НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп’ютерні системи та мережі”
спеціальності 123 “Комп’ютерна інженерія”

на тему: Система моніторингу даних та виявлення підозрілих чи критичних точок

Виконав : студент __4__ курсу, групи _ІО-з91_
(шифр групи)

_____Семенов Артемій Ігорович_____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____асистент Пустовіт О.М._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) ст. викладач Виноградов Ю.М._____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2023 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Інженерія програмного забезпечення комп’ютерних систем”

спеціальності 123 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2023 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Семенова Артемія Ігоровича

1. Тема проєкту *Система моніторингу даних та виявлення підозрілих чи критичних точок*
керівник проєкту асистент Пустовіт О.М.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 31 травня 2023 року №2102-с

2. Термін здачі студентом закінченого проєкту *08 червня 2023 р.*

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Огляд існуючих методів мониторінгу даних.

Розділ 2. Огляд алгоритмів та технологій для розробки системи.

Розділ 3. Деталі розробки системи.

Розділ 4. Дослідження та аналіз розробленої системи.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) структурна системи (структурна схема), діаграма класів (функціональна схема), алгоритм дій програмного забезпечення (принципова схема).

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Виноградов Ю.М.		

7. Дата видачі завдання «10» лютого 2023 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*10.12.2022-15.02.2023*	
2.	*Вивчення та аналіз завдання*	*15.02.2023-15.03.2023*	
3.	*Розробка архітектури та загальної структури системи*	*15.03.2023-25.04.2023*	
4.	*Розробка структур окремих підсистем*	*25.04.2023-5.05.2023*	
5.	*Програмна реалізація системи*	*5.04.2023-15.05.2023*	
6.	*Оформлення пояснювальної записки*	*15.05.2023-30.05.2023*	
7.	*Захист програмного продукту*	*05.06.2023*	
8.	*Передзахист*	*06.06.2023*	
9.	*Захист*	*22.06.2023*	

Студент-дипломник _____ Артемій Семенов
(підпис)

Керівник проєкту _____ Олександр Пустовіт
(підпис)

АНОТАЦІЯ

У даній роботі було детально розглянуто застосування систем машинного навчання як заміні чітко заданих правил для спотереження за потоками даних. Були проаналізовані їхні слабкі та сильні сторони. На основі аналізу було вибрано два кращих методи побудови нейромережі, які лягли в основу вирішення задачі побудови системи моніторінгу. В результаті роботи було створено декілька моделей що можуть спостерігати за потоком даних, орієнтуючись на тренувальні набори даних різного розміру, створені з використанням правил різної складності. Також був створений додаток що дозволяє вручну переглядати, редагувати чи додавати дані та оновлювати модель. Була проаналізована залежність ефективності нейромережі в залежності від використаної мептодики навчання, складності правил що мають емулюватися та розміру тренувальної вибірки. Програмний продукт був розроблений на мові Python.

Ключові слова: машинне навчання, штучний інтелект, користувацький інтерфейс, Python, TensorFlow.

ANNOTATION

In this project, the application of machine learning systems as a replacement for clearly defined rules for monitoring data flows was considered in detail. Their weaknesses and strengths were analyzed. Based on the analysis, the two best methods of building a neural network were chosen, which formed the basis of solving the problem of building a monitoring system. As a result of the work, several models were created that can observe the flow of data, focusing on training data sets of different sizes, created using rules of different complexity. An application was also created that allows you to manually view, edit or add data and update the model. The software product was developed in the Python language.

Keywords: machine learning, artificial intelligence, user interface, Python, TensorFlow.

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпля	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467200.002 ТЗ***	Система моніторингу даних	3		
			та виявлення підозрілих чи			
			критичних точок			
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	Система моніторингу даних	61		
			та виявлення підозрілих чи			
			критичних точок			
			Пояснювальна записка			
	А4	***ІАЛЦ.467200.004 Д1***	Система моніторингу даних	1		
			та виявлення підозрілих чи			
			критичних точок			
			Структура системи			
			(Структурна схема)			
	А4	***ІАЛЦ.4672008.005 Д2***	Система моніторингу даних	1		
			та виявлення підозрілих чи			
			критичних точок			
			Діаграма класів			
			(функціональна схема)			
	А4	***ІАЛЦ.467200.006 Д3***	Система моніторингу даних	1		
			та виявлення підозрілих чи			
			критичних точок			
			Алгоритм дій програмного			
			забезпечення			
			(принципова схема)			
	А4	***ІАЛЦ.467200.007 Д4***	Система моніторингу даних	25		
			та виявлення підозрілих чи			
			критичних точок			
			Текст програмного коду			

					ІАЛЦ.467200.001 ОА
Зм	***Лист***	***№ докум.***	***Підп***	***Дата***	
Розроб		Семенов А.І			*Система моніторингу даних та виявлення підозрілих чи критичних точок* Опис альбому
Перев		Пустовіт О.М.			

Літ.	Аркуш	Аркушів
	1	1

***КПІ* ім. Ігоря Сікорського**
ФІОТ ІО-з91

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Система моніторингу даних та виявлення підозрілих чи критичних точок*»

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку системи моніторингу даних та виявлення підозрілих чи критичних точок, а також на подальшу підтримку та вдосконалення розробленої системи.

Областю застосування цієї системи є аналіз результатів тестування продуктивності в межах системи безперервної інтеграції, але данна система може бути потенційно адаптована для інших видів систем спостереження з даними та правилами аналогічноїх складності: систем ППО, систем обробки результатів медичних аналізів, систем моніторінгу мережевого трафіку та ін.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», який був затверджений факультетом "Інформатики та обчислювальної техніки" кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробка системи моніторингу даних та виявлення підозрілих чи критичних точок, що дозволить ефективно вирішувати дану задачу.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є офіційні документації, публікації та статті в мережі Інтернет на дану тему, науково-технічна література.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Простий і інтуїтивно-зрозумілий інтерфейс системи.
- Надати можливість користувачам передавати вхідні дані для пошуку потоків і отримувати у відповідь результат.
- Надати можливість користувачам власноруч конфігурувати параметри (пере)навчання нейромережі.
- Надати вичерпну та зрозумілу документацію для розробленого продукту.
- Документація має залишити можливість підключення системи до різних джерел і форматів вхідних даних

5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac чи Linux.
- Python 3.9 чи вище.

5.3. Вимоги до апаратної частини

- ЦП не менше ніж Intel Core i5-10400F.
- ROM не менше ніж 64 ГБ.
- RAM не менше ніж 4 ГБ.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	10.12.2022-15.02.2023
Вивчення та аналіз завдання	15.02.2023-15.03.2023
Розробка архітектури та загальної структури системи	15.03.2023-25.04.2023
Розробка структур окремих частин системи	25.04.2023-5.05.2023
Програмна реалізація системи	5.04.2023-15.05.2023
Виправлення помилок	15.05.2023-30.05.2023
Оформлення пояснювальної записки	10.12.2022-15.02.2023

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Система моніторингу даних та виявлення підозрілих чи критичних точок*»

Київ – 2023

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ	Програмне Забезпечення
CI	(Continious Integration) безперервне впровадження
MS	(Monitoring System) Система для спостереження (мониторінгу)
CM	(Classifying Model) Модель класифікації
MD	(Mocked Data) Макет даних
GUI	(Graphical User Interface) Графічний інтерфейс користувача
API	(Application Programming Interface) Прикладний програмний інтерфейс
NTE	(Non Transferrable Expertise) Експертиза не придатна до передачі конвенційними методами («чуйка»).
SPI	(Single Page Interface) Інтерфейс однієї сторінки

ВСТУП

Автоматизація рутинних задач є серед найважливіших змін в сучасній цивілізації, які були досягнуті за допомогою введення інформаційних систем. В останні роки за допомогою систем машинного навчання стала можливою автоматизація задач що раніше вважалися не придатними для ефективної автоматизації.

Серед задач що стали більш доступними за допомогою машинного навчання можна виділити наступні категорії:

- Класифікація
- Кластеризація
- Транскрипція
- Машинний переклад
- Виявлення аномалії
- Синтез і вибірка
- Оцінка щільності ймовірності та функції маси ймовірності
- Зіставлення подібності
- Групування спільного входження
- Каузальне моделювання

Якщо говорити про практичні

Можна запропонувати наступну класифікацію систем спостереження за потоками даних (MS) з точки зору категоризації даних системою:

- Системи ручного спостереження (категоризація даних самою системою відсутня)
- Статичні системи що категоризують дані за допомогою заданих правил

- Динамічні системи що категоризують дані за допомогою машинного навчання
- Комбіновані системи

Дана робота являє собою огляд систем та інструментів що використовуються або можуть бути використани для вирішення задач класифікації та пошуку аномалій, а також огляд їх сильних та слабих сторін з практичною реалізацією.

Також в межах даної роботи була розроблена система що комбінує використання глибокого навчання та ручного вводу для мониторінгу даних з метою оцінки спроможностей використання машинного навчання як засобу передачі NTE від оператора-людини до автоматизованої системи.

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ МЕТОДІВ МОНИТОРІНГУ ПОТОКІВ ДАНИХ

1.1 Визначення понять

Більшість автоматизованих систем що використовуються в сучасному світі для збору та аналізу інформації працюють з потоками даних. Це може бути статистичний аналіз, відеоспостереження, медичні чи військові автоматизовані системи, аналіз ринкових тенденцій на біржі, системи безперервної інтеграції під час розробки ПЗ та ін.

Основні поняття які використовуються в контексті роботи з потоками даних:

Да́ні [1] (англ. data) — це формалізоване подання інформації, яке придатне для інтерпретування, передачі чи обробки за участю людини або автоматичними засобами.

Потік даних - рух даних через систему, що складається з програмного, апаратного забезпечення або їх комбінації.

Точка даних [2] — це дискретна одиниця інформації. У загальному сенсі будь-який окремий факт є точкою даних. Термін «точка даних» приблизно еквівалентний датуму, формі однини даних.

У статистичному або аналітичному контексті це фактична інформація, отримана в результаті вимірювання або дослідження, і може бути представлена у вигляді числових даних, статистичного відображення або графіка.

Простіше кажучи, точка даних може бути числом, словом або навіть фізичним об’єктом. Важливо те, що його можна відрізнити від інших точок даних.

Моніторинг (англ. monitoring — контроль, від лат. monitor — той, хто попереджає, застерігає, радник, консультант) — регулярне спостереження за

станом природ., тех. і соціальних процесів з метою їх оцінки, контролю та прогнозування. Здійснюється за допомогою електронних та ін. тех. засобів з назем., мор. і косм. станцій. Розрізняють глобальний, регіональний і локальний рівні М. Найпоширеніші системи М.: за станом навкол. природ, середовища, громад, думки з певних питань, злочинності. Порядок проведення М. регулюється відповідними правилами. Інформація, отримана за допомогою М., кладеться в основу рішень, що приймаються держ. органами, політ, партіями, громад. організаціями тощо.

В контексті предмету досліджень ми розглядаємо моніторинг природних та технічних процесів, зокрема аналіз даних отрисманих в результаті моніторингу.

1.2 Види та структура потоків даних

1.2.1 Аналогові дані

Аналогові дані – це дані, які представлені фізичним способом. Якщо цифрові дані — це набір окремих символів, аналогові дані зберігаються на фізичних носіях, будь то канавки на поверхні вінілової платівки, магнітна стрічка касети відеомагнітофона чи інші нецифрові носії.

Одна з головних ідей сучасного технологічного світу, що швидко розвивається, полягає в тому, що більшість природних явищ світу можна перевести в цифровий текст, зображення, відео, звук тощо. Наприклад, фізичні рухи об’єктів можна змоделювати в просторовій симуляції, а аудіо та відео в реальному часі можна захоплювати за допомогою ряду систем і пристроїв.

Аналогові дані також можуть бути відомі як органічні дані або дані реального світу.

Один із способів охарактеризувати аналогові дані полягає в тому, що вони просто існують без вимірювання. Щоб аналогові дані були перетворені в

цифрову форму, вони повинні бути захоплені та відтворені за допомогою спеціальних технологій. Як правило, цифрові дані використовують просту двійкову систему для побудови наборів даних, які представляють аудіо- чи відеовведення.

Для простого прикладу різниці між аналоговими та цифровими даними розглянемо рух води. Аналогові дані — це фактична водна поверхня в русі, яку органи чуття сприймають як зміни фізичних рухів, а також кольору, текстури та навіть запаху самої води. Цифровий формат перетворював би фізичний рух, властивості кольору або обидва в набори даних, які імітували б ці властивості в апаратному інтерфейсі або зберігали їх для дослідницьких цілей.

Хоча деякі нові технології можуть стирати межу між аналоговими та цифровими даними, основна природа аналогових даних завжди буде архетипом, на якому базується цифрове перетворення. Іншими словами, хоча цифрові дані можуть імітувати та відтворювати аналогові дані, вони надзвичайно обмежені у своїй здатності всебічно відтворювати аналогові дані.

Перетворення аналогових даних на цифрові є необхідним для будь-якої подальшої їх цифрової обробки.

1.2.2 Цифрові потоки даних

Потоки даних працюють різними способами в багатьох сучасних технологіях із галузевими стандартами для підтримки широких глобальних мереж та індивідуального доступу.

Багато видів потоків даних контролюються за допомогою пакетної системи. Звичайні бездротові платформи 3G і 4G, а також передачі в Інтернеті складаються з цих наборів пакетів даних, які обробляються певним чином. Наприклад, пакети зазвичай включають заголовки, які ідентифікують походження або передбачуваного одержувача, а також іншу інформацію, яка може зробити обробку потоку даних більш ефективною.

Багато IT-фахівців різних типів займаються контролем потоку даних і моніторингом його використання. Мережні адміністратори спостерігають за даними, що надходять у мережу та з неї, або проходять через топологію мережі. Ті, хто займається IT-завданнями, пов’язаними з Інтернетом, дивляться на те, як глобальна мережа обробляє потоки даних. Навіть внутрішні (не мережеві, не IT) групи, такі як бухгалтерія та дослідники, можуть виконувати багато роботи щодо потоків даних, оскільки вони намагаються обробляти різну інформацію та обробляти її різними способами. Технологічні журналісти надають інформацію про стандартні швидкості передачі потоку даних та інші галузеві конвенції, а громадськість дивиться на те, як обробка потоку даних відіграє роль у нових технологіях.

Термін «потік» В контексті комп’ютерних систем використовується кількома подібними способами:

«Редагування потоку», як у sed, awk і perl. Потокове редагування обробляє файл або файли на місці, без необхідності завантажувати файл(и) в інтерфейс користувача. Одним із прикладів такого використання є пошук і заміна всіх файлів у каталозі з командного рядка.

В Unix і споріднених системах на основі мови C потік є джерелом або приймачем даних, як правило, окремих байтів або символів. Потоки — це абстракція, яка використовується під час читання або запису файлів або спілкування через мережеві сокети. Стандартні потоки — це три потоки, доступні для всіх програм.

Пристрої вводу/виводу можна інтерпретувати як потоки, оскільки вони виробляють або споживають потенційно необмежену кількість даних з часом.

В об'єктно-орієнтованому програмуванні вхідні потоки зазвичай реалізуються як ітератори.

