Міністерство освіти і науки України
Університет митної справи та фінансів
Факультет інноваційних технологій
Кафедра комп`ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота магістра

на тему: «Навчання нейронної моделі для прогнозування трафіку в хмарних обчислювальних системах»

Виконала: студентка групи ІПЗ 23-1зм
Попова Д.Є.
(прізвище та ініціали)
Спеціальність: 121 «Інженерія програмного забезпечення»
(шифр і назва напряму підготовки, спеціальності)
Керівник: д.т.н. доц. Яковенко В.О.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент: Дніпровський державний технічний університет
(місце роботи)
доцент кафедри математичного моделювання та системного аналізу
(посада)
к.т.н., доц. Волосова Н.М.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2025

АНОТАЦІЯ

Попова Д.Є. Навчання нейронної моделі для прогнозування трафіку в хмарних обчислювальних системах.

Дипломна робота на здобуття освітнього ступеня магістр за спеціальністю 121 «Інженерія програмного забезпечення» – Університет митної справи та фінансів, Дніпро, 2025.

Кваліфікаційна робота присвячена дослідженню методів прогнозування мережевого трафіку в хмарних обчислювальних системах з використанням нейронних мереж.

У ході роботи проведено глибокий аналіз предметної області, зокрема основних принципів роботи хмарних обчислень, моделей надання послуг (IaaS, PaaS, SaaS) та структури мережевого трафіку. Особливу увагу приділено факторам, що впливають на трафік у хмарних середовищах, таким як географічне розташування користувачів, протоколи передачі даних, механізми безпеки та особливості архітектури хмарних систем.

Основним науковим результатом є розробка нейронної моделі для прогнозування трафіку в хмарних обчислювальних середовищах. Запропонована модель побудована на базі глибинних нейронних мереж і оптимізована для роботи з великими обсягами даних. Проведено навчання і тестування моделі на основі реальних даних з використанням сучасних метрик оцінки точності, таких як середня абсолютна помилка (MAE) та середньоквадратичне відхилення (RMSE). Запропоновані рішення сприяють удосконаленню методів прогнозування мережевого трафіку та можуть бути використані для подальшого розвитку систем управління трафіком.

Ключові слова: нейронні мережі, прогнозування трафіку, хмарні обчислення, машинне навчання, оптимізація ресурсів, управління мережею, глибинне навчання, аналітика великих даних.

ABSTRACT

Popova D.E. Training a neural model for traffic forecasting in cloud computing systems.

Diploma thesis for the degree of Master in specialty 121 “Software Engineering” - University of Customs and Finance, Dnipro, 2025.

The master’s thesis is devoted to the study of methods for predicting network traffic in cloud computing systems using neural networks.

In the course of the work, an in-depth analysis of the subject area was carried out, including the basic principles of cloud computing, service delivery models (IaaS, PaaS, SaaS) and the structure of network traffic. Particular attention is paid to the factors that affect traffic in cloud environments, such as the geographical location of users, data transfer protocols, security mechanisms, and features of cloud system architecture.

The main scientific result is the development of a neural model for traffic forecasting in cloud computing environments. The proposed model is built on the basis of deep neural networks and is optimized for working with large amounts of data. The model was trained and tested on real data using modern accuracy metrics such as mean absolute error (MAE) and root mean square error (RMSE). The proposed solutions contribute to the improvement of network traffic forecasting methods and can be used for the further development of traffic management systems.

Keywords: neural networks, traffic forecasting, cloud computing, machine learning, resource optimization, network management, deep learning, big data analytics.

ЗМІСТ

ВСТУП

У сучасному світі, з огляду на глобальний розвиток інформаційних технологій, особливо в контексті хмарних обчислювальних систем, зростає потреба у ефективних методах управління та прогнозування мережевого трафіку. Різке збільшення обсягів даних, що передаються через глобальні інформаційні мережі, а також зростання вимог до якості обслуговування та ефективності ресурсів у хмарних середовищах, ставлять перед дослідниками і розробниками нові виклики в області мережевого адміністрування та інтелектуальних систем прогнозування [1, 2]. Ці фактори, у свою чергу, роблять актуальним і необхідним застосування передових методів машинного навчання, зокрема нейронних мереж, для оптимізації процесів прогнозування трафіку в хмарних обчислювальних середовищах.

Хмарні обчислювальні системи мають велику кількість переваг, таких як масштабованість, гнучкість, доступність і ефективність використання ресурсів. Однак одночасно з цим, вони стикаються з проблемами, пов’язаними з перегрузкою мережі, затримками в передачі даних, а також неефективним розподілом ресурсів. Для того, щоб забезпечити стабільність та надійність функціонування таких систем, важливо мати можливість прогнозувати майбутні навантаження на мережу та відповідним чином управляти трафіком. У зв’язку з цим, застосування нейронних мереж для прогнозування трафіку стає не тільки теоретично цікавою, але й практично важливою задачею.

Актуальність теми цієї кваліфікаційної роботи зумовлена глобальним розвитком хмарних обчислювальних технологій і стрімким зростанням мережевого трафіку в різних сферах діяльності. Прогнозування трафіку у хмарних системах дозволяє не лише оптимізувати використання мережевих ресурсів, а й забезпечити ефективне управління доступом, моніторингом навантаження та масштабуванням. Завдяки своїй здатності до навчання на великих обсягах даних, нейронні мережі демонструють високі результати в

задачах прогнозування, в тому числі в контексті мережевого трафіку. Однак на сьогоднішній день існує потреба в удосконаленні методів і моделей, які б дозволяли врахувати всі чинники, що впливають на ефективність прогнозування в реальному часі.

Метою цієї роботи є розробка нейронної моделі для ефективного прогнозування трафіку в хмарних обчислювальних системах. Для досягнення поставленої мети необхідно вирішити низку конкретних завдань:

– аналіз існуючих підходів до прогнозування трафіку в хмарних системах та вибір найбільш ефективних методів машинного навчання;

– розробка архітектури нейронної мережі, яка здатна обробляти великі обсяги даних і враховувати специфіку мережевого трафіку в хмарних середовищах;

– проведення експериментів і оцінка ефективності розробленої моделі на реальних даних;

– порівняння результатів нейронної мережі з іншими методами прогнозування для визначення її конкурентних переваг;

– оптимізація моделі для досягнення високої точності прогнозування при мінімальних витратах ресурсів.

Об’єктом дослідження є хмарні обчислювальні системи, зокрема процеси, пов’язані з передачею даних і управлінням трафіком у таких системах. Предметом дослідження є методи і моделі прогнозування трафіку в хмарних обчислювальних середовищах, засновані на нейронних мережах та інших методах машинного навчання.

В роботі використовуватимуться методи машинного навчання, зокрема глибинного навчання та нейронних мереж. Це дозволить розробити модель, яка зможе ефективно прогнозувати трафік на основі великих обсягів історичних даних. Для аналізу ефективності запропонованої моделі будуть застосовуватися стандартні метрики точності, такі як середня абсолютна помилка (MAE), середнє квадратичне відхилення (RMSE) та інші, що

дозволяють оцінити її ефективність у порівнянні з іншими методами прогнозування. Крім того, в роботі буде використовуватися метод крос-валідації для оптимізації гіперпараметрів нейронної мережі та забезпечення стабільності результатів.

Практична значимість цієї роботи полягає в можливості створення ефективних інструментів для прогнозування мережевого трафіку в хмарних обчислювальних системах. Застосування запропонованої нейронної моделі дозволить операторам хмарних платформ оптимізувати процеси управління ресурсами, знижуючи ймовірність перевантаження мережі, зменшуючи затримки і підвищуючи загальну ефективність системи. Це має важливе значення для забезпечення стабільної та безперебійної роботи хмарних сервісів, що використовуються в бізнесі, науці, освіті та інших сферах. Крім того, дослідження може стати основою для подальших розробок у галузі інтелектуальних систем управління трафіком.

Наукова новизна роботи полягає в розробці нової нейронної моделі, яка дозволяє не тільки прогнозувати трафік у хмарних системах з високою точністю, але й адаптуватися до зміни умов роботи та навантаження в реальному часі. Вперше в рамках цієї роботи пропонується комбіноване використання глибинних нейронних мереж з іншими методами аналізу даних для досягнення більш точних результатів у порівнянні з існуючими підходами. Також новизною є застосування розробленої моделі для реальних умов роботи хмарних обчислювальних середовищ, що відкриває нові можливості для вдосконалення мережевого адміністрування і забезпечення стабільності в таких системах.

Структура кваліфікаційної роботи. Кваліфікаційна робота складається з трьох розділів. Обсяг роботи – 94 сторінки. Дипломна робота містить 13 рисунків та 6 таблиць. Список використаних джерел містить 15 посилань.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Характеристика хмарних платформ

Хмарні платформи стали невід'ємною частиною сучасних інформаційних технологій, забезпечуючи організаціям і користувачам доступ до потужних ресурсів обчислювальних потужностей, зберігання даних, а також різноманітних сервісів без необхідності володіти або управляти фізичними інфраструктурами [1]. Вони включають в себе набір технологій, сервісів і інструментів, які забезпечують надання обчислювальних ресурсів через Інтернет, даючи можливість користувачам доступ до таких ресурсів за вимогою. Хмарні платформи дозволяють організаціям знижувати витрати на придбання і обслуговування фізичного обладнання, зменшувати складність управління інфраструктурою та забезпечувати гнучкість у масштабуванні обчислювальних потужностей залежно від потреб.

Одна з основних характеристик хмарних платформ полягає в їх здатності забезпечувати доступ до ресурсів за моделлю «як послуга» (as a service). Існує кілька основних типів хмарних послуг, серед яких інфраструктура як послуга (IaaS), платформа як послуга (PaaS) і програмне забезпечення як послуга (SaaS). Кожен з цих типів надає різний рівень абстракції та контролю для користувачів, дозволяючи їм вибирати найбільш підходящий для своїх потреб.

Інфраструктура як послуга (IaaS) представляє собою базовий рівень хмарних послуг, при якому постачальник надає віртуалізовані обчислювальні ресурси, включаючи сервери, мережеві компоненти, сховища та інші необхідні елементи інфраструктури. Це дозволяє користувачам орендувати ресурси за потребою без необхідності їх придбання та обслуговування. На цьому рівні користувачі отримують повний контроль над операційними системами і програмним забезпеченням, яке працює на цих ресурсах, проте

відповідальність за їх управління залишається за ними. Це робить IaaS ідеальним рішенням для організацій, які потребують великої гнучкості та контролю над своїми інфраструктурними компонентами.

Платформа як послуга (PaaS) надає не тільки інфраструктуру, а й додаткові сервіси, необхідні для розробки, тестування та впровадження додатків. PaaS дозволяє розробникам зосередитися на створенні програмного забезпечення, не турбуючись про управління апаратними ресурсами чи операційними системами. Це дозволяє спростити процес розробки програм і зменшити час, необхідний для їх випуску. PaaS часто включає в себе інструменти для автоматизації розгортання, моніторингу та масштабування додатків, що робить її особливо корисною для стартапів і компаній, що швидко розвиваються.

Програмне забезпечення як послуга (SaaS) пропонує користувачам доступ до готових додатків, які працюють на хмарній платформі. Ці додатки доступні через веббраузер, що дозволяє використовувати їх без необхідності встановлення або підтримки локального програмного забезпечення. Класичні приклади SaaS включають в себе електронну пошту, CRM-системи, офісні пакети, засоби для відеоконференцій та інші вебсервіси [1, 2]. Перевагою SaaS є те, що користувачі не повинні турбуватися про оновлення програмного забезпечення або його обслуговування, оскільки всі ці задачі виконуються постачальником послуг.

Одним із основних аспектів, які визначають переваги хмарних платформ, є їхня здатність до масштабування. Це означає, що ресурси можуть бути швидко і безперешкодно додані або видалені залежно від змін у потребах користувача. Масштабування в хмарних платформах може бути горизонтальним (додавання нових одиниць обчислювальних потужностей) або вертикальним (збільшення потужності окремих серверів). Це дає змогу організаціям ефективно адаптувати свою інфраструктуру до змін у навантаженні без необхідності в інвестуванні в нові фізичні ресурси.

Важливою характеристикою хмарних платформ є також висока доступність та надійність сервісів. Відомі хмарні постачальники, такі як Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, пропонують інфраструктуру з багаторівневою архітектурою, що забезпечує безперервність роботи навіть у разі відмови окремих компонентів. Крім того, хмарні платформи забезпечують механізми для резервного копіювання даних, відновлення після збоїв і балансування навантаження, що робить їх ідеальним вибором для критичних додатків і систем [3].

Ще однією важливою характеристикою хмарних платформ є їхня безпека. Ключові постачальники хмарних послуг зазвичай забезпечують високий рівень захисту даних, що зберігаються і передаються через їхні сервіси. Це включає в себе використання шифрування, двофакторної автентифікації, систем моніторингу і реагування на інциденти, а також регулярні аудити і сертифікації. Однак, на відміну від традиційних внутрішніх інфраструктур, безпека в хмарі є спільною відповідальністю між постачальником послуг і клієнтом. Організації повинні дотримуватись рекомендованих практик щодо безпеки, забезпечувати належну політику доступу до даних і постійно моніторити використання своїх ресурсів.

Хмарні платформи також мають важливу роль у розвитку інноваційних технологій, таких як штучний інтелект (ШІ), машинне навчання та аналіз великих даних. Багато хмарних постачальників пропонують спеціалізовані сервіси і інструменти для розробки і використання моделей ШІ, що значно спрощує процес їх впровадження і використання. Хмарні платформи дозволяють масштабувати обчислення, необхідні для тренування складних моделей, що є важливим для таких областей, як обробка природної мови, розпізнавання зображень, аналіз даних у реальному часі та інші.

Крім того, хмарні технології стають важливою складовою для організацій, що прагнуть до цифрової трансформації. Вони дозволяють швидко адаптуватися до змінюваних умов ринку, автоматизувати бізнес-

процеси, вдосконалювати обслуговування клієнтів і запускати нові послуги та продукти. У цьому контексті хмарні платформи забезпечують не лише технологічну, але й стратегічну перевагу, дозволяючи організаціям бути більш гнучкими, ефективними та інноваційними [3, 4].

З огляду на численні переваги, хмарні платформи також стикаються з певними викликами та обмеженнями. Одним із основних факторів, який може бути обмежуючим, є залежність від Інтернет-з'єднання. Оскільки всі операції в хмарі виконуються через Інтернет, проблеми з підключенням або швидкістю передачі даних можуть впливати на доступність та ефективність послуг. Крім того, хоча хмарні платформи забезпечують високий рівень безпеки, зберігання чутливих даних в хмарі може бути джерелом побоювань щодо захисту приватності та відповідності вимогам регулювання. Для цього постачальники послуг пропонують різноманітні варіанти розміщення даних, включаючи приватні та гібридні хмари, які дозволяють організаціям зберігати критично важливу інформацію в більш контролюваних умовах.

Узагальнюючи, можна стверджувати, що хмарні платформи представляють собою ключову технологію для розвитку сучасних інформаційних технологій. Вони надають значні переваги, зокрема в плані масштабування, доступності, безпеки та гнучкості, що робить їх основою для численних інновацій і цифрових трансформацій. Проте використання хмарних платформ також вимагає ретельного управління безпекою, відповідності нормативним вимогам і забезпечення надійного Інтернет-з'єднання.

1.2 Принципи роботи хмарних обчислень

Принципи роботи хмарних обчислень є основою для розуміння того, як сучасні технології дозволяють здійснювати ефективне управління обчислювальними ресурсами, зберіганням даних та іншими IT-ресурсами через Інтернет. Хмарні обчислення – це модель надання обчислювальних

потужностей, яка дозволяє користувачам отримувати доступ до ресурсів через мережу (зазвичай Інтернет) без необхідності володіти або підтримувати фізичне обладнання та програмне забезпечення. Хмара надає ресурси за запитом, з максимальною гнучкістю і масштабованістю, а також із можливістю оплачувати лише фактично використані ресурси, що робить її привабливою для підприємств і організацій, а також для окремих користувачів [2-4].

Основним принципом роботи хмарних обчислень є модель «як послуга» (as a service), яка передбачає доступ до різних рівнів інфраструктури, платформ і додатків без необхідності їх локального володіння. Завдяки цьому хмарні платформи дозволяють користувачам орендувати обчислювальні потужності, програмне забезпечення або навіть цілі платформи через Інтернет за принципом підписки або на основі використаного обсягу ресурсів. Хмарні обчислення дозволяють зменшити витрати на купівлю та обслуговування фізичного обладнання, а також надають гнучкість для швидкого масштабування інфраструктури.

Іншою важливою складовою хмарних обчислень є використання віртуалізації. Віртуалізація дозволяє створювати віртуальні машини, які функціонують на фізичних серверах. Це дає змогу ефективно використовувати обчислювальні ресурси та забезпечувати їх максимальну доступність. Віртуалізація створює додатковий рівень абстракції, дозволяючи зберігати на одному фізичному сервері кілька незалежних віртуальних середовищ, кожне з яких може працювати з різними операційними системами та додатками. Таке рішення дозволяє оптимізувати використання ресурсів, що в свою чергу сприяє зниженню витрат на фізичну інфраструктуру і підвищенню гнучкості управління системами.

Масштабованість є ще однією ключовою характеристикою хмарних обчислень. Вона передбачає можливість швидко змінювати обсяг ресурсів, наданих користувачеві, в залежності від його потреб. Масштабування в хмарі

може бути як горизонтальним, так і вертикальним. Горизонтальне масштабування передбачає додавання нових одиниць обчислювальних потужностей, тоді як вертикальне масштабування – збільшення потужності вже існуючих одиниць, таких як сервери або інші інфраструктурні компоненти. Обидва види масштабування дозволяють без проблем адаптувати інфраструктуру до змінних потреб користувачів, без необхідності суттєвих капітальних витрат на фізичне обладнання.

Ще одним важливим принципом є висока доступність і надійність. Хмарні обчислення зазвичай забезпечуються через декілька дата-центрів, що географічно розташовані в різних регіонах або країнах. Така архітектура дозволяє знизити ризик втрати даних або припинення роботи сервісів у разі аварійних ситуацій, оскільки дані можуть бути репліковані та доступні з інших місць у разі відмови одного з дата-центрів [1, 3]. Це забезпечує підвищену доступність для користувачів, оскільки навіть у разі поломки одного з серверів чи компонентів, система продовжує працювати без значних перебоїв.

Принцип спільного використання ресурсів є однією з особливостей хмарних обчислень, що дозволяє постачальникам хмарних послуг ефективно використовувати свої ресурси. Завдяки багатокористувацькому середовищу ресурси, зокрема обчислювальні потужності і сховища, можуть бути розподілені серед багатьох користувачів, що дозволяє знизити витрати та забезпечити більш ефективне використання інфраструктури. Кожен користувач має доступ до свого приватного середовища, але при цьому ресурси фізичних серверів можуть бути спільно використовувані кількома користувачами.

Оскільки дані передаються через Інтернет, питання безпеки стають надзвичайно важливими для хмарних платформ. Основні постачальники хмарних послуг забезпечують високий рівень безпеки, використовуючи шифрування, двофакторну аутентифікацію, системи моніторингу та реагування на інциденти, а також постійні оновлення і патчі для операційних

систем та додатків. Залежно від типу хмари, безпека може забезпечуватись на різних рівнях – від фізичного захисту дата-центрів до захисту даних, що передаються або зберігаються в хмарі.

Інтеграція і автоматизація також є важливими принципами роботи хмарних обчислень. Хмарні платформи часто включають в себе інструменти для автоматизації процесів, таких як розгортання інфраструктури, моніторинг, оновлення програмного забезпечення і масштабування ресурсів. Це дозволяє знизити складність керування інфраструктурою та прискорити запуск нових додатків. Такі інструменти можуть включати в себе системи для автоматичного налаштування середовищ, оркестрацію контейнерів, управління життєвим циклом додатків та інші механізми.

Використання контейнеризації та мікросервісної архітектури є ще одним важливим принципом в роботі хмарних обчислень. Контейнери дозволяють пакувати додатки та їх залежності в самодостатні одиниці, які можуть бути легко розгорнуті в будь-якому середовищі. Мікросервісна архітектура дозволяє створювати додатки, що складаються з незалежних сервісів, кожен з яких може бути окремо масштабований та обслуговуваний [5]. Ці підходи дають можливість покращити гнучкість, масштабованість та надійність додатків, що працюють в хмарному середовищі.

Ще однією важливою складовою принципів хмарних обчислень є адаптивність до різних моделей використання і платіжних схем. Постачальники хмарних послуг надають різноманітні моделі ціноутворення, включаючи оплату за використання, що дозволяє користувачам платити лише за фактично спожиті ресурси. Це може включати в себе розрахунок за кількість оброблених даних, за обсяг зберігання, за години обчислювальних потужностей або за кількість виконаних запитів. Це робить хмарні платформи привабливими для організацій будь-якого розміру, оскільки вони можуть гнучко управляти своїми витратами на ресурси.

Таким чином, принципи роботи хмарних обчислень базуються на таких ключових аспектах, як використання віртуалізації, гнучке масштабування, забезпечення високої доступності, спільне використання ресурсів, високий рівень безпеки та інтеграція з іншими технологіями. Хмарні обчислення дозволяють значно спростити управління інфраструктурою, забезпечити масштабованість і надійність додатків, а також знизити витрати на фізичне обладнання. Всі ці принципи роблять хмари потужним інструментом для вирішення широкого спектру завдань у сфері інформаційних технологій.

