НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.891.3

«До захисту допущено»
В.о. завідувача кафедри
_____ Едуард ЖАРІКОВ
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Методи та програмне забезпечення виявлення аномалій при розгортанні мікросервісу**»**

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІТ-03мп
Клименко Денис Владиславович _____

Керівник:
д.т.н., професор
Стеценко Інна Вячеславівна, _____

Рецензент:
доцент кафедри ОТ, к.т.н., доц.,
Клименко Ірина Анатоліївна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського» Факультет

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних систем»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Едуард ЖАРІКОВ

«___»_____2021 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Клименку Денису Владиславовичу

1. Тема дисертації «Методи та програмне забезпечення виявлення аномалій при розгортанні мікросервісу», науковий керівник дисертації Стеценко Інна Вячеславівна, д.т.н., професор, затверджені наказом по університету від «27» жовтня 2021 р. № 3587-с

2. Термін подання студентом дисертації «6» грудня 2021 р.

3. Об'єкт дослідження: процес виявлення функціональних аномалій мікросервісу під час розгортання.

4. Вихідні дані: технічна література про виявлення функціональних аномалій, розробку мікросервісної архітектури, навчання без вчителя.

5. Перелік завдань, які потрібно розробити: опис предметної області, аналіз існуючих рішень, аналіз та вибір технічних засобів для реалізації системи, розроблення програмного рішення, тестування роботи системи, графічний матеріал.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: 27 рисунків, 23 таблиці, діаграма архітектури нейронної мережі, вихідний код.

7. Орієнтовний перелік публікацій: 1 публікація в фахових виданнях

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Огляд предметної області	07.09.2021 р.	
2.	Аналіз мікросервісної архітектури	21.09.2021 р.	
3.	Аналіз та вибір технічних засобів для реалізації системи	28.09.2021 р.	
4.	Розгортання середовища для розробки та тестування	01.10.2021 р.	
5.	Розроблення програмного рішення	30.11.2021 р.	
6.	Тестування розробленої системи	12.11.2021 р.	
7.	Розроблення стартап – проекту	19.11.2021 р.	
8.	Оформлення текстової документації	01.11.2021 р.	
9.	Виконання експериментальних досліджень	10.11.2021 р.	
10.	Оформлення пояснювальної записки	20.11.2021	
11.	Подання дисертації на попередній захист	22.11.2021	
12.	Подання дисертації на захист	6.12.2021	

Студент Денис КЛИМЕНКО

Науковий керівник Інна СТЕЦЕНКО

РЕФЕРАТ

Магістерська дисертація на тему «Методи та програмне забезпечення виявлення аномалій при розгортанні мікросервісу».

Обсяг роботи: 76 сторінки, 27 ілюстрацій, 23 таблиці, 24 джерел посилань.

Актуальність теми: під час розгортання нової версії сервісу інженер, який виконує розгортання, повинен стежити за великої кількістю змінних, наприклад: навантаження на cpu віртуальної машини, кількість 5xx HTTP помилок, логи сервісу, та відстежувати загальну працездатність системи, для того, щоб визначити чи є аномалії в роботі мікросервісу. Суттєвими недоліками є трудомісткий та наявність людського фактору, тобто аномалія може бути пропущена. Щоб позбутися даних недоліків, необхідно розробити систему, яка зможе автоматично виявляти функціональні аномалії в роботі нової версії сервісу, базуючись на робочих показниках(metrics) та потоці подій.

Мета дослідження: метою магістерської дисертації є зменшення впливу дефектного коду на роботу системи під час розгортання мікросервісу.

Об’єктом розробки є процес виявлення функціональних аномалій мікросервісу під час розгортання.

Предметом дослідження є методи виявлення функціональних аномалій при розгортанні мікросервісу.

Для реалізації поставленої мети сформульовані наступні завдання:

- аналіз існуючих рішень;
- реалізація системи визначення функціональних аномалій на базі алгоритму навчання без вчителя;
- оцінка ефективності розробленої системи.

Наукова новизна результатів магістерської дисертації полягає в тому, що запропоновано вдосконалення методу виявлення аномалій у функціонуванні мікросервісу на основі комбінованого аналізу робочих показників сервісу та потоку подій. Результат досягнутий шляхом розробки модернізованого алгоритму з використанням навчання без вчителя нейронних мереж.

Практичне значення отриманих результатів полягає в тому, що реалізовано систему, яка може автоматично припинити розгортання версії мікросервіса з аномальною поведінкою. Дана система може бути використана в будь-якій системі управління розгортанням з метою зменшення впливу дефектів коду на роботу інформаційної системи.

Публікації: за результатами виконаної роботи було опубліковано 1 наукову статтю.

Клименко Д.В. Методи та програмне забезпечення виявлення аномалій при розгортанні мікросервісу. Перша Всеукраїнська науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2021). Секція кафедри інформатики та програмної інженерії. Матеріали конференції. – Київ. – 2021. 22–26 листопада 2021р. – С.45.

Ключові слова: мікросервіси, мікросервісна архітектура, розгортання, навчання без вчителя.

ABSTRACT

Master's dissertation on "Methods and software for detecting anomalies in the deployment of microservice".

Paper size: 76 pages, 27 illustrations, 23 tables, 24 references.

Actuality of theme: when deploying a new version of the service, the engineer performing the deployment must monitor a large number of variables such as CPU load on the virtual machine, 5xx HTTP errors, service logs, and overall system performance to determine if there are any anomalies in the microservice. Not only does it take a lot of time, but it also has a significant portion of the human factor. Therefore, it is necessary to develop a system that can automatically detect functional anomalies in the new version of the service based on performance indicators (metrics) and the flow of events.

Purpose of the study: the purpose of the master's dissertation is to reduce the impact of defective code on the operation of the system during the deployment of microservice.

The object of development is the process of detecting functional anomalies of the microservice during deployment.

The subject of the study are methods for detecting functional abnormalities in the deployment of microservice.

To achieve this goal, the following tasks are formulated:

- analysis of existing solutions;
- implementation of a system for determining functional anomalies based on the algorithm unsupervised learning;
- evaluation of the effectiveness of the developed system.

The scientific novelty of the results of the master's dissertation is that it is proposed to improve the method of detecting anomalies in the functioning of the microservice on the basis of a combined analysis of the performance of the service and

the flow of events. The result was achieved by developing an upgraded algorithm using unsupervised learning.

The practical significance of the obtained results is that the implemented system can automatically stop the deployment of a version of the microservice with abnormal behavior. This system can be used in any deployment management system to reduce the impact of defects on the operation of the information system.

Publications: as a result of the work done 3 scientific articles were published.

Klymenko D. Methods and software for detecting anomalies in the deployment of microservice. The First All-Ukrainian Scientific and Practical Conference of Young Scientists and Students "Software Engineering and Advanced Information Technologies" (SoftTech-2021). Section of the Department of Informatics and Software Engineering. Conference materials. - Kyiv. - 2021. November 22-26, 2021. - P.45.

Keywords: microservices, microservice architecture, deployment, unsupervised learning.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

API – Application user interface

CPU – Central processor unit

GPU – graphics processing unit

HTTP – Hypertext transfer protocol

Под – Pod в системі Kubernetes

ПММ – приховані моделі Маркова

LSTM – Long Short-term memory

РНМ – рекурентні нейроні мережі

REST – Representational state transfer

URL – Uniform Resource Locator

Dropout – випадання активації нейронів

ВСТУП

Мікросервісна архітектура надає надійну та масштабовану платформу для розробки складних інформаційних систем. Проте основна перевага мікросервісів – розподіл системи на велику к-ть малих сервісів, водночас є і слабкістю. Розгортання нової версії одного з сотні мікросервісів може призвести до каскадної нестабільності всієї системи. Щоб запобігти цьому, необхідно мати інструмент, який зможе за допомогою аналізу поведінки системи виявити функціональні аномалії, на основі яких можна прийняти рішення про припинення розгортання та повернення системи в стабільний стан.

Виявлення аномальної поведінки, яка веде до збоїв в системі, є складним процесом, адже кожен сервіс генерує сотні та навіть тисячі показників та подій кожну хвилину. Це ускладнює інженерам процес пошуку проблеми, що в свою чергу може призвести до значних фінансових втрат. Саме тому проблема автоматичного виявлення аномалій є дуже актуальною проблемою та потребує уваги.

Об’єктом розробки є процес виявлення функціональних аномалій мікросервісу під час розгортання.

Предметом дослідження є методи виявлення функціональних аномалій при розгортанні мікросервісу.

Метою магістерської дисертації є зменшення впливу дефектного коду на роботу системи під час розгортання мікросервісу. Дана мета досягається за рахунок аналізу часових показників нероздільно разом з потоком подій.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

У даному розділі розглянуто методи виявлення аномалій в роботі мікросервісу. Оскільки дана робота безпосередньо стосується мікросервісів, то необхідно також ознайомитись і з мікросервісною архітектурою. Додатково розглядається платформа оркестрації мікросервісів Kubernetes, оскільки вона використовується для тестування та розробки системи.

1.1 Методи виявлення аномалій під час розгортання мікросервісів

В даному підрозділі розглядаються методи виявлення аномалій в роботі мікросервісу, їх переваги та недоліки.

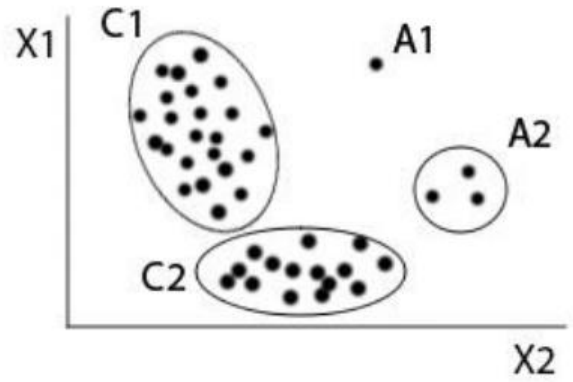
Задача виявлення аномалій часових даних - це пошук непередбачених значень, які сигналізують, що система відхилилась від бажаної поведінки. Відхилення поведінки від норми може бути викликане різними чинниками або дефектами, наприклад: збільшення навантаження, зловмисні атаки, тощо. Однак, слід зауважити, що аномалії не завжди можуть бути пов’язані з негативними причинами: деякі дії в системі або недетермінована логіка роботи є нормальними, але можуть бути класифіковані як аномалії. Щоб уникнути великої кількості помилково позитивних аномалій, необхідно ретельно обирати порогові значення реальних та передбачених даних.

Аномалії поділяються на такі типи:

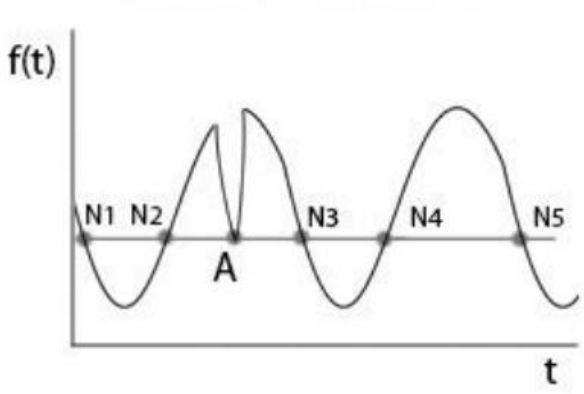
- точкові аномалії – індивідуальні сигнали, що мають значне відхилення від передбачених значень. Такі аномалії досить легко відстежити, тож існує безліч способів для їх виявлення;
- контекстні аномалії – сигнали можуть бути одночасно аномальними та не аномальними, в залежності від контексту, в якому відбуваються. Для виявлення таких аномалій необхідно використовувати умовні моделі, в яких визначається контекст по значенню атрибутів вхідного сигналу, наприклад: пора року, джерело, стан системи та інші;

- колективні аномалії - сигнали які можуть не бути аномаліями окремо від інших, але є аномальними, якщо зустрічаються в наборі або послідовності сигналів.

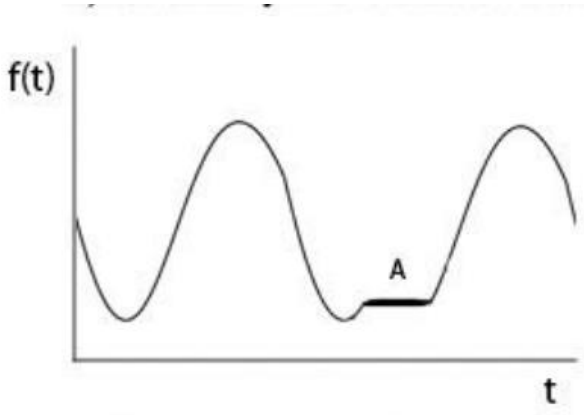
На рисунку 1.1 зображено види аномалій.



а) точкові аномалії



б) контекстні аномалії



в) колективні аномалії

Рисунок 1.1 – Види аномалії [24]

В залежності від наявності чіткого маркування даних використовують наступні алгоритми навчання моделей виявлення:

- Навчання з вчителем - передбачає наявність промаркованих даних з нормальними та аномальними даними, на основі яких створюється модель класифікації відповідних класів. Труднощами такого підходу є необхідність рівномірного розподілу нормальних та аномальних даних. Частіше набори даних мають малу частину аномальних даних в порівнянні з нормальними, що призводить до поганої класифікації нових даних. Модель стає більш схильною класифікувати все як клас, якого було більше в навчальній вибірці;
- Навчання з частковим використанням вчителя – дані для тренування представляють лише нормальний клас. Модель, навчившись на таких даних, може визначити належність нових даних до нормального класу, а все інше як аномалії;
- Навчання без вчителя – використовується при відсутності промаркованих даних, та виходить з припущення, що аномальні дані дуже рідкісні. Аномалії визначаються на основі ступеню віддаленості від передбачених моделлю значень.

Виділяють такі методи виявлення аномалій:

- класифікація;

Метод базується на тому, що в вибірці даних наявні класи, які можна чітко віднести до нормальних; усі виміри, які не підпадають під жоден клас, вважаються аномальними. Пошук аномалій відбувається у два етапи: навчання та розпізнавання. Модель, навчившись на маркованих даних, визначає належність нових вимірів до відомих класів, та в протилежному випадку маркує як аномалію.

Найчастіше використовують такі алгоритми: наївний байєсівський підхід, метод опорних векторів, дерева рішень.

Також використовують систему нечіткої логіки, коли різниця між нормальним та аномальним розмита. В таких системах кожен екземпляр певною

мірою є одночасно нормальним та аномальним, в залежності від віддаленості під скупчення нормальних інтервалів.

- Кластерізації.

Метод передбачає групування нормальних екземплярів в кластери таким чином, щоб нормальні екземпляри були ближче до центру кластеру, а аномальні якнайдалі. Коли аномальних екземплярів багато, вони утворюють власні кластери. При цьому нормальними екземплярами вважаються ті, що формують великі та щільні кластери, а аномальними - маленькі та розрізнені. Найпопулярнішим є метод К-середніх.

Визначення аномалій в мікросервісній архітектурі потребує від методів розуміння контексту та впливу минулих подій.

1.1.1 Методи із використанням моделей Маркова

Модель Маркова - це кінцевий автомат з дискретною кількістю станів, які змінюються в дискретному часі. Таким чином, описується послідовність подій, в якій кожний наступний стан залежить від попередніх[1,2]. Моделі Маркова прийнято зображати як граф, приклад наведено на рисунку 1.2.

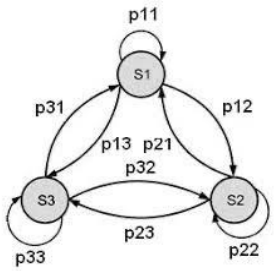


Рисунок 1.2 – Граф Марковсього процесу [16]

Модель Маркова можна описати як матрицю ймовірностей переходу від одного стану до іншого. Таким чином, знаючи поточний стан, можна отримати набір ймовірностей настання наступного. Якщо справжній наступний стан не відповідає стану з найбільшою ймовірністю, то можна стверджувати, що такий стан є аномальним[3].

Однак моделі Маркова не є надійними детекторами аномалій через те, що такі моделі є статичними, та розглядають стан без контексту та не враховують часових змін[4].

Для вирішення проблеми відсутності контексту використовують моделі Маркова вищих порядків. Модель станів будується таким чином, щоб один стан відображав декілька станів одночасно. Наприклад, якщо є два стани A та B, модель першого порядку буде мати стани A,B та 2^2 кількість ймовірностей переходу. Модель другого порядку буде мати стани A,B, AA, BB, AB, BA та 6^2. Очевидно, що кількість станів зростає як $N^2 - 1$, де N – порядок моделі. Це робить моделі високих порядків не придатними через швидке зростання розміру та моделі, та повільне зростання врахування історичних даних[5].

