Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2023 р

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «**МОДЕЛЬ ВИЯВЛЕННЯ НЕСПРАВНОСТЕЙ У ПРИСТРОЯХ IOT**»

Захищено на засіданні
Атестаційної комісії № 42
протокол № __ від __.12.2023 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ **СКОБ Ю. О.**.

Виконав:
студент 2 курсу, групи КУ– 61
Галузь знань: 15 – Автоматизація та приладобудування
Спеціальність: 151 – «Автоматизація та комп'ютерно-інтегровані технології»
ІСАЄВ Владислав Сергійович

Керівник:
к.т.н., доцент
Бикова Тетяна Володимирівна

Рецензент:
в.о. завідувача кафедри теоретичної та прикладної інформатики к.т.н., доц.
Меняйлов Євген Сергійович

Харків – 2023

АНОТАЦІЯ

Пояснювальна записка до магістерської атестаційної роботи складається зі вступу, трьох розділів, висновку, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 102 сторінки із яких 50 сторінок основної частини, 1 таблиця, 13 рисунків, 26 найменувань списку використаних джерел на 3 сторінках і 5 додатків на 42 сторінках.

Розумні міста, розумні домівка та розміні речі поступово стають реальністю. «Розумним» річчю вважається річ, що наповнена датчиками та виконавчими механізмами, які вимірюють і контролюють споживання різних джерел енергії та параметри навколишнього середовища до саморегулювання майже без участі людини. На сьогоднішній день існує багато міст у всьому світі, де технологія IoT сама проводить вимірювання повітря, трафіку, денного світла, споживання газу, електроенергії та води тощо. Оскільки немає простого та надійного способу перевірити, чи ці датчики працюють належним чином, без їх навмисної активації, необхідно запровадити більш зручні процедури контролю, тестування та пошуку похибок.

В ході виконання кваліфікаційної роботи були використані задача регресіїї, алгоритм LSTM (Long Short-Term Memory), середня абсолютна похибка і нейронне навчання. Програмне забезпечення розроблено за допомогою мови Python. Також були використані наступні бібліотеки: scikit-learn, NumPy, Pandas, Tensorflow і інші.

У ході виконання роботи отримано: створена та покращена модель прогнозування даних з сенсорів пристроїв IoT, отримані та покращені метрики програмної моделі.

Головним напрямком використання є покращення технології IoT, довговічності приладів IoT та зниження вартості їх обслуговування.

Ключові слова: МОДЕЛЬ ВИЯВЛЕННЯ ПОХИБОК, МАШИННЕ НАВЧАННЯ, LSTM, ГЛИБОКЕ НАВЧАННЯ, МЕТРИКИ ОЦІНКИ ЯКОСТІ МОДЕЛЕЙ.

ABSTRACT

The explanatory note to the master's thesis consists of an introduction, three sections, a conclusion, a list of used sources and four appendices. The total volume of the work is 102 pages, of which 50 pages are the main part, 1 table, 13 figures, 26 names of the list of used sources on 3 pages, and 5 appendices on 42 pages.

Smart cities, smart homes and exchangeable things are gradually becoming a reality. A "smart" thing is a thing filled with sensors and actuators that measure and control the consumption of various energy sources and environmental parameters to self-regulate with almost no human intervention. Today, there are many cities around the world where IoT technology itself measures air, traffic, daylight, gas, electricity and water consumption, etc. On the other hand, sensors such as gas and smoke detectors, or security surveillance examples, must be constantly checked for accuracy. Since there is no simple and reliable way to verify that these sensors are working properly without deliberately activating them, more convenient monitoring, testing and troubleshooting procedures must be implemented.

In the course of the qualification work, regression problem, LSTM (Long Short-Term Memory) algorithm, average absolute error and neural learning were used. The software is developed using the Python language. The following libraries were also used: scikit-learn, NumPy, Pandas, Tensorflow and others.

In the course of the work, the following results were obtained: a model for forecasting data from sensors of IoT devices was created and improved, metrics of the software model were obtained and improved.

The main direction of use is the improvement of IoT technology, the durability of IoT devices and the reduction of their maintenance costs.

Keywords: ERROR DETECTION MODEL, MACHINE LEARNING, LSTM, DEEP LEARNING, MODEL QUALITY ASSESSMENT METRICS.

ЗМІСТ

ВСТУП

Інтернет речей чи IoT (Internet of Things) — це термін, який з’явився наприкінці 1990-х та початку 2000-х коків для позначення ініціативи, метою якої є глобальне підключення фізичних пристроїв, транспортних засобів та інших предметів, які можуть підключатися до Інтернету, у однорідну комунікаційну платформу, яка використовується для обмінюватися даними між собою. Щороку валовий капітал IoT зростає на мільярди і, як очікується, досягне кількох трильйонів у найближчі кілька років. У міру інтеграції нових технологій деякі концепції та елементи, що використовуються в IoT, виключаються без оцінки. Технологія IoT являється відносно новою концепцією.

Термін IoT сильно змінився з часу його першої згадки. Коли він був представлений, він мав на увазі об’єкти, з’єднані за допомогою радіочастотної ідентифікації RFID(radio-frequency identification), з властивістю їх унікальної ідентифікації та сумісності.

На сьогоднішній день IoT – це глобальна мережева інфраструктура, що постійно змінюється, із можливостями самоналаштування, які використовуються для підключення тих самих «речей», що можуть бути так віртуальними так і фізичними. «Речі» спілкуються через стандартні та сумісні протоколи, мають деякі фізичні атрибути та унікальні ідентифікатори.

Оскільки технологія IoT все ще є новою концепцією, її стандартизація була у пріоритеті багатьох організацій та урядів: наприклад, Американський національний інститут стандартів або та регулятивні зусилля в Європі у 2016 році (що включали Міжнародний Союз електрозв'язку, Міжнародну організацію стандартизації, Європейський комітет з електротехнічної стандартизації).

Розумні міста, розумні домівка та розміні речі поступово стають реальністю. «Розумним» річчю вважається річ, що наповнена датчиками та

виконавчими механізмами, які вимірюють і контролюють споживання різних джерел енергії та параметри навколишнього середовища до саморегулювання майже без участі людини. На сьогоднішній день існує багато міст у всьому світі, де технологія IoT сама проводить вимірювання повітря, трафіку, денного світла, споживання газу, електроенергії та води тощо.

Включаючи більше датчиків у наше повсякденне життя, ми зможемо замінити людське судження спеціальною вимірювальною системою та датчиками, створеними для конкретних сфер. На жаль, з часом ці датчики доведеться замінити або відкалібрувати заново, що в більшості залежить від вартості датчиків. Більшість наших повсякденних вимірювань, таких як температура тіла або кімнатна температура й вологість, не мають бути дуже точними, тому їх, в багатьох випадках, не потрібно постійно калібрувати. З іншого боку, датчики, такі як детектори газу та диму, чи приклади спостереження безпеки, повинні постійно перевірятися на точність. Оскільки немає простого та надійного способу перевірити, чи ці датчики працюють належним чином, без їх навмисної активації, необхідно запровадити більш зручні процедури контролю, тестування та пошуку похибок. Датчики, які здатні самокалібруватися, також будуть схильні до старіння, але їхні статичні та динамічні параметри мають бути набагато точнішими протягом тривалого часу. Очікувана тривалість життя цих пристроїв залежить від точності їх компонентів, що самокалібруються. Таким чином ми можемо продовжити термін служби домашнього чи промислового датчика, не збільшуючи вартість їх обслуговування, але водночас збільшуючи вартість виробництва та точність, коли це необхідно.

Але в деяких випадках така підхід не є раціональним. Деякі датчики, що використовуються в екстремальних умовах, повинні бути відкалібровані навченим персоналом у певних умовах навколишнього середовища. У такому

випадку необхідно взяти ці датчики та відправити їх на повторне калібрування лише в ті лабораторії, які можуть забезпечити такі умови середовища. Цей підхід є занадто дорогим іа створює логістичні проблеми, якщо це робити у великих масштабах.

На даний момент у галузевих застосуваннях, якщо датчик потрібно відкалібрувати вручну, його зазвичай потрібно вилучити з передбачуваного середовища, надіслати в сертифіковану акредитовану калібрувальну лабораторію, повторно відкалібрувати та надіслати назад власнику для відновлення. У цьому процесі є багато очевидних проблем:

• Датчик видалено з оригінальної системи, і система може не працювати без нього

• Датчик повторно калібрується в середовищі, яке може не імітувати середовище, у якому спочатку встановлено датчик

• Процес повторного калібрування сертифікованою лабораторією іноді є дорогим

• Датчик може бути пошкоджений під час транспортування та повернутись до власника несправним

• Весь цей процес займає багато часу, якщо сертифікована лабораторія знаходиться далеко

• Повторне калібрування має виконуватися навченим персоналом, який часто невеликий.

Ряд дослідників поділяють думку про те, що слід відмовитися від традиційних методів періодичних калібрувань.

Було розроблено багато спроб дистанційного калібрування, але жодна з них не була успішною на ринку, а саме з причин, зазначених вище. В ідеалі датчики повинні мати можливість автоматичного калібрування або, принаймні, мати певну резервну систему, таку як алгоритм для зміни даних у випадку, якщо показання виявляться неточними. Інший варіант - надіслати сповіщення

адміністратору, щоб можна було застосувати відповідні дії. Проблемою, що найбільше не враховується, є не інформаційна безпека переданого вимірювання, а його атрибут

невизначеність та її метрологічну надійність. Іншими словами, проблема полягає в низькій ймовірності того, що дані, отримані від датчиків під час інтервалу калібрування, можна вважати надійними. Не всі отримані дані повинні мати однакову вагу в процесі обчислень, які використовуються в промисловості високої точності.

Саме тому остається актуалціллю цієї роботи є дослідження датчиків пристроїв IoT у пошуках похибок за допомогою машинного навчання. Отже, завдання пошука похибок у пристроях IoT актуальною як для даної, так і для багатьох інших прикладних галузей.

Метою цієї роботи є застосування і удосконалення моделей машинного навчання, що прогнозують вихід похибки вимірювання сенсорів IoT за допустимий діапазон.

Об’єкт роботи – процесси виявлення несправностей на основі даних з різних пристроїв IoT.

Предмет роботи – модель виявлення потенційних похибок у пристроях IoT.

Методами дослідження є: методи аналізу моделей машинного навчання та синтезу проектування та модернізації систем автоматизації та комп’ютерно-інтегрованих технологій.

Предмет дослідження – модель виявлення несправностей у пристроях IoT

Завдання для виконання:

- Виконати аналіз наявних датасетів пристроїв IoT за допомогою програмної реалізації
- Перевірити чи зможе програмна модель за допомогою машинного навчання розраховати невідомі майбутні показники сенсорів IoT

- Створити та покрашіти
- Зробити загальний аналіз отриманних результатів

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ ЗАДАЧІ РЕГРЕСІЇ ДЛЯ ПОШУКУ ПОХИБОК У ПРИСТРОЯХ ІОТ

1.1 Вибір задачі машинного навчання

Задача класифікації є однією з основних задач в області машинного навчання, і її основною метою є призначення вхідних даних до одного з певних класів чи категорій. Задача класифікації є формою навчання з учителем, оскільки модель навчається на основі позначених прикладів (навчальних даних), де для кожного вхідного зразка відомий його клас чи категорія.

Основними етапами вирішення задачі класифікації є:

- **Навчання моделі**

Підготовка навчальних даних, що складається з пар вхідних зразків і відповідних міток класів.

Вибір моделі класифікації, такої як логістична регресія, метод опорних векторів (SVM), рішаючі дерева, нейронні мережі тощо.

Навчання моделі на навчальних даних для того, щоб вона могла встановити зв'язки між вхідними ознаками і класами.

- **Тестування моделі**

Використання тестових даних (які модель не бачила під час навчання) для оцінки її продуктивності.

Обчислення метрик класифікації, таких як точність, чутливість, специфічність, F1-мера тощо, для оцінки того, наскільки добре модель класифікує дані.

- **Прогнозування**

Після успішного навчання та тестування модель може бути використана для класифікації нових, раніше не бачених даних.

Прикладами задач класифікації можуть бути:

- **Бінарна класифікація**: Розпізнавання спаму в електронній пошті (два класи: спам чи не спам).
- **Мультикласова класифікація**: Розпізнавання рукописних цифр (де кожна цифра представляє окремий клас).
- **Класифікація на основі тексту**: Класифікація статей за темою (наприклад, наука, технології, спорт і т.д.).

У процесі розвитку та вдосконалення моделей класифікації використовуються різні техніки, включаючи вибір важливих ознак, налаштування параметрів моделі та використання складніших архітектур, таких як глибокі нейронні мережі для розв'язання складних завдань класифікації.

Задача регресії також є однією з основних задач в машинному навчанні, і вона відрізняється від задачі класифікації тим, що цільова змінна (вихід) є неперервною числовою величиною. У задачі регресії модель намагається побудувати функцію, яка передбачатиме або апроксимуватиме значення цільової змінної на основі вхідних ознак.

Основні аспекти задачі регресії включають:

Навчання моделі

Підготовка навчальних даних, що містять пари вхідних зразків і відповідних числових значень цільової змінної.

Вибір моделі регресії, такої як лінійна регресія, регресія методом опорних векторів (SVR), рішаючі дерева, глибокі нейронні мережі тощо.

Навчання моделі з метою знаходження оптимальних ваг та параметрів для відображення залежності між вхідними ознаками і цільовою змінною.

Тестування моделі

Використання тестових даних для оцінки точності та ефективності моделі в передбаченні значень цільової змінної.

Оцінка метрик регресії, таких як середньоквадратична помилка (MSE), середня абсолютна помилка (MAE), коефіцієнт детермінації (R-squared) і т. д.

Прогнозування

Після навчання і тестування модель може використовуватися для прогнозування значень цільової змінної для нових даних.

Приклади задач регресії можуть включати:

Прогнозування ціни акцій. Вхідними ознаками можуть бути різноманітні фінансові показники, а вихідною змінною - ціна акції в певний момент часу.

Прогнозування температури. Вхідні ознаки можуть включати дані про атмосферний тиск, вологість, швидкість вітру і т. д., а цільовою змінною буде температура.

Оцінка вартості нерухомості. Вхідні ознаки можуть включати площу будинку, кількість кімнат, розташування тощо, а вихідною змінною - вартість нерухомості.

У процесі вирішення задач регресії важливо правильно вибирати моделі, налаштовувати їх параметри і аналізувати результати для досягнення точних та надійних прогнозів.

Задача регресії може бути більш придатною для прогнозування даних з сенсорів пристроїв Інтернету речей (IoT) у порівнянні з задачею класифікації з кількох причин:

Неперервні значення

Дані, які зазвичай отримуються з сенсорів, частіше мають неперервний характер, такі як температура, вологість, або рівень освітленості. Задача регресії ефективно працює з прогнозуванням числових значень, тоді як класифікація розглядає задачі, де вихідна змінна належить до обмеженого множини класів.

Більше інформації

Задача регресії може дозволити враховувати більше інформації, оскільки вона працює з конкретними числовими значеннями. У випадку класифікації, модель

вирішує, до якого класу відноситься вхідний зразок, і вся додаткова інформація про числові характеристики може втрачатися.

