НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

«До захисту допущено»

Завідувач кафедри

_____ Олег ЧЕРТОВ

«___» _____ 2021 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Наука про дані та математичне моделювання»
спеціальності 113 «Прикладна математика»

на тему: «Математичне та програмне забезпечення системи ідентифікації аномалій в даних вимірювання»

Виконав:
студент IV курсу, групи КМ-71
Бірук Станіслав Володимирович _____

Керівник:
Старший викладач,
Ладогубець Тетяна Сергіївна _____

Консультант з нормоконтролю:
Старший викладач,
Мальчиков Володимир Вікторович _____

Рецензент:
Доцент кафедри ПЗКС, канд. техн. наук, доцент
Онай Микола Володимирович _____

Засвідчую, що в цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.
Бірук С.В. _____

Київ — 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 113 «Прикладна математика»

Освітньо-професійна програма «Наука про дані та математичне моделювання»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олег ЧЕРТОВ

«___» _____ 2021 р.

ЗАВДАННЯ

на дипломну роботу студенту

Біруку Станіславу Володимировичу

1. Тема роботи: «Математичне та програмне забезпечення системи ідентифікації аномалій в даних вимірювання», керівник роботи Ладогубець Тетяна Сергіївна, старший викладач, затверджені наказом по університету від «25» травня 2021 р. № 1331-С.

2. Термін подання студентом роботи: «9» червня 2021 р.

3. Вихідні дані до роботи: система повинна працювати з багатовимірними часовими рядами в реальному часі, мінімальна точність виявлення аномалій 75%

4. Зміст роботи: виконати аналіз існуючих методів розв’язання задачі, вибрати метод ідентифікації аномалій в багатовимірних часових рядах, спроектувати та реалізувати систему, провести тестування на реальних даних.

5. Перелік ілюстративного матеріалу: архітектурні графи нейронних мереж, схема взаємодії модулів системи, знімки екранних форм

6. Дата видачі завдання: «01» лютого 2021 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за тематикою та збір даних	10.03.2021	
2	Проведення порівняльного аналізу математичних методів ідентифікації аномалій	31.03.2021	
3	Підготовка матеріалів першого розділу	10.04.2021	
4	Підготовка матеріалів другого розділу	20.04.2021	
5	Розробка програмного забезпечення	05.05.2021	
6	Тестування програмного забезпечення	10.05.2021	
7	Підготовка матеріалів третього розділу	15.05.2021	
8	Підготовка матеріалів четвертого розділу	20.05.2021	
9	Оформлення пояснювальної записки	01.06.2021	

Студент _____ Станіслав БІРУК

Керівник роботи _____ Тетяна ЛАДОГУБЕЦЬ

АНОТАЦІЯ

Дипломну роботу виконано на 53 аркушах, вона містить 2 додатки та перелік посилань на використані джерела з 34 найменувань. У роботі наведено 14 рисунків та 7 таблиць.

Метою даної дипломної роботи є підвищення ефективності ідентифікації аномалій в даних вимірювання.

У роботі проведено аналіз існуючих методів ідентифікації аномалій в часових рядах, зокрема, методи подібності, статистичні, зменшення розмірності, передбачувальні та генеративні моделі. Виконано їх порівняння з погляду особливостей застосування та вимог, які ставляться до розроблюваної системи. Для розв’язання задачі було вибрано компекс з двох методів: генеративну нейронну мережу варіаційного автоенкодера та SPOT (Streaming Peaks-over-Threshold) алгоритм. Розроблено систему та виконано тестування на реальних наборах даних.

Ключові слова: аномалія, багатовимірний часовий ряд, варіаційний автоенкодер, SPOT алгоритм.

ABSTRACT

The thesis is presented in 53 pages. It contains 2 appendixes and bibliography of 34 references. 14 figures and 7 tables are given in this paper.

The goal of this thesis is to increase the efficiency the identification of anomalies of measurement data.

The paper analyzes existing methods to identify anomalies in time series: methods of similarity, statistical, dimensional reduction, predictive and generative models. Comperison of that methods is made in terms of application features and requirements related to the developed system. In order to find solution of considered task generative neural network of the variation autoencoder and the SPOT (Streaming Peaks-over-Threshold) algorithm were used. The system is tested on real data sets.

Keywords: anomaly, multidimensional time series, variational autoencoder, SPOT algorithm

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ....8
ВСТУП....9
1 ПОСТАНОВКА ЗАДАЧІ....10
2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ВИЯВЛЕННЯ АНОМАЛІЙ....11
2.1 Часові ряди та їх властивості....11
2.2 Типи аномалій....12
2.3 Одновимірні часові ряди....13
2.3.1 Методи подібності....14
2.3.2 Статистичні методи....14
2.3.3 Передбачувальні моделі....15
2.4 Багатовимірні часові ряди....16
2.4.1 Одновимірні підходи....17
2.4.2 Багатовимірні підходи....18
2.5 Порівняльна характеристика....20
2.6 Висновки до розділу....22
3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ....24
3.1 Опис проблеми....24
3.2 Обробка вхідних даних....25
3.3 Варіаційні автоенкодери....25
3.4 Теорія екстремальних значень....29
3.4.1 Розподіл екстремальних значень....29
3.4.2 Peaks Over-Threshold (POT)....30
3.4.3 Оцінка параметрів розподілу....31
3.4.4 SPOT алгоритм....32

3.5 Висновки до розділу.....33
4 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.....34
4.1 Опис бібліотек.....35
4.2 Опис функціональних елементів.....35
4.2.1 Encoder.....37
4.2.2 Decoder.....38
4.2.3 SPOT.....39
4.4 Висновки до розділу.....43
5 ВИПРОБУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ.....44
5.1 Тестова вибірка.....44
5.2 Метрики якості.....44
5.3 Аналіз результатів.....45
5.4 Загальні результати.....47
5.5 Висновки до розділу.....48
ВИСНОВКИ.....49
ПЕРЕЛІК ПОСИЛАНЬ.....50
Додаток А Лістинги програм.....54
Додаток Б Ілюстративний матеріал.....62

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

GAN (Generative Adversarial Network) – генеративно-змагальна мережа

GRU (Gated Recurrent Units) – мережа вентильних рекурентних вузлів

LSTM (Long Short-Term Memory) – мережа довгої короткочасної пам’яті

POT(Peaks Over-Threshold) – піки, що перевищують критичне значення

RNN (Recurrent Neural Network) – рекурентна нейронна мережа

SPOT (Sreaming Peaks Over-Threshold) – потокові піки, що перевищують критичне значення

VAE (Variational Autoencoder) – мережа варіаційного автоенкодера

ВСТУП

Останні досягнення в галузі технологій дозволяють збирати і накопичувати велику кількість даних в різних галузях застосування. Особливе значення мають дані представлені у вигляді часових рядів. І справді, даний формат даних широко використовується у різних сферах, від енергетики та фінансів до охорони здоров’я та хмарних обчислень.

Зі збільшенням даних, збільшується і кількість форм аналізу, що дозволяють виявляти стратегічно важливу інформацію. Виявлення аномалій в часових рядах також належить до цих форм аналізу.

Методи ідентифікації аномалій дозволяють виявляти нехарактерні для процесу спостереження, що в загальному випадку несуть критичну інформацію. Так оперативне виявлення і реагування на спостереження даного характеру можуть збільшити прибуток компанії або уникнути грандіозних збитків, а комусь просто зберегти життя. Деякі з еталонних прикладів застосування наведено нижче:

- моніторинг стану обладнання;
- відстежування шахрайських транзакцій;
- моніторинг несподіваних коливань на фінансовому ринку акцій;
- облік показників стану здоров’я;
- аналіз телеметричних даних;

Саме тому створення системи ідентифікації аномалій в часових рядах є актуальною задачею.

1 ПОСТАНОВКА ЗАДАЧІ

Метою даної дипломної роботи є підвищення ефективності ідентифікації аномалій в даних вимірювання.

Для досягнення мети необхідно виконати наступні завдання:

а) розглянути існуючі підходи та математичні методи виявлення аномалій;

б) спроектувати та реалізувати цільову систему;

в) провести випробування розробленої системи та зробити висновки.

Розроблена система повинна виконувати наступні функції:

- ітеративно ідентифікувати аномалії (online режим);
- виявляти аномалії в багатовимірних часових рядах;
- виявляти не менше 75% наявних аномалій;
- не враховувати додаткової інформацію про структуру даних (розподіл, діапазон значень, наявність або відсутність аномалій);

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ВИЯВЛЕННЯ АНОМАЛІЙ

Для виявлення аномалій використовують напрацювання в сферах статистики, обробки сигналів та машинного навчання. Основна задача, яка ставиться перед методама ідентифікації аномалій формулюється наступним чином: «Виявити спостереження, які не узгоджуються з тими, що вважаються нормальними або ймовірними з точки зору розподілу».

В загальному випадку не існує чіткої межі між даними, які визначені як нормальні, і тими, які відносять до категорії аномальних, що безумовно є основною проблемою при вирішенні даної задачі. Крім того, аномалії є достатньо рідкими подіями, що призводить до незбалансованості навчальної вибірки в сторону звичайних спостережень. Тому навчання без учителя, на практиці зустрічається значно частіше ніж з учителем. Також виникає ряд обмежень при роботі з даними, які описують процеси в реальному часі, що є предметом даної роботи.

2.1 Часові ряди та їх властивості

Часовий ряд (англ. «time series») – це послідовність значень деякого показника, які впорядковані в часі. Коли кожному моменту часу відповідає декілька значень різних показників, то ряд називають багатовимірним. Кожне спостереження називають елементом ряду, а послідовність значень, які належать до різних показників називають рівнями ряду [1]. Існує ряд особливостей, які відрізняють часові ряди від інших типів даних.

По-перше, елементи часового ряду зазвичай містять додатковий шум [2]. Методи обробки сигналів, такі як пониження розмірності, вейвлет аналіз або фільтрація, можуть бути використані для вирішення цієї проблеми. Однак при цьому треба звернути увагу, що велика кількість інформації може бути втрачена.

По-друге, маючи певний часовий ряд, немає цілковитої впевненості, що досліджуваний процес описується за тими законами, які були отримані з наявної кількості інформації [3]. Наприклад, у фінансових даних при спостереженні за одиничним процесом, який описує лише невеликий аспект системи, ймовірно недостатньо інформації для прогнозування поведінки всієї системи.

По-третє, часові ряди мають явну залежність від часу [3]. Іншими словами маючи вхідні дані x(t) в момент часу t, передбачення моделі буде y(t), однак ідентичні вхідні дані в майбутньому будуть мати неідентичні передбачення моделі. Щоб вирішити цю проблему, модель повинна додатково приймати на вхід минулі дані або ж зберігати інформацію про попередні спостереження.

По-четверте, більшість часових рядів нестаціонарні, тобто такі характеристики даних, як середнє, дисперсія та частота змінюються протягом часу [2].

Часові ряди, як тип даних мають індивідуальні характеристики, які ускладнюють проведення аналізу та моделювання. Тому часові ряди є активним об’єктом аналізу серед науковців, що породило широкий спектр методів та підходів для взаємодії з ними.

2.2 Типи аномалій

Важливим аспектом при розгляді методів виявлення аномалій є поняття аномалії. Аномалії можна класифікувати на наступні три класи:

- Точкова аномалія. Якщо одиничний елемент ряду знаходиться окремо відносно решти спостережень, то даний елемент називають точковою аномалією [4]. Даний тип аномалій найчастіше зустрічається в роботах дослідників. Прикладом точкової аномалії може бути одинична транзакція великого об’єму, яка не попадає в діапазон попередніх транзакцій власника банківського рахунку;
- Контекстуальні (умовні) аномалії. Коли елемент ряду є аномалією лише в

певному контексті [4]. Контекст визначається умовними атрибутами, які зазвичай відносяться до часу або місця знаходження. Наприклад, кількість пасажирів таксі на Різдв’яні свята та будні дні;

- Колективні аномалії. Якщо послідовність взаємопов’язаних елементів ряду, які значно виділяються із загальної закономірності називаються колективною аномалією [4]. При цьому, по одинці ці спостереження можуть не відноситись до точкових аномалій. Наприклад, нехарактерна зміна електричного імпульсу, що виникає в окремих кардіоміцитах та відслідковується за допомогою кардіограми серця.

2.3 Одновимірні часові ряди

Загальна схема методів для одновимірних часових рядів зображена на рисунку 2.1

Рисунок 2.1 – Загальна схема одновимірних методів

2.3.1 Методи подібності

Усі методи даного класу базуються на понятті функції відстані або ж подібності між двома наборами даних. Функція відстані/подібності не обов'язково повинна бути метричною, але додатно визначена і симетрична [5].

Методи даної групи ідентифікують елемент ряду як аномалія, якщо на відстані *R* від цього спостереження знаходиться менше τ елементів.

$$\left\{x \in X | d\left(x, x_t\right) \leqslant R\right\} \leqslant \tau, \tag{2.1}$$

де d – функція відстані/подібності; x_t – елемент ряду в момент часу t; X – множина всіх елементів ряду; $R \in \mathbb{R}^+$.

Даний клас методів широко використовується у нечасових даних, оскільки концепція сусідства в порядкованих даних є значно складнішою. Однак Анджіулі та Фасетті [6, 7], а згодом і Ішімцев [8] застосували цей підхід із ковзаючим вікном, що дозволило ітеративно визначати чи новий елемент знаходиться по сусідству із попередніми спостереженнями.

2.3.2 Статистичні методи

Виявлення аномалій за допомогою статистичних підходів базується на оцінці функції щільності досліджуваної величини та припущені, що допустимі значення зустрічаються значно частіше ніж аномальні. Піментел[5] і Чандола[4] виділяють дві підкатегорії статистичних методів: параметричні та непараметричні (табл. 2.1).

Параметричні методи. Методи даного класу припускають, що дані згенеровані з розподілу з параметрами θ. Для неперервних значень найчастіше використовують гаусівський розподіл. Для визначення параметрів θ використовують

метод максимальної правдоподібності (maximum likelihood estimates). В загальному випадку дані не обов’язково описуються нормальним розподілом, тому використовують гаусівські змішані моделі (англ. Gaussian Mixture Model, GMM) моделі або інші типи розподілів, такі як гамма, розподіл Пуассона, Ст’юдента або Вейбула.

Непараметричні методи. Непараметричні підходи не передбачають, що структура моделі є фіксованою протягом часу, тобто модель зі збільшенням складності досліджуваних даних весь час змінюється. Найпростішим непараметричним підходом є використання гістограми, яка графічно відображає табличні частоти.

Таблиця 2.1 – Перелік статистичних методів

Тип	Назва
Параметричні	Змішана модель (Mixture model)
Параметричні	Теорія екстремальних значень (Extreme value theory)
Параметричні	Модель станів простору (State-space models)
Непараметричні	Ядрова оцінка густини розподілу (Kernel density estimators)
Непараметричні	Негативний відбір (Negative selection)

2.3.3 Передбачувальні моделі

Методи даного класу використовують найбільш інтуітивно зрозуміле визначення аномалії. Елемент ряду є аномалією, якщо відхилення очікуваного значення від спостережуваного перевищує попередньо визначене критичне

значення:

$$\left|x_t - \hat{x}_t\right| < \tau \qquad (2.2)$$

де x_t – елемент ряду; – $\hat{x}_t$очікуване значення.

Особливість методів даної групи полягає в тому, що вони складаються із двох частин. Перша частина обчислює очікуване значення на основі попередніх спостережень $\hat{x}_t$, а друга знаходить критичне значення τ . Зазвичай друга складова алгоритму використовує статистичні методи для залишків (residuals).

Наприклад, в роботі був представлений [9] алгоритм DeepAnT, який використовує згорткові нейронні мережі (англ. Convolutional Neural Network, CNNs) для прогнозування наступного значення. Ахмад в співавторстві [10] запропонували модель ієрархічної тимчасової пам’яті (англ. Hierarchical Temporal Memory), яка використовує теорію Гебба для навчання. Структура моделі складається із деревовидної ієрархії рівнів, які здатні накопичувати знання про складні залежності між вхідними даними.

2.4 Багатовимірні часові ряди

Багатовимірні часові ряди мають два і більше рівні, які зазвичай не є статистично незалежними. Тому використання одновимірних методів без додаткових модифікацій може призвести до втрати інформації.

Так не беручи до уваги залежність між рівнями часового ряду, Хундман в співавторстві [11] застосували рекурентну нейронну мережу до кожного потоку телеметричних даних. Додатково автор представив динамічний метод визначення порогового значення для ідентифікації аномалій. Однак існують підходи, які дозволяють досягти кращих результатів.

2.4.1 Одновимірні підходи

Щоб уникнути втрати інформації і при цьому використовувати методи одновимірного аналізу, деякі дослідники використовують методи попередньої обробки для пошуку нового набору незалежних змінних (рівнів ряду). Зазвичай це підходи, що базуються на зменшенні розмірності.

В роботі [12] було запропоновано інкрементний алгоритм головних компонент (англ. Principal Component Analysis) для визначення нових незалежних рівнів ряду. Для безпосереднього виявлення аномалій в подальшому можна застосувати авторегресивні моделі до отриманого представлення.

Подібним чином, Барагона та Баттаглія [13] запропонували використовувати аналіз незалежних компонентів (англ. Independent Component Analysis), щоб отримати набір незалежних адативних негаусівських компонент. Виявлення аномалій здійснюється для кожної компоненти окремо, якщо окреме значення компоненти відхиляється від середнього цієї ж компоненти більш ніж на 4.47 стандартні відхилення, то дане спостереження є аномальним.

Алгоритм зменшення розмірності за допомогою проекцій [14] здійснює пошук найкращої проекції, яка відділяє аномалії від іншої групи спостережень. Пошук оптимального рішення еквівалентний максимізації або мінімізації коефіцієнту ексцесу. Потім ітеративно застосовуються одновимірні статистичні тести до кожного спроектованого рівня для виявлення аномалій.

Інші методи виконують перетворення багатовимірних часових рядів в одновимірні. Лу в співавторстві [15] запропонував трансформацію, яка використовує взаємокореляційну функцію між сусідніми в часі елементами ряду. Елемент ряду одновимірної послідовності є аномалією, якщо він слабо корелює із відповідним елементом початкової послідовності. Порогове значення коефіцієнта кореляції визначається на кожній ітерації за допомогою методу Оцу.