У мові Scheme та деяких інших мовах потік — це ліниво обчислена або затримана послідовність елементів даних. Потік можна використовувати так

само, як і список, але наступні елементи обчислюються лише за потреби. Тому потоки можуть представляти нескінченні послідовності та серії.[3]

У стандартній бібліотеці Smalltalk і в інших мовах програмування потік є зовнішнім ітератором. Як і в Scheme, потоки можуть представляти кінцеві або нескінченні послідовності.

Потокова обробка — у паралельній обробці, особливо в обробці графіки, термін потік застосовується як до апаратного, так і до програмного забезпечення. Там він визначає квазібезперервний потік даних, який обробляється мовою програмування потоку даних, щойно стан програми відповідає початковій умові потоку.

1.2.3 Структура цифрових потоків даних

В узагальненому вигляді потік даних в контексті моніторингу даних можна представити як конструкцію, що складається з метаданих потоку і точок даних.

Метадані потоку містять загальну інформацію про дані шо передаються, інформацію про атрибути окремих точок, спосіб їх впорядкування і.т.п.

Точки даних – містять безпосередньо дані що будуть підлягати подальшій обробці.

Потік даних містить різні набори даних, які залежать від вибраного формату даних.

Атрибути – кожен атрибут потоку даних представляє певний тип даних, напр. ідентифікатор сегмента/точки даних, позначка часу, геодані.

Атрибут Timestamp (маркер часу) допомагає визначити, коли сталася подія.

Ідентифікатор суб’єкта – це ідентифікатор, закодований алгоритмом, витягнутий із файлу cookie.

Необроблені дані містять інформацію безпосередньо від постачальника даних без обробки ні алгоритмом, ні людиною.

Оброблені дані – це дані, які були підготовлені (якимось чином змінені, підтверджені чи очищені) для використання в майбутніх діях.

1.2.4 Точки даних

Для роботи з цифровими потоками даних потрібне розбиття потоку на точки даних – самостійні об’єкти які можуть бути проаналізовані окремо (хоча не виключене одночасне розглядання декількох точок для спостереження за дінамікою процесів що спостерігаються.

В деяких системах точка даних у потоці є самоочевидною через природу даного потоку і співпадає з елементом потока (напр. окремий вимір що передається вимірювальною апаратурою), а в інших виділення точки даних є частиною задачі з конвертації необроблених (у.т.ч. аналогових) даних.

Чіткого критерію вибору оптімальних точок даних для аналізу не існує оскільки це залежить від контексту, у якому використовується точка даних. Однак є деякі загальні характеристики, які часто вважаються важливими:

Точки даних мають бути точними.

Вони мають бути репрезентативними для статистичної сукупності чи явища, що вивчається.

Вони повинні бути вільними від упередженості.

Вони мають бути своєчасними.

Вони повинні бути простими для розуміння та використання.

В контексті мониторингу потоків даних важливо зазничити що розмір і частота тоячок мають бути такими щоб засіб моніторингу міг вчасно їх обробляти. З одного боку ця вимога стосується запобіганню передачі монітору для поглибленого аналізу дублюючихся чи неінформативних даних, а з іншого боку – є вимогою до засобів моніторингу, які мають мати відповідні обсягу даних що достежуються обчислювальні потужності.

1.2.5 Кластеризація потоків

Деякі види потоків мають забагато точок для ефективної їх обробки, тому фактичні точки іноді об’єднуються у пакети для подальшої їх обробки Процес розбиття потоків на точки придатні до обробки та аналізу у безперервних птоках (напр. відеопотоках) називається кластерізацією.

Проблема кластеризації потоку даних визначається як:

Вхідні дані: послідовність з n точок метричного простору та ціле число k.

Вихідні дані: k центрів у наборі з n точок, щоб мінімізувати суму відстаней від точок даних до їхніх найближчих центрів кластерів.

Це потокова версія задачі k-медіани.

Існують наступні популярні алгоритми розбиття потоків даних на кластери:

STREAM [4] — це алгоритм для кластеризації потоків даних, описаний Гухою, Мішрою, Мотвані та О'Каллаханом, який забезпечує апроксимацію постійного фактора для проблеми k-медіан за один прохід і з використанням невеликого простору.

BIRCH: створює ієрархічну структуру даних для поступової кластеризації вхідних точок, використовуючи доступну пам’ять і мінімізуючи необхідний обсяг вводу-виводу. Складність алгоритму полягає в тому O(N), оскільки одного проходу достатньо, щоб отримати хорошу кластеризацію (хоча результати можна покращити, дозволивши кілька проходів).

COBWEB: це метод інкрементної кластеризації, який зберігає ієрархічну модель кластеризації у формі класифікаційного дерева. Для кожної нової точки COBWEB спускається по дереву, попутно оновлює вузли та шукає найкращий вузол для розміщення точки (використовуючи функцію утиліти категорії).

C2ICM: будує структуру кластеризації з плоским розділенням, вибираючи деякі об’єкти як початкові числа/ініціатори кластерів, а ненасіннєве значення призначається початковому значенню, яке забезпечує найвище покриття, додавання нових об’єктів може ввести нові початкові значення та фальсифікувати деякі існуючі старі початкові значення під час інкрементальної

кластеризації нових об’єктів, і члени фальсифікованих кластерів призначаються одному з існуючих нових/старих насінь.

1.3 Засоби моніторингу потоків даних

1.3.1 Ручний моніторінг

Ми розглянули загальні поняття що стосуються потоків даних. Тепер треба розглянути методи спостереження за ними. Найпростішим та найочевиднішим є «ручний» метор спостереження – коли людина-спостерігач безпосередньо спостерігає потік даних у форматі, який може сприймати людина. Цей метод використовується у тих випадках коли обсяг даних які необхідно спостерігати (розмір та тривалість потоку) є незначним та знаходження інформації що шукається у ньому не може бути алгоритмізовано за час та бюджет кращі ніж використання людини-спостерігача.

Простим прикладом такого моніторингу є охоронець що дивиться на камери спостереження: у багатьох випадках правила за якими активність що потребує додаткової уваги є поза межами спроможностей системи спостереження: в той час як навіть примітивний датчик руху може виконати початковий аналіз, який може бути достатнім для прийняття рішень в простих ситуаціях (напр. «на об’єкті спостереження не має бути нікого», але більш складні правила (напр. дії дозволені тим чи іншим особам на об’єкті спостереження в залежності від їх повноважень) можуть бути поза можливостями сучасних систем що є доступними для широких користувачів.

Інший приклад ручного спостереження – переклад легко оцифрованих даних у вигляд більш легкий для сприйняття людиною для прийняття спостерігачем рішень. Дані що важко спостерігати в необробленому вигляді, можна представити у стислому вигляді чи у вигляді графіку.

Розглянемо кілька прикладів інструментів що дозволяють подібні перетворення:

Microsoft Excel та подібні йому програми: дозволюють створення графіків багатьох видів з даних заповнених у таблиці. Ці засоби є досить легкими у використанні користувачем що не має досвіду у програмуванні, але використання результуючих файлів як частини існуючої системи для подальшої обробки чи комбінація з іншими файлами може бути проблематичною. Зазвичай подібного типу файли є кінцевим результатом що періодично видається системою моніторінгу, сумуючи спостереження за останній час.

Бібліотеки для створення графічних відтворень:

Matplotlib — це бібліотека для побудови графіків для Python. Він використовується разом із NumPy для забезпечення середовища, яке є ефективною альтернативою з відкритим кодом для MatLab. Його також можна використовувати з такими наборами графічних інструментів, як PyQt і wxPython.

Модуль Matplotlib вперше був написаний Джоном Д. Хантером. З 2012 року основним розробником є Майкл Дреттбум. Наразі Matplotlib ver. 1.5.1 є доступною стабільною версією. Пакет доступний у двійковому вигляді, а також у формі вихідного коду на www.matplotlib.org [5].

The Boost Graph Library (BGL) (C++) - Стандартизований загальний інтерфейс для обходу графів є надзвичайно важливим для заохочення повторного використання графових алгоритмів і структур даних. Частиною бібліотеки Boost Graph є загальний інтерфейс, який дозволяє отримати доступ до структури графа, але приховує деталі реалізації. Це «відкритий» інтерфейс у тому сенсі, що будь-яка бібліотека графів, яка реалізує цей інтерфейс, буде сумісна з загальними алгоритмами BGL та іншими алгоритмами, які також використовують цей інтерфейс. BGL надає деякі класи графів загального призначення, які відповідають цьому інтерфейсу, але вони не призначені бути

«єдиними» класами графів; звичайно будуть інші класи графів, які будуть кращими для певних ситуацій.[6]

Перевагою такого підходу є більш широкі можливості інтеграції результату та результат у форматі зображення. Недоліком є те що створення подібних відображень потребує більш глибоких знань мов програмування що виклристовуються і результуючий файл не зберігає контексту (вхідних даних).

Ручне спостереження є невід'ємною частиною майже будь-якої системи моніторингу але зазвичай є частиною комбінованого підходу, у більшості систем що мають справу з обробкою великого обсягу вхідних даних людина-спостерігач має справу з обробленими даними, даними що виділені системою або має підтвердити рішення системи моніторингу для дій.

1.3.2 Моніторінг на основі завчасно заданих правил

Значна частина MS працює за заданими користувачем або завчасно вбудованими правилами. Це можуть бути як прості правила (напр. відсутність чи наявність сигналу) так і складні (напр. сумарний розмір пакетів що проходять через вузол мережі що спостерігається за певний період з розміром викликаючим реакцію чи сигнал що змінюється в залежності від часу). Розглянемо кілька подібних автоматизованих систем моніторингу.

Zabbix [7] – це програмний інструмент із відкритим кодом для моніторингу IT-інфраструктури, такої як мережі, сервери, віртуальні машини та хмарні служби. Zabbix збирає та відображає основні показники.

Nagios [8] – це MS що має відкритий код, і є призначеною для моніторингу комп'ютерних систем та мереж: спостереження, контролю стану обчислювальних вузлів та служб, оповіщення адміністратора у тому випадку, якщо якісь із служб припиняють (або відновлюють) свою роботу.

Nagios спочатку була створена під ім'ям Netsaint, її основний розробник - Етано Галстад (англ. Ethan Galstad). Він і команда розробників займаються підтримкою системи, розробкою плагінів. Окрім того сторонні розробники створили ряд неофіційних плагінів для даної системи.

Спочатку Nagios була розроблена для Linux, але має сумісність з іншими unix-based іншими ОС, такими як Sun Solaris, FreeBSD, та інші.

Ці системи дозволяють задавати складні правила за якими спостерігаються процеси у комп'ютерних мережах і навіть автоматично виконуівати деякі прості дії якщо будуть зустрінутів певні критерії.

Також у деяких системах засоби моніторінгу є вторинними до їх основної але дозволяють більш ефективне їх використання, напр.

Jenkins - сервер автоматизації що має відкритий програмний код. Це допомагає автоматизувати частини розробки програмного забезпечення, пов'язані зі створенням, тестуванням і розгортанням, сприяючи безперервній інтеграції та безперервній доставці. Ця серверна система працює в контейнерах сервлетів, таких як Apache Tomcat. Система також підтримує чисельні інструменти контролю версій, зокрема AccuRev, Mercurial, Perforce, CVS, Subversion, Git, ClearCase та інших, і здібний виконувати проекти на основі Apache Ant, Apache Maven чи sbt, а також різні сценарії, та пакетні команди у ОС Windows. Частиною функціоналу є моніторинг «трубопроводів» - відображення результатів виконань наборів інструкцій у форматі що легко сприймається людиною. Система виконує прості спостереження за кодом результату виконання коду що запускається і відображає його як закодовану кольорами таблицю з рядками відповідаючими окремому запуску і стовпцями відповідаючими етапу. Також під час виконання тестів система виділяє тести що завершилися невдачно, що значно полегшує аналіз результатів.

1.3.3 Моніторінг з використанням систем машинного навчання

Інтелектуальний аналіз потоків даних (також відомий як потокове навчання) — це процес вилучення структур знань із безперервних швидких записів даних. Потік даних — це впорядкована послідовність екземплярів, які в багатьох програмах інтелектуального аналізу потоків даних можна прочитати лише один раз або невелику кількість разів, використовуючи обмежені обчислювальні можливості та можливості зберігання.

У багатьох MS інтелектуального аналізу потоків даних мета полягає в тому, щоб передбачити клас або значення нових екземплярів у потоці даних, маючи певні знання про членство в класі або значення попередніх екземплярів у потоці даних. Методи машинного навчання можна використовувати для автоматизованого вивчення цього завдання прогнозування з позначених прикладів. Часто концепції з поетапного навчання застосовуються для того, щоб впоратися зі структурними змінами, онлайн-навчанням і вимогами в реальному часі. У багатьох програмах, особливо в нестаціонарних середовищах, розподіл, що лежить в основі екземплярів, або правила, що лежать в основі їхнього маркування, можуть змінюватися з часом, тобто мета прогнозу, клас, який потрібно передбачити, або цільове значення, яке потрібно передбачити, можуть змінитися з часом. Цю проблему називають дрейфом концепції. Виявлення дрейфу концепції є центральною проблемою інтелектуального аналізу потоків даних. Інші проблеми, які виникають під час застосування машинного навчання до потокових даних, включають: частково та затримано позначені дані, відновлення після відхилень концепції та тимчасові залежності.

Інтелектуальний аналіз потоків даних можна вважати підгалуззю інтелектуального аналізу даних, машинного навчання та відкриття знань.

Серед найбільш відомих систем для аналізу потоків даних з використанням машинного навчання є:

MOA (Massive Online Analysis) - безкоштовне програмне забезпечення з відкритим вихідним кодом, призначене для видобутку потоків даних із концептуальним дрейфом, розробленим на Java. Він має кілька алгоритмів машинного навчання (класифікація, регресія, кластеризація, виявлення викидів і системи рекомендацій). Крім того, він містить метод попереднього оцінювання, методи дрейфу концепції EDDM, засіб читання реальних наборів даних ARFF і генератори штучних потоків як концепції SEA, STAGGER, обертову гіперплощину, випадкове дерево та функції на основі випадкового радіуса. MOA підтримує двонаправлену взаємодію з Weka (машинне навчання).

scikit-multiflow - Фреймворк машинного навчання для даних із кількома виходами/багатьма мітками та потоковими даними, реалізованим на Python. scikit-multiflow містить генератори потоків, методи навчання потоків для одноцільових і багатоцільових, концептуальні детектори дрейфу, методи оцінки та візуалізації. (розробку припинено)

StreamDM - фреймворк із відкритим вихідним кодом для видобутку великих потоків даних, який використовує розширення Spark Streaming основного API Spark. Одна з переваг StreamDM у порівнянні з існуючими фреймворками полягає в тому, що він отримує безпосередню користь від Spark Streaming API, який вирішує більшість складних проблем базових джерел даних, таких як несправні дані та відновлення після збоїв.

RapidMiner: комерційне програмне забезпечення для виявлення знань, інтелектуального аналізу даних і машинного навчання, яке також включає інтелектуальний аналіз потоків даних, вивчення концепцій, що змінюються в часі, і концепцію відстеження дрейфу (якщо використовується в поєднанні з додатком інтелектуального аналізу потоку даних (раніше: плагін Concept Drift))

RiverML: River — це бібліотека Python для онлайн-машинного навчання. Це результат злиття creme та scikit-multiflow. Амбіції River — стати основною бібліотекою для машинного навчання потокових даних.

