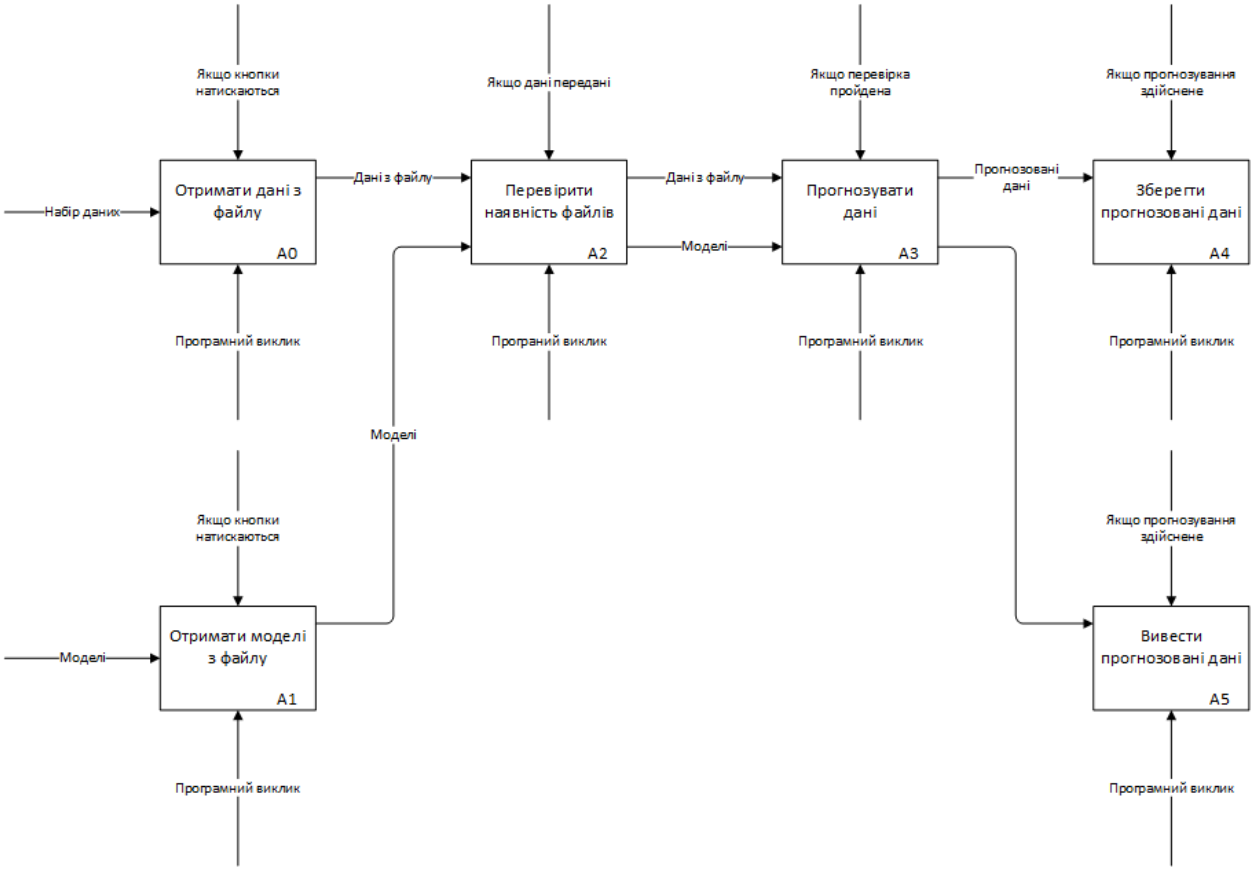


Рисунок 2.22 – Функціональна схема у процесі прогнозування даних

Основні функції у процесі прогнозування даних:

— отримати дані з файлу;

— отримати моделі з файлу;

— перевірити наявність файлів;

— прогнозувати дані;

— зберегти прогнозовані дані;

— вивести прогнозовані дані.

На рисунку 2.23 показана функціональна схема у процесі оцінки економічної ефективності програмного комплексу.

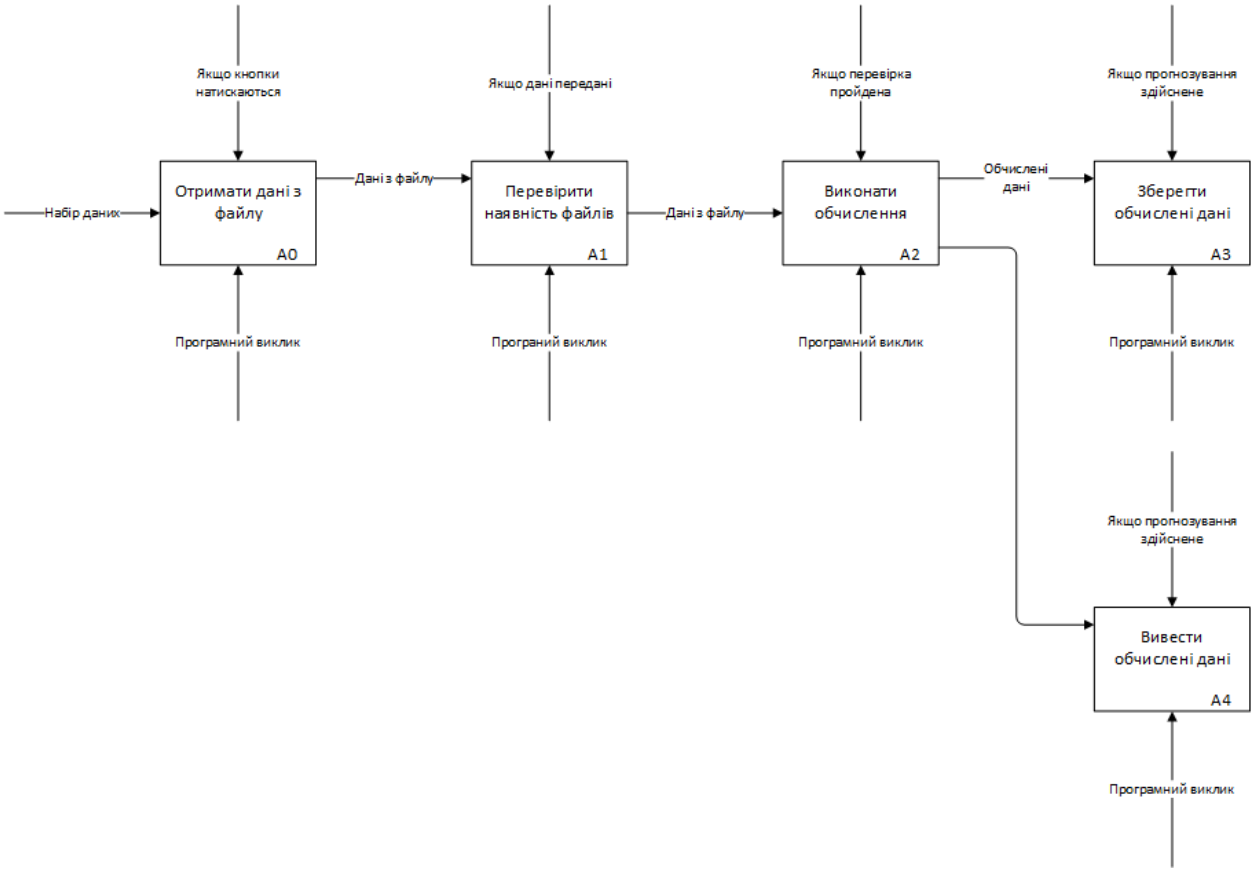


Рисунок 2.23 – Функціональна схема у процесі оцінки економічної ефективності програмного комплексу

Основні функції у процесі прогнозування даних:

— отримати дані з файлу;

— перевірити наявність файлів;

— обчислити дані;

— зберегти обчислені дані;

— вивести обчислені дані.

Функціональні схеми, показані на рисунках 2.19 – 2.23, були розроблені за допомогою програмного комплексу Microsoft Visio [20].

2.8 Моделювання структури програмного комплексу

Структурна схема інформаційної системи – це графічне представлення логічної структури інформаційної системи або її основних компонентів, що показує послідовність виконання команд, взаємозв’язки між частинами програми та потік даних. Така схема дозволяє зрозуміти, як працює інформаційна система в загальних рисах, а також полегшує її розробку, налагодження та обслуговування [27].

Основними елементами структурної схеми програмного комплексу, зазвичай, є:

— процес – прямокутник, що відображає окрему операцію або крок [27];

— рішення – ромб, що представляє точку вибору з кількох можливих варіантів [27];

— потік даних – стрілка, що вказує на напрямок руху інформації або виконання команди [27];

— підпрограма – прямокутник з подвійною рамкою, який позначає блок коду, що виконує певні задачі [27].

На рисунку 2.24 показана структурна схема програмного комплексу. Визначені такі основні компоненти програмного комплексу:

— головне меню;

— обробник даних;

— створення набору даних;

— генератор даних;

— аналіз набору даних;

— текстовий аналізатор набору даних;

— візуальний аналізатор набору даних;

— візуальний аналізатор моделей;

— створення моделей нейронних мереж;

— конструктор моделей;

— прогнозування даних;

— виклик моделей;

— оцінка економічної ефективності;

— економічний калькулятор;

— файлові операції.

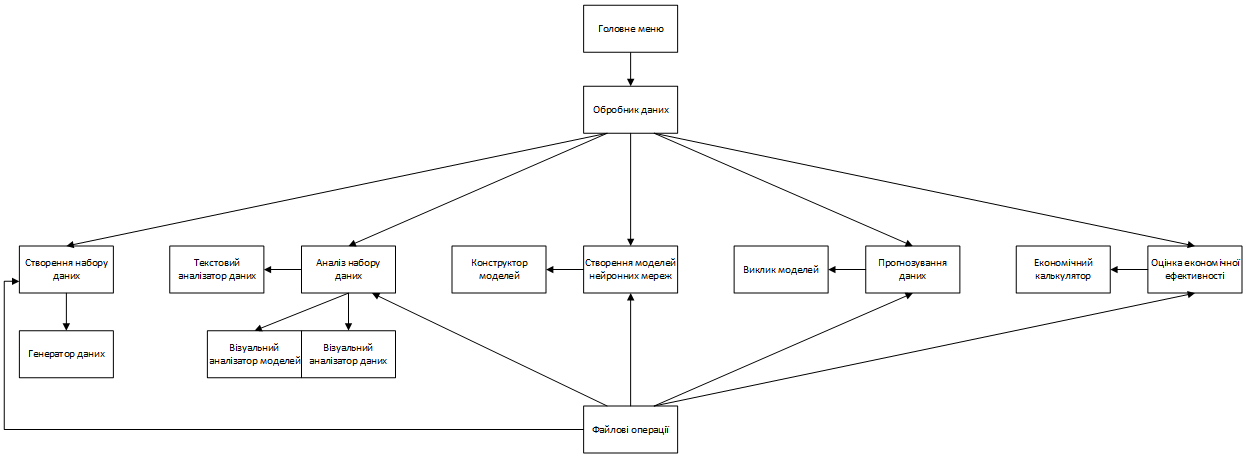


Рисунок 2.24 – Структурна схема програмного комплексу

Структурна схеми програмного комплексу, показані на рисунку 2.24, була розроблена за допомогою програмного комплексу Microsoft Visio [20].

2.9 Моделювання архітектури програмного комплексу

Діаграма класів – це один із ключових засобів у моделюванні програмного комплексу в уніфікований мові моделювання [28]. Вона використовується для візуалізації архітектури інформаційної системи, описі класів і відношень між ними [28].

Основними елементами діаграми класів, зазвичай, є:

— клас – елемент, представлений у вигляді прямокутнику, який поділяється на три секції: ім’я класу, атрибути класу, методи класу [28];

— відношення між класами – лінія або стрілка, яка з’єднують два класи між собою [28].

Зазвичай, на діаграмі класі відображають такі типи відношень між класами:

— асоціація – зв’язок між двома класами, відображає відношення типу «має» або «користується» [28];

— агрегація – слабкий зв’язок типу «частина-ціле», коли один клас є частиною іншого, але може існувати незалежно [28];

— композиція – сильний зв’язок типу «частина-ціле», коли частини не можуть існувати окремо від цілого [28];

— наслідування – показує, що один клас є підкласом іншого [28];

— реалізація – зв’язок між класом та інтерфейсом, показує, що клас реалізує інтерфейс [28].

На рисунку 2.25 показана приблизна діаграма класів розроблюваного програмного комплексу. На діаграмі класів, показаній на рисунку 2.25, визначені такі основні класи:

— TimestampGenerator – клас для генерації часового ряду в наборі даних;

— RoomGenerator – клас для генерації кімнат у наборі даних;

— OutsideTemperatureGenerator – клас для генерації температури довкілля у наборі даних;

— BoilerStatus – клас для визначення режиму роботи системи нагрівання води у наборі даних;

— HeatingCoolingStatus – клас для визначення режимів роботи системи опалення і системи кондиціонування в наборі даних;

— ResidentPresence – клас для симуляції присутності людей в кімнатах для набору даних;

— RoomEnvironment – клас для симуляції різного стану кімнат для набору даних;

— DatasetGenerator – клас для здійснення генерації набору даних з урахуванням початкових параметрів;

— NumericSeriesGenerator – клас для генерації числового ряду з урахуванням початкових параметрів;

— TextDataAnalyzer – клас для здійснення текстового аналізу набору даних;

— VizualDataAnalyzer – клас для здійснення візуального аналізу набору даних;

— VizualModelAnalyzer – клас для здійснення візуального аналізу моделей нейронних мереж;

— DataProcessor – клас для здійснення обробки даних;

— DatasetProcessor – клас для здійснення обробки набору даних;

— DatasetReader – клас для здійснення зчитування набору даних із файлу;

— DatasetSaver – клас для здійснення запису набору даних у файл;

— ModelLoader – клас для здійснення завантаження моделі нейронної мережі з файлу;

— ModelSaver – клас для здійснення збереження моделі нейронної мережі у файл;

— PlotSaver – клас для здійснення запису графіку у файл;

— StringSaver – клас для здійснення збереження текстового рядку в файл;

— EconomicCalculator – клас для здійснення обчислень економічних показників;

— EnergyConsumptionCalculator – клас для здійснення обчислення витрат енергії пристроями;

— MathUtils – клас для здійснення додаткових математичних операцій;

— NeuralNetworkModel – абстрактний клас для опису стандартної моделі штучної нейронної мережі;

— BoilerStatusModel – клас-модель нейронної мережі для здійснення прогнозування режиму роботи системи нагрівання води;

— HeatingCoolingStatusModel – клас-модель нейронної мережі для здійснення прогнозування режимів роботи системи опалення та системи кондиціонування;

— Window – абстрактний клас для опису стандартного вікна;

— MainWindow – клас, який являє собою головне вікно графічного інтерфейсу користувача у програмному комплексі;

— DatasetCreatorWindow – клас, який являє собою вікно для здійснення створення набору даних у графічному інтерфейсі користувача програмного комплексу;

— ModelsCreationWindow – клас, який являє собою вікно для здійснення створення моделей нейронних мереж у графічному інтерфейсі користувача програмного комплексу;

— DisplayModelResultsWindow – клас, який являє собою вікно для здійснення перегляду прогнозованих даних у графічному інтерфейсі користувача програмного комплексу;

— EconomicParametersSetterWindow – клас, який являє собою вікно для здійснення введення економічних параметрів у графічному інтерфейсі користувача програмного комплексу;

— EconomicParametersDisplayWindow – клас, який являє собою вікно для здійснення виведення економічних параметрів у графічному інтерфейсі користувача програмного комплексу;

— Messagebox – клас, який являє собою вікно для виведення повідомлень у графічному інтерфейсі користувача програмного комплексу.

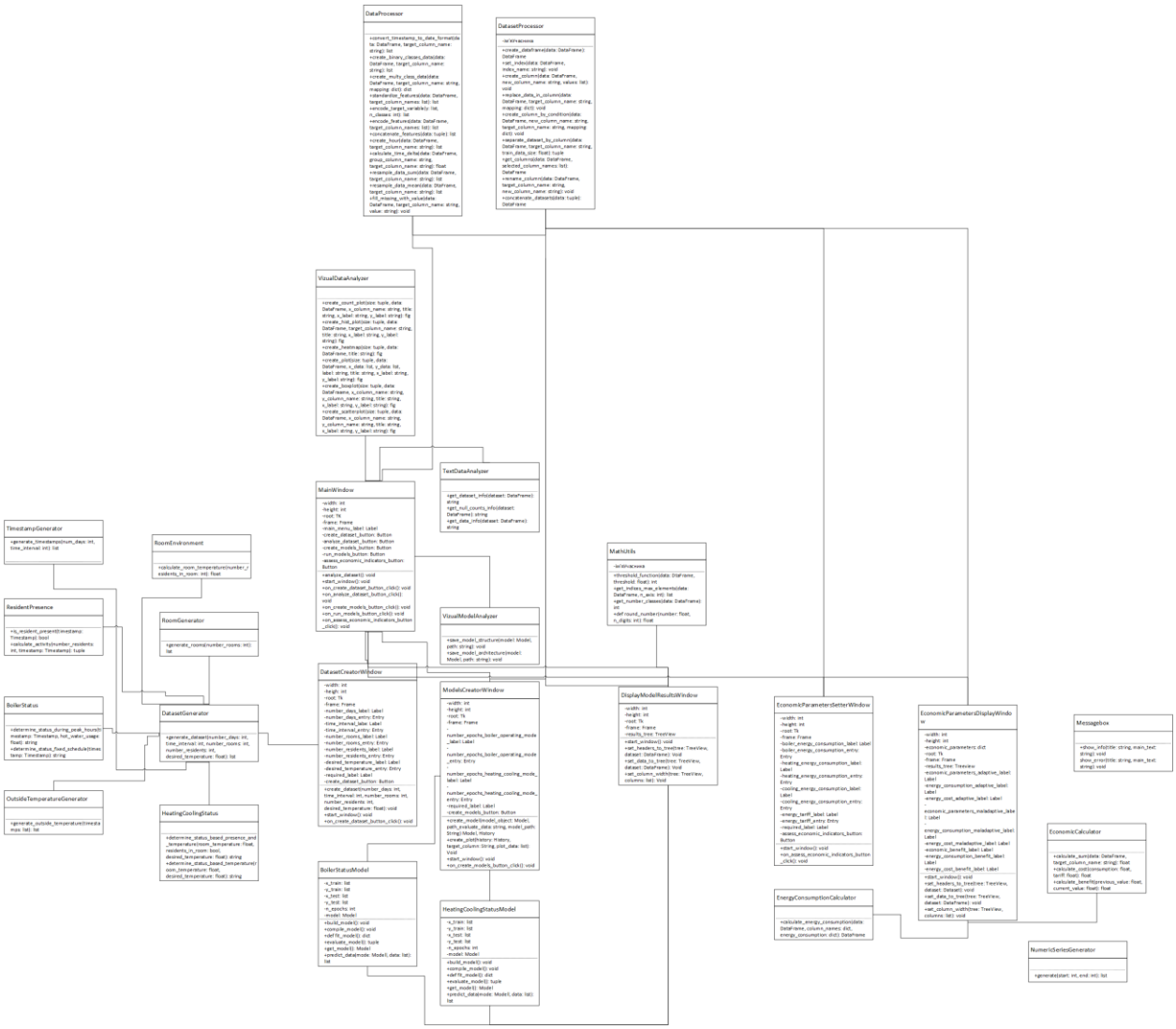


Рисунок 2.25 – Діаграма класів

Діаграма класів, показані на рисунку 2.25, була розроблена за допомогою програмного комплексу Microsoft Visio [20].

2.10 Моделювання штучних нейронних мереж

Нейронна мережа – це математична модель, натхненна будовою і функціональністю біологічного мозку, яка використовується для обробки даних і вирішення різноманітних задач [29]. Нейронна мережа складається з множини взаємопов’язаних вузлів, які називають нейронами, організованих у шари [29]. Основна ідея нейронних мереж полягає у здатності навчатися і знаходити закономірності в даних [29].

нейронні мережі мають широкий спектр застосувань у різних галузях – таких, як: комп'ютерний зір, обробка природної мови, розпізнавання мови, рекомендаційні системи, біоінформатика та фінансова аналітика [29]. Їхня популярність зумовлена високою здатністю адаптуватися до різних типів даних і складних задач [29].

Основними компонентами нейронної мережі, зазвичай, є:

— шари нейронів. Вхідний шар отримує дані. Приховані шари виконують обчислення, трансформуючи дані, щоб виявити приховані закономірності. Вихідний шар видає результат [29];

— нейрони. Кожен нейрон отримує вхідні сигнали, зважує їх, додає зсув і передає результат через активаційну функцію, яка вирішує, чи активувати нейрон [29];

— активаційні функції. Визначають нелінійність моделі. Популярні функції: ReLU, Sigmoid, Softmax [29];

— ваги та навчання. Вага – це параметри, які визначають силу зв'язку між нейронами. Нейронна мережа навчається шляхом оновлення ваг під час процесу оптимізації (наприклад, за допомогою алгоритмів зворотного поширення помилки та градієнтного спуску) [29].

На рисунку 2.26 показана архітектура нейронної мережі для здійснення прогнозування режиму роботи системи нагрівання води. Нейронна мережа має такі шари:

— вхідний шар. Кількість входів: 27;

— прихований шир. Кількість нейронів: 64. Функція активації: ReLU. Метод регуляризації: L2. Коефіцієнт регуляризації: 0,01;

— шар виключення нейронів. Імовірність виключення нейронів: 0,4;

— прихований шир. Кількість нейронів: 32. Функція активації: ReLU. Метод регуляризації: L2. Коефіцієнт регуляризації: 0,01;

— шар виключення нейронів. Імовірність виключення нейронів: 0,4;

— вихідний шар. Кількість нейронів: 1. Функція активації: Softmax.

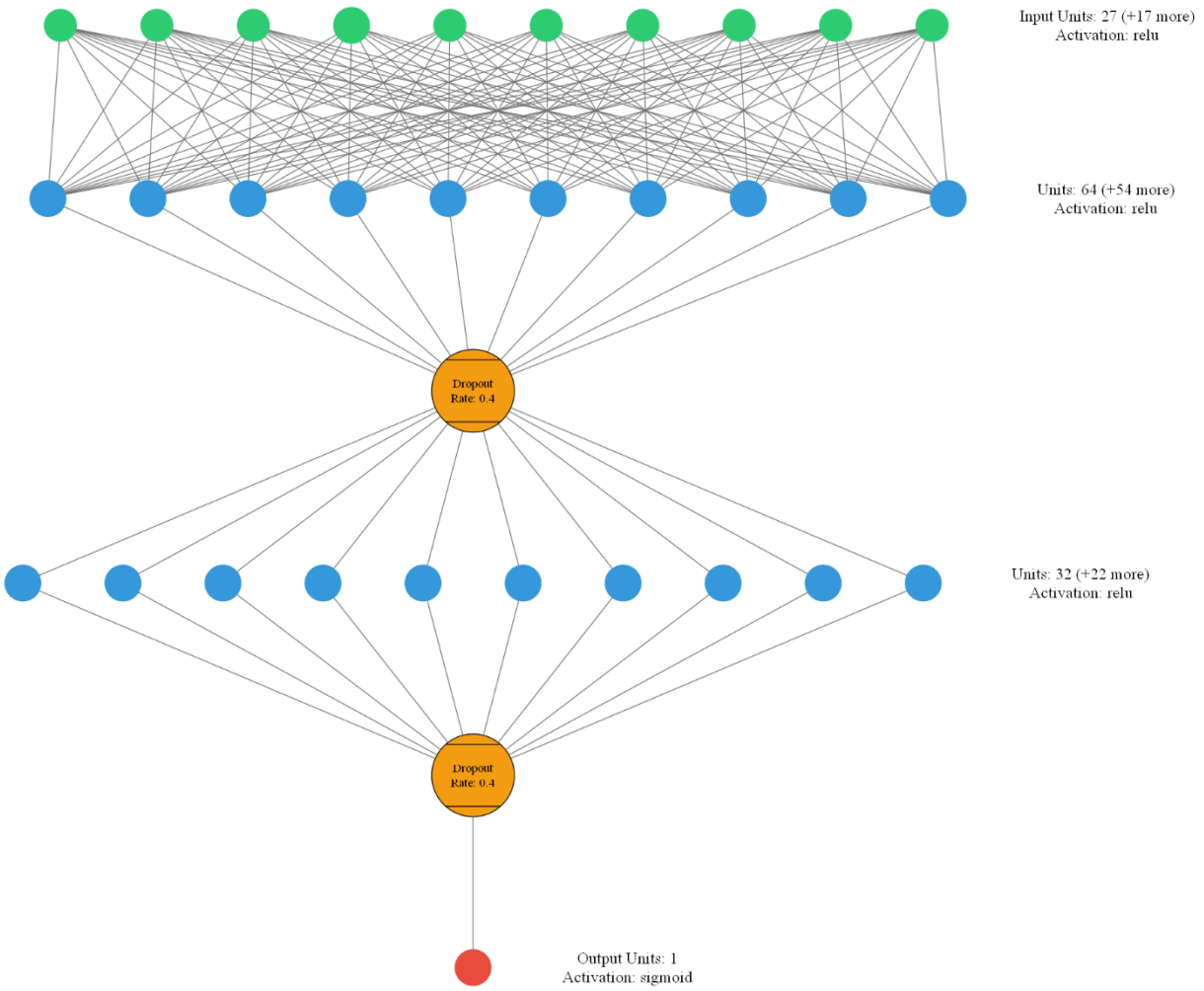


Рисунок 2.26 – Архітектура нейронної мережі для здійснення прогнозування режиму роботи системи нагрівання води

На рисунку 2.27 показана архітектура нейронної мережі для здійснення прогнозування режимів роботи системи опалення та системи кондиціонування. Нейронна мережа має такі шари:

— вхідний шар. Кількість входів: 10;

— прихований шир. Кількість нейронів: 64. Функція активації: ReLU. Метод регуляризації: L2. Коефіцієнт регуляризації: 0,01;

— шар виключення нейронів. Імовірність виключення нейронів: 0,4;

— прихований шир. Кількість нейронів: 32. Функція активації: ReLU. Метод регуляризації: L2. Коефіцієнт регуляризації: 0,01;

— шар виключення нейронів. Імовірність виключення нейронів: 0,4;

— вихідний шар. Кількість нейронів: 3. Функція активації: Softmax.