1.3 Основні моделі хмарних обчислень

Існуючі моделі хмарних обчислень наведені в таблиці 1.1. Також в таблиці 1.2 наведено переваги та приклади до конкретних моделей хмарних обчислень [4-6].

Таблиця 1.1

Моделі хмарних обчислень

Модель	Опис
IaaS (Infrastructure as a Service)	IaaS надає основну інфраструктуру, таку як віртуальні машини, мережі, сервери, сховища. Користувачі можуть орендувати ці ресурси через Інтернет.
PaaS (Platform as a Service)	PaaS надає платформу для розробки, тестування і розгортання додатків без необхідності управління інфраструктурою.
SaaS (Software as a Service)	SaaS надає готові програми, доступні через Інтернет без необхідності їх встановлення і управління ними на локальних пристроях.
FaaS (Function as a Service)	FaaS дозволяє запускати функції або фрагменти коду на вимогу без управління серверами або інфраструктурою.
CaaS (Container as a Service)	CaaS надає управління контейнерами для розгортання додатків без необхідності керувати інфраструктурою.

Таблиця 1.2

Переваги та приклади до моделей хмарних обчислень

Модель	Переваги	Приклади постачальників	Застосування
IaaS (Infrastructure as a Service)	Гнучкість, низькі витрати на фізичні ресурси, можливість масштабування, контроль над ОС і програмним забезпеченням.	Amazon Web Services (AWS), Microsoft Azure, Google Cloud	Розгортання серверів, управління даними, великий обсяг обчислень, бізнес-додатки
PaaS (Platform as a Service)	Простота в розробці, зменшення витрат на інфраструктуру, підтримка інструментів для розробників.	Google App Engine, Microsoft Azure, Heroku	Розробка та хостинг вебдодатків, автоматизація CI/CD, інтеграція з базами даних
SaaS (Software as a Service)	Зниження витрат на ліцензії, автоматичні оновлення, простота використання, доступність через веббраузер.	Google Workspace, Microsoft 365, Salesforce	Електронна пошта, CRM-системи, офісні програми, онлайн-спільна робота
FaaS (Function as a Service)	Часова ефективність, мінімізація витрат на інфраструктуру, масштабованість за потребою.	AWS Lambda, Google Cloud Functions, Azure Functions	Обробка подій, безсерверна обробка даних, автоматизація робочих процесів
CaaS (Container as a Service)	Легке управління контейнерами, автоматичне масштабування, підтримка різних середовищ розгортання.	Google Kubernetes Engine, Azure Kubernetes Service, Amazon EKS	Розгортання мікросервісів, хмарні додатки, автоматизація управління контейнерами

1.4 Огляд структури мережевого трафіку в хмарах

Огляд структури мережевого трафіку в хмарах є важливим аспектом для розуміння того, як відбувається передача даних в середовищі хмарних обчислень. З розвитком хмарних технологій і переходом підприємств на хмарні сервіси, мережевий трафік стає одним із критичних елементів для забезпечення ефективності, безпеки та надійності роботи [6, 7]. Мережевий трафік в хмарах охоплює всі аспекти передачі даних між користувачами, серверами, додатками, а також між різними компонентами інфраструктури хмари. Він включає в себе комунікації між локальними клієнтами і віддаленими дата-центрами, а також між різними підсистемами хмарної інфраструктури, такими як віртуальні машини, сервіси баз даних, обчислювальні ресурси та мережеві шлюзи. Структура мережевого трафіку в хмарах є надзвичайно складною, оскільки вона включає в себе численні компоненти, що взаємодіють між собою за допомогою різних протоколів і технологій.

Одним з ключових аспектів структури мережевого трафіку є те, що хмара зазвичай є глобальною мережею, що складається з кількох географічно розподілених дата-центрів. Це забезпечує високу доступність і масштабованість сервісів, однак також додає складності в управлінні мережевим трафіком. Кожен з цих дата-центрів може мати свою внутрішню мережу, а зв’язок між ними здійснюється через спеціалізовані канали. Це означає, що трафік між різними компонентами хмари може проходити через різні фізичні та логічні шари, що значно ускладнює його моніторинг, керування і оптимізацію.

Основними елементами мережевого трафіку є вхідний і вихідний трафік. Вхідний трафік – це дані, що надходять з Інтернету або з локальних мереж до хмарних ресурсів. Вихідний трафік – це дані, які хмарні ресурси передають назад до користувачів або інших зовнішніх систем. Зазвичай вхідний трафік

вимагає більших зусиль для фільтрації, аналізу та керування, оскільки він може містити непередбачувані запити, включаючи атаки типу DDoS (розподілені відмови в обслуговуванні) [1, 4, 5], в той час як вихідний трафік більш контрольований і часто використовується для виконання запитів користувачів.

Мережевий трафік у хмарі також поділяється на внутрішній і зовнішній. Внутрішній трафік відбувається між різними компонентами однієї хмарної інфраструктури, такими як віртуальні машини, сервіси зберігання даних, бази даних, сервіси додатків та інші. Цей трафік передає дані на великі відстані всередині хмари або між її окремими частинами. Зовнішній трафік, у свою чергу, стосується комунікацій з зовнішніми користувачами або іншими системами поза межами хмари. Це може бути обмін даними з клієнтами або з іншими хмарними сервісами, що розташовані в різних географічних регіонах або навіть у інших хмарних провайдерів.

Особливу увагу в структурі мережевого трафіку заслуговує використання різних мережевих протоколів. Хмара використовує стандартні інтернет-протоколи, такі як TCP/IP, для забезпечення передачі даних між різними вузлами мережі, а також може використовувати більш спеціалізовані протоколи для ефективної організації трафіку. Наприклад, віртуалізація мережі в хмарах часто використовує протоколи, орієнтовані на забезпечення ізоляції трафіку між різними віртуальними машинами або контейнерами, такими як VXLAN (Virtual Extensible LAN), що дозволяє розширювати мережеву інфраструктуру хмари, не залежачи від фізичних обмежень.

Протоколи на рівні додатків також можуть істотно впливати на структуру мережевого трафіку в хмарах. Наприклад, використання HTTP/HTTPS для обміну даними між користувачами і вебсервісами створює специфічні патерни трафіку, які включають в себе запити на отримання даних, передачу файлів або відправку форм. В той час як для передачі великих обсягів даних між різними хмарними сервісами можуть використовуватися протоколи

на кшталт FTP або SFTP, які краще підходять для великих файлів та надають більше контролю за передачею даних [7].

Ще одним важливим аспектом мережевого трафіку в хмарах є використання концепції зон безпеки. Хмара, як правило, має багаторівневу систему безпеки, включаючи брандмауери, шлюзи, системи виявлення вторгнень (IDS) та інші механізми для захисту трафіку від несанкціонованого доступу або атак. Трафік між різними частинами хмари може бути ізольований за допомогою віртуальних приватних мереж (VPN), що забезпечують шифрування даних і контрольоване з'єднання між різними компонентами.

Оскільки хмара підтримує різні моделі обчислень, такі як IaaS, PaaS і SaaS, мережевий трафік в кожній з цих моделей може мати різні характеристики. У моделі IaaS користувачі мають більше контролю над віртуальними машинами та іншими обчислювальними ресурсами, що дозволяє їм управляти трафіком між цими ресурсами. У моделі PaaS трафік в основному стосується взаємодії з платформою для розробки та розгортання додатків, а в моделі SaaS переважно здійснюється обмін даними між кінцевими користувачами і вебдодатками, розміщеними в хмарі.

З огляду на ці фактори, ефективне управління мережевим трафіком є важливою складовою для забезпечення оптимальної роботи хмари. Важливою частиною цього є моніторинг і аналіз мережевого трафіку для виявлення аномалій, таких як надмірне навантаження на канали зв'язку, а також для забезпечення безпеки і запобігання атакам. Інструменти моніторингу, такі як системи управління мережею та аналізатори трафіку, дають змогу хмарним провайдерам та адміністраторам користувачів оптимізувати мережевий трафік, зменшуючи затримки і покращуючи ефективність.

Крім того, структура мережевого трафіку повинна враховувати питання масштабованості і гнучкості. Хмара повинна мати можливість швидко адаптуватися до змін у обсягах трафіку, які виникають через зміни в попиті на ресурси або через зміни в роботі додатків. Це вимагає впровадження

технологій, які дозволяють автоматично масштабувати мережеві ресурси, забезпечуючи їх відповідність потребам користувачів [8].

Впровадження нових технологій, таких як 5G, віртуалізація мереж, мережі з підтримкою контейнерів і мікросервісів, а також використання хмарних сервіси для керування мережами, дозволяє покращити структуру мережевого трафіку в хмарах, підвищити ефективність, знизити затримки і забезпечити більш стабільне і безпечне обслуговування користувачів. Трафік між сервісами і користувачами буде ставати більш розподіленим і більш динамічним, з максимальною орієнтацією на гнучкість, безпеку і ефективність у взаємодії з великими даними і обчислювальними ресурсами.

1.5 Аналіз досліджень

Стаття [1] зосереджена на проблемі прогнозування трафіку, яке є важливою складовою інтелектуальних транспортних систем (ITS) та має велике значення для покращення досвіду користувачів і управління міським рухом. Для вирішення цієї проблеми пропонується нова модель на основі адаптації доменів за допомогою технології edge computing, що є перспективною для забезпечення реального часу та точних прогнозів потоку трафіку. В роботі розроблено функції оптимізації для карти ознак і дискримінатора, які дозволяють адаптувати характеристики трафіку з різних джерел даних, використовуючи методи суперечливої адаптації доменів. Це дозволяє краще узгоджувати ознаки цільового і вихідного доменів при прогнозуванні таких параметрів, як потік трафіку та рівень заповнюваності. Застосувавши цю модель до реальних даних, автори показують, що запропонована методика забезпечує кращу точність прогнозування, навіть коли кількість доступних навчальних даних обмежена, порівняно з іншими алгоритмами.

Стаття [2] розглядає методи прогнозування трафіку, що є важливою складовою інтелектуальних транспортних систем (ITS), оскільки це має велике значення для покращення досвіду користувачів та управління міським рухом. Вона пропонує нову модель, яка використовує просторово-часові графові згорткові мережі (STGCN-EC) в системі edge computing для прогнозування потоку трафіку. Спершу, автори моделюють дорожню мережу як граф, поділяючи її на підграфи, зважаючи на просторові кореляції, що дозволяє виконувати прогнозування трафіку кожним вузлом окремо. Далі, з урахуванням географічної інформації та часової схожості потоку трафіку, розробляється просторово-часова графова згорткова мережа для ефективного виявлення просторово-часових особливостей у кожному підграфі для прогнозування потоку трафіку. Для підвищення ефективності навчання застосовується перенесене навчання для спільного використання моделей між різними вузлами. Результати симуляцій на реальних даних показують, що запропонований підхід покращує точність прогнозування та ефективність навчання в системі edge computing.

Стаття [3] досліджує прогнозування трафіку споживачів мобільних мереж для оптимізації використання послуг, зокрема для аналізу мобільного трафіку, що є важливим для розкриття особливостей міського середовища. У роботі використовуються методи прогнозування часових рядів, зокрема авторегресивно-інтегровані рухомі середні (ARIMA) та варіанти довго- та короткочасної пам’яті (LSTM). Дослідження поділяється на дві частини: спочатку за допомогою моделі ARIMA мобільний трафік розкладається на чотири основні компоненти – сезонність, тренд, залишкову частину та шум. Далі на основі зібраних даних тренується рекурентна нейронна мережа LSTM, яка прогнозує кількість користувачів мобільного трафіку на наступні десять років. Запропонований підхід дозволяє отримати глибоке розуміння використання мобільного трафіку в великих міських агломераціях і допомагає у прийнятті рішень на основі прогнозів майбутнього трафіку.

Стаття [4] розглядає віртуалізацію мережі як основний елемент архітектури майбутніх бездротових комунікаційних мереж, яка є ключовою для реалізації різноманітних додатків і технологій 5G та майбутніх мереж, таких як мережевий поділ (Network Slicing), Edge Computing і програмно-орієнтовані мережі (SDN). У таких складних системах автоматизація мережі є необхідною для подолання проблем управління мережами. Прогнозування мережевого трафіку є одним з важливих проактивних підходів у керуванні та плануванні мереж. У роботі пропонується реальний тестовий стенд для прогнозування трафіку бездротової мережі на основі попередньо натренованих моделей глибокого навчання. В якості середовища для тренування та тестування використовується віртуалізована ядрова мережа 5G. Прогнозування мережевого трафіку здійснюється під час того, як обладнання користувача виконує різні дії. Крім результатів моделей прогнозування, обговорюються пропозиції щодо вдосконалення тестового стенду та можливі напрямки його використання в майбутньому.

Стаття [5] розглядає Інтернет транспортних засобів (IoV) як новітню концепцію, що є розподіленою мережею різних транспортних засобів, оснащених датчиками, актуаторами, технологіями та додатками для з’єднання і обміну даними через Інтернет. Головною метою IoV є створення платформи для покращення комунікації та якості обслуговування (QoS) для транспортних засобів, пішоходів та дорожньої інфраструктури в реальному часі через канали взаємодії, такі як Vehicle-to-Vehicle (V2V), Vehicle-to-Pedestrian (V2P), Vehicle-to-Infrastructure (V2I), Vehicle-to-Network (V2N) та Vehicle-to-Cloud (V2C). Однак зростаюча кількість транспортних послуг створює серйозні проблеми для дослідників, такі як прогнозування трафіку в реальному часі, розміщення послуг, безпека, надійність та маршрутизація. Основна увага цієї роботи зосереджена на проблемі прогнозування трафіку в кількох регіонах для багатьох типів транспортних засобів одночасно. Розглянуті моделі прогнозування трафіку дозволяють здійснювати проактивне управління

системами, забезпечуючи точні прогнози в реальному часі, а також аналізувати щільність трафіку в просторі та часі для різних типів транспортних засобів. Проте існуючі дослідження не можуть одночасно прогнозувати трафік для багатьох регіонів і класів транспортних засобів. Метою цієї роботи є створення проактивної платформи, яка здатна приймати рішення на основі інтегрованої моделі прогнозування трафіку, що поєднує графові нейронні мережі (GNN) і Gated Recurrent Units (GRU), тобто модель прогнозування трафіку на основі просторово-часових графових нейронних мереж (STGNN). Модель STGNN використовує GNN для обробки просторових даних, що складаються з щільності багатокласового трафіку, що дозволяє отримати графові вбудовування, які потім використовуються в GRU для виділення часових ознак. Такий підхід дозволяє прогнозувати щільність трафіку для багатьох класів транспортних засобів і забезпечує проактивність платформи IoV. Крім того, результати виконаних експериментів показують, що на основі запропонованої моделі прогнозування трафіку можна створити інтелектуальну платформу, здатну точно аналізувати складний та нелінійний трафік.

Стаття [6] розглядає автономні та з’єднані автомобілі (ACC) разом із технологією edge computing (EC) як перспективне рішення для досягнення екологічно чистого та інтелектуального транспорту в смарт-містах. Основною метою роботи є вирішення проблеми короткострокового прогнозування трафіку, яке є важливим для успішної реалізації додатків ACC, в рамках архітектури мультидоступного EC (MEC), що має обмеження, відмінні від традиційного хмарного обчислення. Для цього спершу створено експериментальну платформу, орієнтовану на дані, яка сприяє розробці алгоритмів прогнозування трафіку. Далі пропонується нова модель короткострокового прогнозування трафіку, яка інтегрує модель світлофора та модель швидкості транспортних засобів, з урахуванням обмежених обчислювальних ресурсів серверів MEC. Автори зазначають, що вплив

світлофорів є складним і не був детально досліджений у більшості, якщо не всіх, попередніх роботах. У цій роботі моделюється час черги, коли водій під'їжджає до перехрестя і зіштовхується з червоним світлом. Крім того, для прогнозування швидкості транспортного засобу пропонується нова модель з низькою складністю, що враховує періодичні характеристики та просторово-часові кореляції динамічних дорожніх подій. Результати експериментів показують, що запропонована модель прогнозування швидкості транспорту досягає майже такої ж точності, як відома модель Long Short-Term Memory (LSTM), але вимагає значно менше обчислювальних ресурсів. Результати експериментів також підтверджують, що, оскільки інтегрована модель враховує ефекти світлофора та індивідуальну поведінку водія, вона більш ефективно відображає реальні зміни в дорожній ситуації.

Стаття [7] присвячена прогнозуванню швидкості потоку трафіку, що є важливою задачею для інтелектуальних транспортних систем. Однак прогнозування швидкості потоку трафіку стикається з низкою проблем, таких як динамічність даних у часі та просторі, а також накопичення помилок при довгострокових прогнозах. В роботі запропоновано модель на основі механізму уваги та згорткових мереж (ACN), яка поєднує згорткові шари з механізмом уваги для виявлення просторово-часових характеристик даних. Модель використовує шар уваги для моделювання кореляції між прогнозованими даними і історичними даними, що дозволяє визначити важливі ознаки та оптимізувати результат. Експерименти на двох реальних наборах даних показали, що модель ACN дозволяє покращити точність прогнозування швидкості трафіку.

Стаття [8] розглядає проблему прогнозування трафіку, яка є важливим аспектом інтелектуальних транспортних систем. У останні роки було запропоновано багато моделей для покращення точності прогнозів, однак більшість із них мають недоліки: вони зосереджуються лише на залежностях між вузлами, ігноруючи залежності між ребрами, а також недостатньо

враховують високодинамічні просторові залежності в трафік мережах у часі. У цій роботі запропоновано нову модель – багатошарову динамічну просторово-часову графову згорткову мережу з вбудовуванням ознак ребер (MDSTGCN). У просторовому вимірі модель створює динамічну матрицю суміжності та гіперграф, використовуючи дифузійне згортання для виявлення просторових кореляцій. В тимчасовому вимірі розроблено багатошарову тимчасову згорткову архітектуру для захоплення тимчасових динамік трафіку на різних масштабах. Експерименти на чотирьох реальних наборах даних показали, що запропонована модель перевищує за точністю існуючі базові моделі.

Стаття [9] розглядає проблему дорожнього трафіку, яка є однією з найбільших сучасних викликів через такі фактори, як збільшення кількості транспортних засобів, обмежена пропускна здатність доріг та некординація світлофорів. Ці проблеми призводять до зайвих витрат пального та негативного впливу на навколишнє середовище. В дослідженні використовується набір даних про трафік з Kaggle, на основі якого застосовуються методи машинного навчання (ML) та глибокого навчання (DL) для вирішення проблеми заторів. Для прогнозування напрямку руху трафіку на певний день були реалізовані різні моделі, зокрема машини опорних векторів (SVM), випадкові ліси (RF), дерева рішень (DT), наївний баєсів класифікатор (NB) та штучні нейронні мережі (ANN). Аналіз точності реалізованих моделей показав, що модель ANN показує кращі результати порівняно з іншими базовими моделями машинного навчання. Результати зберігаються в хмарному сховищі для постійних записів, до яких користувачі можуть отримати доступ для моніторингу трафіку та уникнення заторів.

Стаття [10] розглядає короткострокове прогнозування різних атрибутів трафіку, що є одним з основних інструментів у плануванні транспортних систем. Зокрема, точне прогнозування потоку трафіку є основою для ефективного управління рухом. У роботі розглядається випадок

короткострокового прогнозування потоку трафіку на міських магістралях у Бремені (Німеччина), використовуючи реальні дані, зібрані з 7 детекторів на основі петель, встановлених у центрі міста. Для вирішення цієї задачі запропоновано два різних підходи – лінійну регресію та графові згорткові нейронні мережі. Обидві моделі застосовуються до даних за 11 тижнів, з яких три тижні використовуються для тестування точності моделей. Експериментальні результати показують, що обидві моделі демонструють схожу ефективність і в середньому досягають 19% середньої абсолютної відсоткової помилки під час пік часу, зокрема вранці та ввечері.

Стаття [11] розглядає проблему прогнозування потоку трафіку як типову задачу прогнозування просторово-часових даних, оскільки дані про трафік демонструють складні патерни, а обробка трафік-графів є складною задачею. Більшість існуючих методів прогнозування потоку трафіку не здатні адекватно моделювати динамічні просторово-часові особливості, що призводить до незадовільних результатів прогнозів. У роботі запропоновано модель Attention Based Multi-Unit Spatial-Temporal Network (AMU-STN), яка покликана вирішити проблему прогнозування потоку трафіку. Модель складається з трьох основних блоків: просторово-часового блоку уваги для ефективного захоплення динамічних просторово-часових кореляцій, блоку витягування просторово-часових ознак для видобування довготривалих ознак і блоку прогнозування для передбачення стану ознак. Для обробки довгострокових прогнозів модель використовує кілька шарів блоків уваги та витягування ознак, застосовуючи залишкові з’єднання для запобігання зникненню градієнтів при збільшенні глибини мережі. Експериментальні результати на двох реальних наборах даних з системи вимірювання продуктивності Caltrans (PeMS) демонструють, що запропонована модель перевищує за ефективністю сучасні методи. Також була проведена аналіз вагою матриці уваги, результати якого підтверджують ефективність механізму уваги та підвищують інтерпретованість моделі.