Машинне навчання дозволяє працювати з складними системами класифікації, доповнюючи чи навіть повністю заміняючи інші методи.

1.3.4 Комбіновані методи

Комбінуючи методи моніторингу, можна досягти оптимальних за ціною (обчислювальний час та людино-години) та якістю (мінімізація кількості хибних висновків) результатів. Більшість розглянутих попередньо засобів моніторингу є в деякому сенсі комбінованими: людина приймає остаточне рішення щодо класифікації результатів та дій що мають виконуватися, хоча не завжди має можливість редагування результатів.

Комбіновані системи зазвичай розподіляють методи моніторингу між ти чи іншим способом спостереження, але зазвичай не дають можливості комбінувати їх: напр. спостереження вручну проводиться за результатами моніторингу що відбуваються на нижчих рівнях системи, а методи машинного навчання часто комбінуються з формально заданими правилами.

1.4 Нейромережі

Нейронні мережі є функціональною одиницею Deep Learning і, як відомо, імітують поведінку людського мозку для вирішення складних проблем, керованих даними.

Вхідні дані обробляються різними шарами штучних нейронів, складених разом, щоб отримати бажаний результат.

Від розпізнавання мовлення та розпізнавання осіб до охорони здоров’я та маркетингу, нейронні мережі використовувалися в різноманітних областях. [11]

В межах данної роботи найцікавішими є системи що виконують задачі категоризації, але тип задач що виконується не є повністю залежним від вихідного типу даних.

В той час як нейронні мережі виконують задачі, що є складними чи неможливими для програмування конвенційними методами (такі як генерація зодражень), не всі задачі для них є настільки складними, іноді інформація що аналізується є доволі простою (напр. вектор з кількох цифр для мереж що опрацьовують координати). Розмір як віхідних так і вихідних даних може бути яким завгодно (в межах розумного).

1.4.1 Стандартні нейронні мережі

Персептрон - нейронна мережа, яка застосовує математичну операцію до вхідного значення, надаючи вихідну змінну.

Мережі прямого зв’язку – багаторівнева нейронна мережа, де інформація рухається зліва направо, іншими словами, у прямому напрямку. Вхідні значення проходять через ряд прихованих шарів на шляху до вихідного шару.

Залишкові мережі (ResNet) – глибока мережа прямого зв’язку із сотнями рівнів.

1.4.2 Рекурентні нейронні мережі

Повторювані нейронні мережі (RNN) запам’ятовують раніше вивчені прогнози, щоб допомогти зробити майбутні прогнози з точністю.

Мережа довготривалої короткочасної пам’яті (LSTM) – LSTM додає додаткові структури або ворота до RNN для покращення можливостей пам’яті.

Мережа стану відлуння (ESN) – тип прихованих шарів RNN, які рідко з’єднані.

1.4.3 Згорткові нейронні мережі

Згорткові нейронні мережі (CNN) – це тип прямої мережі, який використовується для аналізу зображень і обробки мови. Є приховані згорткові шари, які формують ConvNets і виявляють шаблони. CNN використовують такі функції, як грані, форми та текстури для виявлення візерунків. Приклади CNN включають:

AlexNet – містить кілька згорткових шарів, призначених для розпізнавання зображень.

Група візуальної геометрії (VGG) - VGG схожа на AlexNet, але має більше шарів вузьких звивин.

Капсульні мережі – містять вкладені капсули (групи нейронів) для створення більш потужної CNN.

1.4.4 Генеративні змагальні мережі

Генеративні суперницькі мережі (GAN) – це тип неконтрольованого навчання, де дані генеруються на основі шаблонів, які було виявлено на основі вхідних даних. GAN складається з двох основних частин, які конкурують одна з одною:

Генератор – створює синтетичні дані на етапі навчання моделі. Він візьме випадкові набори даних і згенерує трансформоване зображення.

Дискримінатор - вирішує, чи є створені зображення підробленими чи справжніми.

GAN використовуються, щоб допомогти передбачити, яким може бути наступний кадр у відео, створити текст у зображення або перевести зображення в зображення.

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі були розглянути типи потоків даних та систем їх моніторингу.

Різновиди можливих потоків даних, їх характеристики та задачі які виконуються шляхом їх спостереження.

Ручний моніторинг має перевагою те що не виникає потреби в переводі потоку даних з аналогового в цифровий вигляд (якщо початковий формат даних не є цифровим), спостерігач-людина може використовувати неформально задані правила та здоровий глузд для знаходження проблем а також має значно більші юридичні повноваження ніж автоматизована система. Найбільш розповсюджений випадок – спостереження за одним чи кількома відеоканалами з метою охорони. Проблема цього методу – залежність від людського фактору (втома, халатність, навмисний саботаж виконавцем) та лінійно зростаюча вартість при масштабуванні. Також значною проблемою є втрата експертизи (надійності знаходження та класифікації проблем) у випадку заміни людини – оператора коли критерії подій що потребують реакції під час спостереження не є чітко сформульованими та є NTE оператора.

Моніторинг автоматичними системами з чітко заданими правилами – легко масштабується, є надійним і дешевим у підтримці. Розповсюджені системи з таким підходом – мережеві екрани та медична апаратура підтримки життєдіяльності. З недоліків такого підходу – необхідність максимально повного покриття можливих ситуацій правилами та точного їх формулювання. Така система не може працювати з нечітко визначеними правилами і може мати проблеми з змінами правил через особливості реалізації.

Моніторинг системами машинного навчання дозволяє отримати масштабованість та швидкодію характерні для автоматизованих систем без потреби в чітко сформульованих у вигляді який може сприйматися комп’ютерною системою правилах. Останнім часом подібні системи

використовуються для вдосконалення систем відеоспостереження, економічного аналізу (напр. під час високочастотного трейдінгу на біржі), систем медичної діагностики. Недоліком системи є її непрозорість і необхідність якісноих тренувальних даних в достатній кількості, які не завжди можна зібрати перед запуском системи.

Комбіновані методи дозволяють об’єднати підходи, нейтралізуючи слабкі місця того чи іншого підходу. Як приклад – система відеоспостереження з датчиками руху, яка потребує уваги спостерігача-людини лише у випадку знайденого руху, що дозволяє одному оператору спостерігати значно більшу кількість відеоканалів.

Для цілей даної роботи ми розглянули розглянути комбіновані методи моніторингу потоків, зокрема комбінацію ручного моніторингу та машинного навчання. Це дозволить порівняти ефективність правил що задаються вручну і правил що знаходяться методами машинного навчання та оцінити загальну доцільність методів машинного навчання у системах моніторінгу в залежності від обсягу даних та складності правил.

РОЗДІЛ 2. ОГЛЯД АЛГОРИТМІВ ТА ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ.

2.1 Загальні вимоги до системи

Система яка буде розроблена в процесі даної роботи має бути частиною системи безперервної інтеграції програмного забезпечення для вбудованої системи: аналізувати результати тестування продуктивності вбудованих систем. Система має аналізувати результати тестів що надходять в контексті історії вимірів (заміри швидкодії певних операцій) та сортувати їх по наступним категоріям:

- Добре (вимір є в межах норми і не потребує додаткової уваги)
- Увага (вимір є в межах норми але показує підозрілі тенденції що потребують додаткової уваги)
- Проблема (значення є поза нормою і потребує створення квитку та сигналізує проблему в ПЗ)
- Помилка (значення є поза нормою і є неможливим результатом, що сигналізує помилку вимірювання чи тесту)

Користувач має мати можливість редагувати висновки системи вручну а сама система має враховувати результати ручного вводу при подальшому аналізі.

Для даної роботи система буде робити висновки лише за допомогою системи машинного навчання, так як першочерговою ціллю стоїть оцінка ефективності алгоритмів машинного навчання для подібних задач.

Вхідні дані вимірів будуть емульовані, а результати будуть зберігатися локально – вбудова системи моніторінгу у систему безперервної інтеграції є потенційним практичним використанням даного проекту але в цілях дотримання авторських прав та отримання більш репрезентативних тестових результатів будуть використані «заглушки».

2.2 Вибір інструментів

2.2.1 Вибір мови програмування

Python – високорівнева мова програмування що має широкі можливості в плані обробці даних. Її було обрано для даної системи через наявність численних тематичних бібліотек та добру ознайомленість з нею автора дипломної роботи.

Python став домінуючою мовою програмування для впровадження систем машинного навчання, і це не дарма. Його простота, універсальність і широка бібліотечна підтримка роблять його найкращим вибором для розробки систем машинного навчання.

Однією з ключових причин, чому Python користується перевагою в спільноті машинного навчання, є його простота використання та читабельність. Python має чистий та інтуїтивно зрозумілий синтаксис, який дозволяє розробникам висловлювати складні ідеї в стислій формі. Це полегшує як початківцям, так і досвідченим програмістам розуміння та підтримку коду машинного навчання. Крім того, велика та активна спільнота Python надає обширну документацію та ресурси, що спрощує вивчення та вирішення проблем.

Універсальність Python є ще однією важливою перевагою машинного навчання. Це мова програмування загального призначення, що означає, що її можна використовувати для широкого спектру програм, крім машинного навчання. Ця гнучкість дозволяє розробникам використовувати існуючий код і бібліотеки Python, підвищуючи продуктивність і скорочуючи час розробки. Крім того, сумісність Python з іншими мовами програмування забезпечує бездоганну інтеграцію з існуючими системами, що робить його ідеальним

вибором для розгортання моделей машинного навчання у виробничих середовищах.

Широка бібліотечна підтримка, доступна в Python, кардинально змінює гру для практиків машинного навчання. Бібліотеки, такі як NumPy, pandas і scikit-learn, надають потужні інструменти для обробки даних, аналізу та навчання моделі. Ці бібліотеки спрощують складні завдання, дозволяючи розробникам зосередитися на високорівневих концепціях і алгоритмах, а не на низькорівневих реалізаціях. Крім того, найпопулярніша бібліотека машинного навчання Python, TensorFlow, забезпечує надійну екосистему для побудови та розгортання моделей глибокого навчання, що ще більше розширює можливості Python у цій галузі.

Ще однією важливою перевагою Python для машинного навчання є потужна підтримка візуалізації та аналізу даних. Такі бібліотеки, як Matplotlib і Seaborn, дозволяють розробникам створювати інформативні та візуально привабливі графіки та діаграми, допомагаючи досліджувати дані та оцінювати моделі. Інтеграція Python із Jupyter Notebook, інтерактивним обчислювальним середовищем, сприяє безперебійному робочому процесу, дозволяючи розробникам поєднувати код, візуалізацію та документацію в одному блокноті.

Крім того, прийняття Python у науковому співтоваристві призвело до появи численних дослідницьких статей, навчальних посібників і попередньо навчених моделей, що полегшує практикам використання найсучасніших методів і включення їх у власні системи.

Підсумовуючи, простота, універсальність, широка бібліотечна підтримка та сильна спільнота роблять Python чудовим вибором для розробки систем машинного навчання. Його простота використання в поєднанні з наявністю потужних бібліотек дозволяє розробникам швидко створювати прототипи та розгортати складні моделі машинного навчання. Оскільки попит на машинне навчання продовжує зростати, популярність Python у цій галузі, ймовірно,

збережеться, що робить його розумною інвестицією для тих, хто прагне створювати надійні та ефективні системи машинного навчання.

2.2.2 Вибір бібліотек та технологій машинного навчання

Як бібліотека машинного навчання буде використана TensorFlow. Ця платформа машинного навчання має гарну підтримку та добре інтегрується з python аплікаціями.

TensorFlow — це платформа машинного навчання з відкритим кодом, розроблена Google. Він надає комплексну екосистему інструментів і бібліотек для створення та розгортання моделей машинного навчання. Він підтримує різні архітектури нейронних мереж і пропонує API високого рівня для розробки моделей, а також можливості нижчого рівня для детального керування. TensorFlow набув широкої популярності та широко використовується як у дослідженнях, так і в промисловості для таких завдань, як розпізнавання зображень, обробка природної мови та глибоке навчання [10]. Приклад архітектурм можна побачити на рис. 2.1

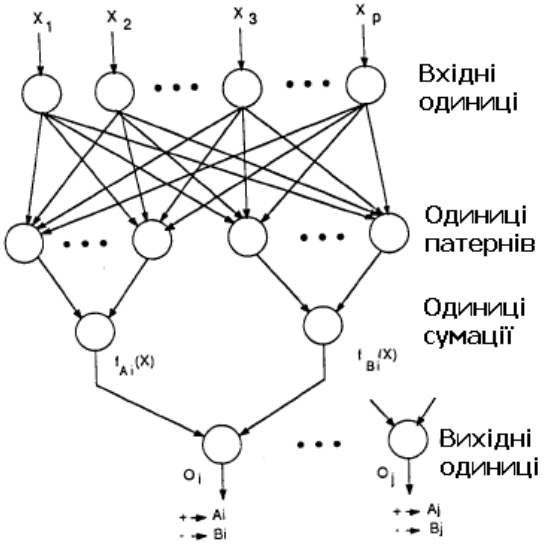


Рисунок 2.1 – Приклад архітектури нейромережі що вирішує задачу класифікації

TensorFlow особливо відомий своїми ефективними обчисленнями на великомасштабних наборах даних за допомогою обчислювальних графіків і тензорів.

Для поставлених перед нами задач буде достатнім використання звичайної нейромережі глибогого навчання з кількома прихованими щільними шарами (Багатошаровий персептрон), так як і вхідна і вихідня інформація є досить простими структурно: вектор на вході та одиничне значення на виході.

Універсальна апроксимаційна теорема [12] говорить: «Нейронна мережа з прихованим шаром може представляти будь-які функції» (з достатньою кількістю правильних параметрів). Однак у більшості випадків у нас недостатньо даних для наших проблем. Якщо ми це зробимо, ми можемо вирішити ці проблеми за допомогою таблиці перегляду. Тому нам потрібно машинне або глибоке навчання, щоб вивчити наближені функції для вирішення проблеми.

Було показано, що глибокі мережі перевершують дрібні мережі за однакової кількості параметрів і навчальних даних; глибокі мережі потребують експоненціально меншої кількості параметрів і складності вибірки для досягнення аналогічної продуктивності [13]. Але чому?

Один прихований рівень застосовує функцію нелінійної активації до лінійної комбінації вхідних даних. Кожен прихований шар можна розглядати як модульну функцію. Глибока мережа з багатьох прихованих рівнів схожа на стек кількох функцій, які можуть досягати більш складних функцій з тією ж кількістю параметрів порівняно з дрібною мережею. Іншими словами, глибокі мережі використовують параметри більш ефективно. Слід зазначити, що якщо тренувальних даних надмірно достатньо або проблема досить проста для вивчення дрібних мереж, тоді ви можете побачити однакову продуктивність між глибокими та дрібними мережами.

Для прикладу класифікації зображень і згорткових нейронних мереж початкові рівні вивчають функції низького рівня, такі як ребра та криві, а пізніші рівні вивчають функції високого рівня, які повторно використовують функції низького рівня для створення форми об’єкта. Подібне явище було виявлено в розпізнаванні мови. Початкові рівні вивчають манеру артикуляції, а наступні рівні вивчають різні фонеми на основі комбінацій різних ознак низького рівня. Тому глибокі мережі з багатьма прихованими шарами полегшують вивчення таких функцій високого рівня, ніж дрібні мережі.