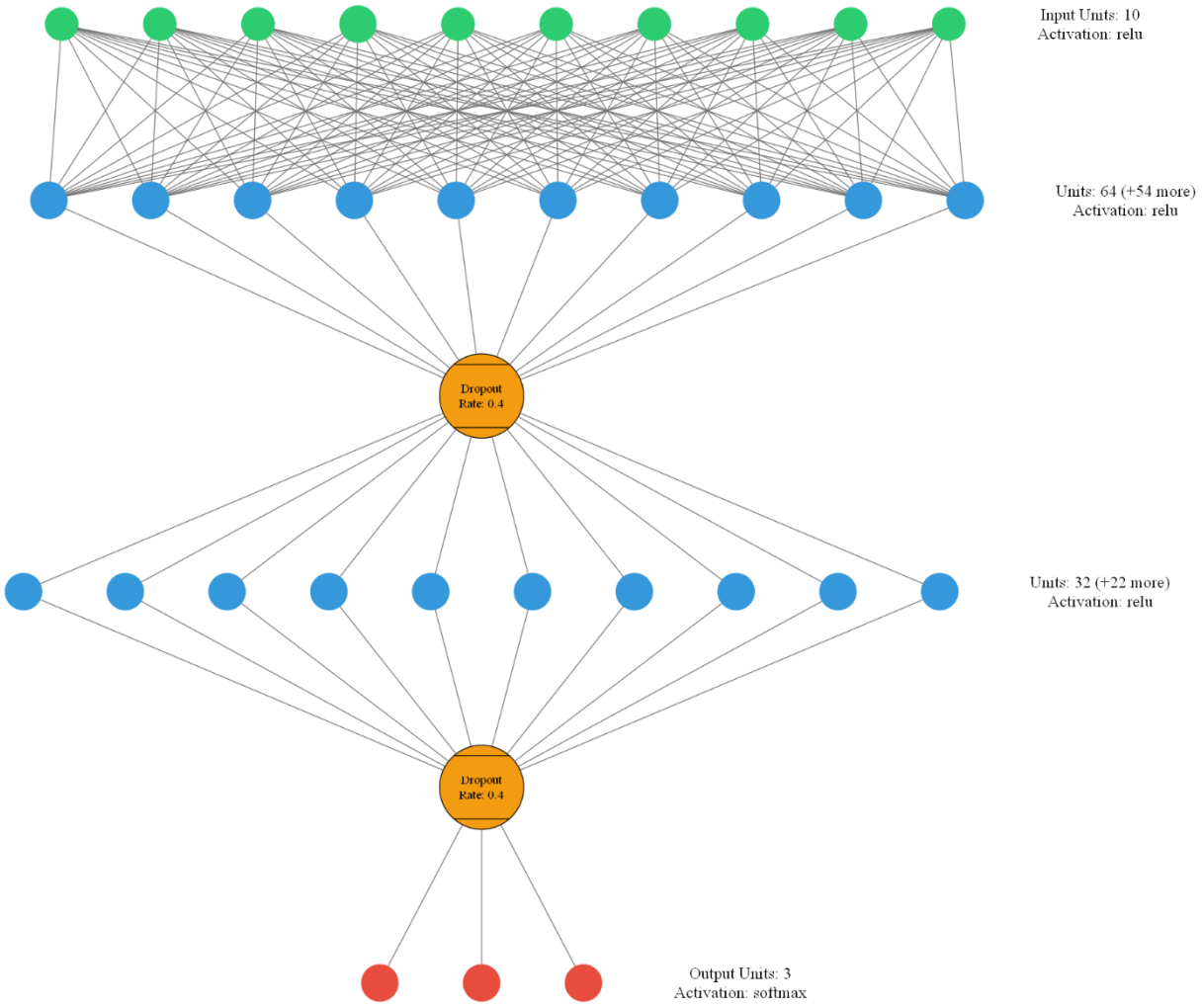


Рисунок 2.27 – Архітектура нейронної мережі для здійснення прогнозування режимів роботи системи опалення та системи кондиціонування

Архітектури нейронних мереж, показані на рисунку 2.26 і рисунку 2.27, були розроблені за допомогою програмного рішення keras-visualizer [30].

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ

У цьому розділі цієї кваліфікаційної роботи описана розробка основних складових програмного комплексу за допомогою мови програмування Python [31] Усі програмні модулі розробляються в інтегрованому середовищі розробки Jetbrains Pycharm [32].

3.1 Генерація набору даних

3.1.1 Генерація часового ряду

Клас TimestampGenerator був створений для генерації часових міток із заданими параметрами (див. Додаток А.1.1).

Метод generate_timestamps – це статичний метод, який не потребує створення об’єкту класу для його використання. Він генерує список (або DatetimeIndex) часових міток у межах певного періоду часу [33].

3.1.2 Генерація кімнат

Клас RoomGenerator був створений, для автоматичної генерації списку кімнат із заданою кількістю (див. Додаток А.1.2).

Метод generate_rooms – це статичний метод, що не потребує створення екземпляру класу для його використання. Він генерує список кімнат із назвами у форматі «room N», де N – порядковий номер кімнати [31].

3.1.3 Генерація зовнішньої температури

Клас OutsideTemperatureGenerator був створений для генерації даних про зовнішню температуру для заданих часових міток. Він створює модель зміни температури на основі сезонного коливання, додаючи випадкові коливання для реалістичності (див. Додаток А.1.3).

Метод generate_outside_temperature – це статичний метод, який створює часову серію температур (об'єкт pandas.Series), відповідну до заданих часових міток [34].

3.1.4 Симуляція присутності мешканців

Клас ResidentPresence моделює присутність мешканців у приміщенні та їхню активність, враховуючи час і ймовірності поведінки. Він визначає, чи присутній мешканець, і розраховує кількість людей у кімнаті та використання гарячої води в певний момент часу (див. Додаток А.1.4).

Методи класу:

— метод: is_resident_present визначає, чи присутній мешканець у заданий час [31]. Вхідний параметр: timestamp – об’єкт datetime, який представляє дату й час. Результат: повертає True, якщо мешканець присутній, і False в іншому випадку;

— метод calculate_activity моделює активність мешканців у заданий час, враховуючи їхню присутність, перебування в кімнаті, і використання гарячої води [31]. Вхідні параметри: number_residents – кількість мешканців у приміщенні (ціле число). timestamp: об’єкт datetime, який представляє дату й час. Результат: повертає кортеж: number_residents_in_room – кількість мешканців у кімнаті, hot_water_usage – використання гарячої води в літрах.

3.1.5 Симуляція середовища в приміщеннях

Клас RoomEnvironment моделює температуру в приміщенні залежно від кількості мешканців у кімнаті та випадкових факторів. Він використовується для симуляції змін у температурі кімнати в реалістичних умовах (див. Додаток А.1.5).

Метод calculate_room_temperature обчислює температуру в кімнаті залежно від кількості мешканців і випадкових змін [31]. Вхідний параметр:

number_residents_in_room – кількість мешканців у кімнаті (ціле число). Результат: повертає значення температури в кімнаті.

3.1.6 Симуляція керування системою нагрівання води

Клас BoilerStatus визначає стан системи нагрівання води (увімкнена чи вимкнена) залежно від споживання гарячої води або за фіксованим розкладом. Він використовується для моделювання роботи системи нагрівання води в різних режимах (див. Додаток А.1.6).

Методи класу:

— метод determine_status_during_peak_hours визначає стан системи нагрівання води, враховуючи споживання гарячої води і час доби [31]. Вхідні параметри: timestamp – об’єкт datetime, який представляє поточну дату й час, hot_water_usage – кількість використаної гарячої води в літрах (числове значення). Результат: повертає «on» або «off» залежно від умов;

— метод determine_status_fixed_schedule визначає стан системи нагрівання води за фіксованим розкладом [31]. Вхідний параметр: timestamp– об’єкт datetime, який представляє поточну дату й час. Результат: повертає «on» або «off» залежно від часу.

3.1.7 Симуляція керування системами опалення та кондиціонування

Клас HeatingCoolingStatus відповідає за визначення стану системи опалення та кондиціонування залежно від температури в кімнаті, кількості мешканців і бажаної температури. Він використовується для моделювання управління кліматом у приміщенні (див. Додаток А.1.7).

Методи класу:

— метод determine_status_based_presence_and_temperature визначає, чи потрібно увімкнути опалення, кондиціонування або вимкнути систему, враховуючи: поточну температуру в кімнаті, кількість мешканців у кімнаті,

бажану температуру [31]. Вхідні параметри: room_temperature – поточна температура в кімнаті (число), residents_in_room – кількість мешканців у кімнаті (ціле число), desired_temperature – бажана температура в кімнаті (число). Результат: повертає «heating», «cooling» або «off»;

— метод: determine_status_based_temperature визначає стан системи опалення та кондиціонування лише на основі температури в кімнаті та бажаної температури [31]. Вхідні параметри: room_temperature – поточна температура в кімнаті (число), desired_temperature – бажана температура в кімнаті (число). Результат: повертає «heating», «cooling» або «off».

3.2 Аналіз набору даних і моделей

3.2.1 Текстовий аналіз набору даних

Клас TextDataAnalyzer використовується для аналізу даних у вигляді текстових звітів. Він надає різні методи для отримання інформації про набір даних (DataFrame) у зручному текстовому форматі (див. Додаток А.2.1).

Методи класу:

— метод get_dataset_info отримує базову інформацію про набір даних (наприклад, типи стовпців, кількість ненульових значень, розмір даних тощо) і повертає її як текст [33]. Вхідний параметр: dataset – об’єкт pandas.DataFrame, який необхідно проаналізувати. Результат: повертає текстову версію інформації, отриманої з info();

— метод get_null_counts_info визначає кількість пропущених (NaN) значень у кожному стовпці набору даних і повертає цю інформацію як текст [33]. Вхідний параметр: dataset об’єкт pandas.DataFrame, який необхідно проаналізувати. Результат: повертає текстовий звіт із кількістю пропущених значень по стовпцях;

— метод get_data_info надає описову статистику для числових стовпців набору даних і повертає її у вигляді тексту [33]. Вхідний параметр:

dataset – об’єкт pandas.DataFrame, який необхідно проаналізувати. Результат: повертає текстовий звіт із описовою статистикою.

3.2.2 Візуальний аналіз набору даних

Клас VizualDataAnalyzer є засобом для візуалізації даних (див. Додаток А.2.2). Він містить методи для створення різноманітних типів графіків – таких, як: гістограми, теплові карти, лінійні графіки, діаграми розподілу тощо. Цей клас побудований на основі бібліотек Matplotlib та Seaborn [35].

Методи класу:

— метод create_count_plot створює стовпчасту діаграму для підрахунку кількостей у категоріях. Вхідні параметри: size – розмір графіку, data – набір даних, x_column_name – ім’я колонки для осі X, title – заголовок графіку, x_label – підпис осі X, y_label – підпис осі Y. Результат: повертає об’єкт Figure [36];

— метод create_hist_plot створює гістограму для числової змінної з лінією згладженого розподілу. Вхідні параметри: size – розмір графіку, data – набір даних, target_column_name – ім'я колонки, яка аналізується, title – заголовок графіку, x_label – підпис осі X, y_label – підпис осі Y. Результат: повертає об’єкт Figure [36];

— метод create_heatmap створює теплову карту кореляцій між числовими змінними. Вхідні параметри: size – розмір графіку, data – набір даних, title – заголовок графіку. Результат: повертає об’єкт Figure [36];

— метод create_plot створює лінійний графік для відображення змін однієї змінної відносно іншої. Вхідні параметри: size – розмір графіку, data – набір даних, x_data – ім’я колонки для осі X, y_data – ім’я колонки для осі Y, label – мітка для лінії, title – заголовок графіку, x_label – підпис *осі* X, y_label – підпис осі Y. Результат: повертає об’єкт Figure [36];

— метод create_boxplot створює коробкову діаграму, яка показує розподіл числової змінної відносно категорійної. Вхідні параметри: size –

розмір графіку, data – набір даних, x_column_name –ім’я колонки для осі X, y_column_name – ім’я колонки для осі Y, title – заголовок графіку, x_label – підпис осі X, y_label – підпис осі Y. Результат: повертає об’єкт Figure [36];

— метод create_scatterplot створює діаграму розсіювання для аналізу взаємозв’язків між двома змінними. Вхідні параметри: size – розмір графіку, data – набір даних, x_column_name – ім’я колонки для осі X, y_column_name – ім’я колонки для осі Y, title – заголовок графіку, x_label – підпис осі X, y_label – підпис осі Y. Результат: повертає об’єкт Figure [36]**.**

3.2.3 Візуальний аналіз моделей штучних нейронних мереж

Клас VizualModelAnalyzer є засобом для візуалізації структури та архітектури моделей штучних нейронних мереж, створених за допомогою Keras. Він містить два статичні методи для створення та збереження графічних представлень моделей (див. Додаток А.2.3).

— метод save_model_structure використовує функцію plot_model з Keras для візуалізації структури моделі. Вхідні параметри: model – модель Keras, яку потрібно візуалізувати, path – шлях до файлу для збереження візуалізації [37];

— метод save_model_architecture використовує пакет keras_visualizer для створення схеми архітектури моделі. Вхідні параметри: model – модель Keras, яку потрібно візуалізувати, path – шлях до файлу для збереження схеми [30].

3.3 Обробка набору даних

3.3.1 Обробка даних

Клас DataProcessor є засобом для обробки та підготовки даних, які можуть використовуватися в машинному навчанні та аналізі. Він містить набір статичних методів, які виконують різноманітні задачі, пов’язані з обробкою

дат, числових даних, категорійних змінних та роботою з часовими рядами (див. Додаток А.3.1).

Методи класу:

— метод convert_timestamp_to_date_format конвертує стовпець з часовими позначками в формат datetime. Приймає DataFrame і назву стовпця, повертає стовпець із перетвореними даними [33];

— метод create_hour витягує години з даних типу datetime [33];

— метод calculate_time_delta обчислює часову різницю між послідовними значеннями у групі, виражену в годинах [33];

— метод resample_data_sum здійсню перевибірку даних із сумуванням значень за день [33];

— метод resample_data_mean здійснює перевибірку даних із обчисленням середнього арифметичного значення за день [33];

— метод create_binary_classes_data перетворює цільовий стовпець у бінарні класи (0 або 1) залежно від умови [33];

— метод standardize_features нормалізує числові змінні до стандартного розподілу з середнім 0 і дисперсією 1 за допомогою StandardScaler [38];

— метод create_multy_class_data створює мапінг унікальних значень цільового стовпця в числові класи [33];

— метод encode_target_variable кодує цільову змінну в формат one-hot-encoding з використанням LabelEncoder і to_categorical [39];

— метод encode_features кодує зазначені стовпці з використанням one-hot-encoding [40];

— метод concatenate_features об’єднує декілька наборів даних у єдину структуру [34];

— метод fill_missing_with_value заповнює пропущені значення в зазначеному стовпці певним значенням [33].

3.3.2 Обробка набору даних

Клас DatasetProcessor є засобом для обробки та маніпуляції з даними у форматі pandas.DataFrame (див. Додаток А.3.2). Він містить набір статичних методів, які виконують різноманітні операції з даними, зокрема створення, зміну, розділення та об’єднання даних.

Методи класу:

— метод create_dataframe створює DataFrame з вхідних даних (наприклад, списків або словників) [33];

— метод set_index встановлює вказаний стовпець як індекс у DataFrame [33];

— метод create_column додає новий стовпець у DataFrame з вказаними значеннями [33];

— метод replace_data_in_column замінює значення в цільовому стовпці відповідно до заданих умов [33];

— метод create_column_by_condition створює новий стовпець, значення якого формуються шляхом формування словнику значень з іншого стовпця [33];

— метод rename_column перейменовує цільовий стовпець у DataFrame [33];

— метод separate_dataset_by_column розділяє DataFrame на тренувальний і тестовий набори даних за цільовим стовпцем [33];

— метод get_columns вибирає зазначені стовпці з DataFrame [33];

— метод concatenate_datasets об’єднує декілька DataFrame по стовпцях [33].

3.4 Файлові операції

3.4.1 Читання набору даних із файлу

Клас DatasetReader є засобом для зчитування наборів даних із файлів у форматі CSV (див. Додаток А.4.1). Він надає один статичний метод, який дозволяє завантажувати дані з автоматичним розпізнаванням дат.

Метод read_dataset зчитує файл CSV за вказаним шляхом і повертає його у вигляді pandas.DataFrame. Вхідні параметри: path – шлях до CSV-файлу, date_column_name – назва стовпця, який містить дані у форматі часу/дати. Результат: повертає об’єкт типу pandas.DataFrame із завантаженими даними [33].

3.4.2 Запис набору даних у файл

Клас DatasetSaver є засобом для збереження наборів даних у файли формату CSV (див. Додаток А.4.2). Він забезпечує простий інтерфейс для збереження даних, представлених у вигляді об’єкту pandas.DataFrame.

Метод save_dataset зберігає вхідний DataFrame у файл CSV за вказаним шляхом. Вхідні параметри: data – об’єкт типу pandas.DataFrame, який потрібно зберегти, path – шлях до файлу, де дані будуть збережені [33].

3.4.3 Завантаження моделі із файлу

Клас ModelLoader є засобом для завантаження збережених моделей, створених за допомогою бібліотеки Keras (див. Додаток А.4.3). Він забезпечує простий спосіб відновлення раніше збережених моделей для подальшого використання.

Метод load_model завантажує модель Keras зі збереженого файлу за вказаним шляхом. Вхідні параметри: path – шлях до файлу моделі. Результат: повертає об’єкт моделі Keras, готовий для використання [41].

3.4.4 Збереження моделі у файл

Клас ModelSaver засобом для збереження моделей, створених за допомогою бібліотеки Keras (див. Додаток А.4.4). Він забезпечує простий спосіб зберегти модель у файл для подальшого використання.

Метод save_model зберігає модель Keras у файл за вказаним шляхом. Вхідні параметри: model – об’єкт моделі Keras, яку потрібно зберегти, path – шлях до файлу, куди буде збережена модель [41].

3.4.5 Збереження графіку у файл

Клас PlotSaver є засобом для збереження графіків, створених за допомогою бібліотеки Matplotlib. Він дозволяє легко зберігати візуалізації у вигляді файлів різних форматів, наприклад, PNG, JPG, PDF тощо, і автоматично закриває графік після збереження, щоб уникнути проблем із пам’яттю (див. Додаток А.4.5).

Метод save_plot зберігає переданий графік за вказаним шляхом у заданому форматі. Вхідні параметри: fig – об’єкт типу matplotlib.figure.Figure, який потрібно зберегти, path – шлях до файлу, куди буде збережений графік [35].

3.4.6 Збереження текстового рядку у файл

Клас StringSaver є засобом для збереження текстових даних у файл (див. Додаток А.4.6). Він дозволяє записувати рядки у файли з можливістю вибору режиму запису (наприклад, створення нового файлу або додавання до існуючого).

Метод save_string зберігає текстовий рядок у файл за вказаним шляхом у вибраному режимі. Вхідні параметри: data – рядок тексту, який потрібно записати у файл, path – шлях до файлу, куди буде записаний текст, mode – режим запису у файл [42].

3.5 Побудова моделей штучних нейронних мереж

3.5.1 Розробка моделі для прогнозування режиму роботи системи нагрівання води

Клас BoilerStatusModel використовується для задач бінарної класифікації (наприклад, визначення режиму роботи системи нагрівання води: увімкнена або вимкнена) (див. Додаток А.5.1).

Методи класу:

— конструктор класу __init__ ініціалізує дані для тренування і тестування, а також кількості епох. Містить атрибут model, що зберігає створену модель [31];

— метод build_model створює нейронну мережу [43];

1) вхідний шар із розміром, що відповідає кількості ознак у тренувальному наборі [43];

2) два прихованих шари із 64 і 32 нейронами, функцією активації ReLU та регуляризацією L2 [43];

3) щари Dropout для запобігання перенавчанню (із ймовірністю 0.4) [43];

4) вихідний шар із одним нейроном та активацією «sigmoid» [43];

— метод compile_model: компілює модель із функцією втрат binary_crossentropy, оптимізатором «adam» та метрикою «accuracy» [44];

— метод fit_model навчає модель на тренувальних даних протягом заданої кількості епох [44]. Результат: повертає об’єкт history, що містить метрики навчання;

— метод evaluate_model оцінює модель на тестових даних [44]. Результат: повертає втрати і точність;

— метод get_model повертає побудовану модель [31].

3.5.2 Розробка моделі для прогнозування режимів роботи систем опалення та кондиціонування

Клас HeatingCoolingStatusModel використовується для задач багатокласової класифікації (наприклад, визначення режиму роботи системи нагрівання води: увімкнена система опалення, увімкнена система кондиціонування або вимкнені всі системи) (див. Додаток А.5.2).

Методи класу:

— конструктор класу __init__ ініціалізує дані для тренування і тестування, а також кількості епох. Містить атрибут model, що зберігає створену модель [31];

— метод build_model створює нейронну мережу [43];

1) вхідний шар із розміром, що відповідає кількості ознак у тренувальному наборі [43];

2) два прихованих шари із 64 і 32 нейронами, функцією активації ReLU та регуляризацією L2 [43];

3) щари Dropout для запобігання перенавчанню (із ймовірністю 0.4) [43];

4) вихідний шар із трьома нейронами та активацією «softmax» [43];

— метод compile_model: компілює модель із функцією втрат categorical_crossentropy, оптимізатором «adam» та метрикою «accuracy» [44];

— метод fit_model навчає модель на тренувальних даних протягом заданої кількості епох [44]. Результат: повертає об’єкт history, що містить метрики навчання;

— метод evaluate_model оцінює модель на тестових даних [44]. Результат: повертає втрати і точність;

метод get_model повертає побудовану модель [31].

3.6 Математичні операції

3.6.1 обчислення використання енергії

Клас EnergyConsumptionCalculator є засобом для обчислення споживання енергії на основі даних про режим роботи системи нагрівання води, режимів роботи систем опалення та кондиціонування і часу роботи (див. Додаток А.6.1). Він забезпечує обчислення та додавання нової колонки з результатами обчислення споживання енергії до наданого набору даних.

Метод calculate_energy_consumption обчислює споживання енергії для різних умов роботи системи нагрівання води і систем опалення та кондиціонування [34]. Вхідні параметри: data – набір даних, що містить інформацію про режими роботи систем і тривалість роботи систем, column_names – словник із назвами колонок, що використовуються для умов, energy_consumption – словник із даними про споживання для різних систем. Результат: повертає модифікований DataFrame, у який додано нову колонку з обчисленим споживання енергії.

3.6.2 обчислення економічних параметрів

Клас EconomicCalculator є засобом для виконання економічних розрахунків, пов’язаних із сумарними значеннями, витратами та економічною вигодою (див. Додаток А.6.2). Він містить статичні методи для виконання обчислень із використанням вхідних даних.