Отже методи на основі моделей Маркова не відходять для виявлення аномалій в мікросервісній архітектурі, оскільки не враховують минулі стани системи.

1.1.2 Методи із застосуванням рекурентних нейронних мереж

Для аналізу послідовностей в нейронних мережах використовуються рекурентні шари, які мають внутрішню пам’ять, завдяки якій модель навчається розуміти зв'язок в послідовних даних. Однак, звичайні рекурентні шари мало де використовуються через те, що внутрішній стан шару віддає найбільшу перевагу попередньому стану, не враховуючи довготривалі зв’язки.

На рисунку 1.3 зображено схему роботи рекурентного шару:

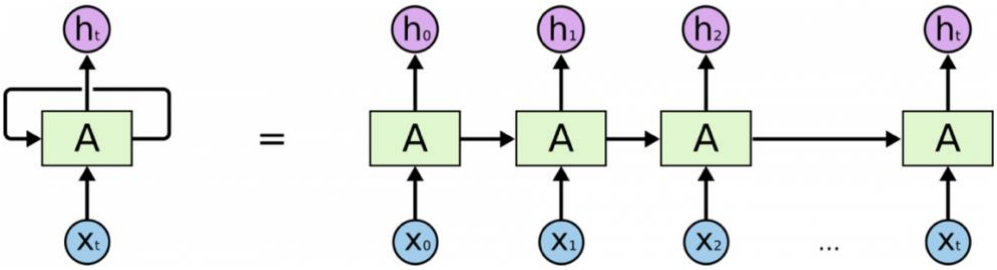


Рисунок 1.3 – Розгорнута схема рекурентного шару [19]

Як вдосконалення існуючої рекурентної моделі було розроблено LSTM – long short-term memory шари, які контролюють, як змінювати внутрішній стан для найкращого врахування нещодавніх вхідних даних та довготривалих в’язків. Це досягається за допомогою 2 кроків:

- на першому кроці минулий стан та вхідні значення проходять через так званий фільтр(сігмоідна функція, яка визначає, яку кількість даних «пам’ятати» з минулого стану, а яку забути);
- другим кроком є створення нового стану на базі пам’яті та наступного вхідного значення.[6] На рисунку 1.4 зображено схему роботи LSTM шару.

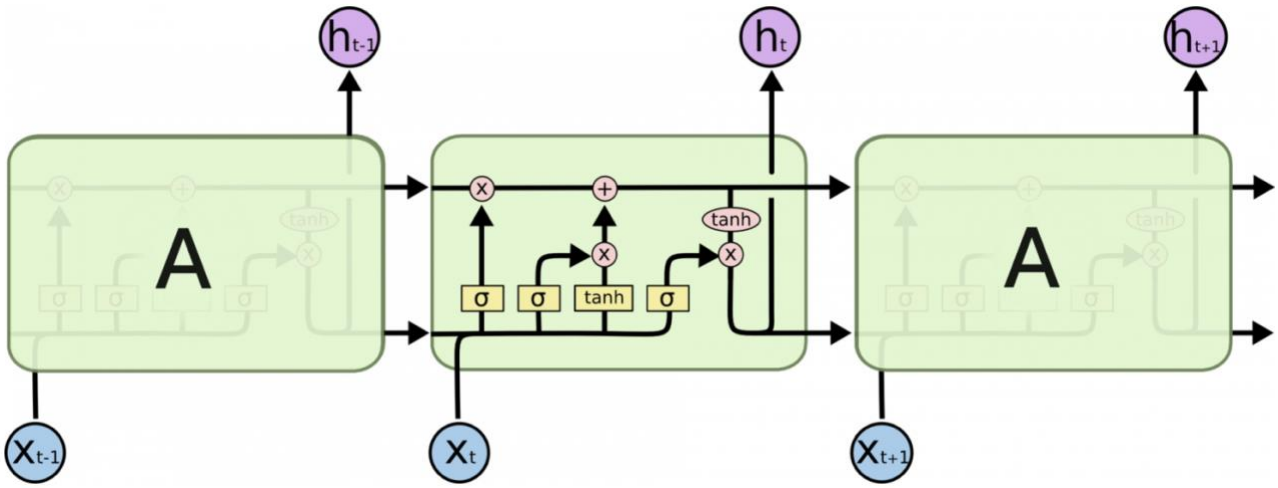


Рисунок 1.4 – Розгорнута схема рекурентного шару [19]

На відміну від моделей Маркова, дані при вході в нейронну мережу потребують підготовки(нормалізація, аугментація тощо) та процесу embedding(«упакування») в нормалізований числовий простір фіксованого розміру.

Навчання моделі відбувається зразками фіксованої тривалості, розмір яких відображає, наскільки далеко модель дивиться в минуле для генерації наступного значення.

Як правило, формується пара encoder-decoder, які працюють наступним чином: encoder перетворює вхідні часові послідовності в так зване ядро, в якому будуть закодовані всі особливості вхідного сигналу в фіксованому n-мірному

просторі, що являє собою матрицю. Decoder в свою чергу використовує ядро для створення нових значень. Таким чином, сформувавши ядро, його можна повторно використовувати в інших моделях, наприклад замість передбачення одного елементу часового ряду, передбачати декілька.

Отримані передбачення, виконані моделлю, являють собою відновлений сигнал, який використовується як еталон для порівняння з виміряними значеннями. В процесі навчання за допомогою середньої квадратичної помилки (або перехресна ентропія для бінарних вимірів) визначається порогове значення відхилення відновленого сигналу та значень тестової вибірки. Таким чином будь-який сигнал, який відрізняється від передбаченого моделлю більше, ніж на порогове значення, вважається аномальним.

Для розробки та використання нейронних мереж необхідно розуміти, що таке функція втрат, які є методи регуляризації, та як оцінити якість моделі.

Нейроні мережі виконують задачу мінімізації, для якої необхідно мати функцію, яку потрібно мінімізувати. Така функція буде мати назву функції витрат або функції помилок. Функція витрат має охоплювати всі аспекти модельованого процесу для того, щоб її мінімізація означала покращення моделі процесу. Неправильно обрана функція витрат може призвести до незадовільного результату.

Для задачі відновлення часового ряду подій, в якості функції помилки доречно використовувати бінарну крос ентропію(Binary crossentropy)[20]. Обчислюється за формулою:

$$\hat{\ell}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(p_i) + (1-y_i)\log(1-p_i)] \quad (1.1)$$

де i - індексує зразки/спостереження, j - індексує класи, а y є міткою зразка;

Під час навчання моделей може виникнути ситуація, коли модель просто «запам’ятовує» всі значення з тренувальної вибірки. Для попередження цього використовують методи регуляризації Dropout та Early Stopping[21].

Метод dropout випадковим чином виключає обрані нейрони мережі. Таким чином, внесок цих нейронів ігнорується. При прямому проходженні нейронної мережі функції активації цих нейронів випадають(dropout) та при зворотному розповсюдженні не оновлюються. Вважається, що через це інші нейрони повинні будуть компенсувати втрату, що призводить до більшої незалежності та зменшення чутливості до ваг конкретних нейронів. В результаті модель має кращу узагальнюючу характеристику та буде менш схильна до перенавчання.

Early stopping зупиняє процес навчання, якщо функція втрат починає збільшуватися замість того, щоб зменшуватися. Таким чином цей метод «захищає» модель від виходу з зони локального мінімуму, в якому спостерігались покращення. Однак, така поведінка може спричинити ситуацію, коли модель може застрягти в неоптимальному мінімумі.

1.2 Аналіз мікросервісної архітектури

В даному підрозділі розглядається мікросервісна архітектура, її переваги та недоліки.

Мікросервісна архітектура – це підхід, при якому інформаційна система будується з невеликих сервісів, які відповідають за певний бізнес-процес або сутність. Кожен мікросервіс може бути написаний на будь-який мові програмування чи з використанням будь-яких технологій. Розгортання, розробка та тестування проводиться ізольовано для пришвидшення циклу розробки мікросервісу[7].

На рисунку 1.5 зображено найпростіший приклад мікросервісної архітектури:

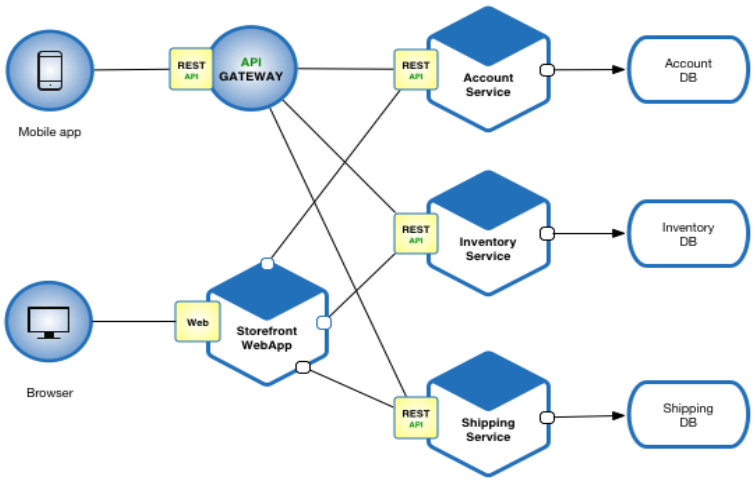


Рисунок 1.5 – Найпростіший приклад мікросервісної архітектури[17]

- API Gateway – це сервіс, який реалізує шаблон фасаду[8] для програмних інтерфейсів мікросервісів, таким чином, щоб для зовнішнього користувача здавалося, що інтерфейс реалізує єдина сутність;
- сервіси inventory, shipping, account реалізують відповідні бізнес процеси та цілком володіють ними, надаючи REST API інтерфейс;
- бази даних – Використання прямих підключень до баз даних сторонніх сервісів порушує ізольованість, оскільки йде в обхід публічного інтерфейсу. З часом це може призвести до дефектів через зміни в базі даних сервісом, який володіє цими даними, а прямі користувачів бази даних отримають помилки та не зможуть працювати. Тому є доречним мати одну базу даних для одного сервісу;
- сервіс для веб додатку – такі сервіси частіше за все не мають бізнес логіки, а лише надають інтерфейс для веб браузера, щоб не засмічувати інші сервіси специфічною для веб додатків логікою.

Переваги даної архітектури:

- швидкість – створити новий мікросервіс відносно просто та швидко. Також, мікросервіс швидко включається в роботу завдяки відокремленому циклу розробки та розгортання;
- гнучкість –ізольована розробка дає змогу швидко додавати та видаляти функціонал. Ізольоване розгортання дозволяє змінювати версії мікросервісів, не

змінюючи інших частин системи та швидко повертати стару версію в разі виявлення проблем;

- технології – для кожного мікросервісу можна обрати ту технологію, яка є оптимальною для вирішення поставленої задачі;
- масштабованість – кількість реплік одного сервісу залежить від навантаження на нього. Це дозволяє обробляти більше запитів та здешевити процес. Приклад зображений на рисунку 1.6.

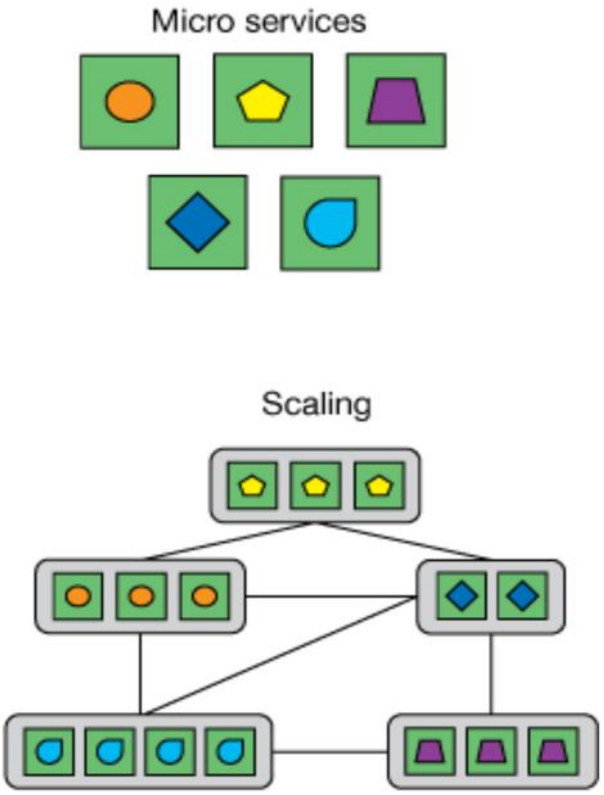


Рисунок 1.6 – Приклад масштабування [18]

Варто зауважити, що мікросервісна архітектура збільшує складність системи та складніше стає виявляти аномалії та дефекти в роботі системи.

1.3 Оркестрація мікросервісів в Kubernetes

Kubernetes - це оркестратор контейнерних застосунків, який бере на себе відповідальність автоматизованої підтримки бажаного стану кластеру

мікросервісів завдяки простій конфігурації текстових файлів. Стрімкий розвиток платформи та популярність серед розробників вплинули на її становлення як стандарту для розгортання мікросервісів [9].

Kubernetes надає наступний функціонал:

- виявлення сервісів та балансування навантаження: kubernetes може мати власний DNS сервіс, за допомогою якого мікросервіси можуть знайти одне одного не знаючи реальної IP адреси. Також, для кожного сервісу можна встановити балансувальник навантажень;
- гнучка система персистентних даних: змонтувати можна будь-яке сховище - локальне або хмарне. Kubernetes самостійно визначає, яким чином фізично змонтувати томи жорстких дисків чи S3 провайдерів;
- постійний моніторинг реального та бажаного стану кластеру, розгортаючи або знищуючи ресурси для досягнення бажаного стану;
- автоматичне розподілення обчислюваних ресурсів оптимальним способом: маючи декілька нод, поди будь розміщені в залежності від вимог щодо CPU та віртуальної пам’яті.

Для моніторингу кластеру частіше за все використовують Prometheus. Prometheus – система з відкритим вихідним кодом для моніторингу та сповіщення. Дані зберігаються в базі даних реального часу оптимізовану для часових рядів[10,11].

Kubernetes складається з нод, які утворюють кластер. Кожна нода має в собі проксі сервіс та kubelet сервіс, які відповідальні за синхронізацію стану ноди та конфігурації. Контрольна панель Kubernetes відповідає за планування створення подів та, за потреби, додавання або вилучення нод. Дані конфігурації зберігаються в базі даних etcd, також Kubernetes надає API для зміни конфігурації та взаємодію з кластером. Схема Kubernetes кластеру зображена на рисунку 1.7.

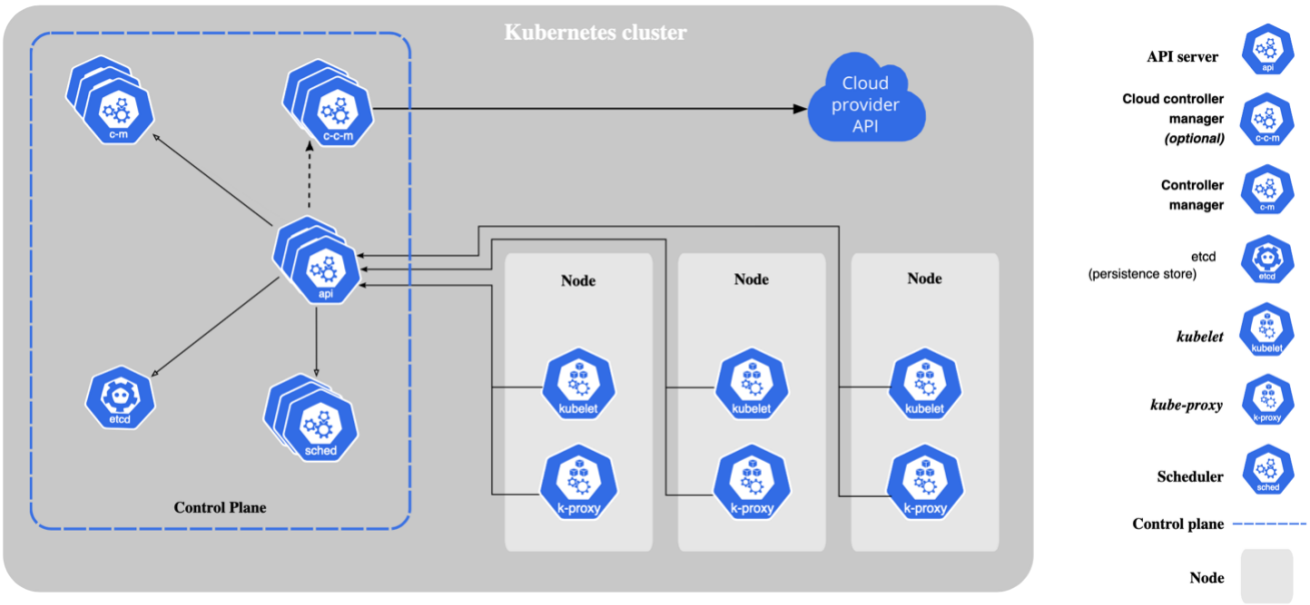


Рисунок 1.7 – Схема Kubernetes кластеру [9]

1.4 Висновок

В даному розділі було розглянуто поняття аномалій та методи їх виявлення. Більш детально описано методи Ланцюгів Маркова та рекурентні мережі. Розглянуто мікросервісну архітектуру та її властивості. Також представлено короткий опис системи оркестрації контейнерами Kubernetes.