Таким чином, глибокі мережі перевершують поверхневі мережі (з однаковою кількістю параметрів і даних), оскільки кілька прихованих рівнів призводять до багаторазово використовуваних модульних функцій, які забезпечують кращу ефективність у використанні параметрів і вивчення зв’язку між входом і виходом.

2.2.3 Вибір засобів візуалізації

Для візуалізації (побудова графіків) буде використана бібліотека **matplotlib**. Ця бібліотека є стандартною для візуалізації під час роботи з даними у python-аплікаціях. Вона дозволяє будувати широкий спектр графічних репрезентацій та легко інтегрується з іншими бібіліотеками для роботи з даними, такими як **numpy**.

Окрім того, дана бібліотека дозволяє швидку інтеграцію з популярними GUI та дає можливості для налаштування шрифтів і стилів, що може бути важливим при розробці комерційних продуктів.

В нашому випадку ключовими факторами є простота використання і достатній набір можливих форматів репрезентації даних цією бібліотекою.

На рис. 2.2 можна побачити приклади репрезентації інформації за допомогою даної бібіліотеки.

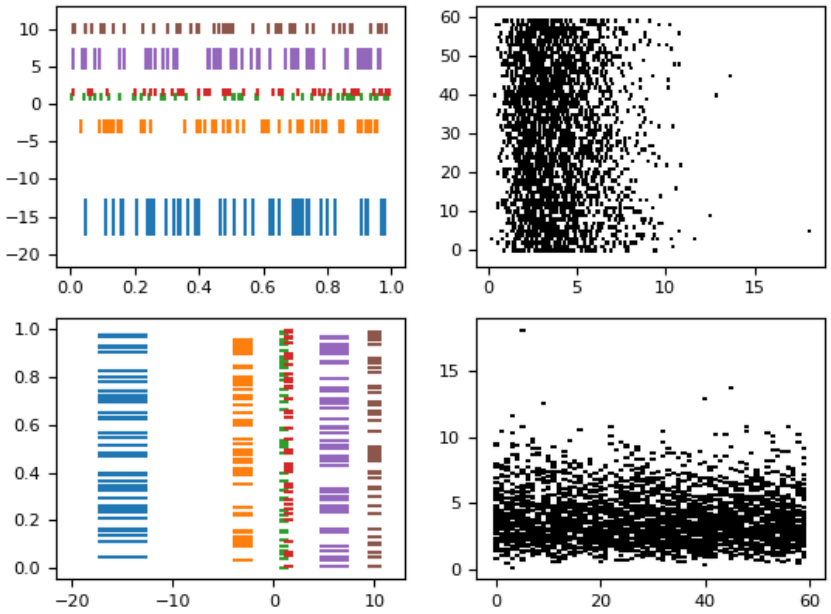


Рисунок 2.2 – Приклади графіків побудованих з допомогою matplotlib

2.2.4 Вибір інструменту для взаємодії з користувачем

Tkinter — це бібліотека Python, яка забезпечує простий і ефективний спосіб створення графічних інтерфейсів користувача (GUI). Ось деякі ключові переваги використання Tkinter:

Простий у вивченні та використанні: Tkinter має відносно простий та інтуїтивно зрозумілий API, що робить його доступним як для початківців, так і для досвідчених розробників. Його синтаксис простий, що дозволяє розробникам швидко створювати графічні інтерфейси без великих знань кодування.

Сумісність із різними платформами: Tkinter входить до стандартних дистрибутивів Python, що робить його доступним у основних операційних системах, таких як Windows, macOS і Linux. Це гарантує безперебійну роботу програм із графічним інтерфейсом користувача, розроблених за допомогою Tkinter, на різних платформах.

Широкий вибір віджетів: Tkinter пропонує багатий набір вбудованих віджетів, таких як кнопки, мітки, поля введення, прапорці тощо. Ці віджети можна легко налаштувати та поєднати для створення інтерактивних та візуально привабливих GUI. (рис.2.3)

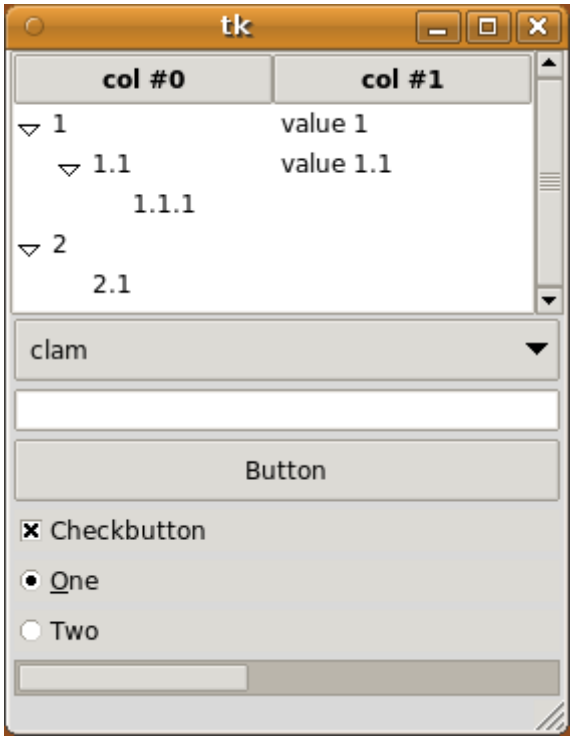


Рисунок 2.3 – Приклади GUI розробленого за допомогою tkinter

Налаштування та оформлення тем: Tkinter надає параметри для налаштування зовнішнього вигляду віджетів, включаючи кольори, шрифти та розміри. Він також підтримує тематизацію, що дозволяє розробникам створювати послідовні та візуально цілісні інтерфейси.

Інтеграція з Python: Tkinter легко інтегрується з Python, дозволяючи розробникам використовувати всю потужність мови для розробки програм. Він надає механізми для обробки подій, зв’язування даних і взаємодії з іншими бібліотеками Python.

Активна спільнота та ресурси: Tkinter має велике та активне співтовариство розробників, а це означає, що доступно багато ресурсів, включаючи навчальні посібники, документацію та приклади. Це полегшує пошук допомоги та вивчення найкращих практик.

Загалом Tkinter пропонує зручний і зручний спосіб створення додатків із графічним інтерфейсом на Python, що робить його популярним вибором для розробників, які хочуть створювати додатки для настільних комп'ютерів із графічним інтерфейсом.

Ще одним плюсом для просунутого користувача є те що python – інтерпретована мова і користувачу простого додатка нескладно самостійно зробити потрібні зміни. В той час як це не є важливим фактором для комерційного ПЗ, це є фактором «за» для внутрішніх інструментів для розробки ПЗ чи наукової діяльності.

Користувач може легко додати нові дані що відображаючи просто змінивши сирцевий код, якщо додаток постачається в некомпільованому вигляді.

ВИСНОВОК ДО РОЗДІЛУ 2

В другому розділі ми обрали інструменти що дозволять нам створити систему, що дозволить ефективно комбінувати ручний моніторинг за допомогою побудованих графіків і аналіз тих самих даних за допомогою машинного навчання.

Вибір Python як мови розробки дозволить мати доступ до численних бібліотек для роботи з даними а також пришвидшить швидкість розробки і подальшої модифікації системи: інтерпретована мова дозволяє швидко вносити та тестувати необхідні зміни.

Вибір кросплатформенної бібліотеки для створення користувацького інтерфейсу дозволить легко адаптувати систему під різні ОС.

Як система машинного навчання була обрана Tensorflow – ця система має можливість оперувати багатьма типами нейромереж і дозволяє широкий спектр операцій, а також збереження і завантаження моделей і наборів даних, що є важливим для імплементації системи, особливо в умовах використання штучно згенерованих даних в якості вхідних.

РОЗДІЛ 3. ДЕТАЛІ РОЗРОБКИ СИСТЕМИ

Відповідно до досліджених матеріалів та обраних інструментів ми переходимо до розробки програмного продукту.

3.1 Розробка програмних компонентів MD

Макетовані дані що використовуються в даній програмі мають емулювати API що повертає значення виміру та базу даних що зберігає історію таких вимірів. Для цього будуть імплементовані класи що можуть виконувати задачі генерації і зберігання інформації подібної до реальної. Припущення з яким ми працюємо – що правила за якими заповнюються датасети не будуть доступні у системі що має робити подальші висновки так як є не формалізованою NTE, тому вони можуть бути використані лише для генерації тренувальних даних (історія спостережень і аналізу) але не під час власно роботи системи.

3.1.1 Клас DataPoint

Це – утілітарний клас для роботи з даними на початковому етапі до перетворення в масиви що придатні для використання для тренування моделі. містить поля value – змінна типу список що містить останні 6 значень вимірюваної величини та resolution – висновок з аналізу значень. Висновком за замовчанням є NOT_SET (0).

3.1.2 Клас Resolution

Цей клас наслідується від типу Enum і є змінною з обмеженою кількістю можливих значень типу int кожній з яких відповідає один з висновків аналізу значень. На рис. 3.1 представлено список значень які можуть бути використані як висновок

3.1.3 Клас ValueGen

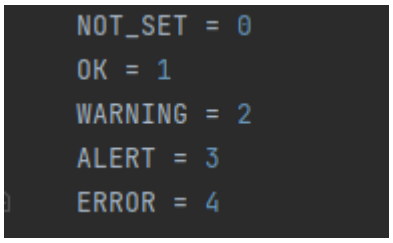


Рисунок 3.1 – Можливі стани класу Resolution

Клас ValueGen є простою імплементацією класа-генератора значень подібних тим що можуть бути згенеровані в результаті вимірювань продуктивності в CI системі.

Ці значення мають бути випадковими і змінюватися в межах від 0 до нескінченості, з найбільш ймовірними значеннями що можуть бути в «нормальному» діапазоні від 1 до 1000 та наближені до попередніх значень та «ненормальними значеннями» вище 1000 чи дорівнюючими 0.

Це досягається шляхом генерації попереднього значення як зміни попереднього значення на довільний процент (з більш ймовірними малими і менш ймовірними великими змінами) На рис 3.2 можна побачити можливі діапазони змін попереднього значення (1 = 0-100%, 0.15 = 0-15%). Ймовірності кожного класу змін (з найменшої по найбільшу є 0.85, 0.13, 0.2)

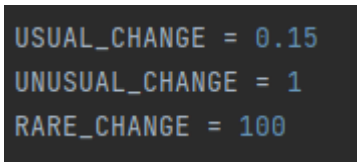


Рисунок 3.2 – Можливі діапазони змін

Якщо значення є поза межами нормального діапазону воно має 75% шанс повернутися до норми.

Клас містить наступні атрибути і методи:

- __init__ - ініціація класу. В цей метод також передається перше значення з якого будуть генеруватися наступні
- last_value – останнє виміряне значення
- gen_next – генерує і повертає нове значення а також заміняє останнє значення новим (так що наступний виклик цього методу згенерує зміни основані вже на новому значенні)
- rand_change – повертає значення, основане на останньому значенні, зміненому згідно вищеописаній логіці.

3.1.4 Скрипт values_to_dataset

Скрипт values_to_dataset використовується для генерації кількох наборів правил що можуть бути використані для порівняння ефективності машинного навчання для кодифікації NTE різних ступеней складності:

Набір можливих використаних правил:

- within_bounds
- not_rounded
- is_valid
- is_stable
- not_accelerating
- not_stuck
- not_growing

Ці правила – індівідуальні перевірки за різними параметрами, що в результаті повертають Resolution того чи іншого типу (рис.3.3 як приклад)

```python
# Відхилення від середнього проміж минулих 5 значень не більше ніж 20%
#
def is_stable(datapoint) -> Resolution:
    avg = sum(datapoint[:5])/5
    if abs(datapoint[5] - avg) < avg*0.2:
        return Resolution.OK
    else:
        return Resolution.WARNING
```

Рисунок 3.3 – Приклад правила

Набори правил:

- ruleset0
- ruleset1
- ruleset2
- ruleset3

Набори правил – то списки (list) правил що містять 2-7 попередньо зазначених правил

Скрипт використовує наступні методи:

- check_for_rules – перевірка індівідуальної точки даних на відповідність правилам з набору правил, повертає найбільший Resolution (напр. якщо 3 правила повертають 0 “OK” а одне – 3 “ALERT” то метод поверне 3
- mark_data_with_ruleset – перевірити масив з точок даних за допомогою check_for_rules і заповнити результат анлізу в кожній з них.
- create_dataset – створити датасет tensorflow.data.Dataset на основі масиву даних і зберегти його на жорсткий диск (рис. 3.4). Пізніше ці дані можна використати для навчання нейромережі та завантажити у аплікацію для відображення чи редагування

Основний виконавчий метод скрипта використовує послідовність даних раніше згенеровану за допомогою ValueGen і створює окремий датасет для кожного з наборів правил.

Пізніше ці дані можуть бути використані як для оцінки ефективності машиннного навчання для з використанням лише моделі так і в якості завантажених даних в застосунку що дозволяє їх редагування та розширення.

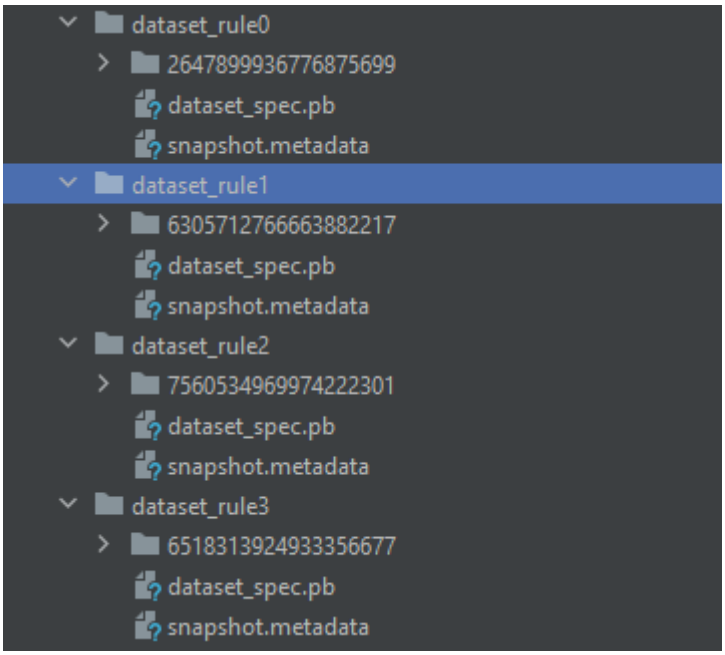


Рисунок 3.4 – Результат виконання

3.2 Розробка програмних компонентів СМ

Далі буде описано основний клас та допоміжний тестувальний скрипт для роботи з моделлю що має передбачати правила за якмим

3.2.1 Клас Model

Клас Model є основним класом для роботи з датасетами та генерації моделей на їх основі. Цей клас є інтерфейсом для роботи з tf.data.Dataset і tf.keras.models, адаптований під умови роботи і містячий основні методи для

завантаження датасету та перетворення його на придатні для редагування та тренування формат, тренування на його основі послідовної моделі та її збереження чи завантаження. На рис 3.5 можна побачити діаграму класу.