Методи класу:

— метод calculate_sum обчислює суму значень у вказаному стовпці набору даних [33]. Вхідні параметри: data – об’єкт pandas.DataFrame, що містить дані, target_column_name – назва стовпця, для якого обчислюється сума. Результат: повертає сумарне значення;

— метод calculate_cost обчислює вартість, засновану на споживанні і тарифі [31]. Вхідні параметри: consumption – кількість спожитого ресурсу,

tariff – тариф на одиницю ресурсу. Результат: повертає: загальну вартість як добуток споживання і тарифу;

— метод calculate_benefit обчислює економічну вигоду шляхом порівняння попереднього значення із поточним [31]. Вхідні параметри: previous_value – попереднє значення, current_value – поточне значення. Результат: повертає різницю між попереднім і поточним значенням, що відображає економічну вигоду.

3.6.3 Додаткові математичні операції

Клас MathUtils є засобом для виконання різноманітних математичних операцій, використовуваних у задачах обробки даних і машинного навчання (див. Додаток А.6.3). Він надає статичні методи для роботи з пороговими функціями, масивами, обчисленнями кількості унікальних класів і округлення чисел.

Методи класу:

— метод threshold_function застосовує порогову функцію до масиву даних [34]. Вхідні параметри: data – масив числових даних, threshold – поріг, за яким значення будуть класифіковані. Результат: повертає масив із бінарними мітками:1 або 0;

— метод get_indices_max_elements знаходить індекси максимальних елементів уздовж заданої осі масиву [34]. Вхідні параметри: data – масив, у якому потрібно знайти максимальні елементи, n_axis – вісь, уздовж якої здійснюється пошук. Результат: повертає масив індексів максимальних елементів;

— метод get_number_classes розраховує кількість унікальних класів у масиві даних [34]. Вхідні параметри: data – масив із класами або категоріями. Результат: повертає ціле число, що відображає кількість унікальних значень у масиві;

— метод round_number округлює число до заданої кількості десяткових знаків [45]. Вхідні параметри: number – число, яке потрібно округлити, n_digits – кількість десяткових знаків для округлення. Результат: повертає: округлене число.

3.7 Графічний інтерфейс користувача

3.7.1 Графічний інтерфейс користувача навігації у програмному комплексі

Клас MainWindow створює головне вікно графічного інтерфейсу користувача програмного комплексу для дослідження енергетичної ефективності. Інтерфейс користувача містить заголовок і набір кнопок, кожна з яких відповідає за певну функціональність програмного комплексу.

Елементи інтерфейсу користувача:

— текстова мітка. Текст: «Main Menu». Розташований у верхній частині вікна [46];

— кнопка «Create Dataset» відкриває вікно для створення нового набору даних [47];

— кнопка «Analyze Dataset» запускає функцію аналізу даних та збереження результатів [47];

— кнопка «Create Models» відкриває вікно для створення моделей штучних нейронних мереж [47];

— кнопка «Run Models» відкриває вікно для перегляду результатів роботи моделей [47];

— кнопка «Assess Economic Indicators» відкриває вікно для оцінки економічних показників [47].

3.7.2 Графічний інтерфейс користувача для створення набору даних

Клас DatasetCreatorWindow створює вікно для введення параметрів і генерації набору даних. Це вікно забезпечує зручний інтерфейс для користувача, дозволяючи вводити параметри та зберігати отримані дані у вигляді наборів для тренувальних і тестових цілей.

Елементи інтерфейсу користувача:

— текстове поле «Number of days» – кількість днів, на які генерується набір даних [48];

— текстове поле «Number of rooms» – кількість кімнат у будівлі [48];

— текстове поле «Number of residents» – кількість мешканців у будівлі [48];

— текстове поле «Desired temperature» – бажана температура в кімнатах [48];

— текстові мітки описують призначення кожного текстового поля [46];

— кнопка Create Dataset дозволяє користувачу згенерувати набір даних, натиснувши кнопку [47].

3.7.3 Графічний інтерфейс користувача для створення моделей штучних нейронних мереж

Клас ModelsCreatorWindow створює вікно для генерації моделей штучних нейронних мереж. Інтерфейс дозволяє користувачам вказувати кількість епох для навчання моделей і автоматично виконувати повний цикл створення, навчання, оцінювання, збереження моделей та побудови графіків їхньої роботи.

Елементи інтерфейсу користувача:

— текстове поле «Number of epochs (Boiler operating mode)» – кількість епох для моделі прогнозування режиму роботи системи нагрівання води [48];

— текстове поле «Number of epochs (Heating and air conditioning mode)» –кількість епох для для моделі прогнозування режимів роботи систем опалення та кондиціонування [48];

— текстові мітки описують призначення кожного текстового поля [46];

— кнопка «Create Models» дозволяє користувачу створити моделі, натиснувши кнопку [47].

3.7.4 Графічний інтерфейс користувача для перегляду результатів роботи моделей штучних нейронних мереж

Клас DisplayModelResultsWindow створює інтерфейс для демонстрації результатів роботи моделей штучних нейронних мереж. Це вікно дозволяє користувачеві переглядати результати прогнозувань у вигляді таблиці, яка містить початкові дані тестового набору та відповідні прогнози моделей.

Елементи інтерфейсу користувача:

— таблиця результатів відображає тестові дані та результати прогнозувань моделей [49];

— смуга прокрутки – вертикальна смуга прокрутки дозволяє переглядати велику кількість записів у таблиці [50].

3.7.5 Графічний інтерфейс користувача для введення економічних параметрів

Клас EconomicParametersSetterWindow створює інтерфейс для введення економічних параметрів, пов’язаних із споживанням енергії – таких, як споживання енергії різними системами і тариф на енергію. Ці параметри використовуються для оцінки економічних показників.

Елементи інтерфейсу користувача:

— текстове поле «Boiler energy consumption» – споживання енергії системою нагрівання води [48];

— текстове поле «Heating system energy consumption» – споживання енергії системою опалення [48];

— текстове поле «Cooling system energy consumption» – споживання енергії системою охолодження [48];

— текстове поле «Energy tariff» – тариф на енергію [48];

— текстові мітки вказують на призначення кожного текстового поля [46];

— кнопка «Assess Economic Indicators» при натисканні викликає необхідні методи для здійснення обчислення економічних параметрів [47].

3.7.6 Графічний інтерфейс користувача для виведення економічних показників

Клас EconomicParametersDisplayWindow створює вікно для відображення економічних показників – таких, як енергоспоживання та витрати з урахуванням різних режимів роботи систем (адаптивного і неадаптивного). Він дозволяє користувачеві переглядати таблицю з розрахунками та зведені дані економічних переваг.

Елементи інтерфейсу користувача:

— таблиця результатів відображає дані про споживання енергії кімнатами у різних режимах [49];

— смуга прокрутки – вертикальна смуга прокрутки для перегляду великого обсягу даних у таблиці [50];

— текстові мітки відображають основні економічні показники. Адаптивний режим: споживання енергії, витрати на енергію. Неадаптивний режим: споживання енергії, витрати на енергію. Економічні вигоди: зекономлене споживання енергії, зекономлені витрати [46].

4 АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМНОГО КОМПЛЕКСУ

У цьому розділі цієї кваліфікаційної роботи показані результати роботи розробленого програмного комплексу за деяких визначених параметрів.

4.1 Навігація у програмному комплексі

У цьому підрозділі цієї кваліфікаційної роботи описане головне меню розробленого програмного комплексу.

На рисунку 4.1 показане головне меню розробленого програмного комплексу.

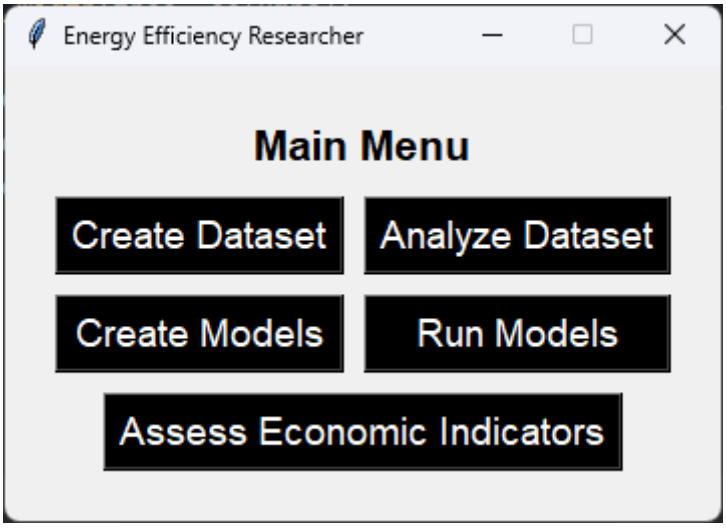


Рисунок 4.1 – Головне меню програмного комплексу

Із головного меню доступні такі основні функції програмного комплексу:

— створення набору даних;
— аналіз набору даних;
— створення та аналіз моделей штучних нейронних мереж;
— перегляд результатів роботи моделей штучних нейронних мереж;
— перегляд економічних показників.

4.2 Створення набору даних

У цьому підрозділі цієї кваліфікаційної роботи показаний процес і результати створення набору даних.

У таблиці 4.1 показані параметри та їхні значення, які враховувалися при створенні набору даних.

Таблиця 4.1 – Параметри та їхні значення для генерації набору даних

Параметр	Значення
Кількість днів, днів	30
Кількість кімнат, шт	5
Кількість мешканців, шт	4
Бажана температура, °C	22

На рисунку 4.2 показаний графічний інтерфейс для встановлення значень параметрів набору даних.

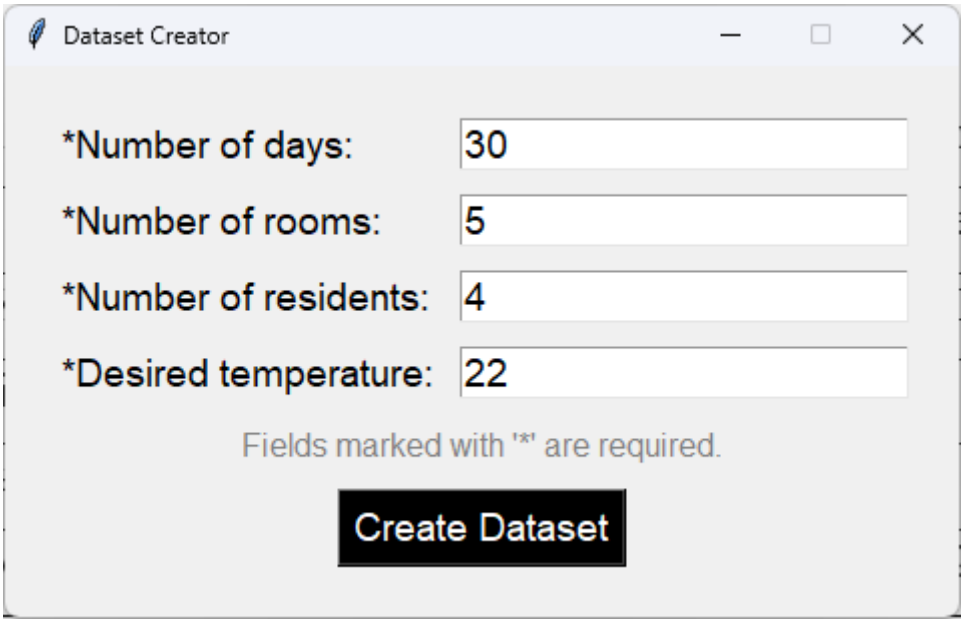


Рисунок 4.2 – Графічний інтерфейс для встановлення значень параметрів набору даних

У таблиці 4.2 показані сутності набору даних та їхні типи, які створюються за допомогою програмного комплексу.

Таблиця 4.2 – Результати створення набору даних

Сутність	Тип даних
Часова мітка	Дата і час
Назва кімнати	Рядок
Кількість мешканців, які перебувають у кімнаті	Ціле число
Температура в кімнаті	Дробове число
Температура довкілля	Дробове число
Використання гарячої води	Дробове число
Режим роботи системи нагрівання води (адаптивний)	Рядок
Режим роботи системи нагрівання води (звичайний)	Рядок
Режим роботи системи опалення і кондиціонування (адаптивний)	Рядок
Режим роботи системи опалення і кондиціонування (звичайний)	Рядок

Після генерації усі дані зберігаються у файл формату «.csv». Всього створюється 216005 записів.

4.3 Аналіз набору даних

У цьому підрозділі цієї кваліфікаційної роботи описаний аналіз набору даних. Для здійснення аналізу набору даних необхідно натиснути відповідну кнопку «Analyze Dataset» в головному меню розробленого програмного комплексу (рис. 4.1).

Аналіз набору даних був проведений для глибокого розуміння результатів роботи алгоритму генерації набору даних. Основна мета аналізу полягає в тому, щоб:

— визначити закономірності у поведінці мешканців та роботі пристроїв;

— оцінити вплив зовнішніх факторів – таких, як температура довкілля, на внутрішній мікроклімат будинку;

— порівняти режими роботи пристроїв;

— виявити можливості оптимізації для подальшого зниження витрат і підвищення комфорту.

Були здійснені такі види аналізу набору даних:

— аналіз середньодобового використання гарячої води (див. Додаток Б.1);

— аналіз середньодобової температури довкілля (див. Додаток Б.2);

— аналіз середньодобової температури в приміщенні (див. Додаток Б.3);

— аналіз розподілу режиму роботи системи нагрівання води (за адаптивним графіком) (див. Додаток Б.4);

— аналіз розподілу режиму роботи системи нагрівання води (за звичайним графіком) (див. Додаток Б.5);

— аналіз розподілу температури довкілля (див. Додаток Б.6);

— аналіз розподілу температури в приміщеннях (див. Додаток Б.7);

— аналіз залежності температури в приміщенні від температури довкілля (див. Додаток Б.8);

— аналіз розподілу кількості мешканців (див. Додаток Б.9);

— аналіз розподілу режимів роботи системи опалення та системи кондиціонування (за адаптивним графіком) (див. Додаток Б.10);

— аналіз розподілу режимів роботи системи опалення та системи кондиціонування (за звичайним графіком) (див. Додаток Б.11);

— аналіз залежності використання гарячої води від кількості мешканців (див. Додаток Б.12);

— аналіз кореляції між числовими даними у наборі (див. Додаток Б.13).

4.4 Створення моделей штучних нейронних мереж

У цьому підрозділі цієї кваліфікаційної роботи показаний процес і результати створення моделей штучних нейронних мереж для прогнозування режимів роботи системи нагрівання води, системи опалення та системи кондиціонування.

У таблиці 4.3 показані параметри та їхні значення, необхідні для створення моделей штучних нейронних мереж.

Таблиця 4.3 – Параметри та їхні значення для створення моделей штучних нейронних мереж

Параметр	Значення
Кількість епох (модель для прогнозування режиму роботи системи нагрівання води), шт	5
Кількість епох (модель для прогнозування режимів роботи системи опалення та системи кондиціонування), шт	5

На рисунку 4.3 показаний графічний інтерфейс розробленого програмного комплексу для створення моделей штучних нейронних мереж.

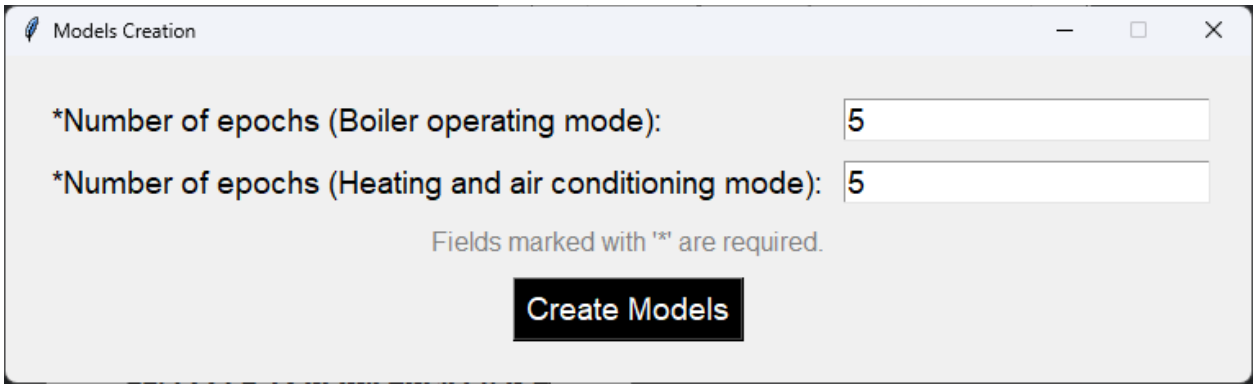


Рисунок 4.3 – Графічний інтерфейс для створення моделей штучних нейронних мереж

У таблиці 4.4 показані результати перевірки моделей штучних нейронних мереж для прогнозування режимів роботи систем нагрівання води, опалення та кондиціонування на тестових даних. Для здійснення тестування моделей штучних нейронних мереж використовуються 43200 записів із набору даних.

Таблиця 4.4 – Результати тестування моделей штучних нейронних мереж

Модель	Точність (%)	Втрати
Модель для здійснення прогнозування режиму роботи системи нагрівання води	99	0,01
Модель для здійснення прогнозування режимів роботи системи опалення та системи кондиціонування	99	0,02

Також, окрім файлів моделей, результатом роботи модулю є:

— графіки залежності точності моделей від кількості епох на навчальних даних (див. Додаток В.1 і Додаток В.3);

— графіки залежності втрат моделей від кількості епох на навчальних даних (див. Додаток В.2 і Додаток В.4);

— структури моделей штучних нейронних мереж (див. Додаток Г.1 і Додаток Г.2).

4.5 Прогнозування даних

У цьому підрозділі цієї кваліфікаційної роботи показані результати роботи моделей штучних нейронних мереж, а саме:

— прогнозування режиму роботи системи нагрівання води;

— прогнозування режимів роботи системи опалення та системи кондиціонування.

Для прогнозування режиму роботи системи нагрівання води використовуються такі дані:

— дані про назву кімнати;

— дані про присутність мешканців у кімнаті;

— дані про використання гарячої води.

Для прогнозування режимів роботи системи опалення та системи кондиціонування використовуються такі дані:

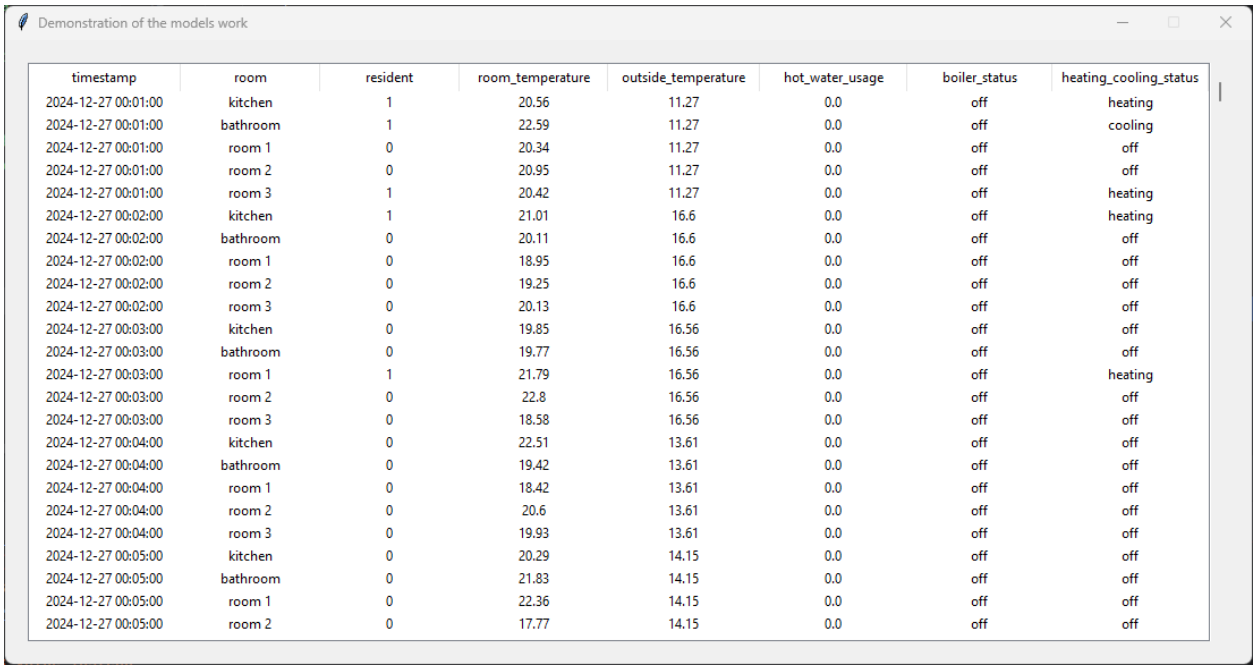
— дані про назву кімнати;

— дані про присутність мешканців у кімнаті;

— дані про температуру довкілля;

— дані про температуру в кімнаті.

На рисунку 4.4 показаний графічний інтерфейс розробленого програмного комплексу з результатами прогнозування режимів роботи системи нагрівання води, системи опалення та системи кондиціонування.



timestamp	room	resident	room_temperature	outside_temperature	hot_water_usage	boiler_status	heating_cooling_status
2024-12-27 00:01:00	kitchen	1	20.56	11.27	0.0	off	heating
2024-12-27 00:01:00	bathroom	1	22.59	11.27	0.0	off	cooling
2024-12-27 00:01:00	room 1	0	20.34	11.27	0.0	off	off
2024-12-27 00:01:00	room 2	0	20.95	11.27	0.0	off	off
2024-12-27 00:01:00	room 3	1	20.42	11.27	0.0	off	heating
2024-12-27 00:02:00	kitchen	1	21.01	16.6	0.0	off	heating
2024-12-27 00:02:00	bathroom	0	20.11	16.6	0.0	off	off
2024-12-27 00:02:00	room 1	0	18.95	16.6	0.0	off	off
2024-12-27 00:02:00	room 2	0	19.25	16.6	0.0	off	off
2024-12-27 00:02:00	room 3	0	20.13	16.6	0.0	off	off
2024-12-27 00:03:00	kitchen	0	19.85	16.56	0.0	off	off
2024-12-27 00:03:00	bathroom	0	19.77	16.56	0.0	off	off
2024-12-27 00:03:00	room 1	1	21.79	16.56	0.0	off	heating
2024-12-27 00:03:00	room 2	0	22.8	16.56	0.0	off	off
2024-12-27 00:03:00	room 3	0	18.58	16.56	0.0	off	off
2024-12-27 00:04:00	kitchen	0	22.51	13.61	0.0	off	off
2024-12-27 00:04:00	bathroom	0	19.42	13.61	0.0	off	off
2024-12-27 00:04:00	room 1	0	18.42	13.61	0.0	off	off
2024-12-27 00:04:00	room 2	0	20.6	13.61	0.0	off	off
2024-12-27 00:04:00	room 3	0	19.93	13.61	0.0	off	off
2024-12-27 00:05:00	kitchen	0	20.29	14.15	0.0	off	off
2024-12-27 00:05:00	bathroom	0	21.83	14.15	0.0	off	off
2024-12-27 00:05:00	room 1	0	22.36	14.15	0.0	off	off
2024-12-27 00:05:00	room 2	0	17.77	14.15	0.0	off	off

Рисунок 4.4 Графічний інтерфейс з результатами прогнозування режимів роботи систем нагрівання води опалення та кондиціонування

На рисунку 4.4 показані такі дані:

— часова мітка;

— назва кімнати;

— кількість присутніх мешканців у кімнаті;

— температура в кімнаті;

— температура довкілля;

— використання гарячої води;

— режим роботи системи нагрівання води;

— режими роботи систем опалення та кондиціонування.

4.6 Аналіз економічної ефективності оптимізації режимів роботи систем

У цьому підрозділі цієї кваліфікаційної роботи показаний процес і результати аналізу економічної ефективності оптимізації режимів роботи системи нагрівання води, системи опалення та системи кондиціонування.

У таблиці 4.5 показані параметри та їхні значення, необхідні для здійснення аналізу економічної ефективності оптимізації режимів роботи системи нагрівання води, системи опалення та системи кондиціонування.

Таблиця 4.5 – Параметри та їхні значення для здійснення оцінки економічної ефективності оптимізації режимів роботи систем нагрівання води, опалення та кондиціонування

Параметр	Значення
Споживання енергії системою нагрівання води, кВт-год	1,5
Споживання енергії системою опалення, кВт-год	2
Споживання енергії системою кондиціонування, кВт-год	0.5
Вартість енергії, грн	2,64

На рисунку 4.5 показаний графічний інтерфейс розробленого програмного комплексу для встановлення параметрів та їхніх значень для здійснення аналізу економічної ефективності оптимізації режимів роботи системи нагрівання води, системи опалення та системи кондиціонування.