На даний час існує багато інструментів для виявлення аномалій та моніторингу мікросервісів. Але більшість методів та програм розглядають показники та події окремо. Необхідно розробити досконаліший метод, який включатиме в себе зв'язок подій з показниками, оскільки існують аномалії, які потребують більше контексту для виявлення.

2 ПРОЕКТУВАННЯ СИСТЕМИ РОЗГОРТАННЯ З ВИЯВЛЕННЯМ АНОМАЛІЙ

В даному розділі розробляється структурна схема системи, а також аналізується кожен з її компонентів.

2.1 Аналіз вимог до системи виявлення аномалій

Аналіз вимог - це процес визначення очікуваних результатів, потреб, та побажань замовника, щоб визначити всі конфліктні вимоги та можливі проблеми до початку проектування або розробки. Аналіз вимог може включати в себе прототипування, інтерв'ю з зацікавленими особами, створення прецедентів та специфікації. Визначені вимоги повинні бути задокументовані в несуперечливій формі та зрозумілі для всіх учасників розробки.

Вимоги можуть поділятися на наступні категорії:

- вимоги споживача - фактичний опис побажань споживачів щодо цілей, середовища, ефективності системи;
- архітектурні вимоги - як системні обмеження впливають на систему;
- структурні вимоги - як структурні обмеження впливають на систему;
- поведінкові вимоги - як саме система повинна себе поводити в тій чи іншій ситуації;
- функціональні вимоги - необхідної задачі, дії, чи діяльності які мають виконуватись;
- нефункціональні вимоги - оцінка системи за її критеріями, а не поведінкою;
- вимоги продуктивності - вимоги в термінах кількості, якості, охоплення, своєчасності чи готовності.

Далі детально розглянуто функціональні та нефункціональні вимоги до системи.

Основні функціональні вимоги:

- система повинна збирати та аналізувати дані потоку подій з мікросервісу;
- система повинна збирати та аналізувати показники мікросервісу;
- система повинна підтримувати можливість роботи в будь-якій системі розгортання;
- система повинна приймати на вхід дані від зовнішніх систем та обробляти їх швидко та ефективно;
- система повинна надавати сучасні інтерфейси для інтеграції;
- бажано, щоб система могла бути розгорнута в хмарному середовищі;
- система має бути модульною для забезпечення зручного її розгортання та легкої інтеграцій з іншими системами.

Основні нефункціональні вимоги:

- система має обробляти вхідні дані в реальному часі або близькому до реального;
- кількість даних від зовнішніх систем має не перевищувати 1000 одиниць на секунду.

2.2 Вибір мови програмування

В даному розділі розглядається вибір мови програмування для реалізації системи виявлення аномалій. Важливо обрати мову, яка однаково комфортна для реалізації алгоритмів машинного навчання та реалізації back-end додатків. Можна виділити наступні вимоги до мови програмування:

- підтримка бібліотек машинного навчання;
- наявність готових фреймворків для реалізації back-end додатків;
- підтримка та активна спільнота.

Серед мов, які розглядаються в даному розділі – Python та R. Кожна з мов в цілому задовольняє вимогам. Після порівняння було обрано Python, оскільки він має більшу кількість бібліотек для машинного навчання.

2.2.1 Аналіз мови програмування Python

Далі розглядається мова програмування Python. Python – інтерпретована, динамічна типізація, об’єктно-орієнтована мова програмування високого рівня.[15] Python є дуже простою в вивчені та використанні мовою завдяки простому синтаксису та динамічній типізації. Кількість бібліотек для будь-яких потреб росте з кожним днем, адже активно підтримується спільнотою.

Де-факто Python є стандартом для машинного навчання та розробки нейронних мереж в світі. Завдяки таким бібліотекам як pyTourch та TersorFlow немає потреби використовувати будь-яку іншу мову чи фреймворк.

Основні переваги даної мови:

- проста у використанні;
- активна спільнота ;
- дуже широкий набір бібліотек для роботи з машинним навчанням.

Недоліки:

- повільна робота;
- динамічна типізація;
- інтерпретована мова.

2.2.2 Аналіз мови програмування R

R – інтерпретована, динамічна мова програмування. Найчастіше використовується для статистичних обчислень та аналізу даних. Має широкий набір бібліотек та надійну спільноту. Також надає можливість підключати бібліотеки на C та C++ для задач, які потребують значної швидкості роботи[12].

Робота з даними в R значно простіше, ніж в інших мовах програмування, оскільки такі об’єкти як вектори, матриці, тензори включені в мову на рівні синтаксису, що дозволяє запобігти нагромаджень коду для вирішення типових задач обходу матриць, перемноження векторів та інші.

Основні переваги даної мови:

- проста у використанні та роботи з форматами даних;
- швидкий цикл редагування-тестування-відлагодження;
- простота роботи з даними;
- наявність спеціалізованої IDE.

Недоліки:

- динамічна типізація;
- відсутня підтримка найсучасніших методів машинного навчання;
- швидкість роботи.

2.3 Тестовий об’єкт

В якості тестового об’єкту було розроблено декілька простих мікросервісів. Система складається з web сервісу, api сервісу та сервісу, який збирає метрики та потік подій від api та web.

На рисунку 2.1 зображено схему системи.

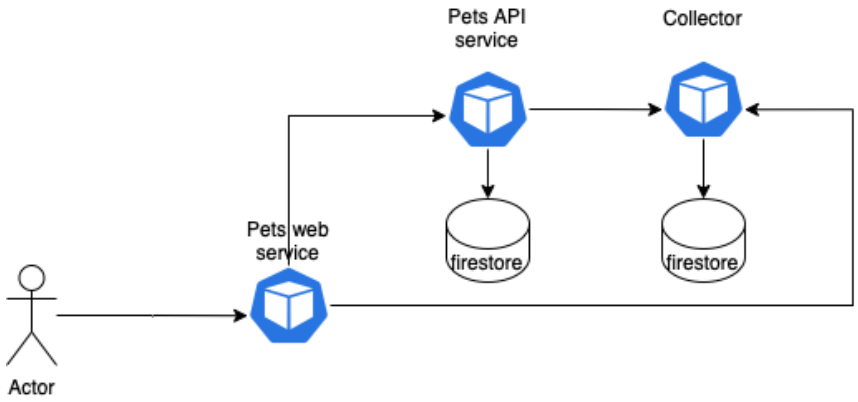


Рисунок 2.1 – Схема мікросервісів

Кожен мікросервіс збирається в Docker контейнер, на рисунку 2.2 зображено приклад Dockerfile для API сервісу. Контейнер виконує побудову сервісу, відкриває 8080 порт, та запускає відповідний бінарний файл сервісу.

```
FROM golang:1.17.0-alpine3.13
RUN mkdir /app
ADD . /app
WORKDIR /app
EXPOSE 8080
RUN cd api && go build -o main .
CMD ["/app/api/main"]
```

Рисунок 2.2 – Приклад Dockerfile для API сервісу

Для розгортання контейнерів сервісів в Kubernetes було розроблено helm chart для кожного сервісу, на рисунку 2.3 зображено файлову структуру. Кожен сервіс складається з наступних компонентів:

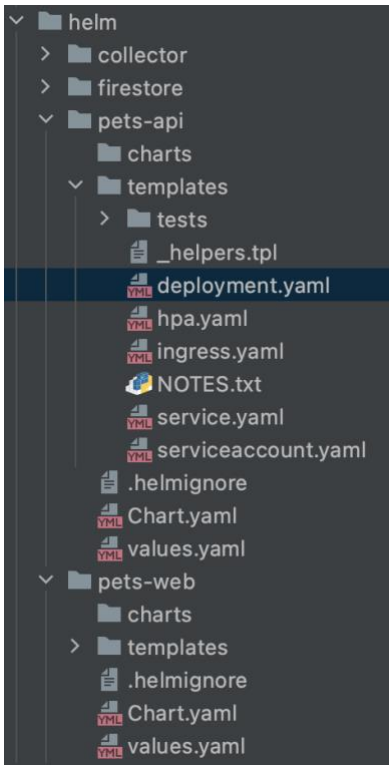


Рисунок 2.3 – Файлова структура helm chart сервісів

- deployement (розгортання) – описує створення подів для Kubernetes, визначення імені та тегу контейнеру, змінні середовища, кількість реплік, проби працездатності;

- hpa (Автоматичне горизонтальне масштабування) – описує правила збільшення кількості реплік в залежності від кількості спожитих мікросервісом ресурсів;
- ingress – описує мережеві налаштування для доступу до сервісу з зовнішньої мережі;
- service – описує мережеві налаштування для доступу сервісу з внутрішньої мережі;
- service account – описує створення сервісного користувача в системі Kuberenets для конкретного мікросервісу.

Функціонал системи являє собою простий CRUD сервіс, який надає веб сторінку для перегляду тварин та додавання їх в базу даних. В якості бази даних було обрано Firestore через свою простоту та швидкість роботи.

На рисунку 2.4 зображено веб сторінка web сервісу.

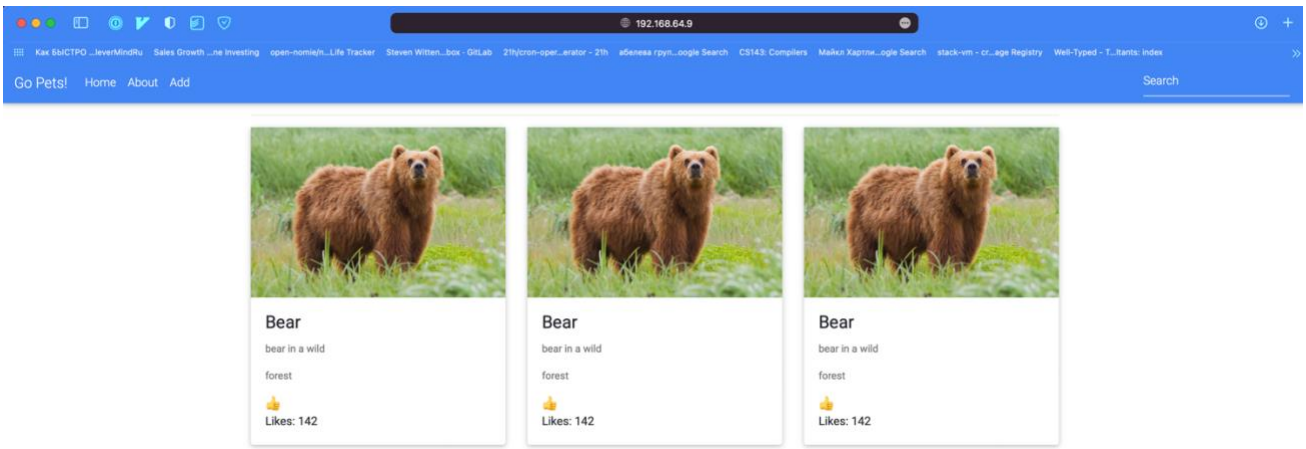


Рисунок 2.4 – Веб сторінка перегляду тварин

API сервіс має swagger сторінку, де перераховані всі методи. Також swagger надає можливість виконати запит з браузера, що спрощує тестування.

На рисунку 2.5 зображено swagger панель API сервісу.

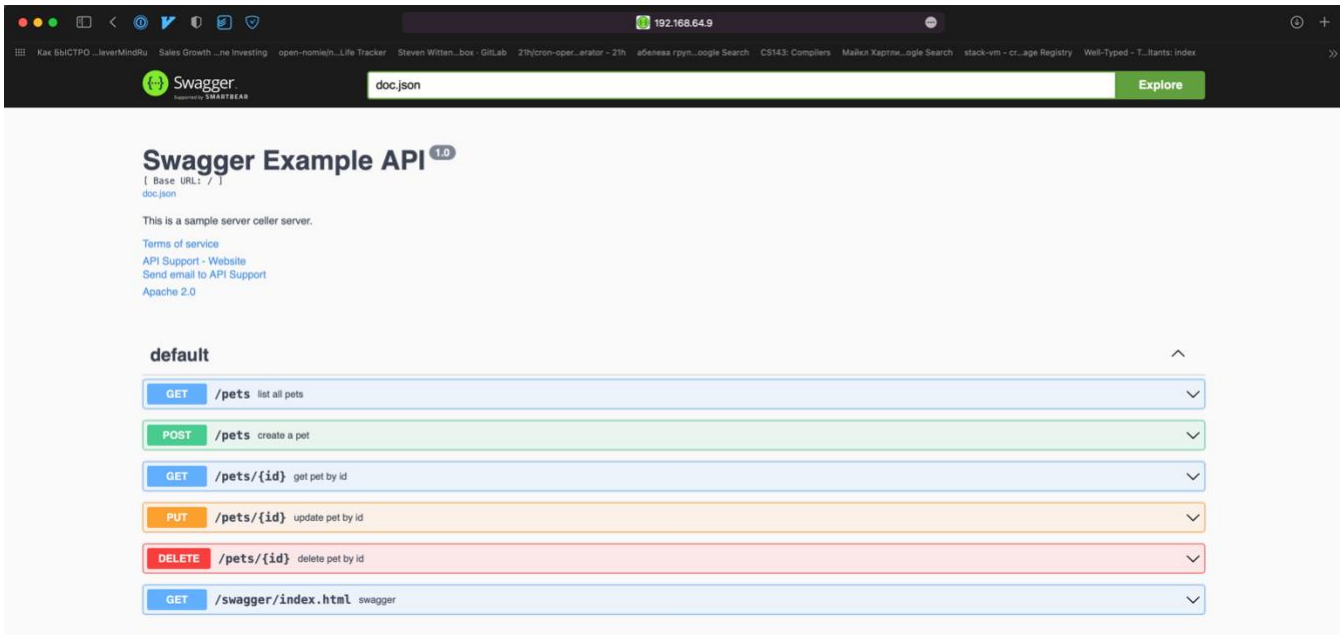


Рисунок 2.5 – Swagger панель API сервісу.

В якості інструменту тестування було обрано Locust, який надає можливість просто реалізувати тести для навантаження та імітації користувацької активності[14].

На рисунку 2.6 зображено результат тестування на якому видно, що система здатна обробити 45 запитів в секунду. Тест проводився з 1000 конкурентних запитів.

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/	737	0	40000	44000	34738	114	57237	107175	8.8	0
POST	/add	653	0	47000	61000	42148	211	65617	107175	3.1	0
GET	/pets	1076	0	38000	43000	34226	100	50790	28110	14	0
POST	/pets	2649	0	3900	12000	5192	36	22943	300	18.2	0
	Aggregated	**5115**	**0**	**9800**	**45000**	**20275**	**36**	**65617**	**35193**	**44.1**	**0**

Рисунок 2.6 – Статистика locust тестів.

2.3.1 Налаштування тестового середовища

Для розгортання використано Kubernetes та minikube[13]. Створено кластер з 1 нодою 8 CPU та 10 gb пам’яті. В кластері розгорнуто тестові мікросервіси та інструмент тестування Lotus.

У результаті використання Kubernetes кластеру значно збільшується швидкість розробки та перевірки змін завдяки простому механізму розгортання нових версій, який представляє з себе побудову нового зображення контейнера та просту зміну версії в yaml файлі налаштувань helm chart.

2.4 Проектування алгоритму виявлення аномалій

2.4.1 Проектування архітектури нейронної мережі передбачення показників та подій

Для аналізу потоку подій та метрик використовується алгоритм навчання без вчителя, схема нейронної мережі зображеної на рисунку 2.7.