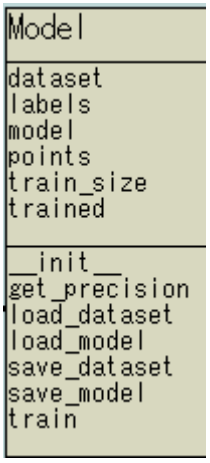


Рисунок 3.5 UML діаграма класу Model

Клас Model містить наступні атрибути та методи:

- dataset – атрибут типу tf.data.Dataset, дані для тренування що завантажуються
- train_size – розмір тренувального набору (використовуються останні train_size вимірів з завантаженого набору даних чи увексь набір даних якщо він менший)
- model - атрибут типу tf.keras.models.Sequential, модель для передбачень результатів аналізу що тренується на існуючих даних чи завантажується з файлу відповідного обраному джерелу даних
- points – numpy масив що відображає точки даних, який може редагуватися
- labels - numpy масив що відображає збережені результати аналізу точок даних, який може редагуватися
- trained – bool змінна що відображає наявність натренованої моделі
- load_dataset – обрати джерело даних, в умовах макетних даних то є завантаженням датасету з жорсткого диску

- save_dataset – зберегти дані. В умовах використання макетних даних то є збереження поточних точок та результатів їх аналізу на жорсткий диск
- train – тренування нейромережі (багатошаровий персептрон) на основі існуючих даних. (рис. 3.6)

- get_precision – оцінка точності моделі на основі завантажених даних
- save_model – зберегти модель
- load_model – завантажити модель

```
self.model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, input_shape=[6, ], activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(5),
    tf.keras.layers.Softmax()
])
self.model.compile(optimizer=tf.keras.optimizers.Adam(),
                   loss=tf.keras.losses.SparseCategoricalCrossentropy(),
                   metrics=['accuracy'])
```

Рисунок 3.6 Архітектура моделі навчання представлена в програмному коді

3.2.3 Скрипт generate_models

Даний скрипт використовується для тестування ефективності різних конфігурацій нейромережі, тренуючи та оцінюючи модель на згенерованих наборах даних з різною кількістю епох, розміром пачки даних та розміром тренувального набору даних. Детальніше результати виконання цього скрипту я розгляну в розділі 4

3.3 Розробка програмних компонентів GUI

Далі буде описано основний клас що відповідає за взаємодію апліації з користувачем та вивід графічного відображення даних.

Клас Window

Цей клас (рис 3.7) відповідає за взаємодію моделі з користувачем і генератором даних і є розширенням базового класу tkinter Tk.

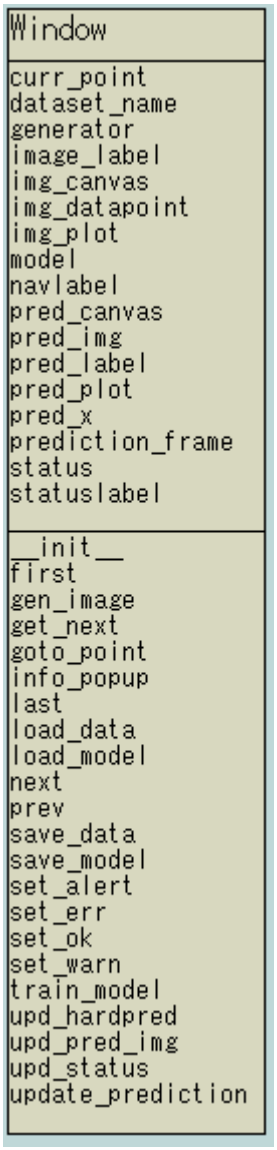


Рисунок 3.7 UML діаграма класу Window

В класі представлени наступні атрибути і методи:

Атрибути:

- curr_point – шндекс поточної
- dataset_name

- generator – об’єкт ValueGen який використовується як заміна вхідного потоку даних
- image_label – назва набору даних який в даний момент підвантажено
- img_canvas, img_datapoint, img_plot – об’єкти що використовуються для побудови графічного відображення поточної точки даних (останнього виміра та історії попередніх 5 вимірів) та інтеграції tkinter з matplotlib
- model – об’єкт-модель через який відбувається взаємодія з датасетами та моделями для передбачень
- navlabel – поточна позиція (відображена точка даних в наборі даних)
- pred_canvas, pred_img, pred_plot, pred_x – об’єкти використані для відображення розподілення ймовірностей передбачення, згенерованого моделлю
- pred_label, prediction_frame – кольорово-текстове відображення найбільш ймовірного варіанту згідно моделі
- status, statuslabel - кольорово-текстове відображення результату аналізу що був завантажений або введений користувачем

Методи:

- __init__ - ініціалізація вікна та завантаження зображення - плейсхолдера
- first – переміщення на першу точку даних з завантажених
- gen_image – генерація зображення згідно поточно обраній точці даних
- get_next – отримання точки даних з не визнваченим результатом аналізу від генератору даних і додання її в кінець. Переміщення на цю точку.

- goto_point – переміщення в задану точку даних. Відображення графічної репрезентації і результату аналізу (статусу) точки. За наявності моделі – генерація і відображення передбачення для цієї точки.
- info_popup – виклик вспливаючого вікна (зокрема у випадку помилки)
- last – навігацію на останню з доступних точок даних
- load_data – завантаження набору даних з жорсткого диску
- load_model – завантаження натренованої моделі передбачень з жорсткого диску
- next – навігація на наступну точку даних
- prev - навігація на попередню точку даних
- save_data – збереження набору даних на жорсткий диск
- save_model - збереження натренованої моделі передбачень на жорсткий диск
- set_alert – зміна статусу аналізу обраної точки даних на 3
- set_err – зміна статусу аналізу обраної точки даних на 4
- set_ok – зміна статусу аналізу обраної точки даних на 1
- set_warn – зміна статусу аналізу обраної точки даних на 2
- train_model – тренування моделі машинного навчання на поточних даних
- upd_hardpred – оновлення відображення найймовірнішого результату аналізу з передбачених моделлю
- upd_pred_img – оновлення графічного відображення результатау аналізу моделлю
- upd_status – відображення збереженого результату аналізу точки
- update_prediction – отримання і оновлення передбачень для поточної точки

ВИСНОВОК ДО РОЗДІЛУ 3

В результаті проведеної роботи щодо розробки програмного забезпечення системи моніторингу даних та виявлення підозрілих чи критичних точок можна виокремити наступне:

- Реалізовано раніше обгрунтовану (у Розділі II) нейромережу глибокого навчання.
- Реализовано генерацію макетних даних аналогічним тим що надходять з реальних систем CI
- Реализовано GUI який дозволяє завантажувати, генерувати дані та
- Наведено UML діаграми розроблених класів.
- Детально описано атрибути, методи та властивості класів.
- Були продемонстровані фрагменти коду як скріншоти з середовища розробки що було використано для розробки.

РОЗДІЛ 4. ДОСЛІДЖЕННЯ ТА АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ

Базуючись на розроблений системі моніторингу даних та виявлення підозрілих чи критичних точок ми можемо дослідити ефективність заміни NTE людини-спостерігача на передбачення, зроблені на основі багатошарового персептрону що був натренований на даних які були оброблені спостерігачем в залежності від складності правил.

4.1 Дослідження складності обробки різних наборів правил

У табл. 4.1 я навів абсолютні і відносні часи обчислення відповідності правилам перевірки ста тисяч згенерованих значень для різних наборів. Так як різні правила мають різну обчислювальну складність, як міру складності набору правил ми будемо використовувати час обчислення відносно найшвидшого та найпростішого з них. Ним є набор 0, який видає значення:

4 Error якщо останнє значення =0,

1 OK якщо останнє значення в діапазоні [1, 1000],

3 Alert якщо поточне значення більше 1000 і не враховує історію значень.

Порівняно з ним інші набори правил є більш складними та орієнтуються як на поточне так і на історичні значення а також на похідні з них виміри.

Із зростанням складності наборів також хростає кількість елементів вектору що впливають на результат аналізу та складність формул (просте порівняння у наборі 0 проти кількох аріфметичних операцій у більшості правил набору 3).

Таблиця 4.1 –Відносна обчислювальна складність наборів правил

Кількість правил	Час обробки 100000 вимірів, с	Відносна складність (кратність найшвидщому часу обробки)
2	0.115	1
3	0.158	1.382
4	0.365	3.182
7	0.718	6.26

Можна бачити що в силу різної складності обчислення різних правил залежність складності обчислення набору правил від їх кількості не є линійною, тому в подальших вимірах будемо використовувати відносну складність як первісну характеристику набру правил що досліджуються системою.

4.2 Налаштування моделі для найефективнішого передбачення

Згенерувавши модельні дані та побудувавши датасети з них, ми дослідимо відносну ефективність навчання для різних кількостей епох та розмірів пачек тренувальних даних для налаштування моделі. Ми використаємо найскладніший набір правил та розмір тренувального набору 1000 (для пришвидшення обчислення)

У таблиці 4.2 ми побачимо результати оцінки точності моделей в залежності від кількості епох (epoch) та розміру пачки даних (batch).

Таблиця 4.2 – Калібровка моделі

Epochs	Batch size	Точність
10	10	0.661
10	20	0.697
10	50	0.665
10	100	0.694
50	10	0.812
50	20	0.812
50	50	0.751
50	100	0.790
100	10	0.840
100	20	0.835
100	50	0.835
100	100	0.819
500	10	0.887
500	20	0.881
500	50	0.872
500	100	0.821

В той час як зменшення розміру пачки збільшує точність, чя зміна незначна відносно збільшення кількості епох, в той час як швидкість навчання моделі прямо пропорційна. Тому для навчання моделі було обрано 500 епох та розмір пачки 50.

4.3 Порівняльний аналіз ефективності машинного навчання та потрібної для досягнення ефективності кількості вхідних даних

Відкалібрувавши модель, можна дослідити залежність точності передбачень від складності правил розмітки вхідних даних та розміру тренувального датасету. Модель була запущена з раніше визначеними параметрами кількості епох і розміру пачки 500 та 50 що дають оптимальне співвідношення точності і часу виконання. Час виконання не вимірюється так як не є важливим (перетренування моделі буде відбуватися нечасто і не є важливим в даному випадку, поки є в межах 5 хв) Результати цього дослідження можна побачити у таблиці 4.3

Таблиця 4.3 – Вплив кількості ітерацій на точність і час виконання алгоритму

Складність правил розмітки	Кількість тренувальних точок даних	Точність передбачень моделі
1	500	0.943
1	1000	0.956
1	5000	0.979
1	10000	0.991
1.382	500	0.855
1.382	1000	0.883
1.382	5000	0.965
1.382	10000	0.967
3.182	500	0.868
3.182	1000	0.912
3.182	5000	0.962
3.182	10000	0.98
6.26	500	0.832
6.26	1000	0.865
6.26	5000	0.965
6.26	10000	0.974

Очікувано, точність передбачень зростає з розміром набору тренувальних даних, але може досягти достатньо високої точності щоб вважатися надійною лише з тисячами точок даних для більш складних наборів правил. Це можна пояснити тим що для більш складних та численних правил буде менше випадків де вони змінять результат аналізу, тобто на малих вибірках частина правил може відтворитися недостатню для навчання кількість разів. Зміна швидкості навчання моделі може підвищити ефективність навчання на малих вибірках але збільшить вразливість до шуму.

4.4 Огляд інтерфейсу програми

На рисунку 4.1 зображений розроблений програмний інтерфейс користувача на першому запуску програми. Користувач має обрати джерело даних з якого буде отримані початкові дані для розмітки та/або аналізу

Рисунок 4.1 – Програмний інтерфейс користувача на старті програми

Застосунок є аплікацією однієї сторінки тому частина полей заповнена даними-прикладом або пусті. У відсутності завантаженого набору даних чми

моделі ми маємо виключно плейсхолдер що показує як буде виглядати завантажена точка даних. Від користувача очікується що він завантажить набір даних за допомогою LOAD DATA (в нашому випадку відкриється селектор в якому треба обрати папку що містить датасет.

На рисунку 4.2 можна побачити вигляд програми з завантаженим датасетом але відсутною моделлю: поля «передбаячення» пусті, так як відсутна натренована модель, але кнопки навігації активні. На цьому етапі

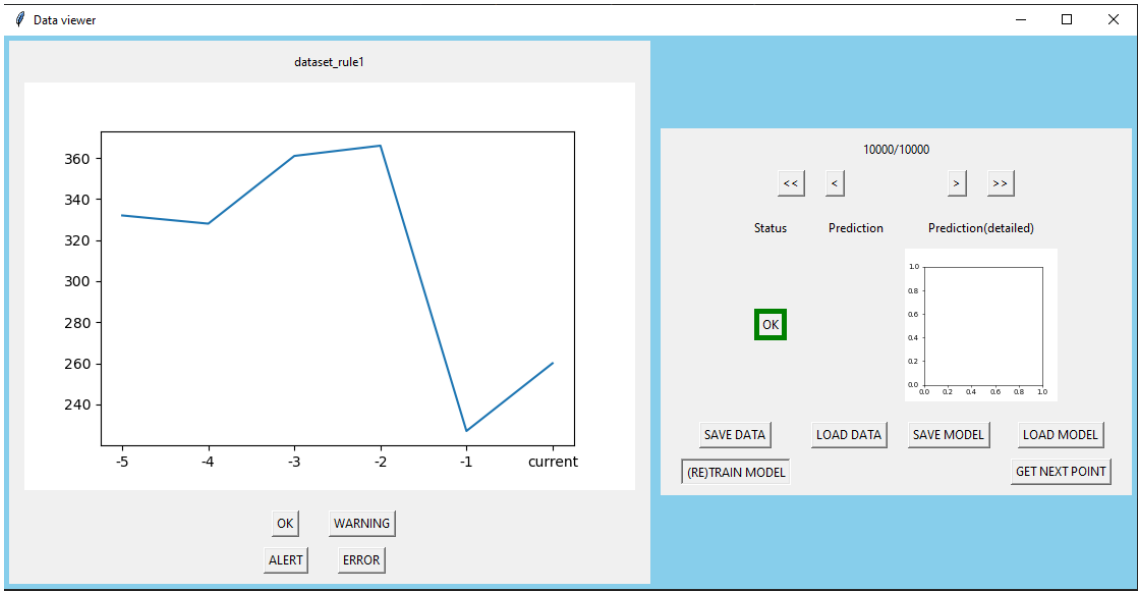


Рисунок 4.2 – Програмний інтерфейс користувача з завантаженим датасетом

користувач може як отримувати нові дані (get next point) так і редагувати результати аналізу старих чи нових даних за допомогою кнопок зміни результатів аналізу:

- OK
- WARNING
- ALERT
- ERROR

Також користувач має два засоби завантажити модель:

- TRAIN MODEL – тренувати виходячи з наявних даних, використовуючи найновіші завантажені в оперативну пам’ять дані
- LOAD MODEL – отримати вже натреновану модель яка була раніше збережена на жорсткий диск

Окрім того користувач може зберегти дані (SAVE DATA) переписавши дані що є наявними в раніше завантаженому джерелі даних тими що були розширені чи відредаговані користувачем.

Також користувач може навігувати даними за допомогою стрілок навігації, переходячи на попередні чи наступні точки даних, чи взагалі стрибаючи на першу чи останню.