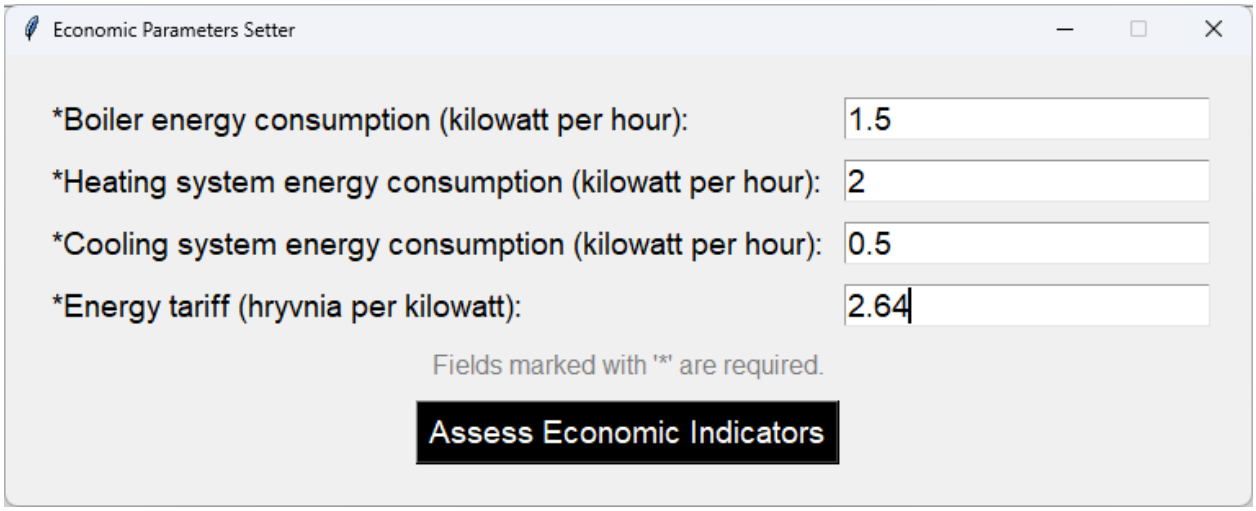


Рисунок 4.5 – Графічний інтерфейс для встановлення значень параметрів для здійснення оцінки економічної ефективності програмного комплексу

На рисунку 4.6 показані результати оцінки економічної ефективності оптимізації системи нагрівання води, системи опалення та системи кондиціонування.

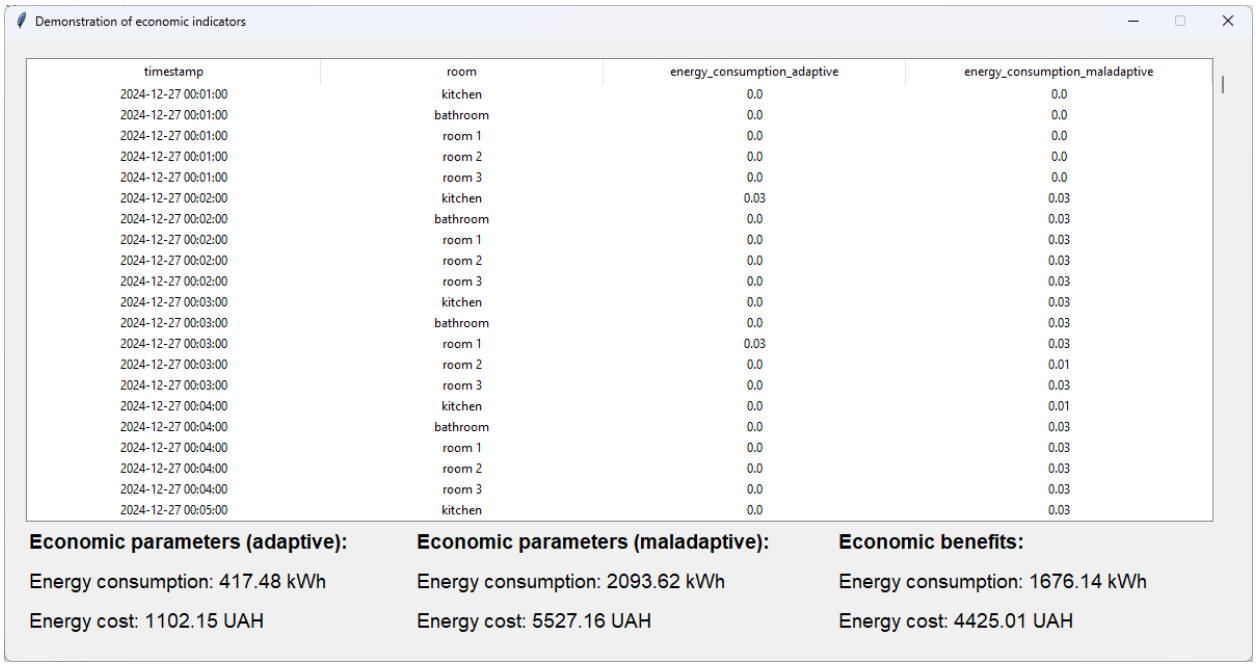


Рисунок 4.6 – Демонстрація оцінки економічної ефективності програмного комплексу

На рисунку 4.6 у таблиці показані такі дані:

— часова мітка;

— назва кімнати;

— споживання енергії (адаптивний режим);

— споживання енергії (звичайний режим).

У таблиці 4.6 показані результати оцінки економічної ефективності оптимізації режимів роботи системи нагрівання води, системи опалення та системи кондиціонування.

Таблиця 4.6 – Результати оцінки економічної ефективності оптимізації режимів роботи систем нагрівання води, опалення та кондиціонування

Показник	Адаптивний режим	Звичайний режим	Економічна вигода
Споживання енергії, кВт-год	417,48	2093,62	1676,14
Вартість енергії, грн	1102,15	5527,16	4425,01

Згідно із даними, показаними у таблиці 4.6, можна дійти висновку, що оптимізація режимів роботи системи нагрівання води, системи опалення і системи кондиціонування економічно виправдана. Економічна вигода становить: 19,94%.

4.7 Узагальнення результатів

У межах розробки програмного комплексу, імовірно, були досягнуті деякі науково-технічні результати, які можуть знайти застосування у галузі енергетичної ефективності та інтелектуальних систем керування в житлових приміщеннях. Нижче наведені основні результати розробки програмного комплексу:

— розробка методології генерації та моделювання даних для симуляції системи «Розумний будинок». Наукова цінність: Запропоновано методику генерації синтетичних даних, яка дозволяє моделювати реалістичну поведінку мешканців та роботу систем нагрівання води, опалення та кондиціонування. Ця методика враховує різноманітні фактори – такі, як: зовнішня температура, присутність мешканців, їхню активність та використання ресурсів. Технічна цінність: створено програмний комплекс, який дозволяє генерувати великі обсяги даних для тренування і тестування моделей штучних нейронних мереж;

— розробка алгоритму оптимізованого керування споживанням енергії. Наукова цінність: створений алгоритм, який враховує час, присутність мешканців та використання гарячої води для керування системами нагрівання води, опалення та кондиціонування. Технічна цінність: впровадження такого алгоритму в системи «Розумний будинок» може зменшити споживання енергії без втрати комфорту для мешканців;

— оцінка економічної ефективності оптимізації режимів роботи систем нагрівання води, опалення та кондиціонування. Наукова цінність: проведено комплексний аналіз економічної ефективності застосування алгоритму, зокрема розрахунок зекономленої енергії та коштів. Технічна цінність: результати аналізу можуть бути використані розробниками та споживачами для ухвалення обґрунтованих рішень щодо впровадження систем «Розумний будинок».

Тож, можна дійти висновку, що були досягнуті результати, які можуть знайти застосування в галузі енергетичної ефективності та інтелектуальному керуванні будинком. Запропоновані методології та алгоритми, потенційно, сприяють оптимізації споживання енергії та можуть бути корисними для подальших досліджень і впроваджень у цій галузі.

5 ВИЗНАЧЕННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ПРОГРАМНОГО КОМПЛЕКСУ

5.1 Розрахунок собівартості розробки програмного комплексу

Згідно із прикладом, наведеним у навчальних матеріалах із дисципліни «Інноваційний менеджмент в індустрії програмного забезпечення», собівартість розробки програмного комплексу складається із таких статей втрат:

— комплектуючі вироби;

— витрати на електроенергію;

— основна та додаткова заробітна плата;

— відрахування в соціальні фонди;

— загальновиробничі витрати [52].

У таблиці 5.1 наведені деякі вхідні дані для здійснення обчислення собівартості розробки програмного комплексу.

Таблиця 5.1 – Вхідні дані

Найменування початкових даних	Показник	Джерело отримання
Трудомісткість складання програмного комплексу, днів	40	Фактичні витрати часу на розробку програмного комплексу: 2 місяці по 20 робочих днів
Місячна ставка укладача програмного комплексу, грн	40000,00	Дані, визначені роботодавцем
Кількість годин в місяці, год	160	20 робочих днів по 8 годин
Додаткова заробітна плата (%)	10	Дані, визначені роботодавцем

Продовження таблиці 5.1

Найменування початкових даних	Показник	Джерело отримання
Відрахування до соціальних фондів (%)	22	Закон України «Про збір та облік єдиного внеску на загальнообов'язкове державне соціальне страхування»
Загальновиробничі витрати, (%)	100	Дані, визначені роботодавцем
Податок на додану вартість (%)	20	Податковий кодекс України

Для здійснення розробки програмного комплексу необхідний 1 компакт-диск, вартість якого становить: 75,00 грн.

Вартість комплектуючи виробів, зазвичай, обчислюється за формулою:

$$З_k = \sum Ц_k \cdot n_k\,, \tag{5.1}$$

де $З_k$ – витрати на комплектуючі вироби, грн;

$Ц_k$ – ціна за 1 одиницю комплектуючих виробів, грн;

n_k – кількість комплектуючих виробів деякого типу, шт [52].

Тож, враховуючи вищезазначені вхідні дані та параметри і алгоритм, визначені у формулу 5.1, здійснюється обчислення вартості комплектуючих виробів:

$$З_k = 75 \cdot 1 = 75, \tag{5.2}$$

Результат: вартість комплектуючих виробів дорівнює 75,00 грн.

Для здійснення розробки програмного комплексу необхідна електрична енергія: для роботи комп’ютера та освітлення. Витрати на електричну енергію можна обчислити за формулою 5.3:

$$В_Б = Р_В \sum W_i \cdot t\,, \tag{5.3}$$

де $В_Б$ – витрати на електричну енергію, грн;

$Р_В$ – ціна за 1 кВт-год електричної енергії, грн;

W_i – потужність задіяних пристроїв, кВт;

t – час роботи задіяних пристроїв, год [52].

Враховуючи параметри визначені у формулі 5.3, можна визначити вхідні дані для обчислення витрат на електричну енергію:

— ціна за 1 кВт-год електричної енергії: 4,32 грн;

— потужність комп’ютеру: 0,6 кВт;

— потужність освітлення: 0,03 кВт;

— кількість годин: 160 (табл. 5.).

Враховуючи алгоритм і параметри, визначені у формулі 5.3, а також визначені вхідні дані, можна здійснити обчислення вартості електричної енергії:

$$В_Б = 4{,}32 \cdot (0{,}6 + 0{,}03) \cdot 160 = 435{,}46, \tag{5.4}$$

Результат: вартість електричної енергії дорівнює 435,46 грн.

Для початку необхідно обчислити погодинну тарифну ставку розробника програмного комплексу. Це можна зробити за формулою:

$$l_{год} = \frac{З_{осн}}{T_{год}}, \tag{5.5}$$

де $l_{год}$ – погодинна тарифна ставка, грн;

$З_{осн}$ – основна заробітна плата, грн;

$T_{год}$ – кількість годин, год [52].

Враховуючи алгоритм і параметри, визначені у формулі 5.5, а також визначені вхідні дані, можна здійснити обчислення погодинної тарифної ставки розробника програмного комплексу:

$$l_{год} = \frac{40000}{160} = 250, \quad (5.6)$$

Результат: погодинна тарифна ставка розробника програмного комплексу дорівнює 250 грн.

Враховуючи формулу 5.5, можна вивести формулу для обчислення основної заробітної плати розробника програмного комплексу:

$$З_{осн} = l_{год} \cdot T_{год}, \quad (5.7)$$

де $З_{осн}$ – основна заробітна плата, грн;

$l_{год}$ – погодинна тарифна ставка, грн;

$T_{год}$ – кількість годин, год [52].

Тепер, коли виведена формула для обчислення основної заробітної плати, можна визначити вхідні дані:

— погодинна тарифна ставка: 250 грн (формула 5.6);

— кількість годин: 160 годин (табл. 5.1).

Враховуючи алгоритм і параметри, визначені у формулі 5.7, а також визначені вхідні дані, можна здійснити обчислення основної заробітної плати:

$$З_{осн} = 250 \cdot 160 = 40000, \quad (5.8)$$

Результат: основна заробітна плата розробника програмного комплексу дорівнює 40000,00 грн.

Значення додаткової заробітної плати розробника програмного комплексу можна обчислити:

$$З_{дод} = \frac{З_{осн} \cdot Д\%}{100}, \quad (5.9)$$

де $З_{дод}$ – додаткова заробітна плата розробника програмного комплексу, грн;

$З_{осн}$ – основна заробітна плата розробника програмного комплексу, грн;

Д% – відсоток додаткової заробітної плати розробника програмного комплексу (%) [52].

Враховуючи параметри визначені у формулі 5.9, можна визначити вхідні дані для обчислення додаткової заробітної плати:

— основна заробітна плата: 40000 грн (формула. 5.8);

— відсоток додаткової заробітної плати: 10% (табл. 5.1).

Враховуючи алгоритм і параметри, визначені у формулі 5.9, а також визначені вхідні дані, можна здійснити обчислення додаткової заробітної плати розробника програмного комплексу:

$$З_{дод} = \frac{40000 \cdot 10}{100} = 4000, \quad (5.10)$$

Результат: додаткова заробітна плата розробника програмного комплексу дорівнює 4000 грн.

Значення суми відрахувань у соціальні фонди можна обчислити:

$$З_{соц} = \frac{(З_{осн} + З_{дод}) \cdot С\%}{100}, \quad (5.11)$$

де $З_{соц}$ – сума відрахувань у соціальні фонди, грн;

$З_{осн}$ – основна заробітна плата розробника програмного комплексу, грн;

$З_{дод}$ – додаткова заробітна плата розробника програмного комплексу, грн;

$С\%$ – відсоток відрахувань у соціальні фонди (%) [52].

Враховуючи параметри визначені у формулі 5.11, можна визначити вхідні дані для обчислення суми відрахувань у соціальні фонди:

— Основна заробітна плата розробника програмного комплексу: 40000 грн (формула 5.8);

— Додаткова заробітна плата розробника програмного комплексу: 40000 грн (формула 5.10);

— Відсоток відрахувань у соціальні фонди: 22% (табл. 5.1).

Враховуючи алгоритм і параметри, визначені у формулі 5.11, а також визначені вхідні дані, можна здійснити обчислення суми відрахувань у соціальні фонди:

$$З_{соц} = \frac{(40000 + 4000) \cdot 22}{100} = 9680, \quad (5.12)$$

Результат: сума відрахувань у соціальні фонди дорівнює 9680 грн.

Значення суми загальновиробничих витрат можна обчислити:

$$З_{заг} = \frac{З_{осн} \cdot H_1\%}{100}, \quad (5.13)$$

де $З_{заг}$ – сума загальновиробничих витрат, грн;

$З_{осн}$ – основна заробітна плата розробника програмного комплексу, грн;

$H_1\%$ – відсоток загальновиробничих витрат (%) [52].

Враховуючи параметри визначені у формулі 5.13, можна визначити вхідні дані для обчислення суми загальновиробничих витрат:

— Основна заробітна плата розробника програмного комплексу: 40000 грн (формула 5.8);

— Відсоток загальновиробничих витрат: 100% (табл. 5.1).

Враховуючи алгоритм і параметри, визначені у формулі 5.13, а також визначені вхідні дані, можна здійснити обчислення суми загальновиробничих витрат:

$$З_{заг} = \frac{40000 \cdot 100}{100} = 40000, \quad (5.14)$$

Результат: сума загальновиробничих витрат дорівнює 40000 грн.

Виробнича собівартість обчислюється шляхом додавання усіх раніше обчислених статей витрат на розробку програмного комплексу:

$$S_{nn} = 75 + 435{,}46 + 40000 + 4000 + 9680 + 40000 = 94190{,}46, \quad (5.15)$$

де S_{nn} – виробнича собівартість [52].

Результат: виробнича собівартість розробки програмного комплексу дорівнює 94190,46 грн.

Тепер, коли обчислені усі статті витрат, із яких складається собівартість розробки програмного комплексу, доречним буде занести всі ці значення в таблицю 5.2.

Таблиця 5.2 – Планова калькуляція

Стаття калькуляції	Сума, грн
Стаття 1 – комплектуючі вироби	75
Стаття 2 – витрати на електричну енергію	435,46
Стаття 3 – основна заробітна плата	40000
Стаття 4 – додаткова заробітна плата	4000

Продовження таблиці 5.2

Стаття калькуляції	Сума, грн
Стаття 5 – відрахування в соціальні фонди	9680
Стаття калькуляції	Сума, грн
Стаття 6 – загальновиробничі витрати	40000
Виробнича собівартість, 1 місяць	94190,46
Собівартість програмного комплексу	188380,92

Враховуючи те, що розробка програмного комплексу триває 2 місяці, розрахункова собівартість його розробки складає 188380,92 грн.

5.2 Обчислення інноваційного ефекту від впровадження програмного комплексу

У таблиці 5.3 наведений комплект обладнання, необхідний для роботи розробленого програмного комплексу, а також вартість комплектуючих.

Таблиця 5.3 – Пристрої для забезпечення роботи розробленого програмного комплексу та їхня вартість

Пристрій	Вартість, грн
Комп’ютер	30000
Монітор	6000
Комп’ютерна мишка	1500
Клавіатура	2500

Вартість обладнання, необхідного для забезпечення роботи розробленого програмного комплексу можна обчислити:

$$P_a = P_m \cdot n, \tag{5.16}$$

де P_a – вартість всього обладнання, грн;

P_m – вартість одного комплекту обладнання, грн;

n – кількість комплектів обладнання, шт [52].

Враховуючи вхідні дані (табл. 5.3) та алгоритм і параметри, визначені у формулі 5.16, можна здійснити обчислення загальної вартості обладнання, необхідного для забезпечення роботи розробленого програмного комплексу:

$$P_a = (30000 + 6000 + 1500 + 2500) \cdot 1 = 40000, \quad (5.17)$$

Результат: загальна вартість обладнання, необхідного для забезпечення роботи розробленого програмного комплексу дорівнює 40000,00 грн.

Тепер, використовуючи результати обчислень розробки програмного комплексу, а також вартість обладнання для забезпечення роботи розробленого програмного комплексу, можна обчисли суму інвестицій:

$$I = C_p + C_e, \quad (5.18)$$

де I – сума інвестицій, грн;

C_p – витрати на розробку програмного комплексу, грн;

C_e – сума витрат на обладнання для забезпечення експлуатації програмного комплексу, грн [53].

Враховуючи вхідні дані (табл. 5.2 і табл. 5.3) та алгоритм і параметри, визначені у формулі 5.18, можна здійснити обчислення суми інвестицій:

$$I = 188380{,}92 + 40000 = 228380{,}92, \quad (5.19)$$

Результат: сума інвестицій дорівнює 228380,92 грн.

Для того, щоб визначити економічний ефект від застосування адаптивного режиму керування системою нагрівання води, системою

опалення та системою кондиціонування, необхідно, для початку, з’ясувати які витрати на енергію становлять у звичайному та адаптивному режимах.

Як зазначено у таблиці 4.6, витрати на енергії за 20% місяця (тобто 6 днів) склади:

— в адаптивному режимі: 1102,15 грн;

— у звичайному режимі 5527,16 грн.

Маючи всі необхідні вхідні дані, можна визначити формулу для обчислення економії витрат на енергію:

$$E = E_0 - E_1, \tag{5.20}$$

де E – економія витрат на енергію, грн;

E_0 – витрати на енергію без оптимізації, грн;

E_1 – витрати на енергію після оптимізації, грн [54].

Враховуючи вхідні дані та алгоритм і параметри, визначені у формулі 5.20, можна здійснити обчислення суми економії витрат на енергію:

$$E = 5527{,}16 - 1102{,}15 = 4425{,}01, \tag{5.21}$$

Результат: сума економії витрат за енергію дорівнює 4425,01 грн.

Але треба враховувати, що це економія на витратах за енергію за 6 днів. Для забезпечення коректності подальших обчислень, необхідно обчислити щоденну економію витрат на енергію:

$$E = \frac{4425{,}01}{6} = 737{,}50, \tag{5.22}$$

Результат: щоденна економія на витратах за енергію дорівнює 737,50 грн.

Тепер можна визначити формулу для обчислення періоду окупності інвестицій:

$$T_p = \frac{I_0}{E}, \tag{5.23}$$

де T_p – період окупності інвестицій, днів;

I_0 – сума інвестицій, грн;

E – сума економії витрат на енергію, грн [54].

Враховуючи вхідні дані та алгоритм і параметри, визначені у формулі 5.23, можна здійснити обчислення періоду окупності інвестицій:

$$T_p = \frac{228380{,}92}{737{,}50} = 309{,}66, \tag{5.24}$$

Результат: період окупності інвестицій дорівнює 309,66 днів.

Тож, аналізуючи результати, отримані після здійснених обчислень у підрозділі 5.1 та підрозділі 5.2 цієї кваліфікаційної роботи, можна дійти висновку, що запропонований метод керування системами нагрівання води, опалення та кондиціонування, може бути ефективним з економічної точки зору.

ВИСНОВОК

Під час виконання цієї кваліфікаційної роботи був розроблений програмний комплекс, який дозволяє проводити моделювання та аналіз енергетичної ефективності систем «Розумний будинок». Основною метою цієї роботи було створення засобу, який дозволяє оцінювати вплив різних параметрів на споживання енергії та вивчати можливі сценарії функціонування системи.

Основні результати цієї кваліфікаційної роботи такі:

— був виконаний аналіз вимог до програмного комплексу;

— було здійснене проектування програмного комплексу;

— був створений інтуїтивно зрозумілий інтерфейс користувача;

— були реалізовані принципи генерації даних, які імітують поведінку системи «Розумного будинку»;

— були реалізовані засоби для здійснення аналізу набору даних;

— були реалізовані моделі штучних нейронних мереж для прогнозування режимів роботи системи нагрівання води, системи опалення та системи кондиціонування;

— було здійснене порівняння адаптивного та звичайного режимів роботи системи нагрівання води, системи опалення та системи кондиціонування з економічної точки зору.

Отримані результати демонструють, що створена модель може бути достатньо ефективним засобом для здійснення аналізу можливих сценаріїв функціонування систем «Розумний будинок». Вона дозволяє вивчати потенціал енергетичної ефективності й слугує основою для подальших досліджень і розробок у цій галузі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Kushnir I. METHODICAL REGULATION AND ANALYTICAL AND INFORMATION SUPPORT OF "SMART HOME" PROJECTS IN THE MODERN CONSTRUCTION DEVELOPMENT SYSTEM. *Management of Development of Complex Systems*. 2022. No. 49. P. 97–104. URL: https://doi.org/10.32347/2412-9933.2022.49.97-104 (дата звернення: 23.09.2024).

2. World electricity generation by source 2022 | Statista. Statista. URL: https://www.statista.com/statistics/269811/world-electricity-production-by-energy-source/ (дата звернення: 11.06.2024).

3. Fossil fuels. Our World in Data. URL: https://ourworldindata.org/fossil-fuels (дата звернення: 11.06.2024).

4. Impacts of Power Plants | Climate Connection. Climate Connection. URL: https://climateconnection.org.in/content/impacts-power-plants (дата звернення: 11.06.2024).

5. Smart Home – Worldwide | Statista Market Forecast. Statista. URL: https://www.statista.com/outlook/cmo/smart-home/worldwide (дата звернення: 11.06.2024).

6. Цілі сталого розвитку. Цілі сталого розвитку | Організація Об’єднаних Націй в Україні. URL: https://ukraine.un.org/uk/sdgs (дата звернення: 11.06.2024).

7. Hlybovets A. M., Mogolivskiy V. O. Analysing of Systems for Maintaining a Smart Home. *Control systems and computers*. 2019. No. 5 (283). P. 30–37. URL: https://doi.org/10.15407/csc.2019.05.030 (дана звернення: 11.06.2024).