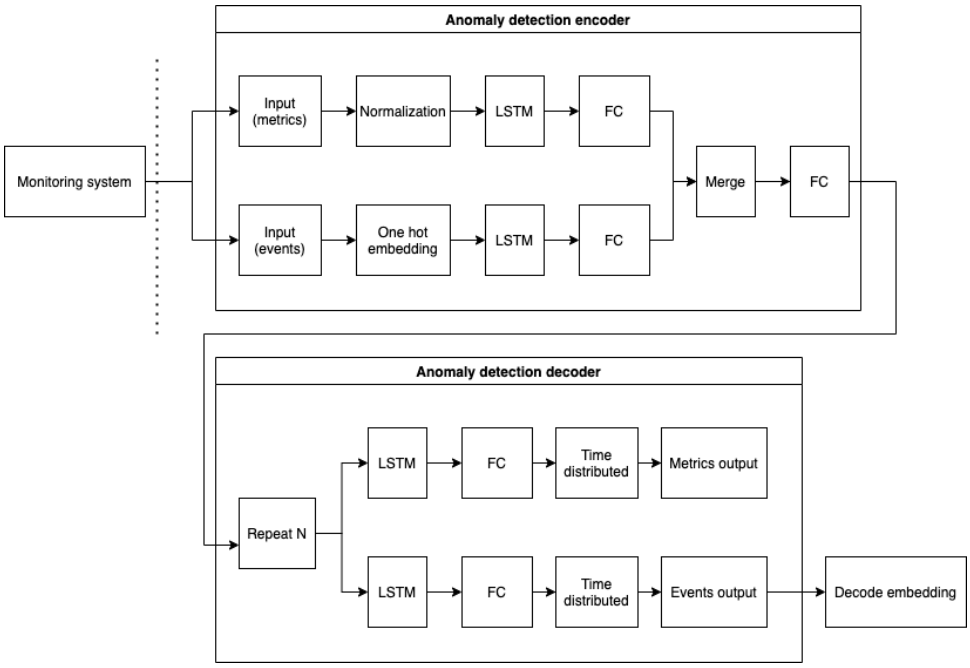


Рисунок 2.7 – Схема нейронної мережі

На вхід подається часова послідовність потоку подій та метрик. Далі необроблені дані проходять через шари «упакування» в числовий нормалізований простір. Упаковані дані проходять через рекурентні шари, на виході яких отримуємо матрицю фіксованого розміру, яку можна передати в повнозв'язні шари та конкатенацію для отримання фінальної матриці. Після

цього відбувається відновлення сигналу. В залежності від кількості відновлених значень архітектуру можна спростити, прибравши повторювач та додаткові LSTM шари.

2.4.2 Алгоритм роботи системи розгортання з виявленням аномалій

UML діаграма послідовностей наведена на рисунку 2.8.

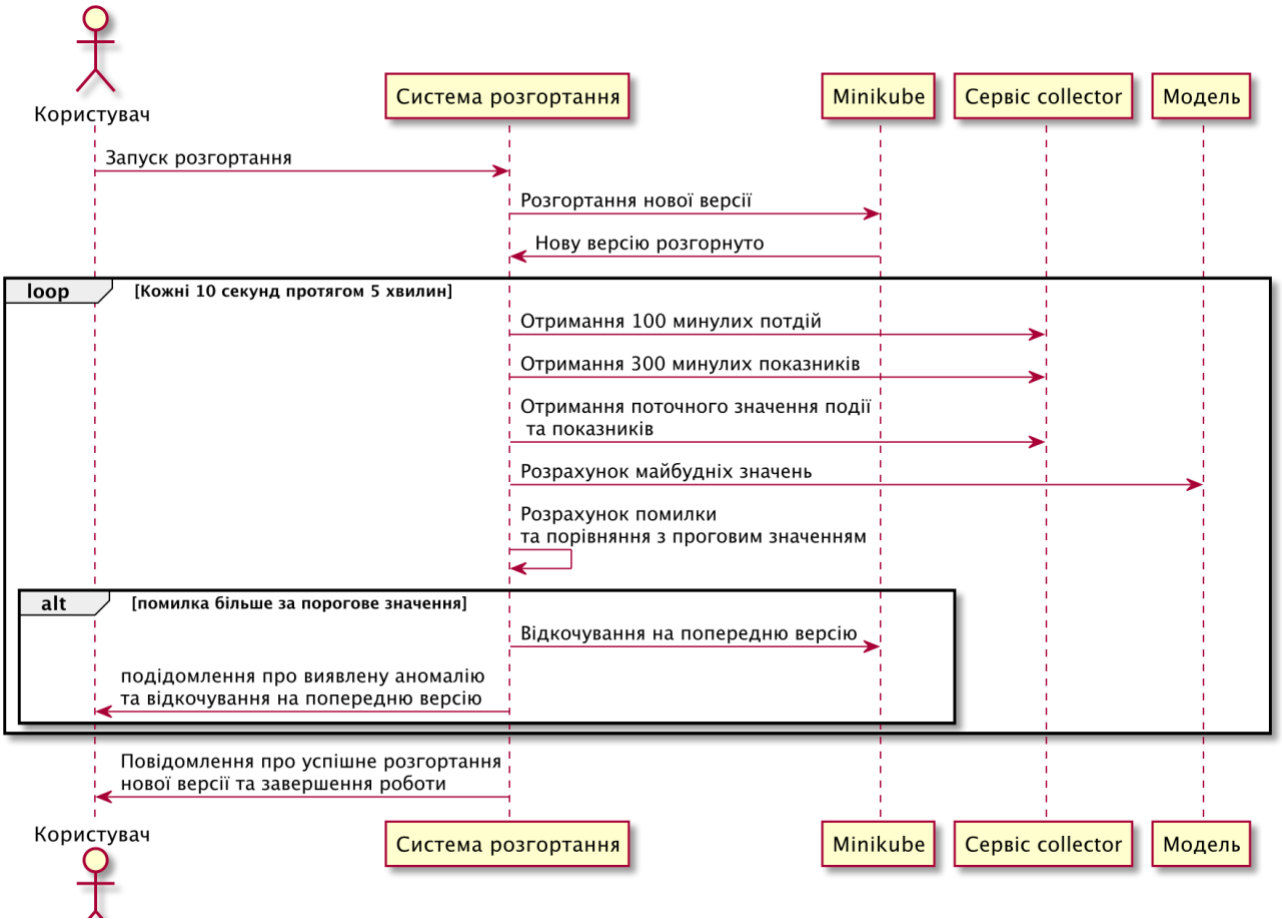


Рисунок 2.8 – UML діаграма послідовностей алгоритму розгортання.

При виявленні будь-яких помилок, що унеможливлюють процес виявлення аномалій, система розгортання повертає попередню версію сервісу та закінчує роботу. В разі відсутності необхідної кількості минулих значень система буде очікувати їх появи.

2.5 Висновки

В даному розділі були описані вимоги до системи. Алгоритм виявлення функціональних аномалій в роботі мікросервісу під час розгортання повністю відповідає поставленим вимогам. Також розроблено схему системи та проаналізовано варіанти мов програмування та вибрано Python для реалізації системи.

Суттєвим покращенням методу може бути реалізація перетворення класів подій в н-вимірний простір. Після цього додавання нових класів подій потребуватиме лише розрахунок одного нового вектору в н-вимірному просторі замість повного перенавчання моделі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ РОЗГОРТАННЯ

В даному розділі описується аналіз тренувальних даних для моделі, реалізація моделі, та пошук порогового значення для класифікації аномалій.

3.1 Засоби розробки

Розробка проводилась обраною в розділі 2.2. мовою програмування Python. Для обробки даних та реалізації моделі нейронної мережі використано середовище розробки Google Colab.

Google Colab – це безкоштовний (також має платну версію) браузерний застосунок, який надає доступ для виконання Python коду на віртуальних машинах з наявним GPU, який пришвидшує навчання та використання нейронних мереж та інших алгоритмів машинного навчання, які потребують значної кількості матричних розрахунків[23].

Google Colab надає всі необхідні пакети для розробки нейронних мереж та не потребує попереднього налаштування чи самостійного управління віртуальними машинами. На рисунку 3.1 зображено інтерфейс Google Colab.

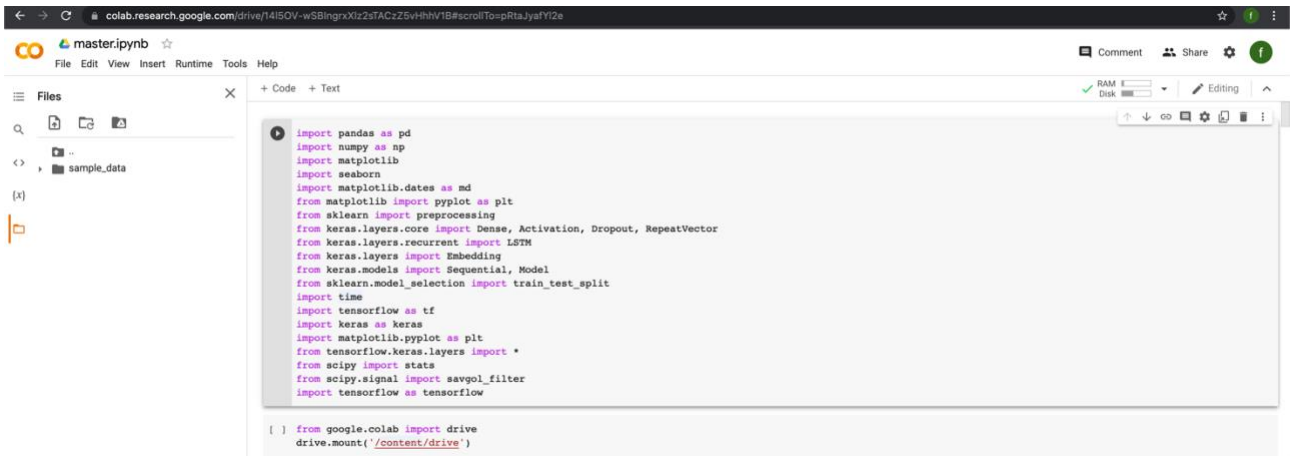


Рисунок 3.1 – Інтерфейс Google Colab

Для реалізації моделі нейронної мережі обрано популярну зв'язку бібліотек Tensorflow та Keras. Tensorflow – це бібліотека з відкритим вихідним кодом, яка надає інструменти для реалізації алгоритмів машинного навчання та має значний фокус на глибинне навчання. В свою чергу Keras надає простий та зрозумілий інтерфейс для створення нейронних мереж, виключаючи потребу реалізації низькорівневих речей таких, як розрахунок функції втрат та застосування розрахованого значення для оберненого проходу по нейронній мережі.

Також для розробки системи розгортання та тестового об'єкту було використано IntelliJ IDE, яка надає широкий спектр інструментів, таких як: компілятор для багатьох мов, редактор тексту, систему контролю версій, додатки для роботи з Kubernetes та інші.

3.2 Тренувальні дані

Тренувальні дані було отримано під час роботи тестового об'єкта, який викликався за допомогою тестового інструменту locust, з змінними в часі налаштуваннями кількості паралельних запитів та користувачів для того, щоб симулювати реальний трафік. Тести виконують запити як до API, так і до web сервісу, створюючи та отримуючи створені екземпляри (рис. 3.2).

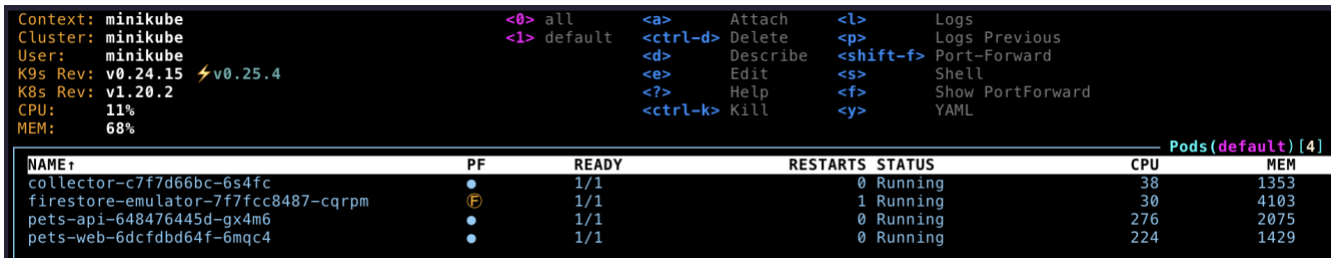


Рисунок 3.2 – Запущені тестові мікросервіси в minikube

Дані показників попередньо нормалізуються. Дані подій проходять процес кодування one hot векторізація.

Додатково всі монотонно зростаючі показники було перетворено в різницю між замірами, адже за допомогою такого перетворення моделі показують значне покращення в точності.

Так як для передбачення майбутніх значень модель повинна мати дані з минулого, з рядів подій та показників були сформовані порції по 100 та 300 минулих значень відповідно. Таку кількість минулих значень було обрано експериментально. Чим більша кількість минулих значень - тим точніша модель. Однак, при збільшенні кількості минулих значень збільшується час очікування даних для початку передбачення. На рисунку 3.3 зображено графіки частини тренувальних даних.

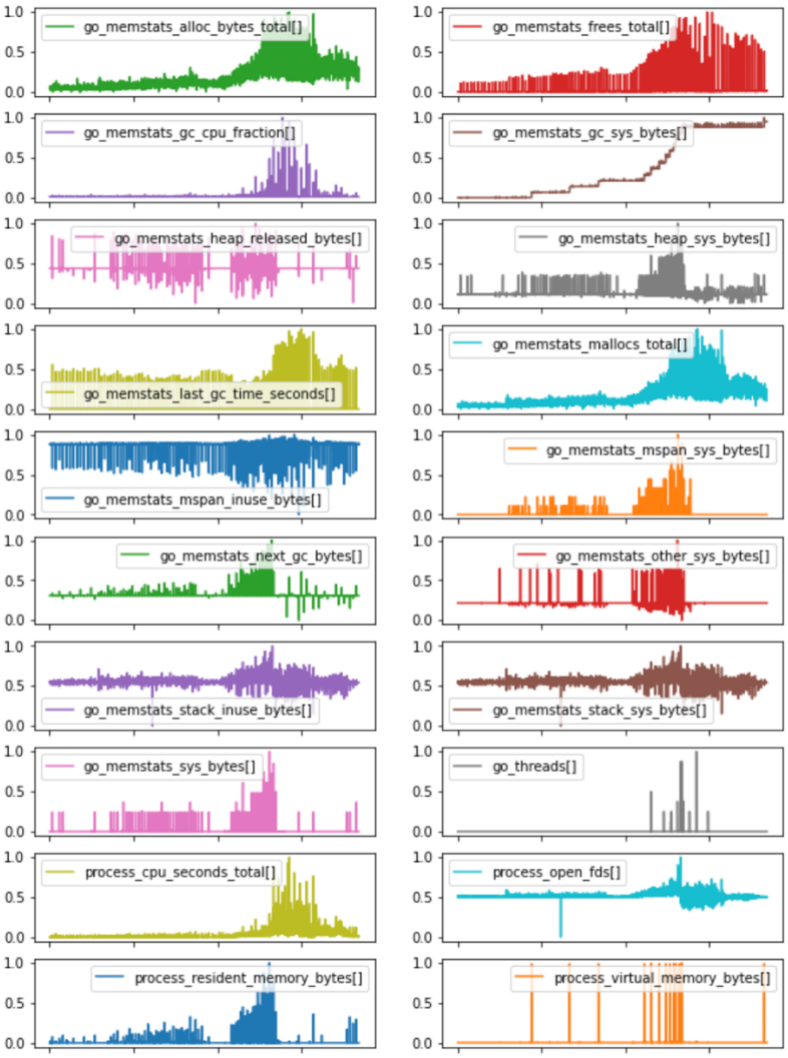


Рисунок 3.3 – Графік тренувальних даних(вісь X – час, вісь Y – нормовані значення)

3.3 Реалізація моделі передбачення показників та подій

В розділі 2.4. описано загальну схему реалізації детектора аномалій з використанням LSTM шарів нейронної мережі. На основі даного алгоритму розроблено архітектуру моделі нейронної мережі, яка поєднує потік подій та показники роботи мікросервісу. В додатку А зображено візуалізацію моделі на базі фреймворку Keras.

Модель складається з двох паралельних входів та двох виходів окремо для потоку подій та показників. Енкодер потоку подій, вхід якого позначено “metrics” має два шари: LSTM та повно-зв’язний шар, який забезпечує правильність розміру для формування ядра. Енкодер показників, позначено “events”, також має два LSTM шари, однак також має dropout для запобігання перенавчання та збільшення стабільності моделі. Іншою суттєвою різницею є функції втрат. Для подій це бінарна кроссентропія, бо події закодовані як one not вектор, та для показників середня квадратична похибка. Вагові коефіцієнти функції помилки є однаковими для досягнення рівномірного навчання.

Навчання відбувається порціями(batch) по 100, оскільки «пам’ять» LSTM зберігається для однієї порції. Збільшення значення порції навчання може призвести до небажаної чутливості LSTM шарів. Також, з тренувальної вибірки взято 10% для валідації після кожної епохи навчання.