Після завантаження чи тренування моделі (рис 4.3) стають активними всі елементи інтерфейсу:

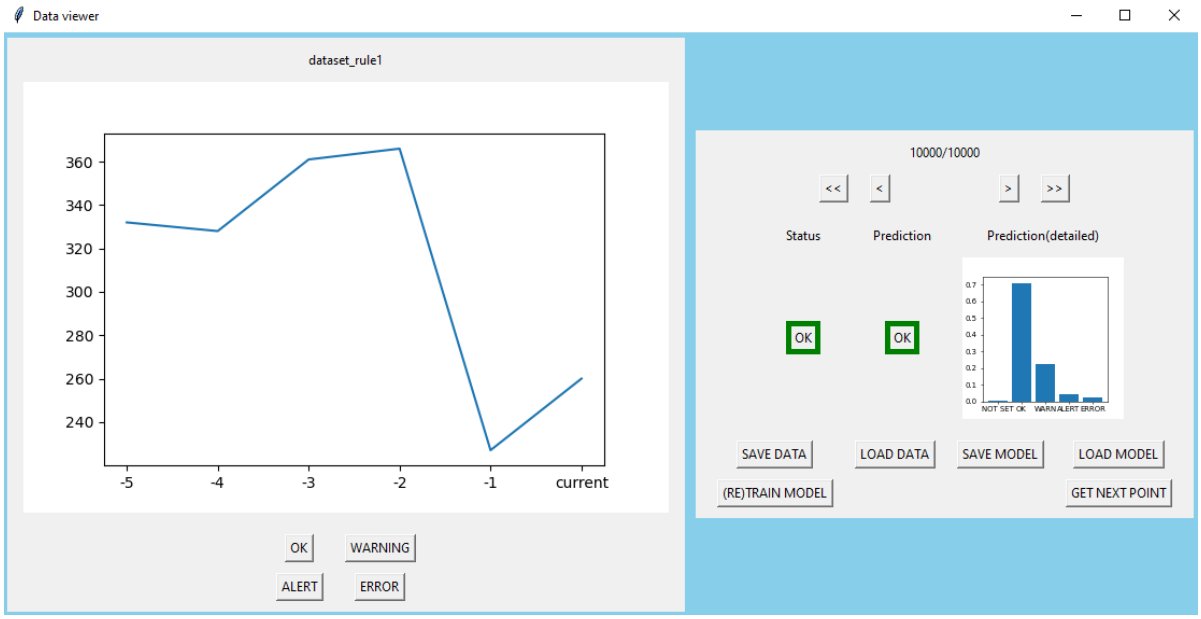


Рисунок 4.3 – Інтерфейс користувача з підключеною моделлю

З’являється ймовірнісна оцінка різних результатів аналізу (деталізоване передбачення) та найймовірніший з них (передбачення) згідно моделі. Таким чином отримуючи нові дані, користувач має попередній результат аналізу який може підтвердити чи спростувати. Кнопка TRAIN MODEL залишається активною для того щоб користувач міг оновити модель у випадку якщо внаслідок її недосконалості (напр. через малу кількість даних чи зміну правил)

серед останніх результатів аналізу багато незбіжностей статусу запропонованого системою та виставленого людиною.

На рис. 4.4 можна побачити вигляд програми яка отримала нову точку даних і зробила для неї передбачення, готової до отримання висновку користувача.

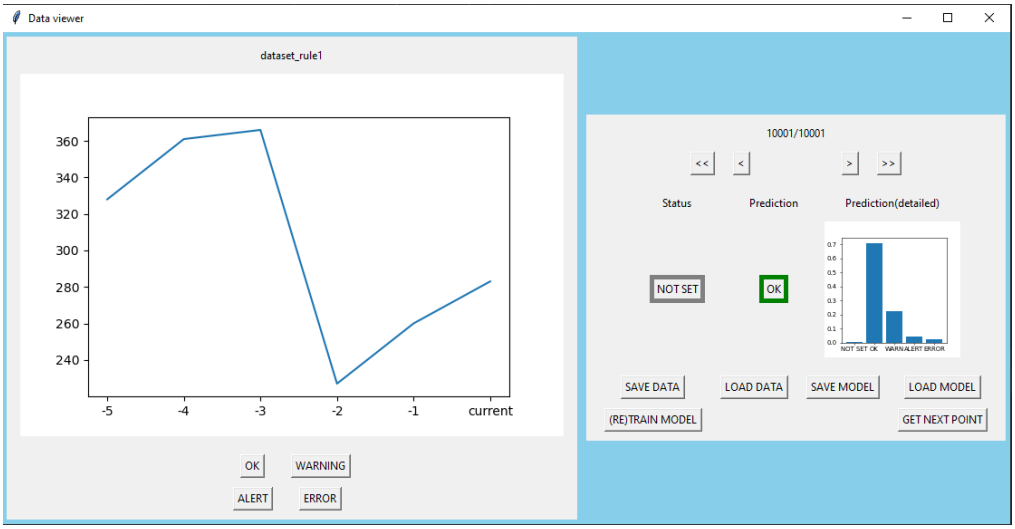


Рисунок 4.4 – Інтерфейс користувача з не проаналізованою точкою даних

У тому випадку коли користувач намагається здійснити не можливу в поточному стані дію (наприклад тренувати модель за відсутності завантажених даних чи навігувати на позицію поза індексом масиву доступних даних) – програма видає вспливаюче вікно з попередженням. Як приклад – рис.4.5

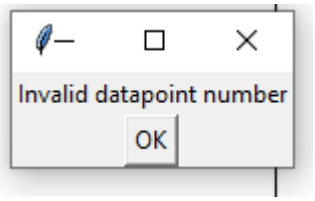


Рисунок 4.5 – Вспливаюче вікно з описом помилки

4.5 Рекомендації щодо розвитку та вдосконалення додатка

У для покращення інтеграції додатку можна внести кілька змін:

- Адаптувати завантаження, збереження і отримання даних для роботи з системами CI
- Вдосконалити інтерфейс, щоб упростити навігацію у великих датасетах (навігація по набору даних за допомогою введення числа, скроллбару чи кнопки «перейти до наступного необробленого результату виміру»
- Додати більш детальний індикатор процесу тренування моделі, так як тренування може зайняти багато часу, особливо на малопотужних машинах
- Додати можливість паралельного спостереження кількох потоків даних з окремими моделями
- Після впровадження в реальну систему продовжити оптимізацію архітектури нейромережі для більш швидкого та ефективного її навчання
- Комбінувати результати машинного навчання з жорстко заданими правилами для простих випадків які можуть бути легко сформульовані та запрограмовані (використання виключно машинного навчання було зроблено в цілях тестування даного подходу)
- Додати можливість (перемикач) для автоматичного заповнення статусу передбаченням

ВИСНОВОК ДО РОЗДІЛУ 4

У четвертому розділі представлений готовий застосунок. Був показаний процес відладки моделі навчання, генерації макетів даних, а також проаналізована ефективність моделі як функції складності правил та розміру тренувального датасету.

Був оглянутий користувацький інтерфейс та основні юзкейси які можуть відбутися під час використання ПЗ, розглянуті деталі інтерфейсу та їх взаємодії. Дане ПЗ не має переходу між екранами і всі дії відбуваються в межах однієї сторінки, різні стани відрізняються тим які команди кативні в даний момент.

Обраний формат взаємодії з користувачем дозволяє застосунку як бути інструментом для ручного аналізу, так і використовувати нейромережу як радника чи навіть заміну людині якщо вона досягне достатній точності.

Також були відокремлені недоліки поточної імплементації та методи їх усунення і перспективи вбудови в систему СІ у майбутньому. Нажаль взаємодія з реальною системою є поза межами цілей даної роботи але впровадження її в реальне виробництво може дещо підвищити якість аналізу даних у деяких системах.

ВИСНОВКИ

Під час виконання дипломної роботи було створено систему аналізу даних та пошуку підозрілих та критичних точок з використанням машинного навчання. Дана система надає можливість за допомогою настільного застосунку передавати людську експертизу в аналізі даних моделі машинного навчання, не перериваючи звичайний процес спостереження.

Таким чином система, будучи вбудованою в процес безперервної інтеграції, може поступово переходити від ролі радника до повної чи майже повної заміни людини-спостерігача-оператора.

У першому розділі були розглянуті задачі моніторінгу даних та засоби їх вирішення. Були розглянуті переваги і недоліки різних категорій систем моониторінгу та загальних підходів а також проведений огляд систем машинного навчання.

У другому розділі була детально розглянута задача та технології що будуть використані для реализації її вирішення. Були обрані та обгрунтовані для використання такі прості і надійні інструменти як мова програмування Python, бібліотека машинного навчання TensorFlow, а також засоби графічного відображення та архітектура нейромережі глибокого навчання що найбільш підходить для структури даних яку було обрано.

У третьому розділі було реалізовано генерацію макетів даних як заміни сстеми в яку може інтегруватися даний застосунок, модель машинного навчання і застосунок що дозволяє взаємодію з користувачем. Були наведені діаграми класів та показані значення що використовувалися для генерації макетних даних. Були продемонстровані фрагменти коду у вигляді скріншотів з середи розробки.

Також була обгрунтована необхідність використання макетних даних та представлені засоби їх генерації.

У четвертому розділі ми провели аналіз розробленої системи. У розділі була проаналізована відносна складність згенерованих макетних даних, ефективність нейромережі в визначенні правил та показан процесм і результати калібровки нейромережі.

Також був продемонстрований користукацьеий інтерфейс і зроблені висновки щодо можливостей подальшого розвитку ПЗ та інтеграції його в реальні системи.

Розроблена система є кросплатформеною та легко розгортаємою, відповідає заданим на початку вимогам і є аналогом систем що реально використовуються у сфері тестування продуктивності ПЗ (але відрізняється від них використанням глибокого навчання).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Юридична енциклопедія - Шемшученко Ю.С. [Електронний ресурс] //– Режим доступу до ресурсу: https://leksika.com.ua/legal/
2. Технопедія [Електронний ресурс] Режим доступу до ресурсу: https://www.techopedia.com/definition/
3. Bifet, Albert; Gavaldà, Ricard; Holmes, Geoff; Pfahringer, Bernhard (2018). Machine Learning for Data Streams with Practical Examples in MOA. Adaptive Computation and Machine Learning. // Машинне навчання для потоків даних із практичними прикладами в MOA. Адаптивні обчислення та машинне навчання MIT Press. с. 288. ISBN 9780262037792.
4. Guha, S.; Mishra, N.; Motwani, R.; O'Callaghan, L. (2000). "Clustering Data Streams" // “Кластерізація потоків даних”. Proceedings of the Annual Symposium on Foundations of Computer Science: 359–366.
5. https://matplotlib.org/ [Електронний ресурс] // – Режим доступу до ресурсу: https://matplotlib.org
6. boost.org [Електронний ресурс] Бібліотеки мови C++ // – Режим доступу до ресурсу: https://www.boost.org/doc/libs/1_73_0/libs/graph/doc/index.html
7. https://www.zabbix.com/ [Електронний ресурс] // – Режим доступу до ресурсу: https://www.zabbix.com/
8. https://www.nagios.org/ / [Електронний ресурс] // – Режим доступу до ресурсу: https://www.nagios.org/
9. https://wiki.jenkins-ci.org/ [Електронний ресурс] // – Режим доступу до ресурсу: https://wiki.jenkins-ci.org/
10. https://www.tensorflow.org/ [Електронний ресурс] // – Режим доступу до ресурсу: https://www.tensorflow.org/

11. https://www.v7labs.com/blog/neural-network-architectures-guide [Електронний ресурс] // – Режим доступу до ресурсу: https://www.v7labs.com/blog/neural-network-architectures-guide

12. K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” Neural networks, vol. 2, no. 5, pp. 359–366, 1989.

13. H. Mhaskar, Q. Liao, and T. Poggio, “When and why are deep networks better than shallow ones?,” in Proceedings of the AAAI conference on artificial intelligence, 2017, vol. 31, no. 1.

ДОДАТОК 1

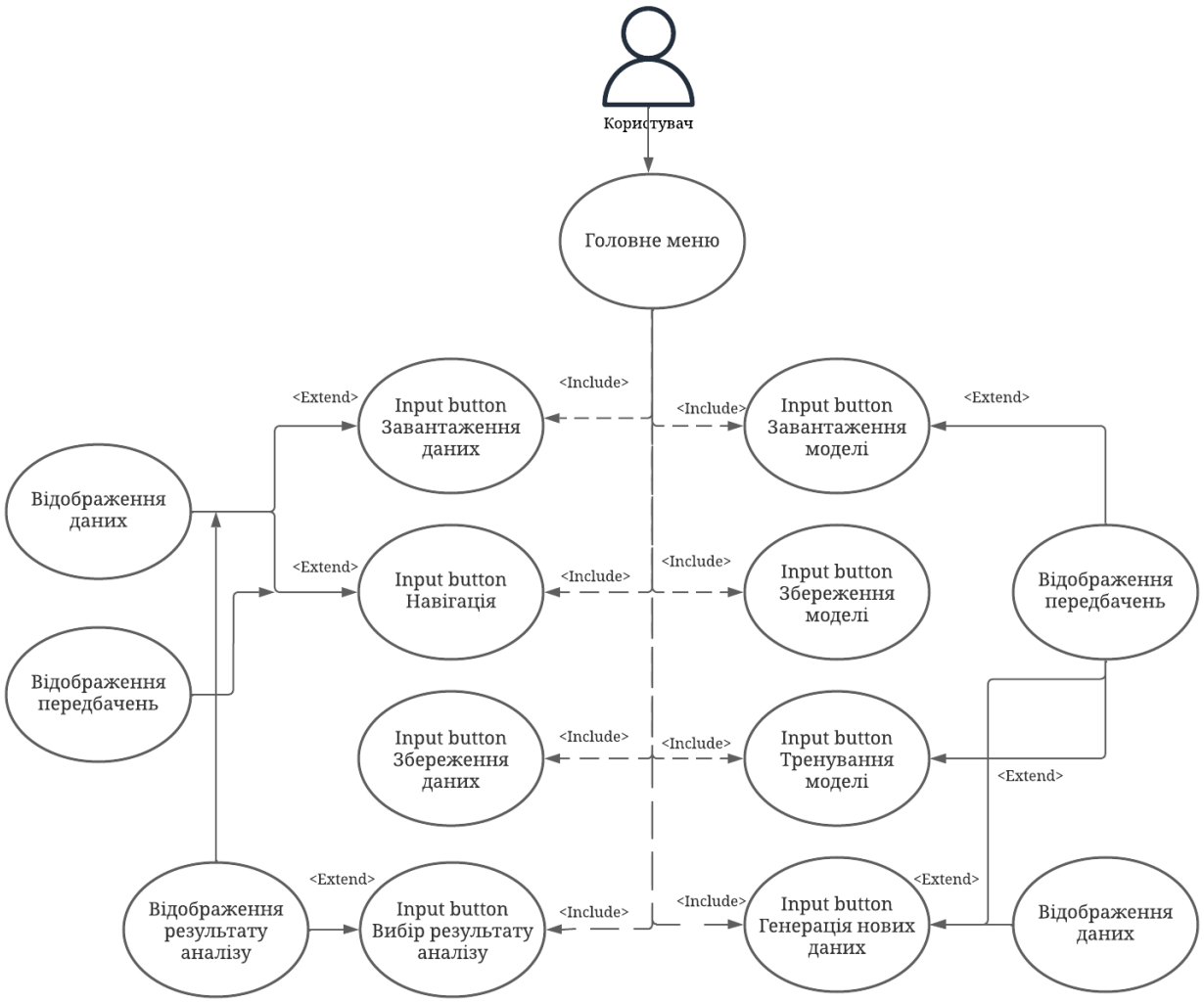
Система моніторингу даних та виявлення підозрілих чи критичних точок