8. Розумний будинок: переваги та недоліки. *BUDUEMO*. URL: https://buduemo.com/ua/news/smart_systems/what-is-a-smart-home.html (дата звернення: 11.06.2024).

9. Мелконова І. В., Романченко Ю. А. Підвищення енергоефективності житлових будівель. *ВІСНИК СХІДНОУКРАЇНСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ імені Володимира Даля*. 2021. № 5 (269). С. 17–19. URL: https://doi.org/10.33216/1998-7927-2021-269-5-17-19 (дата звернення: 11.06.2024).

10. Копець Г. Р. Сучасні проблеми забезпечення енергоефективності у муніципальному секторі міст України. 2008. URL: https://vlp.com.ua/files/22_29.pdf (дата звернення: 11.06.2024).

11. Бобровнікова К. Ю. Методи забезпечення енергоефективності та енергозбереження в системі розумного будинку. *Комп'ютерні системи та інформаційні технології*. 2020. № 1. С. 53–58. URL: https://elar.khmnu.edu.ua/handle/123456789/9605 (дата звернення: 11.06.2024).

12. Побоченко Л. М. «Розумне місто» («розумний будинок») та його енергетична складова: світовий досвід. *Стратегія розвитку України*. 2016. № 1. С. 141–145. URL: http://nbuv.gov.ua/UJRN/sru_2016_1_27 (дата звернення: 11.06.2024).

13. Кириленко О. М., Дмитренко Е. Д. Проблеми підвищення енергоефективності та енергозбереження України. 2014. № 18. URL: https://dspace.nau.edu.ua/bitstream/NAU/16929/1/13.pdf (дата звернення: 11.06.2024).

14. Алдошина А. А. Особливості створення «розумного міста» в житловому масиві Русанівка. *Молодий вчений*. 2019. № 11. С. 290–293. URL: https://molodyivchenyi.ua/index.php/journal/article/view/1445/1412 (дата звернення: 11.06.2024).

15. Теслюк В. М., Казарян А. Г. Вибір оптимального типу штучної нейронної мережі для автоматизованих систем «розумного будинку». *Науковий вісник НЛТУ України*. 2020. № 30. С. 90–93. URL: https://doi.org/10.36930/40300515 (дата звернення: 11.06.2024).

16. Клюванський Є. Г. Оцінка ризиків проєкту впровадження системи управління температурою повітря в приміщенні. *Радіоелектроніка та молодь у XXI столітті : матеріали 27-го міжнар. молодіж. форуму, 10–12 травня 2023 р.* 2023. Т. 6, № 27. С. 210–211. URL: https://openarchive.nure.ua/handle/document/24688 (дата звернення: 11.06.2024).

17. Воробйов Д. О. Розробка системи управління «Розумний дім» для приватного будинку. 2019. URL: https://ea.donntu.edu.ua/bitstream/123456789/33401/1/Воробйов.pdf (дата звернення: 11.06.2024).

18. Гайдай М. Ю. Роль міжнародних організацій у стимулюванні розвитку «зеленої економіки» в країнах світу. *Україна і світ: перспективи та стратегії розвитку: електронний збірник наукових праць*. 2018. Т. 1, № 6. С. 205–218. URL: http://er.nau.edu.ua/handle/NAU/37745 (дата звернення: 11.06.2024).

19. Ковальов Ю., Дроздовська Т. Розумне середовище: тенденції, дослідження, оцінювання, навчання. *Міжнародна науково-практична конференція АКТУАЛЬНІ ПРОБЛЕМИ СУЧАСНОГО ДИЗАЙНУ*. 2020. URL: https://er.knutd.edu.ua/bitstream/123456789/16141/1/APSD2020_V2_P195-201.pdf (дата звернення: 11.06.2024).

20. Visio. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/ru-ru/office/client-developer/visio/visio-home (дата звернення: 28.11.2024).

21. Опорні точки зору. *StudFiles*. URL: https://studfile.net/preview/9862899/page:2/ (дата звернення: 11.06.2024).

22. Weiss M. A. Data Structures & Algorithm Analysis in C++. Pearson, 2017. 656 p.

23. What is Use Case Diagram?. Ideal Modeling & Diagramming Tool for Agile Team Collaboration. URL: https://www.visual-paradigm.com/guide/uml-

unified-modeling-language/what-is-use-case-diagram/ (дата звернення: 18.11.2024).

24. What is Activity Diagram?. Ideal Modeling & Diagramming Tool for Agile Team Collaboration. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/ (дата звернення: 18.11.2024).

25. What is a Data Flow Diagram. *Lucidchart*. URL: https://www.lucidchart.com/pages/data-flow-diagram (дата звернення: 11.06.2024).

26. IDEF0 Functional Modeling. *Aarongilly*. URL: https://aarongilly.com/gillespedia/idef0/ (дата звернення: 11.06.2024).

27. CS Odessa. Program Structure Diagram. https://www.conceptdraw.com. URL: https://www.conceptdraw.com/How-To-Guide/program-structure-diagram-jps (дата звернення: 27.11.2024).

28. UML Class Diagram Tutorial. Ideal Modeling & Diagramming Tool for Agile Team Collaboration. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/ (дата звернення: 28.11.2024).

29. Chollet F. Deep Learning with Python, Second Edition. Manning Publications Co. LLC, 2021.

30. Keras Visualizer Library. Google Colab. URL: https://colab.research.google.com/drive/1S35LDBXLV89nAfOvOURJJsbGpC49nFGG?usp=sharing#scrollTo=2T8uYriVNetR (дата звернення: 30.11.2024).

31. Python 3.13.0 Documentation. Python 3.13.0 Documentation. URL: https://docs.python.org/3/ (дата звернення: 01.12.2024).

32. Getting started | PyCharm. PyCharm Help. URL: https://www.jetbrains.com/help/pycharm/getting-started.html (дата звернення: 01.12.2024).

33. pandas documentation – pandas 2.2.3 documentation. pandas - Python Data Analysis Library. URL: https://pandas.pydata.org/docs/ (дата звернення: 01.12.2024).

34. NumPy Documentation. NumPy. URL: https://numpy.org/doc/ (дата звернення: 01.12.2024).

35. Matplotlib documentation – Matplotlib 3.9.3 documentation. Matplotlib – Visualization with Python. URL: https://matplotlib.org/stable/index.html (дата звернення: 01.12.2024).

36. seaborn: statistical data visualization – seaborn 0.13.2 documentation. seaborn: statistical data visualization – seaborn 0.13.2 documentation. URL: https://seaborn.pydata.org/ (дата звернення: 01.12.2024).

37. Keras documentation: Model plotting utilities. Keras: Deep Learning for humans. URL: https://keras.io/api/utils/model_plotting_utils/ (дата звернення: 01.12.2024).

38. StandardScaler. scikit-learn. URL: https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.StandardScaler.html (дата звернення: 01.12.2024).

39. LabelEncoder. scikit-learn. URL: https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.LabelEncoder.html (дата звернення: 01.12.2024).

40. OneHotEncoder. scikit-learn. URL: https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.OneHotEncoder.html (дата звернення: 01.12.2024).

41. Keras documentation: Whole model saving & loading. Keras: Deep Learning for humans. URL: https://keras.io/api/models/model_saving_apis/model_saving_and_loading/ (дата звернення: 01.12.2024).

42. open Function – Python Tips 0.1 documentation. Intermediate Python – Python Tips 0.1 documentation. URL:

https://book.pythontips.com/en/latest/open_function.html (дата звернення: 01.12.2024).

43. Keras documentation: The Sequential model. Keras: Deep Learning for humans. URL: https://keras.io/guides/sequential_model/ (дата звернення: 02.12.2024).

44. Keras documentation: Model training APIs. Keras: Deep Learning for humans. URL: https://keras.io/api/models/model_training_apis/ (дата звернення: 02.12.2024).

45. math Mathematical functions. Python documentation. URL: https://docs.python.org/3/library/math.html (дата звернення: 02.12.2024).

46. Tkinter Label. URL: https://www.tutorialspoint.com/python/tk_label.htm (дата звернення: 02.12.2024).

47. Button – Tk tutorial 2020 documentation. Welcome to this Tk tutorial! – Tk tutorial 2020 documentation. URL: https://tk-tutorial.readthedocs.io/en/latest/button/button.html (дата звернення: 02.12.2024).

48. Tkinter Entry. URL: https://www.tutorialspoint.com/python/tk_entry.htm (дата звернення: 02.12.2024).

49. Treeview – Tk tutorial 2020 documentation. Welcome to this Tk tutorial! – Tk tutorial 2020 documentation. URL: https://tk-tutorial.readthedocs.io/en/latest/tree/tree.html (дата звернення: 02.12.2024).

50. Scrollbar – Tk tutorial 2020 documentation. Welcome to this Tk tutorial! – Tk tutorial 2020 documentation. URL: https://tk-tutorial.readthedocs.io/en/latest/scrollbar/scrollbar.html (дата звернення: 02.12.2024).

51. tkinter â□□ Python interface to Tcl/Tk. Python documentation. URL: https://docs.python.org/uk/3/library/tkinter.html (дата звернення: 02.12.2024).

52. Шамрай О. В. МЕТОДИЧНІ ВКАЗІВКИ до виконання індивідуальних завдань з дисципліни «Інноваційний менеджмент в індустрії програмного забезпечення» для студентів усіх форм навчання за

спеціальністю 121 – «Інженерія програмного забезпечення». Кривий Ріг : Криворізький національний університет, 2022. 42 с.

53. ТЕМА 1 ТЕОРЕТИЧНІ ОСНОВИ ІНВЕСТУВАННЯ. Головна | Elib LNTU. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/ENP%20finish/page6.html (дата звернення: 04.12.2024).

54. Економічний ефект - Бібліотека BukLib.net. Головна - Бібліотека BukLib.net. URL: https://buklib.net/books/31853/#:~:text=Ет%20=%20Рт%20–%20Вт,,здійснення%20заходів%20щодо%20реалізації%20інвестиції (дата звернення: 04.12.2024).

Додаток А

Вихідний код розробленого програмного комплексу

А.1 Генерація набору даних

А.1.1 Клас TimestampGenerator

```
import pandas as pd
import datetime


class TimestampGenerator:
    @staticmethod
    def generate_timestamps(num_days, time_interval):
        start_date = datetime.datetime.now().replace(hour=0,
minute=0, second=0, microsecond=0)
        end_date = start_date +
datetime.timedelta(days=num_days)
        date_range = pd.date_range(start=start_date,
end=end_date, freq=f'{time_interval}min')
        return date_range
```

А.1.2 Клас RoomGenerator

```
import random


class RoomGenerator:
    @staticmethod
    def generate_rooms(number_rooms):
        room_types = [
            "bedroom",
            "kitchen",
            "bathroom",
            "living room",
            "cabinet"
        ]
        rooms = [
            {
                "name": "kitchen",
                "type": "kitchen"
            },
            {
                "name": "bathroom",
                "type": "bathroom"
            }
        ]
        for number_room in range(number_rooms - 2):
```

```
            room_name = f"room {number_room + 1}"
            room_type = random.choice([
                "bedroom",
                "living room",
                "cabinet"
            ])
            rooms.append({
                "name": room_name,
                "type": room_type
            })
        return rooms
```

A.1.3 Клас OutsideTemperatureGenerator

```
import pandas as pd
import numpy as np


class OutsideTemperatureGenerator:
    @staticmethod
    def generate_outside_temperature(timestamps):
        outside_temperature = []
        for timestamp in timestamps:
            temperature = 15 + 10 * np.sin(2 * np.pi *
timestamp.timetuple().tm_yday / 365)
            temperature += np.random.normal(0, 2)
            outside_temperature.append(temperature)
        return pd.Series(outside_temperature, index=timestamps)
```

A.1.4 Клас ResidentPresence

```
import random


class ResidentPresence:
    @staticmethod
    def is_resident_present(timestamp):
        weekday = timestamp.weekday()
        hour = timestamp.hour
        if weekday < 5 and 9 <= hour < 18:
            return False
        return random.random() > 0.15

    @staticmethod
    def calculate_activity(number_residents, timestamp,
room_type):
        number_residents_in_room = 0
        hot_water_usage = 0
        for _ in range(number_residents):
            presence =
ResidentPresence.is_resident_present(timestamp)
```

```
            if presence:
                in_room = random.choices([0, 1], weights=[0.95,
0.05])[0]
                if in_room:
                    number_residents_in_room += 1
                    target_rooms = [
                        "kitchen",
                        "bathroom"
                    ]
                    if room_type in target_rooms:
                        uses_hot_water = random.choices([0, 1],
weights=[0.95, 0.05])[0]
                        if uses_hot_water:
                            hot_water_usage +=
random.uniform(1, 20)
        return number_residents_in_room, hot_water_usage
```

A.1.5 Клас RoomEnvironment

```
import random
import numpy as np


class RoomEnvironment:
    @staticmethod
    def calculate_room_temperature(number_residents_in_room):
        room_temperature = 20
        if number_residents_in_room > 0:
            room_temperature += random.uniform(0, 2)
        else:
            room_temperature += random.uniform(-2, 2)
        room_temperature += np.random.normal(0, 0.5)
        return room_temperature
```

A.1.6 Клас BoilerStatus

```
class BoilerStatus:
    @staticmethod
    def determine_status_during_peak_hours(timestamp,
hot_water_usage):
        hour = timestamp.hour
        if hot_water_usage > 0:
            return "on"
        else:
            if 6 <= hour < 9 or 18 <= hour < 22:
                return "on"
            else:
                return "off"

    @staticmethod
    def determine_status_fixed_schedule(timestamp):
```

```
        hour = timestamp.hour
        if 6 <= hour < 22:
            return "on"
        else:
            return "off"
```

А.1.7 Клас HeatingCoolingStatus

```
class HeatingCoolingStatus:
    @staticmethod
    def
determine_status_based_presence_and_temperature(room_temperatur
e, residents_in_room, desired_temperature):
        if residents_in_room > 0:
            if room_temperature < desired_temperature:
                return "heating"
            elif room_temperature > desired_temperature:
                return "cooling"
        return "off"

    @staticmethod
    def determine_status_based_temperature(room_temperature,
desired_temperature):
        if room_temperature < desired_temperature:
            return "heating"
        elif room_temperature > desired_temperature:
            return "cooling"
        return "off"
```

А.2 Аналіз набору даних і моделей

А.2.1 Клас TextDataAnalyzer

```
from io import StringIO


class TextDataAnalyzer:
    @staticmethod
    def get_dataset_info(dataset):
        buffer = StringIO()
        dataset.info(buf=buffer)
        buffer_value = buffer.getvalue()
        return buffer_value

    @staticmethod
    def get_null_counts_info(dataset):
        null_counts = dataset.isnull().sum()
        null_counts_string = null_counts.to_string()
        return null_counts_string
```

```
    @staticmethod
    def get_data_info(dataset):
        describe_dataset = dataset.describe().T
        describe_dataset_string = describe_dataset.to_string()
        return describe_dataset_string
```

A.2.2 Клас VizualDataAnalyzer

```
import matplotlib.pyplot as plt
import seaborn as sns


class VizualDataAnalyzer:
    @staticmethod
    def create_count_plot(size, data, x_column_name, title,
x_label, y_label):
        fig, ax = plt.subplots(figsize=size)
        sns.countplot(x=x_column_name, data=data, ax=ax)
        ax.set_title(title)
        ax.set_xlabel(x_label)
        ax.set_ylabel(y_label)
        ax.grid(True)
        return fig

    @staticmethod
    def create_hist_plot(size, data, target_column_name, title,
x_label, y_label):
        fig, ax = plt.subplots(figsize=size)
        sns.histplot(data[target_column_name], bins=30,
kde=True, ax=ax)
        ax.set_title(title)
        ax.set_xlabel(x_label)
        ax.set_ylabel(y_label)
        ax.grid(True)
        return fig

    @staticmethod
    def create_heatmap(size, data, title):
        fig, ax = plt.subplots(figsize=size)
        numeric_columns = data.select_dtypes(include=[
            "float64",
            "int64"]
        ).columns
        corr_matrix = data[numeric_columns].corr()
        sns.heatmap(corr_matrix, annot=True, cmap='coolwarm',
ax=ax)
        ax.set_title(title)
        return fig

    @staticmethod
    def create_plot(size, data, x_data, y_data, label, title,
x_label, y_label):
```

```
        fig, ax = plt.subplots(figsize=size)
        sns.lineplot(data=data, x=x_data, y=y_data,
label=label, marker="o", ax=ax)
        ax.set_title(title)
        ax.set_xlabel(x_label)
        ax.set_ylabel(y_label)
        ax.legend()
        ax.grid(True)
        return fig

    @staticmethod
    def create_boxplot(size, data, x_column_name,
y_column_name, title, x_label, y_label):
        fig, ax = plt.subplots(figsize=size)
        sns.boxplot(x=x_column_name, y=y_column_name,
data=data, ax=ax)
        ax.set_title(title)
        ax.set_xlabel(x_label)
        ax.set_ylabel(y_label)
        ax.grid(True)
        return fig

    @staticmethod
    def create_scatterplot(size, data, x_column_name,
y_column_name, title, x_label, y_label):
        fig, ax = plt.subplots(figsize=size)
        sns.scatterplot(x=x_column_name, y=y_column_name,
data=data, ax=ax)
        ax.set_title(title)
        ax.set_xlabel(x_label)
        ax.set_ylabel(y_label)
        ax.grid(True)
        return fig
```

A.2.3 Клас VizualModelAnalyzer

```
from keras.src.utils import plot_model
from keras_visualizer import visualizer


class VizualModelAnalyzer:
    @staticmethod
    def save_model_structure(model, path):
        plot_model(
            model,
            to_file=path,
            show_shapes=True,
            show_dtype=True,
            show_layer_names=True,
            rankdir="LR",
            expand_nested=False,
            dpi=450,
```

```
            show_layer_activations=True,
            show_trainable=True,
        )

    @staticmethod
    def save_model_architecture(model, path):
        visualizer(
            model,
            file_name=path,
            file_format='png',
            view=False
        )
```

А.3 Обробка набору даних

А.3.1 Клас DataProcessor

```
import numpy as np
import pandas as pd
from keras.src.utils import to_categorical
from sklearn.preprocessing import StandardScaler, LabelEncoder,
OneHotEncoder


class DataProcessor:
    @staticmethod
    def convert_timestamp_to_date_format(data,
target_column_name):
        timestamp_format =
pd.to_datetime(data[target_column_name])
        return timestamp_format

    @staticmethod
    def create_binary_classes_data(data, target_column_name):
        binary_value = data[target_column_name].apply(lambda x:
1 if x > 0 else 0)
        return binary_value

    @staticmethod
    def create_multy_class_data(data, target_column_name,
mapping=None):
        if mapping is None:
            mapping = {}
        unique_values = data[target_column_name].unique()
        for unique_value in unique_values:
            if unique_value not in mapping:
                new_key = unique_value
                new_value = len(mapping) + 1
                mapping[new_key] = new_value
        return mapping
```

```
    @staticmethod
    def standardize_features(data, target_column_names):
        scaler = StandardScaler()
        data = scaler.fit_transform(data[target_column_names])
        return data

    @staticmethod
    def encode_target_variable(y, n_classes):
        label_encoder = LabelEncoder()
        label_encoded = label_encoder.fit_transform(y)
        categorical_data = to_categorical(label_encoded,
num_classes=n_classes)
        return categorical_data

    @staticmethod
    def encode_features(data, target_column_names):
        encoder = OneHotEncoder(sparse_output=False)
        features_encoded =
encoder.fit_transform(data[target_column_names])
        return features_encoded

    @staticmethod
    def concatenate_features(data):
        features = np.hstack(data)
        return features

    @staticmethod
    def create_hour(data, target_column_name):
        hour = data[target_column_name].dt.hour
        return hour

    @staticmethod
    def calculate_time_delta(data, group_column_name,
target_column_name):
        time_delta =
data.groupby(group_column_name)[target_column_name].diff().dt.t
otal_seconds() / 3600.0
        return time_delta

    @staticmethod
    def resample_data_sum(data, target_column_name):
        resampled_data =
data[target_column_name].resample("D").sum().reset_index()
        return resampled_data

    @staticmethod
    def resample_data_mean(data, target_column_name):
        resampled_data =
data[target_column_name].resample("D").mean().reset_index()
        return resampled_data
```

```
    @staticmethod
    def fill_missing_with_value(data, target_column_name,
value):
        data[target_column_name] =
data[target_column_name].fillna(value)
```

A.3.2 Клас DatasetProcessor

```
import pandas as pd


class DatasetProcessor:
    @staticmethod
    def create_dataframe(data):
        dataframe = pd.DataFrame(data)
        return dataframe

    @staticmethod
    def set_index(data, index_name):
        data.set_index(index_name, inplace=True)

    @staticmethod
    def create_column(data, new_column_name, values):
        data.loc[:, new_column_name] = values

    @staticmethod
    def replace_data_in_column(data, target_column_name,
mapping):
        data.loc[:, target_column_name] =
data[target_column_name].replace(mapping)

    @staticmethod
    def create_column_by_condition(data, new_column_name,
target_column_name, mapping):
        data[new_column_name] =
data[target_column_name].map(mapping)

    @staticmethod
    def separate_dataset_by_column(data, target_column_name,
train_data_size=0.8):
        split_column =
data[target_column_name].quantile(train_data_size)
        train_data = data[data[target_column_name] <=
split_column].reset_index(drop=True)
        test_data = data[data[target_column_name] >
split_column].reset_index(drop=True)
        return train_data, test_data

    @staticmethod
    def get_columns(data, selected_column_names):
        selected_columns = data[selected_column_names]
        return selected_columns
```

```
    @staticmethod
    def rename_column(data, target_column_name,
new_column_name):
        data.rename(columns={
            target_column_name: new_column_name
        }, inplace=True)

    @staticmethod
    def concatenate_datasets(data):
        dataset = pd.concat(data, axis=1)
        return dataset
```

А.4 Файлові операції

А.4.1 Клас DatasetReader

```
import pandas as pd


class DatasetReader:
    @staticmethod
    def read_dataset(path, date_column_name="timestamp"):
        dataset = pd.read_csv(path,
parse_dates=[date_column_name])
        return dataset
```

А.4.2 Клас DatasetSaver

```
class DatasetSaver:
    @staticmethod
    def save_dataset(data, path):
        data.to_csv(path, index=False)
```

А.4.3 Клас ModelLoader

```
from keras.src.saving import load_model


class ModelLoader:
    @staticmethod
    def load_model(path):
        model = load_model(path)
        return model
```

А.4.4 Клас ModelSaver

```
class ModelSaver:
    @staticmethod
```

```
    def save_model(model, path):
        model.save(path)
```

А.4.5 Клас PlotSaver

```
from matplotlib import pyplot as plt


class PlotSaver:
    @staticmethod
    def save_plot(fig, path):
        fig.savefig(path, bbox_inches="tight")
        plt.close(fig)
```

А.4.6 Клас StringSaver

```
class StringSaver:
    @staticmethod
    def save_string(data_string, path, mode):
        with open(path, mode) as file:
            file.write(data_string)
```

А.5 Побудова моделей штучних нейронних мереж

А.5.1 Клас BoilerStatusModel

```
from keras import Sequential, Input
from keras.src.layers import Dense, Dropout
from keras.src.regularizers import regularizers
from modeling.neural_network_model import NeuralNetworkModel


class BoilerStatusModel(NeuralNetworkModel):
    def __init__(self, x_train, y_train, x_test, y_test,
n_epochs):
        self.x_train = x_train
        self.y_train = y_train
        self.x_test = x_test
        self.y_test = y_test
        self.n_epochs = n_epochs
        self.model = None

    def build_model(self):
        self.model = Sequential()
        self.model.add(Input(shape=(self.x_train.shape[1],)))
        self.model.add(Dense(64, activation="relu",
kernel_regularizer=regularizers.L2(0.01)))
        self.model.add(Dropout(0.4))
```

```
        self.model.add(Dense(32, activation="relu",
kernel_regularizer=regularizers.L2(0.01)))
        self.model.add(Dropout(0.4))
        self.model.add(Dense(1, activation="sigmoid"))

    def compile_model(self):
        self.model.compile(loss="binary_crossentropy",
optimizer="adam", metrics=["accuracy"])

    def fit_model(self):
        history = self.model.fit(self.x_train, self.y_train,
epochs=self.n_epochs, batch_size=32)
        return history

    def evaluate_model(self):
        loss, accuracy = self.model.evaluate(self.x_test,
self.y_test)
        return loss, accuracy

    def get_model(self):
        return self.model
```