2.4.2 Багатовимірні підходи

На відмінну від одновимірних підходів, багатовимірні використовують інформацію про всі рівні часового ряду цілком, при цьому не використовуючи додаткових трансформацій. Основна частина багатовимірних методів представлена нейронними мережами.

Рекурентні нейронні мережі. Рекурентні нейронні мережі (англ. Recurrent Neural Networks, RNNs)– це особливий клас нейронних мереж, які призначені для обробки послідовних наборів значень. RNNs містять змінні пам’яті, щоб зберігати минулу інформацію, яка разом з теперішніми вхідними даними використовується для обчислення вихідних значень. На практиці використовують дві архітектурні модифікації: довга короткочасна пам’ять (англ. Long Short-Term Memory, LSTM) та вентильні рекурентні вузли (англ. Gated Recurrent Units, GRU).

Малхотра в співавторстві [16] запропонували LSTM модель з декількома шарами, яка навчається на нормальних даних. Обчислені залишки (різниця між модельним значенням і реальним) на навчальній вибірці, описуються багатовимірним гаусівським розподілом. Таким чином, використовуючи перевірку статистичних гіпотез підбирається відповідний рівень значущості, який на тестовій вибірці дозволяє досягти адекватних результатів.

Пізніше, Ерген в співавторстві [17] запропонували гібридний підхід, який використовує LSTM модель для послідовних вхідних значень і метод опорних векторів (англ. Support Vector Machine, SVM) в якості детектора аномалій. Особливість даного рішення унікальна в тому плані, що навчання LSTM та SVM відбувається одночасно.

Згорткові нейронні мережі. Згорткові нейронні мережі – це категорія нейронних мереж, які широко використовуються для обробки зображень. CNNs здатні розпізнавати складні патерни за допомогою послідовних перетворень вхідних даних.

Хоча RNNs класично розглядаються як найкращий підхід при роботі із

часовими послідовностями, дослідження [18] показують, що CNNs можуть досягати кращих результатів ніж LSTM. CNNs найчастіше використовуються для вилучення взаємозв’язків в зображеннях, однак вони можуть бути застосовані для пошуку складних прихованих взаємозв’язків у даних з часовою залежністю. У цьому випадку CNNs іноді об’єднують з деякими RNNs. Наприклад, ConvLSTM[19].

Автоенкодери. Автоенкодери (англ. Autoencoder, AE) – це категорія нейронних мереж, які виконують нелінійне зменшення розмірності. Найчастіше використовуються для виявлення аномалій [5]. Вони представляють вхідні дані в латентному просторі, а потім намагаються відновити початкове представлення.

Останні методи виявлення аномалій на основі AE, включаючи гібридні, досягають значно кращих результатів в порівнянні зі звичайними RNNs та CNNs.

У випадку використання AE із RNNs, варто згадати дослідження Мальхотра в співавторстві [20]. Автори запропонували модель LSTM Encoder-Decoder, яка виконує реконструкції початкових даних. Навчання здійснюється на вибірці без аномалій, таким чином похибки реконструкції для аномальних значень є значно більшими. Потім використовуючи перевірку статистичних гіпотез підбирається відповідний рівень значущості, який на валідаційній вибірці дає найкращі результати.

У роботі [21] представлено Deep Autoencoder (DAE) для виявлення аномалій в багатовимірних часових рядах. DAE – це багатошаровий автоенкодер, який здатний розпізнавати декілька рівнів абстракції вхідних даних. При роботі з часовими рядами, використовують методи перехресних ковзаючих вікон як попередній етап підготовки вхідних даних. На наступних етапах, середня помилка реконструкції для всіх рівнів часового ряду використовується як оцінка аномалії.

Генеративні моделі. Генеративні моделі – це категорія нейронних мереж, які намагаються створювати екземпляри даних, які схожі на ті, що знаходяться в оригінальній вибірці.

Генаративно-змагальні мережі (англ. Generative adversarial networks, GANs) – це один із видів генеративних мереж, які запропонував Гудфеллоу [22]. Вони складаються з двох конкуруючих моделей: генератора та дискрімінатора. Генератор

намагається змоделювати дані, які як найкраще відповідають навчальним даним, тоді як дискрімінатор вчиться розрізняти згенеровані дані та реальні. Останні роботи виявлення аномалій за допомогою генеративно-змагальних мереж наведені в [23-27].

Варіаційні автоенкодери (англ. Variational Autoencoder, VAE) – це один із видів генеративних мереж, які широко використовуються для виявлення аномалій. Архітектура мережі створена на основі класичних AE, хоча з математичної точки зору вони цілком різні. VAE намагається створити нові набори даних, використовуючи латентний простір, який зазвичай відповідає гаусівському розподілу. Аномалії досить погано відновлюються через процес генерації, тому як наслідок VAE дозволяє визначити який з елементів ряду значно відхиляється від очікуваних.

Використання RNNs в якості енкодера та декодера, дозволило узагальнити здібності VAE для часових рядів. Одна із модифікацій зустрічається під терміном стохастична нейронна мережа (Stochastic Recurrent Network) [28].

Припускається, що латентний простір описується гаусівським розподілом, що в певній мірі є обмеженням, тому існує ряд модифікацій, які дозволяють уникнути даного припущення. Останні роботи намагалися моделювати більш складні розподіли латентного простору за допомогою енергетичних моделей [29] або змішаних моделей[30].

2.5 Порівняльна характеристика

Список методів розглянутих в попередніх підрозділах наведено на рисунку 2.2. Модифікації методів не розглядаються.

Рисунок 2.2 – Список розглянутих методів

Враховуючи кількість різноманітних методів та необмежену кількість архітектурних реалізацій, порівняльна характеристика здійснюється між категоріями методів.

Критерії, які враховуються при аналізі:

- вимоги до системи;
- характеристики часових рядів;
- особливості методів;

Таблиця 2.2 – Порівняльна характеристика методів

Методи	Залежність між рівнями ряду	Необхідність навчання	Наявність аномалій в навчальні вибірці	Критичне значення
Подібності	Не враховує	Не потрібно	--	Задається вручну

Продовження таблиці 2.2

Методи	Залежність між рівнями ряду	Необхідність навчання	Наявність аномалій в навчальні вибірці	Критичне значення
Статистичні	Не враховує	Залежить від типу	Залежить від типу	Залежить від типу
Зменшення розмірності	Враховує	Залежить від типу	Впливає на результат	Не використовується
Передбачувальні	Враховує	Потрібно	Впливає на результат	Задається вручну
Генеративні	Враховує	Потрібно	Не впливає	Задається вручну

2.6 Висновки до розділу

На основі проведеного порівняльного аналізу можна зробити висновок, що єдиного методу вирішення задачі, який задовільняє всім критеріям, банально, не існує. Для досягнення мети необхідно задіяти комплекс методів.

Кореляція, яка існує між різними рівнями часового ряду не може бути врахована за допомогою статистичних методів та методів подібності. І оскільки, всі методи, що залишились в тій чи іншій степені передбачають навчання на навчальній вибірці, вплив аномалій на кінцевий результат стає критичним. Тому пошук конкретного методу здійснювався серед генеративних моделей.

Проаналізувавши особливості генеративно-змагальних мереж, можна зробити наступні висновки:

- параметри моделі коливаються і ніколи не сходяться до єдиного варіанту;
- генерується обмежена кількість екземплярів даних;
- навчання дискрімінатора випереджає навчання генератора, і як наслідок, генератор не може навчитись генерувати якісні об’єкти;

Таким чином було вибрано модель варіаційного автоенкодера, що широко використовується для виявлення аномалій і немає проблем наведених для генеративно-змагальних мереж.

В якості другої складової алгоритму (вибір критичного значення) було використано один із алгоритмів теорії екстремальних значень. Даний вибір не вимагає припущень про розподіл спостережень, а відповідно є універсальним для різних наборів даних.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Опис проблеми

Нехай $x = [x_1, x_2, \ldots, x_n]$ багатовимірний часовий ряд, який має *n* спостережень кожне з яких є *m*-вимірним вектором $x_t = [x_t^1, x_t^2, \ldots, x_t^m]$. Будемо позначати $x_{t+T:t}$ послідовність спостережень із T елементів $\{x_t, x_{t+1}, \ldots, x_{t+T-1}, x_{t+T}\}$. Необхідно з'ясувати чи елемент ряду x_{t+T} є точковою аномалією.

Загальний підхід вирішення поставленої задачі, складається з наступних етапів:

а) здійснити попередню обробку вхідних даних:

1) привести часовий ряд до стаціонарного вигляду;

2) провести маштабування вхідних параметрів;

б) визначити параметри розподілу (μ, σ^2) для спостереження x_{t+T} за допомогою нейронної мережі варіаційного автоенкодера;

в) обчислити логарифм ймовірності, що x_{t+T} знаходиться в даному діапазоні $\log N(x_{t+T}|\mu, \sigma^2)$;

г) обчислити логарифм ймовірності, що значення латентних змінних $[z_1^T, z_2^T, \ldots, z_k^T]$ для x_{t+T} описуються багатовимірним нормальним розподілом $\log N([z_1^T, z_2^T, \ldots, z_k^T]|0, I)$;

д) застосувати SPOT (Steaming POT) алгоритм для отриманих значень на етапі в) або г) (вибір користувача), щоб з'ясувати мітку («аномальне значення», «допустиме значення»);

3.2 Обробка вхідних даних

Особливості обробки вхідних даних пов’язані з особливостями навчання нейронних мереж. З певних міркувань варіант, коли середнє значення знаходиться близько біля нуля є найкращим. Розглядаючи випадок із нестаціонарними рядами, дане значення змінне у часі, що впливає на процес навчання нейронних мереж.

Враховуючи вище наведені міркування, обробка відбувається за наступним планом:

а) провести тест Дікі-Фуллера (рівень значущості 0.01) для кожного рівня початкового ряду;

б) провести диференціювання, якщо р-значення буде меншим 0.01;

в) провести стандартизацію за наступною формулою:

$$z = \frac{x - \mu}{\sigma}, \tag{3.1}$$

де μ – середнє значення; σ – стандартне відхилення значень конкретного рівня ряду

Варто зауважити, що маштабування вхідних даних до діапазону [0, 1] не є вірним. Оскільки аномальні значення не обмежені ні зверху ні знизу, тому процес нормалізації відбувався за формулою 3.1

3.3 Варіаційні автоенкодери

Варіаційні автоенкодери відносяться до класу генеративних моделей, які використовують архітектуру автоенкодерів з певними вдосконаленями та баєвський апарат формування припущень про структуру даних. Вперше були запропоновані в 2013 році Кінгма та Резенде [31].

Автори роботи розглядали процес генерації нових даних $\{x_i\}_i^N$ зі сторони ймовірностних моделей. Загальна схема зображена на наступному рисунку.

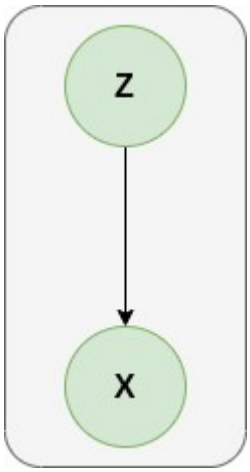

Рисунок 3.1 – Концепція генеративних моделей

Формально, ілюстрацію на рисунку 3.1 можна інтерпретувати як послідовність дій:

а) згенерувати значення z_i із розподілу p(z);

б) згенерувати значення x_i із розподілу p(x|z);

Власне, автори роботи взяли за генеруючу модель спільний розподіл p(x,z) між змінними x та z, який можна представити в наступному вигляді:

$$p(x,z)=p(x|z)\,p(z), \tag{3.2}$$

де p(z) прийнято називати апріорним розподілом, а p(x|z) – правдоподібним (likelihood) розподілом ймовірності.

Основне завдання, яке виникло при розгляді даної моделі, було з’ясувати найкраще значення z для генерації спостережуваного x. Інакше кажучи, з’ясувати апостеріорний розподіл p(z|x). Використовуючи теорему Баєса, отримуємо наступне представлення:

$$p(z|x)=\frac{p(x|z)\cdot p(z)}{p(x)}, \tag{3.3}$$

Основна проблема, яка виникала при безпосередньому використанні теореми Баєса полягала в обчисленні відособленого розподілу p(x). В роботі [31] наведено дві причини, чому необхідно виконати велику кількість обчислень. Тому, Кінгма та Резенде запропонували інший підхід, який використовує архітектуру автоенкодерів для пошуку розподілу $q_\theta(z|x)$, що апроксимує дійсний розподіл p(z|x). Загальна схема зображена на рисунку 3.2

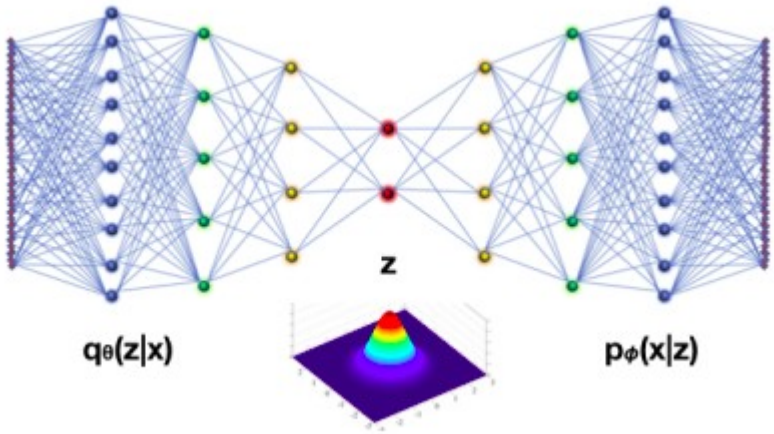

Рисунок 3.2 – Архітектура варіаційного автоенкодера [32]

Для оцінки якості апроксимації використовують дивергенцію Кульбака - Лейблера.

$$D_{KL}\left(q_\theta(z|x_i)\parallel p(z|x_i)\right)=\int q_\theta(z|x_i)\log\left(\frac{p(z|x_i)}{q_\theta(z|x_i)}\right)dz\geq 0 \tag{3.4}$$

Виконавши всі перетворення наведені в роботі [32], в кінцевому результаті отримаємо нерівність:

$$\log\left(p(x_i)\right) \geq D_{KL}\left(q_\theta(z|x_i) \parallel p(z)\right) + E_{z\sim q_\theta(z|x_i)}\left[\log p_\phi(x_i|z)\right] \tag{3.5}$$

Праву частину даної нерівність називають нижньою межею припущень (Evidence Lower Bound – ELBO).

На практиці в якості латентного простору вибирають гаусівський розподіл, що дозволяє спростити нерівність 3.5.

Таким чином:

$$p(z) \to \frac{1}{\sqrt{2\pi\sigma_p^2}} e^{\frac{-(x-\mu_p)^2}{2\sigma_p^2}}, \tag{3.6}$$

$$q_\theta(z|x_i) \to \frac{1}{\sqrt{2\pi\sigma_q^2}} e^{\frac{-(x-\mu_q)^2}{2\sigma_q^2}} \tag{3.7}$$

Тоді дивергенція Кульбака-Лейблера між $q_\theta(z|x_i)$ та $p(z)$ дорівнюватиме:

$$D_{KL}\left(q_\theta(z|x_i) \parallel p(z)\right) = \log\left(\frac{\sigma_q}{\sigma_p}\right) - \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} + \frac{1}{2} \tag{3.8}$$

І якщо взяти $\sigma_p = 1$ і $\mu_p = 0$, то кінцевий вигляд функціоналу, який необхідно мінімізувати матиме наступний вигляд:

$$G = -\sum_{j=1}^{J} \frac{1}{2}\left[1 + \log\left(\sigma_j^2\right) - \sigma_j^2 - \mu_j^2\right] - \frac{1}{L}\sum_{l} E_{z\sim q_\theta(z|x_i)}\left[\log p_\phi\left(x_i|z^{(i,l)}\right)\right], \tag{3.9}$$

де J – розмірність вектора z; L – число згенерованих значень x.

Навчання ж VAE полягає в тому, щоб знайти такі параметри θ та ϕ, щоб функціонал 3.9 досягав мінімального значення на існуючих даних.

3.4 Теорія екстремальних значень

Теорія екстремальних значень належить до статистичних інструментів, які дозволяють дізнатися порогове значення (квантиль) для ідентифікації особливо рідких спостережень без додаткових припущень про розподіл випадкової величини.

3.4.1 Розподіл екстремальних значень

Припустимо, що X – випадкова велична із комулятивною функцією розподілу F(x) = P(X ≤ x), тоді одиничні екстремальні спостереження будуть знаходитися на кінцях даного розподілу і матимуть наступну комулятивну функцію $\widetilde{F}(x)=1-F(x)=P(X>x)$. Необхідно знайти таке значення z_q, що ймовірність зустріти спостереження $x>z_q$ є меншим ніж q, тобто $P(X > z_q) < q$.

В роботі Фішера та Тіппета (1928р.) було доведено, що спостереження на кінцях розподілу можна описати за допомогою розподілу екстремальних значень (Extreme Value Distributions – EVD). Формулюється даний розподіл наступним чином:

$$G_\gamma : x \to \exp\left(-(1+\gamma x)^{\frac{-1}{\gamma}}\right), \gamma \in R, 1+\gamma x>0, \tag{3.10}$$

де параметр γ називають індексом екстремальних значень.

Таким чином, початкова задача дещо спрощується, оскільки не потрібно визначати розподіл $\widetilde{F}(x)$. Однак, необхідно знайти таке значення γ, щоб G_γ найкращим чином описувало поведінку спостережень на кінцях розподілу. Ілюстрація на рисунку 3.3 відображає концепцію даного підходу.

Перші підходи для пошуку параметру γ запропонував Хіл та Пікандс [33],

однак їхні методи працюють для обмеженої кількості функцій розподілу і в загальному випадку не гарантують збіжності апроксимації. Тому використовують інший підхід Peaks Over-Threshold (POT).

