МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИКОЛАЇВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет менеджменту

Кафедра економічної кібернетики, комп'ютерних наук та інформаційних технологій

Кваліфікаційна наукова
праця на правах рукопису

КРАВЧИНА Максим Вікторович

УДК 621.392.71:004.451.1(043.2)

КВАЛІФІКАЦІЙНА РОБОТА

ЗАСТОСУВАННЯ МАШИННОГО НАВЧАННЯ ДЛЯ ПРОГНОЗУВАННЯ ЗАТОРІВ НА ДОРОГАХ У МІСТІ

Спеціальність 122 «Комп'ютерні науки»
Галузь знань – 12 «Інформаційні технології»

Подається на здобуття освітнього ступеня «Бакалавр»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело _____ М.В. Кравчина

Науковий керівник:
Тищенко Світлана Іванівна, кандидат педагогічних наук, доцент

Завідувач кафедри:
Тищенко Світлана Іванівна, кандидат педагогічних наук, доцент.

Миколаїв–2025

АНОТАЦІЯ

Кравчина М.В. Застосування машинного навчання для прогнозування заторів на дорогах у місті – Кваліфікаційна праця на правах рукопису.

Робота на здобуття освітнього ступеня бакалавра за спеціальністю 122 «Комп'ютерні науки». – Миколаївський національний аграрний університет, Миколаїв 2025.

Ця кваліфікаційна робота присвячена розробці системи прогнозування заторів на дорогах у міських умовах із використанням методів машинного навчання. У сучасному світі, де рівень урбанізації стрімко зростає, проблема перевантаженості дорожньої інфраструктури є надзвичайно актуальною. Забезпечення своєчасної та точної інформації про можливі затори сприяє підвищенню ефективності дорожнього руху та зменшенню негативного впливу на довкілля.

У даній роботі проведено аналіз предметної області, зокрема досліджено природу та динаміку міських транспортних потоків. Визначено вимоги до програмної системи, описано джерела збору даних про дорожню ситуацію (GPS-дані, сенсори, камери, API Google Maps), розглянуто методи попередньої обробки інформації та виділення ключових ознак. Особливу увагу приділено вибору та реалізації алгоритмів машинного навчання, таких як Random Forest, для класифікації стану дорожнього руху та прогнозування виникнення заторів

Розроблено архітектуру системи прогнозування, здійснено моделювання структури даних, реалізовано програмну частину проєкту з використанням мов програмування Python та бібліотек машинного навчання. Описано процес навчання та тестування моделі, розроблено програмний інтерфейс для виведення результатів та інтеграції з реальними даними

Результатом роботи є функціональна система, яка дозволяє прогнозувати затори на основі аналізу історичних та поточних даних про дорожній рух. Реалізований інтерфейс забезпечує зручний доступ до прогнозної інформації, що може бути використано для підвищення ефективності роботи транспортних служб або створення мобільних застосунків для водіїв.

Ця робота демонструє актуальність впровадження інтелектуальних систем у сферу транспортної логістики, а також надає практичні рекомендації та приклади реалізації інноваційних підходів до аналізу трафіку з використанням сучасних технологій машинного навчання.

Ключові слова: машинне навчання, прогнозування заторів, інтелектуальні транспортні системи, аналіз дорожнього трафіку, нейронні мережі, обробка даних у реальному часі, великі дані (Big Data), моделювання трафіку, оптимізація дорожнього руху, транспортна аналітика.

ABSTRACT

Kravchyna M.V. Applying machine learning to predict traffic congestion in the city – Bachelor's Qualification Thesis.

Work for obtaining a bachelor's degree in the specialty 122 'Computer Science' – Mykolaiv National Agrarian University, Mykolaiv 2025.

This bachelor thesis is devoted to the development of a system for predicting traffic congestion in urban areas using machine learning methods. In today's world, where the level of urbanisation is rapidly increasing, the problem of congestion on the road infrastructure is extremely relevant. Providing timely and accurate information about possible congestion helps to improve road traffic efficiency and reduce the negative impact on the environment.

This paper analyses the subject area, in particular, the nature and dynamics of urban traffic flows. The requirements for the software system are defined, the sources of traffic data collection (GPS data, sensors, cameras, Google Maps API) are described, and methods of preliminary information processing and key features extraction are considered. Particular attention is paid to the selection and implementation of machine learning algorithms, such as Random Forest, for classifying traffic conditions and predicting congestion.

The architecture of the forecasting system is developed, the data structure is modelled, and the software part of the project is implemented using Python programming languages and machine learning libraries. The process of training and

testing the model was described, and a software interface for displaying results and integrating with real data was developed

The result of the work is a functional system that allows predicting traffic jams based on the analysis of historical and current traffic data. The implemented interface provides convenient access to forecast information that can be used to improve the efficiency of transport services or create mobile applications for drivers.

This work demonstrates the relevance of introducing intelligent systems in the field of transport logistics, and provides practical recommendations and examples of implementing innovative approaches to traffic analysis using modern machine learning technologies.

Keywords: machine learning, traffic congestion prediction, intelligent transportation systems, traffic flow analysis, neural networks, real-time data processing, big data, traffic modeling, traffic optimization, transportation analytics.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасні міста стрімко розвиваються, що супроводжується інтенсивним зростанням кількості транспортних засобів, збільшенням щільності дорожнього руху та ускладненням транспортної інфраструктури. Ці чинники призводять до виникнення дорожніх заторів, які суттєво впливають на ефективність функціонування міського транспорту, екологічну ситуацію, економічні витрати та якість життя населення. Щоденні затори втрачають години продуктивного часу, спричиняють зростання витрат на паливо, збільшують рівень стресу водіїв та пасажирів, а також сприяють підвищенню рівня забруднення повітря через додаткові викиди від транспортних засобів. У зв'язку з цим, прогнозування заторів на дорогах набуває особливої актуальності. Можливість заздалегідь передбачити перевантаження окремих ділянок дорожньої мережі дає змогу ефективніше управляти транспортними потоками, оптимізувати маршрути пересування, зменшити затримки та покращити загальну мобільність у місті. Однак точне прогнозування дорожніх ситуацій є надзвичайно складним завданням через велику кількість факторів, що впливають на рух транспорту: час доби, погодні умови, дорожні події, поведінка водіїв, зміни інфраструктури тощо. Зараз активно розвиваються інтелектуальні транспортні системи, які використовують новітні досягнення в галузі інформаційних технологій, зокрема машинного навчання, Big Data та інтернету речей, для покращення управління дорожнім рухом. Методи машинного навчання, які дозволяють аналізувати великі масиви даних і знаходити закономірності в них, відкривають нові можливості для точного та ефективного прогнозування заторів на дорогах.

Метою кваліфікаційної роботи є розробка системи, що на основі історичних і реальних даних про дорожній рух зможе передбачити ймовірність виникнення затору на певній ділянці дороги.

Для досягнення цієї мети було проведено огляд сучасних підходів до аналізу транспортних потоків, обрано релевантні алгоритми машинного навчання, зібрано та проаналізовано набір вхідних даних, розроблено модель

прогнозування, а також реалізовано програмну систему для її інтеграції та тестування на практичних прикладах.

Для досягнення мети необхідно виконати такі *завдання:*

– провести огляд існуючих підходів для прогнозування заторів на дорогах у місті, зокрема з використанням моделі машинного навчання Random Forest;

– проаналізувати існуючі технічні рішення що застосовуються у сервісах для прогнозування заторів таких як Google Maps;

– розробити архітектуру системи прогнозування, що включає:

  – скрипт збору, і обробки маршрутів, використовуючи Google Maps API;

  – скрипт навчання та створення моделі за допомогою ансамблевого методу Random Forest;

  – скрипт побудування, та прогнозування маршрутів на карті Києва;

  – запуск інтерфейсу на базі бібліотеки Streamlit.

– Провести оцінку моделі за допомогою кросс-валідації, F1-score.

Предметом дослідження є моделі та алгоритми машинного навчання, здатні виявляти закономірності та робити прогнози щодо стану дорожнього руху.

Об'єктом дослідження є транспортні потоки у межах міста. У якості основних методів дослідження використано алгоритм Random Forest як базовий класифікатор, а також інструменти для збирання даних із Google Maps API, їх обробки та візуалізації.

Структура і обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел, додатків. Основний зміст викладено на 53 сторінках комп’ютерного тексту. Робота включає 4 таблиці, 14 рисунків, 2 додатки. Список використаних джерел налічує 30 найменувань.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ МАШИННОГО НАВЧАННЯ ДЛЯ ПРОГНОЗУВАННЯ ЗАТОРІВ

1.1 Сутність поняття транспортних потоків у міських умовах

Транспортні потоки в умовах сучасного міста є динамічним і нестабільним процесом, що змінюється у просторі та часі, реагуючи на низку зовнішніх подразників, зокрема зміну погодних умов, графік роботи установ, аварії, ремонтні роботи, події культурного або спортивного характеру, а також поведінкові фактори учасників дорожнього руху. Оцінювання та моделювання транспортних потоків потребує розгляду руху як сукупності множинних об’єктів, які мають:

- траєкторію;
- швидкість;
- інерцію;
- індивідуальну логіку прийняття рішень.

Рух в умовах міста, на відміну від автомагістралей чи заміських трас, має більш виражену фрагментарність та змінність: щільність транспорту може кардинально змінюватися навіть у межах кількох хвилин або кварталів, що ускладнює передбачення загального стану дорожньої мережі. У таких умовах аналітики змушені враховувати не лише поточні показники інтенсивності руху, а й контекстуальні чинники, які безпосередньо впливають на швидкість пересування та час перебування транспортного засобу в тій чи іншій зоні. У містах чітко простежується залежність структури дорожнього трафіку від добових, тижневих та сезонних циклів. Наприклад, у години пік навантаження суттєво перевищує середні показники, що часто спричиняє затори навіть у заздалегідь не проблемних зонах. При цьому, локальні події, як-от аварії, дорожні роботи або погодні катаклізми, можуть раптово змінити напрям і щільність руху, трансформуючи потік в неочікуваних напрямках. Сповільнення або часті зупинки можуть бути індикаторами підвищеного навантаження на певній ділянці дороги, що, своєю чергою, вказує на ймовірність формування затору. Висока

інтенсивність при низькій швидкості свідчить про стан перевантаження, тоді як зниження інтенсивності при нормальній швидкості може означати тимчасове зменшення трафіку без ознак затору. Якість, обсяг і різноманітність даних визначають можливості прогнозування заторів, моделювання транспортних потоків та прийняття оперативних управлінських рішень. Отримання актуальної та точної інформації про стан доріг, інтенсивність руху, середню швидкість транспорту, а також просторові та часові характеристики транспортних потоків потребує впровадження інноваційних технологій, що дозволяють автоматизовано фіксувати ключові параметри дорожньої ситуації.

Завдяки розгалуженій системі фіксації трафіку, що базуються на різноманітних типах фізичних сенсорів, які вбудовуються у дорожнє полотно або встановлюються на транспортних вузлах вдається безперервно контролювати кількість транспортних засобів, що проходять певну ділянку дороги, вимірювати швидкість руху автомобілів, визначати їх тип та клас, а також реєструвати часові характеристики потоку. Ці дані надходять у централізовані обчислювальні системи, де вони обробляються у реальному часі для подальшого аналізу.

GPS (англ. Global Positioning System) — сукупність радіоелектронних засобів, що дозволяє визначати положення та швидкість руху об'єкта на поверхні Землі або в атмосфері.

Рисунок 1.1 Архітектура GPS

Положення об'єкта обчислюється завдяки використанню розміщеного на ньому GPS-приймача, який приймає та обробляє сигнали супутників космічного сегменту GPS-системи глобального позиціонування. Для визначення точних параметрів орбіт супутників та керування GPS-системою вона в своєму складі має наземні центри управління.[1] Пристрої, які встановлюються у автомобілях, зокрема в рамках систем GPS-моніторингу, дозволяють здійснювати трекінг маршрутів пересування, визначати геолокацію у режимі реального часу та реєструвати швидкісні характеристики. Дані, зібрані у такий спосіб, формують цінний масив інформації для подальшого аналізу, оскільки дають змогу не лише оцінити загальний стан дорожнього руху, але й виявляти закономірності у поведінці водіїв, реагування на затори, вибір альтернативних маршрутів тощо. Крім того, використання навігаційних технологій дозволяє створювати історичні карти навантаження транспортної мережі, що стають основою для побудови моделей прогнозування.