Нейтралізація проблеми дисбалансу

У випадку класифікації може виникнути проблема дисбалансу класів, особливо якщо один клас значно переважає інші. Задача регресії може бути менш чутливою до цього, оскільки вона не обмежена кількістю класів і може прогнозувати широкий спектр числових значень.

Простота інтерпретації

Результати задачі регресії можуть бути легше інтерпретовані, оскільки вони представляють собою конкретні числа. Це особливо важливо в області IoT, де розуміння значень та їх впливу може бути критичним для прийняття рішень. Зважаючи на цю інформацію, можна вважати що задача регресії подходить більше для програмної реалізації моделі пошука несправностей в приладах IoT.

1.2 Задача регресії

У данній роботі для попередження несправностей у приладах IoT використовується задача регресії з алгоритмом LSTM (Long Short-Term Memory). Задача регресії в машинному навчанні спрямована на передбачення числових значень на основі вхідних даних. Це означає, що модель намагається побудувати функціональну залежність між вхідними ознаками і вихідними значеннями. Найти оптимальну модель для прогнозування числових результатів є ключовим аспектом задачі регресії.

У лінійній регресії модель передбачає лінійну залежність між вхідними ознаками (позначеними як X) та вихідним значенням (позначеним як Y). Формула виглядає наступним чином:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \varepsilon \qquad (1.1)$$

де

- Y є вихідним значенням(змінною, яку намагаються передбачити),
- X_1, X_2,…,X_n є вихідними данним,
- β_0 є ознаками(вхідними змінними),
- β_1, β_2,…,β_n є коефіцієтнати регресії, що показують, наскільки сильно кожна ознака впливає на вихід,
- ε є помилкою, яка представляє собою непоняснену частину моделі.

Мета полягає в знаходженні значень β_1, β_2,…,β_n, які мінімізують суму квадратів різниць між спостережуваними та передбаченими значеннями.

Задачі регресії широко використовуються в різних галузях, і їхнє вивчення дозволяє точно прогнозувати числові значення на основі вхідних ознак. Порівнюючи задачу регресії з задачею класифікації, можна виділити декілька переваг задачі регресії у порівнянні з класифікацією, а саме:

- Нумеричні результати

Однією з основних переваг задачі регресії є здатність передбачати нумеричні значення. Це корисно в ситуаціях, де потрібно прогнозувати реальні числові дані, наприклад, ціни на нерухомість, температуру, часові ряди тощо.

- Континуальний простір відповідей

У задачах регресії відповіді лежать в континуальному просторі, що відрізняється від дискретних класів у задачах класифікації. Це дозволяє моделім бути більш гнучкими у вирішенні широкого спектру завдань.

- Оцінка похибки

Оцінка ефективності регресійних моделей може використовувати різноманітні метрики, такі як середньоквадратична помилка (MSE) чи коефіцієнт детермінації (R^2), що дозволяє краще оцінити точність моделі.

- Наявність Трендів та Залежностей

Регресія дозволяє виявляти тренди та залежності між ознаками, що може бути важливим для розуміння структури даних та прийняття відповідних рішень.

- Оцінка надійності прогнозів

Для задачі регресії важливо визначити, наскільки можна довіряти прогнозам моделі. Це можна робити за допомогою довірчих інтервалів, дисперсії прогнозів та інших метрик, що відображають ступінь невизначеності.

- Аналіз важливості ознак

Вивчення того, які ознаки або змінні найбільше впливають на прогнози моделі. Це може допомогти в розумінні важливості різних аспектів вхідних даних для задачі регресії.

- Робота з викидами

Виявлення та обробка викидів може бути критичним для моделей регресії, оскільки вони можуть значно впливати на точність прогнозів.

- Використання ансамблей

Комбінування декількох моделей регресії в ансамблі може поліпшити загальну ефективність і надійність прогнозів, особливо при використанні методів, таких як випадковий ліс або градієнтний бустінг.

- Техніки зменшення розмірності

У випадку наявності великої кількості ознак може бути корисним використовувати техніки зменшення розмірності, такі як вибір ознак або важливість ознак, для зменшення обсягу обчислень та підвищення ефективності.

- Ітераційна модельна розробка

Задача регресії може розглядатися як ітеративний процес, де можливо варто проводити аналіз та вдосконалення моделі в залежності від результатів тестування та змін в даних.

Ці аспекти допомагають детальніше розуміти та оптимізувати задачі регресії для конкретного застосування, забезпечуючи точні та надійні результати прогнозування.

Задача регресії часто є дуже корисною для прогнозування даних сенсорів пристроїв Інтернету речей (IoT). У контексті IoT, де пристрої можуть збирати величезні обсяги даних, важливо мати ефективні моделі для аналізу та передбачення цих даних. Існують багато причин, чому задача регресії може бути корисною для прогнозування даних сенсорів IoT, а саме:

- Предбачення нумеричних величин

Багато з даних, зібраних сенсорами IoT, представляють собою нумеричні значення, такі як температура, вологість, тиск, освітленість тощо. Задача регресії ідеально підходить для прогнозування цих неперервних числових даних.

- Адаптація до змін

Сенсори можуть реагувати на різні зовнішні впливи, і їхні вимірювання можуть змінюватися з часом. Задача регресії дозволяє моделям адаптуватися до змін у вхідних даних та вчитися новим паттернам.

- Оцінка точності прогнозів

Використання метрик регресії, таких як середньоквадратична помилка (MSE) чи коефіцієнт детермінації (R^2), дозволяє об'єктивно оцінити точність моделі та визначити, наскільки вона добре прогнозує реальні величини.

- Моделювання складних залежностей

Задача регресії дозволяє моделям виявляти складні залежності та взаємодії між різними сенсорами, що є важливим для повного розуміння вихідних даних.

- Прогнозування заходів та управління ресурсами

Результати задачі регресії можуть бути використані для прогнозування подій або станів, що дозволяє вчасно реагувати на зміни, зменшуючи ризики та оптимізуючи управління ресурсами.

- Використання алгоритмів машинного навчання

У задачі регресії для прогнозування даних сенсорів IoT застосовуються різні алгоритми машинного навчання. Наприклад, лінійна регресія, метод опорних векторів (SVR), рішення дерева, ансамблеві методи (наприклад, випадковий ліс) та нейронні мережі. Кожен з цих підходів має свої переваги та обмеження, і вибір конкретного алгоритму може залежати від конкретних особливостей задачі та характеристик даних.

- Обробка даних

Ефективна обробка даних є ключовою для успішної задачі регресії. Вона включає в себе видалення аномалій, обробку відсутніх даних, нормалізацію та інші техніки, які покращують якість вхідних даних для навчання моделі.

- Автоматизація та виробниче впровадження:

Після створення та навчання моделі регресії важливо враховувати аспекти автоматизації та можливості виробничого впровадження. Це включає в себе оптимізацію моделі для швидкості прогнозування, масштабування для роботи з великими обсягами даних та інші практичні аспекти використання моделі в реальному часі.

- Захист та приватність даних:

У зв'язку із збільшенням обсягу даних, які збираються сенсорами, важливо приділяти увагу питанням захисту і приватності даних. Застосування алгоритмів та методів, які дозволяють прогнозувати дані, не розголошуючи особистої інформації, може бути ключовим аспектом в розробці систем IoT.

Загалом, задача регресії є потужним інструментом для вирішення проблем прогнозування даних сенсорів в Інтернеті речей, де важливо мати точні та адаптивні моделі.

1.3 Алгоритм LSTM

Існує дуже багато алгоритмів для задачі регресії, і їх вибір залежить від конкретних властивостей даних та характеристик задачі. Найпоширенішими з популярних алгоритмів для вирішення задачі регресії являються алгоритми лінійної регресії (Linear Regression), дерева рішень (Decision Trees), випадковий ліс (Random Forest), градієнтний бустінг (Gradient Boosting), лінійна регресія з регуляризацією (Lasso, Ridge, Elastic Net), метод опорних векторів (Support Vector Machines - SVM), К-найближчих сусідів (K-Nearest Neighbors - KNN) та глибокі нейронні мережі.

Лінійна регресія є одним з найпростіших і широко використовуваних методів в статистиці та машинному навчанні для вирішення задачі регресії. Цей метод використовує лінійну функцію для апроксимації залежності між незалежними ознаками (вхідними змінними) та залежною змінною (вихідним значенням). У лінійній регресії припускається, що залежна змінна може бути лінійною комбінацією вхідних змінних. Лінійна регресія є ефективним інструментом для простих задач, де залежність між ознаками та вихідним значенням може бути апроксимована лінійно. Якщо залежність більше складна, можуть використовуватися більш складні моделі, такі як поліноміальна регресія, дерева рішень, нейронні мережі та інші.

Дерево рішень у задачі регресії - це модель машинного навчання, яка використовується для прогнозування числового значення (кількісної змінної). В основі дерева рішень лежить структура дерева, де кожен вузол представляє

рішення або тестову умову, а кожний листок дерева містить прогнозоване числове значення. Дерева рішень в регресії добре підходять для виявлення складних залежностей та нелінійних взаємозв'язків в даних. Вони є інтерпретованими та легко зрозумілими, але можуть схильні до перенавчання, особливо якщо дерево дуже глибоке та складне.

Випадковий ліс у задачі регресії - це ансамбль моделей, який базується на деревах рішень та використовується для прогнозування числових значень (регресії). Випадковий ліс об'єднує кілька дерев рішень, кожне з яких обчислює прогноз, і вирішує завдання за допомогою голосування або середнього значення. Хоча випадкові ліси і є потужним і ефективним методом в багатьох випадках, вони також мають свої недоліки. Деякі з найзначущих недоліків є велика обчислювальна складність та споживання ресурсів, неефективність для розріджених даних та тривалість тренування.

Градієнтний бустінг - це ансамбльний метод машинного навчання, який використовує послідовне навчання слабких моделей з метою покращення загальної ефективності моделі. Головна ідея полягає в тому, щоб навчати нові моделі для корекції помилок, які робить попередня модель або ансамбль моделей. Він є потужним методом, який часто виявляється дуже ефективним, але також вимагає уваги до гіперпараметрів та може бути чутливим до перенавчання.

Лінійна регресія з регуляризацією - це модифікація класичної лінійної регресії, яка включає додаткові терміни у функцію втрати з метою обмежити величину коефіцієнтів регресії. Це робиться для запобігання перенавчанню та поліпшення стійкості моделі. Але у той же час лінійна регресія з регуляризацією неефективна при великій кількості ознак, може бути чутливою до викидів у даних. Одиночний великий викид може значно вплинути на коефіцієнти регресії і, відповідно, на якість моделі. Також з основних обмежень лінійної регресії полягає в тому, що вона покладається на лінійні залежності між ознаками та

цільовою змінною. Якщо відносини в даних складніші, лінійна регресія може бути менш ефективною.

Основна ідея методу опорних векторів у задачі регресії полягає в тому, щоб знаходити гиперплощину, яка максимально відокремлює точки даних у просторі функціональних ознак (вхідних змінних) і в той же час мінімізує помилки прогнозу для точок, які потрапляють в допустиму область. Метод опорних векторівчасто використовуються в задачах класифікації, особливо коли дані мають складну структуру та можливо не є лінійно роздільними. Однак вони також можуть бути застосовані до задач регресії та інших завдань машинного навчання.

Алгоритм К-найближчих сусідів є простим та ефективним методом для класифікації та регресії в машинному навчанні. Основна ідея полягає в тому, щоб призначити клас (у випадку класифікації) або визначити значення (у випадку регресії) нового прикладу, виходячи з його "найближчих сусідів" у просторі ознак. Алгоритм К-найближчих сусідів має кілька недоліків, особливо в контексті задачі регресії: чутливий до шуму, великі вимоги до обчислювальних ресурсів, неефективність при великій кількості ознак.

У наявній задачі регресії для пошуку похибок у пристроях IoT використовується алгоритм LSTM (Long Short-Term Memory). LSTM є особливим типом рекурентних нейронних мереж, спроектованим для роботи з послідовнісними даними, такими як часові ряди. LSTM є алгоритмом, який можна використовувати як у задачі класифікації, так і у задачі регресії в залежності від конкретної задачі.

LSTM (Long Short-Term Memory) є типом рекурентних нейронних мереж, які здатні моделювати складні залежності в часових послідовностях. У задачі регресії, де потрібно прогнозувати числові значення, можна використовувати LSTM. Загальна структура формули для LSTM у задачі регресії виглядає так:

$$y_t = f(W_{hy} \cdot h_t + b_{hy}) \quad (1.2)$$

де

- y_t є прогнозоване значення на часі,
- h_t - стан прихований LSTM на часі t,
- W_{hy} - матриця ваг LSTM для зв'язку між станом прихованим та прогнозованим значенням,
- b_{hy} - зміщення для прогнозованого значення,
- f - функція активації.

У випадку регресії, LSTM може бути використаний для прогнозування числових значень в майбутньому, особливо в тих випадках, коли дані мають часовий компонент (часові ряди). Наприклад, прогнозування цін акцій, температур, обсягів продажів і т. д.

Основна різниця між використанням LSTM у задачі класифікації та регресії полягає у форматі виходу моделі. У задачі класифікації, LSTM може виводити ймовірності або класи категорій. У задачі регресії, LSTM виводить числові значення.

В порівнянні з іншими алгоритмами, LSTM має деякі унікальні переваги, зокрема для завдань, пов'язаних з обробкою послідовностей та часових рядів. Ось деякі з його основних переваг:

- Обробка довгострокових залежностей

LSTM спеціально створений для вирішення проблем довгострокових залежностей в даних. Це дозволяє ефективно моделювати та запам'ятовувати інформацію з попередніх часових кроків, що є важливим для багатьох завдань, таких як прогнозування часових рядів.

- Здатність розпізнавати патерни

LSTM в здатний автоматично вивчати та розпізнавати складні патерни в послідовностях даних, такі як тенденції, цикли та інші нелінійні залежності, що робить його ефективним для задач прогнозування.

- Обробка великого обсягу даних

LSTM може ефективно працювати з великими обсягами даних, включаючи великі часові ряди. Це дозволяє використовувати модель для прогнозування в умовах великої кількості даних.

- Можливість моделювання послідовностей різної довжини

LSTM може обробляти послідовності різної довжини, що робить його більш гнучким у порівнянні з традиційними методами, які можуть бути обмеженими фіксованою довжиною вводу.

- Захист від зникнення та вибуху градієнту

Архітектура LSTM включає механізми для запобігання проблемам, таким як зникнення та вибух градієнту, що можуть виникати при тренуванні глибоких нейронних мереж.

- Можливість роботи з одноразовими подіями (Events)

LSTM може ефективно обробляти вхідні дані, де важливою є не тільки поточна інформація, але й події, які сталися в минулому, що робить його корисним для задач, пов'язаних з подіями.

Висновки до розділу 1

В розділі була розглянута та проаналізована задача регресії та задача класифікації, на предмет того, чому саме задача регресіїзадача регресії підходить для моделі пошука несправностей у пристрояї IoT. Також розглядаються різні алгоритми задачі регресії та пояснюється чому саме алгоритм LSTM підходить для даної моделі.

