Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

«Затверджую»
в.о. завідуючого кафедри
комп'ютерних систем та робототехніки
_____ к. ф.-м. н., доцент Максим Хруслов
«___» червня 2025 р.

# Пояснювальна записка

**до кваліфікаційної роботи**
**бакалавра**

на тему: «**Комп'ютерна модель системи управління мережею міського освітлення»**

Спеціальність 123 – Комп'ютерна інженерія.
Галузь знань: 12 – Інформаційні технології.
Освітня програма «Комп'ютерна інженерія».

**Захищено на засіданні**
**Екзаменаційної комісії № 44**
протокол № __ від __.06.2025 р.
Оцінка _____ / _____
**Голова Екзаменаційної комісії**
_____ **ЧУГАЙ А.М.**

**Виконав:**
Студент групи КІ– 41
**ІСАЧЕНКО Владислав Русланович**

**Керівник:** к.е.н, доцент
закладу вищої освіти кафедри
комп'ютерних систем та робототехніки
**ЧУБ Ольга Ігорівна**

**Рецензент:** професор кафедри
теоретичної та прикладної інформатики
Харківського національного
університету імені В.Н. Каразіна, д.т.н.,
професор
**ФРОЛОВ В'ячеслав Вікторович**

Харків – 2025

# АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається з трьох основних розділів, висновків, списку використаних джерел та двох додатків. Обсяг роботи становить 74 сторінки, з яких 50 сторінки є основною частиною, що включає 25 рисунків, 1 таблицю та 18 найменувань у списку використаних джерел та чотири додатки.

***Мета кваліфікаційної роботи*** полягає у створенні, на основі методів машинного навчання, моделі для покращення та збільшення гнучкості графіків міського комунального освітлення.

***Об'єкт дослідження*** – процес функціонування системи міського освітлення в умовах зовнішніх факторів, що змінюються залежно від погодних умов, часу доби, календарної дати та географічної широти.

***Предмет дослідження*** – методи та підходи до комп'ютерного моделювання процесів управління міським освітленням з урахуванням змін природного освітлення в різних умовах.

***Проблема***, яку розв'язує кваліфікаційна робота, полягає в покращенні графіків роботи системи міського комунального освітлення шляхом впровадження комп'ютерної моделі, заснованої на методах машинного навчання.

***Область застосування*** – результати кваліфікаційної роботи можуть бути використані при проєктуванні та вдосконаленні систем автоматизованого управління міським освітленням. Запропонована комп'ютерна модель може бути адаптована для використання в муніципальних службах, енергетичних підприємствах та інтелектуальних системах «розумного міста» з метою підвищення енергоефективності, покращенні графіків освітлення та зниження експлуатаційних витрат.

***Ключові слова***: МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ПРОГНОЗУВАННЯ, МІСЬКЕ ОСВІТЛЕННЯ, СТАЛЕ МІСТО, API.

# ABSTRACT

The explanatory note to the bachelor's thesis consists of three main sections, conclusions, a list of references and two appendices. The volume of the work is 74 pages, of which 50 pages are the main part, including 25 figures, 1 table and 18 references in the list of references and four appendices.

The purpose of the qualification work is to create a model based on machine learning methods to improve and increase the flexibility of urban public lighting schedules.

The object of research is the process of functioning of the urban lighting system under external factors that vary depending on weather conditions, time of day, calendar date and geographical latitude.

The subject of the study is methods and approaches to computer modeling of urban lighting management processes, taking into account changes in natural light in different conditions.

The problem solved by the qualification work is to improve the schedules of the urban public lighting system by implementing a computer model based on machine learning methods.

Area of application – the results of the qualification work can be used in the design and improvement of automated urban lighting control systems. The proposed computer model can be adapted for use in municipal services, energy companies, and smart city intelligent systems to increase energy efficiency, improve lighting schedules, and reduce operating costs.



Keywords: MACHINE LEARNING, NEURAL NETWORKS, FORECASTING, URBAN LIGHTING, SUSTAINABLE CITY, API.

**ЗМІСТ**

## ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

| | | |
|---|---|---|
| MLP | - | Multilayer perceptron; |
| API | - | Application Programming Interface; |
| ReLU | - | Rectified Linear Unit; |
| ELU | - | Exponential Linear Unit; |
| DataFrame | - | a data structure constructed with rows and columns; |
| MSE | - | Mean Squared Error; |
| LED | - | Light-emitting diode |
| ДБН | - | Державні будівельні норми |

# ВСТУП

**Актуальність дослідження.** Сучасний світ розвивається дуже швидко, особливо швидко розвиваються різноманітні інформаційні технології. Комп'ютерні технології вже давно стали частиною нашого повсякденного життя, а нейронні мережі впевнено, з кожним днем, захоплюють все більшу і більшу частку нашого життя та нашого перебування у цифровому світі. Нейронні мережі вже можна зустріти навіть у холодильнику та пральній машині.

Робота полягає у розробці автоматизованої системи аналізу даних та подальшому прогнозуванні стану роботи міського комунального освітлення. Ця система базується на методах машинного навчання. Її ціль вдосконалити графіки роботи підприємств міського освітлення комунального типу задля збереження енерго- та грошових ресурсів.

**Об'єкт дослідження** – процес функціонування системи міського освітлення в умовах зовнішніх факторів, що змінюються залежно від погодних умов, часу доби, календарної дати та географічної широти.

**Предмет дослідження** – методи та підходи до комп'ютерного моделювання процесів управління міським освітленням з урахуванням змін природного освітлення в різних умовах.

**Область застосування** – результати дослідження можуть бути використані при проєктуванні та вдосконаленні систем автоматизованого управління міським освітленням. Запропонована комп'ютерна модель може бути адаптована для використання в муніципальних службах, енергетичних підприємствах та інтелектуальних системах «розумного міста» з метою підвищення енергоефективності, покращенні графіків освітлення та зниження експлуатаційних витрат.

**Мета дослідження** – створення, на основі методів машинного навчання, моделі для покращення та збільшення гнучкості графіків міського комунального освітлення.

Згідно з метою дослідження було визначено та виконано такі **завдання**:

1. Аналіз основних методів управління інфраструктурою міського освітлення;
2. Огляд сучасних технологічних рішень для підвищення ефективності управління;
3. Дослідження підходів до вибору ефективної технологічної платформи для реалізації моделі системи управління мережею міського освітлення;
4. Аналіз наявних відкритих наборів даних, придатних для навчання моделі управління мережею міського освітлення;
5. Формування власного датасету для навчання та валідації комп'ютерної моделі системи управління міським освітленням;
6. Розробка комп'ютерної моделі системи управління мережею міського освітлення;
7. Проведення тестування моделі та оцінка її ефективності;
8. Аналіз програмних рішень для підвищення економічності та функціональної ефективності системи освітлення.

Для розв'язання поставлених завдань нами були використані такі методи дослідження: теоретико-критичний аналіз наукової літератури, методи математичного моделювання, методи класу Data mining, методи зіставлення, узагальнення та синтезування інформації.

Нейронні мережі в цій ситуації як найкраще підходять для аналізу та прогнозування стану освітлення, адже вони можуть дуже ефективно адже вони можуть дуже ефективно врівноважувати важливість різних факторів на кінцевий результат, інтегруючи просторово-часові характеристики навколишнього середовища, наприклад погодні умови, час та дата чи географічна широта міста для якого проводяться розрахунки, забезпечуючи тим самим адаптивне та економне планування графіка включення й вимкнення міського комунального освітлення.

# РОЗДІЛ 1
# ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1. Машинне навчання

Сам термін машинного навчання існує вже майже сімдесят років. Його запропонував один з піонерів галузі штучного інтелекту – Артур Семюель [1]. З того часу наука зробила декілька якісних стрибків у розвитку усього що пов’язано з комп’ютерними науками: від матеріальної бази до алгоритмів обробки та зберігання інформації. Але ідеї, що були запропоновані піонерами лежать в основі технології машинного навчання та нейронних мереж навіть зараз. Так, наприклад, Дональд Гебб у 1949 році в своїй книзі «The Organization of Behavior: A Neuropsychological Theory» запропонував теоретичну модель з штучних нейронів та механізми взаємодії між ними. Алан Тюрінг у 1950 році вперше використав поняття штучного інтелекту [2]. І таких піонерів було дуже багато, їх наукові здобутки до сих пір дуже сильно впливають на вивчення та створення «самонавчальних програм».

Тепер машинне навчання це окремий розділ комп’ютерних наук, який дуже швидко розвивається і вже дуже сильно впливає на більшість сфер людського життя та господарства. Програми на основі машинного навчання присутні у наших смартфонах, соціальних мережах, на заводах та фабриках, та навіть у сільському господарстві та міській інфраструктурі.

Поняття «машинне навчання» охоплює автоматичне виявлення закономірностей у даних [3]. У загальному сенсі, навчання являє собою процес трансформації накопиченого досвіду в знання або компетенції. Алгоритм навчання використовує тренувальні дані як вхідний досвід, а результатом є набір знань, що зазвичай втілюється у вигляді комп'ютерної програми, здатної виконувати певні, задумані програмістом, функції.

Для цього використовуються різні алгоритми, що дозволяють системам аналізувати дані, вдосконалювати їх інтерпретацію та генерувати прогнози. У

процесі обробки великих обсягів даних створюється модель, яка виступає основою для виконання цільових завдань.

Машинне навчання ефективно вирішує такі виклики, як складність завдань і потреба в адаптивності. Складність завдань пов’язана з проблемами, які є надто комплексними для традиційного програмування, зокрема розпізнавання мовлення, обробка зображень чи аналіз великих і складних наборів даних. Водночас адаптивність дозволяє інструментам машинного навчання динамічно коригувати свою поведінку залежно від вхідних даних. На відміну від статичних запрограмованих систем, ці інструменти здатні реагувати на зміни в середовищі, забезпечуючи гнучкість і ефективність.

### 1.2. Аналіз існуючих типів нейронних мереж

Нейронні мережі є одним із ключових інструментів сучасного машинного навчання, що дозволяють вирішувати широкий спектр завдань – від класифікації та регресії до розпізнавання образів і генерації даних. Вони базуються на моделюванні структури мозку, складаючись із шарів взаємопов’язаних вузлів-нейронів, які обробляють інформацію, передаючи її від входу до виходу. Для розробки комп’ютерної моделі управління мережею міського комунального освітлення, аналіз різних типів нейронних мереж є необхідним кроком.

Основні типи нейронних мереж:

- Перцептрон

Перцептрон це найпростіша з нейронних мереж, яку тільки можна уявити. Вона складається з одного шару нейронів і розроблена спеціально для задач лінійної класифікації. У такій моделі, вхідні дані зважуються, сумуються, а результат пропускається через порогову функцію активації.

Звичайно, що така модель, дуже проста у реалізації, не вимагає великих обчислювальних ресурсів та швидко навчається на лінійно-віддільних даних, але відсутність багатошаровості робить цю нейронну мережу дуже неефективною у питаннях обробки складних нелінійних закономірностей.

- Багатошаровий перцептрон

Багатошаровий перцептрон є розширенням описаного вище одношарового перцептрону. Така модель включає в себе один або декілька прихованих шарів з функціями активації.

Найголовніша перевага такої моделі полягає в універсальності: багатошаровий перцептрон здатен до вирішення майже будь-якої функції за наявності достатньої кількості шарів та нейронів. Його дуже часто використовують при обчислені табличних даних та в задачах регресії та класифікації. Але з великою універсальністю приходить велика відповідальність. Багатошаровий перцептрон вимагає дуже тонкого та кропітливого налаштування гіперпараметрів. Також він схильний до перенавчання, а витрати обчислювальних ресурсів для навчання дуже швидко зростають з кількістю як шарів, так і нейронів.

- Рекурентні нейронні мережі

Рекурентні нейронні мережі призначені для обробки послідовних даних, таких як часові ряди чи текст. Мережі з такою архітектурою мають зворотні зв’язки, які дозволяють зберігати інформацію про попередні входи даних, також вони добре підходять для вирішення задач, які потребують урахування контексту або залежностей між вхідними даними. Але такі моделі дуже вимогливі до обчислювальних ресурсів при навчанні.

### 1.3. Обґрунтування вибору моделі нейронної мережі

Задача, яку потрібно вирішити, полягає у бінарній класифікації. Модель повинна визначити чи потрібно ввімкнути, тобто обрати клас 1, чи вимкнути світло – обрати клас 0.

Дані на яких базується прогнозування табличні, серед них є 3 числові ознаки: час, день року та географічна широта. А також одна категоріальна ознака – стан погоди.

До розроблюваної комп’ютерної моделі є ще одна вимога – низькі обчислювальні потреби. Для того, щоб запускати нейронну мережу навіть на низькопродуктивних комп’ютерних системах.

Враховуючи усі вищезазначені фактори як архітектуру нейронної мережі для комп’ютерної моделі прогнозування графіків роботи комунальної системи освітлення було обрано багатошаровий перцептрон (MLP – multilayer perceptron). Також, як спосіб навчання для моделі було обрано контрольоване навчання.

Багатошаровий перцептрон складається з:

- вхідного шару, який приймає вхідні дані у вигляді вектору ознак;
- прихованих шарів, які обробляють інформацію з попередніх шарів. Кількість прихованих шарів та нейронів у них визначає складність багатошарового перцептрону;
- вихідний шар, який формує кінцевий результат моделі.

Кожен нейрон у такій нейронній мережі пов’язаний з усіма нейронами наступного шару, таким чином утворюючи повнозв’язну архітектуру.

Використання MLP дозволяє моделювати складні взаємозв’язки між вхідними параметрами. Нелінійна апроксимація, реалізована через приховані шари, забезпечує високу точність прогнозування в умовах змінних зовнішніх факторів. MLP характеризується низькими вимогами до обчислень під час інференсу, оскільки передбачає лише матричні операції та застосування функцій активації. Це забезпечує швидкодію та ефективність. MLP дозволяє легко адаптувати модель до змін у задачі шляхом коригування кількості шарів, нейронів або функцій активації. Багатошаровий перцептрон обрано як основну архітектуру для моделі управління комунальним освітленням завдяки його відповідності задачі бінарної класифікації, ефективній роботі з табличними даними, здатності до нелінійної апроксимації, низьким обчислювальним вимогам та гнучкості налаштування. Цей вибір забезпечує надійність і практичність розробленого рішення [4].

### 1.4. Основні характеристики та господарський вплив моделі

Розробка комп'ютерної моделі управління мережею міського комунального освітлення на основі нейронної мережі передбачає низку ключових характеристик, які можуть мати значний вплив на міське господарство. Ці характеристики спрямовані не лише на підвищення технічної ефективності системи, але й на сприяння економічному та екологічному розвитку міста.