У процесі отримання транспортної інформації активно використовується відкрите джерело даних, зокрема від глобального картографічного сервісу Google Maps. Він надає доступ до агрегованої інформації про завантаженість доріг, середню швидкість руху в різні періоди часу, а також пропонує API-інтерфейси для інтеграції цих даних у сторонні аналітичні системи. У поєднанні з іншими джерелами ці дані дозволяють формувати повноцінну картину транспортної ситуації в місті як у поточний момент часу, так і в ретроспективному аналізі.[2]

1.2 Огляд методів машинного навчання у сфері дорожнього руху

Машинне навчання – це розділ штучного інтелекту, який дає змогу комп’ютерним системам навчатися на даних і покращувати свої результати без явного програмування (не потрібно писати інструкції для кожної задачі). Воно працює на основі алгоритмів (спеціальних правил), які аналізують великі обсяги інформації, шукають у ній закономірності та використовують ці знання для

ухвалення рішень або прогнозування. ML можна порівняти з тим, як людина вчиться на власному досвіді.[3] Щоб добре навчити машину, потрібні:

- Данні - потрібна неймовірна кількість данних, а саме – десятки тисяч прикладів. На основі аналізу сотень тисяч GPS-маршрутів система може «зрозуміти», що у певний час на конкретному перехресті часто виникають затори через перевантаження сусідньої ділянки, хоча візуально ситуація може виглядати задовільно.[21]
- Ознаки - ціна, якість, матеріал, будь що, що може охарактеризувати потрібні нам предмети. Машина повинна знати, на що конкретно їй слід звернути увагу. У випадку аналізу дорожнього трафіку такими ознаками можуть бути середня швидкість потоку, щільність руху, наявність світлофорів чи пішохідних переходів, середній час подорожі тощо.
- Алгоритм – одну й ту ж задачу, зазвичай, можна розв'язати різними способами. Від вибору способу залежить лише швидкість і якість роботи. [4]

Крім того, важливою складовою є валідація результатів – перевірка того, наскільки добре модель справляється із завданням на нових, раніше не бачених прикладах. У цьому контексті застосовуються різні метрики точності, серед яких часто використовують середньоквадратичну похибку, середню абсолютну похибку або відсоток правильних передбачень. Серед найпоширеніших алгоритмів, що застосовуються для задач аналізу транспортних потоків, виділяються методи на основі дерев рішень та ансамблевих моделей, таких як Random Forest.

Метод дерев рішень є одним із найпростіших і найбільш інтерпретованих інструментів класифікації та регресії. Його перевага полягає в здатності працювати як з числовими, так і з категоріальними даними, не вимагаючи масштабування чи складного попереднього оброблення. Водночас основним недоліком дерев рішень є їхня схильність до переобучення, особливо коли модель має велику глибину, що знижує її узагальнювальні властивості на нових даних. Дерева рішень працюють на основі алгоритмічного підходу, який розбиває набір даних на окремі точки даних на основі різних критеріїв. Ці розбиття виконуються за допомогою різних змінних або різних функцій набору даних.[5]

Ансамблеве прогнозування – це підхід до моделювання, який поєднує джерела даних, моделі різних типів з альтернативними припущеннями, використовуючи різні методи розпізнавання образів.[15]

Random Forest — це широко використовуваний алгоритм машинного навчання, який об'єднує результати кількох дерев рішень для отримання одного результату.[19] Це ансамблевий підхід, який комбінує велику кількість дерев рішень у єдину узагальнену модель. Такий підхід дозволяє значно зменшити проблему перенавчання та підвищити точність прогнозу.

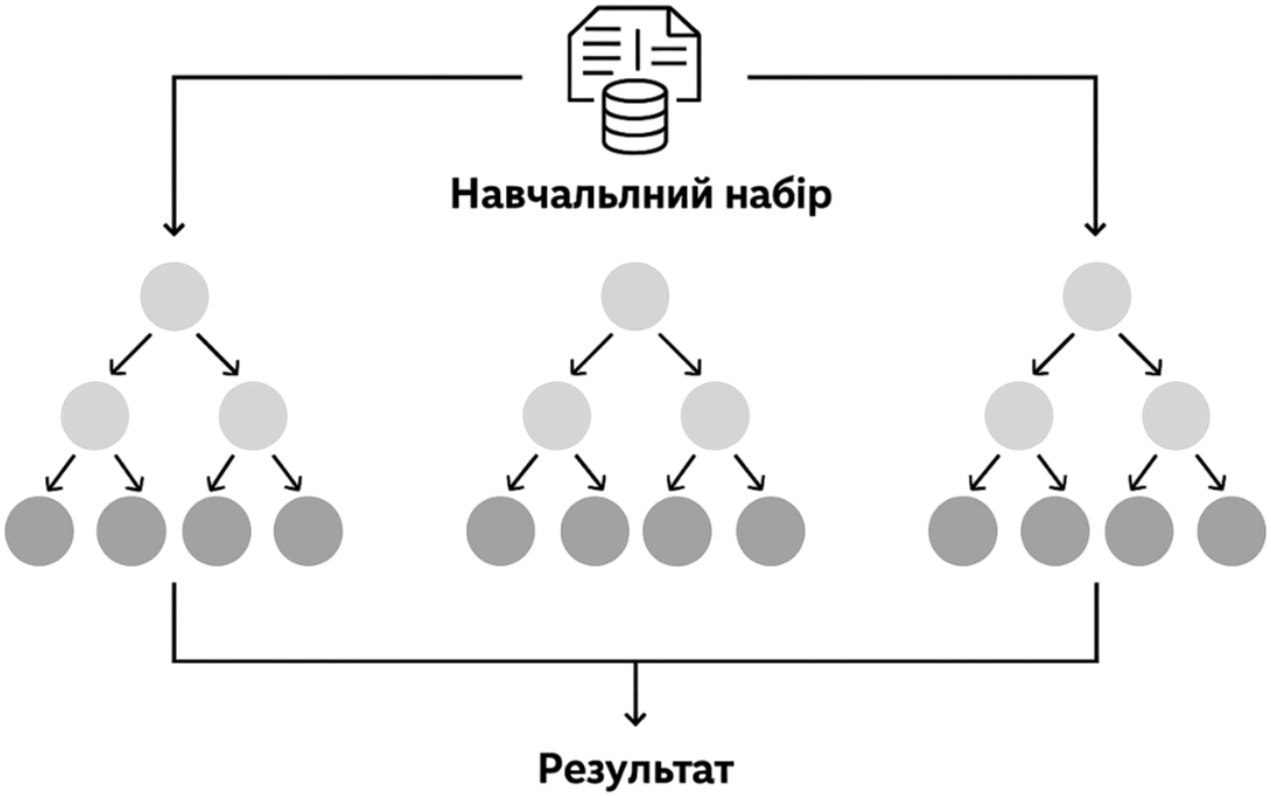


Рисунок 1.2 – Дерева рішень

Його ключовою перевагою в порівнянні з окремими деревами рішень є підвищена точність, стійкість до перенавчання, а також здатність адекватно працювати з різними типами даних. У контексті прогнозування заторів і класифікації дорожніх умов, Random Forest демонструє особливо високі результати, оскільки може ефективно враховувати численні параметри, що впливають на стан дорожнього руху, та автоматично виявляти важливість кожного з них. Його недоліком, є складність інтерпретації фінального рішення та

зниження продуктивності на даних із сильною часовою залежністю. У випадку з транспортними даними, модель має справу з великим обсягом вхідної інформації:

- час доби,
- день тижня,
- щільність трафіку,
- швидкість транспортних засобів,
- географічні координати та характеристики конкретної ділянки дороги.

Враховуючи, що ці ознаки можуть мати як лінійні, так і складні, взаємозалежні впливи, Random Forest надає інструменти для автоматичного виявлення таких взаємозв'язків без необхідності попереднього аналізу або створення складних математичних моделей. Завдяки своїй структурі з великої кількості незалежних дерев, які будуються на основі випадкових підвибірок ознак і даних, алгоритм здатен «згладжувати» шум у даних та забезпечувати стабільний прогноз навіть у випадку часткової втрати або спотворення інформації.

Random Forest у завданнях транспортної класифікації має здатність оцінювати важливість ознак (feature importance).[6] У ситуаціях, коли модель оперує десятками або сотнями характеристик, ця властивість дозволяє зрозуміти, які саме фактори мають найбільший вплив на формування затору або інших дорожніх ситуацій. Наприклад, модель може показати, що головними причинами заторів є час пік, вузькість дорожнього полотна чи інтенсивність руху на сусідніх вулицях, що, у свою чергу, може бути використано для подальшої оптимізації маршрутів або перегляду інфраструктурних рішень. У порівнянні з більш складними нейронними мережами, такими як LSTM, Random Forest є менш вимогливим до обчислювальних ресурсів, що дозволяє впроваджувати його в системи, які мають обмеження щодо обробки інформації в реальному часі. Хоча точність таких моделей може бути трохи нижчою в задачах з високою динамікою, у випадках з обробкою історичних даних або регулярного моніторингу дорожньої ситуації цей підхід демонструє цілком задовільні результати.

У випадках, коли дані мають послідовну природу, особливо якщо це часові ряди, найбільш ефективними виявляються нейронні мережі з довготривалою короткочасною пам'яттю (LSTM). Цей клас рекурентних мереж здатний зберігати

та передавати інформацію через великі інтервали часу, що робить його ідеальним для прогнозування транспортних подій, які залежать від історичного контексту. Модель LSTM виявляє приховані залежності між подіями в часовому потоці, що дозволяє їй передбачати затори на основі попередніх станів дорожнього руху. Проте ці моделі потребують великої кількості даних для успішного навчання, а також відрізняються складністю у налаштуванні та значними вимогами до обчислювальних ресурсів.

Модель SVM є представленням зразків як точок у просторі, відображених таким чином, що зразки з окремих категорій розділено чистою прогалиною, яка є щонайширшою. Нові зразки тоді відображуються до цього ж простору, й робиться передбачення про їхню належність до категорії на основі того, на який бік прогалини вони потрапляють.[20]

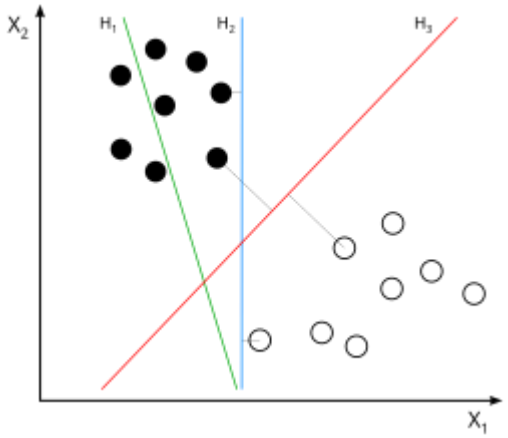


Рисунок 1.3 Приклад роботи SVM методу

Основна ідея методу SVM полягає в знаходженні гіперплощини, яка максимально відокремлює дані одного класу від іншого у багатовимірному просторі ознак. Це досягається шляхом максимізації запасу розділення (margin), тобто відстані між найближчими точками кожного класу (які називаються опорними векторами) та гіперплощиною.

Таблиця 1.1 Порівняльна характеристика методів машинного навчання

Метод	Точність	Інтерпретація	Витрати ресурсів	Схильність до перенавчання
SVM	Висока	Висока	Високі	Низька

Random Forest	Висока	Середня	Середні	Низька
LSTM	Дуже висока	Низька	Дуже високі	Залежить від налаштувань моделі

1.3 Використання історичних данних для прогнозування заторів

Для успішного навчання моделей машинного навчання, спрямованих на прогнозування заторів на дорогах, необхідно забезпечити їх якісними і змістовними даними, які відображають реальні умови транспортного середовища. Якщо дані погано піддаються навчанню, наприклад, нерепрезентативні, неякісні, нерелевантні ознаки або їх недостатня кількість для навчання, то моделі машинного навчання можуть стати марними або матимуть нижчу точність.[7] Саме тому особливу увагу слід приділити характеристикам даних, які використовуються для побудови такої моделі. В першу чергу, модель повинна мати доступ до інформації, яка відображає динаміку дорожнього руху у просторі та часі. Ці дані мають бути прив’язані до конкретних географічних координат та часових міток, що дозволяє моделі враховувати змінність дорожнього навантаження протягом доби, тижня або навіть пори року. Така інформація, крім того повинна передавати кількісні показники транспортної активності, наприклад середню швидкість руху на певній ділянці, рівень завантаженості дороги, щільність транспортного потоку або кількість транспортних засобів, що проходять за одиницю часу. Подібні характеристики дозволяють моделі навчитися розпізнавати ситуації, які супроводжують утворення заторів, та відрізняти їх від нормального потоку.