Модель має значну кількість мета параметрів таких, як: кількість, розмір виходу шарів LSTM, Dense, відсоток погашень активація в шарі Dropout, функції активації. Всі мета параметри підібрані експериментальним шляхом таким чином, щоб досягти найменшої помилки під час тестування моделі. Загальний обсяг параметрів для тренування моделі(вагові коефіцієнти та зміщення) становить 4,933,006.

На рисунку 3.4 зображено код моделі.

```
losses = {
  "category_output": "binary_crossentropy",
  "metrics_output": "mse",
}
lossWeights = {
    "category_output": 1.0,
    "metrics_output": 1.0
}

input1 = keras.layers.Input(shape=(10, 47), name="events")
input2 = keras.layers.Input(shape=(300, 7), name="metrics")

inputx = LSTM(512, input_shape=(10, 47), return_sequences=True)(input1)
inputx = LSTM(256, return_sequences=False)(inputx)
inputx = Dense(1000)(inputx)


inputy = LSTM(128, input_shape=(300, 7), return_sequences=True)(input2)
inputy = Dropout(0.2)(inputy)
inputy = LSTM(128, return_sequences=False)(inputy)
inputy = Dense(1000)(inputy)


input = Concatenate()([inputx, inputy])

x1 = Dropout(0.2)(input)
x1 = Dense(100)(x1)
x1 = Dense(7)(x1)
x1 = Activation('linear', name="metrics_output")(x1)


x2 = Dense(1000, activation='relu')(input)
x2 = Dense(200, activation='relu')(x2)
x2 = Dense(47, activation='sigmoid', name="category_output")(x2)


model = Model(inputs=[input1, input2], outputs=[x1, x2])
model.compile(optimizer='adam', loss=losses, loss_weights=lossWeights, metrics=["accuracy"])
```

Рисунок 3.4 – Код моделі на базі фреймворку TensorFlow та Keras

3.4 Результат тренування моделі

Тренування проходило порціями(batch) по 1000 з 4-ма епохами. В результаті тренування досягнуто значення загальної функції втрат 0.0174, ф-цій втрат подій 0.0009, та ф-цій втрат показників 0.0164. На рисунку 3.5 зображено процес зменшення загальної ф-цій втрат в залежності від епохи.

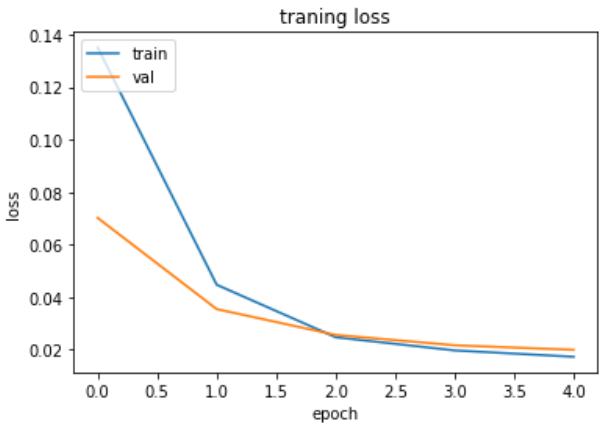


Рисунок 3.5 – Графік функції втрат навчання моделі

На рисунках 3.6-3.7 Зображено тенденцію збільшення точності моделі під час навчання, та досягнуто точність >0.9, що є достатньо високою точністю.

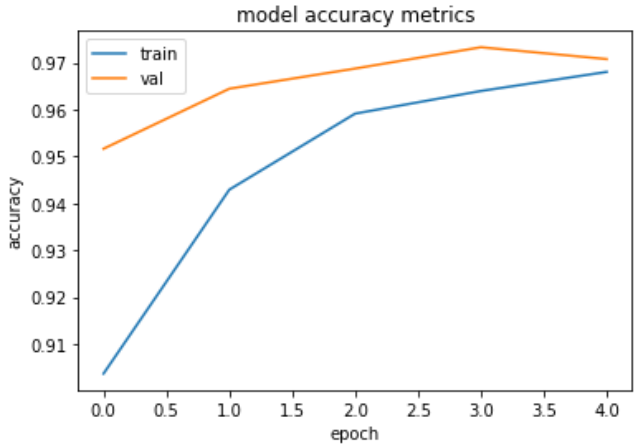


Рисунок 3.6 – Точність передбачення показників

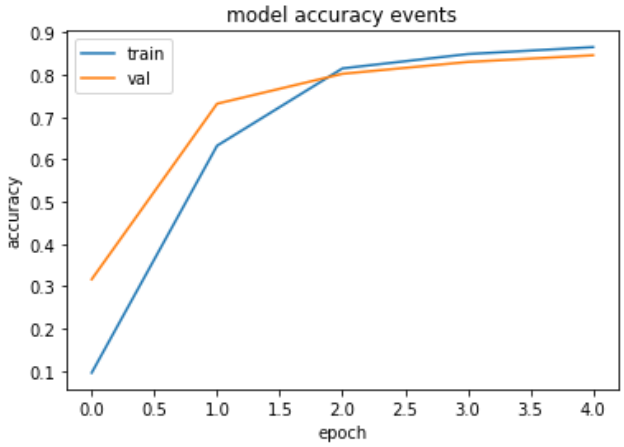


Рисунок 3.7 – Точність передбачення подій

3.5 Виявлення аномалій показників та подій

Маючи готову модель, можна розрахувати порогове значення аномальних даних. Порогове значення обирається наступним чином: модель передбачує тренувальні дані, розраховується середня квадратична помилка для показників роботи та бінарна кроссентропія для потоку подій. Розрахунок відбувається для кожного елементу даних: з отриманого ряду помилок можна побудувати гістограму, по якій буде обиратися порогове значення. Чим менше порогове значення, тим більшу кількість даних буде класифіковано як аномальні. Чим більше порогове значення, тим більше аномалій може бути пропущено. Підбір порогового значення є суто експериментальним процесом, а обране значення може бути змінене за потреби. На рисунках 3.8-3.9 зображено гістограми помилок для тренувальної та тестової вибірки. Можна побачити, що більшість значень знаходиться в діапазоні від 0 до 0,05, однак порогове значення 0,05 може призвести до великої кількості хибно позитивних класифікацій.

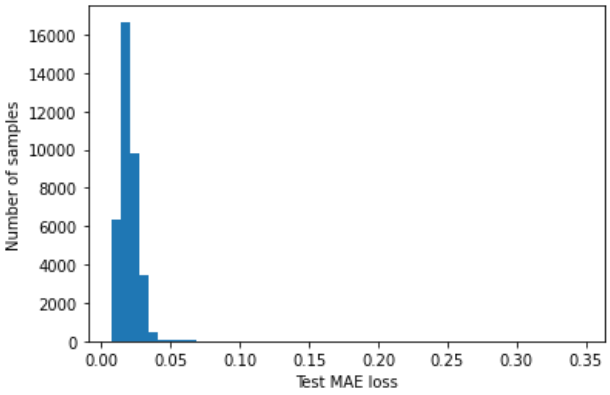


Рисунок 3.8 – Гістограма помилок тестової вибірки

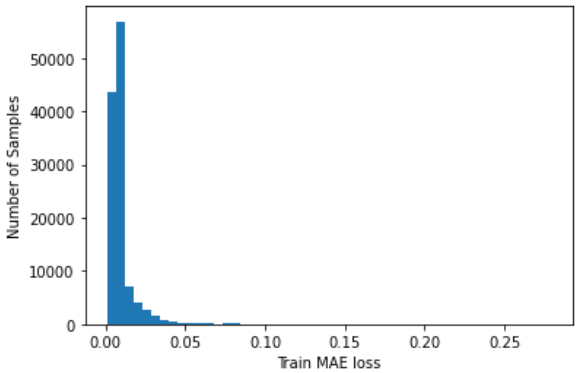


Рисунок 3.9 – Гістограма помилок тренувальної вибірки

Було обрано порогове значення 0.3 для подій та 0.68 для показників. На рисунку 3.10 зображені знайдені аномалії на тестовій вибірці.

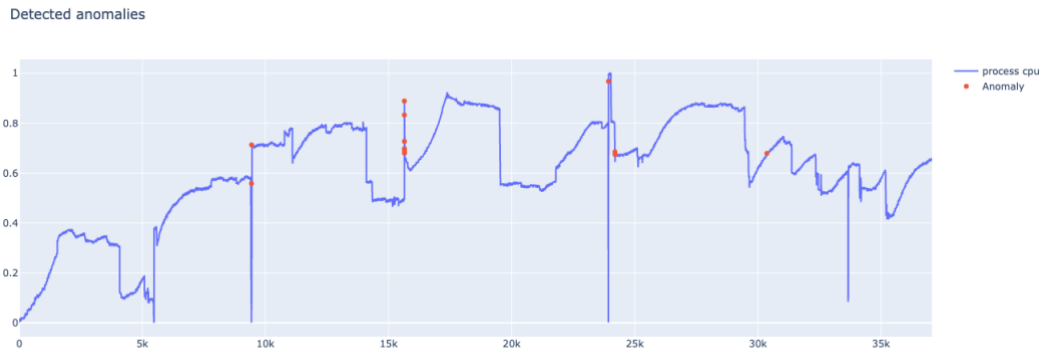


Рисунок 3.10 – Виявлені аномалії в тестовій вибірці

3.6 Тестування системи розгортання з виявленням аномалій

В ході роботи було розроблено інструмент командного рядка, який отримує на вході назву сервісу, шлях до helm chart, та версію. Під час роботи програми виконується розгортання заданої версії та починається пошук аномалій

протягом заданого часу. Отримані дані з сервісу collector проходять такий самий процес попередньої обробки, як і навчальні дані, а саме нормалізація та перетворення в one hot кодування для подій. В разі наявності аномалії виконується операція відкочування до старої версії. Якщо аномалій не було знайдено, нова версія залишається в кластері та програма завершує роботу.

На рисунку 3.11 зображено роботу програми та відкочування через наявну аномалію.

```
deploying...
b'Release "pets-api" has been upgraded. Happy Helming!\nNAME: pets-api\nLAST DEPLOYED: Mon M
 running these commands:\n  export NODE_PORT=$(kubectl get --namespace default -o jsonpath='
 jsonpath="{.items[0].status.addresses[0].address}")\n  echo http://$NODE_IP:$NODE_PORT\n'
new version has been deployed
0.47240223188418895 0.47771477556996267
no anomalies so far
0.1837684471781055 0.4863747305536491
no anomalies so far
0.38586958587014425 0.25695126215222747
no anomalies so far
0.07896487976540811 0.2628732402123005
no anomalies so far
1.0078916757064462 0.351575212030125
Anomaly detected rolling back to previous version
b'Rollback was a success! Happy Helming!\n'
```

Рисунок 3.11 – Робота програми розгортання

3.7 Порівняння комбінованого та некомбінованого методу виявлення аномалій

Для порівняння комбінованого та некомбінованого методу вибрано наступні критерії:

- якість навчання та швидкість сходження моделі;
- виявлення аномально високого навантаження;
- виявлення логічної помилки.

Архітектуру комбінованої моделі розглянуто в розділі 3.3. Схема некомбінованих моделей наведена на рисунку 3.12. Це ті самі моделі, але без конкатенації в єдине ядро.

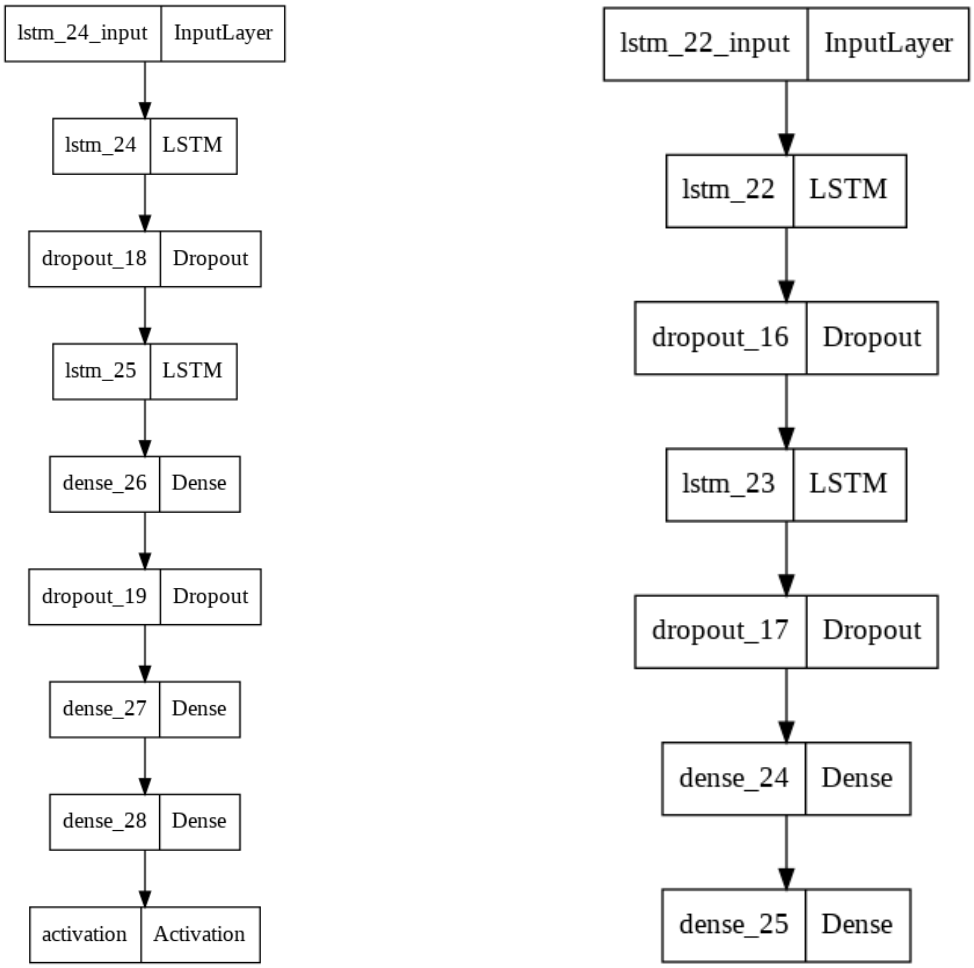


Рисунок 3.12 – Схема не комбінованих моделей

- Виявлення аномально високого навантаження;

Для цього експерименту було встановлено 1000 паралельних воркерів в locust налаштуваннях. Було проведено 10 запусків, в усіх 10 комбінована та некомбінована моделі успішно виявили аномалію.

- Якість навчання та швидкість сходження моделі;

Для перевірки цього критерію було зібрано вибірку даних під час роботи мікросервісів та запущеним тестом продуктивності locust. Для уникнення фактору перенавчання розмір вибірки невеликий. В таблиці 3.1. наведено порівняння показників навчання моделей.

Таблиця 3.1 – Порівняння якості навчання моделей

Епоха	Комбінована Модель			Некомбінована модель подій		Некомбінована Модель показників	
	Загальна ф-ція втрат	Точність подій	Точність показників	Ф-ція втрат	Точність подій	Ф-ція втрат	Точність Показників
1)	0,065	0,899	0,332	0,204	0,786	0,012	0,413
2)	0,051	0,898	0,284	0,078	0,882	0,007	0,274
3)	0,043	0,906	0,414	0,049	0,934	0,007	0,179
4)	0,039	0,916	0,357	0,038	0,941	0,007	0,138
5)	0,036	0,909	0,462	0,033	0,938	0,007	0,245

В результаті можна сказати, що комбінована модель однаково якісно передбачує події як і некомбінована, однак краще за некомбіновану передбачує показники.

- Виявлення логічної помилки.

В ході експерименту в коді API сервісу було використано цикл, який генерує 50 подій про неуспішну валідації запиту, симулюючи таким чином невірну логіку обробки виключних ситуацій. Експеримент було проведено 10 разів, в таблиці 3.2-3. Зображено матриці невідповідності для комбінованої та некомбінованої моделі.

Таблиця 3.2 – Матриця невідповідності комбінованої моделі

Дійсність / Передбачення	Аномалія	Не аномалія
Аномалія	12	3
Не аномалія	6	6

Таблиця 3.3 – Матриця невідповідності некомбінованої моделі

Дійсність / Передбачення	Аномалія	Не аномалія
Аномалія	9	7
Не аномалія	3	8

Для оцінка точності класифікації використовується F-score:

- precision;

$$Precision = \frac{TP}{TP + FP} \text{ [22]}, \quad (3.1)$$

де TP – кількість істинно позитивних значень; FP – кількість хибно позитивних значень;

- recall;

$$Recall = \frac{TP}{TP + FN} \text{ [22]}, \quad (3.2)$$

де TP – кількість істинно позитивних значень; FN – кількість хибно негативних значень;

- F-score – загальний критерій якості, середнє гармонійне precision та recall.

$$f\text{-}score = 2 * (\frac{precision * recall}{precision + recall}) \text{ [22]}, \quad (3.3)$$

де precision – влучність; recall – повнота;

В результаті експерименту можна побачити, що некомбінована модель має більшу схильність до хибно позитивних ніж комбінована, та загалом обидві

моделі мають значну кількість хибних передбачень. Значення F-score для комбінованої моделі вище ніж у некомбінованої. Це означає, що комбінована модель краще справляється з виявленням аномалій.