Структура системи (структурна схема)

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2023 р



					ІАЛЦ.467200.004 Д1		
		№ докум.	Підпис	Дата			
Розробив		Семенов А.І.			**Система моніторингу даних та виявлення підозрілих чи критичних точок**	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Пустовіт О.М..			Структура системи (структурна схема)		КПІ ім. Ігоря Сікорського, ФІОТ, ІО-з91
Н. Контр.		Виноградов Ю.М.					
Затвердив							

ДОДАТОК 2

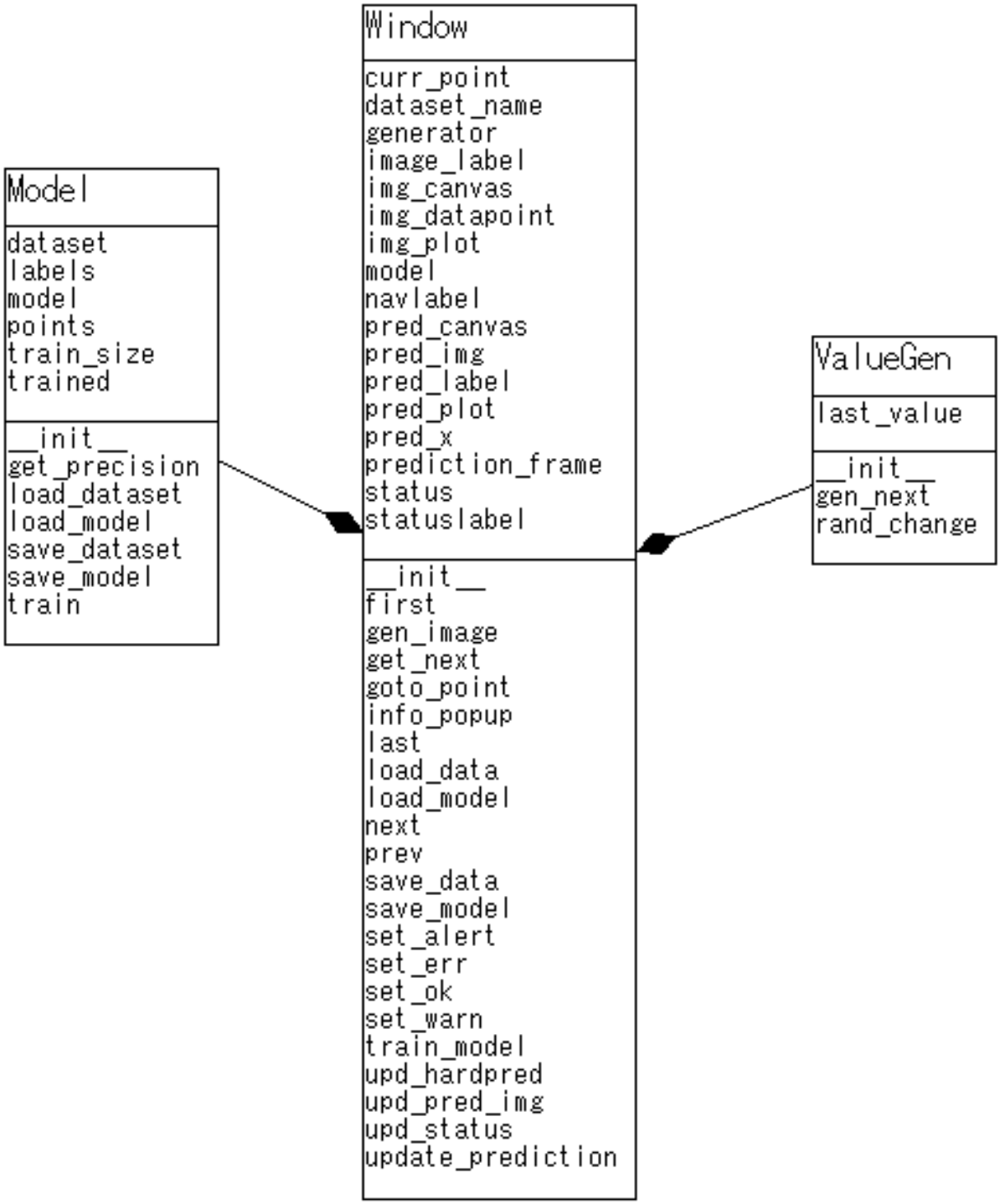
Система моніторингу даних та виявлення підозрілих чи критичних точок

Діаграма класів (функціональна схема)

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2023 р



					ІАЛЦ.467200.005 Д2	
		№ докум.	Підпис	Дата		
Розробив		Семенов А.І.			**Система моніторингу даних та виявлення підозрілих чи критичних точок** Діаграма класів (функціональна схема)	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Пустовіт О.М...				
Н. Контр.		Виноградов Ю.М.				КПІ ім. Ігоря Сікорського, ФІОТ, ІО-з91
Затвердив						

ДОДАТОК 3

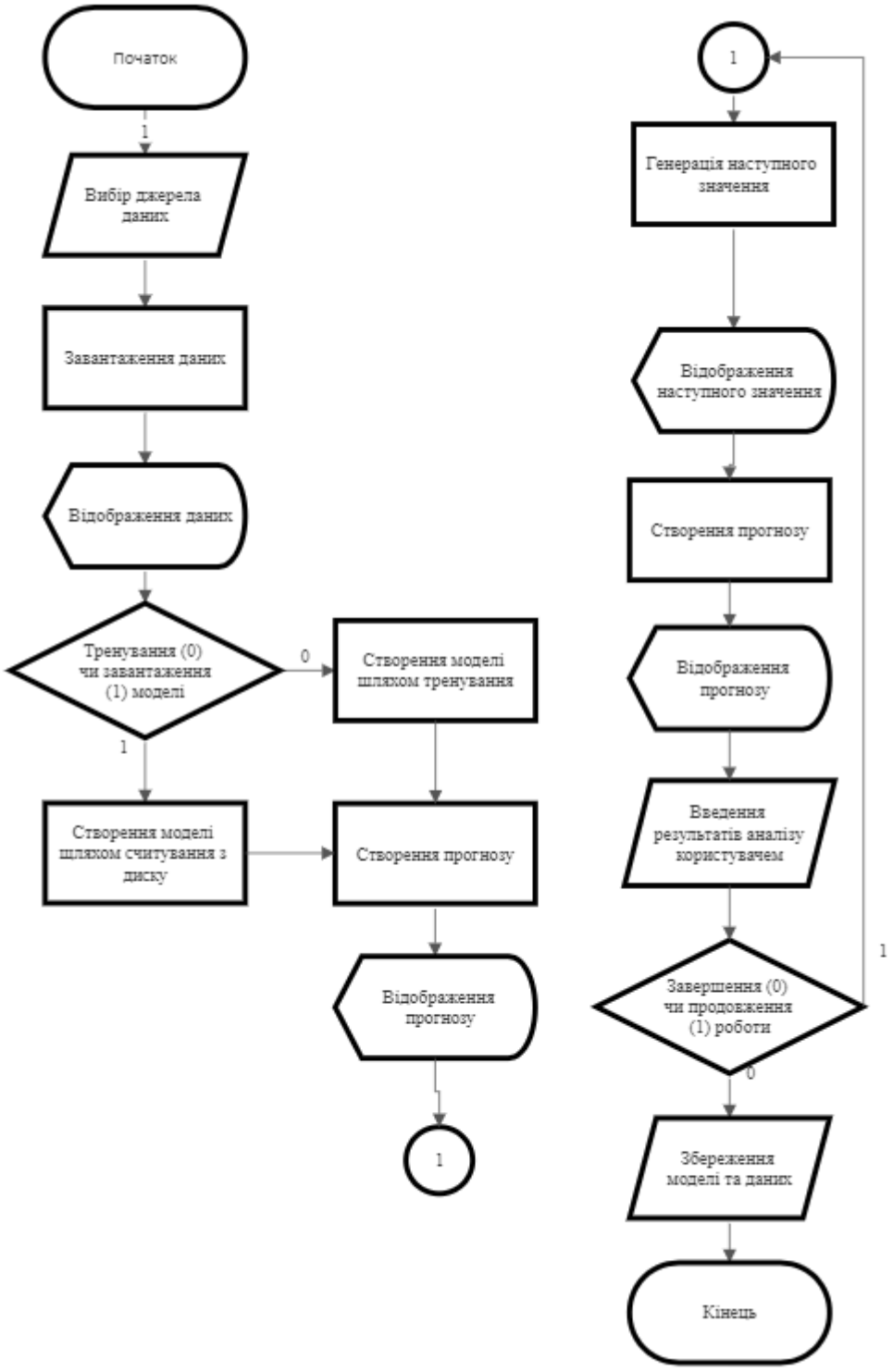
Система моніторингу даних та виявлення підозрілих чи критичних точок

Алгоритм дій програмного забезпечення (принципова схема)

ІАЛЦ.467200.006 Д3

Аркушів 1

Київ 2023 р



					ІАЛЦ.467200.006 Д3		
		№ докум.	Підпис	Дата			
Розробив		Семенов А.І			**Система моніторингу даних та виявлення підозрілих чи критичних точок** Алгоритм дій програмного забезпечення (принципова схема)	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Пустовіт О.М.					
						КПІ ім. Ігоря Сікорського, ФІОТ, ІО-з91	
Н. Контр.		Виноградов Ю.М.					
Затвердив							