A.5.2 Клас HeatingCoolingStatusModel

```
from keras import Sequential, Input
from keras.src.layers import Dense, Dropout
from keras.src.regularizers import regularizers
from modeling.neural_network_model import NeuralNetworkModel


class HeatingCoolingStatusModel(NeuralNetworkModel):
    def __init__(self, x_train, y_train, x_test, y_test,
n_epochs):
        self.x_train = x_train
        self.y_train = y_train
        self.x_test = x_test
        self.y_test = y_test
        self.n_epochs = n_epochs
        self.model = None

    def build_model(self):
        self.model = Sequential()
        self.model.add(Input(shape=(self.x_train.shape[1],)))
        self.model.add(Dense(64, activation="relu",
kernel_regularizer=regularizers.L2(0.01)))
        self.model.add(Dropout(0.4))
        self.model.add(Dense(32, activation="relu",
kernel_regularizer=regularizers.L2(0.01)))
        self.model.add(Dropout(0.4))
        self.model.add(Dense(3, activation="softmax"))

    def compile_model(self):
```

```
        self.model.compile(loss="categorical_crossentropy",
optimizer="adam", metrics=["accuracy"])

    def fit_model(self):
        history = self.model.fit(self.x_train, self.y_train,
epochs=self.n_epochs, batch_size=32)
        return history

    def evaluate_model(self):
        loss, accuracy = self.model.evaluate(self.x_test,
self.y_test)
        return loss, accuracy

    def get_model(self):
        return self.model
```

А.6 Математичні операції

А.6.1 Клас EnergyConsumptionCalculator

```
import numpy as np
from mathematical_operations.MathUtils import MathUtils


class EnergyConsumptionCalculator:
    @staticmethod
    def calculate_energy_consumption(data, column_names,
energy_consumption):
        boiler_condition =
(data[column_names["boiler_status_column_name"]] == "on")
        heating_condition =
(data[column_names["heating_cooling_status_column_name"]] ==
"heating")
        cooling_condition =
(data[column_names["heating_cooling_status_column_name"]] ==
'cooling')
        data[column_names["result"]] = np.select(
            [
                boiler_condition & heating_condition,
                boiler_condition & cooling_condition,
                boiler_condition,
                heating_condition,
                cooling_condition
            ],
            [
                MathUtils.round_number(data['time_delta'] *
(energy_consumption["boiler_energy_consumption"] +
energy_consumption["heating_energy_consumption"]), 2),
```

```
                MathUtils.round_number(data['time_delta'] *
(energy_consumption["boiler_energy_consumption"] +
energy_consumption["cooling_energy_consumption"]), 2),
                MathUtils.round_number(data['time_delta'] *
energy_consumption["boiler_energy_consumption"], 2),
                MathUtils.round_number(data['time_delta'] *
energy_consumption["heating_energy_consumption"], 2),
                MathUtils.round_number(data['time_delta'] *
energy_consumption["cooling_energy_consumption"], 2)
            ],
            default=0
        )
        return data
```

A.6.2 Клас EconomicCalculator

```
class EconomicCalculator:
    @staticmethod
    def calculate_sum(data, target_column_name):
        sum_value = data[target_column_name].sum()
        return sum_value

    @staticmethod
    def calculate_cost(consumption, tariff):
        cost_value = consumption * tariff
        return cost_value

    @staticmethod
    def calculate_benefit(previous_value, current_value):
        benefit_value = previous_value - current_value
        return benefit_value
```

A.6.3 Клас MathUtils

```
import numpy as np


class MathUtils:
    @staticmethod
    def threshold_function(data, threshold):
        labels = np.where(data >= threshold, 1, 0)
        return labels

    @staticmethod
    def get_indices_max_elements(data, n_axis):
        index_max_element = np.argmax(data, axis=n_axis)
        return index_max_element

    @staticmethod
    def get_number_classes(data):
        unique_values = np.unique(data)
```

```
        number_classes = len(unique_values)
        return number_classes

    @staticmethod
    def round_number(number, n_digits):
        rounded_number = round(number, ndigits=n_digits)
        return rounded_number
```

Додаток Б

Результати аналізу набору даних

На рисунку Б.1 показаний результат аналізу середньодобового використання гарячої води. На осі Y визначена кількість використаної гарячої води в літрах, на осі X визначені дати.

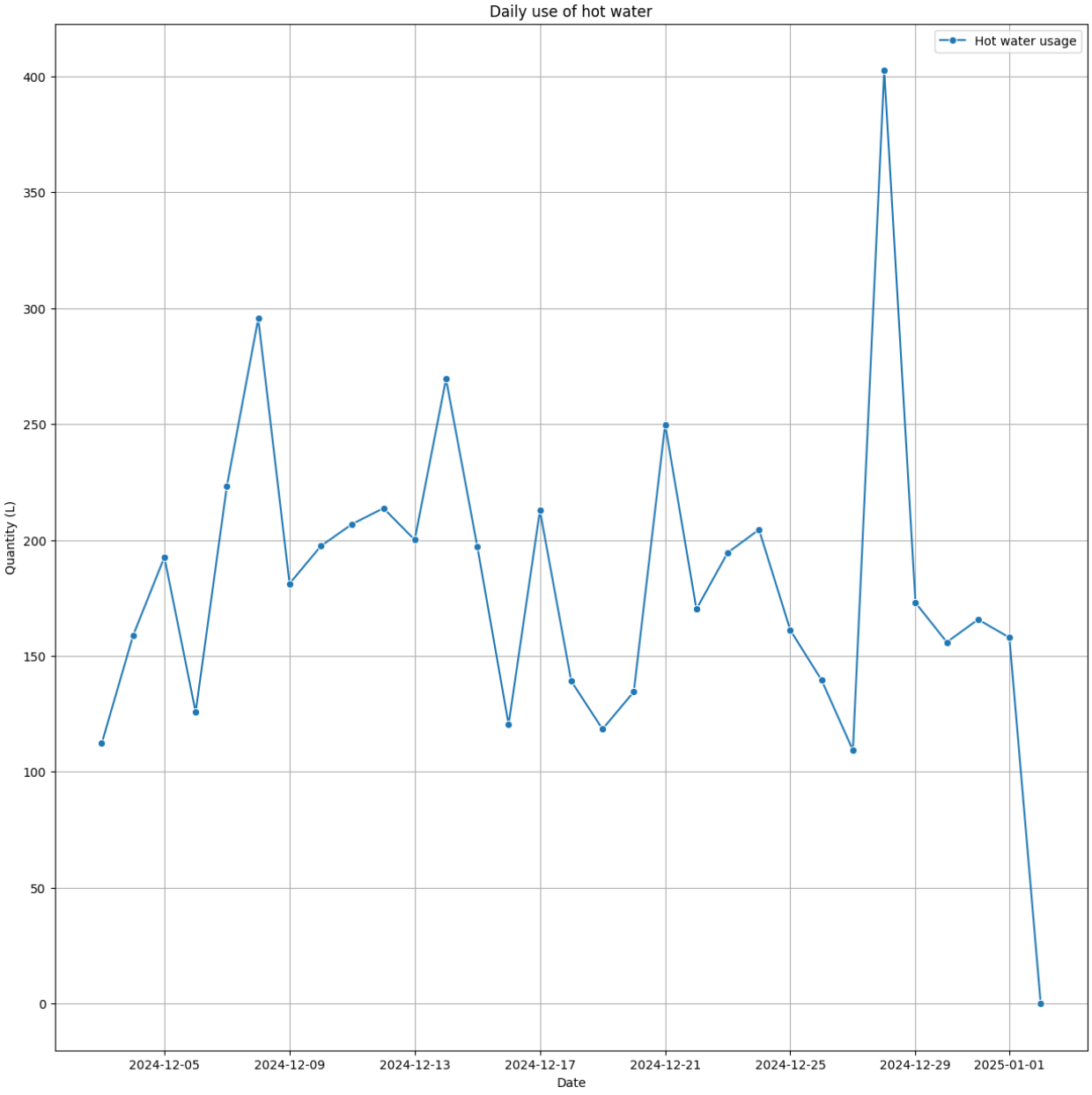


Рисунок Б.1 – Аналіз середньодобового використання гарячої води

На рисунку Б.2 показаний результат аналізу середньодобової температури довкілля. На осі Y визначена температура в градусах Цельсію, на осі X визначені дати.

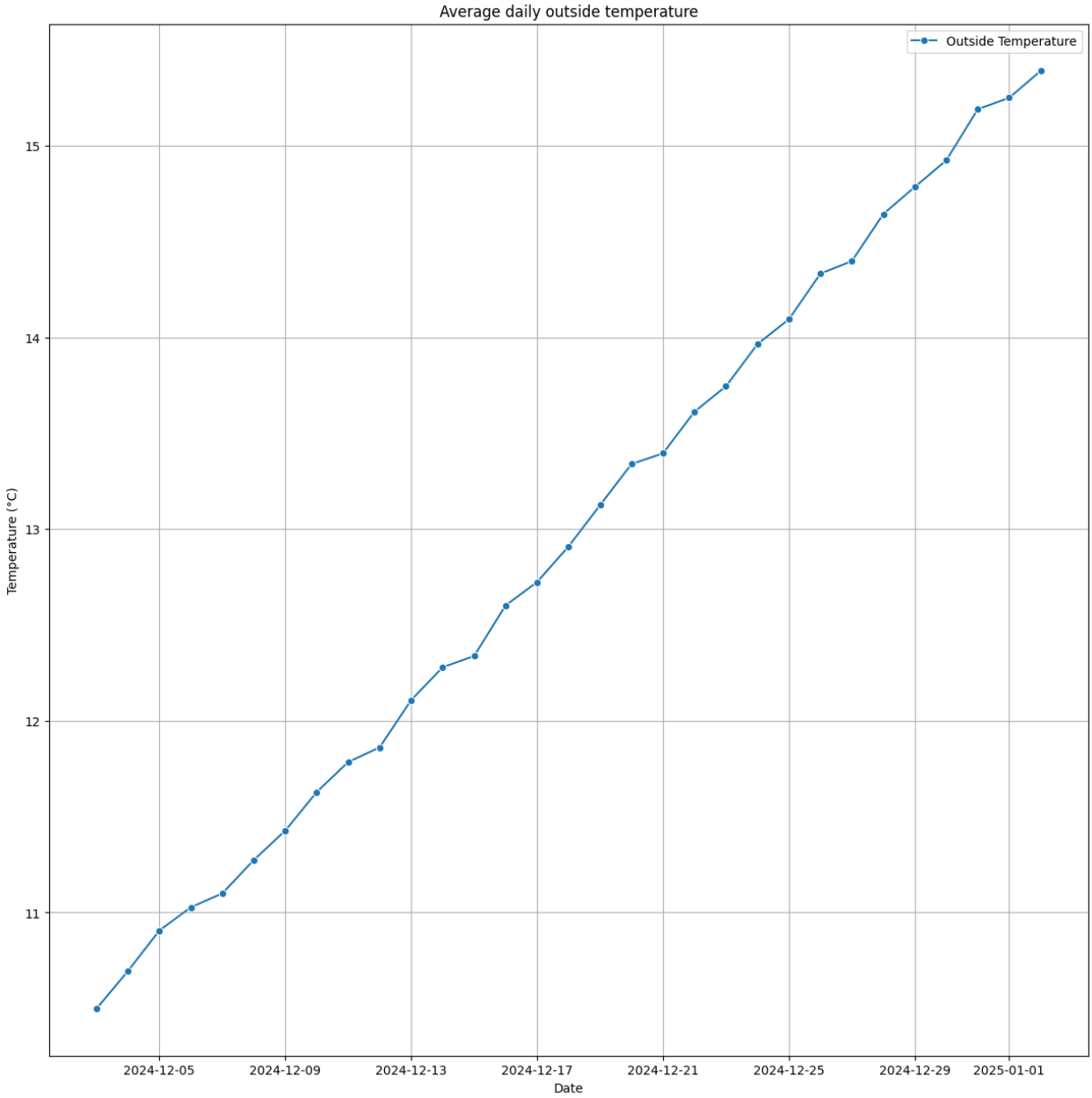


Рисунок Б.2 – Аналіз середньодобової температури довкілля

На рисунку Б.3 показаний результат аналізу середньодобової температури в приміщеннях. На осі Y визначена температура в градусах Цельсію, на осі X визначені дати.

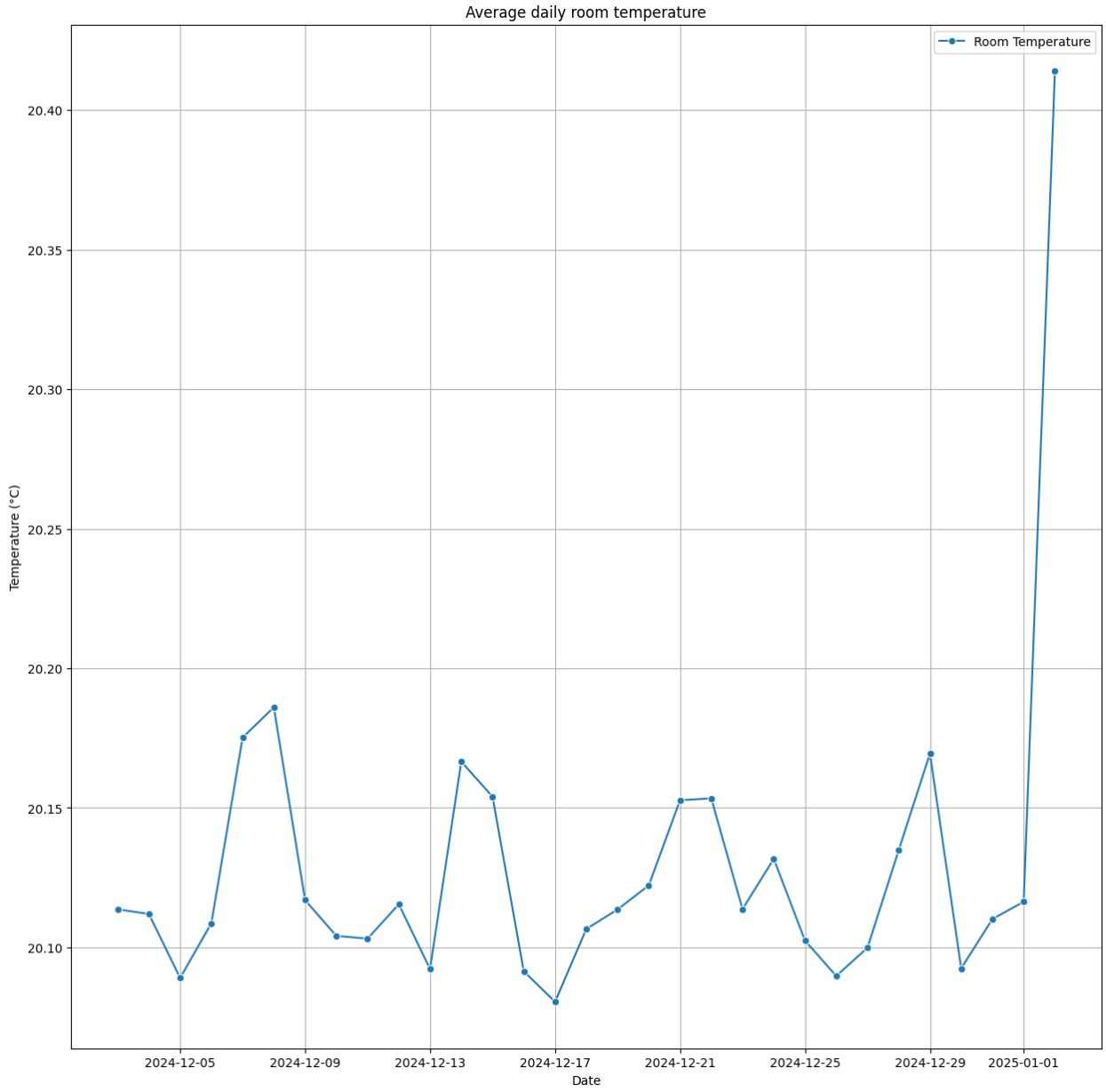


Рисунок Б.3 – Аналіз середньодобової температури в приміщеннях

На рисунку Б.4 показаний результат аналізу розподілу режиму роботи системи нагрівання води за адаптивним графіком. На осі Y визначена кількість випадків режиму роботи системи нагрівання води, на осі X визначені режими роботи системи нагрівання води.

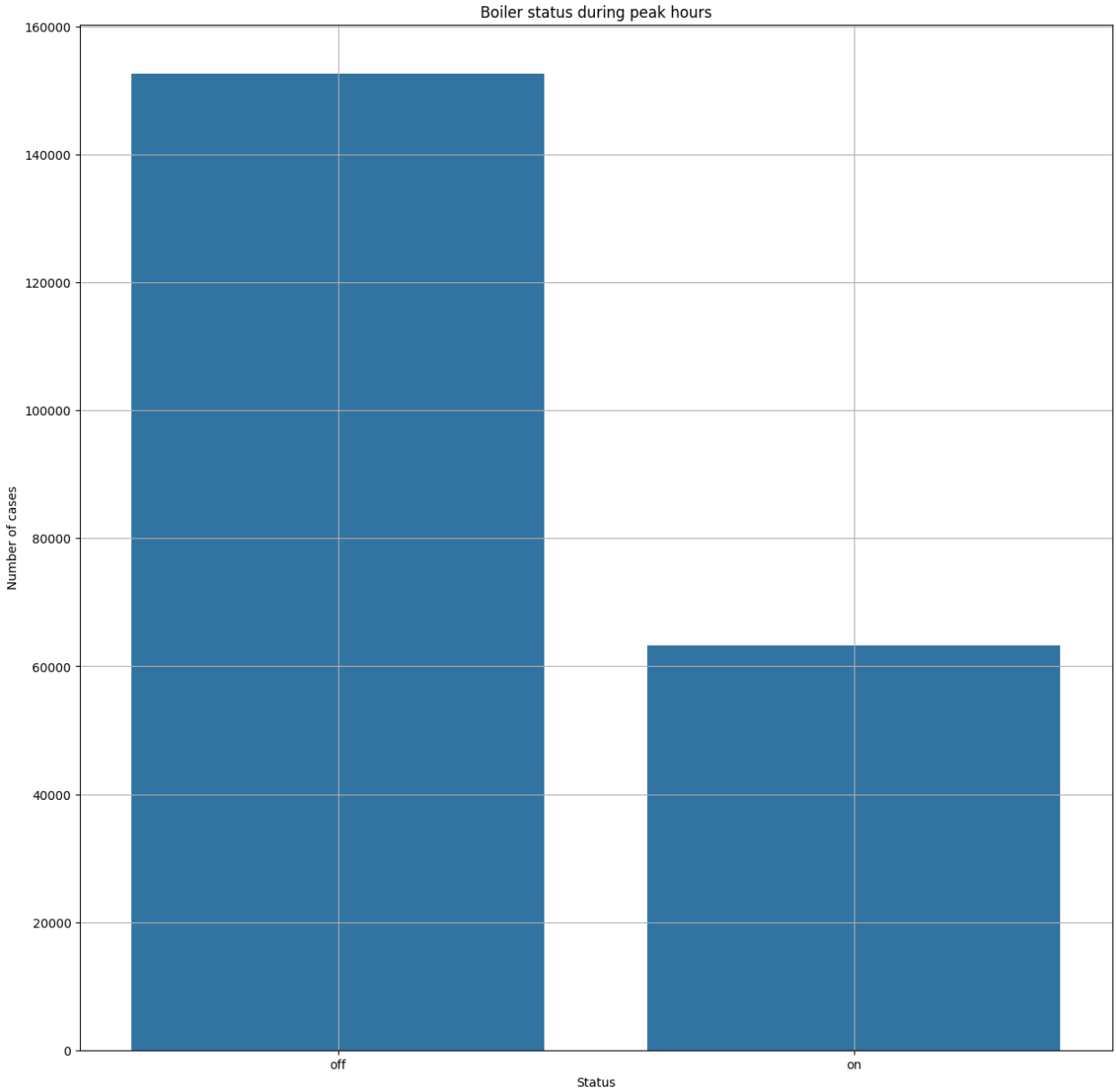


Рисунок Б.4 – Аналіз розподілу режимів роботи системи нагрівання води за адаптивним графіком

На рисунку Б.5 показаний результат аналізу розподілу режиму роботи системи нагрівання води за звичайним графіком. На осі Y визначена кількість випадків режиму роботи системи нагрівання води, на осі X визначені режими роботи системи нагрівання води.

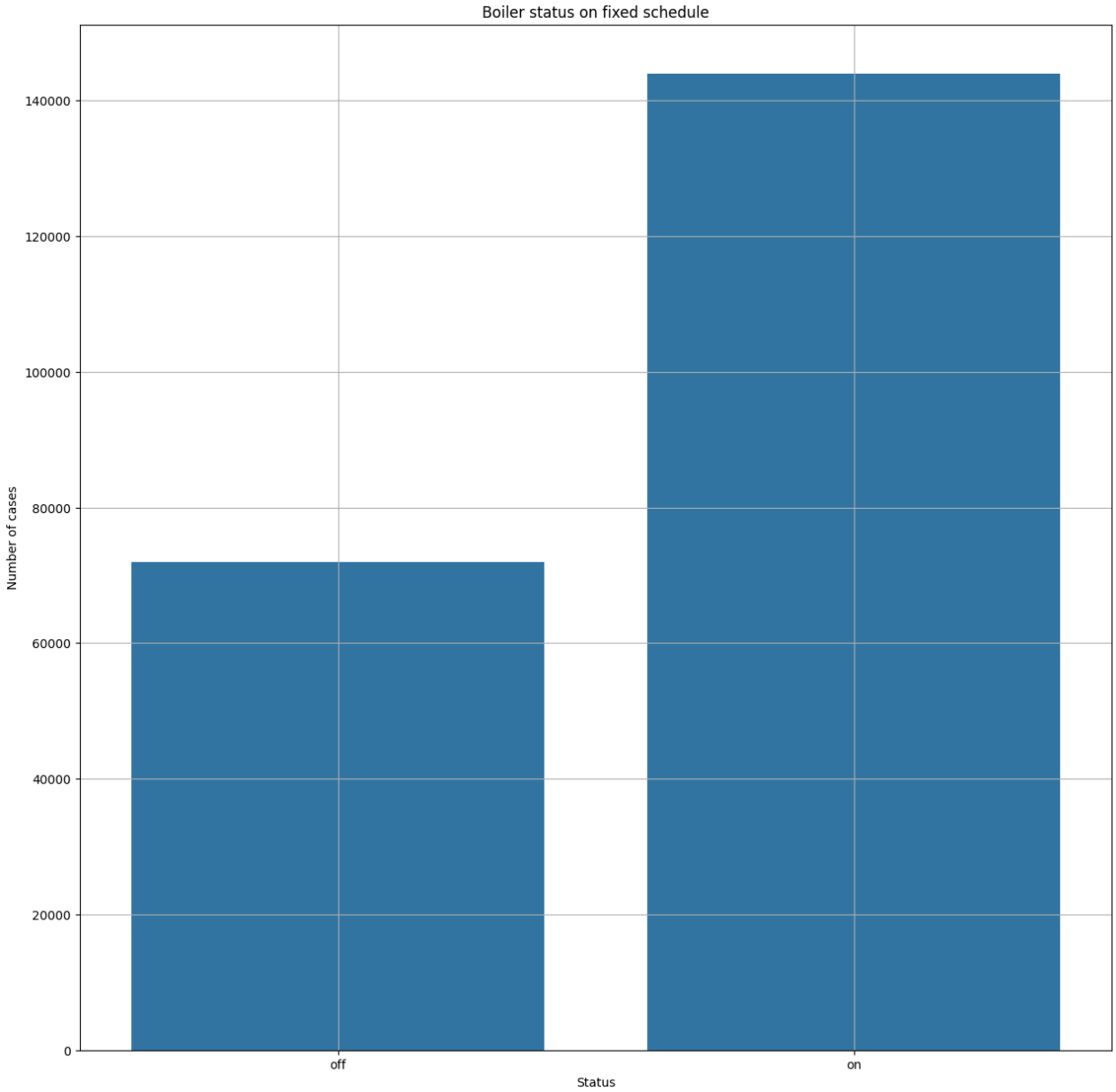


Рисунок Б.5 – Аналіз розподілу режимів роботи системи нагрівання води за звичайним графіком

На рисунку Б.6 показані результати аналізу розподілу температури довкілля. На осі Y показана кількість випадків певної температури, на осі X показна температура в градусах Цельсію.