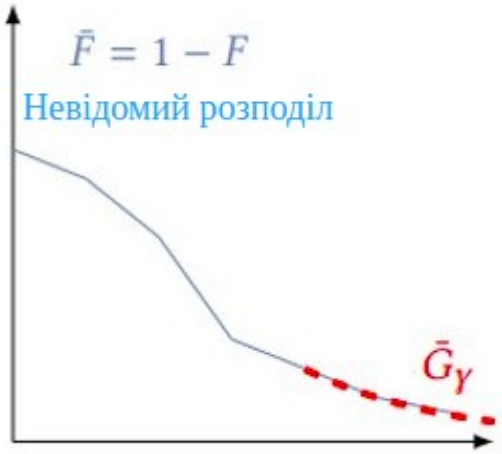

Рисунок 3.3 – Апроксимація невідомого розподілу

3.4.2 Peaks Over-Threshold (POT)

Альтернативою для підходу, що використовує розподіл екстремальних значень може слугувати Peaks Over Threshold [33], який на відмінну від першого намагається апроксимувати узагальнений розподіл Парето для значень X, які перевищують порогове значення t. Тобто, наступний розподіл:

$$\widetilde{F}_t(x) = P(X - t > x | X > t) \underset{t \to \tau}{\sim} \left(1 + \frac{\gamma x}{\sigma}\right)^{\frac{-1}{\gamma}}, \tag{3.11}$$

де t – порогове значення; X - t – різниця між X_i та пороговим значенням t; τ – границя початково розподілу *F(x);* γ та σ параметри узагальненого розподілу Паретто (в даному випадку μ = 0).

Таким чином, оцінивши параметри γ та σ, квантиль z_q можна знайти за наступною формулою:

$$z_q \simeq t + \frac{\hat{\sigma}}{\hat{\gamma}}\left(\left(\frac{qn}{N_t}\right)^{-\hat{\gamma}} - 1\right), \tag{3.12}$$

де t – порогове значення; q – бажана ймовірність; n – загальна кількість спостережень; N_t – кількість «піків»(спостережень X, які перевищують t).

Для пошуку параметрів γ та σ можна використати метод моментів або інші підходи. Однак, в даній роботі буде розглянуто метод максимальної правдоподібності.

3.4.3 Оцінка параметрів розподілу

Маючи вибірку {X1, X2, …, X_n} із n спостережень і функцію щільності f_θ (в нашому випадку узагальнений розподіл Паретто), функція максимальної правдоподібності матиме наступний вигляд:

$$logI = -N_t log\sigma - \left(1 + \frac{1}{\gamma}\right)\sum_{i=1}^{N_t} \log\left(1 + \frac{\gamma}{\sigma} Y_i\right), \tag{3.13}$$

де Y_i – різниця між спостереженням X_i та пороговим значенням t коли $X_i > t$.

Оскільки, оптимізація здійснюється чисельними методами для зменшення впливу похибок обчислень на кінцевий результат та підвищення стійкості збіжності використовують підхід Грімшов [34].

Основна ідея полягає в зведенні багатовимірної задачі оптимізації до одновимірного випадку. Нехай $l(\gamma,\sigma) = logI(\gamma,\sigma)$, тоді екстремум можна знайти розв’язавши систему $\nabla l(\gamma,\sigma)$. Грімшов показав, якщо існує розв’зок $(\tilde{\gamma},\tilde{\sigma})$ цієї системи

, то $\tilde{x}=\tilde{\gamma}/\tilde{\sigma}$ є розв’язком рівняння $u(x)\cdot v(x)=1$ де:

$$u(x)=\frac{1}{N_t}\sum_{i=1}^{N_t}\frac{1}{1+x\cdot Y_i}, \tag{3.14}$$

$$v(x)=1+\frac{1}{N_t}\sum_{i=0}^{n_t}\log(1+x\cdot Y_i) \tag{3.15}$$

Таким чином, знайшовши розв’язок $\tilde{x}$, можна знайти значення параметрів $\tilde{\gamma}=v(\tilde{x})-1$, $\tilde{\sigma}=\tilde{\gamma}/\tilde{x}$

3.4.4 SPOT алгоритм

SPOT (Sreaming Peaks Over-Threshold)[33] модифікація алгоритму POT, яка оновлює параметри розподілу (див. формулу 3.11) ітеративно. Кожне нове спостереження так чи інакше впливає на форму узагальненого розподілу Паретто і необхідно здійснювати переобчислення параметрів γ та σ. Загальна схему роботи зображено на рисунку 3.4

Алгоритм складається із двох частин:

а) процес налаштування. На даному етапі обчислюється значення t та z_q. А відповідно і початкові параметри γ та σ узагальненого розподілу Паретто;

б) ітеративний процес. На даному етапі, отримуючи нові спостереження, відбувається оновлення параметрів γ та σ, а також обчислюється нове значення квантилю z_q;

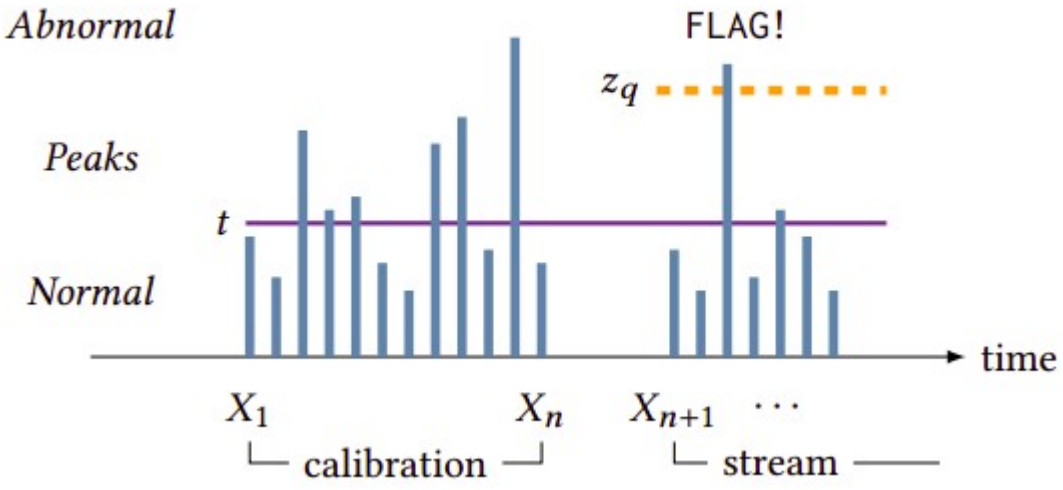

Рисунок 3.4 – Загальна схема роботи SPOT алгоритму [33]

3.5 Висновки до розділу

У розділі було формалізовано задачу ідентифікації аномалій в багатовимірних часових рядах. Запропоновано етапи вирішення поставленої задачі.

Розглянуто особливості роботи варіаційних автоенкодерів. Наведено основні поняття теорії екстремальних значень. Розглянуто чисельні методи пошуку параметрів розподілу.

4 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Загальну структура системи представлено на рис. 4.1. Враховуючи безперервний процес надходження нових вхідних даних та специфіку навчання моделі VAE , систему було розділено на три умовні частини.

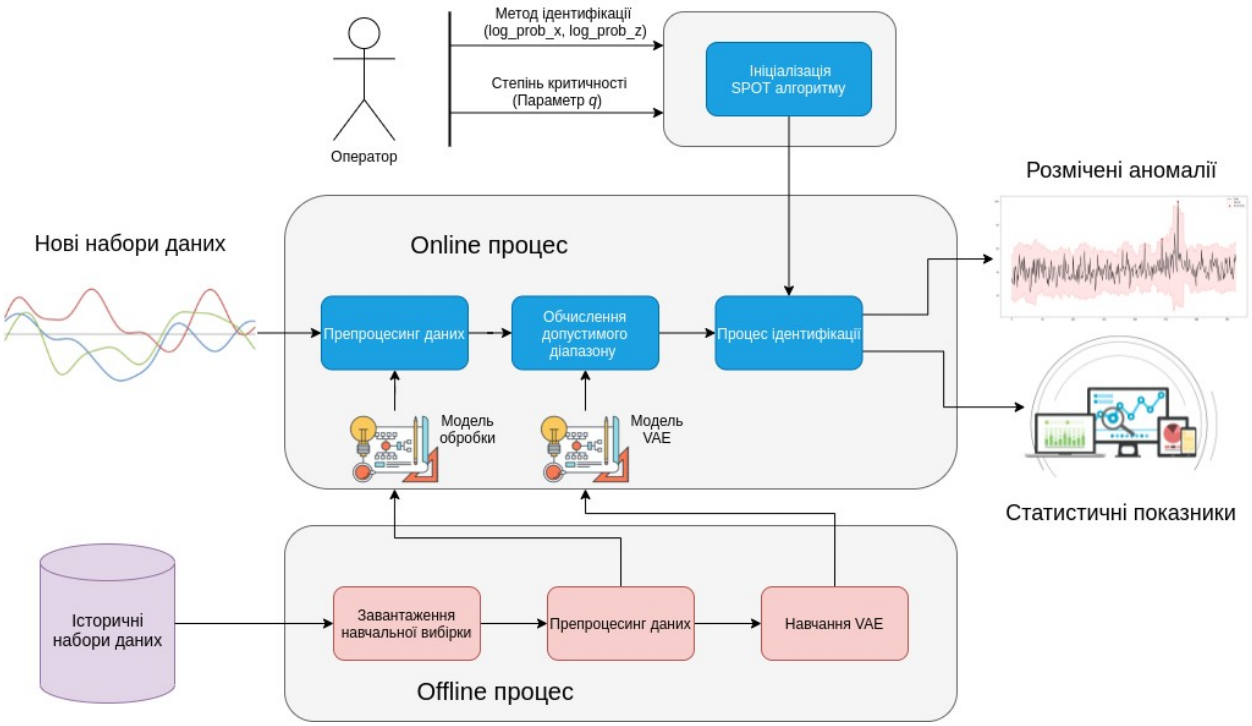

Рисунок 4.1 – Структура системи

Offline частина системи безпосередньо взаємодіє з історичними даними для їх подальшої обробки та навчанням VAE моделі. Додатково зберігається модель обробки, яка містить інформацію про наявність стаціонарності або її відсутність для кожного рівня ряду, а також параметри стандартизації вхідних даних.

Online частина системи взаємодіє із потоковими даними, які в кожен момент часу надходять до системи. Використовуючи модель обробки, дані стандартизуються і за необхідності диференціюються. На наступному етапі за допомогою VAE моделі обчислюються параметри розподілу і передаються в якості аргументів до SPOT алгоритму. Обчисливши нижню межу допустимих значень для

розподілу, який вибрав оператор, кожний елемент ряду ідентифікується як аномальне або допустиме значення. В результатах відображається інформація про діапазон допустимих значень для кожного рівня ряду та ідентифіковані аномалії.

4.1 Опис бібліотек

Програмна реалізація системи повністю написана на мові програмування python. Був задіяний наступний перелік сторонніх бібліотек:

- numpy – містить широкий спектор методів для роботи з матрицями, підтримує векторні обчислення. Використовується для здійснення проміжних обчислень;
- pandas – надбудова над numpy для безпосередньої взаємодії з даними та їх обробки. Використовується для завантаження даних, обробки та збереження;
- sklearn – бібліотека машинного навчання. Використовується для оцінки якості отриманих результатів.
- torch – бібліотека машинного навчання. Використовується для розробки та навчання VAE моделі;
- matplotlib – бібліотека для створення 2D та 3D зображень. Використовується для візуалізації проміжних та кінцевих результатів роботи;
- streamlit – бібліотека для розробки web-застосунків. Використовується для взаємодії з користувачем та представлення результатів;

4.2 Опис функціональних елементів

Розробка програмного забезпечення здійснювалася згідно принципів об’єктно орієнтованого програмування, тому опис функціональних елементів буде

здійснюватися на рівні класів і методів.

До основних елементів системи відносяться наступні класи: Encoder, Decoder, VAE та SPOT.

Складовими компонентами VAE (рис. 4.2). є Encoder та Decoder. Даний клас був створений для спрощення обчислень градієнтів та оновлення параметрів моделі

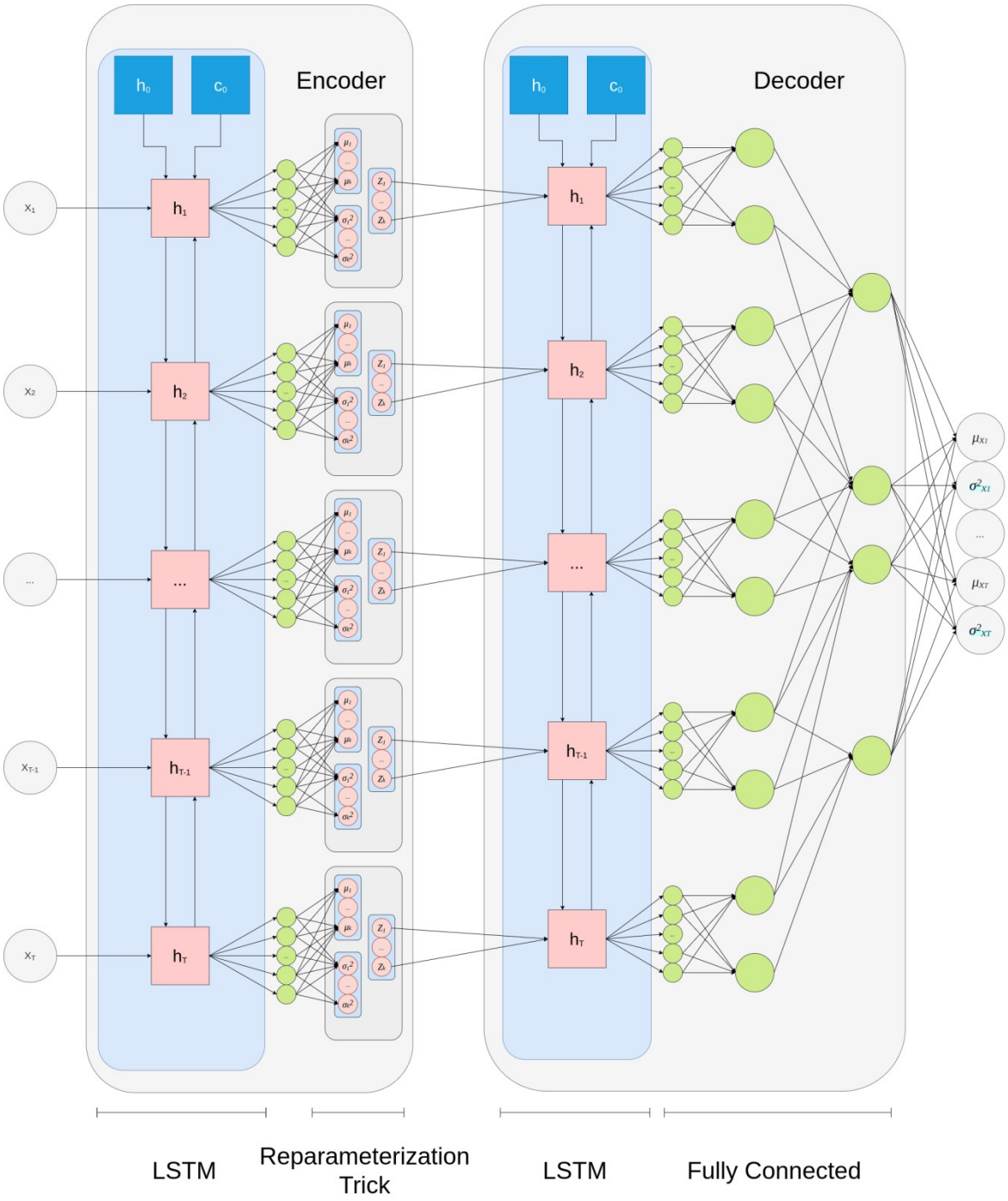

Рисунок 4.2 – Загальна схема варіаційного автоенкодера

4.2.1 Encoder

Клас Encoder (рис. 4.3). наслідує батьківський клас nn.Module, який є базовим для всіх нейронних мереж реалізованих за допомогою фреймворку машинного навчання pytorch (бібліотека torch). Згідно вимог описаних в документації до даного програмного забезпечення кожний модуль нейронної мережі обов’язково повинен мати метод __init__(self) та forward(self) за виключенням лінійних моделей. Перший метод описує компоненти з яких складається модель, а другий визначає порядок обчислень. Опис даного класу (моделі) і наступних буде здійснюватися відносно цих двох методів.

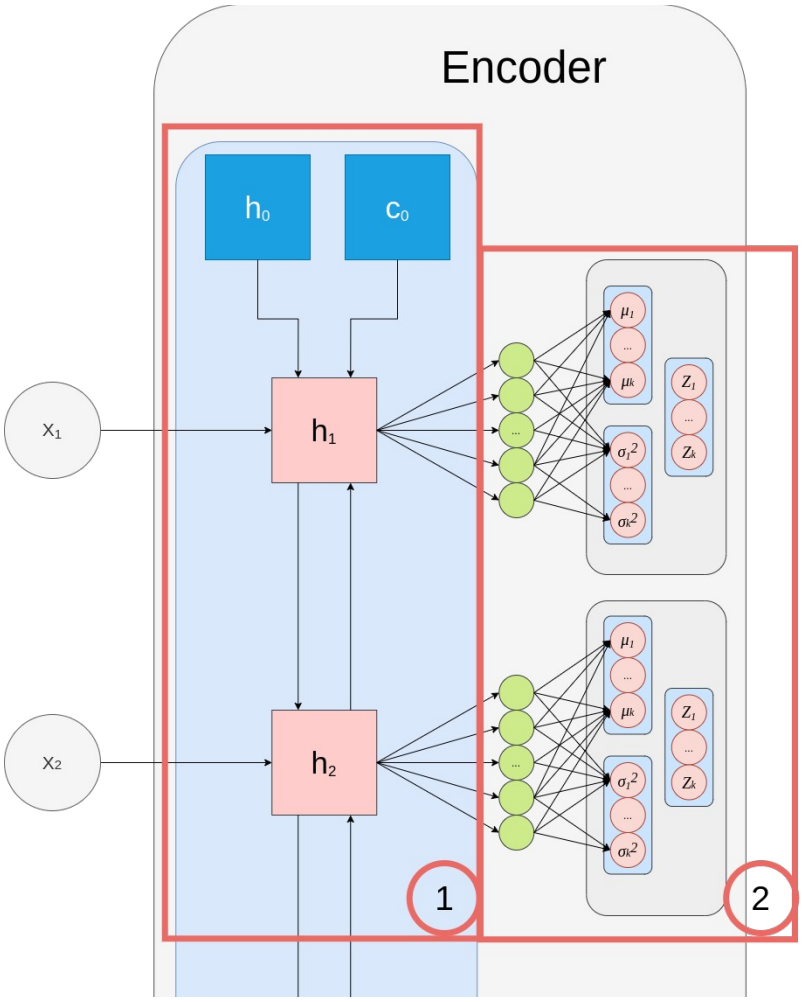

Рисунок 4.3 – Базові елементи Енкодера

__init__: Модель складається із двох ключових компонентів: nn.LSTM (1) та nn.ModuleList (2). Останній містить послідовність компонентів, які обчислюють середнє значення та дисперсію для кожного прихованого стану nn.LSTM. В якості вхідних параметрів виступають параметри: input_size, hidden_size, num_layers, seq_length та z_size.

forward: Схематичну структуру графа обчисленнь наведено на рисунку 4.3. В якості вхідних параметрів виступає матриця X розмірності [batch_size, seq_length, input_size], де batch_size – кількість прикладів, seq_length – розмірність плаваючого вікна та input_size – кількість рівнів часового ряду. Вихідні параметри: матриця z розмірності [batch_size, seq_length, z_size], де z_size – розмірність латентного простору.