Система управління мережею міського освітлення на основі нейронної мережі характеризується високою точністю прогнозування, швидкістю прийняття рішень, адаптивністю до змінних умов, можливістю інтеграції з існуючою інфраструктурою та масштабованістю. Точність прогнозування забезпечується здатністю моделі визначати необхідність увімкнення або вимкнення освітлення з урахуванням погодних умов, часу доби, дня року та географічного положення, що сприяє покращеному використанню ресурсів. Швидкість прийняття рішень досягається завдяки обробці даних у реальному часі, що дозволяє оперативно реагувати на зміни зовнішніх умов, таких як зміна стану погоди. Адаптивність проявляється в автоматичному врахуванні сезонних і погодних факторів, що впливають на потребу в освітленні. Можливість інтеграції з існуючою інфраструктурою забезпечує сумісність моделі з поточними системами без значних витрат на модернізацію. Масштабованість дозволяє розширювати застосування моделі на різні райони або міста, що підвищує її практичну цінність.

Розроблювана модель управління освітленням може впливати на міське господарство в кількох напрямах. Оптимізація роботи освітлювальних приладів сприяє раціональному використанню ресурсів, що є важливим для міст із високим енергоспоживанням [5]. Ефективне управління освітленням також може підвищувати безпеку громадян у темний час доби або за несприятливих погодних умов, зменшуючи ризики аварій і злочинності [6, 7]. Крім того, належне освітлення покращує якість життя мешканців, роблячи міське середовище комфортнішим і привабливішим, що може позитивно

позначитися на туристичній привабливості міста. Інтеграція моделі в концепцію "розумного міста" підтримує загальну тенденцію до покращення роботи міських процесів за допомогою технологій, сприяючи модернізації та сталому розвитку.

Теоретичне впровадження моделі управління освітленням на основі нейронної мережі може сприяти значному економічному ефекту. Точне прогнозування потреби в освітленні дозволяє уникати надмірного споживання електроенергії, що потенційно скорочує витрати. Оптимальне використання приладів також подовжує їхній термін служби, зменшуючи витрати на технічне обслуговування та заміну обладнання. Покращення інфраструктури шляхом інтеграції такої моделі може підвищити привабливість міста для туристів та інвесторів, що сприяє економічному зростанню. Таким чином, модель має потенціал забезпечити економію ресурсів і створити умови для їх перерозподілу на інші міські потреби [5].

Екологічний вплив моделі полягає в кількох аспектах. Зменшення споживання електроенергії сприяє скороченню викидів вуглекислого газу, що є важливим кроком у боротьбі зі зміною клімату та відповідає цілям сталого розвитку. Покращення режиму освітлення також може зменшити світлове забруднення, що позитивно впливає на екосистеми та здоров’я населення. Ці фактори підкреслюють екологічну значущість розробки [8].

Сучасні міста стикаються з викликами управління інфраструктурою, такими як недостатня гнучкість традиційних методів, зростання цін на енергоносії, можливі пошкодження або знищення енергетичної інфраструктури та потреба в підвищенні безпеки й комфорту мешканців. Розроблювана модель управління освітленням на основі нейронної мережі може сприяти вирішенню цих проблем шляхом забезпечення точного, адаптивного та ефективного контролю. Теоретичний вплив такої моделі полягає в підвищенні енергоефективності, безпеки та привабливості міського середовища.

Варто зазначити, що зекономлені міські ресурси можна буде перерозподілити на ремонт чи модернізацію зношених елементів енергетичної інфраструктури. Або направити на інші соціально значимі проекти. Наприклад, медична допомога, проекти модернізації шкіл, садочків, коледжів, університетів, міський благоустрій та багато-багато інших галузей міського господарства.

### 1.5 Огляд та аналіз публікацій

Зазначеною проблемою покращення енергоефективності міських мереж комунального освітлення займалось багато науковців з різних галузей людського господарства. У ході написання кваліфікаційної роботи автор ознайомився з науковими результатами:

- Яцишин В. С. (2024) Керування електроспоживання в SmartGrid

У своєму дослідженні автор використовує алгоритми машинного навчання для збільшення ефективності використання електроенергії, зменшення навантаження на енергетичну інфраструктуру. Дослідник використовує історичні дані зібрані під час вимірювань для керування споживанням електроенергії різними пристроями, також досліджується можливість прогнозувати споживання електроенергії за допомогою нейронних мереж. У першому розділі своєї роботи автор заявляє про можливу максимальну ефективність у 10-15%.

- Pragna Labani Sikdar,· Samarjit Kar,· Parag Kumar Guha Thakurta (2024) ANNEGA: an artificial neural network embedded genetic algorithm approach for energy efficient street lighting

У ході розробки рішення для підвищення використання електроенергії системою міського освітлення автори роботи використовують комбінацію алгоритмів машинного навчання із генетичними алгоритмами, таким чином дослідники наполягають на тому, що досягли дуже хороших результатів зменшивши енергоспоживання на 41.94%, а помилки під час тренування нейронної мережі на 94.8% порівняно з існуючими моделями.

- Fouad Agramelal, Fouad Agramelal, Youssef Moubarak, Saad Abouzahir (2023) Smart Street Light Control: A Review on Methods, Innovations, and Extended Applications

У даній роботі автори дослідили різні вже існуючі методи управління вуличним освітленням, у тому числі і за допомогою алгоритмів машинного навчання. Наведено досить великий список різноманітних способів управління. Розглядаються як питання економії електроенергії, так і покращення міського середовища. Наведені методи пропонують різний вплив на шукані покращення, який досягається різними способами. Наприклад, одним з методів пропонується зменшувати інтенсивність освітлення в залежності від активності трафіку на дорозі: чим менше трафіку тим менше інтенсивність. Інший метод використовує датчики освітлення для коригування інтенсивності освітлення. Загалом у дослідницькій роботі наведено більше 120 різних методів та різних параметрів цих методів для управління вуличним освітленням.

- Muhammad Asif, Sarmad Shams, Samreen Hussain, Jawad Ali Bhatti, Munaf Rashid, Muhammad Zeeshan-ul-Haque (2022) Adaptive Control of Streetlights Using Deep Learning for the Optimization of Energy Consumption during Late Hours

У науковій праці описують спосіб адаптивного контролю системи міського освітлення за допомогою використання глибокого навчання. Уся робота побудована навколо використання камери для фіксації об’єктів, що з’являються на вулиці, після чого відео передається до нейронної мережі, яка вирішує, що саме з’явилось на екрані, після чого відповідно до характеру об’єкту, змінюється інтенсивність освітлення. Дослідники стверджують, що під час експериментального використання розробленої методики досягли середнього показника економії електроенергії у 31.25%.

**Висновки до розділу 1**

У першому розділі було здійснено теоретичний аналіз основних понять, пов’язаних із машинним навчанням та штучними нейронними мережами, що становлять фундамент для побудови інтелектуальних систем. Розглянуто структуру та принципи функціонування нейронних мереж, особливу увагу приділено багатошаровому перцептрону (MLP) як найбільш доцільній моделі для задач прогнозування у сфері автоматизованого керування міським освітленням. Встановлено, що MLP завдяки своїй архітектурі здатен ефективно обробляти вхідні параметри, такі як час доби, дата, погодні умови тощо, що дозволяє йому адаптуватися до змін зовнішнього середовища та приймати обґрунтовані рішення щодо вмикання або вимикання освітлення. Проведений аналіз літературних джерел засвідчив актуальність та перспективність використання нейромережевих технологій у сфері енергозбереження, зокрема для покращення ефективності роботи комунального освітлення.

# РОЗДІЛ 2
# РОЗРОБКА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

## 2.1 Розробка навчального набору даних

Навчальний набір даних є одним із найважливіших компонентів у процесі розробки моделей нейронних мереж, оскільки саме він визначає якість, ефективність і здатність моделі до узагальнення під час виконання завдань. Добре підготовлений і ретельно розмічений набір даних не лише сприяє швидкому й точному навчанню моделі, а й забезпечує її стабільну продуктивність на нових, раніше небачених даних.

У процесі навчання нейронних мереж навчальний набір даних слугує основою для оптимізації параметрів моделі. Цей процес відбувається шляхом ітеративного зменшення функції втрат за допомогою алгоритмів, таких як зворотне поширення помилки (*backpropagation*) [11]. Кожен приклад із навчального набору використовується для обчислення градієнтів, які коригують модель, дозволяючи їй поступово наближатися до оптимального стану. Тобто, можемо сказати, що якість і структура даних безпосередньо впливають на те, наскільки точно модель зможе відобразити залежності між вхідними ознаками та цільовими значеннями.

Для того щоб навчальний набір даних відповідав потребам нейронної мережі, він має відповідати кільком основним вимогам. По-перше, дані повинні бути репрезентативними, тобто відображати реальний розподіл прикладів, з якими модель зіткнеться в практичних умовах. По-друге, обсяг даних має бути достатнім, щоб забезпечити адекватне покриття простору ознак і запобігти перенавчанню, особливо у випадку складних глибоких архітектур. Точність розмітки є критично важливою для контрольованого навчання, адже помилки в мітках можуть призвести до неправильного налаштування параметрів моделі, що знизить її продуктивність [12].

Було вирішено згенерувати датасет вручну за допомогою програмних засобів мови Python, після чого провести розмітку даних.

Для генерування даних було використано дві бібліотеки мови програмування Python:

- Pandas;
- Random.

Бібліотека *Random* забезпечує можливість створення випадкових значень, що дозволяє генерувати різноманітні комбінації даних. Завдяки цьому вдається імітувати широкий спектр умов, які можуть впливати на досліджувані характеристики, такі як погода, сприяючи формуванню реалістичного набору даних.

У свою чергу, бібліотека *Pandas* застосовується для ефективної роботи з табличними даними. Вона надає інструменти для створення структурованих об’єктів, таких як *DataFrame*, які полегшують зберігання, обробку та аналіз великих обсягів інформації. У цьому коді її основна функція полягає у формуванні згенерованих даних у зручний формат та їх збереженні у вигляді файлу формату *.csv*, що забезпечує сумісність із подальшими етапами аналізу та машинного навчання.

Таким чином, спільна робота цих бібліотек дозволила створити та організувати набір даних, який слугує основою для навчання нейронної моделі, поєднуючи генерацію різноманітних умов із їх структуризацією для подальшого використання.

Для розроблюваної моделі було вирішено використовувати 4 параметри, а саме:

- Погода;
- Час;
- Дата;
- Географічна широта.

Ці параметри мають дуже сильний вплив на рівень освітлення і відповідно на час в який система міського комунального освітлення повинна вмикатися та вимикатися. Але при цьому для отримання цих параметрів нам не потрібно використовувати коштовну інфраструктуру, що забезпечує низьку

вартість та легкість встановлення та використання розробленої моделі (рис. 2.1).

Код починається з визначення основних параметрів: weather_options – це масив який містить у собі усі стани погоди, які повинні бути у наборі даних. Наприклад, «clear» – ясно, «stormy» – шторм, «snow» – сніг тощо; total_records – визначає кількість записів, які планується згенерувати. Було обрано 5000 для того щоб гарантувати покриття усіх можливих сценаріїв та забезпечити повне та рівномірне навчання моделі нейронної мережі.

Далі створюється три списки для часу, дати та географічних широт. Кожне значення у кожному списку обирається випадково, але є одне можна сказати виключення. Для днів року застосовується більш складний алгоритм. Для рівномірного покриття набором даних усіх сезонів року обирається 6 днів для кожного місяця, а щоб визначити скільки днів у місяці використовується досить проста функція *days_in_month*, яка приймає місяць та рік і виводить кількість днів у місяці. Це було створено для того, щоб виключити ситуації з некоректними даними. Такими як, наприклад 30 лютого або 31 червня.


Рисунок 2.1 – Блок-схема програми створення навчального набору даних

Після підготовки списків запускається цикл генерації записів, де для кожного нового запису випадково обирається стан погоди, час, дата та географічна широта. Після чого відбувається збереження згенерованих записів у файл формату *.csv*.

## 2.2 Розмічення створеного набору даних

Після створення набору даних наступним кроком є підготовка цього набору даних, зокрема їх розмічення. Розмічання даних є критично важливим етапом, оскільки воно забезпечує надання моделі точних міток, які слугують основою для навчання в задачах контрольованого машинного навчання. Якісна розмітка дозволяє алгоритму виявляти закономірності та залежності між вхідними ознаками та цільовими значеннями, що підвищує точність прогнозів.

### 2.2.1. Програма для розмітки створеного набору даних

Було написано програму для розмітки набору даних, який було створено за допомогою іншої програми описаної у пункті 2.1. Враховуючи, що модель нейронної мережі виконуватиме бінарну класифікацію на основі чотирьох змінних можна зробити висновок, що програма з розмітки набору даних повинна приймати чотири змінні, а саме:

- Погода
- Час
- Дата
- Географічна широта

Та видавати результат у вигляді бінарної змінної, тобто 0 або 1.

Для цього треба обчислювати час сходу та заходу Сонця у даний день року на даній географічній широті під час даного погодного стану. Таким чином, потрібно розробити алгоритм, який буде вираховувати час сходу і заходу Сонця у даній точці земної кулі у даний день року, а потім в залежності від погодних умов визначити чи потрібно ввімкнути світло у даний момент часу.

Функція обчислення часу сходу і заходу Сонця приймає день року та географічну широту та повертає відповідно час сходу і заходу Сонця. Вона включає в себе декілька кроків.

Першим кроком широта переводиться з градусів до радіанів.

Другим кроком визначається деклінація Сонця. Для обчислення часу сходу й заходу Сонця важливе поняття сонячної деклінації – кута між променями Сонця та площиною екватора. [13].

Щоб моделювати зміну деклінації, використовують періодичну функцію – синус, бо зміна кута відбувається синусоїдально. Може виникнути питання: «Чому саме 360 поділити на 365?». Тому що Земля обертається навколо Сонця приблизно за 365 днів, описуючи за цей час повне коло – тобто 360°. Щоб знайти, на якому «кутовому положенні» орбіти знаходиться Земля у день року n треба поділити 360 градусів на 365 днів та потім помножити на різницю між поточним днем та днем весіннього рівнодення, коли деклінація Сонця дорівнює 0 (2.1):

$$\delta = 23.44^\circ \times sin\left(\frac{360}{365} \times (n - 81)\right) \quad (2.1)$$

Третім кроком обчислюється омега – годинний кут при сході/заході Сонця (2.2).

$$\cos(\omega) = -\tan(\varphi) \times \tan(\delta), \quad (2.2)$$

де φ – географічна широта у радіанах

δ – деклінація Сонця

Четвертим кроком обчислюється тривалість дня (2.3):

$$L_{day} = \frac{2\omega}{15} \quad (2.3)$$

П’ятим кроком обчислюються моменти сходу і заходу сонця. Береться сонячний зеніт – 12:00 та відповідно для моменту сходу сонця віднімається довжина дня поділена на 2, а для моменту заходу сонця додається (2.4 та 2.5).

$$\text{Час сходу Сонця} = 12 - \frac{L_{day}}{2} \quad (2.4)$$

$$\text{Час заходу Сонця} = 12 + \frac{L_{day}}{2} \quad (2.5)$$

Шостим кроком отримані значення переводяться у хвилини, для більш зручного подальшого використання, шляхом множення на 60. Після чого функція повертає відповідні значення.