Стаття [12] розглядає проблему прогнозування трафіку в мобільних мережах у контексті зростаючої складності та масштабів таких мереж завдяки технологіям 4G та 5G, а також Інтернету речей (IoT), які збільшують мобільний трафік в десятки разів кожні п’ять років. Для ефективного управління такими великими та складними мережами важливим є досягнення самокерованих, інтелектуальних автономних мереж. Точне прогнозування трафіку дозволяє приймати автономні рішення у плануванні, управлінні, розподілі ресурсів та адаптаціях мережі. У роботі запропоновано ефективну систему прогнозування трафіку мобільної мережі, яка базується на вибраних показниках ефективності (KPIs), що мають високу кореляцію з трафіком. Система використовує алгоритм кластеризації на основі сітки для групування клітин за їх просторовим розташуванням. У статті порівнюються кілька методів обробки часових рядів для прогнозування таких KPIs, як максимальна кількість підключених користувачів і обсяг передаваних даних PDCP. Для порівняння використовуються різні моделі: модель сталості як базова, сезонна авторегресивно-інтегрована модель (SARIMA) як класичний статистичний метод та методи машинного навчання, такі як LSTM (Long-Short Term Memory) і GRU (Gated Recurrent Unit). Результати експериментів показують, що архітектури рекурентних нейронних мереж (LSTM і GRU) дають обіцяючі результати, значно перевершуючи класичні статистичні моделі прогнозування. Модель GRU, навчена на одному кластері, була успішно застосована для прогнозування KPI інших кластерів з подібною динамічною поведінкою, що продемонструвало ефективність трансферного навчання.

Стаття [13] розглядає проблему точного короткострокового прогнозування потоку трафіку, що є важливим для надання інформації про стан доріг в найближчому майбутньому, що дозволяє інтелектуальним транспортним засобам планувати і коригувати маршрути для запобігання заторам. Багато моделей для прогнозування короткострокового потоку трафіку вже було запропоновано, але більшість з них фокусується на

прогнозуванні всього трафіку в мережі, що може призвести до кількох проблем, таких як велика складність і масштаб мережі, що ускладнює навчання моделі, а також високі вимоги до обчислювальних потужностей центрального сервера, що може збільшити навантаження на сервери та ризик витоку конфіденційних даних. У цій роботі запропоновано модель Spatial-Temporal Attention Graph Convolution Network на Edge Cloud (STAGCN-EC), яка спершу розбиває всю мережу трафіку на декілька частин для зменшення її масштабу та складності. Потім кожна частина мережі призначається певному дорожньому пристрою (RSU) для навчання, що дозволяє уникнути необхідності обробляти всі дані на центральному сервері. Крім того, застосовано просторово-часову увагу та модуль витягування ознак, що підходить для пристроїв з обмеженими обчислювальними потужностями, таких як RSU, для захоплення просторово-часових залежностей і прогнозування потоку трафіку. Модель перевірена на двох наборів даних, зібраних на шосе в районах 7 і 4 в Каліфорнії, і результати експериментів показали, що модель добре справляється як з точністю прогнозування, так і з ефективністю порівняно з п’ятьма базовими методами.

Стаття [14] розглядає проблему прогнозування потоку трафіку, що є однією з основних задач у системах інтелектуального транспорту (ITS). Основною проблемою є покращення точності моделей та захоплення динамічних тимчасових і нелінійних просторових залежностей для поліпшення точності прогнозування потоку трафіку. Використання реальних даних є одним із способів покращити точність моделювання просторово-часових кореляцій. Однак реальні дані про потік трафіку не є строго періодичними через випадкові фактори, що може призводити до відхилень. У роботі запропоновано модель Spatial-Temporal Similar Graph Attention Network (STSGAN), яка вирішує цю проблему, фокусуючись на захопленні та моделюванні тимчасових порушень у реальних періодичних даних. Модуль просторово-часових графових згорток у STSGAN дозволяє захоплювати

локальні просторово-часові взаємозв’язки в даних про трафік, а модуль періодичної схожої уваги – справлятися з нелінійними даними потоку трафіку. Експерименти на трьох наборах даних показали, що запропонована модель демонструє найкращу ефективність серед усіх методів.

Стаття [15] розглядає проблему прогнозування мережевого трафіку в умовах постійного зростання обсягів мобільних даних через розвиток мобільних сервісів. Точне прогнозування мережевого трафіку є критично важливим для ефективного планування мереж і розподілу ресурсів. Модель Informer, заснована на архітектурі Transformer, була недавно оптимізована для прогнозування часових рядів, але має недолік, оскільки акцентує шум у кожному окремому часовому ряді, що призводить до перенавчання або зменшення надійності, що негативно впливає на точність прогнозів. Для вирішення цих проблем запропоновано адаптивну модель прогнозування мережевого трафіку під назвою VSNT. Ця модель використовує метод Варіаційної Модальної Декомпозиції (VMD) для адаптивного розкладу оригінального мережевого трафіку на компоненти Інтригуючих Модальних Функцій (IMF). Для збереження взаємозв’язків між цими компонентами здійснюється комплексне навчання представлень перед тим, як вони потрапляють до нейронної мережі. Мережа використовує структуру кодувальника-декодувальника і інтегрує інструменти, такі як SCI-Block та механізми уваги. SCI-Block допомагає витягувати багатошарові мережеві атрибути, а механізм уваги акцентує важливі ознаки. Порівняно з більш складною моделлю Informer, VSNT знижує значення середньоквадратичної помилки (RMSE) і середньої абсолютної відсоткової помилки (MAPE) на 51% і 35% відповідно, що робить її надійним рішенням для точного прогнозування мережевого трафіку і відкриває нові можливості для досліджень у сфері прогнозування мережевого трафіку та аналізу часових рядів.

1.6 Висновки до першого розділу

Отже, після аналізу предметної області можна навести наступні висновки:

1. Розділ містить ґрунтовний аналіз хмарних платформ, який дає змогу зрозуміти їх функціональну структуру, компоненти та основні характеристики. Це дозволяє отримати чітке уявлення про технології, які складають основу хмарних обчислень, а також про їх потенціал для реалізації різноманітних задач у сучасному світі;

2. Огляд принципів роботи хмарних обчислень розкриває ключові механізми взаємодії між користувачами і хмарними ресурсами. З цього випливає важливість розуміння основних принципів, таких як віртуалізація, автоматичне масштабування, абстракція і віддалене управління, що є необхідними для забезпечення ефективності та безпеки у хмарних системах;

3. Визначення основних моделей хмарних обчислень (IaaS, PaaS, SaaS) дозволяє окреслити різні способи надання обчислювальних ресурсів і послуг кінцевим користувачам, що є основою для вибору оптимальної моделі в залежності від потреб конкретних бізнес-процесів або технічних вимог;

4. Поглиблений аналіз структури мережевого трафіку в хмарах демонструє, як важливо забезпечити стабільну і ефективну передачу даних між різними компонентами хмари та її користувачами. Це є основою для забезпечення продуктивності та масштабованості хмарних платформ, особливо в умовах великих обсягів даних та високих вимог до пропускної здатності.

Таким чином, розділ висвітлює ключові аспекти хмарних обчислень, що дає змогу розробити стратегії для їх ефективного використання, оптимізації та забезпечення стабільної роботи, враховуючи безліч факторів, що впливають на їх функціонування.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ МЕТОДІВ ПРОГНОЗУВАННЯ ТРАФІКУ В ХМАРНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ

2.1 Традиційні методи прогнозування трафіку

Традиційні методи прогнозування трафіку [8-10], які були започатковані ще на початку, доцільно представити у вигляді таблиці 2.1.

Таблиця 2.1

Традиційні методи прогнозування трафіку

Метод прогнозування	Опис	Застосування	Приклад
Метод середнього значення	Цей метод базується на використанні середніх значень для прогнозування трафіку.	Короткострокові прогнози для стабільного трафіку.	Прогнозування кількості відвідувань вебсайту.
Метод екстраполяції	Метод використовує дані про минулі тенденції для прогнозування майбутніх значень.	Прогнозування для довгострокових періодів.	Прогнозування обсягів трафіку на основі трендів.
Метод експоненціального згладжування	Враховує як рівень трафіку, так і його зміни в часі.	Прогнозування при наявності даних з сезонними змінами.	Прогнозування попиту в інтернет-магазині.
Метод рухомого середнього	Використовує середнє значення за останній період для прогнозування.	Прогнозування на основі короткострокових даних.	Прогнозування кількості відвідувань сайту на наступний день.

Продовження таблиці 2.1

Метод регресії	Метод побудови математичної моделі на основі даних для прогнозування.	Прогнозування на основі факторів, що впливають на трафік.	Прогнозування відвідуваності залежно від сезонних акцій.
Метод авторегресії	Використовує минулі значення для передбачення майбутніх значень.	Прогнозування на основі великої кількості минулих даних.	Прогнозування попиту за останні кілька місяців.
Метод авторегресії з ковзним середнім (ARIMA)	Моделює часового ряду з урахуванням як авторегресії, так і ковзного середнього.	Прогнозування при сильних коливаннях трафіку.	Прогнозування трафіку в умовах нестабільності.
Метод Байєсівських мереж	Використовує ймовірнісні моделі для прогнозування залежностей між змінними.	Прогнозування за умов значної невизначеності.	Прогнозування трафіку за допомогою ймовірнісних моделей.
Метод теорії масового обслуговування	Моделює трафік як процес обслуговування запитів або подій.	Прогнозування для систем з великим навантаженням.	Прогнозування навантаження на сервери.

Таблиця 2.2 наводить порівняльну характеристику переваг та недоліків перших, традиційних методів прогнозування трафіку [11].

Таблиця 2.2

Порівняльна характеристика переваг та недоліків традиційних методів прогнозування трафіку

Метод прогнозування	Переваги	Недоліки
Метод середнього значення	Простота, швидкість розрахунку.	Не враховує трендів і сезонних коливань.
Метод екстраполяції	Можливість прогнозувати тренди.	Вимагає великої кількості даних для точності.
Метод експоненціального згладжування	Враховує сезонні коливання та тренди.	Може бути складним в налаштуванні та обчисленнях.
Метод рухомого середнього	Простота реалізації та швидкість обчислень.	Не враховує сезонність або довгострокові тренди.
Метод регресії	Може враховувати декілька змінних, що впливають на трафік.	Потрібна велика кількість даних для точності моделі.
Метод авторегресії	Здатний адаптуватися до змін в трендах.	Потребує складних налаштувань.
Метод авторегресії з ковзним середнім (ARIMA)	Добре працює з нестаціонарними даними.	Складність у реалізації та налаштуванні.
Метод Байєсівських мереж	Здатність враховувати складні взаємозв'язки між змінними.	Складність в побудові моделі.
Метод теорії масового обслуговування	Застосовність до систем з обмеженими ресурсами.	Не враховує багато зовнішніх факторів.

2.2 Методи машинного навчання в прогнозуванні трафіку

Машинне навчання є однією з основних складових сучасної науки про дані, яка активно застосовується в багатьох галузях, включаючи прогнозування трафіку. Це є надзвичайно важливою задачею, оскільки точне

прогнозування трафіку дозволяє ефективно планувати інфраструктурні ресурси, розподіляти навантаження в реальному часі та знижувати витрати, що пов’язані з перевантаженням систем чи недооцінкою трафіку. Відтак, використання методів машинного навчання для прогнозування трафіку стає дедалі актуальнішим, оскільки традиційні підходи часто не здатні справлятися з такими масштабами та складністю даних, як це може зробити автоматизована модель [12].

Машинне навчання надає змогу прогнозувати трафік на основі аналізу великих обсягів історичних даних і виявлення прихованих закономірностей, що є надзвичайно корисним для виявлення трендів та сезонних коливань, а також для швидкого реагування на зміни в умовах інтернет-активності. Однією з головних переваг машинного навчання є його здатність адаптуватися до змінних умов без необхідності вручну налаштовувати модель під кожен новий набір даних, що значно спрощує процес прогнозування.

Прогнозування трафіку за допомогою машинного навчання охоплює різноманітні техніки та підходи, кожен з яких має свої сильні та слабкі сторони залежно від конкретних умов використання. Одним із найбільш поширених методів є регресійні моделі, які дозволяють передбачати значення майбутнього трафіку на основі вхідних змінних. Регресія в машинному навчанні використовується для моделювання залежностей між незалежними змінними та прогнозованим значенням [13]. Наприклад, лінійна регресія може бути використана для того, щоб визначити, як змінюється трафік на основі зовнішніх факторів, таких як час доби, день тижня, пори року, спеціальні події чи зміни в контенті вебсайту. Однак лінійна регресія може бути недостатньо ефективною для прогнозування складних і нелінійних залежностей, що характерно для реальних даних, де взаємодії між змінними можуть бути значно складнішими.

Для більш складних ситуацій, де взаємозв’язки між змінними є нелінійними, доцільно використовувати методи, засновані на деревах рішень,

такі як випадкові ліси та бустінг. Випадкові ліси складаються з великої кількості дерев рішень, кожне з яких є окремою моделлю, що визначає трафік на основі певного набору ознак. Перевагою цього методу є його здатність працювати з великими обсягами даних, зокрема в умовах високої кількості вхідних змінних. Випадкові ліси можуть ефективно справлятися з задачами, де є значний рівень шуму або де не існує чіткої лінійної залежності між змінними. Бустінг, з іншого боку, є технікою, що створює ансамбль моделей, де кожна наступна модель намагається коригувати помилки попередньої. Це дозволяє досягати більш високої точності в прогнозуванні трафіку, оскільки кожен новий крок у процесі навчання фокусується на тих аспектах даних, які були помилково передбачені на попередніх етапах [14].

Іншими популярними методами машинного навчання, що використовуються в прогнозуванні трафіку, є нейронні мережі. Нейронні мережі, зокрема глибокі нейронні мережі, є потужним інструментом для розв’язування складних задач прогнозування, оскільки вони здатні виявляти глибокі і складні взаємозв’язки між даними. Глибокі нейронні мережі використовують багатошарову архітектуру, що дозволяє автоматично видобувати корисні ознаки з сирих даних, таких як лог файли або реальні показники трафіку. Це може бути особливо корисно для передбачення високочастотного трафіку, коли кожен запит є потенційно важливим для моделі. Проте варто зазначити, що тренування глибоких нейронних мереж потребує значних обчислювальних ресурсів і часу, що може бути обмеженням для деяких організацій.

У сучасних системах прогнозування трафіку також широко використовуються рекурентні нейронні мережі (RNN) та їхні вдосконалені варіанти, такі як LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit). Ці моделі особливо ефективні для обробки послідовних даних, таких як серії даних про трафік, що мають часову залежність. Рекурентні нейронні мережі здатні враховувати попередні значення в часовому ряду для

прогнозування майбутніх значень, що робить їх ідеальними для задач, де важлива історія та динаміка зміни трафіку. Наприклад, такі мережі можуть передбачити зростання чи зниження трафіку на вебсайті на основі попередніх даних, що дозволяє краще планувати навантаження на сервери чи інші інфраструктурні компоненти.

Застосування методів машинного навчання в прогнозуванні трафіку також включає класифікацію даних для виявлення аномалій, що можуть вказувати на надмірне навантаження або злочинні атаки, такі як DDoS-атаки. Для цього використовуються алгоритми, здатні ідентифікувати аномальні патерни в даних. Одним із таких підходів є методи класифікації на основі підтримки векторних машин (SVM) або кластери на основі k-середніх. Ці методи дозволяють класифікувати дані на нормальні та аномальні, що дає змогу оперативно реагувати на несподівані збої або атаки, забезпечуючи безперервну роботу системи [13, 14].

Іншою важливою областю є використання гібридних методів машинного навчання для прогнозування трафіку. Гібридні моделі комбінують переваги кількох різних алгоритмів, щоб досягти кращої точності прогнозів. Наприклад, комбінація глибоких нейронних мереж з методами машинного навчання на основі дерев рішень дозволяє підвищити якість прогнозування, враховуючи як нелінійні взаємозв’язки, так і важливість вхідних змінних. Такий підхід є особливо корисним, коли дані мають різноманітну природу та містять як структуровану, так і неструктуровану інформацію.

У процесі впровадження методів машинного навчання для прогнозування трафіку важливо також звертати увагу на такі фактори, як вибір відповідних ознак, а також розподіл даних для тренування і тестування моделей. Оскільки дані часто мають не тільки різноманітні змінні, але й велику кількість пропусків або аномалій, необхідно застосовувати методи очищення даних та попередньої обробки, щоб забезпечити максимальну ефективність моделей. Одним із важливих етапів є також оптимізація моделей, коли на

основі тренувальних даних визначаються найбільш ефективні гіперпараметри для кожного конкретного методу машинного навчання.

Таким чином, використання методів машинного навчання для прогнозування трафіку є потужним інструментом, який дозволяє зберігати ефективність та точність прогнозів навіть у складних і мінливих умовах. Враховуючи постійне зростання обсягів даних і розвиток технологій, машинне навчання забезпечує нові можливості для вдосконалення процесів планування і управління трафіком, що є критично важливим для багатьох індустрій, від веброзробки до телекомунікацій та кібербезпеки.

2.3 Огляд сучасних підходів до прогнозування в хмарних системах

Прогнозування в хмарних системах є важливою складовою частиною управління такими системами, оскільки правильне передбачення майбутніх подій і змін може істотно підвищити ефективність їх роботи, знизити витрати та підвищити рівень надійності. Хмарні обчислювальні технології, які за останнє десятиліття стали основою для багатьох підприємств та організацій, мають велику кількість компонентів і є високодинамічними системами. Тому прогнозування їхнього стану, навантаження, витрат і навіть безпеки стає ключовим аспектом для забезпечення належної роботи таких інфраструктур. Оскільки хмарні системи включають в себе різноманітні технології і моделі, існує безліч підходів до прогнозування, які можуть враховувати різні аспекти діяльності таких систем [13-15].

Першим аспектом, на який слід звернути увагу при прогнозуванні в хмарних системах, є необхідність в обробці великих обсягів даних. Хмарні обчислення створюють величезні потоки даних, які потребують ефективних алгоритмів для обробки і аналізу. Одним із сучасних підходів до прогнозування є використання методів машинного навчання, які дозволяють з використанням історичних даних створювати моделі, здатні передбачати

майбутні події або зміни. Моделі машинного навчання можуть бути навчанням під наглядом або без нагляду, залежно від доступності мічених даних, а також можуть включати в себе глибоке навчання для роботи з складними наборами даних, такими як логи сервісів, метадані або інформація про запити користувачів.

Методи машинного навчання в хмарних системах застосовуються для прогнозування безлічі різних аспектів, зокрема для визначення навантаження на сервери, аналізу трафіку в мережах, моніторингу стану баз даних, прогнозування вимог до ресурсів і навіть для прогнозування можливих атак. Наприклад, аналіз трафіку в хмарних системах може допомогти передбачити збільшення навантаження в певний період часу, що дозволяє завчасно збільшити ресурси і знизити ймовірність перевантаження. Існують також моделі, які дозволяють прогнозувати потреби в обчислювальних ресурсах в залежності від різних змін, таких як збільшення кількості користувачів або зміни в умовах роботи програмного забезпечення. Прогнозування витрат на ресурси, яке забезпечує моделі машинного навчання, може зменшити операційні витрати, дозволяючи організаціям ефективніше керувати своїми хмарними ресурсами та оптимізувати витрати.

Іншим важливим аспектом є прогнозування стану безпеки хмарних систем. Хмари надають широкий спектр сервісів для зберігання даних, обробки запитів, виконання додатків, і з огляду на їх відкриту та публічну природу, питання безпеки є надзвичайно важливими. Прогнозування можливих атак або вразливостей може допомогти запобігти загрозам до того, як вони можуть бути використані зловмисниками [15]. Сучасні методи безпеки, такі як виявлення аномалій або використання автоматизованих систем для визначення патернів атак, також ґрунтуються на використанні методів машинного навчання. Наприклад, алгоритми, що працюють на основі аналізу великих даних, здатні виявляти підозрілі патерни у вхідному трафіку або в запитах, які можуть вказувати на DDoS-атаки чи спроби

несанкціонованого доступу до системи. Аналіз аномалій у хмарних системах дозволяє прогнозувати небажані події та вчасно вживати заходів для запобігання можливим інцидентам безпеки.

Прогнозування в хмарних системах також включає в себе такі аспекти, як управління ресурсами та забезпечення якості обслуговування. Однією з ключових переваг хмарних систем є їх здатність до масштабування, тобто здатність автоматично коригувати кількість доступних ресурсів у відповідь на зміну навантаження. Це означає, що для забезпечення стабільної роботи системи необхідно передбачити, скільки ресурсів буде потрібно в кожний конкретний момент часу. Для цього використовуються прогностичні моделі, що визначають на основі попередніх даних, скільки ресурсів буде необхідно для обробки запитів у майбутньому. Наприклад, для вебсервісів або хмарних баз даних, де кількість запитів може змінюватися з часом, прогностичні моделі можуть передбачити необхідність у збільшенні обчислювальних потужностей або збільшенні обсягу пам’яті для обробки додаткових запитів.