4.2.2 Decoder

Decoder був реалізований за схожими принципами, які були описані в попередньому підрозділі. Ключова відмінність полягає в тому, що кожен прихований стан nn.LSTM об’єднується в повнозв’язаний шар (2), який в кінцевому результаті обчислює середнє та дисперсія для кожного вхідного значення ряду.

__init__: Модель складається із двох блоків: nn.LSTM (1) та nn.Sequential (2). В якості вхідних параметрів виступають наступні змінні: z_size, hidden_size, num_layers, seq_length та output_size.

forward: Схематичну структуру графа обчислень наведено на рисунку 4.4. В якості вхідних параметрів виступає матриця Z розмірності [batch_size, seq_length, z_size], де batch_size – кількість прикладів, seq_length – розмірність плаваючого вікна та z_size – розмірність латентного простору. Вихідні параметри: матриця з середніми значеннями розмірності [batch_size, seq_length, output_size] і анологічна для дисперсії. В даному випадку параметри input_size та output_size співпадають.

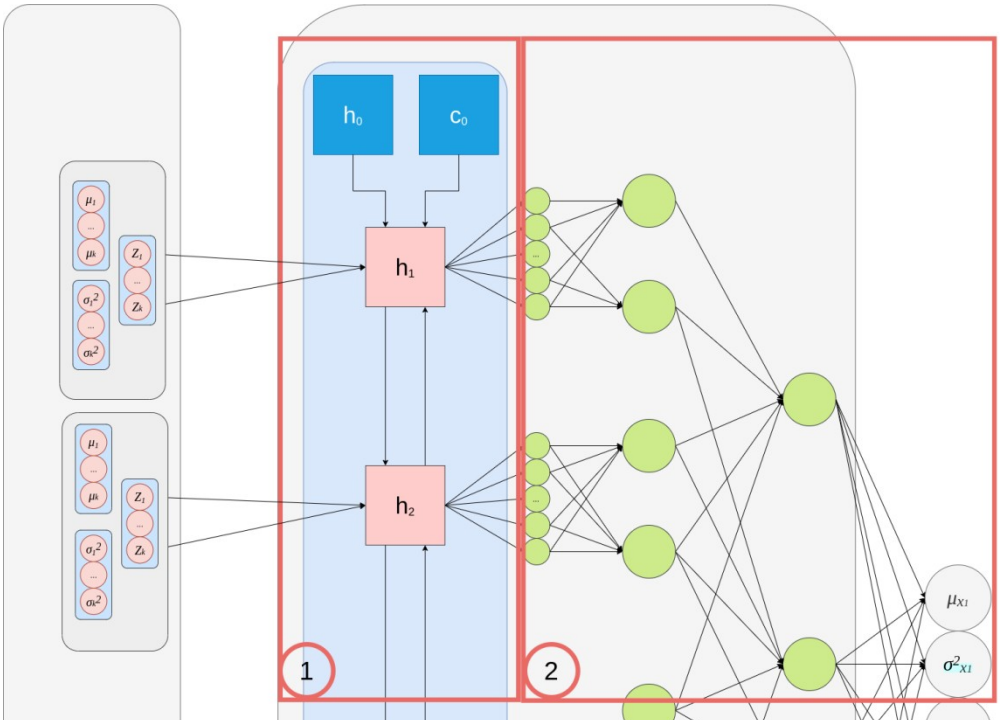

Рисунок 4.4 – Базові елементи Декодера

4.2.3 SPOT

Клас SPOT містить три приватні та три публічні методи. Короткий опис кожного з методів наведений у наступній таблиці.

Таблиця 4.1 – Методи SPOT

Назва	Опис
fit(init_data, data)	Здійснює перевірку типів даних та проводить конвертацію до єдиного формату. Приймає на вхід два набори даних. Один для проведення калібрації, а другий для безпосереднього аналізу
initialize()	Підраховує кількість «піків» на даних для калібрації, а також знаходить початкове критичне значення

Продовження таблиці 4.1

Назва	Опис
run()	Точка запуску SPOT алгоритму
_log_likelihood (Y, gamma, sigma)	Обчислює значення функції максимальної правдоподібності для узагальненого розподілу Паретто з параметрами gamma та sigma
_grimshaw (epsilon, n_points)	Здійснює пошук параметрів γ (gamma) та σ (sigma). В якості аргументів приймає epsilon – допустиму похибку та n_points – максимальна кількість коренів рівняння
_quantile (gamma, sigma)	Обчислює значення квантилю z_q

4.3 Опис інтерфейсу користувача

При проектуванні інтерфейсу насамперед враховувались види діяльності користувача (оператора) із системою. Основні з них можна сформулювати наступним чином:

- прийняття рішень;
- аналіз отриманих рішень;

Таким чином, основні задачі, які ставилась при розробці інтерфейсу формулювалась наступним чином:

- реалізувати віджети керування;
- реалізувати засоби представлення проміжних та кінцевих результатів (графіки, таблиці та поля інформування про теперішній стан системи);

На початковому етапі взаємодії користувача із системою виводиться повідомлення про необхідність вибрати тестовий датафрейм, який в подальшому виступатиме об’єктом досліджень. Реалізовано дану функціональність за допомогою

випадаючого списку із переліком усіх наявних датафреймів (рис. 4.5).

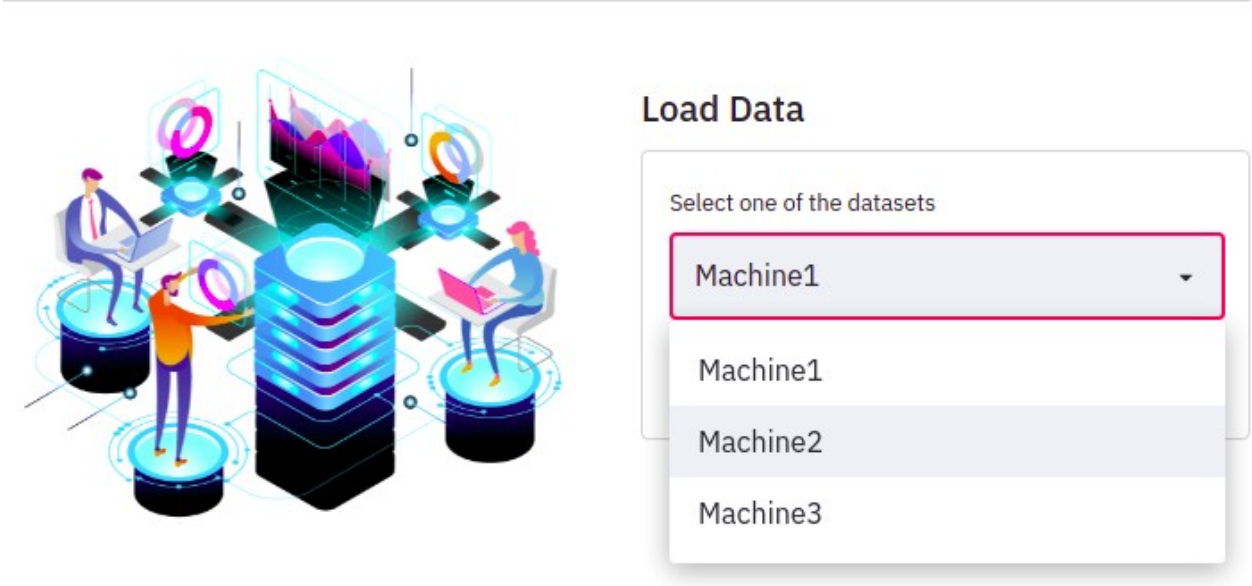

Рисунок 4.5 – Вибір датасету

Після підтвердження вибору користувачем відображається представлення даних. Таким чином, користувач може самостійно робити припущення про розподіл значень та присутні аномалії. Користувач може збільшити маштаб зображення нажавши на віджет у верхній правій частині ілюстрації. В якості прикладу може слугувати зображення на рисунку 4.6.

Іншим функціональним блоком є форма вибору гіперпараметрів для SPOT алгоритму (рис. 4.7). Список використаних віджетів:

- радіо-кнопки для вибору назви розподілу;
- повзунок вибору значень із допустимого діапазону;

Вибравши віджети вище вказаних типів, не потрібно проводити валідацію вхідних даних від користувача. Додатково діапазон значень дозволяє оператору робити вибір без додаткових знань про структуру алгоритму.

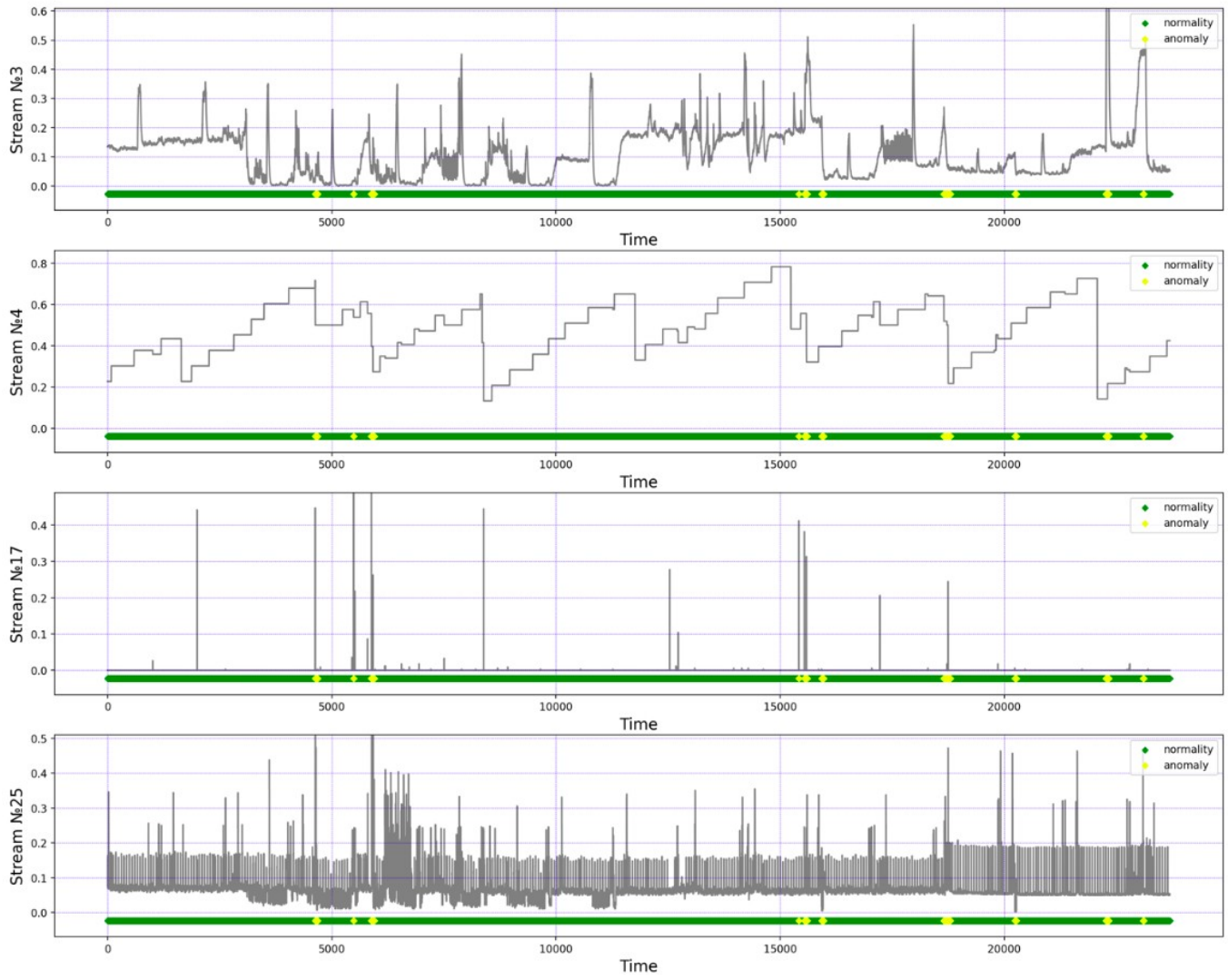

Рисунок 4.6 – Графічне представлення датасету

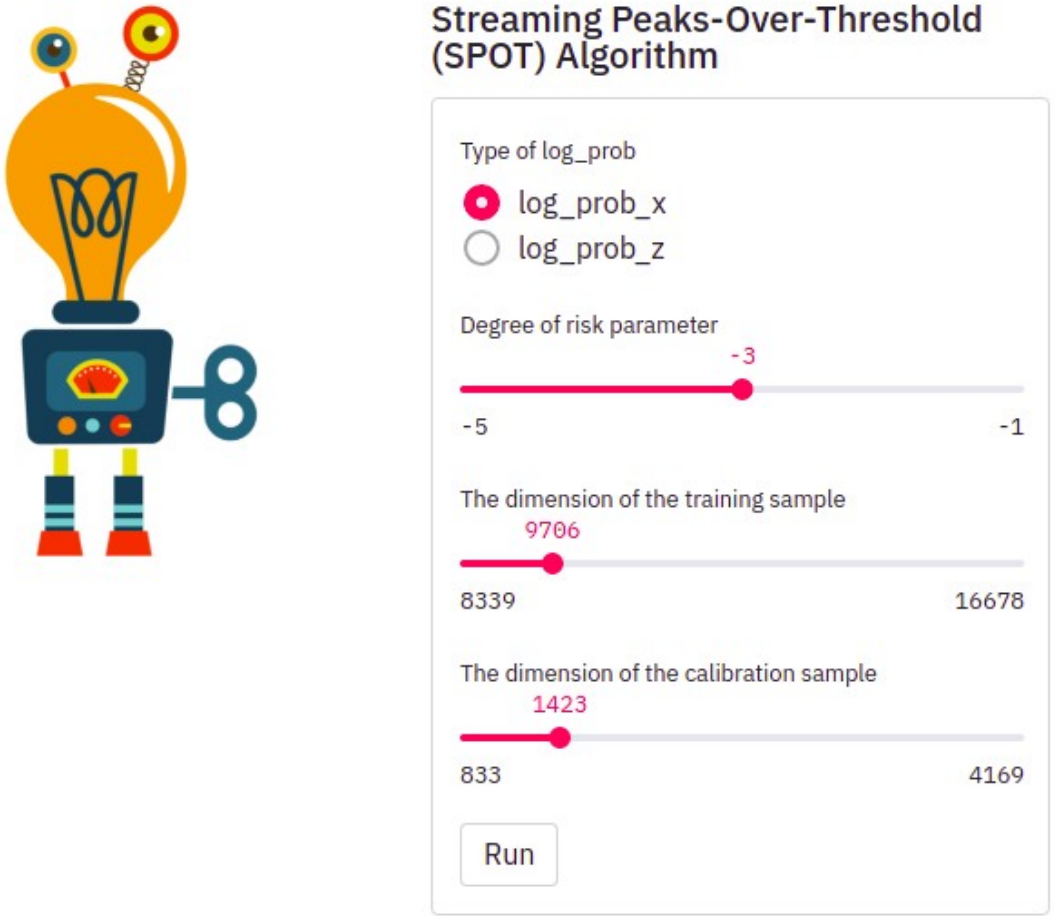

Рисунок 4.7 – Форма вибору гіперпараметрів

4.4 Висновки до розділу

У розділі було описано архітектуру системи ідентифікації аномалій в багатовимірних часових рядах. Наведено деталі реалізації нейронної мережі варіаційного автоенкодера та SPOT алгоритму. Представлено екранні форми інтерфейсу користувача.

5 ВИПРОБУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

5.1 Тестова вибірка

Перевірка працездатності системи відбувається на трьох датасетах, які описують процес роботи серверного обладнання.

Кількість досліджуваних показників для кожного з датасетів відрізняється. Їхні назви відсутні, що як наслідок не дозволяє робити припущення ні про допустимий діапазон значень, ні про динаміку або характер змін. Найважливіша особливість полягає в тому, що кожен із наборів даних містить мітки для кожного спостереження про принадлежність до нормального або аномального значення. Детальна характеристика кожного із датасетів наведена в таблиці 5.1

Таблиця 5.1 – Характеристика вибірки

Назва	Кількість показників	Об’єм навчальної вибірки	Об’єм тестової вибірки	Кількість аномалій
Machine1	5	8000	20479	2694
Machine2	4	8000	15694	542
Machine3	3	8000	15703	817

5.2 Метрики якості

Для оцінювання ефективності роботи системи використовується Precision, Recall та F1 оцінки. Дані метрики можна обчислити за допомогою матриці невідповідності.

5.3 Аналіз результатів

Для проведення повного аналізу розглянемо один із датасетів, наприклад, Machine2. Результати роботи VAE моделі наведено на наступному рисунку.