ДОДАТОК 4

Система моніторингу даних та виявлення підозрілих чи критичних точок

Текст програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 25

Київ 2023 р

```python
from dataclasses import dataclass
from enum import Enum

"""Обмеження очікуваного діапазону значень"""
LOWER_LIMIT = 1
UPPER_LIMIT = 1000

MIN_POSSIBLE = LOWER_LIMIT
MAX_POSSIBLE = 10000

USUAL_CHANGE = 0.15
UNUSUAL_CHANGE = 1
RARE_CHANGE = 100


"""
Значення точки даних складається з власно значення і висновку аналізу цього значення
NOT_SET = не визначено
OK = значення в межах норми і не потребує уваги
WARNING = значення в межах норми і потребує уваги
ALERT = значення не в межах норми і не є помилкою виміру
ERROR = значення не в межах норми і є помилкою виміру
"""


class Resolution(Enum):
    NOT_SET = 0
    OK = 1
    WARNING = 2
    ALERT = 3
    ERROR = 4


STATUS_COLORS = {
    0: 'gray',
    1: 'green',
    2: 'yellow',
    3: 'red',
    4: 'black'
}

STATUS_TEXT = {
    0: 'NOT SET',
    1: 'OK',
    2: 'WARN',
    3: 'ALERT',
    4: 'ERROR'
}

@dataclass
class DataPoint:
    value: []
    resolution: Resolution = Resolution.NOT_SET
```

```
from random import randint, random
from data_lib import *
import pickle


DATA_RANGE = 10000


class ValueGen:
    """Клас що генерує випадкові числа за заданими правилами"""
    def __init__(self, start_point):
        self.last_value = start_point

    def gen_next(self):
        new_value = self.rand_change(self.last_value)
        self.last_value = new_value
        return new_value

    @staticmethod
    def rand_change(prev_value: int) -> int:
        """Випадковим чином змінити значення числа
        на 0%-10000%: більш ймовірні незначні зміни"""
        if prev_value < LOWER_LIMIT or prev_value > UPPER_LIMIT:
            if random() > 0.25:  # 75% шанс повернення до норми якщо вийшло за межі
                return randint(LOWER_LIMIT, UPPER_LIMIT)
        change_rate = random()
        if change_rate < 0.85:
            change = random()*USUAL_CHANGE*prev_value
        elif 0.85 <= change_rate < 0.98:
            change = random()*UNUSUAL_CHANGE*prev_value
        else:
            change = random() * RARE_CHANGE * prev_value
        change_dir = random()
        if change_dir > 0.4:
            value = int(prev_value + change)
        else:
            value = int(prev_value - change)
        if value < 0:
            return 0
        elif value > MAX_POSSIBLE:
            return MAX_POSSIBLE
        else:
            return value
```

```
if __name__ == "__main__":
    """Генеруємо масив з DATA_RANGE значень даних для обробки"""
    history = [randint(LOWER_LIMIT, UPPER_LIMIT)]
    generator = ValueGen(history[0])
    for _ in range(4):
        history.append(generator.gen_next())
    data_generated = []
    for _ in range(DATA_RANGE):
        history.append(generator.gen_next())
        data_generated.append(DataPoint(history))
        history = history[1:]
    with open('raw_data', 'wb') as outfile:  # 'wb' with pickle, 'w' for text format
        # outfile.write(str(data_generated))
        pickle.dump(data_generated, outfile)
```

```
import pickle
import numpy
import time
from data_lib import LOWER_LIMIT, UPPER_LIMIT, Resolution, MIN_POSSIBLE, MAX_POSSIBLE
# TensorFlow and tf.keras
import tensorflow as tf
import matplotlib

matplotlib.use('TkAgg')


# Значення не більше 1000
def within_bounds(datapoint) -> Resolution:
    if LOWER_LIMIT <= datapoint[5] <= UPPER_LIMIT:
        return Resolution.OK
    else:
        return Resolution.ALERT


# Поточне і попередні значення кратні 32
def not_rounded(datapoint) -> Resolution:
    if datapoint[5] % 32 == 0 and datapoint[4] % 32 == 0:
        return Resolution.ALERT
    else:
        return Resolution.OK


# Вважати значення невалідним якщо воно менше о або більше 100000
#
def is_valid(datapoint) -> Resolution:
    if MIN_POSSIBLE <= datapoint[5] <= MAX_POSSIBLE:
        return Resolution.OK
    else:
        return Resolution.ERROR


# Відхилення від середнього проміж минулих 5 значень не більше ніж 20%
#
def is_stable(datapoint) -> Resolution:
    avg = sum(datapoint[:5])/5
    if abs(datapoint[5] - avg) < avg*0.2:
        return Resolution.OK
    else:
```

```
        return Resolution.WARNING


# Остання зміна значення більш ніж вдвічі більша за будь-яку з 4 попередніх
#
def not_accelerating(datapoint) -> Resolution:
    maxdiff = 0
    for i in range(1, 5):
        maxdiff = max(abs(datapoint[i] - datapoint[i-1]), maxdiff)
    if abs(datapoint[5] - datapoint[4]) <= maxdiff * 2:
        return Resolution.OK
    else:
        return Resolution.WARNING


# Відсутність змін також вважати підозріолю
#
def not_stuck(datapoint) -> Resolution:
    maxdiff = 0
    for i in range(1, 5):
        maxdiff = max(abs(datapoint[i] - datapoint[i-1]), maxdiff)
    if datapoint[5] == datapoint[4]:
        return Resolution.WARNING
    else:
        return Resolution.OK


# Зростає протягом 3 останніх вимірів
#
def not_growing(datapoint) -> Resolution:
    if datapoint[5] > datapoint[4] > datapoint[3]:
        return Resolution.WARNING
    else:
        return Resolution.OK


ruleset0 = [
    is_valid,
    within_bounds
]

ruleset1 = [
    is_valid,
```

```
    within_bounds,
    not_growing,
]

ruleset2 = [
    is_valid,
    within_bounds,
    is_stable,
    not_stuck,
]

ruleset3 = [
    is_valid,
    within_bounds,
    is_stable,
    not_growing,
    not_accelerating,
    not_stuck,
    not_rounded,
]


def check_for_rules(ruleset_used, datapoint) -> Resolution:
    resolution = Resolution.NOT_SET
    for rule in ruleset_used:
        curr_resolution = rule(datapoint)
        if curr_resolution == Resolution.ERROR:
            return curr_resolution
        elif curr_resolution == Resolution.ALERT:
            resolution = curr_resolution
        elif curr_resolution == Resolution.WARNING and resolution == Resolution.OK:
            resolution = curr_resolution
        elif curr_resolution == Resolution.OK and resolution not in (Resolution.WARNING,
Resolution.ALERT):
            resolution = curr_resolution
    return resolution


def mark_data_with_ruleset(raw_data, ruleset_used):
    for datapoint in raw_data:
        datapoint.resolution = check_for_rules(ruleset_used, datapoint.value)
    return raw_data
```

```
def create_dataset(dataset_name, raw_data):
    train_arr = []
    train_labels = []
    for point in raw_data:
        train_arr.append(point.value)
        train_labels.append(point.resolution.value)
    train_arr = numpy.array(train_arr)
    train_labels = numpy.array(train_labels)
    train_dataset = tf.data.Dataset.from_tensor_slices(
        {"datapoint": train_arr, "label": train_labels},
        name=f"{dataset_name}"
    )
    return train_dataset


if __name__ == "__main__":
    rulesets = [ruleset0, ruleset1, ruleset2, ruleset3]
    complexities = []
    for rn, ruleset in enumerate(rulesets):
        with open('raw_data', 'rb') as datafile:
            unmarked_data = pickle.load(datafile)
        data = mark_data_with_ruleset(unmarked_data, ruleset)
        setname = f'dataset_rule{rn}'
        train = create_dataset(setname, data)
        tf.data.Dataset.save(train, f'datasets/{setname}')

    with open('raw_data', 'rb') as datafile:
        unmarked_data = pickle.load(datafile)
        compl_average = [0, 0, 0, 0]
        for i, ruleset in enumerate(rulesets):
            time_passed = 0

            start = time.time()
            for j in range(5000):
                check_for_rules(ruleset, unmarked_data[j].value)
            end = time.time()
            time_passed += end - start
            compl_average[i] = time_passed/5000

        print(compl_average)
```

```
import tensorflow as tf
import numpy as np


class Model:
    """Клас що зберігає набор даних і модель та проводить операції з ними"""
    dataset: tf.data.Dataset
    train_size: int = 5000
    model: tf.keras.models.Sequential
    trained: bool = False

    def __init__(self):
        self.points = []
        self.labels = []

    def load_dataset(self, dataset_path):
        self.dataset = tf.data.Dataset.load(dataset_path)
        num_data = self.dataset.as_numpy_iterator()

        points = []
        labels = []
        for elem in num_data:
            points.append(elem['datapoint'])
            labels.append(elem['label'])
        self.points = np.array(points)
        self.labels = np.array(labels)

    def save_dataset(self, dataset_path, dataset_name=None):
        if not dataset_name:
            dataset_name = dataset_path.split('/')[0]
        self.dataset = tf.data.Dataset.from_tensor_slices(
            {"datapoint": self.points, "label": self.labels},
            name=f"{dataset_name}"
        )
        tf.data.Dataset.save(self.dataset, f'datasets/{dataset_name}')

    def train(self, epochs=500, batch_size=50, verbose=0):
        if not self.dataset:
            raise RuntimeError("No dataset loaded")
        normalizer = tf.keras.layers.Normalization(input_shape=[6, ], axis=-1)
        normalizer.adapt(self.points)
        self.model = tf.keras.models.Sequential([
            tf.keras.layers.Dense(64, input_shape=[6, ], activation='relu'),
```

```
            tf.keras.layers.Dense(64, activation='relu'),
            tf.keras.layers.Dense(64, activation='relu'),
            tf.keras.layers.Dense(5),
            tf.keras.layers.Softmax()
        ])
        self.model.compile(optimizer=tf.keras.optimizers.Adam(),
                           loss=tf.keras.losses.SparseCategoricalCrossentropy(),
                           metrics=['accuracy'])

        train_points = self.points[-self.train_size:]
        train_labels = self.labels[-self.train_size:]
        self.model.fit(train_points,  train_labels,  epochs=epochs,  batch_size=batch_size,
verbose=verbose)
        self.trained = True

    def get_precision(self):
        _, b = self.model.evaluate(self.points, self.labels)
        print(f"Accuracy: {b}")
        return b

    def save_model(self, save_path):
        self.model.save(save_path)

    def load_model(self, load_path):
        self.model = tf.keras.models.load_model(load_path)
```

```
from model import Model


if __name__ == "__main__":
    """Тестування точності передбачень для """
    train_set_size = [500, 1000, 5000, 10000]

    test_results = []
    for runesetnum in range(4):
        model = Model()
        model.load_dataset(dataset_path=f'datasets/dataset_rule{runesetnum}')
        #  for size in train_set_size:
        for size in train_set_size:
            model.train_size = size
            model.train(verbose=1)
            precision = model.get_precision()
            test_results.append({
                'epochs': 500,
                'batch': 50,
                'dataset': runesetnum,
                'train_size': size,
                'precision': precision
            })
    for result in test_results:
        print(result)
```

```
from tkinter import Tk, Button, Label, Frame, filedialog, Toplevel
from data_lib import STATUS_COLORS, STATUS_TEXT
from tensorflow.python.framework.errors_impl import NotFoundError
import numpy as np
from generate_model import Model
from gen_values import ValueGen
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

PLACEHOLDER = [1, 5, 6, 17, 8, 24]
DATAPOINT_X = [-5, -4, -3, -2, -1, 'current']


class Window(Tk):
    curr_point: int
    model: Model
    dataset_name: str
    generator: ValueGen

    def __init__(self):
        super().__init__()
        self.title("Data viewer")
        self.config(background="skyblue")

        self.model = Model()

        left_side = Frame(self)
        left_side.grid(row=0, column=0, padx=5, pady=5)
        right_side = Frame(self)
        right_side.grid(row=0, column=1, padx=5, pady=5)

        image_frame = Frame(left_side, width=400, height=400)
        image_frame.grid(row=0, column=0, padx=5, pady=5)
        self.image_label = Label(image_frame, text="Placeholder data")
        self.image_label.grid(row=0, column=0, padx=5, pady=5)
        # load image to be "edited"
        self.img_datapoint = Figure(figsize=(6, 4), dpi=100)
        self.img_plot = self.img_datapoint.add_subplot(111)
        self.img_plot.plot(DATAPOINT_X, PLACEHOLDER)
        self.img_canvas = FigureCanvasTkAgg(self.img_datapoint, master=self.image_frame)
        self.img_canvas.draw()
        self.img_canvas.get_tk_widget().grid(row=2, column=0, padx=10, pady=5)
```

```
        resolution_frame = Frame(left_side, width=400, height=300)
        resolution_frame.grid(row=1, column=0, padx=10, pady=5)
        okbutton = Button(resolution_frame, text="OK", command=self.set_ok)
        okbutton.grid(row=0, column=0, padx=10, pady=5)
        warnbutton = Button(resolution_frame, text="WARNING", command=self.set_warn)
        warnbutton.grid(row=0, column=1, padx=10, pady=5)
        alertbutton = Button(resolution_frame, text="ALERT", command=self.set_alert)
        alertbutton.grid(row=1, column=0, padx=10, pady=5)
        errorbutton = Button(resolution_frame, text="ERROR", command=self.set_err)
        errorbutton.grid(row=1, column=1, padx=10, pady=5)

        nav_frame = Frame(right_side, width=400, height=100)
        nav_frame.grid(row=0, padx=10, pady=5, sticky='N')
        firstbutton = Button(nav_frame, text="<<", command=self.first)
        firstbutton.grid(row=1, column=0, padx=10, pady=5)
        prevbutton = Button(nav_frame, text="<", command=self.prev)
        prevbutton.grid(row=1, column=1, padx=10, pady=5)
        self.navlabel = Label(nav_frame, text="n/a")
        self.navlabel.grid(row=0, column=2, padx=5, pady=5)
        nextbutton = Button(nav_frame, text=">", command=self.next)
        nextbutton.grid(row=1, column=3, padx=10, pady=5)
        lastbutton = Button(nav_frame, text=">>", command=self.last)
        lastbutton.grid(row=1, column=4, padx=10, pady=5)

        status_frame = Frame(right_side, width=400, height=250)
        status_frame.grid(row=1, padx=10, pady=5)
        Label(status_frame, text="Status").grid(row=0, column=0, padx=5, pady=5)
        self.status = Frame(status_frame, width=100, height=100)
        self.status.grid(row=1, column=0, padx=30, pady=30)
        self.statuslabel = Label(self.status)
        self.statuslabel.grid(row=0, column=0, padx=5, pady=5)
        Label(status_frame, text="Prediction").grid(row=0, column=1, padx=5, pady=5)
        self.prediction_frame = Frame(status_frame, width=200, height=200)
        self.prediction_frame.grid(row=1, column=1, padx=30, pady=30)
        self.pred_label = Label(self.prediction_frame)
        self.pred_label.grid(row=0, column=0, padx=5, pady=5)
        Label(status_frame,   text="Prediction(detailed)").grid(row=0,   column=2,   padx=5,
pady=5)
        self.pred_img = Figure(figsize=(3, 3), dpi=50)
        self.pred_plot = self.pred_img.add_subplot(111)
        self.pred_x = [STATUS_TEXT[t] for t in range(5)]
        self.pred_canvas = FigureCanvasTkAgg(self.pred_img, master=status_frame)
        self.pred_canvas.get_tk_widget().grid(row=1, column=2, padx=10, pady=5)
```

```
        fileop_frame = Frame(right_side, width=400, height=100)
        fileop_frame.grid(row=2, padx=10, pady=5)
        load_data_button = Button(fileop_frame, text="LOAD DATA", command=self.load_data)
        load_data_button.grid(row=0, column=1, padx=10, pady=5)
        save_data_button = Button(fileop_frame, text="SAVE DATA", command=self.save_data)
        save_data_button.grid(row=0, column=0, padx=10, pady=5)
        load_model_button = Button(fileop_frame, text="LOAD MODEL", command=self.load_model)
        load_model_button.grid(row=0, column=3, padx=10, pady=5)
        save_model_button = Button(fileop_frame, text="SAVE MODEL", command=self.save_model)
        save_model_button.grid(row=0, column=2, padx=10, pady=5)
        train_model_button      =      Button(fileop_frame,      text="(RE)TRAIN      MODEL",
command=self.train_model)
        train_model_button.grid(row=1, column=0, padx=10, pady=5)
        train_model_button     =     Button(fileop_frame,     text="GET     NEXT     POINT",
command=self.get_next)
        train_model_button.grid(row=1, column=3, padx=10, pady=5)

    def save_data(self):
        self.model.dataset.save(f'datasets/{self.dataset_name}')

    def load_data(self):
        folder_selected = filedialog.askdirectory()
        self.dataset_name = folder_selected.split('/')[-1]
        if not folder_selected:
            return
        try:
            self.model.load_dataset(folder_selected)
        except NotFoundError:
            self.info_popup(title='Error',  text=f'{folder_selected}  doesnt  contain  valid
dataset')
        self.image_label.config(text=f'{self.dataset_name}')
        self.last()

    def save_model(self):
        self.model.save_model(f'models/model_{self.dataset_name}')

    def load_model(self):
        self.model.load_model(f'models/model_{self.dataset_name}')

    def set_alert(self):
        self.model.labels[self.curr_point] = 3
        self.upd_status(3)
```

```
def set_warn(self):
    self.model.labels[self.curr_point] = 2
    self.upd_status(2)

def set_ok(self):
    self.model.labels[self.curr_point] = 1
    self.upd_status(1)

def set_err(self):
    self.model.labels[self.curr_point] = 4
    self.upd_status(4)

def prev(self):
    if self.goto_point(self.curr_point-1):
        self.curr_point = self.curr_point-1

def next(self):
    if self.goto_point(self.curr_point+1):
        self.curr_point = self.curr_point+1

def first(self):
    if self.goto_point(0):
        self.curr_point = 0

def last(self):
    if self.goto_point(len(self.model.points) - 1):
        self.curr_point = len(self.model.points) - 1

def gen_image(self, datapoint: list):
    self.img_datapoint.clf()
    self.img_plot = self.img_datapoint.add_subplot(111)
    self.img_plot.plot(DATAPOINT_X, datapoint)
    self.img_canvas.draw()

def upd_pred_img(self, pred_array):
    self.pred_img.clf()
    self.pred_plot = self.pred_img.add_subplot(111)
    self.pred_plot.bar(self.pred_x, pred_array)
    self.pred_canvas.draw()

def upd_status(self, status: int):
    self.status.config(bg=STATUS_COLORS[status])
```

```
        self.statuslabel.config(text=STATUS_TEXT[status])

    def upd_hardpred(self, status):
        self.prediction_frame.config(bg=STATUS_COLORS[status])
        self.pred_label.config(text=STATUS_TEXT[status])

    def goto_point(self, point_num: int):
        if point_num > len(self.model.points) - 1 or point_num < 0:
            self.info_popup(title='Error', text=f'Invalid datapoint number')
            return False
        # Оновити номер що відображається
        self.navlabel.config(text=f'{point_num+1}/{len(self.model.points)}')
        # Згенерувати нове зображення і статус
        self.gen_image(self.model.points[point_num])
        self.upd_status(self.model.labels[point_num])
        # Оновити передбачення якщо завантажена модель
        if hasattr(self.model, 'model'):
            self.update_prediction(point_num)
        return True

    def update_prediction(self, point_num: int):
        point = np.array([self.model.points[point_num]])
        predictions = self.model.model.predict(point)[0]
        hard_pred = np.argmax(predictions)
        self.upd_hardpred(hard_pred)
        self.upd_pred_img(predictions)

    def train_model(self):
        if not hasattr(self.model, 'dataset'):
            self.info_popup(title='Error', text=f'No data loaded')
        self.model.train_size = min(len(self.model.points), self.model.train_size)
        self.model.train()
        self.update_prediction(self.curr_point)

    @staticmethod
    def info_popup(title: str, text: str):
        win = Toplevel()
        win.wm_title(title)

        lab = Label(win, text=text)
        lab.grid(row=0, column=0)

        b = Button(win, text="OK", command=win.destroy)
```

```
        b.grid(row=1, column=0)

    def get_next(self):
        if not hasattr(self, 'generator'):
            self.generator = ValueGen(self.model.points[-1][-1])
        new_value = self.generator.gen_next()
        new_datapoint = list(self.model.points[-1][1:])
        new_datapoint.append(new_value)
        self.model.points = np.append(self.model.points, [new_datapoint], axis=0)
        self.model.labels = np.append(self.model.labels, [0])
        self.last()


if __name__ == "__main__":
    # Start the event loop.
    window = Window()
    window.mainloop()
```