Сьомим кроком відбувається перевірка на літній час. Якщо поточна дата входить у зазначений в функції діапазон, то до часу сходу і заходу Сонця додається 60 хвилин для компенсації зміщення астрономічного часу відносно місцевого.

Восьмим кроком відбувається корекція за допомогою калібрувальної моделі, алгоритм якої буде описано далі.

Дев’ятим кроком відбувається корекція за станом погоди. У функції описано кожний стан погоди та відповідні значення корекції для моменту вмикання та вимикання світла.

Десятим кроком відбувається фінальне обчислення часу вмикання та вимикання світла. Після чого виконується перевірка чи поточний час більше та менше ніж відповідно час вмикання та вимикання світла, якщо так то функція повертає 1 – світло ввімкнути, якщо ні, то 0 – світло вимкнути.

Одинадцятим кроком зберігаються усі дані розмітки у окрему колонку, після чого зберігається сам файл з розміченими даними у форматі .csv. (рис. 2.2)

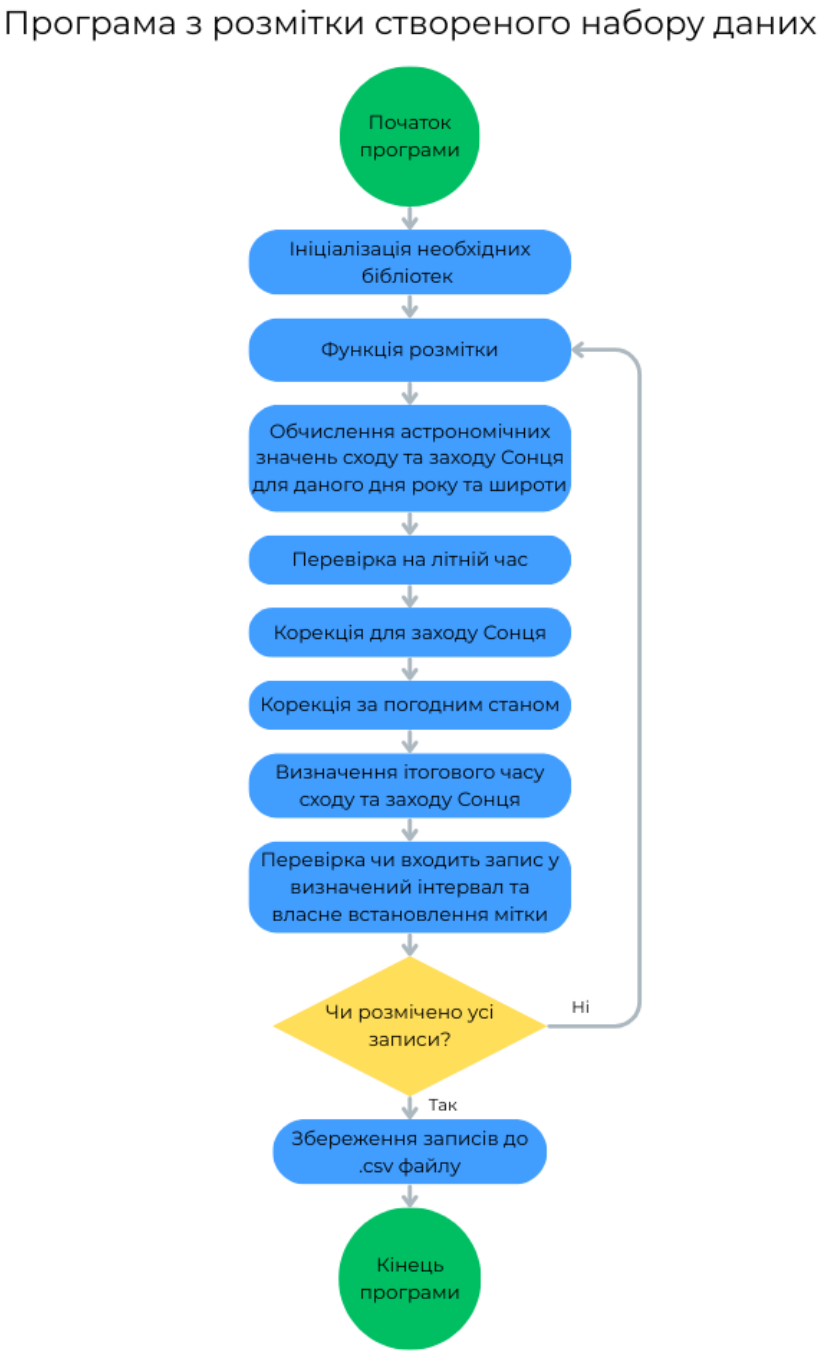


Рисунок 2.2 – Блок-схема програми розмітки створеного набору даних

### 2.2.2. Калібрувальна модель для заходу Сонця

Після початкового розмічення даних та тренування моделі на цих даних було помічено неприємну тенденцію: початковий алгоритм розмітки за якоїсь причини не міг коректно визначити час заходу сонця через, що виникали викривлення даних і відповідно неточності у роботі моделі згодом.

Для вирішення цієї проблеми використано статистичну модель лінійної регресії. Для навчання цієї моделі взято історичні дані з набору даних GlobalWeatherRepository [9].

Перед цим історичні дані фільтруються. Залишається 4 колонки, а саме: [country], [sunrise], [sunset], [day_of_year]. У самій програмі навчання калібрувальної моделі повторюються деякі кроки з безпосередньо програми розмітки. Тут також визначається час астрономічного сходу та заходу Сонця, робиться корекція на літній час за необхідності і перетворюється часовий формат HH:MM на прості хвилини.

Основний алгоритм програми включає в себе:

- розрахування теоретичного часу заходу Сонця;
- додаються нелінійні ознаки для моделювання сезонних змін у корекції часу заходу Сонця;
- навчається модель лінійної регресії.

У програмі використовується вбудована функція *LinearRegression* бібліотеки *sklearn*. Модель намагається мінімізувати помилку між передбаченими значеннями і реальними значеннямих [10]. Для цього використовується функція втрат, а саме середньоквадратична помилка MSE (2.6).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_{i_{pred}})^2 \qquad (2.6)$$

Метою лінійної регресії є знайти такі ваги ($w_i$), щоб середньоквадратична помилка була найменшою. Враховуючи, що ми передаємо ознаки ['day_of_year', 'latitude', 'day_of_year_sin', 'day_of_year_cos'], то лінійна регресія знаходить ваги для рівняння (2.7):

$$sunsetCorr = w_0 + w_1 \times day_{of_{year}} + w_2 \times latitude + \\ + w_3 \times day\_of\_year\_sin + w_3 \times day\_of\_year\_cos \quad (2.7)$$

Власне після навчання модель зберігається і потім використовується у програмі розмітки з такими самими ознаками, але значення вже звісно не історичні, а згенеровані.

### 2.3 Розробка моделі нейронної мережі

Як вже було зазначено у пункті 1.3 для моделі нейронної мережі було обрано архітектуру багатошарового перцептрону, або як його ще називають Multi-Layer Perceptron (MLP). Така архітектура дозволяє моделювати складні нелінійні взаємозв'язки між вхідними параметрами, також MLP моделі характеризуються високою точністю прогнозування та низькими вимогами до обчислювальних ресурсів, що у свою чергу забезпечує швидку та ефективну роботу програми.

#### 2.3.1 Архітектура моделі нейронної мережі

Багатошаровий перцептрон легко піддається адаптації до потрібних параметрів шляхом зміни кількості прихованих шарів та нейронів у шарах. А враховуючи, що навчальні дані представлені у вигляді таблиці вибір багатошарового перцептрон у як архітектури для моделі нейронної мережі є оптимальним та зваженим.

Під час розробки моделі нейронної мережі було випробувано декілька різних варіантів. Сюди входять моделі з різною кількістю шарів та нейронів у них, різні функції активації та інше.

Але було вирішено зупинитися на наступній моделі (рис. 2.3):

```
# --- Структура нейронної моделі ---
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dropout(0.3),
    Dense(16, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Компіляція моделі
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])

# --- Навчання моделі ---
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=120, batch_size=32, class_weight=class_weights,
                    validation_split=0.25, callbacks=[early_stopping])
```

Рисунок 2.3 – Скріншот коду структури нейронної мережі

Така структура показала найточніші результати під час навчання. Це послідовна нейронна мережа, що складається з 4 повнозв'язних шарів нейронів. Перший шар є вхідним, що задається параметром input_shape та складається з 64 нейронів, другий шар складається з 32 нейронів, третій шар з 16 нейронів. Усі шари мають функцію активації *Relu* (f(x) = max(0,x), окрім останнього. Останній шар є вихідним, складається з 1 нейрона з *логістичною* (сигмоїдною) функцією активації. У якості порогу активації для вихідного шару взято число 0.5 (2.8):

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

Також присутні так звані *Dropout* шари, які запобігають перенавчанню моделі [14, 15]. Механізм досить простий: програма випадковим чином вимикає певну зазначену кількість нейронів, у цьому випадку 30%, таким чином за кожної епохи виходить нова мережа. Це стимулює модель не

покладатися на певні стійкі комбінації нейронів, що сприяє більш узагальненому та стійкому навчанню.

Компіляція моделі відбувається з оптимізатором Adam та швидкістю навчання у 0,001. Функція втрат Binary Crossentropy. Функція втрат використовується, щоб оцінити наскільки сильно відрізняються результати отримані моделлю від істинних значень (2.9):

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \times \log(\hat{y}_i) + (1 - y_i) \times \log(1 - \hat{y}_i)] \quad (2.9)$$

Під час безпосередньо самого навчання застосовується завчасна зупинка (EarlyStopping) з показником терпіння у десять епох. Ця функція забезпечує зупинку навчання моделі, якщо показник валідаційних втрат не покращується десять епох підряд та повертає ваги моделі за найкращого показника валідаційних втрат.

Валідаційна вибірка становить 25%, розмір групи – 32, загальна кількість епох відведена на навчання становить 120, але зазвичай процес навчання завершується швидше.

Також у програмі застовується One-Hot encoding для ознаки погоди для того, щоб перетворити категоріальну змінну у числову, що у свою чергу сприяє кращому навчанню моделі [16].

### 2.3.2 Оцінка моделі нейронної мережі

Після навчання моделі здійснюється оцінка її точності на тестовій вибірці. Для оцінки використовуємо 4 метрики:

- Accuracy – це частка правильних передбачень відносно усіх передбачень
- Precision – вимірює яка частка передбачень, позначених як позитивні, дійсно є такими

- Recall – вимірює яка частка всіх позитивних прикладів була передбачена моделлю

- F1-score – оцінює компроміс між метриками Precision та Recall

Ці дві метрики досить часто можуть «сперечатися» між собою. Наприклад, якщо модель стає дуже обережною і передбачає позитивний клас лише тоді, коли дуже впевнена, Precision зростає, але Recall падає, бо багато позитивних прикладів пропускаються

### 2.3.3 Порівняння нейронної мережі з різними гіперпараметрами

Як вже було зазначено на рис. 2.3 для фінальної нейронної мережі було обрано специфічні параметри, але у процесі розробки було перебрано декілька інших комбінацій гіперпараметрів.

Для оцінки ефективності нейронних мереж з різними гіперпараметрами використовуються ті ж самі метрики.

Також для кожної комбінації було створено *confusion_matrix*, або матриця плутанини (рис. 2.4):

```
# Структура моделі
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.4),
    Dense(64, activation='relu'),
    Dropout(0.4),
    Dense(32, activation='relu'),
    Dropout(0.4),
    Dense(16, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Компіляція моделі
model.compile(optimizer=Adam(learning_rate=0.0005), loss='binary_crossentropy', metrics=['accuracy'])

# Навчання моделі
history = model.fit(X_train, y_train, epochs=120, batch_size=64, class_weight=class_weights,
                    validation_split=0.25, callbacks=[early_stopping])
```

Рисунок 2.4 – Скріншот коду структури першої нейронної мережі

Першу нейронну мережу можна охарактеризувати збільшеною кількістю нейронів у шарах. Також збільшено Dropout, зменшено швидкість

навчання та величину партії. У результаті отримали матрицю плутанини (рис. 2.5):

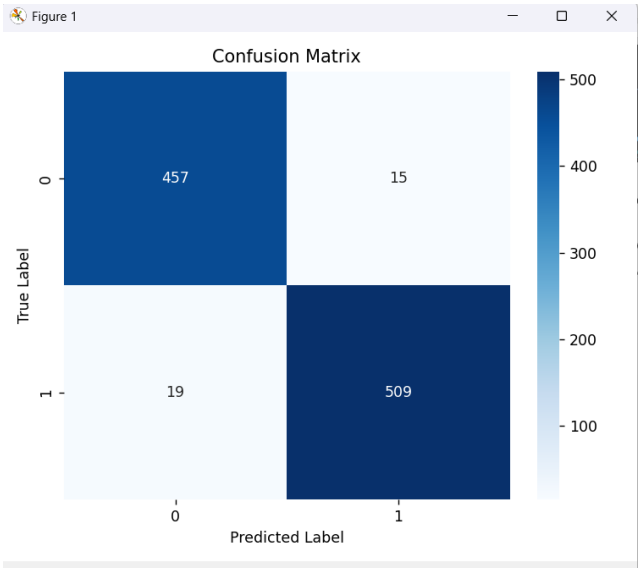


Рисунок 2.5 – Матриця плутанини для першої нейронної мережі

Значення метрик для першої нейронної мережі наведені на рис. 2.6:

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.96 | 0.97 | 0.96 | 472 |
| 1 | 0.97 | 0.96 | 0.97 | 528 |
| accuracy | | | 0.97 | 1000 |
| macro avg | 0.97 | 0.97 | 0.97 | 1000 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1000 |

Рисунок 2.6 – Скріншот метрик для першої нейронної мережі

Загалом, можна сказати, що така нейронна мережа показує гарні результати, на рівні з обраною, але займає більше часу на тренування та сама збережена нейронна мережа займає більше місця на носії даних (рис. 2.7):

На рис. 2.10 наведена структура другої нейронної мережі.

```
# Структура моделі
model = Sequential([
    Dense(64, activation='elu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),
    Dense(32, activation='elu'),
    Dropout(0.3),
    Dense(16, activation='elu'),
    Dense(1, activation='sigmoid')
])

# Компіляція моделі
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])

# Навчання моделі
history = model.fit(X_train, y_train, epochs=120, batch_size=16, class_weight=class_weights,
                    validation_split=0.25, callbacks=[early_stopping])
```