Моделі прогнозування можуть бути побудовані з використанням традиційних статистичних методів або на основі більш складних алгоритмів, таких як згорткові нейронні мережі або рекурентні нейронні мережі, які використовуються для обробки часових рядів і можуть ефективно справлятися з даними, що мають часову залежність. Наприклад, для передбачення майбутніх навантажень на сервери або попиту на ресурси можуть використовуватися рекурентні нейронні мережі LSTM або GRU, які здатні зберігати інформацію про попередні стани системи та враховувати її при прийнятті рішень про майбутнє. Ще однією важливою частиною прогнозування в хмарних системах є інтеграція з іншими технологіями, такими як інтернет речей (IoT). Хмари все більше інтегруються з розумними пристроями, що дозволяє здійснювати аналіз та прогнозування не тільки на основі даних, що надходять із серверів або додатків, але й з фізичних пристроїв, таких як сенсори, пристрої моніторингу та датчики. Інтернет речей

дозволяє зібрати величезну кількість даних з різних джерел, що потім можуть бути оброблені та використані для прогнозування майбутніх змін у хмарних системах. Наприклад, в системах моніторингу енергоспоживання в хмарах дані, отримані від сенсорів, можуть використовуватися для прогнозування потреб у енергетичних ресурсах або для визначення оптимальних режимів роботи серверів. Технології хмарних обчислень також дозволяють використовувати багато різних типів моделей прогнозування в реальному часі. Це дозволяє організаціям на хмарних платформах швидко адаптуватися до змін в навантаженні або запитах, що приходять до системи, забезпечуючи необхідну продуктивність без додаткових витрат. Така здатність адаптуватися в реальному часі є важливою перевагою хмарних систем, оскільки дозволяє забезпечити стабільну роботу навіть при великих коливаннях навантаження або при виникненні неочікуваних ситуацій [12, 13].

Таким чином, прогнозування в хмарних системах є важливим аспектом для забезпечення їх ефективної роботи, надійності та безпеки. Сучасні підходи до прогнозування включають використання методів машинного навчання для прогнозування навантаження, витрат на ресурси, безпеки, якості обслуговування та інтеграції з іншими технологіями. Вони дозволяють здійснювати точні прогнози і забезпечувати стабільність та ефективність хмарних інфраструктур. З розвитком нових технологій і методів обробки даних прогнозування в хмарних системах буде тільки вдосконалюватися, відкриваючи нові можливості для зниження витрат і підвищення ефективності в таких системах.

2.4 Огляд нейронних мереж і їх архітектур

Нейронні мережі є однією з основних концепцій у галузі штучного інтелекту та машинного навчання. Вони являють собою клас моделей, натхнених біологічними нейронними мережами, які обробляють інформацію

в мозку. Нейронні мережі здатні виконувати широкий спектр завдань, таких як класифікація, регресія, виявлення аномалій, генерація зображень, машинний переклад та багато інших. Вони є важливою частиною сучасних підходів до аналізу даних, зокрема в галузях комп'ютерного зору, обробки природної мови та робототехніки [7-10]. Протягом останніх десятиліть розвиток нейронних мереж значно змінився, завдяки удосконаленню обчислювальних потужностей, новим алгоритмам навчання та здобуткам в науці про дані. Огляд нейронних мереж і їх архітектур, з урахуванням сучасних тенденцій і досягнень, дозволяє зрозуміти, чому вони стали настільки популярними і як саме вони функціонують.

Нейронні мережі складаються з великої кількості з'єднаних між собою нейронів або вузлів, які організовані в шари. Основна ідея нейронної мережі полягає в імітації роботи біологічного мозку, де нейрони обробляють і передають інформацію за допомогою електричних імпульсів. Кожен нейрон в мережі приймає сигнал від інших нейронів або зовнішніх джерел, обробляє цей сигнал через математичну функцію активації і передає результат наступним нейронам. У результаті такого взаємодії нейронів, нейронна мережа здатна вирішувати складні задачі, навіть якщо вона складається з простих елементів. Нейронні мережі можна класифікувати за різними ознаками, включаючи типи архітектур, спосіб навчання, застосування та інші параметри. Однією з найбільш загальних класифікацій є поділ на одношарові та багатошарові мережі. Одношарові мережі складаються лише з одного шару нейронів, які безпосередньо з'єднані з вхідними та вихідними даними. Ці мережі є простими і можуть застосовуватися для вирішення задач лінійної класифікації або регресії. Однак для більш складних задач, таких як виявлення складних патернів в даних, потрібні багатошарові мережі, де існує кілька шарів нейронів між вхідними та вихідними даними. Багатошарові нейронні мережі здатні виявляти нелінійні залежності між змінними, що робить їх значно більш потужними та універсальними.

Основою багатошарових нейронних мереж є концепція так званих «глибоких нейронних мереж» або «deep learning». Глибоке навчання передбачає використання мереж з великою кількістю шарів, що дозволяє моделі навчатися більш складним представленням даних на різних рівнях абстракції. Це дозволяє глибоким мережам, таким як згорткові нейронні мережі (CNN) або рекурентні нейронні мережі (RNN), вирішувати складні завдання, пов’язані з обробкою зображень, текстів або інших типів даних [11].

Однією з основних архітектур нейронних мереж є перцептрон. Перцептрон є найпростішою формою нейронної мережі і складається з одного шару нейронів, що використовуються для класифікації двох класів. Хоча перцептрон є обмеженим і не здатний вирішувати складні задачі, він є основою для більш складних багатошарових архітектур. Розширенням перцептрона є багатошаровий перцептрон, який складається з декількох шарів нейронів, де кожен шар відповідає за обробку все більш складних ознак вхідних даних. Така мережа здатна вирішувати нелінійні задачі класифікації завдяки використанню нелінійних функцій активації.

Важливою архітектурою є згорткова нейронна мережа (CNN), яка використовується для обробки даних, що мають топологічну структуру, зокрема зображення. CNN складаються з кількох типів шарів: згорткових, підвибіркових і повнозв’язних. Згорткові шари відповідають за виділення ознак з вхідних даних, таких як краї, текстури чи форми, використовуючи операцію згортки. Підвибіркові шари зменшують розмірність даних і допомагають мережі зберігати важливу інформацію, знижуючи обчислювальні витрати. Повнозв’язні шари відповідають за фінальну класифікацію або регресію, на основі ознак, що були виділені на попередніх етапах. CNN стали однією з найбільш популярних архітектур для завдань, пов’язаних із комп’ютерним зором, таких як класифікація зображень, детекція об’єктів, сегментація зображень та інші. Вони також використовуються в ряді інших застосувань, таких як обробка відео, медичні зображення та навіть для

задач, пов’язаних із обробкою природної мови, таких як виявлення емоцій в текстах або класифікація новин [12].

Ще однією важливою архітектурою є рекурентні нейронні мережі (RNN), які здатні обробляти послідовні дані, що мають часову залежність. Відмінною рисою RNN є те, що вони мають зв’язки між своїми нейронами на різних етапах, що дозволяє їм зберігати інформацію про попередні стани мережі. Ці мережі широко використовуються в задачах, де важливі не лише окремі спостереження, але й їхня послідовність. Наприклад, вони можуть бути застосовані для обробки тексту, де кожне слово залежить від попереднього, або для передбачення майбутніх значень в часових рядах. Однак стандартні RNN мають обмеження через проблему затухаючого градієнта, що ускладнює їх навчання на довгих послідовностях. Для подолання цієї проблеми були розроблені покращені варіанти RNN, такі як LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit), які мають спеціальні механізми для збереження інформації на тривалих проміжках часу. Іншою важливою архітектурою є автоенкодери, які використовуються для навчання компактних представлень даних. Автоенкодери складаються з двох основних частин: кодувальника, який зменшує розмірність вхідних даних, та декодувальника, який відновлює дані до початкового вигляду. Вони широко використовуються для задач векторизації даних, таких як стиснення зображень, зниження розмірності даних або виявлення аномалій. З розвитком технологій і методів обробки даних нейронні мережі продовжують еволюціонувати. Архітектури стають дедалі складнішими і потужнішими, здатними вирішувати завдання, що раніше здавалися неможливими для машинного навчання. Наприклад, генеративні змагальні мережі (GAN), які використовуються для генерації нових даних на основі існуючих, отримали величезне поширення в таких сферах, як генерація зображень, відео та музики. GAN складаються з двох нейронних мереж: генератора, який створює нові дані, та дискримінатора, який оцінює, наскільки реалістичними є ці дані.

Таким чином, нейронні мережі та їх архітектури є потужними інструментами для вирішення широкого спектра задач. Їх здатність автоматично вчитися з даних та виявляти складні залежності робить їх незамінними в багатьох сферах, від медицини до фінансів, від обробки зображень до обробки природної мови. З розвитком обчислювальних потужностей та нових методів навчання нейронні мережі продовжують розширювати свої можливості, відкриваючи нові горизонти для досліджень та інновацій.

2.5 Використання глибинного навчання для прогнозування

Глибинне навчання є потужним напрямком у сфері машинного навчання, яке здійснило справжню революцію в багатьох галузях, таких як комп’ютерний зір, обробка природної мови, медицина та фінансові технології. Суть глибинного навчання полягає в використанні багатошарових нейронних мереж, які здатні автоматично навчатися представленням даних на різних рівнях абстракції [13]. Глибинне навчання має значний вплив на різні методи прогнозування, оскільки дозволяє створювати моделі, які можуть передбачати майбутні події, оцінювати ризики, передбачати зміни в поведінці систем або навіть генерувати нові дані на основі існуючих. У порівнянні з традиційними методами прогнозування, які часто вимагають спеціальної обробки даних і експертних знань для формулювання математичних моделей, глибинне навчання здатне автоматично витягувати важливі характеристики з великих обсягів неструктурованих даних, таких як зображення, текст або часові ряди. Це дозволяє зменшити вплив людського фактора та підвищити точність прогнозів у складних і нестандартних задачах.

Одним з основних напрямків використання глибинного навчання є прогнозування на основі часових рядів. Часові ряди – це дані, що містять інформацію про події, які відбуваються в часі, такі як ціни на фінансові активи,

температура повітря, обсяги продажів, трафік на вебсайті тощо. Прогнозування на основі часових рядів є важливим для розв’язання багатьох задач, таких як фінансове прогнозування, прогнозування попиту на товари і послуги, передбачення природних катастроф або аналіз поведінки споживачів. Традиційні методи, такі як авторегресивні моделі (AR), моделі ковзаючого середнього (MA) або гібридні моделі ARIMA, здатні давати непогані результати в умовах лінійних залежностей між змінними. Однак для складних, нелінійних та багатофакторних систем, де присутні приховані зв’язки між величинами, традиційні методи можуть бути менш ефективними.

Глибинні нейронні мережі, зокрема рекурентні нейронні мережі (RNN), показали надзвичайно високі результати при прогнозуванні часових рядів. RNN володіють властивістю зберігати інформацію про попередні стани в процесі обробки послідовностей, що дозволяє їм ефективно передбачати майбутні значення на основі попередніх спостережень. Однак класичні RNN мають обмеження у вигляді проблеми затухаючого градієнта, що ускладнює навчання на довгих послідовностях [14]. Для подолання цієї проблеми були розроблені вдосконалені архітектури, такі як Long Short-Term Memory (LSTM) та Gated Recurrent Units (GRU), які використовують спеціальні механізми для збереження та оновлення інформації про стан мережі, що дозволяє їм ефективно працювати з довгими послідовностями даних і прогнозувати довгострокові залежності. LSTM і GRU застосовуються для прогнозування в таких областях, як фінансові ринки, передбачення попиту на ресурси, прогнозування кліматичних змін, а також для обробки складних природних мовних даних.

Крім рекурентних мереж, інші архітектури глибинного навчання також застосовуються для прогнозування в різних сферах. Наприклад, згорткові нейронні мережі (CNN), хоча традиційно використовуються для обробки зображень, також виявили свою ефективність у задачах прогнозування. У випадку часових рядів, CNN використовуються для автоматичного виділення

локальних ознак або патернів з даних, що дає можливість прогнозувати зміни на основі виявлених залежностей. Згорткові мережі зазвичай працюють в парі з іншими типами мереж, такими як LSTM або GRU, що дозволяє моделі зберігати як просторові, так і часові залежності. Іншою архітектурою, яка активно використовується в задачах прогнозування, є трансформери. Спочатку трансформери були розроблені для задач обробки природної мови, зокрема для машинного перекладу, однак їх гнучкість і здатність до паралельної обробки даних швидко зробили їх популярними для прогнозування в інших галузях. Трансформери використовують механізм уваги (attention), що дозволяє моделі зосереджуватись на найбільш релевантних частинах вхідних даних для прийняття рішень. Завдяки такому механізму, трансформери можуть ефективно працювати з великими обсягами даних, враховуючи взаємодії між елементами на різних етапах послідовності. Це робить трансформери особливо корисними для задач, де важливі як глобальні, так і локальні залежності в даних. Вони застосовуються для прогнозування в фінансових ринках, аналізі тексту, створенні рекомендаційних систем та в прогнозуванні попиту на продукти. Особливість глибинного навчання в контексті прогнозування полягає в його здатності автоматично знаходити важливі патерни в даних без необхідності ручного вибору ознак. Це значно зменшує потребу в експертних знаннях та попередній обробці даних, що характерно для традиційних підходів до прогнозування. В той же час, глибинне навчання вимагає значних обчислювальних потужностей для навчання моделей, а також великого обсягу даних для досягнення високої точності. Недостатня кількість даних або неякісні дані можуть призвести до переобучення моделі або її низької здатності до генералізації. Для вирішення цих проблем були розроблені різноманітні методи регуляризації та техніки зменшення розміру моделі, що дозволяють ефективно використовувати глибинне навчання навіть при обмежених ресурсах. Окрім того, техніки як transfer learning (перенос навчання) дозволяють використовувати попередньо

навчальні моделі, адаптуючи їх до нових задач за допомогою меншої кількості даних [15]. Одним з важливих аспектів застосування глибинного навчання для прогнозування є проблема інтерпретованості моделей. Оскільки глибинні моделі складаються з великої кількості параметрів і складних шарів, вони часто є «чорними ящиками», тобто важко зрозуміти, як саме модель приймає рішення. Однак в останні роки були розроблені методи для покращення інтерпретованості, такі як техніки візуалізації нейронних мереж, методи важливості ознак та інші підходи, які дозволяють зрозуміти, які саме фактори впливають на результат прогнозування. Це є особливо важливим у таких критичних сферах, як медицина або фінанси, де необхідно пояснити, чому модель зробила певне передбачення, щоб забезпечити довіру користувачів і відповідність регуляторним вимогам. Слід також зазначити, що глибинне навчання має широкі можливості для інтеграції з іншими сучасними технологіями. Наприклад, в поєднанні з великими даними (big data) та хмарними обчисленнями, моделі глибинного навчання можуть ефективно обробляти величезні масиви інформації та робити прогнози на основі різноманітних джерел даних. Така інтеграція дозволяє створювати прогностичні моделі, які працюють у реальному часі, аналізуючи потоки даних з різних датчиків, соціальних мереж або інших джерел.

Таким чином, глибинне навчання стає невід’ємною частиною сучасних підходів до прогнозування. Завдяки своїй здатності обробляти великі обсяги неструктурованих даних і знаходити складні взаємозв’язки, воно відкриває нові горизонти для створення точних і ефективних прогнозних моделей у різних сферах. Однак, для досягнення максимальних результатів важливо враховувати потребу в великих даних, обчислювальних ресурсах та методах підвищення інтерпретованості моделей, що дозволяє забезпечити їх ефективне та етичне застосування.

2.6 Техніки навчання нейронних мереж

Техніки навчання нейронних мереж [12-14] представлено в таблиці 2.3.

Таблиця 2.3

Техніки навчання нейронних мереж

Техніка навчання	Опис	Застосування	Переваги
Підкріплене навчання	Процес навчання, при якому агент навчається виконувати завдання, отримуючи нагороду або покарання залежно від своїх дій.	Ігри, робототехніка, автономні транспортні засоби	Може ефективно навчати агентів без попередньої інформації, адаптується до змінних умов
Навчання з учителем	Моделі навчаються на основі мічених даних, де кожен вхід має відповідний вихід.	Класифікація, регресія, розпізнавання образів	Висока точність при правильній підготовці даних, простота у впровадженні
Навчання без учителя	Моделі намагаються знайти структуру в неорганізованих даних без явних міток.	Кластеризація, зниження розмірності, аномалії	Добре підходить для нових, неструктуро-ваних даних, не потребує міток
Напів-підкріплене навчання	Поєднує підкріплене та навчання з учителем, де мітки частково присутні, а решта даних – без міток.	Медична діагностика, розпізнавання обличчя	Ефективне у випадках з обмеженою кількістю мічених даних

Продовження таблиці 2.3

Генеративні моделі	Моделі, що генерують нові дані на основі вивчених розподілів існуючих даних.	Генерація зображень, синтез тексту, штучні зображення	Можливість створювати нові дані, які схожі на реальні
Глибоке навчання	Тип навчання, що використовує багатошарові нейронні мережі для автоматичного виділення ознак.	Обробка зображень, тексту, відео	Здатність до виявлення складних залежностей, висока точність
Супервізоване навчання	Моделі навчаються на чітко позначених даних для прогнозування або класифікації.	Прогнозування, виявлення шаблонів	Точність прогнозів, інтуїтивна побудова моделей
Безконтрольне навчання	Навчання без наявності міток даних, яке дозволяє системі самостійно знаходити закономірності.	Аналіз великих даних, пошук аномалій	Може виявляти нові патерни, ефективне для складних задач
Перенос навчання	Перенос знань з однієї моделі на іншу для скорочення часу навчання на нових даних.	Обробка нових задач із обмеженими даними	Зменшує потребу в великих наборах даних для нових задач
Невиконуване навчання	Моделі працюють з даними без конкретних цілей чи вихідних значень, покладаючись на інші ознаки для прогнозування.	Автономні системи	Здатність до самостійного навчання без явних цілей

2.7 Оцінка ефективності нейронних мереж в задачах прогнозування

Оцінка ефективності нейронних мереж в задачах прогнозування є важливим аспектом, що визначає їх здатність до адекватного відображення реальних залежностей у даних та забезпечення точних передбачень. Прогнозування, яке полягає у передбаченні майбутніх значень або трендів на основі історичних даних, є одним з найбільш розповсюджених застосувань машинного навчання, і нейронні мережі в даному контексті демонструють виняткову ефективність, завдяки здатності обробляти складні, нелінійні зв'язки між величинами, що важко описуються традиційними статистичними методами.

Нейронні мережі, зокрема їх глибокі варіанти, забезпечують високу точність при вирішенні задач прогнозування завдяки своїй здатності автоматично навчатися на великому обсязі даних та виявляти приховані закономірності, які часто неможливо виявити традиційними підходами. Оцінка ефективності нейронних мереж у задачах прогнозування передбачає використання кількох методів, які допомагають зрозуміти, наскільки добре модель працює на даній задачі, і наскільки вона може бути застосована в реальних умовах [10]. Важливими аспектами оцінки ефективності є точність прогнозів, здатність до генералізації, час обчислень, стійкість до шуму в даних і здатність до роботи з неструктурованими даними.

Одним із ключових критеріїв оцінки ефективності нейронних мереж є точність прогнозування. Точність вимірюється за допомогою різних метрик, таких як середньоквадратична помилка (MSE), середня абсолютна помилка (MAE), кореляція між прогнозованими і реальними значеннями, коефіцієнт детермінації (R^2) тощо. Серед них середньоквадратична помилка є найбільш популярною, оскільки вона надає чітке уявлення про різницю між прогнозованими значеннями та фактичними результатами. Однак важливо враховувати, що кожен з цих показників має свої переваги та недоліки.

Наприклад, середньоквадратична помилка є дуже чутливою до великих відхилень, що може призвести до переоцінки незначних помилок у прогнозах. Водночас середня абсолютна помилка дає більш стійку метрику до великих відхилень, що може бути корисно в задачах, де важливо зберігати рівновагу між точністю та стійкістю до аномальних даних.

Для оцінки ефективності нейронних мереж також важливе значення має здатність моделі до генералізації, тобто її здатність працювати з новими, невідомими даними. Нейронні мережі, особливо при глибокому навчанні, мають схильність до переобучення, що означає, що модель надто добре адаптується до навчального набору даних, але втрачає здатність правильно прогнозувати на нових, невидимих для неї даних. Тому для оцінки генералізації застосовують техніки, такі як крос-валідація, де дані діляться на кілька частин, і модель тестується на різних підмножинах даних. Крос-валідація допомагає знизити ризик переобучення і оцінити, як добре модель справляється з новими, не баченими під час навчання даними [9].

Крім того, ефективність нейронних мереж оцінюється через здатність моделі працювати з неструктурованими даними. Зокрема, в задачах прогнозування можуть використовуватися дані, які мають складну структуру, такі як зображення, текст або часові ряди. Нейронні мережі є ефективними інструментами для таких задач, оскільки вони можуть автоматично витягувати корисні ознаки з неструктурованих даних, що робить їх потужним інструментом у різних сферах. Наприклад, у задачах прогнозування на основі зображень нейронні мережі здатні витягувати важливі ознаки, такі як контури, текстури або форми об’єктів, що дозволяє створювати моделі, які можуть передбачати зміни в зображеннях з високою точністю. У задачах прогнозування часових рядів нейронні мережі можуть автоматично навчатися на даних про минулі значення та виявляти складні залежності, які є важкими для традиційних методів аналізу.

Важливим аспектом оцінки ефективності нейронних мереж є час навчання та обчислювальні ресурси, необхідні для створення моделей. Глибоке навчання, яке часто використовується в нейронних мережах, вимагає значних обчислювальних потужностей, особливо для великих наборів даних або складних моделей. Тому ефективність моделей також оцінюється з точки зору швидкості навчання, зокрема порівняно з іншими методами машинного навчання, такими як підтримка векторних машин або дерева рішень. Зважаючи на високі обчислювальні витрати, важливо оцінити, чи є вигідним використання нейронних мереж у порівнянні з іншими підходами, якщо точність не покращується суттєво. Ще одним важливим аспектом є стійкість моделі до шуму в даних. В реальних умовах дані, на основі яких здійснюється прогнозування, часто містять помилки або неточності, які можуть впливати на якість прогнозів. Нейронні мережі мають здатність працювати з шумними даними, однак для цього необхідно використовувати методи регуляризації та оптимізації, які допомагають знизити ризик переобучення і підвищити стійкість до шуму. Використання таких методів, як Dropout або Batch Normalization, дозволяє зробити модель більш стабільною і здатною до коректної роботи в умовах шумних або неповних даних [7, 9, 10].