РОЗДІЛ 2
ПРОГРАМНА РЕАЛІЗАЦІЯ МОДЕЛІ ПОШУКУ ПОХИБОК

2.1 Вхідні дані

Для реалізації моделі пошука похибок у пристроях IoT Python - це високорівнева, інтерпретована мова програмування з широким застосуванням у різних галузях, включаючи розробку веб-додатків, аналіз даних, наукові обчислення та машинне навчання. Ось кілька ключових рис Python, які роблять його виключним серед інших мов програмування:

- Простота вивчення та читання: Python має простий та чіткий синтаксис, який полегшує вивчення програмування, а також зрозуміння та підтримку коду. Це робить Python доступним для широкого кола користувачів, включаючи науковців, інженерів та аналітиків.
- Велика екосистема бібліотек: У світі машинного навчання Python визначається широкою екосистемою бібліотек, таких як NumPy, pandas, Matplotlib та, зокрема, TensorFlow і PyTorch для глибокого навчання. Ці бібліотеки надають потужні інструменти для обробки даних, візуалізації та розробки моделей машинного навчання.
- Мультипарадигмальність: Python підтримує різні парадигми програмування, такі як процедурне програмування, об'єктно-орієнтоване програмування та функціональне програмування. Це надає розробникам гнучкість у виборі підходів при розробці програм.
- Спільнота та підтримка: Python має велику та активну спільноту розробників, що призводить до швидкої реакції на нові технологічні виклики, виправлення помилок та розробки нових функцій. Це забезпечує високу якість та стабільність мови.

- Відкритий код та безкоштовність: Python - відкрита мова програмування, що дозволяє вільно використовувати, модифікувати та розповсюджувати її. Це забезпечує широкий розповсюдження та високу доступність.
- У сукупності ці риси роблять Python однією з найбільш популярних мов для машинного навчання, сприяючи широкому прийняттю та розвитку нових методів і технік у галузі штучного інтелекту.

На початку програми проводиться імпортування таких бібліотек: timedelta, numpy, tensorflow, StandardScaler, mean_absolute_error, pyplot та List.

- «timedelta» є об'єктом в Python, який представляє різницю між двома датами або часовими мітками. Цей об'єкт дозволяє виконувати арифметичні операції з часом, такі як додавання або віднімання певної кількості днів, годин, хвилин чи інших одиниць виміру часу. У кваліфікаційній роботі timedelta використовується для того щоб відняти частину часового інтервалу від основного датасету для утворення тренирувального та валідаціоного датафреймів по часовому ряду
- NumPy (Numerical Python) - це бібліотека для мови програмування Python, яка надає підтримку для роботи з великими, багатовимірними масивами та матрицями, а також високорівневі математичні функції для роботи з цими структурами даних. NumPy є однією з основних бібліотек для наукових обчислень та обробки даних в Python. NumPy в кваліфікаційній роботі використовується для роботи з масивами та матрицями, а також для певних операцій над числовими даними.
- TensorFlow - це відкрита бібліотека для машинного навчання та глибокого навчання, розроблена компанією Google. Вона надає інструменти для створення та тренування штучних нейронних мереж, які використовуються для вирішення різних завдань, таких як класифікація, регресія, кластеризація та інші.

У кваліфікаційній роботі TensorFlow використовується для створення та тренування моделі глибокого навчання (LSTM) для задачі регресії.

- StandardScaler є частиною бібліотеки scikit-learn і представляє собою метод стандартизації даних. Стандартизація - це процес приведення даних до стандартного нормального розподілу, де середнє значення дорівнює 0, а стандартне відхилення - 1. Основна ідея стандартизації полягає в тому, щоб забезпечити те, що кожен атрибут (ознака) у наборі даних має середнє значення, яке дорівнює 0, і стандартне відхилення, яке дорівнює 1. Це дозволяє легше порівнювати та інтерпретувати значення різних ознак. Конкретно StandardScaler використовується для виконання такої стандартизації. Він працює на основі знаходження середнього значення та стандартного відхилення для кожного атрибуту в наборі даних і застосовує зміни до кожного значення, щоб стандартизувати його.
- mean_absolute_error (середня абсолютна похибка) - це метрика оцінки точності регресійних моделей. Вона вимірює середню абсолютну різницю між прогнозованими та фактичними значеннями. У кваліфікаційній роботі ця метрика використовується для оцінки ефективності моделі LSTM під час тренування та валідації.
- pyplot - це модуль бібліотеки matplotlib, який надає функції для створення графіків та візуалізації даних. У кваліфікаційній роботі pyplot використовується для побудови графіків для аналізу тренування моделі.
- List в наведеному коді використовується для аннотації типу даних. Аннотації типу дозволяють вказати, який тип даних повинен бути використаний для певної змінної чи аргумента функції.

Для перевірки роботи моделі ми використовуємо декілька вхідних даних - датасетів сенсорів датчиків пристроїв IoT, а саме датасети data1.csv що містить дані вимірювання сенсорів анемометра(вітроміра) на швидкість вітру та Fareng.csv, що містить данні вимірювання кімнатної температури з сенсорів

термостата у градусах Фаренгейту. Частковий вигляд датасетів «Fareng.csv» та «data1.csv» зображено на рисунках 2.1 та 2.2 відносно.

DATE	Fareng_temp
1/1/1985	72.5052
2/1/1985	70.672
3/1/1985	62.4502
4/1/1985	57.4714
5/1/1985	55.3151
6/1/1985	58.0904
7/1/1985	62.6202
8/1/1985	63.2485
9/1/1985	60.5846
10/1/1985	56.3154
11/1/1985	58.0005
12/1/1985	68.7145
1/1/1986	73.3057
2/1/1986	67.9869
3/1/1986	62.2221
4/1/1986	57.0329
5/1/1986	55.8137
6/1/1986	59.9005
7/1/1986	65.7655
8/1/1986	64.4816
9/1/1986	61.0005
10/1/1986	57.5322
11/1/1986	59.3417
12/1/1986	68.1354
1/1/1987	73.8152
2/1/1987	70.062
3/1/1987	65.61
4/1/1987	60.1586
5/1/1987	58.8734
6/1/1987	63.8918
7/1/1987	68.8694
8/1/1987	70.0669
9/1/1987	64.1151
10/1/1987	60.3789
11/1/1987	62.4643
12/1/1987	70.5777
1/1/1988	79.8703
2/1/1988	76.1622
3/1/1988	70.2928
4/1/1988	63.2384

Рис 2.1 Частковий вигляд датасету «Fareng.csv»

Date	Speed
1977-01-03	1.1584
1977-01-04	1.1746
1977-01-05	1.1584
1977-01-06	1.1499
1977-01-07	1.1584
1977-01-10	1.1499
1977-01-11	1.1584
1977-01-12	1.1417
1977-01-13	1.1417
1977-01-14	1.1584
1977-01-17	1.1417
1977-01-18	1.1334
1977-01-19	1.1172
1977-01-20	1.1087
1977-01-21	1.1008
1977-01-24	1.0924
1977-01-25	1.0924
1977-01-26	1.0845
1977-01-27	1.0597
1977-01-28	1.0431
1977-01-31	1.0514
1977-02-01	1.0514
1977-02-02	1.0597
1977-02-03	1.0597
1977-02-04	1.0597
1977-02-07	1.0597
1977-02-08	1.0675
1977-02-09	1.0675
1977-02-10	1.0597
1977-02-11	1.0675
1977-02-14	1.0597
1977-02-15	1.0597
1977-02-16	1.0675
1977-02-17	1.0762
1977-02-18	1.0675
1977-02-22	1.0762
1977-02-23	1.0762
1977-02-24	1.0675
1977-02-25	1.0514
1977-02-28	1.0762

Рис 2.2 Частковий вигляд датасету «data1.csv»

Перед початком роботи ми проводимо попередній аналіз вхідних даних. Попередня обробка даних — це підготовка необроблених даних для моделі машинного навчання та процес вивчення та визначення патернів, властивостей та характеристик даних для отримання інсайтів та прийняття обґрунтованих

рішень. У програмному коді кваліфікаційної роботи аналіз даних включає кілька етапів:

- Читання та завантаження даних

Ми завантажуемо дані з CSV-файлу за допомогою бібліотеки Pandas. Дані виглядають, що містять стовпець "DATE" з датами та стовпець "Fareng_temp" з температурними значеннями.

- Виведення перших 5 рядків даних:

Це дозволяє швидко переглянути перші 5 рядків даних для отримання загального уявлення про їх структуру та зміст.

- Статистичний аналіз:

Виведення мінімальної та максимальної дати в наборі даних, щоб визначити період, який охоплює дані.

- Фільтрація даних за останні N років:

Створення нового фрейму даних, який містить лише дані за останні N років для наступного тренирування валідаційного та тренувального датафреймів.

- Виведення інформації про розмір фільтрованого набору даних:

Виведення розміру (кількість рядків та стовпців) фільтрованого фрейму даних.

- Виведення мінімальної та максимальної дати в новому наборі даних:

Перевірка мінімальної та максимальної дат в новому наборі даних для підтвердження правильності фільтрації.

Ці етапи надають можливість вивчити основні характеристики та властивості даних перед подальшою обробкою та моделюванням. Тому попередня обробка даних є найважливішим кроком під час створення моделі машинного навчання.

2.2 Розробка моделі

Код містить кілька етапів підготовки даних для подальшого використання в моделі машинного навчання. Вони включають у себе:

- Розділення даних на тренувальний та валідаційний набори.
- Визначення розміру тренувального набору як 80% від загального розміру даних.

Створення окремих фреймів даних для тренувального та валідаційного наборів. Тренувальний (навчальний) набір та валідаційний (перевірочний) набір - це два основних набори даних, які використовуються в процесі машинного навчання для навчання та оцінки моделей. Розділення даних на ці два набори є важливою практикою для визначення ефективності моделі та її здатності генералізувати знання на нові дані. Цілю тренувальний набора є навчання моделі на даних и вивчення патернів та ваг параметрів моделі. Він характеризується великою кількістьтю прикладів та використовується для тренування та оптимізації моделі. Завдяки ньому модель "бачить" ці дані під час навчання та намагається здобути з них корисні знання.

Цілю валідаційного набіра є оцінка ефективності моделі під час навчання та виявлення перенавчання (overfitting) та налаштування параметрів моделі. Валідаційних набір характерихується тим, що він використовується для внутрішнього тестування під час навчання, але не використовується для тренування, але дозволяє визначити, наскільки добре модель генералізує знання на нові дані. Також валідаційний набір допомагає виявити перенавчання, якщо модель "перезанадто" адаптована до тренувальних даних та втрачає здатність генералізувати.

Розділення на тренувальний та валідаційний набори є важливим кроком у розробці моделей машинного навчання, оскільки воно дозволяє оцінювати точність моделі на даних, які вона не бачила під час навчання. Це допомагає

уникнути перенавчання та забезпечує кращу загальну ефективність моделі на нових наборах даних.

Нормалізація даних за допомогою StandardScaler.

Створення екземпляру StandardScaler та навчання його на тренувальному наборі температурних значень. Це допомагає нормалізувати дані та забезпечити стандартний розподіл.

Нормалізація даних за допомогою StandardScaler - це процес перетворення числових даних так, щоб вони мали середнє значення (середнє арифметичне) 0 і стандартне відхилення 1. Цей метод нормалізації важливий для багатьох алгоритмів машинного навчання, які чутливі до масштабування даних.

Стандартний спосіб нормалізації полягає в застосуванні Z-перетворення (Z-score transformation) за допомогою StandardScaler, який використовується для стандартизації даних. Процедура стандартизації призводить до того, що значення кожного ознаки у наборі даних мають середнє значення 0 та стандартне відхилення 1. Формула Z-перетворення для одного значення x полягає у відніманні середнього значення (μ) та поділі на стандартне відхилення (σ):

$$z = \frac{x-\mu}{\sigma} \tag{2.2}$$

де:

x - оригінальне значення ознаки,

μ - середнє значення ознаки,

σ - стандартне відхилення ознаки.

Створення функції make_dataset для побудови часових серій та батчів.

Створення функції, яка приймає на вхід фрейм даних та параметри для побудови часових серій та батчів. Використовується tf.keras.preprocessing.timeseries_dataset_from_array для створення датасету для моделі.

Цілю функції є побудова датасету для тренування моделі на основі часових серій та використання векторів ознак та міток для навчання та оцінювання моделі машинного навчання.

Параметри функції:

- df: Фрейм даних, який містить часові ряди.
- window_size: Розмір вікна для побудови часових серій.
- batch_size: Розмір батча для навчання моделі.
- use_scaler: Чи використовувати нормалізацію даних (за замовчуванням True).
- shuffle: Чи перемішувати дані у батчах (за замовчуванням True).

Процесом роботи функції являються вибір відповідних ознак та підготовка даних для введення в модель, нормалізація даних, якщо вказано параметр use_scaler та використання tf.keras.preprocessing.timeseries_dataset_from_array для побудови датасету часових серій та батчів.У випадку з часовими серіями кожен батч представляє собою вікно заданого розміру, що зміщується по часу.

Функція використовується для створення тренувальних та валідаційних датасетів, які подаються моделі для навчання та оцінки.

Ця функція являється важливою, оскільки спрощує підготовку даних для моделі, дозволяючи зручно визначити розмір вікна та батча.

Відмінності використання:

Параметр use_scaler надає можливість включення або виключення нормалізації даних в датасеті.

Параметр shuffle визначає, чи потрібно перемішувати дані в батчах, що може бути важливим для уникнення порушень кореляцій в даних.

- Створення прикладного датасету для тренування:

Створення прикладного датасету з використанням функції make_dataset. Виведення прикладу функцій та міток для перевірки.

- Використання прикладного датасету для перевірки розмірності функцій та міток:

Виведення розмірності функцій та міток у прикладному датасеті. Це допомагає переконатися, що дані підходять для введення в модель.

Ці етапи підготовки даних є важливими перед використанням їх для навчання моделі машинного навчання. Нормалізація та побудова часових серій допомагають моделі ефективно вивчати патерни в даних та робити прогнози.

Далі створюється та навчається модель машинного навчання для прогнозування температурних значень на основі часових рядів. Цей процес складається з:

- Створення моделі

Використовується бібліотека TensorFlow та її високорівневий API Keras для створення моделі. Модель є послідовною моделлю (Sequential), що дозволяє лінійно стекувати шари. «tf.keras.models.Sequential» використовується для створення послідовної моделі, в якій шари додаються послідовно один за одним. Далі використовується перший шар LSTM (Long Short-Term Memory) з 32 або 64 нейронами, в залежності від конкретного випадку. Dense шари для прогнозування значень. Цей шар використовується для прогнозування одного числового значення - даних з сенсорів IoT, оскільки вихід моделі повинен бути одновимірним числовим значенням.

Ця частина програмної реалізації визначає архітектуру моделі, яка має один LSTM шар та один Dense шар. LSTM використовується для аналізу часових залежностей у вхідних даних, а Dense шар використовується для прогнозування значення температури на основі аналізу.

Модель може бути налаштована для використання різної кількості нейронів у LSTM, а також можуть бути додані додаткові шари чи змінені параметри, відповідно до вимог конкретної задачі.

Компіляція моделі

Компіляція моделі в нейронних мережах — це процес визначення параметрів, необхідних для тренування моделі. Під час компіляції визначаються такі аспекти, як оптимізатор (алгоритм оптимізації), функція втрат (loss function) та метрики (metrics), які будуть використовуватися для оцінки ефективності моделі під час тренування та валідації.