Рисунок 2.10 – Скріншот коду структури другої нейронної мережі

Для ще однієї комбінації було вирішено змінити функцію активації. Тепер замість *relu* встановлено *elu* (2.14):

$$ELU(x) = \begin{cases} x, & \text{якщо } x > 0 \\ a(e^x - 1), & \text{якщо } x \leq 0 \end{cases} \quad (2.14)$$

Матриця плутанини для другої нейронної мережі представлена на рис. 2.11:

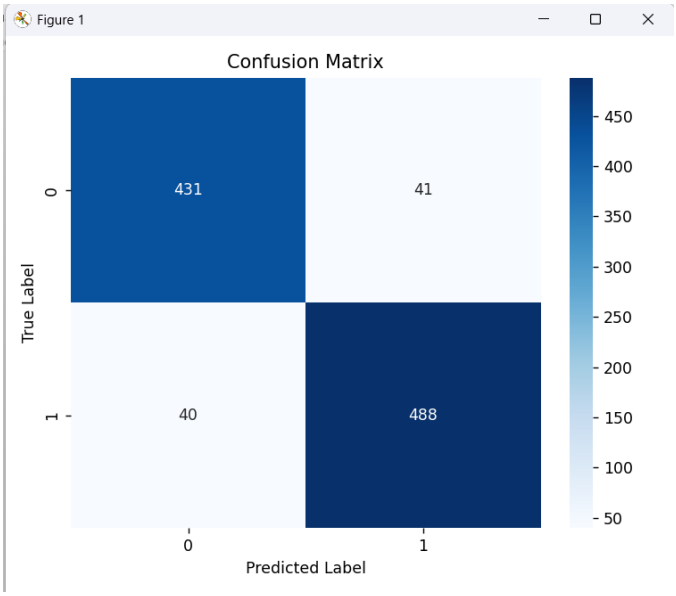


Рисунок 2.11 – Матриця плутанини для другої нейронної мережі

Метрики для другої нейронної мережі зазначені на рис. 2.12:

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.91 | 0.91 | 472 |
| 1 | 0.92 | 0.92 | 0.92 | 528 |
| accuracy | | | 0.92 | 1000 |
| macro avg | 0.92 | 0.92 | 0.92 | 1000 |
| weighted avg | 0.92 | 0.92 | 0.92 | 1000 |

Рисунок 2.12 – Скріншот метрик для другої нейронної мережі

Зміна функції активації сильно впливає на точність мережі значно підвищуючи шанси на помилку, але загалом точність все ще задовільна.

Тепер для порівняння наведемо матрицю плутанин (рис. 2.13) та метрики для обраної структури нейронної мережі (рис. 2.14):

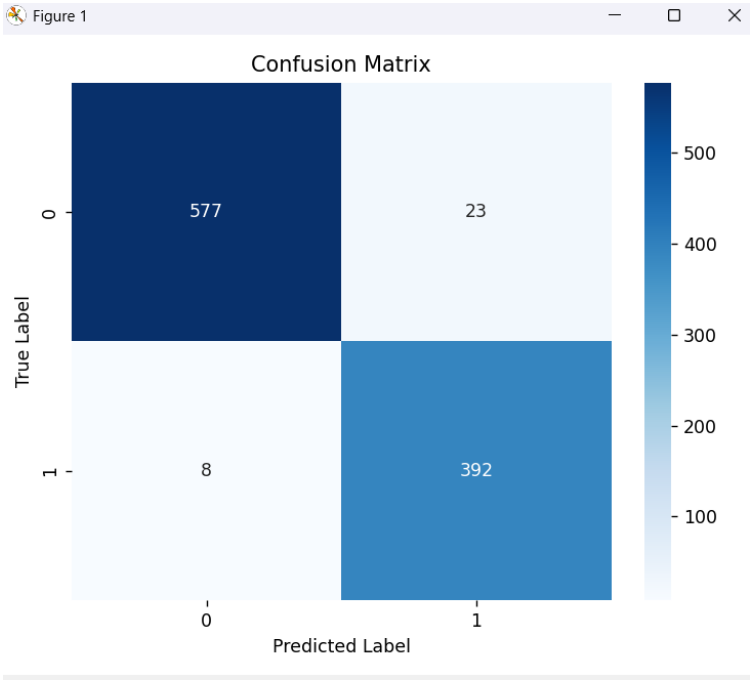


Рисунок 2.13 – Матриця плутанини для обраної нейронної мережі

| Classification Report: | | | | |
|---|---|---|---|---|
| | precision | recall | f1-score | support |
| 0 | 0.99 | 0.96 | 0.97 | 600 |
| 1 | 0.94 | 0.98 | 0.96 | 400 |
| accuracy | | | 0.97 | 1000 |
| macro avg | 0.97 | 0.97 | 0.97 | 1000 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1000 |

Рисунок 2.14 – Скріншот метрик для обраної нейронної мережі

Бачимо, що результати обраної нейронної мережі кращі з усіх протестованих, тому для подальшого використання було обрано саме цю структуру.

## 2.4 Створення користувацької програми

Після створення нейронної мережі для прогнозування графіків роботи системи міського комунального освітлення треба вирішити ще декілька питань.

Програма повинна:

- брати час та дату з персонального комп’ютера користувача;
- приймати назву населеного пункту та повертати стан погоди та географічну широту цього місця;
- виводити прогноз.

Програму було вирішено писати мовою програмування Python.

### 2.4.1 Отримання поточної дати та часу

Першу вимогу досить просто реалізувати за допомогою бібліотеки *datetime*. У коді це буде виглядати як *now = datetime.datetime.now()* – такий запис повертає поточний час системи у форматі (2025, 5, 26, 13, 54), що означає 26 травня 2025 року, 13:54, але такий запис не підходить для передачі у нейронну мережу, тому за допомогою допоміжних функції тієї ж бібліотеки відбувається розбиття цього запису на два окремих.

Функція *timetuple().tm_yday* бере рядок (2025, 5, 26, 13, 54) та повертає порядковий номер поточної дати – для 26 травня 2025 року це буде 146 день року.

Функція *strftime('%H:%M')* бере той самий рядок та повертає запис у форматі HH:MM – наприклад 13:54.

Після таких кроків ці дві ознаки готові до передавання у нейронну мережу і подальшого прогнозування.

### 2.4.2 Отримання географічної широти та стану погоди

Початково для отримання географічної широти та стану погоди планувалось використовувати тільки одне API, але у процесі роботи з OpenWeatherAPI виявилось, що список міст, містечок та селищ не повний для

України, але цей API все ще може представляти стан погоди для точки на планеті за координатами.

Таким чином було прийнято рішення про імплементування ще одного API, а саме GeocodingAPI [17]. Цей API представляє найбільш повне покриття усіх міст, містечок та селищ України, та, що мабуть стало найважливішим дає змогу фільтрувати населенні пункти за областю, таким чином можна уникнути помилок із випадковим отриманням прогнозу погоди з міста іншої області з такою ж назвою.

Власне сам процес отримання стану погоди та географічної широти для подальшого використання у прогнозуванні за допомогою розробленої нейронної мережі характеризується наступними діями:

- Користувач обирає область та вводить назву міста;
- Формується запит до GeocodingAPI використовуючи назву міста, область та тег UA, який позначає Україну;
- GeocodingAPI повертає географічну широту та довготу шуканого міста;
- Формується запит до OpenWeatherAPI використовуючи географічну широту та довготу;
- OpenWeatherAPI повертає стан погоди у шуканій точці.

Власне на цьому завершується процес отримання даних. Але є декілька критичних моментів.

По-перше, географічна широта зберігається для подальшої передачі у нейронну мережу на етапі використання GeocodingAPI.

По-друге, при тренуванні нейронної мережі було використано OneHotEncoding для кодування стану погоди у числову ознаку. На щастя, бібліотека *pickle* надає змогу зберегти конфігурацію кодувальника, щоб забезпечити відповідне кодування і для вже працюючої програми. Тому останнім кроком попередньої обробки даних є OneHot кодування стану погоди.

Імпортується кодувальник, після чого засобами бібліотеки *sklearn.preprocessing* кодується номінальна ознака стану погоди у числову.

На рис. 2.15 представлена блок-схема передачі даних до нейронної мережі:

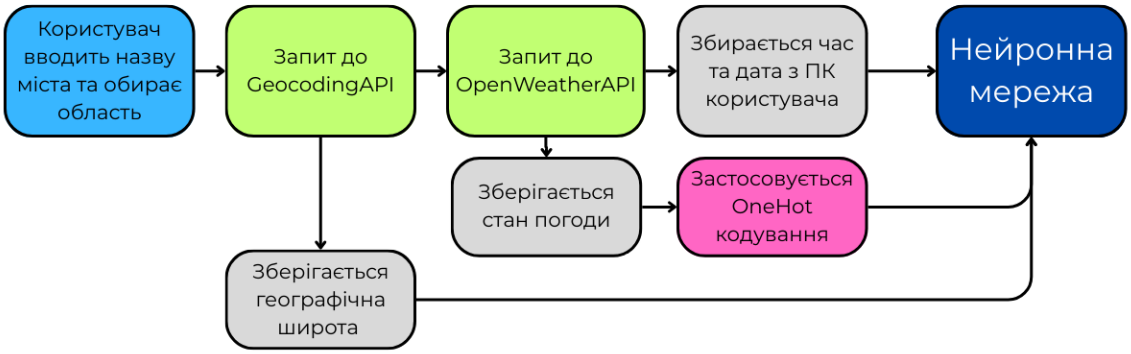


Рисунок 2.15 – Блок-схема передачі даних до нейронної мережі

### 2.4.3 Прогнозування графіків роботи міської системи комунального освітлення

На даному етапі програма може зробити запити до API та дати свій прогноз на момент часу у який цей запит було відправлено, але це не підходить для складання графіків роботи.

OpenWeatherAPI може видавати прогноз на кожну годину на 24 години вперед, тому для складання графіків використовується функція *forecast_day*.

Першим кроком відбувається ініціалізація часу та дати. Зберігається поточний час системи користувача, встановлюється крок у 10 хвилин та встановлюється кінцевий час – через 24 години після поточного.

Другим кроком робиться один запит до GeocodingAPI з метою отримати географічну широту та довготу шуканого міста та зберегти широту.

Третім кроком ініціалізується цикл та його змінні:

- recs = [] – список записів для зберігання;
- t = now – поточний час;

- last_hour = None – змінна для відстеження часу останнього прогнозу погоди;
- mapped = None – змінна для зберігання погоди;
- lat = None – змінна для зберігання широти.

Тіло циклу:

- Перевіряється чи відрізняється поточна година від останньої використаної, якщо так викликається функція get_weather_data, яка робить запит до OpenWeatherAPI;
- Формується словник з вхідними даними для моделі;
- Викликається функція predict_lighting у яку передається словник;
- Нейронна мережа робить прогноз;
- Відповідний запис додається до recs = [].
- оновлення часу на крок у 10 хвилин і цикл запускається знову.

На рис. 2.16 представлена блок-схема функції створення графіку роботи системи міського комунального освітлення:

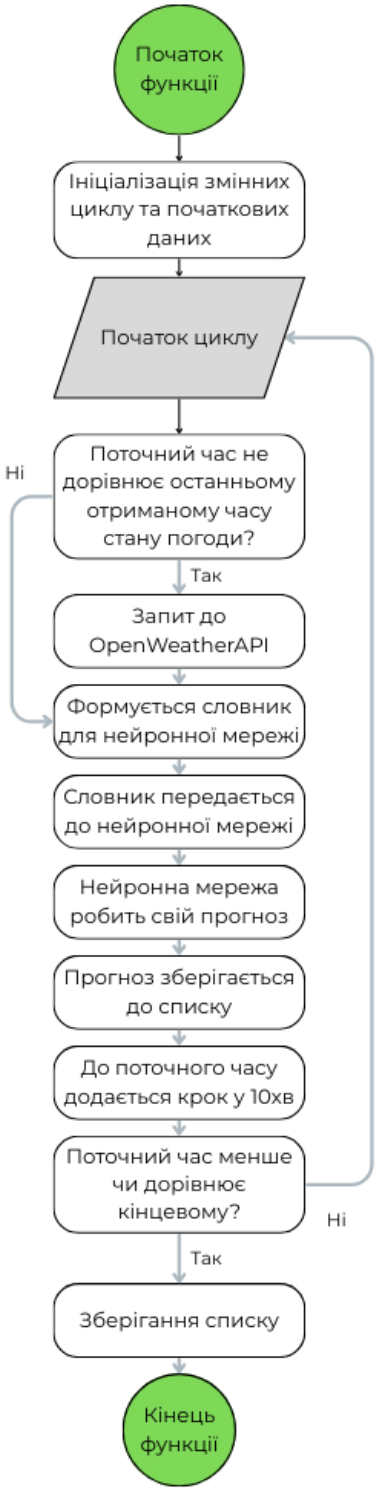


Рисунок 2.4.2 – Блок-схема функції створення графіку роботи системи міського комунального освітлення

У результаті отримаємо список записів із датою, часом та прогнозом нейронної мережі, який у подальшому можна буде вивести користувачеві, або використовувати для інших цілей.

**Висновки до розділу 2**

У розділі два описано процес створення для навчання нейронної мережі була процедури генерації та розмітки даних, що включає урахування погодних умов, часу доби, пори року та астрономічного заходу сонця. Також було створено декілька різних нейронних мереж, що відрізнялися за гіперпараметрами, порівняно та обрано найкращу з них. Після чого створено користувацьку програму.

# РОЗДІЛ 3
# ОГЛЯД ФУНКЦІОНАЛУ ТА ТЕСТУВАННЯ

## 3.1 Користувацький інтерфейс програми

Звичайно, що у програми повинен бути інтерфейс, через який користувач зможе взаємодіяти з програмою у зручний спосіб та отримувати результати (рис. 3.1).

Інтерфейс програми було реалізовано мовою програмування *Python* за допомогою методів та функцій бібліотеки *tkinter*.