Нейронні мережі також демонструють високу гнучкість у вирішенні різноманітних задач прогнозування. Завдяки різноманітним архітектурам, таким як згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) або трансформери, вони можуть бути адаптовані до специфіки конкретної задачі. Наприклад, для прогнозування часових рядів найефективнішими є рекурентні нейронні мережі, оскільки вони здатні враховувати часову залежність в даних. Для обробки зображень або відео часто використовуються згорткові мережі, які добре справляються з виявленням локальних ознак. У свою чергу, трансформери показали себе дуже ефективними в задачах прогнозування, де важливі глобальні залежності та складні взаємодії між елементами даних, що зустрічаються в тексті або часових рядах.

Загалом, оцінка ефективності нейронних мереж в задачах прогнозування повинна охоплювати кілька аспектів: точність моделі, її здатність до генералізації, обчислювальні витрати, стійкість до шуму, а також здатність працювати з різноманітними типами даних. Крім того, для ефективного використання нейронних мереж необхідно враховувати специфіку конкретної задачі, особливості даних та ресурси, що є у розпорядженні. Нейронні мережі показують надзвичайно високі результати в задачах прогнозування, особливо коли мають справу з великими обсягами складних даних, але для досягнення оптимальних результатів важливо використовувати правильні методи навчання, регуляризації та оцінки ефективності.

2.8 Висновки до другого розділу

Розділ демонструє поступову еволюцію підходів до прогнозування трафіку, починаючи від традиційних статистичних методів і поступово переходячи до більш складних і потужних технік машинного навчання, нейронних мереж і глибинного навчання. Це вказує на те, що з розвитком обчислювальних можливостей та збільшенням обсягу даних традиційні методи не завжди можуть забезпечити необхідну точність і ефективність, тому використовуються більш інноваційні підходи.

Методи машинного навчання починають заміщати традиційні методи завдяки своїй здатності обробляти великі обсяги даних, знаходити складні нелінійні зв’язки та забезпечувати кращі результати прогнозування. Це підкреслює важливість застосування сучасних технологій для аналізу трафіку в умовах хмарних систем, де трафік є дуже динамічним і змінюється під впливом численних факторів. Огляд сучасних підходів до прогнозування в хмарних системах підкреслює необхідність адаптації методів прогнозування до специфіки хмарних середовищ. Це включає врахування різноманітних факторів, таких як обробка великих даних, оптимізація ресурсів та зниження

навантаження на сервери в реальному часі. Хмарні системи є важливим контекстом для розвитку прогнозування трафіку, оскільки вони можуть забезпечити масштабованість та гнучкість для інтеграції передових методів.

Огляд нейронних мереж та їх архітектур вказує на те, що нейронні мережі, зокрема їх більш складні архітектури, є потужними інструментами для прогнозування трафіку. Вони здатні виявляти складні залежності між параметрами та адаптуватися до змін в обсягах даних. Це є особливо важливим для хмарних систем, де динамічність і складність даних постійно змінюється;

Підрозділ 2.5 акцентує увагу на глибинному навчанні як ключовому підході для прогнозування, що дозволяє не тільки покращити точність моделей, але й автоматизувати процеси навчання на великих і складних даних. Глибинне навчання дозволяє нейронним мережам глибше розуміти структуру даних і краще прогнозувати навіть у найскладніших ситуаціях.

Підрозділ 2.6 висвітлює важливість технік навчання нейронних мереж, таких як регуляризація, оптимізація та використання різних алгоритмів навчання. Ці техніки дозволяють підвищити ефективність нейронних мереж і забезпечити їхню стійкість до переобучення, що є важливим для створення високоточних прогнозних моделей для хмарних систем.

Останній підрозділ підкреслює необхідність ретельної оцінки ефективності нейронних мереж у задачах прогнозування. Важливими аспектами є точність прогнозів, здатність до генералізації, стійкість до шуму в даних, а також швидкість обчислень. Це дозволяє зрозуміти, чи є нейронні мережі оптимальними для конкретних умов і задач прогнозування трафіку в хмарних середовищах.

Загалом, проведений аналіз другого розділу показує, як різні підходи до прогнозування трафіку розвиваються від традиційних методів до більш складних і гнучких технологій машинного навчання та глибинного навчання, які стають незамінними інструментами в умовах динамічних та високонавантажених хмарних обчислювальних систем.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Постановка задачі

Загальною метою проєкту є розроблення програмного інструмента для прогнозування трафіку хмарних сервісів на основі синтетичних даних. Програма повинна використовувати різні алгоритми машинного навчання для порівняння їхньої ефективності, а також мати функції для візуалізації, оцінки моделей, виявлення аномалій та інтерактивного прогнозування.

Основні завдання проєкту полягають у:

1) синтетично згенерувати часові дані для імітації роботи хмарних сервісів, з урахуванням:

– циклічних компонентів (наприклад, використання CPU та пам’яті);

– трендових змін (вхідний/вихідний трафік мережі);

– випадкових коливань навантаження (виклики API, кількість користувачів);

Дані повинні включати часові мітки та ключові параметри, що впливають на обсяг трафіку.

2) передобробка даних:

– розділити дані на навчальну та тестову вибірки;

– здійснити масштабування числових ознак за допомогою standardscaler або minmaxscaler;

– перетворити категоріальні ознаки у числовий формат за допомогою техніки one-hot encoding;

3) реалізувати навчання та порівняння різних моделей для регресійного аналізу:

– random forest regressor (з використанням grid search для підбору гіперпараметрів),

– gradient boosting regressor,

– support vector regressor (svr),

– linear regression;

4) використовувати метрики оцінки:

– середньоквадратична помилка (mse);

– середня абсолютна помилка (mae);

– коефіцієнт детермінації (r^2);

– середня абсолютна відсоткова помилка (mape);

5) створити графічні візуалізації для наочного представлення:

– співвідношення реальних та прогнозованих значень трафіку;

– розподілу похибок;

– важливості ознак для моделей на основі random forest;

– результатів крос-валідації моделей;

6) реалізувати функцію для прогнозування обсягів трафіку на заданий період (наприклад, на 24-48 годин) з використанням найкращої моделі. Забезпечити можливість автоматичної генерації вхідних даних для прогнозування;

7) додати функціонал для виявлення аномальних значень у трафіку на основі аналізу залишків (різниць між реальними та прогнозованими значеннями);

8) реалізувати інтерфейс для користувача з можливістю:

– виконання прогнозів на майбутні періоди;

– налаштування параметрів прогнозування (наприклад, вибір моделі чи кількості періодів);

– порівняння прогнозів різних моделей на одному графіку;

9) додати функції для збереження навченої моделі у файл (joblib) та її завантаження для подальшого використання.

У результаті розробки має бути створений програмний інструмент, що дозволяє:

1) генерувати (якщо відсутні) складні синтетичні дані для задач прогнозування;

2) автоматизувати навчання, порівняння та оцінку моделей;

3) прогнозувати обсяг трафіку з використанням найкращих моделей;

4) виявляти аномалії у трафіку;

5) візуалізувати результати та забезпечити можливість інтерактивного прогнозування для користувачів.

Мова програмування: Python.

Використані бібліотеки: NumPy, Pandas, Matplotlib, Seaborn, scikit-learn, joblib, logging.

3.2 Опис використаних бібліотек

Програма, представлена у вихідному коді, є комплексною системою прогнозування трафіку хмарних сервісів з використанням методів машинного навчання та сучасних аналітичних технологій. Основою цієї системи є Python – універсальна мова програмування, яка завдяки своїй багатій екосистемі інструментів, бібліотек і фреймворків є ключовим вибором для розробки рішень у галузі машинного навчання, аналітики даних та наукових обчислень. У цьому випадку Python використовується для побудови та інтеграції модулів, які реалізують функціонал генерації даних, попередньої обробки, навчання моделей машинного навчання, оцінювання їх ефективності, візуалізації результатів, а також прогнозування та виявлення аномалій.

Бібліотека NumPy є ключовим компонентом системи для роботи з числовими даними. NumPy забезпечує ефективні інструменти для створення та обробки багатовимірних масивів, виконання математичних операцій, а також генерації випадкових чисел. У коді вона використовується для створення синтетичних даних, включаючи генерацію циклічних і трендових числових послідовностей. Зокрема, функція np.sin дозволяє моделювати

синусоїдальні патерни, що імітують циклічність використання ресурсів, таких як CPU та пам’ять. Також np.random забезпечує створення випадкових даних з різними розподілами, що є основою для синтетичних показників, зокрема випадкового шуму, який додається до даних для наближення до реальних сценаріїв.

Для представлення даних у структурованому вигляді використовується Pandas. Ця бібліотека є основою для обробки та аналізу структурованих даних у форматі таблиць, що забезпечує зручну інтеграцію з іншими компонентами системи. Вихідні дані представлені у форматі DataFrame, що дозволяє виконувати операції з фільтрації, обчислення нових ознак, злиття стовпців та обробки часових рядів. Також Pandas використовується для трансформації категоріальних ознак у числові значення шляхом One-Hot Encoding через функцію pd.get_dummies, що є критично важливим етапом у підготовці даних для моделей машинного навчання.

Matplotlib і Seaborn є основними інструментами для візуалізації даних. Matplotlib надає можливості для створення базових графіків, таких як лінійні діаграми, гістограми та діаграми розсіювання. У системі Matplotlib використовується для створення візуалізацій розподілу помилок, діаграм важливості ознак та прогнозованих значень трафіку. Seaborn, як розширення Matplotlib, забезпечує створення естетично привабливих і інформативних графіків завдяки високорівневим API для візуалізації розподілів даних та трендів.

Для машинного навчання у програмі застосовуються інструменти з Scikit-learn – провідної бібліотеки Python для вирішення завдань машинного навчання. Scikit-learn забезпечує широкий спектр алгоритмів для класифікації, регресії, кластеризації та оцінки моделей. Основними моделями у коді є RandomForestRegressor, GradientBoostingRegressor, LinearRegression і SVR, які використовуються для прогнозування трафіку на основі ознак, згенерованих у процесі синтетичної генерації даних. Модель RandomForestRegressor є

ансамблевим методом, що поєднує кілька дерев рішень для зменшення ризику переобчислення та підвищення точності прогнозування. Gradient Boosting, на відміну від Random Forest, виконує послідовне навчання моделей, що коригують помилки попередніх ітерацій, що забезпечує високу точність за рахунок ефективного навчання на малих вибірках. Linear Regression є базовою моделлю, яка дозволяє оцінити лінійні залежності між ознаками, тоді як SVR використовує метод опорних векторів для моделювання нелінійних патернів у даних завдяки ядру RBF.

Для попередньої обробки даних застосовується StandardScaler та MinMaxScaler з Scikit-learn. StandardScaler забезпечує стандартизацію даних шляхом перетворення їх до нульового середнього значення та одиничного стандартного відхилення, що є важливим для алгоритмів, чутливих до масштабування, таких як SVM. MinMaxScaler перетворює дані у заданий діапазон, що є корисним для алгоритмів, які працюють з обмеженими значеннями.

Процес навчання моделей включає GridSearchCV, який забезпечує пошук оптимальних гіперпараметрів моделі за допомогою крос-валідації. У прикладі для RandomForestRegressor підбираються значення параметрів, таких як n_estimators і max_depth, що дозволяє знайти найкращу конфігурацію моделі для конкретного набору даних. Крос-валідація дозволяє мінімізувати проблему перенавчання, розділяючи навчальні дані на кілька підвибірок для оцінки якості моделі на різних сегментах.

Для оцінки продуктивності моделей використовуються метрики з Scikit-learn, серед яких mean_squared_error (MSE), mean_absolute_error (MAE), r2_score і mean_absolute_percentage_error (MAPE). MSE та MAE є класичними метриками помилок регресії, де MSE акцентує увагу на великих відхиленнях завдяки квадратичній формі. R2 Score є показником пояснювальної здатності моделі, що відображає відношення дисперсії прогнозованих значень до

фактичних. MAPE використовується для обчислення середнього абсолютного відсоткового відхилення.

Для збереження та відновлення моделей використовується Joblib – бібліотека для серіалізації Python-об’єктів. Збереження моделі разом зі стандартним масштабувальником забезпечує можливість повторного використання без необхідності повторного навчання на нових даних. Це особливо важливо для прогнозування в режимі реального часу або інтеграції з іншими системами.

Особливу увагу приділено генерації синтетичних даних, які імітують реальні сценарії використання хмарних ресурсів. За допомогою циклічних функцій і випадкових трендів моделюються патерни використання CPU, пам’яті, мережевих ресурсів, а також обсягу API-запитів. Синтетичні дані збагачуються додатковими ознаками, такими як тип хмарного сервісу (IaaS, PaaS, SaaS, FaaS) і регіон. Такий підхід дозволяє відтворити складні залежності між ознаками і на виході отримати набір даних, який є ідеальним для тестування моделей машинного навчання.

Для аналізу аномалій у трафіку використовується залишковий метод, який обчислює різницю між фактичними та прогнозованими значеннями трафіку. Визначення аномалій здійснюється через обчислення порогу на основі середнього значення залишків і їх стандартного відхилення. Візуалізація аномалій на часовій шкалі дозволяє виявити моменти, коли трафік виходить за межі звичайного діапазону, що є важливим для моніторингу продуктивності хмарних систем.

Крім того, програмний код включає функціонал для інтерактивного режиму прогнозування, де користувач може налаштувати параметри прогнозу, вибрати модель і переглянути результати у зручному форматі. Це підвищує гнучкість системи та її адаптацію до різних сценаріїв використання.

Таким чином, система використовує сучасний стек технологій для забезпечення ефективного прогнозування трафіку, навчання моделей

машинного навчання та виявлення аномалій. Комплексне використання Python, NumPy, Pandas, Scikit-learn, Matplotlib і Seaborn забезпечує гнучкість, ефективність і масштабованість системи, що робить її потужним інструментом для аналізу і прогнозування даних у хмарних середовищах.

3.3 Архітектура системи

Архітектура розробленого програмного коду є складною, модульною та ефективно структурованою системою, яка побудована для розв’язання завдання аналізу, прогнозування та виявлення аномалій у хмарному трафіку. Її побудова заснована на використанні принципів об’єктно-орієнтованого програмування, процедурного підходу, а також функціональних компонентів для забезпечення максимальної гнучкості, продуктивності та надійності системи. В основу архітектури закладено чіткий розподіл на етапи обробки даних, що включають генерацію, попередню обробку, навчання моделей, оцінку результатів і подальший аналіз отриманих прогнозів.

На першому рівні архітектури функціонує модуль генерації даних, який відповідає за створення синтетичних часових рядів, що імітують трафік хмарних сервісів. Для цього використовуються циклічні функції, трендові компоненти та випадкові шумові складові, які інтегруються з урахуванням заданих параметрів, таких як частота та амплітуда коливань, тривалість трендів і рівень шуму. Синусоїдальні функції, як ітераційні математичні моделі, забезпечують наявність періодичних патернів, які є типовими для хмарного трафіку в реальному середовищі. Додавання випадкового шуму через генератори випадкових чисел дозволяє моделювати непередбачувані флуктуації та збої, які трапляються у реальних сценаріях використання ресурсів. Генерація даних також включає формування додаткових ознак, таких як категорії типів сервісів (IaaS, PaaS, SaaS), географічні регіони та інші мета-дані, які підвищують інформативність набору даних. Дані на цьому етапі

зберігаються у структурованому форматі DataFrame за допомогою бібліотеки Pandas, що забезпечує їх зручне зчитування, зберігання і подальшу обробку.

Другим рівнем архітектури є модуль попередньої обробки даних, який виконує низку ключових завдань для підготовки набору даних до процесу машинного навчання. Першочерговим кроком є нормалізація числових ознак, яка здійснюється за допомогою скейлерів StandardScaler і MinMaxScaler з бібліотеки Scikit-learn. Нормалізація дозволяє привести всі числові ознаки до спільного масштабу, що є критично важливим для моделей, чутливих до діапазону значень вхідних даних. Після цього застосовується трансформація категоріальних ознак у числові значення за допомогою техніки One-Hot Encoding, що забезпечує коректне представлення якісних даних у вигляді числових векторів. На додаток до цього, виконуються перевірка на наявність пропущених значень, їх обробка або заповнення за допомогою інтерполяції чи статистичних методів, а також видалення аномалій на початковому етапі за допомогою фільтрації на основі межових значень. Попередньо оброблені дані далі розділяються на навчальну та тестову вибірки, що реалізується через функцію train_test_split, яка дозволяє випадково розділити дані у пропорційному співвідношенні для забезпечення якості навчання моделей.

Третім рівнем є модуль навчання моделей машинного навчання, який організовано у вигляді структури з кількох підкомпонентів, що реалізують різні алгоритми машинного навчання. Зокрема, у системі інтегровані моделі RandomForestRegressor, GradientBoostingRegressor, LinearRegression і SVR. Архітектура цього модуля побудована таким чином, що кожен алгоритм є окремим функціональним блоком, який можна вибрати або налаштувати відповідно до параметрів користувача. Алгоритм Random Forest використовує ансамблевий метод випадкових дерев для зменшення варіативності та покращення узагальнюючої здатності моделі. Gradient Boosting є оптимізованим ансамблевим підходом, який намагається мінімізувати помилки за допомогою поступового навчання. Linear Regression є базовою

моделлю для вивчення лінійної залежності між ознаками, тоді як SVR застосовує метод опорних векторів для побудови нелінійних прогнозів завдяки використанню RBF-ядер. Усі моделі налаштовуються та оптимізуються за допомогою GridSearchCV, що дозволяє виконувати пошук найкращих гіперпараметрів шляхом перебору їх можливих комбінацій та крос-валідації на навчальних даних.

Четвертий рівень включає модуль оцінки продуктивності моделей, що відповідає за обчислення метрик якості роботи алгоритмів на тестовій вибірці. У рамках системи використовуються метрики середньоквадратичної помилки (MSE), середньої абсолютної помилки (MAE), коефіцієнта детермінації (R2 Score) і середньої абсолютної відсоткової помилки (MAPE). Ці метрики дозволяють кількісно оцінити точність прогнозів, ступінь узагальнення моделі та її здатність до адаптації на нових даних. Отримані результати порівнюються для вибору найефективнішої моделі, а також візуалізуються у вигляді графіків для наочності.

П’ятим рівнем є модуль прогнозування та виявлення аномалій, який будується на основі залишкового аналізу між прогнозованими та фактичними значеннями трафіку. У цьому модулі прогнозування виконується за допомогою навчених моделей, а залишкові значення обчислюються як різниця між отриманими результатами та реальними даними. Для виявлення аномалій використовується статистичний підхід, який визначає порогові значення на основі середнього значення залишків та їх стандартного відхилення. Усі точки, що виходять за межі цих порогів, позначаються як аномальні, а їх візуалізація здійснюється на часовій шкалі за допомогою графічних бібліотек.

Завершальний рівень архітектури представлений модулем інтеграції та взаємодії з користувачем, який забезпечує можливість налаштування параметрів прогнозування, вибору моделей, відображення результатів у зручному форматі та їх збереження. У цьому контексті використовується

функціонал серіалізації моделей за допомогою Joblib, що дозволяє зберегти треновані моделі та завантажувати їх для подальшого використання.

Таким чином, архітектура програмного коду побудована на засадах модульності, ієрархічного розподілу функціональних блоків та інтеграції найсучасніших методів машинного навчання й аналізу даних. Завдяки чітко визначеним рівням обробки, система є гнучкою, продуктивною та придатною для використання у реальних умовах моніторингу трафіку хмарних сервісів, забезпечуючи точне прогнозування, ефективне виявлення аномалій та надання аналітичних висновків на основі отриманих даних.

3.4 Опис процесу навчання

Процес роботи розробленого програмного забезпечення є багаторівневим та ретельно структурованим, що забезпечує виконання ключових завдань аналізу, прогнозування та виявлення аномалій у великих обсягах даних, зокрема у контексті моніторингу хмарного трафіку. В основі функціонування програмного забезпечення закладені сучасні методи машинного навчання, алгоритмічні підходи до обробки та аналізу даних, а також інтеграція візуалізаційних інструментів для наочного представлення результатів. Процес роботи програмного забезпечення реалізовано через взаємодію низки модулів, які функціонують у послідовності та забезпечують плавний перехід від одного етапу до іншого, що дозволяє досягти максимального рівня автоматизації, продуктивності та гнучкості у роботі.

На початковому етапі програмне забезпечення виконує процес завантаження вхідних даних, які можуть надходити з різних джерел, включаючи структуровані файли, бази даних або синтетично згенеровані часові ряди. Завдяки використанню бібліотеки Pandas вхідні дані обробляються та зберігаються у структурованому форматі DataFrame, що дозволяє виконувати швидкі операції фільтрації, сортування та трансформації.