У сучасних дослідженнях у збиранні маршрутів відіграють GPS-технології та сервіси геолокації, серед яких особливо виділяється Google Maps API, яку я обрав за основу для своєї кваліфікаційної роботи. Використання цього сервісу дозволяє автоматизувати процес збору даних і забезпечити систематичне накопичення інформації про маршрути. До того ж, Google Maps API надає змогу

формувати запити з урахуванням конкретного часу доби, що дозволяє моделювати рух транспорту в умовах, максимально наближених до реальних.

GPS-датчики дають змогу визначати точне місцезнаходження транспортного засобу, його швидкості та пройденого маршруту.[8] Історичні маршрути, зібрані за допомогою GPS, містять послідовність координат, кожна з яких супроводжується часовою позначкою. Завдяки цьому можна з високою точністю визначити швидкість переміщення, час перебування на певних ділянках дороги, а також виявити зони, де часто виникає уповільнення чи зупинка руху. Такі дані дозволяють не лише аналізувати просторово-часові характеристики транспортних потоків, а й виявляти аномалії у русі, які можуть свідчити про наявність заторів або інших обмежень. Процес інтеграції GPS-даних із сервісами на кшталт Google Maps вимагає попередньої обробки та нормалізації даних.

Нормалізація даних – це процес організації даних у реляційній БД з метою мінімізації надлишковості даних, поліпшення цілісності даних і забезпечення ефективності виконання запитів.[9] Сирі GPS-логи часто містять похибки, дублікати або пропуски координат, що може викликати спотворення вхідних параметрів для моделей машинного навчання. Важливо забезпечити стабільність інтервалів між точками маршруту, узгодженість часових міток, а також правильну географічну прив’язку до дорожньої мережі. Така підготовка гарантує, що модель отримує не лише велику кількість даних, але й їхню якісну структуру.

У сукупності, процес збирання історичних маршрутів є поєднанням технологічних, аналітичних та правових аспектів. Він формує фундамент для побудови точної системи прогнозування заторів, яка може адаптуватися до змін транспортної ситуації, навчатись на минулому досвіді та забезпечувати оперативне реагування на потенційні загрози заторів у майбутньому. Такий підхід дозволяє не лише глибше зрозуміти поведінку транспортних потоків, а й створити ефективну аналітичну модель, що враховує численні фактори міського дорожнього середовища. Проблема точності даних полягає насамперед у варіативності джерел, з яких ці дані отримуються. Наприклад, супутникові системи, такі як GPS, дуже чутливі до фізичних перешкод. У міських районах, де будівлі створюють так звані «міські каньйони», сигнали можуть відбиватися від

стін і поверхонь, що призводить до так званого ефекту мультишляху. Це знижує точність визначення місцезнаходження, оскільки приймач отримує сигнали з затримкою або з кількох джерел одночасно. Подібна ситуація може виникати й у густих лісах або під землею, де доступ до супутникових сигналів взагалі може бути відсутній.[10] Крім того, у даних можуть бути пропущені значення — частково або повністю відсутні записи про певні часові інтервали, географічні ділянки чи параметри трафіку, що ускладнює формування повноцінного вектору ознак для навчання моделі. У деяких випадках ці пропуски спричинені технічними збоями у системах моніторингу, людським фактором при введенні інформації або відсутністю належної інфраструктури збору даних у певному регіоні. Окрему проблему становить неконсистентність історичних записів, що виникає у випадках, коли структура або формат зібраних даних змінювався з часом. Наприклад, одні записи можуть мати точну прив'язку до часу з похвилинною точністю, тоді як інші — лише до години. Або ж одна частина даних може містити додаткові параметри, такі як тип транспортного засобу чи погодні умови, в той час як інша — лише базову інформацію про швидкість і координати. Така несумісність змушує розробника або значно ускладнювати процес передобробки даних, або ж обмежуватись лише тим підмножиною інформації, що має повну відповідність усім записам, що, у свою чергу, ще більше звужує можливості моделі.

1.4 Системи прогнозування заторів у містах та їх обмеження

Системи прогнозування заторів на дорогах активно впроваджуються у повсякденну практику транспортного менеджменту багатьох країн, переважно у великих мегаполісах, де проблема перевантаженості дорожньої мережі набуває особливої гостроти. Найбільш відомими є такі рішення, як Google Maps, Waze, HERE Technologies, TomTom Traffic та інші, які вже продемонстрували свою ефективність у наданні візуалізованої інформації про поточну дорожню ситуацію, попередження про затори або пропозицію альтернативних маршрутів. Основою для роботи таких систем слугують геолокаційні дані, дані з мобільних пристроїв користувачів, інформація від дорожніх сенсорів, камер спостереження,

супутникових джерел та навіть погодних сервісів.[11] Більшість систем дотримується моделі «дані в реальному часі + історичні патерни», що дає змогу не лише відображати теперішній стан трафіку, але й робити короткотермінові прогнози на основі накопиченого досвіду. У процесі роботи алгоритми аналізують швидкість руху транспортних засобів на окремих ділянках, частоту гальмувань, час перебування в конкретній зоні, а також фіксують різкі зміни у поведінці трафіку, що може свідчити про аварію, затор або інші ускладнення. Такі системи часто мають можливість об'єднувати велику кількість джерел, тим самим забезпечуючи широку картину дорожньої ситуації у реальному часі. Характерною рисою деяких платформ є використання колективної взаємодії користувачів, коли кожен учасник дорожнього руху виконує функцію сенсора. Користувачі, пересуваючись із ввімкненими додатками, несвідомо передають геодані та швидкість пересування, що дозволяє системі будувати карти завантаженості. Крім того, деякі сервіси надають можливість користувачам вручну повідомляти про ДТП, затори, ремонтні роботи, що доповнює картину дорожньої ситуації суб’єктивною, але цінною інформацією.[12]

Попри очевидні переваги, такі системи мають низку обмежень. Частою проблемою є складність масштабування таких систем на нові географічні регіони, особливо ті, що мають слабку інформаційну інфраструктуру. У деяких містах відсутні централізовані цифрові платформи або не проводиться регулярний збір статистичних даних про транспортні потоки. Це означає, що навіть за наявності програмного забезпечення та технічної можливості аналізу, модель просто не матиме необхідного обсягу інформації для ефективного функціонування. У кращому випадку система базуватиметься на застарілих або фрагментованих даних, що значно знижує точність прогнозів. Тому створення локалізованої системи потребує не лише перекладу або налаштування інтерфейсу, а й глибокої адаптації самого алгоритмічного ядра, врахування місцевих особливостей дорожнього середовища, розробки власних картографічних шарів та логік реагування на динамічні зміни у русі. Універсальні моделі, що добре функціонують у великих містах США чи Європи, нерідко втрачають точність при застосуванні в умовах іншої урбаністичної структури, де інтенсивність руху,

розташування об’їзних доріг, наявність маршрутного громадського транспорту, а також поведінкові особливості водіїв значно відрізняються. Наприклад, щільність забудови у центрі Києва або Львова суттєво відрізняється від типової дорожньої мережі Лондона чи Берліна, що змушує переглядати логіку побудови маршрутів та оцінки навантаження на дороги. Ба більше, навіть культурні відмінності, як-от рівень дотримання правил дорожнього руху, можуть суттєво вплинути на надійність прогнозів, адже багато систем працюють на припущеннях, що трафік розподіляється рівномірно або прогнозовано.

Машинне навчання, а особливо глибинне навчання та рекурентні нейронні мережі (такі як LSTM), висувають серйозні вимоги до обчислювальних ресурсів, включно з оперативною пам’яттю, графічними процесорами та загальною продуктивністю серверної інфраструктури.[13] Для більш наочного розуміння основних ресурсних обмежень існуючих систем прогнозування заторів наведемо порівняльну таблицю, що охоплює компоненти, які впливають на продуктивність та точність таких систем:

Таблиця 1.2 Порівняльна таблиця з компонентами що впливають на продуктивність

Категорія ресурсу	Суть обмеження	Наслідки для системи
Обчислювальні потужності	Обмежений доступ до серверів, GPU, процесорів для обробки великих даних	Зменшення швидкості прогнозу, затримки в обробці, неможливість працювати в реальному часі
Оперативна пам’ять	Недостатній обсяг пам’яті для одночасної обробки кількох потоків даних	Виникнення помилок під час роботи моделі, зниження стабільності системи
Обсяг даних	Відсутність повноцінної історичної бази або	Низька точність прогнозу, відсутність

	неякісні джерела інформації	адаптації до нетипових ситуацій
Мережеві ресурси	Повільна передача даних від сенсорів, нестабільний інтернет-зв’язок	Порушення безперервності збору даних, втрата частини інформації
Сховище даних	Обмежена ємність для довготривалого зберігання історичних записів	Неможливість здійснювати довготривалий аналіз, ускладнення повторного навчання моделей

Надмірна концентрація на складності алгоритмів без урахування реальних обмежень апаратного та інформаційного середовища може призвести до низької ефективності таких рішень у практичних умовах. Саме тому сучасні дослідження повинні базуватися на поєднанні оптимізованих моделей, які не потребують надмірних ресурсів, із добре структурованою системою збору та обробки даних.

Більшість існуючих рішень побудовані на закритих екосистемах із власними наборами API, обмеженнями доступу до сервісів або комерційними ліцензіями, що унеможливлює їх широке розгортання в окремих містах чи адаптацію під відкриті локальні платформи. Це створює додаткові складнощі для муніципалітетів, які прагнуть створити власну систему на базі відкритих даних, адже доводиться або погоджуватись на зовнішні комерційні умови, або будувати все «з нуля».[22]

Частина з сервісів орієнтована на масового споживача і має простий користувацький інтерфейс, інші ж доступні лише через спеціалізовані API з технічними обмеженнями, ліцензійними вимогами або необхідністю оформлення платних підписок. В окремих випадках ці сервіси інтегровані безпосередньо у мобільні додатки для навігації, що полегшує їх використання, але водночас обмежує доступ до сирих даних або аналітичних моделей для сторонніх розробників. Це ускладнює можливість глибокої кастомізації або адаптації рішень

під конкретні задачі, зокрема в академічних дослідженнях чи для реалізації локальних ініціатив у містах із меншою цифровою інфраструктурою. Інтегрованість сучасних систем у реальні інфраструктури відображає ступінь їх взаємодії з іншими елементами міського середовища, такими як розумне регулювання світлофорів, оперативне управління транспортними потоками або інформаційні табло для водіїв. У великих мегаполісах такі рішення часто поєднані із системами відеоспостереження, GPS-моніторингу, мобільними застосунками та сервісами сповіщень, що забезпечує високий рівень динамічного управління.

РОЗДІЛ 2
АНАЛІЗ ДАННИХ ПРО ДОРОЖНІЙ РУХ ТА ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Збір даних про дорожній рух (GPS, сенсори, камери)

Одним з найбільш дієвих рішень для отримання достовірних і актуальних даних щодо руху автомобільного транспорту є використання послуг Google Maps Platform, зокрема послуг Google Maps Directions API, Google Maps Distance Matrix API та Google Traffic Data, які забезпечують доступ до даних щодо маршрутів, швидкості руху, передбачуваний та фактичний час подорожі, інтенсивність руху на окремих ділянках дороги, а також дані щодо поточних дорожніх подій.[14] Контакт з послугами Google API відбувається шляхом надсилання HTTP-запитів з точними географічними координатами початку і кінця шляху, а також додатковими параметрами для забезпечення відповіді залежно від системи. до них зокрема належать тип транспорту, тип отриманих даних, мова інтерфейсу, часові знаки для попередження майбутнього трафіку, а також параметри, що забезпечують отримання інформації про найкращі маршрути залежно від заторів, аварій чи технічного ремонту доріг. У процесі інтеграції з служби Google API ми маємо дотримуватись виконання політики використання та ліміту, встановленого фахівцями служби. Умовно, наприклад, існує ліміт запитів, яким підлягає регулювання кількості звернень до API за певний проміжок часу, а також загальної кількості запитів за добу, тиждень або місяць. Ліміти такі потребують встановлення залежно від обраної тарифної плівки та виду доступу до API. Для обробки запитів слід використовувати платний план, що забезпечує більші квоти та доступ до додаткової функціональності.[27]

В моїй розробленій системі збір інформації щодо шляхів між окремими точками міста був використаний скрипт, який програмно надсилає запит до Google Directions API, опрацьовує відповідь та зберігає дані в базу даних.