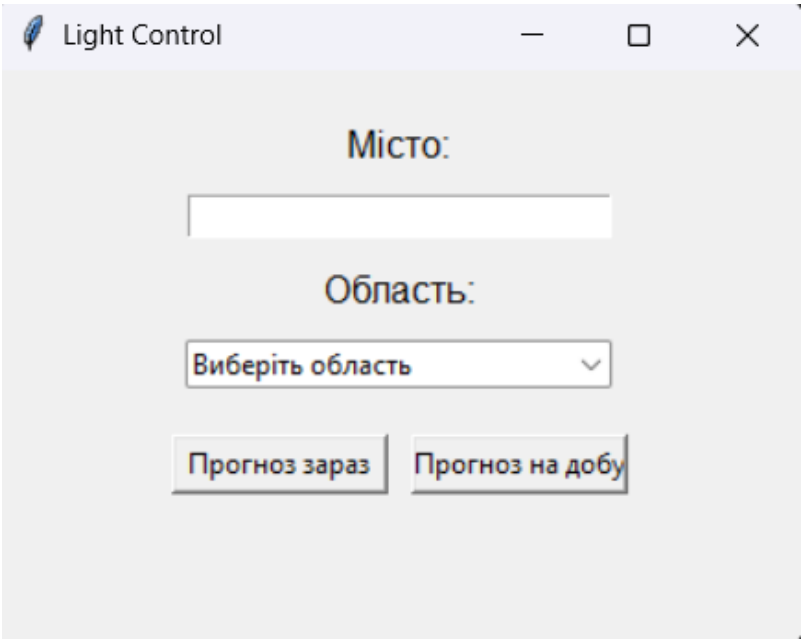


Рисунок 3.1 – Початкове вікно користувацького інтерфейсу програми

Запустивши програму перед користувачем з’являється вікно як на рис. 3.1. У інтерфейсі присутні одне поле для вводу назви населеного пункту ручним набором на клавіатурі, одне поле зі списком, який випадає, для вибору області та дві кнопки (рис. 3.2):

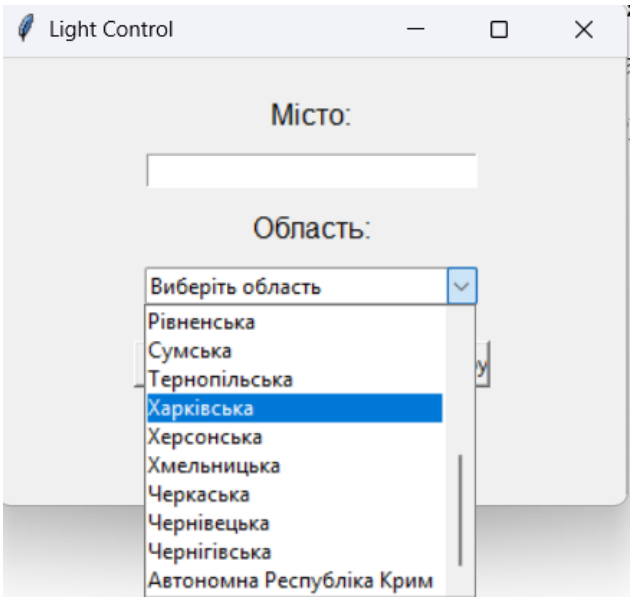


Рисунок 3.2 – Список, який випадає, для вибору області

Загальний алгоритм роботи користувача у програмі можна описати наступним набором кроків:

1. Користувач запускає програму;
2. Користувач обирає область;
3. Користувач вводить назву населеного пункту;
4. Користувач натискає на кнопку прогнозу.

Для користувача доступні дві кнопки для запуску функцій прогнозу.

Перша кнопка «Прогноз зараз» запускає функцію, яка робить один запит до API та повертає значення прогнозу на поточний момент часу (3.3):

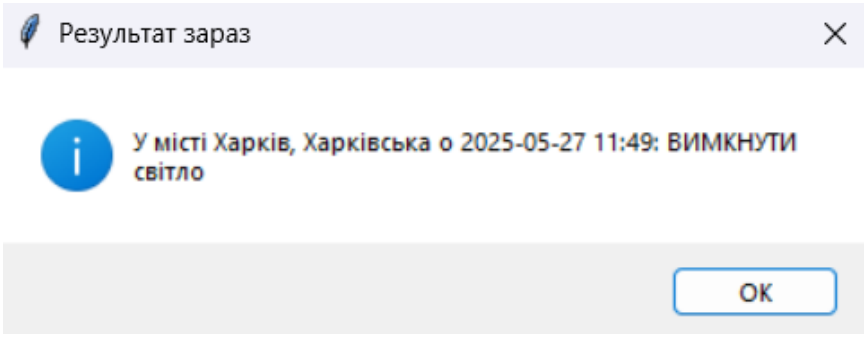


Рисунок 3.3 – Вікно результату за натисканням кнопки «Прогноз зараз»

Користувач отримує нове вікно з повідомленням (рис. 3.3), або так званий MessageBox у якому може подивитися результат прогнозу для заданого населеного пункту, області, дати та часу.

Друга кнопка «прогноз на добу» робить запит кожну годину на наступні 24 години та виводить користувачу таблицю у якій зазначається прогноз нейронної мережі для кожного інтервалу у 10 хвилин (рис. 3.4):

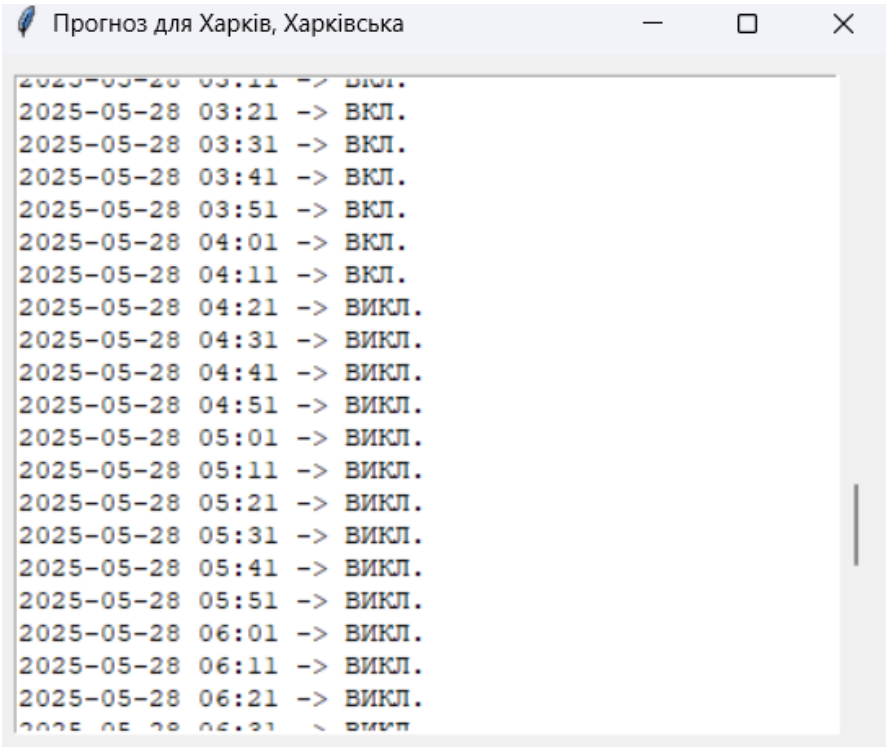


Рисунок 3.4 – Вікно з прогнозом на наступну добу

Користувач також отримує нове вікно з повідомленням, але тут вже відображається весь список прогнозів на наступну добу. Зазначається дата та час для кожного прогнозу, щоб користувач міг легше орієнтуватися у цьому масиві даних. Якщо провести математичні розрахунки, то отримаємо, що для кроку у 10 хвилин програма виводить 144 прогнози, тому такий принцип ідентифікації кожного прогнозу досить обґрунтований. Моменти часу у які нейронна мережа вважає, що потрібно ввімкнути світло позначені міткою «ВКЛ», а моменти часу у які потрібно вимкнути світло відповідно «ВИКЛ».

### 3.2 Тестування програмного продукту

Тест № 1: Введемо назву населеного пункту, якого не існує у певній області.

Нехай населений пункт – Хубей. Область – Вінницька область. Очікуваний результат – «Не вдалося знайти місто Хубей в області Вінницька» (рис. 3.5 та 3.6):

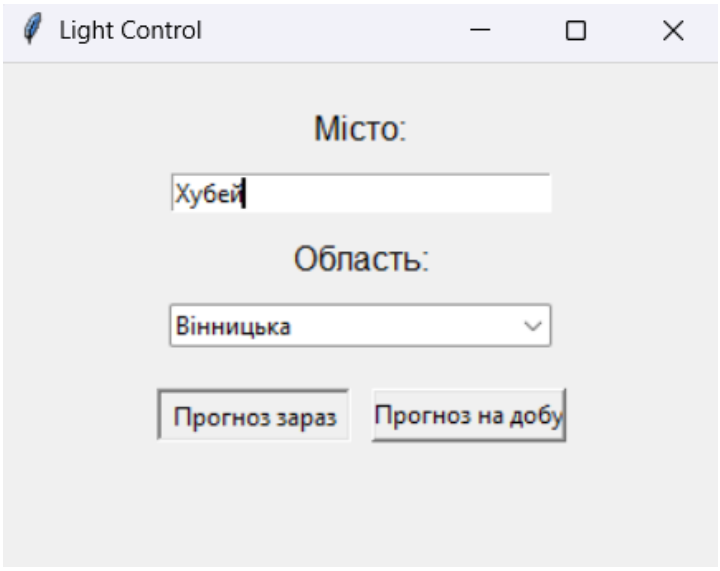


Рисунок 3.5 – Ввід даних для тесту № 1

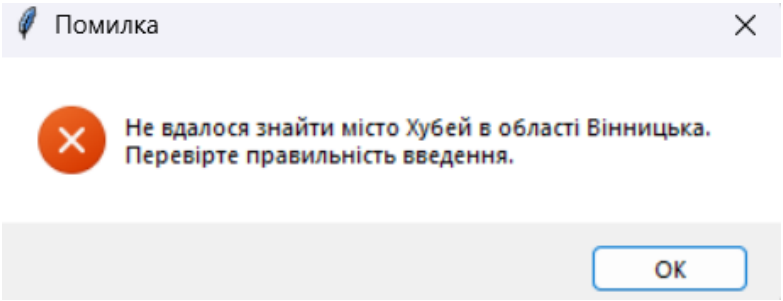


Рисунок 3.6 – Результат тесту № 1

Бачимо, що очікування від вводу неправильних даних відповідають дійсності. Програма не завершила свою роботу після неправильного вводу

даних та дає можливість виправити помилку. Цей тест однаково працює і для «Прогноз зараз», і для «Прогноз на добу».

Тест № 2: Запуск виконання програми без доступу до мережі Інтернет.

Враховуючи, що програмний виріб використовує з'єднання з віддаленими серверами для отримання стану погоди та географічних координат населеного пункту вважається важливим перевірити, як працює програма в умовах відсутності доступу до мережі Інтернет.

Введемо деякі дані та запустимо виконання програми. Отримали наступне повідомлення (рис. 3.7):


Рисунок 3.7 – Результат тесту № 2

Програма повідомила користувача про те, що HTTP запит до API не пройшов у відповідному повідомленні. Програма не закрилась та готова до повторного відправлення запиту, як тільки з'єднання з мережею Інтернет відновиться.

Після відновлення з'єднання з мережею Інтернет програма працює у нормальному режимі (рис. 3.8):

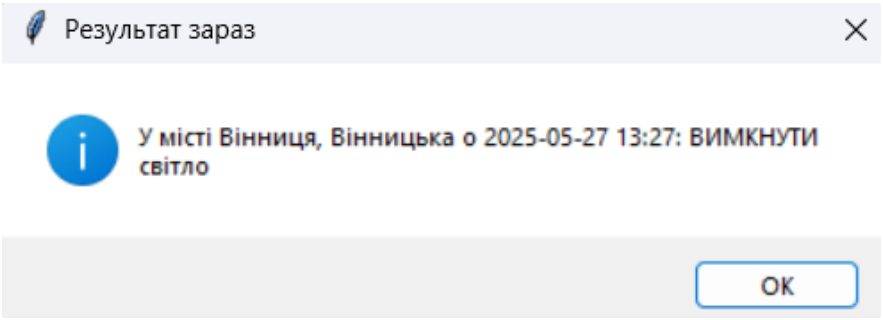


Рисунок 3.8 – Результат тесту № 2 після відновлення з'єднання з мережею Інтернет

Тест № 3: Перевірка дійсності прогнозу нейронної мережі.

Варто перевірити чи дійсно нейронна мережа правильно розраховує час роботи міської системи комунального освітлення. Для цього проведемо два розрахунка для міста Вінниця, Вінницької області. Порівнювати результати будемо з даними сайту timeanddate.com та функцію *sun* на цьому сайті, яка дозволяє передивитися повну характеристику Сонця на конкретний день. Існує такий термін як громадські сутінки, або civil twilight. Очікуємо, що «Прогноз на добу» буде вимикати світло у період ранкових громадських сутінок, а вмикати у період вечірніх громадських сутінок. Також очікуємо, що «Прогноз зараз» на момент часу 27 травня 2025 року 13:38 повідомить про необхідність вимкнути світло.

Почнемо з «Прогноз зараз». Було введено: населений пункт – Вінниця, область Вінницька (рис. 3.9):

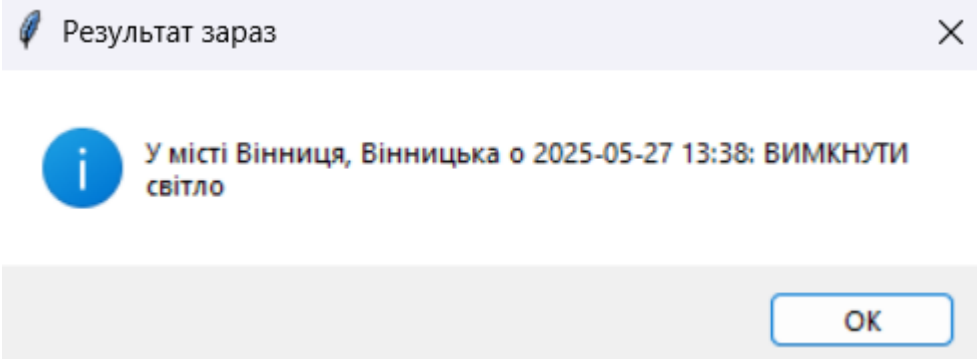


Рисунок 3.9 – Результат тесту № 3 для «Прогноз зараз»

Бачимо, що у випадку з «Прогноз зараз» очікування від програми та дійсний порядок речей співпали – програма повідомила про необхідність вимкнути світло.

Протестуємо «Прогноз на добу». Було введено: населений пункт – Вінниця, область Вінницька.

Результат тесту № 3 для вечірнього вимкнення світла наведений на рис.3.10, а для ранкового вимкнення світла – на рис. 3.11.