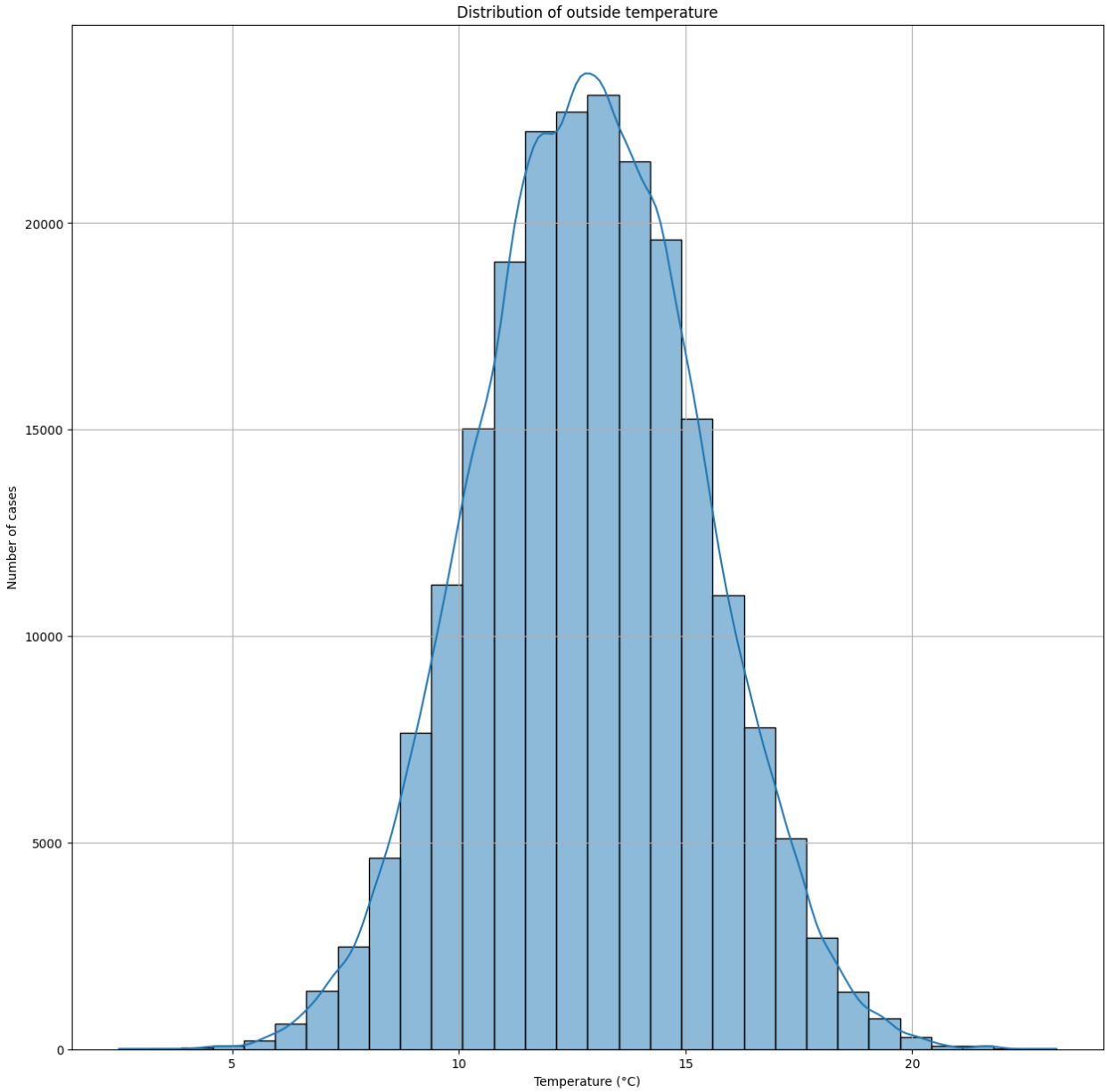


Рисунок Б.6 – Аналіз розподіл температури довкілля

На рисунку Б.7 показані результати аналізу розподілу температури в приміщеннях. На осі Y показана кількість випадків певної температури, на осі X показна температура в градусах Цельсію.

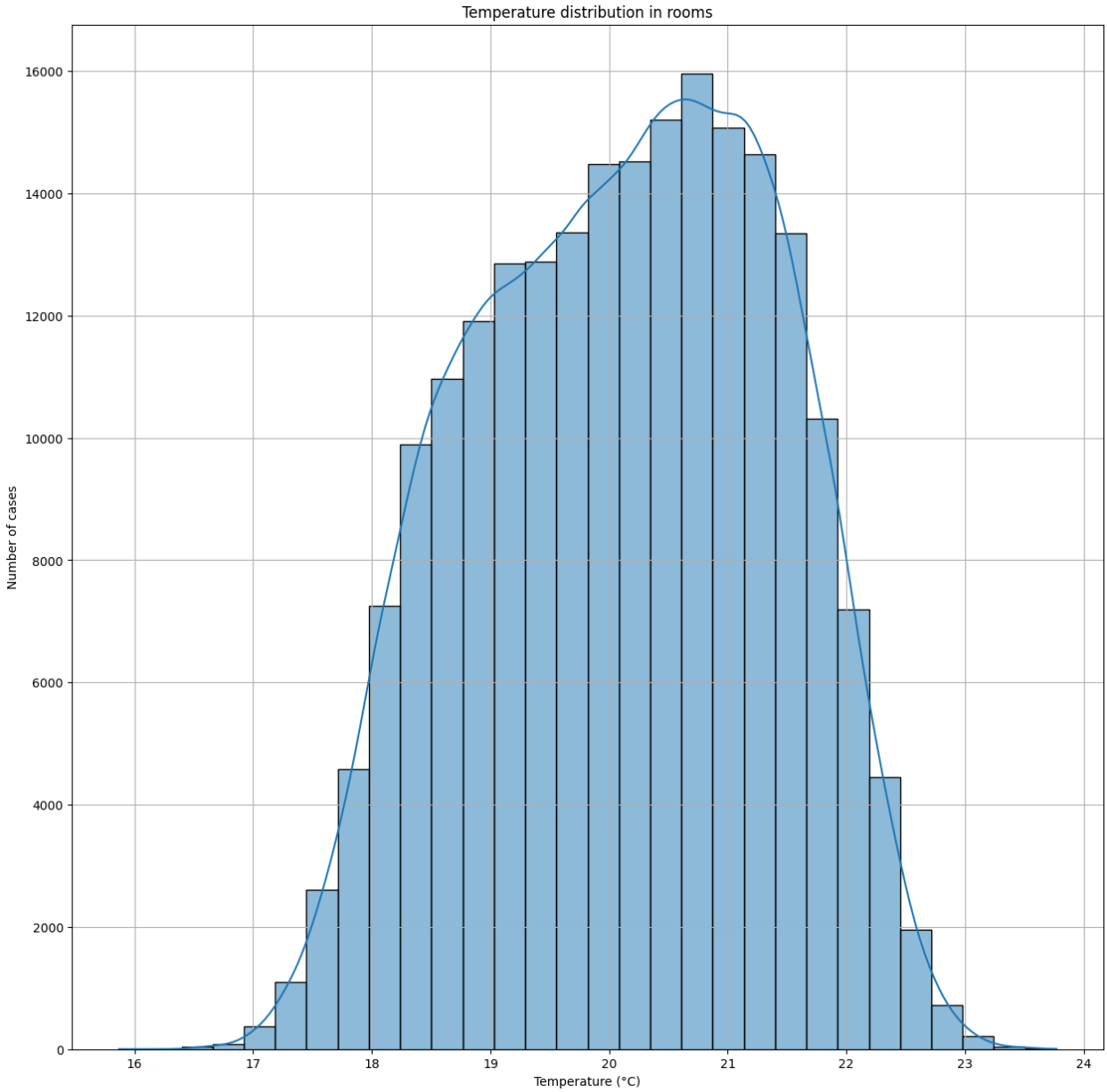


Рисунок Б.7 – Аналіз розподілу температури в приміщеннях

На рисунку Б.8 показані результати аналізу залежності температури в приміщенні від температури довкілля. На осі Y показана температура в приміщенні в градусах Цельсію, на осі X показна температура довкілля в градусах Цельсію.

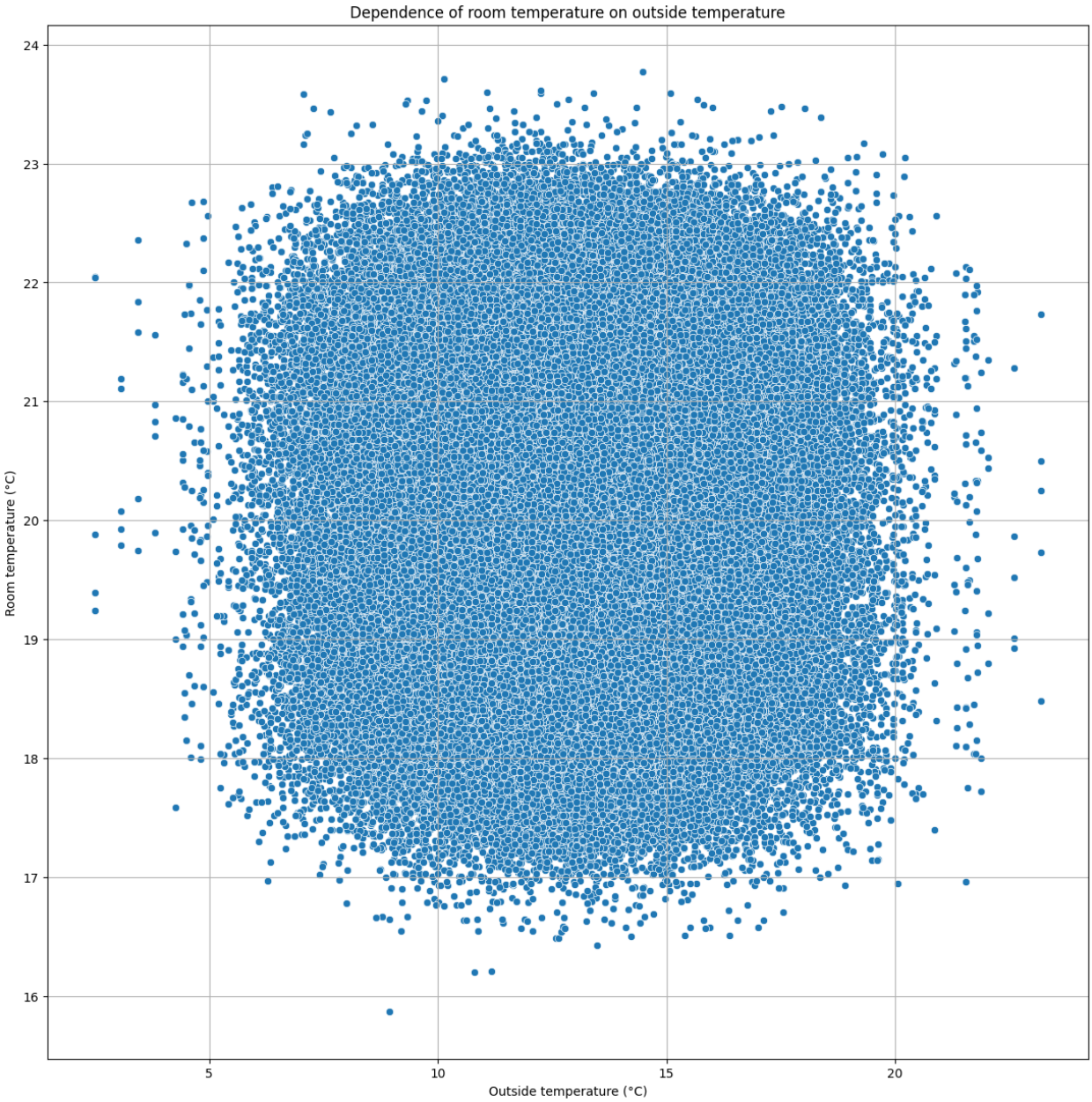


Рисунок Б.8 – Аналіз залежності температури в приміщенні від температури довкілля

На рисунку Б.9 показані результати аналізу розподілу кількості мешканців На осі Y показана кількість випадків, на осі X показна кількість мешканців.

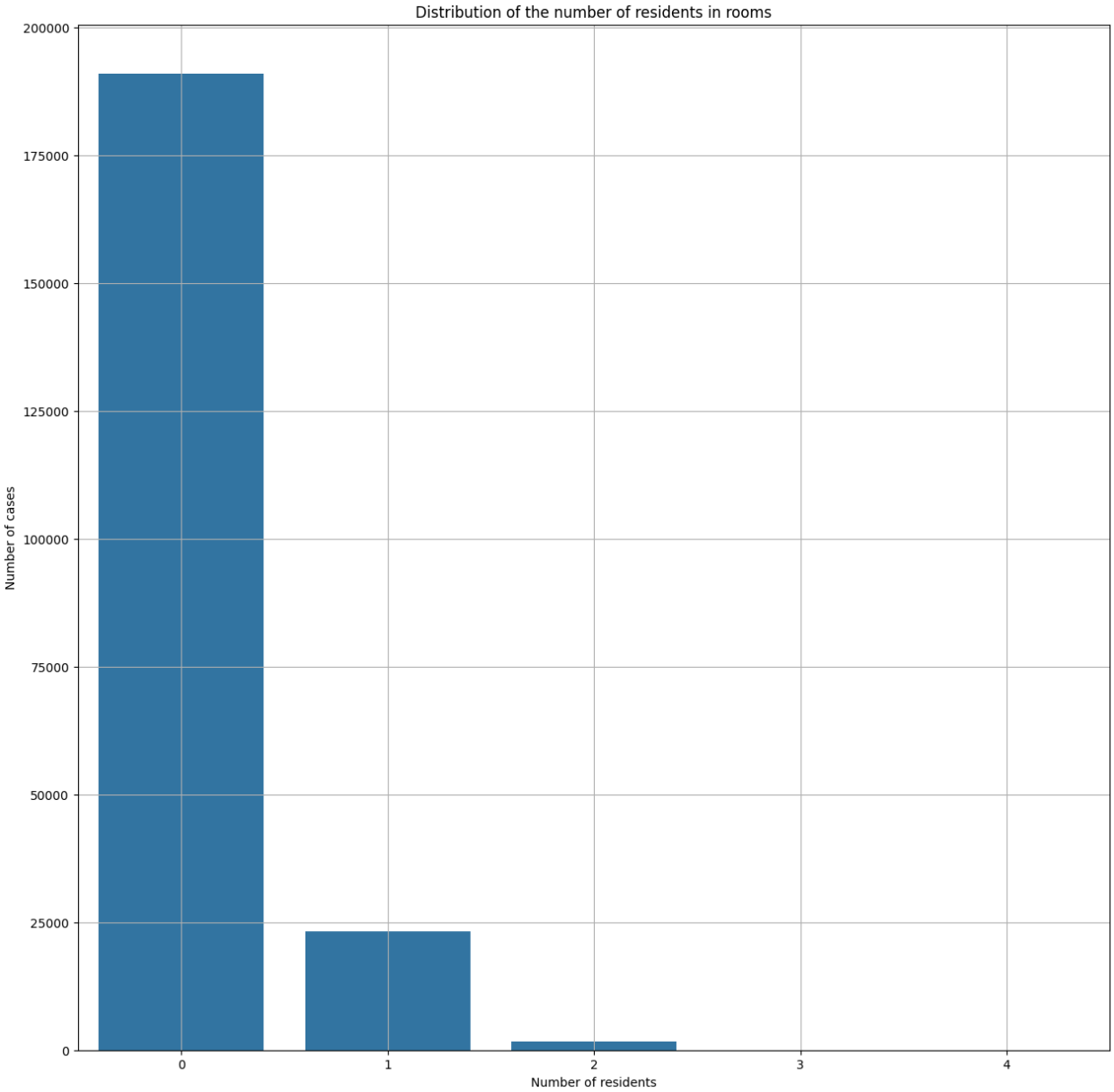


Рисунок Б.9 – Аналіз розподілу кількості мешканців

На рисунку Б.10 показаний результат аналізу розподілу режимів роботи системи опалення та системи кондиціонування за адаптивним графіком. На осі Y визначена кількість випадків режиму роботи системи опалення та системи кондиціонування, на осі X визначені режими роботи системи опалення та системи кондиціонування.

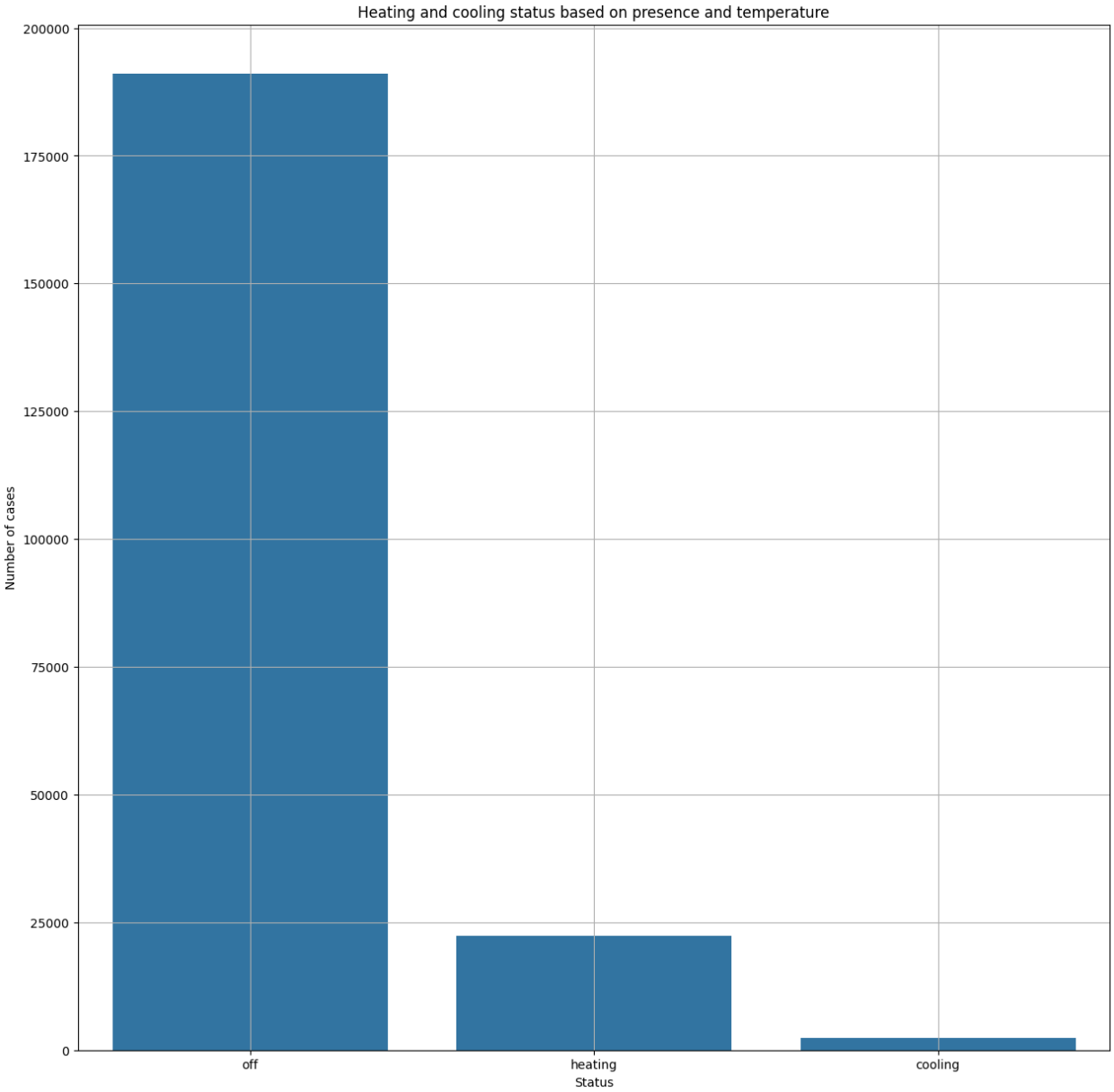


Рисунок Б.10 – Аналіз розподілу режимів роботи системи опалення та системи кондиціонування за адаптивним графіком

На рисунку Б.11 показаний результат аналізу розподілу режимів роботи системи опалення та системи кондиціонування за звичайним графіком. На осі Y визначена кількість випадків режиму роботи системи опалення та системи кондиціонування, на осі X визначені режими роботи системи опалення та системи кондиціонування.

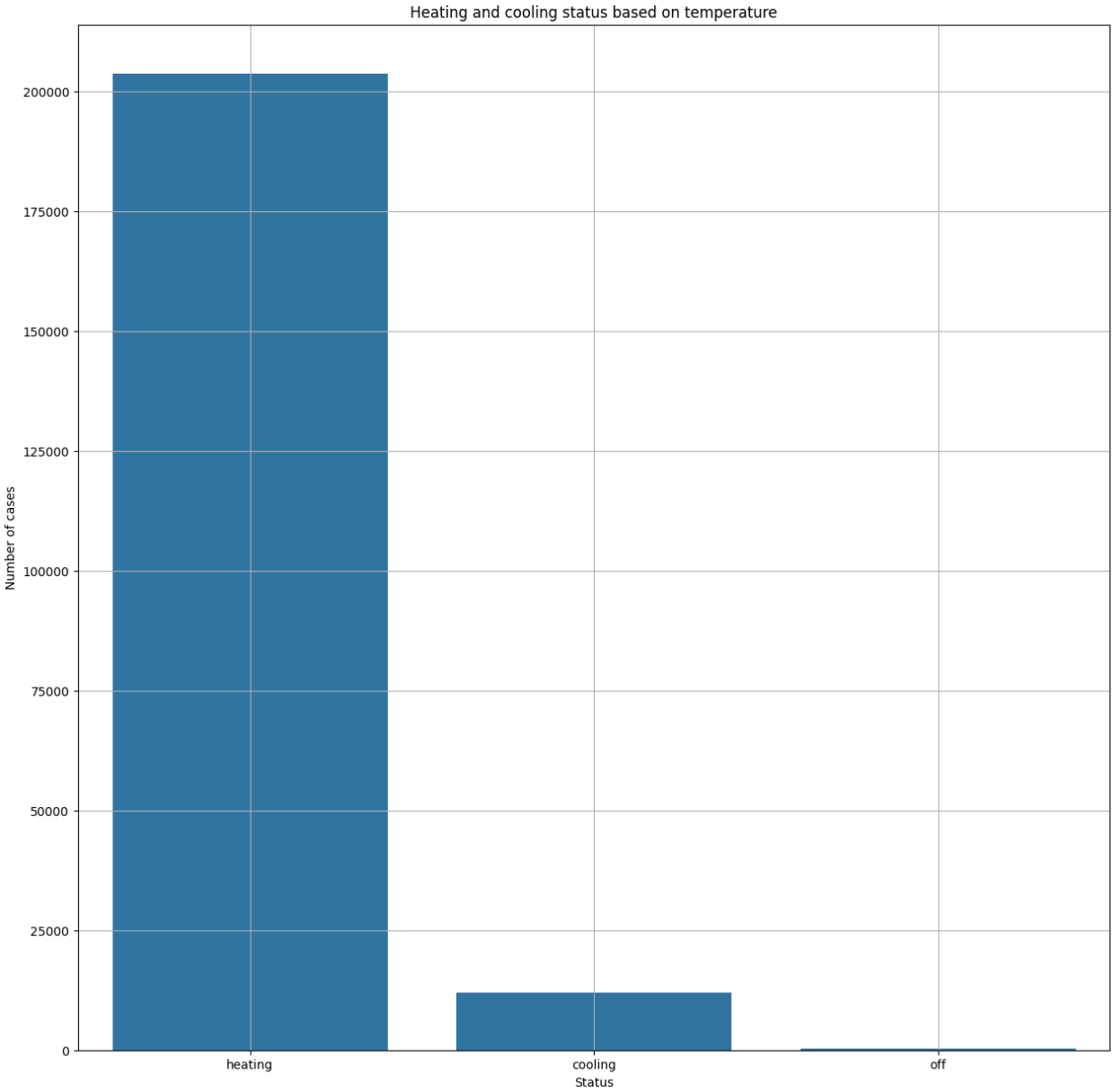


Рисунок Б.11 – Аналіз розподілу режимів роботи системи опалення та системи кондиціонування за звичайним графіком

На рисунку Б.12 показані результати аналізу залежності використання гарячої води від кількості мешканців. На осі Y показана кількість використаної гарячої води в літрах, на осі X показна кількість мешканців у приміщенні.