У програмної реалізації цей процес відбувається за допомогою методу compile об'єкта моделі у функції compile_and_fit.

Під час компіляції визначаються декілька параметрів:

- ***Втрата (Loss)***

У цьому випадку використовується середньоквадратична функція втрати (tf.losses.MeanSquaredError()). Ця функція вимірює середнє значення квадратів різниці між прогнозованими та фактичними значеннями. Оптимізатор намагатиметься мінімізувати цю втрату під час тренування.

- ***Оптимізатор (Optimizer)***

Для оптимізації використовується алгоритм Adam (tf.optimizers.Adam()). Adam є ефективним алгоритмом оптимізації, який адаптує швидкість навчання для кожного параметра моделі.

- ***Метрики (Metrics)***

Визначається метрика для оцінки ефективності моделі. У цьому випадку використовується середньо-абсолютна похибка (tf.metrics.MeanAbsoluteError()), яка вимірює середню абсолютну різницю між прогнозованими та фактичними значеннями.

Після компіляції модель готова до тренування. Процес тренування розпочинається викликом методу fit, який використовує навчальні дані для адаптації параметрів моделі до зазначених критеріїв.

- ***Тренування моделі:***

Тренування моделі у коді відбувається за допомогою методу fit з бібліотеки TensorFlow Keras. Цей метод використовується для передачі навчальних даних моделі та оптимізації параметрів моделі на основі цих даних.

Основні параметри що приведені у цьому методу:

train_ds:

Це навчальний набір даних, який передається моделі для тренування. Навчальний набір підготовлюється за допомогою функції make_dataset та містить вікна даних та відповідні значення цільової змінної.

epochs=num_epochs:

Параметр, який визначає кількість епох тренування. Епоха - це одна ітерація над усім навчальним набором даних. Зазначений параметр num_epochs визначає, скільки разів весь навчальний набір буде поданий моделі для тренування.

validation_data=val_ds:

Це валідаційний набір даних, який використовується для оцінки ефективності моделі під час тренування. Валідація відбувається після кожної епохи, і результати виводяться в історію тренування.

verbose=0:

Це параметр, який визначає рівень деталізації виводу під час тренування. У даному випадку verbose=0 означає, що вивід буде вимкнено, інші значення (наприклад, verbose=1) можуть виводити більше інформації під час тренування.

Після завершення тренування метод fit повертає об'єкт історії history, який містить метрики та значення втрат на кожній епохі. Це може використовуватися для подальшого аналізу результатів тренування та візуалізації.

- ***Оцінка моделі:***

Оцінка моделі відбувається за допомогою методу evaluate, який використовується для визначення ефективності моделі на навчальних та валідаційних даних. Виклики evaluate проводять оцінку моделі на вказаних

наборах даних (у цьому випадку, навчальному та валідаційному наборах) і виводять значення втрат та метрик, які були визначені під час компіляції моделі.

Ці оцінки можуть допомогти визначити, чи модель навчилася на навчальних даних та чи можна очікувати, що вона буде ефективною на нових, раніше не бачених даних. Оцінка також включає значення втрат та метрики, такі як середній абсолют.

- ***Побудова графіків середньої абсолютної похибки (MAE) для тренування та валідації під час епох.***

У програмній моделі plt.plot(history.history['mean_absolute_error']) та plt.plot(history.history['val_mean_absolute_error']) використовуються для побудови графіків залежності середньої абсолютної похибки (MAE) від кількості епох для тренування та валідації моделі.

plt.plot(history.history['mean_absolute_error']):

Ця лінія коду відповідає за побудову графіка MAE на тренуванні. history.history['mean_absolute_error'] містить значення MAE для кожної епохи під час тренування моделі. Графік дозволяє відстежувати, як MAE змінюється від епохи до епохи на тренувальному наборі даних.

plt.plot(history.history['val_mean_absolute_error']):

Ця лінія коду відповідає за побудову графіка MAE на валідації. history.history['val_mean_absolute_error'] містить значення MAE для кожної епохи під час оцінки моделі на валідаційному наборі даних. Графік дозволяє відстежувати, як MAE змінюється на валідації, що може допомогти визначити, чи модель перенавчалася (overfitting) на тренувальних даних або якість її прогнозів на нових даних.

2.3 Вдосконалення моделі

Вдосконалення моделі - це процес налаштування та покращення архітектури нейронної мережі для досягнення кращих результатів на конкретній

задачі. Цей процес включає в себе декілька кроків, які можна розглядати як "етапи вдосконалення моделі":

Етап 1: Базова модель

Починається з визначення базової архітектури моделі. У вказаному коді це є модель з одним шаром LSTM та одним Dense шаром.

Етап 2: Компіляція та тренування базової моделі

Модель компілюється з використанням певних параметрів, таких як оптимізатор, функція втрат та метрики.

Після цього модель тренується на навчальних даних.

Етап 3: Оцінка базової моделі

Проводиться оцінка ефективності базової моделі на навчальних та валідаційних даних.

Етап 4: Вдосконалення моделі

На основі аналізу результатів базової моделі вносяться зміни в архітектуру або параметри моделі з метою поліпшення її ефективності.

Наприклад, в коді проводяться декілька етапів вдосконалення, де змінюється кількість шарів LSTM та Dense шарів, додається dropout, тощо.

Цей процес ітеративно повторюється з метою досягнення найкращої можливої ефективності моделі на конкретній задачі прогнозування данних з пристроїв IoT.

Висновок до розділу 2

В розділі було описано уявлення вхідних даних та їх обробка. Основну увагу приділено моментам, які відбуваються у коді під час створення моделі.

Також було розказано про вдосконалення моделі, які конкретні дії здійснюються при покращенні моделі на конкретній задачі прогнозування данних з пристроїв IoT.

РОЗДІЛ 3.
АНАЛІЗ ОТРИМАННИХ РЕЗУЛЬТАТІВ

3.1 Отриманні результати вимірювання

Результати для датасету Fareng.csv, данних з сенсорів термостату IoT у градусах Фаренгейта

На рис. 3.1 зображена візуалізація процесса тренувального набору моделі, що використовує у якості вхідних даних датасет «Fareng.csv», і яка прошла 100 епох.

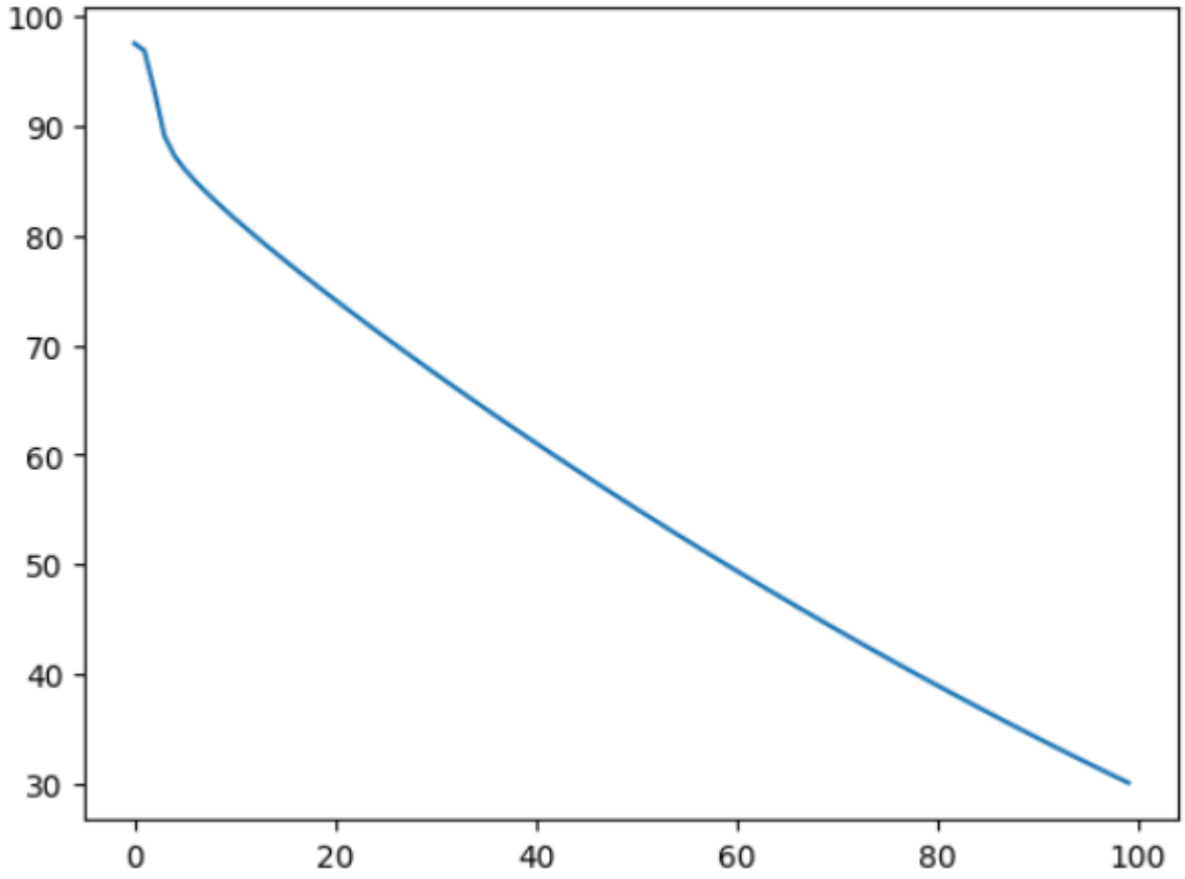

Рис 3.1. Mean_absolute_error1

На рис. 3. 2 зображена візуалізація процесса валідаційного набору моделі, що використовує у якості вхідних даних датасет «Fareng.csv», і яка прошла 100 епох.

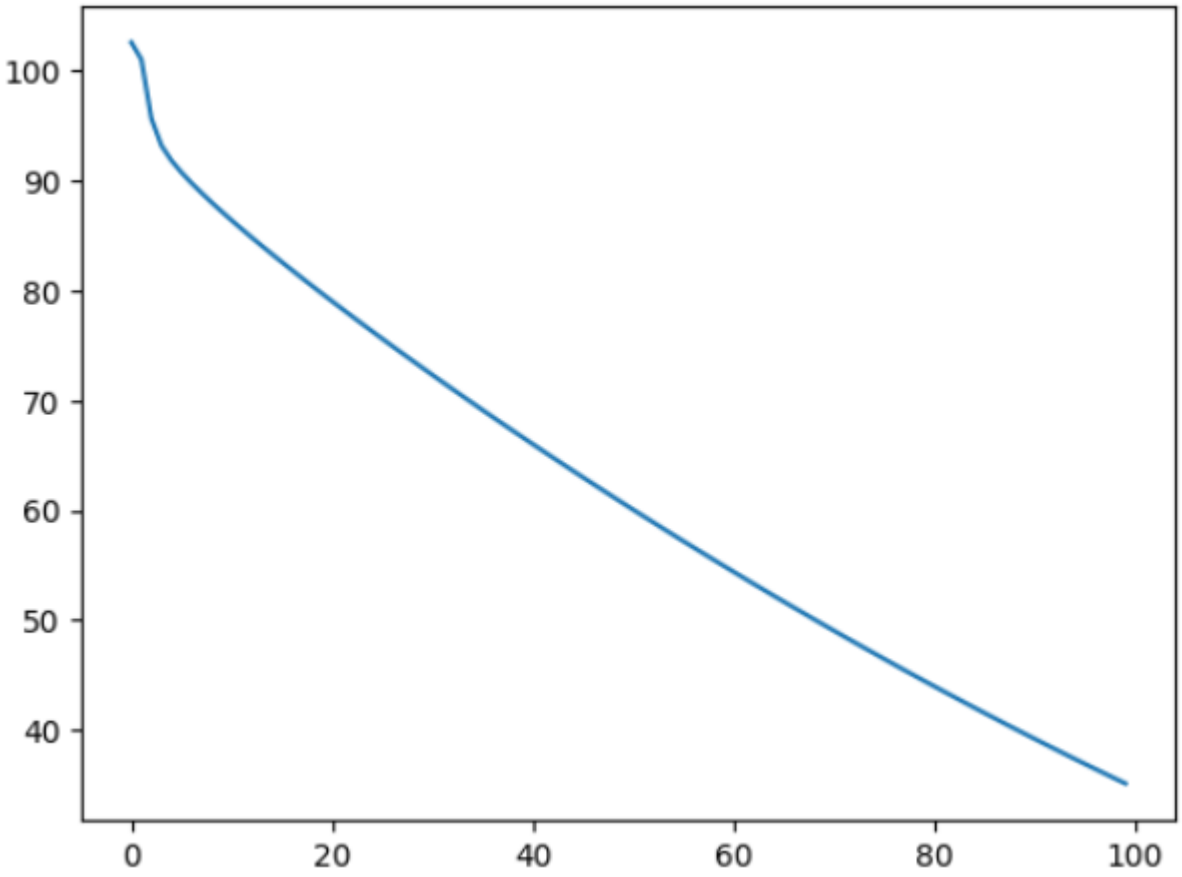

Рис 3. 2. Val_mean_absolute_error1

На рис. 3.3 зображена візуалізація процесса тренувального набору моделі, що використовує у якості вхідних даних датасет «Fareng.csv», і яка прошла 500 епох.

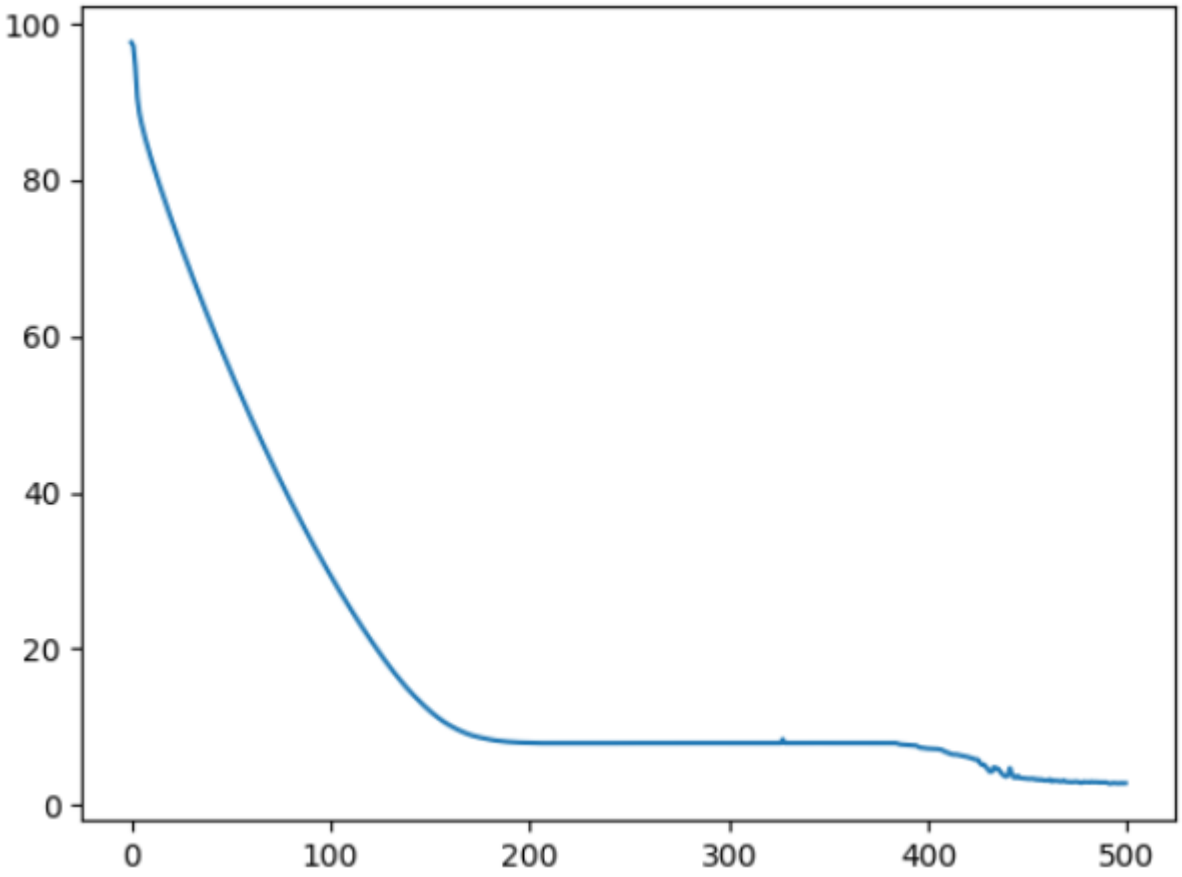

Рис 3.3. Mean_absolute_error2

На рис. 3.4 зображена візуалізація процесса валідаційного набору моделі, що використовує у якості вхідних даних датасет «Fareng.csv», і яка прошла 500 епох.