В таблиці 3.4. наведені значення критерію якості моделей.

Таблиця 3.4 – F-score моделей

	Precision	Recall	F-score
Комбінована	0,80	0,67	0,73
Не комбінована	0,56	0,75	0,64

3.8 Висновки

В даному розділі було описано дані навчання моделі, попередню обробку даних, реалізацію та тренування моделі, та підбір порогового значення аномалій. Продемонстровано роботу програми розгортання.

В результаті тестування можна сказати, що даний метод виявлення аномалій під час розгортання мікросервісів працездатний, але є нестійким та потребує постійного навчання. Вибір порогових значень не має чіткого критерію та незахищений від великої кількості хибно позитивних спрацювань.

Різниця між комбінованою та некомбінованою моделлю наявна, F-score для комбінованої моделі вища, що каже про більшу якість моделі. Отже, можна зробити висновок, що комбінація подій та показників в одну модель має позитивний вплив на якість класифікації аномалій.

4 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ

4.1 Опис ідеї проєкту

Таблиця 4.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка SAAS який надає користувачам нотифікації в разі виявлення аномалій в роботі мікросервісу на основі даних які надає користувач	1. Виявлення дефектного коду	Дозволяє зменшити вплив дефектного коду на інформаційну систему та уникнути порушень в роботі, що можуть призвести до фінансових втрат
	2. Система моніторингу	Своєчасне отримання інформації, що до стану системи

Таблиця 4.2 – Опис ідеї стартап-проєкту

№	Техніко-економічні характеристики ідеї	Продукція конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проєкт	Конкурент 1	Конкурент 2	Конкурент 3			
1	Точність роботи	Низька	Середня	Висока	Висока	-		
2	Наявність інтернету	+	-	-	+		-	
3	Вартість	Низька	Середня	Висока	Середня			+
4	Швидкість	Висока	Середня	Висока	Висока		-	

4.2 Технологічний аудит ідеї проєкту

Таблиця 4.3 – Технологічна здійсненність ідеї проєкту

№	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	SAAS	Мова програмування Python	Наявна	Доступна Безкоштовна
2		Фреймворк для створення моделей машинного навчання tensorflow	Наявна	Доступна Безкоштовна
3		Хмарний постачальник GCP	Наявна	Доступна Платна
Обрана технологія реалізації ідеї проєкту мова програмування Python, розгортаня в хмарному середовищі Google Cloud Platform, який хоч і платний, але має безкоштовні ліміти, для реалізацій алгоритмів машинного навчання обрано фреймворк tensorflow				

Висновок: технологічна реалізація продукту – можлива

4.3 Аналіз ринкових можливостей запуску стартап-проєкту

Таблиця 4.4 – Попередня характеристика потенційного ринку

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн./ум.од	59,83 млрд/рік
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Немає
5	Специфічні вимоги до стандартизації та сертифікації	ISO/IEC 27001
6	Середня норма рентабельності в галузі або по ринку, %	20%

Висновок: відсутність обмежень, помірної складності сертифікація, та середня рентабельність вказують на те, що створення стартапу є вигідним.

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проєкту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці цільових груп клієнтів	Вимоги споживачів до товару
1	Стрімке зростання розміру мікросервісних систем	Компанії які розробляють власне програмне забезпечення	Формат даних моніторингу сервісів	Програмний інтерфейс для інтеграції, надійність роботи, можливість збільшувати кількість даних для аналізу
2	Значне ускладнення та зростання кількості програмного коду	Галузі програмного забезпечення з підвищеною складністю систем та високими вимогами надійності, такі фінтех, та мед. установи	Стандарти безпеки	

Таблиця 4.6 – Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Поява великого конкурента	Наявність конкурента який має значно більше ресурсів для навчання кращих моделей	Продаж компанії конкуренту, Публікація коду в відритий доступ та вихід з ринку
2	Не робоча економічна модель	Закінчення грошей та невеликий прибуток від продажів	Зменшення собівартості, Зменшення штату робітників, Пошук партнера для залучення інвестування

Таблиця 4.7 — Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Удосконалення методів машинного навчання	Поява нових алгоритмів та інструментів для реалізації продукту	Інвестування в нові технології, розробка нових версій продукту, Створення науково-дослідницького відділу
2	Позитивна реакція на продукт у клієнтів	Збільшення зацікавленості та довіри до продукту	Збільшення витрат на маркетинг, Спрощення купівлі послуг

Кінець таблиці Таблиця 4.7

№	Фактор	Зміст можливості	Можлива реакція компанії
3	Використання готових рішень компанії Google	Спрощення кодової бази за допомогою використання готових рішень для мікросервісної архітектури та машинного навчання	Додаткові витрати на готові рішення, спрощення та пришвидшення процесу розробки та розгортання

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	Тип конкуренції: монополістична	Товар є аналогом уже наявних на ринку від інших компаній із деякими відмінностями функціоналу; Невеликі бар'єри для входу в галузь; Кількість фірм конкурентів невелика;	Розробка продукту із ширшим та зручнішим функціоналом, ніж у фірм конкурентів; Ціна не має бути значно вищою і має корелювати із ціною фірм-конкурентів; Зробити функціонал з різним рівнем доступу, дати на вибір декілька варіантів;
2	Рівень конкурентної боротьби: світовий	Аналоги були розроблені інтернаціональними фірмами;	Продукт має бути доступний для якнайбільшої кількості країн; Реалізація мультимовності;
3	Галузева ознака: внутрішньогалузева	Даний продукт є актуальним для сфери розробки IT додатків \ продуктів	Зрозумілий орієнтований на споживача інтерфейс; Цілодобова онлайн підтримка; Підтримка якнайбільшої кількості методів взаємодії з середовищем розробки
4	Конкуренція за видами товарів: товарно-видова	Конкуренція між товарами одного виду.	Підтримка інновацій/функціоналу, яких не мають аналоги продукту; Максимально прості інтерфейси;
5	Характер конкурентних переваг: цінова та не цінова	Цінові переваги – точкова комерціалізація; Не цінова – унікальний функціонал, якого немає у аналогів	Платні ліцензії на конкретний функціонал з чітко вираженим строком дії; Унікальний функціонал;

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Компанії, що надають продукти моніторингу роботи мікросервісів : Google, Kibana Cloud, Netflix	Рекламні кампанії, організації Google та Kibana, є досить розумними та дорогими. Питання інвестицій.	Як постачальника визначити лише GCP, треба стежити за зміною технологій та бути готовими до downtime-в. Це напряму впливає на розробку.	Клієнтами може бути будь-яка компанія, що використовую для розробки мікросервісну архітектуру та зацікавлена у якісному моніторингу роботи мікросервісів -географія досить широка, можуть бути як вітчизняні компанії, так і інтернаціональні.	Доступні аналоги, розроблені прямими конкурентами, але не замінники
Висновки		Потенційні конкуренти-компанії, що створюють власний продукт для моніторингу роботи мікросервісів та планують продати/відкрити вихідний код.	Необхідність пристосовуватись до змін у GCP є обов’язковою при роботі з ними.	Клієнти зацікавлені у простоті та зручності продукту.	Не наявні у аналогів

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Простота інтерфейсу інтеграції	Простота інтеграції зменшує час розробки коду для інтеграції, що зменшує додаткову вартість та складність підтримки для клієнта
2	Інтеграція з системою клієнта	Інтеграція з програмними інтерфейсами клієнта збільшує гнучкість системи
3	Надання додаткової аналітики даних клієнта	Додаткова аналітика даних надає цінну інформацію, щодо роботи системи клієнта

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін системи виявлення аномалій мікросервісів

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	+1	+2	+3
1	Простота інтерфейсу інтеграції	15			+				
2	Інтеграція з системою клієнта	10					+		
3	Надання додаткової аналітики даних клієнта	12				+			

Таблиця 4.12 – SWOT аналіз стартап-проєкту

Сильні сторони (S): – простота інтерфейсу інтеграції; – швидкість роботи; – додаткова аналітика.	Слабкі сторони (W): – вузькоспеціалізована система; – клієнти повинні адаптувати свої системи.
Можливості (O): – вдосконалення алгоритму; – створення науково-дослідницького відділу, що привабить висококваліфікованих працівників.	Загрози (T): – поява безкоштовних аналогів; – велика компанія конкурент, що зможе диктувати умови на ринку.

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проєкту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Пробні плани з лімітованою кількістю запитів	Головний ресурс – працівники, даний ресурс - наявний	1-2 місяці
2	Виступи на професійних доповідях	Головний ресурс – час, даний ресурс - наявний	1-2 місяці
3	Ведення технічного блогу компанії	Головний ресурс – час, даний ресурс - наявний	4-6 місяці
4	Реклама	Головний ресурс – час, гроші даний ресурс - наявні	2-3 місяці

4.4 Розроблення ринкової стратегії проєкту

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Компанії які мають малу кількість мікросервісів	Вартість є критичною	Середній попит	Конкуренти мають середню інтенсивність в сегменті	Вхід в сегмент складний, необхідно зменшити вартість та надавати тимчасові знижки

Кінець таблиці Таблиця 4.14

2	Компанії які мають середню та велику кількість мікросервісів	Критичним є точність та надійність продукту	Високий попит		Вхід в сегмент середньої складності, необхідно мати кваліфіковану команду для надійної інтеграції з системою клієнта
3	Стартапи з мікросервісною архітектурою	Критичним є вартість та простота	Низький попит		Складний вхід в сегмент, потребує інвестування в маркетинг

Які цільові групи обрано: 2

Відповідно до проведеного аналізу можна зробити висновок, що підходящою цільовою групою для розповсюдження даного програмного продукту є компанії з середньою та великою кількістю мікросервісів. Такі компанії цінують надійність та точність, та готові платити більше за систему яка допомагає уникати дефектів та збоїв в роботі власних продуктів. На основі цього обрано стратегію таргетовиного маркетингу, адже є чіткий запит від клієнтів.

Таблиця 4.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Широка картина процесів мікросервісів	Наростаюча	Зниження ступеню замінності товару завдяки забезпеченню гнучкості системи; Прихильність клієнтів, адже для нього це зручно; Комфортні для клієнта характеристики продукту	Стратегія диференціації

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні, оскільки є аналоги, але дані аналоги не мають деякого необхідного функціоналу	Ціллю є переманити споживачів у конкурентів, щоб задовольнити їх потреби, та знайти нових	Ні	Стратегія заняття конкурентної ніші

Таблиця 4.17 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту
1	Простота інтерфейсу інтеграції	Стратегія диференціації	Дешевше та швидше для клієнта, спрощення інтеграції	Простота, швидкість,
2	Інтеграція з системою клієнта	Стратегія диференціації	Гнучкість системи	Гнучкість, комфорт
3	Додаткова аналітика	Стратегія диференціації	Наявність додаткової аналітики для кращого бачення процесів	Точність, якість

4.5 Розроблення маркетингової програми стартап-проєкту

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Швидка інтеграція	Простий інтерфейс інтеграції	Спрощений інтерфейс інтеграції, що зекономить клієнту час та гроші + спростить підтримку у майбутньому
2	Гнучкість системи	Інтеграція із системою клієнта	Можливість використовувати дані клієнта, якщо вони мають власний моніторинг
3	Широке бачення процесів компанії	Додаткова аналітика	Нові показники, наприклад зв’язок події з метриками

4.6 Висновки по розділу

В четвертому розділі описано стратегії та підходи з розроблення стартап-проєкту, визначення цільової аудиторії, обрано стратегії позиціонування, проаналізовано ринок та обрано альтернативи впровадження на ринок. Розроблено SWOT аналіз та виявлено конкурентів та методи боротьби з ними.

ВИСНОВОК

У ході виконання магістерської дисертації було розглянуто питання, пов’язані з виявленням аномалій в мікросервісній архітектурі. Спроектовано алгоритм виявлення аномалій з використанням показників роботи та потоком подій мікросервісу.

Для розробки моделі було використано:

- мову програмування Python;
- tensorflow;
- keras.

Розроблено модель з використанням LSTM шарів та комбінованим ядром, двома входами та двома виходами для передбачення майбутніх екземплярів. Виконано експериментальне порівняння комбінованих та некомбінованих моделей, та виявлено, що комбінована модель краще виконує виявлення аномалій та має кращі результати навчання, що робить її доцільною для використання в поставленій задачі.

Розроблено програмне забезпечення розгортання мікросервісу з виявленням аномалій та автоматичним відкочуванням при наявності аномалії.

Розроблено тестовий об’єкт та проведено тестування моделі.

Проведений маркетинговий аналіз стартап-проєкту. Проведено опис ідеї проєкту, технологічний аудит ідеї проєкту, аналіз ринкових можливостей запуску стартап-проєкту. Розроблено ринкову стратегію проєкту та маркетингову програму стартап-проєкту.

Проведено порівняльний аналіз комбінованої та не комбінованої моделі передбачення подій та показників в результаті якого, виявлено, що комбінована модель якісніше класифікує аномалії.

Наукова новизна одержаних результатів магістерської дисертації:

вперше:

– запропоновано комбінований Енкодер числових та категоріальних даних.

удосконалено:

– модель виявлення аномалій роботи мікросервісів.

Практична значимість одержаних результатів полягає в створенні інструмента автоматизованого розгортання, що дозволяє зменшити ризики впливу дефектного коду на роботу інформаційної системи.

Майбутніми покращеннями можуть бути:

- реалізація гнучкої системи адаптації моделі під зміни кількості показників та класів подій;
- постійне навчання моделі в процесі роботи системи мікросервісів, для запобігання деградації моделі та появи хибно позитивних результатів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) A Classification Framework for Anomaly Detection / S. Haykin, X. B. Li // Proc. IEEE. — 1995. — Вип. 83, № 1. — С. 95–122.

2) Data Clustering: Algorithms and Applications / C. K. Reddy, F. Group; Boca Raton. — CRC Press, 2014. — 49 с. — ISBN 9781466558212.

3) A new look at the statistical model identification / H. Akaike // IEEE Trans. Automat. Contr. — 1974. — Вип. 19, № 6. — С. 716–723.

4) Fast Outlier Detection in High Dimensional Spaces BT - Principles of Data Mining and Knowledge Discovery / F. Angiulli, C. Pizzuti ; Elomaa T., Mannila H., Toivonen H. — Springer Berlin Heidelberg, 2002. — ISBN 978-3540456810.

5) Applied Econometrics / D. Asteriou. — Springer, 2015. — 400 с. — ISBN 978-1137415462.

6) Application of Long Short-Term Memory (LSTM) neural network for flood forecasting / X. H. Le, H. V. Ho, G. Lee, S. Jung // Water (Switzerland). — 2019. — Вип. 11, № 7.

7) Microservices [Електронний ресурс] / Martin F. – Режим доступу до ресурсу: https://martinfowler.com/articles/microservices.html

8) Pattern: API Gateway/Backend for Frontends [Електронний ресурс] / Chris R. – Режим доступу до ресурсу: https://microservices.io/patterns/apigateway.html

9) Що таке Kubernetes? – [Електронний ресурс] – Режим доступу до ресурсу: https://kubernetes.io/uk/docs/concepts/overview/what-is-kubernetes/

10) Capture and visualize metrics using Prometheus and Grafana - [Електронний ресурс] – Режим доступу до ресурсу: https://docs.particular.net/samples/logging/prometheus-grafana/

11) What is Prometheus? – [Електронний ресурс] – Режим доступу до ресурсу: https://prometheus.io/docs/introduction/overview/

12) R for Data Science: Import, Tidy, Transform, Visualize, and Model Data / Wickham G., Hadley G. — O'Reilly Media, 2017. — 350 с. — ISBN 978-1491910399.

13) Minikube [Електронний ресурс] – Режим доступу до ресурсу https://kubernetes.io/docs/setup/minikube/.

14) Locust: An open source load testing tool [Електронний ресурс] – Режим доступу до ресурсу: https://locust.io.

15) A Python Book: Beginning Python, Advanced Python, and Python Exercises / D. Kuhlman. — Platypus Global Media, 2013. — 1-227 с. — ISBN 978-0984221233.