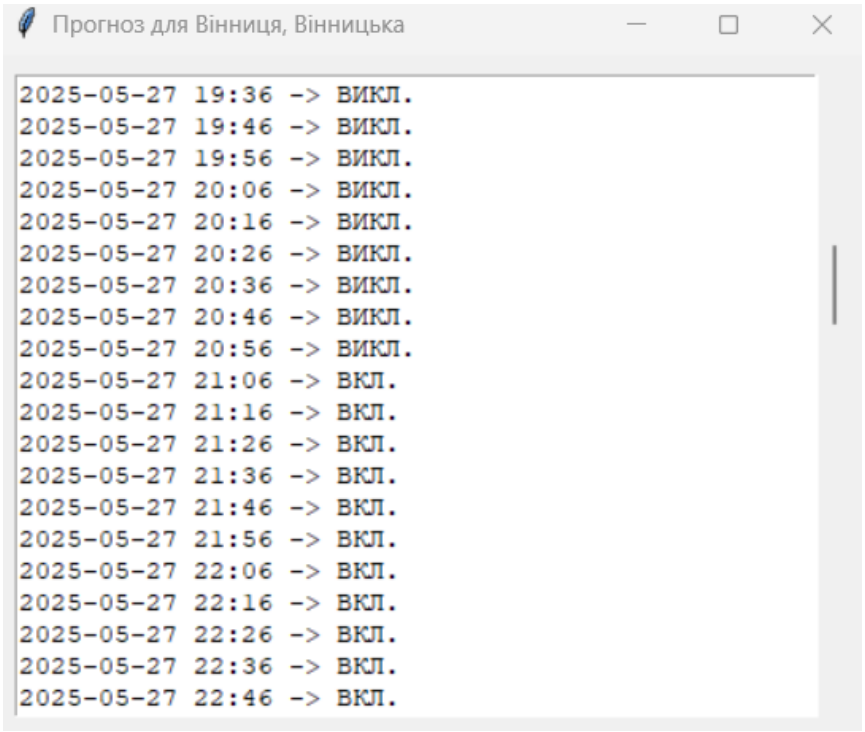


Рисунок 3.10 – Результат тесту № 3 для «Прогноз на добу» вечірнє вмикання світла

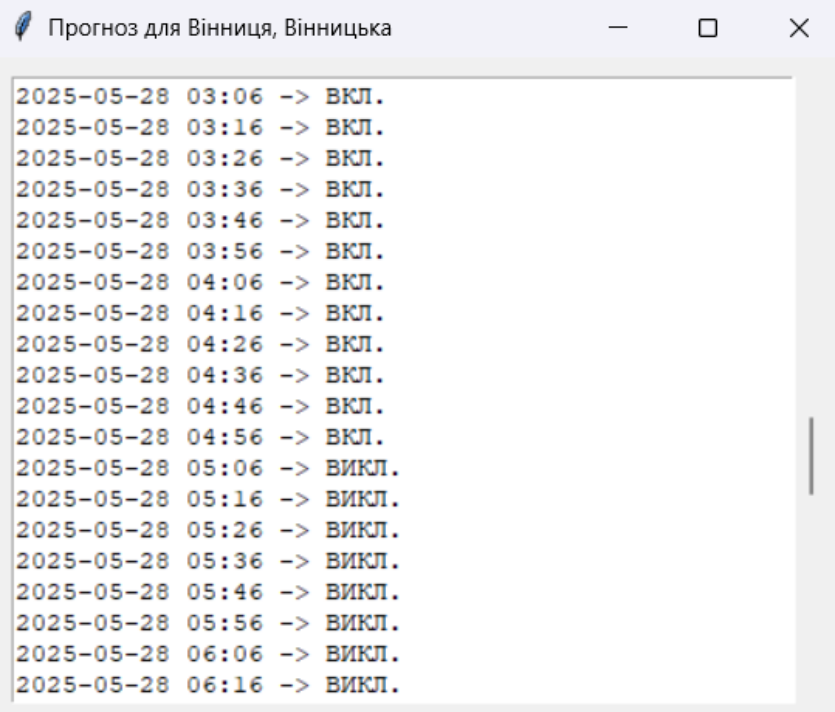


Рисунок 3.11 – Результат тесту №3 для «Прогноз на добу» ранкове вимикання світла

Зафіксуємо: вечірнє вмикання світла – 21:06; ранкове вимикання світла – 05:06. Порівняємо ці значення із даними сайту timeanddate.com (3.12):

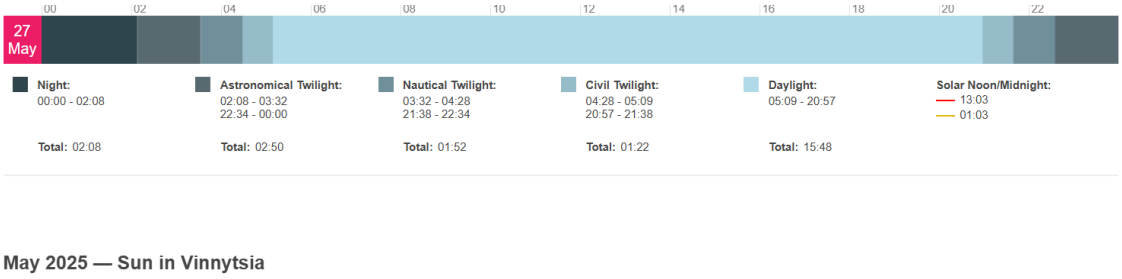


Рисунок 3.12 – Дані сайту timeanddate.com

Можна побачити, що обидва зафіксовані числа входять у період часу, який називають громадськими сутінками. З чого можна зробити висновок, що обчислення проходять коректно та співпадають з дійсними значеннями.

### 3.3 Економічний вплив

Для оцінки економічного впливу проведемо заміри. По-перше потрібно визначити час вмикання та вимикання світла у місті Мелітополь, Запорізької області. Застосувавши метод спостереження можемо сказати, що ранкове вимикання світла відбувається у 4:30, а вечірнє вмикання світла відбувається у 21:30. Ці часові мітки актуальні лише з другої половини квітня та до кінця липня.

Для проведення контрольних замірів та обчислень економічного впливу візьмемо проміжок часу з 28.04.2025 до 28.05.2025. Це тридцять один день. Протягом цього часу будемо запускати програму кожного дня приблизно у 13:00 для того щоб гарантувати, що програма захопить і вечірнє вмикання світла і ранкове вмикання світла. Побудуємо Таблицю 3.1:

*Таблиця 3.1*

Порівняльний аналіз комунальних та прогнозованих графіків увімкнення і вимкнення міського освітлення

| Дата | Комунальне вимикання | Прогнозоване вимикання | Комунальне вмикання | Прогнозоване вмикання |
|---|---|---|---|---|
| 28.04.2025 | 4:30 | 4:26 | 21:15:00 | 21:32 |
| 29.04.2025 | 4:30 | 4:29 | 21:15:00 | 21:31 |
| 30.04.2025 | 4:30 | 4:27 | 21:15:00 | 21:34 |
| 01.05.2025 | 4:30 | 4:26 | 21:15:00 | 21:38 |
| 02.05.2025 | 4:30 | 4:24 | 21:15:00 | 21:37 |
| 03.05.2025 | 4:30 | 4:25 | 21:15:00 | 21:37 |
| 04.05.2025 | 4:30 | 4:23 | 21:15:00 | 21:34 |
| 05.05.2025 | 4:30 | 4:35 | 21:15:00 | 21:24 |
| 06.05.2025 | 4:30 | 4:36 | 21:15:00 | 21:23 |
| 07.05.2025 | 4:30 | 4:25 | 21:15:00 | 21:38 |

Продовження таблиці 3.1

| | | | | |
|---|---|---|---|---|
| 08.05.2025 | 4:30 | 4:23 | 21:15:00 | 21:36 |
| 09.05.2025 | 4:30 | 4:24 | 21:15:00 | 21:35 |
| 10.05.2025 | 4:30 | 4:26 | 21:15:00 | 21:33 |
| 11.05.2025 | 4:30 | 4:23 | 21:15:00 | 21:17 |
| 12.05.2025 | 4:30 | 4:26 | 21:15:00 | 21:35 |
| 13.05.2025 | 4:30 | 4:23 | 21:15:00 | 21:37 |
| 14.05.2025 | 4:30 | 4:24 | 21:15:00 | 21:40 |
| 15.05.2025 | 4:30 | 4:27 | 21:15:00 | 21:38 |
| 16.05.2025 | 4:30 | 4:32 | 21:15:00 | 21:25 |
| 17.05.2025 | 4:30 | 4:37 | 21:15:00 | 21:40 |
| 18.05.2025 | 4:30 | 4:21 | 21:15:00 | 21:42 |
| 19.05.2025 | 4:30 | 4:25 | 21:15:00 | 21:40 |
| 20.05.2025 | 4:30 | 4:21 | 21:15:00 | 21:44 |
| 21.05.2025 | 4:30 | 4:22 | 21:15:00 | 21:42 |
| 22.05.2025 | 4:30 | 4:23 | 21:15:00 | 21:43 |
| 23.05.2025 | 4:30 | 4:21 | 21:15:00 | 21:37 |
| 24.05.2025 | 4:30 | 4:22 | 21:15:00 | 21:38 |
| 25.05.2025 | 4:30 | 4:21 | 21:15:00 | 21:44 |
| 26.05.2025 | 4:30 | 4:15 | 21:15:00 | 21:44 |
| 27.05.2025 | 4:30 | 4:14 | 21:15:00 | 21:43 |
| 28.05.2025 | 4:30 | 4:42 | 21:15:00 | 21:42 |

Після побудови отримали таблицю з 5 колонками: [Дата], [Комунальне вимикання], [Прогнозоване вимикання], [Комунальне вмикання], [Прогнозоване вмикання].

Тепер обчислимо скільки хвилин економимо для кожного дня. Для цього будемо віднімати від комунального вимикання прогнозоване та від прогнозованого вмикання комунальне. Після чого порахуємо суму. Отримали,

що загальна економія на ранковому вимиканні становить 142 хвилини, загальна економія на вечірньому вмиканні становить 658 хвилин, що в цілому складає 800 хвилин за один місяць, або 13,33 годин на місяць.

Для подальших розрахунків візьмемо ліхтар вуличного освітлення *LED-Straßenleuchte PRO, 80 W, 12000 lm, 5000 K , IP66*. Подібні зазвичай використовують у системах міського комунального освітлення. Знаємо, що цей ліхтар витрачає на годину 80 W електроенергії, тобто для одного такого ліхтаря економія на місяць буде складати 1066 W. Відповідно для 1000 таких ліхтарів економія складатиме 1066 kW електроенергії.

Візьмемо невеликий населений пункт наприклад Лозова, Харківської області. За допомогою сервісу GoogleMaps виміряємо довжину відрізка вулиці Миру, що входить до складу населеного пункту, вона складає 2200 метрів. Розміщення ліхтарів на цій вулиці одностороннє. За ДБН «Природне і штучне освітлення» [18] відстань при односторонньому розміщенні складає 25-30м для опори висотою у 6м. Візьмемо відстань у 30м. Розділимо 2200 на 30 та отримаємо, що на вулиці Миру встановлено 73 ліхтарі.

Використовуючи дані контрольних замірів обчислили, що світло горить на вулиці 7,25 годин, тоді можемо обчислити, що без застосування розробленої програми кожен ліхтар буде працювати приблизно 225 годин у місяць, а 73 ліхтарі за місяць споживатимуть 1 312 540 W. Якщо ж застосувати розроблену програму, то кожен ліхтар за місяць буде працювати 212 годин, а 73 ліхтарі за місяць споживатимуть 1 236 620 W. Різниця складатиме 75 920 W, або 5,78%.

Візьмемо ціну у 4,32 грн/kW·год, виходить, що за один місяць на одній вулиці буде зекономлено 328 гривень. Якщо ж масштабувати роботу програми наприклад на довжину вулиць у 100км, що зазвичай відповідає довжину вулиць невеликих містечок, то отримаємо, що тільки за один місяць громада зекономить 14 974 гривні. І процент економії зростає при використанні більш потужних приладів освітлення.

**Висновки до розділу 3**

У розділі три було детально описано користувацький інтерфейс розробленої програми. Проведено низку тестів, які показали, що програма стійка до неправильного вводу та ненадійного з’єднання з мережею інтернет. Також було проаналізовано теоретичний економічний вплив, який може здійснити розроблена програма на прикладі одного відрізку вулиці у реальному місті України.

# ВИСНОВКИ

У кваліфікаційній роботі було розроблено комп'ютерну модель для управління мережею міського освітлення з використанням нейронних мереж, що є актуальним рішенням у контексті підвищення енергоефективності та економії ресурсів у сучасних містах. Робота охопила всі ключові етапи: від теоретичного аналізу предметної області до практичної реалізації моделі та оцінки її економічного ефекту.

Розробка моделі є актуальною через зростаючу потребу в зменшенні енергоспоживання міських систем, зокрема комунального освітлення, що дозволяє зменшити витрати та підвищити ефективність використання ресурсів. Робота полягає у створенні адаптивної системи на основі багатошарового перцептрону (MLP), яка враховує зовнішні фактори (погоду, час, дату, географічну широту) та використовує згенерований вручну набір даних із застосуванням калібрувальної моделі для підвищення точності прогнозів. Практичне значення роботи полягає в розробці програмного продукту, який може бути інтегрований у муніципальні служби та системи "розумного міста" для автоматизації управління освітленням.

У першому розділі проведено аналіз існуючих типів нейронних мереж, серед яких обґрунтовано вибір багатошарового перцептрону (MLP) як оптимальної архітектури для задачі. Вибір базується на:

- Ефективності роботи з табличними даними, що відповідає формату вхідних параметрів (час, дата, погода, широта);
- Низьких обчислювальних потребах, що забезпечує можливість запуску моделі на малопотужних пристроях;
- Гнучкості налаштування, що дозволяє адаптувати модель до змін умов.

Також розглянуто господарський вплив моделі, який включає економію електроенергії, підвищення безпеки та комфорту мешканців, а також

зменшення екологічного навантаження завдяки скороченню світлового забруднення та викидів $CO_2$.

У другому розділі описано створення навчального набору даних, який через відсутність відповідних відкритих джерел було згенеровано вручну за допомогою Python. Процес включав в себе генерацію даних, де було враховано чотири параметри (погода, час, дата, географічна широта), що впливають на потребу в освітленні та розмітку, під час якої було розроблено алгоритм обчислення часу сходу та заходу Сонця з корекцією на погодні умови та літній час. Для підвищення точності визначення часу заходу Сонця створено калібрувальну модель на основі лінійної регресії, яка використовує історичні дані.

Архітектура нейронної мережі (MLP) складається з чотирьох шарів (64, 32, 16 нейронів із функцією активації ReLU та вихідний шар із сигмоїдною функцією). Проведено порівняння різних гіперпараметрів, що дозволило обрати оптимальну конфігурацію з точністю прогнозування понад 95% за такими метриками як: Accuracy, Precision, Recall та F1-Score. Розроблено користувацьку програму, яка інтегрує дані з GeocodingAPI та OpenWeatherAPI та видає прогнози в реальному часі або на добу вперед із кроком у 10 хвилин.