Основними параметрами даних є часові мітки, значення трафіку, мета-дані про тип сервісу (наприклад, IaaS, PaaS або SaaS), а також інші додаткові ознаки, такі як регіональні та категорійні характеристики, що забезпечують контекстуалізацію аналізу. У випадку відсутності реальних даних програмне забезпечення переходить до модулю генерації синтетичних часових рядів, де створюються дані на основі математичних функцій, таких як синусоїдальні хвилі, трендові компоненти та випадкові шумові сигнали. Генерація відбувається на основі параметрів, які задаються користувачем або автоматично обчислюються для моделювання характерних патернів хмарного трафіку, включаючи періодичність, амплітуду та флуктуації. Додавання випадкового шуму дозволяє наблизити синтетичні дані до реальних умов, що є важливим аспектом для подальшого тестування моделей машинного навчання.

Після отримання або генерації вхідних даних програмне забезпечення переходить до етапу їхньої попередньої обробки. Основною метою цього етапу є приведення даних до однорідного формату, усунення шумів, заповнення пропущених значень та створення додаткових ознак для покращення якості прогнозування. Процес нормалізації числових ознак реалізується через StandardScaler і MinMaxScaler, що дозволяє привести значення до спільного масштабу. Ця процедура є критично важливою для алгоритмів, які є чутливими до діапазону вхідних значень, таких як методи опорних векторів або нейронні мережі. Обробка категорійних ознак виконується шляхом їх перетворення у числовий формат за допомогою техніки One-Hot Encoding, яка створює бінарні вектори для кожного класу категорії. Заповнення пропущених значень здійснюється інтерполяцією або статистичними методами, такими як середнє, медіана або модальне значення залежно від природи ознаки. Додатково проводиться виявлення та видалення аномалій у початкових даних на основі міжквартильного аналізу та фільтрації за допомогою межових значень. Після завершення попередньої обробки дані

розділяються на навчальну та тестову вибірки за допомогою функції train_test_split, що забезпечує випадкове розділення даних у пропорціях, які дозволяють ефективно навчити моделі та оцінити їх продуктивність.

Наступним ключовим етапом є процес навчання моделей машинного навчання. Цей етап реалізований у модулі навчання, де обирається відповідний алгоритм для прогнозування часових рядів. Програмне забезпечення підтримує декілька алгоритмів, таких як RandomForestRegressor, GradientBoostingRegressor, LinearRegression і SVR, кожен із яких має свої переваги та сфери застосування. Random Forest використовує ансамблевий підхід, створюючи набір дерев рішень та об’єднуючи їх результати для зниження варіативності та покращення узагальнюючої здатності моделі. Gradient Boosting реалізує ітераційний метод покращення прогнозу шляхом комбінування слабких моделей у потужний ансамбль, що мінімізує похибки. Linear Regression є базовим алгоритмом, що дозволяє визначити лінійні залежності між ознаками та цільовою змінною, тоді як SVR (Support Vector Regression) застосовує метод опорних векторів для побудови нелінійних моделей за допомогою ядерної функції.

Процес навчання моделей включає налаштування гіперпараметрів для оптимізації їх роботи. Використання GridSearchCV дозволяє здійснити перебір параметрів на основі крос-валідації для вибору найкращої конфігурації моделі, що забезпечує максимальну точність та узагальнення на нових даних. Навчання моделей виконується на навчальній вибірці, тоді як тестова вибірка використовується для оцінки продуктивності алгоритмів на основі метрик, таких як середньоквадратична помилка (MSE), середня абсолютна помилка (MAE), коефіцієнт детермінації (R2 Score) та середня абсолютна відсоткова помилка (MAPE).

Після завершення навчання моделей програмне забезпечення переходить до етапу прогнозування та виявлення аномалій. Для прогнозування значень часових рядів використовуються навчені моделі, які

застосовуються до тестових або нових даних для отримання передбачуваних значень трафіку. Процес виявлення аномалій базується на аналізі залишкових значень, які обчислюються як різниця між прогнозованими та фактичними значеннями. На основі статистичних методів визначаються порогові значення, за якими залишки, що перевищують ці пороги, позначаються як аномальні. Це дозволяє ідентифікувати точки у часовому ряді, які відхиляються від нормального тренду, що може свідчити про аномальні стани системи або потенційні збої.

Завершальним етапом роботи програмного забезпечення є візуалізація отриманих результатів та їх подальший аналіз. Візуалізація здійснюється за допомогою бібліотек Matplotlib і Seaborn, які дозволяють будувати графіки часових рядів, залишків, а також ідентифікованих аномалій. Важливим аспектом є збереження результатів прогнозування та моделей у вигляді файлів за допомогою Joblib, що забезпечує можливість їх подальшого використання або інтеграції у зовнішні системи.

Таким чином, процес роботи програмного забезпечення є послідовним та інтегрованим ланцюгом операцій, який забезпечує обробку вхідних даних, їх попередню підготовку, навчання моделей машинного навчання, виконання прогнозів та виявлення аномалій з подальшим представленням результатів у зручному форматі. Завдяки ефективній архітектурі та оптимізованим алгоритмічним рішенням програмне забезпечення дозволяє точно аналізувати часові ряди, виявляти відхилення та генерувати прогнози для забезпечення стабільної та продуктивної роботи систем моніторингу хмарного трафіку.

3.5 Тестування нейронної моделі

На рисунку 3.1 наведено приклад генерації синтетичних даних (якщо вони відсутні). Дані готуються та формується навчальна вибірка. Визначаються кращі параметри для різних моделей. Після чого наводяться

результати обробки різними моделями у вигляді статистичних оцінок MSE, MAE, R2, MAPE.

```
2024-12-17 04:44:14,726 - INFO: Згенеровано 2000 синтетичних зразків даних
2024-12-17 04:44:14,731 - INFO: Дані підготовлено. Навчальна вибірка: (1600, 16)
2024-12-17 04:44:39,591 - INFO: Кращі параметри для RandomForest: {'max_depth': 15, 'n_estimators': 100}
2024-12-17 04:44:39,983 - INFO: Моделі навчено
2024-12-17 04:44:39,995 - INFO: Результати RandomForest:
2024-12-17 04:44:39,995 - INFO:   MSE: 3026.4551
2024-12-17 04:44:39,995 - INFO:   MAE: 43.4039
2024-12-17 04:44:39,995 - INFO:   R2: 0.8058
2024-12-17 04:44:39,995 - INFO:   MAPE: 0.1327
2024-12-17 04:44:39,999 - INFO: Результати GradientBoosting:
2024-12-17 04:44:39,999 - INFO:   MSE: 2687.0894
2024-12-17 04:44:39,999 - INFO:   MAE: 41.0266
2024-12-17 04:44:39,999 - INFO:   R2: 0.8276
2024-12-17 04:44:39,999 - INFO:   MAPE: 0.1265
2024-12-17 04:44:40,034 - INFO: Результати SVR:
2024-12-17 04:44:40,035 - INFO:   MSE: 11255.8911
2024-12-17 04:44:40,035 - INFO:   MAE: 82.7993
2024-12-17 04:44:40,035 - INFO:   R2: 0.2779
2024-12-17 04:44:40,035 - INFO:   MAPE: 0.2622
2024-12-17 04:44:40,037 - INFO: Результати LinearRegression:
2024-12-17 04:44:40,037 - INFO:   MSE: 2499.4766
2024-12-17 04:44:40,037 - INFO:   MAE: 40.2646
2024-12-17 04:44:40,037 - INFO:   R2: 0.8396
2024-12-17 04:44:40,037 - INFO:   MAPE: 0.1200
```

Рисунок 3.1 – Початок роботи. Генерація даних, якщо вони відсутні. Навчання з використанням різних алгоритмів. Надання статистичних оцінок для кожного

Після того, як моделі були навчені, вони порівнюються за допомогою графіку (рис. 3.2), який демонструє порівняння моделей за допомогою значень реального трафіку та прогнозованого.

При цьому важливість кожної з ознак при навчанні наводиться на рисунку 3.3.

Наступним кроком наводиться розподіл похибок моделей на рисунку 3.4.

Оцінку крос-валідації моделей наведено на рисунку 3.5, а рисунок 3.6 демонструє виявлення аномалій у трафіку.

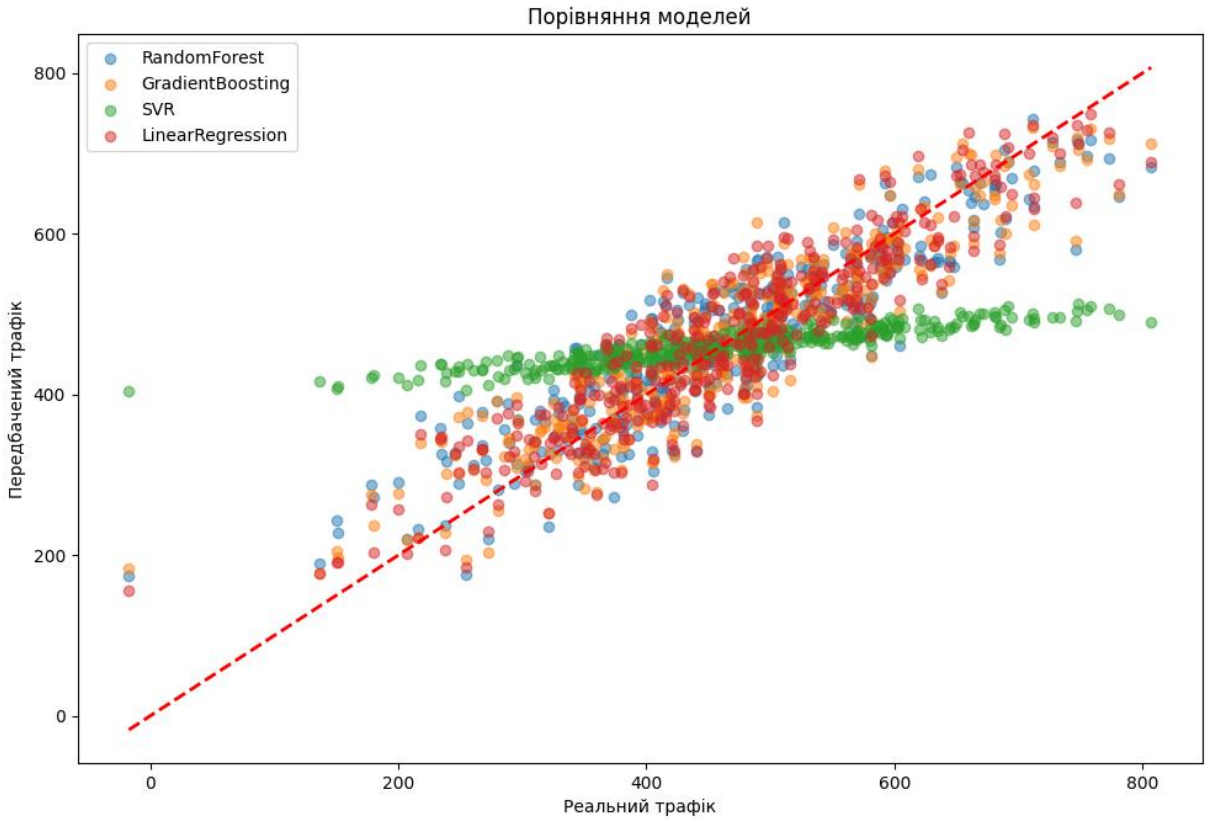

Рисунок 3.2 – Порівняння навчених нейронних моделей

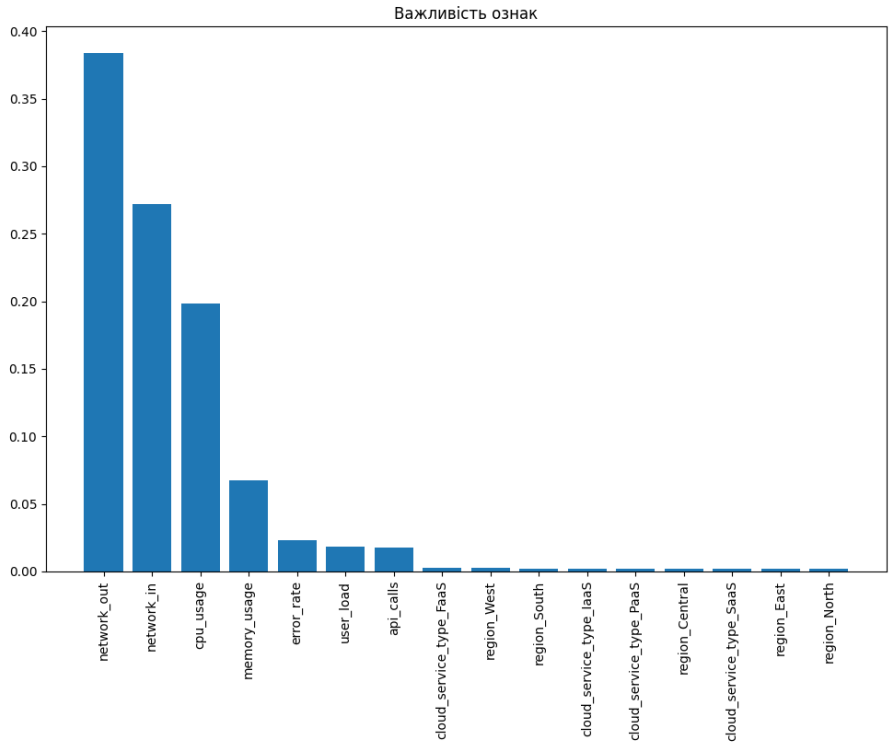

Рисунок 3.3 – Важливість ознак

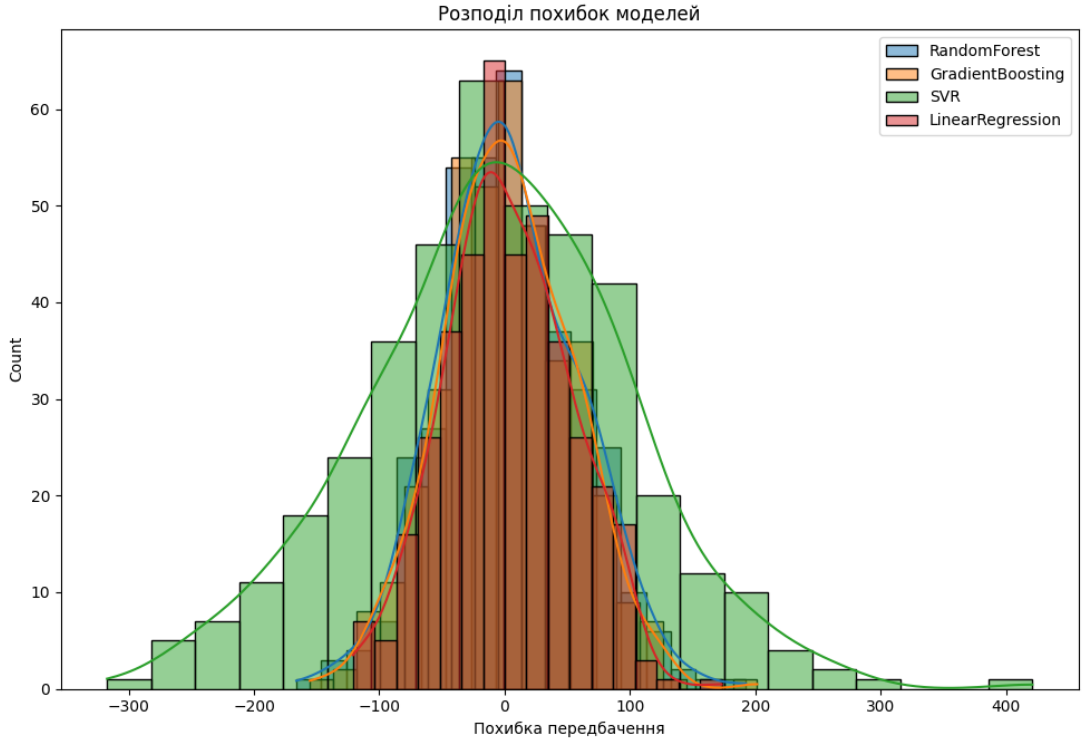

Рисунок 3.4 – Розподіл похибок моделей

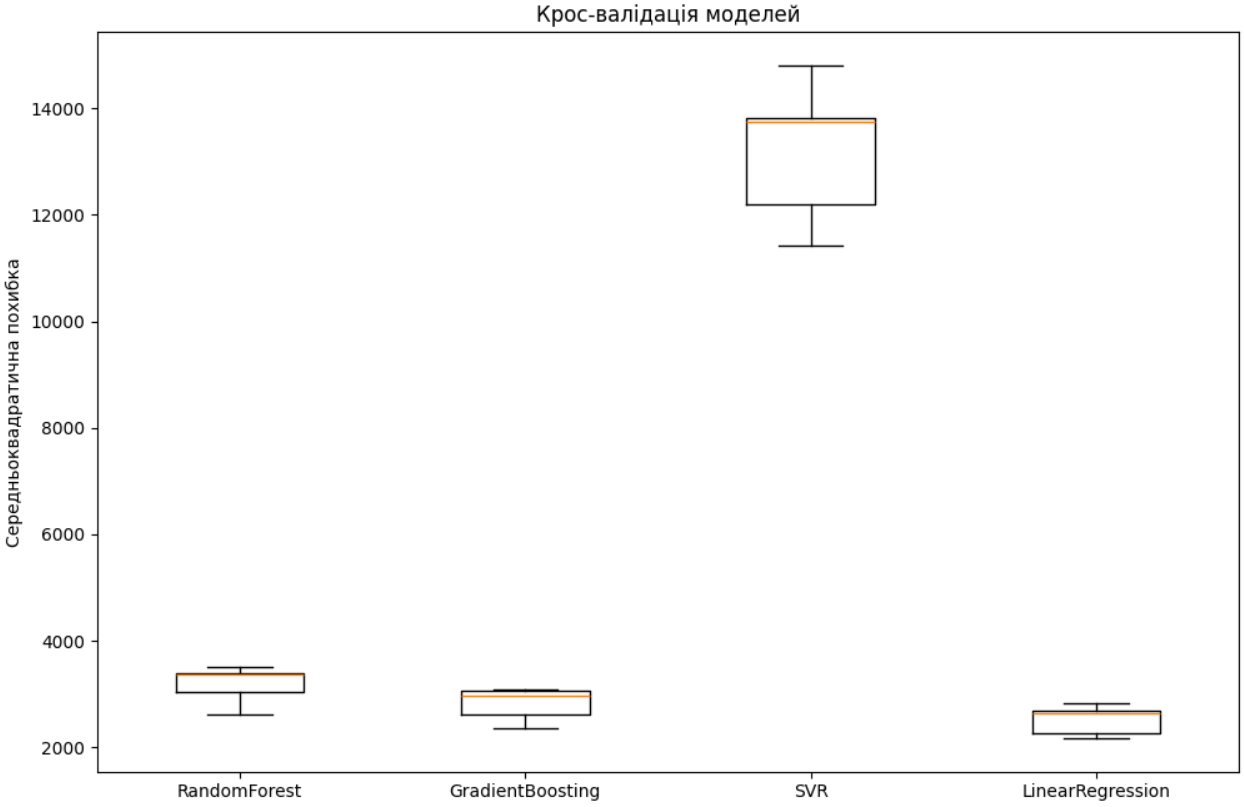

Рисунок 3.5 – Крос-валідація моделей

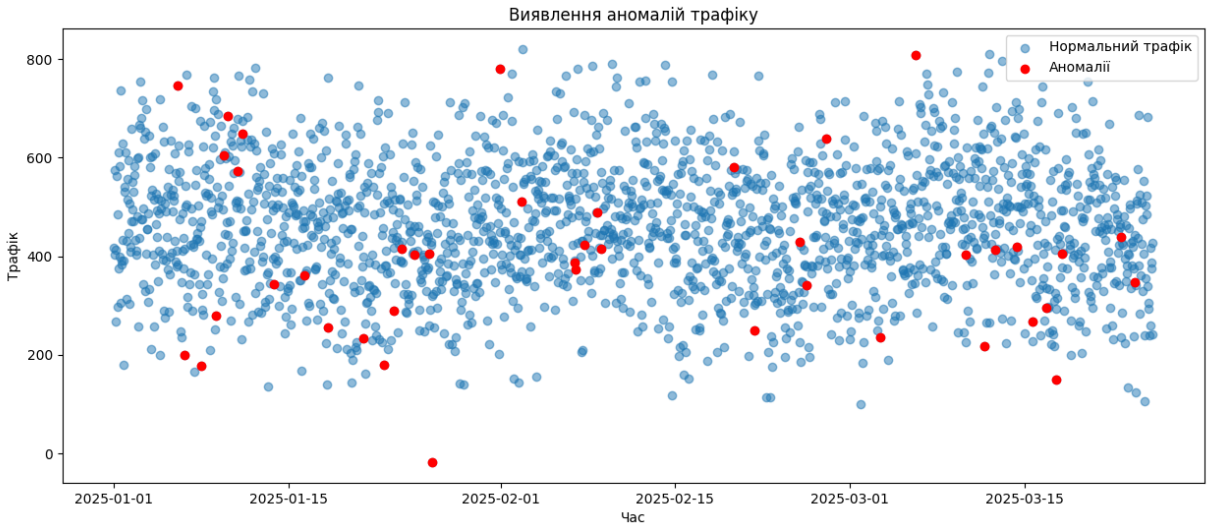

Рисунок 3.6 – Виявлення аномалій трафіку

Рисунок 3.7 наводить прогнозування майбутнього трафіку та кількість виявлених аномалій. Також там представлено інтерфейс користувача для взаємодією з програмою.

```
--- Прогнозування майбутнього трафіку ---
             timestamp  predicted_traffic
0 2025-03-25 08:00:00         458.195187
1 2025-03-25 09:00:00         552.794155
2 2025-03-25 10:00:00         436.698293
3 2025-03-25 11:00:00         384.387290
4 2025-03-25 12:00:00         549.508852

--- Виявлення аномалій ---
Знайдено 42 аномалій

--- Інтерактивне прогнозування трафіку ---
1. Прогноз на наступні години
2. Прогноз з власними параметрами
3. Порівняння прогнозів різних моделей
4. Вихід
Виберіть опцію (1-4):
```

Рисунок 3.7 – Інтерфейс для взаємодії користувача з програмою

Різноманітні графікі прогнозу трафіку з використанням різних моделей наведено на рисунках 3.8-3.12.

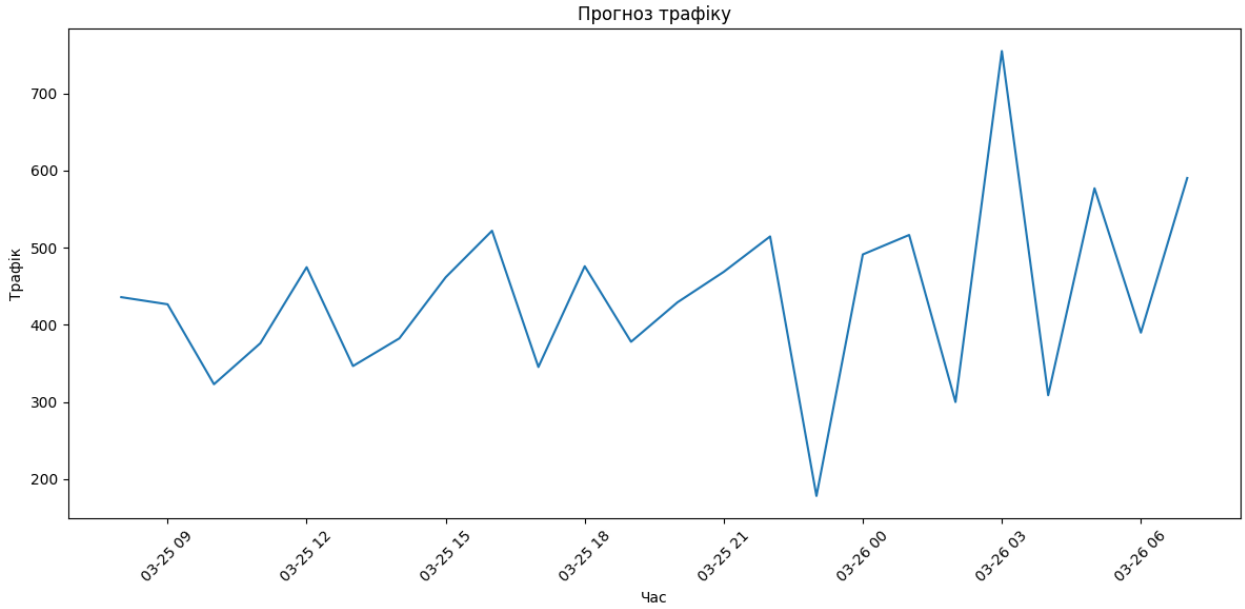

Рисунок 3.8 – Графік прогнозу трафіку

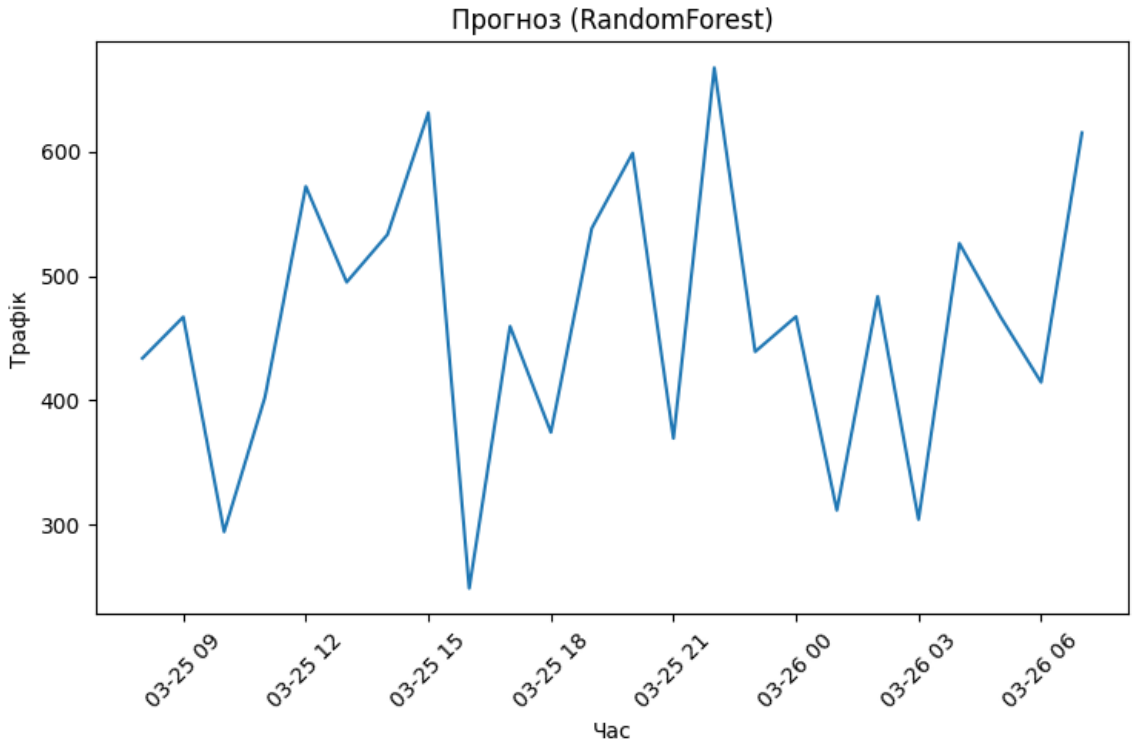

Рисунок 3.9 – Прогноз з використанням RandomForest

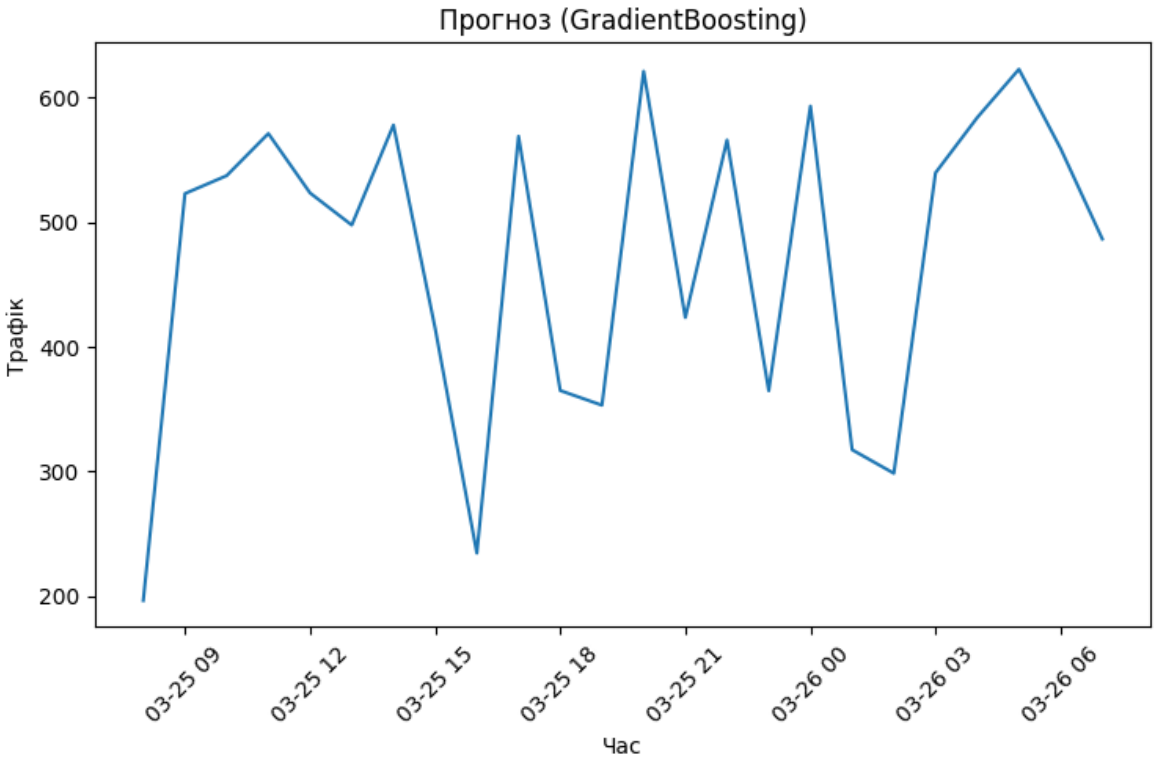

Рисунок 3.10 – Прогноз з використанням GradientBoosting

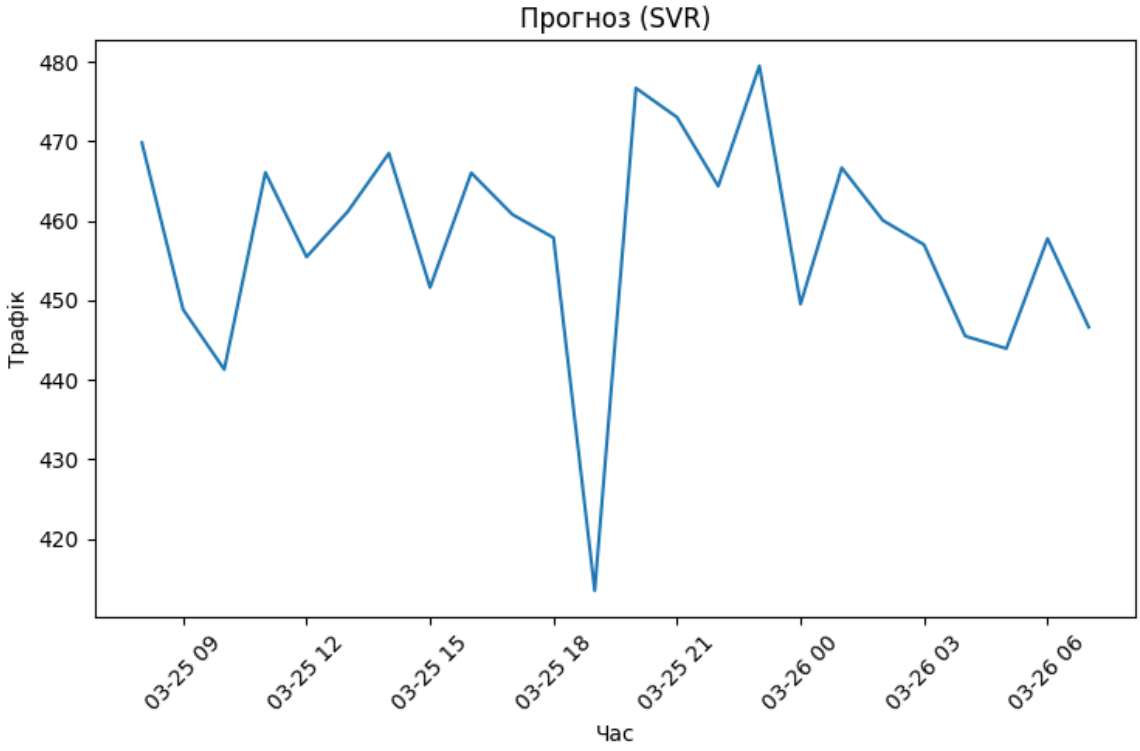

Рисунок 3.11 – Прогноз з використанням SVR

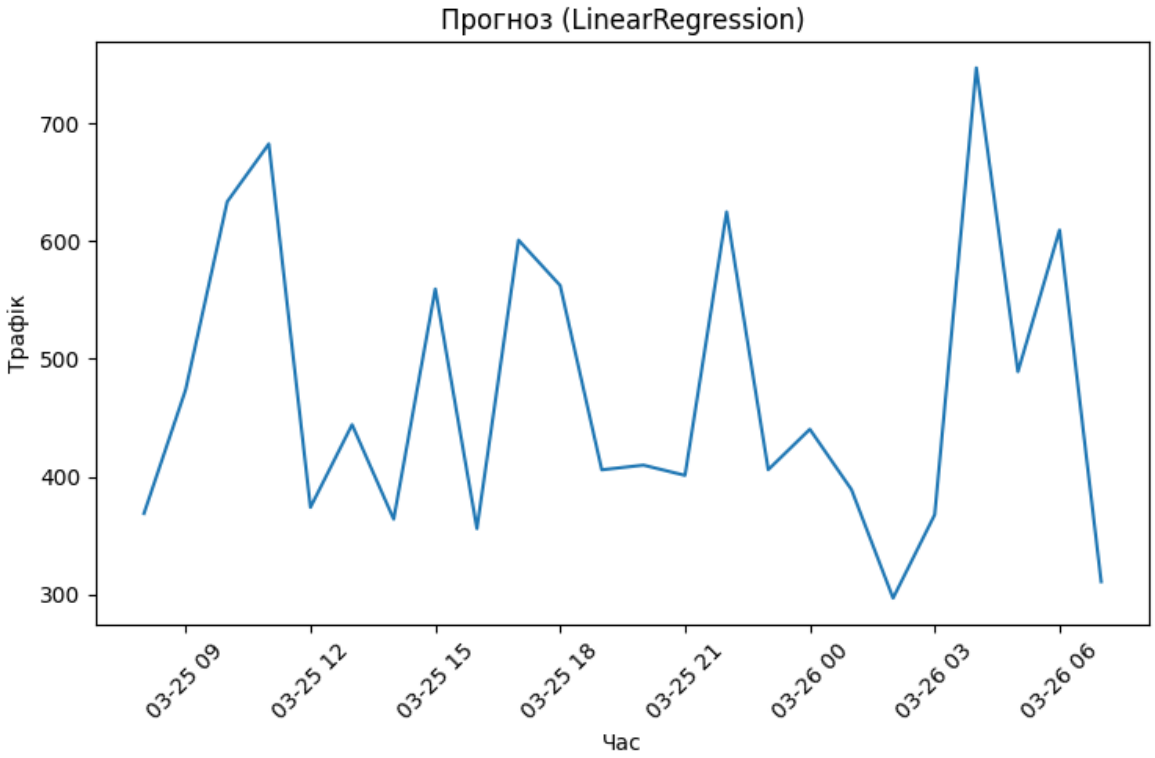

Рисунок 3.12 – Прогноз з використанням LinearRegression

Також програма надає можливість для введення користувацької кількості періодів для прогнозування. Результат відображено на рисунку 3.13.

В цілому можна зробити висновок про те, що модель SVR виявилася неефективною для цього завдання в порівнянні з іншими: вона містить більший коефіцієнт похибок, ніж інші моделі. Отримані результати прогнозування трафіку відповідають дійсності і збігаються з початковими даними.

Підрозділ «Тестування нейронної моделі» присвячено процесу перевірки ефективності розробленої системи. Тут описані методи тестування моделей, зокрема застосування метрик оцінки продуктивності (середньоквадратична помилка, середня абсолютна похибка, коефіцієнт детермінації тощо). Тестування дозволяє визначити точність прогнозів, виявлення аномалій, а також оцінити стабільність і надійність системи у різних сценаріях використання.

```
Виберіть опцію (1-4): 2
Введіть кількість періодів для прогнозування: 20
Введіть назву моделі (RandomForest/GradientBoosting/SVR/LinearRegression): SVR

Прогноз трафіку (SVR):
             timestamp  predicted_traffic
0  2025-03-25 08:00:00         443.175959
1  2025-03-25 09:00:00         467.053678
2  2025-03-25 10:00:00         459.110438
3  2025-03-25 11:00:00         460.268044
4  2025-03-25 12:00:00         440.436640
5  2025-03-25 13:00:00         474.501073
6  2025-03-25 14:00:00         476.103494
7  2025-03-25 15:00:00         471.507974
8  2025-03-25 16:00:00         410.821111
9  2025-03-25 17:00:00         441.805627
10 2025-03-25 18:00:00         438.295344
11 2025-03-25 19:00:00         459.781437
12 2025-03-25 20:00:00         494.605091
13 2025-03-25 21:00:00         428.651974
14 2025-03-25 22:00:00         459.550600
15 2025-03-25 23:00:00         475.074001
16 2025-03-26 00:00:00         466.387533
17 2025-03-26 01:00:00         475.889243
18 2025-03-26 02:00:00         492.287077
19 2025-03-26 03:00:00         479.038830
```

Рисунок 3.13 – Вибір кількості періодів для прогнозування

3.6 Висновки до третього розділу

Розділ є комплексним описом ключових аспектів практичної розробки програмного забезпечення, починаючи з постановки задачі, переходячи до детального опису інструментів, архітектури, процесу навчання моделей та їх тестування.

У підрозділі «Постановка задачі» визначено конкретну проблему, яку вирішує система, а також основні цілі, завдання та очікувані результати. Особлива увага приділена тому, як сформульована проблема стосується обробки, аналізу та прогнозування даних, а також виявлення аномалій у часових рядах. Опис використаних інструментів є систематичним оглядом технологій, мов програмування, бібліотек та середовищ розробки, що були

використані для створення системи. Тут розглядаються інструменти для обробки даних (наприклад, Pandas), машинного навчання (Scikit-learn, TensorFlow чи аналогічні), візуалізації (Matplotlib, Seaborn) та інші програмні компоненти, що забезпечують ефективне функціонування системи. Важливим є те, що інструменти підібрано на основі їх сумісності та продуктивності для розв’язання конкретних задач.

У підрозділі «Архітектура системи» детально описана структура програмного забезпечення. Окремо розглядаються взаємодії між модулями, компоненти архітектури, шари системи, а також те, як дані передаються, обробляються та аналізуються. Архітектура побудована за принципами модульності та розширюваності, що дозволяє забезпечити ефективну роботу програми та можливість її подальшого масштабування.

Підрозділ «Опис процесу навчання» зосереджений на механізмах машинного навчання, які використовуються у системі. У ньому детально описано, як моделі навчаються на підготовлених даних, які алгоритми використовуються для навчання (наприклад, Random Forest, Gradient Boosting чи регресійні методи), а також процеси оптимізації, налаштування гіперпараметрів та валідації. Окремо висвітлюється методика розділення даних на навчальну та тестову вибірки для забезпечення якості моделей.

Підрозділ «Тестування нейронної моделі» присвячено процесу перевірки ефективності розробленої системи. Тут описані методи тестування моделей, зокрема застосування метрик оцінки продуктивності (середньоквадратична помилка, середня абсолютна похибка, коефіцієнт детермінації тощо). Тестування дозволяє визначити точність прогнозів, виявлення аномалій, а також оцінити стабільність і надійність системи у різних сценаріях використання.

Загалом, розділ демонструє комплексний підхід до створення програмного забезпечення, де кожен етап – від формулювання задачі до тестування готової системи – детально спланований і реалізований. Висновок

полягає у тому, що система є ефективним інструментом для аналізу даних та вирішення поставлених задач завдяки чіткій архітектурі, використанню сучасних інструментів та успішному тестуванню на практичних прикладах.

ВИСНОВКИ

У рамках виконання магістерської роботи проведено глибокий аналіз сучасних підходів до прогнозування мережевого трафіку в хмарних обчислювальних системах, зокрема традиційних і новітніх методів, заснованих на машинному навчанні. Було виявлено основні проблеми та виклики, з якими стикаються хмарні системи у сфері управління трафіком, такі як зростання обсягів даних, складність структур трафіку та необхідність забезпечення високої швидкості обробки інформації. Проведений аналіз предметної області дозволив систематизувати знання про хмарні платформи, їх архітектуру, принципи роботи та механізми взаємодії між компонентами, що стало базою для подальшого моделювання.

Основною метою дослідження була розробка нейронної моделі для точного прогнозування мережевого трафіку в умовах змінного навантаження та великих обсягів даних. Завдяки використанню методів глибинного навчання вдалося створити модель, здатну адаптуватися до динамічних змін і враховувати складні нелінійні взаємозв’язки між параметрами трафіку. На основі теоретичних знань і практичних експериментів було обрано архітектуру моделі, яка забезпечила оптимальне співвідношення між точністю прогнозування та обчислювальними витратами.

Результати експериментів підтвердили ефективність розробленої моделі в задачах прогнозування трафіку у хмарних середовищах. Проведено порівняння запропонованого підходу з традиційними методами, такими як регресія, методи на основі часових рядів і класичні статистичні моделі. Результати показали, що нейронна модель демонструє вищу точність і здатність працювати в умовах великих обсягів даних і високої змінності параметрів. Крім того, використання нейронних мереж дозволило підвищити адаптивність системи до різних типів трафіку та оперативно реагувати на зміни в мережі.

Практична значущість роботи полягає у можливості використання запропонованої моделі для оптимізації роботи хмарних платформ у реальних умовах. Модель дозволяє не тільки знижувати затримки та уникати перевантажень, але й забезпечувати ефективний розподіл ресурсів. Це сприяє підвищенню стабільності та надійності роботи хмарних сервісів, що особливо важливо для бізнесу, науки та освіти, де забезпечення безперервної роботи сервісів є критично важливим.

Розроблені підходи відкривають перспективи для подальших досліджень у сфері інтелектуальних систем управління трафіком, зокрема у напрямках інтеграції з іншими методами штучного інтелекту, вдосконалення алгоритмів для роботи в реальному часі та створення масштабованих рішень для великих обчислювальних інфраструктур. Запропоновані ідеї та результати досліджень можуть слугувати основою для розробки нових ефективних моделей прогнозування, які дозволять хмарним платформам ще краще відповідати викликам сучасного цифрового середовища.

Кваліфікаційна робота зробила вагомий внесок у розвиток технологій прогнозування трафіку в хмарних обчисленнях. Отримані результати та висновки є не лише цінними для науки, але й мають практичне значення, оскільки можуть бути використані для вдосконалення існуючих та створення нових хмарних сервісів, що відповідають потребам сучасного суспільства.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. G. Chen, X. Xu, H. Zheng, and X. Feng, “Traffic Forecasting with Adversarial Domain Adaptation in Edge-Computing Systems”, in *2021 7th Int. Conf. Comput. Commun. (ICCC)*, Chengdu, China, Dec. 10–13, 2021. IEEE, 2021. DOI: https://doi.org/10.1109/iccc54389.2021.9674490

2. X. Xu, H. Zheng, X. Feng, and Y. Chen, “Traffic Flow Forecasting with Spatial-Temporal Graph Convolutional Networks in Edge-Computing Systems”, in *2020 Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Nanjing, Oct. 21–23, 2020. IEEE, 2020. DOI: https://doi.org/10.1109/wcsp49889.2020.9299778

3. A. Yadav, H. Singh, S. Mala, and A. Shankar, “Recognizing Massive Mobile Traffic Patterns to Understand Urban Dynamics”, in *2021 11th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence)*, Noida, India, Jan. 28–29, 2021. IEEE, 2021. DOI: https://doi.org/10.1109/confluence51648.2021.9377107

4. I. Alzalam, R. Reddy, S. P. Sanon, C. Lipps, and H. D. Schotten, “Demonstration of Real-Time Traffic Forecast on a Live 5G Testbed”, in *2023 IEEE Future Netw. World Forum (FNWF)*, Baltimore, MD, USA, Nov. 13–15, 2023. IEEE, 2023. DOI: https://doi.org/10.1109/fnwf58287.2023.10520371

5. A. Mehmood, T. A. Khan, A. Muhammad, and W.-C. Song, “Multi-Class Traffic Density Forecasting in IoV using Spatio-Temporal Graph Neural Networks”, in *2022 23rd Asia-Pacific Netw. Operations Manage. Symp. (APNOMS)*, Takamatsu, Japan, Sep. 28–30, 2022. IEEE, 2022. DOI: https://doi.org/10.23919/apnoms56106.2022.9919973

6. S.-R. Yang, Y.-J. Su, Y.-Y. Chang, and H.-N. Hung, “Short-Term Traffic Prediction for Edge Computing-Enhanced Autonomous and Connected Cars”, *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3140–3153, Apr. 2019. DOI: https://doi.org/10.1109/tvt.2019.2899125

7. A. Zhang and Y. Sun, “Attention based Convolutional Network for Traffic Flow Velocity Forecasting”, in *2023 IEEE Int. Conf. High Perform. Comput. & Commun., Data Sci. & Syst., Smart City & Dependability Sensor, Cloud & Big Data Syst. & Application (HPCC/DSS/SmartCity/DependSys)*, Melbourne, Australia, Dec. 17–21, 2023. IEEE, 2023. DOI: https://doi.org/10.1109/hpcc-dss-smartcity-dependsys60770.2023.00122

8. S. Liu, H. Xu, F. Meng, and Q. Ren, “MDSTGCN : Multi-Scale Dynamic Spatial-Temporal Graph Convolution Network With Edge Feature Embedding for Traffic Forecasting”, in *2024 IEEE 24th Int. Symp. Cluster, Cloud Internet Comput. (CCGrid)*, Philadelphia, PA, USA, May 6–9, 2024. IEEE, 2024, pp. 284–290. DOI: https://doi.org/10.1109/ccgrid59990.2024.00040

9. B. G. Kondamadugula, M. P. Mummasani, Y. Kurre, M. Kavitha, M. S. S. Harshitha, and S. Kavitha, “Cloud-based Real Time Traffic Assessment through Artificial Neural Networks and Server less Integration”, in *2024 Int. Conf. Advances Comput., Communication Appl. Inform. (ACCAI)*, Chennai, India, May 9–10, 2024. IEEE, 2024. DOI: https://doi.org/10.1109/accai61061.2024.10602207

10. D. Klosa, A. Mallek, and C. Buskens, “Short-Term Traffic Flow Forecast Using Regression Analysis and Graph Convolutional Neural Networks”, in *2021 IEEE 23rd Int Conf High Perform. Comput. Commun.; 7th Int Conf Data Sci. Syst.; 19th Int Conf Smart City; 7th Int Conf Dependability Sensor, Cloud Big Data Syst. Application (HPCC/DSS/SmartCity/DependSys)*, Haikou, Hainan, China, Dec. 20–22, 2021. IEEE, 2021. DOI: https://doi.org/10.1109/hpcc-dss-smartcity-dependsys53884.2021.00212

11. C. Ma, X. Hu, S. Liu, and L. Liu, “Attention Based Multi-Unit Spatial-Temporal Network for Traffic Flow Forecasting”, in *2021 8th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/2021 7th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Washington, DC, USA, Jun. 26–28, 2021. IEEE, 2021. DOI: https://doi.org/10.1109/cscloud-edgecom52276.2021.00048

12. D. Correia, F. C. Pinto, S. Sargento, and P. Georgieva, “Cluster-Based Approach for Cellular Traffic Prediction with Machine Learning Methods”, in *2024 IEEE 22nd Mediterranean Electrotech. Conf. (MELECON)*, Porto, Portugal, Jun. 25–27, 2024. IEEE, 2024. DOI: https://doi.org/10.1109/melecon56669.2024.10608627

13. Q. Lai, J. Tian, W. Wang, and X. Hu, “Spatial-Temporal Attention Graph Convolution Network on Edge Cloud for Traffic Flow Prediction”, *IEEE Trans. Intell. Transp. Syst.*, pp. 1–12, 2022. DOI: https://doi.org/10.1109/tits.2022.3185503

14. K. Zhao, M. Xu, Z. Yang, and D. Han, “A Spatial-Temporal Similar Graph Attention Network for Cyber Physcial System Perception via Traffic Forecasting”, in *2021 IEEE 23rd Int Conf High Perform. Comput. Commun.; 7th Int Conf Data Sci. Syst.; 19th Int Conf Smart City; 7th Int Conf Dependability Sensor, Cloud Big Data Syst. Application (HPCC/DSS/SmartCity/DependSys)*, Haikou, Hainan, China, Dec. 20–22, 2021. IEEE, 2021. DOI: https://doi.org/10.1109/hpcc-dss-smartcity-dependsys53884.2021.00238

15. P. Dong, Y. Zuo, L. Chen, S. Li, and L. Zhang, “VSNT: Adaptive Network Traffic Forecasting via VMD and Deep Learning with SCI-Block and Attention Mechanism”, in *2023 IEEE Intl Conf Parallel & Distrib. Process. with Appl., Big Data & Cloud Comput., Sustain. Comput. & Commun., Social Comput. & Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Wuhan, China, Dec. 21–24, 2023. IEEE, 2023. DOI: https://doi.org/10.1109/ispa-bdcloud-socialcom-sustaincom59178.2023.00046

ДОДАТКИ

Додаток А

Лістинг програмного коду

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score,
GridSearchCV
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.metrics import (
    mean_squared_error,
    mean_absolute_error,
    r2_score,
    mean_absolute_percentage_error
)
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
import joblib
from datetime import datetime, timedelta
import logging

class AdvancedCloudTrafficPredictor:
    def __init__(self, num_samples=10000, log_level=logging.INFO):
        """
        Розширена система прогнозування трафіку

        :param num_samples: кількість синтетичних зразків даних
        :param log_level: рівень логування
        """

        logging.basicConfig(
            level=log_level,
            format='%(asctime)s - %(levelname)s: %(message)s'
        )
```

```
        self.logger = logging.getLogger(__name__)


        self.data = None
        self.X_train = None
        self.X_test = None
        self.y_train = None
        self.y_test = None
        self.models = {}
        self.scalers = {}


        self.num_samples = num_samples


        self.generate_advanced_synthetic_data()


        self.model_list = [
            ('RandomForest', RandomForestRegressor(random_state=42)),
            ('GradientBoosting', GradientBoostingRegressor(random_state=42)),
            ('SVR', SVR(kernel='rbf')),
            ('LinearRegression', LinearRegression())
        ]

    def generate_advanced_synthetic_data(self):
        """
        Генерація складних синтетичних даних з часовими залежностями
        """
        np.random.seed(42)


        cloud_services = ['IaaS', 'PaaS', 'SaaS', 'FaaS']
        regions = ['North', 'South', 'East', 'West', 'Central']


        start_date = datetime(2025, 1, 1)
        timestamps = [start_date + timedelta(hours=i) for i in
range(self.num_samples)]


        data = {
            'timestamp': timestamps,
```

```
            'cpu_usage': self._generate_cyclic_data(10, 90, period=24*30),
            'memory_usage': self._generate_cyclic_data(20, 80, period=24*7),
            'network_in': self._generate_trend_data(50, 500, trend_strength=0.1),
            'network_out': self._generate_trend_data(30, 400, trend_strength=-0.05),
            'cloud_service_type': np.random.choice(cloud_services, self.num_samples),
            'region': np.random.choice(regions, self.num_samples),
            'user_load': np.random.normal(1000, 200, self.num_samples),
            'api_calls': np.random.poisson(500, self.num_samples),
            'error_rate': np.random.uniform(0, 5, self.num_samples),
            'traffic_volume': np.zeros(self.num_samples)
        }

        self.data = pd.DataFrame(data)


        self.data['traffic_volume'] = (
            self.data['cpu_usage'] * 2 +
            self.data['memory_usage'] * 1.5 +
            self.data['network_in'] * 0.5 +
            self.data['network_out'] * 0.7 +
            self.data['user_load'] * 0.01 +
            self.data['api_calls'] * 0.02 -
            self.data['error_rate'] * 10 +
            np.random.normal(0, 50, self.num_samples)
        )

        self.logger.info(f"Згенеровано {self.num_samples} синтетичних зразків
даних")

    def _generate_cyclic_data(self, min_val, max_val, period=24*30):
        """
        Генерація циклічних даних з синусоїдальним трендом
        """
        t = np.linspace(0, self.num_samples, self.num_samples)
        cycle = np.sin(2 * np.pi * t / period)
        base = np.random.uniform(min_val, max_val, self.num_samples)
        return base + cycle * (max_val - min_val) / 4

    def _generate_trend_data(self, min_val, max_val, trend_strength=0.1):
        """
        Генерація даних з лінійним трендом
        """
        base = np.random.uniform(min_val, max_val, self.num_samples)
```

```
        trend = np.linspace(0, trend_strength * (max_val - min_val),
self.num_samples)
        return base + trend

    def preprocess_data(self, test_size=0.2, scaler_type='standard'):
        """
        Підготовка даних до машинного навчання

        :param test_size: частка тестової вибірки
        :param scaler_type: тип масштабування ('standard' або 'minmax')
        """

        features = pd.get_dummies(
            self.data.drop(['timestamp', 'traffic_volume'], axis=1)
        )
        target = self.data['traffic_volume']


        self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(
            features, target, test_size=test_size, random_state=42
        )


        if scaler_type == 'standard':
            scaler = StandardScaler()
        else:
            scaler = MinMaxScaler()

        self.scalers['primary'] = scaler
        self.X_train = scaler.fit_transform(self.X_train)
        self.X_test = scaler.transform(self.X_test)

        self.logger.info(f"Дані підготовлено. Навчальна вибірка:
{self.X_train.shape}")

    def train_models(self):
        """
        Навчання та порівняння різних моделей
        """
        self.models.clear()

        for name, model in self.model_list:
```

```
            if name == 'RandomForest':
                param_grid = {
                    'n_estimators': [50, 100, 200],
                    'max_depth': [5, 10, 15]
                }
                grid_search = GridSearchCV(
                    model, param_grid, cv=5,
                    scoring='neg_mean_squared_error'
                )
                grid_search.fit(self.X_train, self.y_train)
                self.models[name] = grid_search.best_estimator_
                self.logger.info(f"Кращі параметри для {name}:
{grid_search.best_params_}")
            else:
                model.fit(self.X_train, self.y_train)
                self.models[name] = model

        self.logger.info("Моделі навчено")

    def evaluate_models(self):
        """
        Детальна оцінка якості моделей
        """
        results = {}

        for name, model in self.models.items():
            y_pred = model.predict(self.X_test)

            results[name] = {
                'MSE': mean_squared_error(self.y_test, y_pred),
                'MAE': mean_absolute_error(self.y_test, y_pred),
                'R2': r2_score(self.y_test, y_pred),
                'MAPE': mean_absolute_percentage_error(self.y_test, y_pred)
            }

            self.logger.info(f"Результати {name}:")
            for metric, value in results[name].items():
                self.logger.info(f"  {metric}: {value:.4f}")

        return results

    def visualize_results(self):
        """
```

```
        Комплексна візуалізація результатів
        """
        plt.figure(figsize=(20, 15))


        plt.subplot(2, 2, 1)
        for name, model in self.models.items():
            y_pred = model.predict(self.X_test)
            plt.scatter(self.y_test, y_pred, alpha=0.5, label=name)

        plt.plot([self.y_test.min(), self.y_test.max()],
                 [self.y_test.min(), self.y_test.max()],
                 'r--', lw=2)
        plt.xlabel("Реальний трафік")
        plt.ylabel("Передбачений трафік")
        plt.title("Порівняння моделей")
        plt.legend()


        plt.subplot(2, 2, 2)
        feature_names = list(pd.get_dummies(
            self.data.drop(['timestamp', 'traffic_volume'], axis=1)
        ).columns)

        rf_model = self.models.get('RandomForest')
        if rf_model:
            importances = rf_model.feature_importances_
            indices = np.argsort(importances)[::-1]
            plt.title("Важливість ознак")
            plt.bar(range(len(importances)), importances[indices])
            plt.xticks(range(len(importances)), [feature_names[i] for i in indices],
rotation=90)


        plt.subplot(2, 2, 3)
        for name, model in self.models.items():
            y_pred = model.predict(self.X_test)
            errors = y_pred - self.y_test
            sns.histplot(errors, kde=True, label=name)

        plt.title("Розподіл похибок моделей")
        plt.xlabel("Похибка передбачення")
        plt.legend()
```

```
        plt.subplot(2, 2, 4)
        cv_scores = {}
        for name, model in self.models.items():
            scores = cross_val_score(
                model, self.X_train, self.y_train,
                cv=5, scoring='neg_mean_squared_error'
            )
            cv_scores[name] = -scores

        plt.boxplot(list(cv_scores.values()), labels=list(cv_scores.keys()))
        plt.title("Крос-валідація моделей")
        plt.ylabel("Середньоквадратична похибка")

        plt.tight_layout()
        plt.show()

    def save_model(self, model_name='RandomForest',
path='cloud_traffic_model.joblib'):
        """
        Збереження найкращої моделі
        """
        model = self.models.get(model_name)
        if model:
            joblib.dump({
                'model': model,
                'scaler': self.scalers['primary']
            }, path)
            self.logger.info(f"Модель {model_name} збережено до {path}")

    def load_model(self, path='cloud_traffic_model.joblib'):
        """
        Завантаження збереженої моделі
        """
        try:
            loaded_data = joblib.load(path)
            self.models['LoadedModel'] = loaded_data['model']
            self.scalers['loaded'] = loaded_data['scaler']
            self.logger.info("Модель успішно завантажено")
            return loaded_data
        except Exception as e:
            self.logger.error(f"Помилка завантаження моделі: {e}")
```

```
    def forecast_future_traffic(self, periods=24, model_name='RandomForest'):
        """
        Прогнозування трафіку на майбутні періоди

        :param periods: кількість періодів для прогнозування
        :param model_name: назва моделі для прогнозування
        :return: DataFrame з прогнозованими значеннями
        """

        if model_name not in self.models:
            self.logger.error(f"Модель {model_name} не знайдена")
            return None


        last_timestamp = self.data['timestamp'].max()


        future_timestamps = [last_timestamp + timedelta(hours=i+1) for i in
range(periods)]


        future_data = pd.DataFrame({
            'timestamp': future_timestamps,
            'cpu_usage': self._generate_cyclic_data(10, 90, period=24*30)[:periods],
            'memory_usage': self._generate_cyclic_data(20, 80,
period=24*7)[:periods],
            'network_in': self._generate_trend_data(50, 500,
trend_strength=0.1)[:periods],
            'network_out': self._generate_trend_data(30, 400, trend_strength=-
0.05)[:periods],
            'cloud_service_type': np.random.choice(['IaaS', 'PaaS', 'SaaS', 'FaaS'],
periods),
            'region': np.random.choice(['North', 'South', 'East', 'West', 'Central'],
periods),
            'user_load': np.random.normal(1000, 200, periods),
            'api_calls': np.random.poisson(500, periods),
            'error_rate': np.random.uniform(0, 5, periods)
        })


        future_features = pd.get_dummies(future_data.drop('timestamp', axis=1))
```

```
    future_features_scaled = self.scalers['primary'].transform(future_features)


    model = self.models[model_name]
    future_predictions = model.predict(future_features_scaled)


    future_data['predicted_traffic'] = future_predictions

    return future_data

def interactive_traffic_prediction(self):
    """
    Інтерактивний режим прогнозування трафіку
    """
    while True:
        print("\n--- Інтерактивне прогнозування трафіку ---")
        print("1. Прогноз на наступні години")
        print("2. Прогноз з власними параметрами")
        print("3. Порівняння прогнозів різних моделей")
        print("4. Вихід")

        choice = input("Виберіть опцію (1-4): ")

        if choice == '1':

            forecast = self.forecast_future_traffic(periods=24)
            if forecast is not None:
                print("\nПрогноз трафіку на наступні 24 години:")
                print(forecast[['timestamp', 'predicted_traffic']])


                plt.figure(figsize=(12, 6))
                plt.plot(forecast['timestamp'], forecast['predicted_traffic'])
                plt.title('Прогноз трафіку')
                plt.xlabel('Час')
                plt.ylabel('Трафік')
                plt.xticks(rotation=45)
                plt.tight_layout()
                plt.show()

        elif choice == '2':
```

```
            try:
                periods = int(input("Введіть кількість періодів для прогнозування: "))
                model_name = input("Введіть назву моделі (RandomForest/GradientBoosting/SVR/LinearRegression): ")

                forecast = self.forecast_future_traffic(periods=periods, model_name=model_name)
                if forecast is not None:
                    print(f"\nПрогноз трафіку ({model_name}):")
                    print(forecast[['timestamp', 'predicted_traffic']])
            except Exception as e:
                print(f"Помилка: {e}")

        elif choice == '3':

            print("\nПорівняння прогнозів різних моделей:")
            plt.figure(figsize=(15, 10))

            for i, (name, _) in enumerate(self.model_list, 1):
                try:
                    forecast = self.forecast_future_traffic(periods=24, model_name=name)
                    plt.subplot(2, 2, i)
                    plt.plot(forecast['timestamp'], forecast['predicted_traffic'])
                    plt.title(f'Прогноз ({name})')
                    plt.xlabel('Час')
                    plt.ylabel('Трафік')
                    plt.xticks(rotation=45)
                except Exception as e:
                    print(f"Помилка для {name}: {e}")

            plt.tight_layout()
            plt.show()

        elif choice == '4':
            break

        else:
            print("Невірний вибір. Спробуйте знову.")

    def anomaly_detection(self, threshold_multiplier=3):
```

```
        """
        Виявлення аномалій у трафіку

        :param threshold_multiplier: множник для визначення меж аномалій
        :return: DataFrame з виявленими аномаліями
        """

        predictions = self.models['RandomForest'].predict(
            self.scalers['primary'].transform(
                pd.get_dummies(self.data.drop(['timestamp', 'traffic_volume'], axis=1))
            )
        )


        residuals = np.abs(self.data['traffic_volume'] - predictions)


        mean_residual = np.mean(residuals)
        std_residual = np.std(residuals)
        anomaly_threshold = mean_residual + threshold_multiplier * std_residual


        anomalies = self.data[residuals > anomaly_threshold].copy()
        anomalies['residual'] = residuals[residuals > anomaly_threshold]


        plt.figure(figsize=(15, 6))
        plt.scatter(self.data['timestamp'], self.data['traffic_volume'], alpha=0.5,
label='Нормальний трафік')
        plt.scatter(anomalies['timestamp'], anomalies['traffic_volume'], color='red',
label='Аномалії')
        plt.title('Виявлення аномалій трафіку')
        plt.xlabel('Час')
        plt.ylabel('Трафік')
        plt.legend()
        plt.show()

        return anomalies

def main():

    predictor = AdvancedCloudTrafficPredictor(num_samples=2000)
    predictor.preprocess_data(test_size=0.2, scaler_type='standard')
```

```
    predictor.train_models()
    predictor.evaluate_models()


    print("\n--- Прогнозування майбутнього трафіку ---")
    future_forecast = predictor.forecast_future_traffic(periods=48)
    print(future_forecast[['timestamp', 'predicted_traffic']].head())

    print("\n--- Виявлення аномалій ---")
    anomalies = predictor.anomaly_detection()
    print(f"Знайдено {len(anomalies)} аномалій")


    predictor.interactive_traffic_prediction()

if __name__ == "__main__":
    main()
```