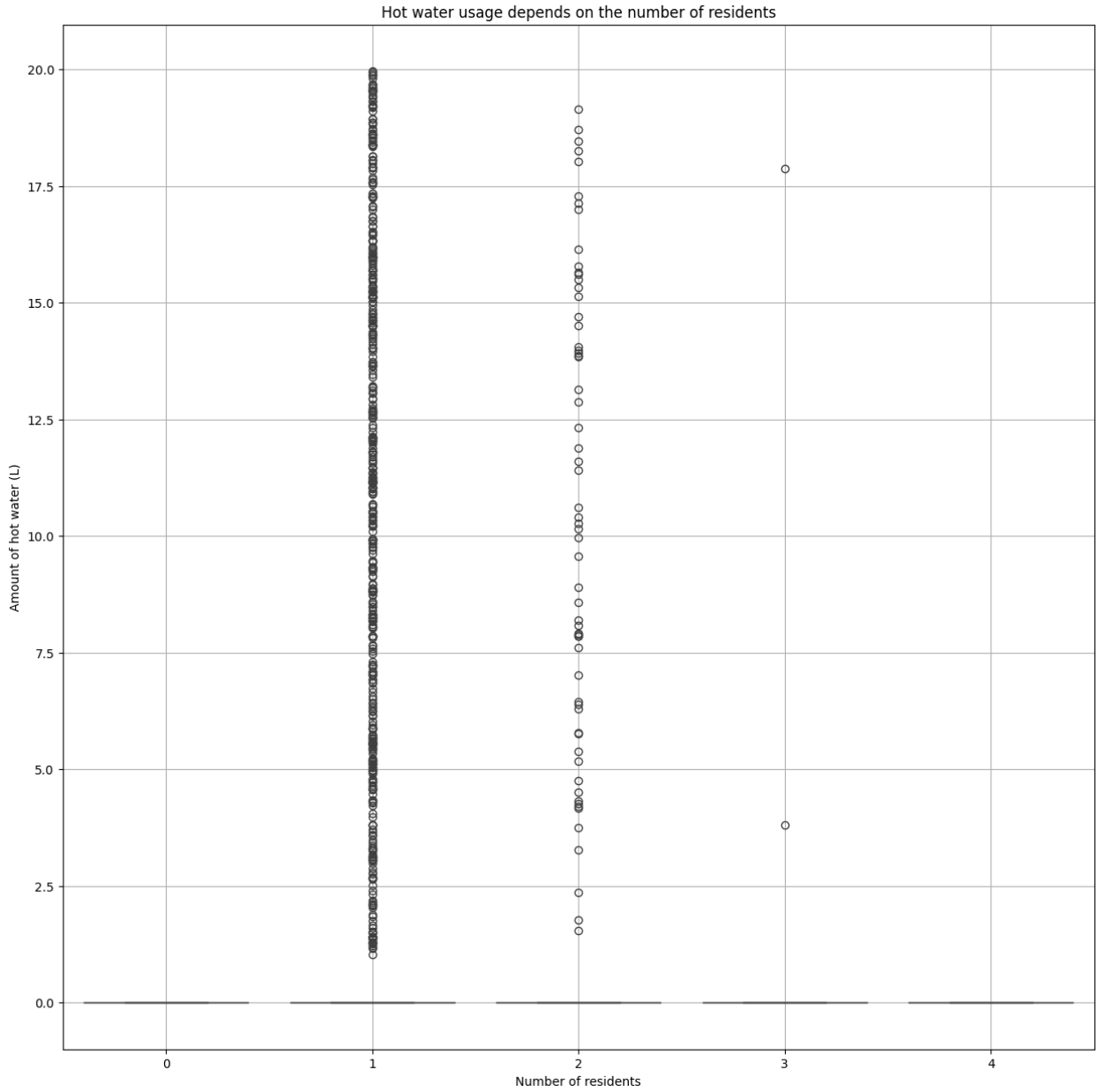


Рисунок Б.12 – Аналіз залежності використаної гарячої води від кількості мешканців у приміщенні

На рисунку Б.13 показані результати аналізу кореляції між числовими значеннями у наборі даних.

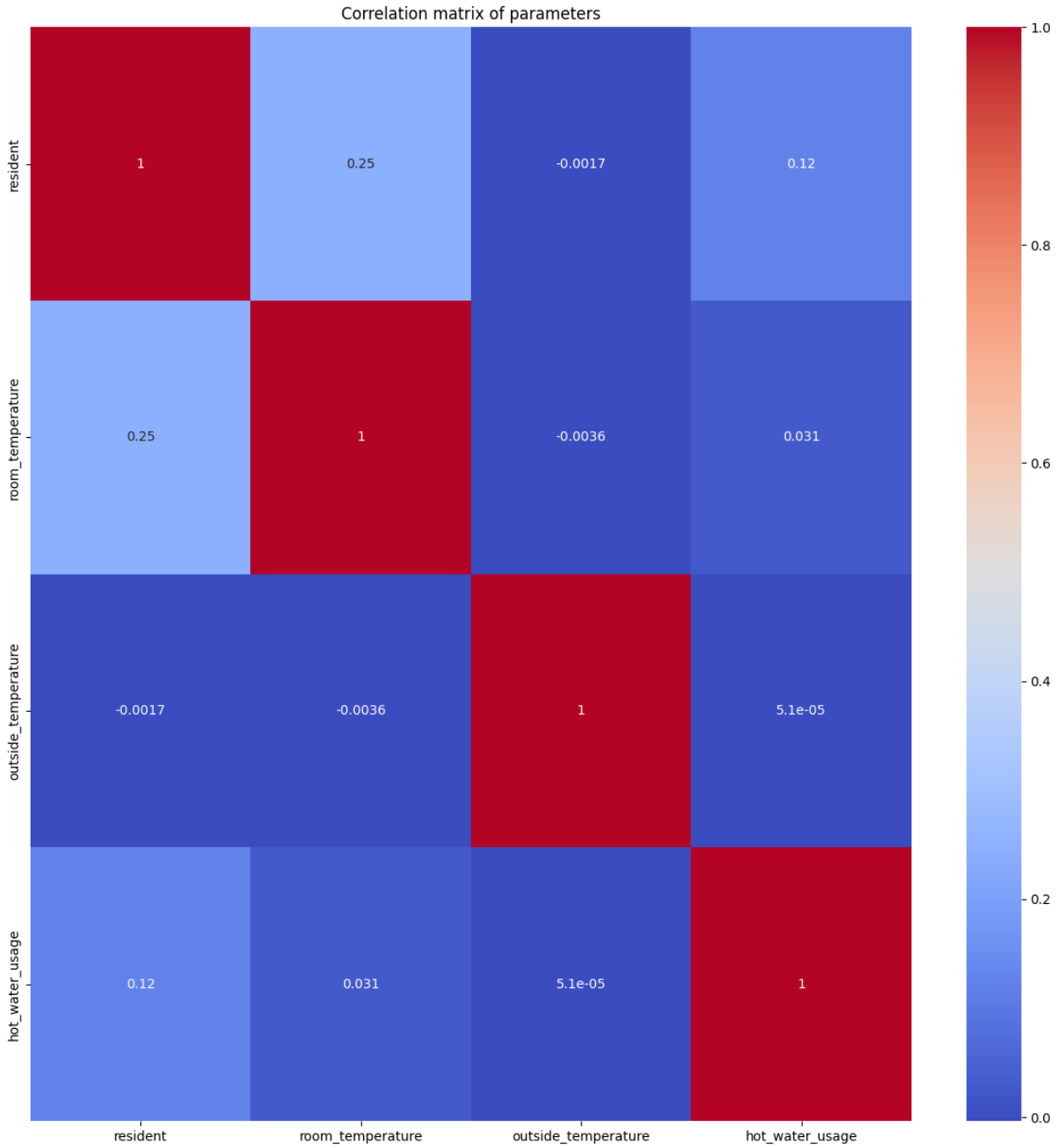


Рисунок Б.13 – Аналіз кореляції між числовими значеннями у наборі даних

Додаток В

Результати аналізу процесу навчання моделей

На рисунку В.1 показана залежність точності моделі від кількості епох (на прикладі навчання моделі для прогнозування режиму роботи системи нагрівання води). На осі Y показана точність моделі, на осі X показані епохи.

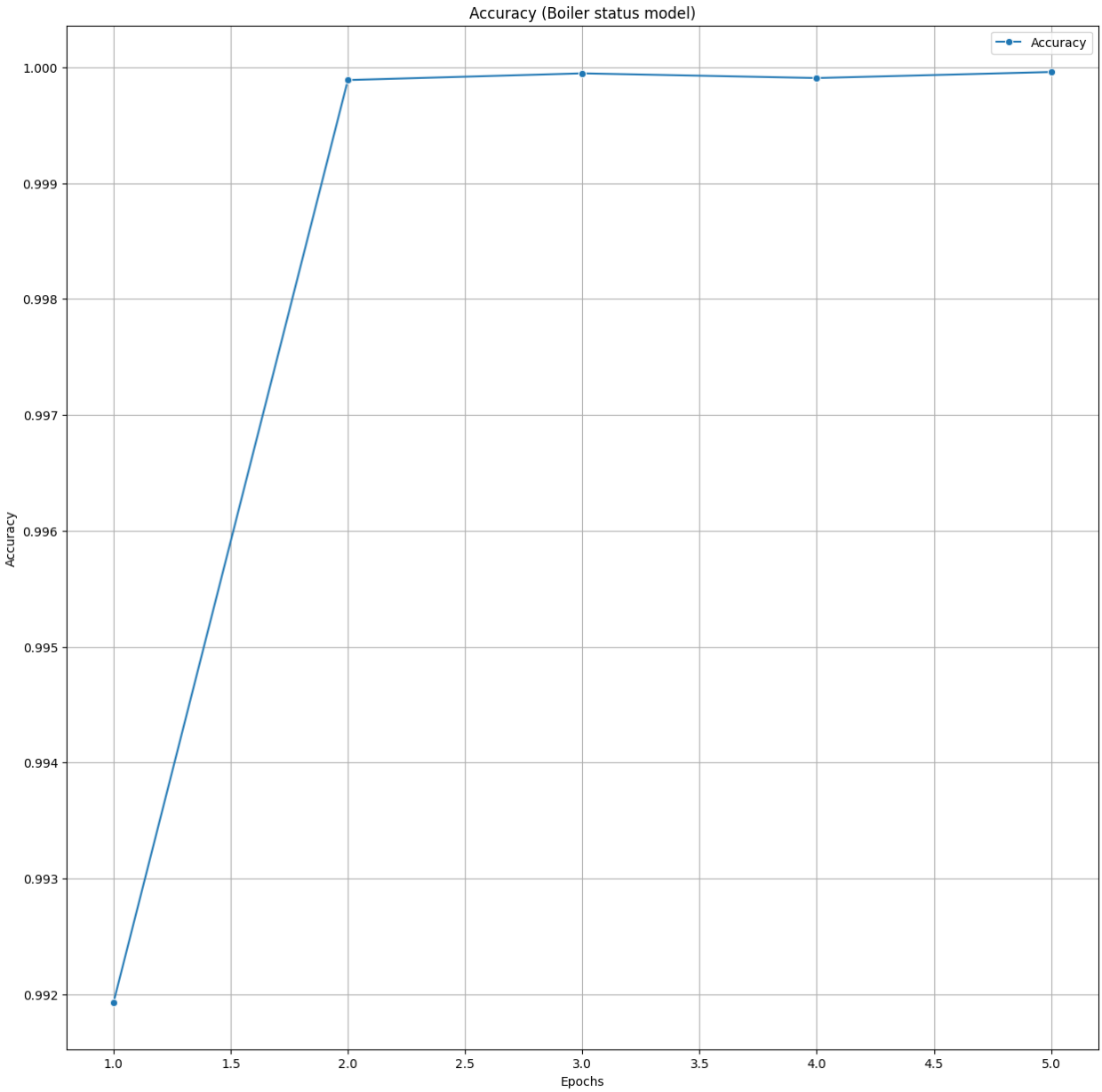


Рисунок В.1 – Точність моделі для прогнозування режиму роботи системи нагрівання води

На рисунку В.2 показана залежність втрат моделі від кількості епох (на прикладі навчання моделі для прогнозування режиму роботи системи нагрівання води). На осі Y показані втрати моделі, на осі X показані епохи.

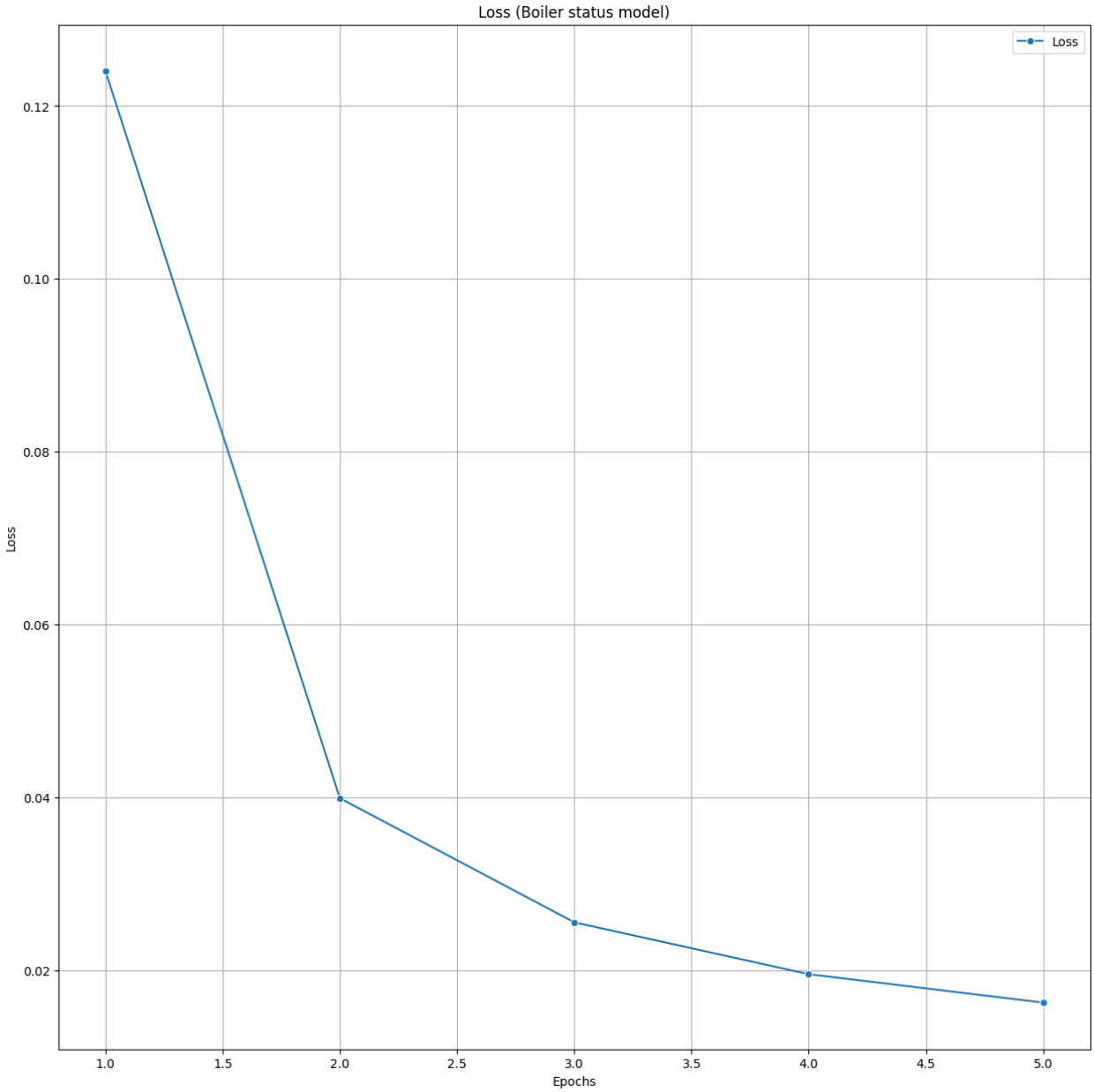


Рисунок В.2 – Втрат моделі для прогнозування режиму роботи системи нагрівання води

На рисунку В.3 показана залежність точності моделі від кількості епох (на прикладі навчання моделі для прогнозування режимів роботи системи

опалення та системи кондиціонування). На осі Y показана точність моделі, на осі X показані епохи.

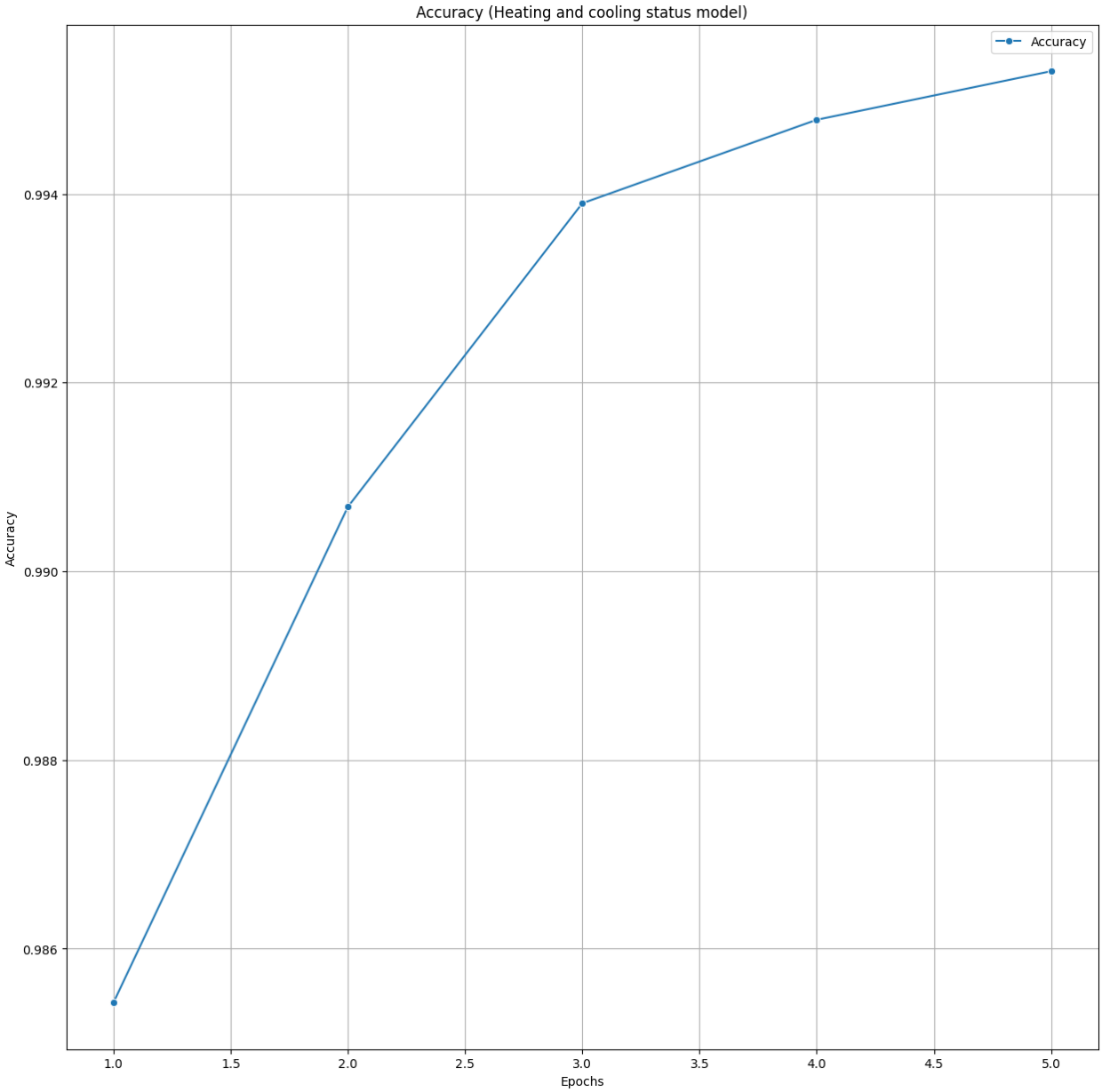


Рисунок В.3 – Точність моделі для прогнозування режимів роботи системи опалення та системи кондиціонування

На рисунку В.4 показана залежність втрат моделі від кількості епох (на прикладі навчання моделі для прогнозування режимів роботи системи опалення та системи кондиціонування). На осі Y показані втрати моделі, на осі X показані епохи.

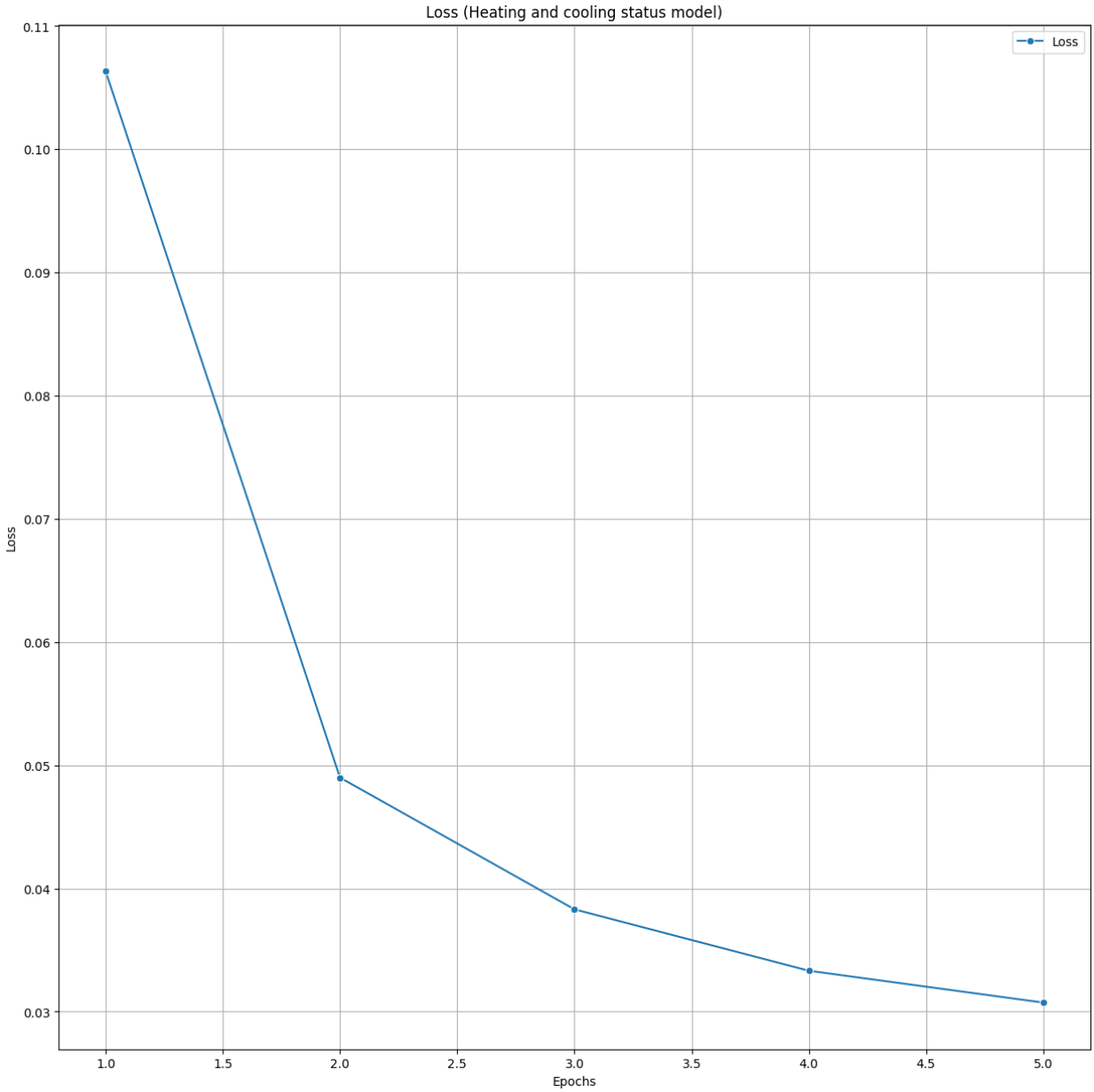


Рисунок В.4 – Втрати моделі для прогнозування режимів роботи системи опалення та системи кондиціонування

Додаток Г

Структура моделей штучних нейронних мереж

На рисунку Г.1 показана структура моделі штучної нейронної мережі для здійснення прогнозування режиму роботи системи нагрівання води.

Рисунок Г.1 – Структура моделі для прогнозування режиму роботи системи нагрівання води

На рисунку Г.2 показана структура моделі штучної нейронної мережі для здійснення прогнозування режимів роботи системи опалення та системи кондиціонування.

Рисунок Г.2 – Структура моделі для прогнозування режимів роботи системи опалення та системи кондиціонування