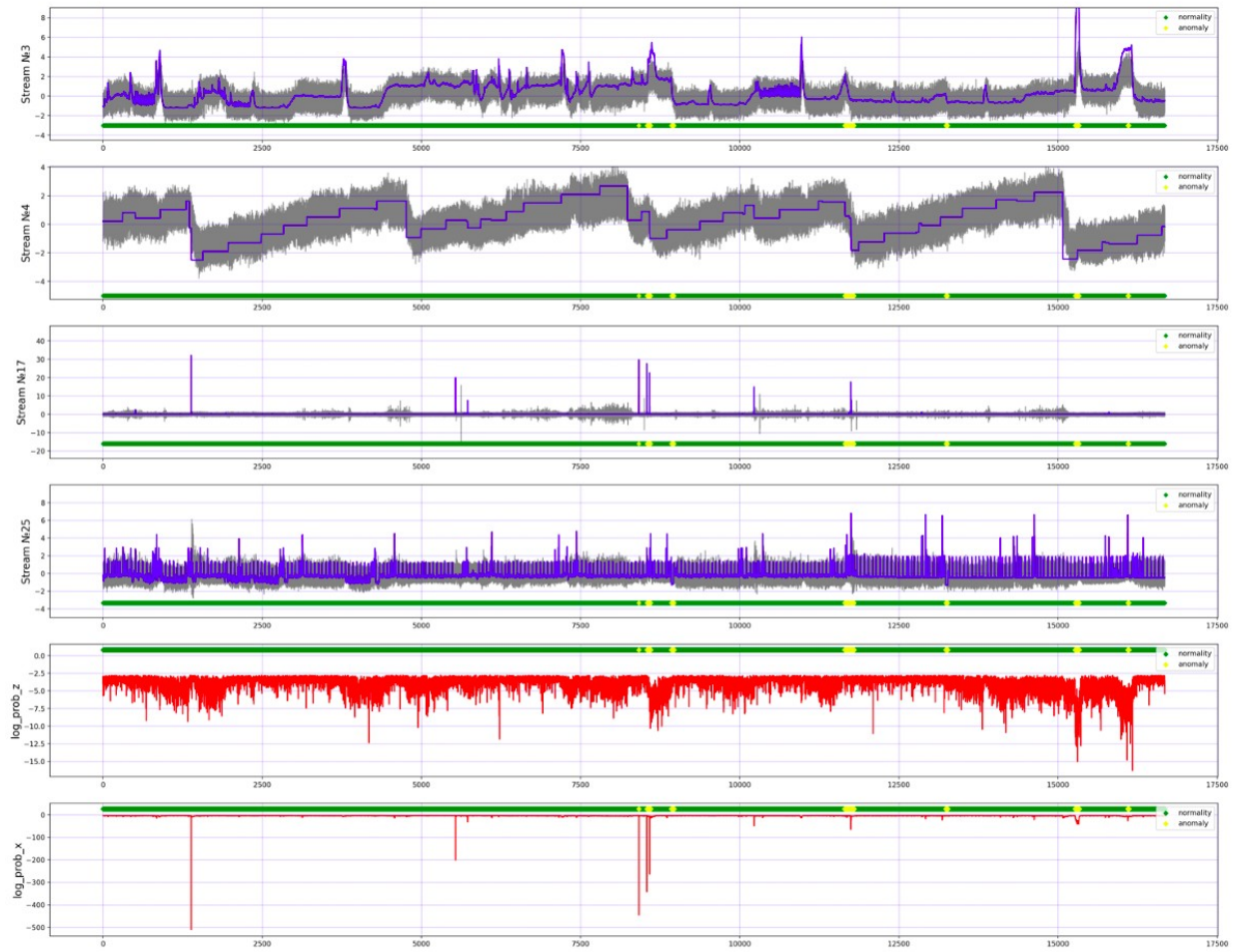

Рисунок 5.1 – Результати роботи VAE

Безпосередньо, перед тим як проводити аналіз отриманих результатів введемо наступні позначення:

- синя лінія – спостережуване значення показника;
- червона лінія – значення логарифма ймовірності;
- сіра зона – допустима множина значень;
- зелено-жовтий бар – мітка розміщення достовірних аномальних значень;

При аналізі результатів, представлених в даному форматі, варто насамперед звернути увагу на розміщення жовтих міток. Вони ідентифікують достовірні аномальні значення. Наш інтерес полягає в тому, щоб навпроти їх абсолютне значення логарифма ймовірності було якомога більшим. Дане припущення стає інтуїтивно зрозумілим коли провести логарифмування ймовірності.

Спостереження, які приймають значення за межами допустимої зони (сіра зона) відносяться з меншою ймовірністю до нормальних ніж ті що знаходять в ній. Таким чином, SPOT алгоритм з легкістю може їх ідентифікувати як аномалії. Варто звернути увагу, коли однин із показників в конкретний момент часу знаходиться на значній відстані від допустимого діапазону, а інші показники в цей же момент приймають допустимі значення, то не обов’язкво дане спостереження є аномальним. Кінцевий висновок робиться на основі багатовимірного розподілу.

Отримані значення логарифма ймовірності, передаються до SPOT алгоритму. Результати роботи для параметрів ($q = 10^{-3}$ та n_init = 1981) зображено на рисунку 5.2.

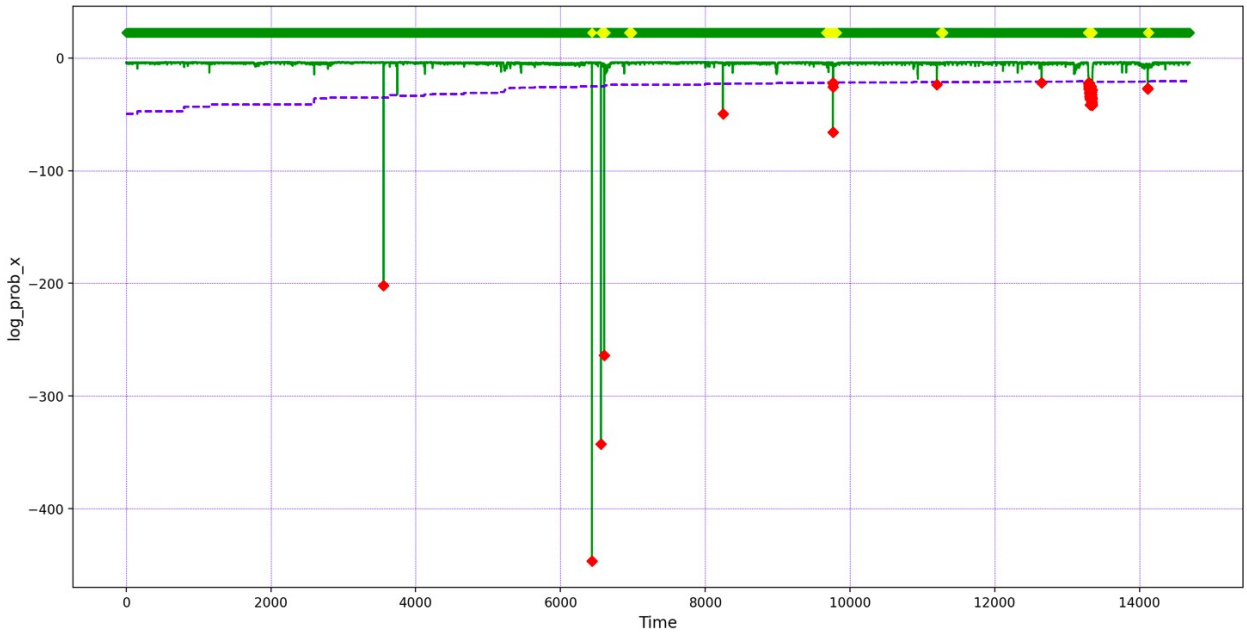

Рисунок 5.2 Результати роботи SPOT алгоритму

Матриця невідповідності та чисельні показники якості отриманих результатів наведено в таблиці 5.2 та 5.3 відповідно.

Таблиця 5.2 – Матриця невідповідності

		Дійсні класи	
		Аномалія	Не аномалія
Передбачувальні класи	Аномалія	424	263
	Не аномалія	39	13431

Таблиця 5.3 – Показники якості

Датасет	Precision	Recall	F1-score
Machine2	0.579	0.919	0.710

5.4 Загальні результати

Результати тестування системи на всіх датасетах для різник значень ризику (параметр q) наведено в таблиці 5.3.

Таблиця 5.4 – Результати тестування

Параметр ризику	Датасет	Precision	Recall	F1-score
0.01	Machine1	0.583	0.936	0.718
	Machine2	0.214	0.956	0.349
	Machine3	0.169	0.944	0.286

Продовження таблиці 5.4

Параметр ризику	Датасет	Precision	Recall	F1-score
0.001	Machine1	0.717	0.906	0.800
	Machine2	0.579	0.919	0.710
	Machine3	0.317	0.963	0.477
0.0001	Machine1	0.777	0.826	0.801
	Machine2	0.754	0.844	0.797
	Machine3	0.462	0.969	0.626

5.5 Висновки до розділу

У розділі наведено характеристики наборів даних для тестування системи. Детально проаналізовано результати роботи для одного з датасетів. Наведено значення метрик якості для різних рівнів ризику.

В найгіршому випадку відношення виявлених аномалій до загальної кількості (recall) дорівнює 82.6%. Зі зменшенням рівня ризику кількість спостережень, які були помилково ідентифіковані як аномальні (precision) зменшується. Найкращі результати було отримано на датесеті Machine1 для рівня ризику 0.0001.

ВИСНОВКИ

У рамках дипломного проектування:

а) здійснено аналіз підходів та методів ідентифікації аномалій в багатовимірних часових рядах. Серед усіх розглянутих методів обрано комбінацію із двох методів: нейронну мережу варіаційного автоенкодера та SPOT алгоритм теорії екстремальних значень;

б) на основі вибраних методів реалізовано систему ідентифікації аномалій. Розроблена система ітеративно взаємодіє з вхідними даними, виконує їх обробку та класифікує кожне спостереження як аномальне або допустиме значення;

в) Проведено тестування системи на розмічених даних. Результати роботи представлено в графічному та табличному форматі. Відношення вірно виявлених аномалій перевищує 82.6%;

ПЕРЕЛІК ПОСИЛАНЬ

1. Юрченко М.Є. Прогнозування та аналіз часових рядів: підручник / М.Є. Юрченко; М-во освіти і науки України, Чернігів. нац. технологічний ун-т. – Чернігів: ЧНТУ, 2018. – 88 с.

2. Cook A. Anomaly Detection for IoT Time-Series Data: A Survey [Text]/ A. Cook, G. Misirli, Z. Fan. // IEEE Internet of Things Journal. – 2020. – Vol. 7, №7. – pp. 6481–6494.

3. Längkvist M. A review of unsupervised feature learning and deep learning for time-series modeling [Text]/ M. Längkvist, L. Karlsson, A. Loutfi. // Pattern Recognition Letters. – 2014. – Vol. 42.– pp. 11–24.

4. Chandola V. Anomaly detection [Text]/ V. Chandola, A. Banerjee, V. Kumar. // ACM Computing Surveys. – 2009. – Vol. 41, №3. – pp. 1–58.

5. A review of novelty detection [Text]/ M.Pimentel, D. Clifton, L. Clifton, L. Tarassenko. // Signal Processing. – 2014. – Vol. 99. – pp. 215–249.

6. Angiulli F. Detecting distance-based outliers in streams of data / F. Angiulli, F. Fassetti. // Proceedings of the sixteenth ACM conference on Conference on information and knowledge management - CIKM '07, Lisbon, Portugal 06-10 November. – 2007. – pp. 811-820

7. Angiulli F. Distance-based outlier queries in data streams: the novel task and algorithms [Text]/ F. Angiulli, F. Fassetti. // Data Mining and Knowledge Discovery. – 2010. –Vol. 20, №2. – pp. 290–324.

8. Ishimtsev V. Conformal k-NN Anomaly Detector for Univariate Data Streams [Electronic resource]/ V. Ishimtsev, A. Bernstein, E. Burnaev, I. Nazarov// Proceedings of the Sixth Workshop on Conformal and Probabilistic Prediction and Applications, PMLR 60. – 2017. – Mode of access: http://proceedings.mlr.press/v60/ishimtsev17a.html. – (15.05.2021)

9. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series [Text]/ M.Munir, S. Siddiqui, A. Dengel, S. Ahmed. // IEEE Access. – 2019.

– Vol. 7. – pp. 1991–2005.

10. Unsupervised real-time anomaly detection for streaming data [Text]/ S.Ahmad, A. Lavin, S. Purdy, Z. Agha. // Neurocomputing. – 2017. – Vol. 262.– pp. 134–147.

11. Hundman K. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding [Text]/ K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom. // Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '18. – August 19–23, 2018, London, United Kingdom

12. Papadimitriou S. Streaming Pattern Discovery in Multiple Time-Series [Text]/ S. Papadimitriou J. Sun, C. Faloutsos. // In Proceedings of the 31st international conference on Very large data bases (VLDB '05). VLDB Endowment, pp. 697–708.

13. Baragona R. Outliers Detection in Multivariate Time Series by Independent Component Analysis [Text]/ R. Baragona, F. Battaglia. // Neural Computation. – 2007. – Vol. 19, №7. – pp. 1962–1984.

14. Galeano P. Outlier Detection in Multivariate Time Series by Projection Pursuit [Text]/ P. Galeano, D. Peña, R. Tsay. // Journal of the American Statistical Association. – 2006. – Vol. 101, №474. – pp. 654–669.

15. An Outlier Detection Algorithm Based on Cross-Correlation Analysis for Time Series Dataset [Text]/ H.Lu, Y. Liu, Z. Fei, C. Guan. // IEEE Access. – 2018. – Vol. 6.– pp. 53593–53610.

16. Yang H. Robust and Adaptive Online Time Series Prediction with Long Short-Term Memory [Text]/ H. Yang, Z. Pan, Q. Tao. // Computational Intelligence and Neuroscience. – Val. 2017. – 2017. – pp. 1–9.

17. Ergen T. Unsupervised Anomaly Detection With LSTM Neural Networks [Text]/ T. Ergen, S. Kozat. // IEEE Transactions on Neural Networks and Learning Systems. – 2020. – Vol. 31, №8. – pp. 3127–3141.

18. Shaojie Bai. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling [Electronic resource]. – 2018. – Mode of access: https://arxiv.org/abs/1803.01271. – (18.05 .2021)

19. Convolutional LSTM network: A machine learning approach for precipitation

nowcasting[Text] / X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, W. C. Woo. // Advances in Neural Information Processing Systems. – 2015. – Vol. 2015. – pp.802-810.

20. Malhotra P. LSTM-based encoder-decoder for multi-sensor anomaly detection [Text]/ P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, G. Shroff. // Accepted at ICML 2016 Anomaly Detection Workshop. New York, NY, USA. – 2016.

21. Reddy K.K. Anomaly Detection and Fault Disambiguation in Large Flight Data: A Multi-modal Deep Auto-encoder Approach [Text] / K.K. Reddy, S. Sarkar, V. Venugopalan, M. Giering.// Annual Conference of the PHM Society. – 2016-- Val. 8, №1

22. Goodfellow I. Generative Adversarial Networks [Electronic resource]/ I. Goodfellow, J. Pouget-Abadie, M. Mirza //. – 2014. – Mode of access: https://arxiv.org/abs/1406.2661. – (19.05 .2021)

23. Geiger A. TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks [Electronic resource]/ A. Geiger, D. Liu, S. Alnegheimish, A. Cuesta-Infante, K. Veeramachaneni //. – 2020. – Mode of access: https://arxiv.org/abs/2009.07769. – (21.05 .2021)

24. Li D. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks [Text]/ D. Li , D. Chen, B. Jin, L. Shi, J. Goh, S.K. Ng. // In International Conference on Artificial Neural Networks. – 2019. – pp. 703-716

25. Jethava V. Easy High-Dimensional Likelihood-Free Inference [Electronic resource]. – 2017. – Mode of access: https://arxiv.org/abs/1711.11139. – (21.05 .2021)

26. Balagopalan A. ReGAN: RE[LAX|BAR|INFORCE] based Sequence Generation using GANs [Electronic resource]. – 2018. – Mode of access: https://arxiv.org/abs/1805.02788. – (21.05 .2021)

27. Niu Z. LSTM-Based VAE-GAN for Time-Series Anomaly Detection [Text]/ Z. Niu, K. Yu, X. Wu. // Sensors. – 2020. – Vol. 20, №13. – pp. 3738.

28. Su Y. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network[Text]/ Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun. // Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '19. – August 4–8, Anchorage, AK, USA. – 2019. – pp. 2828-2837

29. Zhai S. Deep Structured Energy Based Models for Anomaly Detection [Text]/ S. Zhai, Y. Cheng, W. Lu, Z. Zhang. // Proceedings of The 33rd International Conference on Machine Learning, PMLR 48. – 2016. – pp.1100-1109

30. Nasios N. Variational learning for Gaussian mixture models [Text]/ N. Nasios, A. Bors. // IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics). – 2006. – Vol. 36, №4. – pp. 849–862.

31. Kingma D.P. Auto-Encoding Variational Bayes [Text] / D.P. Kingma, M. Welling. // Conference proceedings: papers accepted to the International Conference on Learning Representations (ICLR). – 2014

32. Odaibo S. Tutorial: Deriving the Standard Variational Autoencoder (VAE) Loss Function [Electronic resource] / Stephen Odaibo. – 2019. – Mode of access: https://arxiv.org/abs/1907.08956. – (23.05 .2021)

33. Siffer A. Anomaly Detection in Streams with Extreme Value Theory [Text]/ A. Siffer, P.A. Fouque, A. Termier, C. Largouet. // Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17. – August 13–17, Halifax, NS, Canada. – 2017. – pp.1067-1075

34. Grimshaw S.D. Computing Maximum Likelihood Estimates for the Generalized Pareto Distribution [Text]/ Scott D. Grimshaw. // Technometrics. – 1993. – Val.35, №2.– pp. 185–191.

Додаток А Лістинги програм

```
# dataset.py
import numpy as np
import pandas as pd

import torch
from torch.utils.data import Dataset


class NominalDataset(Dataset):

    def __init__(self, x: np.array, seq_length: int, step: int):
        super(NominalDataset, self).__init__()

        if not isinstance(x, np.ndarray):
            raise Exception("X isn't numpy.ndarray ")

        self.n_samples = x.shape[0] - step * (seq_length - 1) - 1
        self.data = [
            (
                torch.from_numpy(x[i:i+(seq_length*step):step, :].astype(np.float32)),
                torch.from_numpy(x[i+((seq_length-1)*step)+1, :].astype(np.float32))
            )

            for i in range(self.n_samples)
        ]



    def __getitem__(self, index: int) -> tuple:
        return self.data[index][0], self.data[index][1]


    def __len__(self) -> int:
        return self.n_samples


#encoder.py
import torch
import torch.nn as nn
import torch.nn.functional as F


class Encoder(nn.Module):

    def __init__(self, input_size: int, hidden_size: int,
                 num_layers: int, seq_length: int, z_size: int):
        super(Encoder, self).__init__()

        self.z_size = z_size
        self.num_layers = num_layers
        self.hidden_size = hidden_size

        self.lstm = nn.LSTM(input_size, hidden_size, num_layers,
                                    batch_first=True, bidirectional=True)

        # mu
        self.m_layers = nn.ModuleList([

            nn.Sequential(

                nn.Linear(2*hidden_size, hidden_size),
                nn.LeakyReLU(0.1),
                nn.Linear(hidden_size, z_size)

            )

            for _ in range(seq_length)
        ])

        # logvar
        self.v_layers = nn.ModuleList([
```

```
                nn.Sequential(

                        nn.Linear(2*hidden_size, hidden_size),
                        nn.LeakyReLU(0.1),
                        nn.Linear(hidden_size, z_size)

                )

                for _ in range(seq_length)
                ])


        def forward(self, x: torch.tensor) -> torch.tensor:

                h0 = torch.zeros(2 * self.num_layers, x.size(0), self.hidden_size)
                c0 = torch.zeros(2 * self.num_layers, x.size(0), self.hidden_size)

                output, _ = self.lstm(x, (h0, c0))
                z = torch.zeros(x.size(0), x.size(1), self.z_size)

                mu = torch.zeros(x.size(0), x.size(1), self.z_size)
                logvar = torch.zeros(x.size(0), x.size(1), self.z_size)

                for indx, (m, v) in enumerate(zip(self.m_layers, self.v_layers)):
                        mu[:, indx, :]  = m(output[:, indx, :])
                        logvar[:, indx, :] = F.softplus(v(output[:, indx, :]))

                        # reparametrazation trick
                        std = torch.exp(logvar[:, indx, :] / 2)
                        #eps = torch.randn_like(std)
                        eps = 1e-2

                        z[:, indx, :] = mu[:, indx, :] + eps+std

                return z, mu, logvar


# decoder
import torch
import torch.nn as nn
import torch.nn.functional as F

class Decoder(nn.Module):

        def __init__(self, z_size:  int, hidden_size: int,
                                num_layers: int, seq_length: int, output_size: int):
                super(Decoder, self).__init__()

                self.num_layers = num_layers
                self.output_size = output_size
                self.hidden_size = hidden_size

                self.lstm = nn.LSTM(z_size, hidden_size, num_layers,
                                                batch_first=True, bidirectional=True)


                self.layers = nn.ModuleList([
                        nn.Sequential(

                                nn.Linear(hidden_size * 2, hidden_size),
                                nn.LeakyReLU(0.2)

                        )

                        for _ in range(seq_length)

                ])


                self.h2 = output_size * 2
                self.h4 = output_size * 4

                self.m = nn.Sequential(

                        nn.Flatten(),
                        nn.Linear(
                                seq_length * hidden_size,
                                seq_length * hidden_size, bias=False
```

```
                ),
                nn.LeakyReLU(0.2),
                nn.BatchNorm1d(seq_length * hidden_size),

                nn.Linear(
                    seq_length * hidden_size,
                    seq_length * self.h4
                ),
                nn.LeakyReLU(0.2),
            )


            # mu
            self.m_layers = nn.ModuleList([
                nn.Linear(self.h4, output_size) for _ in range(seq_length)
            ])

            # logvar
            self.v_layers = nn.ModuleList([
                nn.Linear(self.h4, output_size) for _ in range(seq_length)
            ])


    def forward(self, z: torch.tensor) -> torch.tensor:

            h0 = torch.zeros(2*self.num_layers, z.size(0), self.hidden_size)
            c0 = torch.zeros(2*self.num_layers, z.size(0), self.hidden_size)

            output, _ = self.lstm(z, (h0, c0))
            hidden1 = torch.zeros(z.size(0), z.size(1), self.hidden_size)

            for indx, m in enumerate(self.layers):
                hidden1[:, indx, :] = m(output[:, indx, :])

            hidden2 = self.m(hidden1)
            hidden2 = hidden2.view(z.size(0), z.size(1), -1)

            mu = torch.zeros(z.size(0), z.size(1), self.output_size)
            logvar = torch.zeros(z.size(0), z.size(1), self.output_size)


            for indx, (m, v) in enumerate(zip(self.m_layers, self.v_layers)):
                mu[:, indx, :]  = m(hidden2[:, indx, :])
                logvar[:, indx, :] = F.softplus(v(hidden2[:, indx, :]))

            return mu, logvar


## spot.py
import tqdm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import minimize


class biSPOT:


    self.proba = q
    self.data = None
    self.init_data = None
    self.n = 0
    nonedict =  {'up':None,'down':None}
    self.markers = None


    self.extreme_quantile = dict.copy(nonedict)
    self.init_threshold = dict.copy(nonedict)
    self.peaks = dict.copy(nonedict)
    self.gamma = dict.copy(nonedict)
    self.sigma = dict.copy(nonedict)
    self.Nt = {'up':0,'down':0}



  def __str__(self):
    s = ''
    s += 'Streaming Peaks-Over-Threshold Object\n'
```

```
    s += 'Detection level q = %s\n' % self.proba
    if self.data is not None:
        s += 'Data imported : Yes\n'
        s += '\t initialization  : %s values\n' % self.init_data.size
        s += '\t stream : %s values\n' % self.data.size
    else:
        s += 'Data imported : No\n'
        return s


    if self.n == 0:
        s += 'Algorithm initialized : No\n'
    else:
        s += 'Algorithm initialized : Yes\n'
        s += '\t initial threshold : %s\n' % self.init_threshold


        r = self.n-self.init_data.size
        if r > 0:
            s += 'Algorithm run : Yes\n'
            s += '\t number of observations : %s (%.2f %%)\n' % (r,100*r/self.n)
            s += '\t triggered alarms : %s (%.2f %%)\n' % (len(self.alarm),100*len(self.alarm)/self.n)
        else:
            s += '\t number of peaks  : %s\n' % self.Nt
            s += '\t upper extreme quantile : %s\n' % self.extreme_quantile['up']
            s += '\t lower extreme quantile : %s\n' % self.extreme_quantile['down']
            s += 'Algorithm run : No\n'
    return s



def fit(self,init_data,data):


    if isinstance(data,list):
        self.data = np.array(data)
    elif isinstance(data,np.ndarray):
        self.data = data
    elif isinstance(data,pd.Series):
        self.data = data.values
    else:
        print('This data format (%s) is not supported' % type(data))
        return


    if isinstance(init_data,list):
        self.init_data = np.array(init_data)
    elif isinstance(init_data,np.ndarray):
        self.init_data = init_data
    elif isinstance(init_data,pd.Series):
        self.init_data = init_data.values
    elif isinstance(init_data,int):
        self.init_data = self.data[:init_data]
        self.data = self.data[init_data:]
    elif isinstance(init_data,float) & (init_data<1) & (init_data>0):
        r = int(init_data*data.size)
        self.init_data = self.data[:r]
        self.data = self.data[r:]
    else:
        print('The initial data cannot be set')
        return



def initialize(self, verbose = True):


    n_init = self.init_data.size


    S = np.sort(self.init_data)     # we sort X to get the empirical quantile
    self.init_threshold['up'] = S[int(0.98*n_init)] # t is fixed for the whole algorithm
    self.init_threshold['down'] = S[int(0.02*n_init)] # t is fixed for the whole algorithm

    # initial peaks
    self.peaks['up'] = self.init_data[self.init_data>self.init_threshold['up']]-self.init_threshold['up']
    self.peaks['down'] = -(self.init_data[self.init_data<self.init_threshold['down']]-self.init_threshold['down'])
    self.Nt['up'] = self.peaks['up'].size
    self.Nt['down'] = self.peaks['down'].size
    self.n = n_init
```

```
    if verbose:
        print('Initial threshold : %s' % self.init_threshold)
        print('Number of peaks : %s' % self.Nt)
        print('Grimshaw maximum log-likelihood estimation ... ', end = '')

    l = {'up':None,'down':None}
    for side in ['up','down']:
        g,s,l[side] = self._grimshaw(side)
        self.extreme_quantile[side] = self._quantile(side,g,s)
        self.gamma[side] = g
        self.sigma[side] = s

    ltab = 20
    form = ('\t'+'%20s' + '%20.2f' + '%20.2f')
    if verbose:
        print('[done]')
        print('\t' + 'Parameters'.rjust(ltab) + 'Upper'.rjust(ltab) + 'Lower'.rjust(ltab))
        print('\t' + '-'*ltab*3)
        print(form % (chr(0x03B3),self.gamma['up'],self.gamma['down']))
        print(form % (chr(0x03C3),self.sigma['up'],self.sigma['down']))
        print(form % ('likelihood',l['up'],l['down']))
        print(form % ('Extreme quantile',self.extreme_quantile['up'],self.extreme_quantile['down']))
        print('\t' + '-'*ltab*3)
    return




def _rootsFinder(fun,jac,bounds,npoints,method):

    if method == 'regular':
        step = (bounds[1]-bounds[0])/(npoints+1)
        X0 = np.arange(bounds[0]+step,bounds[1],step)
    elif method == 'random':
        X0 = np.random.uniform(bounds[0],bounds[1],npoints)

    def objFun(X,f,jac):
        g = 0
        j = np.zeros(X.shape)
        i = 0
        for x in X:
            fx = f(x)
            g = g+fx**2
            j[i] = 2*fx*jac(x)
            i = i+1
        return g,j
    opt = minimize(lambda X:objFun(X,fun,jac), X0,
                   method='L-BFGS-B',
                   jac=True, bounds=[bounds]*len(X0))

    X = opt.x
    np.round(X,decimals = 5)
    return np.unique(X)



def _log_likelihood(Y,gamma,sigma):

    n = Y.size
    if gamma != 0:
        tau = gamma/sigma
        L = -n * np.log(sigma) - ( 1 + (1/gamma) ) * ( np.log(1+tau*Y) ).sum()
    else:
        L = n * ( 1 + np.log(Y.mean()) )
    return L


def _grimshaw(self,side,epsilon = 1e-8, n_points = 10):
```

```
def u(s):
    return 1 + np.log(s).mean()


def v(s):
    return np.mean(1/s)


def w(Y,t):
    s = 1+t*Y
    us = u(s)
    vs = v(s)
    return us*vs-1


def jac_w(Y,t):
    s = 1+t*Y
    us = u(s)
    vs = v(s)
    jac_us = (1/t)*(1-vs)
    jac_vs = (1/t)*(-vs+np.mean(1/s**2))
    return us*jac_vs+vs*jac_us



Ym = self.peaks[side].min()
YM = self.peaks[side].max()
Ymean = self.peaks[side].mean()



a = -1/YM
if abs(a)<2*epsilon:
    epsilon = abs(a)/n_points


a = a + epsilon
b = 2*(Ymean-Ym)/(Ymean*Ym)
c = 2*(Ymean-Ym)/(Ym**2)


# We look for possible roots
left_zeros = biSPOT._rootsFinder(lambda t: w(self.peaks[side],t),
                    lambda t: jac_w(self.peaks[side],t),
                    (a+epsilon,-epsilon),
                    n_points,'regular')


right_zeros = biSPOT._rootsFinder(lambda t: w(self.peaks[side],t),
                     lambda t: jac_w(self.peaks[side],t),
                     (b,c),
                     n_points,'regular')


# all the possible roots
zeros = np.concatenate((left_zeros,right_zeros))


# 0 is always a solution so we initialize with it
gamma_best = 0
sigma_best = Ymean
ll_best = biSPOT._log_likelihood(self.peaks[side],gamma_best,sigma_best)


# we look for better candidates
for z in zeros:
    gamma = u(1+z*self.peaks[side])-1
    sigma = gamma/z
    ll = biSPOT._log_likelihood(self.peaks[side],gamma,sigma)
    if ll>ll_best:
        gamma_best = gamma
        sigma_best = sigma
        ll_best = ll


return gamma_best,sigma_best,ll_best
```

```
def _quantile(self,side,gamma,sigma):


    if side == 'up':
        r = self.n * self.proba / self.Nt[side]
        if gamma != 0:
            return self.init_threshold['up'] + (sigma/gamma)*(pow(r,-gamma)-1)
        else:
            return self.init_threshold['up'] - sigma*np.log(r)
    elif side == 'down':
        r = self.n * self.proba / self.Nt[side]
        if gamma != 0:
            return self.init_threshold['down'] - (sigma/gamma)*(pow(r,-gamma)-1)
        else:
            return self.init_threshold['down'] + sigma*np.log(r)
    else:
        print('error : the side is not right')



def run(self, with_alarm = True):


    if (self.n>self.init_data.size):
        print('Warning : the algorithm seems to have already been run, you \
        should initialize before running again')
        return {}


    # list of the thresholds
    thup = []
    thdown = []
    alarm = []
    # Loop over the stream
    for i in range(self.data.size):


        # If the observed value exceeds the current threshold (alarm case)
        if self.data[i]>self.extreme_quantile['up'] :
            # if we want to alarm, we put it in the alarm list
            if with_alarm:
                alarm.append(i)
            # otherwise we add it in the peaks
            else:
                self.peaks['up'] = np.append(self.peaks['up'],self.data[i]-self.init_threshold['up'])
                self.Nt['up'] += 1
                self.n += 1
                # and we update the thresholds

                g,s,l = self._grimshaw('up')
                self.extreme_quantile['up'] = self._quantile('up',g,s)

        # case where the value exceeds the initial threshold but not the alarm ones
        elif self.data[i]>self.init_threshold['up']:
                # we add it in the peaks
                self.peaks['up'] = np.append(self.peaks['up'],self.data[i]-self.init_threshold['up'])
                self.Nt['up'] += 1
                self.n += 1
                # and we update the thresholds

                g,s,l = self._grimshaw('up')
                self.extreme_quantile['up'] = self._quantile('up',g,s)


        elif self.data[i]<self.extreme_quantile['down'] :
            # if we want to alarm, we put it in the alarm list
            if with_alarm:
                alarm.append(i)
            # otherwise we add it in the peaks
            else:
                self.peaks['down'] = np.append(self.peaks['down'],-(self.data[i]-self.init_threshold['down']))
                self.Nt['down'] += 1
                self.n += 1
                # and we update the thresholds

                g,s,l = self._grimshaw('down')
                self.extreme_quantile['down'] = self._quantile('down',g,s)

        # case where the value exceeds the initial threshold but not the alarm ones
```

```
    elif self.data[i]<self.init_threshold['down']:
            # we add it in the peaks
            self.peaks['down'] = np.append(self.peaks['down'],-(self.data[i]-self.init_threshold['down']))
            self.Nt['down'] += 1
            self.n += 1
            # and we update the thresholds

            g,s,l = self._grimshaw('down')
            self.extreme_quantile['down'] = self._quantile('down',g,s)
    else:
       self.n += 1



    thup.append(self.extreme_quantile['up']) # thresholds record
    thdown.append(self.extreme_quantile['down']) # thresholds record


    yield i

self.markers =  {'upper_thresholds' : thup,'lower_thresholds' : thdown, 'alarms': alarm}
```

Додаток Б Ілюстративний матеріал

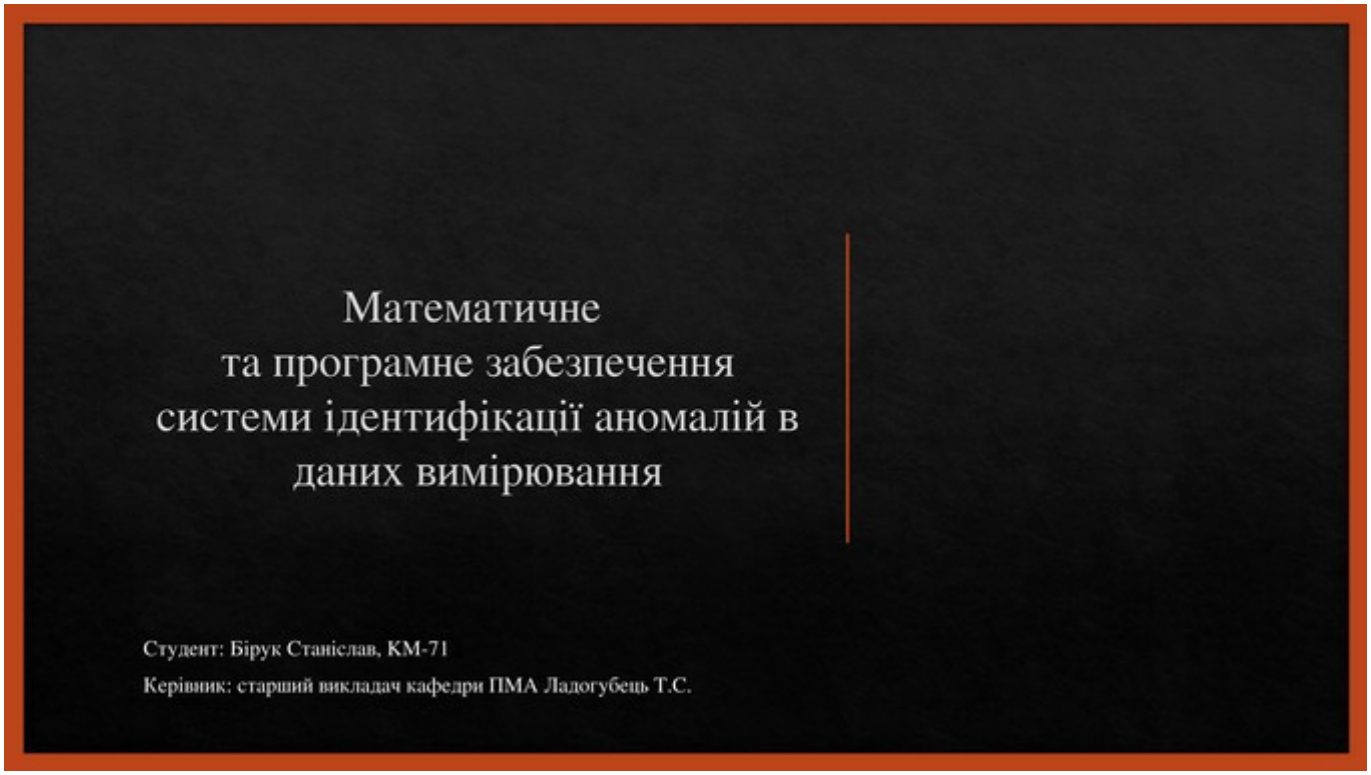

Рисунок Б.1 – Слайд 1

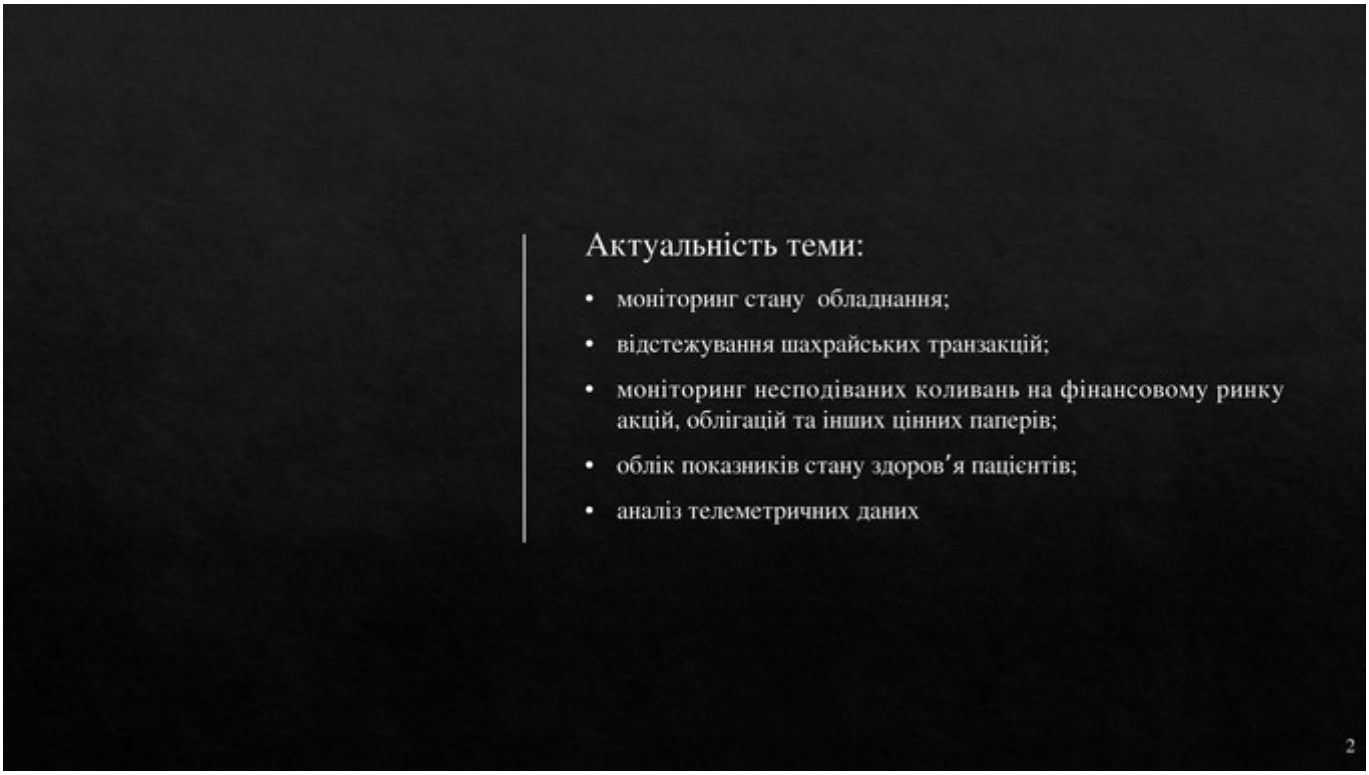

Рисунок Б.2 – Слайд 2

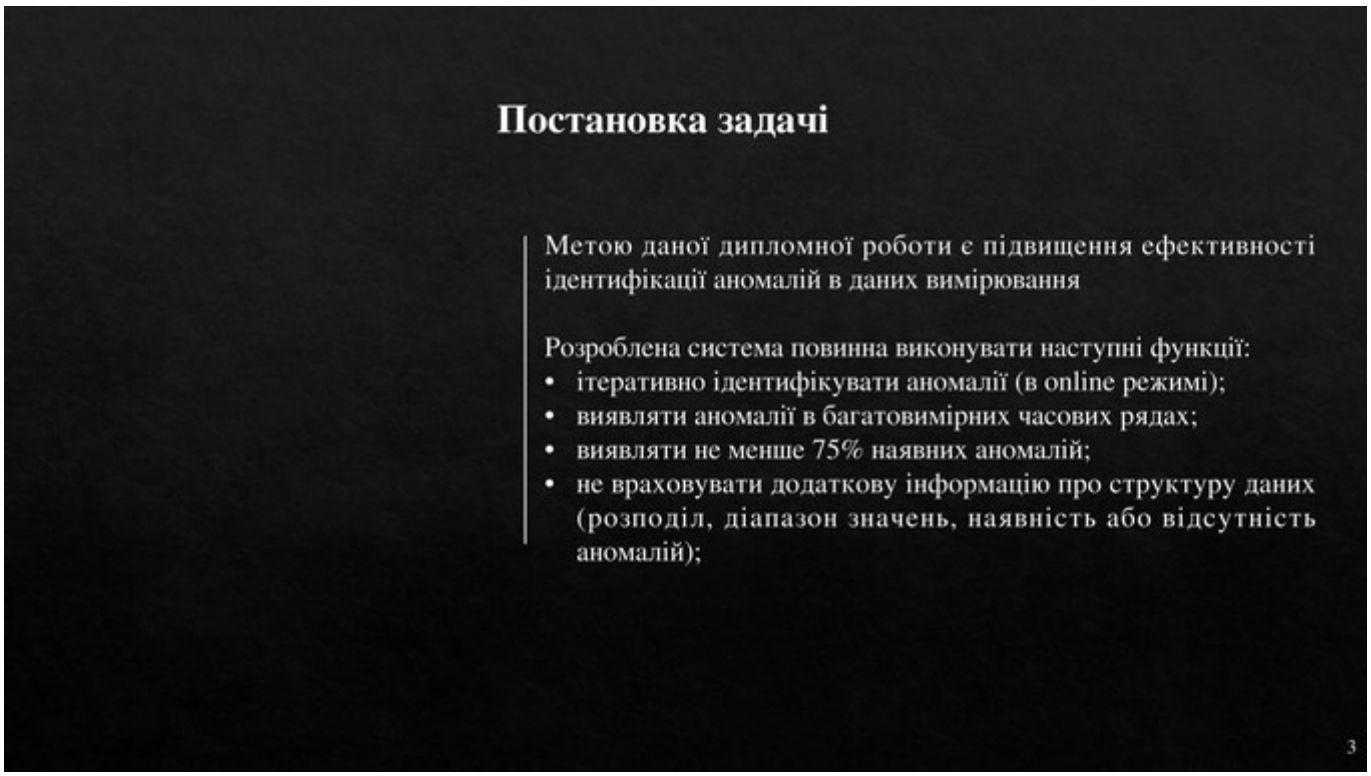

Рисунок Б.3 – Слайд 3

Рисунок Б.4 – Слайд 4

Порівняльна характеристика

Методи	Залежність між рівнями ряду	Необхідність навчання	Наявність аномалій в навчальній вибірці	Критичне значення
Подібності	Не враховує	Не потрібно	--	Задається вручну
Статистичні	Не враховує	Залежить від типу*	Залежить від типу*	Залежить від типу*
Зменшення розмірності	Враховує	Залежить від типу	Впливає на результат	Не використовується
Передбачувальні	Враховує	Потрібно	Впливає на результат	Задається вручну
Генеративні	Враховує	Потрібно	Не впливає	Задається вручну

Тип*: параметричні та непараметричні методи

5

Рисунок Б.5 – Слайд 5

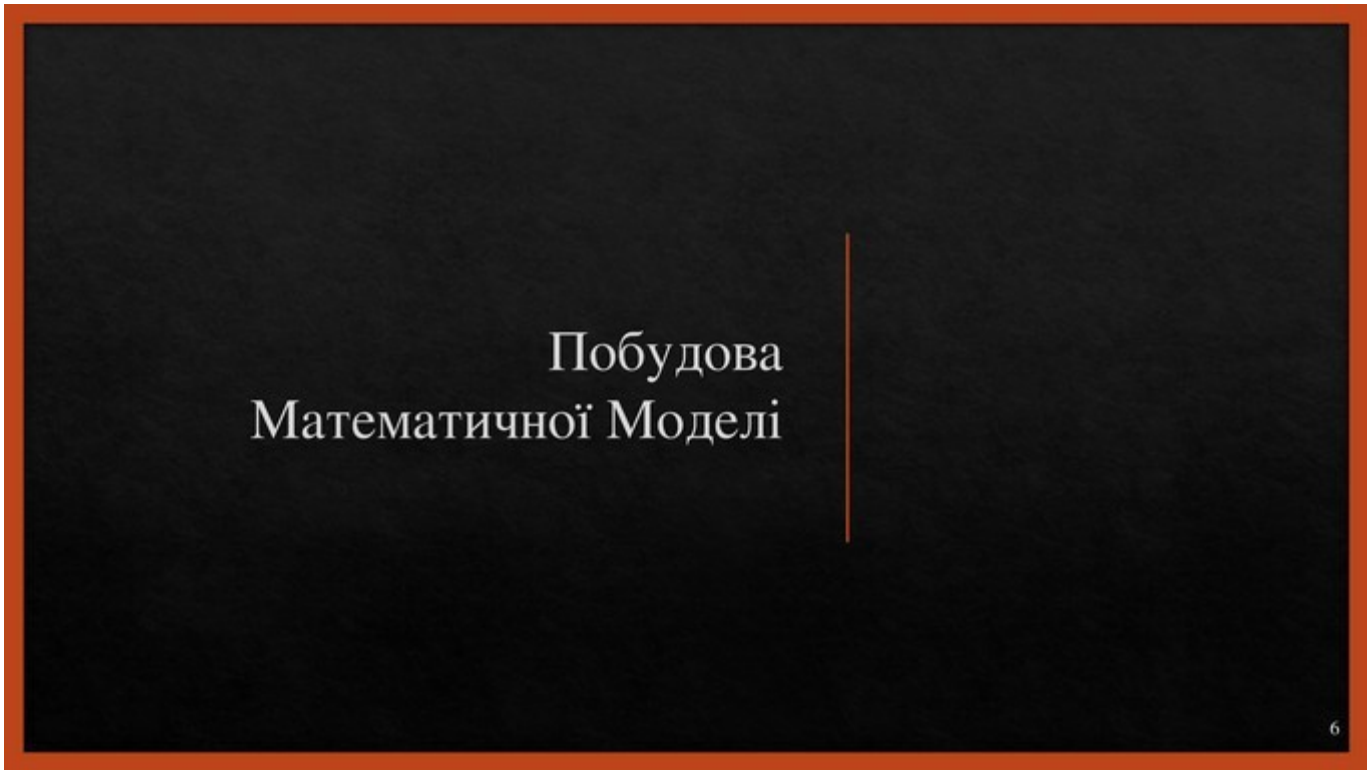

Рисунок Б.6 – Слайд 6

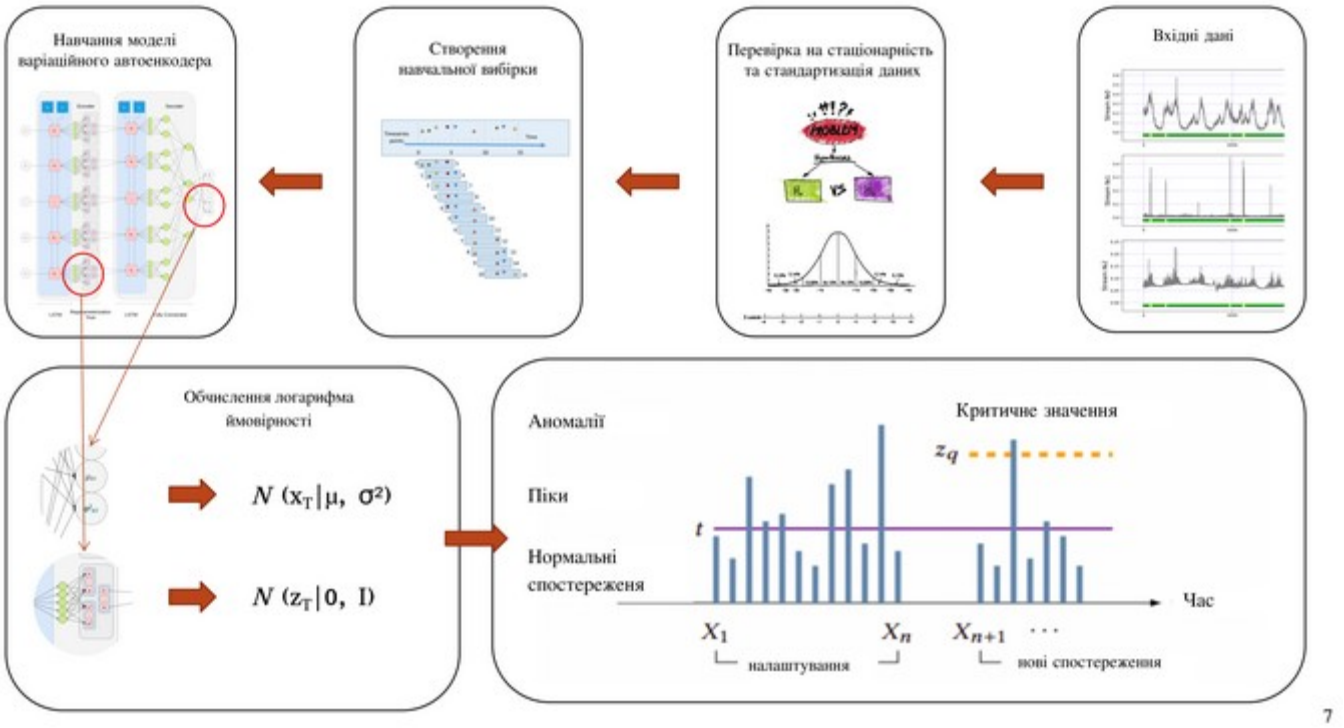

Рисунок Б.7 – Слайд 7

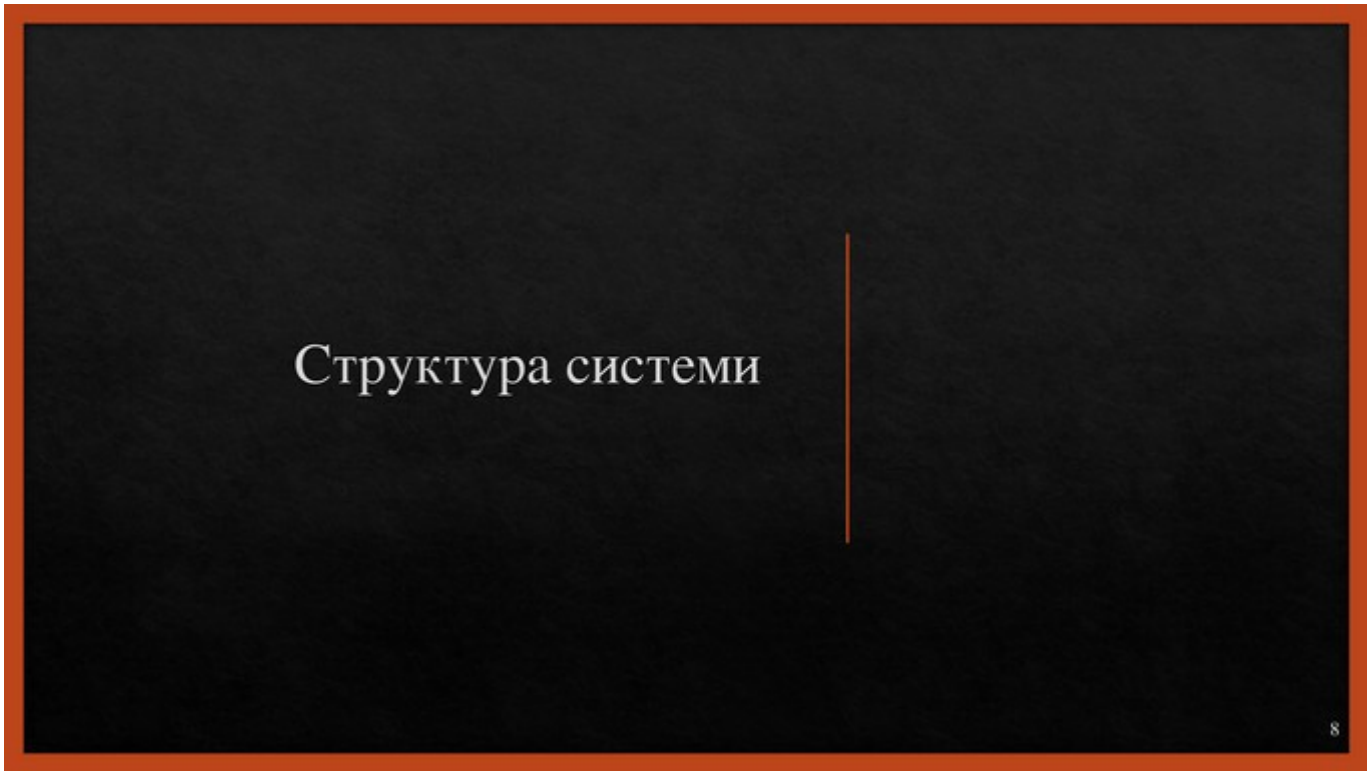

Рисунок Б.8 – Слайд 8

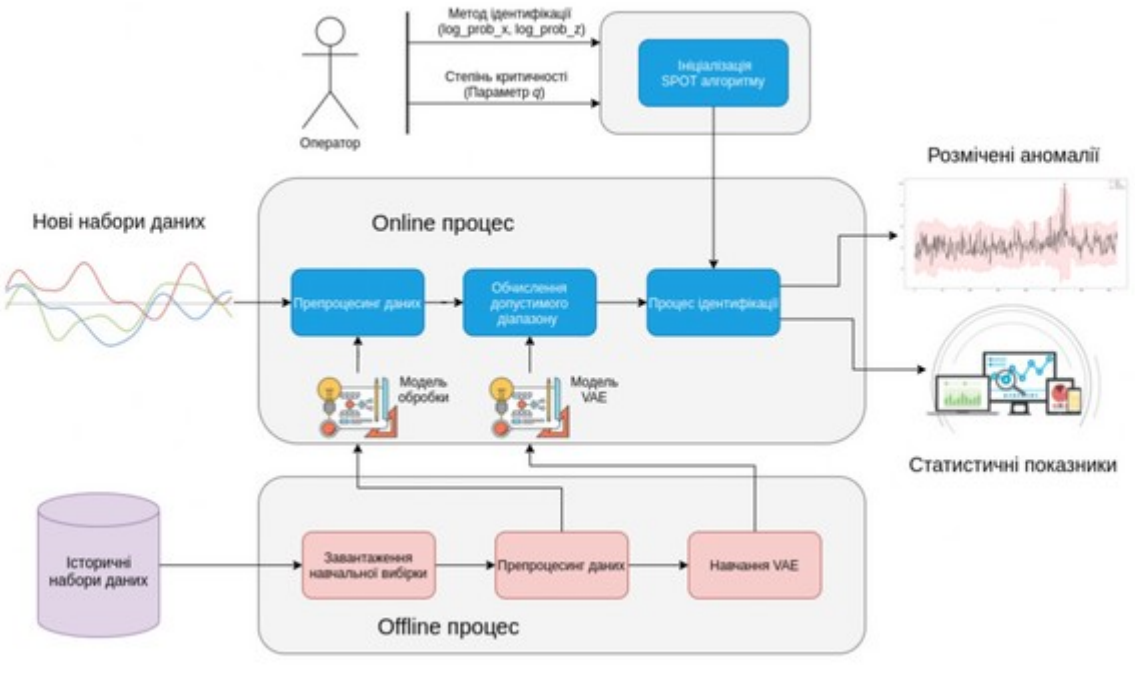

Рисунок Б.9 – Слайд 9

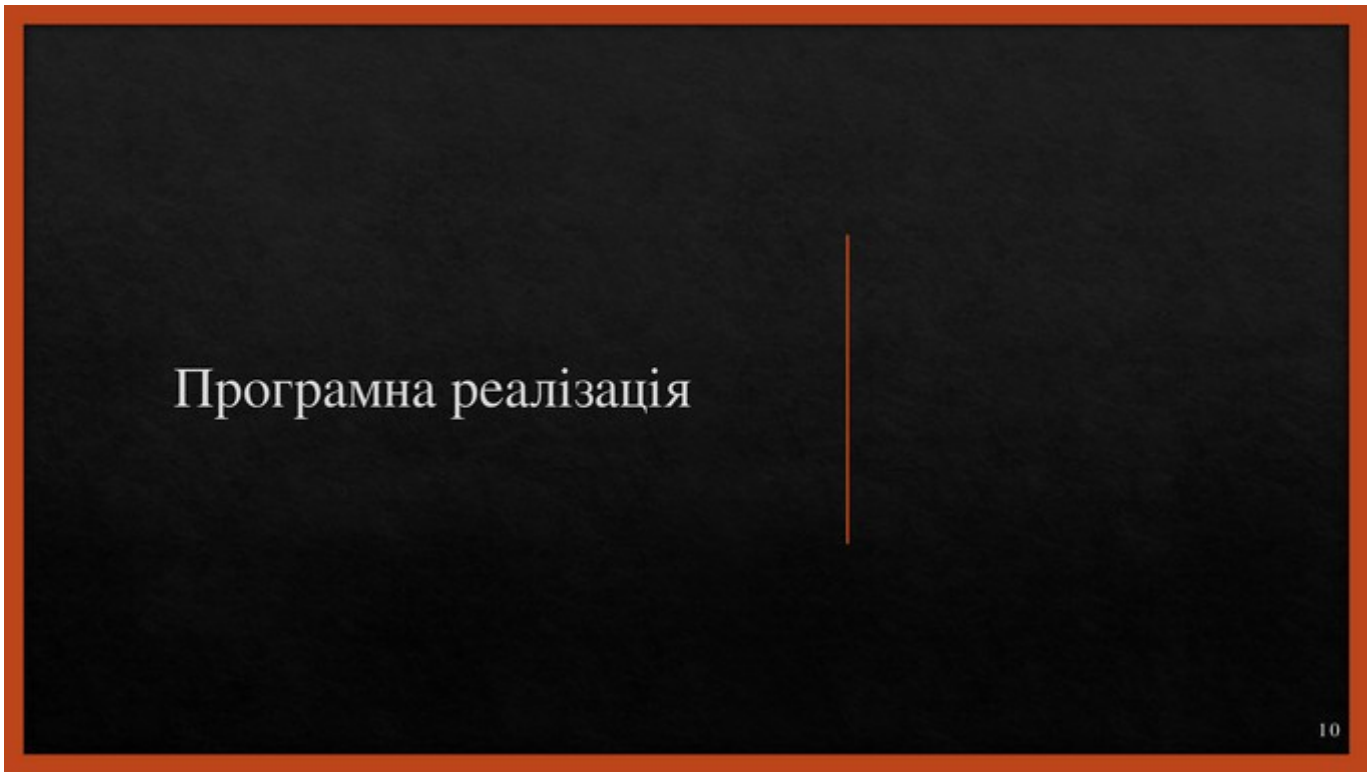

Рисунок Б.10 – Слайд 10

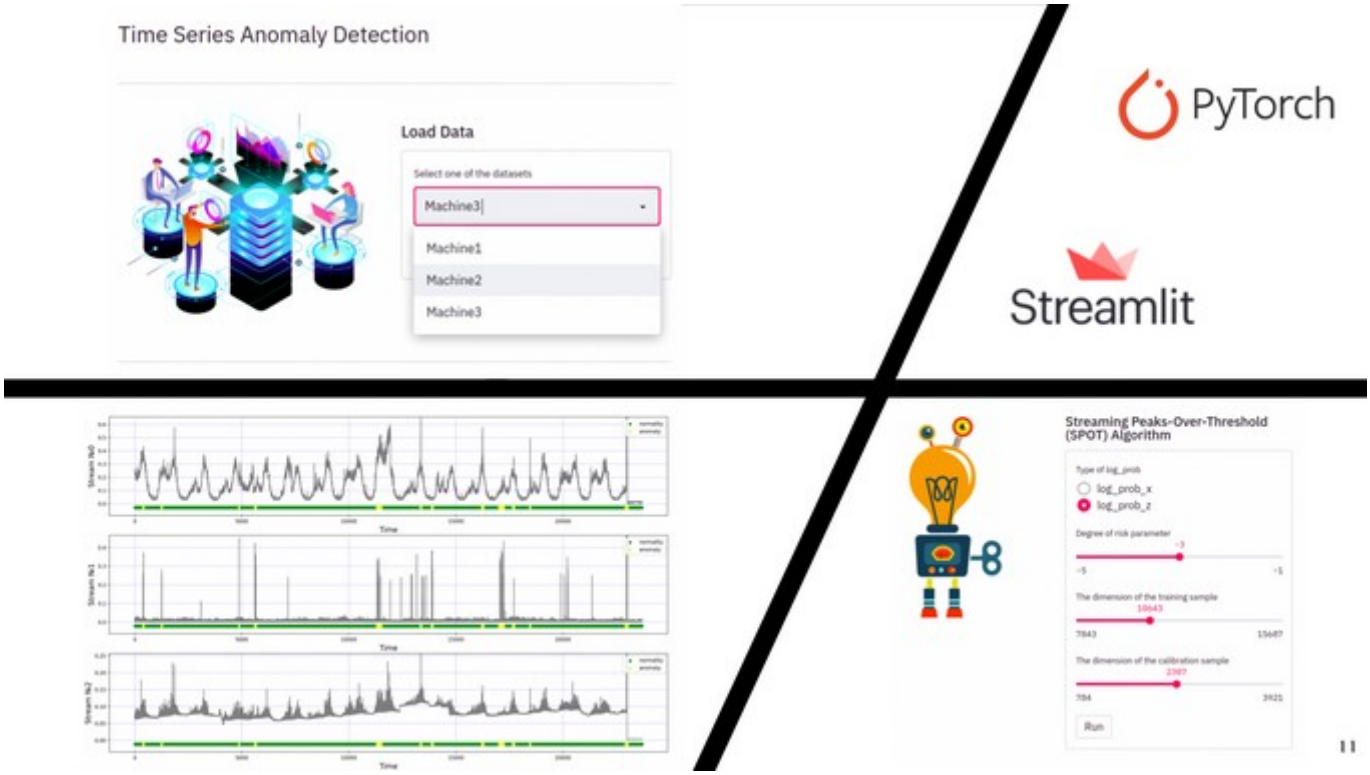

Рисунок Б.11 – Слайд 11

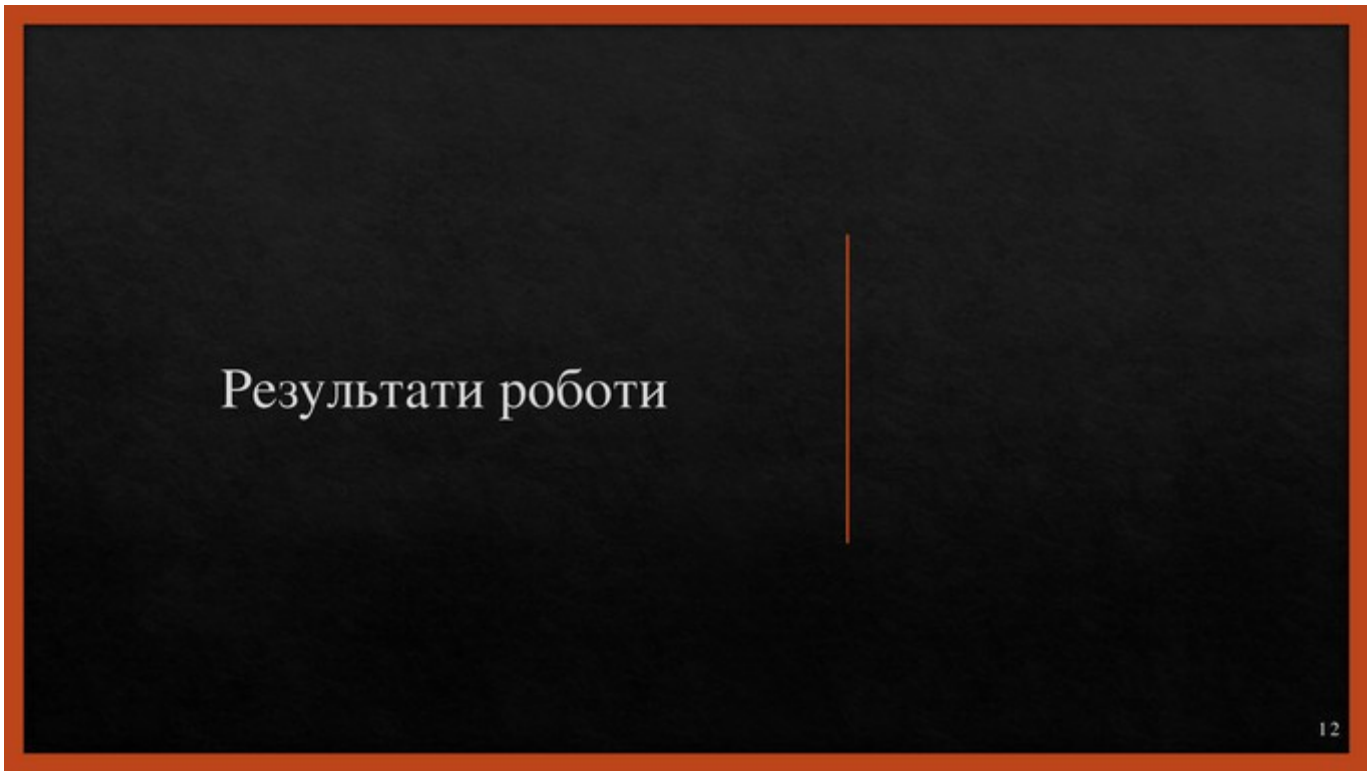

Рисунок Б.12 – Слайд 12

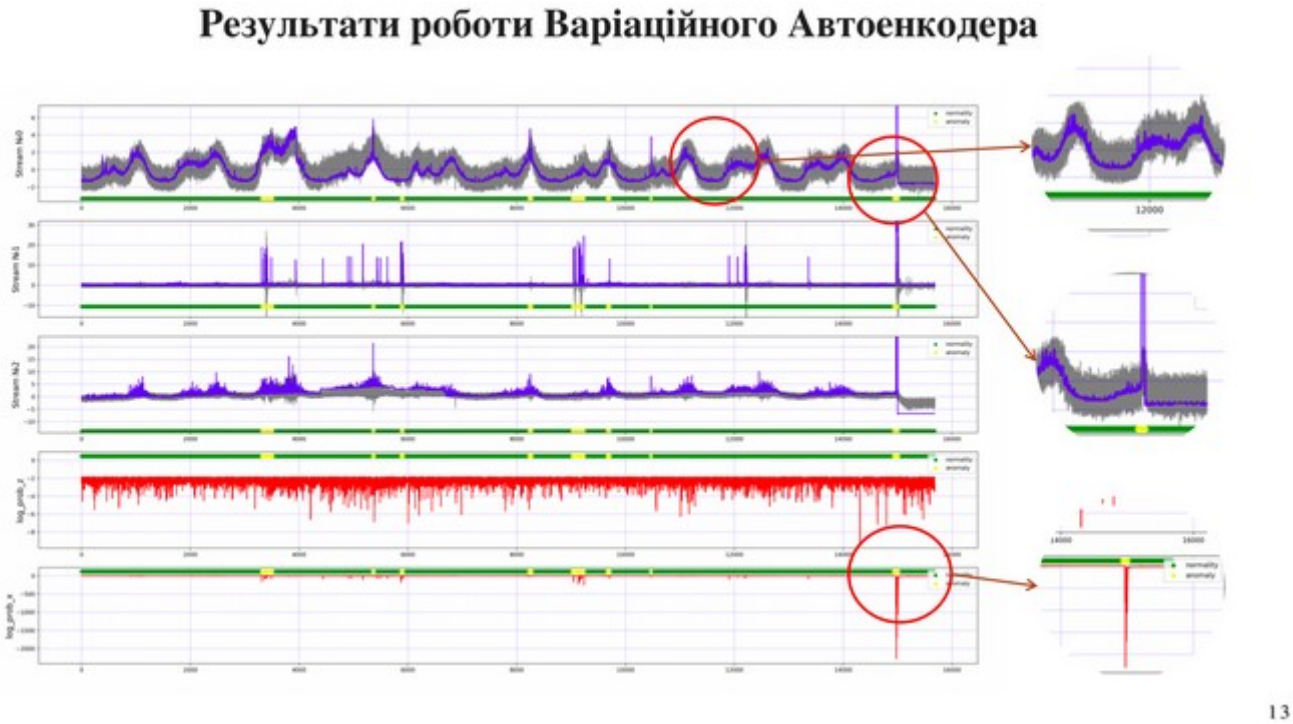

Рисунок Б.13 – Слайд 13

Рисунок Б.14 – Слайд 14

Результати роботи системи

Метрики якості

Влучність (precision)	Повнота (recall)	F1- оцінка
0.904	0.978	0.939

Матриця невідповідності

		Дійсні класи	
		Аномалія	Не аномалія
Передбачувальні класи	Аномалія	526	56
	Не аномалія	12	13647

15

Рисунок Б.15 – Слайд 15

Висновки

У рамках дипломного проектування:

- Здійснено аналіз існуючих підходів та методів виявлення аномалій;
- Спроектовано та реалізовано цільову систему;
- Перевірено працездатність системи на реальних даних;
- Відношення виявлених аномалій перевищує 82.6%

16

Рисунок Б.16 – Слайд 16

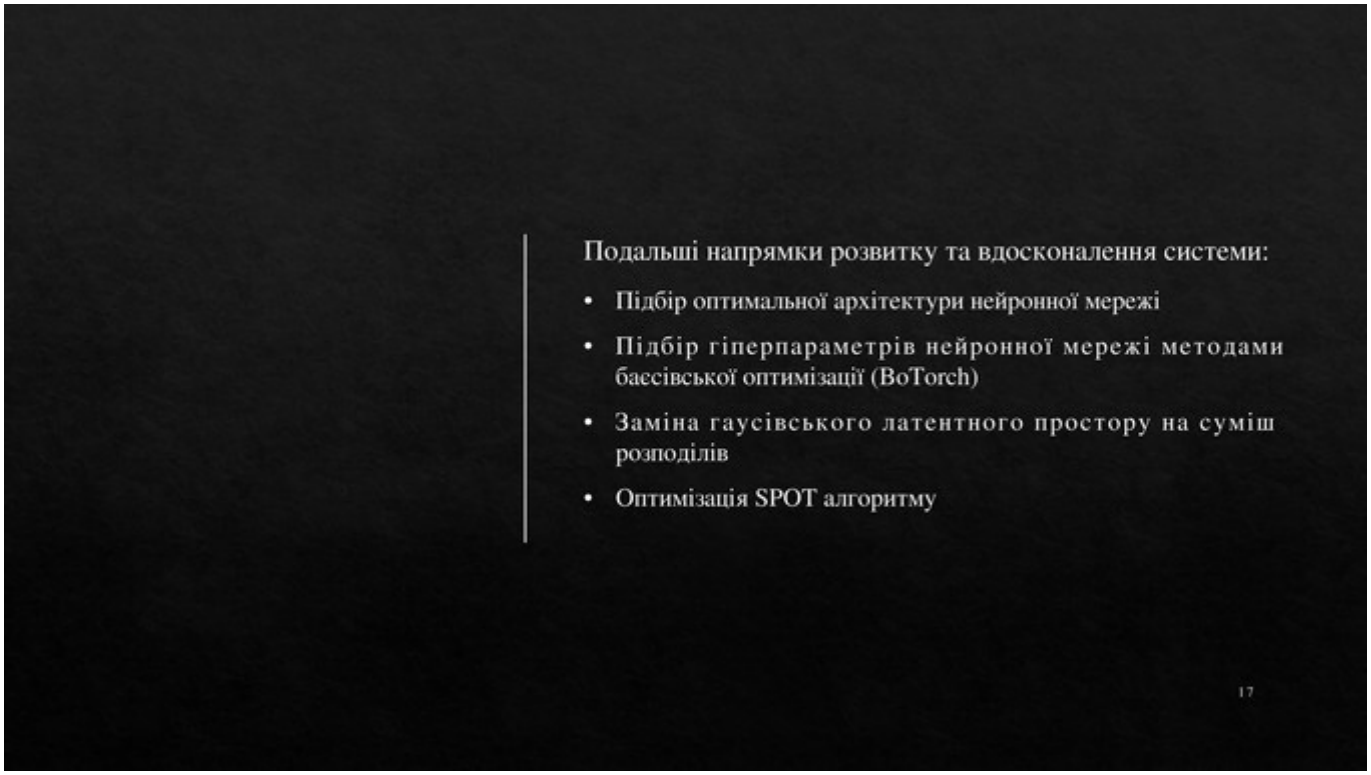

Рисунок Б.17 – Слайд 17

Рисунок Б.18 – Слайд 18