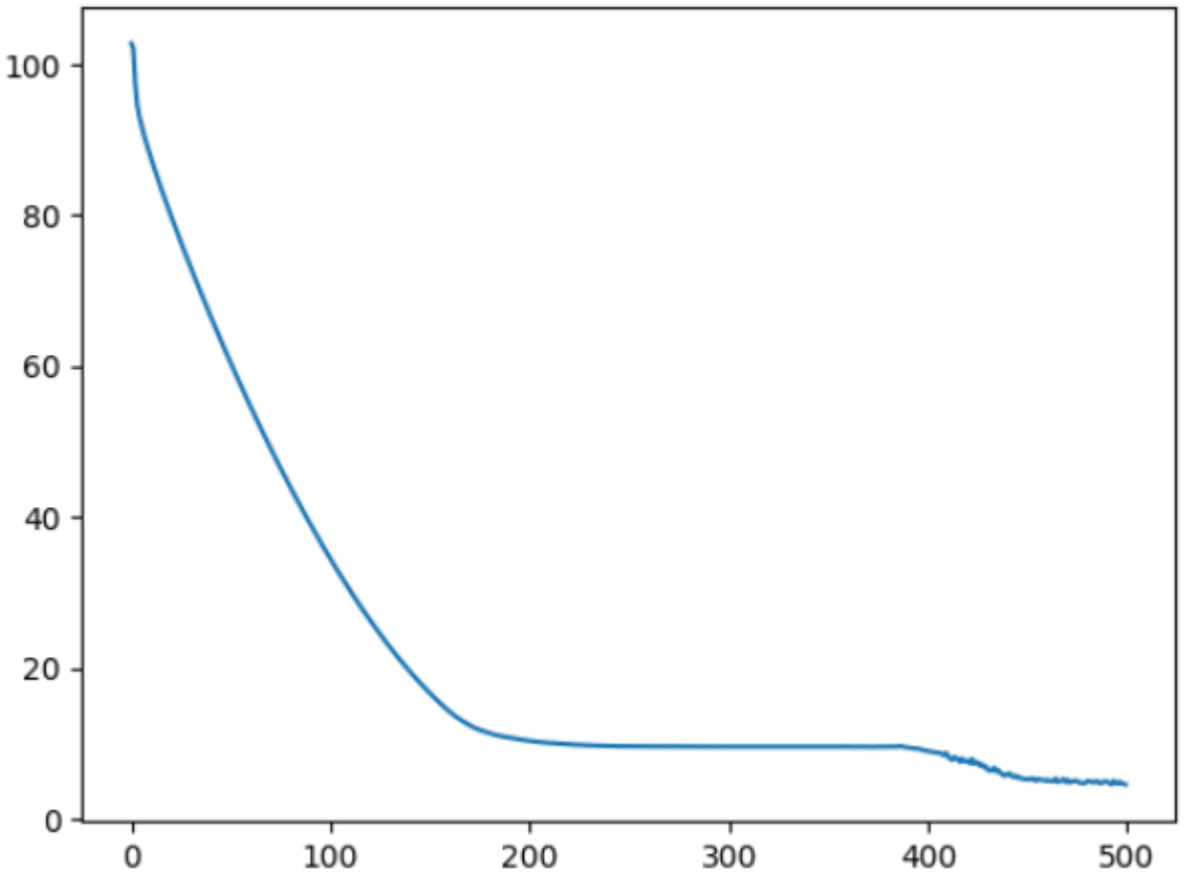

Рис 3.4. Val_mean_absolute_error2

На рис. 3.5 зображена візуалізація процесса тренувального набору моделі, що використовує у якості вхідних даних датасет «Fareng.csv», і яка прошла 500 епох з додаванням dropout шару для покращення продуктивності моделі.

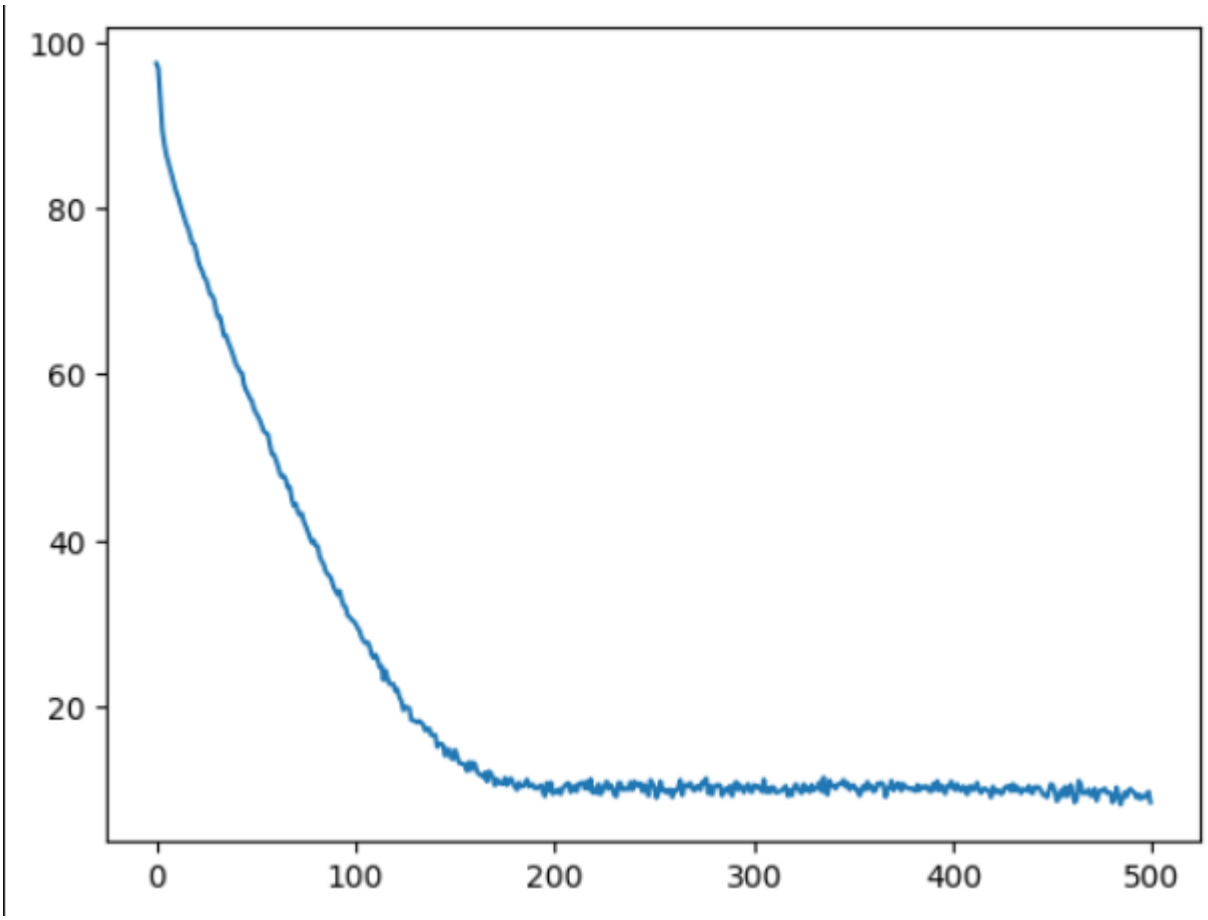

Рис 3. 5. Mean_absolute_error3

На рис. 3.6 зображена візуалізація процесса валідаційного набору моделі, що використовує у якості вхідних даних датасет «Fareng.csv», і яка прошла 500 епох з додаванням dropout шару для покращення продуктивності моделі.

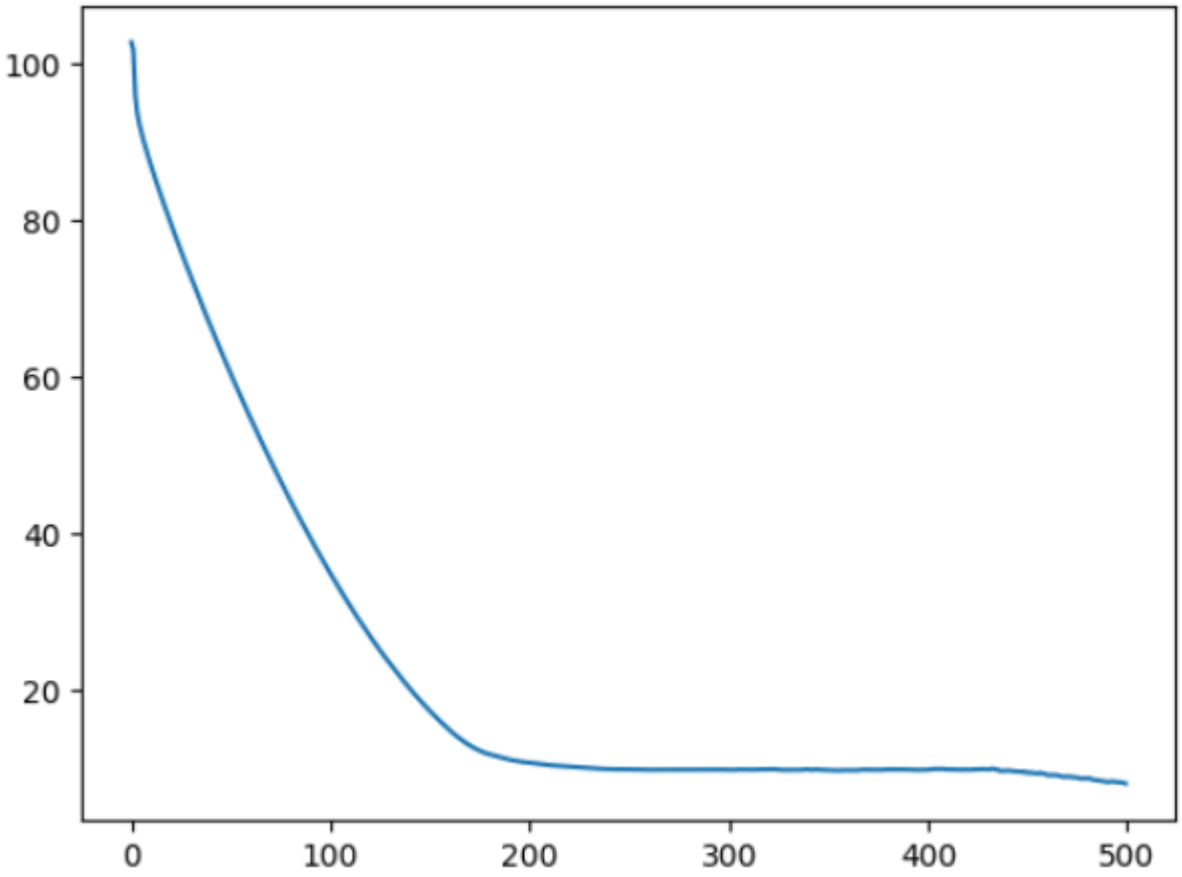

Рис 3.6. Val_mean_absolute_error3

Результати для датасету data1.csv, данних з сенсорів вітроміру IoT у метрах на годину.

На рис. 3.7 зображена візуалізація процесса тренувального набору моделі, що використовує у якості вхідних даних датасет «data1.csv», і яка прошла 100 епох.

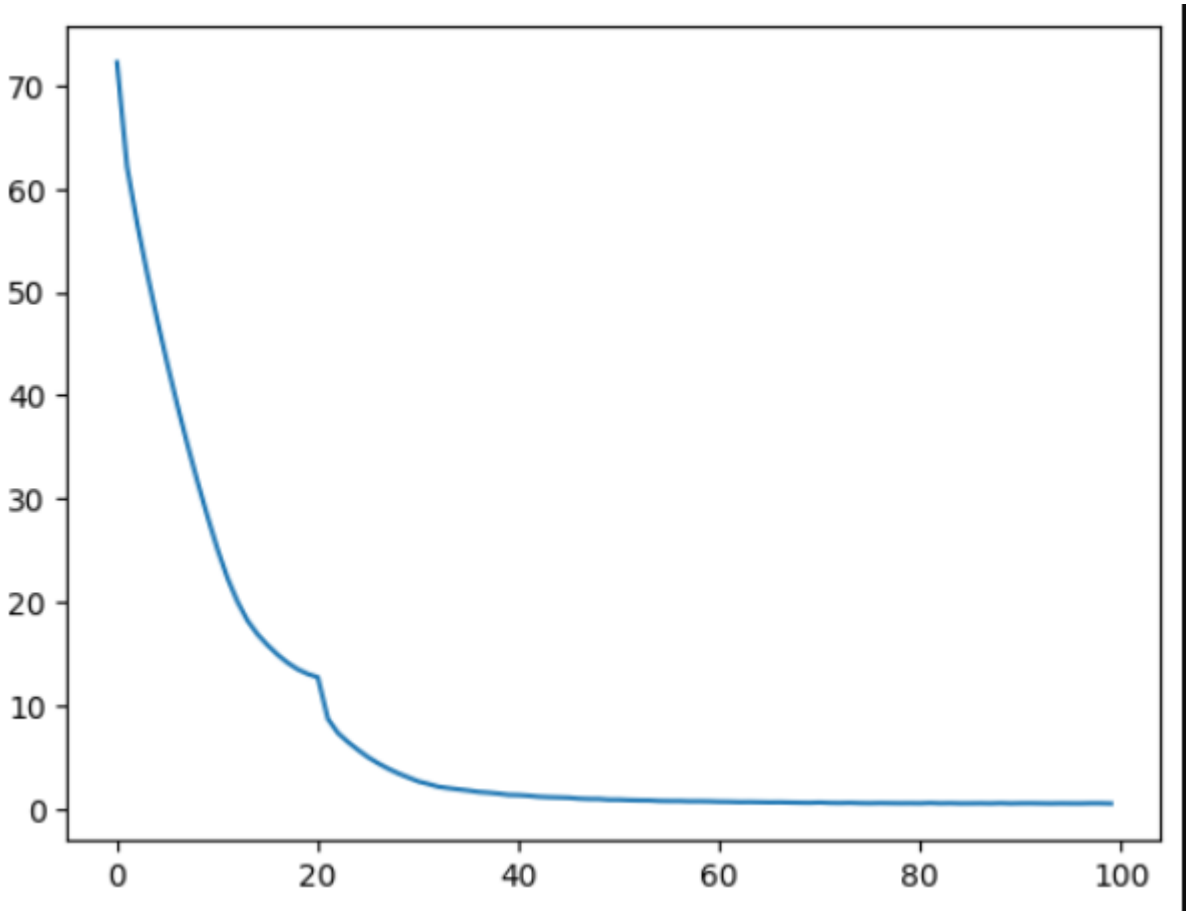

Рис 3.7. Mean_absolute_error1

На рис. 3.8 зображена візуалізація процесса валідаційного набору моделі, що використовує у якості вхідних даних датасет «data1.csv», і яка прошла 100 епох.

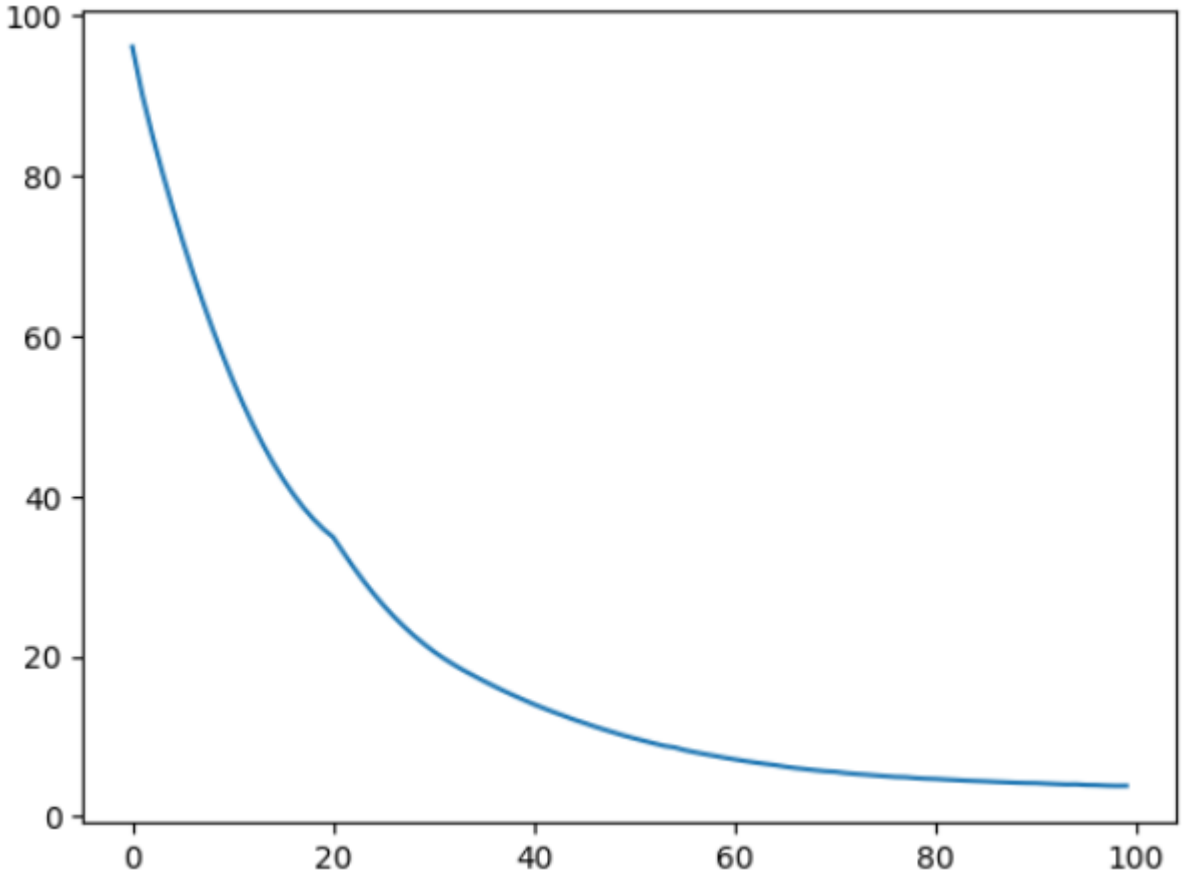

Рис. 3.8. Val_mean_absolute_error1

На рис. 3.9 зображена візуалізація процесса тренувального набору моделі, що використовує у якості вхідних даних датасет «data1.csv», і яка прошла 500 епох.

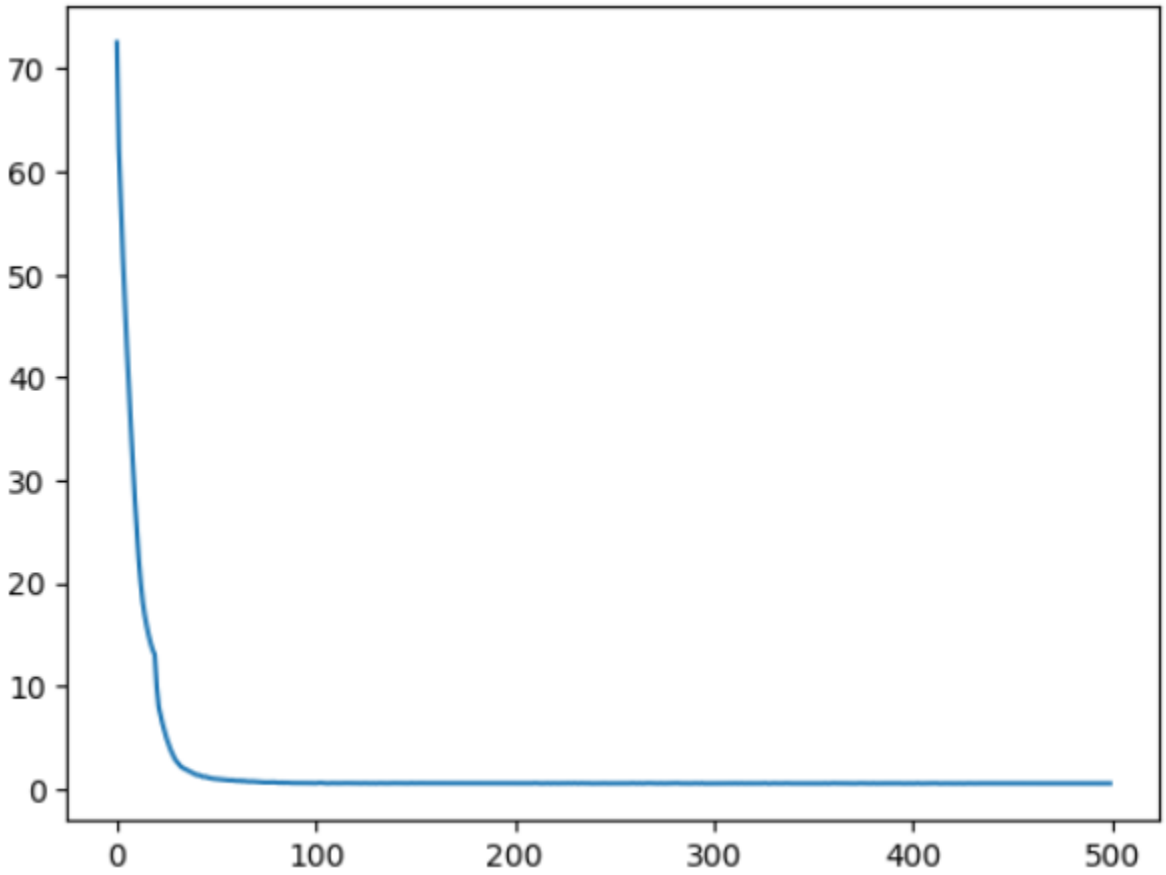

Рис 3.9. Mean_absolute_error2

На рис. 3.10 зображена візуалізація процесса валідаційного набору моделі, що використовує у якості вхідних даних датасет «data1.csv», і яка прошла 500 епох.

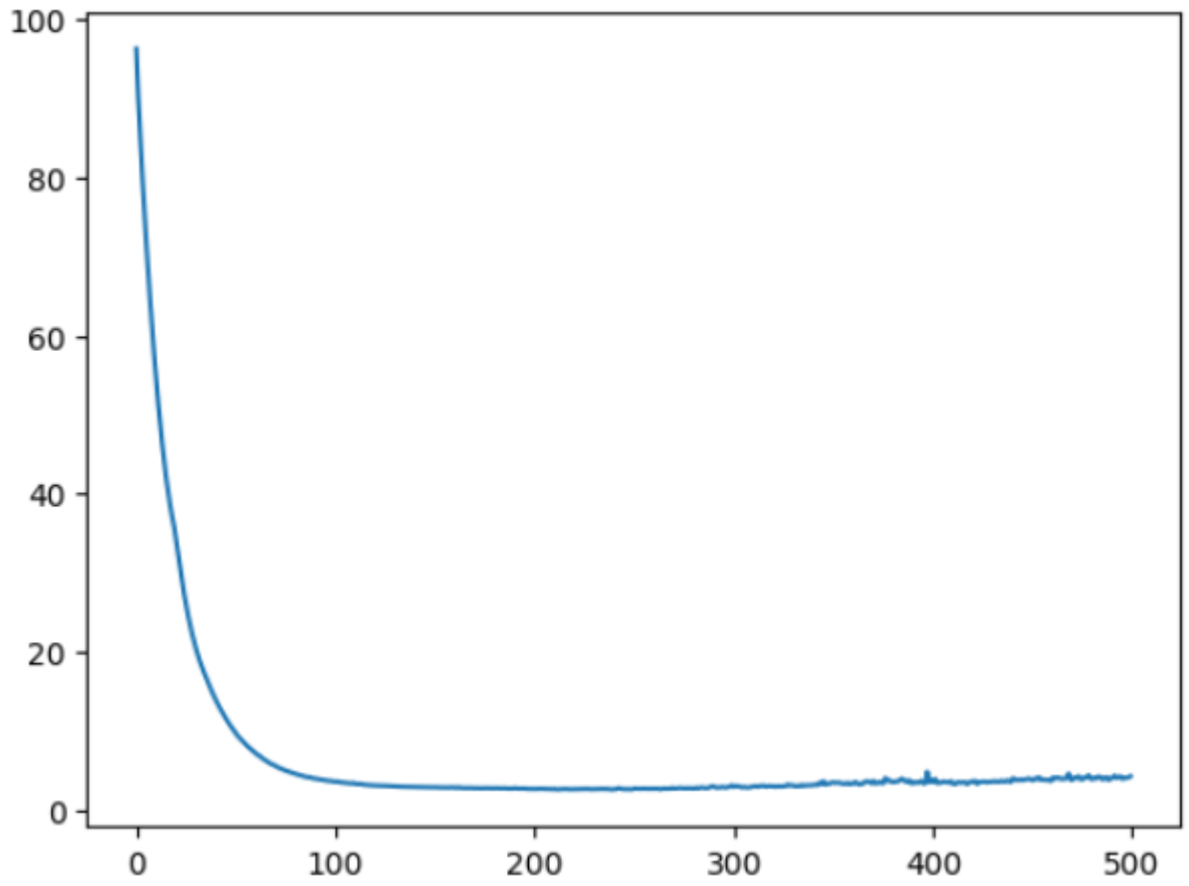

Рис 3. 10. Val_mean_absolute_error2

На рис. 3.11 зображена візуалізація процесса тренувального набору моделі, що використовує у якості вхідних даних датасет «data1.csv», і яка прошла 500 епох з додаванням dropout шару для покращення продуктивності моделі.

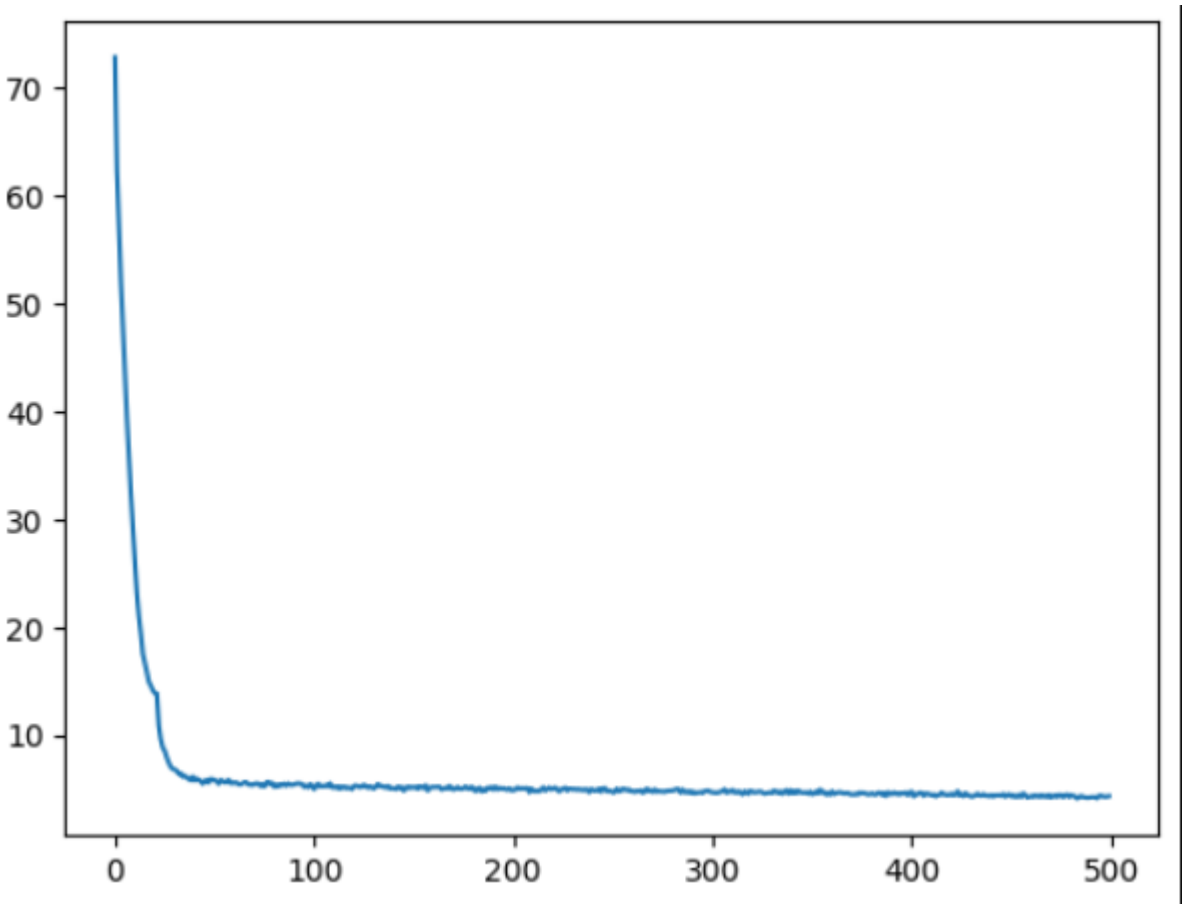

Рис. 3.11. Mean_absolute_error3

На рис. 3.12 зображена візуалізація процесса валідаційного набору моделі, що використовує у якості вхідних даних датасет «data1.csv», і яка прошла 500 епох з додаванням dropout шару для покращення продуктивності моделі.