Рисунок 2.1 – Принцип роботи модуля collect_routes

Отримані дані, зокрема географічні координати, час у дорозі з урахуванням трафіку та без нього, а також трафік-індикатори, використовуються як вхідні змінні для алгоритму машинного навчання, що дозволяє формувати ознаки, які мають прямий вплив на прогнозування заторів.

Від обраного способу збереження інформації залежить ефективність подальшого аналізу, швидкість обробки даних, зручність доступу до них, а також можливість масштабування системи у майбутньому. На початковому етапі збирання інформації, коли мова йде про невеликі обсяги даних або прототипування моделі, зручно використовувати табличні формати, зокрема у вигляді текстових файлів. На практиці найпоширенішим форматом для зберігання структурованих даних є формат CSV, який дозволяє представляти дані у вигляді таблиць з фіксованими колонками.[23] Простота цього формату забезпечує його широке застосування при обміні даними між різними системами, при підключенні до аналітичних інструментів, а також у процесі первинної перевірки або візуального перегляду. Проте з часом, коли виникає потреба в обробці більш складних ієрархічних структур або великого обсягу параметрів, прості формати стають недостатніми.

Наприклад, під час інтеграції з Google Maps API кожна точка маршруту може містити не лише координати, а й супутню інформацію — від часу прибуття до умов дорожнього покриття. Зберігати таку інформацію у табличному вигляді не лише складно, а іноді й неможливо без втрати частини контексту. Вибір між реляційними та нереляційними системами зберігання залежить від характеру даних та вимог до масштабованості. Наприклад, при роботі зі структурованими даними про трафік із фіксованими ознаками, такими як час, місце, швидкість руху та щільність потоку, доцільно застосовувати реляційні бази даних. Реляційні бази даних (РБД) – один із найпоширеніших типів баз даних, що використовуються для зберігання, організації та управління інформацією.[9]

2.2 Аналіз зібраних данних та виділення ключових факторів

У процесі аналізу транспортних потоків найбільш поширеними показниками є швидкість транспорту, інтенсивність руху, щільність на окремих ділянках, кількість зупинок та середній час перебування. Ці характеристики створюють загальне уявлення про динаміку дорожнього руху та служать основою для розробки моделей. Однак ці параметри, хоч і важливі, не можуть самостійно визначити наявність або відсутність затору. Тільки всебічний аналіз їх взаємозв’язків дає змогу отримати цілісну картину. При використанні моделей на основі ансамблевих методів, таких як Random Forest, можливо визначити, які саме параметри мають найвищу вагу у прийнятті рішення щодо класифікації певного стану дорожнього руху як "навантаженого". Ці моделі автоматично враховують кореляції між вхідними змінними та виводом, що дозволяє не лише отримати високу точність, але й інтерпретувати результат.[24]

Під час роботи з даними, було встановлено, що значне зниження швидкості у певні часові періоди, особливо у пікові години, з високою ймовірністю вказує на наявність або наближення до стану затору. Водночас, один лише показник швидкості не завжди дає повну картину: наприклад, низька швидкість у центрі міста може бути типовою навіть при відсутності критичних перевантажень. Саме тому доцільним є одночасне врахування супутніх факторів, таких як кількість автомобілів на певному відрізку дороги, час доби, день тижня, погодні умови, а

також наявність інфраструктурних обмежень, як-от ремонтні роботи або аварійні ситуації.

Визначення вагомості ознак дозволяє зрозуміти, які з параметрів мають найбільший вплив на результат прогнозу, а також допомагає оптимізувати саму модель. Усунення надлишкових або слабоінформативних ознак не лише знижує ризик перенавчання, але й покращує швидкодію системи та полегшує її інтерпретацію. У випадку прогнозування заторів мова йде про численні чинники, серед яких можна згадати географічне положення, швидкість руху транспортних засобів, густоту потоку, пору доби, день тижня, погодні умови, наявність дорожніх робіт, подій або аварій. Однак далеко не всі ці параметри однаково впливають на кінцеве рішення моделі. Тому важливо не просто включити ці змінні у процес навчання, а й дослідити їхню реальну цінність. Алгоритм Random Forest, використаний у рамках даного дослідження, має вбудований механізм для оцінки важливості ознак, який ґрунтується на зменшенні критерію імпурності в кожному дереві ансамблю. Коли певна ознака сприяє істотному поділу вибірки в дереві рішень, вона отримує вищий коефіцієнт важливості. В результаті аналізу всі ознаки можуть бути ранжовані за рівнем їхнього впливу на точність прогнозу. Така інформація дозволяє зробити висновки не лише про технічну сторону моделі, а й про поведінкові та інфраструктурні аспекти дорожнього руху, які справді мають значення у формуванні заторів.

До чинників, що заслуговують уваги відносять день тижня, час доби, тип дорожньої інфраструктури, а також загальна транспортна навантаженість на певному відрізку вулиці чи автомагістралі. Їхній взаємозв’язок із рівнем заторів потребує глибокого аналізу, оскільки саме ці характеристики формують специфіку міського руху та дозволяють зробити прогноз більш точним і прив’язаним до конкретних обставин. У робочі дні активність транспорту різко зростає у ранкові та вечірні години, що зумовлено потребою населення дістатися до місця роботи або навчання, а потім повернутися додому. Така регулярність спричиняє чітко виражені пікові навантаження, які характерні для центральних вулиць і магістралей. У вихідні дні трафік, як правило, менш інтенсивний, проте в

окремих районах із великою кількістю торгових центрів чи зон відпочинку спостерігається локальне зростання транспортного навантаження у денні години.

Основні магістралі мають вищу пропускну здатність, але водночас є головними маршрутами масового пересування, що робить їх потенційно вразливими до перевантажень. Вторинні дороги, навпаки, можуть бути менш навантаженими, проте їх вузька геометрія та наявність регульованих перехресть часто стають причинами локальних затримок. Особливу увагу слід звертати на дороги, що проходять біля вузлів громадського транспорту, шкіл, лікарень або торгових центрів — ці зони є осередками підвищеної активності, незалежно від загальної щільності руху. Таким чином, для точної оцінки дорожньої ситуації слід не просто класифікувати дороги, а й враховувати функціональні особливості їх розташування.[30] Показники поточної навантаженості дороги, які можуть бути отримані завдяки GPS-даним або інформації з дорожніх сенсорів, відображають реальну ситуацію на відміну від історичних середніх значень. Важливо, що навантаження має не лінійний характер: навіть незначне перевищення порогу пропускної спроможності може призвести до лавиноподібного ефекту зупинки транспорту. Тому ефективні алгоритми прогнозування повинні враховувати не тільки абсолютні значення навантаження, але й динаміку змін — швидкість їх зростання або зменшення в реальному часі.

2.3 Проектування алгоритму для прогнозування заторів

Успішність реалізації всієї системи значною мірою залежить від того, наскільки обрана модель здатна точно відображати складну динаміку міського дорожнього руху, виявляти приховані закономірності та робити прогнози з урахуванням численних змінних, притаманних транспортному середовищу. На початковому етапі моделювання увага була зосереджена на алгоритмах, які зарекомендували себе як ефективні при роботі з табличними наборами даних, що містять як числові, так і категоріальні ознаки. Такі алгоритми характеризуються високою стійкістю до викидів у даних, відносною простотою реалізації та інтерпретованістю результатів, що важливо з точки зору подальшої валідації моделі. Проте у процесі аналізу виникла необхідність врахування часових

залежностей, що вимагає використання моделей, здатних працювати з послідовностями та історією змін стану об’єкта прогнозування.

У ході дослідження вдалося з’ясувати, що модель, яка використовує ансамблеве навчання та принцип спільного застосування кількох незалежних слабких класифікаторів, є найбільш ефективною для аналізу транспортних потоків у міських умовах. Ця модель може враховувати багато вхідних параметрів, зменшувати ймовірність перенавчання і показує хорошу здатність до узагальнення. Крім цього, сучасні бібліотеки машинного навчання забезпечують хорошу підтримку для її реалізації, а результати можна легко тлумачити й оцінювати. Вибраний підхід надає можливість адаптувати модель до нових даних, поступово підвищуючи її ефективність. З практичного боку, це означає, що система здатна вдосконалюватися з плином часу. Вона аналізує нові сценарії заторів і реагує на зміни в поведінці водіїв, розвиток інфраструктури або коливання погодних умов.

Архітектура даних – це структурована система, яка визначає, як дані збираються, зберігаються, управляються та використовуються в організації.[16] Вона визначає, яким чином дані з різних джерел, таких як GPS-вимірювання, API сервісів дорожнього руху, сенсори чи відеоаналітика, потрапляють у систему, як ці дані зберігаються, очищуються, перетворюються та використовуються для побудови моделей. Архітектура повинна забезпечувати стабільний механізм отримання вхідних даних у різноманітних форматах та з неоднорідних джерел. Оскільки джерела трафіку часто мають різні структури та часову розбіжність, важливо передбачити компоненти для синхронізації та нормалізації цих даних.[26] Одним із головних завдань є формування уніфікованого вигляду входу, де кожен запис містить ключові характеристики дорожньої ситуації, такі як координати ділянки, час доби, день тижня, середня швидкість руху, кількість транспортних засобів та інші релевантні параметри. Визначення релевантних ознак є критично важливим, оскільки саме від якості їх формування залежить точність майбутнього прогнозу.

Схема тренування моделі охоплює логіку вибору алгоритму, початкову ініціалізацію параметрів, цикл навчання, а також механізми збереження і валідації

проміжних результатів. Під час тренування моделі важливою умовою є оптимальне налаштування її гіперпараметрів, таких як кількість дерев у лісі, глибина дерева, мінімальна кількість зразків для розгалуження та мінімальна кількість зразків на листку. Параметри, які задаються до початку навчання моделі, впливають на її здатність узагальнювати інформацію, уникати перенавчання або недонавчання та забезпечувати стабільні результати прогнозування у різних умовах. У контексті даної роботи, основна модель побудована на основі алгоритму Random Forest, який є ансамблевим методом, що використовує велику кількість дерев рішень для прийняття колективного рішення. Параметризація цієї моделі охоплює низку критично важливих налаштувань, які визначають її структуру та поведінку під час навчання. Одним із ключових параметрів є кількість дерев у лісі. Цей параметр визначає, скільки незалежних дерев буде побудовано для подальшого голосування при прийнятті рішень. Зі збільшенням кількості дерев модель здатна покращити узагальнення, оскільки кожне нове дерево додає додатковий рівень варіативності. Водночас надмірна кількість дерев може призвести до зайвих обчислювальних витрат, що є важливим аспектом при роботі з великими обсягами даних у реальному часі.

Максимальна глибина дерев регулює, наскільки детально модель може поділити простір ознак під час навчання. Занадто велика глибина може спричинити перенавчання, коли модель надто точно відображає тренувальні дані, включаючи шум і випадкові аномалії, тоді як занадто мала — може не дозволити моделі розкрити складні закономірності у даних, що знижує її загальну точність. Пошук оптимального балансу між складністю моделі та її здатністю до узагальнення є одним із центральних завдань при налаштуванні алгоритму. Крім кількості дерев і глибини, важливим є і обсяг ознак, які доступні кожному дереву для розгляду при побудові розгалужень. Він дозволяє забезпечити різноманітність дерев, оскільки різні дерева аналізують різні підмножини ознак, що покращує загальну здатність моделі до узагальнення. У свою чергу, варіативність серед дерев сприяє зниженню корельованості між окремими моделями всередині ансамблю, що позитивно впливає на якість прогнозу.

