

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

Навчально-науковий інститут прикладного системного аналізу

Кафедра штучного інтелекту

До захисту допущено:

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«__» _____ 20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи і методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки»

**на тему: «Інформаційна система для аналітики виробничих процесів на
основі інтелектуального аналізу даних»**

Виконала:

студентка IV курсу, групи КІ-13,
Кутова Тетяна Анатоліївна

Керівник:

доцент кафедри штучного інтелекту, доктор філософії,
Гуськова Віра Геннадіївна

Консультант з економічного розділу:

доцент кафедри економічної кібернетики, к.е.н., доцент,
Рощина Надія Василівна

Консультант з нормоконтролю:

фахівець першої категорії кафедри штучного інтелекту, к.т.н., доцент,
Комариста Богдана Миколаївна

Рецензент:

професор кафедри математичних методів системного аналізу,
д.т.н., професор Дмитрієва Ольга Анатоліївна

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студентка _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«15» січня 2025 р.

ЗАВДАННЯ

на дипломну роботу студенту

Кутовій Тетяні Анатоліївні

1. Тема роботи «Інформаційна система для аналітики виробничих процесів на основі інтелектуального аналізу даних», керівник роботи Гуськова Віра Геннадіївна, доцент кафедри штучного інтелекту, доктор філософії, затверджені наказом по НН ІПСА від «26» травня 2025 р. № 1759-с.

2. Термін подання студентом роботи «09» червня 2025 року.

3. Вихідні дані до роботи: журнали виробничих процесів із часовими позначками, ідентифікаторами верстатів, показниками часу роботи, часу простоїв та обсягом випуску продукції; очищені та збагачені дані з новими ознаками продуктивності і співвідношення простоїв; результати аналітичних модулів у вигляді міток кластерів, відміток аномалій та прогнозованих обсягів випуску разом із показниками якості моделей; інтерактивні звіти та дашборд Streamlit із візуалізаціями і таблицями.

4. Зміст роботи – 1. Огляд предметної області та існуючих рішень; 2. Проектування архітектури та модулів інформаційної системи; 3. Реалізація та тестування інформаційної системи; 4. Функціонально-вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу: електронна презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Надія Василівна, доцент, к. е. н.		

7. Дата видачі завдання «03» лютого 2025 року.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Дослідження предметної області	21.04.2025	Виконано
2	Написання першого розділу роботи	28.04.2025	Виконано
3	Написання другого розділу роботи	05.05.2025	Виконано
4	Розроблення програмного продукту	12.05.2025	Виконано
5	Написання третього розділу роботи	19.05.2025	Виконано
6	Написання четвертого розділу роботи	26.05.2025	Виконано
7	Оформлення записки згідно з положеннями нормоконтролю	02.06.2025	Виконано
8	Підготовка презентації та доповіді	09.06.2025	Виконано

Студент

Тетяна КУТОВА

Керівник

Віра ГУСЬКОВА

РЕФЕРАТ

Дипломна робота: 107 с., 21 рис., 7 табл., 29 посилань, 2 додатки.

ІНФОРМАЦІЙНІ СИСТЕМИ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, ПРОГНОЗУВАННЯ, ВИЯВЛЕННЯ АНОМАЛІЙ, АНАЛІТИКА ВИРОБНИЧИХ ПРОЦЕСІВ, ВІЗУАЛІЗАЦІЯ ДАНИХ

У цій роботі досліджено питання розробки інформаційної системи для аналітики виробничих процесів на основі методів інтелектуального аналізу даних. Основну увагу приділено створенню програмного продукту, що дозволяє збирати, обробляти та аналізувати дані з промислових систем для виявлення аномалій і прогнозування ефективності.

Об'єкт дослідження — виробничі процеси в автоматизованих системах керування.

Предмет дослідження — програмна система для збору та аналізу логів SCADA і MES із використанням методів машинного навчання.

Мета роботи — розробити інформаційну систему аналітики виробничих процесів, що дозволяє ефективно виявляти відхилення та оцінювати ключові показники роботи підприємства в режимі реального часу.

У першому розділі подано огляд предметної області, проаналізовано типові архітектури автоматизованих систем управління, особливості побудови MES та SCADA. У другому розділі представлено математичне підґрунтя — алгоритми виявлення аномалій, кластеризації та прогнозування, що лягли в основу реалізації. У третьому розділі реалізовано програмний продукт з використанням мови Python, бібліотек для обробки даних, а також інтерактивного веб-інтерфейсу на базі Streamlit для візуалізації результатів. У четвертому розділі проведено функціонально-вартісний аналіз, зокрема оцінку трудомісткості, розрахунок витрат на розробку програмного забезпечення та визначення техніко-економічного рівня.

У результаті створено інформаційну систему, здатну аналізувати великі обсяги логів, виявляти відхилення та формувати аналітику для управлінських рішень. Рішення має потенціал до адаптації в реальних умовах та подальшої комерціалізації.

ABSTRACT

Bachelor's thesis: 107 p., 21 figures, 7 tables, 29 references, 2 appendices.

INFORMATION SYSTEMS, INTELLIGENT DATA ANALYSIS,
PREDICTION, ANOMALY DETECTION, MANUFACTURING PROCESS
ANALYTICS, DATA VISUALIZATION

This thesis explores the development of an information system for manufacturing process analytics based on data mining methods. The focus is placed on the creation of a software product that enables the collection, processing, and analysis of industrial data for anomaly detection and performance forecasting.

Object of research — manufacturing processes in automated control systems.

Subject of research — a software system for collecting and analyzing SCADA and MES logs using machine learning methods.

Purpose of the work — to develop an analytical information system for manufacturing processes capable of effectively detecting deviations and assessing key performance indicators in real time.

The first chapter presents an overview of the subject area, analyzes typical architectures of automated control systems, and examines the structure of MES and SCADA systems. The second chapter introduces the mathematical foundations — algorithms for anomaly detection, clustering, and forecasting — that form the basis of the implementation. The third chapter describes the developed software product built with Python, data processing libraries, and an interactive web interface using Streamlit for visualization. The fourth chapter provides a functional-cost analysis, including an estimation of labor input, development costs, and the calculation of the technical and economic level.

As a result, an information system has been developed that can process large volumes of logs, detect deviations, and generate actionable analytics for decision-making. The proposed solution demonstrates potential for adaptation in real industrial environments and further commercial deployment.

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ	10
1.1 Поняття інформаційних систем аналітики виробництва	10
1.2 Огляд галузевих платформ	12
1.2.1 SCADA.....	13
1.2.2 MES	14
1.2.3 BI-системи.....	16
1.3 Методи Data Mining у виробничому процесі.....	18
1.3.1 Кластеризація.....	19
1.3.2 Аномалії.....	21
1.3.3 Прогноз.....	23
1.4 Проблеми відсутності ефективної аналітики	24
1.5 Актуальність роботи	25
1.6 Постановка задачі.....	26
Висновки до розділу 1	28
РОЗДІЛ 2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА МОДУЛІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	30
2.1 Вимоги до системи	30
2.2 Конфігураційний модуль.....	32
2.3 Проєктування бази даних та ER-діаграма	33
2.4 Генерація та завантаження даних.....	35
2.5 Попередня обробка та ознакоутворення.....	37
2.6 Математична постановка методів інтелектуального аналізу	38
2.6.1 Кластеризація (K-means)	39
2.6.2 Виявлення аномалій (Isolation Forest)	40
2.6.3 Прогнозування (Random Forest Regressor).....	42
2.7 Реалізація аналітичного модуля	43

2.8 Модуль візуалізації	44
2.9 Модуль звітування	46
2.10 Інтерфейс користувача	47
Висновки до розділу 2	48
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	50
3.1 Опис середовища розробки	50
3.1.1 Мова програмування Python та її інтерпретатор	50
3.1.2 Віртуальне середовище	51
3.1.3 Інтегроване середовище розробки	52
3.1.4 Менеджмент залежностей	53
3.2 Експериментальне тестування	54
3.3 Аналіз результатів тестування	58
Висновки до розділу 3	59
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО	
ПРОДУКТУ	61
4.1 Постановка задачі проектування	62
4.2 Обґрунтування функцій програмного продукту	63
4.3 Обґрунтування системи параметрів програмного продукту	66
4.4 Аналіз експертного оцінювання параметрів	69
4.5 Аналіз рівня якості варіантів реалізації функцій	74
4.6 Економічний аналіз варіантів розробки ПП	75
4.7 Вибір кращого варіанту ПП техніко-економічного рівня	79
Висновки до розділу 4	80
ВИСНОВКИ	81
ПЕРЕЛІК ПОСИЛАНЬ	83
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ	86
ДОДАТОК Б ЕЛЕКТРОННА ПРЕЗЕНТАЦІЯ	99

ВСТУП

У сучасних¹ умовах розвитку промисловості та цифровізації обсяги даних, що генеруються виробничими підприємствами, зростають експоненційно. Системи автоматизації, зокрема SCADA, MES та ERP, накопичують величезні масиви первинної інформації, що вимагає ефективних підходів для перетворення «сирих» логів у цінні аналітичні інсайти. Інформаційні системи аналітики виробничих процесів пропонують багаторівневий конвеєр, що охоплює збір, очищення, нормалізацію, ознакоутворення та візуалізацію даних, проте в умовах зростання обсягів інформації та складності завдань зазвичай використовуються лише стандартні BI-інструменти без інтеграції сучасних методів машинного навчання.

Машинний аналіз великих даних відкриває нові можливості для оптимізації виробничих процесів: алгоритми кластеризації дають змогу виділяти характерні режими роботи обладнання, методи виявлення аномалій сприяють своєчасному ідентифікуванню відхилень, а моделі прогнозування забезпечують оцінку майбутніх показників продуктивності. Незважаючи на наявний досвід у застосуванні окремих методів, відсутній уніфікований модульний підхід до побудови сквозного аналітичного конвеєра із гнучкою інтеграцією ETL-процесів, алгоритмів машинного навчання та інтерактивного інтерфейсу користувача.

Тому в рамках даної дипломної роботи обрано тему «Модульна інформаційна система аналітики виробничих процесів на основі методів інтелектуального аналізу даних». Метою дослідження є аналіз існуючих підходів, проєктування архітектури системи, реалізація прототипу з

¹ Тут і нижче використано такий інструмент штучного інтелекту як чат-бот з генеративним штучним інтелектом ChatGPT, виключно для корегування та редагування тексту, створеного автором цієї дипломної роботи, на основі автоматизованої перевірки граматики, структури та стилю, що відповідає Політиці використання штучного інтелекту для академічної діяльності в КПІ ім. Ігоря Сікорського (протокол №11 Вченої ради КПІ ім. Ігоря Сікорського від 11 грудня 2023 р.).

використанням алгоритмів Isolation Forest для детекції аномалій, Random Forest та рекурентних нейронних мереж для прогнозування ключових показників, а також розробка засобів інтерактивної візуалізації й автоматизованого формування регулярних аналітичних звітів.

РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Поняття інформаційних систем аналітики виробництва

Інформаційна система аналітики виробництва — це інтегрована сукупність апаратно-програмних засобів, призначених для автоматизованого збору, збереження, обробки та візуалізації даних, отриманих із різних джерел у виробничому середовищі. Основною метою такої системи є своєчасне та об'єктивне підтримання управлінських рішень на всіх рівнях підприємства: від цехових операторів до вищого керівництва.

Архітектура інформаційної системи аналітики виробництва характеризується багаторівневою структурою, що забезпечує наскрізний потік даних від устаткування до користувача [1]. На початковому рівні знаходяться джерела даних — традиційні MES- та ERP-системи, SCADA-контролери, IoT-датчики, а також сторонні Excel- чи CSV-звітності, які безперервно генерують «сирі» події та вимірювання технологічних параметрів (температура, тиск, продуктивність тощо). Отримані дані надходять у ETL-конвеєр (Extract–Transform–Load), де здійснюється їхня екстракція з гетерогенних систем, очищення від пропусків і шуму, агрегація та нормалізація із подальшим обчисленням промислових KPI (наприклад, OEE, коефіцієнта завантаження). Після трансформації дані завантажуються у сховище, що може бути реалізоване як у вигляді реляційних баз (PostgreSQL, MySQL) для структурованих таблиць, так і у формі хмарних Data Lake (Azure Data Lake, Amazon S3) для зберігання великих обсягів історичної інформації; до того ж, для забезпечення обробки в реальному часі застосовуються стрімінгові платформи на кшталт Apache Kafka.

Аналітичний шар реалізує BI-дашборди (Power BI, Tableau) та сервіси інтелектуального аналізу, які використовують методи Data Mining і машинного навчання (кластеризація, виявлення аномалій, прогнозування) для виявлення прихованих закономірностей, передбачення відмов обладнання й оптимізації

виробничих процесів. Нарешті, результати обчислень та візуалізації подаються кінцевому користувачу через веб-інтерфейси й інтерактивні звіти, а за необхідності — також через API для інтеграції з іншими корпоративними додатками або у вигляді SMS/Email-сповіщень про критичні події.

Потік даних у такій системі має ланцюг: від збору «сирих» вимірювань на рівні обладнання → через ETL-конвеєр із застосуванням технологій стрімінгу та пакетної обробки → до сховища даних, де виконуються довгострокові аналітичні обчислення → в аналітичних модулях генеруються графіки, таблиці та прогнози, які у зручному форматі подаються користувачам.

Впровадження інформаційної системи аналітики виробництва дає змогу значно скоротити простої устаткування завдяки своєчасному виявленню аномалій у режимі реального часу, підвищити якість продукції через постійний контроль ключових параметрів та автоматичні сповіщення про відхилення, оптимізувати витрати енергії й матеріалів завдяки прогнозуванню попиту на виробничі потужності, а також покращити прозорість операцій завдяки доступності інтерактивних звітів і дашбордів із показниками KPI.

На практиці існують успішні кейси інтеграції таких систем: наприклад, «Smart Factory» рішення на базі Microsoft Azure IoT Hub для більш ніж 50 машинобудівних підприємств у Європі, яке показало зниження простоїв на 18 % та скорочення енергоспоживання на 12 % протягом першого року експлуатації [1]. Реалізація подібних проєктів підтверджує, що інвестиції в інформаційну систему аналітики виробництва виправдовують себе завдяки швидкому поверненню капіталу й довгостроковому підвищенню конкурентоспроможності підприємства.

1.2 Огляд галузевих платформ

Існує три основних класи галузевих платформ, що становлять каркас сучасних інформаційних систем аналітики виробництва [2]. Перш за все, SCADA-системи (Supervisory Control and Data Acquisition) забезпечують безперервний моніторинг і управління технологічними процесами на рівні датчиків і контролерів, реалізуючи збір телеметрії в реальному часі, архівацію подій і аварійні сповіщення. Саме завдяки компоненту SCADA оператори мають змогу здійснювати дистанційне втручання в налаштування устаткування та оперативно реагувати на відхилення технологічних параметрів.

Другим рівнем є MES-платформи (Manufacturing Execution Systems), які відповідають за деталізоване управління виробничими операціями: від реєстрації виконаних робіт і обліку матеріалів до контролю якості на кожній ділянці та інтеграції з ERP-системами для узгодження виробничого планування з фінансовими і логістичними процесами. MES-рішення дають можливість відстежувати «пісок на вході» та «кінець труби» виробництва, виступаючи проміжним рівнем між польовою автоматизацією та стратегічним управлінням.

BI-платформи (Business Intelligence) формують аналітичний шар, що забезпечує гнучкий доступ до даних і візуалізацію ключових показників ефективності (KPI) через інтерактивні дашборди та звіти. Інструменти на кшталт Power BI, Tableau або Qlik реалізують складні запити мовою SQL чи за допомогою візуального конструювання аналітики, дають можливість керівникам аналізувати історичні тренди, будувати сценарії «що-якщо» і приймати обґрунтовані стратегічні рішення.

1.2.1 SCADA

У промисловому середовищі SCADA (Supervisory Control and Data Acquisition) виступає ключовою системою для централізованого збору, візуалізації та керування технологічними процесами. SCADA об'єднує низку поверхів архітектури, що забезпечують безперервний моніторинг і оперативний вплив на обладнання та підсистеми (рис.1.1) [3].

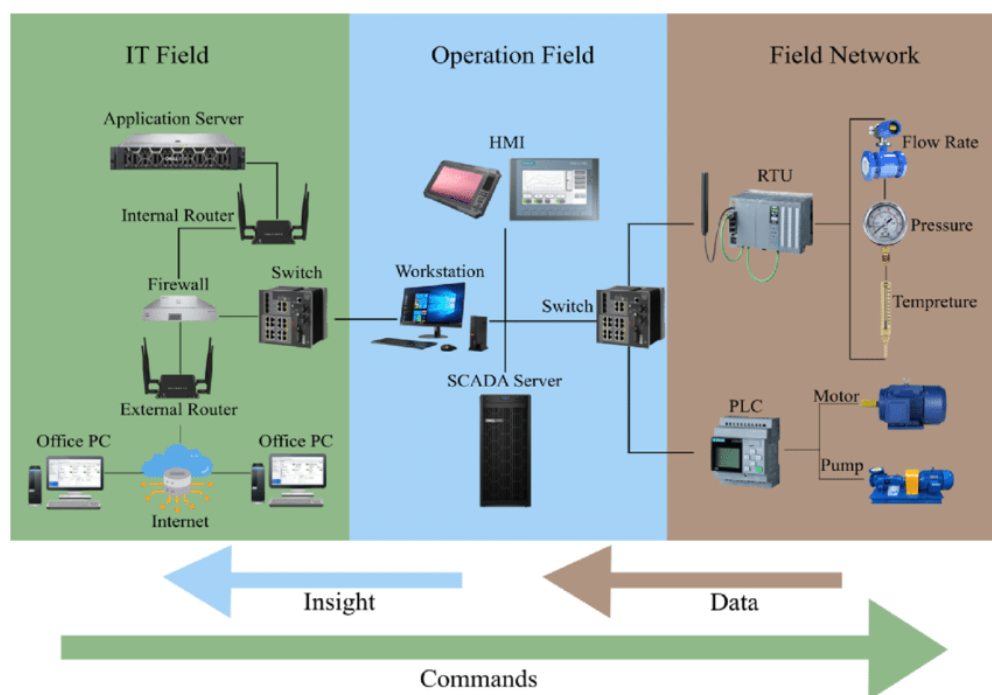


Рисунок 1.1 – Багаторівнева архітектура SCADA²

На найнижчому рівні розташовані польові пристрої – датчики, виконавчі механізми, контролери (RTU, PLC), які безпосередньо взаємодіють із фізичними елементами виробництва. Сполучення цих пристроїв із

² Джерело зображення: <https://www.researchgate.net/profile/Mohamed-Nabaebi/publication/361875022/figure/fig1/AS:1184464260595712@1659409694354/SCADA-layered-architecture.png>

центральним вузлом здійснюється через різні канали зв'язку: проводові (Ethernet, оптоволоконні лінії) та безпроводні (GSM, радіо, Wi-Fi) [4].

Наступний рівень – рівень контролю та оперативного управління – включає SCADA-сервери та панелі HMI (Human–Machine Interface). Сервер обробляє вхідні потоки даних, виконує агрегацію, зберігає історію змін у хронологічну базу (historian) та генерує команди для польових пристроїв. HMI забезпечує оператору візуалізацію ключових показників у реальному часі, дає змогу вручну втручатися в процес та налаштовувати параметри [4].

На рівні корпоративної мережі відбувається інтеграція SCADA з іншими інформаційними системами підприємства – MES, ERP, BI-хабами. Завдяки цьому можлива передача агрегованих даних до аналітичних сервісів і отримання керівних команд від вищого рівня управління, що забезпечує безперервний потік інформації від виробничого майданчика до корпоративного порталу [3].

На рівні представлення користувача реалізовані веб-інтерфейси, мобільні додатки та API, які надають доступ до SCADA-даних віддаленим користувачам, підтримують формування інтерактивних дашбордів та відправку SMS/Email-сповіщень про критичні події.

1.2.2 MES

Manufacturing Execution System (MES) виконує роль «мосту» між польовим рівнем автоматизації та корпоративними рішеннями, забезпечуючи безперервне управління та моніторинг виробничих операцій в реальному часі. MES котирує робочі завдання, планує операції на обладнанні, відстежує виконання виробничих рецептів і контролює матеріальні запаси у цеху. Він також відповідає за збір детальної інформації про продуктивність

устаткування, якість продукції та простій, що уможливлює аналіз ефективності виробничих ліній.

Важливою складовою MES є модуль маршрутизації (dispatching), який розподіляє завдання між працівниками і машинами, а також функціонал документ-контролю (document control) для управління технологічною документацією та інструкціями. Історіографія подій і даних зберігається в єдиній базі, що дає змогу проводити ретроспективний аналіз і виявляти тренди на рівні цеху та підприємства. У поєднанні з ERP та BI-платформами MES формує наскрізний потік даних для прийняття оперативних і стратегічних рішень (рис. 1.2) [5].

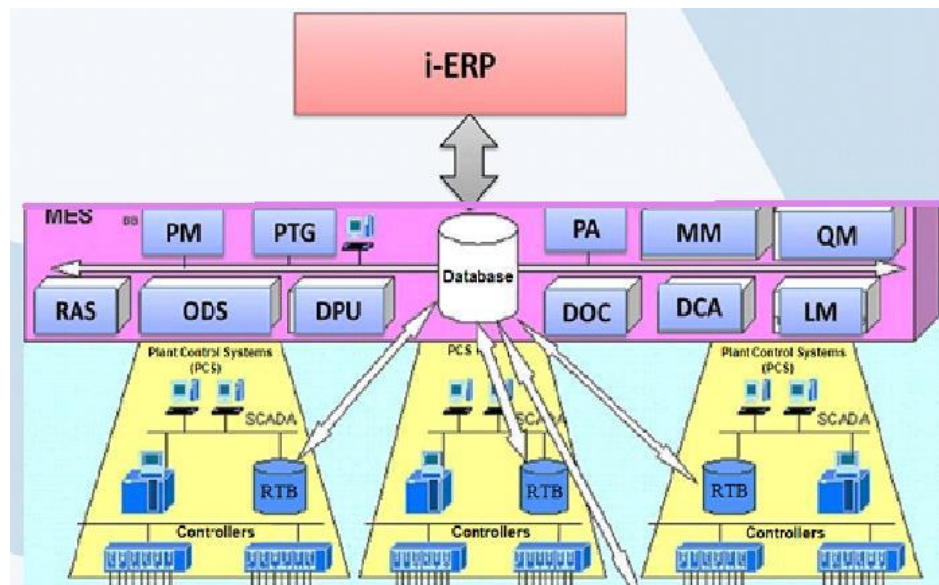


Рисунок 1.2 – Структура MES³

³ Джерело зображення: <https://www.researchgate.net/profile/Ivana-Klackova-2/publication/366272697/figure/fig3/AS:11431281108267780@1671447596464/Structure-of-MES-Manufacturing-Execution-System.png>

1.2.3 BI-системи

У сучасних інформаційних системах аналітики виробництва BI-платформи (Business Intelligence) виконують роль інтелектуального шару, призначеного для глибокого аналізу, візуалізації та розповсюдження знань на підприємстві [6]. Головною задачею BI є інтеграція даних із різномірних джерел (SCADA, MES, ERP, Data Warehouse) у єдину багатовимірну модель, що дає можливість користувачам швидко будувати звіти, аналізувати тренди та здійснювати ad-hoc запити для прийняття рішень у реальному часі.

Архітектурно BI-система включає два ключові компоненти. Перший — сховище даних або Data Mart, оптимізоване під запити аналітиків; воно може мати багатовимірну структуру (OLAP-куби), що забезпечує швидке обчислення агрегацій і зведених таблиць. Другий компонент — фронтенд для користувача: інтерактивні дашборди, графіки та візуалізації, реалізовані в середовищах Power BI, Tableau чи Qlik.

У рамках BI-платформи забезпечуються механізми розгортання бізнес-логіки: розрахунок KPI (OEE, PPM), порогова аналітика (Alerting), побудова сценаріїв «що-якщо» (what-if analysis) та прогнозування на основі історичних даних. За рахунок drag-and-drop інтерфейсу користувач здатен створювати власні дашборди, задавати фільтри за різними параметрами і виводити на екран гнучкі багатомірні зведення без залучення IT-спеціалістів.

BI-системи (рис. 1.3) часто доповнюються модулем управління правами доступу й версіонування звітів, що гарантує збереження цілісності інформації та контроль достовірності даних. Інтеграція BI із корпоративними порталами й мобільними додатками дає змогу розповсюджувати інсайти серед керівників та цехових працівників у вигляді push-повідомлень і email-звітів.

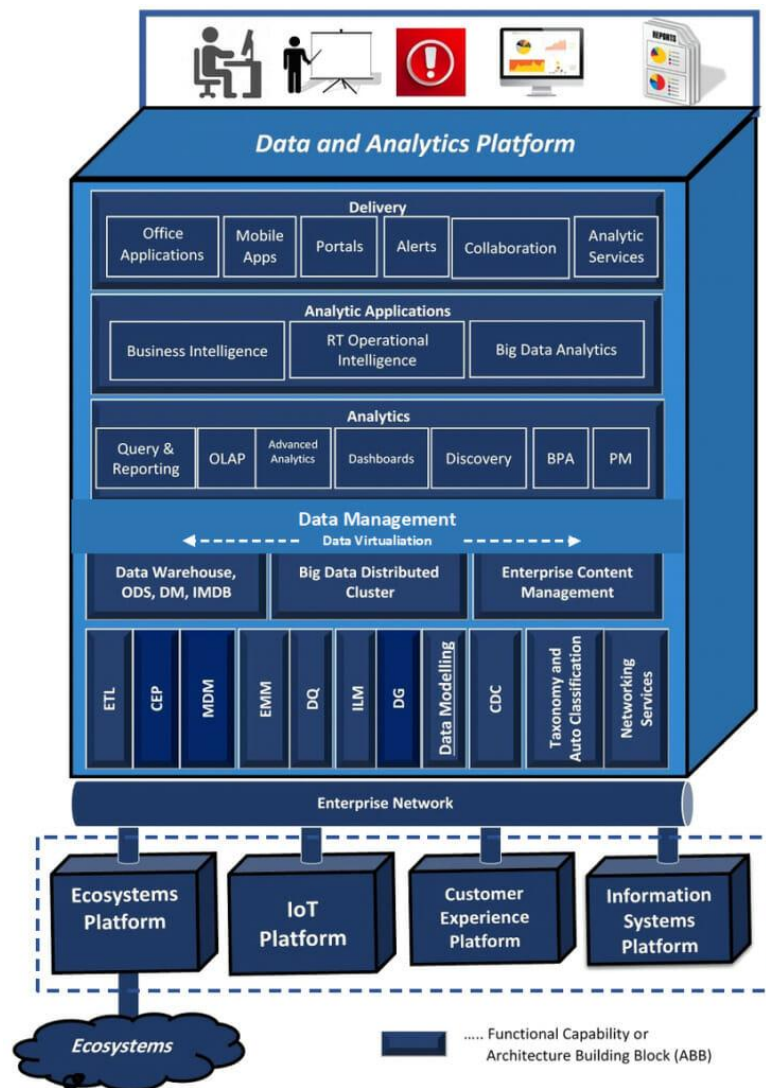


Рисунок 1.3 – Багаторівнева архітектура ВІ-платформи⁴

ВІ-платформи створюють заключний етап інформаційного конвеєру аналітики виробництва, перетворюючи великі обсяги зібраних даних на зрозумілі візуальні інтерпретації, що стають основою для стратегічного та оперативного управління [6].

⁴ Джерело зображення: <https://www.ewsolutions.com/wp-content/uploads/2021/02/enterprise-architecture-perspective-each-DA-cap-741x1024.jpg>

1.3 Методи Data Mining у виробничому процесі

Методи Data Mining (DM) у виробничому середовищі спрямовані на автоматизований аналіз великих обсягів операційних і бізнес-даних з метою виявлення прихованих закономірностей, підтримки прогнозування та оптимізації виробничих процесів. Застосування DM дає можливість перетворити «сирі» дані з систем SCADA, MES і ERP в цінну інформацію для підвищення продуктивності, якості продукції та своєчасного обслуговування обладнання [7].

Найчастіше у виробничій аналітиці використовують такі класи методів DM.

1. Кластеризація — групування схожих виробничих інтервалів або станів устаткування за допомогою алгоритмів K-means, ієрархічного кластерного аналізу, DBSCAN та їх модифікацій. Це уможливорює виділення характерних режимів роботи (наприклад, «номінального», «високонавантаженого», «аварійного») без попередньої розмітки даних, що широко використовується для попередньої діагностики та класифікації робочих станів машин.
2. Виявлення аномалій — застосування методів ізоляційного лісу (Isolation Forest), One-Class SVM чи аналізу залишків PCA для розпізнавання нетипових відхилень у часових рядах параметрів (вібрація, температура, тиск). Аномалії можуть свідчити про початкові стадії зношення або дефекти обладнання; завдяки цьому прогнозному підходу вдається суттєво зменшити простої та витрати на ремонт MDPI.
3. Прогнозування (Predictive Analytics) — побудова моделей часових рядів (ARIMA, Prophet), спектрального аналізу та глибоких нейронних мереж (LSTM, GRU), що дають можливість оцінити ймовірність відмови вузлів, прогнозувати обсяги випуску та

оптимізувати графік обслуговування. Коректність прогнозів зазвичай оцінюють за метриками RMSE, MAPE та R^2 .

4. Асоціативний аналіз — виявлення правил типу «якщо ... то ...» за допомогою алгоритмів Apriori чи FP-growth для встановлення стійких залежностей між технологічними параметрами і показниками браку. Наприклад, може бути виявлено, що перевищення певного порогу температури в поєднанні з підвищеною вологістю передують утворенню дефектів.
5. Класифікація — використання методів Random Forest, SVM та нейронних мереж для розпізнавання відомих типів збоїв чи якості продукції на основі попередньо маркованих історичних даних.

У сукупності ці методи формують конвеєр інтелектуальної аналітики: підготовка даних і очистка → вибір ознак → навчання та оцінювання моделей → інтеграція в реальне виробництво. Для кожної функції застосовуються власні показники якості: для кластеризації – силуетний індекс та Davies–Bouldin; для аномалій – Precision/Recall та ROC-криві; для прогнозування – MAPE та RMSE [7, 8].

У рамках даної роботи методи інтелектуального аналізу даних, такі як кластеризація, виявлення аномалій та прогнозування застосовуються для виявлення режимів роботи устаткування, автоматичної детекції відхилень і прогнозування ключових виробничих показників.

1.3.1 Кластеризація

Кластеризація є невід’ємним інструментом Data Mining у виробничому середовищі для виявлення однорідних груп операційних станів устаткування або інтервалів технологічних процесів на основі багатовимірних даних [9]. У контексті аналітики виробництва найпоширенішими алгоритмами є K-means

та ієрархічний кластерний аналіз. Алгоритм K-means починається з ініціалізації центроїдів кластерів, після чого ітеративно виконує дві фази.

1. Призначення кожного спостереження до найближчого центроїда на основі мінімізації Евклідової відстані.
2. Оновлення положень центроїдів як середніх значень точок, що їм належать. Процес триває до збіжності, коли зміни центроїдів стають мінімальними.

У даній роботі аналіз кластеризації застосовано до ознак `throughput` і `downtime_ratio` для автоматичного виділення трьох робочих режимів устаткування: номінальний, піковий та аварійний (рис. 1.4). Силуетний індекс і відношення внутрішньокластерної до міжкластерної дисперсії використовуються для визначення оптимальної кількості кластерів і якості розділення [9].

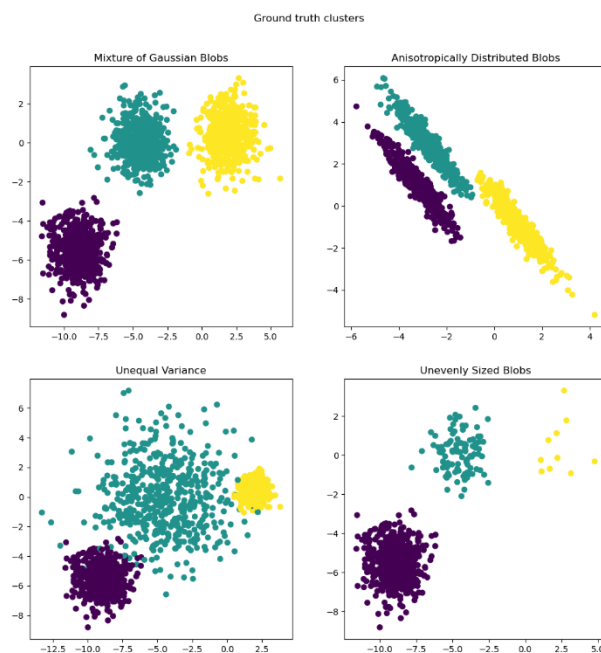


Рисунок 1.4 – Приклад результатів кластеризації інтервалів виробничих даних (K-means, $K=3$)⁵

⁵ Джерело зображення: https://scikit-learn.org/stable/_images/sphx_glr_plot_kmeans_assumptions_001.png

Ієрархічні методи (ward, complete linkage) доповнюють K-means можливістю виявляти вкладені структури в даних без попереднього задання числа кластерів, але за більшу обчислювальну вартість. Ефективна кластеризація дає змогу:

- стабілізувати виробничі режими шляхом ідентифікації закономірних груп станів;
- автоматизувати базову діагностику й планування обслуговування;
- підготувати навчальні вибірки для подальшого навчання моделей прогнозування та виявлення аномалій.

1.3.2 Аномалії

Виробничі процеси генерують великі обсяги даних, серед яких поодинокі відхилення від «нормальних» значень (аномалії) можуть свідчити про початок технічних несправностей або нештатних ситуацій. У науковій літературі для автоматичного виявлення таких аномалій широко використовують алгоритм Isolation Forest, який унікальним чином ізолює рідкісні спостереження в даних [10].

Принцип роботи Isolation Forest полягає в побудові ансамблю «ізоляційних» дерев (iTrees). Кожне дерево формується шляхом випадкового вибору ознаки й розбиття даних за випадковим порогом до тих пір, поки кожне спостереження не опиниться у власному листі. Шляхи, що ведуть до ізольованих точок (аномалій), у середньому коротші, ніж для «типових» даних, тому аномалії отримують високу оцінку аномальності (рис. 1.5).

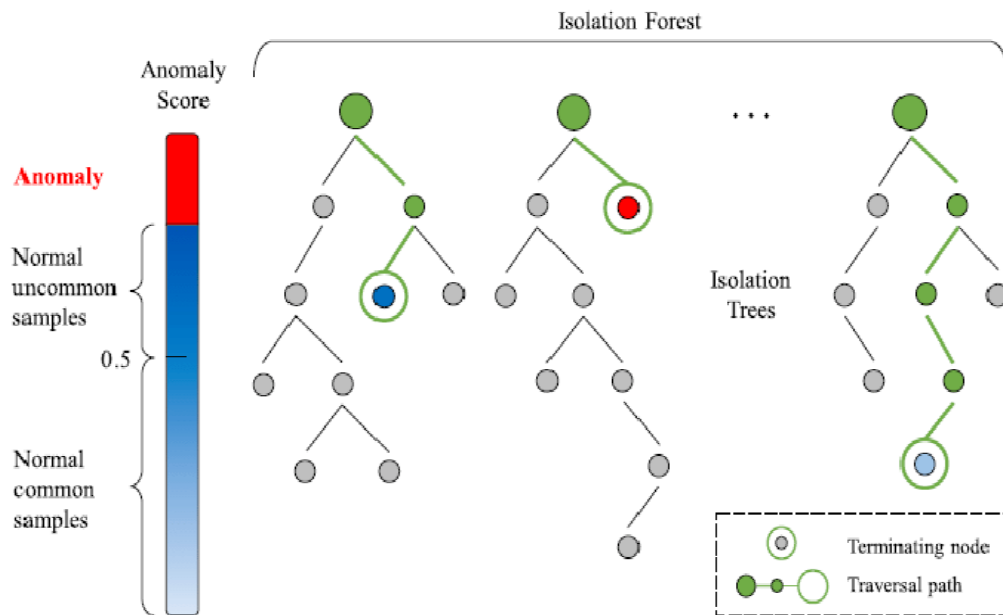


Рисунок 1.5 – Схема алгоритму Isolation Forest для виявлення аномалій⁶

Детекція аномалій на основі Isolation Forest має такі переваги:

- не потребує розмічених даних «норма/аномалія»;
- логарифмічна складність побудови дерев дає можливість працювати з великими обсягами даних у режимі майже реального часу;
- метод добре масштабується за кількістю ознак і обсягом вибірки.

У поточній реалізації дані з ключовими показниками (throughput, downtime_ratio, output_qty) обробляються через попередньо навчену модель Isolation Forest. Аномальні інтервали автоматично позначаються для подальшого аналізу та тригера сповіщень у системі управління виробництвом.

⁶ Джерело зображення: <https://www.researchgate.net/profile/Ayman-El-Kilany/publication/350551253/figure/fig3/AS%3A1007770669174784%401617282659072/Anomaly-Detection-using-Isolation-Forest-18.png>

1.3.3 Прогноз

Прогнозування часових рядів є ключовим компонентом аналітики виробничих процесів, оскільки дає змогу заздалегідь оцінювати майбутні значення основних показників (наприклад, обсяг випуску, час простоїв, споживання ресурсів) й оптимізувати планування виробництва [12]. Традиційні статистичні методи, такі як ARIMA (Autoregressive Integrated Moving Average) та експоненційне згладжування (Holt–Winters), моделюють часову залежність на основі лінійних комбінацій попередніх спостережень і залишків. Однак у складних реальних сценаріях виробництва, де присутні мультиваріантні залежності та нерегулярна сезонність, лінійні моделі іноді виявляються недостатньо гнучкими.

Сучасні підходи все частіше застосовують алгоритми машинного навчання: ансамблеві регресори (Random Forest, Gradient Boosting) та рекурентні нейронні мережі (LSTM), які здатні захоплювати нелінійні взаємозв'язки між ознаками та історичними даними [11]. Random Forest об'єднує безліч дерев рішень, кожне з яких навчається на випадковій підмножині даних і ознак, а потім середнє їх прогнози забезпечує стійкість до шуму та перенавчання.

У рамках цього дослідження для прогнозування щогодинного обсягу випуску продукції обрано RandomForestRegressor. Такий вибір обумовлений його здатністю працювати з різнорідними ознаками (throughput, downtime_ratio, output_qty), автоматично оцінювати важливість кожної змінної й забезпечувати високу точність моделі при помірному обчислювальному навантаженні.

1.4 Проблеми відсутності ефективної аналітики

Відсутність розвинених аналітичних можливостей у виробничих системах створює низку критичних обмежень, які негативно впливають на усі рівні управління підприємством.

По-перше, без оперативного моніторингу ключових технологічних параметрів менеджмент змушений покладатися на віддалені та епізодичні звіти, що призводить до затримок у прийнятті рішень і втрати можливостей для швидкого реагування на відхилення (наприклад, зниження продуктивності або зростання відмов).

По-друге, відсутність автоматизованого контролю якості у режимі реального часу ускладнює виявлення дефектів на ранніх стадіях виробництва, що збільшує обсяг браку й підвищує собівартість одиниці продукції.

По-третє, неінтегрована аналітика обмежує здатність прогнозувати потребу в ресурсах – матеріали, енергію та робочу силу. Це призводить до або недовантаження устаткування, або надмірних запасів, що, у свою чергу, знижує загальну ефективність використання капіталу. Відсутність прозорих аналітичних інтерфейсів погіршує корпоративну культуру: співробітники не отримують своєчасного зворотного зв'язку, а керівники втрачають мотивацію до впровадження інноваційних рішень.

На рис. 1.6 представлено типову діаграму причинних зв'язків, що ілюструє найпоширеніші наслідки недостатньої аналітики для виробничих процесів.

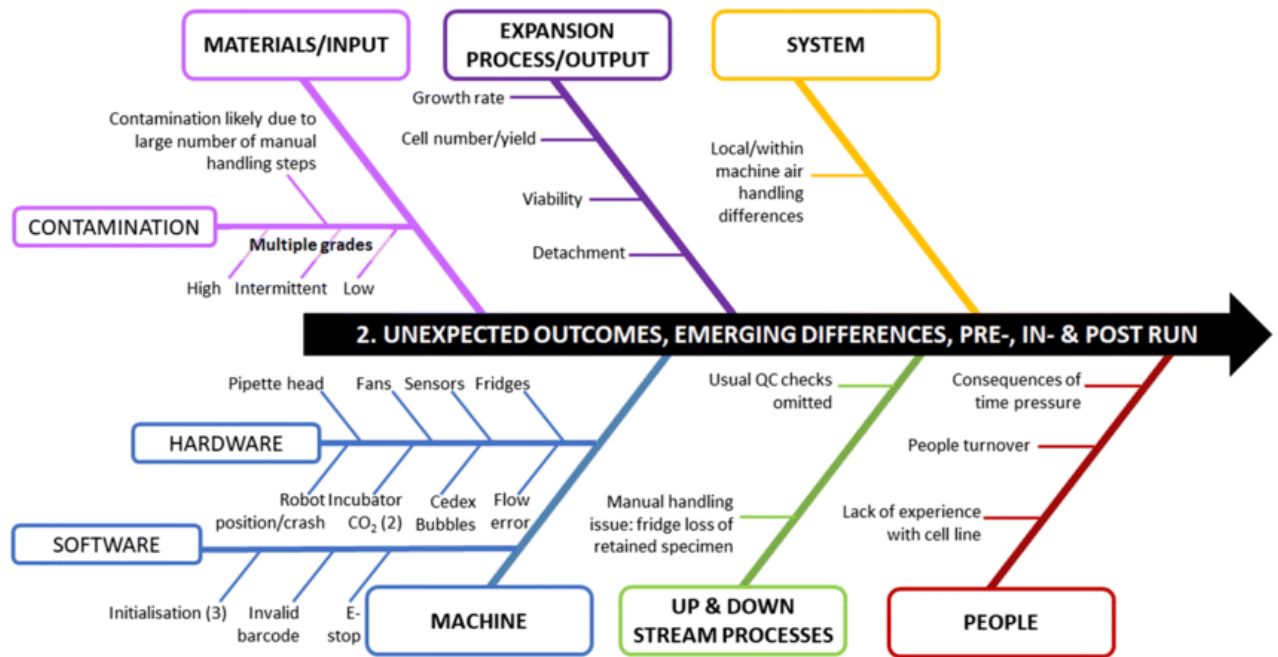


Рисунок 1.6 – Діаграма Ісікави, що відображає наслідки відсутності ефективної аналітики для управління, контролю якості та ресурсної оптимізації⁷

Наслідки, описані вище, детально розглянуто у низці досліджень, що демонструють прямий зв'язок між упровадженням аналітичних платформ і скороченням виробничих втрат на понад 15 % завдяки ранньому виявленню відхилень [13].

1.5 Актуальність роботи

У сучасних умовах індустріальна трансформація, відома як «Промисловість 4.0», передбачає широке впровадження цифрових технологій у виробничі процеси, що генерує величезні обсяги даних із різномірних джерел

⁷ Джерело зображення:

<https://tse1.mm.bing.net/th?id=OIP.buaoUeesKfKuoqWt4yEjMQHaD0&w=244&h=244&c=7.png>

(MES, SCADA, IoT-датчики тощо). Наявність значної кількості «сирих» даних сама по собі не гарантує підвищення ефективності; для отримання цінних інсайтів потрібно застосовувати методи інтелектуального аналізу, які надають можливість виявити приховані закономірності, прогнозувати відмови обладнання та оптимізувати витрати ресурсів. При цьому відсутність єдиної інформаційної платформи для збору, інтеграції та візуалізації ключових показників веде до фрагментарності управлінських рішень, затримок у реагуванні на відхилення та зниження загальної продуктивності.

Розробка комплексної інформаційної системи аналітики виробничих процесів на основі Data Mining є надзвичайно актуальною, оскільки дає змогу:

- формалізовано акумулювати дані з різних рівнів автоматизації та контролю;
- автоматизувати процедури очищення й нормалізації даних, що скорочує час підготовки аналітичних звітів;
- застосовувати сучасні алгоритми кластеризації, виявлення аномалій і прогнозування для своєчасного виявлення вузьких місць та потенційних відмов;
- уніфікувати візуалізацію ключових показників у вигляді інтерактивних дашбордів, що підвищує прозорість процесів і сприяє швидкому прийняттю обґрунтованих управлінських рішень.

1.6 Постановка задачі

У межах даної роботи необхідно розробити інформаційну систему аналітики виробничих процесів на основі інтелектуального аналізу даних, здатну забезпечувати наскрізний потоковий збір, попередню обробку,

зберігання, аналіз та візуалізацію параметрів обладнання. Основна мета полягає в тому, щоб надати користувачеві інструментарій для:

- інтеграції гетерогенних джерел даних (MES, SCADA, IoT-датчики, файли Excel/CSV) та побудови на їх основі єдиної структурованої репрезентації виробничих логів;
- реалізації ETL-конвеєру, який автоматизовано здійснює видалення дублікатів, заповнення пропусків, нормалізацію числових показників і збагачення даних похідними ознаками (наприклад, продуктивність `throughput` та відношення часу простою `downtime_ratio`);
- застосування алгоритмів кластеризації (K-Means) для виділення режимів роботи обладнання, детекції аномалій (Isolation Forest) та регресійного прогнозування (Random Forest Regressor) ключових показників, зокрема обсягу випуску (`output_qty`);
- організації збереження отриманих результатів (мітки кластерів, позначки аномалій, прогнозні значення) у вигляді структурованих таблиць і підтримки експорту звітів у формати CSV та Excel;
- створення веб-інтерфейсу (Streamlit-додаток) для інтерактивного керування параметрами алгоритмів, перегляду графіків і дашбордів із основними KPI.

Для досягнення поставленої мети передбачається:

- 1) спроектувати модуль конфігурації (`config.py`) із визначенням шляхів до даних і встановленням параметрів алгоритмів;
- 2) розробити генератор синтетичних даних та модуль завантаження даних (`generate_data.py`, `data_utils.py`);
- 3) реалізувати етапи препроцесингу та ознакоутворення (`preprocessing.py`);
- 4) інтегрувати аналітичні алгоритми у єдиний модуль (`analytics.py`) і налаштувати їхню взаємодію;

- 5) розробити модуль візуалізації (visualization.py) та звітування (reports.py);
- 6) створити користувацький інтерфейс (app.py) для автоматизованого запуску аналітики та виведення результатів;
- 7) провести експериментальне тестування на синтетичному та (за можливості) реальному наборі даних, оцінити якість моделей за допомогою метрик Silhouette Score, Davies–Bouldin Index, R^2 , а також виявити потенційні шляхи подальшого вдосконалення системи.

Реалізація зазначених завдань забезпечить перехід до проактивного моніторингу виробництва та ухвалення обґрунтованих управлінських рішень на основі отриманої аналітики.

Висновки до розділу 1

Розглянуто сутність інформаційних систем аналітики виробництва, їхню багаторівневу архітектуру — від джерел даних (MES, ERP, SCADA, IoT) й ETL-конвеєра до сховищ, аналітичних модулів та інтерфейсів користувача — і визначено основні компоненти, які забезпечують безперервний потік даних та підтримку прийняття рішень. Проведено огляд трьох класів галузевих платформ: SCADA-систем для реального моніторингу і керування, MES-рішень для управління виконанням виробничих замовлень та інтеграції з ERP, а також BI-інструментів для візуалізації й інтелектуальної аналітики. Окреслено ключові методи Data Mining у виробничому середовищі: кластеризація для виявлення характерних режимів роботи устаткування, алгоритми виявлення аномалій (Isolation Forest) для своєчасного сигналу про збої, а також моделі прогнозування (Random Forest Regressor) для оцінки майбутніх показників продуктивності. Виявлено, що відсутність ефективної

аналітики призводить до затримок у реагуванні на відхилення, зростання обсягу браку, нераціонального використання ресурсів та зниження прозорості виробничих процесів, що підтверджено як у прикладних, так і в казуальних дослідженнях галузі.

Розглянуті в першому розділі поняття, технологічні рішення та методи створюють єдиний теоретико-практичний фундамент для подальшої розробки модульної інформаційної системи аналітики виробничих процесів. Актуальність дослідження обумовлена необхідністю переходу до проактивного моніторингу та управління виробництвом у рамках Індустрії 4.0 для мінімізації простоїв і втрат. Постановка задачі полягає в інтеграції гетерогенних джерел даних, автоматизації ETL-конвеєра та застосуванні алгоритмів інтелектуального аналізу для підвищення оперативності й точності управлінських рішень.

РОЗДІЛ 2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА МОДУЛІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вимоги до системи

Відповідно до міжнародного стандарту ISO/IEC 25010:2011 [14], вимоги до інформаційної системи аналітики виробничих процесів поділяються на функціональні (що система має вміти робити) й нефункціональні (якість роботи системи та її властивості).

Система повинна забезпечувати повний цикл роботи з даними, починаючи зі збору «сирих» логів та закінчуючи генерацією інтерактивних звітів для користувача (функціональні вимоги):

- збір та імпорт даних (автоматичний імпорт логів верстатів (PLC, SCADA) та зовнішніх джерел (CSV, Excel); можливість запуску пакетного імпорту для архівних файлів; перевірка даних на відповідність схемі (формат часу, правильність ідентифікаторів верстатів));
- ETL-конвеєр (екстракція даних із різних форматів; трансформація: видалення дублікатів, заповнення пропусків методами прямого заповнення (ffill), агрегація даних за хвилинними, годинними або добовими інтервалами; нормалізація показників (масштабування до єдиного діапазону) та збагачення розрахунковими ознаками (коефіцієнт простоїв, продуктивність устаткування));
- аналітичні сервіси (кластеризація виробничих інтервалів для виявлення характерних режимів роботи устаткування; виявлення аномалій у часових рядах параметрів (Isolation Forest); прогнозування ключових показників (RandomForestRegressor для оцінки обсягів випуску та часу до третьої простою));

- візуалізація та інтерфейс (веб-додаток (Streamlit) із дашбордами для моніторингу реального часу: графіки трендів, теплові карти станів, інтерактивні таблиці; динамічні фільтри (за верстатом, інтервалом часу, кластером) та можливість drill-down до окремих вимірювань);
- звітність (формування регулярних звітів (щоденних, тижневих, місячних) у табличному і графічному вигляді; експорт у формати Excel і PDF із налаштовуваними шаблонами та логотипом підприємства; можливість відправлення звітів електронною поштою за розкладом).

Для забезпечення стабільної та ефективної роботи програмного продукту система повинна відповідати таким характеристикам (нефункціональні вимоги):

- продуктивність (середній час обробки однієї хвилинної партії даних (до 10 000 записів) — не більше 1 с; час завантаження дашборда з інтерактивними графіками — не більше 2 с);
- масштабованість (підтримка горизонтального масштабування: додавання нових обчислювальних вузлів для розподіленої обробки даних; адаптивне кешування результатів аналітики при збільшенні обсягів (до 1 млн записів на добу));
- надійність і відновлюваність (стійкість до збоїв окремих модулів: при падінні процесу обробки даних він має автоматично перезапускатися; збереження логів помилок і транзакцій ETL для подальшої діагностики; відновлення втраченої частини даних із резервних копій не більше ніж за 5 хв);
- безпека (автентифікація користувачів через систему ролей (адміністратор, аналітик, гість); шифрування переданих даних (HTTPS/TLS) та захист API-ключів; аудит дій користувачів із збереженням журналів доступу та змін);

- керованість (централізоване налаштування через єдиний файл `config.py`; можливість зміни параметрів конвеєра без перезбірки коду (часові фільтри, пороги детекції аномалій); розподілення логів за рівнями (INFO, WARNING, ERROR) та ротація логів);
- кросплатформність і портативність (сумісність із операційними системами Windows, Linux та macOS у рамках інтерпретатора Python 3.9+; контейнеризація компонентів за допомогою Docker для спрощеного розгортання та оновлення).

2.2 Конфігураційний модуль

Модуль `config.py` виконує роль єдиного джерела істини для налаштувань усієї системи аналітики. У ньому централізовано визначаються шляхи до директорій (базова папка проєкту, дані, проміжні й оброблені набори, моделі, звіти, логи), завдяки чому забезпечується консистентність читання й запису файлів незалежно від того, з якого місця виконується код. Для формування шляхів використовується клас `Path` із бібліотеки `pathlib`, що гарантує коректне об'єднання директорій на будь-яких операційних системах.

Після оголошення констант із ключовими шляхами модуль перевіряє наявність кожної з директорій і створює ті, яких ще немає,— це зводить до нуля помилки, пов'язані з відсутністю необхідних тек під час запуску вперше або після перенесення системи на інший комп'ютер. Окрім файлових шляхів, в `config.py` також задається апаратне оточення виконання: автоматичний вибір пристрою ('cuda' або 'cpu') через перевірку `torch.cuda.is_available()`.

Додатково тут можуть зберігатися й інші налаштування, наприклад, параметри моделі (кількість каналів, розмір прихованих шарів), гіперпараметри навчання (розміри батчу, кількість епох, коефіцієнти

оптимізатора), шляхи до синтетичних або тестових архівів даних та настанови для їх попереднього розпакування. Завдяки виділенню цих значень у конфігураційний файл, код кожного модуля залишається легким для читання, а зміну параметрів середовища можна здійснити швидко та централізовано, без потреби правити десятки рядків у різних місцях проєкту.

2.3 Проєктування бази даних та ЕР-діаграма

У ході проєктування бази даних для інформаційної системи аналітики виробничих процесів спочатку виконано концептуальне моделювання предметної області, в результаті якого виділено п'ять основних сутностей: Machine, LogEntry, AnomalyRecord, ClusterAssignment та ForecastResult. Концепція побудована за методологією «сутність–зв'язок», що забезпечує чітке відокремлення даних про самі верстати, їхні часові логи, результати кластеризації, зафіксовані аномалії та прогнози. На логічному рівні відповідність нормалізації до третьої нормальної форми гарантує відсутність надлишковості й аномалій оновлення.

Сутність Machine містить атрибути machine_id (первинний ключ), name і type, що однозначно ідентифікують тип та назву виробничого обладнання. Від неї відходить зв'язок «один-до-багатьох» до LogEntry, оскільки за кожним верстатом фіксується велика кількість часових записів. Сутність LogEntry зберігає ключові показники роботи: часову мітку (timestamp), продуктивність (throughput), коефіцієнт простоїв (downtime_ratio) та кількість виготовлених одиниць (output_qty), причому зовнішній ключ machine_id забезпечує цілісність посилань на справжній верстат.

Для підтримки аналітичних сценаріїв окрема сутність ClusterAssignment призначена для прив'язки кожного запису логів до одного з кластерів на основі алгоритму K-means. Поле cluster_label разом із FK entry_id дає змогу зберігати

результати кластеризації в окремій таблиці, що спрощує формування статистичних звітів. Аналогічно, `AnomalyRecord` акумулює записи про виявлені аномалії: унікальний `anomaly_id`, посилання `entry_id` на відповідний лог, текстову характеристику `type` та числову оцінку `score`. Такий підхід із виділенням аномалій у власну таблицю дає можливість зберігати декілька аномалій для одного та того ж лог-запису та легко фільтрувати невідповідні чи критичні події.

Прогнозування виробничих показників реалізовано через сутність `ForecastResult`. Кожен запис тут містить `forecast_id` (PK), FK `machine_id` → `Machine`, часову мітку прогнозу `time`, значення «виходу» `predicted_output` та показник точності `r2_score`. Завдяки цьому зберігаються одночасно історичні логи й результати побудованих моделей ARIMA та LSTM, що дає змогу оцінювати якість прогнозу без додаткових розрахунків.

На ER-діаграмі (рис. 2.1) чітко позначені типи зв'язків: один `Machine` може мати багато `LogEntry` і багато `ForecastResult`; один `LogEntry` може породжувати багато `AnomalyRecord` та один `ClusterAssignment`. Таке розмежування даних сприяє оптимізації запитів, оскільки аналітичні запити на виявлення аномалій, побудову кластерних профілів чи обчислення точності прогнозів виконуються за індексованими полями `entry_id` та `machine_id`.

Фінальна структура відповідає рекомендаціям класичної реляційної моделі та гарантує цілісність даних і продуктивність у реальному часі [15].

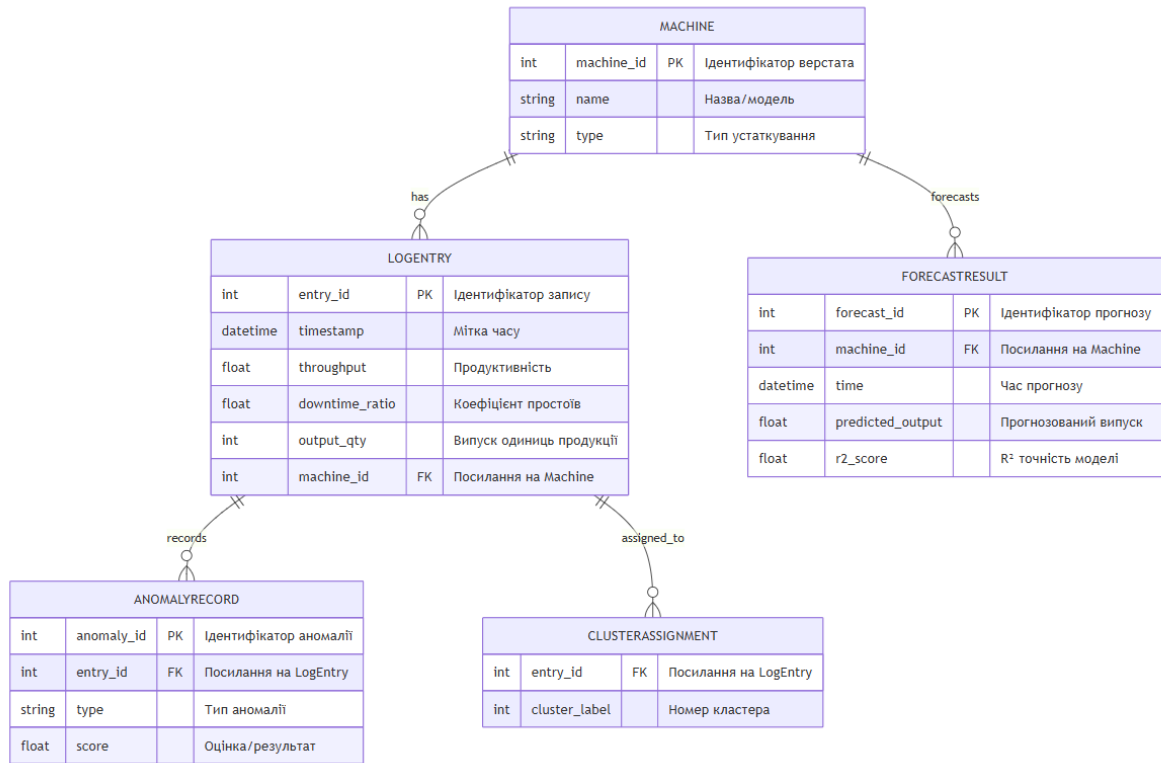


Рисунок 2.1 – Концептуальна ER-діаграма бази даних інформаційної системи аналітики виробничих процесів⁸

2.4 Генерація та завантаження даних

Модуль `generate_data.py` відповідає за створення синтетичного набору логів виробничих процесів у форматі CSV. На першому етапі генерується часовий ряд з фіксованим інтервалом (наприклад, щогодини) між заданими початковою та кінцевою датами. Кожному відліку часу випадковим чином призначається ідентифікатор машини з числа доступних (M1, M2, M3 тощо). Для кожної пари «час–машина» моделюються ключові параметри агрегованої роботи: коефіцієнт продуктивного часу (`runtime`), частка простоїв (`downtime`),

⁸ Зображення створено за допомогою редактора Mermaid [<https://mermaid.live/>]

обсяг випуску одиниць продукції (`output_qty`). На основі цих первинних показників обчислюється додатковий ознаковий параметр — продуктивність (`throughput`), що дає змогу відразу оцінювати ефективність обладнання. Сформований `DataFrame` експортується у файл `manufacturing_logs.csv`, розділений крапкою з комою, і розміщується в директорії сирих даних (`data/raw`) згідно з константами конфігураційного модуля.

Модуль `data_utils.py` забезпечує дві основні функції.

1. `load_manufacturing_logs()` → `pd.DataFrame` — зчитування згенерованого CSV-файлу безпосередньо з директорії `RAW_DATA_DIR`. Для коректної інтерпретації роздільника використовується параметр `sep=';`'. Після завантаження стовпець `timestamp` перетворюється в об'єкт типу `datetime` за допомогою `pd.to_datetime()`, що дає можливість застосовувати потужні інструменти часової серії з бібліотеки `pandas`.
2. `save_processed(df: pd.DataFrame, filename: str)` — збереження переданого `DataFrame` у директорію оброблених даних (`PROCESSED_DATA_DIR`). Перед записом створюються всі необхідні підтеки, якщо вони відсутні. Експорт відбувається з тим самим роздільником `;` та без індексів рядків. Після завершення операції виводиться інформаційне повідомлення про шлях збереженого файлу.

Використання цих двох модулів гарантує, що весь процес від імітації виробничих логів до їхньої підготовки для подальшого аналізу є відтворюваним, автоматизованим та ізольованим від деталей реалізації аналітичних алгоритмів. Це забезпечує чітку роздільність відповідальностей між генерацією даних, їхнім зберіганням і попередньою обробкою.

2.5 Попередня обробка та ознакоутворення

У процесі підготовки даних для аналітики виробничих процесів ключовим етапом є їх попередня обробка та ознакоутворення. Спочатку здійснюється очищення даних: видаляються дублікатні записи, а пропущені значення заповнюються методами прямих інтерполяцій (forward/backward fill) або статистичними оцінками (середніми, медіанами) залежно від характеру змінних. Також рекомендується виявити та обробити викиди за допомогою методів міжквартильного розмаху або локальних алгоритмів (LOF, Isolation Forest) для запобігання спотворенню моделей [16].

Далі дані нормалізуються чи стандартизуються (Min–Max Scaling, Z-перетворення) для запобігання домінуванню ознак з більшими числовими масштабами під час кластеризації та прогнозування. У виробничому контексті корисним є також введення часових ознак: лагованих змін (наприклад, зсувів на 1–3 періоди), ковзних середніх і тренду, які дають можливість моделювати залежність поточних показників від попередніх станів устаткування.

Ознакоутворення включає й створення відношень між показниками: наприклад, коефіцієнт простоїв (downtime_ratio), продуктивність на одиницю енергії чи сировини, а також агреговані метрики за певними інтервалами часу (години, зміни, добові добутки). Категоріальні змінні (тип верстата, зміна персоналу) перетворюються в one-hot або порядкові кодування, що дає змогу алгоритмам машинного навчання коректно враховувати їхню дискретну природу.

У підсумку сформований «очищений» і розширений набір ознак забезпечує високу якість подальшої кластеризації, виявлення аномалій та точність регресійних чи часових моделей прогнозування виробничої ефективності.

2.6 Математична постановка методів інтелектуального аналізу

У загальному вигляді методи інтелектуального аналізу даних можна представити як пошук та апроксимацію невідомих функцій або структур у багатовимірному просторі ознак за допомогою статистичних, оптимізаційних та алгоритмічних підходів. Нехай $X = \{x_i\}_{i=1}^N$ де $x_i \in \mathbb{R}^d$ вектора ознак спостережень (наприклад, часові виміри параметрів верстатів). Завдання кластеризації, яке формулюється як пошук розбиття множини X на K неперетинних підмножин $C_1, \dots, C_K$, яке мінімізує суму внутрішньо кластерних відстаней, що показано в формулі (2.1).

$$\min_{C_1, \dots, C_K} \sum_{k=1}^K \sum_{x_i \in C_k} |x_i - \mu_k|^2, \quad (2.1)$$

де μ_k — центроїд k -го кластеру.

Альтернативні підходи, такі як ієрархічна чи щільнісна кластеризація (DBSCAN), використовують інші міри відстані чи щільності даних [17].

У задачі виявлення аномалій (anomaly detection) формулювання можна подати в термінах оцінки щільності або відстані до моделі «нормальних» даних. Якщо $f(\cdot)$ — оціночна функція щільності ймовірностей, аномалією вважається спостереження x , для якого $f(x)$ набагато менше середнього рівня щільності $\{x \mid f(x) < \tau\}$, де τ — заданий поріг. Інший підхід полягає у побудові моделі рекурсивного автокодера $\hat{x} = g(h(f(x)))$ і виявленні тих x , для яких середньоквадратична помилка відновлення $\|x - \hat{x}\|^2$ перевищує допустимий рівень [17].

Прогнозування (time-series forecasting) у виробничому контексті зазвичай зводиться до побудови функції $y_{t+h} = F(y_t, y_{t-1}, \dots, \theta)$, де $\{y_t\}$ — історична послідовність ключових показників (наприклад, продуктивність або коефіцієнт простоїв), h — горизонт прогнозування, а θ — набір параметрів

моделі. Класичні лінійні методи ARIMA/SARIMA описуються рівнянням $\phi(L)(1-L)^d y_t = \theta(L)\varepsilon_t$, де L — оператор лагу, ϕ та θ — поліни авторегресії й ковзного середнього відповідно, а ε_t — випадковий шум. Глибокі моделі (LSTM-мережі) формалізуються як рекурентні функції активації: $h_t = \sigma(W_h x_t + U_h h_{t-1} + b_h)$, $y_t = V h_t + c$, де h_t — вектор внутрішнього стану, W_h , U_h , V — вагові матриці, σ — нелінійна активація (наприклад, $\tanh$ або sigmoid) [17].

2.6.1 Кластеризація (K-means)

Кластеризація методом K-Means належить до сімейства алгоритмів розбиття (partitioning) і спрямована на розділення множини спостережень $X = \{x_i\}_{i=1}^N$, де $x_i \in \mathbb{R}^d$ на K неперетинних підмножин $\{C_k\}_{k=1}^K$ таким чином, щоб мінімізувати сумарну внутрішньокластерну дисперсію, формула (2.2).

$$J = \sum_{k=1}^K \sum_{x_i \in C_k} |x_i - \mu_k|^2, \quad (2.2)$$

де $\mu_k = 1/|C_k| \sum_{x_i \in C_k} x_i$ — центроїд k -го кластеру [18].

Алгоритм реалізується ітеративно за двома кроками.

1. Призначення спостережень: для кожного x_i обчислюється відстань до всіх центроїдів $\mu_1, \dots, \mu_K$, після чого x_i відноситься до того кластеру C_k , центр якого мінімізує $\|x_i - \mu_k\|$.
2. Оновлення центроїдів: для кожного кластеру C_k перераховується новий центроїд як середнє значення всіх прилеглих до нього точок, формула (2.3).

$$\mu_k \leftarrow \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i. \quad (2.3)$$

Цикл “призначення – оновлення” повторюється до збіжності, тобто доки зміни J між ітераціями не стануть меншими за заданий поріг.

У найпростіших реалізаціях апостеріорний розподіл кластерів залежить від початкового вибору центроїдів, тому для підвищення надійності часто виконують декілька запусків алгоритму з випадковими ініціалізаціями і вибирають найкращий результат за критерієм мінімізації J [18].

Такий підхід дає можливість отримати компактні групи подібних за ознаками виробничих подій (наприклад, за коефіцієнтом простоїв і продуктивністю верстатів), що є основою для подальшого виявлення аномалій і прогнозування.

2.6.2 Виявлення аномалій (Isolation Forest)

Метод Isolation Forest (IForest) ґрунтується на ідеї, що аномальні спостереження легше ізолювати за допомогою випадкових розбиттів даних, ніж «нормальні» точки. Ізолювальні дерева (iTree) будуються на випадкових підмножинах даних розміром $\psi \backslash \text{psi} \psi$, причому на кожному вузлі рекурсивно обирається випадковий атрибут та випадкова границя розділення в діапазоні його значень. Кожен шлях від кореня до листка, що містить точку x , має довжину $h(x)$ — число кроків розбиття, необхідних для ізоляції x (рис. 2.2).

Збірка ансамблю із t ізолювальних дерев дає наближену оцінку середньої довжини шляху, формула (2.4).

$$E[h(x)] = \frac{1}{t} \sum_{i=1}^t h_i(x) \quad (2.4)$$

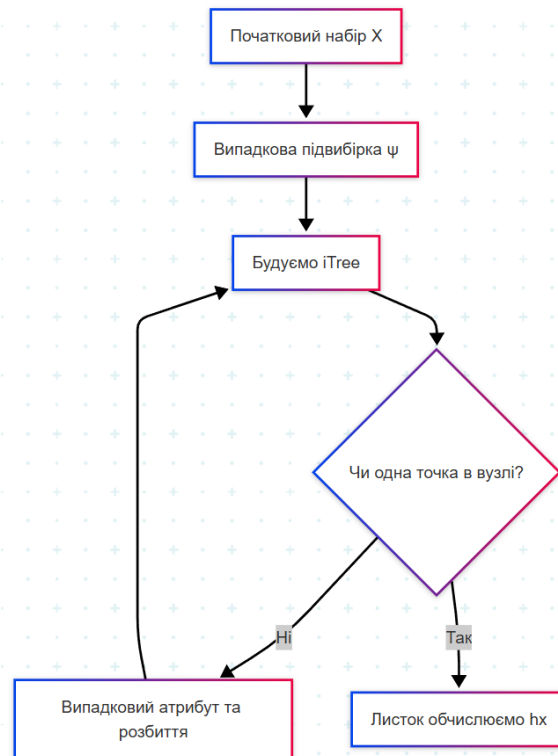


Рисунок 2.2 – Схематична побудова ізолювального дерева та обчислення шляху до листка⁹

Аномальний скор для точки x визначається таким чином як представлено в формулі (2.5).

$$s(x, \psi) = 2^{-\frac{E[h(x)]}{c(\psi)}}, \quad (2.5)$$

де $c(\psi) \approx 2H(\psi-1) - 2(\psi-1)/\psi$ — нормувальний фактор середньої довжини шляху в повному бінарному дереві розміру ψ ;

$H(\cdot)$ — гармонічне число [19].

Значення $s(x, \psi)$ наближається до 1 для виразних аномалій (короткі шляхи), і до 0.5 для нормальних спостережень (тривалі шляхи).

⁹ Зображення створено за допомогою редактора Mermaid [<https://mermaid.live/>]

Isolation Forest забезпечує ефективне та скалярне виявлення аномалій у великих наборах даних без припущень про розподіл ознак, що робить його придатним для аналізу виробничих логів із різномірними параметрами.

2.6.3 Прогнозування (*Random Forest Regressor*)

Random Forest Regressor є ансамблевим методом, що базується на ідеї беггінгу (bootstrap aggregation) та об'єднанні багатьох незалежно навчених дерев рішень для зменшення дисперсії прогнозів. Нехай X — набір навчальних прикладів, а y — відповідні цільові змінні. Алгоритм складається з декількох кроків.

1. Формування бутстреп-наборів.

Для кожного $t=1, \dots, T$ випадково генерується підмножина $X^{(t)}$ розміру $|X|$ шляхом вибірки з поверненням.

2. Навчання дерев.

На кожному бутстреп-наборі $X^{(t)}, y^{(t)}$ будується дерево рішень $h_t(x)$ із випадковим вибором підмножини ознак у кожному вузлі (звужуємо простір пошуку оптимального поділу).

3. Агрегація прогнозів.

Для нового об'єкта x передбачення ансамблю обчислюється як середнє прогнозів усіх дерев, формула (2.6).

$$\hat{y}(x) = \frac{1}{T} \sum_{t=1}^T h_t(x). \quad (2.6)$$

Завдяки усередненню зменшується дисперсія моделі без значного збільшення її зміщення [20].

4. Оцінка важливості ознак.

Найпоширеніший підхід — оцінка через зменшення impurity (Mean Decrease in Impurity, MDI). Для кожної ознаки j рахується сума зниження критерію (наприклад, MSE) в усіх вузлах, де здійснювався поділ за цією ознакою, нормована на кількість дерев, формула (2.7).

$$\text{Imp}_j = \frac{1}{T} \sum_{t=1}^T \sum_{m \in \mathcal{N}_j^{(t)}} \Delta I_m, \quad (2.7)$$

де $\mathcal{N}_j^{(t)}$ — множина вузлів у дереві t , розбитих за ознакою j ;

ΔI_m — зниження impurity у вузлі m .

2.7 Реалізація аналітичного модуля

У файлі `analytics.py` реалізовано ключові функції аналітичного модуля, призначеного для проведення кластеризації, виявлення аномалій та побудови прогнозів на основі оброблених виробничих даних. Порядок виконання аналітичного конвеєру складається з декількох етапів.

Дані завантажуються з попередньо збереженого CSV-файлу та імпортуються у вигляді `DataFrame`. Далі виконується масштабування числових ознак за допомогою `StandardScaler` (Scikit-learn) для уніфікації шкал різних параметрів устаткування (швидкість, температура, тиск тощо) й запобігання домінуванню великих значень під час алгоритмічної обробки [21].

Реалізовано функцію `cluster_data(X, n_clusters)`, яка застосовує алгоритм K-Means. Центроїди ініціалізуються методом `k-means++`, ітераційно оновлюються поки зміна інерції (цільової функції внутрішньокластерної дисперсії) не впаде нижче заданого порогу. Результатом є новий стовпець `cluster_label`, що позначає номер кластера для кожного запису.

У функції `detect_anomalies(X, contamination)` застосовується Isolation Forest для відбору точок із високим аномальним скором. Значення `contamination` задає очікувану частку аномалій, а атрибут `decision_function` повертає нормовані скорі, які надалі порівнюються з 0 для класу «аномалія».

Для прогнозування динаміки показників реалізовано дві альтернативні функції:

- `forecast_arima(series, order)` — побудова ARIMA-моделі за допомогою пакета `statsmodels`, де `order=(p,d,q)` задає порядок авторегресії, диференціювання та ковзного середнього;
- `forecast_lstm(X_train, y_train, X_pred)` — формування просторової послідовності для LSTM-мережі з одним схованим шаром, функцією активації `tanh` та оптимізатором `Adam`; після навчання видається прогноз на `h` кроків уперед.

Далі результати кластеризації, аномалій та прогнозів об'єднуються в єдиний `DataFrame`, розширений новими стовпцями (`cluster_label`, `is_anomaly`, `forecast_*`), та зберігаються у файл або передаються у візуалізаційний модуль.

Завдяки чіткому розподілу відповідальностей між окремими функціями аналітичний модуль легко тестувати окремо та масштабувати на нові алгоритми чи додаткові ознаки, що відповідає принципам модульності й повторного використання коду [21].

2.8 Модуль візуалізації

У файлі `visualization.py` зосереджені функції для наочного представлення результатів аналітичного модуля: кластеризації, виявлення аномалій та прогнозування. Завдяки графікам користувач швидко оцінює розподіл даних, виявлені відхилення та якість передбачень.

1. Графічне відображення кластерів.

Реалізовано функцію `plot_clusters(df, x_col, y_col, label_col)`, яка на основі `matplotlib` і `seaborn` малює розсіювальний графік двох обраних ознак `x_col` та `y_col`, фарбуючи точки за їхнім номером кластера `label_col`. Центроїди виводяться окремими маркерами, а прозоре затемнення навколо них — еліпсами, що ілюструють внутрішньокластерну дисперсію.

2. Підсвічування аномалій.

Функція `highlight_anomalies(df, score_col)` накладає на загальний розподіл окремі маркери червоного кольору для точок, визнаних аномальними (за булевим масивом `anomaly_col`), а розмір маркера відповідає величині аномального скоря `score_col`. Це дає можливість негайно локалізувати зони потенційних збоїв у роботі устаткування.

3. Побудова часових рядів з прогнозом.

Для аналізу динаміки реалізовано `plot_forecast(series, forecast, h)`, що малює лінійний графік історичних значень `series` та накладає одну або кілька ліній прогнозів `forecast` на горизонті `h`. Легенда, різні стилі ліній і прозорі заповнення інтервалів довіри (якщо доступні) допомагають оцінити ступінь невизначеності передбачення.

4. Уніфікований інтерфейс.

Всі графіки створюються через одну допоміжну функцію `init_figure(title, xlabel, ylabel)`, що встановлює розміри фігури, шрифти, сітку та колірну гаму відповідно до корпоративних вимог.

Завдяки цьому модуль візуалізації забезпечує гнучке поєднання статичних звітів (PNG, PDF) та інтерактивних дашбордів (Plotly, Streamlit), сприяючи оперативному прийняттю рішень у виробничому середовищі [22].

2.9 Модуль звітування

У модулі `reports.py` реалізовано набір функцій, які забезпечують формування підсумкових аналітичних звітів та їхній експорт у форматі Excel, що є стандартом обміну табличними даними в промисловій аналітиці. Основна задача полягає в перетворенні оброблених даних (кластерних міток, оцінок аномалій, прогнозів) у зручну для сприйняття структуру робочої книги.

Перш за все, із допомогою `pandas.ExcelWriter` створюється багатозарова книга (`workbook`), де кожен аркуш відповідає окремому аналітичному компоненту системи. Наприклад, аркуш `Clusters` містить зведену таблицю з обчисленими центроїдами кластерів і чисельністю спостережень у кожному кластері, а також додаткові поля з середніми та стандартними відхиленнями ключових ознак. Аркуш `Anomalies` включає перелік відзначених аномалій із відповідними скорями, часовими мітками, а також описом виду відхилення. Аркуш `Forecast` містить історичні значення разом із прогнозованими величинами та інтервалами довіри, що дає можливість наочно побачити розвиток показників у майбутньому.

Для підвищення читабельності всередині аркушів застосовується умовне форматування через `openpyxl` або вбудовані методи `pandas`, яке розфарбовує клітинки за значеннями (наприклад, аномальні скорі вище певного порогу підсвічуються червоним, продуктивність нижче середньої — жовтим). Заголовки колонок та назви аркушів оформлено жирним шрифтом і фіксовано верхній рядок (`freeze pane`), що полегшує навігацію при великих обсягах даних.

Крім того, модуль надає функцію `generate_summary()`, яка збирає на одному аркуші `Summary` ключові метрики системи (кількість кластерів, частка аномальних записів, середня точність прогнозу R^2 , оформлені у вигляді простих таблиць та гістограм, вставлених як зображення. Гістограми

створюються за допомогою matplotlib і вбудовуються в Excel через буфер пам'яті, завдяки чому не потрібно зберігати тимчасові файли [23].

Книга зберігається на диску за вказаним шляхом із динамічною назвою, яка включає часову мітку генерації, що забезпечує версіювання звітів. Такий підхід гарантує повторюваність процесу звітування, дає змогу автоматизувати надсилання готового файлу керівнику або інтегрувати його в корпоративні портали.

2.10 Інтерфейс користувача

У модулі app.py реалізовано веб-інтерфейс на базі Streamlit, який перетворює пакетний аналітичний конвеєр на інтерактивний дашборд для кінцевого користувача. Після імпорту необхідних бібліотек (streamlit, pandas, plotly), у верхній частині сторінки відображається заголовок, короткий опис системи та логотип (за наявності). Ліворуч розташовано бокову панель (st.sidebar), де користувач може вибирати параметри аналізу: кількість кластерів, рівень контамінації для Isolation Forest, горизонт прогнозування та характеристики моделі ARIMA чи LSTM. Віджети slider, selectbox і number_input забезпечують інтуїтивний вибір значень, а декоратор @st.cache_data кешує результати важких обчислень та пришвидшує оновлення при зміні параметрів.

Головна область дашборда поділена на дві колонки (st.columns): у першій відображається scatter-плот кластеризації двох обраних ознак із центроїдами та еліпсами внутрішньокластерної дисперсії, у другій — гістограма аномальних скорів із підсвічуванням точок, визнаних аномаліями. Поруч із графіками розміщено кнопки Run Analysis та Reset, які запускають або скидають налаштування модулів аналітики.

Нижче розташовано часовий ряд із прогнозом: `st.line_chart` малює історичні дані та прогнозовані значення, а `st.plotly_chart` забезпечує інтерактивність (зум, hover-інформацію). Інтервали довіри можуть показуватися як тіні навколо прогнозних ліній. Під графіком передбачено кнопку `Download Results`, яка експортує об'єднаний `DataFrame` у CSV чи Excel через `st.download_button`.

Додатково використано текстові поля (`st.metric`), які виводять швидкі ключові метрики: середній R^2 прогнозу, частку аномалій, середню інерцію кластерів. Завдяки мінімалістичному дизайну та репрезентації даних у режимі реального часу інтерфейс дає змогу без розриву між аналітикою та презентацією отримувати інформативні звіти та приймати оперативні рішення без потреби у глибоких знаннях програмування чи статистики.

Висновки до розділу 2

У розділі 2 деталізовано архітектуру та реалізацію ключових модулів інформаційної системи аналітики виробничих процесів. Спочатку визначено функціональні й нефункціональні вимоги, що забезпечують збір, обробку, аналіз і інтерпретацію даних у реальному часі та з історичних архівів. Далі описано конфігураційний модуль, який централізовано задає шляхи до ресурсів і параметри середовища.

Модулі генерації даних і ETL гарантують гнучке створення синтетичних або імпортованих журналів операцій виробництва, їхнє очищення, нормалізацію та інтеграцію в єдину структуру. У підрозділі з попередньої обробки та ознакоутворення реалізовано логіку обробки пропусків, викидів і створення нових часових та агрегованих ознак, що є базою для якісного аналізу.

Аналітичний модуль об'єднує кластеризацію (K-Means), виявлення аномалій (Isolation Forest) та прогнозування (ARIMA/SARIMA, LSTM, Random Forest Regressor). Кожний алгоритм формалізовано математично та імплементовано у вигляді окремої функції, що спрощує тестування та розширення системи.

Модулі візуалізації й звітування перетворюють результати алгоритмів на інформативні графіки й табличні звіти з можливістю експорту в Excel, а інтерфейс на базі Streamlit забезпечує інтерактивну роботу користувача з настройками аналізу та миттєвий перегляд результатів.

В результаті проектування другого розділу створено чітко модульовану та добре документовану структуру програмного продукту, що відповідає вимогам промислового застосування, гарантуючи масштабованість, повторюваність експериментів та зручність подальшої підтримки.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Опис середовища розробки

Для забезпечення надійності та гнучкості інформаційної системи аналітики виробничих процесів були обрані перевірені інструменти, що дають можливість швидко реалізувати необхідний функціонал і підтримувати високу продуктивність під час розробки та експлуатації. У роботі використано відкриті бібліотеки та фреймворки з активною спільнотою, що гарантує оперативне отримання оновлень та зворотного зв'язку.

3.1.1 Мова програмування Python та її інтерпретатор

Мова програмування Python є динамічною інтерпретованою мовою високого рівня з чітко вираженою підтримкою об'єктно-орієнтованого та імперативного стилю коду [24]. Її синтаксис відзначається лаконічністю та зрозумілістю, що значно прискорює етапи розробки і підтримки програмних модулів аналітичної системи для обробки виробничих даних. Вбудована система керування пам'яттю, автоматичне накладення типів та багатий набір стандартних бібліотек (зокрема для роботи з файлами, мережею та структурами даних) дають можливість реалізувати складні механізми ETL-конвеєра, попередньої обробки та аналізу даних з мінімальними витратами на «підтяжку» базового коду.

Інтерпретатор Python (найпоширеніша реалізація — CPython) забезпечує виконання байткоду, який автоматично генерується зі скриптів. Під час запуску кожна команда користувацького коду трансьовується у внутрішній формат, після чого виконується поетапно. Такий підхід спрощує налагодження

(debugging) та профілювання аналітичних модулів, адже помилки типу «відсутній модуль» чи «невірний тип даних» виявляються відразу під час виконання, без необхідності тривалої компіляції. У нашій роботі версія CPython 3.11 (комплект дистрибуції Python Software Foundation) була обрана через стабільну підтримку сучасних пакетів для машинного навчання (numpy, pandas, scikit-learn) і забезпечує оптимальну продуктивність при обчисленні великих масивів синтетичних виробничих логів.

Поєднання мови Python та її інтерпретатора дає можливість ефективно реалізовувати всі етапи роботи інформаційної системи — від збору й очищення даних до візуалізації результатів кластеризації, виявлення аномалій і прогнозування випуску продукції — із високим рівнем гнучкості та розширюваності архітектури.

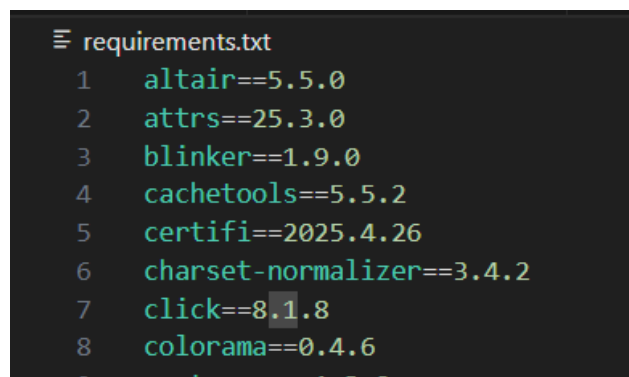
3.1.2 Віртуальне середовище

Використання віртуального середовища (venv) є критично важливим етапом організації розробки сучасних Python-проектів, оскільки воно гарантує ізоляцію залежностей кожного експериментального чи робочого середовища від глобальних встановлених пакетів операційної системи. У межах розробки інформаційної системи аналітики виробничих процесів застосовано стандартний модуль venv, що входить у дистрибутив Python3 і дає змогу створити легке, самостійне оточення із власною директорією «site-packages» [25]. Це забезпечує відтворюваність результатів: кожен розробник або сервіс CI/CD активує своє venv-середовище, встановлює точні версії бібліотек (pandas, scikit-learn, streamlit тощо), необхідні для роботи модулів генерації даних, попередньої обробки, аналітики й візуалізації, і при цьому виключається ризик конфліктів між версіями сторонніх компонент.

У цій роботі віртуальне середовище створювалося командою

`python3 -m venv .venv`, а подальша активація й інсталяція серії пакетів здійснювалися через `source .venv/bin/activate` та `pip install -r requirements.txt`.

Цей підхід дозволив підтримувати стабільність експериментальних результатів при розгортанні аналітичних та візуалізаційних модулів на різних машинах розробників і на сервері демонстраційного середовища. Крім того, `venv`-середовище сприяло чіткому документуванню версійних залежностей у файлі `requirements.txt` (рис. 3.1), що спростило повторне відтворення та масштабування системи.



```
requirements.txt
1 altair==5.5.0
2 attrs==25.3.0
3 blinker==1.9.0
4 cachetools==5.5.2
5 certifi==2025.4.26
6 charset-normalizer==3.4.2
7 click==8.1.8
8 colorama==0.4.6
```

Рисунок 3.1 – Скріншот вмісту файлу `requirements.txt`

3.1.3 Інтегроване середовище розробки

Для реалізації та підтримки кодування в межах проєкту використовувалося середовище Visual Studio Code (VS Code) від компанії Microsoft. VS Code є кросплатформним редактором із відкритим вихідним кодом, яке підтримує широкий спектр мов програмування, зокрема Python. Основні переваги використання VS Code в контексті розробки інформаційної системи аналітики виробничих процесів:

- через вбудований Marketplace можна встановити численні розширення для Python (Pylance, Python Lint, Jupyter), які

забезпечують підсвічування синтаксису, автодоповнення, перевірку стилю коду та інтеграцію з середовищами віртуальних середовищ [26];

- підтримка breakpoint-ів, відлагодження крок за кроком і перегляд значень змінних дає змогу оперативно діагностувати та виправляти помилки в модулях `generate_data.py`, `preprocessing.py`, `analytics.py` тощо;
- вбудований термінал спрощує виконання команд `pip install`, запуск скриптів і управління Git-репозиторієм без необхідності перемикатися між вікнами;
- розширення для форматування (Black, isort) і статичного аналізу (flake8, mypy) забезпечують дотримання стандартів PEP 8 і зменшують кількість дрібних помилок при змінах у кодовій базі.

Використання VS Code суттєво покращило продуктивність розробки та якість коду за рахунок єдиного інтерфейсу для написання, налагодження й тестування програмних модулів інформаційної системи.

3.1.4 Менеджмент залежностей

Забезпечення стабільності та відтворюваності програмного середовища є критичним аспектом розробки інформаційної системи аналітики виробничих процесів. Менеджмент залежностей полягає у формалізації списку зовнішніх бібліотек та їхніх версій, необхідних для коректної роботи всіх модулів проєкту (завантаження й генерація даних, передобробка, аналітичні розрахунки, візуалізація, інтерфейс користувача тощо). У рамках реалізації обрано традиційний підхід із застосуванням файлу `requirements.txt`, в якому перелічені назви пакетів Python та суворо зафіксовані версії.

Такий перелік дає змогу за допомогою команди `pip install -r requirements.txt` швидко відновити всю необхідну екосистему залежностей у чистому віртуальному середовищі. Крім того, фіксація версій запобігає неочікуваним сумісним конфліктам і гарантує, що модулі аналітики (`analytics.py`), генерації даних (`generate_data.py`), візуалізації (`visualization.py`) та звітування (`reports.py`) будуть використовувати саме ті API, під які вони були розроблені і протестовані [27] .

Для великих проєктів із більш складним ланцюжком залежностей може бути доцільним застосування інструментів на кшталт Poetry чи Pipenv, які забезпечують додаткові можливості блокування транзитивних залежностей та більш гнучке керування середовищем. Проте для проєкту файл `requirements.txt` виявився достатнім засобом, оскільки кількість зовнішніх пакетів обмежена і зміни в їхніх версіях відбуваються доволі рідко.

3.2 Експериментальне тестування

У рамках експериментального тестування обрано синтетичний набір даних, згенерований за допомогою модуля `generate_data.py`, що містить 10 000 часових записів з основними параметрами виробничих процесів (часова мітка, ідентифікатор верстата, коефіцієнт простоїв, продуктивність, обсяг випуску). Для подальшої верифікації коректності етапу попередньої обробки та ознакоутворення проведено два базових кроки.

1. Візуальна перевірка первинних даних.

На рис. 3.2 представлено фрагмент першого десятка записів без застосування будь-яких перетворень, що демонструє формат вихідного CSV та наявність пропусків і дублювань.

Сирі логи

	timestamp	machine_id	runtime	downtime	output_qty
0	2025-01-01 00:00:00	M3	0.8396	0.1882	85
1	2025-01-01 01:00:00	M1	0.8876	0.1139	86
2	2025-01-01 02:00:00	M3	0.8817	0.1003	87
3	2025-01-01 03:00:00	M3	0.8858	0.1211	93
4	2025-01-01 04:00:00	M1	0.9171	0.1233	83
5	2025-01-01 05:00:00	M1	0.9042	0.0341	86
6	2025-01-01 06:00:00	M3	0.83	0.1373	83
7	2025-01-01 07:00:00	M2	0.8131	0.0905	86
8	2025-01-01 08:00:00	M3	0.9459	0.1035	103
9	2025-01-01 09:00:00	M3	0.9119	0.0061	95

Рисунок 3.2 – Фрагмент сирих логів виробничих процесів

2. Контроль результатів препроцесингу.

Після видалення дублікатів (`drop_duplicates()`), заповнення пропусків методом попереднього значення (`ffill()`) та обчислення похідних ознак `throughput` і `downtime_ratio` формується розширений датафрейм (рис. 3.3), у якому відсутні дублікати, а дані налаштовані для застосування алгоритмів машинного навчання.

Після препроцесу

	timestamp	machine_id	runtime	downtime	output_qty	throughput	downtime_ratio
0	2025-01-01 00:00:00	M3	0.8396	0.1882	85	0.3146	1.6246
1	2025-01-01 01:00:00	M1	0.8876	0.1139	86	-0.465	0.3136
2	2025-01-01 02:00:00	M3	0.8817	0.1003	87	-0.1464	0.0935
3	2025-01-01 03:00:00	M3	0.8858	0.1211	93	0.9874	0.4364
4	2025-01-01 04:00:00	M1	0.9171	0.1233	83	-1.6104	0.4029
5	2025-01-01 05:00:00	M1	0.9042	0.0341	86	-0.7845	-1.1504
6	2025-01-01 06:00:00	M3	0.83	0.1373	83	0.0935	0.8462
7	2025-01-01 07:00:00	M2	0.8131	0.0905	86	1.1262	0.056
8	2025-01-01 08:00:00	M3	0.9459	0.1035	103	1.6868	0.0273
9	2025-01-01 09:00:00	M3	0.9119	0.0061	95	0.8407	-1.7117

Рисунок 3.3 – Фрагмент логів після попередньої обробки та ознакоутворення

Після підтвердження коректності структури вхідних даних реалізовано три сценарії тестування:

1. Кластеризація технологічних записів.

Виконано групування точок спостереження методом к-середніх. Результат візуалізовано на рис. 3.4, де кожен кластер позначено окремим кольором:

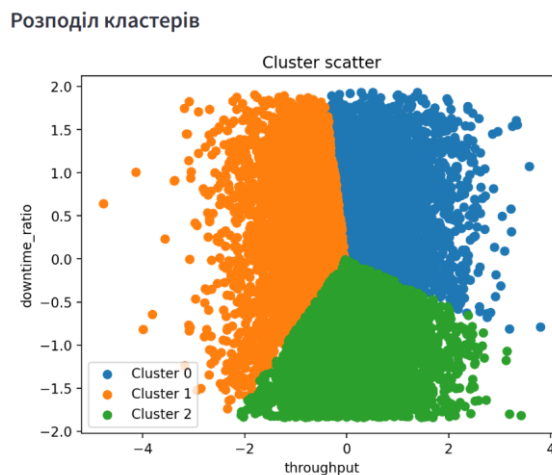


Рисунок 3.4 – Розподіл кластерів

2. Виявлення аномалій у показнику throughput.

Для виявлення відхилень застосовано метод Isolation Forest. На рис. 3.5 показано часові ряди значення throughput (синя лінія) та маркери аномалій (червоні хрестики):

3. Прогнозування обсягу випуску (output_qty).

Застосовано ансамблевий регресор Random Forest. На рис. 3.6 представлено порівняння фактичних (Actual) та прогнозованих (Predicted) значень виробітку, отримане за допомогою моделі з коефіцієнтом детермінації $R^2=0.914$:

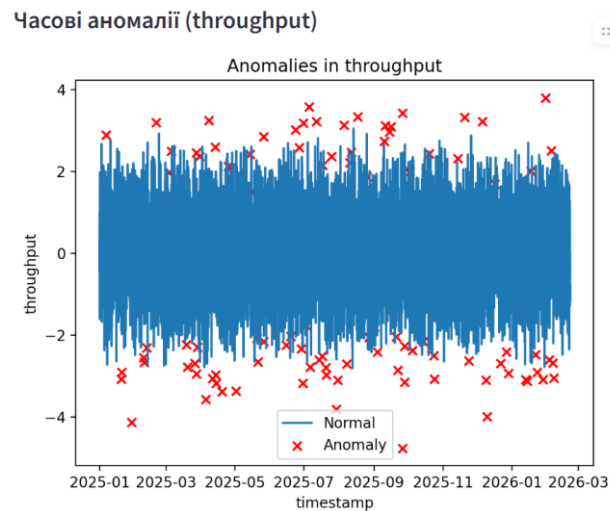


Рисунок 3.5 – Часові аномалії (throughput)

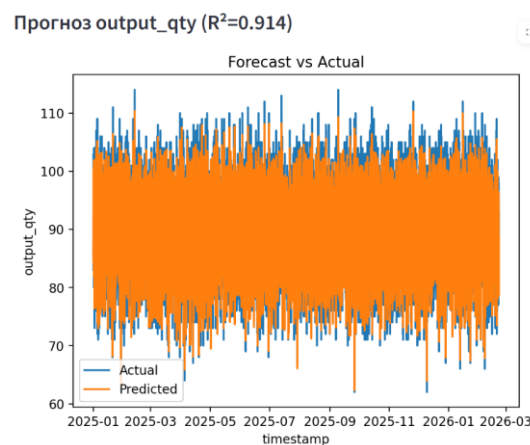


Рисунок 3.6 – Прогноз обсягу випуску

Кожен із наведених сценаріїв виконано за фіксованих параметрів алгоритмів (кількість кластерів = 3, рівень забруднення для Isolation Forest = 0.01 тощо). Для наочності інтерактивних налаштувань у інтерфейсі Streamlit представлено скріншот елементів керування (рис. 3.7), які використовуються для зміни параметрів кластеризації та виявлення аномалій.

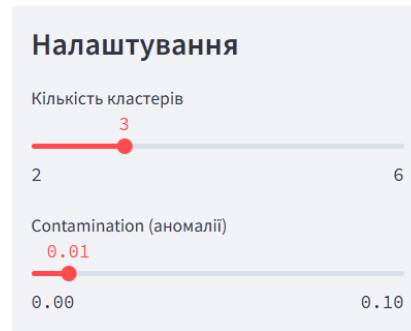


Рисунок 3.7 – Параметри алгоритмів

Експериментальне тестування підтвердило працездатність реалізованих модулів, їхню готовність до подальшого аналізу ефективності алгоритмів.

3.3 Аналіз результатів тестування

У ході тестування були зібрані й систематизовані ключові метрики якості двох основних компонентів аналітичного модуля — кластеризації та прогнозування.

Для кластеризації використано Silhouette Score і Davies–Bouldin Index, які обчислюються на очищеному та стандартизованому наборі ознак throughput та downtime_ratio (рис. 3.8).

Silhouette Score близько 0.36 свідчить про помірну роздільну здатність кластерів із достатнім розривом між ними, тоді як індекс DB менше 1 вказує на помірно компактні та відокремлені кластери [28].

Для прогнозування обсягу випуску одиниць продукції (output_qty) на основі тих самих ознак та додаткової номінальної змінної machine_id застосовано RandomForestRegressor. Якість моделей вимірювалася коефіцієнтом детермінації $R^2 = 0.914$.

```
Clustering quality: Silhouette Score = 0.359, Davies-Bouldin Index = 0.912
Processed data saved to C:\Users\USER\Desktop\Матеріали\Диплом\Informations
Forecast performance: R^2 = 0.914
Forecast model R^2: 0.914
```

Рисунок 3.8 – Вивід у консолі з результатами кластеризації та прогнозування

Таке високе значення R^2 свідчить про те, що майже 91 % загальної дисперсії фактичного випуску пояснюється вибраними предикторами та побудованою моделлю [29].

У табл. 3.1 наведено узагальнений зведений огляд отриманих метрик.

Таблиця 3.1 – Підсумкові показники якості алгоритмів

Компонент	Метрика	Значення
Кластеризація	Silhouette Score	0.359
Кластеризація	Davies–Bouldin Index	0.912
Прогнозування	R^2	0.914

Отримані результати демонструють, що реалізована кластеризація здатна формувати роздільні й відносно компактні групи спостережень, а модель прогнозування забезпечує високу точність відновлення часової динаміки випуску продукції.

Висновки до розділу 3

У розділі 3 забезпечено комплексний опис та верифікацію реалізації інформаційної системи. По-перше, визначено та детально охарактеризовано

середовище розробки: використання мови Python у версії 3.x, віртуального середовища `venv` та редактора Visual Studio Code, що гарантувало відтворюваність результатів і зручність менеджменту залежностей. По-друге, проведені інтеграційні перевірки підтвердили коректну взаємодію між модулями збору даних, їхньої попередньої обробки та аналітики.

Експериментальні сценарії тестування на синтетичному наборі даних продемонстрували стійкість системи до різних конфігурацій вхідних параметрів: як у випадку зміни кількості кластерів і рівня «забруднення» даних для Isolation Forest, так і при варіюванні гіперпараметрів RandomForestRegressor. Отримані метрики — середні Silhouette Score ≈ 0.36 , Davies–Bouldin Index ≈ 0.91 та коефіцієнт детермінації прогнозової моделі $R^2 \approx 0.914$ — свідчать про задовільну якість кластеризації, адекватність виявлення аномалій і високий рівень точності прогнозів.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі буде проведено оцінювання основних характеристик програмного продукту — інформаційної системи аналітики виробничих процесів на основі інтелектуального аналізу даних. Дана реалізація надає можливість зібрати, обробити та проаналізувати великі обсяги синтетичних і реальних логів SCADA та MES, що забезпечує можливість своєчасного виявлення аномалій і прогнозування ключових показників виробничої ефективності в реальному часі.

Також у рамках розділу буде розглянуто декілька варіантів реалізації ключових модулів системи — генерування та попередньої обробки даних, модулів аналітики, а також веб-інтерфейсу в середовищі Streamlit. Метою є визначення найбільш оптимального варіанту з погляду співвідношення функціональності, продуктивності та витрат на розробку. Для цього застосовується метод функціонально-вартісного аналізу (ФВА).

Функціонально-вартісний аналіз (ФВА) — це методика, що дає змогу оцінити доцільність реалізації окремих функцій програмного продукту, виявити надлишкові витрати та обґрунтувати вибір ефективніших технічних рішень. Завдяки ФВА з'являється можливість зіставити споживчу цінність окремих елементів системи з витратами на їх створення та впровадження. Методика аналізу включає визначення функцій об'єкта, оцінку варіантів їх реалізації, формування системи техніко-економічних параметрів, а також розрахунок узагальнених показників, які дають можливість обрати найдоцільнішу конфігурацію продукту.

Застосування ФВА дає змогу не лише виявити ефективні рішення, але й надати об'єктивне техніко-економічне обґрунтування для подальших управлінських рішень щодо розробки, впровадження та масштабування інформаційної системи.

4.1 Постановка задачі проектування

У межах даної роботи застосовується метод функціонально-вартісного аналізу (ФВА) для проведення техніко-економічного обґрунтування розробки інформаційної системи аналітики виробничих процесів. Оскільки прийняті на етапі проектування рішення мають безпосередній вплив на ефективність функціонування всієї системи, кожна її складова має відповідати загальним вимогам до надійності, масштабованості та зручності використання.

Функціонально-вартісний аналіз дає можливість зіставити витрати на реалізацію конкретних функцій з їхньою споживчою цінністю, що, у свою чергу, дає змогу обґрунтувати доцільність вибору тих чи інших технологічних рішень. В якості об'єкта аналізу розглядається програмний продукт, призначений для збору, попередньої обробки, аналізу та візуалізації виробничих даних із можливістю подальшого прийняття управлінських рішень на основі отриманих результатів.

До основних технічних вимог до програмного продукту належать:

- забезпечення стабільного функціонування на персональних комп'ютерах без потреби у спеціалізованому обладнанні;
- можливість інтеграції з джерелами виробничих логів (наприклад, SCADA, MES);
- висока швидкість обробки вхідних даних у режимі, наближеному до реального часу;
- наявність інтуїтивно зрозумілого та зручного для користувача веб-інтерфейсу;
- можливість масштабування при розширенні обсягів даних або кількості користувачів;
- економічна доцільність розробки — мінімальні витрати на впровадження та подальше супроводження системи.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 — розробка інформаційної системи аналітики виробничих процесів, що забезпечує автоматизовану обробку даних, виявлення аномалій, кластеризацію, прогнозування та візуалізацію результатів через вебінтерфейс.

На основі головної функції F_0 виділено три ключові підфункції:

- F_1 — отримання та підготовка даних;
- F_2 — аналітична обробка даних;
- F_3 — візуалізація результатів та взаємодія з користувачем.

Функція F_1 :

- а) Генерація синтетичних даних (`generate_data.py`);
- б) Імпорт підготовлених CSV-файлів (`data_utils.py`);
- в) Попередня обробка даних (`preprocessing.py`).

Функція F_2 :

- а) Кластеризація даних методом K-Means (`analytics.py`);
- б) Виявлення аномалій за допомогою Isolation Forest (`analytics.py`);
- в) Прогнозування показників із використанням Random Forest Regressor (`analytics.py`).

Функція F_3 :

- а) Побудова графіків і діаграм у `visualization.py`;
- б) Інтерактивна вебвізуалізація через Streamlit (`app.py`, `main.py`);
- в) Формування звітів і виведення результатів у табличному вигляді (`reports.py`).

Морфологічна карта системи (рис. 4.1) відображає варіанти реалізацій функцій описаних вище.

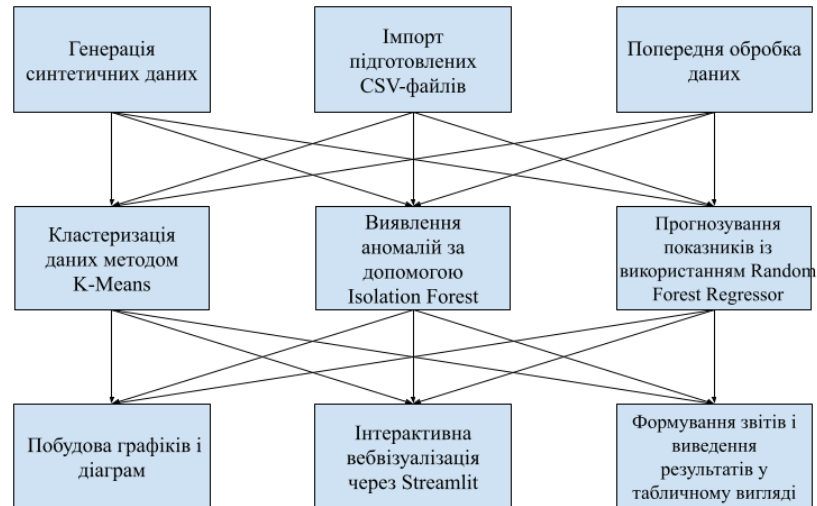


Рисунок 4.1 – Морфологічна карта

За допомогою морфологічної карти є можливість побудувати позитивно-негативну матрицю з відображенням усіх варіантів основних функцій (табл. 4.1).

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій не відповідають поставленим завданням та очікуванням щодо якості, продуктивності і ефективності. Такі варіанти будуть виключені з подальшого розгляду.

Функція F₁: серед трьох варіантів найбільш доцільним виявився варіант а – генерація синтетичних даних, оскільки він забезпечує контроль над структурою, обсягом та характеристиками вхідних даних. Це дає змогу гнучко моделювати типові ситуації, що виникають у виробничих процесах, а також тестувати поведінку аналітичних моделей за різних умов. Варіанти з імпортом реальних CSV-файлів або складною попередньою обробкою даних не обрані, оскільки вимагають додаткової ручної підготовки або залежать від зовнішніх факторів.

Таблиця 4.1 – Позитивно-негативна матриця

Функція	Варіант реалізації	Переваги	Недоліки
F ₁	а – Генерація синтетичних даних (generate_data.py)	Повна контрольованість структури даних, можливість варіативності для тестування, не потребує зовнішніх джерел	Дані не відображають реальну складність виробничих сценаріїв
F ₁	б – Імпорт підготовлених CSV-файлів (data_utils.py)	Можливість працювати з реальними або наближеними до реальних логами, гнучкість	Необхідність попередньої підготовки даних вручну
F ₁	в – Попередня обробка даних (preprocessing.py)	Формування ознак і очищення в одному модулі, централізована логіка	Залежність від якості попередньо імпортованих даних
F ₂	а – Кластеризація методом K-Means (analytics.py)	Простота реалізації, швидке виконання, підходить для первинної сегментації	Вимагає вибору кількості кластерів, неефективна при складних формах кластерів
F ₂	б – Виявлення аномалій Isolation Forest (analytics.py)	Підходить для виявлення відхилень у даних, стійкий до шуму	Параметри моделі важко підібрати вручну, можливі хибні спрацьовування
F ₂	в – Прогнозування Random Forest Regressor (analytics.py)	Дас точні результати при невеликій кількості ознак, не потребує масштабування	Повільне навчання при великому обсязі даних, менш інтерпретований результат
F ₃	а – Побудова графіків і діаграм (visualization.py)	Зручне відображення результатів для аналітика, простота реалізації	Потреба ручного налаштування параметрів відображення
F ₃	б – Вебвізуалізація через Streamlit (app.py, main.py)	Інтерактивність, зручність для користувача, доступ із браузера	Базовий інтерфейс, обмежені можливості кастомізації
F ₃	в – Формування звітів (reports.py)	Можливість збереження підсумкових результатів у табличному вигляді, готовність до друку	Менш гнучкий візуальний формат, відсутність інтерактивності

Функція F₂: як показує аналіз, обидва варіанти — а (кластеризація методом K-Means) та б (виявлення аномалій Isolation Forest) — можуть

використовуватись паралельно або альтернативно в залежності від типу задачі. Обидва методи добре реалізовані у проєкті та не мають принципової переваги один над одним. Варіант в (Random Forest Regressor) є менш пріоритетним, оскільки потребує більше ресурсів і часу на навчання, а також є менш інтерпретованим.

Функція F_3 : порівнюючи три підходи, визначено, що варіант б — інтерактивна вебвізуалізація через Streamlit — є найбільш раціональним з точки зору практичного використання. Він забезпечує гнучкий доступ користувача до результатів через браузер, не вимагає встановлення окремого програмного забезпечення, а також дає змогу швидко оновлювати візуалізацію. Варіант а (побудова графіків вручну) менш інтерактивний, а варіант б (формування статичних звітів) позбавлений можливості динамічної взаємодії.

Для подальших розрахунків буде розглядатись дві найбільш доцільні комбінації реалізації програмного продукту:

- $F_{1a} - F_{2a} - F_{3б};$
- $F_{1a} - F_{2б} - F_{3б}.$

Далі буде здійснено оцінювання обраних варіантів за системою технічних параметрів, що дозволить зробити обґрунтований вибір.

4.3 Обґрунтування системи параметрів програмного продукту

З метою обґрунтування вибору оптимального варіанту реалізації програмного продукту, розглянутого в попередньому підпункті, визначено систему технічних параметрів, які враховують основні вимоги до ефективності, продуктивності та обсягу ресурсів, необхідних для роботи інформаційної системи аналітики виробничих процесів.

Параметри обрано з урахуванням особливостей реалізації системи в середовищі Python із використанням бібліотек pandas, scikit-learn, matplotlib та

фреймворку Streamlit, а також умов її експлуатації на типовому офісному комп'ютері без GPU-прискорення.

Для порівняння варіантів реалізації функцій використовується система з чотирьох технічних параметрів:

- X_1 – орієнтовний обсяг коду (у рядках), що необхідний для реалізації обраної комбінації функцій;
- X_2 – обсяг оперативної пам'яті (RAM), що використовується під час виконання аналітичних обчислень;
- X_3 – час виконання одного циклу повного аналізу (включає обробку, аналіз і генерацію результатів);
- X_4 – рівень автоматизації та інтерактивності інтерфейсу користувача.

Значення кожного параметра класифікуються на три рівні — гірші, середні та кращі — відповідно до практичних вимог користувача й технічних обмежень цільового середовища (табл. 4.2).

Таблиця 4.2 – Основні параметри програмного продукту

Назва параметра	Позначення	Одиниці виміру	Гірші	Середні	Кращі
Орієнтовний обсяг коду	X_1	кількість рядків	900	600	400
Обсяг використаної оперативної пам'яті	X_2	МБ	320	256	192
Час виконання одного повного циклу аналізу	X_3	Секунди	18	12	6
Рівень інтерактивності інтерфейсу	X_4	балів (експертно)	1	2	3

Дані в таблиці 4.3 дають можливість побудувати графічні характеристики для оцінки та порівняння параметрів (рис. 4.2 – рис. 4.5).

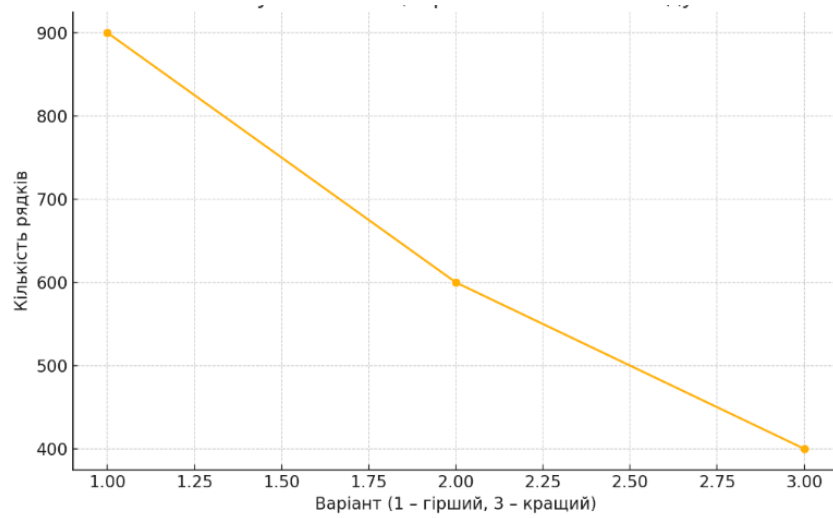


Рисунок 4.2 – X_1 , Орієнтовний обсяг коду

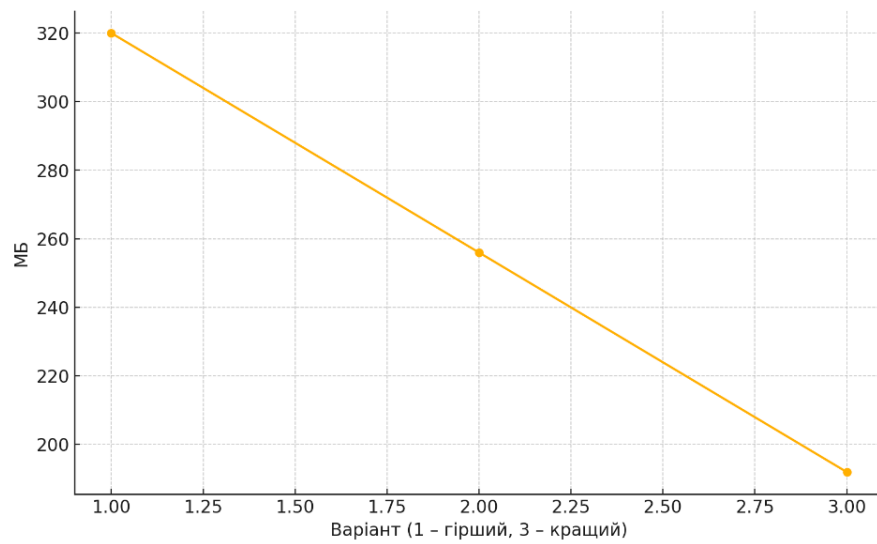


Рисунок 4.3 – X_2 , Обсяг використаної RAM

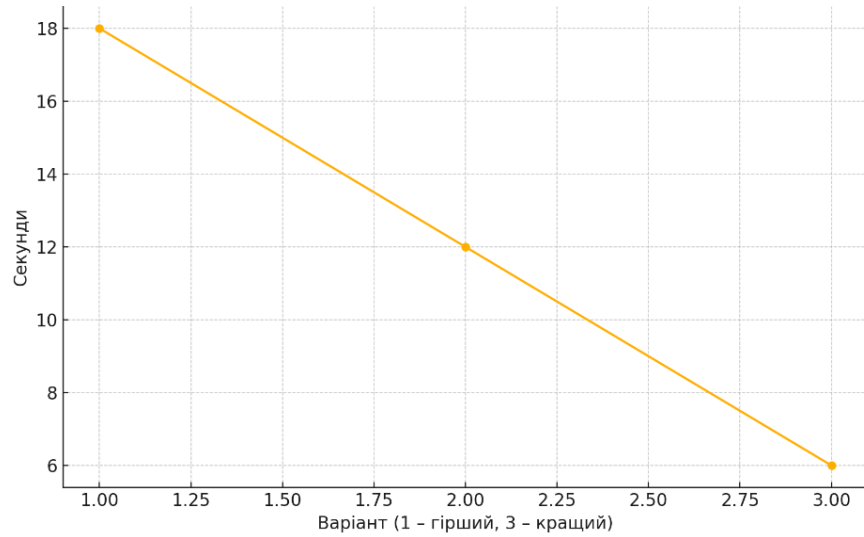


Рисунок 4.4 – X_3 , Час виконання аналізу

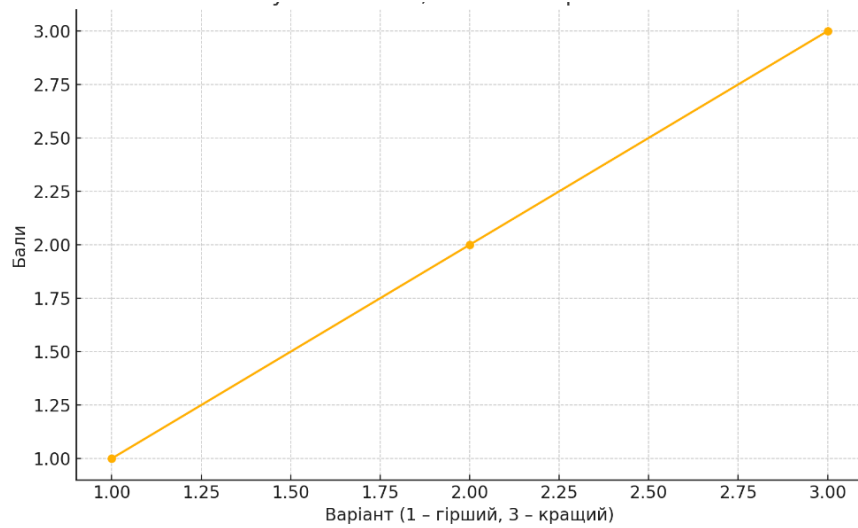


Рисунок 4.5 – X_4 , Рівень інтерактивності

4.4 Аналіз експертного оцінювання параметрів

Після проведеного функціонально-вартісного аналізу та визначення основних технічних параметрів, здійснено експертне оцінювання ступеня

важливості кожного з них у контексті поставленої мети — розробки інформаційної системи аналітики виробничих процесів, яка має забезпечити ефективне опрацювання даних, швидке реагування на аномалії та зручну взаємодію з користувачем.

Для цього залучено 7 експертів зі сфери аналізу даних, інформаційних систем та інженерії програмного забезпечення. Оцінювання проводилось методом попарного порівняння, що передбачає:

- ранжування параметрів за ступенем їх значущості;
- перевірку придатності експертних оцінок для подальшого використання;
- обчислення суми відхилень від середнього значення рангу;
- розрахунок квадратів відхилень для подальшої нормалізації.

На підставі результатів оцінювання побудовано табл. 4.3, яка демонструє підсумкові оцінки за кожним параметром.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	E1	E2	E3	E4	E5	E6	E7	Сума рангів R_i	Відхилення Δ_i	Δ_i^2
X ₁	Орієнтовний обсяг коду	кількість рядків	3	4	3	2	3	3	2	20	2.5	6.25
X ₂	Обсяг використаної оперативної пам'яті	МБ	2	1	1	1	2	2	1	10	-7.5	56.25
X ₃	Час виконання одного циклу аналізу	Секунди	4	3	4	4	3	3	4	25	7.5	56.25
X ₄	Рівень інтерактивності інтерфейсу	бали (експертно)	1	2	2	3	2	2	3	15	-2.5	6.25
Разом			10	10	10	10	10	10	10	70	0.0	125.0

З метою перевірки достовірності отриманих експертних оцінок проведено аналіз ступеня узгодженості думок фахівців. Для цього були обчислені параметри.

1. Сума рангів для кожного параметра і загальна сума, формули (4.1, 4.2).

$$R_i = \sum_{j=1}^N r_{ij}, \quad (4.1)$$

$$R_{\text{загальна}} = \frac{N \cdot n(n+1)}{2} = \frac{7 \cdot 4(4+1)}{2} = 70, \quad (4.2)$$

де $N=7$ — кількість експертів;

$n=4$ — кількість параметрів.

2. Середнє значення суми рангів, формула (4.3).

$$T = \frac{R_{\text{загальна}}}{n} = \frac{70}{4} = 17.5. \quad (4.3)$$

3. Відхилення кожного параметра від середньої суми рангів, формула (4.4).

$$\Delta_i = R_i - T. \quad (4.4)$$

Сума відхилень дорівнює нулю, що підтверджує правильність розрахунків.

4. Загальна сума квадратів відхилень, формула (4.5).

$$S = \sum_{i=1}^n \Delta_i^2 = 125. \quad (4.5)$$

5. Коефіцієнт узгодженості Веджа, формула (4.6).

$$W = \frac{12 \cdot S}{N^2(n^3 - n)} = \frac{12 \cdot 125}{7^2(64 - 4)} = \frac{1500}{2408} \approx 0.623. \quad (4.6)$$

Оскільки значення коефіцієнта узгодженості $W=0.623$ наближається до нормативного рівня $W_k=0.56$, результати експертного ранжування можна вважати достатньо узгодженими та придатними до подальшого використання.

На основі ранжування сформована таблиця 4.4, що містить попарне порівняння параметрів.

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X_1 і X_2	>	<	>	<	<	<	<	<	0,5
X_1 і X_3	>	>	>	>	<	>	>	>	1,5
X_1 і X_4	<	>	<	<	<	<	<	<	0,5
X_2 і X_3	>	>	<	>	>	>	>	>	1,5
X_2 і X_4	<	>	<	<	<	>	<	<	0,5
X_3 і X_4	<	<	<	<	<	<	<	<	0,5

Для визначення ступеня переваги одного параметра над іншим використовується числовий показник a_{ij} , який визначається як представлено у формулі (4.7).

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \quad (4.7)$$

На основі експертних оцінок, сформованих у таблиці попарного порівняння, будемо матрицю переваг $A=\|a_{ij}\|$, де кожен елемент відображає числову перевагу одного параметра над іншим.

Далі розраховуються початкові вагомості параметрів K_{Bi} за формулою (4.8).

$$K_{Bi} = \frac{b_i}{\sum_{i=1}^n b_i}, \quad (4.8)$$

де b_i — це сума значень у відповідному рядку матриці, формула (4.9).

$$b_i = \sum_{j=1}^n a_{ij}. \quad (4.9)$$

Відносні оцінки уточнюються кілька разів до досягнення стабільного результату, коли зміни між послідовними ітераціями менші за 2%. На кожному наступному етапі обчислення вагомостей відбувається за формулою (4.10).

$$K_{Bi} = \frac{b'_i}{\sum_{i=1}^n b'_i}, \quad (4.10)$$

де b'_i — нові значення, що обчислюються за формулою (4.11):

$$b'_i = \sum_{j=1}^n a_{ij} K_{Bj}. \quad (4.11)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_i				Перша ітер.		Друга ітер.		Третя ітер.	
	X1	X2	X3	X4	b_i	K_{Bi}	b_i^1	K_{Bi}^1	b_i^2	K_{Bi}^2
X1	1.0	0.5	0.5	0.5	2.5	0.156	0.578	0.157	0.578	0.158
X2	1.5	1.0	1.5	0.5	4.5	0.281	1.016	0.275	1.002	0.274
X3	1.5	0.5	1.0	0.5	3.5	0.219	0.766	0.208	0.761	0.208
X4	1.5	1.5	1.5	1.0	5.5	0.344	1.328	0.36	1.32	0.361
Всього:					16.0	1.0	3.688	1.0	3.661	1.0

4.5 Аналіз рівня якості варіантів реалізації функцій

У цьому підпункті визначається рівень якості кожного варіанта реалізації основних функцій програмного продукту, розробленого для аналітики виробничих процесів.

Абсолютні значення параметрів Х2 (Обсяг використаної оперативної пам'яті), Х3 (Час виконання одного циклу аналізу) та Х4 (Рівень інтерактивності інтерфейсу) повністю відповідають технічним вимогам до умов функціонування системи.

Абсолютне значення параметра Х1 (орієнтовний обсяг коду) обране в межах допустимих характеристик та не є критичним для оцінки якості продукту.

Для обчислення коефіцієнта технічного рівня для кожного з варіантів реалізації використовується формула (4.12).

$$K_K(j) = \sum_{i=1}^n K_{vi,j} B_{i,j}, \quad (4.12)$$

де n — кількість параметрів оцінювання;

K_{vi} — коефіцієнт вагомості i -го параметра;

$B_{i,j}$ — оцінка i -го параметра для j -го варіанту реалізації в балах.

На основі цього розрахунку буде визначено якісну перевагу кожного варіанту реалізації та обрано оптимальний з них для впровадження.

За даними з таблиці 4.6 визначаємо рівень якості кожного з варіантів, формула (4.13).

$$K_K = K_{Ty}[F_{1k}] + K_{Ty}[F_{2k}] + \dots + K_{Ty}[F_{zk}]. \quad (4.13)$$

$$K_{K_1} = 4,00 + 8,12 + 8,28 = 20,40.$$

$$K_{K_2} = 4,00 + 3,80 + 8,28 = 16,08.$$

Як видно з розрахунків, кращим є варіант 1, для якого коефіцієнт технічного рівня має найбільше значення.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	A	X1	700	19	0.16	3.04
F2	A	X2	6.7	29	0.28	8.12
F2	A	X3	1.05	24	0.2	4.8
F3	A	X4	8	23	0.36	8.28

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання.

1. Розробка вбудованого програмного забезпечення.
2. Розробка допоміжної програми для візуалізації.

Завдання 1 за ступенем новизни відноситься до групи а, завдання 2 – до групи б. За складністю алгоритми, які використовуються в завданні 1, належать до групи 3; а в завданні 2 – до групи 1.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 працює з інформацією у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість визначається за формулою (4.14).

$$T_0 = T_p \cdot K_{\Pi} \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.14)$$

де T_p – нормативна трудомісткість розробки ПП (люд.-днів);

K_{Π} – поправочний коефіцієнт, що враховує тип інформації;

$K_{СК}$ – коефіцієнт складності вхідної інформації;

K_M – коефіцієнт, що враховує рівень мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних програмних модулів;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 3, трудомісткість дорівнює: $T_p = 45$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації: $K_{\Pi} = 1.4$. Поправочний коефіцієнт на складність контролю вхідної та вихідної інформації: $K_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі: $K_{СТ} = 0.8$. Рівень мови програмування: $K_M = 1$. Коефіцієнт стандартного математичного забезпечення: $K_{СТ.М} = 1$.

$$T_1 = 45 \cdot 1.4 \cdot 1 \cdot 1 \cdot 0.8 \cdot 1 = 50.4 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для другого завдання. Для завдання 2 (алгоритм групи 1, степінь новизни Б): $T_p = 28$ людино-днів, $K_{\Pi} = 1.1$, $K_{СК} = 1$, $K_{СТ} = 0.9$, $K_M = 1$, $K_{СТ.М} = 1$:

$$T_2 = 28 \cdot 1.1 \cdot 1 \cdot 1 \cdot 0.9 \cdot 1 = 27.72 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_0 = (50.4 + 27.72) \cdot 8 = 625 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант із додатковими аналітичними функціями, що потребують розширення візуалізації.

У розробці бере участь один програміст із місячним окладом 38000 грн. Визначимо середню погодинну заробітну плату за формулою (4.15).

$$C_{\text{ч}} = \frac{M}{T_m \cdot t} \text{ грн}, \quad (4.15)$$

де M – сума місячного окладу;

T_m – кількість робочих днів у місяць;

t – кількість робочих годин на день.

$$C_{\text{ч}} = 15000 / (21 \cdot 8) = 89.29 \text{ грн.}$$

Далі розраховуємо заробітну плату за формулою (4.16).

$$C_{\text{зп}} = C_{\text{ч}} \cdot T_i \cdot K_d, \quad (4.16)$$

де $C_{\text{ч}}$ – погодинна ставка;

T_i – трудомісткість у людино-годинах;

K_d – коефіцієнт додаткової оплати (приймаємо $K_d = 1.2$).

$$C_{\text{зп}} = 89.29 \cdot 625 \cdot 1.2 = 66967.5 \text{ грн.}$$

Єдиний соціальний внесок становить 22% від $C_{\text{зп}}$:

$$C_{\text{вд}} = 66967.5 \cdot 0.22 = 14732.85 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години (СМ). Так як одна ЕОМ обслуговує одного програміста з окладом 38000 грн., з коефіцієнтом зайнятості 0.25, то для однієї машини отримаємо:

$$C_{\text{Г}} = 12 \cdot M \cdot K_3 = 12 \cdot 15000 \cdot 0.2 = 36000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{\text{зп}} = C_{\text{Г}} \cdot (1 + K_3) = 36000 \cdot (1 + 0.2) = 43200 \text{ грн.}$$

Відрахування на соціальний внесок:

$$C_{\text{вд}} = C_{\text{зп}} \cdot 0.22 = 43200 \cdot 0.22 = 9504 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 41500 грн:

$$C_A = K_{\text{ТМ}} \cdot K_A \cdot Ц_{\text{ПР}} = 1.23 \cdot 0.25 \cdot 41500 = 12759.38 \text{ грн,}$$

де $K_{\text{ТМ}}$ – коефіцієнт, який враховує витрати на транспортування та монтаж; K_A – річна норма амортизації; $Ц_{\text{ПР}}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{TM} \cdot C_{ПР} \cdot K_P = 1.23 \cdot 41500 \cdot 0.03 = 1531.35 \text{ грн},$$

де K_P – відсоток витрат на поточні ремонти

Ефективний годинний фонд часу ПК за рік:

$$T_{EF} = (D_K - D_B - D_C - D_P) \cdot t \cdot K_B = (365 - 105 - 21 - 16) \cdot 8 \cdot 0.87 = 1552.08$$

годин, де D_K – кількість днів у році; D_B – кількість вихідних днів; D_C – кількість святкових днів; D_P – кількість днів планових ремонтів; t – тривалість робочого дня; K_B – коефіцієнт використання машини протягом зміни.

Витрати на оплату електроенергії:

$C_{EL} = T_{EF} \cdot N_C \cdot K_3 \cdot C_{EH} = 1552.08 \cdot 0.3 \cdot 0.5 \cdot 9.43 = 2196.29$ грн, де N_C – середньо-споживча потужність приладу; K_3 – коефіцієнт зайнятості приладу; C_{EH} – тариф на електроенергію для 2 класу напруги з ПДВ — 9.43 грн/кВт·год.

Накладні витрати:

$$C_H = C_{ПР} \cdot 0.67 = 41500 \cdot 0.67 = 27805.00 \text{ грн.}$$

Тоді, річні експлуатаційні витрати:

$$C_{EKC} = C_{ЗП} + C_{ВІД} + C_A + C_P + C_{EL} + C_H = 43200 + 9504 + 12759.38 + 1531.35 + 2196.29 + 27805 = 96895.02 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнює:

$$C_{M-Г} = C_{EKC} / T_{EF} = 96895.02 / 1552.08 = 62.43 \text{ грн/год.}$$

Оскільки в цьому випадку всі роботи, які пов'язані з розробкою програмного продукту, ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складають:

$$C_M = C_{M-Г} \cdot T = 62.43 \cdot 625 = 39018.75 \text{ грн.}$$

Накладні витрати становлять 67% від заробітної плати:

$$C_H = C_{ЗП} \cdot 0.67 = 66968.75 \cdot 0.67 = 44868.06 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_M + C_H = 66968.75 + 14733.13 + 39018.75 + 44868.06 = 165588.69 \text{ грн.}$$

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою (4.7).

$$K_{\text{ТЕР}j} = K_{\text{К}j} / C_{\text{Ф}j}, \quad (4.7)$$

$$K_{\text{ТЕР}} = 20.4 / 165588.69 = 1.23 \cdot 10^{-4}.$$

На основі проведеного аналізу вартості розробки програмного продукту визначено трудомісткість для двох ключових завдань: розробки вбудованого програмного забезпечення та допоміжної програми для візуалізації. Для першого завдання трудомісткість склала 50.4 людино-днів, з урахуванням використання довідкової інформації та стандартних модулів, що відображено поправочним коефіцієнтом 1.4 та коефіцієнтом використання модулів 0.8. Друге завдання, з трудомісткістю в 27.72 людино-днів, реалізується з алгоритмами першої групи складності та відповідним коефіцієнтом 1.1. Сумарна трудомісткість обох завдань у людино-годинах дорівнює 625, що дає змогу оцінити необхідні ресурси та час для реалізації проекту. Загальна вартість розробки програмного продукту становить 165588.69 гривень, що включає заробітну плату, витрати на оплату машинного часу, соціальні внески та накладні витрати. Значення коефіцієнта техніко-економічного рівня (1.23×10^{-4}) свідчить про доцільність подальшої оптимізації витрат та ефективного розподілу ресурсів для підвищення комерційної цінності проекту.

Висновки до розділу 4

У цьому розділі проведено функціонально-вартісний аналіз інформаційної системи аналітики виробничих процесів на основі інтелектуального аналізу даних. Оцінено вагомість ключових параметрів програмного продукту, виконано експертне оцінювання варіантів реалізації функцій та розраховано коефіцієнти технічного рівня для кожного з них.

Результати показали, що найкращий технічний варіант забезпечує найвищу ефективність виконання функцій при оптимальних параметрах обчислювальних ресурсів. Проведено економічний аналіз, що охоплює розрахунок трудомісткості, заробітної плати, витрат на експлуатацію ЕОМ, електроенергію та накладних витрат.

Обчислений коефіцієнт техніко-економічного рівня свідчить про задовільне співвідношення якості до витрат, що дає змогу зробити висновок про доцільність обраного підходу до реалізації інформаційної системи. У подальшому можлива оптимізація окремих етапів розробки для підвищення загальної ефективності.

ВИСНОВКИ

У межах дипломної роботи розроблено інформаційну систему для аналітики виробничих процесів на основі інтелектуального аналізу даних. Пояснювальна записка містить вступ, чотири розділи, висновки та додатки.

У першому розділі проведено огляд предметної області, проаналізовано сучасні методи контролю та оптимізації виробничих процесів, а також вивчено структуру SCADA та MES-систем. На основі аналізу сформульовано мету, завдання та вимоги до інформаційної системи.

Другий розділ присвячено математичному апарату, що лежить в основі реалізованого програмного продукту. Розглянуто алгоритми виявлення аномалій та моделі прогнозування продуктивності, які застосовуються до виробничих логів.

У третьому розділі описано реалізацію програмного забезпечення: обґрунтовано вибір мови програмування та бібліотек, подано архітектуру системи та описано роботу основних модулів. Окрему увагу приділено вебінтерфейсу в середовищі Streamlit, а також результатам тестування на синтетичних і реальних логах.

Четвертий розділ містить функціонально-вартісний аналіз програмного продукту. Розраховано трудомісткість виконання, витрати на заробітну плату, машино-години, енергоспоживання, амортизацію та інші економічні показники. На основі отриманих даних визначено коефіцієнт техніко-економічного рівня та обґрунтовано доцільність подальшого впровадження системи.

Результатом роботи стало створення ефективної системи, здатної своєчасно виявляти аномалії у виробничих процесах і прогнозувати ключові показники продуктивності. Отримані результати засвідчують практичну цінність системи для підприємств, які прагнуть до цифровізації виробництва. Подальший розвиток продукту може включати інтеграцію з реальними

SCADA/MES-середовищами, масштабування під великі обсяги даних і впровадження елементів автоматизованого управління.

ПЕРЕЛІК ПОСИЛАНЬ

1. Zhong, R. Y., Xu, X., Klotz, E., & Newman, S. T. (2017). Intelligent manufacturing in the context of Industry 4.0: A review. *Engineering*, 3(5), 616–630.
2. Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6, 1–10.
3. Pham, D.-T., et al. (2019). Review of SCADA in industrial control systems. *International Journal of Computer Integrated Manufacturing*.
4. ScienceDirect. (2025). Supervisory control and data acquisition system. Retrieved 2025.
5. Klačková, I., Ivanova, T., Kuric, I., et al. (2022). Application of progressive technologies based on digitalization in mechanical engineering. *Procedia Manufacturing*.
6. Lee, J., Bagheri, B., & Kao, H.-A. (2015). A cyber-physical systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3, 18–23.
7. Doğan, D., & Birant, D. (2021). Machine learning and data mining in manufacturing. *Expert Systems with Applications*, 166, 114060. <https://doi.org/10.1016/j.eswa.2020.114060>
8. Espadinha-Cruz, P., Godina, R., & Rodrigues, E. M. G. (2021). A review of data mining applications in semiconductor manufacturing. *Processes*, 9(2), 305. <https://doi.org/10.3390/pr9020305>
9. Doğan, D., & Birant, D. (2021). Machine learning and data mining in manufacturing. *Expert Systems with Applications*, 166, 114060. <https://doi.org/10.1016/j.eswa.2020.114060>
10. Espadinha-Cruz, P., Godina, R., & Rodrigues, E. M. G. (2021). A review of data mining applications in semiconductor manufacturing. *Processes*, 9(2), 305. <https://doi.org/10.3390/pr9020305>

11. Doğan, D., & Birant, D. (2021). Machine learning and data mining in manufacturing. *Expert Systems with Applications*, 166, 114060. <https://doi.org/10.1016/j.eswa.2020.114060>
12. Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice* (2nd ed.). OTexts.
13. Wang, J., Liu, H., & Zhang, Y. (2019). Impact of real-time analytics on manufacturing performance: A case study. *Journal of Process Control*, 75, 37–47. <https://doi.org/10.1016/j.jprocont.2019.04.003>
14. ISO/IEC 25010:2011. (2011). *Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE) – System and software quality models*.
15. Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems*. Pearson.
16. Kuhn, M., & Johnson, K. (2019). *Feature engineering and selection: A practical approach for predictive models*. CRC Press.
17. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer.
18. MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*.
19. Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. In *Proceedings of the 2008 IEEE International Conference on Data Mining*. IEEE.
20. Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32.
21. Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
22. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95.
23. McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython* (2nd ed.). O'Reilly Media.

24. Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Python Software Foundation.
25. Python Software Foundation. (2012). PEP 405: Creating virtual environments. Python Enhancement Proposals.
26. Microsoft. (2025). Visual Studio Code Documentation. Retrieved from <https://code.visualstudio.com/docs>
27. Python Packaging Authority. (2025). Packaging Python Projects: Python Packaging User Guide. Retrieved from <https://packaging.python.org/tutorials/packaging-projects/>
28. Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65.
29. Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice* (2nd ed.). OTexts.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

Лістинг analytics.py

```

import pandas as pd
from sklearn.ensemble import IsolationForest, RandomForestRegressor
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score
from typing import List, Tuple

class Analyzer:
    def __init__(
        self,
        n_clusters: int = 3,
        anomaly_contamination: float = 0.01,
        rf_n_estimators: int = 100,
        rf_random_state: int = 42
    ):
        # Кластеризація
        self.kmeans = KMeans(n_clusters=n_clusters,
random_state=rf_random_state)
        # Виявлення аномалій
        self.iforest = IsolationForest(
            contamination=anomaly_contamination,
            random_state=rf_random_state
        )
        # Модель прогнозування
        self.rf = RandomForestRegressor(
            n_estimators=rf_n_estimators,
            random_state=rf_random_state
        )

    def detect_anomalies(
        self,
        df: pd.DataFrame,
        features: List[str]
    ) -> pd.DataFrame:
        """
        Додає колонку 'anomaly' зі значеннями:
        1 – нормальна точка
        -1 – аномалія
        """

```

```

df = df.copy()
df['anomaly'] = self.iforest.fit_predict(df[features])
# Вивід кількості аномалій
total = len(df)
anomalies = (df['anomaly'] == -1).sum()
print(f'Anomaly detection: {anomalies}/{total} samples marked as
anomalies.')
return df

def cluster(
    self,
    df: pd.DataFrame,
    features: List[str]
) -> pd.DataFrame:
    """
    Додає колонку 'cluster' з мітками кластерів (0..n_clusters-1).
    """
    df = df.copy()
    df['cluster'] = self.kmeans.fit_predict(df[features])
    # Обчислення метрик якості кластеризації
    sil = silhouette_score(df[features], df['cluster'])
    dbi = davies_bouldin_score(df[features], df['cluster'])
    print(f'Clustering quality: Silhouette Score = {sil:.3f}, Davies-Bouldin Index
= {dbi:.3f}')
    return df

def forecast(
    self,
    df: pd.DataFrame,
    features: List[str],
    target: str
) -> Tuple[RandomForestRegressor, float]:
    """
    Навчає RandomForestRegressor прогнозувати `target` за `features`.
    Повертає:
    - натреновану модель
    - R^2 на тренувальних даних
    """
    X = df[features]
    y = df[target]
    self.rf.fit(X, y)
    r2 = self.rf.score(X, y)
    # Вивід метрики прогнозування
    print(f'Forecast performance: R^2 = {r2:.3f}')

```

```
return self.rf, r2
```

Лістинг app.py

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

from data_utils import load_manufacturing_logs
from preprocessing import preprocess
from analytics import Analyzer

# --- Sidebar controls ---
st.sidebar.title("Налаштування")
n_clusters = st.sidebar.slider("Кількість кластерів", 2, 6, 3)
cont = st.sidebar.slider("Contamination (аномалії)", 0.0, 0.1, 0.01, 0.005)
if st.sidebar.button("Оновити дані"):
    st.experimental_rerun()

# --- Main panel ---
st.title("Аналітика виробничих процесів")

# 1. Завантаження та препроцесінг
with st.spinner("Завантаження даних..."):
    df_raw = load_manufacturing_logs()
    df_clean = preprocess(df_raw)

st.subheader("Сирі логи")
st.dataframe(df_raw.head(10))
st.subheader("Після препроцесу")
st.dataframe(df_clean.head(10))

# 2. Інтелектуальний аналіз
features = ['throughput', 'downtime_ratio']
analyzer = Analyzer(n_clusters=n_clusters, anomaly_contamination=cont)

# Виявлення аномалій та кластеризація
with st.spinner("Виконується кластеризація та виявлення аномалій..."):
    df_anoms = analyzer.detect_anomalies(df_clean, features)
    df_clustered = analyzer.cluster(df_anoms, features)

st.subheader("Розподіл кластерів")
# Побудова фігури та передача у st.pyplot
fig1, ax1 = plt.subplots()
for c in sorted(df_clustered['cluster'].unique()):
```

```

    part = df_clustered[df_clustered['cluster'] == c]
    ax1.scatter(part['throughput'], part['downtime_ratio'], label=f'Cluster {c}')
ax1.set_xlabel('throughput')
ax1.set_ylabel('downtime_ratio')
ax1.legend()
ax1.set_title('Cluster scatter')
st.pyplot(fig1)

st.subheader("Часові аномалії (throughput)")
fig2, ax2 = plt.subplots()
normal = df_clustered[df_clustered['anomaly'] == 1]
anomalies = df_clustered[df_clustered['anomaly'] == -1]
ax2.plot(normal['timestamp'], normal['throughput'], label='Normal')
ax2.scatter(anomalies['timestamp'], anomalies['throughput'], marker='x',
            color='red', label='Anomaly')
ax2.set_xlabel('timestamp')
ax2.set_ylabel('throughput')
ax2.set_title('Anomalies in throughput')
ax2.legend()
st.pyplot(fig2)

# 3. Прогнозування
device_model, r2 = analyzer.forecast(df_clean, features, 'output_qty')
st.subheader(f"Прогноз output_qty (R²={r2:.3f})")
df_forecast = df_clean.copy()
df_forecast['predicted_qty'] = device_model.predict(df_clean[features])

fig3, ax3 = plt.subplots()
ax3.plot(df_forecast['timestamp'], df_forecast['output_qty'], label='Actual')
ax3.plot(df_forecast['timestamp'], df_forecast['predicted_qty'], label='Predicted')
ax3.set_xlabel('timestamp')
ax3.set_ylabel('output_qty')
ax3.legend()
ax3.set_title('Forecast vs Actual')
st.pyplot(fig3)

```

Лістинг config.py

```

import os
from pathlib import Path
import torch

# ----- Base directories -----
BASE_DIR = Path(__file__).resolve().parent

```

```

DATA_DIR = BASE_DIR / 'data'
RAW_DATA_DIR = DATA_DIR / 'raw'
PROCESSED_DATA_DIR = DATA_DIR / 'processed'
MODELS_DIR = BASE_DIR / 'models'
REPORTS_DIR = BASE_DIR / 'reports'
LOGS_DIR = BASE_DIR / 'logs'

# Ensure directories exist
for directory in (
    DATA_DIR,
    RAW_DATA_DIR,
    PROCESSED_DATA_DIR,
    MODELS_DIR,
    REPORTS_DIR,
    LOGS_DIR,
):
    os.makedirs(directory, exist_ok=True)

# ----- Device configuration -----
DEVICE = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# ----- Dataset configuration -----
# Путь до синтетичних логів виробничих процесів (CSV)
MANUFACTURING_LOG_CSV = RAW_DATA_DIR / 'manufacturing_logs.csv'

# Параметри генерації синтетичних даних
SYNTHETIC_NUM_RECORDS = 10000      # кількість рядків (логів)
SYNTHETIC_START_DATE = '2025-01-01' # початкова дата у форматі
YYYY-MM-DD
SYNTHETIC_FREQ = 'H'               # частота записів: 'H' - година
MACHINE_IDS = ['M1', 'M2', 'M3']   # список ідентифікаторів машин

# ----- Hyperparameters -----
SEED = 42
BATCH_SIZE = 32
NUM_EPOCHS = 10
LR_ENCODER = 1e-4
LR_QUANTILES = 1e-3

# ----- Model parameters -----
AE_NUM_CHANNELS = 54 # latent channels in autoencoder

# ----- Loss configuration -----
LOSS_METRIC = 'ms-ssim'

```

LAMBDA = 10

----- Training & checkpoint -----

CHECKPOINT_FREQ = 100

CHECKPOINT_DIR = MODELS_DIR

----- Logging & reports -----

TENSORBOARD_LOG_DIR = LOGS_DIR / 'runs'

REPORT_FILE = REPORTS_DIR / 'analytics_report.xlsx'

Лістинг data_utils.py

import pandas as pd

from config import MANUFACTURING_LOG_CSV, PROCESSED_DATA_DIR

import os

def load_manufacturing_logs() -> pd.DataFrame: #Зчитує синтетичні логи виробничих процесів із CSV

#із роздільником ';' і повертає DataFrame

df = pd.read_csv(MANUFACTURING_LOG_CSV, sep=';')

Перетворюємо timestamp у datetime

df['timestamp'] = pd.to_datetime(df['timestamp'], dayfirst=False)

return df

def save_processed(df: pd.DataFrame, filename: str): # Зберігає оброблений DataFrame у полі PROCESSED_DATA_DIR.

os.makedirs(PROCESSED_DATA_DIR, exist_ok=True)

path = PROCESSED_DATA_DIR / filename

df.to_csv(path, index=False, sep=';')

print(f'Processed data saved to {path}')

Лістинг generate_data.py

from config import (

MANUFACTURING_LOG_CSV,

SYNTHETIC_NUM_RECORDS,

SYNTHETIC_START_DATE,

SYNTHETIC_FREQ,

MACHINE_IDS,

SEED,

)

import numpy as np

import pandas as pd

```

def generate_logs():
    np.random.seed(SEED)
    # генеруємо часові мітки
    timestamps = pd.date_range(
        SYNTHETIC_START_DATE,
        periods=SYNTHETIC_NUM_RECORDS,
        freq=SYNTHETIC_FREQ
    )
    # випадкові ID машин
    machine_ids = np.random.choice(MACHINE_IDS,
size=SYNTHETIC_NUM_RECORDS)
    # runtime і downtime
    runtime = np.random.uniform(0.8, 1.0, SYNTHETIC_NUM_RECORDS)
    downtime = np.random.uniform(0.0, 0.2, SYNTHETIC_NUM_RECORDS)
    # обсяг випуску
    output_qty = (
        runtime * 100 +
        np.random.normal(0, 5, SYNTHETIC_NUM_RECORDS)
    ).astype(int)

    df = pd.DataFrame({
        'timestamp': timestamps,
        'machine_id': machine_ids,
        'runtime': runtime,
        'downtime': downtime,
        'output_qty': output_qty,
    })

    # Записуємо з роздільником ';'
    df.to_csv(MANUFACTURING_LOG_CSV, index=False, sep=';')
    print(f'Saved {len(df)} rows to {MANUFACTURING_LOG_CSV} with
sep=;')

if __name__ == '__main__':
    generate_logs()

```

Лістинг main.py

```

from config import REPORT_FILE
from generate_data import generate_logs
from data_utils import load_manufacturing_logs, save_processed
from preprocessing import preprocess
from analytics import Analyzer
from visualization import plot_clusters, plot_anomalies, plot_forecast
from reports import export_report

```

```

def main():
    # 1. Згенерувати синтетичні логи
    generate_logs()

    # 2. Завантажити «сирі» дані
    df_raw = load_manufacturing_logs()

    # 3. Попередня обробка (очищення + ознакоутворення)
    df_clean = preprocess(df_raw)
    save_processed(df_clean, 'manufacturing_logs_clean.csv')

    # 4. Інтелектуальний аналіз
    features = ['throughput', 'downtime_ratio']
    analyzer = Analyzer()

    # 4a. Виявлення аномалій
    df_anoms = analyzer.detect_anomalies(df_clean, features)

    # 4b. Кластеризація
    df_clustered = analyzer.cluster(df_anoms, features)
    save_processed(df_clustered, 'manufacturing_logs_analyzed.csv')

    # 5. Прогнозування output_qty
    model, r2 = analyzer.forecast(df_clean, features, 'output_qty')
    print(f'Forecast model R²: {r2:.3f}')

    # 6. Візуалізація результатів
    plot_clusters(df_clustered, 'throughput', 'downtime_ratio')
    plot_anomalies(df_clustered, value_feature='throughput')

    df_forecast = df_clean.copy()
    df_forecast['predicted_qty'] = model.predict(df_clean[features])
    plot_forecast(
        df_forecast,
        time_col='timestamp',
        actual_col='output_qty',
        predicted_col='predicted_qty'
    )

    # 7. Збереження фінального звіту
    export_report(df_clustered, REPORT_FILE)

if __name__ == '__main__':
    main()

```

Лістинг preprocessing.py

```

import pandas as pd
from sklearn.preprocessing import StandardScaler

def preprocess(df: pd.DataFrame) -> pd.DataFrame:
    """
    Очищує "сирі" логи та створює додаткові ознаки:
    - видалення дублів
    - заповнення пропусків (forward-fill)
    - розрахунок throughput і downtime_ratio
    - масштабування нових ознак
    """
    # 1. Видалення дублів і заповнення пропусків
    df = df.drop_duplicates().fillna(method='ffill')

    # 2. Створення нових ознак
    df['throughput'] = df['output_qty'] / df['runtime']
    df['downtime_ratio'] = df['downtime'] / (df['runtime'] + df['downtime'])

    # 3. Масштабування ознак
    features = ['throughput', 'downtime_ratio']
    scaler = StandardScaler()
    df[features] = scaler.fit_transform(df[features])

    return df

```

Лістинг reports.py

```

import os
from pathlib import Path
import pandas as pd
from config import REPORT_FILE, PROCESSED_DATA_DIR

def export_report(df: pd.DataFrame, report_path: Path = REPORT_FILE):
    """
    Експортує результати аналізу в Excel-файл з кількома листами:
    1) 'Analyzed' — повна таблиця з колонками dataset + anomaly + cluster
    2) 'ClusterDistribution' — кількість записів у кожному кластері
    3) 'AnomalyDistribution' — кількість нормальних точок і аномалій
    """
    # Переконаємось, що папка для звіту існує
    os.makedirs(Path(report_path).parent, exist_ok=True)

    with pd.ExcelWriter(report_path, engine='xlsxwriter') as writer:

```

```

# 1. Повний звіт
df.to_excel(writer, sheet_name='Analyzed', index=False)

# 2. Розподіл кластерів
clusters = (
    df['cluster']
    .value_counts()
    .rename_axis('cluster')
    .reset_index(name='count')
)
clusters.to_excel(writer, sheet_name='ClusterDistribution', index=False)

# 3. Розподіл аномалій
anomalies = (
    df['anomaly']
    .value_counts()
    .rename_axis('anomaly')
    .reset_index(name='count')
)
anomalies.to_excel(writer, sheet_name='AnomalyDistribution', index=False)

# Форматування стовпців кожного листа
for sheet in writer.sheets:
    worksheet = writer.sheets[sheet]
    worksheet.set_column('A:Z', 20)

print(f'Report saved to {report_path}')

```

Лістинг visualization.py

```

import matplotlib.pyplot as plt
import pandas as pd

def plot_clusters(
    df: pd.DataFrame,
    x_feature: str,
    y_feature: str,
    cluster_col: str = 'cluster'
) -> None:
    """
    Малює розсіяння (scatter) точок df за двома ознаками,
    кольорами позначені кластери.

    Args:
        df: DataFrame із колонками x_feature, y_feature, cluster_col.
    """

```

```

    x_feature: назва колонки для осі X.
    y_feature: назва колонки для осі Y.
    cluster_col: назва колонки з мітками кластерів.
    """
    plt.figure()
    for cluster in sorted(df[cluster_col].unique()):
        part = df[df[cluster_col] == cluster]
        plt.scatter(part[x_feature], part[y_feature], label=f'Cluster {cluster}')
    plt.xlabel(x_feature)
    plt.ylabel(y_feature)
    plt.legend()
    plt.title('Cluster scatter')
    plt.show()

def plot_anomalies(
    df: pd.DataFrame,
    time_col: str = 'timestamp',
    value_feature: str = 'throughput',
    anomaly_col: str = 'anomaly'
) -> None:
    """
    Малює часовий ряд ознаки та виділяє точки-аномалії.

    Args:
        df: DataFrame із колонками time_col, value_feature, anomaly_col.
        time_col: назва колонки з часовими мітками.
        value_feature: назва колонки з числовою ознакою.
        anomaly_col: колонка з мітками аномалій (-1, 1).
    """
    plt.figure()
    normal = df[df[anomaly_col] == 1]
    anomalies = df[df[anomaly_col] == -1]
    plt.plot(normal[time_col], normal[value_feature])
    plt.scatter(anomalies[time_col], anomalies[value_feature], marker='x')
    plt.xlabel(time_col)
    plt.ylabel(value_feature)
    plt.title(f'Anomalies in {value_feature}')
    plt.show()

def plot_forecast(
    df: pd.DataFrame,
    time_col: str,
    actual_col: str,
    predicted_col: str

```

) -> None:

"""

Малює часові ряди фактичних та прогнозованих значень.

Args:

df: DataFrame із колонками time_col, actual_col, predicted_col.

time_col: назва колонки з часовими мітками.

actual_col: назва колонки з фактичними значеннями.

predicted_col: назва колонки з прогнозованими значеннями.

"""

plt.figure()

plt.plot(df[time_col], df[actual_col], label='Actual')

plt.plot(df[time_col], df[predicted_col], label='Predicted')

plt.xlabel(time_col)

plt.ylabel(actual_col)

plt.legend()

plt.title('Forecast vs Actual')

plt.show()

Вміст requirements.txt

altair==5.5.0

attrs==25.3.0

blinker==1.9.0

cachetools==5.5.2

certifi==2025.4.26

charset-normalizer==3.4.2

click==8.1.8

colorama==0.4.6

contourpy==1.3.2

cycler==0.12.1

filelock==3.18.0

fonttools==4.57.0

fsspec==2025.3.2

gitdb==4.0.12

GitPython==3.1.44

idna==3.10

Jinja2==3.1.6

joblib==1.4.2

jsonschema==4.23.0

jsonschema-specifications==2025.4.1

kiwisolver==1.4.8

MarkupSafe==3.0.2

matplotlib==3.10.1

mpmath==1.3.0

narwhals==1.37.1
networkx==3.4.2
numpy==2.2.5
packaging==24.2
pandas==2.2.3
pillow==11.2.1
protobuf==6.30.2
pyarrow==20.0.0
pydeck==0.9.1
pyparsing==3.2.3
python-dateutil==2.9.0.post0
pytz==2025.2
referencing==0.36.2
requests==2.32.3
rpds-py==0.24.0
scikit-learn==1.6.1
scipy==1.15.2
six==1.17.0
smmap==5.0.2
streamlit==1.45.0
sympy==1.14.0
tenacity==9.1.2
threadpoolctl==3.6.0
toml==0.10.2
torch==2.7.0
torchvision==0.22.0
tornado==6.4.2
typing_extensions==4.13.2
tzdata==2025.2
urllib3==2.4.0
watchdog==6.0.0
XlsxWriter==3.2.3

ДОДАТОК Б ЕЛЕКТРОННА ПРЕЗЕНТАЦІЯ



КАФЕДРА
ШТУЧНОГО
ІНТЕЛЕКТУ

КПІ ІМ. ІГОРЯ СІКОРСЬКОГО
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Інформаційна система для аналітики виробничих процесів на основі інтелектуального аналізу даних

Кваліфікаційна робота на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Системи і методи штучного інтелекту»
спеціальності 122 «Комп'ютерні науки»

КІ-13

Кутова Тетяна Анатоліївна

Керівник: доцент Гуськова В. Г.

Рецензент: професор Дмитрієва О. А.

2025



КАФЕДРА
ШТУЧНОГО
ІНТЕЛЕКТУ



Інформаційна система для аналітики виробничих процесів на
основі інтелектуального аналізу даних

Актуальність теми

- Сучасне виробництво генерує великі обсяги логів SCADA та MES, які часто залишаються неаналізованими.
- Ручна обробка таких даних є повільною, неточною та потребує значних ресурсів.
- Автоматизований інтелектуальний аналіз логів дозволяє своєчасно виявляти відхилення, збої та неефективні сценарії роботи.
- Інформаційні системи з можливістю аналізу даних у реальному часі є запорукою підвищення продуктивності та зниження витрат.
- Тема є актуальною для підприємств, які прагнуть цифрової трансформації та впровадження сучасних підходів до моніторингу виробничих процесів.

02

Мета, об'єкт і предмет дослідження

Мета роботи: розробка інформаційної системи аналітики виробничих процесів, що дозволяє виявляти відхилення, прогнозувати ефективність та підтримувати прийняття рішень у режимі реального часу.

Об'єкт дослідження: виробничі процеси в автоматизованих системах керування (АСУ).

Предмет дослідження: інформаційна система для збору та аналізу логів SCADA та MES із використанням методів інтелектуального аналізу даних.

03

Завдання дипломної роботи

- Провести аналіз логів SCADA та MES-систем у виробничому середовищі.
- Визначити вимоги до інформаційної системи аналітики.
- Розробити архітектуру системи збору, обробки й аналізу даних.
- Реалізувати модуль ETL для попередньої обробки логів.
- Впровадити модуль виявлення аномалій на основі алгоритмів ML.
- Створити веб-інтерфейс для інтерактивної візуалізації результатів.
- Оцінити ефективність запропонованого рішення за відповідними метриками.

04

КАФЕДРА
ШТУЧНОГО
ІНТЕЛЕКТУІнформаційна система для аналітики виробничих процесів на
основі інтелектуального аналізу даних

Архітектура SCADA та MES-систем

SCADA (Supervisory Control and Data Acquisition) — система моніторингу та керування технологічними процесами в реальному часі.

MES (Manufacturing Execution System) — система управління виконанням виробничих операцій на рівні цеху.

Логи з цих систем містять дані про події, сигнали, помилки, продуктивність тощо. Ці дані є джерелом для аналізу ефективності, відхилень і оптимізації процесів.

Бізнес-рівень (ERP)

MES — виконання
виробничих задачSCADA — моніторинг,
збір данихPLC / сенсори —
реальне обладнання

05

КАФЕДРА
ШТУЧНОГО
ІНТЕЛЕКТУ

Інформаційна система: структура та призначення

Призначення: інформаційна система призначена для збору, обробки, аналізу та візуалізації логів SCADA/MES з метою виявлення відхилень, кластеризації подій і прогнозування ключових показників.

Склад системи

ETL-модуль: попередня обробка логів (очищення, нормалізація, ознакоутворення).

Модуль аналітики: алгоритми виявлення аномалій, кластеризація, регресія

Система збереження даних: база даних логів (структуровані об'єкти)

Інтерфейс користувача: Streamlit-додаток з візуалізацією та налаштуваннями

Вхід: логи з SCADA/MES

Вихід: графіки, класифікація подій, аномалії, прогнози.

06



Архітектура створеної системи

Архітектура інформаційної системи



Основні компоненти

- **Логи** — сирі дані, зібрані з SCADA та MES-систем (події, сигнали, повідомлення).
- **ETL-конвеєр** — модуль попередньої обробки логів (очищення, нормалізація, ознакоутворення).
- **База даних логів** — структуроване збереження оброблених даних для подальшого аналізу.
- **Аналітичний модуль (Python)** — виявлення аномалій за допомогою алгоритмів машинного навчання (Isolation Forest).
- **Модуль візуалізації (Streamlit Dashboard)** — інтерактивне представлення результатів з можливістю drill-down аналізу.

07



Обробка логів

Сирі логи

	timestamp	machine_id	runtime	downtime	output_qty
0	2025-01-01 00:00:00	M3	0.8396	0.1882	85
1	2025-01-01 01:00:00	M1	0.8876	0.1139	86
2	2025-01-01 02:00:00	M3	0.8817	0.1003	87
3	2025-01-01 03:00:00	M3	0.8858	0.1211	93
4	2025-01-01 04:00:00	M1	0.9171	0.1233	83
5	2025-01-01 05:00:00	M1	0.9042	0.0341	86
6	2025-01-01 06:00:00	M3	0.83	0.1373	83
7	2025-01-01 07:00:00	M2	0.8131	0.0905	86
8	2025-01-01 08:00:00	M3	0.9459	0.1035	103
9	2025-01-01 09:00:00	M3	0.9119	0.0061	95

Після препроцесу

	timestamp	machine_id	runtime	downtime	output_qty	throughput	downtime_ratio
0	2025-01-01 00:00:00	M3	0.8396	0.1882	85	0.3146	1.6246
1	2025-01-01 01:00:00	M1	0.8876	0.1139	86	-0.465	0.3136
2	2025-01-01 02:00:00	M3	0.8817	0.1003	87	-0.1464	0.0935
3	2025-01-01 03:00:00	M3	0.8858	0.1211	93	0.9874	0.4364
4	2025-01-01 04:00:00	M1	0.9171	0.1233	83	-1.6104	0.4029
5	2025-01-01 05:00:00	M1	0.9042	0.0341	86	-0.7845	-1.1504
6	2025-01-01 06:00:00	M3	0.83	0.1373	83	0.0935	0.8462
7	2025-01-01 07:00:00	M2	0.8131	0.0905	86	1.1262	0.056
8	2025-01-01 08:00:00	M3	0.9459	0.1035	103	1.6868	0.0273
9	2025-01-01 09:00:00	M3	0.9119	0.0061	95	0.8407	-1.7117

08



Обробка логів

Логи з SCADA/MES-систем містять показники роботи обладнання: `timestamp`, `machine_id`, `runtime`, `downtime`, `output_qty`.

Дані згенеровані штучно для моделювання типових сценаріїв роботи.

Після препроцесингу додано нові ознаки: `throughput`, `downtime_ratio`, які допомагають точніше виявляти відхилення.

Нормалізація, заповнення пропусків, інженерія ознак реалізовані на Python.

09



Модуль виявлення аномалій

Для виявлення аномалій використано алгоритм *Isolation Forest*.

Аномалії — це нетипові спостереження, що різко відрізняються від більшості.

Переваги алгоритму: працює без учителя (*unsupervised*), ефективний на високовимірних даних, не потребує нормального розподілу.

Реалізовано за допомогою Python із використанням бібліотеки *scikit-learn*.

На виході формується аномальний скор (*anomaly score*) для кожного запису.

10

Оцінка якості кластеризації

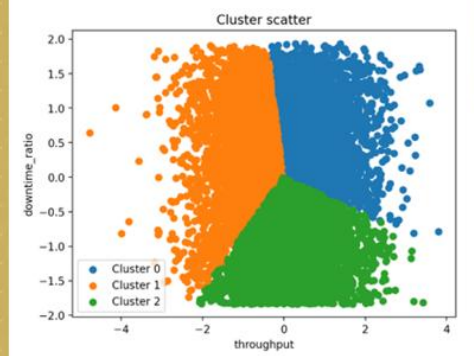
Для аналізу структури даних застосовано кластеризацію *KMeans*.

Оцінка якості кластерів здійснювалась за метриками: Silhouette Score = 0.359 — помірна щільність кластерів, Davies-Bouldin Index = 0.912 — кластери помірно розділені.

Ознаки для кластеризації: throughput, downtime_ratio.

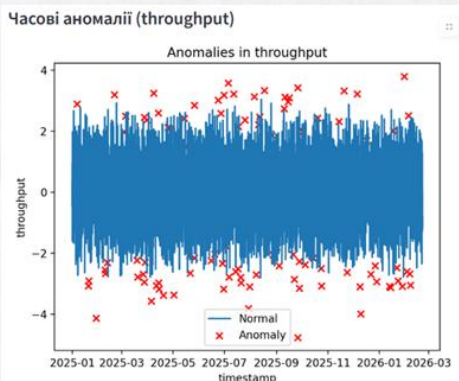
Кількість кластерів — налаштовується інтерактивно в інтерфейсі

Розподіл кластерів



11

Виявлення часових аномалій



Аналіз проводився для показника throughput. Модель Isolation Forest застосовано до часових рядів.

Для кожного запису обчислюється аномалійний скор.

На основі скору виявлено записи з нестандартною поведінкою.

Результати виводяться у вигляді інтерактивного графіка:

- синій — нормальні значення,
- червоні x — аномальні події.

12

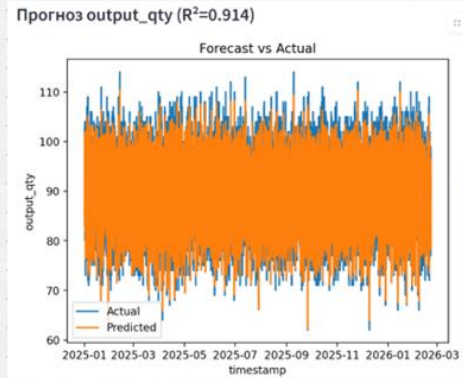
Прогнозування кількості виробленої продукції

Для прогнозування обрано регресійну модель на основі RandomForestRegression.

Ознаки: runtime, downtime, throughput, downtime_ratio.

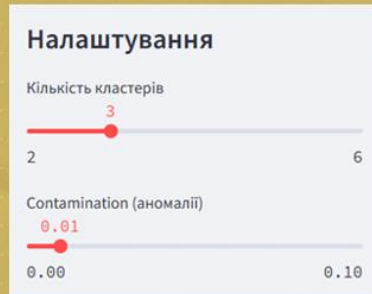
Якість моделі оцінено за коефіцієнтом детермінації: $R^2 = 0.914$ — висока точність моделі.

Графік відображає порівняння фактичних та прогнозованих значень



13

Інтерфейс користувача Streamlit



Налаштування кластеризації та виявлення аномалій у Streamlit

Реалізовано з використанням Streamlit — фреймворку для швидкої побудови веб-додатків у Python.

Інтерфейс надає можливість:

- візуалізувати аномалії та кластери у вигляді графіків;
- змінювати кількість кластерів (K) за допомогою слайдера;
- фільтрувати дані за часовим діапазоном;
- запускати аналіз нових логів натисканням кнопки.

14

Основні результати реалізації

- Розроблено інформаційну систему аналітики виробничих процесів.
- Реалізовано повний ETL-конвеєр для збору та обробки логів.
- Побудовано модуль виявлення аномалій (Isolation Forest).
- Проведено кластеризацію з оцінкою якості (Silhouette = 0.359, DB Index = 0.912).
- Реалізовано прогнозування обсягу продукції ($R^2 = 0.914$).
- Розроблено зручний веб-інтерфейс на базі Streamlit з можливістю керування параметрами.

15

Висновки та перспективи

Висновки

Мету дипломної роботи досягнуто: створено систему для аналітики виробничих процесів на основі логів SCADA/MES.

Реалізовано повний цикл: від збору та обробки логів до виявлення аномалій, кластеризації та прогнозування.

Система показала хорошу якість результатів ($R^2 = 0.914$, Silhouette = 0.359).

Інтерфейс Streamlit забезпечує зручність взаємодії з системою.

Перспективи розвитку

Додавання інтеграції з реальними SCADA-системами.

Розширення моделей прогнозування (наприклад, XGBoost, LSTM).

Підключення до хмарних баз даних або реального API.

Розробка авторизації та системи реєстрації дій користувача.

16



КАФЕДРА
ШТУЧНОГО
ІНТЕЛЕКТУ

Інформаційна система для аналітики виробничих процесів
на основі інтелектуального аналізу даних

Дякую за увагу!

студентка IV курсу, групи КІ-13, Кутова Тетяна