16) Економіка і суспільство / Геселева Н.В., Пронюк Г.В.: Особливості марковського моделювання для оцінювання надійності технічних систем – 2018. – 968 с

17) Pattern: Mocroservice 2018 [Електронний ресурс] / Michael P. – Режим доступу до ресурсу: https://microservices.io/patterns/microservices.html

18) Scalability Design with MicroServices [Електронний ресурс] – Режим доступу до ресурсу: https://apilama.com/2018/07/18/scalability-design-with-microservices/

19) Understanding LSTM Networks [Електронний ресурс] – Режим доступу до ресурсу: https://colah.github.io/posts/2015-08-Understanding-LSTMs

20) Neural smithing : supervised learning in feedforward artificial neural networks / R. D. Reed, R. J. Marks. — MIT Press, 1999. — 346 с. — ISBN 978-0262181907.

21) How many samples are needed to build a classifier: A general sequential approach / W. J. Fu, E. R. Dougherty, B. Mallick, R. J. Carroll // Bioinformatics. — 2005. — Вип. 21, № 1. — С. 63–70.

22) Ji Y. ContamDE-lm: Linear model-based differential gene expression analysis using next-generation RNA-seq data from contaminated tumor samples / Y. Ji, C. Yu, H. Zhang // Bioinformatics. — 2020. — Вип. 36, № 8. — С. 2492–2499.

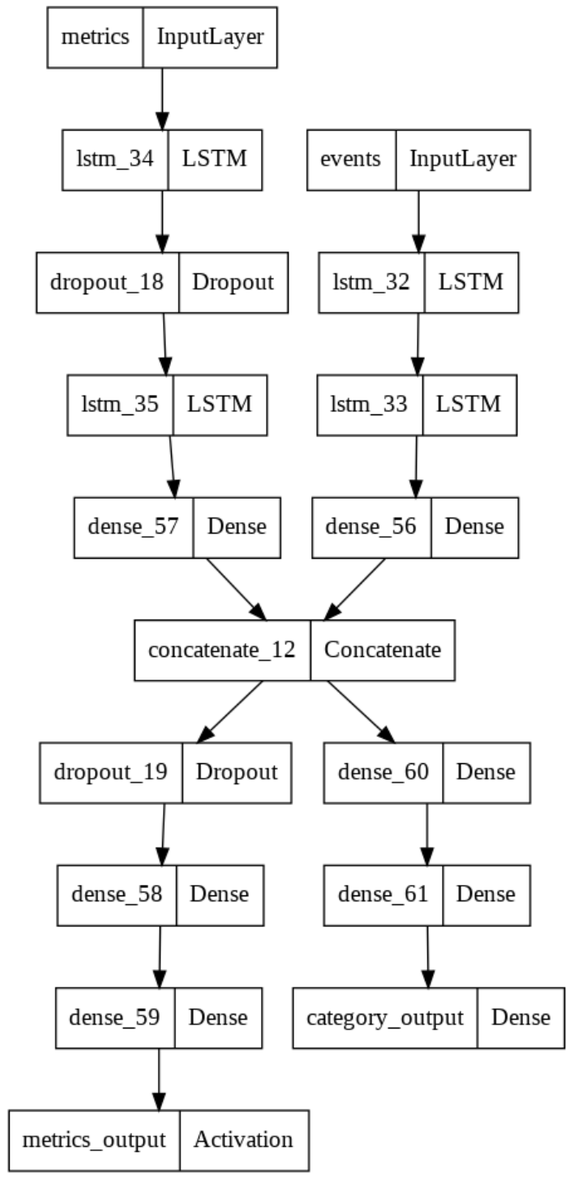
23) Welcome to Colaboratory [Електронний ресурс] – Режим доступу до ресурсу: https://colab.research.google.com/?utm_source=scs-index.

24) Обзор методов обнаружения аномалий в потоках данных / В. П. Шкодырев, К. И. Ягафаров, В. А. Баштовенко, Е. Э. Ильина // Proc. Second Conf. Softw. Eng. Inf. Manag. — 2017. — Вип. 1864. — С. 7.

ДОДАТОК А

Графічний матеріал

Схема моделі нейронної мережі передбачення подій та показників



Демонстраційний плакат до магістерської дисертації
на тему «Методи та програмне забезпечення виявлення аномалій при розгортанні мікросервісу »

Виконав студент гр. ІТ-03мп Клименко Д.В.

Керівник Стеценко І. В.

ДОДАТОК Б

Вихідний код

Model.py

```
!pip3 install pandas

!pip3 install tensorflow --upgrade --force-reinstall

!pip3 install seaborn

!pip3 install sklearn

import pandas as pd
import numpy as np
import matplotlib
import seaborn
import matplotlib.dates as md
from matplotlib import pyplot as plt
from sklearn import preprocessing
from keras.layers.core import Dense, Activation, Dropout, RepeatVector
from keras.layers.recurrent import LSTM
from keras.layers import Embedding
from keras.models import Sequential, Model
from sklearn.model_selection import train_test_split
import time
import tensorflow as tf
import keras as keras
import matplotlib.pyplot as plt
from tensorflow.keras.layers import *
from scipy import stats
from scipy.signal import savgol_filter
import tensorflow as tensorflow
import requests
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
import joblib
from google.colab import drive
from IPython.display import HTML, display
import plotly.graph_objects as go

drive.mount('/content/drive')

def set_css():
  display(HTML('''
  <style>
    pre {
        white-space: pre-wrap;
    }
  </style>
  '''))
get_ipython().events.register('pre_run_cell', set_css)

def binary_cross_entropy(yhat: np.ndarray, y: np.ndarray) -> float:
    return -(y * np.log(yhat) + (1 - y) * np.log(1 - yhat))

def norm(colm):
  return (colm-colm.min())/(colm.max()-colm.min())

def make_data(frame):
  data = pd.DataFrame()
  data["memory_space_usage"] = norm(frame[frame.columns[5]])
  data["memory_space_activity"] = norm(frame[frame.columns[18]])
  data["real_heap_committed"] = norm(frame[frame.columns[20]])
```

```
  data["memory_space_usage_2"] =  norm(frame[frame.columns[36]])
  data["physical_mem_act"] =  norm(frame[frame.columns[30]])
  data["process_cpu"] =  norm(frame[frame.columns[34]])
  data["rel_physical_mem_usage"] =  norm(frame["go_goroutines[]"])
  return train_test_split(data, test_size=0.2, shuffle=False)

TIME_STEPS=100
def create_sequences(X, time_steps=TIME_STEPS):
    Xs, ys = [], []
    for i in range(len(X)-time_steps):
        Xs.append(X.iloc[i:(i+time_steps)].values)
        ys.append(X.iloc[i+time_steps].values)

    return np.array(Xs), np.array(ys)

"""# EVENTS"""

df = pd.read_csv("/content/drive/MyDrive/events_go3.csv")

events = pd.get_dummies(df[df.columns[0]])

events.head()

trainE, testE = train_test_split(events, test_size=0.2, shuffle=False)

X_trainE, y_trainE = create_sequences(trainE, 100)
X_testE, y_testE = create_sequences(testE, 100)
print(f'Training shape: {X_trainE.shape}')
print(f'Testing shape: {X_testE.shape}')
print(f'Training y shape: {y_trainE.shape}')
print(f'Testing y shape: {y_testE.shape}')

modelE = Sequential()
modelE.add(LSTM(128, input_shape=(100, 12), return_sequences=True))
modelE.add(Dropout(0.2))
modelE.add(LSTM(128, return_sequences=False))
modelE.add(Dropout(0.2))
modelE.add(Dense(1000, activation='relu'))
modelE.add(Dense(12, activation='sigmoid'))
modelE.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
modelE.summary()

historyE = modelE.fit(X_trainE, y_trainE, epochs=5, batch_size=100,
validation_split=0.1, shuffle=True)

modelE.save("./drive/MyDrive/model_E_for_test")

np.save('./drive/MyDrive/history-E.npy', historyE.history)

modelE.evaluate(X_testE, y_test, verbose=1)

"""# Metrics

"""

df = pd.read_csv("/content/drive/MyDrive/result2.csv")

columns = ['go_goroutines[]', 'go_memstats_alloc_bytes[]',
           'go_memstats_alloc_bytes_total[]', 'go_memstats_frees_total[]',
           'go_memstats_gc_cpu_fraction[]', 'go_memstats_gc_sys_bytes[]',
           'go_memstats_heap_alloc_bytes[]', 'go_memstats_heap_idle_bytes[]',
           'go_memstats_heap_inuse_bytes[]', 'go_memstats_heap_objects[]',
```

```
            'go_memstats_heap_released_bytes[]', 'go_memstats_heap_sys_bytes[]',
            'go_memstats_last_gc_time_seconds[]', 'go_memstats_mallocs_total[]',
            'go_memstats_mspan_inuse_bytes[]', 'go_memstats_mspan_sys_bytes[]',
            'go_memstats_next_gc_bytes[]', 'go_memstats_other_sys_bytes[]',
            'go_memstats_stack_inuse_bytes[]', 'go_memstats_stack_sys_bytes[]',
            'go_memstats_sys_bytes[]', 'go_threads[]',
            'process_cpu_seconds_total[]', 'process_open_fds[]',
            'process_resident_memory_bytes[]', 'process_start_time_seconds[]',
            'process_virtual_memory_bytes[]',
             'go_goroutines[]',]

df.plot.hist(subplots=True, figsize=(30,20), layout=(10,10))

to_diff = ['go_gc_duration_seconds[]-0.000000',
 'gin_requests_total[name:"code" value:"201"  name:"handler" value:"takeoff-
projects/denys-klymenko/api/pets.(*Controller).createPet-fm"  name:"host"
value:"192.168.64.9:31001"  name:"method" value:"POST"  name:"url" value:"/pets"
]',
 'gin_requests_total[name:"code" value:"200"  name:"handler" value:"takeoff-
projects/denys-klymenko/api/pets.(*Controller).getPets-fm"  name:"host"
value:"pets-api:8080"  name:"method" value:"GET"  name:"url" value:"/pets" ]',
 'go_memstats_lookups_total[]',
 'go_memstats_mcache_sys_bytes[]',
 'go_memstats_alloc_bytes_total[]',
 'go_memstats_mspan_inuse_bytes[]',
 'promhttp_metric_handler_requests_total[name:"code" value:"503" ]',
 'process_start_time_seconds[]',
 'go_memstats_heap_released_bytes[]',
 'go_gc_duration_seconds[]-0.250000',
 'go_gc_duration_seconds[]-0.750000',
 'go_memstats_stack_sys_bytes[]',
 'process_resident_memory_bytes[]',
 'go_memstats_heap_sys_bytes[]',
 'go_memstats_sys_bytes[]',
 'go_memstats_last_gc_time_seconds[]',
 'go_gc_duration_seconds[]-0.500000',
 'go_memstats_mspan_sys_bytes[]',
 'go_memstats_other_sys_bytes[]',
 'process_cpu_seconds_total[]',
 'process_open_fds[]',
 'go_memstats_stack_inuse_bytes[]',
 'go_gc_duration_seconds[]-1.000000',
 'gin_requests_total[name:"code" value:"200"  name:"handler" value:"takeoff-
projects/denys-klymenko/api/pets.handleRequests.func1"  name:"host"
value:"172.17.0.7:8080"  name:"method" value:"GET"  name:"url" value:"/health"
]',
 'promhttp_metric_handler_requests_total[name:"code" value:"200" ]',
 'go_threads[]',
 'go_memstats_mcache_inuse_bytes[]',
 'go_memstats_gc_cpu_fraction[]',
 'go_memstats_next_gc_bytes[]',
 'promhttp_metric_handler_requests_total[name:"code" value:"500" ]',
 'process_max_fds[]',
 'gin_requests_total[name:"code" value:"400"  name:"handler" value:"takeoff-
projects/denys-klymenko/api/pets.(*Controller).createPet-fm"  name:"host"
value:"192.168.64.9:31001"  name:"method" value:"POST"  name:"url" value:"/pets"
]',
 'go_memstats_mallocs_total[]',
 'gin_requests_total[name:"code" value:"200"  name:"handler" value:"takeoff-
projects/denys-klymenko/api/pets.(*Controller).getPets-fm"  name:"host"
value:"192.168.64.9:31001"  name:"method" value:"GET"  name:"url" value:"/pets"
]',
 'go_memstats_frees_total[]',
```

```
 'process_virtual_memory_bytes[]',
 'go_goroutines[]',
 'gin_requests_total[name:"code" value:"201"  name:"handler" value:"takeoff-
projects/denys-klymenko/api/pets.(*Controller).createPet-fm"  name:"host"
value:"pets-api:8080"  name:"method" value:"POST"  name:"url" value:"/pets" ]']
to_diff = list(set(to_diff))

df = df.drop(columns=['process_virtual_memory_max_bytes[]',
'go_memstats_heap_inuse_bytes[]','go_memstats_heap_idle_bytes[]',
'go_memstats_heap_objects[]',     'go_memstats_heap_alloc_bytes[]',])

for col in to_diff:
  df[col] = df[col].diff()

df = df.dropna()

df2.plot(subplots=True, figsize=(10, 40), layout=(25, 2))

normalized_df=(df-df.min())/(df.max()-df.min())
normalized_df = normalized_df.dropna(axis=1, how="any")

normalized_df.plot(subplots=True, figsize=(10, 40), layout=(25, 2))

len(list(set(normalized_df.columns).intersection(set(to_diff))))

normalized_df.columns.size

trainM, testM = train_test_split(normalized_df, test_size=0.2, shuffle=False)

X_trainM, y_trainM = create_sequences(trainM, 300)
X_testM, y_testM = create_sequences(testM, 300)

X_trainM.shape

modelM = Sequential()
modelM.add(LSTM(256, input_shape=(300, y_trainM.shape[1]),
return_sequences=True))
modelM.add(Dropout(0.2))
modelM.add(LSTM(256, return_sequences=True))
modelM.add(LSTM(256, return_sequences=False))
modelM.add(Dense(1000))
modelM.add(Dropout(0.2))
modelM.add(Dense(100))
modelM.add(Dense(y_trainM.shape[1]))
modelM.add(Activation('linear'))
modelM.compile(optimizer='adam', loss='mse', metrics=["accuracy"])
modelM.summary()

history = modelM.fit(X_trainM, y_trainM, epochs=10, batch_size=100,
validation_split=0.1, shuffle=True)

modelM.save("./drive/MyDrive/model_M_for_test")

min_max_scaler = preprocessing.MinMaxScaler()
min_max_scaler = min_max_scaler.fit(df)

scaler_filename = "scaler_for_test.save"
joblib.dump(min_max_scaler, scaler_filename)

modelM.evaluate(X_testM, y_testM)

"""# Combined
```

```
"""

losses = {
   "category_output": "binary_crossentropy",
   "metrics_output": "mse",
}
lossWeights = {
    "category_output": 1.0,
    "metrics_output": 1.0
}

input1 = keras.layers.Input(shape=(100, 12), name="events")
input2 = keras.layers.Input(shape=(300, 34), name="metrics")

inputx = LSTM(512, input_shape=(100, 12), return_sequences=True)(input1)
inputx = LSTM(256, return_sequences=True)(inputx)
inputx = LSTM(256, return_sequences=False)(inputx)
inputx = Dropout(0.2)(inputx)
inputx = Dense(1000)(inputx)


inputy = LSTM(256, input_shape=(300, 34), return_sequences=True)(input2)
inputy = Dropout(0.2)(inputy)
inputy = LSTM(256, return_sequences=False)(inputy)
inputy = Dense(1000)(inputy)


input = Concatenate()([inputx, inputy])

x1 = Dropout(0.2)(input)
x1 = Dense(1000)(x1)
x1 = Dense(34)(x1)
x1 = Activation('linear', name="metrics_output")(x1)


x2 = Dense(1000, activation='relu')(input)
x2 = Dense(500, activation='relu')(x2)
x2 = Dense(12, activation='sigmoid', name="category_output")(x2)


model = Model(inputs=[input1, input2], outputs=[x1, x2])
model.compile(optimizer='adam', loss=losses, loss_weights=lossWeights,
metrics=["accuracy"])
model.summary()

X_trainM = np.repeat(X_trainM, 7, axis=0)
y_trainM = np.repeat(y_trainM, 7, axis=0)

X_testM = np.repeat(X_testM, 7, axis=0)
y_testM = np.repeat(y_testM, 7, axis=0)

truncate = min(X_trainE.shape[0], X_trainM.shape[0])

history = model.fit(x={"events": X_trainE[0:truncate], "metrics":
X_trainM[0:truncate]}, y={"metrics_output": y_trainM[0:truncate],
"category_output": y_trainE[0:truncate]}, epochs=5, batch_size=100,
validation_split=0.1, shuffle=False)

truncate_test = min(X_testE.shape[0], X_testM.shape[0])

model.evaluate(x={"events": X_testE[0:truncate_test], "metrics":
X_testM[0:truncate_test]}, y={"category_output": y_testE[0:truncate_test],
"metrics_output": y_testM[0:truncate_test]}, verbose=1)
```

```
train_pred = model.predict(x={"events": X_trainE[0:truncate], "metrics":
X_trainM[0:truncate]}, verbose=1)

test_pred = model.predict(x={"events": X_testE[0:truncate_test], "metrics":
X_testM[0:truncate_test]}, verbose=1)

train_mae_lossM = np.mean(np.abs(train_pred[0] - y_trainM[0:truncate]), axis=1)
train_mae_lossE = np.array([np.mean(arr) for arr in
binary_cross_entropy(train_pred[1][0:truncate], y_trainE[0:truncate]) ])
train_mae_loss = train_mae_lossE + train_mae_lossM
plt.hist(train_mae_loss, bins=50)
plt.xlabel('Train MAE loss')
plt.ylabel('Number of Samples');

threshold = np.max(train_mae_loss)

print(threshold)

test_score_df = pd.DataFrame(y_trainM[0:truncate])
test_score_df['loss'] = train_mae_loss
test_score_df['threshold'] = threshold
test_score_df['anomaly'] = test_score_df['loss'] > test_score_df['threshold']
test_score_df['Value'] = y_trainM[0:truncate, 6 ]

fig = go.Figure()
fig.add_trace(go.Scatter(y=test_score_df['loss'], name='Test loss'))
fig.add_trace(go.Scatter(y=test_score_df['threshold'], name='Threshold'))
fig.update_layout(showlegend=True, title='Test loss vs. Threshold')
fig.show()

anomalies = test_score_df.loc[test_score_df['anomaly'] == True]
anomalies.shape

fig = go.Figure()
fig.add_trace(go.Scatter( y=test_score_df['Value'], name='metric'))
fig.add_trace(go.Scatter(x = anomalies.index, y=anomalies['Value'],
mode='markers', name='Anomaly'))
fig.update_layout(showlegend=True, title='Detected anomalies')
fig.show()

plt.plot(history.history['category_output_accuracy'])
plt.plot(history.history['val_category_output_accuracy'])
plt.title('model accuracy events')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

plt.plot(history.history['metrics_output_accuracy'])
plt.plot(history.history['val_metrics_output_accuracy'])
plt.title('model accuracy metrics')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('traning loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
```

```
plt.show()

np.save('./drive/MyDrive/history-combine.npy', history.history)

model.save("./drive/MyDrive/combine_model")

model = keras.models.load_model("./drive/MyDrive/combine-model")

history = model.fit(x={"events": X_event[0:118668], "metrics":
X_train[0:118668]}, y=y_train[0:118668], epochs=10, batch_size=1000,
validation_split=0.1, shuffle=True)

tf.keras.utils.plot_model(
    model,
    to_file="model.png",
    show_shapes=False,
    show_dtype=False,
    show_layer_names=True,
    rankdir="TB",
    expand_nested=False,
    dpi=96,
    layer_range=None,
)
```