2.4 Оцінка точності прогнозування та налаштування моделі

Побудова ефективної моделі машинного навчання, яка б забезпечувала надійне прогнозування заторів на дорогах, вимагає не лише правильного вибору алгоритму та якісних вхідних даних, а й ретельної перевірки її узагальнюючої здатності. Інструментом для досягнення цієї мети є крос-валідація — метод, що дозволяє оцінити продуктивність моделі на незалежних даних і зменшити ризик перенавчання. Особливу актуальність вона набуває у контексті задач, де використовується обмежений обсяг репрезентативної інформації або коли дані містять значну кількість варіативних факторів, як це відбувається при моделюванні трафіку. У традиційному процесі навчання модель створюється на основі певних підмножин даних (тренувального набору), після чого її точність перевіряється на окремому наборі (тестовому). Такий підхід хоча й є поширеним, однак може спричинити викривлену оцінку якості моделі, оскільки результати залежать від конкретного розподілу вибірки. У випадку ж із крос-валідацією весь набір даних ділиться на кілька частин, і кожна з них по черзі використовується як тестова, тоді решта — як тренувальна. Завдяки цьому забезпечується більш об’єктивне й стабільне вимірювання ефективності алгоритму, адже модель багаторазово перевіряється на різних підмножинах даних.

Крос-валідації дозволяє краще зрозуміти, як обрана модель реагує на зміни у вхідних параметрах, зокрема при зміні структури транспортного потоку, часу доби або погодних умов.[25] Оскільки дані, що стосуються дорожнього руху, можуть бути нерівномірно розподілені у часі або містити сезонні та просторові коливання. З її допомогою можна визначити, чи не навчається модель лише на окремих сценаріях і чи здатна вона адекватно узагальнювати закономірності, властиві іншим, не представленим у тренувальному наборі ситуаціям. Під час моделювання, наприклад, за допомогою Random Forest або нейронних мереж, ефективність алгоритму сильно залежить від таких параметрів, як глибина дерева, кількість дерев чи кількість нейронів у прихованому шарі. Проведення серії тренувань із різними конфігураціями, в поєднанні з оцінкою на валідаційних підмножинах, дозволяє визначити найкращі комбінації, які забезпечують збалансовану продуктивність без втрати узагальнення. Це також допомагає

уникнути ситуації, коли модель показує відмінні результати на тренувальних даних, але виявляється неефективною при застосуванні до нових реальних сценаріїв.

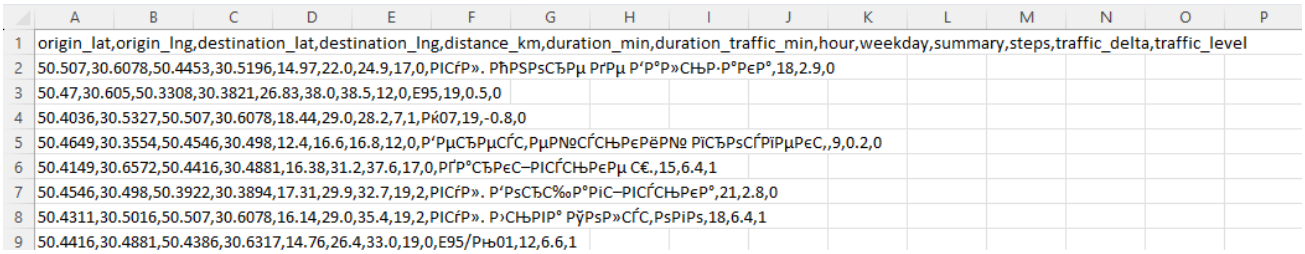
Першочергово необхідно розглянути співвідношення між передбаченим результатом та фактичним станом дорожнього руху. Якщо модель класифікує стан дороги як "затор" тоді, коли його насправді не було, це свідчить про наявність помилок першого роду (false positives).[28] Такі ситуації можуть призводити до зайвого перенаправлення трафіку або створення непотрібних обмежень, що негативно впливає на ефективність управління транспортом. У протилежному випадку, коли система не виявляє затор, який справді мав місце, маємо справу з помилками другого роду (false negatives).[29] Саме вони вважаються найбільш критичними для систем прогнозування, оскільки ставлять під загрозу достовірність інформації, на якій базується подальше планування маршруту. Часто моделі помиляються в години-пік, коли трафік має динамічну, нерівномірну структуру, і навіть невеликі затримки можуть з часом призвести до значного скупчення транспорту, яке не було передбачене за наявними історичними даними.

Також буває, якщо у навчальному наборі переважали приклади "чистих" доріг, модель може виявитися менш чутливою до ознак заторів. Такий дисбаланс призводить до зниження точності прогнозування у критичних ситуаціях, що й пояснює високий відсоток помилок другого роду. І навпаки — якщо модель перенавчилася на ознаках заторів, вона може надто часто передбачати їх виникнення, що викликає недовіру з боку користувачів. У випадку, якщо модель містить занадто багато параметрів або глибокі дерева ухвалення рішень, виникає ризик перенавчання (overfitting).[17] У такій ситуації модель показує високу точність на тренувальних даних, але погано узагальнює на нові приклади. Це легко виявити шляхом порівняння результатів на навчальному та тестовому наборах. Якщо різниця між ними є суттєвою, необхідно спростити модель, зменшити кількість ознак або застосувати регуляризацію.

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ ПРОГНОЗУВАННЯ ЗАТОРІВ

3.1 Реалізація моделі машинного навчання

Процес написання програмного забезпечення для навчання розпочинається з попередньої підготовки даних, які використовуватимуться для формування навчальної вибірки. За допомогою модулю collect_routes та платформи Google Maps Platform API були зібрані дані, а потім сформовані в набір вхідних параметрів для навчання моделі у вигляді структурованих таблиць, у форматі CSV, придатних для машинного аналізу.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	origin_lat,origin_lng,destination_lat,destination_lng,distance_km,duration_min,duration_traffic_min,hour,weekday,summary,steps,traffic_delta,traffic_level															
2	50.507,30.6078,50.4453,30.5196,14.97,22.0,24.9,17,0,PICґР». РђPSPsСЪРµ РґРµ Р'Р°Р»СЊР·Р°РєР°,18,2.9,0															
3	50.47,30.605,50.3308,30.3821,26.83,38.0,38.5,12,0,E95,19,0.5,0															
4	50.4036,30.5327,50.507,30.6078,18.44,29.0,28.2,7,1,Рќ07,19,-0.8,0															
5	50.4649,30.3554,50.4546,30.498,12.4,16.6,16.8,12,0,Р'РµСЪРµСЃС,РµРNєСЃСЊРєРёРNє РїСЪРsСЃРїРµРєС,,9,0.2,0															
6	50.4149,30.6572,50.4416,30.4881,16.38,31.2,37.6,17,0,РЃР°СЪРєС–РІСЃСЊРєРµ С€.,15,6.4,1															
7	50.4546,30.498,50.3922,30.3894,17.31,29.9,32.7,19,2,PICґР». Р'РsСЪС‰Р°РіС–РІСЃСЊРєР°,21,2.8,0															
8	50.4311,30.5016,50.507,30.6078,16.14,29.0,35.4,19,2,PICґР». Р›СЊРІР° РўРsР»СЃС,РsРіРs,18,6.4,1															
9	50.4416,30.4881,50.4386,30.6317,14.76,26.4,33.0,19,0,E95/Рњ01,12,6.6,1															

Рисунок 3.1 – Вигляд бази даних формату CSV

В процесі підготовки даних наступним кроком є їх поділ на навчальну та тестову вибірки в модулі train_model. Це дає змогу отримати об'єктивну оцінку роботи моделі після її тренування. У коді здійснюється імпорт необхідних бібліотек, зокрема таких як:

Таблиця 3.1 – Бібліотеки що використовуються в модулі «train_model»

Бібліотека	Призначення
pandas	Використовується для обробки, фільтрації, трансформації та структурування табличних даних. Завдяки pandas можливо легко імпортувати CSV-файли з дорожніми даними, аналізувати окремі стовпці,

	групувати інформацію за часовими або географічними ознаками, а також готувати вибірки для навчання моделі.
numpy	Забезпечує підтримку ефективних числових операцій, таких як масивні обчислення, лінійна алгебра та статистичний аналіз. Застосовується для оптимізації роботи з векторизованими даними та обчислення показників, необхідних для попередньої обробки або аналізу точності моделі.
scit-learn	Основна бібліотека для побудови моделей машинного навчання. Саме в її рамках реалізовано модель Random Forest, яка була використана у роботі. Крім цього, бібліотека забезпечує інструменти для розділення вибірок, налаштування гіперпараметрів, перевірки точності, валідації моделей, побудови метрик якості тощо.
joblib	Служить для серіалізації об’єктів моделі машинного навчання. Дозволяє зберігати навчену модель у вигляді файлу та згодом швидко завантажувати її без необхідності повторного навчання. Це особливо корисно для інтеграції моделі у веб-додатки або автоматизовані системи аналізу трафіку.

Після навчання моделі на зібраних і попередньо оброблених даних виникає необхідність у збереженні отриманого результату для подальшого використання, без потреби повторного проходження процесу навчання. Це збереження дозволяє використовувати модель у реальному часі, інтегрувати її в програмні рішення, тестувати на нових даних або передавати іншим системам.

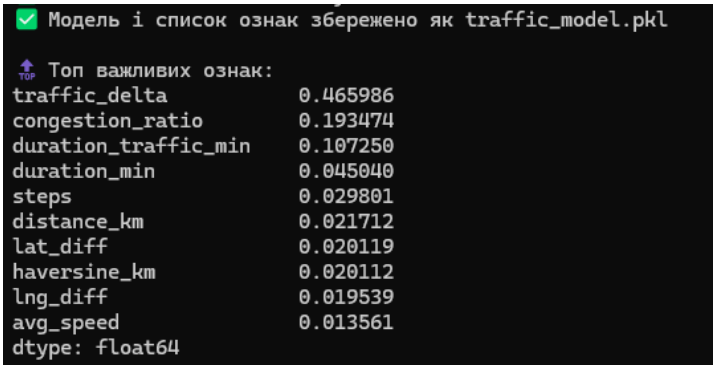


Рисунок 3.2 – Приклад навчання, виведення результатів, та збереження моделі на нашій базі даних

У процесі імплементації системи для прогнозування заторів було обрано формат збереження моделі .pkl, що є скороченою формою слова Pickle. Це вбудований модуль у мові програмування Python, який використовується для серіалізації даних. Цей формат дозволяє зберігати довільні об'єкти Python, включно зі складними структурами, такими як навчена модель машинного навчання, із усіма її параметрами, вагами та внутрішнім станом. Таким чином, після навчання, модель може бути легко записана у файл, який пізніше можна завантажити для здійснення прогнозів без необхідності повторного тренування. Сам процес збереження моделі в .pkl включає створення відповідного файлу, у який за допомогою функцій модуля joblib записується об'єкт моделі.

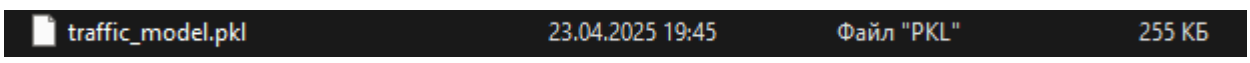


Рисунок 3.3 – Новостворена навчена модель

Після серіалізації (перетворення будь-якої структури даних у послідовність бітів) збережений файл стає частиною системи — його можна завантажити в

іншому середовищі або передати як компонент програмного забезпечення, що значно пришвидшує процес розгортання та тестування моделі.

Ця модель зберігає всі внутрішні налаштування, включно з гіперпараметрами, структурою моделі, вхідними ознаками та логікою прийняття рішень. Це забезпечує відтворюваність експериментів та стабільну роботу моделі навіть через тривалий час після початкового навчання. У процесі побудови системи прогнозування заторів ця можливість є критично важливою, оскільки прогнозування повинно виконуватись швидко та без затримок, особливо якщо система працює в умовах реального часу.