У третьому розділі продемонстровано користувацький інтерфейс, реалізований за допомогою бібліотеки tkinter, який дозволяє вводити назву міста та області, отримувати поточний прогноз або графік на 24 години. Тестування показало, що програма правильно обробляє неправильні дані (наприклад, неіснуючі міста) та відсутність доступу до Інтернету, видаючи відповідні повідомлення. А порівняння з реальними даними (timeanddate.com) підтвердило відповідність часу вмикання/вимикання періодам громадських сутінок.

Економічний ефект

Аналіз економічного впливу проведено на прикладі міста Мелітополь за період із 28.04.2025 по 28.05.2025. Результати показали:

- Економія часу роботи освітлення склала 13,33 години на місяць (800 хвилин);
- Для одного ліхтаря (80 Вт) економія становить 1066 Вт/місяць, а для 1000 ліхтарів – 1066 кВт;
- На прикладі вулиці довжиною 2200 м (73 ліхтарі) економія склала 75,92 кВт (5,78% від загального споживання на місяць), що еквівалентно 328 грн за ціною 4,32 грн/кВт·год;

При масштабуванні на 100 км вулиць економія досягає 14 974 грн/місяць, що підтверджує значний потенціал моделі для муніципальних бюджетів.

Розроблена комп'ютерна модель є ефективним інструментом для кращого управління міським освітленням. Вона демонструє високу точність прогнозування, адаптивність до змінних умов та практичну застосовність. Модель має потенціал для впровадження в реальні системи, сприяючи економії ресурсів, підвищенню енергоефективності та покращенню міського середовища. Подальший розвиток може включати інтеграцію з датчиками реального часу та розширення функціоналу для інших аспектів "розумного міста". Звичайно, що у розробленої моделі є слабкі сторони, але сильні сторони та можливості до майбутніх покращень загалом перевершують недоліки.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Annals of the History of Computing. IEEE. 1992. Volume: 14, Issue: 3.(дата звернення – 03.03.2025).
2. Turing A. Computing Machinery and Intelligence. 1950.(дата звернення – 26.02.2025).
3. Murphy K. P. Probabilistic Machine Learning: An Introduction. 2022.(дата звернення – 12.03.2025).
4. Almeida L. B. Multilayer perceptrons. Handbook of Neural Computation. 1996. (дата звернення – 16.03.2025).
5. Bluhm R., Krause M. Top lights: Bright cities and their contribution to economic development. 2022. (дата звернення – 16.03.2025).
6. Cozens P. A Critical Review of Street Lighting, Crime and Fear of Crime in the British City. Crime Prevention and Community Safety: An International Journal. 2003. [Електронний ресурс] Режим доступу URL: https://doi.org/10.1057/PALGRAVE.CPCS.8140143. ( дата звернення – 20.03.2025).
7. 7. Welsh B. C., Farrington D. P. effects of improved street lighting on crime. 2008. (дата звернення – 15.04.2025).
8. Zielinska-Dabkowska K. M., Bobkowska K. Rethinking Sustainable Cities at Night: Paradigm Shifts in Urban Design and City Lighting. 2022.( дата звернення – 21.04.2025).
9. Elgiriyewithana N. World Weather Repository. [Електронний ресурс] Режим доступу URL: https://www.kaggle.com/datasets/nelgiriyewithana/global-weather-repository.
10. Supervised learning. scklit-learn. [Електронний ресурс] Режим доступу URL: https://scikit-

learn.org/stable/modules/svm.html#regression. ( дата звернення – 26.04.2025).

11. Wythoff B. J. Backpropagation neural networks: A tutorial. Chemometrics and Intelligent Laboratory Systems. 1993. Volume 18, Issue 2 (дата звернення – 27.04.2025).

12. A survey on dataset quality in machine learning / Y. Gong et al. Information and Software Technology. 2023. Volume 162 (дата звернення – 28.04.2025).

13. Kagali B. A., Shivalingaswamy T. Determination of the Declination of the Sun on a Given Day. European Journal of Physics Education. 2011. Volume 3, Issue 1 (дата звернення – 30.04.2025).

14. Srivastava N., Hinton G. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014 (дата звернення – 04.05.2025).

15. Srivastava N. Improving Neural Networks with Dropout : Masters degree. Toronto, 2013. [Електронний ресурс] Режим доступу URL: https://www.cs.toronto.edu/~nitish/msc_thesis.pdf (дата звернення - 05.05.2025).

16. Gu B., Sung Y. Enhanced Reinforcement Learning Method Combining One-Hot Encoding-Based Vectors for CNN-Based Alternative High-Level Decisions. Advanced Intelligent Imaging Technology II. 2021. [Електронний ресурс] Режим доступу URL: https://doi.org/10.3390/app11031291 (дата звернення - 05.05.2025).

17. Geocoding API overview. developers.google. [Електронний ресурс] Режим доступу URL: https://developers.google.com/maps/documentation/geocoding/overview (дата звернення - 08.05.2025).

18. ДБН В.2.5-28-2018. Природне і штучне освітлення. На заміну ДБН В.2.5-28-2006 ; чинний від 2019-03-01. Вид. офіц. 2018 (дата звернення – 18.05.2025).

# ДОДАТКИ

**Додаток А**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) Бакалавр
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«02» жовтня 2024 року

## З А В Д А Н Н Я
## НА КВАЛІФІКАЦІЙНУ РОБОТУ

**Ісаченко Владислав Русланович**
(прізвище, ім'я, по батькові студента)

1. Тема роботи **«Комп'ютерна модель системи управління мережею міського освітлення»**

керівник роботи Чуб Ольга Ігорівна, к.е.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету ***№ 4101-5/962 від 16.04.2025***

2. Строк подання студентом роботи ***30 травня 2025 року***

3. Перелік питань, які потрібно розробити:

1. Аналіз основних методів управління інфраструктурою міського освітлення;
2. Огляд сучасних технологічних рішень для підвищення ефективності управління;
3. Дослідження підходів до вибору ефективної технологічної платформи для реалізації моделі системи управління мережею міського освітлення;
4. Аналіз наявних відкритих наборів даних, придатних для навчання моделі управління мережею міського освітлення;
5. Формування власного датасету для навчання та валідації комп'ютерної моделі системи управління міським освітленням;

6. Розробка комп’ютерної моделі системи управління мережею міського освітлення;
7. Проведення тестування моделі та оцінка її ефективності;
8. Аналіз програмних рішень для підвищення економічності та функціональної ефективності системи освітлення.

4. План роботи:

| № з/п | Назви етапів роботи | Термін виконання етапів роботи |
|---|---|---|
| 1. | Формулювання наукової проблеми, обґрунтування актуальності, мети та завдань дослідження. Визначення об’єкта, предмета та методів дослідження | 05.09.2024 – 02.10.2024 |
| 2. | Пошук та аналіз методичної літератури щодо побудови комп’ютерної моделі управління мережею міського освітлення | 02.10.2024 – 24.10.2024 |
| 3. | Огляд і порівняльний аналіз наявних програмних рішень для управління мережею міського освітлення | 25.10.2024 – 09.11.2024 |
| 4. | Аналіз технологічних платформ і обґрунтування вибору найдоцільнішої для реалізації нейронної мережі в моделі управління міським освітленням | 10.11.2024 – 29.11.2024 |
| 5. | Пошук набору даних для тренування нейромережі для комп’ютерної моделі | 30.11.2024 – 05.01.2025 |
| 6. | Формування власного датасету для навчання та валідації комп’ютерної моделі | 06.01.2025 – 28.01.2025 |
| 7. | Розробка комп’ютерної моделі для управління мережею міського освітлення на основі нейромережі | 29.01.2025 – 01.02.2025 |
| 8. | Тестування та оцінка ефективності розробленої комп’ютерної моделі | 02.02.2025 – 07.02.2025 |
| 9. | Аналіз програмних рішень для підвищення економічності та функціональної ефективності системи освітлення | 08.02.2025 – 13.02.2025 |
| 10. | Виклад матеріалу кваліфікаційної роботи | 14.02.2025 – 30.04.2025 |
| 11. | Оформлення звіту про переддипломну практику | 01.05.2025 – 20.05.2025 |
| 12. | Передзахист кваліфікаційної роботи | 20.05.2025 – 28.05.2025 |
| 13. | Остаточне оформлення пояснювальної записки та проходження перевірки на антиплагіат | до 30.05.2025 |

5. Дата видачі завдання ***02 жовтня 2024 року.***

Студент В.Р. Ісаченко

Керівник роботи О.І. Чуб

ДОДАТОК Б

Затверджую

_____

«_____»_____2024 р.

**Технічне завдання**
**на розробку програмного виробу**

**«Комп'ютерна модель системи управління мережею міського освітлення»**

| Назва розділу | Назва та зміст підрозділу |
|---|---|
| 1. Вступ | 1.1. Назва програмного виробу:<br>«Комп'ютерна модель системи управління мережею міського освітлення»<br><br>1.2. Галузь застосування:<br>Комп’ютерна модель системи управління мережею міського освітлення може бути застосовано у наступних галузях людської діяльності: міське управління, інтеграція в «розумні міста», енергетичний сектор, транспортна інфраструктура, сільське господарство, приватні домогосподарства, паркове керування |
| 2. Підстава для розробки | 2.1. Навчальний план за спеціальністю:<br>123 – Комп’ютерна інженерія<br><br>2.2. Завдання на кваліфікаційну роботу бакалавра<br>***№ 4101-5/962 від «16» квітня 2025*** |
| 3. Призначення розробки | 3.1. Мета розробки програмного виробу:<br>Метою розробки є створення застосунку, який покликаний покращити графік вимкнення міського освітлення, використовуючи нейронну мережу, GeocodingAPI та OpenWeatherAPI. Застосунок надаватиме рекомендації щодо графіку роботи мережи міського управління.<br><br>3.2. Призначення програмного виробу:<br>Цей програмний виріб призначений для покращення графіків роботи міського освітлення, зменшення впливу суб’єктивної оцінки та людської помилки, зменшення споживання електроенергії мережею міського освітлення та збільшенням безпечності нічного міста.<br><br>3.3. Вихідні дані для розробки :<br>• Набір даних для тренування комп’ютерної моделі;<br>• Використання технологій tensorflow для забезпечення стабільного та якісного навчання комп’ютерної моделі;<br>• Використання технологій GeocodingAPI та OpenWeatherAPi. |
| 4. Технічні вимоги до | 4.1. Вимоги до функціональних характеристик:<br>• Програмний виріб повинен автоматично будувати рекомендаційні графіки роботи системи міського освітлення; |

| | |
|---|---|
| програмного виробу | • Користувач повинен мати можливість вказати місто для якого повинен створюватися графік роботи системи міського освітлення;<br>• Система має забезпечувати класифікацію навколишнього середовища в залежності від чотирьох змінних: стан погоди, час, дата, географічна широта;<br>• Після класифікації має виводитись рекомендація для користувача;<br>• Час класифікації не повинен перевищувати 5 секунд.<br><br>4.2. Вимоги до надійності:<br>• Модель має бути стійкою до неправильного введення даних користувачем;<br>• Обробка винятків у випадку відсутності ключа API, або інших проблем зв'язаних з доступом до мережі Інтернет;<br>• Програмний виріб має зберігати цілісність моделей та вихідних даних навіть у разі збоїв у роботі.<br><br>4.3. Вимоги до умов експлуатації:<br>• Кліматичні умови: Температура навколишнього середовища: від +10°C до +35°C. Відносна вологість: від 20% до 80%, без конденсації вологи;<br>• Програма та модель можуть бути завантажені з репозиторію. Для запуску програми достатньо завантажити архів з програмним забезпеченням та розпакувати його на локальний диск;<br>• Запуск програми здійснюється безпосередньо з локального комп'ютера без потреби в додаткових налаштуваннях або підключеннях до серверів;<br>• Після завантаження програми користувач може одразу почати використовувати модель для побудови графіків роботи міської системи освітлення;<br>• Програма працює в онлайн-режимі, але не потребує постійного доступу до мережі, та зберігає передбачення на добу вперед;<br>• Для користування програмою користувачеві необхідно прочитати інструкцію використання.<br><br>4.4. Вимоги до складу параметрів технічних засобів:<br>• Для виконання програми повинен підходити ПК із будь-якою операційною системою сімейства Windows, Linux/Unix, Mac OS X, OS/2, Amiga. Крім того, для роботи необхідний інтерпретатор Python версії 3.10 або 3.11 та завантажена бібліотека tensorflow;<br>• Для користування мінімальними вимогами є підключення комп'ютера до електромережі, мінімальна оперативна пам'ять 2 ГБ, дисковий простір 1 ГБ, процесор 3.0-3.5 ГГц.<br><br>4.5. Вимоги до інформаційної та програмної сумісності:<br>Програма працює автономно під керуванням сучасних ОС: Windows 10/11, Linux (Ubuntu 20.04+), macOS 10.15+. Потрібен Python 3.10-3.11 (або сумісних) та бібліотеки pandas tensorflow тощо. |

| | |
|---|---|
| | Вхідні дані – «словник» вигляду :<br>"weather": <string>,<br>"day_of_year": <int>,<br>"time": <string>,<br>"Latitude": <float><br><br>4.6. Вимоги до маркування та упаковки<br>Вимоги до маркування та упаковки не пред'являються, оскільки програмний продукт не потребує спеціальної упаковки для навчального чи дослідницького використання.<br><br>4.7. Вимоги до транспортування та зберігання<br>Вимоги до транспортування та зберігання не пред'являються, оскільки програмний продукт є цифровим і не потребує фізичного транспортування чи спеціальних умов зберігання.<br><br>4.8. Спеціальні вимоги<br>Спеціальних вимог до програмного виробу не пред'являються. Програмний продукт є стандартним додатком для класифікації архітектурних стилів та не потребує додаткових специфікацій або унікальних умов для використання. |
| 5. Вимоги до програмної документації | Програмною документацією до виробу «Комп'ютерна модель системи управління мережею міського освітлення» вважати:<br>1) Це Технічне завдання на розробку програмного виробу (представити у вигляді Додатка Б до пояснювальної записки до кваліфікаційної роботи);<br>2)Програму та методику випробувань розробленого програмного виробу (представити у вигляді Додатка В до пояснювальної записки до кваліфікаційної роботи);<br>3) Опис програмного виробу (представити в розділі пояснювальної записки до кваліфікаційної роботи);<br>4) Текст програми (представити в Додатку Г до пояснювальної записки до кваліфікаційної роботи). |
| 6. Техніко-економічні показники | 1) Планується досягти точності на рівні близько 95%, що є конкурентним показником з огляду на складність розмежування стилів. Для порівняння, аналогічні рішення демонструють точність у межах 90–97%.<br>2) Розробка моделі триватиме орієнтовно 3 місяці. Планується використовувати безкоштовне програмне забезпечення та доступне апаратне забезпечення. У разі масштабування – можливі витрати на продуктивніші графічні процесори.<br>3) Очікується, що модель матиме подібний або вищий рівень ефективності, ніж наявні рішення. Вона буде доступнішою для інтеграції в освітні, комунальні та дослідницькі застосунки завдяки спрощеній реалізації. |
| 7. Стадії та етапи розробки | Розробка моделі системи управління мережею міського освітлення проходить у три основні стадії відповідно до ДЕСТ 19.102-77:<br>1) Розробка технічного завдання – формування вимог, постановка задачі, погодження з керівником. |

| | |
|---|---|
| | 2) Робоче проектування – включає програмування, налагодження, розробку документації та випробування.<br>3) Впровадження розробленого виробу – підготовка, передача програми та документації до експлуатації.<br><br>Етапи:<br>1) Розробка програми (створення і підготовка датасету, створення моделі, тренування та валідація моделі, коригування за потреби, тестування, створення звітів з аналітичними результатами роботи моделі);<br>2) Розробка програмної документації (створення пояснювальної записки, візуалізацій результатів);<br>3) Впровадження (випробування) програми (передача готової моделі для практичного або навчального використання). |
| 8. Порядок контролю та приймання | 1. Перевірку ходу розроблення заданих проектних документів керівнику курсової роботи здійснювати 1 раз на 3 тижні;<br>2. Випробування програмного виробу;<br>3. Захист розробленого програмного виробу провести на засіданні ДЕК;<br>4. Розроблені проектні документи подати в електронному вигляді та на паперових носіях в одному примірнику. |

**Виконавець:**
студент групи КІ-41
ІСАЧЕНКО Владислав Русланович

**Замовник:**
к.е.н., доцент кафедри комп'ютерних систем та робототехніки
ЧУБ Ольга Ігорівна

ДОДАТОК В

# Програма та методика випробувань програмного виробу «Комп'ютерна модель системи управління мережею міського освітлення»

## 1. Об'єкт випробувань

1.1. Найменування програмного виробу: «Комп'ютерна модель системи управління мережею міського освітлення».

1.2. Область застосування:

Комп’ютерна модель системи управління мережею міського освітлення може бути застосовано у наступних галузях людської діяльності: міське управління, інтеграція в «розумні міста», енергетичний сектор, транспортна інфраструктура, сільське господарство, приватні домогосподарства, паркове керування.

1.3. Умовне призначення розробки (за потреби)

Цей програмний виріб призначений для надання рекомендацій щодо графіку роботи системи міського освітлення.

## 2. Мета випробувань

Метою випробувань є перевірка відповідності функціональних характеристик розробленого програмного виробу вимогам, зазначеним у Технічному завданні. Зокрема, це підтвердження здатності моделі надавати рекомендації для графіку роботи мережі міського освітлення, враховуючи зовнішні фактори, такі як погода, час, дата, географічна широта. Випробування мають на меті також оцінку ефективності використання обраної архітектури комп’ютерної моделі.

## 3. Загальні положення

3.1. Підстави для проведення випробувань: Наказ по університету про призначення Атестаційної комісії № 44.

3.2. Місце і тривалість випробувань: Випробування проводяться у віртуальному лабораторному середовищі протягом роботи Атестаційної комісії № 44.

3.3. Обсяг випробувань: Приймання та тестування програмного виробу проводяться в обсягах, які відповідають пункту 3.2 цієї Програми.

3.4. Організації, які беруть участь у випробуваннях: Приймання проводиться в процесі засідання Атестаційної комісії № 44 за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

**4. Вимоги до програмного виробу**

1. Програмний виріб повинен автоматично виконувати класифікацію стану навколишнього середовища у місті застосування;
2. Користувач повинен мати можливість вказати місто свого знаходження;
3. Система має забезпечувати класифікацію навколишнього середовища в залежності від чотирьох змінних: стан погоди, час, дата, географічна широта;
4. Після класифікації має виводитись рекомендація до терміналу застосунка;
5. Час класифікації не повинен перевищувати 5 секунд.

**5. Вимоги до програмної документації**

Програмною документацією до виробу «Комп'ютерна модель системи управління мережею міського освітлення» вважати:

1. Це Технічне завдання на розробку програмного виробу (представити у вигляді Додатка Б до пояснювальної записки до кваліфікаційної роботи);
2. Програму та методику випробувань розробленого програмного виробу (представити у вигляді Додатка В до пояснювальної записки до кваліфікаційної роботи);
3. Опис програмного виробу (представити в розділі пояснювальної записки до кваліфікаційної роботи);
4. Текст програми (представити в Додатку Г до пояснювальної записки до кваліфікаційної роботи).

**6. Засоби та порядок випробувань**

6.1. Засоби випробувань

Для проведення випробувань буде використовуватись наступне технічне та програмне забезпечення.

Технічні засоби:

- Сервери або робочі станції з достатньою кількістю оперативної пам'яті та потужними графічними процесорами (GPU) для прискорення обчислень;
- Вихідні зображення архітектурних об’єктів для тестування та класифікації.

Програмні засоби:

- Python – основна мова програмування для реалізації моделі глибокого навчання;
- TensorFlow – відкрита бібліотека машинного навчання;
- Visual Studio Code (VS Code) – середовище для розробки, в якому буде здійснюватися написання коду та його виконання;
- TensorBoard – для візуалізації процесу навчання моделі, включаючи побудову графіків точності та втрат, а також аналіз показників моделі під час навчання. Використовуватиметься через командний рядок або у VS Code для аналізу навчання моделі;
- NumPy, Pandas – для обробки даних та аналізу результатів;
- Matplotlib – для візуалізації результатів тестування та графічного представлення ефективності моделі;
- OpenWeatherAPI – для доступу до метеорологічних даних у реальному часі для всього земного шару.

Програма тестування буде надаватися у вигляді інсталяційної версії розробленого програмного продукту з документацією, яка містить інструкції щодо налаштування середовища для запуску моделі.

## 6.2 Порядок проведення випробувань

Етап № 1 – Перевірка комплектності програмної документації

Перевіряється наявність усіх необхідних документів згідно з Технічним завданням. Перевірка комплектності технічних та програмних засобів здійснюється за критерієм відповідності їх складу вимогам середовища виконання (Visual Studio Code, TensorFlow, TensorBoard, Python 3.10, ОС Windows 10/11);

Етап № 2 – Перевірка якості програмної документації

Якість програмної документації відповідає вимогам ДЕСТ 19.301–79 ЄСПД;

Етап № 3 – Функціональне випробування програмного виробу

На цьому етапі перевіряється працездатність моделі та ступінь виконання функціональних вимог.

6.3. Перевірка працездатності програмного виробу

**Методика проведення Тесту № 1:**

VSCode використовується на етапі розробки та навчання моделі:

- Запуск тестування моделі здійснюється у середовищі Visual Studio Code командою python user.py.
- Після запуску система запитує назву міста у користувача та виконує безпосередньо передбачення.
- Програма виконує:
  - завантаження моделі нейронної мережі;
  - попередню обробку отриманих з GeocodingAPI, OpenWeatherAPI даних;
  - виконання передбачення;
  - вимір часу виконання та обсягу використаної пам’яті;
  - вивід передбаченого класу(0,1) у консоль.

Для першого тесту візьмемо місто Paarl, що знаходиться у ПАР.

Тестування проводиться 30 квітня 2025 року у 19:15.

**Очікуваний результат:**

Система повинна передбачити, що світло повинно бути увімкненим у 19:15 30 квітня. Тобто вивести 1(світло увімкнено).

**Фактичний результат:**

Фактичний результат виконання тесту № 1наведений на рис. В.1:

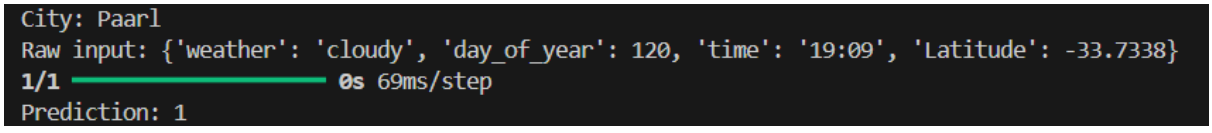


Рис. В.1 – Фактичний результат виконання тесту № 1

**Висновок:**

У результаті виконання Тесту №1 функціональна вимога щодо якості оцінювання навколишнього середовища комп'ютерною моделлю для мережі міського освітлення була виконана.

**Методика проведення Тесту № 2:**

VSCode використовується на етапі розробки та навчання моделі.

- Запуск тестування моделі здійснюється у середовищі Visual Studio Code командою python app.py;
- Після запуску система починає процес навчання моделі. Після закінчення навчання система використовує вбудовані метрики для перевірки правильності роботи нейронної моделі. Наприклад, Accuracy, Precision, Recall, F1-score та створює confusion_matrix;
- Програма виконує:
  - завантаження набору даних;
  - навчання комп'ютерної моделі;
  - порівняння результатів на тестовій та навчальній виборці;
  - складання метрик та їх вивід у консоль.

**Очікуваний результат:**

- Виведення статистики тестування у консоль;
- Відображення confusion_matrix.

**Фактичний результат:**

Фактичні результати виконання тесту № 2 наведені на рис. В.2 та В.3:

```
Test loss: 0.15106350183486938
Test accuracy: 0.9570000171661377
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Precision: 0.9516
Recall:    0.9678
F1-score:  0.9596

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95       472
           1       0.95      0.97      0.96       528

    accuracy                           0.96      1000
   macro avg       0.96      0.96      0.96      1000
weighted avg       0.96      0.96      0.96      1000
```

Рис. В.2 – Перший фактичний результат виконання тесту № 2

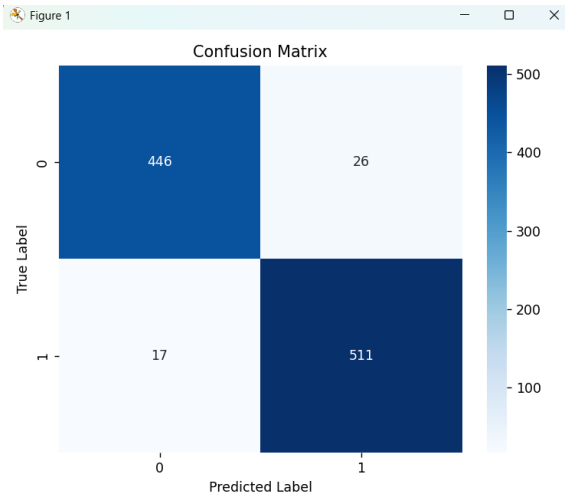


Рис. В.3 – Другий фактичний результат виконання тесту № 2

**Висновок:**

Програма успішно виконала всі етапи тестування моделі. В результаті були отримані точні метрики, такі як точність (accuracy), precision, recall та F1-score, а також був згенерований повний звіт класифікації, включаючи confusion_matrix.

Очікувані результати було досягнено.

**Висновок за результатами проведених випробувань:**

Програмний виріб вважається таким, що пройшов випробування в цілому, якщо успішно виконані Тести 1 та 2. Загалом, тестування показало, що модель працює ефективно та надає точні результати класифікації. Це дозволяє зробити висновок про її готовність до використання в реальних умовах або для подальшого доопрацювання. Рекомендується проводити подальше тестування для оцінки роботи моделі на нових даних та вдосконалення результатів.

**Виконавець:**

студент групи КІ-41

ІСАЧЕНКО Владислав Русланович

ДОДАТОК Г

ЛІСТИНГ ПРОГРАМИ

Фрагмент № 1 – labeling_table.py – Функція розмітки датасету

```
def label_row(row) -> int:
    day_of_year = int(row['day_of_year'])
    latitude = float(row['latitude'])

    # Базові часи з урахуванням DST
    computed_sunrise, computed_sunset = compute_sunrise_sunset(day_of_year, latitude)
    if is_dst(day_of_year):
        computed_sunrise += 60
        computed_sunset += 60

    # Корекція для сходу та заходу сонця
    X_pred = [[day_of_year, latitude, np.sin(2 * np.pi * day_of_year / 365), np.cos(2 * np.pi * day_of_year / 365)]]
    corrected_sunset = computed_sunset #+ model_sunset.predict(X_pred)[0]
    corrected_sunrise = computed_sunrise # + model_sunrise.predict(X_pred)[0]

    # Поправки за погодними умовами
    weather = row['weather']
    if weather in ["cloudy", "fog"]:
        weather_adj_on, weather_adj_off = -30, 30
    elif weather in ["rain", "snow"]:
        weather_adj_on, weather_adj_off = -50, 50
    elif weather in ["rainfall", "blizzard"]:
        weather_adj_on, weather_adj_off = -80, 80
```

```
    elif weather in ["partly_cloudy"]:
        weather_adj_on, weather_adj_off = -20, 20
    else:
        weather_adj_on, weather_adj_off = 0, 0

    # Ітогові часи
    final_on = corrected_sunset + weather_adj_on + 100
    final_off = corrected_sunrise + weather_adj_off - 100

    # Поточний час
    current_time = time_to_minutes(row['time'])
    if current_time is None:
        print(f"Пропущено рядок через помилку парсингу часу: {row}")
        return 0

    # Логіка ввімкнення світла
    return 1 if current_time >= final_on or current_time < final_off else 0
```

Фрагмент №2 – app.py – Фрагмент коду навчання нейронної мережі

```
# --- Структура нейронної моделі ---
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dropout(0.3),
    Dense(16, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Компіляція моделі
```

```
model.compile(optimizer=Adam(learning_rate=0.001),
loss='binary_crossentropy', metrics=['accuracy'])

# --- Навчання моделі ---
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=120, batch_size=32,
class_weight=class_weights,
                    validation_split=0.25, callbacks=[early_stopping])
```

Фрагмент № 3 – user.py – Функція прогнозу

```
def predict_lighting(raw_input):
    df = pd.DataFrame([raw_input])
    df.rename(columns={"Latitude": "latitude"}, inplace=True)

    we = encoder.transform(df[['weather']])
    we_df = pd.DataFrame(we,
columns=encoder.get_feature_names_out(['weather']))

    df['Hour'] = df['time'].str.split(':').str[0].astype(int)
    df['Minute'] = df['time'].str.split(':').str[1].astype(int)

    features = pd.concat([we_df, df[['day_of_year','Hour','Minute','latitude']]],
axis=1)
    features[numeric_cols] = scaler.transform(features[numeric_cols])

    weather_cols = [c for c in features.columns if c.startswith('weather_')]
    features[weather_cols] *= w_weather
    features['day_of_year'] *= w_day_of_year
    features[['Hour','Minute']] *= w_time
```

```
features['latitude'] *= w_latitude

pred = loaded_model.predict(features)
return int((pred>0.5)[0][0])
```