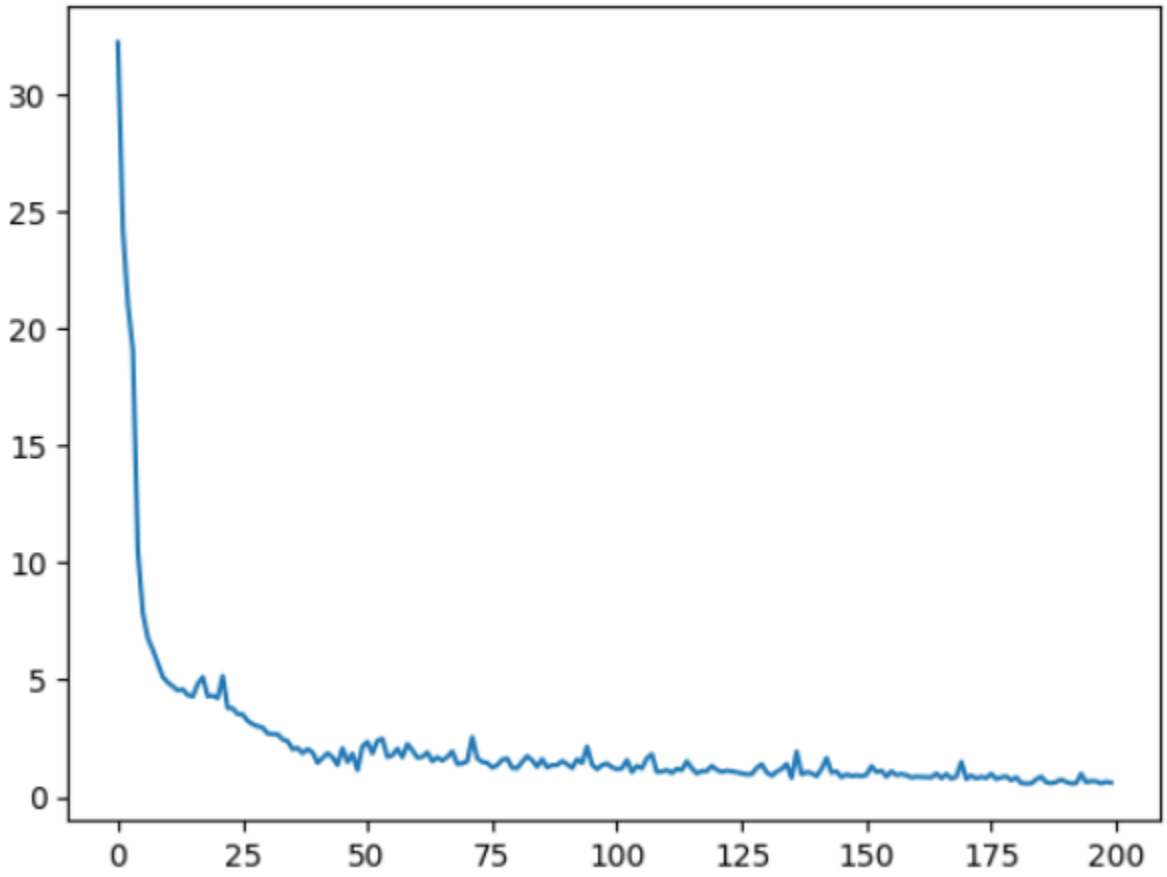

Рис. 3.12. Val_mean_absolute_error3

Уся собрана інформація, яка була отримана у результаті вимірювань знаходиться у таблиці 3.1.

Таблиця 3.1

Складність моделі	Тренувальний набір «Fareng.csv»	Валідаційний набір «Fareng.csv»	Тренувальний набір «data1.csv»	Валідаційний набір «data1.csv»
100 епох	29.9032	35.1561	0.5411	3.8225
500 епох	2.8226	4.5485	0.5079	1.5251
500 епох + шар dropout	6.4811	7.9961	0.4917	4.3537

3.2 Аналіз результатів

В результаті першого випадка вимірювання на датасеті «Fareng.csv» складність моделі складала 32 нейрони, 1 dense шар та 100 епох. В результаті роботи моделі були отримані такі втрати

Тренувальна втрата: 29.9032

Валідаційна втрата: 35.1561

З цього можна зробити висновки що модель перенавчилися, оскільки втрата на валідаційному наборі вища, ніж на тренувальному.

В результаті першого випадка вимірювання на датасеті «Fareng.csv» складність моделі складала 32 нейрони, 1 dense шар та 500 епох. В результаті роботи моделі були отримані такі втрати

Тренувальна втрата: 2.8226

Валідаційна втрата: 4.5484

З цього можна зробити висновки що збільшення кількості епох допомагає покращити результати, оскільки модель має низьку втрату і на тренувальному, і на валідаційному наборах.

В результаті першого випадка вимірювання на датасеті «Fareng.csv» складність моделі складала 32 нейрони, 1 dense шар, 1 dropout шар та 500 епох. В результаті роботи моделі були отримані такі втрати

Тренувальна втрата: 6.4811

Валідаційна втрата: 7.9961

З цього можна зробити що використання dropout може допомогти уникнути перенавчання. Помітна трохи вища втрата на валідаційному наборі через dropout, але це може забезпечити кращу генералізацію.

В результаті першого випадка вимірювання на датасеті «data1.csv» складність моделі складала 32 нейрони, 1 dense шар та 100 епох. В результаті роботи моделі були отримані такі втрати

Тренувальна втрата: 0.5411

Валідаційна втрата: 3.8225

З цього можна зробити висновки що модель перенавчилися, оскільки втрата на валідаційному наборі вища, ніж на тренувальному.

В результаті першого випадка вимірювання на датасеті «data1.csv» складність моделі складала 32 нейрони, 1 dense шар та 500 епох. В результаті роботи моделі були отримані такі втрати

Тренувальна втрата: 0.5079

Валідаційна втрата: 1.5251

З цього можна зробити висновки що збільшення кількості епох кількості епох до 500 було вигідно для моделі та допомогло покращити результати, оскільки модель має низьку втрату і на тренувальному, і на валідаційному наборах.

В результаті першого випадка вимірювання на датасеті «data1.csv» складність моделі складала 32 нейрони, 1 dense шар, 1 dropout шар та 500 епох. В результаті роботи моделі були отримані такі втрати

Тренувальна втрата: 0.4917

Валідаційна втрата: 4.3537

Втрата на валідаційному наборі вища, ніж на тренувальному. З цього можна зробити що використання dropout не допоміг уникнути перенавчання на валідаційному наборі.

Як ми можемо бачити, завдяки програмної реалізації моделі, у обох датасетів покращилася виборка метриків. З додаванням додаткових епох та dropout шару до структури моделі, можна побачити що метрика моделей значно покращилася, хоч у випадку валідаційних наборів можна спострегігати перенавчання.

В результаті проведеного дослідження було виявлено, що візуалізація процесу тренувального набору моделі відіграє ключову роль у вдосконаленні її продуктивності. Чітке відображення динаміки навчання дозволяє ефективно виявляти та усувати проблеми, що виникають під час тренування.

Результати роботи моделі показали значний прогрес у порівнянні з попередніми варіантами. Виявлені зміни у параметрах та архітектурі моделі призвели до покращення її точності та загальної продуктивності. Детальний аналіз отриманих результатів дозволяє визначити оптимальні конфігурації та стратегії тренування для досягнення найкращих показників.

Умови, при яких були отримані кращі результати, також заслуговують на увагу. Встановлено, що оптимальні параметри моделі визначаються не лише характеристиками даних, але й умовами тренування. Ефективне використання ресурсів та оптимальне налаштування гіперпараметрів дозволяють досягти найкращих результатів при обмежених ресурсах.

У цілому, отримані результати підтверджують важливість системного підходу до тренування моделі, включаючи візуалізацію, аналіз результатів та оптимізацію умов тренування. Це дозволяє не лише досягти високої продуктивності моделі, але і підвищити її стабільність та адаптивність до різноманітних умов реального використання.

Висновок до розділу 3

В розділі було описана візуалізація процесса тренувального набору моделі, отриманні результи роботи моделі та при яких умовах вони були отримані для покращення продуктивності моделі. Також було проведено аналіз отриманних результатів.

ВИСНОВОК

Ми успішно створили, протестували та вдосконалили програмну модель для виявлення помилок у пристроях Інтернету речей (IoT). Початкові дані були завантажені з CSV-файлу, піддані аналізу та розділені на тренувальний та валідаційний набори. З метою візуалізації змін у часі ми побудували графік тимчасових рядів, що дозволило ефективно відстежувати динаміку навчання.

Для підготовки даних до тренування моделі використовувався StandardScaler для їх стандартизації. Оптимальні параметри моделі визначалися шляхом декількох експериментів. Ми використовували просту рекурентну нейронну мережу з одним шаром LSTM та одним вихідним шаром. Функція активації для LSTM, а також функція втрат та оптимізатор були визначені для забезпечення ефективного навчання.

Оцінка моделі проводилась на тренувальному та валідаційному наборах даних, використовуючи середню абсолютну помилку (MAE). Ми ретельно вивчали зміну MAE протягом навчання, використовуючи графіки, що надають змогу оцінити продуктивність моделі та її здатність до узагальнення нових даних.

У роботі проведено кілька експериментів, зокрема, із зміною кількості епох та додаванням dropout шару для покращення продуктивності моделі. Результати цих експериментів були виведені графічно для зручного порівняння з базовою моделлю.

Важливим аспектом була систематична візуалізація процесу навчання та використання графіків зміни MAE для тренувальних та валідаційних даних. Це

дозволило ефективно оцінити продуктивність моделі, уникнути перенавчання та підтримувати її стабільність.

Загалом, отримані результати підтверджують високий потенціал використання системного підходу до тренування моделі, де враховуються як параметри даних, так і умови її тренування. Цей підхід дозволяє не лише досягти високої продуктивності, але і підвищити стабільність та адаптивність моделі до різноманітних умов реального використання в IoT пристроях.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How IoT Is Transforming The Manufacturing Industry [Електронний ресурс]. – режим доступу: URL: https://www.forbes.com/sites/forbestechcouncil/2022/09/28/how-iot-is-transforming-the-manufacturing-industry/?sh=784e23c575d7 (Дата звернення: 20.09.2023).
2. Rise of the Internet of Things (IoT) [Електронний ресурс]. – режим доступу: URL: https://www.techradar.com/news/rise-of-the-internet-of-things-iot (Дата звернення: 20.09.2023).
3. Five Most Common Problems IoT Devices Will Encounter in 2020 [Електронний ресурс]. – режим доступу: URL: https://www.iotforall.com/iot-problems (Дата звернення: 20.09.2023).
4. Андрес Міллер, Сара Гвідо, Введення в машинне навчання за допомогою Python, 2016 – 390 с
5. Офіційна документація Python [Електронний ресурс]. – режим доступу: URL: https://docs.python.org/3/ (Дата звернення: 23.09.2023).
6. Офіційна документація TensorFlow [Електронний ресурс]. – режим доступу: URL: https://www.tensorflow.org/ (Дата звернення: 23.09.2023).
7. Pandas Functions in Python: A Toolkit for Data Analysis [Електронний ресурс]. – режим доступу: URL: https://www.geeksforgeeks.org/pandas-functions-in-python/ (Дата звернення: 25.09.2023).
8. Офіційна документація Pandas [Електронний ресурс]. – режим доступу: URL: https://pandas.pydata.org/pandas-docs/stable/index.html (Дата звернення: 27.09.2023).
9. Офіційна документація Scikit-learn [Електронний ресурс]. – режим доступу: URL:

https://scikit-learn.org/stable/supervised_learning.html#supervised-learning (Дата звернення: 29.09.2023).

10. Introduction to Scikit-Learn (sklearn) in Python [Електронний ресурс]. – режим доступу: URL: https://datagy.io/python-scikit-learn-introduction/ (Дата звернення: 05.10.2023).
11. Офіційна документація mathlotlib [Електронний ресурс]. – режим доступу: URL: https://matplotlib.org/stable/users/index.html (Дата звернення: 06.10.2023).
12. Matplotlib Tutorial – Principles of creating any plot with the Matplotlib library [Електронний ресурс]. – режим доступу: URL: https://www.machinelearningplus.com/plots/matplotlib-plotting-tutorial/ (Дата звернення: 06.10.2023).
13. Data Visualization in Python with matplotlib, Seaborn, and Bokeh [Електронний ресурс]. – режим доступу: URL: https://machinelearningmastery.com/data-visualization-in-python-with-matplotlib-seaborn-and-bokeh/ (Дата звернення: 06.10.2023).
14. ML Approaches for Time Series [Електронний ресурс]. – режим доступу: URL: https://towardsdatascience.com/ml-approaches-for-time-series-4d44722e48fe (Дата звернення: 07.10.2023).
15. Time series forecasting | TensorFlow Core [Електронний ресурс]. – режим доступу: URL: https://www.tensorflow.org/tutorials/structured_data/time_series (Дата звернення: 07.10.2023).
16. Time Series Analysis and Forecasting | Data-Driven Insights (Updated 2023) [Електронний ресурс]. – режим доступу: URL: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-to-time-series-analysis/ (Дата звернення: 07.10.2023).

17. How to Make Predictions for Time Series Forecasting with Python [Електронний ресурс]. – режим доступу: URL: https://machinelearningmastery.com/make-predictions-time-series-forecasting-python/ (Дата звернення: 07.10.2023).
18. Model evaluation, model selection, and algorithm selection in machine learning [Електронний ресурс]. – режим доступу: URL: https://sebastianraschka.com/blog/2018/model-evaluation-selection-part4.html (Дата звернення: 10.10.2023).
19. Machine Learning Model Validation – The Data-Centric Approach [Електронний ресурс]. – режим доступу: URL: https://appen.com/blog/machine-learning-model-validation/ (Дата звернення: 12.10.2023).
20. A Gentle Introduction to Long Short-Term Memory Networks by the Experts [Електронний ресурс]. – режим доступу: URL: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/ (Дата звернення: 12.10.2023).
21. Tutorial on LSTMs: A Computational Perspective [Електронний ресурс]. – режим доступу: URL: https://towardsdatascience.com/tutorial-on-lstm-a-computational-perspective-f3417442c2cd (Дата звернення: 12.10.2023).
22. Long short-term memory (LSTM) RNN in Tensorflow [Електронний ресурс]. – режим доступу: URL: https://www.geeksforgeeks.org/long-short-term-memory-lstm-rnn-in-tensorflow/ (Дата звернення: 14.10.2023).
23. Long Short-term Memory [Електронний ресурс]. – режим доступу: URL: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory (Дата звернення: 14.10.2023).
24. 10.1. Long Short-Term Memory (LSTM) [Електронний ресурс]. – режим доступу: URL: https://d2l.ai/chapter_recurrent-modern/lstm.html

25. Data Scaling for Machine Learning [Електронний ресурс]. – режим доступу: URL: https://betterdatascience.com/data-scaling-for-machine-learning/ (Дата звернення: 14.10.2023).
26. Data Visualization using Matplotlib [Електронний ресурс]. – режим доступу: URL: https://www.geeksforgeeks.org/data-visualization-using-matplotlib/ (Дата звернення: 14.10.2023).
27. Understanding Mean Absolute Error (MAE) in Regression: A Practical Guide [Електронний ресурс]. – режим доступу: URL: https://medium.com/@m.waqar.ahmed/understanding-mean-absolute-error-mae-in-regression-a-practical-guide-26e80ebb97df (Дата звернення: 05.11.2023).
28. What Is Mean Absolute Error In Machine Learning [Електронний ресурс]. – режим доступу: URL: https://robots.net/fintech/what-is-mean-absolute-error-in-machine-learning/ (Дата звернення: 05.11.2023).
29. Absolute Error & Mean Absolute Error (MAE) [Електронний ресурс]. – режим доступу: URL: https://www.statisticshowto.com/absolute-error/ (Дата звернення: 05.11.2023).
30. How to Calculate Mean Absolute Error in Python? [Електронний ресурс]. – режим доступу: URL: https://www.geeksforgeeks.org/how-to-calculate-mean-absolute-error-in-python/ (Дата звернення: 05.11.2023).
31. Ісаєв В.С. Модель управління ризиками при реалізації IT-проекту. Праці 8-ої Міжнародній конференції «Комп'ютерне моделювання в наукоємних технологіях (КМНТ-2022)

Додаток А

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

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Магістр**
Галузь знань: **15 – Автоматизація та приладобудування**
Спеціальність: **151 – «Автоматизація та комп'ютерно-інтегровані технології»**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної та прикладної системотехніки
д.т.н., проф. Шматков С. І.

«08 » грудня 2022 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ісаєв Владислав Сергійович
(прізвище, ім'я, по батькові студента)

1. Тема роботи «Модель виявлення несправностей у пристроях IoT»

керівник роботи Бикова Тетяна Володимирівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10» листопада 2023 року № 4101-5/3197

2. Строк подання студентом роботи 28.11.2023

3. Перелік питань, які потрібно розробити)

1) Створення моделі IoT за допомогою мови програмування Python.
2) Розробка алгоритмів пошуку несправностей у моделі IoT на основі різних критеріїв.
3) Використання та імплементування у модель різноматінтних dataset пов'язаних з IoT.
4) Розробка симуляції можливих несправностей у моделі та можливостей для її виправлення.
5) Тестування програми

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Аналіз та підбір літератури та створення літературної бази для розробки моделі	01.01.2023 - 02.02.2023
2	Аналіз проблематики IoT	02.02.2023- 01.04.2023
3	Розробка програмної моделі IoT	01.04.2023 - 12.08.2023
4	ограмна реалізація моделі виявлення несправностей та їх правлення	12.08.2023 - 15.09.2023
5	Тестування та визначення ефективності моделі пошуку правлення IoT	15.09.2023 - 02.10.2023
6	зробка рекомендацій щодо застосування моделі пошуку справностей у пристроях IoT	02.10.2023 - 31.11.2023

5. Дата видачі завдання 8.12.2022

Студент Ісаєв В.С _____

Керівник роботи Бикова Т.В. _____

Додаток Б

Технічне завдання

на розробку програмного виробу
«Модель виявлення несправностей у пристроях IoT»

Назва розділу	Назва і зміст підрозділу
1. Введення	1.1. Назва виробу: Модель виявлення несправностей у пристроях IoT. 1.2. Галузь застосування: автоматизація.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 151 – «Автоматизація та комп'ютерно-інтегровані технології» 2.2. Завдання на кваліфікаційну роботу магістра затверджено наказом ректора № 4101-5/3197 від 10.11.2023.
3. Призначення розробки	3.1. Метою цієї роботи є застосування і удосконалення моделей машинного навчання, що прогнозують вихід похибки вимірювання сенсорів IoT за допустимий діапазон. 3.2. Призначення виробу: використання виробу для Виявлення похибок та прогнозування похибок у роботі пристроїв IoT

4. Технічні вимоги до виробу	5.1. Вимоги до функціональних характеристик: можливість виконати задачу регресії із вхідної вибірки на основі інформаційних критеріїв. 5.2. Вимоги до надійності: забезпечення працездатності методу регрессії при подачі вхідних даних неправильного формату. 5.3. Вимоги до умов експлуатації: встановлення мови програмування Python та необхідних бібліотек для роботи методу регресії. 5.4. Вимоги до складу і параметрів технічних засобів: комп'ютер або ноутбук із 4 ГБ оперативної пам'яті та процесором не нижче Intel Core i3 9-го покоління . 5.5. Вимоги до інформаційної та програмної сумісності: підтримка ОС Linux або Windows 8/10, підтримка Anaconda, Python 3. 5.6. Вимоги до маркування та упаковки: жорстка упаковка з пластмаси, маркування на українській та англійській мовах. 5.7. Вимоги до транспортування і зберігання: транспортування в упаковці будь-яким способом, температура транспортування/зберігання +5-+20 C^o. 5.8. Спеціальні вимоги: відсутні.
5. Вимоги до документації.	Документацією до виробу «Модель виявлення несправностей у пристроях IoT» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити як Додаток Б до пояснювальної записки до дипломної роботи). 2) Програму і методику випробувань розробленого виробу (представити як Додаток В до пояснювальної записки до дипломної роботи).

<table>
<tr><td>6. Техніко-економічні показники</td><td colspan="2">Орієнтовна оцінка ефективності та якості виконуваної
регрісії даних: точність (accuracy) повинна бути вище 90%.</td></tr>
<tr><td rowspan="2">7. Стадії і етапи розробки</td><td>Дата</td><td>Назва
етапу</td></tr>
<tr><td>15.10.22 – 01.11.22

01.11.22 – 01.01.23

01.01.23 – 01.02.23

02.02.23 – 01.03.23

02.03.23 – 15.04.23

19.03.23 – 16.04.23

17.04.23 – 17.05.23

28.04.23 – 19.05.23

22.09.23 – 30.10.23

30.9.23 – 12.11.23</td><td>1. Визначення постановки задачірегрессії даних;
2. Аналітичний огляд існуючихметодів регрісійних даних;
3. Огляд існуючих програмнихзасобів для регресії даних;
4. Формалізація і уявлення вхідних
даних;
5. Підготовка та обробка вхідних
даних для ефективності методу;
6. Опис методу регресії даних;
7. Опис обчислювального алгоритму вирішення задачі.
8. Опис обраного програмного забезпечення для вирішення задачі.
9. Тестування різних датасетів у
програмному засобі для аналізу отриманних даних.
10. Огляд та аналіз отриманих результатів.</td></tr>
</table>

8. Порядок контролю і приймання	Загальні вимоги до приймання розроблюваного виробу: 1) Перевірка ходу розробки програмного виробу керівнику робіт виконувати 1 раз в 3 тижні. 2) Випробування виробу відповідно до Програми і методики випробувань провести на базі комп'ютерного класу або на базі приватного приміщення. 3) Захист розробленого виробу провести на засіданні атестаційної комісії. 4) Пояснювальну записку представити на паперових носіях в одному примірнику та в електронному вигляді.

Виконавець:
студент групи КУ-61

Ісаєв В.С.

Замовник:
к.т.н., доцент кафедри
теоретичної та прикладної
системотехніки

Бикова Т.В

Додаток В

Програма і методика випробувань програмного виробу

«Модель виявлення несправностей у пристроях IoT»

1 Об'єкт випробувань

1.1 Назва виробу: Модель виявлення несправностей у пристроях IoT.
1.2 Галузь застосування: автоматизація.

2. Мета випробувань

Метою цієї роботи є застосування і удосконалення моделей машинного навчання, що прогнозують вихід похибки вимірювання сенсорів IoT за допустимий діапазон.

3. Загальні положення

3.1 Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії та перевірку даного виробу.

3.2 Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу або будь-якого приміщення в період роботи атестаційної комісії.

3.3 Обсяг випробувань

Приймальні випробування програмного виробу проводяться в обсязі відповідному цієї Програми і методики випробувань.

3.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до виробу

4.1 Вимоги до функціональних характеристик:
можливість виконати задачу регресії із вхідної вибірки
на основі інформаційних критеріїв.
4.2 Вимоги до надійності: забезпечення працездатності методу регрессії при подачі вхідних даних неправильного формату.
4.3 Вимоги до умов експлуатації: встановлення мови програмування Python та необхідних бібліотек для роботи методу регресії.
4.4 Вимоги до складу і параметрів технічних засобів: комп’ютер або ноутбук із 4 ГБ оперативної пам’яті та процесором не нижче Intel Core i3 9-го покоління .
4.5 Вимоги до інформаційної та програмної сумісності: підтримка ОС Linux або Windows 8/10, підтримка Anaconda, Python 3.
4.6 Вимоги до маркування та упаковки: жорстка упаковка з пластмаси, маркування на українській та англійській мовах.

4.7 Вимоги до транспортування і зберігання: транспортування в упаковці будь-яким способом, температура транспортування/зберігання +5-+20 Cº.

4.8 Спеціальні вимоги: відсутні.

5. Вимоги до документації

Документацією до виробу «Модель виявлення несправностей у пристроях IoT» вважати:

3) Справжнє Технічне завдання на розробку виробу (представити як Додаток Б до пояснювальної записки до дипломної роботи).

4) Програму і методику випробувань розробленого виробу (представити як Додаток В до пояснювальної записки до дипломної роботи).

6. Засоби і порядок випробувань

Засоби випробувань представлено на ПК на яких встановлено наступні програмні засоби: GoogleColab, Python. В програмному засобі GoogleColab встановлені необхідні бібліотеки для запуску, такі як numpy, scikit-learn, pandas, matplotlib, seaborn, scipy.

6.2 Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:

- ознайомчий (1-й етап);

- власне випробування програмного виробу (2-й етап).

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1) Перевірку комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.

2) Виконання тесту:

2.1) Запустити термінал або команду строку.

2.2) Проглянути написаний код та перевірити на наявність помилок.

2.3) Перевірити наявність встановлених бібліотек шляхом запуску програми на даних, які згенеровані випадковим чином

3) Якість програмної документації перевіряється на відповідність до стандартів ISO/IEC 26514:2008.

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1) Перевірку відповідності технічних характеристик програми вимогам технічного завдання.

2) Перевірку ступеня виконання функціональних вимог до програми.

3) Методику проведення перевірок:

3.1) Запустити термінал або команду строку.

3.2) Перевірити наявність встановлених бібліотек шляхом запуску програми на даних, які завантажені у програмну модель.

3.3) Порядок проведення випробувань:

- Запустити програму за допомогою терміналу з використанням вхідних данних з сенсорів пристроїв IoT та моделі пошуку несправностей у пристроях IoT.

- Перевірити працездатність розробленого програмного продукту.
- Перевірити правильність роботи нейронної мережі.
- Перевірити правильність візуалізації результатів.

4) Якщо перевірки на першому та другому етапах виконано успішно, то виріб вважається таким, що пройшов випробування.

Виконавець:

студент групи КУ-61 Ісаєв В.С. _____

Додаток Г

ЛІСТІНГ КОДУ.

«Модель пошуку несправностей у пристроях IoT»

```
import pandas as pd
from datetime import timedelta
import numpy as np
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error
from matplotlib import pyplot as plt
from typing import List
df = pd.read_csv("data1.csv", parse_dates=["Date"])
df.plot(x= "Date", y="Speed")
df.head(5)
df["Date"].min(), df["Date"].max()
df_6_yr = df[df["Date"] > df["Date"].max() - timedelta(days=365 * 6)]
df_6_yr.shape
df_6_yr["Date"].min(), df_6_yr["Date"].max()
train_size = int(df_6_yr.shape[0] * 0.8)
train_df = df_6_yr.iloc[:train_size]
val_df = df_6_yr.iloc[train_size:]
train_df.shape, val_df.shape
train_df["Date"].min(), train_df["Date"].max(), val_df["Date"].min(),
val_df["Date"].max()
scaler = StandardScaler()
scaler.fit(train_df[["Speed"]])
def make_dataset(
   df,
    window_size,
    batch_size,
    use_scaler=True,
    shuffle=True
    ):
  features = df[["Speed"]].iloc[:-window_size]
  if use_scaler:
    features = scaler.transform(features)
  data = np.array(features, dtype=np.float32)
  ds = tf.keras.preprocessing.timeseries_dataset_from_array(
     data=data,
     targets=df["Speed"].iloc[window_size:],
     sequence_length=window_size,
```

```
            sequence_stride=1,
            shuffle=shuffle,
            batch_size=batch_size)
        return ds

    example_ds = make_dataset(df=train_df, window_size=3, batch_size=2, use_scaler=False, shuffle=False)
    example_feature, example_label = next(example_ds.as_numpy_iterator())
    example_feature.shape
    example_label.shape
    train_df["Speed"].iloc[:6]
    print(example_feature[0])
    print(example_label[0])
    print(example_feature[1])
    print(example_label[1])

    window_size = 10
    batch_size = 8
    train_ds = make_dataset(df=train_df, window_size=window_size, batch_size=batch_size, use_scaler=True, shuffle=True)
    val_ds = make_dataset(df=val_df, window_size=window_size, batch_size=batch_size, use_scaler=True, shuffle=True)

    lstm_model = tf.keras.models.Sequential([
        tf.keras.layers.LSTM(32, return_sequences=False),
        tf.keras.layers.Dense(1)
    ])

    def compile_and_fit(model, train_ds, val_ds, num_epochs: int = 20):
        model.compile(
            loss=tf.losses.MeanSquaredError(),
            optimizer=tf.optimizers.Adam(),
            metrics=[tf.metrics.MeanAbsoluteError()]
            )
        history = model.fit(
            train_ds,
            epochs=num_epochs,
            validation_data=val_ds,
            verbose=0
            )
        return history
```

```
        history = compile_and_fit(lstm_model, train_ds, val_ds,
num_epochs=100)
        plt.plot(history.history['mean_absolute_error'])
        plt.plot(history.history['val_mean_absolute_error'])
        lstm_model.evaluate(train_ds)
        lstm_model.evaluate(val_ds)
        lstm_model = tf.keras.models.Sequential([
            tf.keras.layers.LSTM(32, return_sequences=False),
            tf.keras.layers.Dense(1)
        ])

        history = compile_and_fit(lstm_model, train_ds, val_ds,
num_epochs=500)
        lstm_model.evaluate(train_ds)

        ltm_model.evaluate(val_ds)
        plt.plot(history.history['mean_absolute_error'])
        plt.plot(history.history['val_mean_absolute_error'])
        lstm_model = tf.keras.models.Sequential([
            tf.keras.layers.LSTM(32, return_sequences=False),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(1)
        ])
        history = compile_and_fit(lstm_model, train_ds, val_ds,
num_epochs=500)
        lstm_model.evaluate(train_ds)
        lstm_model.evaluate(val_ds)
        plt.plot(history.history['mean_absolute_error'])
        plt.plot(history.history['val_mean_absolute_error'])
```