Збережена модель у цьому форматі містить всю необхідну інформацію для відтворення процесу прогнозування без необхідності повторного навчання. Наступним кроком є її завантаження для подальшого практичного використання. Реалізація цього процесу відбувається у модулі predictor.py, який виконує функцію ядра всієї системи прогнозування. Приклад коду для завантаження моделі у модуль predictor.py, для прогнозування заторів:

```
model_data = joblib.load("traffic_model.pkl")
model = model_data["model"]
feature_names = model_data["feature_names"]
```

Саме тут відбувається відтворення навченої моделі, обробка нових вхідних даних у реальному часі та видача результату прогнозу у зручному форматі. Завантаження моделі — це не лише формальна операція відкриття збереженого файлу, а повноцінний етап, що вимагає ретельного підходу до забезпечення сумісності між структурою моделі, типами вхідних даних і середовищем виконання. Після збереження моделі ключовим завданням predictor.py є забезпечення її повноцінного функціонування у складі програмного комплексу. При запуску модуля виконується ініціалізація необхідного середовища, підключення відповідних бібліотек, після чого здійснюється безпосереднє завантаження моделі з файлової системи. У межах цієї кваліфікаційної роботи файл predictor.py виконує роль модуля зв’язку між навченою моделлю та практичними задачами прогнозування.

3.2 Інтеграція з реальними даними про трафік

У рамках даної роботи реалізовано модуль під назвою collect_routes, основне призначення якого полягає в автоматизованому отриманні даних про маршрути на основі інтерфейсу Google Maps Directions API. Цей інструмент дозволяє не лише збирати інформацію про заплановані поїздки між заданими точками, але й отримувати низку важливих параметрів, які характеризують дорожню ситуацію, зокрема тривалість маршруту, довжину шляху, можливі затримки, альтернативні варіанти руху та часову прив’язку до моменту запиту.

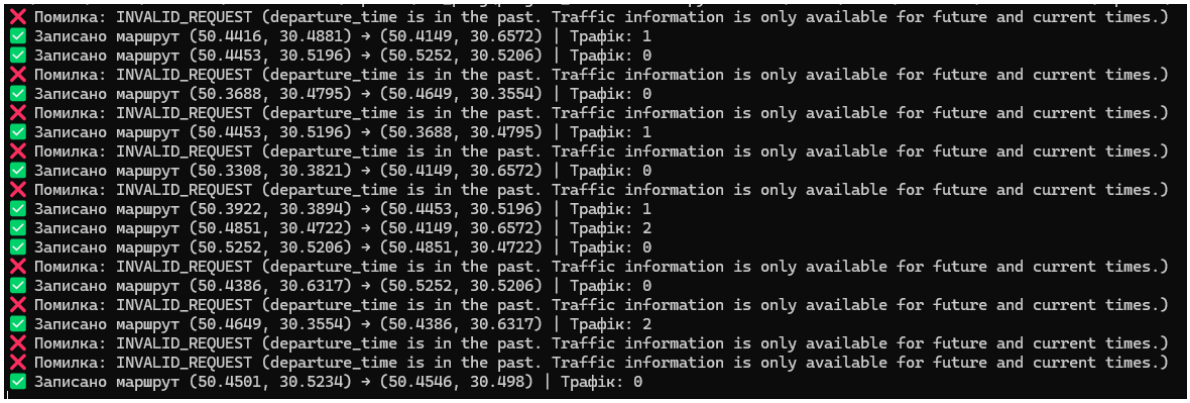


Рисунок 3.4 – Процес збору маршрутів

Модуль працює за принципом послідовного звернення до API з різними координатами початкових і кінцевих точок, що дозволяє емулювати велику кількість потенційних маршрутів у межах певної географічної зони — зокрема, в умовах міста. Після кожного запиту система отримує структуровану відповідь у форматі CSV, яку потім обробляє для витягування релевантних параметрів. До таких параметрів, як правило, належать координати початку й завершення маршруту, загальна тривалість поїздки в хвилинах, довжина маршруту в метрах, орієнтовна кількість затримок (якщо вони передбачаються системою Google), а також набір полігонів шляху, який може використовуватися для візуалізації або детальнішого аналізу дорожньої ситуації по всьому маршруту. Особливістю даного модуля є його здатність враховувати реальні умови дорожнього руху, зокрема затори, що актуальні на момент виконання запиту. Зібрані маршрути

зберігаються у форматі CSV, що забезпечує зручність подальшої обробки в рамках pipeline машинного навчання.

Таблиця 3.2 – Колонки що зберігаються в базі даних, а також їх опис

Назва колонки	Опис даних
origin_lat	Широта точки відправлення
origin_lng	Довгота точки відправлення
destination_lat	Широта точки призначення
destination_lng	Довгота точки призначення
distance_km	Відстань між точками у кілометрах
duration_min	Час поїздки без урахування трафіку (у хвилинах)
duration_traffic_min	Час поїздки з урахуванням трафіку (у хвилинах)
hour	Година доби, коли було отримано маршрут
weekday	День тижня (0 = неділя, 1 = понеділок і т.д.)
summary	Короткий опис маршруту (може містити назви вулиць)
steps	Покроковий опис маршруту
traffic_delta	Різниця між часом у трафіку та без нього (у хвилинах)
traffic_level	Рівень трафіку (0 = низький, 1 = середній, 2 = високий)

Важливо зазначити, що реалізація модуля базується на офіційній бібліотеці googlemaps для Python, яка надає стабільний і зручний доступ до сервісів Google.18

Запити до API виконувалися через HTTP-протокол із зазначенням координат початкової та кінцевої позиції, а також параметрів, що дозволяють

враховувати дорожню ситуацію саме в момент запиту. Приклад коду для збору маршрутів через API було наведено нижче:

```
locations = [
    # Центр
    (50.4501, 30.5234),  # Майдан Незалежності
    (50.4453, 30.5196),  # Золоті Ворота
    (50.4547, 30.5162),  # Арсенальна

    # Вокзали та транспортні вузли
    (50.4416, 30.4881),  # Центральний вокзал
    (50.4036, 30.5327),  # Видубичі
    (50.4546, 30.4980),  # Шулявка

    # Житлові райони
    (50.4386, 30.6317),  # Лівобережна
    (50.4149, 30.6572),  # Дарниця
    (50.4700, 30.6050),  # Троєщина
    (50.4851, 30.4722),  # Оболонь
    (50.4649, 30.3554),  # Виноградар
    (50.4311, 30.5016),  # Голосіїв
    (50.4200, 30.5200),  # Солом'янка

    # Периферія та зелені зони
    (50.3688, 30.4795),  # Конча-Заспа
    (50.5070, 30.6078),  # Броварський ліс
    (50.5252, 30.5206),  # Пуща-Водиця
    (50.3922, 30.3894),  # Жуляни
    (50.3308, 30.3821),  # Бортничі
]
```

З технічної точки зору, витяг даних було реалізовано за допомогою спеціального модуля collect_routes, написаного мовою Python. Цей скрипт звертається до Google Directions API, отримує у відповідь структуровані CSV-дані, які містять детальну інформацію про маршрут, включно з довжиною дороги,

часом у дорозі за звичайних умов, передбаченим часом у дорозі з урахуванням трафіку, переліком сегментів маршруту та їхнім станом.

Особливістю взаємодії з Google API є наявність обмежень на кількість запитів та швидкість їх виконання. Для дотримання встановлених квот було реалізовано механізм контролю навантаження: скрипт виконує запити з паузами, зберігає стан обробки та може відновити роботу з місця, де було зупинено у разі перевищення ліміту. Таким чином, забезпечено стабільну роботу процесу збору даних у довгостроковому періоді, що дозволяє накопичувати історичні дані без втрати якості. Приклад обмеження запитів на хвилину було реалізовано за допомогою програмного коду, наведеного нижче:

```
time.sleep(1.2)
```

Модель потребує точної кількості ознак у правильному порядку та відповідного масштабу значень. Наприклад, дані про швидкість руху, координати, час доби або тип дороги мають бути представлені в числовому вигляді, і в ідеалі — масштабовані відповідно до заданого діапазону, щоб уникнути зміщення акцентів у прогнозуванні. Приклад коду з модуля train_model, який вказує на послідовність даних, а також на ціль, на яку потрібно орієнтуватись:

```
features = [
    'origin_lat', 'origin_lng',
    'destination_lat', 'destination_lng',
    'distance_km', 'duration_min', 'duration_traffic_min',
    'hour', 'weekday', 'steps', 'traffic_delta',
        'lat_diff',   'lng_diff',   'haversine_km',   'avg_speed',
'congestion_ratio'
]
target = 'traffic_level'
```

3.3 Розробка інтерфейсу візуалізації

Одним із ключових завдань під час створення системи прогнозування заторів є забезпечення зручної та інтуїтивно зрозумілої взаємодії користувача з

програмним забезпеченням. У даній роботі для реалізації веб-інтерфейсу було обрано інструмент Streamlit, який є сучасною бібліотекою для створення інтерфейсів у сфері Data Science та машинного навчання. Приклад програмного коду, який імпортує бібліотеку Streamlit для додатку:

```
import streamlit as st
from streamlit_folium import st_folium
```

На відміну від класичних веб-фреймворків, таких як Flask або Django, Streamlit дозволяє будувати повноцінні інтерактивні веб-застосунки без необхідності використання HTML, CSS або JavaScript. Цей підхід значно прискорює розробку, оскільки вся логіка інтерфейсу реалізується винятково мовою Python. Інтерфейс було реалізовано у модулі app, який служить основним скриптом запуску додатку. Після ініціалізації середовища відображається заголовок програми та інструкція для користувача щодо вибору початкової та кінцевої точки маршруту. В основі інтерфейсу лежить інтерактивна мапа, створена за допомогою бібліотек folium та streamlit_folium. Мапа Києва відображається у вікні додатку, і користувач має змогу за допомогою кліків на ній позначити дві координати, що відповідають старту та фінішу маршруту.

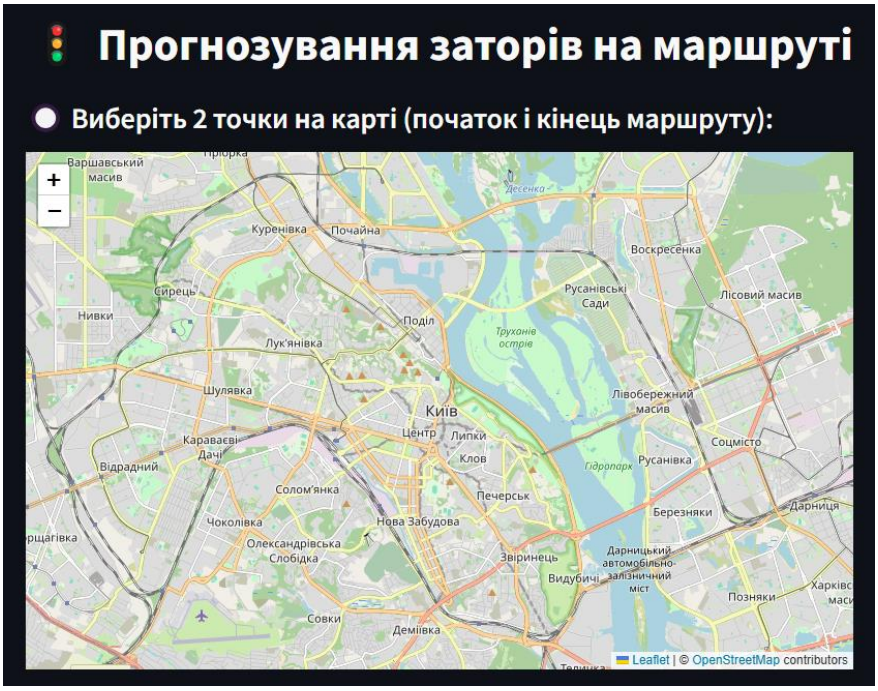


Рисунок 3.5 – Мапа Києва, що відображається у вікні додатку

Наступним етапом ми повинні вибрати дві точки на мапі.

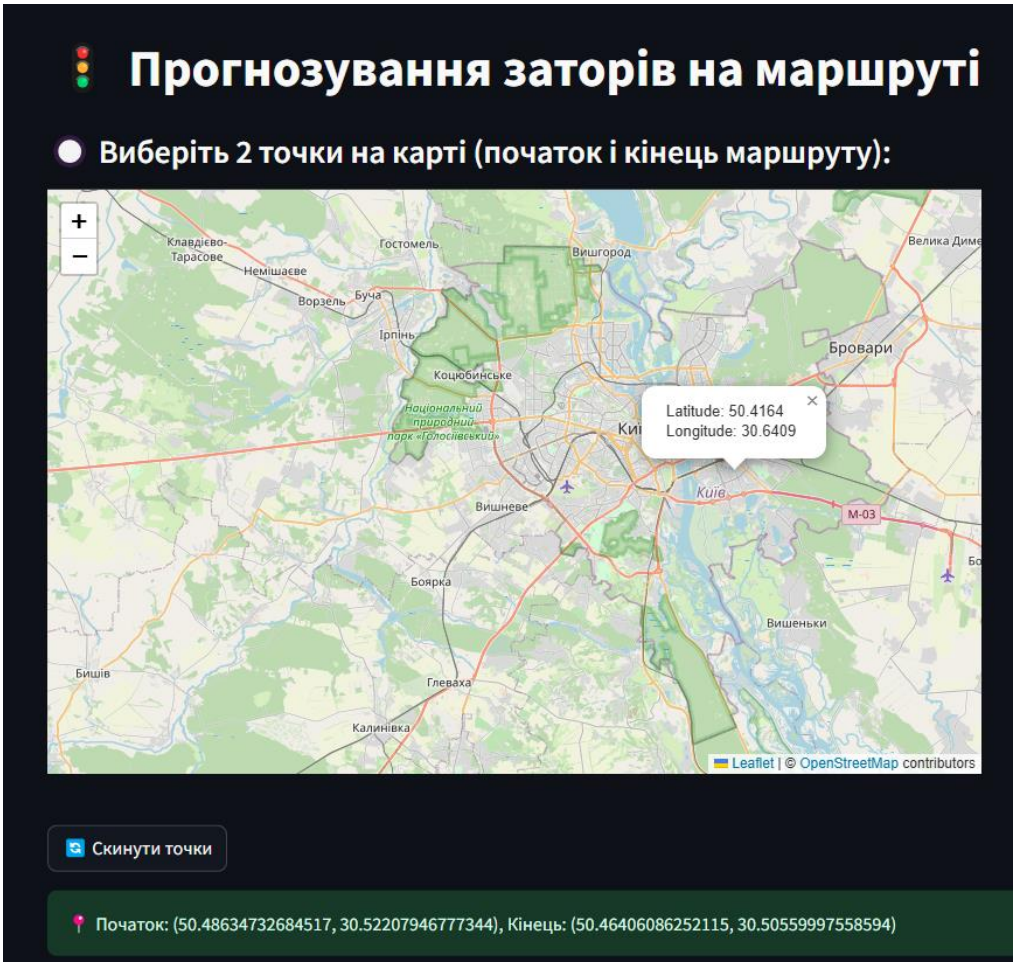


Рисунок 3.6 – Вибір точок на мапі

Streamlit підтримує збереження даних сесії за допомогою змінної st.session_state, що дозволяє відстежувати кількість обраних точок та реагувати на події, ініційовані користувачем. Приклад коду наведено нижче:

```
st.session_state.clicks = []
```

Після вибору двох координат програма автоматично обробляє запит: координати передаються до функції get_route_info(), в модулі google_traffic яка отримує детальну інформацію про маршрут, зокрема сегменти дороги, час подорожі та ймовірність заторів. Цей процес супроводжується індикацією завантаження для кращого користувацького досвіду. Приклад програмному коду, що бере інформацію про маршрут з Random Forest моделі traffic_model.pkl:

```
def get_route_info(origin, destination):
```

```
    _, feature_names = load_model_and_features()
    now = datetime.now()
       directions_result = gmaps.directions(origin, destination,
mode="driving", departure_time=now)

    if not directions_result:
        raise ValueError("Маршрут не знайдено")

    leg = directions_result[0]['legs'][0]
    steps = leg['steps']

    segments = []
    for step in steps:
        coords = polyline.decode(step['polyline']['points'])
        distance_km = step['distance']['value'] / 1000
        duration_min = step['duration']['value'] / 60
        speed = distance_km / ((duration_min / 60) + 1e-6)

        features = {
            'distance': distance_km,
            'duration': duration_min,
            'speed': speed,
            'hour': now.hour,
            'day_of_week': now.weekday(),
            'avg_speed': speed,
             'congestion_ratio': duration_min / (distance_km + 1e-
6),
            'destination_lat': step['end_location']['lat'],
            'destination_lng': step['end_location']['lng'],
            'distance_km': distance_km
        }

             full_features = {k: features.get(k, 0) for k in
feature_names}
        pred = random.choice([0, 1, 2])  # Тестовий прогноз
```

```
        segments.append({
            "coords": coords,
            "congestion": pred,
            "features": full_features
        })

    return segments
```

Після завершення обробки запиту маршрут виводиться на мапі за допомогою функції build_map(), google_traffic яка генерує HTML-представлення маршруту та вбудовує його безпосередньо у вікно Streamlit-додатку. Таким чином, користувач отримує не лише числовий прогноз, а й візуальне уявлення про маршрут із позначенням потенційно проблемних ділянок.

```
def build_map(segments):
    if not segments:
        raise ValueError("Список сегментів порожній")

           m     =     folium.Map(location=segments[0]['coords'][0],
zoom_start=13)

    color_map = {
        0: "green",
        1: "orange",
        2: "red"
    }

    for segment in segments:
        coords = segment['coords']
        color = color_map.get(segment["congestion"], "gray")
        for i in range(len(coords) - 1):
                    folium.PolyLine([coords[i],   coords[i   +   1]],
color=color, weight=6).add_to(m)

                            folium.Marker(segments[0]['coords'][0],
tooltip="Початок").add_to(m)
```

```
                            folium.Marker(segments[-1]['coords'][-1],
tooltip="Кінець").add_to(m)

    return m
```

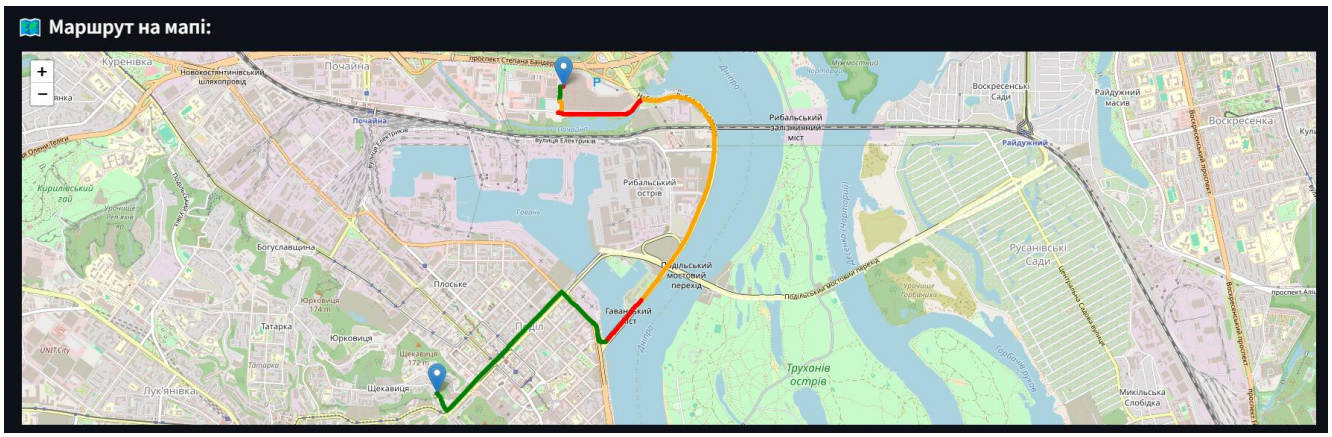


Рисунок 3.7 – Будування маршруту за нашими точками, і прогнозування затору

На даному рисунку !!! побудовано наш маршрут за допомогою функції build_map(), на якому:

- Зелений – немає затору
- Жовтий – можливе навантаження трафіку
- Червоний – затор

Кнопка «Скинути точки» виконує роль механізму скидання попередніх дій користувача, дозволяючи очистити список натиснутих координат та повторно виконати вибір початкової й кінцевої точки маршруту. Вона розміщена безпосередньо під інтерактивною мапою та доступна в будь-який момент до моменту завершення вибору обох точок. Її реалізація у коді здійснена за допомогою наступної функції:

```
if st.button(" Скинути точки"):
    st.session_state.clicks = []
```

Коли користувач натискає кнопку, активується блок логіки, що встановлює змінну сесії st.session_state.clicks у порожній список. Це означає, що будь-які

попередньо вибрані координати анулюються, і програма повертається у початковий стан, чекаючи на новий вибір точок.

Важливою перевагою використання Streamlit є його здатність швидко інтегруватися з іншими модулями системи, зокрема з модулями машинного навчання та обробки даних. Завдяки цьому користувач може взаємодіяти з усією системою прогнозування у режимі реального часу без необхідності складного налаштування або розгортання інфраструктури. Streamlit також дозволяє гнучко змінювати параметри інтерфейсу, легко адаптувати застосунок під потреби користувача та швидко впроваджувати нові функціональні можливості.

3.4 Оцінка ефективності моделі

Аналіз результатів класифікації дозволяє визначити не лише загальний рівень точності алгоритму, але й ефективність його роботи при розпізнаванні кожного з окремих класів. У рамках тестування було реалізовано трирівневу шкалу класифікації:

- перший клас відповідав ситуаціям без заторів,
- другий — помірному рівню завантаження,
- третій — ситуаціям із високою ймовірністю утворення значного затору.

Рисунок 3.8 – Трирівнева шкала реалізації

Після навчання моделі на зібраних та оброблених даних, було проведено оцінювання на незалежній тестовій вибірці. Приклад програмного коду що реалізує тестування моделі на тестовій вибірці:

```
# Розділення на train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Показниками, що були враховані, стали точність (precision), повнота (recall), F1-міра та загальна точність класифікації (accuracy). Їхнє значення є

репрезентативним для розуміння того, як добре модель розрізняє між собою різні рівні інтенсивності дорожнього руху. Отримані результати свідчать про високу здатність моделі точно ідентифікувати класи, що не пов’язані з критичним трафіком. Наприклад, дорожні умови без заторів класифікувались з абсолютною точністю та повнотою, що вказує на те, що модель впевнено розпізнає нормальні транспортні потоки. Подібна ситуація спостерігається і в середньому класі, де, незважаючи на дещо знижене значення precision, рівень recall залишається високим. Водночас найбільш складним для розпізнавання виявився клас, пов’язаний із сильними заторами. Модель частково втрачала точність саме у випадках інтенсивного трафіку, що цілком логічно, з огляду на значно меншу кількість відповідних прикладів у навчальній вибірці. Це також може бути наслідком високої схожості деяких характеристик транспортного потоку між помірним і надмірним завантаженням, що ускладнює коректне класифікування.

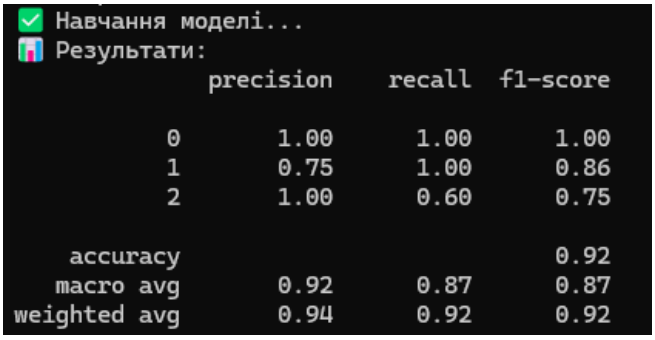


Рисунок 3.9 – Результати проведення тестувань моделі

Варто звернути увагу також на значення F1-міри, яке враховує як точність, так і повноту, забезпечуючи більш збалансовану оцінку. У випадку з цією моделлю, макро-усереднене значення F1-міри становило 0.87, а зважене середнє — 0.92, що свідчить про високий рівень узгодженості між передбаченнями моделі та фактичними значеннями, особливо у випадках з переважаючими класами.

Також модель треба перевірити на стійкість до змін у вхідних даних, і саме для цього застосовується метод крос-валідації, який дозволяє оцінити, наскільки добре модель здатна узагальнювати знання, отримані під час навчання, та

наскільки вона є стабільною при поданні нових, ще не бачених даних. Приклад програмного коду, що реалізує крос-валідацію наведено нижче:

```
# Крос-валідація
scores = cross_val_score(model, X, y, cv=5)
print(f"  Cross-validation  accuracy:  {scores.mean():.2f}  (+/-
{scores.std():.2f})")
```

У цьому контексті крос-валідація не лише служить інструментом перевірки точності, але й виступає своєрідним фільтром, що дозволяє виявити приховані слабкі місця моделі, які можуть залишитися непоміченими при класичному поділі вибірки на тренувальну та тестову частини. Суть цього методу полягає в тому, що навчальна вибірка розбивається на кілька рівних частин, одна з яких на кожній ітерації використовується для тестування, а решта — для навчання. Таким чином, модель послідовно перевіряється на кожному фрагменті даних, і в результаті обчислює середнє значення точності, а також стандартне відхилення цієї метрики. Отримані результати дають змогу не лише побачити середній рівень ефективності, а й зрозуміти, наскільки коливається якість передбачень при зміні підмножини навчальних даних.

Рисунок 3.10 – Результати крос-валідації

Середній показник точності, досягнутий у межах усіх ітерацій, становив 96%, а стандартне відхилення не перевищувало чотирьох відсоткових пунктів. Такий результат свідчить про те, що незалежно від конкретного розподілу даних модель здатна забезпечувати однаково якісний прогноз. Низький рівень коливань у метриках підтверджує, що процес навчання не супроводжувався перенавчанням або переорієнтацією на специфіку окремих фрагментів навчального набору. Завдяки цій перевірці стало очевидним, що модель демонструє не лише високу загальну точність, але й виявляє здатність адаптуватися до нових умов.

ВИСНОВКИ

На основі вивчення сучасних тенденцій розвитку інтелектуальних транспортних систем і аналізу наукових підходів до обробки транспортних потоків було встановлено, що ефективне передбачення перевантаження дорожніх ділянок є ключовим чинником для підвищення якості міського руху, скорочення часу поїздок та зменшення негативного впливу транспорту на довкілля.

У рамках дослідження було проведено аналіз основних методів машинного навчання, придатних для обробки великих обсягів даних, зокрема тих, що формуються в результаті транспортної активності в умовах міста. Особливу увагу приділено алгоритму Random Forest, як одному з найбільш інтерпретованих та ефективних засобів побудови прогнозної моделі у випадках, коли важливо враховувати множинні ознаки та складні взаємозв'язки між ними. Оцінюючи історичні дані про дорожній рух, GPS-координати та інформацію з сенсорів, модель була навчена розпізнавати характерні ознаки, які свідчать про наявність або високу ймовірність виникнення затору.

Було реалізовано систему, що включає збір, обробку та структурування даних, побудову моделі машинного навчання, а також інтерфейс, який дозволяє користувачеві взаємодіяти з системою прогнозування у доступний та зрозумілий спосіб. Проведене тестування на реальних даних з використанням API Google Maps продемонструвало, що модель здатна генерувати прогнози з високим рівнем точності, відображаючи ситуації дорожнього навантаження, які значною мірою збігаються з фактичним станом на дорогах. Таким чином, система підтвердила свою здатність бути ефективним інструментом у процесі прийняття рішень для міського планування, служб оперативного реагування, а також навігаційних сервісів.

Отже досягнуті результати свідчать про те, що машинне навчання має вагомий потенціал у сфері транспортного прогнозування, дозволяючи не лише отримувати статистичні оцінки, а й будувати динамічні моделі, що враховують часові коливання та контекстуальні фактори. Завдяки об'єднанню методів аналізу даних, алгоритмічного прогнозування та сучасних засобів візуалізації вдалося створити систему, яка не тільки виявляє закономірності в минулих транспортних

подіях, а й здатна адаптуватися до нових умов у режимі реального часу. Запропоноване рішення є універсальним за своєю структурою та може бути масштабованим для різних міст, регіонів і транспортних систем.

СПИСОК ЛІТЕРАТУРИ

1. GPS – Вікіпедія : офіційний сайт [Електронний ресурс]. – 2024. – Режим доступу: https://uk.wikipedia.org/wiki/GPS (дата звернення: 17.04.2025).
2. Svennerberg G. Beginning Google Maps API 3. – Apress, 2010. – С. 3.
3. Машинне навчання: що це таке, як працює і для чого використовується : офіційний сайт [Електронний ресурс]. – 2023. – Режим доступу: https://goit.global/ua/articles/mashynne-navchannia-shcho-tse-take-iak-pratsiuie-i-dlia-choho-vykorystovuietsia/ (дата звернення: 17.04.2025).
4. Петришин В. С., Поліщук Д. О., Ніколюк П. К. Машинне навчання // Комп'ютерні технології обробки даних. – 2022. – С. 167–170.
5. Що таке дерево рішень : офіційний сайт [Електронний ресурс]. – 2023. – Режим доступу: https://www.unite.ai/uk/%D1%89%D0%BE-%D1%82%D0%B0%D0%BA%D0%B5-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE-%D1%80%D1%96%D1%88%D0%B5%D0%BD%D1%8C/ (дата звернення: 20.04.2025).
6. AlSagri H., Ykhlef M. Quantifying feature importance for detecting depression using random forest // International Journal of Advanced Computer Science and Applications. – 2020. – С. 629.
7. Гуменюк К. В., Січко Т. В. Основи машинного навчання // Комп'ютерні технології обробки даних. – 2022. – С. 142–144.
8. Запара О. С. Дослідження методів відстеження телефонів за допомогою GPS та через відправку пакетів даних за назвою мережі. – 2024.
9. Реляційні бази даних : офіційний сайт [Електронний ресурс]. – 2023. – Режим доступу: https://foxminded.ua/reliatsiini-bazy-danykh/ (дата звернення: 20.04.2025).
10. Шолотюк Г. С. Спосіб перехресної геолокації. – 2024. – С. 22.
11. Holovina O. Сучасні технології в управлінні транспортною логістикою // International Science Journal of Management, Economics & Finance. – 2023. – Т. 2, №3. – С. 35–42.

12. Іващенко Д. С. Система прогнозування трафіку на основі просторово-часових моделей. – 2021. – С. 13.
13. Сеник А. Дослідження ефективності моделей обробки даних у системі моніторингу комп'ютерних мереж // Herald of Khmelnytskyi National University. Technical sciences. – 2024. – Вип. 335.3 (1). – С. 186–201.
14. Wang F., Xu Y. Estimating O–D travel time matrix by Google Maps API: implementation, advantages, and implications // Annals of GIS. – 2011. – Vol. 17, No. 4. – P. 199–209.
15. Що таке ансамблевий підхід до прогнозування : офіційний сайт [Електронний ресурс]. – 2023. – Режим доступу: https://chowk.lis.v.ua/maysternist/shho-take-ansambleviy-pidkhid-do-prognozuvannya.html (дата звернення: 23.04.2025).
16. Архітектура даних: короткий огляд : офіційний сайт [Електронний ресурс]. – 2023. – Режим доступу: https://data-life-ua.com/uncategorized/%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85-%D0%BA%D0%BE%D1%80%D0%BE%D1%82%D0%BA%D0%B8%D0%B9-%D0%BE%D0%B3%D0%BB%D1%8F%D0%B4/ (дата звернення: 23.04.2025).
17. Bajorath J. Integration of virtual and high-throughput screening // Journal of Chemical Information and Computer Sciences. – 2004.
18. Google Maps API : офіційний сайт [Електронний ресурс]. – 2024. – Режим доступу: https://pypi.org/project/googlemaps/ (дата звернення: 25.04.2025).
19. What is Random Forest [Електронний ресурс]. – Режим доступу: https://dida.do/what-is-random-forest (дата звернення: 25.04.2025).
20. Метод опорних векторів – Вікіпедія : офіційний сайт [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Метод_опорних_векторів (дата звернення: 26.04.2025).

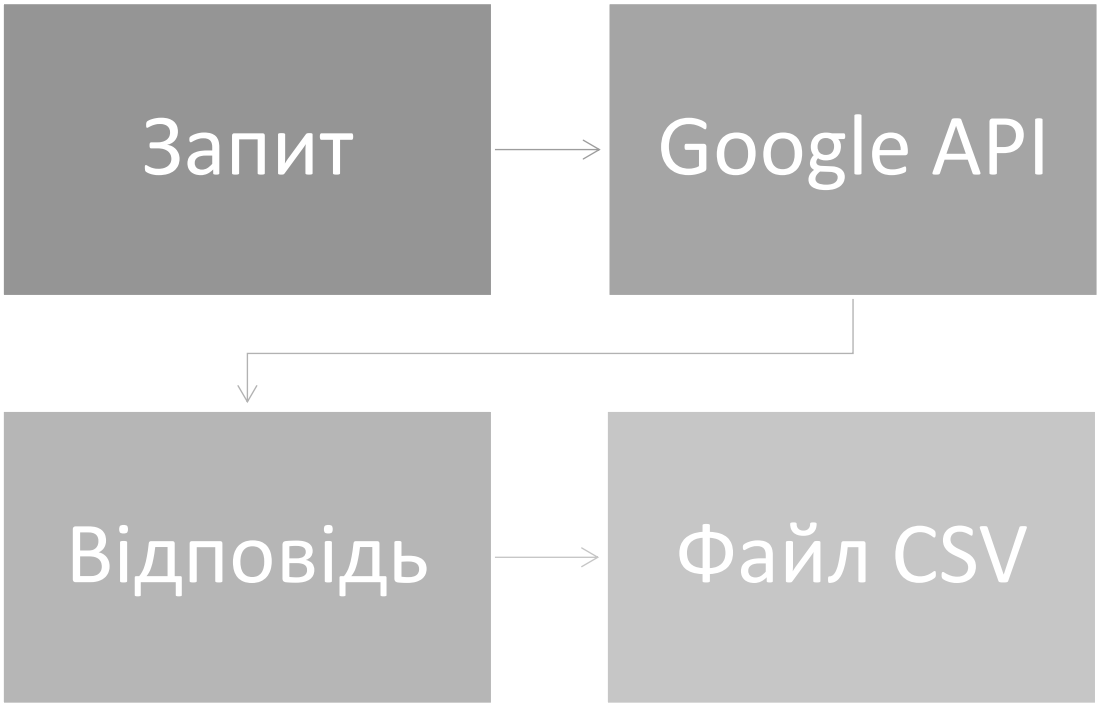
21. Данкевич В. Є., Данкевич А. Є. Інтернет речей та штучний інтелект як ключові елементи інноваційного розвитку підприємств в епоху цифрових викликів // Актуальні проблеми економіки. – 2024. – № 7. – С. 165–173.
22. Këpuska V. Int. Journal of Engineering Research and Application. – 2017. – Vol. 7, Issue 3 (Part -2), March. – P. 20–24.
23. Mahmud S.M. Hasan, et al. CSV-ANNOTATE: Generate annotated tables from CSV file // 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD). – IEEE, 2018. – P. 71–75.
24. Iranzad R., Liu X. A review of random forest-based feature selection methods for data science education and applications // International Journal of Data Science and Analytics. – 2024. – P. 1–15.
25. Bates S., Hastie T., Tibshirani R. Cross-validation: what does it estimate and how well does it do it? // Journal of the American Statistical Association. – 2024. – Vol. 119, No. 546. – P. 1434–1445.
26. Жаровський Р. О., Цірка І. П. Архітектура системи збору, передачі та зберігання даних водоспоживання у багатоквартирних будинках // Збірник тез доповідей XIII Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“. – 2024. – С. 455.
27. Muñoz-Villamizar A., et al. Integration of Google Maps API with mathematical modeling for solving the Real-Time VRP // Transportation Research Procedia. – 2024. – Vol. 78. – P. 32–39.
28. Olateju O., et al. Combating the challenges of false positives in AI-driven anomaly detection systems and enhancing data security in the cloud [Електронний ресурс]. – 2024. – Режим доступу: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4859958 (дата звернення: 28.04.2025).
29. Giray L. The Problem with False Positives: AI Detection Unfairly Accuses Scholars of AI Plagiarism // The Serials Librarian. – 2024. – Vol. 85, No. 5-6. – P. 181–189.

30. Sloan S., Talkhani R. R., Huang T., Engert J., Laurance W. F. Mapping remote roads using artificial intelligence and satellite imagery // Remote Sensing. – 2024. – Vol. 16, No. 5. – P. 839.

ДОДАТКИ

ДОДАТОК А

Схема збору маршрутів для навчання моделі



ДОДАТОК Б

Схема структури програми