Deployer.py

```
import pandas as pd
import numpy as np
import keras as keras
import requests
from sklearn.preprocessing import OneHotEncoder
import joblib
import subprocess
import time
import sys

columns = [
    'gin_requests_total[name:"code" value:"200"  name:"handler" value:"my-
projects/denys-klymenko/api/pets.(*Controller).getPets-fm"  name:"host"
value:"192.168.64.9:31001"  name:"method" value:"GET"  name:"url" value:"/pets"
]',
    'gin_requests_total[name:"code" value:"200"  name:"handler" value:"my-
projects/denys-klymenko/api/pets.(*Controller).getPets-fm"  name:"host"
value:"pets-api:8080"  name:"method" value:"GET"  name:"url" value:"/pets" ]',
    'gin_requests_total[name:"code" value:"200"  name:"handler" value:"my-
projects/denys-klymenko/api/pets.handleRequests.func1"  name:"host"
value:"172.17.0.7:8080"  name:"method" value:"GET"  name:"url" value:"/health"
]',
    'gin_requests_total[name:"code" value:"201"  name:"handler" value:"my-
projects/denys-klymenko/api/pets.(*Controller).createPet-fm"  name:"host"
value:"192.168.64.9:31001"  name:"method" value:"POST"  name:"url" value:"/pets"
]',
    'gin_requests_total[name:"code" value:"201"  name:"handler" value:"my-
projects/denys-klymenko/api/pets.(*Controller).createPet-fm"  name:"host"
value:"pets-api:8080"  name:"method" value:"POST"  name:"url" value:"/pets" ]',
    'gin_requests_total[name:"code" value:"400"  name:"handler" value:"my-
projects/denys-klymenko/api/pets.(*Controller).createPet-fm"  name:"host"
value:"192.168.64.9:31001"  name:"method" value:"POST"  name:"url" value:"/pets"
]',
```

```
    'go_gc_duration_seconds[]-0.250000',
    'go_gc_duration_seconds[]-0.500000',
    'go_gc_duration_seconds[]-0.750000',
    'go_gc_duration_seconds[]-1.000000', 'go_goroutines[]',
    'go_memstats_alloc_bytes[]', 'go_memstats_alloc_bytes_total[]',
    'go_memstats_frees_total[]', 'go_memstats_gc_cpu_fraction[]',
    'go_memstats_gc_sys_bytes[]', 'go_memstats_heap_released_bytes[]',
    'go_memstats_heap_sys_bytes[]', 'go_memstats_last_gc_time_seconds[]',
    'go_memstats_mallocs_total[]', 'go_memstats_mspan_inuse_bytes[]',
    'go_memstats_mspan_sys_bytes[]', 'go_memstats_next_gc_bytes[]',
    'go_memstats_other_sys_bytes[]', 'go_memstats_stack_inuse_bytes[]',
    'go_memstats_stack_sys_bytes[]', 'go_memstats_sys_bytes[]',
    'go_threads[]', 'process_cpu_seconds_total[]', 'process_open_fds[]',
    'process_resident_memory_bytes[]', 'process_virtual_memory_bytes[]',
    'promhttp_metric_handler_requests_in_flight[]',
    'promhttp_metric_handler_requests_total[name:"code" value:"200" ]']

columns_to_rate = [
    'gin_requests_total[name:"code" value:"201"  name:"handler" value:"my-
projects/denys-klymenko/api/pets.(*Controller).createPet-fm"  name:"host"
value:"192.168.64.9:31001"  name:"method" value:"POST"  name:"url" value:"/pets"
]',
    'gin_requests_total[name:"code" value:"200"  name:"handler" value:"my-
projects/denys-klymenko/api/pets.(*Controller).getPets-fm"  name:"host"
value:"pets-api:8080"  name:"method" value:"GET"  name:"url" value:"/pets" ]',
    'go_memstats_alloc_bytes_total[]',
    'go_memstats_mspan_inuse_bytes[]',
    'go_memstats_heap_released_bytes[]',
    'go_gc_duration_seconds[]-0.250000',
    'go_gc_duration_seconds[]-0.750000',
    'go_memstats_stack_sys_bytes[]',
    'process_resident_memory_bytes[]',
    'go_memstats_sys_bytes[]',
    'go_memstats_heap_sys_bytes[]',
    'go_memstats_last_gc_time_seconds[]',
    'go_gc_duration_seconds[]-0.500000',
    'go_memstats_mspan_sys_bytes[]',
    'go_memstats_other_sys_bytes[]',
    'process_cpu_seconds_total[]',
    'go_memstats_stack_inuse_bytes[]',
    'process_open_fds[]',
    'go_gc_duration_seconds[]-1.000000',
    'gin_requests_total[name:"code" value:"200"  name:"handler" value:"my-
projects/denys-klymenko/api/pets.handleRequests.func1"  name:"host"
value:"172.17.0.7:8080"  name:"method" value:"GET"  name:"url" value:"/health"
]',
    'promhttp_metric_handler_requests_total[name:"code" value:"200" ]',
    'go_threads[]',
    'go_memstats_gc_cpu_fraction[]',
    'go_memstats_next_gc_bytes[]',
    'gin_requests_total[name:"code" value:"400"  name:"handler" value:"my-
projects/denys-klymenko/api/pets.(*Controller).createPet-fm"  name:"host"
value:"192.168.64.9:31001"  name:"method" value:"POST"  name:"url" value:"/pets"
]',
    'go_memstats_mallocs_total[]',
    'gin_requests_total[name:"code" value:"200"  name:"handler" value:"my-
projects/denys-klymenko/api/pets.(*Controller).getPets-fm"  name:"host"
value:"192.168.64.9:31001"  name:"method" value:"GET"  name:"url" value:"/pets"
]',
    'go_memstats_frees_total[]',
    'process_virtual_memory_bytes[]',
    'go_goroutines[]',
    'gin_requests_total[name:"code" value:"201"  name:"handler" value:"my-
```

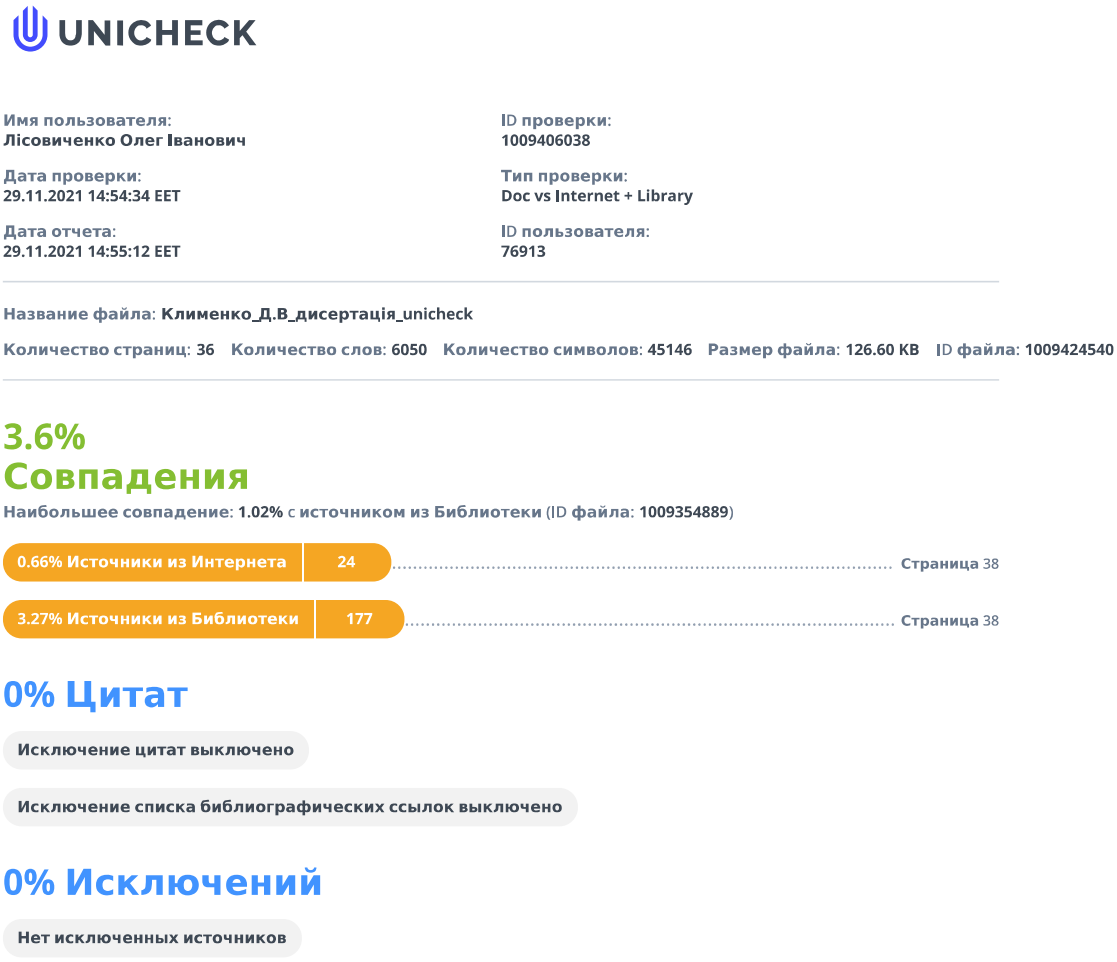
```
projects/denys-klymenko/api/pets.(*Controller).createPet-fm"  name:"host"
value:"pets-api:8080"  name:"method" value:"POST"  name:"url" value:"/pets" ]']


def binary_cross_entropy(yhat, y):
    return -(y * np.log(yhat) + (1 - y) * np.log(1 - yhat))


eventsThreshold = 1.2
metricsThreshold = 0.3
model = keras.models.load_model("./combine_model")
modelE = keras.models.load_model("./model_E_for_test")
modelM = keras.models.load_model("./model_M_for_test")

scaler = joblib.load("scaler_for_test.save")
enc = OneHotEncoder()
enc.fit(pd.DataFrame(['pets-api.create.pet-created', 'pets-api.create.request-
received',
                      'pets-api.create.response-sent', 'pets-api.create.sent-to-
dal',
                      'pets-api.create.validation-failed',
                      'pets-api.list.request-received', 'pets-api.list.response-
sent',
                      'pets-web.create.pet-added', 'pets-web.create.request-
received',
                      'pets-web.create.request-sent-to-pets-api',
                      'pets-web.list.request-received', 'pets-web.list.response-
sent']))

def get_metrics():
    response = requests.get(url="http://192.168.64.9:31003/metrics?limit=301")
    data = response.json()["metrics"]
    metrics = pd.DataFrame([x['observation'] for x in data])[columns]
    for col in columns_to_rate:
        metrics[col] = metrics[col].diff()
    metrics = pd.DataFrame(scaler.fit_transform(metrics))
    return metrics


def get_events():
    response = requests.get(url="http://192.168.64.9:31003/events?limit=101")
    data = response.json()["events"]
    events = pd.DataFrame([x['name'] for x in data])
    events = pd.DataFrame(enc.transform(events).toarray())
    return events


def check_model_E(events):
    pred = modelE.predict(np.array([events[:-1]]))

    events_error = np.mean(binary_cross_entropy(pred, np.array([events.iloc[-
1].values])))

    print("Model E", events_error, events_error > eventsThreshold)
    return events_error > eventsThreshold


def check_model_M(metrics):
    pred = modelM.predict(np.array([metrics[:-1]]))

    metrics_error = np.mean(np.abs(pred - np.array([metrics.iloc[-1].values])),
axis=1)[0]
```

```
    print("Model M", metrics_error, metrics_error > metricsThreshold)
    return metrics_error > metricsThreshold


def check_for_anomaly(events, metrics):
    pred = model.predict(x={"events": np.array([events[:-1]]), "metrics":
np.array([metrics[:-1]])})

    events_error = np.mean(binary_cross_entropy(pred[1], np.array([events.iloc[-
1].values])))
    metrics_error = np.mean(np.abs(pred[0] - np.array([metrics.iloc[-
1].values])), axis=1)[0]

    print("Combine", events_error, metrics_error)
    return events_error + metrics_error > eventsThreshold + metricsThreshold


def rollback():
    process = subprocess.Popen(['helm', "rollback", "pets-api", "0"],
                               stdout=subprocess.PIPE,
                               stderr=subprocess.PIPE)

    stdout, stderr = process.communicate()

    print(stdout)

    if process.returncode != 0:
        print(strerr)


def deploy(service, version):
    process = subprocess.Popen(['helm', "upgrade", "--set",
f"image.tag={version}", service, "../ops/helm/pets-api"],
                               stdout=subprocess.PIPE,
                               stderr=subprocess.PIPE)

    stdout, stderr = process.communicate()

    print(stdout)
    print(stderr)


if __name__ == "__main__":
    serviceName = sys.argv[1]
    version = sys.argv[2]
    print("deploying...")
    deploy(serviceName, version)
    print("new version has been deployed")
    time.sleep(60)

    try:
        t_end = time.time() + 60 * 2
        while time.time() < t_end:
            events = get_events()
            metrics = get_metrics()
            check_model_E(events)
            check_model_M(metrics)
            if check_for_anomaly(events, metrics):
                print("Anomaly detected rolling back to previous version")
                rollback()
                exit(1)
            else:
                print("no anomalies so far")
```

```
    print("No anomalies detected new version deployed")
except Exception as e:
    print(e)
    rollback()
```

ДОДАТОК В

Результати перевірки роботи на співпадіння



UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1009406038

Дата проверки:
29.11.2021 14:54:34 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
29.11.2021 14:55:12 EET

ID пользователя:
76913

Название файла: **Клименко_Д.В_дисертація_unicheck**

Количество страниц: **36** Количество слов: **6050** Количество символов: **45146** Размер файла: **126.60 KB** ID файла: **1009424540**

3.6%
Совпадения

Наибольшее совпадение: **1.02%** с источником из Библиотеки (ID файла: **1009354889**)

0.66% Источники из Интернета 24 ..... Страница 38

3.27% Источники из Библиотеки 177 ..... Страница 38

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников