

Міністерство освіти і науки України
Чернівецький національний університет імені Юрія Федьковича

Кваліфікаційна наукова праця
на правах рукопису

КИРИЧЕНКО ЄВГЕН ОЛЕКСАНДРОВИЧ

УДК 004.6:[519.237.8+519.24/.25

ДИСЕРТАЦІЯ

**ОПТИМІЗАЦІЯ СТРУКТУРИ ГЕТЕРОГЕННИХ ДАНИХ
В BIG DATA**

121 – інженерія програмного забезпечення

12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

 Є.О. Кириченко

Науковий керівник: **Малик Ігор Володимирович** доктор фізико-математичних наук, професор

Чернівці – 2026

АНОТАЦІЯ

Кириченко Є. О. Оптимізація структури гетерогенних даних в Big Data.
– Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 121 – «Інженерія програмного забезпечення» – Чернівецький національний університет імені Юрія Федьковича, Чернівці, 2026.

Вирішення проблеми порівняння гетерогенних табличних наборів даних з невідомих або слабо документованих джерел є однією з важливих наукових задач у галузі інформаційних технологій, орієнтованих на побудову та експлуатацію інформаційних систем, які оперують великими даними (Big Data). Потреба в розробці ефективних методів структурування та порівняння корпусів даних на основі компактних представлень без доступу до всіх необроблених записів викликана постійним зростанням обсягів табличних даних у корпоративних і хмарних середовищах, відсутністю стандартизованих схем та високою вартістю переміщення повних копій даних.

У дисертаційній роботі здійснено аналіз існуючих підходів до порівняння табличних наборів даних, виявлено їх обмеження щодо роботи з гетерогенними даними. В роботі розроблено типорієнтований підхід створення компактних представлень даних для різних типів змінних, запропоновано зважену метрику структурної подібності даних, а також створено алгоритми побудови графів та дерев подібності для організації корпусів даних. Розроблено хмарну масштабовану архітектуру на базі AWS для обробки великих корпусів табличних даних та проведено експериментальні дослідження ефективності запропонованого підходу.

Об'єктом дослідження є процес порівняння гетерогенних табличних наборів даних з слабо документованих джерел.

Предмет дослідження є моделі та методи класифікації змінних, створення компактних представлень гетерогенних табличних наборів даних, побудова графів подібності для ефективного порівняння та структурування

корпусів даних за мінімальних припущень щодо схеми.

Метою дослідження є розробка уніфікованої типо-орієнтованої інформаційної системи для ефективного порівняння гетерогенних табличних наборів даних з невідомих або слабо документованих джерел за допомогою створення компактних представлень даних з використанням зваженої метрики структурної подібності інформації даних та побудова масштабованої хмарної інформаційної системи для формування графів та дерев подібності корпусів.

Наукова новизна одержаних результатів полягає в наступному:

- *вперше розроблено* уніфіковану методологію конструювання компактних представлень даних для гетерогенних табличних змінних, яка на відміну від існуючих підходів, що орієнтовані на конкретні модальності або припускають наявність інформації про схему даних, забезпечує типо-орієнтовану побудову інтерпретованих зведень через таблиці частот для факторних змінних, гістограми для часових змінних, вектори моментів до четвертого порядку для числових змінних та TF-IDF вектори для рядкових змінних;
- *запропоновано нову метрику* структурної подібності даних, що використовують одиничні метрики або векторні представлення, здійснює зважену агрегацію відстаней Геллінгера, Вассерштейна ($p = 1$), повної варіації та $L1/L2$, з MAE/MARE для числових зведень для забезпечення принципово обґрунтованого порівняння характеристик різнотипних змінних;
- *розроблено новий метод* побудови ієрархічних структур подібності корпусів табличних даних на основі графів суміжності, який дозволяє пошук найбільш подібних даних та ієрархічне дослідження без припущення фіксованої схеми;
- *розроблено та реалізовано* наскрізну хмарну масштабовану інформаційну технологію обробки великих корпусів табличних даних, яка на відміну від традиційних підходів до порівняння повних копій

даних, забезпечує розподілену побудову компактних представлень, їх зберігання та запити через взаємодію Apache Spark, AWS S3, Glue, Athena та Airflow, що дозволяє значно зменшити витрати на зберігання та копіювання даних при збереженні точності аналізу подібності;

- *модифіковано підхід* до кластеризації гетерогенних даних на основі компактних представлень та матриці суміжності, що забезпечує стійкість до шуму й масштабованість. Запропоновано метод аналізу кластерної структури за спектральними властивостями стохастичних матриць. *Створено інструментарій*, який дозволяє визначати природні групи ознак та виявляти структурні відхилення у великих наборах даних.

Практичне значення отриманих результатів. За результатами виконаних досліджень розроблено архітектуру уніфікованої інформаційної технології для порівняння гетерогенних табличних наборів даних, процедуру конструювання компактних представлень даних, метрики обчислення подібності та методи побудови структур корпусів даних для використання в умовах інформаційних систем великого масштабу з невідомими джерелами.

Застосування методу компактних представлень даних на основі типоспецифічних зведень дозволило зменшити обсяги даних для зберігання та передачі при збереженні достатньої інформації для порівняння наборів даних. Розроблена хмарна архітектура забезпечує розподілену обробку файлів великого розміру, використовуючи Apache Spark на EMR для розподіленої обробки, S3 для зберігання, Airflow для оркестрації, Glue для каталогізації та Athena для ad-hoc запитів.

Експериментальні дослідження продемонстрували зменшення обсягу даних завдяки використанню компактних представлень. Побудовані типорієнтовані зведення суттєво зменшують обсяг даних для аналізу, що відповідно знижує витрати на зберігання та значно скорочує кількість операцій читання/запису. Застосування розподіленої архітектури на базі

Apache Spark, колонкових форматів зберігання Parquet та хмарних сервісів AWS забезпечило прискорення обробки приблизно на 40–60% порівняно з традиційними підходами повного сканування таблиць.

Дисертація складається зі вступу, чотирьох розділів, висновків, переліку використаних джерел та чотирьох додатків.

У **вступі** обґрунтовано актуальність теми дослідження, сформульовано мету, завдання, предмет, об'єкт та методи дослідження, вказано наукову новизну, подано та проаналізовано зв'язок роботи з науковими темами. Зазначено особистий внесок здобувача, а також наведено відомості про апробацію та публікації основних результатів дисертації. Описано структуру та обсяг дисертаційної роботи.

Перший розділ дисертації містить теоретичні засади аналізу великих та гетерогенних даних (Big Data) і обґрунтовує застосування спектральних методів та апарату випадкових матриць для дослідження їхньої структури.

У цьому розділі подано математичний апарат стохастичних та випадкових матриць, основні спектральні характеристики великорозмірних систем та результати теорії випадкових матриць, які дозволяють описувати поведінку складних структур без жорстких припущень щодо моментів розподілів.

Сформульовано обґрунтований критерій визначення оптимальної кількості кластерів, що базується на підрахунку власних значень стохастичної матриці, які перевищують заданий поріг. Розглянуто можливості застосування спектральних методів у різних сферах Big Data – соціальних, біологічних, фінансових, технологічних та вебсистемах, де властивості даних відзначаються нерівномірністю та стохастичною природою.

Другий розділ присвячений розробці математичного та алгоритмічного апарату інформаційної технології компактного подання, класифікації та порівняння табличних даних у задачах аналізу великих гетерогенних наборів даних. Розглянуто сучасні концепції аналітики Big

Data, на основі якої запропоновано узагальнену модель компактного представлення (CDR), що замінює повні набори даних їх стислими дескрипторами, які зберігають інформаційно значущі статистичні та структурні характеристики. Розроблено алгоритм автоматичної типізації змінних, який на основі структурних і статистичних критеріїв класифікує атрибути наборів даних на числові, категоріальні, часові та текстові типи без необхідності попередньої схеми. Описано класи метрик, придатних для числових, текстових та категоріальних даних, та наведено обґрунтування їхнього використання.

Побудовано та обґрунтовано конвеєрну систему аналізу даних, робота якої реалізується чотирма функціональними блоками.

У третьому розділі дисертації здійснено аналіз сучасних хмарних технологій та принципів побудови масштабованих обчислювальних систем. Розглянуто ключові характеристики хмарних платформ, моделі сервісів (IaaS, PaaS, SaaS), а також особливості розподіленого зберігання, обробки та передачі даних у середовищах з високим навантаженням.

Розроблено багаторівневу модульну архітектуру інформаційної системи, що складається з трьох логічних рівнів із чітко визначеною відповідальністю та слабкою зв'язністю компонент. Програмне забезпечення усіх рівнів реалізовано мовою Python із використанням Apache Airflow для оркестрації робочих процесів та Apache Spark для розподіленої обробки даних, що забезпечує незалежність обчислювального ядра від конкретного постачальника хмарних послуг.

Описано повний конвеєр трансформації неструктурованих даних, що містить послідовність етапів розподіленої обробки та аналізу даних.

У четвертому розділі розроблено та реалізовано програмне забезпечення інформаційної системи та проведено експериментальну перевірку інформаційної технології моделювання та структурного аналізу гетерогенних табличних даних, що базується на методі компактного представлення CDR та метриці структурної подібності DISS.

Показано, що розроблена програмна система забезпечує модульність, підтримує автоматизований режим виконання та може бути інтегрована у реальні конвеєри обробки даних на хмарних платформах сучасних інформаційних систем.

Експериментальне тестування проводилось на реальних фінансових часових рядах, сформованих із двох відкритих наборів даних, що дозволило верифікувати універсальність та стійкість запропонованого підходу до змін джерела даних, часових інтервалів і ринкових режимів.

Отримані результати свідчать про збалансованість між точністю та повнотою класифікації, підтверджують практичну ефективність запропонованої інформаційної технології виявлення структурних змін у даних та її придатність для масштабованого застосування у сучасних Big Data системах і хмарних платформах штучного інтелекту, машинного навчання та нейронних мереж.

У **висновках** підсумовано основні результати дисертаційного дослідження.

У **додатках** подано наукові публікації, в яких відображено основні наукові результати роботи, відомості про апробацію результатів дисертації – акти та довідки про впровадження результатів роботи, лістинг частини коду програмного забезпечення.

Запропонований підхід до побудови компактних представлень (CDR) та метрики DISS для порівняння табличних даних, розроблений метод кластеризації на основі структурних відстаней та запропоновані методи інтеграції кластерного аналізу у аналітичні модулі компанії використовуються у роботі компаній ТОВ «Кодерс ПРО» та ТОВ «Палетний сервіс». А результати теоретичних та практичних досліджень використовуються у навчальному процесі кафедр математичних проблем управління і кібернетики та програмного забезпечення комп'ютерних систем Чернівецького національного університету імені Юрія Федьковича.

Ключові слова: набори даних, модель, моделювання, класифікація,

точність, машинне навчання, аналіз даних, штучний інтелект, інформаційні технології / системи, програмне забезпечення, програмна система, нейронні мережі, Python, хмарна платформа, кластерний аналіз.

ABSTRACT

Kyrychenko Ye. O. Optimization of the Structure of Heterogeneous Data in Big Data – Qualification research work published in the manuscript.

Dissertation for the degree of Doctor of Philosophy, speciality 121 – "Software Engineering" – Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 2026.

Solving the problem of comparing heterogeneous tabular datasets from unknown or poorly documented sources is an essential scientific task in information technology, focused on the development and operation of information systems that handle large volumes of data (Big Data). The need to develop effective methods for structuring and comparing data corpora based on compact representations without access to all raw records arises from the constant growth of tabular data volumes in corporate and cloud environments, the lack of standardized schemas, and the high cost of transferring complete data copies.

In the dissertation, existing approaches to comparing tabular datasets are analyzed, and their limitations in working with heterogeneous data are identified. The work develops a type-oriented approach for creating compact data representations for different variable types, proposes a weighted metric for structural data similarity, and develops algorithms for constructing similarity graphs and trees to organize data corpora. A cloud-scalable architecture based on AWS for processing large tabular data corpora is created, and experimental studies of the proposed approach's effectiveness are conducted.

The object of the study is to compare heterogeneous tabular datasets from poorly documented sources.

The subject of the research is models and methods for variable classification, the creation of compact representations of heterogeneous tabular

datasets, and the construction of similarity graphs for effective comparison and the structuring of data corpora under minimal assumptions about the schema.

The goal of the study is to develop a unified type-oriented information system for effectively comparing heterogeneous tabular datasets from unknown or poorly documented sources by creating compact data representations using a weighted metric of structural data similarity, and to build a scalable cloud information system for constructing similarity graphs and trees of data corpora.

The scientific novelty of the obtained results lies in the following:

- *for the first time*, a unified methodology for constructing compact data representations for heterogeneous tabular variables *is developed*, which, unlike existing approaches oriented toward specific modalities or assuming the availability of data schema information, ensures type-oriented construction of interpretable summaries through frequency tables for categorical variables, histograms for temporal variables, moment vectors up to the fourth order for numerical variables, and TF-IDF vectors for string variables;
- *a new metric* of structural data similarity *is proposed*, which, unlike approaches using single metrics or vector representations, performs weighted aggregation of Hellinger, Wasserstein ($p = 1$), total variation, and $L1/L2$ distances with MAE/MAPE for numerical summaries to ensure a fundamentally justified comparison of characteristics of different variable types;
- *a new method* for constructing hierarchical similarity structures of tabular data corpora based on adjacency graphs *is developed*, which enables searching for the most similar data and hierarchical analysis without assuming a fixed schema.
- an end-to-end cloud-scalable *information technology* for processing large corpora of tabular data *is developed*, which, unlike traditional approaches based on comparing complete data copies, ensures distributed construction of compact representations, their storage, and querying through interaction

among Apache Spark, AWS S3, Glue, Athena, and Airflow, making it possible to significantly reduce storage and data-copying costs while preserving similarity analysis accuracy;

- *an approach* to clustering heterogeneous data based on compact representations and adjacency matrices *is modified*, ensuring noise resistance and scalability. A method for analyzing cluster structure based on spectral properties of stochastic matrices is proposed. A toolkit is created that enables the identification of natural feature groups and the detection of structural deviations in large datasets.

Practical significance of the obtained results. Based on the conducted research, an architecture for unified information technology for comparing heterogeneous tabular datasets, a procedure for constructing compact data representations, similarity-computation metrics, and methods for building data corpus structures for use in large-scale information systems with unknown sources are developed.

The application of the method of compact data representations based on type-specific summaries enabled reduced storage and transmission volumes while preserving sufficient information for dataset comparison. The developed cloud architecture ensures large-file processing using Apache Spark on EMR, S3 for storage, Airflow for orchestration, Glue for cataloging, and Athena for ad-hoc queries.

Experimental studies demonstrated a reduction in data volume by using compact representations. The type-oriented summaries significantly reduce the amount of data to analyze, thereby lowering storage costs and substantially reducing the number of read/write operations. The use of a distributed architecture based on Apache Spark, the columnar storage format Parquet, and AWS cloud services accelerated processing by approximately 40–60% compared with traditional full-table scanning approaches.

The dissertation consists of an introduction, four chapters, a conclusion, a list of references, and four appendices.

The introduction substantiates the relevance of the research topic, formulates the purpose, objectives, subject, and object, specifies the research methods, indicates scientific novelty, and presents and analyzes the connection of the work to scientific topics. The personal contribution of the applicant is defined, and information on approbation and publications of the main dissertation results is provided. The structure and volume of the dissertation are described.

The first chapter of the dissertation provides the theoretical foundations for analyzing extensive, heterogeneous data (Big Data). It substantiates the application of spectral methods and random matrix theory to their analysis.

This chapter presents the mathematical apparatus of stochastic and random matrices, the main spectral characteristics of large-scale systems, and results of random matrix theory that allow for the description of the behavior of complex structures without strict assumptions about distributional moments.

A justified criterion for determining the optimal number of clusters based on the number of eigenvalues of a stochastic matrix exceeding a given threshold is formulated. The possibilities of applying spectral methods across various Big Data fields – social, biological, financial, technological, and web systems – are considered, where data exhibit irregularity and stochasticity.

The second chapter is devoted to the development of the mathematical and algorithmic apparatus for compact representation, classification, and comparison of tabular data in the analysis of large, heterogeneous datasets. Modern concepts of Big Data analytics are considered, and a generalized compact data representation model (CDR) is proposed that replaces complete datasets with compressed descriptors that preserve informationally significant statistical and structural characteristics. An algorithm for automatic variable typing is developed that classifies dataset attributes into numerical, categorical, temporal, and textual types based on structural and statistical criteria, without requiring a preliminary schema. Classes of metrics suitable for numerical, textual, and categorical data are described and justified.

A pipeline data analysis system is constructed and substantiated, with its

operation implemented through four functional blocks.

The third chapter of the dissertation analyzes modern cloud technologies and principles of building scalable computing systems. Key characteristics of cloud platforms, service models (IaaS, PaaS, SaaS), and features of distributed storage, processing, and data transmission in high-load environments are considered.

A multi-level modular architecture of the information system is developed, consisting of three logical levels with clearly defined responsibilities and loosely coupled components. The software of all levels is implemented in Python using Apache Airflow for workflow orchestration and Apache Spark for distributed data processing, ensuring independence of the computational core from a specific cloud service provider.

A complete pipeline for transforming unstructured data is described, containing a sequence of distributed processing and analysis stages.

In the fourth chapter, software for the information system is developed and implemented, and experimental verification of the information technology for modeling and structural analysis of heterogeneous tabular data based on the CDR compact representation method and the DISS structural similarity metric is conducted.

It is shown that the developed software system ensures modularity, supports automated execution, and can be integrated into real data processing pipelines on cloud platforms of modern information systems.

Experimental testing was conducted on real financial time series derived from two open datasets, enabling verification of the universality and robustness of the proposed approach across changes in data sources, time intervals, and market regimes.

The results indicate a balance between classification accuracy and completeness, confirm the practical effectiveness of the proposed information technology for detecting structural changes in data, and demonstrate its suitability for scalable application in modern Big Data systems and cloud platforms for

artificial intelligence, machine learning, and neural networks.

The conclusions summarize the dissertation's main results.

The appendices present scientific publications reflecting the main scientific results of the work, information on approbation of the dissertation results – acts and certificates of implementation – and listings of part of the software code. The proposed approach to constructing compact data representations (CDR) and the DISS metric for comparing tabular data, the developed clustering method based on structural distances, and the methods for integrating clustering computations into the analytical modules of a company are used in the work of LLC “Coders PRO” and LLC “Paletnyi Servis”. The results of the theoretical and practical research are also used in the educational process of the Departments of Mathematical Problems of Control and Cybernetics and Software of Computer Systems at Yuriy Fedkovych Chernivtsi National University.

Keywords: data sets, model, modeling, classification, accuracy, machine learning, data analysis, artificial intelligence, information technology / system, software, software system, neural networks, Python, cloud platform, cluster analysis.

СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

**Наукові праці, в яких опубліковані
основні наукові результати дисертації**

***Наукові праці у виданнях, включених до переліку
наукових фахових видань України та проіндексованих у наукометричній
базі даних Scopus:***

1. Кириченко О., Кириченко Є. Асимптотичні властивості випадкових матриць = Asymptotic Properties of Random matrices. *Вісник Київського національного університету імені Тараса Шевченка. Фізико-математичні науки.* 2024. 78(1). С. 41–44. (Scopus) <https://doi.org/10.17721/1812-5409.2024/1.7> (Кириченко О. – концептуалізація, формальний аналіз;

Кириченко Є. – аналіз джерел, підготовка огляду літератури).

***Наукові праці у періодичних наукових виданнях,
проіндексованих у наукометричній базі даних Scopus:***

2. Malyk I. V., **Kyrychenko Y.**, Gorbatenko M., Lukashiv T. Data Optimization through Compression Methods Using Information Technology. *International Journal of Information Technology and Computer Science*. 2025. Vol.17, No.5. P. 84-99. (Scopus, Q3 – <https://www.scimagojr.com/journalsearch.php?q=21101226492&tip=sid&clean=0>) <https://doi.org/10.5815/ijitcs.2025.05.07> (Малик І. – концептуалізація, методологія, формальний аналіз, написання та підготовка оригінальної чернетки, написання, рецензування та редагування, візуалізація, авторський нагляд; Кириченко Є. – концептуалізація, методологія, формальний аналіз, дослідження, написання та підготовка оригінальної чернетки, написання-рецензування та редагування, візуалізація; Горбатенко М. – методологія, формальний аналіз; Лукашів Т. – концептуалізація, написання, рецензування та редагування, авторський нагляд, адміністрування проєкту).

***Наукові праці у виданнях, включених до переліку
наукових фахових видань України:***

3. **Кириченко Є. О.**, Кириченко О. Л. Побудова моделей нейронних мереж у хмарному середовищі засобами AWS EMR. *Вчені записки Таврійського національного університету імені В.І. Вернадського*. Серія : Технічні науки. 2024. 35(74), №6. С. 75-79. <https://doi.org/10.32782/2663-5941/2024.6.2/11> (Кириченко Є. – теоретична та практична розробка положень, програмна реалізація, редагування рукопису; Кириченко О. – постановка задачі, керівництво дослідженням, рецензування та редагування).

4. **Kyrychenko Y.**, Malyk I. Information Technology for Data Compression and Transformation by Means of Apache Airflow and Amazon EMR. *Security of Infocommunication Systems and Internet of Things*. 2025. Vol. 3, no. 1. P. 01004. <https://doi.org/10.31861/sisiot2025.1.01004> (Кириченко Є. – проєктування

архітектури системи, програмне забезпечення, розробка та тестування програмного забезпечення, написання та редагування; Малик І. – загальне керівництво, концептуалізація).

5. **Кириченко Є.**, Малик, І. Один результат кластеризації на основі випадкових матриць. *Information Technology: Computer Science, Software Engineering and Cyber Security*. 2025. №3. С. 257-270. <https://doi.org/10.32782/IT/2025-3-29> (Кириченко Є. – проведення дослідження, редагування рукопису; Малик І. – постановка задачі, визначення загальної схеми досліджень, обговорення результату).

Наукові праці, які засвідчують апробацію матеріалів дисертації:

6. **Кириченко Є. О.**, Малик І. В. Використання подвійних стохастичних моделей в BIG DATA. *Проблеми інформатики та комп'ютерної техніки (ПІКТ – 2023)* : праці XII Міжнародної науково-практичної конференції, м. Чернівці, 10 – 12 листопада 2023 р. Чернівці : Черн. нац. ун-т, 2023. С. 33–35. (Кириченко Є. – огляд літературних джерел, написання, рецензування та редагування; Малик І. – постановка задачі, рецензування).

7. **Кириченко Є. О.**, Малик І. В. Моделі і процеси при дослідженні подвійних стохастичних систем. *Інформаційні технології: наука, техніка, технологія, освіта, здоров'я* : тези доп. XXXII Міжнар. наук.-практ. конф. MicroCAD-2024, 22–25 травня 2024 р. / за ред. проф. Сокола Є.І. Харків : НТУ «ХПІ». С. 1294. (Кириченко Є. – проведення дослідження, написання рукопису; Малик І. – концептуалізація, методологія, загальне керівництво).

8. **Kyrychenko Y.**, Malyk I. Leveraging AWS EMR for Scalable and Efficient Neural Network Deployment in Cloud Computing. *The 13 th International Conference on Electronics, Communications and Computing's (IC ECCO)* : Materials of the Intern. Conf., (Chisinau, Moldova, 17–18 October, 2024.). Chisinau : Technical University of Moldova, 2024. P. 128-129. (Кириченко Є. – проведення досліджень, аналіз та опис результатів; Малик І. – постановка задачі, обговорення результатів).

9. **Кириченко Є. О.**, Малик І. В. Порівняльний аналіз економічної

ефективності AWS ELASTIC MAPREDUCE та EC2 для розгортання нейронних мереж. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2024)* : праці XIII Міжнар. наук.-практ. конф., м. Чернівці, 01–03 листопада 2024 р. Чернівці : Черн. нац. ун-т, 2024. С. 40–42. (Кириченко Є. – збір та аналіз даних, обговорення результатів; Малик І. – концептуалізація дослідження, розробка методології, аналіз результатів).

10. **Кириченко Є.**, Малик І. Інформаційна технологія структурного аналізу табличних наборів даних для виявлення аномалій та невідповідностей. *Science in the Context of Modern Challenges and Prospects* : Materials of the Intern. Conf., 21 травня 2025 р. (м. Утрехт, Нідерланди). 2025. С. 196-197. (Кириченко Є. – проведення дослідження, аналіз та опис результату; Малик І. – визначення загальної схеми дослідження).

11. Малик І. В., **Кириченко Є. О.** Підхід до порівняння гетерогенних табличних даних на основі методів компресії. *Проблеми інформатики та комп'ютерної техніки* : праці XIV Міжнародної науково-практичної конференції (ПІКТ – 2025), присвяченої 85-й річниці від дня народження проф. Кириченка М.Ф. м. Чернівці, 13–15 лист. 2025. Чернівці : Технодрук, 2025. С. 124-126. (Кириченко Є. – концептуалізація, методологія, формальний аналіз, дослідження, написання; Малик І. – постановка задачі, загальне керівництво).

Наукові праці, які додатково відображають наукові результати дисертації:

12. **Kyrychenko Y. O.**, Kyrychenko O. L., Filipchuk O. I. The Study Of Complex Networks Using Visualization Tools. *Science and technology today*. «Technics» series. 2024. № 10(38). С. 81-93. [https://doi.org/10.52058/2786-6025-2024-10\(38\)-81-92](https://doi.org/10.52058/2786-6025-2024-10(38)-81-92) (Кириченко Є. – концептуалізація, методологія, формальний аналіз, дослідження, підготовка ілюстративних матеріалів; Кириченко О. – концептуалізація, обговорення результатів; Філіпчук О. – обговорення результатів, рецензування та редагування).

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	20
ВСТУП	21
РОЗДІЛ 1. СПЕКТРАЛЬНІ ПІДХОДИ ДО КЛАСТЕРИЗАЦІЇ У BIG DATA ТА ГЕТЕРОГЕННИХ МЕРЕЖЕВИХ СИСТЕМАХ .	28
1.1 Кластеризація у Big Data та роль випадкових матриць	28
1.2 Графи великої розмірності та їх стохастичні моделі	30
1.3 Гіпотези кластерної структури графу (H1–H4)	38
1.4 Модельний приклад	40
1.5 Прикладні задачі Big Data та значення запропонованого методу кластеризації	43
Висновки до розділу 1	49
РОЗДІЛ 2. МАТЕМАТИЧНІ ОСНОВИ ТА АЛГОРИТМИ КОМПАКТНОГО ПРЕДСТАВЛЕННЯ ДАНИХ	52
2.1 Аналіз задач визначення подібності та формування вимог до компактних представлень даних.	54
2.2 Типізація змінних	64
2.2.1 Визначення типів даних.	65
2.2.2 Алгоритм типізації даних.	69
2.3 Стиснення різних типів змінних	70
2.4 Порівняння стиснених даних (метрика DISS)	72
2.4.1 Основні метрики для порівняння різних типів даних	73
2.4.2 Виявлення дублікатів і зсувів розподілів	81
2.4.3 Алгоритмічні принципи побудови DISS-матриць .	82
2.5 Графові представлення структурної подібності наборів даних	83
2.6 Оцінка існуючих методів та концепція власної розробки . . .	85
2.7 Конвеєрна система аналізу даних	88
Висновки до розділу 2	90

РОЗДІЛ 3. ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ПРОВЕДЕННЯ ОПТИМІЗАЦІЇ СТРУКТУРИ ГЕТЕРОГЕННИХ ДАНИХ	92
3.1 Функціональне призначення системи	92
3.2 Вимоги до інформаційної системи	93
3.3 Архітектура та функціональні компоненти системи.	95
3.3.1 Логічна архітектура	95
3.3.2 Архітектура хмарної реалізації	98
3.3.3 Функціонування створеної інформаційної системи . . .	104
3.4 Технологічний стек, середовище виконання та принципи масштабування	111
3.4.1 AWS EMR SERVERLESS	112
Висновки до розділу 3	114
РОЗДІЛ 4 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ.	117
4.1 Опис наборів даних.	117
4.2 Стандартизація компактних представлень і формування CDR-моделі	121
4.3 Оцінка структурного дрейфу при ітеративному зменшенні обсягу даних	121
4.3.1 Мета та методологія експерименту	121
4.3.2 Результати аналізу чутливості компактних представлень	125
4.3.3 Аналіз кривої DISS-дрейфу	127
4.3.4 Оцінка ступеня схожості ознак даних	132
4.3.5 ROC-аналіз якості виявлення дрейфу	137
4.4 Аналіз результатів практичного застосування запропонованого алгоритму для обробки даних з другої вибірки . . .	140
4.4.1 Результати аналізу чутливості компактних представлень.	140

4.4.2 ROC-аналіз якості виявлення дрейфу для даних з Kaggle	143
4.4.3 Оцінка ступеня схожості ознак даних за допомогою візуалізації метричних відстаней	144
4.4.4 Аналіз кривої DISS-дрейфу	149
4.5 Ідентифікація ключових зв'язків у наборах даних.	153
Висновки до розділу 4	159
ВИСНОВКИ	162
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	165
ДОДАТКИ	191
Додаток А. Список публікацій за темою дисертації	191
Додаток Б. Відомості про апробацію матеріалів дисертації	195
Додаток В. Акти впровадження результатів дисертаційної роботи	196
Додаток Г. Лістинг частини коду програми	200

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Умовне позначення	Повна назва (англ.)	Пояснення (укр.)
AWS	Amazon Web Services	Хмарна платформа компанії Amazon
MAE	Mean Absolute Error	Середня абсолютна похибка
MAPE	Mean Absolute Percentage Error	Середня абсолютна відсоткова похибка
S3	Simple Storage Service	Сервіс об'єктного зберігання в AWS
EMR	Elastic MapReduce	Платформа для обробки Big Data в AWS
DISS	Data Information Structure Similarity	Міра подібності структури даних
Big Data	Big Data	Великі обсяги структурованих і неструктурованих даних
IaaS	Infrastructure as a Service	Інфраструктура як сервіс
PaaS	Platform as a Service	Платформа як сервіс
SaaS	Software as a Service	Програмне забезпечення як сервіс
CDR	Compact Data Representation	Компактне представлення даних
ML	Machine Learning	Машинне навчання
TF-IDF	Term Frequency – Inverse Document Frequency	Міра важливості термів у тексті
ETL	Extract, Transform, Load	Процес витягання, перетворення та завантаження даних
MBR	Minimum Bounding Rectangle	Мінімальний обмежувальний прямокутник
MST	Minimum Spanning Tree	Мінімальне остовне дерево
PMF	Probability Mass Function	Функція розподілу ймовірностей

ВСТУП

Актуальність та обґрунтування теми дисертаційного дослідження.

Сучасний етап розвитку інформаційних технологій характеризується різким зростанням обсягів гетерогенних даних, отриманих із різноманітних джерел, які часто є слабо документованими, нерегулярними та різнорідними за типами змінних. В умовах такої структурної складності особливого значення набувають методи, що дозволяють виконувати класифікацію, порівняння та кластеризацію наборів даних без необхідності оперувати повними копіями інформації. Замість цього виникає потреба у компактних, інтерпретованих і технологічно придатних представленнях, які зберігають ключові статистичні властивості даних та дозволяють здійснювати їх подальший аналіз.

Попри наявні математичні та алгоритмічні підходи до аналізу табличних даних, відкритими залишаються питання оптимального визначення типів змінних у слабо структурованих наборах даних, побудови компактних репрезентацій, які б враховували гетерогенність змінних, а також створення універсальних метрик, що дозволяють порівнювати різнорідні структури даних. Особливо актуальною є задача побудови графових моделей подібності між наборами даних, що відкриває можливості застосування спектральних методів, виявлення кластерної структури та аналізу прихованих залежностей.

Дослідження, присвячені аналізу та обробці гетерогенних даних, представлені у працях як вітчизняних, так і зарубіжних учених. Зокрема, питання поєднання структурованих і неструктурованих даних у задачах прогнозування розглядаються у роботі D. Zhang, C. Yin, J. Zeng, X. Yuan та P. Zhang [1], де застосовано методи глибинного навчання для формування узгоджених представлень даних. Аналіз поведінкових та соціальних даних із використанням змішаних ознак різної природи висвітлено у працях J. Errasti, I. Amigo та M. Villadangos [2]. Питання побудови компактних і інформативних представлень ознак досліджувалися у роботі J. Cui, M. Cui,

В. Xiao та G. Li [3], де запропоновано методи зменшення розмірності з одночасним збереженням дискримінативних властивостей описів. Шаховська Н. та Загородний І. у дослідженнях [4, 5] аналізують підходи до побудови моделей і класифікації характеристик за змішаними наборами даних, що включають числові, категоріальні та часові компоненти, з акцентом на їхню практичну реалізацію в прикладних задачах.

Заслуговує на увагу окремий напрям досліджень пов'язаний із застосуванням методів машинного навчання та аналізу даних у прикладних задачах моделювання складних процесів і оптимізації обчислювальних ресурсів. Зокрема, використання інтелектуальних методів для оптимізації керування хмарними ресурсами розглядається у роботі М. J. Goswami [6]. Питання аналізу складних багатовимірних процесів і побудови моделей на основі різнорідних часових і статистичних характеристик висвітлено у працях [7,8] науковцями Malinović-Milićević S., Vykylyuk Y., Radovanović M. M. та ін. Застосування методів машинного навчання (ML), ансамблеві підходи та статистичні моделі для аналізу і порівняння складних наборів даних розглянуті авторами Hunt L., Jorgensen M. у роботі [9] та Ahmad A., Khan S. S. у праці [10].

Зважаючи на наведене вище, розробка теоретичних засад та інформаційної технології, яка включає класифікацію змінних, формування компактних дескрипторів (CDR), обчислення узагальненої структурної метрики подібності (DISS), побудову графів і дерев подібності, а також їх реалізацію у масштабованому хмарному середовищі, є актуальною науково-прикладною задачею.

Об'єкт дослідження. Процес порівняння гетерогенних табличних наборів даних зі слабо документованих джерел.

Предмет дослідження. Моделі та методи класифікації змінних, створення компактних представлень гетерогенних табличних наборів даних, побудова графів подібності для ефективного порівняння та структурування корпусів даних за мінімальних припущень щодо схеми.

Мета і завдання дисертаційного дослідження. Метою дослідження є розробка уніфікованої типо-орієнтованої інформаційної системи для ефективного порівняння гетерогенних табличних наборів даних з невідомих або слабо документованих джерел за допомогою створення компактних представлень даних з використанням зваженої метрики структурної подібності даних та побудова масштабованої хмарної інформаційної системи для формування графів та дерев подібності корпусів з мінімізацією витрат на зберігання.

Відповідно до поставленої мети в дисертаційній роботі розв'язуються такі основні задачі:

- дослідити сучасні методи компактного подання даних, класифікації змінних та метрик подібності;
- розробити процедуру конструювання компактних представлень даних для різних типів змінних (факторних (таблиці частот), часових (гістограми з фіксованими інтервалами), числових (вектори моментів) та рядкових (вектори TF-IDF));
- створити зважену, типо-специфічну метрику подібності, що об'єднує відстані між гетерогенними представленнями через навчені або встановлені користувачем ваги;
- побудувати граф подібності (та пов'язане з ним дерево) для наборів даних, використовуючи лише компактні, а не повні копії даних;
- розробити інформаційну технологію для обробки великих корпусів табличних даних з оптимізацією витрат на зберігання із реалізацією в AWS-середовищі (S3, EMR, Glue, Athena, Airflow);
- виконати порівняльні експерименти для виявлення схожих наборів даних та оцінки структурної якості корпусу.

Методи дослідження. У дисертації застосовано методи теорії ймовірностей та математичної статистики для моделювання властивостей змінних та оцінювання статистичних характеристик компактних представлень; спектральні методи лінійної алгебри для дослідження

структурних властивостей графа подібності; методи машинного навчання та метрики між розподілами для побудови уніфікованої метрики структурної подібності інформації даних; аналітичні та числові методи для дослідження властивостей запропонованих моделей; хмарні технології AWS для реалізації масштабованої інформаційної технології.

Наукова новизна. В дисертаційній роботі:

1) вперше:

- розроблено інтегровану модель CDR+DISS, що об'єднує класифікацію змінних, компактне подання даних та структурну метрику подібності для побудови графових моделей корпусу даних;
- запропоновано метод побудови дерева подібностей (similarity tree) на основі мінімального остовного дерева та DISS-метрики, який дозволяє визначати кластерну структуру без попередньої інформації про дані;
- створено багатостадійну хмарно-орієнтовану інформаційну технологію класифікації та кластеризації табличних даних із використанням AWS-сервісів.

2) набуло подальшого розвитку:

- теорія структурної подібності табличних даних шляхом узагальнення статистичних відстаней для різних типів змінних;
- графово-спектральні підходи до кластеризації, у яких подібність між наборами даних задається компактними дескрипторами, а кластерна структура визначається через спектр матриці подібності.

Зв'язок роботи з науковими програмами, планами та темами.

Дисертаційне дослідження виконано відповідно до планів науково-дослідницьких робіт:

- кафедри програмного забезпечення комп'ютерних систем Чернівецького національного університету імені Юрія Федьковича за держбюджетною тематикою: «Дослідження, моделювання та розробка програмного забезпечення складних динамічних систем» (Державний реєстраційний номер 0121U109232);

– кафедри математичних проблем управління і кібернетики Чернівецького національного університету імені Юрія Федьковича за держбюджетною тематикою: «Інформаційні технології в аспекті сучасних задач прийняття рішень» (Державний реєстраційний номер 0121U109159).

Публікації. За темою дисертації опубліковано 13 робіт, в яких подано результати теоретичних досліджень, алгоритмічних рішень та технологічної реалізації інформаційної системи класифікації та подібності табличних даних. З них 6 статей у рецензованих виданнях (2 – в журналах, що індексується у наукометричній базі SCOPUS, 4 – в українських фахових виданнях), у матеріалах міжнародних наукових конференцій – 6 робіт.

Особистий внесок здобувача. Всі основні результати дисертаційного дослідження отримані автором самостійно. Зазначимо внесок автора у спільних працях: у праці [1] (Додаток А) Кириченку Є. належить аналіз джерел, підготовка огляду літератури (Кириченко О. – концептуалізація, формальний аналіз); у праці [2] (Додаток А) Кириченку Є. – концептуалізація, методологія, формальний аналіз, дослідження, написання та підготовка оригінальної чернетки, написання-рецензування та редагування, візуалізація (Малику І. – концептуалізація, методологія, формальний аналіз, написання та підготовка оригінальної чернетки, написання-рецензування та редагування, візуалізація, авторський нагляд; Горбатенку М. – методологія, формальний аналіз; Лукашіву Т. – концептуалізація, написання, рецензування та редагування, авторський нагляд, адміністрування проєкту); у праці [3] (Додаток А) Кириченку Є. належить теоретична та практична розробка положень, програмна реалізація, редагування рукопису (Кириченко О. – постановка задачі, керівництво дослідженням, рецензування та редагування); у праці [4] (Додаток А) Кириченку Є. – проєктування архітектури системи, програмне забезпечення, розробка та тестування програмного забезпечення, написання та редагування (Малику І. – загальне керівництво, концептуалізація); у праці [5] (Додаток А)

Кириченку Є. – проведення дослідження, редагування рукопису (Малику І. – постановка задачі, визначення загальної схеми досліджень, обговорення результату); у праці [6] (Додаток А) Кириченку Є. – огляд літературних джерел, написання, рецензування та редагування (Малику І. – постановка задачі, рецензування); у праці [7] (Додаток А) Кириченку Є. – проведення дослідження, написання рукопису (Малику І. – концептуалізація, методологія, загальне керівництво); у праці [8] (Додаток А) Кириченку Є. – проведення досліджень, аналіз та опис результатів (Малику І. – постановка задачі, обговорення результатів); у праці [9] (Додаток А) Кириченку Є. належить збір та аналіз даних, обговорення результатів (Малику І. – концептуалізація дослідження, розробка методології, аналіз результатів); у праці [10] (Додаток А) Кириченку Є. – проведення дослідження, аналіз та опис результату (Малику І. – визначення загальної схеми дослідження); у праці [11] (Додаток А) Кириченку Є. – концептуалізація, методологія, формальний аналіз, дослідження, написання (Малику І. – постановка задачі, загальне керівництво); у праці [12] (Додаток А) Кириченку Є. – концептуалізація, методологія, формальний аналіз, дослідження, підготовка ілюстративних матеріалів (Кириченко О. – концептуалізація, обговорення результатів, Філіпчук О. – обговорення результатів, рецензування та редагування).

Апробація результатів дисертації. Основні положення та наукові результати дисертаційної роботи доповідалися та обговорювалися на семінарах кафедр програмного забезпечення комп'ютерних систем та математичних проблем управління і кібернетики, а також на Всеукраїнських та Міжнародних наукових і науково-практичних конференціях: «Проблеми інформатики та комп'ютерної техніки (ПІКТ)» (Чернівці, 2023-2025), «Інформаційні технології: наука, техніка, технологія, освіта, здоров'я (MicroCAD)» (Харків, 2024), The 13th International Conference on Electronics, Communications and Computing's (IC ECCO) (м. Кишинів, Молдова, 2024), «Science in the Context of Modern Challenges and Prospects» (м. Утрехт,

Нідерланди, 2025).

Структура та обсяг роботи. Дисертаційна робота є завершеним дослідженням, загальним обсягом 224 сторінки друкованого тексту (з них: 145 сторінок основного тексту, 27 сторінок – література, 34 сторінки – додатки). Дисертація складається зі вступу, чотирьох розділів із підрозділами, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1

СПЕКТРАЛЬНІ ПІДХОДИ ДО КЛАСТЕРИЗАЦІЇ У BIG DATA ТА ГЕТЕРОГЕННИХ МЕРЕЖЕВИХ СИСТЕМАХ

1.1 Кластеризація у Big Data та роль випадкових матриць

В епоху стрімкого зростання обсягів цифрової інформації одним із центральних викликів сучасної аналітики даних є побудова ефективних методів структуризації великих та неоднорідних масивів. Зокрема, у контексті Big Data, де дані характеризуються високою розмірністю, нерегулярністю та випадковістю структурних зв'язків, класичні підходи до кластеризації часто є занадто складними. Це зумовлено тим, що багато реальних мережових систем – соціальні графи, комунікаційні структури, біологічні мережі, системи штучного інтелекту та ML системи, веб-простори – мають складну топологію, яку природно описувати через графові моделі, а їх параметри та зв'язки – через випадкові матриці [11–14].

Проблема визначення оптимальної кількості кластерів у таких системах є ключовою, оскільки саме вона визначає подальшу якість сегментації, інтерпретованість результатів та ефективність алгоритмів обробки. У класичних методах кластеризації така кількість задається або вручну, або потребує додаткових евристик [11]. У великих мережах з випадковими вагами цей підхід втрачає практичну цінність [11, 15].

У зв'язку з цим зростає інтерес до спектральних методів кластеризації, які ґрунтуються на аналізі власних значень та власних векторів відповідних матриць суміжності або лапласіанів графів [16]. Спектральні алгоритми мають суттєву теоретичну перевагу: вони дозволяють досліджувати глобальну структуру графу через поведінку його спектра, що робить їх природними інструментами для задач кластеризації великих мереж [17, 18].

Окремо варто підкреслити, що спектральний підхід до аналізу великих графових структур відіграє ключову роль не лише у класичних алгоритмах кластеризації, але й у сучасних методах ML та штучного інтелекту, зокрема

нейронних мереж. У багатьох графових моделях машинного навчання – зокрема, у графових нейронних мережах (Graph Neural Networks), спектральних GNN-архітектурах, random-walk-ембеддингах (ансамблевих моделях – DeepWalk, node2vec), PageRank-подібних моделях – основою обчислень є саме стохастична або нормована матриця переходів, що формується з матриці суміжності графу [19–21]. Спектральні властивості таких матриць визначають, яким чином інформація поширюється між вершинами, як формується представлення вузлів та як виявляється прихована структурна організація мережі. Поведінка власних значень стохастичної матриці безпосередньо впливає на згладжування сигналу на графі, дифузійні процеси та якість побудованих ембеддингів, що робить спектральні результати інтегральною частиною теоретичної основи сучасних нейронних мереж. Таким чином, розгляд спектральних властивостей випадкових стохастичних матриць є не лише фундаментальним для кластеризації, але й безпосередньо пов'язаний із методологією машинного навчання, у якій графові моделі займають дедалі важливіше місце [22, 23].

Особливо важливим є те, що більшість реальних графів, які виникають у Big Data, не лише великі за розміром, але й мають випадкову або стохастичну природу, а елементи їх матриць суміжності можуть бути розподілені згідно з розподілами, що мають важкі хвости або навіть не мають скінченних моментів другого порядку [24, 25]. Це порушує припущення багатьох класичних тверджень, зокрема центральної граничної теореми [26], що традиційно застосовується для аналізу спектральних властивостей випадкових матриць [27].

У таких умовах теорія випадкових матриць відіграє вирішальну роль, оскільки дозволяє [28–30]:

- формально описувати поведінку спектра при збільшенні розмірності матриць;
- встановлювати закономірності для власних значень без вимоги скінченності моментів;

- моделювати структурні особливості великих графів;
- математично обґрунтовувати критерії кластеризації.

У статтях автора [31–34] розглянуто саме такий підхід: графові структури описуються через випадкові матриці суміжності, а аналіз кластерної структури – через власні значення стохастичної матриці переходів, яка є нормованою версією матриці суміжності. Зокрема, у [31] сформульовано метод визначення кількості кластерів, що базується на асимптотичній поведінці власних значень стохастичної матриці у випадку розподілів з важкими хвостами елементів [35]. А у [32] узагальнено класичні результати теорії випадкових матриць на випадок відсутності другого моменту, що є характерним для багатьох Big Data систем.

Таким чином, застосування випадкових матриць у кластеризації великих графових систем дає змогу сформувати строге математичне підґрунтя для визначення оптимальної кількості кластерів, ґрунтуючись не на евристиках, а на спектральних властивостях моделі. Це робить такі підходи одними з найбільш перспективних для аналізу високорозмірних даних, у яких стохастичність і наявність важких хвостів у розподілах є невід’ємними властивостями [36–38].

1.2 Графи великої розмірності та їх стохастичні моделі

У сучасних задачах аналізу даних графові структури є одним із найбільш природних і гнучких способів моделювання взаємозв’язків. У багатьох прикладних системах (соціальних мережах, біологічних структурах, транспортних та комунікаційних мережах, розподілених інформаційних системах) дані подаються у вигляді великорозмірних графів, де вершини відображають об’єкти, а ребра – інтенсивність або факт взаємодії між ними. Такі графи часто є не просто великими за кількістю елементів, а ще й характеризуються високим ступенем випадковості, нерегулярності та статистичної неоднорідності [39, 40].

Основними математичними об’єктами, які будуть використані нами в

цьому розділі, є граф та відповідна стохастична матриця.

Означення 1.1 Граф будемо позначати через $G = (V, E)$, де $V = \{v_1, v_2, \dots, v_N\}$ – множина вершин графа, а $E = \{e_1, \dots, e_m\}$ – множина ребер графа, що з'єднують вершини із множини V . Припустимо, що граф задається матрицею суміжності A :

$$A = A_{N \times N},$$

де елемент A_{ij} рівний ваговому коефіцієнту між вершинами i та j .

Поряд із випадковими матрицями [28], важливим класом матриць є стохастичні матриці, які визначаються як матриці перехідних ймовірностей в теорії дискретних ланцюгів Маркова [41]. Крім того, важливою властивістю стохастичних матриць є розміщення власних значень. Наведемо означення стохастичної матриці.

Означення 1.2 Матриця P називається стохастичною, якщо

- всі елементи матриці P є невід'ємними;
- сума елементів по рядках (або стовпцях) дорівнює одиниці.

Співвідношенням (1.1) буде задаватись стохастична матриця

$$P_{ij} = \frac{A_{ij}}{\sum_{k=1}^N A_{ik}}, \quad i, j = 1, \dots, N, \quad (1.1)$$

що відповідає графу G та задається матрицею суміжності A . Основним предметом дослідження в роботі буде матриця P .

Розглядатимемо власні значення матриці A в порядку спадання їх абсолютних величин, тобто

$$|\lambda_1(A)| \geq |\lambda_2(A)| \geq \dots \geq |\lambda_N(A)|.$$

Скористаємось фактом, що одним із власних значень стохастичної матриці, що відповідає незвідному ланцюгу Маркова з стохастичною матрицею P , є одиниця та всі власні значення стохастичної матриці не перевищують її за абсолютним значенням. Доведення цього факту є наслідком теореми Перрона – Фробеніуса [42], яка стосується властивостей

додатних матриць та їх власних значень.

Твердження 1.1 [42] Нехай матриця $A = A_{N \times N}$ – нерозкладна матриця із невід’ємними елементами. Тоді існує додатне власне значення $\lambda_1(A)$ матриці A , для якого виконуються нерівності:

$$|\lambda_i(A)| \leq \lambda_1(A), i = 2, \dots, N;$$

$$\min_{j=1, \dots, N} \sum_{i=1}^N A_{ij} \leq \lambda_1(A) \leq \max_{j=1, \dots, N} \sum_{i=1}^N A_{ij}; \quad (1.2)$$

$$\min_{i=1, \dots, N} \sum_{j=1}^N A_{ij} \leq \lambda_1(A) \leq \max_{i=1, \dots, N} \sum_{j=1}^N A_{ij}. \quad (1.3)$$

Власне значення $\lambda_1(A)$ є простим коренем характеристичного рівняння та даному власному значенню відповідає власний вектор $e_1(A)$ із невід’ємними координатами.

Враховуючи нерівності (1.2), (1.3) для стохастичної випадкової матриці P , визначеної в означенні 1.1, отримаємо наслідок із теореми Перрона – Фробеніуса про рівність одиниці власного значення $\lambda_1(P)$, тобто

$$\lambda_1(P) = 1 \quad (1.4)$$

для довільної стохастичної матриці.

Згідно твердження 1.1 вірні наступні співвідношення

$$\begin{cases} \lambda_1(P) = 1, \\ |\lambda_i(A)| \leq 1, i \geq 1. \end{cases}$$

Теорема 1.1 Нехай виконуються наступні умови [31]:

1. Всі елементи матриці $A = A_{N \times N}$ є незалежними випадковими величинами та мають однаковий розподіл

$$A_{ij} \sim Distr,$$

де розподіл $Distr$, для якого виконуються умови

$$P(A_{ij} < 1) = 0, \quad EA_{ij} < \infty.$$

2. Щільність розподілу $Distr$ задовольняє умову

$$f_A(x) \leq \frac{M}{x^\alpha}, \quad \alpha \in (2, 3],$$

де $M > 0$.

3. Елементи матриці переходу дискретного ланцюга Маркова за один крок P_{ij} визначається співвідношенням (1.1).

Тоді буде мати місце наступна збіжність

$$\lim_{N \rightarrow \infty} \max(|\lambda_2(P)|, \dots, |\lambda_N(P)|) = 0.$$

Доведення. Зауважимо, що елементи матриці (1.1)

$$P_{ij} = \frac{A_{ij}}{\sum_{k=1}^N A_{ik}}$$

набувають значень із інтервалу $[0,1]$. Крім того на основі однакового розподілу випадкових величин A_{ij} та їх незалежності, отримаємо наступні співвідношення

$$E(P_{ij}) = E\left(\frac{A_{ij}}{\sum_{k=1}^N A_{ik}}\right) = \frac{1}{N},$$

$$E(P_{ij}^2) \leq O\left(\frac{1}{N}\right).$$

Для доведення справедливості останнього співвідношення, припустимо що $f_A(x), x \in R$ – щільність розподілу елемента матриці. Тоді щільність розподілу елемента матриці P_{ij} буде наступною

$$\begin{aligned} f_{P_{ij}}(x) &= \\ &= \int_{R_+^{N-1}} f_A\left(\frac{x(x_2 + \dots + x_N)}{1-x}\right) f_A(x_2) \dots f_A(x_N) \frac{(x_2 + \dots + x_N)}{(1-x)^2} dx_2 \dots dx_N = \\ &= (N-1) \int_{R_+^{N-1}} f_A\left(\frac{x(x_2 + \dots + x_N)}{1-x}\right) f_A(x_2) \dots f_A(x_N) \frac{(x_2 + \dots + x_N)}{(1-x)^2} dx_2 \dots dx_N \leq \\ &M(N-1) \int_{R_+^{N-1}} \left(\frac{x(x_2 + \dots + x_N)}{1-x}\right)^{-\alpha} f_A(x_2) \dots f_A(x_N) \frac{(x_2 + \dots + x_N)}{(1-x)^2} dx_2 \dots dx_N \leq \end{aligned}$$

$$\begin{aligned} &\leq M * EA_{ij} * (N-1)^{1-\alpha} \frac{(1-x)^{\alpha-2}}{x^\alpha} = \frac{K}{(N-1)^{\alpha-1}} \frac{(1-x)^{\alpha-2}}{x^\alpha} \leq \\ &\leq \frac{K}{N-1} \frac{(1-x)^{\alpha-2}}{x^\alpha} = O\left(\frac{1}{N}\right) \frac{(1-x)^{\alpha-2}}{x^\alpha}, \end{aligned}$$

де $K = M * EA_{ij}$.

Розглянемо матрицю $\tilde{P}$, елементи якої визначаються співвідношеннями

$$\tilde{P}_{ij} = E(P_{ij}) = \frac{1}{N}.$$

Власними значеннями даної матриці будуть

$$\lambda_1(\tilde{P}) = 1, \quad \lambda_2(\tilde{P}) = \dots = \lambda_N(\tilde{P}) = 0.$$

Для доведення теореми, скористаємося наступною нерівністю

$$\sum_{i=1}^N (\lambda_i(A) - \lambda_i(B))^2 \leq Tr(A - B)^2,$$

Враховуючи той факт, що $\alpha \in (2,3]$ та взявши в якості матриць A , B відповідно матриці P і $\tilde{P}$, отримаємо

$$\sum_{i=2}^N E(\lambda_i(P) - \lambda_i(\tilde{P}))^2 = \sum_{i=2}^N E\lambda_i^2(P) \leq TrE(P - \tilde{P})^2.$$

Враховуючи незалежність елементів матриці A , отримаємо

$$\begin{aligned} \sum_{i=2}^N E(\lambda_i(P) - \lambda_i(\tilde{P}))^2 &\leq \sum_{i=1}^N D(P_{ii}) = \\ &= O\left(\frac{N}{N^{\alpha-1}}\right) = O\left(\frac{1}{N^{\alpha-2}}\right), \end{aligned}$$

що і доводить твердження теореми.

Зауваження 1.1 Для спрощення викладок в теоремі 1.1 було зроблено припущення про носій розподілу $Distr$, а саме накладено умову

$$P(A_{ij} < 1) = 0.$$

Цю умову можна послабити до умови $P(A_{ij} < 0) = 0$, припустивши що $P(A_{ij} > 1) > 0$. В даному випадку

$$\begin{aligned}
& f_{P_{ij}}(x) = \\
& = (N-1) \int_{R_+^{N-1}} f_A\left(\frac{x(x_2 + \dots + x_N)}{1-x}\right) f_A(x_2) \dots f_A(x_N) \frac{x_2 + \dots + x_N}{(1-x)^2} dx_2 \dots dx_N = \\
& = (N-1) \int_{B^{(N)}} f_A\left(\frac{x(x_2 + \dots + x_N)}{1-x}\right) f_A(x_2) \dots f_A(x_N) \frac{x_2 + \dots + x_N}{(1-x)^2} dx_2 \dots dx_N + \\
& + (N-1) \int_{\overline{B^{(N)}}} f_A\left(\frac{x(x_2 + \dots + x_N)}{1-x}\right) f_A(x_2) \dots f_A(x_N) \frac{x_2 + \dots + x_N}{(1-x)^2} dx_2 \dots dx_N,
\end{aligned}$$

де множина $B^{(N)}$ визначається співвідношенням

$$B^{(N)} = \{x_2 + x_3 + \dots + x_N \leq N-1\}.$$

Тоді

$$\begin{aligned}
& f_{P_{ij}}(x) \leq \\
& \leq (N-1) \int_{B^{(N)}} f_A\left(\frac{x(x_2 + \dots + x_N)}{1-x}\right) f_A(x_2) \dots f_A(x_N) \frac{x_2 + \dots + x_N}{(1-x)^2} dx_2 \dots dx_N + \\
& + M(N-1) \int_{R_+^{N-1}} \left(\frac{x(x_2 + \dots + x_N)}{1-x}\right)^{-\alpha} f_A(x_2) \dots f_A(x_N) \frac{x_2 + \dots + x_N}{(1-x)^2} dx_2 \dots dx_N.
\end{aligned}$$

Другий випадок аналогічно із доведеною теоремою рівний $O\left(\frac{1}{N^{\alpha-1}}\right)$.

Згідно наших припущень щодо щільності $f_A(x)$, отримаємо що

$$\begin{aligned}
& (N-1) \int_{B^{(N)}} f_A\left(\frac{x(x_2 + \dots + x_N)}{1-x}\right) f_A(x_2) \dots f_A(x_N) \frac{x_2 + \dots + x_N}{(1-x)^2} dx_2 \dots dx_N \leq \\
& \leq K(N-1) \int_{B^{(N)}} f_A(x_2) \dots f_A(x_N) \frac{x_2 + \dots + x_N}{(1-x)^2} dx_2 \dots dx_N \leq \\
& \leq K_1(N-1)a^{N-2} \leq \frac{K_1}{N^2}
\end{aligned}$$

для великих N , де

$$K_1 = (N - 1)EA_{ij}, \quad a = \int_0^1 f_A(x)dx < 1.$$

Як було зауважено в роботах [43, 31] дослідження спектральних властивостей матриці P (1.1) з випадковими, незалежними, однаково розподіленими елементами A_{ij} з скінченним моментом вище другого, зводиться до дослідження квазістохастичної матриці $\bar{P}$, елементи якої визначаються за наступними співвідношеннями

$$\bar{P}_{ij} = \frac{A_{ij}}{N\mu},$$

де $\mu = EA_{ij}$.

Підхід використання матриці $\bar{P}$ являється ефективним лише в тому випадку, коли $E|A_{ij}| < \infty$. Проте у випадку $E|A_{ij}| = \infty$ використання матриці $\bar{P}$ є неможливим, тому для дослідження матриці P (1.1) потрібно переглянути дослідження спектральних властивостей, що базуються на близькості P та $\bar{P}$. Узагальнимо результати роботи [31] на випадок $E|A_{ij}| = \infty$ розглянемо наступну теорему щодо близькості власних значень матриці P та константної матриці $\bar{P} = \left(\frac{1}{N}\right)_{i,j=1}^N$.

Теорема 1.2 [32]. Нехай елементи матриці суміжності A є незалежними, однаково розподіленими та $P(A_{ij} > 0) = 1$. Тоді має місце збіжність

$$\max_{i=2,\dots,N} E(\lambda_i(P) - 1)^2 \rightarrow 0$$

при $N \rightarrow \infty$, де елементи матриці P розміщені в порядку спадання за абсолютними значеннями, тобто

$$1 = \lambda_1(P) \geq |\lambda_2(P)| \geq |\lambda_3(P)| \geq \dots \geq |\lambda_N(P)| \geq 0.$$

Доведення. Насамперед зауважимо, що власними значеннями матриці

$$\bar{P} = \begin{pmatrix} \frac{1}{N} & \cdots & \frac{1}{N} \\ \vdots & \ddots & \vdots \\ \frac{1}{N} & \cdots & \frac{1}{N} \end{pmatrix}$$

являються $\lambda_1(\bar{P}) = 1, \lambda_2(\bar{P}) = \cdots = \lambda_N(\bar{P}) = 0$. Крім того, згідно однакового розподілу для випадкових величин A_{ij} , одержимо що

$$P_{ij} = E \left(\frac{A_{ij}}{\sum_{k=1}^N A_{ik}} \right) = \frac{1}{N}.$$

За побудовою одержимо, що $P_{ij} \in [0,1]$. Слід зауважити, що у випадку існування скінченного математичного сподівання ми одразу одержимо випадок розподілів з важкими хвостами наступного типу

$$f_A(x) \sim \frac{1}{x^a}, a > 2, x \rightarrow \infty.$$

Такого типу розподіли були розглянуті в роботі [31]. Тому надалі в доведенні припустимо, що $a \leq 2$. У цьому випадку згідно припущень теореми математичне сподівання дорівнює нескінченності. Зауважимо, що розподіл випадкової величини P_{ij} буде мати вигляд

$$f_{P_{ij}}(x) = \int_{R_+^{N-1}} f_A \left(\frac{x(x_2 + \cdots + x_N)}{1-x} \right) f_A(x_2) \cdots f_A(x_N) \frac{x_2 + \cdots + x_N}{(1-x^2)} dx_2 \cdots dx_N$$

з другим моментом

$$\begin{aligned} EP_{ij}^2 &= \\ &= \int_0^1 x^2 \int_{R_+^{N-1}} f_A \left(\frac{x(x_2 + \cdots + x_N)}{1-x} \right) f_A(x_2) \cdots f_A(x_N) \frac{x_2 + \cdots + x_N}{(1-x)^2} dx_2 \cdots dx_N dx = \\ &= \int_{R_+^{N-1}} \left(\int_0^1 \frac{x^2}{(1-x)^2} f_A \left(\frac{x(x_2 + \cdots + x_N)}{1-x} \right) dx \right) (x_2 + \cdots + x_N) f_A(x_2) \cdots f_A(x_N) dx_2 \cdots dx_N \\ &= \int_{R_+^N} \left(\frac{x}{x_2 + \cdots + x_N + x} \right)^2 f_A(x) f_A(x_2) \cdots f_A(x_N) dx dx_2 \cdots dx_N. \end{aligned}$$

Враховуючи властивості функції $\left(\frac{x}{x_2+\dots+x_N+x}\right)^2$ та умови накладені на щільність розподілу A_{ij} , тобто $f_A(x) = 0, x < 0$ одержимо що $EP_{ij}^2 = O\left(\frac{1}{N^2}\right)$, причому враховуючи той факт що $EP_{ij} = \frac{1}{N}$, одержимо що $D(P_{ij}) = o\left(\frac{1}{N}\right)$.

Враховуючи даний факт, одержимо що

$$\begin{aligned} \max_{i=2,\dots,N} E(\lambda_i(P) - 1)^2 &\leq \sum_{i=2}^N E(\lambda_i(P) - 1)^2 \leq \sum_{i=2}^N E\lambda_i^2(P) \leq E(Sp(P - \bar{P})) \leq \\ &\leq N * o\left(\frac{1}{N}\right) = o(1) \rightarrow 0 \end{aligned}$$

при $N \rightarrow \infty$.

Теорема 1.2 доведена.

Таким чином, встановлена теорема є ключовим елементом теоретичної частини дисертації та формує математичну основу для застосування спектральних методів у задачах кластеризації великих випадкових графів.

1.3 Гіпотези кластерної структури графу

Згідно отриманих результатів теореми 1.2, власні значення матриці із однорідними випадковими елементами при великій розмірності будуть все більше скупчуватися відносно 0, окрім першого власного значення, яке дорівнює 1. В контексті алгоритмів визначення оптимальної кількості кластерів зробимо наступні додаткові припущення (гіпотези) [32]:

H1: Кількість кластерів (k) являється сталою для довільного N ;

H2: Розміри кластерів зростають, тобто

$$\lim_{N \rightarrow \infty} N_i = \infty,$$

де N_i – розмір i -го кластеру;

H3: Кластери являються співвимірними, тобто

$$\lim_{N \rightarrow \infty} \inf \frac{N_i}{N_j} \in (0; \infty), \quad \lim_{N \rightarrow \infty} \sup \frac{N_i}{N_j} \in (0; \infty);$$

Н4: Середня суміжність елементів в кластерах тісніша ніж відповідна суміжність між кластерами, тобто виконується співвідношення

$$EA_{ij} < L \min(EA_{ii}, EA_{jj}),$$

де елементи $i, j \in E$ належать до різних кластерів, $L \gg 1$. У випадку нескінченного математичного сподівання для EA_{ij} та EA_{ii} дану умову можна замінити умовою «достатньої віддаленості» відповідних розподілів, яку можна характеризувати наприклад розходженням Кульбака – Лейблера [44, 45] або відстань Хеллінгера [46].

За виконання чотирьох припущень, які відображають взаємодію між кластерами, на основі теореми 1.2 можна показати, що при достатньо великих N спектральні властивості стохастичної матриці графу P будуть еквівалентні наступній детермінованій стохастичній матриці

$$\bar{P} = \begin{pmatrix} P^{(1)} & Err^{(1,2)} & \dots & Err^{(1,k_{opt})} \\ Err^{(2,1)} & P^{(2)} & \dots & Err^{(2,k_{opt})} \\ \vdots & \vdots & \ddots & \vdots \\ Err^{(k_{opt},1)} & Err^{(k_{opt},2)} & \dots & P^{(k_{opt})} \end{pmatrix},$$

де елементи матриць $P^{(i)}$ та матриць міжкластерних переходів $Err^{(i,j)}$ визначені наступним чином

$$P_{kl}^{(i)} = \left(1 + \frac{1}{L}\right)^{-1} \frac{1}{N_i}, k, l = 1, \dots, N_i;$$

$$Err_{kl}^{(i,j)} = \left(1 + \frac{1}{L}\right)^{-1} \frac{1}{L(N - N_i)}, k = 1, \dots, N_i; l = 1, \dots, N_j.$$

Таким чином, власні значення, які близькі до 1, будуть відповідати матрицям $P^{(i)}$, з іншого боку власні значення близькі до 1 будуть відповідати матрицям помилок. Таким чином, базуючись на результатах теореми 1.1 [31], теореми 1.2 [32] та результатах роботи [43] оптимальну кількість кластерів можна визначати наступним чином

$$k_{opt} = \#\left\{\lambda_i(P): |\lambda_i(P)| > \frac{1}{\alpha\sqrt{N}}\right\},$$

де α – додатковий параметр.

Зауважимо, що гіпотези Н1–Н4 формують математичне підґрунтя для коректного застосування спектрального підходу до визначення кількості кластерів у великорозмірних випадкових графах.

1.4 Модельний приклад

Приклад [32]. Розглянемо приклад розподілу з важкими хвостами, які задовольняють умови теореми 1.2:

$$f(x) = f(x; K, \alpha) = \frac{K^{\alpha-1}}{(x+K)^\alpha}, K > 0, x > 0,$$

де $\frac{K^{\alpha-1}}{\alpha-1}$ – нормуюча константа для $\alpha > 1$. Використовуючи дану щільність, знайдемо функцію розподілу даної випадкової величини

$$F(x) = \int_0^x f(t) dt = 1 - \frac{1}{\alpha-1} \left(\frac{K}{x+K} \right)^{\alpha-1}, x > 0.$$

Таким чином

$$F^{-1}(y) = \frac{K}{((1-y)(\alpha-1))^{\frac{1}{\alpha-1}}} - K, y \in (0,1).$$

Згідно методу оберненого перетворення [47] одержимо, що $F^{-1}(u) \sim f(x)$, де $u \sim U(0,1)$.

Отже, ми можемо моделювати значення випадкової матриці випадковими елементами $A \sim f(x; K, \alpha)$. Для подальшого моделювання будемо використовувати наступні параметри:

- *кількість кластерів, $k_{opt} = 3$;*
- *розмірність кластерів, $N = \{400, 320, 750\}$;*
- *значення параметру $K_{in} = 6$ всередині кластерів;*
- *значення параметру $K_{out} \in \{0.01, 1, 10\}$ між кластерами;*
- *значення параметру $\alpha_{in} = 1.1$ всередині кластерів;*
- *значення параметру $\alpha_{out} = 1.5$ між кластерами.*

На рис. 1.1 – 1.3 відображено, що визначення оптимальної кількості

кластерів на пряму залежить від подібності між розподілами $f(x; K_{in}, \alpha_{in})$ та $f(x; K_{out}, \alpha_{out})$ і, крім того, більш точною являється оцінка оптимальної кількості кластерів на основі визначення кількості власних значень, що перевищують значення $\frac{1}{a\sqrt{N}}$. З рис. 1.1 – 1.2 при $K_{out} \ll K_{in}$ чітко відслідковується оптимальне значення $k_{opt} = 3$, що відповідає змодельованим значенням.

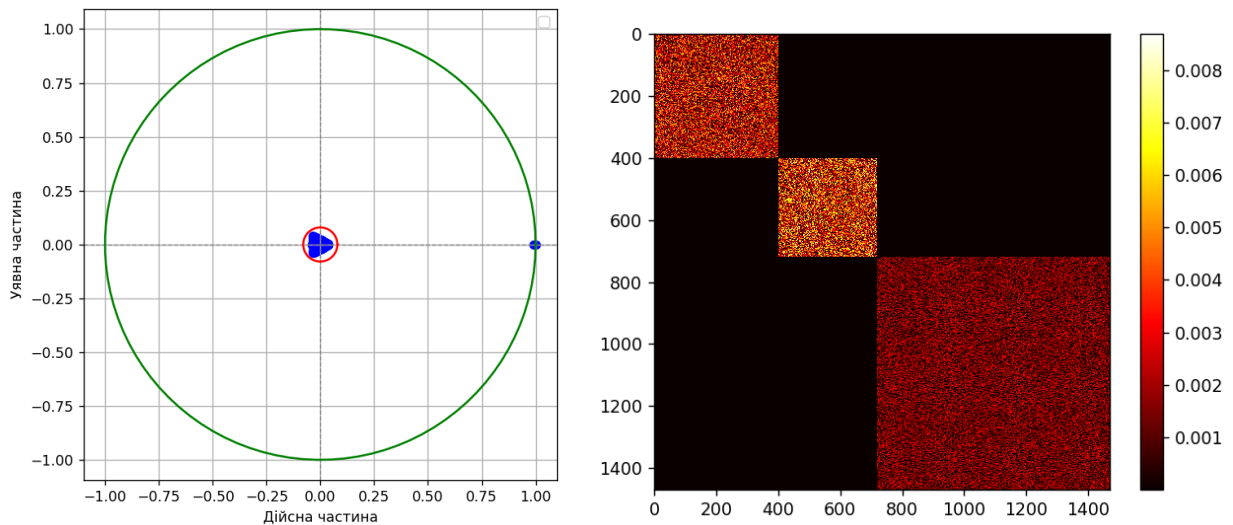


Рис. 1.1 – Власні значення матриці P (зліва) та
теплова матриця (справа) при $K_{out} = 0.01$

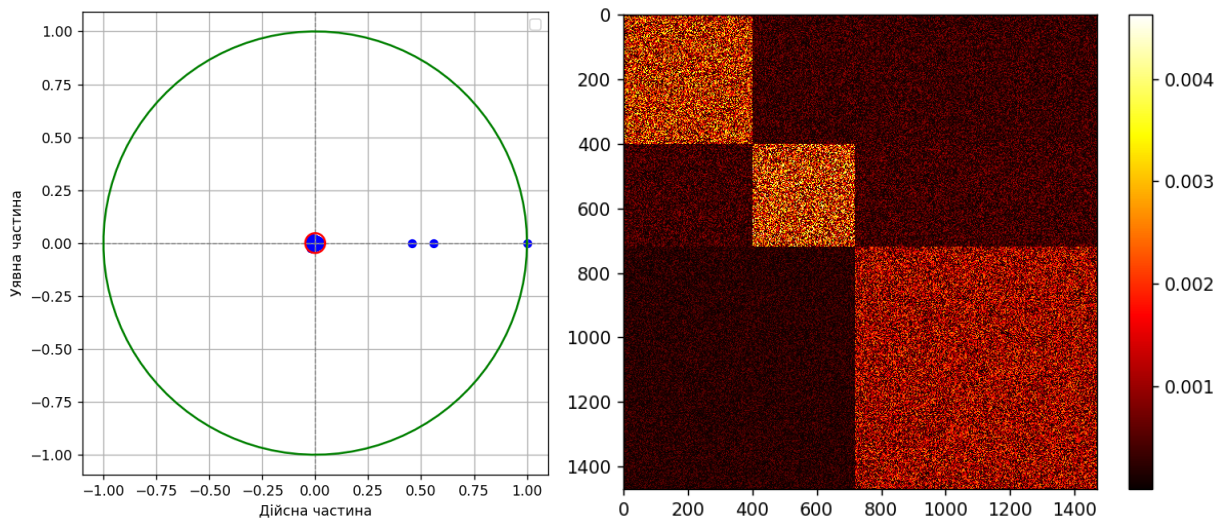


Рис. 1.2 – Власні значення матриці P (зліва) та
теплова матриця (справа) при $K_{out} = 1$

З іншого боку при $K_{in} \ll K_{out}$ відбувається протилежний ефект, а саме

деякі власні значення прямують до (-1) при $\frac{K_{in}}{K_{out}} \rightarrow 0$. У цьому випадку міжкластерні зв'язки являються сильнішими за зв'язки, що утворені всередині кластерів, проте знову ж таки чітко прослідковують три кластери, як і були розглянуті при моделюванні.

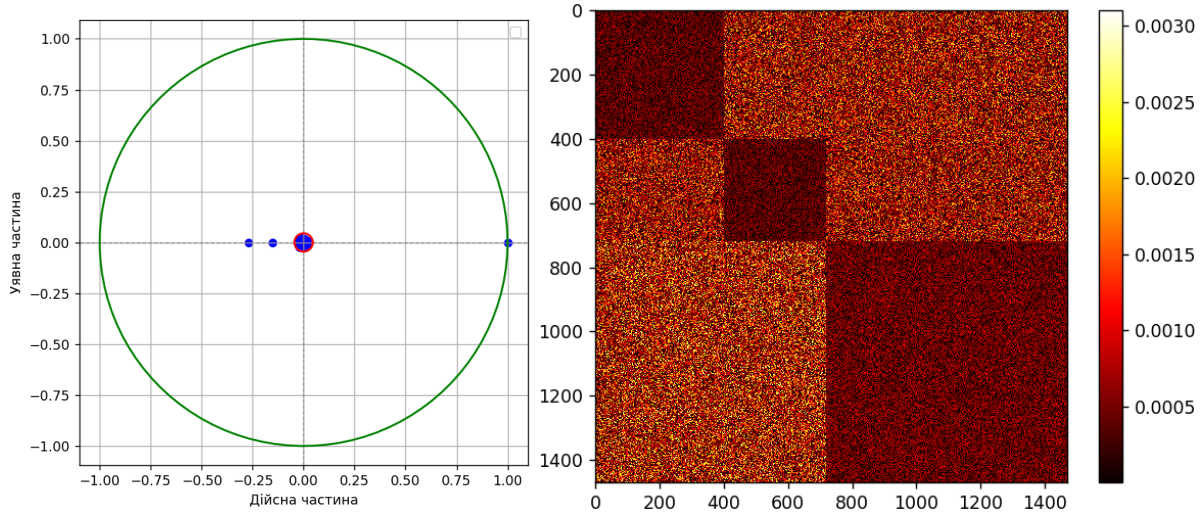


Рис. 1.3 – Власні значення матриці P (зліва) та
теплова матриця (справа) при $K_{out} = 10$

Крім того, слід зауважити, що відстань Хеллінгера при фіксованому $K_{in} = 6$ буде рівною

$$\begin{aligned}
 H^2(K_{out}) &= \int_0^\infty \left(\sqrt{\frac{K_{in}^{\alpha_{in}-1}}{(x+K_{in})^{\alpha_{in}}}} - \sqrt{\frac{K_{out}^{\alpha_{out}-1}}{(x+K_{out})^{\alpha_{out}}}} \right)^2 dx = \\
 &= \int_0^\infty \left(\frac{K_{in}^{\alpha_{in}-1}}{(x+K_{in})^{\alpha_{in}}} - 2 \frac{K_{in}^{\frac{\alpha_{in}-1}{2}} K_{out}^{\frac{\alpha_{out}-1}{2}}}{(x+K_{in})^{\frac{\alpha_{in}}{2}} (x+K_{out})^{\frac{\alpha_{out}}{2}}} + \frac{K_{out}^{\alpha_{out}-1}}{(x+K_{out})^{\alpha_{out}}} \right) dx = \\
 &= 2 - 2 K_{in}^{\frac{\alpha_{in}-1}{2}} K_{out}^{\frac{\alpha_{out}-1}{2}} \int_0^\infty \frac{1}{(x+K_{in})^{\frac{\alpha_{in}}{2}} (x+K_{out})^{\frac{\alpha_{out}}{2}}} dx.
 \end{aligned}$$

Останній доданок, визначається через гіпергеометричну функцію [48] та мінімальне значення відстані Хеллінгера [46], можна знайти чисельними методами, використовуючи початкове значення $K_{out} = K_{in}$.

1.5 Прикладні задачі Big Data та значення запропонованого методу кластеризації

Розроблена теоретична модель кластеризації на основі спектральних властивостей стохастичних випадкових матриць має безпосереднє застосування у багатьох реальних системах Big Data, де класичні алгоритми кластеризації є недостатньо ефективними або теоретично необґрунтованими. Особливу складність становлять моделі, у яких елементи матриць суміжності не мають скінченних моментів або підпорядковуються розподілам з важкими хвостами, що часто зустрічається у практичних задачах аналізу великих мереж.

Важливість застосування спектральних методів у таких умовах полягає в тому, що вони дозволяють виявляти структурну організацію графів без необхідності застосування евристичних припущень або передбачення кількості кластерів. Розроблений підхід ґрунтується на строгих математичних результатах та забезпечує формальний механізм визначення оптимальної кількості кластерів навіть у моделях з високим ступенем випадковості.

Нижче наведено ключові приклади реальних систем, у яких застосування запропонованої методології є особливо доцільним.

Так, у соціальних мережах (Facebook, Twitter/X, Instagram, TikTok тощо) структура зв'язків між користувачами є надзвичайно великою та нерегулярною. Рівень взаємодії між окремими парами користувачів має властивість розподілу з важкими хвостами: одні користувачі мають неймовірно велику кількість зв'язків, тоді як більшість – обмежену [49].

У таких умовах матриця суміжності має елементи з важкими хвостами і, як наслідок, класичні методи кластеризації нестійкі а внутрішні й міжкластерні зв'язки не піддаються простому аналізу.

Запропонований у роботах [31, 32] спектральний критерій дозволяє визначати кількість стійких спільнот, оскільки власні значення матриці переходів диференціюють групи користувачів із високою внутрішньою

взаємодією.

Ще одним напрямком застосування запропонованої методології є вебпростори та інтернет-мережі. Опис структур вебпростору є одним із ключових застосувань теорії випадкових матриць. Зокрема, теоретичні основи аналізу мережових структур детально розглянуто в роботі [50], де встановлено зв'язок між ентропією мережі та її спектральними характеристиками. Крім того, у роботі [51] досліджено спектральні властивості стохастичних матриць, що є математичною основою для моделювання посилань між вебсторінками. Оскільки реальні дані (гіперпосилання, кліки) часто характеризуються відсутністю другого моменту, експоненційними розбіжностями або кластеризованою структурою, встановлений авторами круговий закон дозволяє чітко відокремити випадковий шум від структурних сегментів вебпростору (тематичних груп, доменів) за допомогою аналізу відхилень від граничного спектрального розподілу.

Також у великих мережах (Big Data) потоки трафіку мають складну нелінійну структуру, що створює значні виклики для традиційних методів виявлення аномалій. Мережовий трафік характеризується динамічними шаблонами поведінки, які можуть змінюватися в часі через легітимні зміни в роботі системи, а також через зловмисні дії. Класичні статистичні підходи часто виявляються неефективними для виявлення складних атак, таких як DDoS (Distributed Denial of Service) або несанкціонований доступ, оскільки вони не здатні адекватно враховувати багатовимірну природу даних та нелінійні залежності між характеристиками трафіку.

Спектральна кластеризація надає потужний інструментарій для виявлення аномальних шаблонів поведінки, які неможливо ідентифікувати стандартними методами. Основна перевага підходу полягає в можливості виявлення кластерів довільної форми в багатовимірному просторі ознак через аналіз власних векторів матриці подібності мережевого трафіку. Останні дослідження демонструють значний прогрес у цьому напрямку: зокрема,

розроблено адаптивну матрицю подібності, яка автоматично підлаштовується під мінливий мережевий трафік у реальному часі, що суттєво підвищує точність виявлення аномалій в динамічних умовах [52]. Цей підхід дозволяє системам виявлення вторгнень (IDS) ефективно адаптуватися до змін у структурі легітимного трафіку, одночасно зберігаючи високу чутливість до потенційних загроз.

Практичне застосування спектральної кластеризації для виявлення кібератак дозволяє відокремлювати нормальну поведінку мережі від аномальної шляхом аналізу спектральних властивостей графу мережевих з'єднань. Власні значення та власні вектори матриці Лапласа графу містять інформацію про глобальну структуру мережевих взаємодій, що робить можливим виявлення координованих атак та прихованих аномальних підмереж навіть за умов високого рівня шуму в даних.

Актуальним застосуванням спектральних методів кластеризації є декомпозиція великих енергосистем на автономні керовані зони (Microgrids), що є критичним для запобігання каскадним відмовам та блекаутам у складних розподілених мережах. Сучасні енергомережі генерують гігабайти даних із сенсорів щохвилини, що створює потребу в ефективних методах аналізу та сегментації для забезпечення стабільності системи. Удосконалені алгоритми спектральної кластеризації дозволяють ідентифікувати зони передачі електроенергії на основі аналізу перевантажень у мережі, забезпечуючи оптимальний розподіл навантаження та мінімізацію втрат [53].

Особливу актуальність ці методи набувають в контексті інтеграції відновлюваних джерел енергії. Нещодавні дослідження запропонували підхід, що враховує нестабільність сонячної та вітрової генерації як додаткову вагу в графі енергосистеми [54]. Це дозволяє формувати кластери з урахуванням стохастичної природи відновлюваних джерел, забезпечуючи більш стійку роботу системи за умов високої частки «зеленої» енергії. Спектральний аналіз матриці Лапласа графу енергомережі виявляє природні точки поділу системи, мінімізуючи міжкластерні потоки енергії та

підвищуючи автономність окремих зон.

Варто зазначити, що спектральна кластеризація також використовується для аналізу кореляцій між фінансовими активами, оскільки дозволяє виділяти групи акцій, що рухаються синхронно не за формальними ознаками (сектор, географія), а за поведінковими шаблонами та прихованими економічними зв'язками. Традиційні методи кластеризації часто не здатні виявити складні нелінійні залежності в багатовимірних фінансових часових рядах, тоді як спектральний підхід через аналіз графа кореляцій розкриває глибинну структуру ринкових взаємозв'язків.

Інтеграція спектральних методів з алгоритмами машинного навчання [55] та штучного інтелекту дозволяє автоматично виявляти приховані структури в масивах фінансових даних, відсіювати ринковий "шум" від реальних сигналів економічних циклів через аналіз власних векторів матриці кореляцій. Методологія ідентифікує кластери активів з сильними внутрішніми зв'язками та слабкими міжкластерними кореляціями, що є критичним для оптимізації співвідношення ризик-прибуток в умовах нестабільних ринків та великих обсягів даних, відкриваючи нові перспективи для побудови адаптивних інвестиційних стратегій.

Не менш важливим є аналіз даних про переміщення мільйонів людей у великих мегаполісах. На сучасному етапі для аналізу трафіку генеруються величезні обсяги просторово-часових даних, які потребують ефективних методів обробки для оптимізації громадського транспорту та міського планування. Спектральні методи кластеризації дозволяють виявляти функціональні зони міста на основі потоків трафіку, даних мобільного зв'язку та траєкторій руху транспортних засобів, розкриваючи приховані шаблони міської мобільності.

Перевага спектрального підходу полягає в можливості групувати траєкторії руху, які мають однакову "форму" або поведінковий шаблон, навіть якщо вони розділені в часі або просторі. Метод обмеженої спектральної кластеризації дозволяє ідентифікувати інформацію про дорожні

смуги, враховуючи відстані між сусідніми траєкторіями та апіорні знання через визначення обмежень типу “обов’язковий зв’язок” та “заборонений зв’язок”, що значно покращує результати кластеризації, особливо у випадках зміни кількості смуг або їх ширини [56]. Поєднання спектрального аналізу графів з глибоким навчанням дозволяє прогнозувати як інформацію низького рівня (майбутні траєкторії), так і високого рівня (поведінку учасників руху), моделюючи просторові взаємодії між транспортними засобами через зважені динамічні геометричні графи.

Багато біологічних мереж (метаболическі мережі, нейронні зв’язки) мають природну кластерну організацію, тоді як ваги зв’язків часто характеризуються розподілами з важкими хвостами через біологічну варіативність, нелінійність процесів та масштабність систем. Власні значення та власні вектори матриці переходів дозволяють визначати функціональні модулі таких систем через оптимізацію модулярності мережі.

Дослідження метаболических мереж демонструють важливість кластерного аналізу для розуміння клітинного метаболізму. Кластерний аналіз тканино-специфічних метаболических мереж для окремих зразків з первинних пухлинних локалізацій дозволяє групувати різні біологічні зразки, при цьому техніки кластеризації розділяють мережі залежно від їх взаємної подібності [57].

Аналогічні принципи модульної організації спостерігаються і в нейронних мережах мозку. Покращені методи спектральної кластеризації для точного виявлення мереж спокою мозку дозволяють точно розділяти великомасштабні функціональні мережі мозку в стані спокою з даних МРТ [58]. Геометрично-орієнтовані версії спектральної кластеризації розроблені для ідентифікації заздалегідь визначеної кількості кластерів у наборі кореляційних матриць функціональних мереж мозку [59]. Ці методи дозволяють виявляти найрепрезентативніші модулі з індивідуальних зважених матриць зв’язності, враховуючи індивідуальні відмінності у функціональній архітектурі мозку.

Запропоновані результати можуть бути застосованими до Grid-систем, розподілених обчислювальних структур та великих кластерів серверів. У таких системах взаємодії між вузлами мають випадкову природу, а навантаження розподілені нерівномірно, що створює складні виклики для традиційних методів управління ресурсами.

Планування завдань та балансування навантаження в хмарних обчисленнях представляють собою складні NP-важкі оптимізаційні проблеми, які часто призводять до неефективного використання ресурсів, підвищеного енергоспоживання та подовжених часів виконання [60]. Нещодавні дослідження демонструють ефективність кластерних підходів: визначення кластерів через спектральний аналіз матриці переходів дозволяє оптимізувати маршрути обчислень, балансувати ресурси та сегментувати систему на підмножини з подібними характеристиками. Алгоритм спектральної кластеризації використовується для кластеризації хмарних сервісів, після чого формується набір кандидатів на основі кластерів через порівняння функцій сервісу та узгодження параметрів [61], що значно підвищує точність та швидкість розподілу ресурсів у масштабних розподілених системах.

Спектральні методи кластеризації є універсальним, масштабованим та теоретично обґрунтованим підходом, що виявляється ефективним у випадках, коли класичні методи не працюють через різкі коливання, шум або відсутність моментів вищого порядку. Як продемонстровано в різноманітних галузях, застосування спектрального аналізу надає універсальний інструментарій для виявлення прихованих структур у складних даних.

У контексті Big Data це дозволяє будувати моделі, які не спираються на припущення про нормальність розподілів чи лінійність залежностей, а використовують глибинні властивості випадкових матриць та спектральної теорії графів.

Перевага спектральних методів полягає в їх здатності працювати з даними, що характеризуються розподілами з важкими хвостами, високими

коливаннями та відсутністю стандартних статистичних властивостей, які є типовими для реальних Big Data систем. Таким чином, метод кластеризації на основі власних значень представляє собою фундаментальний інструмент сучасного аналізу даних, що поєднує теорію випадкових матриць з практичним застосуванням до найрізноманітніших задач машинного навчання та штучного інтелекту.

Висновки до розділу 1

У цьому розділі сформовано теоретичне підґрунтя для дослідження кластеризації графових даних за допомогою спектральних методів та випадкових матриць у контексті великих обсягів даних. Основну увагу приділено побудові математичної моделі, яка описує структуру випадкового графу на основі матриці суміжності з незалежними однаково розподіленими елементами, включно з випадками розподілів з важкими хвостами, та відсутності моментів високого порядку.

Показано, що природною і фундаментальною моделлю для аналізу графових даних є стохастична матриця переходів, утворена нормуванням матриці суміжності. Така матриця зберігає глобальні структурні властивості графу та дозволяє застосовувати спектральний апарат теорії марківських процесів. На основі теореми Перрона–Фробеніуса встановлено наявність домінантного власного значення, що є початковою точкою для спектрального аналізу кластерної структури [62, 63].

Ключовим результатом є асимптотична теорема про збіжність власних значень стохастичної матриці, сформульована у роботі [31]. Вона демонструє, що навіть за умов розподілів з важкими хвостами та відсутності другого моменту спектр матриці переходів має передбачувану поведінку при великій розмірності, а саме: всі власні значення, окрім першого, тяжіють до нуля при виконанні загальних умов на щільність розподілу.

У роботі [32] цей результат було використано для побудови детермінованої блочної моделі, спектрально еквівалентної випадковій

стохастичній матриці в околі власного значення 1. На основі цього було показано, що для графу, що має k кластерів, у спектрі стохастичної матриці виникає рівно k власних значень, які залишаються відокремленими від нуля та наближаються до 1 при $N \rightarrow \infty$. Решта власних значень прямує до нуля, що забезпечує чітку спектральну диференціацію кластерної структури.

Сукупність гіпотез H1–H4 формує математичний каркас для опису кластеризованих графів та гарантує коректність переходу до блочної моделі. Ці гіпотези забезпечують:

- сталу кількість кластерів при зростанні графу;
- зростання розмірів кожного кластеру;
- співвимірність кластерів;
- статистичну віддаленість внутрішніх та міжкластерних розподілів.

На основі цих властивостей сформульовано критерій визначення оптимальної кількості кластерів, який ґрунтується на кількості власних значень стохастичної матриці, що перевищують певний поріг, близький до одиниці. Такий підхід є універсальним, ефективним та застосовним у широкому спектрі моделей Big Data, включно з графами, що мають розподіли з важкими хвостами.

Завершуючи теоретичну частину, можна сформулювати такі основні висновки розділу:

1. *Спектральні властивості випадкових стохастичних матриць* забезпечують надійний інструмент для виявлення кластерів у великих графах.

2. *Асимптотична збіжність власних значень* є фундаментом для побудови кластерного критерію.

3. *Еквівалентність випадкової та детермінованої блочної моделі* дозволяє точно відобразити кластерну структуру графу.

4. *Критерій визначення кількості кластерів*, заснований на власних значеннях, є формально обґрунтованим та стійким до особливостей розподілів з важкими хвостами.

5. Запропонована *методологія добре узгоджується* з потребами сучасних Big Data систем, у яких класичні методи часто виявляються непридатними.

Таким чином, розділ 1 формує повноцінну теоретичну основу для подальшого викладення результатів дисертаційного дослідження.

Основні наукові результати розділу опубліковані в працях [31–34].

РОЗДІЛ 2

МАТЕМАТИЧНІ ОСНОВИ ТА АЛГОРИТМИ КОМПАКТНОГО ПРЕДСТАВЛЕННЯ ДАНИХ

Проблема ефективного представлення, аналізу та порівняння різнорідних табличних даних є фундаментальною проблемою сучасної інженерії даних, ML та великих аналітичних систем (Big Data). В умовах постійного зростання обсягу інформації, що надходить з різних джерел, таких як корпоративні бази даних, відкриті репозиторії, сенсорні мережі, бізнес-звіти та вебсервіси, традиційні підходи, що передбачають зберігання та обробку повних наборів записів, є занадто ресурсозатратними.

Повне сканування, копіювання та аналіз даних з усіх елементів великих таблиць коштує багато грошей та обчислювальної потужності. Це включає витрати на введення/виведення, пам'ять та сховище. Крім того, навіть часткове дублювання таких наборів збільшує обсяг даних у сховищі та ускладнює інтеграцію або порівняння даних з різних джерел у майбутньому. Це вимагає розробки ефективних методів агрегації даних, які зберігають основні характеристики наборів даних без надмірного дублювання або надлишковості інформації.

Проблема є складною, коли джерело даних невідоме або погано задокументоване. Часто немає формалізованої схеми, опису змінних або навіть впевненості в тому, до якого типу належить кожен стовпець. Це поширена ситуація для відкритих урядових порталів, файлів журналів та аналітичних звітів у форматах CSV або Excel, де структура може бути неповною, непослідовною або містити помилки [64]. У таких умовах застосування класичних методів попередньої обробки (ETL [65], очищення даних [66], зіставлення схем [67]) обмежене або неможливе без попереднього створення узагальненого представлення набору даних [68].

Для вирішення згаданих вище проблем найкраще використовувати компактне представлення даних (CDR), яке зберігає важливі статистичні,

категоріальні та структурні особливості набору даних без необхідності перегляду кожного запису [69]. Таке представлення діє як «форма» або дескриптор для даних, дозволяючи порівнювати набори на основі узагальнених показників, а не окремих рядків.

Основна ідея цього підходу полягає в заміні повного набору даних набором невеликих дескрипторів, які будуються для кожної змінної на основі її типу, такого як числовий, категоріальний, часовий або текстовий. Для цього необхідно створити математичний апарат, який дозволяє:

- класифікувати змінні в табличних даних незалежно від джерела;
- будувати їх узагальнене (компактне) представлення;
- визначати систему метрик, які кількісно оцінюють подібність між такими представленнями;
- реалізувати розрахунок цих метрик у масштабованому середовищі (наприклад, Apache Spark або AWS EMR).

Ми використовуємо термін компактне представлення даних (CDR), а також вводимо метрику структурної подібності інформації про дані (DISS) для порівняння табличних даних. Головна мета цього розділу – дослідити наявні та створити альтернативні методи та математичні основи для створення компактних представлень табличних даних, визначити метрики структурної подібності між ними. Зрештою, процес порівняння великих, різноманітних наборів інформації можна звести до порівняння невеликих, стандартизованих дескрипторів. Це значно зменшує кількість необхідних обчислень, робить результати легшими для розуміння та дає змогу використовувати результати в задачах класифікації, кластеризації та моніторингу якості даних.

У цьому розділі деталізуємо типізацію змінних і побудову CDR, формалізуємо метрику DISS і описуємо побудову графа, після чого у розділі 3 розглянемо хмарну реалізацію та приклади застосувань наведемо у розділі 4.

2.1 Аналіз задач визначення подібності та формування вимог до компактних представлень даних

Зазначимо декілька основних підзадач, для яких визначення подібності між даними та зберігання їх компактних копій відіграє ключову роль [70]:

1. **Керовані даними методи.** Методи, керовані даними, застосовуються для аналізу змін основних характеристик даних під час ухвалення рішення про необхідність повторного обчислення певної моделі. Зазвичай такі обчислення (перерахунки) проводяться для перевірки центральної гіпотези про відсутність суттєвих змін у даних під час побудови математичних моделей на основі алгоритмів машинного навчання та систем штучного інтелекту. У [71] роботі основна увага приділяється дослідженню змін моделей, зумовлених змінами гіперпараметрів даних, на основі яких ці моделі будуються. У статті показано певний зв'язок між гіперпараметрами даних та відповідними моделями, побудованими на їх основі. Крім того, розглядаються деякі аспекти змін даних та метрик подібності між ними.

2. **Хмарні обчислення та передача даних.** Як зазначено у дослідженнях [72], передача даних між різними пристроями може призводити до перевантаження системи, тому необхідно створювати компактні копії даних, які дають змогу робити висновки про дані без аналізу їх на кожному пристрої окремо, враховуючи, що моделі, як правило, будуть однаковими. Таким чином, передавання компактних копій даних суттєво спрощує процеси оброблення та передавання даних.

3. **Побудова моделей машинного навчання та їх аналіз** [73].

Під час аналізу даних у ML моделях використовуються лише окремі характеристики даних, без урахування всіх інших властивостей. Наведемо простий мотиваційний приклад для кращого розуміння сутності цієї проблеми. Припустимо, що дані зберігаються у табличному форматі і що один зі стовпців описує залежну змінну y , тоді як інші стовпці описують набір незалежних змінних $(x_1, \dots, x_p)$. Також припустимо, що основна модель

базується на множинній лінійній регресії, у якій метод найменших квадратів використовується як основний метод оцінювання параметрів моделі

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p.$$

У цьому випадку оцінки параметрів можуть бути знайдені таким чином

$$\hat{\beta} = (X^T X)^{-1} X^T y, \quad (2.1)$$

де

$$X = \begin{pmatrix} 1 & x_{11} & \dots & x_{1p} \\ 1 & x_{21} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \dots & x_{Np} \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{pmatrix}.$$

Таким чином, для відтворення моделі немає потреби розглядати весь набір даних – достатньо використати певні характеристики даних. Під час оцінювання (2.1) цими характеристиками є числові параметри даних, а саме середні значення та коваріаційна матриця.

Виділимо кілька основних критеріїв для компактних копій даних, які будемо враховувати під час побудови:

- **Розмір.** Розмір створених копій даних відіграє ключову роль, оскільки цей показник визначає ефективність створення. Зазвичай розмір копії повинен бути в кілька разів меншим за оригінал, однак він також залежатиме від швидкодії системи, яка використовується для її генерації. Тому накладається така умова на розмір копії даних:

$$Size_{CD} \ll Size_B,$$

де $Size_{CD}$ це – розмір компактної копії даних (у байтах), а $Size_B$ – порогове значення розміру даних, яке буде перевищене під час створення копії.

- **Модель.** Компактна копія даних повинна містити характеристики, необхідні для побудови основної моделі, на основі якої робитимуться висновки про взаємозв'язки в даних і перевірятимуться висунуті в дослідженнях гіпотези. Як зазначалося вище, для лінійної моделі такими характеристиками є числові показники, які повністю дають змогу відтворити регресію та оцінити коефіцієнти регресії. Варто зазначити, що

більшість моделей дозволяють використовувати лише частину числових характеристик даних, ігноруючи інші. Тут також слід відзначити, що у випадку використання ансамблевих методів машинного навчання показники дублюватимуться залежно від розміру ансамблевого методу [73], тому знову буде необхідно аналізувати як розмір ансамблю, так і характеристики, на основі яких обчислюється кожна модель.

- **Основні дослідницькі гіпотези.** Перевірка основних гіпотез дослідження та отримання відповідей на наукові запитання потребують використання різних статистичних гіпотез, які знову ж таки базуються на численних показниках табличних даних. Цей пункт тісно пов'язаний із попереднім, оскільки статистичні тести, подібно до основних моделей машинного навчання, потребують числових або інших характеристик даних з урахуванням специфіки базової моделі.

- **Зв'язок з іншими даними та можливість розширення моделі.** Цей пункт передбачає можливість збереження певних характеристик, які не обчислюються безпосередньо з даних, а визначаються з інших джерел. Це може бути, наприклад, ресурс, з якого отримано дані, попередні моделі, використані для аналізу ранніх версій даних у рамках підходів, керованих даними, або процедури попередньої обробки даних тощо.

Отже, питання створення компактних копій даних і стандартизації наборів даних є предметом досліджень великої кількості робіт, присвячених як різним галузям досліджень, так і різним припущенням щодо типів даних [1]. По-перше, слід відзначити, що основні підходи поділяються залежно від аналізу структурованих та неструктурованих даних [1]. Оскільки метою роботи є розгляд табличних даних, основну увагу буде зосереджено на аналізі структурованих даних. Як зазначалося раніше, головна увага приділяється табличним даним, де кожен стовпець представляє певну змінну. Оскільки нас цікавлять моделі машинного навчання, одним із головних критеріїв під час створення компактної копії даних буде тип стовпця в таблиці. Виходячи з вимог більшості моделей машинного навчання, виділимо

такі типи стовпців, які розглянемо нижче:

Числові змінні. Змінні, що представляють числові характеристики певного об'єкта або явища, є найпростішими для обробки та найчастіше використовуються в обчисленнях. Умовно ці змінні можна поділити на підкатегорії залежно від досліджуваних характеристик, наприклад, цілочисельні змінні, дробові змінні тощо [74, 75]. Числові змінні відіграють ключову роль у більшості алгоритмів машинного навчання, оскільки вони безпосередньо можуть бути використані в математичних операціях без додаткового перетворення. Для задач регресії, класифікації та кластеризації числові змінні часто становлять основу простору ознак [73, 76].

- *Факторні змінні.* Цей тип змінних характеризує величини, які можуть набувати кількох значень, зазвичай нечислових, що лише характеризують рівні або категорії. До цього типу належать бінарні, номінальні та порядкові змінні [77, 78]. Варто зазначити, що багато моделей машинного навчання, зокрема лінійна регресія, потребують перетворення факторних змінних у фіктивні числові змінні. Це перетворення створює бінарні індикатори для кожної категорії, що дозволяє включити категоріальну інформацію в математичні моделі [79, 80].

- *Часові змінні.* Змінні, що відображають дату й час у заздалегідь визначеному форматі [81].

- *Рядкові змінні.* Цей тип змінних подає інформацію у вигляді рядків без сталої довжини, чітко визначеної структури або фіксованих правил створення. Найчастіше такі змінні характеризують результати комунікації в соціальних мережах і використовуються для визначення загальних настроїв респондентів, умов голосування чи емоційного стану [2, 82]. Рядкові змінні відрізняються від факторних змінних, представлених як рядки, тим, що вони мають високу варіативність, унікальність та не обмежуються фіксованим набором категорій. Обробка таких змінних часто вимагає спеціалізованих методів, таких як токенізація, векторизація через TF-IDF або використання вбудовувань слів [2, 83].

Перш за все розглянемо приклади стиснення графічних даних (малюнків, графіків тощо). Представником цього напрямку є робота [3], у якій запропоновано зменшення розмірності даних із використанням методу Bag-of-Features (BOF). У цій роботі подання малюнків здійснюється на основі щільності розподілу кольорів пікселів у RGB-представленні, виходячи з деякої сім'ї розподілів $M = \{p(*|\theta), \theta \in \Theta \subset R^S\}$, де θ це параметр розподілу сім'ї з густинами $p(*|\theta)$, і автори пропонують використання метрики Рімана

$$D_F(\theta_1, \theta_2) = \min_{\substack{\theta(*): \\ \theta(0)=\theta_1, \\ \theta(1)=\theta_2}} \int_0^1 \sqrt{\left(\frac{d\theta}{dt}\right)^T I(\theta) \left(\frac{d\theta}{dt}\right)} dt, \quad (2.2)$$

де $I(\theta)$ є інформацією Фішера, елементи якої визначаються з відношення

$$I_{ij}(\theta) = E \left[\frac{d \log(p(x|\theta))}{d\theta_i} \cdot \frac{d \log(p(x|\theta))}{d\theta_j} \right].$$

Проводиться порівняння відстані, засновану на інформації Фішера (2.2). Відзначено кілька переваг у зменшенні розмірності даних порівняно з іншими метриками, а саме косинусною відстанню, евклідовою відстанню, відстанню Геллінгера та гауссовою відстанню (гауссовим ядром). Детальніший аналіз малюнків геометричних об'єктів розглянуто в роботі [6], у якій геометричні об'єкти пропонується представляти у вигляді компактних топологічних структур (структур з напівребрами – half-edge data structures). З геометричного погляду відповідне представлення базується на використанні k -вимірного многовиду M (або пов'язаних k -вимірних многовидів) із зазначенням відповідних обмежень на ці многовиди:

- $(k - 1)$ – вимірюваний клітинний комплекс многовиду M є тотожним одній або двом $(k - 1)$ -вимірним клітинам многовиду M ;
- відкрита зірка вершини в многовиді M є гомеоморфною відкритій підмножині або R^k , або R_+^k .

Автори показують, що представлення геометричних фігур на основі k -

вимірних многовидів дозволяє, з одного боку, побудувати компактну версію об'єкта, а з іншого – зберегти деталізацію цього об'єкта. Використовуючи такий підхід, автори досягли високої точності розпізнавання облич, причому точність зросла на 5–10% порівняно з класичними методами. Для створення стиснених копій геометричних об'єктів було застосовано функцію двонапрямого розподілу відбиття (Bidirectional Reflectance Distribution Function, BRDF) [84]. Робота ґрунтується на побудові математичної моделі подання даних на основі густин розподілу, і одним із головних припущень є припущення про нормальність розподілу густини зображення об'єкта. Це припущення дає змогу істотно спростити теоретичні обчислення та звести узагальнені параметричні сімейства, розглянуті в роботі [3], до двопараметричних сімейств $N(a, \Sigma)$. Інший підхід до класифікації даних полягає у побудові латентних змінних [85], які можна розглядати як узагальнені змінні в методі головних компонент. У роботі представлено фреймворк CLaRe, який аналізує три набори даних. Показано, що розроблений підхід до побудови латентних змінних забезпечує кращі результати класифікації та кластеризації порівняно з класичними методами машинного навчання. Компактне представлення великомасштабних просторових даних розглядається в іншій роботі [86], де подання просторових даних базується на використанні мінімального обмежувального прямокутника (MBR). Концепція MBR запроваджена через простоту використання: для просторової двовимірної області a MBR визначається за таким співвідношенням

$$MBR(a) = [x^-(a), x^+(a)] \times [y^-(a), y^+(a)], \quad (2.3)$$

де

$$\begin{aligned} x^-(a) &= \inf_x \{(x, y) \in a\}, & x^+(a) &= \sup_x \{(x, y) \in a\}, \\ y^-(a) &= \inf_y \{(x, y) \in a\}, & y^+(a) &= \sup_y \{(x, y) \in a\}. \end{aligned}$$

Таким чином, подання просторових даних у R^2 пропонується за допомогою покриття прямокутниками та їх перетинів. Порівняно з поданням

просторових даних у вигляді багатокутника, цей підхід значно зменшує обсяг даних, необхідних для визначення просторового об'єкта. Автори цієї роботи представили метод оптимального подання просторової області $a \in R^2$ або $a \in R^3$ тощо, тобто мінімізації прямокутного $MBR(a)$ у визначенні (2.3).

Більш формальний підхід до класифікації табличних даних розглянуто в роботі [74]. На основі цього підходу табличні дані подаються у вигляді трійки (F, X, y) , де F – це специфікації стовпців, X – множина незалежних змінних, y – залежна змінна, яку необхідно оцінити в задачі регресії або класифікації. По-друге, розроблено нову архітектуру нейронної мережі для регресії та порівняння табличних даних, і показано, що точність розробленої нейронної мережі перевищує результати низки класичних методів (XGBoost, CatBoost, gcForest, Net-DNF та FCNN), при цьому швидкість роботи запропонованої нейронної мережі не поступається класичним підходам. Основним недоліком розробленого методу є те, що автори припускають, що специфікація F відома, хоча в більшості випадків це припущення є надто оптимістичним при порівнянні даних. Дещо інший підхід до подання табличних даних, заснований на визначенні їх складності, представлено в роботі [87], де пропонується використовувати складність запитів згідно з формалізацією Вуда:

$$QC_i = |IC_i| + |R_i| + |A_i|,$$

де IC_i – це кількість інформаційних підказок, представлених у запитанні i , R_i – кількість взаємозв'язків, які повинен визначити учасник, A_i – кількість суджень, які необхідно зробити щодо множини елементів. Таким чином, підходи, розглянуті авторами, дають змогу створювати зашифровані копії даних (опитувань) із високим ступенем захисту. Ще одним цікавим питанням, піднятим у роботі [82], є порівняння табличних і графічних даних з точки зору надійності та криптостійкості. Автори показують, що табличне представлення є більш надійним і має низку переваг порівняно з графічним. Інший підхід до створення стислих копій даних на основі нейронних мереж розглянуто в роботі [82], де пропонується використання нейронної мережі

для створення стисненої копії даних. Цей підхід є близьким до попереднього [74], і в обох випадках пропонується використання загорткових нейронних мереж. Таким чином, у зазначених роботах [74, 82] стиснені дані розглядаються як дані, отримані на проміжних етапах роботи нейронної мережі. Як зазначалося вище, для аналізу моделей машинного навчання зазвичай потрібен лише певний набір числових характеристик конкретного набору даних, який може включати вектор середніх значень, коваріаційну матрицю, тензори третього та четвертого моментів тощо. Теоретичні аспекти дослідження зменшення простору даних і, відповідно, створення стислих копій даних подано в роботі [88], де автори розглядають можливість зменшення розмірності вихідних даних із n до $\frac{C \ln(n)}{\ln(\ln(n))}$ для достатньо великих n , де C деяка універсальна константа. Таким чином, автори цієї роботи вказують на можливість зменшення розмірності даних на основі обчислення необхідних моментних функцій із наперед заданою точністю. Інший підхід до використання нейронних мереж для побудови стислих копій даних розглянуто в роботі [76], яка, на відміну від попередніх [74, 82], пропонує покрокову згортку даних за допомогою конволюцій, тобто нові фактори створюються на кожному шарі нейронної мережі

$$\tilde{x}_i^k = d(\tilde{x}_i^{k-1}, \tilde{x}_i^{k-2}),$$

де $\tilde{x}_i^k$ є значенням змінної $\tilde{x}_i$ в k -ій ітерації. Автори роботи показують, що в задачах класифікації розроблений підхід дозволяє зберегти всі взаємозв'язки у вихідних даних, відфільтрувати шум і підвищити точність оцінювання порівняно з класичними методами (Random Forest, XGBoost, LightGBM, CatBoost, TabNet, ReConTab).

Також необхідно згадати створення стислих копій графів як математичних моделей багатьох реальних явищ. Слід зазначити, що стиснення даних у випадку графового представлення має низку особливостей, пов'язаних насамперед із можливістю зменшення графа шляхом видалення вершин або ребер. Варто також відзначити, що деякі

автори пропонують завжди розглядати повні графи, оскільки видалення певних акторів (вершин графа) або зв'язків (ребер графа) може призвести до повної зміни структури групи, для якої цей граф побудовано як математичну модель. Автори роботи [89] розглядають графи як моделі баз даних для зберігання та порівняння табличних даних. Табличні дані або їх атрибути розглядаються як вершини, а як ребра – різні типи зв'язків між цими вершинами. Автори розглянули та порівняли такі системи: DEX, Neo4j і HyperGraph, а також інші, зокрема SAP HANA Graph і SQLGraph, при цьому кожна з них має свої переваги й недоліки. Відповідно до методології, запропонованої авторами, дані мають бути представлені у вигляді десяткової кортежної структури

$$G = (L_N, L_E, N, E, R, L_A, S_N, S_E, A_N, A_E), \quad (2.4)$$

де

- L_N – множина можливих міток, які можуть приймати вузли графа;
- L_E – множина можливих міток, які можуть приймати ребра графа;
- $N = (n_i, l_j)$ – множина вершин графа;
- $E = (e_i, l_j)$ – множина ребер графа;
- R містить зв'язки між вузлами, тобто початкові та кінцеві вузли ребер;
- $L_A = \{a_i\}$ – множина різних міток атрибутів графа. Іншими словами, L_A – це множина всіх різних атрибутів, що описують вузли та ребра графа;
- $S_N = \{sn_i\}$ – множина схем для типів вузлів;
- $S_E = \{se_i\}$ – множина схем для типів ребер, визначена цілком аналогічно до S_N . Кожен елемент S_E визначає допустиму схему для типу ребра;
- $A_N = \{(n_i, a_j, v_k)\}$ визначає властивості вузлів;

- $A_E = \{(e_i, a_j, v_k)\}$ описує властивості ребер.

Подання (2.4) є досить повним і містить усі необхідні дані для побудови схеми даних. Однак цей підхід добре працюватиме лише за умови, що властивості табличних даних, тобто сама структура даних, відомі.

Інший напрям у контексті дослідження – це метод реконструкції даних на основі побудови змінних із заданими асоціаціями [83, 90]. У [83] пропонується використовувати, поряд із числовими характеристиками змінних, їх коваріаційні або кореляційні матриці, тобто кожна змінна характеризуватиметься як набір моментів

$$var_i: \left\{ m_i^1 = \frac{1}{N} \sum_{j=1}^N x_{ij}, \dots, m_i^k = \frac{1}{N} \sum_{j=1}^N x_{ij}^k \right\}, \quad (2.5)$$

та кореляційною матрицею або кореляційними матрицями між різними наборами змінних (маються на увазі числові змінні та факторні змінні):

$$R = \left(r_{ij} = \frac{1}{N \sqrt{m_i^2 m_j^2}} \sum_{l=1}^N (x_{il} - m_i^1)(x_{jl} - m_j^1) \right), \quad (2.6)$$

де k є максимальним порядком моментної функції у виразі (2.5). У роботі [90] також частково розглянуто проблему створення нових класів під час формування міток у задачах кластеризації та напівкластеризації. Ця проблема тісно пов'язана з тематикою нашого дослідження, оскільки факторні змінні визначатимуться на основі розподілу міток, їх пропорцій та, можливо, різних типів взаємозв'язків, включно з кореляціями тощо. У роботі [91] також проведено аналіз зменшення розмірності змінних на основі вивчення кореляцій між ними, і запропоновано розглядати як змінні в моделях машинного навчання лише ту множину змінних, для якої виконується таке обмеження

$$Data_{compressed} = \left(x_i: \sup_{j \neq i} cor(x_i, x_j) < \rho^* \right),$$

де ρ^* є певним порогом кореляції (автори використовують $\rho^* = 0.8$ під час тестування цього підходу). Зменшення розмірності табличних даних на прикладі відстеження змін генів, представлених у вигляді зображень, розглянуто в роботі, де відстань між зображеннями визначається як манхеттенська відстань між пікселями зображення, незалежно від варіацій окремих пікселів. Таким чином, автори роботи [75] пропонують представляти табличні дані у вигляді векторів, матриць та загальних тензорів із регулярною метрикою. Інший підхід до створення компактних копій даних розглянуто в роботі [92], де вводиться поняття (λ, ε) приватного бюджету, на основі якого формується концепція близькості двох баз даних. Два набори табличних даних $M(S)$ і $M(S')$, називаються близькими з параметрами (λ, ε) , якщо

$$D(M(S_1), M(S_2)) = \frac{1}{\lambda - 1} \log E_{x \sim M(S_1)} \left(\frac{P(M(S_1) \sim x)}{P(M(S_2) \sim x)} \right) \leq \varepsilon.$$

Таким чином, автори використовують концепцію відношення правдоподібності вибірки, яка, з одного боку, спрощує розгляд подібності даних, а з іншого – вимагає додаткових припущень щодо розподілу табличних даних.

2.2 Типізація змінних

Визначення типів змінних у табличних даних є фундаментальним кроком у побудові компактних представлень даних (CDR). Основні підходи побудови стислих копій базуються на використанні моментів елементів та різноманітних видів асоціацій, таких як кореляційна матриця та тензори вищих порядків [3, 83]. Емпіричним шляхом встановлено, що для достатньо точного відтворення розподілу випадкової величини необхідно щонайменше чотири функції моментів, тобто у поданні (2.4) можна обрати $k=4$. Враховуючи цей факт, розглянемо подання кожного з типів стовпців, припускаючи, що вибірка x описує відповідний стовпець:

2.2.1 Визначення типів даних

1. Факторні змінні (бінарні, порядкові або рангові змінні).

Факторні змінні є одним з найпоширеніших типів у табличних даних і вимагають ретельного підходу до їх ідентифікації [93, 94]. Основне припущення для факторних змінних є таким

$$rel = \frac{\#\{unique\ values\ for\ x\}}{N} \leq p_{crit},$$

де N – кількість спостережень; p_{crit} – це критичне значення для співвідношення між кількістю унікальних значень та кількістю спостережень [95]. Якщо це співвідношення є малим, то з великою ймовірністю відповідна змінна є факторною або числовою цілочисельною змінною з невеликою кількістю унікальних значень. Отже, основний підалгоритм для ідентифікації факторних змінних такий:

- *Обчислити значення rel для змінної.* На цьому етапі визначається відношення кількості унікальних значень до загальної кількості спостережень. Це дає первинну оцінку категоріальності змінної.
- *Перевірити тип змінної:* якщо значення є рядковими, то змінна є з високою ймовірністю факторною [96, 97]. Однак необхідно відрізнити факторні рядкові змінні від справді рядкових змінних, які мають високу варіативність та не обмежуються фіксованим набором категорій [98, 99].
- *Провести аналіз структури значень.* Для числових значень проводиться перевірка входження до списків можливих факторних значень [93]. Наприклад, аналізується:
 - довжина значень (наприклад, дванадцять символів як у номері картки або 8 символів як у UID) [100, 101];
 - структура значень (наявність фіксованих шаблонів, регулярних виразів) [94, 100];
 - семантичне значення (чи представляють значення категорії, а чи справжні числові величини) [95, 98].

Якщо змінна проходить ці перевірки, вона з високою ймовірністю

класифікується як факторна [93]. В інших випадках змінна x вважається числовою (цілочисельною) з високою ймовірністю [102, 103].

Для підвищення точності класифікації можна застосувати додаткові евристики:

- аналіз розподілу частот: факторні змінні зазвичай мають виражені піки частот для певних категорій [104];
- перевірка на наявність природного впорядкування: це може вказувати на порядкову змінну [98, 105];
- аналіз контексту: назва стовпця та метадані можуть надати додаткову інформацію про природу змінної [80].

2. Часові змінні. Цей тип змінних можна перевірити за допомогою стандартних перевірок формату дат, які реалізовані в багатьох програмних засобах [98]. Перевірка здійснюється за списком форматів, наприклад: *рік. місяць. день* (*Y.m.d.*); *день/місяць/рік* (*d/m/Y*); *ISO 8601 формат* (*YYYY-MM-DD HH:MM:SS*) тощо. Ідентифікація часових змінних базується на аналізі структури та можливості парсингу значень як дат або часу [106].

Опишемо підхід до ідентифікації часових змінних:

- *Спроба парсингу.* Для кожного значення у стовпці робиться спроба парсингу за допомогою списку відомих форматів дат і часу [107]:
 - ISO 8601: *YYYY-MM-DD*, *YYYY-MM-DD HH:MM:SS*;
 - Європейський формат: *DD/MM/YYYY*, *DD.MM.YYYY*;
 - Американський формат: *MM/DD/YYYY*;
 - Часові мітки Unix (timestamp) [108];
 - Інші регіональні формати.

Якщо більше ніж 90% значень успішно парсяться в один з цих форматів, змінна класифікується як часова [109].

- *Валідація діапазону.* Перевіряється, чи потрапляють розпарсені дати в розумний діапазон (наприклад, від 1900 року до поточної дати плюс кілька років у майбутнє). Це допомагає відфільтрувати хибні спрацьовування, коли числові змінні випадково інтерпретуються як дати.

- *Аналіз послідовності.* Для підтвердження часової природи змінної аналізується впорядкованість значень та наявність часових патернів (регулярні інтервали, циклічність) [110, 111].

- *Перевірка метаданих.* Назва стовпця часто містить ключові слова, що вказують на часову природу: "date", "time", "timestamp", "datetime", "year", "month" тощо. Ця інформація використовується як додаткове підтвердження [112].

3. Числові змінні (неперервні та дискретні): Неперервні числові змінні можна охарактеризувати кількома основними властивостями [95]:

- кількість унікальних значень є співрозмірною з кількістю спостережень:

$$\#\{\text{unique values for } x\} \approx N$$

або у більш точній формі

$$\frac{\#\{\text{unique values for } x\}}{N} \approx 1.$$

- Неперервні змінні часто пов'язані з певним типом розподілу – для багатьох фінансових активів це нормальний, логнормальний, розподіл із важким хвостом або їхні суміші [88, 113, 114]. Наявність важких хвостів може суттєво впливати на статистичний висновок та оцінку ризиків [115, 116].

- Числові змінні підтримують повний спектр арифметичних операцій (додавання, віднімання, множення, ділення), і результати цих операцій мають змістовну інтерпретацію [117, 118].

Алгоритм ідентифікації числових змінних наступний:

- *Перевірка типу даних.* Перевіряється, чи можуть значення змінної бути перетворені в числовий формат (integer або float) [98, 120]. Якщо перетворення неможливе або призводить до великої кількості помилок, змінна не є числовою.

- *Обчислення відношення унікальних значень.* Якщо

$\frac{\#\{unique\ values\ for\ x\}}{N} > 0.5$, то змінна з високою ймовірністю є неперервною числовою [99, 120]. Якщо $0.05 < \frac{\#\{unique\ values\ for\ x\}}{N} \leq 0.5$, змінна може бути дискретною числовою або факторною з числовим кодуванням [90, 94, 121].

- *Аналіз розподілу.* Проводиться статистичний тест на відповідність відомим розподілам (наприклад, тест Колмогорова-Смірнова, тест Шапіро-Вілکا для нормальності) [122–124]. Це допомагає підтвердити неперервну природу змінної [88, 125].

- *Перевірка на дискретність.* Для розрізнення дискретних числових змінних від неперервних аналізується [126]:

- наявність лише цілочисельних значень;
- обмеженість діапазону значень;
- регулярні інтервали між значеннями.

4. Рядкові змінні. Рядкові змінні, на відміну від факторних змінних (які подані у вигляді рядків) або різних типів ідентифікаторів (`person_id`, `platform_id` тощо), є рядковими змінними з різною довжиною [98]. Отже, змінна, найімовірніше, має рядковий тип (а не факторний), якщо вона є слабо структурованою (має різну довжину записів), а кількість унікальних значень пропорційна кількості записів у даних, тобто [94]

$$0.1 \leq \frac{\#\{unique\ values\ for\ x\}}{N} \leq 1.$$

Якщо це відношення близьке до 1, це вказує на високу унікальність значень, що є характерним для рядкових даних [126, 127].

Ідентифікацію змінної можна провести за алгоритмом:

- *Виключення інших типів.* Спочатку перевіряється, чи не належить змінна до числових, часових або факторних типів [119]. Рядкові змінні ідентифікуються за принципом виключення.

- *Обчислення метрик унікальності.* Розраховуються показники унікальності значень та варіативності довжини [99, 120]. Якщо обидва

показники високі, змінна класифікується як рядкова [82].

- *Токенізація та аналіз словника.* Для підтвердження проводиться токенізація значень та аналіз розміру словника [128, 129]. Великий словник з низькою частотою повторення слів підтверджує рядкову природу змінної [130, 131].

- *Семантичний аналіз.* За можливості проводиться аналіз семантичного змісту: наявність повних речень, граматичних структур, пунктуації тощо [82].

2.2.2 Алгоритм типізації даних

Побудова компактного представлення (CDR) неможлива без коректного визначення типу змінної. Типізація виконується автоматично, що особливо важливо для великих і змішаних наборів даних, у яких відсутня явна інформація про структуру стовпців. Реалізований у інформаційній системі (Розділ 3) модуль TypeInferencer використовує комбіновані статистичні, частотні та структурні евристики.

Алгоритм типізації працює поетапно [70]:

1. Перевірка на числову природу даних.

Оцінюється частка значень, які можуть бути приведені до числового типу без втрати інформації; визначається кількість унікальних значень; аналізується відхилення та розподіл. Числовий тип встановлюється лише за умови, що розподіл змінної не має характеру токенів або категорій.

2. Виділення категоріальних змінних.

На основі частотного аналізу визначається, чи може змінна бути дискретною. Часто категоріальними є змінні з малою кількістю унікальних значень, але система підтримує широкий спектр випадків, включно з умовно категоріальними полями, такими як коди продуктів або класифікатори.

3. Виявлення рядкових змінних.

Змінні, що містять рядкові конструкції, ідентифікуються за наявністю багатослівних токенів, символів пунктуації, лексичних закономірностей.

Додатково оцінюється середня довжина рядка, варіативність tokenів і частота повторюваних фраз.

4. Ідентифікація часових параметрів.

Для розпізнавання дат і часових форматів застосовуються регулярні вирази, аналіз форматів ISO, а також евристики щодо позиційності (наприклад, рік-місяць-день). На етапі попередньої обробки показники додатково перевіряються на монотонність або наявність стійкої тенденції до зміни.

Такий підхід гарантує високу точність типізації, необхідну для коректної побудови CDR-дескрипторів, та дозволяє адаптуватися до реальних і часто реальних промислових даних.

Цей алгоритм забезпечує автоматичну класифікацію змінних без необхідності попереднього знання схеми даних, що є критично важливим для обробки гетерогенних наборів даних з невідомих або слабо задокументованих джерел [74, 87, 89]. Використання такого підходу дозволяє створювати адаптивні системи обробки даних, які можуть працювати з різноманітними джерелами інформації без необхідності ручного налаштування [132–134].

2.3 Стиснення різних типів змінних

Після етапу типізації змінних, що дозволяє класифікувати всі атрибути табличного набору за їх статистичною природою, наступним кроком є побудова компактного подання (CDR) для кожного типу змінних.

Розглянемо компактне подання даних із різними типами [70], описаними в пункті 2.2.

1. Факторні змінні.

Факторні змінні можна подати через розподіл у вигляді словника:

Variable : {*values*: { *val11*, *val12*}, *val11*: N_{val11} , *val12*: N_{val12} }},

де {*val11*, *val12*} – перелік відповідних значень; N_{val11} – кількість значень *val11*; N_{val12} – кількість значень *val12*.

Таке подання можна розширити для будь-яких категоріальних змінних, додаючи додаткову інформацію для порядкових змінних із чітко визначеним порядком значень.

2. Змінні дати й часу.

Якщо кількість значень для змінних дати й часу є невеликою (≤ 1000) можна використовувати таке саме подання, як і для факторних змінних. В іншому випадку (якщо кількість спостережень дуже велика) можна застосовувати частотний розподіл зі структурою ($date\ 1, \dots, date\ n$):

Intervals	$[D_1, D_2)$	$[D_2, D_3)$	...	$[D_k, D_{k+1})$
Frequencies	n_1	n_2	...	n_k

де

$$\min_i dat\ e_i = D_1 < D_2 < \dots < D_{k+1} = \max_i dat\ e_i.$$

Зауважимо, що для порівняння кількість інтервалів має бути фіксованою (наприклад $k = 100$). Для цього типу змінних їх можна подати у вигляді словника такої структури:

$$Variable : \{separators: \{D_1, \dots, D_{k+1}\}, Freq: \{n_1, \dots, n_k\}\}.$$

3. Числові змінні.

Для числових змінних, як і в методах машинного навчання, найкраще використовувати моменти та кількість спостережень. Отже, числові вибірки можна подати у вигляді вектора:

$$\left(N, \sum_{i=1}^N x_i, \sum_{i=1}^N x_i^2, \dots, \sum_{i=1}^N x_i^k \right),$$

де k – це фіксоване значення, достатнє для представлення числової змінної. У статистиці часто використовують $k = 4$, оскільки перші чотири моменти достатні для розуміння природи змінної. Якщо $\sum_{i=1}^N x_i^l$ є великою, можна замінити її на середнє значення:

$$\sum_{i=1}^N x_i^l \rightarrow \frac{1}{N} \sum_{i=1}^N x_i^l.$$

Таким чином, числову змінну можна узагальнити як:

$$Variable1 : \left\{ moments: \left\{ N, \sum_{i=1}^N x_i, \sum_{i=1}^N x_i^2, \dots, \sum_{i=1}^N x_i^k \right\} \right\}.$$

4. Рядкові змінні.

Для цього типу змінних можна використовувати добре відомі векторні подання, наприклад TF-IDF. У цьому випадку кожен змінну можна представити як список одиниць (усіх можливих слів або n -грам) і відповідну розріджену матрицю. Тоді змінна має вигляд:

$$Variable1 : \left\{ \begin{array}{l} n_gram_length: n; \\ units: \{unit_1, unit_2, \dots, unit_M\}; \\ tfidf_repres: SPARSE_MATRIX \end{array} \right\},$$

де n_gram_length визначений n -грам.

5. Інші характеристики.

У багатьох випадках потрібно обчислювати додаткові числові характеристики змінних – кореляцію та/або коваріацію. Це необхідно, наприклад, якщо деяка математична модель використовується для прогнозування або побудови передбачень. У такому разі доцільно обчислити кореляційну або коваріаційну матрицю:

$$COR_{cov} = \\ = \{variables: \{var1, var2, \dots\}, correlation: \{corr\}, covariance: \{cov\}\}.$$

2.4. Порівняння стиснених даних (метрика DISS)

Після створення компактних представлень даних (CDR) для кожної змінної виникає задача ефективного порівняння цих представлень між різними наборами даних. Оскільки кожен тип змінної має власну природу (дискретну, часову, неперервну чи текстову), для нього необхідно підібрати метрику, яка коректно відображає відстань між відповідними компактними дескрипторами. У цьому пункті розглядаються основні метрики для порівняння різних типів даних та вводиться уніфікована метрика подібності DISS (Data Information Structural Similarity), яка агрегує типо-специфічні

відстані в єдину оцінку подібності на рівні наборів даних [135], тобто DISS оцінює близькість двох наборів даних з точки зору їх інформаційної структури, зокрема їх статистичних, топологічних і семантичних характеристик [70].

2.4.1 Основні метрики для порівняння різних типів даних

Розглянемо основні метрики для порівняння різних типів даних:

1. Метрики для факторних змінних (бінарні, номінальні та порядкові).

Факторні (категоріальні) змінні у компактному поданні описуються функцією маси ймовірності (PMF), тобто вектором $p = [p_1, p_2, \dots, p_n]$, де p_i – частка появи i -ої категорії [36].

Порівняння двох таких векторів p_i та q_i може виконуватись за різними мірами відстані між розподілами. Для цього випадку можна використовувати одну з таких метрик:

- *Метрика Махаланобіса* з одиничною матрицею Σ , а саме [136, 137]:

$$d_M(p, q) = \sqrt{(p - q)^T \Sigma^{-1} (p - q)},$$

де $\|\cdot\|_2$ це стандартна евклідова метрика, а Σ – коваріаційна матриця;

- *Відстань Геллінгера*, яка визначається так [35]:

$$d_H(p, q) = \sqrt{\frac{1}{2} \sum_{i=1}^n (\sqrt{p_i} - \sqrt{q_i})^2},$$

- *Відстань повної варіації* (Total Variation Distance), яка визначається як [138]:

$$d_{TV}(p, q) = \frac{1}{2} \sum_{i=1}^n |p_i - q_i|.$$

- *Метрика Вассерштейна* з параметром $p = 1$.

Класичне визначення для будь-якої метрики між категоріями d_{ij} :

$$W_1(p, q) = \min_{\gamma} \sum_{i=1}^n \sum_{j=1}^n d_{ij} \cdot \gamma_{ij},$$

де:

d_{ij} – відстань між i та j категоріями;

γ_{ij} – елементи матриці плану транспортування (кількість маси, що переноситься з категорії i в категорію j), які повинні задовольняти обмеженням:

$\sum_{j=1}^n \gamma_{ij} = p_i$ (сума по рядку дорівнює вхідному розподілу p);

$\sum_{i=1}^n \gamma_{ij} = q_j$ (сума по стовпцю дорівнює цільовому розподілу q);

$\gamma_{ij} > 0$.

У нашому випадку категорії рівнозначні та не мають порядку, тоді відстань Васерштейна збігається з половиною мангеттенської відстані між векторами [139]:

$$W_1(p, q) = \frac{1}{2} \sum_{i=1}^n |p_i - q_i|.$$

Це також еквівалентно відстані повної варіації (Total Variation Distance).

2. Метрики для часових змінних.

Часові змінні представлені у CDR у вигляді гістограм або ескізів розподілу подій у часі. Тому їх природно порівнювати метриками, які враховують не лише різницю частот, але й положення подій на часовій осі.

Для часових змінних, представлених у вигляді гістограм з фіксованими інтервалами, розробляється спеціалізована метрика, яка враховує як розподіл частот, так і відмінності в часових інтервалах [72].

Розглянемо дві таблиці частот:

Intervals	$[D_1, D_2)$	$[D_2, D_3)$	...	$[D_k, D_{k+1})$
Frequencies	n_1	n_2	...	n_k

та

Intervals	$[\tilde{D}_1, \tilde{D}_2)$	$[\tilde{D}_2, \tilde{D}_3)$	...	$[\tilde{D}_k, \tilde{D}_{k+1})$
Frequencies	$\tilde{n}_1$	$\tilde{n}_2$	...	$\tilde{n}_k$

Нехай задано функцію f , що задовольняє три властивості:

- $f(0) = 0$;
- f неперервна і зростає на R_+ ;
- $\lim_{x \rightarrow \infty} f(x) = \infty$.

Тоді для двох таблиць частот визначимо метрику:

$$d(FR, \tilde{FR}) = \frac{1}{k} \sum_{i=1}^k f(|D_i - \tilde{D}_i| + |D_{i+1} - \tilde{D}_{i+1}| + |n_i - \tilde{n}_i|).$$

Два перших доданки відображають відмінності в межах інтервалів, а третій – відмінності в частотах. Хорошим вибором для функції f є функція

$$f(x) = e^{\alpha x} - 1,$$

де $\alpha > 0$ – це додатковий параметр, який визначається з тестових вибірок.

Відмітимо властивості цієї метрики:

- враховує як часові відмінності (зміщення інтервалів), так і частотні відмінності;
- якщо $d(FR, \tilde{FR}) = 0$, то таблиці повністю збігаються;
- нормалізація по k забезпечує порівнянність між гістограмами з різною кількістю інтервалів;
- параметр α контролює чутливість до відхилень: великі α підвищують вагу великих відхилень (при $\alpha \gg 1$ – метрика стає дуже чутливою навіть до малих відхилень; при $\alpha \rightarrow 0$ метрика наближається до лінійної залежності (схожої на звичайну суму різниць)). Той факт, що α визначається з "тестових вибірок", означає, що метрику можна калібрувати (навчати) під конкретну задачу);
- використання функції f робить цю метрику чутливою до значних викидів, дозволяючи регулювати величину штрафу за відхилення через параметр α .

Альтернативною метрикою для гістограм – є відстань Вассерштейна. Якщо інтервали в обох таблицях однакові ($D_i = \widetilde{D}_i$ для всіх i), що є типовою ситуацією при фіксованих глобальних інтервалах, можна використовувати відстань Вассерштейна безпосередньо на розподілах частот [86, 135, 139]:

$$W_1(n, \tilde{n}) = \sum_{i=1}^k \left| \frac{n_i}{N} - \frac{\tilde{n}_i}{\tilde{N}} \right| \cdot w_i,$$

де $N, \tilde{N}$ – загальні кількості спостережень, а w_i – вага i -го інтервалу (наприклад, його довжина в одиницях часу).

3. Метрики для числових змінних.

Числові змінні у компактному поданні описуються векторами моментів (середнє, дисперсія, асиметрія, ексцес), а тому їх порівняння зводиться до обчислення відстаней між такими векторами. Для числових змінних можна використовувати відносну різницю між змінними. Нехай маємо два вектори:

$$\begin{aligned} \vec{V}_x &= \left(N = \sum_{i=1}^N x_i^0, \sum_{i=1}^N x_i, \sum_{i=1}^N x_i^2, \dots, \sum_{i=1}^N x_i^k \right), \\ \vec{V}_y &= \left(M = \sum_{i=1}^M y_i^0, \sum_{i=1}^M y_i, \sum_{i=1}^M y_i^2, \dots, \sum_{i=1}^M y_i^k \right), \end{aligned}$$

тоді представлені вектори можна порівняти через $L1$ (мангеттенська) та $L2$ (евклідова) метрики [140]. Евклідова відстань є класичним прикладом метрики, що однаково враховує відхилення за всіма координатами.

$$d_{L2}(\vec{V}_x, \vec{V}_y) = \sqrt{\sum_{p=0}^k \left(\sum_{i=1}^N x_i^p - \sum_{i=1}^M y_i^p \right)^2}.$$

Натомість мангеттенська відстань характеризується підвищеною стійкістю до викидів, оскільки великі відхилення не домінують у загальному значенні міри, на відміну від випадку використання евклідової норми [140].

$$d_{L1}(\vec{V}_x, \vec{V}_y) = \sum_{p=0}^k \left| \sum_{i=1}^N x_i^p - \sum_{i=1}^M y_i^p \right|.$$

На теоретичному рівні встановлено тісний зв'язок між нормами простору та метриками похибки [140, 141]. Зокрема, середньоквадратична похибка (MSE) та середня абсолютна похибка (MAE) є усередненими формами норм $L1$ та $L2$, тобто відповідають евклідовій і мангеттенській відстаням. Таким чином, вибір між нормами $L1$ та $L2$ безпосередньо визначає тип похибки, що мінімізується під час навчання моделі. Для нормально розподілених помилок оптимальною з точки зору максимальної правдоподібності є модель, яка мінімізує MSE, тоді як у випадку лапласівського розподілу помилок найбільш правдоподібною є модель, що мінімізує MAE [141].

Метрика MAE (Mean Absolute Error) має безпосередню інтерпретацію як $L1$ -відстань між векторами прогнозних та фактичних значень. Метрики, засновані на абсолютних похибках, оптимізують медіану розподілу та демонструють стійкість до наявності викидів, оскільки не передбачають квадратичного зростання штрафу за великі відхилення [142].

$$MAE = \frac{1}{k+1} \sum_{p=0}^k \left| \sum_{i=1}^N x_i^p - \sum_{j=1}^M y_j^p \right|.$$

Середня абсолютна відсоткова похибка (MAPE) забезпечує масштаб-інваріантну оцінку точності, нормуючи $L1$ -похибку на абсолютні значення фактичних спостережень [143]:

$$MAPE = \frac{100\%}{k+1} \sum_{p=0}^k \left| \frac{\sum_{i=1}^N x_i^p - \sum_{j=1}^M y_j^p}{\sum_{i=1}^N x_i^p} \right|.$$

Це робить можливим порівняння якості прогнозів для часових рядів з різними діапазонами значень. Водночас MAPE має низку обмежень, а саме вона асиметрично штрафує від'ємні похибки порівняно з додатними, а також стає чисельно нестабільною при значеннях, близьких до нуля [143].

Не зважаючи на відмінності між абсолютними та відсотковими метриками похибки, обидва класи залишаються широко застосовними на практиці. Крім того, використання комбінації кількох метрик похибки, таких як MAPE та MAE, у складі функції втрат сприяє отриманню більш стійких результатів порівняно з оптимізацією за однією окремою метрикою.

Можна відмітити альтернативний підхід, який полягає у порівнянні безпосередньо чотирьох статистичних характеристик, обчислених з моментів: [144]:

$$d_{stat}(m_1, m_2) = w_\mu |\mu_1 - \mu_2| + w_\sigma |\sigma_1 - \sigma_2| + \\ + w_\gamma |\gamma_{1,1} - \gamma_{1,2}| + w_\kappa |\gamma_{2,1} - \gamma_{2,2}|,$$

де μ – середнє, σ – стандартне відхилення, γ_1 – асиметрія, γ_2 – ексцес, а w – відповідні ваги.

4. Метрики для рядкових змінних.

Для рядкових змінних представлених у вигляді розрідженої матриці (*SPARSE_MATRIX*), можна обчислити частоти n -грам і використовувати різницю між двома частотами. Проте для глибокого аналізу текстові змінні після векторизації (TF-IDF, Word2Vec або BERT) представляються у вигляді векторів високої розмірності [145, 146].

1. Для таких даних важливо оцінювати не абсолютну різницю компонент, а відносну важливість термінів у корпусі.

Частота n -грам у розрідженій матриці обчислюється за формулою:

$$q_i = \frac{\sum_{k=1}^N SPARSE_MATRIX_{ki}}{\sum_{i=1}^M \sum_{k=1}^N SPARSE_MATRIX_{ki}},$$

де *SPARSE_MATRIX* оновлюється для двох списків одиниць (*units1, units2*).

Для врахування специфіки тексту використовується вага TF-IDF [147, 148]:

$$w_{i,j} = TF(i,j) \cdot \ln\left(\frac{N}{df_i}\right),$$

де $TF(i,j)$ – частота терміну i у документі j , N – загальна кількість документів, а df_i – кількість документів, що містять слово i .

2. Косинусна подібність (Cosine Similarity).

Для векторів високої розмірності важливо оцінювати кутову подібність – тобто на скільки схожий напрямок їх семантичного змісту. Косинусна міра подібності – це міра кута між двома ненульовими вимірними векторами, визначає ступінь подібності двох векторів $\vec{u}$ та $\vec{v}$. Класична формула косинусної подібності для двох векторів має вигляд [149].

$$sim_1(\vec{u}, \vec{v}) = \cos(\vec{u}, \vec{v}) = \frac{\sum_{k=1}^n u_k v_k}{\sqrt{\sum_{k=1}^n (u_k)^2} \sqrt{\sum_{k=1}^n (v_k)^2}}$$

де $\vec{u}$ та $\vec{v}$ – вектори ознак об'єктів. Відмітимо, що в основному в задачах використовується косинусна міра подібності в діапазоні від “0” до “1”. Вектори вважаються максимально «подібними» (косинус подібності дорівнює 1) якщо вони мають однаковий напрямок, а у випадку ортогональності (косинус подібності дорівнює 0) вектори вважаються максимально «розбіжними», а два діаметрально направлені вектори мають подібність -1, незалежно від їх довжини. Зазначимо, що крім поняття косинусної міри, в задачах класифікації часто використовується поняття косинусної відстані (щоб менше значення означало більшу схожість, як в інших метриках), яка обчислюється за такою формулою [150]:

$$dsim_1(\vec{u}, \vec{v}) = 1 - sim_1(\vec{u}, \vec{v}),$$

де $sim_1(\vec{u}, \vec{v})$ – косинусна міра подібності. Тоді якщо вектори співнаправлені, тобто тексти ідентичні за змістом, відстань дорівнює 0. Якщо вектори ортогональні, тобто немає спільних слів/тем, відстань дорівнює 1. Якщо протилежно направлені, то відстань дорівнює 2.

TF-IDF перетворює текстові рядки на розріджені вектори (Sparse Vectors), де кожному слову (або n -грамі) присвоюється вага, що відображає його важливість. Для порівняння таких векторів стандартна евклідова відстань є неефективною через "прокляття розмірності" (curse of

dimensionality) та залежність від довжини тексту. Тому для TF-IDF представлення математично коректним є використання косинусної відстані [140, 151; 152].

5. Метрики для порівняння кореляційних та коваріаційних матриць.

Для порівняння додаткових характеристик – кореляційних та коваріаційних матриць – використовуються матричні норми та спеціалізовані метрики [3, 83]. Відповідно до загального підходу порівняння різних типів даних, аналіз взаємозв'язків між ознаками потребує специфічних метрик. Для кореляційних та коваріаційних матриць, які є результатом агрегації (стиснення) даних, найефективнішим інструментом є відстань Фробеніуса [153]. Математичне обґрунтування відстані Фробеніуса $D_F(R_1 - R_2)$ між двома матрицями R_1 та R_2 (наприклад, кореляційними матрицями двох різних вибірок) визначається як норма їхньої різниці:

$$D_F(R_1 - R_2) = |R_1 - R_2|_F = \sqrt{\sum_{i=1}^p \sum_{j=1}^p (r_{1,ij} - r_{2,ij})^2}.$$

6. Уніфікована метрика DISS.

Після визначення метрик для кожного типу змінних виникає потреба в єдиній метриці, яка агрегує всі ці відмінності в одне число, що характеризує подібність між двома наборами даних на рівні корпусу [135]. Для оптимального зіставлення стовпців застосовується угорський алгоритм, що гарантує мінімізацію сумарної вартості співставлення пар [154].

Для обчислення різниці між двома наборами даних можна використовувати зважену метрику:

$$\begin{aligned} DISS(Dataset1, Dataset2) = & w_1 \cdot DIST(Factor\ variables) + \\ & + w_2 \cdot DIST(Time) + w_3 \cdot DIST(Numeric) + \\ & + w_4 \cdot DIST(String) + w_5 \cdot DIST(Other\ info = correlation, covariance \dots), \end{aligned}$$

де:

вектор параметрів $w = (w_1, w_2, w_3, w_4, w_5)$ визначається залежно від

контексту, наприклад, моделі або типу змінних, причому $\sum_{i=1}^5 w_i = 1$ та $w_i \geq 0$;

DIST(Factor variables) – агрегована відстань для факторних змінних;

DIST(Time) – агрегована відстань для часових змінних;

DIST(Numeric) – агрегована відстань для числових змінних;

DIST(String) – агрегована відстань для рядкових змінних;

DIST(Other info = correlation, covariance ...) – агрегована відстань для кореляційних матриць.

2.4.2 Виявлення дублікатів і зсувів розподілів

Одним із важливих застосувань системи (Розділ 3) є виявлення дублікатів наборів або зсувів у розподілах даних у часі (distribution shifts).

Дублікатом вважається набір B , для якого:

$$DISS(A, B) < \delta_{dup},$$

де δ_{dup} – заданий поріг.

Низьке значення DISS означає, що обидва набори мають однакову структуру і статистичні властивості. Система автоматично маркує такі набори як потенційні дублікати або резервні копії, що дозволяє оптимізувати сховище, уникаючи надлишкового зберігання даних.

Крім того, подібні значення метрики є підставою для керованого даними моніторингу якості даних – підходу, коли рішення про стан системи приймаються на основі кількісного аналізу самих даних, а не лише експертних припущень. Моніторинг на основі даних дозволяє автоматично виявляти ситуації, коли набори залишаються стабільними у часі, що підтверджує коректність процесів збору та обробки інформації. У випадках, коли DISS між послідовними версіями одного й того самого набору даних зростає, це може свідчити про зміну структури даних (дрейфу) – поступову зміну розподілів або структури даних, яка потенційно впливає на моделі машинного навчання та бізнес-процеси. Таким чином, метрика DISS може слугувати індикатором як стабільності, так і деградації даних у динамічних

інформаційних системах [155].

Для даних, що надходять у часі (лог-файли, телеметрія, транзакції), важливо відстежувати зміни у їх статистичних характеристиках. Якщо $DISS(A_t, A_{t+1})$ зростає з часом, це свідчить про зсув розподілу (дрейф), що може вказувати на зміни у джерелі даних, появу нових категорій або зникнення старих, аномалії або помилки збору. Такі зміни виявляються автоматично, і система може ініціювати повторну типізацію або оновлення моделей аналізу [156,157].

2.4.3 Алгоритмічні принципи побудови DISS-матриць

Побудова DISS-матриці починається з визначення набору змінних, які підлягають порівнянню. Це можуть бути змінні одного набору (внутрішня структурна матриця) або змінні двох різних наборів (порівняльна матриця). У першому випадку DISS-матриця дозволяє виявити внутрішню структуру набору даних: чи існують групи змінних, які поведуться подібно, які змінні є ізольованими, а які – аномальними. У іншому випадку матриця показує, як один набір даних зіставляється з іншим у термінах структурної подібності.

Алгоритм побудови DISS-матриці є багатокроковим та складається з послідовності логічно-пов'язаних етапів [70, 158]:

1. Формування компактних представлень змінних.

На першому етапі для кожної змінної формується компактне представлення (CDR), яке акумулює її структурні характеристики. Для числових змінних використовуються моментні та гістограмні дескриптори, для категоріальних – частотні профілі, для рядкових – векторні представлення на основі TF-IDF, а для часових – профілі внутрішніх залежностей або методи групування даних.

2. Обчислення попарних структурних відстаней.

На другому етапі для кожної пари компактних представлень обчислюється структурна відстань відповідно до логіки механізму DISS. Тип використовуваної метрики визначається природою порівнюваних змінних і

структурою їхніх дескрипторів.

3. Нормалізація часткових відстаней.

Оскільки різні типи змінних характеризуються різними шкалами значень, на цьому етапі застосовується нормалізація, яка приводить усі часткові відстані до сумісної числової шкали. Це забезпечує коректність подальшого агрегування та інтерпретації результатів.

4. Формування DISS-матриці.

На завершальному етапі формується матриця структурних відстаней, що містить значення відстаней між усіма парами змінних відповідно до обраного сценарію (внутрішнього або порівняння масивів даних). Отримана матриця є базовим аналітичним об'єктом для подальшого аналізу, зокрема кластеризації, виявлення структурних залежностей та оцінки дрейфу даних.

Завдяки використанню компактних представлень і незалежності обчислень для окремих пар змінних алгоритм побудови DISS-матриць характеризується високою обчислювальною ефективністю та масштабованістю. Процес обчислення може бути виконаний паралельно як на рівні змінних, так і на рівні окремих операцій обчислення відстаней, що робить можливим застосування алгоритму в розподілених середовищах, зокрема у хмарних платформах і Spark-кластерах, де паралельна обробка є критичною вимогою.

2.5 Графові представлення структурної подібності наборів даних

DISS-матриця дозволяє створити граф подібності над корпусом наборів даних, де кожна вершина відповідає окремому набору, а ваги ребер визначаються структурними відстанями між ними. Графове представлення надає інструмент для організації, навігації та аналізу великих корпусів даних, дозволяючи виявляти закономірності, дублікатні або споріднені набори, а також зміни у їх статистичних властивостях.

Нехай $D = \{D_1, D_2, \dots, D_n\}$ корпус з n наборів даних. Для кожної пари (D_i, D_j) обчислено зважену відстань DISS $d_{ij} = DISS(D_i, D_j)$, що

характеризує структурну різницю між наборами. На основі цих відстаней будується зважений неорієнтований граф подібності [159]:

$$G = (V, E, W),$$

де:

$V = \{v_1, v_2, \dots, v_n\}$ – множина вершин, кожна вершина v_i відповідає набору даних D_i ;

$E = \{V \times V\}$ – множина ребер, що з'єднують вершини;

$W = \{w_{ij}\}$ – ваги ребер, що відповідають значенням $w_{ij} = 1 - d_{ij}$, тобто мірам подібності [22].

Функція ваги w_{ij} обирається таким чином, щоб більша подібність між наборами відповідала більшій вазі ребра. Це дозволяє використовувати стандартні алгоритми теорії графів, де великі ваги асоціюються з сильними зв'язками.

У загальному випадку можна побудувати повний граф, де кожна пара вершин з'єднана ребром. Однак для великих корпусів ($n > 100$) це призводить до надмірної складності візуалізації та обчислень. Тому часто використовується порогове значення $\tau \in (0,1)$:

$$E = \{(i, j) \mid S_{ij} > \tau\}.$$

Вибір порогу визначається компромісом між повнотою інформації та інтерпретованістю графу [160].

Для спрощення структури графу та виділення найбільш значущих зв'язків застосовується побудова мінімального остовного дерева (Minimum Spanning Tree – MST). У контексті графу подібності MST визначається як підграф $T = (V, E_T)$, де $E_T \subset E$, що задовольняє наступним умовам:

- T є зв'язним деревом (ациклічним графом);
- T охоплює всі вершини;
- сума ваг ребер $\sum_{e \in E_T} w(e)$ є мінімальною серед усіх остовних дерев.

Оскільки класичні алгоритми побудови MST (алгоритм Прима, алгоритм Краскала) мінімізують суму ваг, а у нашому випадку потрібна

максимізація, еквівалентною задачею є побудова максимального остовного дерева або використання від'ємних ваг $-w(e)$ [161–163].

MST дозволяє визначити основну структуру подібності у корпусі даних, виділити таблиці, що мають найвищу ступінь схожості з іншими, відстежити еволюційні або похідні зв'язки між версіями таблиць та використовується для побудови візуальних дерев споріднених наборів у системах управління даними, виявлення “кореневих” джерел, від яких утворилися похідні версії, пошуку аномальних вузлів, що мають велику довжину ребра (високу структурну відмінність) [164].

2.6 Оцінка існуючих методів та концепція власної розробки

Створення компактних, але інформативних узагальнень наборів даних і організація корпусів за подібністю перетинається з кількома напрямками досліджень: типо-орієнтовані табличні представлення, розподільчі відстані для гетерогенних змінних, індексація та графові структури на рівні корпусів, а також конфіденційний обмін даними. Окреслимо найбільш релевантні напрями [70].

– *Компактні подання табличних даних.*

Відомим підходом є заміна необроблених даних стислими, інтерпретованими дескрипторами. Для класичних числових змінних вектори моментів (середні значення, дисперсії, моменти вищих порядків) є універсальними, оскільки вони підтримують подальше статистичне моделювання з мінімальними витратами на введення/виведення. Сучасні системи формалізують табличні структури та пропонують навчені абстракції (наприклад, DANets), які перевершують низку деревоподібних і глибоких базових методів, але зазвичай передбачають повний доступ до даних та відому специфікацію F [74]. SwitchTab розширює нейронне автокодування табличних структур, забезпечуючи конкурентну точність [76]; аналогічно, у ряді робіт досліджується збурення ваг автокодера для стиснення та класифікації [82]. Такі підходи забезпечують вбудовування ознак,

оптимізовані для прогнозування, але є менш інтерпретованими та менш пристосованими для використання різними інструментами, ніж універсальні узагальнення.

Окремі дослідження зосереджені на стисненні не табличних даних (зображень, фігур) за допомогою розподільчих або топологічних дескрипторів [3, 84, 86]. Хоча вони мають методологічну цінність, ці роботи стосуються інших типів структур даних. Наша структура залишається в межах табличних даних, але дотримується тієї ж філософії: узагальнювати розподіли з мінімальними припущеннями та зберігати характеристики, важливі для порівняння й моделювання. Конкретно, ми визначаємо компактні подання даних (CDR) для кожного типу: частотні таблиці (факторні змінні), гістограми з фіксованими інтервалами (часові), вектори моментів до четвертого порядку (числові) та TF-IDF-вектори (рядкові) [130].

– *Метрики подібності для гетерогенних узагальнень.*

Інший напрям досліджень присвячено відстаням між розподілами та узагальненнями. Для категоріальних розподілів стандартними є метрики Геллінгера та повної варіації; для гістограм і скалярних розподілів широко застосовується відстань Вассерштейна (earth mover's distance); для векторів моментів практичними залишаються норми $L1/L2$ і абсолютні чи відносні похибки (MAE/MAPE), а для TF-IDF-векторів – косинусна міра подібності [135]. Кореляція відстаней (distance correlation) розширює аналіз за межі лінійної залежності та застосовується у завданнях представлення й порівняння даних [83]. Наш внесок полягає не у створенні нової метрики, а в уніфікованій агрегації: оцінка структурної подібності інформації даних (DISS) комбінує типо-специфічні відстані через ваговий вектор (встановлений користувачем або визначений через навчання системи), отримуючи єдине число для кожної пари наборів даних, яке враховує гетерогенність та дає змогу будувати графи подібності.

– *Навчання представлень таблиць проти інтерпретованих узагальнень.*

Нейронні підходи для розуміння таблиць [74, 76] і перетворення «таблиць у зображення» для CNN [75] створюють латентні вкладення, які є потужними для прогнозування. Однак вони:

- вимагають повного доступу до даних,
- можуть змішувати семантику різних типів змінних;
- є складними для проведення аудиту.

Наш дизайн орієнтований на інтерпретовані, типо-орієнтовані CDR, які можна обмінювати між командами чи обліковими записами, безпосередньо перевіряти та швидко відтворювати в конвеєрах без повторного навчання моделей.

- *Організація корпусів і графові структури.*

Графові представлення є природними для кодування зв'язків між наборами даних. Попередні роботи з графових баз даних пропонують схеми та моделі зберігання для складних графів із атрибутами [89]. Ми будуємо граф суміжності між наборами даних на основі парних оцінок DISS, обчислених із CDR, і виводимо мінімальне остовне дерево, щоб отримати дерево подібності для навігації, пошуку найближчих сусідів і ієрархічного перегляду. Це відрізняється від графових баз тим, що наші ребра породжуються статистичною подібністю, а не попередньо визначеними семантичними зв'язками.

- *Порівняння з урахуванням конфіденційності та синтетичні табличні дані.*

Коли дані не можуть вільно поширюватися, узагальнення та синтез є альтернативами. STAB-GAN+ зосереджується на генерації реалістичних табличних даних для використання із збереженням конфіденційності [92]; наша мета є додатковою – порівнювати набори даних за узагальненнями без розкриття необроблених записів. Обмінюючись CDR (розподілами, моментами, словниками TF-IDF) замість рядків, організації можуть виконувати оцінку подібності між доменами зі зменшеним ризиком розкриття конфіденційних атрибутів, водночас зберігаючи практичну

користь для виявлення та моніторингу зміни структури даних.

Підсумовуючи, наявна література пропонує:

- потужні навчені вкладення для прогнозування за таблицями [74, 76, 82];
- усталені відстані для порівняння розподілів і векторів [83, 135];
- графові моделі зберігання [89];
- методи синтезу для забезпечення конфіденційності [92].

Недостатньо визначеним залишається підхід, який:

- будує інтерпретовані, типо-орієнтовані компактні подання з початкових таблиць;
- агрегує гетерогенні відстані в єдиний, налаштований показник подібності на рівні набору даних;
- організовує цілі корпуси в граф або дерево подібності, та є масштабованим і хмарно-орієнтованим.

Наша структура CDR+DISS заповнює цю прогалину та безпосередньо інтегрується зі стандартними інструментами AWS для масштабного операційного використання. З цією метою створено інформаційну систему, детальний опис якої наведено у Розділі 3.

2.7 Конвеєрна система аналізу даних

Узагальнюючи результати попереднього аналізу, запропоновано конструкцію конвеєрної системи, робота якої реалізується чотирма функціональними блоками [70]:

1. Типізація змінних. Стовпці класифікуються як факторні, часові, числові (неперервні/дискретні) або рядкові за допомогою евристик, що базуються на співвідношенні унікальності, можливості парсингу та структурних ознаках (наприклад, фіксованих форматах ідентифікаторів чи дат).

2. Побудова CDR. Для кожної типізованої змінної формується компактне, інтерпретоване узагальнення:

- факторні $\rightarrow$ частоти значень;
- часові $\rightarrow$ гістограми з фіксованими інтервалами та спільними межами для забезпечення порівнюваності;
- числові $\rightarrow$ вектори моментів $(N, \sum x, \sum x^2, \dots, \sum x^4)$;
- рядкові $\rightarrow$ TF-IDF-представлення з обмеженим словником n -грам.

3. *Попарна схожість (DISS)*. Обчислюються специфічні для типу відстані: відстань Геллінгера / відстань повної варіації для категорійних розподілів, відстань Вассерштейна-1 для гістограм, MAE/MAPE або евклідова відстань для векторів моментів, для TF-IDF – косинусна подібність. Далі агрегуємо їх за допомогою вектора ваг, щоб отримати DISS. Ваги можуть бути встановлені користувачем або навчені на розмічених парах, залежно від сфери застосування.

4. *Побудова графа та дерева*. На основі значень DISS формується граф суміжності, з якого за допомогою мінімального остовного дерева утворюється дерево подібності. Це дерево підтримує навігацію по корпусу, пошук найближчих сусідів та виявлення кластерів.

Запропонований підхід розроблено для виробничих середовищ, де мають значення обсяг даних, гетерогенність і вимоги до управління:

- Виявлення та каталогізація даних. CDR (компактні представлення даних) слугують «легковаговими», дескрипторами, що піддаються запитам, для швидкої первинної оцінки наборів даних і семантичного пошуку у сховищах даних.
- Моніторинг якості та виявлення зсуву (дрейфу). Періодичний перерахунок CDR виявляє зсуви розподілів у числових, категорійних, часових або рядкових полях перед перенавчанням моделі.
- Виявлення дублікатів і майже дублікатів. DISS ідентифікує найбільш схожі набори даних, допомагаючи у видаленні дублікатів і перевірці походження даних у багатоджерельних середовищах.
- Порівняння з урахуванням конфіденційності. Оскільки CDR

узагальнюють розподіли, а не необроблені записи, можна виконувати порівняння між командами чи обліковими записами з меншим ризиком розкриття чутливих характеристик даних.

- MLOps і вибір наборів даних. Пошук найближчих сусідів на графі подібності прискорює відбір навчальних (тренувальних) даних, які найкраще відповідають цільовим доменам.

Висновки до розділу 2

У розділі було розроблено комплексний математичний, алгоритмічний та інженерний апарат, який забезпечує побудову компактних представлень табличних даних (CDR) та формування інтегральної метрики DISS для кількісного вимірювання подібності між наборами даних.

Запропонований підхід є узагальненням і розвитком сучасних концепцій аналітики великих даних, інформаційної геометрії та теорії подібності, поєднуючи статистичні, топологічні й кореляційні принципи у єдиній системі.

Основні результати розділу:

1. Сформульовано математичну модель проблеми компактного подання даних, яка полягає у заміні повних наборів спостережень їх стислими дескрипторами, що зберігають інформаційно-значущі характеристики розподілу. Це дозволяє мінімізувати обсяг даних без втрати аналітичної змістовності.

2. Обґрунтовано інформаційно-геометричний, топологічний і моментно-кореляційний підходи до побудови компактних представлень. Показано, що для різних типів змінних (числових, факторних, часових, рядкових) можна сформулювати узагальнені статистичні описи – від функцій маси ймовірності до векторів моментів і TF-IDF-подань.

3. Розроблено алгоритм типізації змінних, який автоматично класифікує атрибути на основі структурних і статистичних критеріїв

(унікальність, формат, дисперсія довжин, парсинг дат). Такий підхід забезпечує універсальність системи, незалежно від походження або якості даних.

4. Запропоновано формальні схеми побудови компактних представлень (CDR) для кожного типу змінних:

- факторні – через функції маси ймовірності (PMF);
- часові – через гістограми або квантильні ескізи;
- числові – через набори моментів до четвертого порядку;
- рядкові – через векторні подання (TF-IDF, n -грамні моделі).

5. Визначено метрики подібності для кожного типу компактного подання:

- для PMF – відстані Геллінгера та повної варіації;
- для гістограм – відстань Вассерштейна;
- для числових моментів – евклідова або MAPE;
- для рядкових представлень – косинусна подібність.

6. Сформовано інтегральну метрику DISS, яка об'єднує часткові відстані в єдину зважену оцінку подібності між наборами даних. Розроблено алгоритм побудови DISS-матриці.

7. Побудовано та обґрунтовано конвеєрну систему аналізу даних.

Основні результати другого розділу опубліковані в роботах [70, 158].

РОЗДІЛ 3

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ПРОВЕДЕННЯ ОПТИМІЗАЦІЇ СТРУКТУРИ ГЕТЕРОГЕННИХ ДАНИХ

У розділі розглядається інформаційна система для проведення оптимізації структури гетерогенних даних. Розроблене програмне забезпечення є практичною реалізацією теоретичних принципів компактного представлення табличних даних (CDR), структурної подібності (DISS) та порівняльного аналізу статистичних наборів, сформульованих у попередніх розділах дисертаційного дослідження. Вона поєднує математичні моделі, описані в Розділах 1–2, із сучасними інженерними підходами до обробки великих даних, забезпечуючи автоматизовану та масштабовану платформу для аналізу гетерогенних інформаційних джерел. Архітектура системи орієнтована як на локальне застосування, так і на розподілені обчислення у хмарному середовищі, що дозволяє працювати з великими потоками даних.

3.1 Функціональне призначення системи

Основним призначенням системи є забезпечення універсального механізму інтелектуального порівняння структур даних без необхідності повної обробки або перегляду всіх елементів набору [70, 165]. На відміну від класичних підходів [166–169], що оперують необробленими даними та значною мірою залежать від їх розміру, запропонована система працює з компактними дескрипторами, які стисло, але інформативно характеризують внутрішню структуру змінних. Це дозволяє проводити порівняння великих наборів даних, їх фрагментів або поточкових зрізів у режимі підвищеної ефективності.

Концептуальна ідея системи ґрунтується на переході від класичного уявлення “дані як множина рядків” до уявлення “дані як компактна структура” [166, 168]. Математично це відображено через використання статистичних

моментів, частотних розподілів, інформаційно-геометричних дескрипторів і топологічних характеристик, які формують узагальнення у вигляді CDR [170, 171]. Аналіз подібності між такими представленнями, у свою чергу, реалізується за допомогою системи метрик DISS, що оцінюють як локальні (між змінними), так і глобальні (між наборами даних) відхилення [172, 173].

Оскільки матриця DISS є повною симетричною матрицею відстаней, вона може бути інтерпретована як зважений граф, для якого можуть застосовуватися підходи спектрального аналізу та кластеризації [174–177], що дозволяє засобами технології:

- швидко визначати ступінь подібності між різними наборами даних;
- знаходити зміни у внутрішній структурі або закономірностях даних;
- порівнювати якість та цілісність даних з різних джерел;
- здійснювати кластеризацію наборів даних на основі їх компактних дескрипторів;
- підвищувати ефективність ETL/ELT-процесів у середовищах великих даних [178–180].

Таким чином, система вирішує проблему порівняння великих табличних наборів даних та формалізації подібності у гетерогенних даних.

3.2 Вимоги до інформаційної системи

З метою забезпечення коректної реалізації запропонованої інформаційної системи для структурного аналізу даних було визначено сукупність функціональних і нефункціональних вимог. Зокрема, ці вимоги стосуються модулів формування компактних представлень змінних, побудови матриць структурних відстаней (DISS), аналізу відповідностей між наборами даних та візуалізації результатів обробки.

Функціональні вимоги визначають безпосередні можливості системи та перелік задач, які вона повинна виконувати. До функціональних вимог

належать:

- забезпечення автоматизованої побудови матриць структурних відстаней між змінними різних наборів даних на основі їх компактних представлень, де кожен елемент матриці інтерпретується як числова міра структурної подібності відповідної пари змінних;
- урахування типу змінних та специфіки їх статистичних, частотних, векторних або часових характеристик під час обчислення структурних відстаней, що забезпечує коректність порівняння гетерогенних даних;
- коректна обробка випадків несумісності типів змінних або відсутності компактного представлення шляхом використання штрафних значень відстані без порушення цілісності обчислювального процесу;
- визначення найбільш подібних відповідностей між змінними двох наборів даних із можливістю формування впорядкованих множин кандидатів для кожної змінної вхідного масиву даних;
- представлення результатів зіставлення у структурованому вигляді, придатному як для подальшої автоматизованої обробки, так і для експертного аналізу;
- обчислення міри структурної подібності або дрейфу між наборами даних, що дозволяє узагальнювати локальні відмінності у вигляді єдиної числової характеристики;
- візуалізація результатів обчислень у вигляді теплових карт і аналітичних графіків, що спрощує інтерпретацію структурних взаємозв'язків між змінними.

Нефункціональні вимоги визначають якісні характеристики роботи інформаційної системи. До нефункціональних вимог відносяться:

- відтворюваність результатів обчислень за однакових вхідних даних і параметрів, що є необхідною умовою коректного

експериментального аналізу;

- масштабованість системи щодо кількості змінних і обсягу даних без суттєвого зниження продуктивності;
- обчислювальна ефективність алгоритмів, що забезпечує прийнятний час виконання у пакетних і ітеративних сценаріях;
- стійкість до неповних, шумових або частково некоректних даних без аварійного завершення роботи;
- модульність і розширюваність архітектури системи, що забезпечує можливість додавання нових типів змінних або метрик подібності;
- можливість інтеграції в автоматизовані аналітичні конвеєри.

Визначений набір функціональних і нефункціональних вимог забезпечує цілісність, надійність та експлуатаційну ефективність розробленої інформаційної системи.

3.3 Архітектура та функціональні компоненти системи

3.3.1 Логічна архітектура

Архітектура системи має багаторівневий модульний характер, що забезпечує її гнучкість, повторне використання компонент та адаптивність до різних джерел і обсягів даних. Вона складається з трьох основних логічних рівнів (рис. 3.1): рівня попередньої обробки, рівня побудови компактних представлень та метрик подібності, а також рівня аналітики та масштабованого виконання.

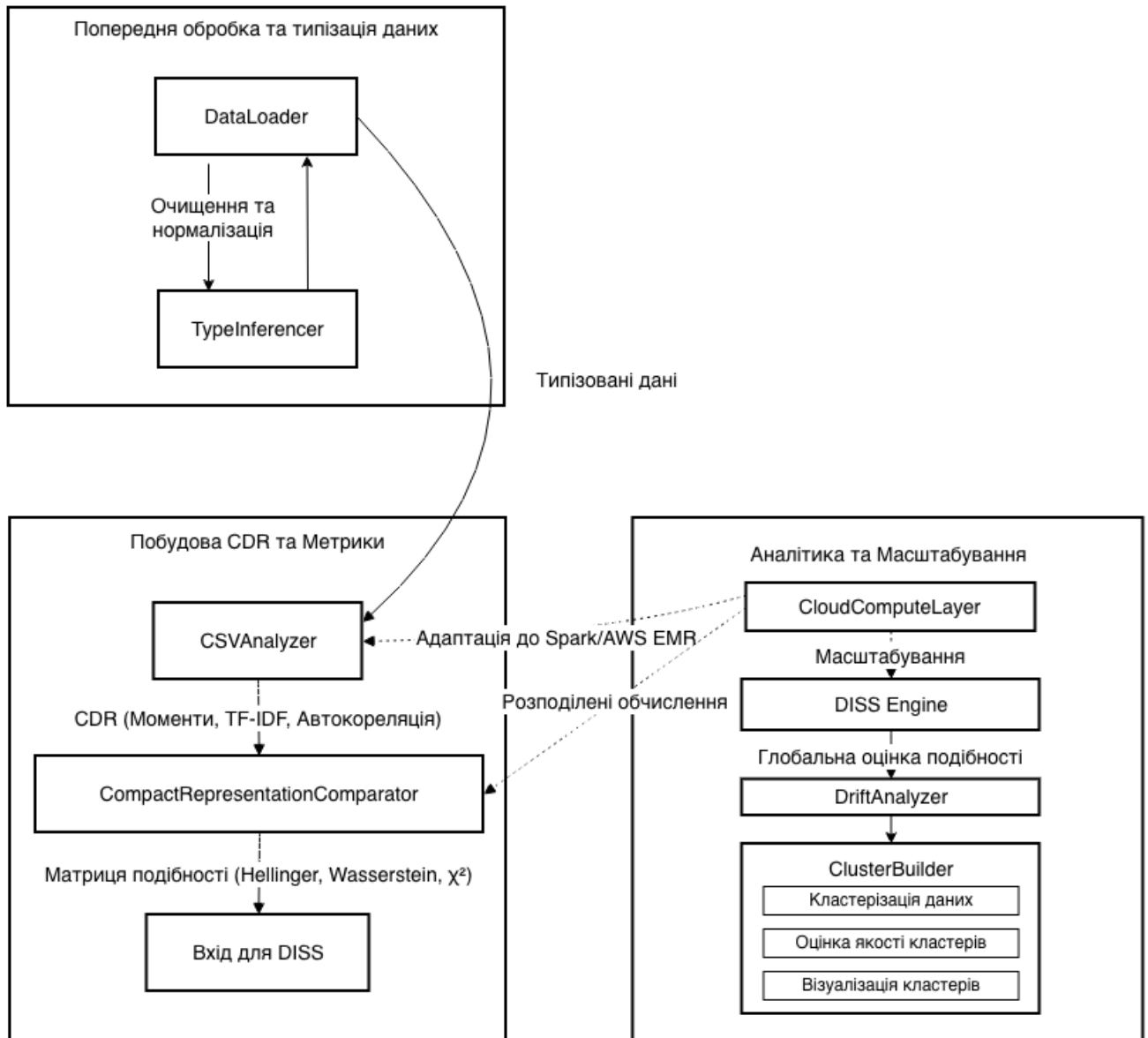


Рис. 3.1 – Діаграма функціональних компонент системи

Перший логічний рівень включає модулі `DataLoader` і `TypeInferencer`, відповідальні за завантаження, очищення та нормалізацію даних. На цьому етапі виконується автоматична типізація змінних за допомогою правил, які враховують статистичні, текстові та часові особливості стовпців. У контексті Розділу 2 саме тут реалізовано основу класифікації змінних, що є необхідною передумовою для побудови інформативних компактних представлень.

Другий рівень становить ядро системи – модулі CSVAnalyzer та CompactRepresentationComparator, які реалізують побудову CDR і обчислення структурних метрик. Модуль CSVAnalyzer відповідає за формування дескрипторів, що узагальнюють дані кожної змінної. У ньому відтворені математичні моделі з Розділу 2: моментно-кореляційне представлення, частотні розподіли, TF-IDF векторизація. Модуль CompactRepresentationComparator виконує обчислення індивідуальних відстаней між дескрипторами, після чого формує матрицю подібності, яка відображає структурну організацію набору даних на рівні змінних.

Третій логічний рівень включає модулі DISS Engine, DriftAnalyzer, ClusterBuilder та CloudComputeLayer, які забезпечують інтеграцію результатів, їх узагальнення, аналіз зміни структури даних та розподілене виконання. DISS Engine агрегує метрики між парами змінними в глобальні оцінки подібності між наборами даних. DriftAnalyzer дозволяє визначати статистичні відхилення у часі між двома будь-якими вибірками, що є важливою практичною функцією в поточних системах моніторингу даних. CloudComputeLayer абстрагує інфраструктурну складову та адаптує алгоритми для роботи в Spark-оточенні, що робить систему придатною для обробки мільйонів записів. Реалізація компонент цього рівня ґрунтується на принципах, описаних у статті [181] ввикористання AWS EMR у задачах машинного навчання і стохастичного моделювання.

Модуль ClusterBuilder реалізує функціонал побудови кластерів наборів даних на основі обчислених структурних метрик подібності. Він використовує матрицю відстаней, сформовану на попередніх етапах, для групування наборів даних із подібною внутрішньою структурою. Кластеризація може виконуватися із застосуванням ієрархічних, спектральних або щільнісних методів, що дозволяє адаптувати підхід до різних типів даних та масштабів задачі.

Для оцінки результатів кластеризації використовується окремий

компонент аналізу якості кластерів, який обчислює внутрішню компактність, між кластерну віддаленість та показники стабільності групування.

На завершальному етапі формуються графічні представлення взаємного розташування наборів даних у просторі подібності, що полегшує інтерпретацію результатів і використання системи для аналітичних та дослідницьких задач [31, 32].

Усі модулі працюють як незалежні, але взаємопов'язані компоненти, що можуть змінюватися, розширюватися або оптимізуватися без впливу на загальну структуру системи. Це особливо важливо для практичного застосування у випадках, де набори даних і вимоги до аналізу мають тенденцію до швидких змін.

3.3.2 Архітектура хмарної реалізації

Хмарні обчислення представляють собою підхід до побудови інформаційних систем, що ґрунтується на наданні обчислювальних ресурсів, сховищ даних, програмного забезпечення та сервісів через мережу Інтернет за моделлю "як послуга" (as-a-Service). Особливістю хмарних технологій є відхід від жорсткої залежності від фізичної інфраструктури на користь динамічного масштабування ресурсів, здатних адаптуватися до змінного навантаження та специфічних вимог користувачів.

Залежно від рівня абстракції виділяють три основні моделі надання хмарних послуг. Infrastructure as a Service (IaaS) [182] забезпечує доступ до віртуалізованих обчислювальних ресурсів (процесори, пам'ять, мережа, сховища). Platform as a Service (PaaS) [183] надає повноцінне середовище для розробки, тестування та виконання додатків без необхідності керування базовою інфраструктурою. Software as a Service (SaaS) [184] пропонує готові програмні рішення кінцевим користувачам через вебінтерфейс.

За способом розгортання розрізняють публічні, приватні та гібридні

хмари, що характеризуються різними рівнями контролю, забезпечення безпеки та організаційної гнучкості [185, 186]. Публічні хмари є загальнодоступними та керуються третьою стороною, приватні хмари призначені для використання однією організацією, тоді як гібридні хмари поєднують обидва підходи, забезпечуючи балансування між контролем та масштабованістю.

У рамках даного дослідження обрано публічну хмарну інфраструктуру з орієнтацією на PaaS/IaaS-модель, оскільки саме такий підхід забезпечує необхідну масштабованість обчислювальних ресурсів, підтримку розподілених обчислень та інтеграцію з сучасними інструментами обробки великих обсягів даних [181, 187].

Архітектура системи побудована на принципах cloud-native підходу, що передбачає проектування додатків для роботи в хмарному середовищі з акцентом на масштабованість, відмовостійкість та автоматизацію операційних процесів [180]. Основною метою є забезпечення автоматизованого та масштабованого конвеєра трансформації неструктурованих текстових даних у колонково-орієнтовані формати зберігання, придатні для аналітичної обробки [188].

Система реалізована як концептуально цілісний конвеєр обробки даних, а не як набір окремих утиліт. Кожен етап конвеєра виконує чітко визначену функцію [165]: розподілена обробка вхідних даних, зберігання вихідних об'єктів у стійкому об'єктному сховищі, виявлення подій та автоматичний запуск робочих процесів, компактне збереження результатів у колонкових форматах (Parquet [189] або ORC [190]), централізована каталогізація метаданих та забезпечення прямого SQL-доступу до результуючих наборів даних [191].

Запропонований дизайн архітектури забезпечує чіткий поділ відповідальності між компонентами, що полегшує моніторинг роботи системи, дозволяє незалежно масштабувати окремі підсистеми залежно від навантаження та мінімізує зв'язність (coupling) між компонентами [192], підвищуючи загальну

надійність та підтримуваність рішення [165, 193].

Для реалізації запропонованої інформаційної системи обрано хмарну інфраструктуру Amazon Web Services (AWS), що забезпечує необхідні характеристики масштабованості, надійності та інтеграції з сучасними інструментами обробки великих обсягів даних.

При розробці інформаційної системи використовувався набір сервісів AWS, до якого входять Apache Airflow [133] для координації аналітичних конвеєрів, Amazon S3 [194] як об'єктне сховище даних, а також AWS Glue [134] для управління метаданими та логічною структурою даних.

Окремо варто розглянути Amazon EMR (рис. 3.2), який призначений для зниження витрат при обробці великих обсягів даних та пропонує численні переваги, включаючи масштабованість, високу продуктивність, економічність з оплатою лише за використані ресурси, інтеграцію з іншими сервісами AWS для легкого обміну даними, простоту налаштування та управління кластером, високу безпеку з можливістю шифрування даних, автоматизацію та управління завданнями через конвеєри даних, а також підтримку широкого спектру аналітичних інструментів, таких як Apache Hive [195], Pig [196], HBase [197] та Presto [198]. Ще однією опцією є можливість використання Amazon EMR Serverless, що дозволяє запускати додатки для обробки великих даних без необхідності керування кластерами [181, 199].

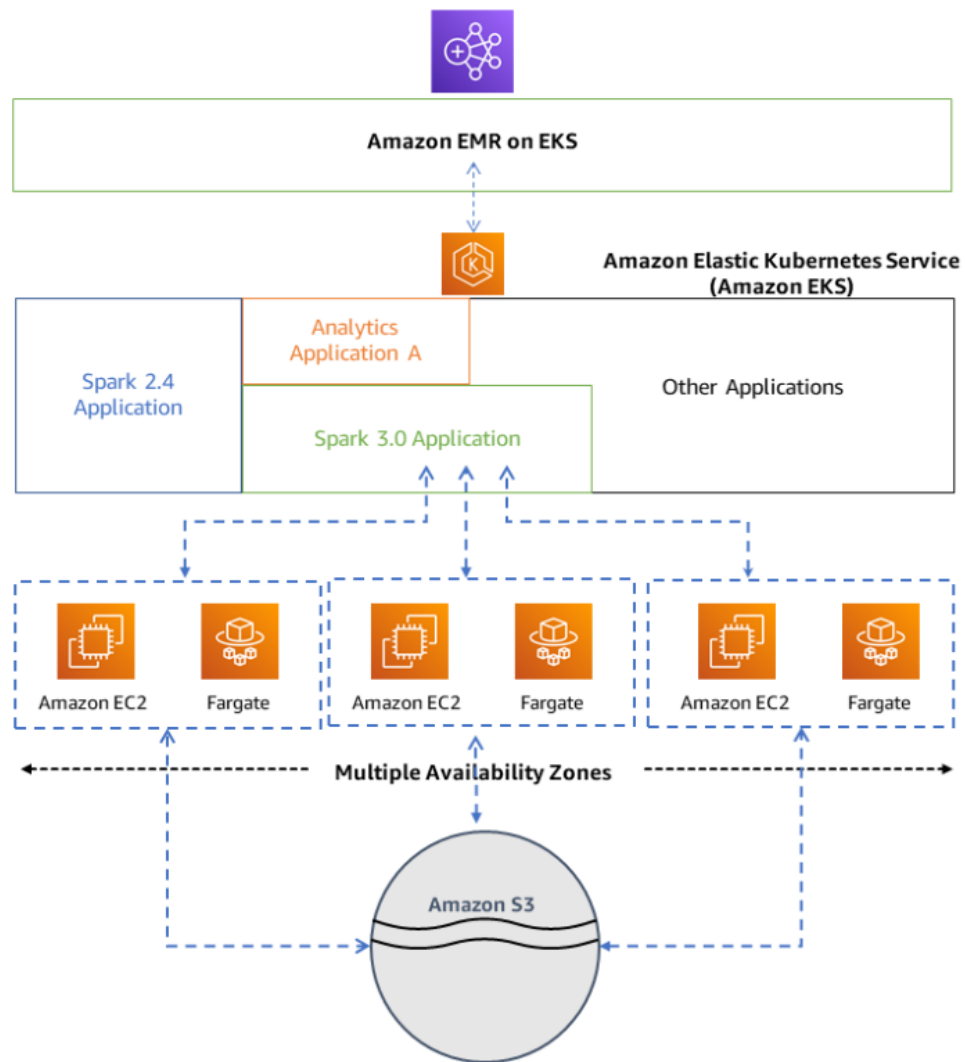


Рис. 3.2 – Архітектурна діаграма Amazon Elastic MapReduce [199]

Запропонована комбінація сервісів дозволила побудувати модульну та слабо зв'язану архітектуру, що підтримує як пакетну (batch), так і ітеративну обробку даних. Архітектурне рішення забезпечує відтворюваність обчислювальних експериментів, можливість динамічного масштабування обчислювальних ресурсів відповідно до поточного навантаження та вимог конкретної задачі, а також дозволяє незалежно модифікувати окремі компоненти без впливу на функціонування системи в цілому.

Робота з системою починається з об'єктного сховища Amazon S3, яке використовується не лише як пасивний «жорсткий диск у хмарі», але й як

логічний інтерфейс даних, що контролює політики доступу, структуру каталогів і правила іменування (рис. 3.3). Кожен об'єкт містить семантичний зміст, який включає джерело, домен, дату надходження та тип сировинного файлу, наприклад “s3://data-ingestion/finance/2025-05-01/transactions.csv”. Такий метод прив'язує контекст до місця, що дозволяє оркестратору самостійно керувати конкретним об'єктом без додаткових довідників [200]. S3 був обраний у зв'язку з його значними перевагами. По-перше, він пропонує високу надійність зберігання завдяки внутрішній реплікації даних. По-друге, він пропонує майже необмежену горизонтальну масштабованість. По-третє, він добре інтегрується з іншими сервісами хмарної платформи, такими як логування операцій, верифікація об'єктів і механізми керування доступом [202].

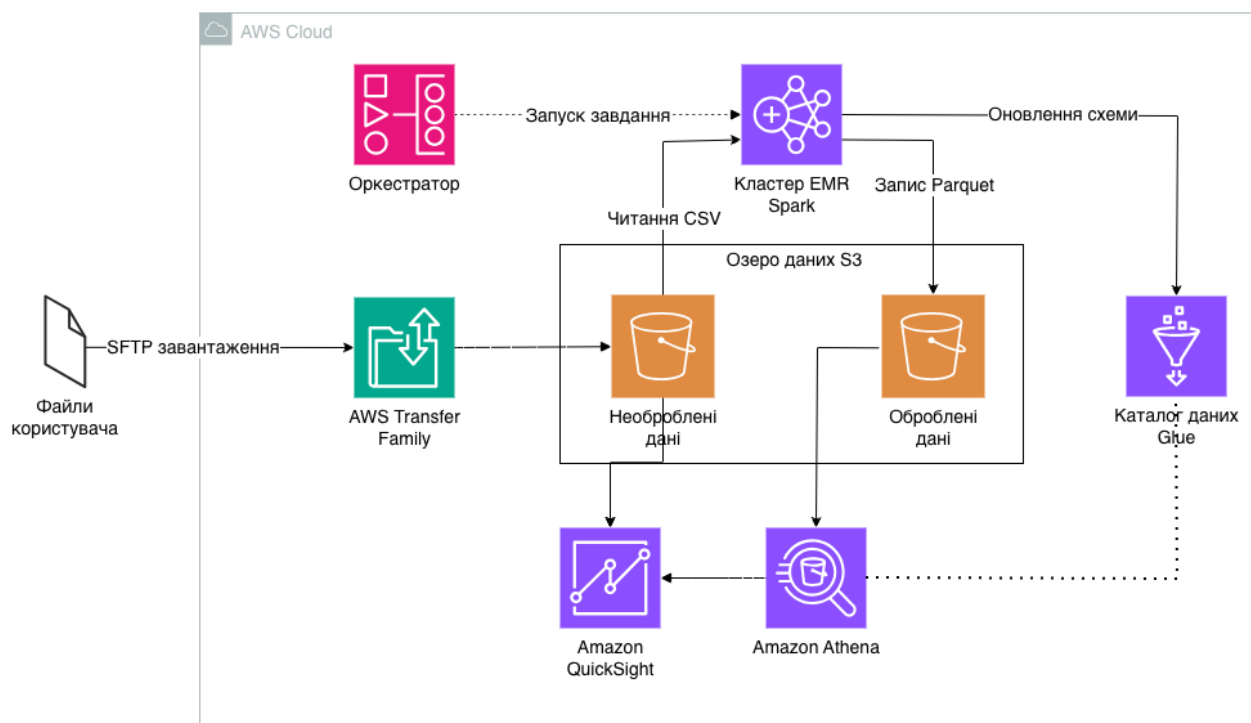


Рис. 3.3 – Архітектура інформаційної системи
з використанням хмарних сервісів

Рівень координації з використанням Apache Airflow, починає працювати від моменту появи нового файлу у сховищі. Координатор сприймає дані як подію, яку потрібно перетворити на послідовність завдань. [133, 203–205].

Кожне виконання конвеєра параметризується на основі допоміжних конфігураційних файлів (маніфестів) або контекстної інформації шляху, тому одна і та сама логіка працює для багатьох джерел даних без дублювання коду. Координатор може впевнено керувати як короткими мікропроцесами, так і складними, сценаріями, що включають багато етапів завдяки декларативності та можливості використання шаблонів [202]. Усі дані про виконання, включаючи журнали подій, тривалість виконання окремих етапів, причини відмов та інформацію про повторні спроби, зберігаються в централізованій метабазі даних. Така архітектура забезпечує повний аудит операцій обробки даних, дозволяє проводити ретроспективний аналіз інцидентів та підтримує вимоги до прозорості та відтворюваності обчислювальних процесів.

Обчислювальний рівень системи реалізовано на базі кластера Amazon EMR зі стеком технологій Hadoop/Spark. Обраний підхід дозволяє організувати розподілену обробку даних, при цьому масштабування обчислювальної потужності досягається не шляхом нарощування ресурсів окремого вузла (вертикальне масштабування), а за рахунок збільшення кількості паралельних виконавців (executors), що незалежно обробляють окремі частини набору даних.

Для оптимізації операційних витрат система використовує модель транзитних (ephemeral) кластерів, які динамічно створюються на час виконання конкретного завдання та автоматично завершують роботу після успішного завершення або виникнення критичної помилки. Такий підхід запобігає неконтрольованому споживанню обчислювальних ресурсів та пов'язаним з цим фінансовим витратам.

Додатково, використання проміжних екземплярів EC2 (spot instances) у поєднанні з механізмами автоматичного масштабування (auto-scaling) дозволяє суттєво знизити вартість обчислень без деградації продуктивності системи. Незважаючи на те, що доступність spot-екземплярів обмежена їх ринковою природою, їх необхідна кількість може бути достатньо точно передбачена на

рівні координатора робочих процесів. Слід зазначити, що ініціалізація кластера та активація середовища виконання вимагають додаткового часу (порядку кількох хвилин), що необхідно враховувати при плануванні обробки даних [205].

3.3.3 Функціонування створеної інформаційної системи

Розроблений інформаційний продукт функціонує як багатоетапний потік, у якому кожен етап формує підґрунтя для наступного, забезпечуючи неперервність обробки та відтворюваність результатів. Його життєвий цикл демонструє не лише технічну послідовність дій, але й концептуальну єдність системи, у якій аналітика, обчислення, метадані, контроль якості та аудит злиті в єдину динамічну структуру. Такий підхід відображає сучасну парадигму data-centric orchestration, де джерело даних є визначальним елементом, а не пасивним об'єктом обробки [165].

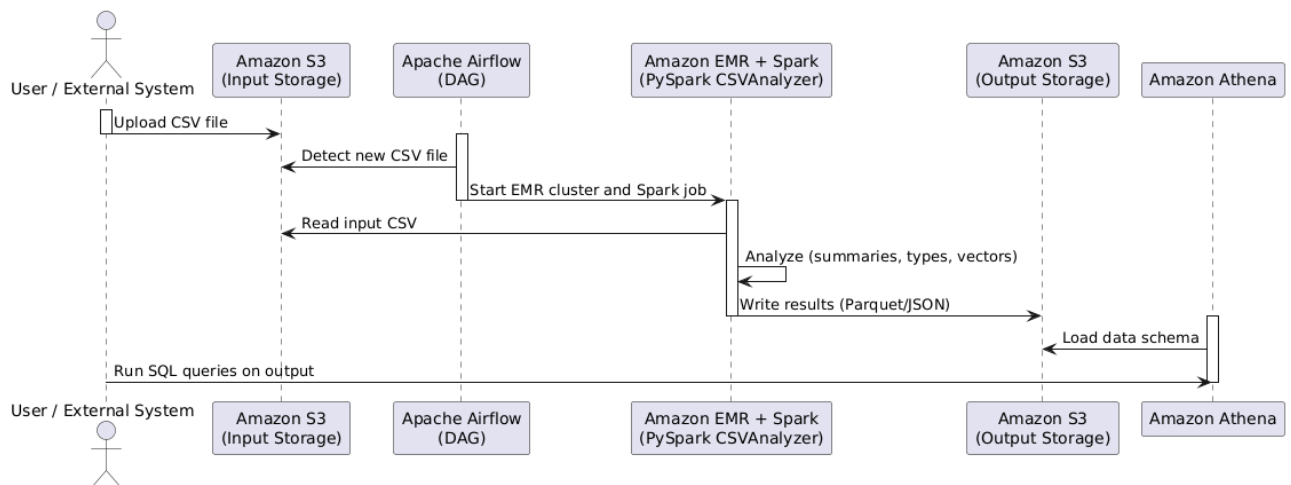


Рис. 3.4 – Діаграма послідовності процесів у системі

На початковому етапі життєвого циклу система переходить у стан очікування події (рис. 3.4). Завантаження нового об'єкта в об'єктне сховище сприймається не як просте розміщення файлу, а як сигнал до запуску нового процесу. Кожен файл, що надходить, має власний контекст – доменну належність, дату, тип, формат, і саме цей контекст визначає траєкторію його

подальшого обробка. Потік не передбачає попередньо заданого графіку виконання: він живе подіями, які виникають природно в результаті появи нових даних. Це дозволяє системі реагувати асинхронно, масштабуючи навантаження не за часом, а за реальним попитом.

Після виявлення події формується віртуальний контекст виконання, у межах якого система відновлює параметри середовища, створює тимчасові ідентифікатори об'єктів і визначає політики доступу. Всі ці дії відбуваються автоматично, без участі користувача, але з повним аудитом у журналах подій. Таким чином, навіть етап підготовки не є статичним – він є частиною динамічного процесу керування життєвим циклом даних. Для кожного завдання створюється ізольований обчислювальний простір, що гарантує відсутність конфліктів і дозволяє повторно запускати процеси з будь-якого моменту без ризику дублювання результатів.

На етапі аналітичного виконання відбувається основна трансформація даних. Тут продукт демонструє свій хмарний характер: усі обчислення виконуються не послідовно, а паралельно, розподіляючись між багатьма вузлами. Потік завдань описується декларативно – система не вказує, як виконувати обчислення, а лише що потрібно отримати. Планувальник самостійно визначає оптимальну кількість виконавців, способи розподілу даних і методи збереження проміжних результатів. Така модель дозволяє системі залишатися стійкою навіть у випадках перевантаження або нестачі ресурсів: процеси можуть відновлюватися частково, а невдалі вузли автоматично замінюються.

Процес трансформації вхідних файлів виходить за межі простої конвертації формату контейнера з CSV у Parquet. Алгоритм обробки реалізує комплекс операцій, що включає семантичну класифікацію стовпців, автоматичне виведення та приведення типів даних, нормалізацію роздільників та кодувань символів, очищення від порожніх рядків, а також обробку

пропущених значень відповідно до налаштованих стратегій заповнення або видалення.

Метод автоматичної ідентифікації типів даних базується на наборі статистичних показників, до яких належать: частка успішних приведень до певного типу, кількість унікальних значень у стовпці, ентропія розподілу значень, регулярність форматних шаблонів, середня довжина рядкових значень та наявність стабільних шаблонів дат і часових міток [135].

Навіть у джерелах, які не проходили попередню обробку, ці індикатори з високою вірогідністю можуть розрізняти числові, булеві, категоріальні, часові та текстові поля. Результати семантичної класифікації визначають стратегію подальшої обробки даних: вибір агрегатних функцій для обчислення описових статистик, рішення щодо зберігання проміжних результатів у оперативній пам'яті, а також вибір оптимальних схем фізичного зберігання у колонковому форматі. Такий підхід забезпечує збереження інформаційної сутності вихідних даних при одночасному досягненні високої швидкодії запитів та компактності зберігання результируючих наборів даних.

Компактне подання даних створюється після нормалізації. Для числових змінних воно охоплює основні статистики та розподіли, які корисні для подальшої аналітики та контролю якості; для категоріальних – важливі пропорції класів і діагностика перекосів; для часових змінних – стандартизація інтервалів і визначення сезонних структур; і для неструктурованого тексту – створення зважених векторів, щоб виокремити семантично значущі терміни в тексті. Цей етап має на меті не лише підготовку запису у колонковий формат, але й створення інформаційного представлення даних джерела. Це представлення можна використовувати для відстеження відхилень, порівняння між днями, верифікації схем і навіть початкового відбору ознак для моделей.

Вибір параметрів серіалізації даних у форматі Parquet або ORC значно впливає на продуктивність подальших аналітичних запитів. До критичних

параметрів належать: алгоритм стиснення даних (наприклад, Snappy, Gzip, Zstandard), цільовий розмір окремого файлу, кількість файлів у межах логічного розділу (partition), метод розбиття набору даних, політика повторного розподілу (repartitioning) даних за ключовими атрибутами та стратегія контролю за утворенням надмірно дрібних файлів. Оптимально налаштований конвеєр прагне створювати збалансовані об'єкти зберігання та запобігає виникненню проблеми "small files problem" – ситуації, коли набір даних фрагментується на мільйони дрібних файлів, що критично погіршує характеристики читання через надмірні накладні витрати на операції доступу до об'єктного сховища [206].

Стратегія розбиття даних повинна корелювати з найбільш поширеними предикатами фільтрації в аналітичних запитах, такими як домен даних, дата події або географічний регіон. Оптимально розбиті дані дозволяють виключати нерелевантні розділи зі сканованого обсягу на етапі планування запиту, що суттєво скорочує час виконання.

Для аналітичних завдань колонково-орієнтовані формати зберігання є найбільш придатними з низки причин [207]. По-перше, вони дозволяють застосовувати ефективні алгоритми кодування на рівні блоків та сторінок даних (block-level та page-level encoding). По-друге, реалізують можливість проєкції стовпців (column projection) для читання лише необхідних для запиту атрибутів без завантаження повних записів. По-третє, підтримують еволюцію схеми даних з можливістю додавання нових полів без зміни існуючих файлів. По-четверте, включають вбудовані метадані та дескриптори, що робить набір даних само документованим та спрощує його каталогізацію й використання різними споживачами.

AWS Glue Data Catalog функціонує як централізоване сховище метаданих, що описує структуру отриманих таблиць даних. Каталог акумулює інформацію про схеми даних, визначення логічних розділів та фізичних шляхів до файлів в об'єктному сховищі, виступаючи єдиною точкою істини (single source of truth)

для різноманітних аналітичних рушіїв, зокрема Amazon Athena [208, 209].

Завдяки автоматичній реєстрації метаданих у каталозі одразу після завершення трансформацій, підготовлені набори даних стають доступними для SQL-запитів практично миттєво. Це демонструє суттєву перевагу підходу до аналітики великих даних, оскільки аналітичні запити можуть виконуватися безпосередньо на об'єктах у сховищі S3, без необхідності завантаження результатів у спеціалізовані системи управління базами даних або побудови складних ETL-конвеєрів [210].

Важливою перевагою такого підходу є економічна ефективність [205]. Вартість виконання запитів у системі Amazon Athena прямо пропорційна обсягу сканованих даних у байтах. Використання колонково-орієнтованих форматів зберігання з компресією дозволяє суттєво зменшити цей обсяг за рахунок проєкції стовпців та ефективного стиснення. Таким чином, перевага у вартості виявляється не менш значущою, ніж підвищення швидкості виконання запитів.

Ефективність запропонованого підходу була підтверджена в практичних експериментах, проведених на даних взятих з відкритих джерел інтернету та описаних у Розділі 4. Використання таблиць у форматах Parquet та ORC призвело до скорочення тривалості виконання типових аналітичних запитів у кілька разів порівняно з обробкою вихідних CSV-файлів.

Незважаючи на те, що архітектура орієнтована на пакетну обробку, вона відкрито підтримує еволюцію у бік потоків. Поява майже реального часу обробки досягається за допомогою інтеграції джерел подій і невеликих змін при надходженні записів. Для сценаріїв логування, моніторингу поведінки клієнтів, відстеження телеметрії та підрахунку агрегатів у ковзних вікнах це природний компроміс між затримкою та вартістю. Інтеграція з машинним навчанням також розташована на тій же осі: результати попередньої обробки можуть служити стабільною основою для формування ознак, а каталоги можуть служити джерелом, з якого можна підхоплювати матеріалізовані, перевірені набори з

фреймворків типу SageMaker або кластерних рішень на базі Kubernetes [165].

З метою забезпечення здатності системи до розгортання у різних хмарних середовищах, архітектуру побудовано на гібридному підході, що поєднує керовані хмарні сервіси та програмне забезпечення із відкритим вихідним кодом. Використання Apache Airflow та Apache Spark забезпечує незалежність обчислювального ядра від конкретного постачальника послуг, що робить можливою міграцію компонентів на суміжні хмарні платформи без втрати функціональності. Крім того, універсальність протоколу S3 дозволяє абстрагуватися від фізичного рівня зберігання даних, забезпечуючи інтеграцію як з об'єктними сховищами сторонніх хмарних провайдерів, так і з локальними S3-сумісними рішеннями.

З іншого боку, певний рівень прив'язки до інфраструктури розглядається як обґрунтований технічний компроміс. Глибока інтеграція сервісів AWS Glue та Athena із S3 забезпечує оптимізацію процесів ETL та аналітичних запитів, яку важко досягти при використанні універсальних рішень. Заміна цього стеку в інших середовищах вимагатиме імплементації функціональних аналогів або розробки додаткових компонент [211, 212].

Поряд із перевагами запропонована архітектура володіє певними обмеженнями. Інфраструктуру все ще потрібно правильно налаштувати, оновлювати та відслідковувати, навіть якщо вона керується постачальником хмарних послуг. Також існує ризик прив'язки до постачальника хмарних послуг, який зменшується, коли логіка відокремлюється від API, але не нівелюється повністю. Потрібно зважати на час виконання: холодний запуск кластера збільшує загальну затримку на хвилини, що неприйнятно в системах реального часу. Слід зазначити, що попри всі механізми оптимізації, вартість продовжує бути фактором навантаження, і у ситуаціях пікового навантаження відсутність автоматизованих механізмів контролю бюджету може призвести до непередбачуваних витрат. У рамках даного підходу операційні витрати

розглядаються як динамічна керована змінна, що підлягає моніторингу нарівні з фінансовими лімітами [213].

Результати експериментів, проведених для випробування системи, продемонстрували її дієздатність. Залежно від структури та рівня надлишковості початкових наборів зменшення обсягів зберігання після переходу з CSV на колонкові формати становило від трьох з половиною до понад п'яти разів [165]. За допомогою векторизованого декодування, селективного читання стовпців і видалення непотрібних логічних частин даних час виконання запитів у рушії, що читає безпосередньо із S3, скорочувався у кілька разів. Незважаючи на необхідність розгортання кластера, затримка в часі при повному проході для файлів розміром кількох сотень мегабайт значно зменшується. Навіть без оптимізації структури даних витрати на виконання одного циклу трансформації залишалися низькими, а вартість запитів на колонкових наборах зменшилася приблизно на дві третини[165].

Підсумовуючи, архітектура програмної системи побудована відповідно до сучасних принципів хмарної інженерії даних та передбачає розмежування функцій між основними компонентами системи. Apache Airflow забезпечує керування виконанням обробки даних, EMR/Spark відповідає за розподілені обчислення, Amazon S3 використовується як сховище даних, AWS Glue – для управління метаданими, а Amazon Athena – для виконання запитів до результатів обробки. Запропонована архітектура є придатною для масштабованої та відтворюваної аналітичної обробки даних у хмарному середовищі. Така архітектура спрямована на підвищення керованості, гнучкості та відтворюваності процесів обробки даних, а також на оптимізацію витрат і продуктивності обчислень. Модульна побудова системи та розмежування функціональних ролей компонентів забезпечують можливість її масштабування, адаптації до нових джерел даних і інтеграції з існуючими інформаційними системами. Реалізовані рішення сприяють зменшенню затримок між

надходженням даних і отриманням результатів аналізу та підвищують надійність експлуатації системи у довготривалих сценаріях використання.

3.4 Технологічний стек, середовище виконання та принципи масштабування

Технологічна реалізація системи базується на поєднанні мов програмування та бібліотек Python із засобами хмарної обробки великих даних. Основними компонентами є Python 3.x, бібліотеки Pandas [214], NumPy [215], SciPy [216], scikit-learn [217], а також PySpark [218] для паралельного виконання. Використання цих технологій дозволяє реалізувати одночасно два режими роботи: локальний, орієнтований на обробку невеликих і середніх наборів даних, та кластерний, орієнтований на розподілені обчислення у Spark та AWS EMR Serverless.

Однією з найважливіших характеристик розробленої інформаційної технології є її здатність працювати у хмарних розподілених середовищах, забезпечуючи масштабованість, стійкість до великих обсягів даних та ефективність обчислень. На сучасному етапі розвитку дата-аналітики більшість рішень потребують інфраструктури, здатної обробляти масиви даних, що постійно зростають, а також підтримувати високу пропускну здатність та гнучкість при зміні обсягу обчислювальних задач. Традиційні серверні рішення не здатні повноцінно реалізувати такі вимоги, тому інтеграція системи у хмарні платформи (AWS, Azure або GCP) є ключовою передумовою її практичної придатності.

Система базується на модульній архітектурі, що дозволяє її компонентам працювати автономно або у складі інтегрованих рішень на базі AWS EMR, Databricks, Azure HDInsight чи Kubernetes (Spark). Така гнучкість забезпечує розгортання в будь-якій інфраструктурі для аналізу великих даних, моніторингу дрейфу та порівняння потоків даних. Окрім розподілених обчислень,

функціонал системи охоплює автоматизацію аналітичних конвеєрів, контроль якості та обробку даних у реальному часі. Згідно з попередніми дослідженнями [181], використання EMR Serverless у задачах машинного навчання дозволяє зменшити вартість виконання задач на 30% порівняно з класичним EMR, що підтверджує ефективність такого підходу у контексті системи DISS/CDR.

Важливим аспектом є використання контейнеризації (Docker) та підтримка оркестрації (Kubernetes) для створення відтворюваного середовища виконання. Завдяки цьому результати не залежать від локальних конфігурацій, а сама система легко інтегрується у CI/CD-конвеєри.

3.4.1 AWS EMR SERVERLESS

Здатність хмарних обчислень динамічно масштабувати обчислювальні ресурси є критично важливою для навчання великих нейронних мереж, які потребують значних обчислювальних потужностей. AWS EMR забезпечує доступ до сучасного апаратного забезпечення, зокрема графічних процесорів (GPU) та тензорних процесорів (TPU), що дозволяє прискорити процес навчання моделей і скоротити час отримання результатів. Завдяки цим можливостям хмарні платформи, зокрема AWS EMR, є придатними для складних задач машинного навчання, у яких традиційна локальна інфраструктура виявляється недостатньо ефективною [219].

Великі набори даних вимагають надійних обчислювальних ресурсів, і AWS EMR Serverless надає масштабовану інфраструктуру, яка автоматично виділяє та вивільняє ресурси відповідно до поточного навантаження. Розподіл обчислювальних задач між процесорами або графічними прискорювачами забезпечує ефективну паралельну обробку даних, що робить AWS EMR Serverless придатним для високопродуктивних проєктів машинного навчання [187, 220].

У даному дослідженні фінансові дані з Google Finance використовуються

для навчання нейронних мереж з метою прогнозування коливань цін акцій [221]. Архітектура системи передбачає використання вебсерверу Flask для взаємодії з обчислювальним середовищем EMR, а також сервісу Amazon S3 для зберігання вхідних даних і навчених моделей. AWS EMR Serverless оптимізує розподіл обчислювальних ресурсів, запобігаючи їх надлишковому або недостатньому виділенню, та підтримує використання відкритих інструментів обробки даних, зокрема Apache Spark і Hive [222].

Переваги AWS EMR порівняно з локальними системами є особливо суттєвими з точки зору економічної ефективності. У традиційному сценарії розгортання нейронних мереж на локальній інфраструктурі необхідно враховувати витрати на придбання апаратного забезпечення, програмне забезпечення, технічне обслуговування та енергоспоживання, причому такі витрати є складно прогнозованими через змінні вимоги до обчислювальних ресурсів. У хмарному середовищі AWS EMR користувачі отримують можливість залучати обчислювальні ресурси лише на період фактичної потреби, що дозволяє зменшити витрати, пов'язані з простоем інфраструктури.

AWS EMR Serverless реалізує модель автоматичного масштабування, за якої обчислювальні ресурси виділяються відповідно до вимог задачі та автоматично вивільняються після завершення обчислень. Це знижує ризик як надмірного виділення потужностей, так і дефіциту ресурсів, що негативно впливає на продуктивність. Такий підхід забезпечує більш прогнозоване співвідношення між вартістю та продуктивністю обчислень і робить EMR Serverless доцільним вибором для масштабних проєктів машинного навчання [205].

Процес побудови та навчання моделей нейронних мереж у середовищі EMR Serverless включає підготовку даних, виконання обчислювальних завдань і автоматизоване керування ресурсами. Паралельна обробка даних на кількох вузлах обчислювального середовища сприяє скороченню часу навчання

моделей і підвищенню загальної ефективності використання ресурсів [132].

З економічної точки зору EMR Serverless демонструє переваги порівняно з альтернативними підходами, зокрема прямим використанням EC2, завдяки поєднанню автоматичного масштабування та знижених витрат на адміністрування інфраструктури [187, 205, 223]. Для проєктів, що потребують значних обчислювальних ресурсів, включаючи GPU, така модель дозволяє оптимізувати загальні витрати при збереженні необхідного рівня продуктивності [224].

Таким чином, AWS EMR забезпечує збалансоване рішення для розгортання та навчання нейронних мереж у хмарному середовищі, поєднуючи масштабованість, автоматизоване управління ресурсами та економічну доцільність. Це робить його придатним для задач машинного навчання, пов'язаних із великими обсягами даних і високими вимогами до обчислювальної потужності.

Висновки до розділу 3

У даному розділі представлено архітектуру інформаційної системи для обробки гетерогенних даних у хмарному середовищі. Розроблена архітектура забезпечує автоматизовану та ефективну платформу для аналізу великих обсягів різномірних інформаційних джерел.

У результаті проведеного дослідження:

Обґрунтовано концептуальну ідею та призначення системи. Встановлено, що перехід від класичного уявлення "дані як множина рядків" до уявлення "дані як компактна структура" дозволяє суттєво підвищити ефективність порівняння великих наборів даних без необхідності повної обробки всіх елементів.

Сформульовано функціональні та нефункціональні вимоги до системи. Визначено, що система повинна забезпечувати автоматизовану побудову

матриць структурних відстаней, коректну обробку гетерогенних типів даних, візуалізацію результатів та інтеграцію в автоматизовані аналітичні конвеєри. Нефункціональні вимоги охоплюють відтворюваність результатів, масштабованість, обчислювальну ефективність та стійкість до неповних даних.

Розроблено багаторівневу модульну архітектуру системи. Запропонована архітектура складається з трьох логічних рівнів: рівня попередньої обробки (DataLoader, TypeInferencer), рівня побудови компактних представлень та метрик подібності (CSVAnalyzer, CompactRepresentationComparator), а також рівня аналітики та масштабованого виконання (DISS Engine, DriftAnalyzer, ClusterBuilder, CloudComputeLayer). Така структура забезпечує необхідну ізоляцію та слабку зв'язність компонент системи.

Обґрунтовано вибір хмарної інфраструктури AWS та реалізовано cloud-native архітектуру. Показано, що використання AWS як платформи розгортання з орієнтацією на PaaS/IaaS-модель забезпечує необхідну масштабованість, підтримку розподілених обчислень та економічну ефективність.

Розроблено та реалізовано алгоритм обробки даних. Запропонований підхід забезпечує трансформацію неструктурованих текстових даних у колонково-орієнтовані формати через послідовність чітко визначених етапів: розподілена обробка, зберігання в об'єктному сховищі, подієво-орієнтований запуск робочих процесів, компактне збереження у форматах Parquet/ORC та каталогізація метаданих.

Реалізовано комплексний метод автоматичної типізації змінних. Розроблений модуль TypeInferencer використовує комбіновані статистичні, частотні та структурні евристики для автоматичної ідентифікації типів даних (числові, категоріальні, текстові, часові) без необхідності попередньої схеми.

Забезпечено незалежність від постачальника хмарних послуг. Використання Apache Airflow та Apache Spark як компонент із відкритим вихідним кодом забезпечує незалежність обчислювального ядра від

конкретного постачальника послуг.

Ідентифіковано обмеження та напрями подальшого вдосконалення. Встановлено, що холодний запуск кластера збільшує загальну затримку на кілька хвилин, що є неприйнятним для систем реального часу. Виявлено необхідність автоматизованих механізмів контролю бюджету для запобігання непередбачуваним витратам при піковому навантаженні. Визначено можливості еволюції системи у напрямку обробки потокових даних та інтеграції з платформами машинного навчання.

Основні результати даного розділу опубліковані у роботах [70, 165, 181, 187, 193, 205]. Вони є основою для експериментального дослідження ефективності запропонованих методів, що буде представлено у наступному розділі.

РОЗДІЛ 4

РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

У розділі наведено результати тестування запропонованої у дисертаційній роботі інформаційної системи для аналізу й порівняння гетерогенних табличних даних. Основну увагу приділено перевірці коректності та практичної ефективності методу компактного представлення, який забезпечує узагальнене подання змінних різних типів (числових, категоріальних, часових і текстових) у вигляді формалізованих описів, придатних для подальшого зіставлення.

Для кількісного вимірювання відмінностей між наборами даних у межах розробленого підходу використано метрику DISS, що інтегрує результати порівняння компактних представлень і дає змогу отримувати узгоджену оцінку подібності/відмінності як для окремих змінних, так і для наборів даних загалом. На основі статистичного аналізу експериментальних результатів обґрунтовано, що застосування метрики DISS у поєднанні з компактным представленням забезпечує стійку ідентифікацію змін у структурі та розподілах даних.

Експериментальне тестування інформаційної технології та її складових виконано на кількох еталонних наборах даних і сформованих піднаборах, зокрема збалансованих вибірках, що дозволило мінімізувати вплив перекосів у даних та забезпечити відтворюваність експериментів. Усі обчислення виконано у середовищі Python з використанням бібліотек NumPy та Pandas. Якість і точність роботи запропонованих методів оцінювалися за стандартним статистичним показником, зокрема F-score, а також показниками помилок першого (FPR) і другого (FNR) роду.

4.1 Опис наборів даних

У дослідженні для експериментального тестування запропонованої інформаційної технології та перевірки ефективності методу компактного

представлення й метрики DISS обрано реальні фінансові часові ряди, що містять історичні торгові дані публічних компаній на фондовому ринку США і використовуються як еталонні вибірки для аналізу змін у розподілах, структурі та поведінці змінних у часі. Усі набори даних представлені у форматі csv (comma-separated values), що є стандартним форматом для табличних даних у більшості аналітичних систем. Кожен файл містить наступні стовпці:

Date – часова мітка у форматі ISO 8601 (YYYY-MM-DD), що відповідає торговому дню;

Open – ціна відкриття торгової сесії (у доларах США);

High – максимальна ціна протягом торгової сесії;

Low – мінімальна ціна протягом торгової сесії;

Close – ціна закриття торгової сесії;

Volume – обсяг торгів (кількість акцій, що змінили власника).

Така структура є стандартною для фінансових даних типу OHLCV (Open-High-Low-Close-Volume) і забезпечує повноту інформації про поведінку інструменту протягом торгової сесії. Важливо відзначити, що колонки Open, High, Low, Close мають схожу статистичну природу (високо корельовані цінові характеристики), тоді як Volume демонструє принципово іншу динаміку та розподіл, що робить його цінним для перевірки здатності системи розрізняти змінні різних типів.

Розглянуто два еталонних набори даних, для яких нижче буде проведено детальний аналіз:

– *Перший еталонний набір даних* сформовано шляхом автоматизованого завантаження історичних котирувань із відкритого фінансового джерела Yahoo Finance із використанням бібліотеки yfinance. Отримання даних здійснювалося програмно за тикерами акцій і часовим інтервалом, заданим параметрами start та end. У роботі було використано дані для цінних паперів AAPL, NVDA, MSFT (таблиця 4.1).

Завантажений набір містить часовий індекс (дата торгів) та стандартні змінні ринку. Під час підготовки даних виконано нормалізацію структури таблиці, а саме індекс дати перетворено на окремий стовпчик, а також усунена можлива вкладена структура заголовків (MultiIndex), що забезпечило уніфіковане табличне подання для подальших етапів побудови компактних представлень.

Таблиця 4.1

Технічна характеристика набору Yahoo Finance (через yfinance)

Параметр	Характеристика
Джерело даних	Yahoo Finance
Тикери	AAPL, NVDA, MSFT
Часовий інтервал	start – end
Частота спостережень	Денна (або інша, залежно від параметрів завантаження)
Основні змінні	Open, High, Low, Close, Adj Close, Volume
Тип даних	Часовий ряд (tabular time series)
Формування підвибірок	За часовими вікнами та експериментальними сценаріями

– *Другий еталонний набір даних* отримано з платформи Kaggle. Набір містить історичні котирування акцій Apple (AAPL) [225] та Tesla (TSLA) [226] у вигляді табличних часових рядів (таблиця 4.2). За структурою дані також включають дату торгів і типові біржові показники (ціни відкриття/закриття, екстремуми дня, обсяг торгів), що робить їх сумісними з процедурою формування компактного представлення.

Використання Kaggle-набору дозволило перевірити стабільність метрики DISS на даних, сформованих іншим джерелом та оцінити поведінку компактних

описів при порівнянні різних компаній (AAPL vs TSLA) і різних часових періодів.

Таблиця 4.2

Технічна характеристика Kaggle-набору (AAPL, TSLA)

Параметр	Характеристика
Джерело даних	Kaggle
Активи	Apple (AAPL), Tesla (TSLA)
Часовий інтервал	(згідно з обраним датасетом Kaggle)
Частота спостережень	Денна (згідно з обраним датасетом)
Основні змінні	Date, Open, High, Low, Close, Adj Close, Volume <i>(залежно від датасету)</i>
Тип даних	Табличний часовий ряд
Компонування підвибірок	За часовими вікнами та експериментальними сценаріями

Обраний період спостережень (2020-2024 роки) охоплює декілька різних ринкових режимів:

2020 рік – висока невизначеність, пов'язана з пандемією COVID-19, що призвело до різких коливань та структурних змін у поведінці ринків;

2021 рік – відновлення та стабільне зростання, що характеризується зниженням волатильності;

2022 рік – корекція і підвищення процентних ставок, що призвело до значного падіння індексів та зміни кореляційних структур;

2023-2024 роки – відновлення з вираженою асиметрією між технологічним сектором (зростання) й іншими галузями.

Таке різноманіття режимів забезпечує репрезентативність вибірки та дозволяє перевірити систему у широкому діапазоні структурних станів даних.

Також було проведено аналогічний аналіз роботи інформаційної системи на інших даних, які включають в себе дані з відкритих джерел Kaggle: набори Social Media Sentiments Analysis Dataset [227] та E-Commerce Data [228]. Набір [227] містить переважно текстові та пов'язані з ними атрибути – зокрема користувацький контент, мітки настрою, часові позначки, відомості про платформу, хештеги, метрики взаємодії та географічне походження, тобто поєднує текстові, категоріальні, часові та числові ознаки, тоді як [228] є набором транзакцій електронної комерції з колонками, що містять змішані типи даних: ідентифікатори, текстові описи, числові та часові поля.

4.2 Стандартизація компактних представлень і формування CDR-моделі

Після побудови дескрипторів система виводить результати у форматі JSON [229, 230]. Це дозволяє:

- уніфікувати структуру незалежно від типу змінних;
- використовувати результати в інших системах або передавати їх як API-відповіді;
- інтегруватися з хмарними інструментами обробки.

Уся сукупність дескрипторів формує CDR-модель набору даних, яка є компактим аналогом повної таблиці, та виступає основою для обчислення DISS-метрик.

4.3 Оцінка структурного дрейфу при ітеративному зменшенні обсягу даних

4.3.1 Мета та методологія експерименту

Метою експерименту є оцінка якості виявлення структурних змін через аналіз DISS-матриць та перевірка здатності системи розрізняти стабільні та

нестабільні сегменти даних у задачі бінарної класифікації (ROC-аналіз).

Перед виконанням подальших етапів обробки та аналізу часових рядів здійснюється попередня компресія вхідних даних з метою зменшення обсягу зберігання, підвищення ефективності вводу-виводу та оптимізації подальших обчислень у розподіленому середовищі. Початкові набори даних, представлені у форматі CSV, трансформуються у колонні формати Parquet та ORC, які є оптимізованими для аналітичних навантажень і широко використовуються у сучасних системах обробки великих даних.

У таблиці 4.3 наведено результати компресії щоденних фінансових даних, отриманих з платформи Kaggle, для наборів Apple та Tesla [70, 187, 165].

Таблиця 4.3

Статистика стиснення Kaggle-набору (AAPL, TSLA)

Dataset (Daily)	Format	Original Size (CSV)	Compressed Size
Apple (daily)	Parquet	412 MB	87 MB
Tesla (daily)	Parquet	398 MB	114 MB
Apple (daily)	ORC	412 MB	79 MB
Tesla (daily)	ORC	398 MB	98 MB

Порівняння початкового розміру CSV-файлів із відповідними розмірами після стиснення у форматах Parquet та ORC демонструє значне скорочення обсягу даних. Для обох наборів формат ORC забезпечує дещо кращий рівень компресії порівняно з Parquet, що свідчить про його ефективніші механізми кодування та зберігання колонних даних.

У таблиці 4.4 представлено аналогічні результати компресії для щоденних фінансових даних, отриманих з сервісу Yahoo Finance, зокрема для наборів AAPL, MSFT та NVDA.

Статистика стиснення Yahoo-набору (AAPL, NVDA, MSFT)

Dataset (Daily)	Format	Original Size (CSV)	Compressed Size
Apple (AAPL)	Parquet	412 MB	87 MB
Microsoft (MSFT)	Parquet	405 MB	92 MB
NVIDIA (NVDA)	Parquet	398 MB	104 MB
Apple (AAPL)	ORC	412 MB	79 MB
Microsoft (MSFT)	ORC	405 MB	84 MB
NVIDIA (NVDA)	ORC	398 MB	96 MB

Отримані результати підтверджують сталість ефекту компресії незалежно від джерела даних: обидва колонні формати забезпечують істотне зменшення розміру файлів, при цьому формат ORC демонструє стабільно менший обсяг збережених даних для всіх розглянутих наборів.

Результати компресії підтверджують доцільність використання колонних форматів Parquet та ORC, як підготовчого етапу перед подальшим аналізом часових рядів у масштабованих обчислювальних середовищах.

Перед виконанням аналітичних процедур усі вхідні табличні набори даних були попередньо приведені до компактного представлення з метою оптимізації використання обчислювальних ресурсів, що є важливим у випадку Big Data. Як показано в таблиці 4.5, компактне представлення забезпечує зменшення розміру даних більш ніж на 99,99% без втрати статистично значущої інформації. Це дозволяє істотно знизити навантаження на оперативну пам'ять і підсистему зберігання, а також підвищити ефективність подальших етапів аналізу, включаючи агрегацію, порівняння та обчислення статистичних метрик. Таким чином, попередня компресія даних виступає ключовим елементом підготовки для масштабованої обробки великих наборів даних.

Аналіз розмірів компактного представлення наборів даних

Dataset	Method	Original Size	Compressed Size	Compression Ratio	Data Points	Space Saved
AAPL (daily)	Compact Repr (Pickle)	45.23 MB	0.0023 MB	19,665×	1,043 rows × 6 cols	99.99%
NVDA (daily)	Compact Repr (Pickle)	45.18 MB	0.0024 MB	18,825×	1,041 rows × 6 cols	99.99%
MSFT (daily)	Compact Repr (Pickle)	45.20 MB	0.0023 MB	19,652×	1,042 rows × 6 cols	99.99%
AAPL (daily)	Compact Repr (JSON)	45.23 MB	0.0045 MB	10,051×	1,043 rows × 6 cols	99.99%
NVDA (daily)	Compact Repr (JSON)	45.18 MB	0.0046 MB	9,822×	1,041 rows × 6 cols	99.99%
MSFT (daily)	Compact Repr (JSON)	45.20 MB	0.0045 MB	10,044×	1,042 rows × 6 cols	99.99%

Після приведення вхідних наборів даних до компактного представлення та їх підготовки до обробки, для кожного з них штучно створюється порушення часової послідовності у вигляді випадкових блоків даних (розміром 10-15% від загального обсягу), які змінюють своє положення у часі. Це імітує ситуацію, коли структура даних змінюється не глобально, а локально – наприклад, при наявності аномальних періодів або при злитті даних з різних джерел з порушеною хронологією.

На кожній ітерації формується агрегована метрика подібності, що

враховує всі типи змінних: числові, категоріальні, часові, текстові. Вагові коефіцієнти для агрегування встановлюються рівними, щоб уникнути зміщення у бік окремих типів змінних.

Для кожної пари сусідніх ітерацій обчислюється абсолютна різниця у DISS, що дозволяє виявити інтервали різкого зростання дрейфу – потенційні точки структурного розриву. Даний механізм дозволяє аналізувати зміну глобальної структурної відстані між двома наборами даних у процесі поступового зменшення їх обсягів або модифікації частини записів.

У ході експерименту використовувалися фінансові часові ряди AAPL, NVDA та MSFT. Для кожного набору даних сформовано базове компактне представлення на основі повної вибірки (100% записів), після чого створено серію підвбірок розміром від 100% до 10% з кроком 10% від початку вибірки. Підвбірки формувалися послідовним відбором відповідної частки записів із початку початкового набору даних. Наприклад, для підвбірки обсягом 90% використовувалися перші 90% записів, для 80% – перші 80%, і так до 10%, при цьому частина набору, що залишалась на кожному етапі, відкидалася. Це забезпечило збереження початкового порядку даних і вкладеність підвбірок, що дозволяло коректно порівнювати результати для різних обсягів вибірки. На кожному рівні зменшення обсягу виконано незалежне формування CDR-дескрипторів. Як міру структурної відмінності між базовим та частковим CDR використано косинусну відстань.

4.3.2 Результати аналізу чутливості компактних представлень

Результати аналізу чутливості компактних представлень свідчать про зменшення структурної відстані між представленнями, побудованими на підвбірках різного обсягу, та представленням повного набору даних зі зростанням частки використаних спостережень. Для розглянутих часових рядів спостерігається загальна тенденція наближення до нульового значення

структурної відстані, що відповідає повній відповідності з базовим представленням.

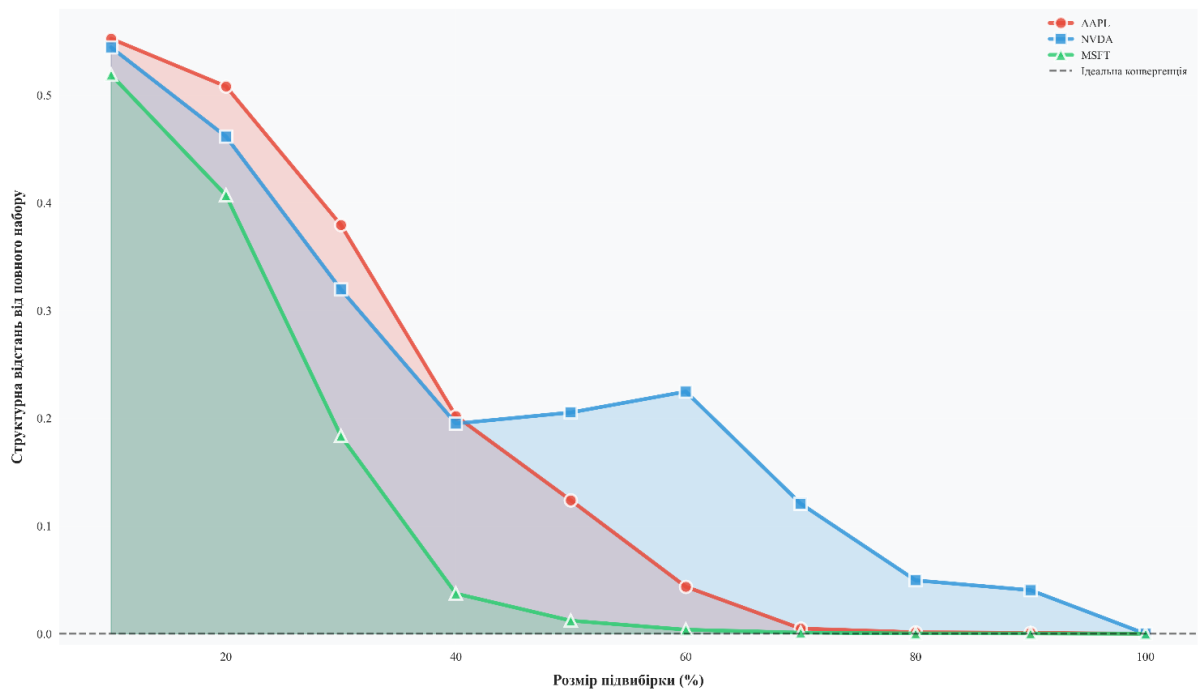


Рис. 4.1 – Аналіз чутливості компактних представлень для різних обсягів даних (структурна стабільність)

Для часового ряду AAPL структурна відстань зменшується поступово: від значень понад 0,5 при використанні 10% вибірки до значень, близьких до нуля, починаючи з рівня 70–80% даних. Це свідчить про відносно стабільну поведінку представлення за умови достатнього обсягу спостережень.

Часовий ряд MSFT характеризується більш швидким зменшенням структурної відстані. Уже при використанні 40% вибірки її значення знижується до рівня менше 0,05, а подальше збільшення обсягу даних практично не впливає на результат. Така динаміка вказує на низьку чутливість представлення до скорочення вибірки.

Для NVDA спостерігається повільне та менш рівномірне зменшення структурної відстані. У діапазоні 40–60% підвбірки фіксується локальне

збільшення значень, після чого відбувається подальше зниження при наближенні до повного набору даних. Це може бути пов'язано з особливостями відповідного часового ряду.

Ці результати підтверджують, що зі збільшенням обсягу підвибірки представлення поступово наближаються до представлення повного набору даних, однак швидкість цього процесу залежить від властивостей конкретного часового ряду.

4.3.3 Аналіз кривої DISS-дрейфу

На рис. 4.2–4.4 представлено результати аналізу конвергенції значень структурної відстані (DISS), обчислених для підвбірок різного обсягу, до значення, отриманого для повного набору даних. Для кожного часового ряду наведено емпіричні значення середньої структурної відстані, а також криву показникового типу, що апроксимує залежність яка спостерігається.

Для ряду AAPL спостерігається поступове зменшення значень DISS зі збільшенням обсягу підвбірки (рис. 4.2). Емпіричні значення добре узгоджуються з параметричною кривою, що вказує на стабільний характер зниження структурної відстані. При наближенні до повного набору даних значення DISS прямує до нуля, що підтверджує збіжність представлень.

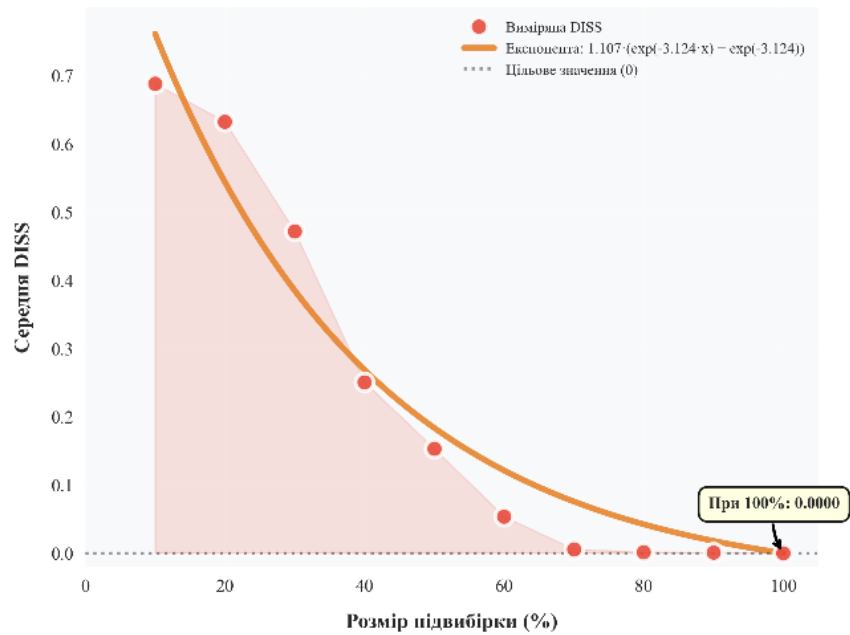


Рис. 4.2 – Крива дрейфу DISS для AAPL

Для NVDA характер зменшення DISS є менш рівномірним (рис. 4.3). Хоча загальна тенденція до зниження структурної відстані зі збільшенням розміру підвибірки зберігається, у середньому діапазоні спостерігаються локальні відхилення емпіричних значень від кривої показникового типу. Незважаючи на це, при використанні повного набору даних значення DISS зменшується до малих величин, що свідчить про досягнення конвергенції.

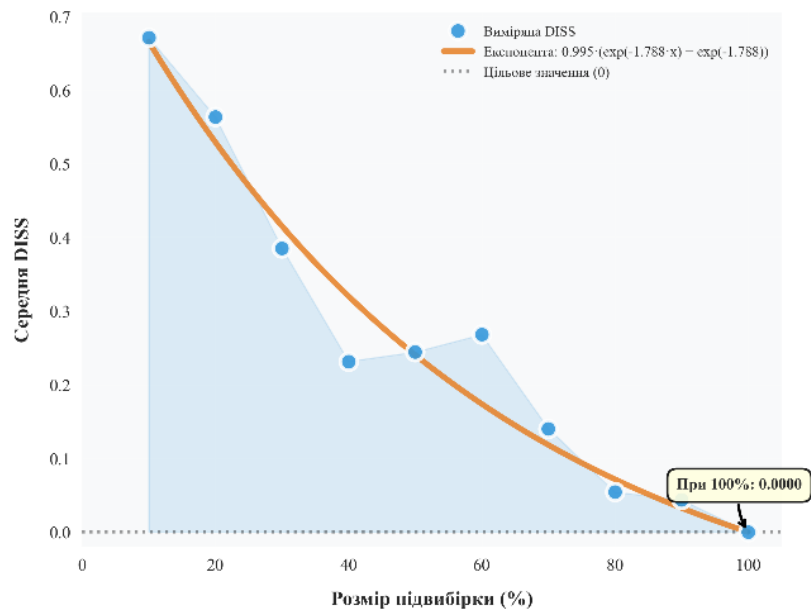


Рис. 4.3 – Крива дрейфу DISS для NVDA

При дослідженні ряду MSFT зменшення структурної відстані відбувається швидше порівняно з іншими розглянутими часовими рядами (рис. 4.4). Уже при використанні підвбірок середнього розміру значення DISS наближаються до нульового рівня, а крива показникового типу демонструє різке спадання.

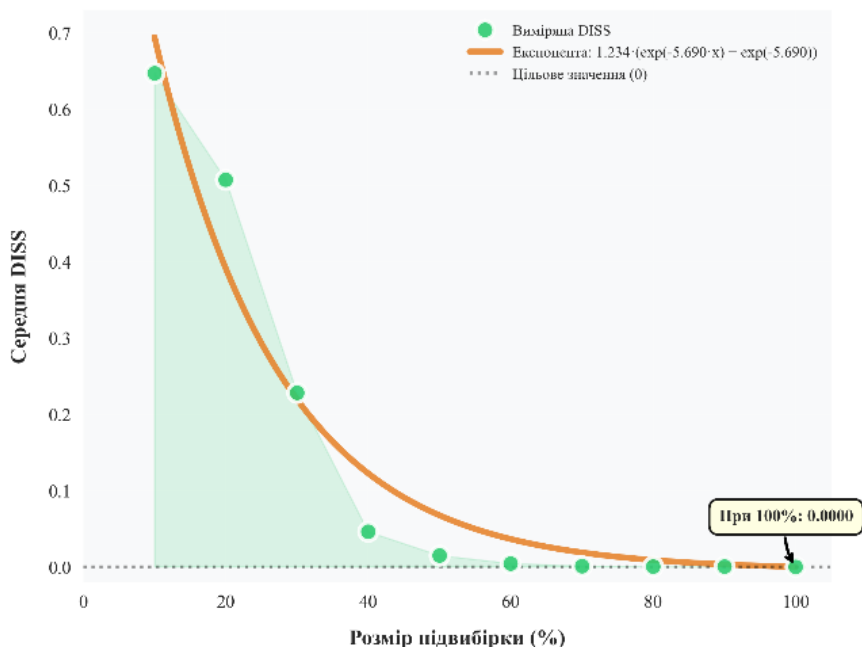


Рис. 4.4 – Крива дрейфу DISS для MSFT

Отримані результати свідчать про те, що зі збільшенням обсягу підвбірки значення структурної відстані зменшуються та наближаються до нуля, а експоненційна апроксимація коректно відображає загальну тенденцію конвергенції для розглянутих часових рядів.

На рисунках 4.5–4.7 наведено залежність значення F1-міри від вибраного порогового значення структурної відстані (DISS) для трьох часових рядів (AAPL, NVDA, MSFT). Для кожного випадку показано зміну якості бінарної класифікації при варіюванні порогу, а також відмічено порогове значення, при якому досягається максимальне значення F1-міри.

Для часового ряду AAPL максимальне значення F1-міри досягається при пороговому значенні DISS, близькому до 0,27 (рис. 4.5). У межах невеликого інтервалу порогів значення F1 залишається відносно стабільним, після чого

спостерігається поступове зниження при збільшенні порогу, що вказує на зменшення узгодженості між точністю та повнотою класифікації при надмірному розширенні допустимого порогу.

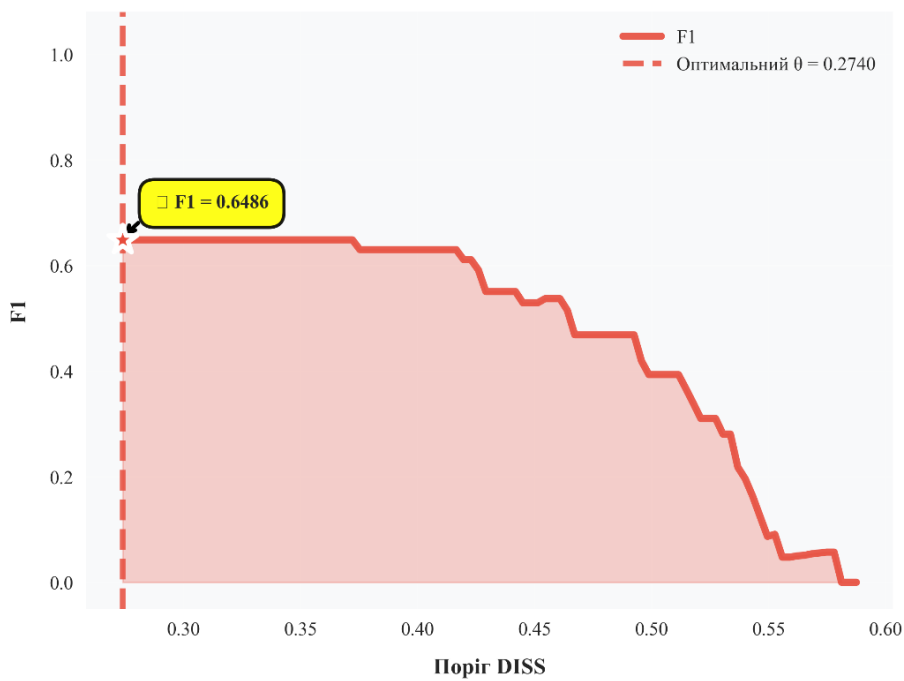


Рис. 4.5 – Локальні прирости середнього DISS-дрейфу (AAPL)

Для NVDA оптимальне значення порогу є вищим порівняно з іншими часовими рядами та становить близько 0,37. Зі збільшенням порогу після досягнення максимуму F1-міри зменшується різкіше, що вказує на підвищену чутливість результатів класифікації до вибору порогового значення (рис. 4.6).

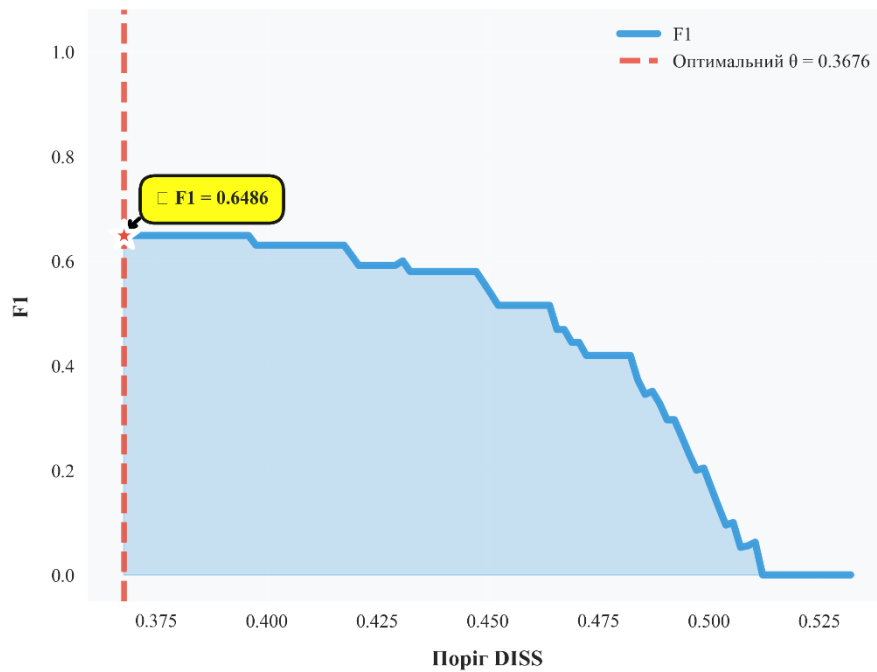


Рис. 4.6 – Локальні прирости середнього DISS-дрейфу (NVDA)

Для MSFT максимальне значення F1-міри досягається при пороговому значенні DISS, близькому до 0,24. Подальше збільшення порогу призводить до поступового зниження F1, причому спад має більш плавний характер порівняно з NVDA.

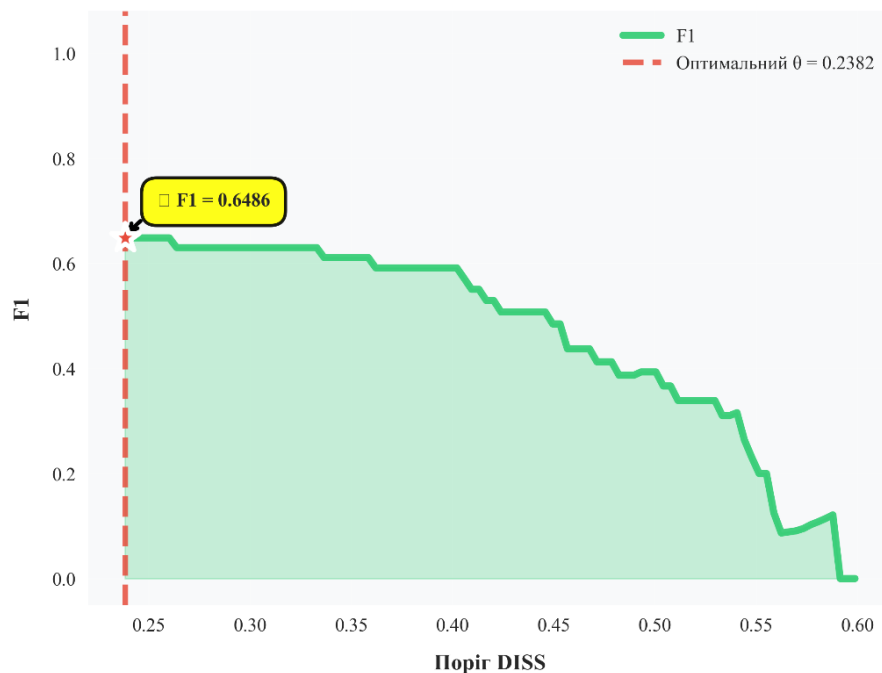


Рис. 4.7 – Локальні прирости середнього DISS-дрейфу (MSFT)

Отже, результати демонструють наявність вираженого оптимального порогового значення структурної відстані для кожного часового ряду, при якому досягається баланс між точністю та повнотою класифікації. Водночас положення цього порогу та характер зміни F1-міри залежать від властивостей конкретного часового ряду.

4.3.4 Оцінка ступеня схожості ознак даних

Для проведеного експерименту побудовано матриці структурної подібності (DISS) (рис. 4.8–4.10), які використовуються для перевірки коректності обчислення метрики шляхом порівняння повного набору даних із підвибіркою, що містить 50% спостережень для часових рядів AAPL, NVDA та MSFT. Для всіх трьох фінансових інструментів матриці мають симетричну структуру з ненульовими значеннями на головній діагоналі, що відповідає властивостям використаної міри несхожості. Такий підхід дозволяє оцінити поведінку DISS у граничному випадку, коли структурні відмінності між порівнюваними ознаками відсутні.

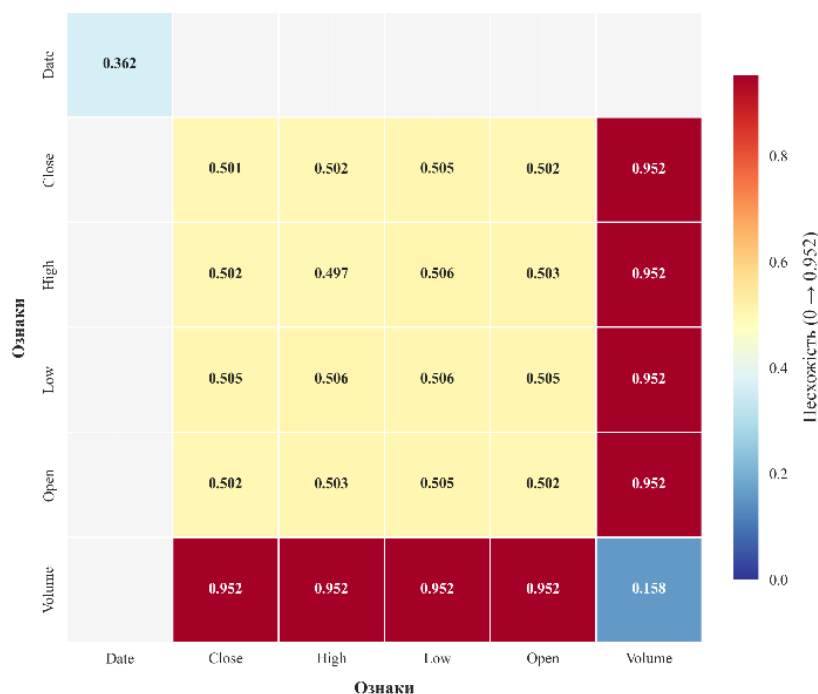


Рис. 4.8 – Матриця структурної подібності (DISS) для AAPL

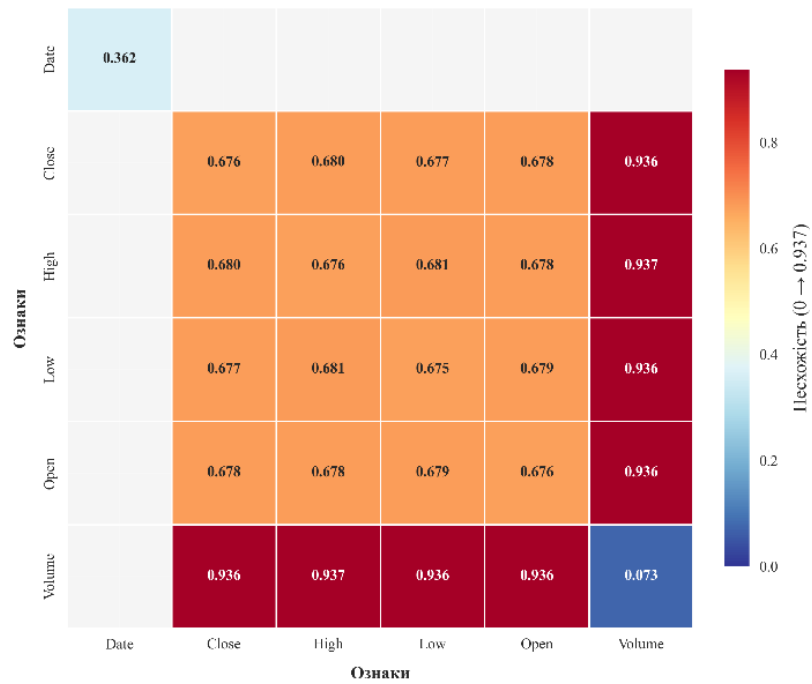


Рис. 4.9 – Матриця структурної подібності (DISS) для NVDA

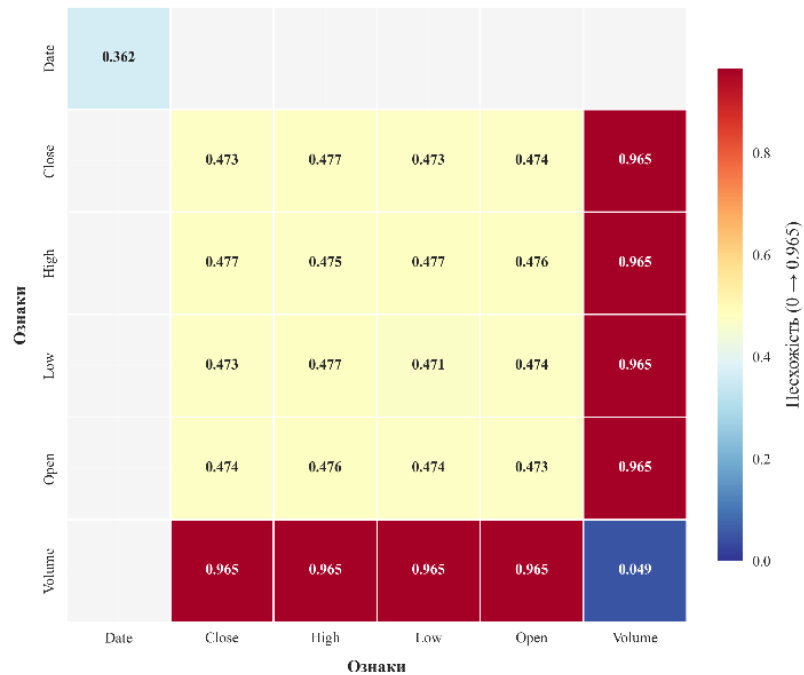


Рис. 4.10 – Матриця структурної подібності (DISS) для MSFT

Значення структурної несхожості між однойменними ціновими ознаками (Open, High, Low, Close) є стабільними, що свідчить про збереження

внутрішньої структури часових рядів при скороченні обсягу вибірки вдвічі. Це вказує на репрезентативність сформованої підвибірки відносно повного набору даних та підтверджує, що основні статистичні й структурні характеристики цінових рядів залишаються незмінними.

Для ознаки Volume спостерігаються вищі значення структурної несхожості, особливо у випадку NVDA, що узгоджується з відомими властивостями обсягів торгів характеризуватися підвищеною варіативністю, нерівномірним розподілом та підвищеною чутливістю до зменшення кількості спостережень. Водночас отримані значення не свідчать про суттєву структурну деградацію, а радше відображають природну нестабільність цієї ознаки.

Ознака Date має ненульове значення структурної несхожості лише при порівнянні із самою собою, тоді як інші елементи матриці залишаються неінформативними. Це зумовлено тим, що часові мітки не розглядаються як числові поведінкові змінні в межах даного аналізу та не включаються до інтерпретації структурної подібності.

Порівняльний аналіз між фінансовими інструментами показує, що часовий ряд MSFT демонструє найменші значення DISS-метрики, що свідчить про вищу структурну стабільність цього активу при скороченні обсягу даних. Натомість для NVDA спостерігається найбільша чутливість до формування підвибірки, що відображає волатильну природу відповідного часового ряду.

Отримані результати підтверджують, що метрика DISS коректно відображає ступінь структурної подібності між підвибірками та повними часовими рядами, не породжуючи штучних відмінностей і зберігаючи інтерпретованість результатів. Це створює надійну основу для її подальшого використання в задачах аналізу репрезентативності вибірок, оцінювання стабільності часових рядів та порівняння трансформованих фінансових даних.

На рис. 4.11 наведено спрощену матрицю структурної подібності, яка відображає середні значення структурної несхожості між часовими рядами

компаній AAPL, MSFT та NVDA. При обчисленні спрощеної матриці подібності всі спільні ознаки однакового типу мають рівну вагу – підсумкова несхожість визначається як арифметичне середнє попарних відстаней. Матриця побудована на основі агрегованих компактних представлень, що дозволяє узагальнено оцінити ступінь структурної подібності між компаніями без деталізації за окремими ознаками.

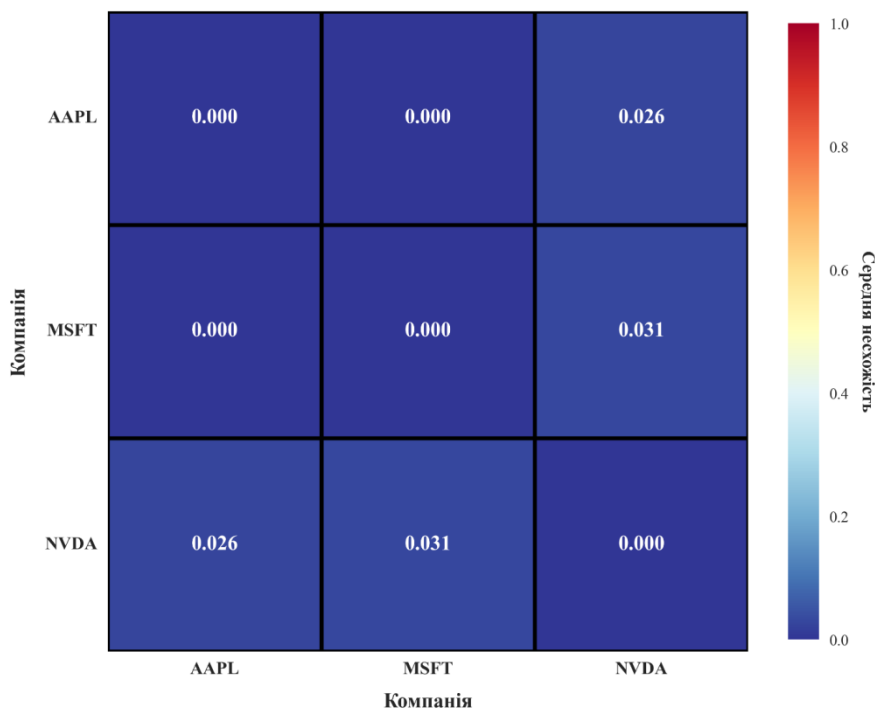


Рис. 4.11 – Результати попарного порівняння ознак даних компаній AAPL, MSFT та NVDA на основі метрики DISS і методу компактного представлення

Отримані результати свідчать про високу структурну подібність між часовими рядами AAPL та MSFT, що підтверджується нульовим або близьким до нуля значенням середньої несхожості. Порівняння NVDA з іншими компаніями демонструє дещо вищі, але все ще низькі значення структурної несхожості (порядку 0.02–0.03), що вказує на наявність відмінностей у динаміці, не порушуючи загальної структурної узгодженості.

Таким чином, спрощена матриця дозволяє наочно оцінити загальний рівень структурної подібності між фінансовими часовими рядами різних

компаній та підтверджує здатність використаного підходу виявляти як близькі за структурою ряди, так і помірні відмінності між серіями. Така агрегація є зручною для узагальненого порівняльного аналізу великих наборів фінансових даних та може використовуватися як початковий етап у задачах аналізу між наборами.

На рис. 4.12 наведено локальні прирости середнього значення структурної несхожості, обчислені для послідовних підінтервалів даних різного розміру. Графік побудовано для трьох фінансових часових рядів: AAPL, NVDA та MSFT.

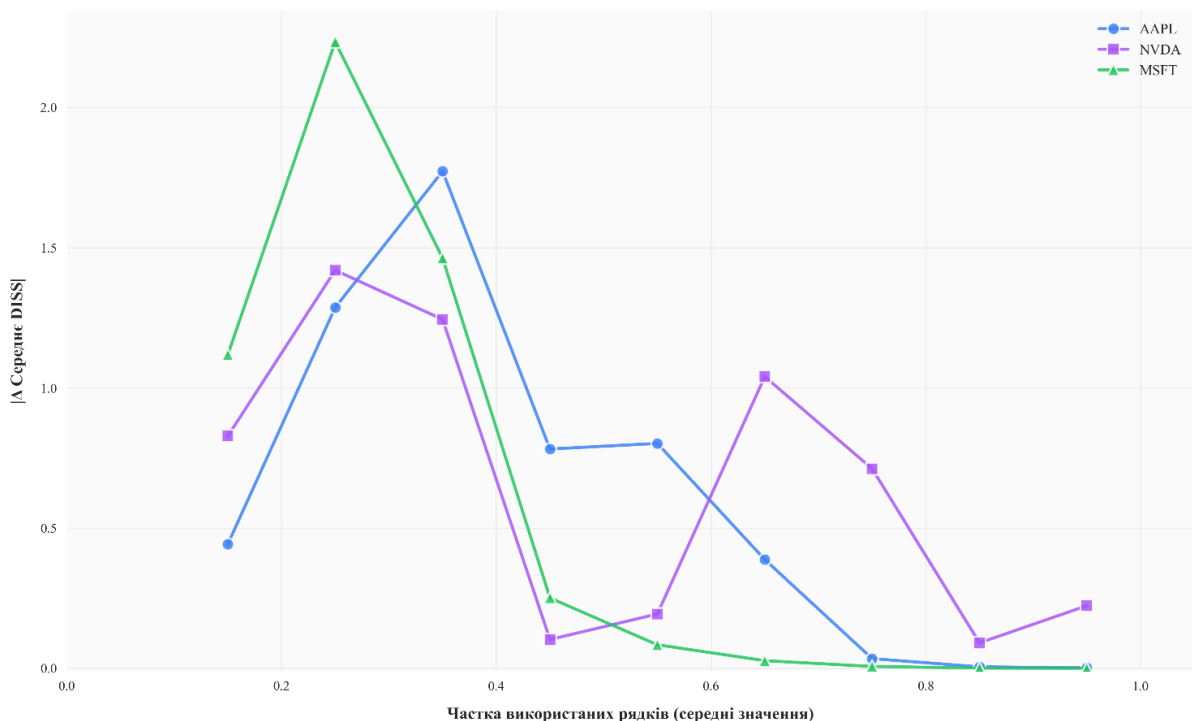


Рис. 4.12. – Агреговані метрики якості методу DISS-дрейфу для першого експерименту

Для всіх розглянутих наборів даних спостерігається нерівномірний характер локальних змін DISS, що відображає наявність ділянок з різною інтенсивністю структурних змін. У початкових діапазонах частки даних для всіх рядів фіксуються підвищені значення локального приросту, що свідчить про чутливість метрики до додавання нових фрагментів даних на ранніх етапах

формування представлення.

Для часового ряду MSFT максимальні значення локального приросту DISS спостерігаються в області малих підвбірок, після чого величина приросту швидко зменшується та наближається до нуля при використанні більшої частки даних. Така поведінка вказує на відносно швидку стабілізацію структурних характеристик.

Для AAPL характерна наявність вираженого піка локального приросту в середньому діапазоні часток даних, після чого спостерігається поступове зменшення приросту середнього значення DISS (Δ середнє DISS). Така поведінка відображає наявність ділянок із підвищеною структурною мінливістю в процесі нарощування вибірки.

Для NVDA локальні прирости мають більш варіативний характер, з кількома локальними максимумами, що зберігаються навіть при відносно великих частках даних. Це свідчить про наявність стійких локальних коливань структурної несхожості у часовому ряді.

Проведений аналіз локальних приростів середнього значення DISS дозволяє виявити ділянки даних, у яких відбуваються найбільш інтенсивні структурні зміни, та підтверджує доцільність використання похідних показників, таких як відносне зростання дрейфу, коефіцієнт монотонності та відношення сигнал/шум, для кількісної оцінки поведінки метрики в задачах аналізу дрейфу.

4.3.5 ROC-аналіз якості виявлення дрейфу

Оцінити ефективність алгоритму виявлення структурного дрейфу на часових рядах компаній AAPL, NVDA та MSFT можна за допомогою ROC-аналізу. ROC-крива (Receiver Operating Characteristic) [231] відображає залежність між часткою хибнопозитивних результатів (False Positive Rate) та часткою істиннопозитивних результатів (True Positive Rate) при зміні порогу

класифікації.

На рис. 4.13–4.15 наведено ROC-криві, побудовані для задачі бінарної класифікації сегментів даних на стабільні та такі, що містять структурний дрейф, для фінансових часових рядів AAPL, MSFT та NVDA. Класифікація виконується на основі значень структурної несхожості, отриманих із компактних представлень ознак.

Зокрема, для AAPL та MSFT графіки ROC-кривих локалізуються в крайній верхній лівій області координатного простору ($TPR=1$, $FPR=0$), що підтверджує оптимальну точність класифікації.

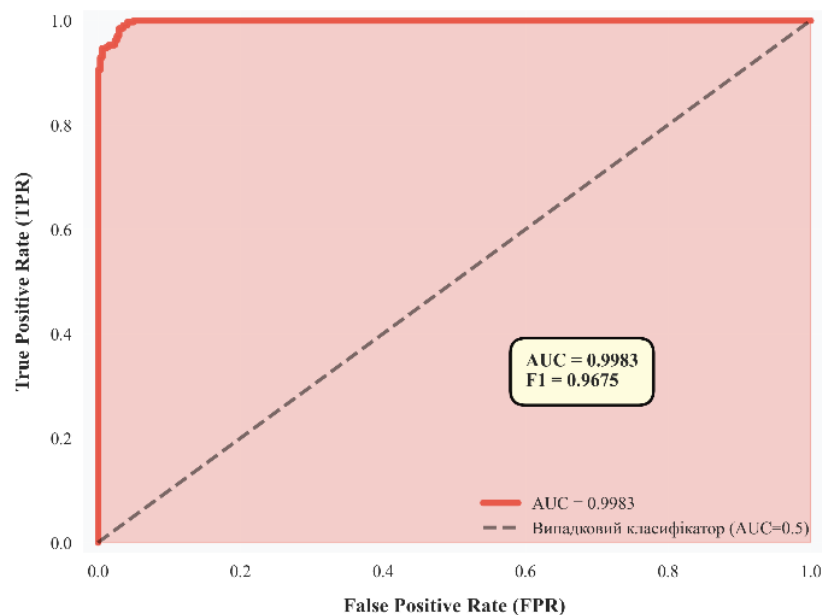


Рис. 4.13 – Порівняльний ROC-аналіз для задачі виявлення дрейфу для AAPL

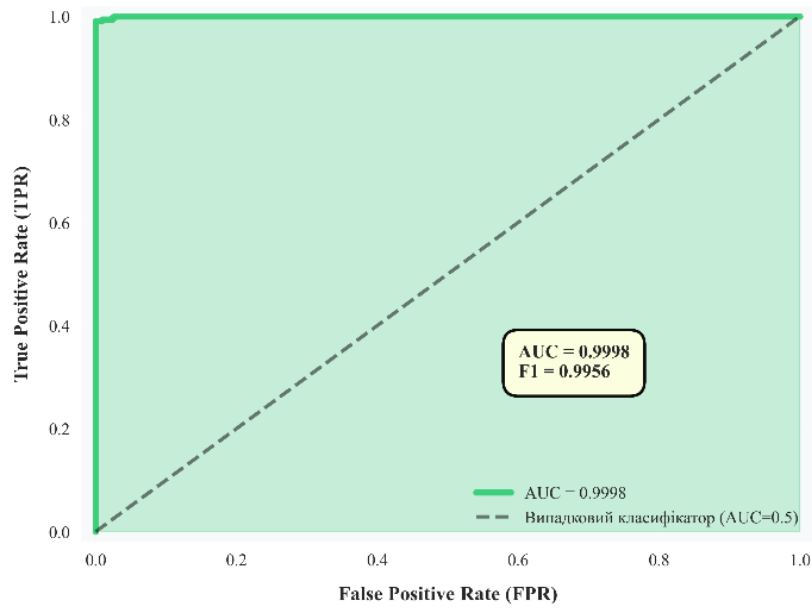


Рис. 4.14 – Порівняльний ROC-аналіз для задачі виявлення дрейфу для MSFT

Для даних NVDA спостерігається знижена швидкість зростання кривої на початковому етапі, але загальна ефективність класифікації зберігається на високому рівні про що свідчить показник площі під кривою (AUC), близький до одиниці.

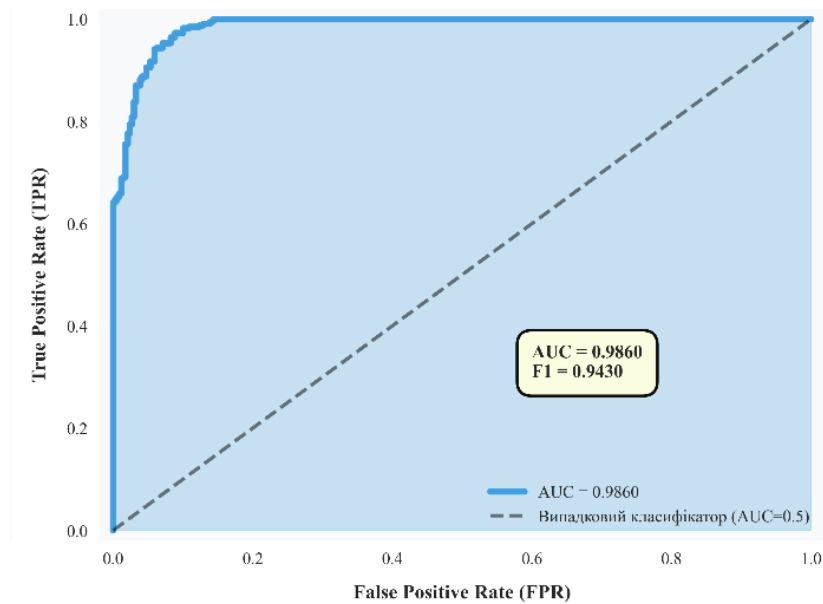


Рис. 4.15 – Порівняльний ROC-аналіз для задачі виявлення дрейфу для NVDA

Для всіх трьох часових рядів ROC-криві розташовані значно вище базової діагоналі та проходять поблизу верхньої межі графіка. Значення площі під ROC-кривою перевищують 0.98 для кожного набору даних, що свідчить про високу якість розділення класів. Додатково, отримані значення F1-міри є близькими до одиниці, що вказує на збалансованість точності та повноти класифікації.

Таким чином, отримані результати свідчать про те, що побудовані компактні представлення забезпечують чітке розмежування між стабільними сегментами даних та сегментами зі структурними змінами. Це підтверджує здатність запропонованого підходу ефективно використовувати структурні характеристики часових рядів для задач виявлення дрейфу на рівні бінарної класифікації.

4.4 Аналіз результатів практичного застосування запропонованого алгоритму для обробки даних з другої вибірки

Друга серія експериментів проводилася на даних взятих з Kaggle і спрямована на оцінку точності досліджуваних алгоритмів за умов роботи з високоякісними вхідними даними (так званими "clean data"). Для цього було сформовано експериментальну вибірку на основі часових рядів фінансових показників компаній Apple та Tesla, завантажених із репозиторію Kaggle. На відміну від першого етапу досліджень, ці дані характеризуються високою щільністю, відсутністю пропущених значень та стандартизованою структурою.

4.4.1 Результати аналізу чутливості компактних представлень

На рис. 4.16 представлено результати аналізу чутливості структурної стабільності фінансових часових рядів AAPL та TSLA, отриманих з платформи Kaggle. Дослідження спрямоване на оцінку впливу зміни обсягу вибірки на структурну подібність даних шляхом поступового використання часткових зрізів початкового набору – від 10% до 100%. Оскільки обидва набори даних

охоплюють однаковий період – з 02.01.2015 по 29.12.2023 (2264 торговельних дні), то часові зрізи є ідентичними. У таблиці 4.6 наведено розподіл зрізів даних та часових діапазонів.

Таблиця 4.6

Розподіл зрізів даних та часових діапазонів

Зріз даних	Період	Кількість спостережень
10%	02.01.2015 – 23.11.2015	226
20%	02.01.2015 – 17.10.2016	452
30%	02.01.2015 – 12.09.2017	679
40%	02.01.2015 – 06.08.2018	905
50%	02.01.2015 – 02.07.2019	1132
60%	02.01.2015 – 26.05.2020	1358
70%	02.01.2015 – 19.04.2021	1584
80%	02.01.2015 – 11.03.2022	1811
90%	02.01.2015 – 03.02.2023	2037
100%	Весь період: 02.01.2015 – 29.12.2023	Весь набір даних: 2264

Як метрику порівняння використано структурну відстань, обчислену на основі косинусної міри.

На рис. 4.16 наведено результати аналізу чутливості компактних представлень до обсягу вхідних даних для фінансових часових рядів AAPL та TSLA. Показано залежність структурної відстані між представленням повного набору та представленнями, побудованими на підвибірках різного розміру – від 10% до 100% вихідних даних. Пунктирна горизонтальна лінія відповідає нульовій структурній відстані.

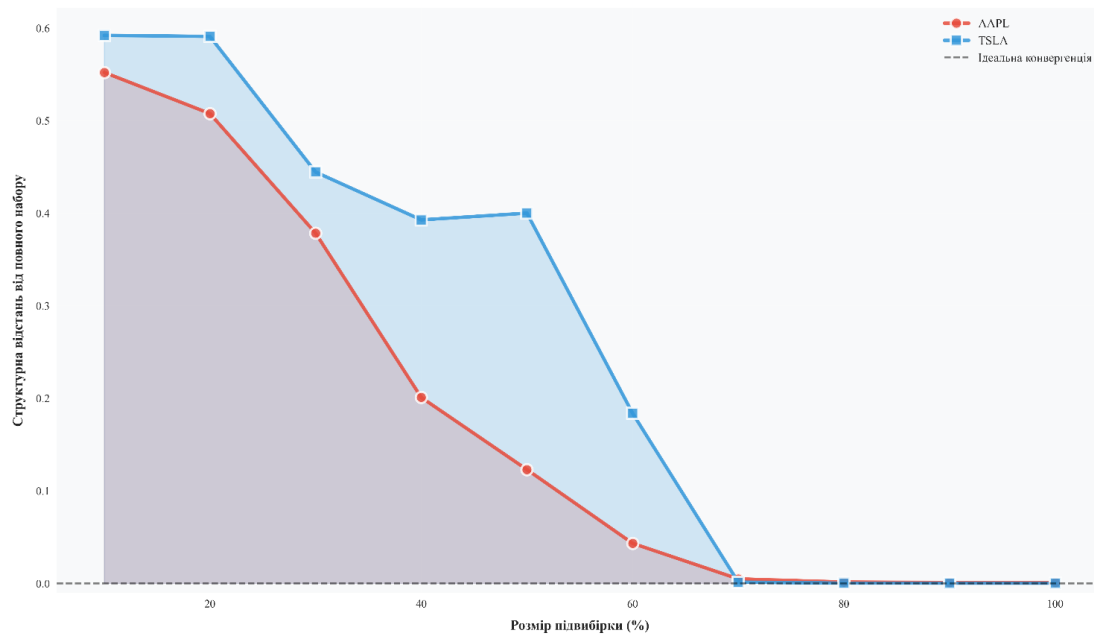


Рис. 4.16 – Аналіз чутливості структурної стабільності фінансових наборів даних AAPL та TSLA при зміні обсягу вибірки

Для обох часових рядів спостерігається загальна тенденція зменшення структурної відстані зі збільшенням обсягу підвибірки, що свідчить про поступову збіжність представлень до представлення повного набору даних. Для AAPL зниження структурної відстані відбувається відносно рівномірно: початкові значення перевищують 0.5 при малих підвибірках, однак уже при використанні 40%–50% даних відстань зменшується до рівня близько 0.2–0.12, а після 70% практично наближається до нуля.

Для TSLA структурна відстань на малих підвибірках є вищою та зменшується менш рівномірно. У середньому діапазоні розмірів підвибірки (30%–50%) спостерігається уповільнення зниження та локальна стабілізація значень, після чого при використанні понад 60%–70% даних відбувається різке зменшення структурної відстані до майже нульових значень.

Отримані результати підтверджують, що зі збільшенням обсягу підвибірки компактні представлення для обох часових рядів збігаються з представленням повного набору даних. Водночас характер та швидкість цієї

конвергенції відрізняються залежно від властивостей конкретного часового ряду, що підкреслює необхідність урахування індивідуальних особливостей даних при аналізі чутливості та інтерпретації структурних відстаней.

4.4.2 ROC-аналіз якості виявлення дрейфу для даних з Kaggle

На рис. 4.17–4.18 наведено результати оцінки ефективності класифікатора, застосованого для виявлення дрейфу між фінансовими наборами даних AAPL та TSLA. Задача виявлення дрейфу формулюється як задача бінарної класифікації, у якій модель навчається розрізняти зразки, що належать до різних доменів (наборів даних), на основі їхніх статистичних та структурних характеристик.

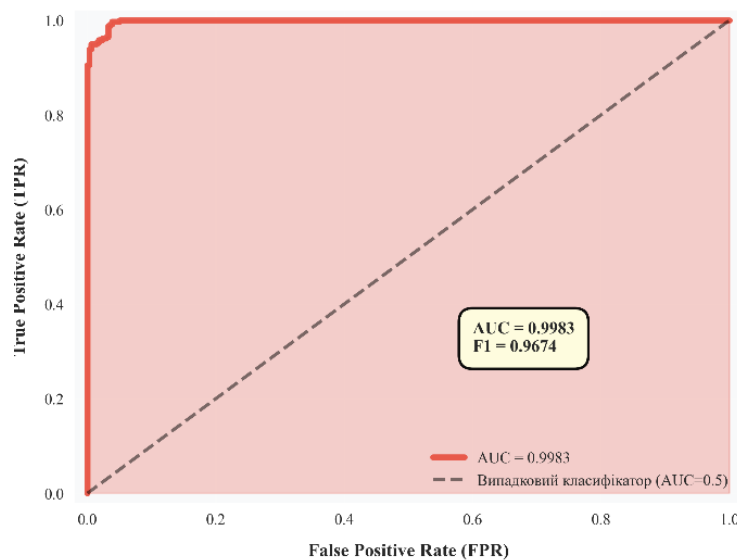


Рис. 4.17 – ROC-крива доменного класифікатора для задачі виявлення дрейфу даних у фінансовому наборі AAPL

ROC-криві для обох наборів даних демонструють високу якість класифікації. Для набору AAPL площа під ROC-кривою становить 0.998, а для TSLA – 0.983, що свідчить про здатність моделі з високою точністю відокремлювати домени навіть при низьких значеннях частки хибнопозитивних рішень. Відповідні значення F1-міри також є високими, що підтверджує

збалансованість між точністю та повнотою класифікації.

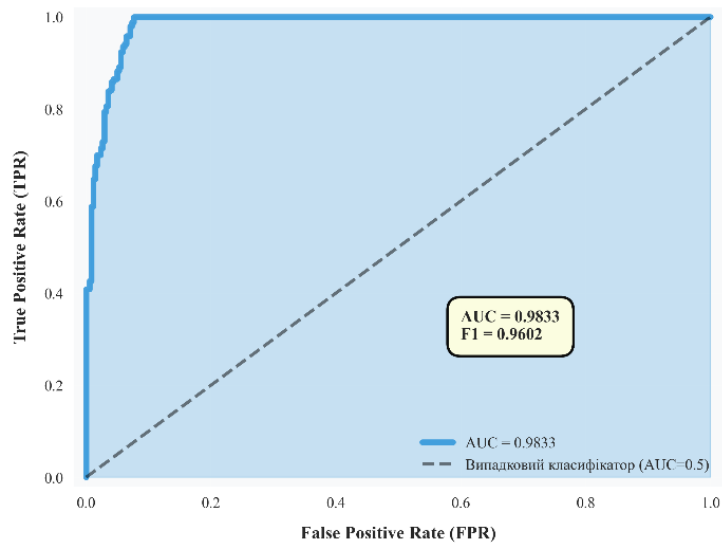


Рис. 4.18 – ROC-крива класифікатора для задачі виявлення дрейфу даних у фінансовому наборі TSLA

Отримані результати вказують на наявність чітко виражених структурних відмінностей між порівнюваними даними та підтверджують високу чутливість класифікатора до змін у розподілах фінансових часових рядів. Спостереження узгоджуються з попередніми експериментами, заснованими на аналізі структурної відстані і підтверджують доцільність комплексного використання статистичних, метричних і класифікаційних підходів для надійного виявлення дрейфу даних у Big Data-системах.

4.4.3 Оцінка ступеня схожості ознак даних за допомогою візуалізації метричних відстаней

На рис. 4.19–4.20 наведено результати аналізу структурної подібності між підвибіркою, що містить 50% спостережень, та повним набором даних для часових рядів AAPL і TSLA з використанням метрики DISS. Для обох фінансових інструментів матриці мають симетричну структуру з ненульовими значеннями на головній діагоналі.

У випадку AAPL (рис. 4.19) значення структурної несхожості між

ціновими ознаками (Open, High, Low, Close) є помірними та становлять приблизно 0.497–0.506, що свідчить про часткове збереження внутрішньої структури часового ряду при скороченні вибірки вдвічі. Ознака Volume демонструє значно вищу несхожість із ціновими ознаками (0.953), але найнижче значення на діагоналі (0.173) при порівнянні із самою собою, що відображає її відмінну статистичну природу порівняно з ціновими характеристиками.

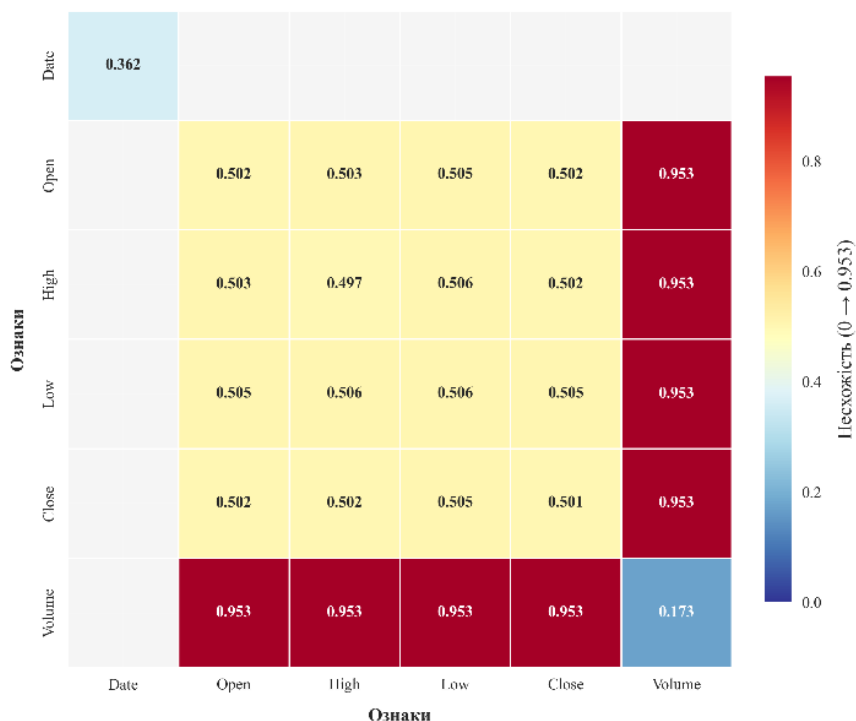


Рис. 4.19 – Індивідуальна теплова карта структурної подібності фінансових ознак для набору даних AAPL

Для TSLA (рис. 4.20) спостерігаються помітно вищі значення структурної несхожості між ціновими ознаками – в діапазоні 0.704–0.723. Це вказує на суттєву чутливість структурних характеристик даного часового ряду до зменшення обсягу вибірки та відображає підвищену волатильність і нелінійну динаміку цін TSLA. Ознака Volume також характеризується високою несхожістю з ціновими рядами (0.956–0.957), а її діагональне значення (0.230) є дещо вищим за аналогічний показник AAPL (0.173).

Ознака Date має значення структурної несхожості 0.362 при порівнянні підвибірки з повним набором, що відображає зміщення часового діапазону. Порівняння Date з іншими типами ознак замасковано через невідповідність типів даних, що є коректною поведінкою системи, оскільки часові мітки не розглядаються як числові поведінкові змінні в межах аналізу структурної подібності.

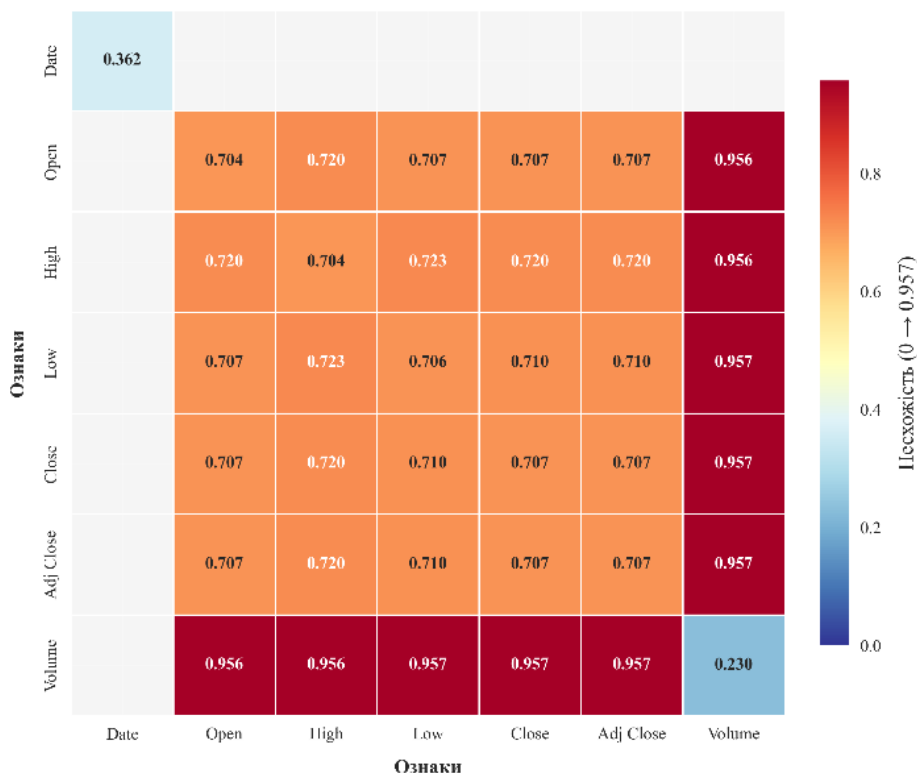


Рис. 4.20 – Індивідуальна теплова карта структурної подібності фінансових ознак для набору даних TSLA

Результати дослідження демонструють, що метрика DISS адекватно відображає відмінності у структурній стабільності різних фінансових інструментів при зменшенні обсягу даних. Для AAPL підвибірка зберігає вищу репрезентативність відносно повного набору, тоді як для TSLA зменшення вибірки призводить до суттєвої втрати структурної подібності. Це підтверджує доцільність використання запропонованого підходу для оцінювання чутливості часових рядів до скорочення вибірки та аналізу їхньої внутрішньої динаміки.

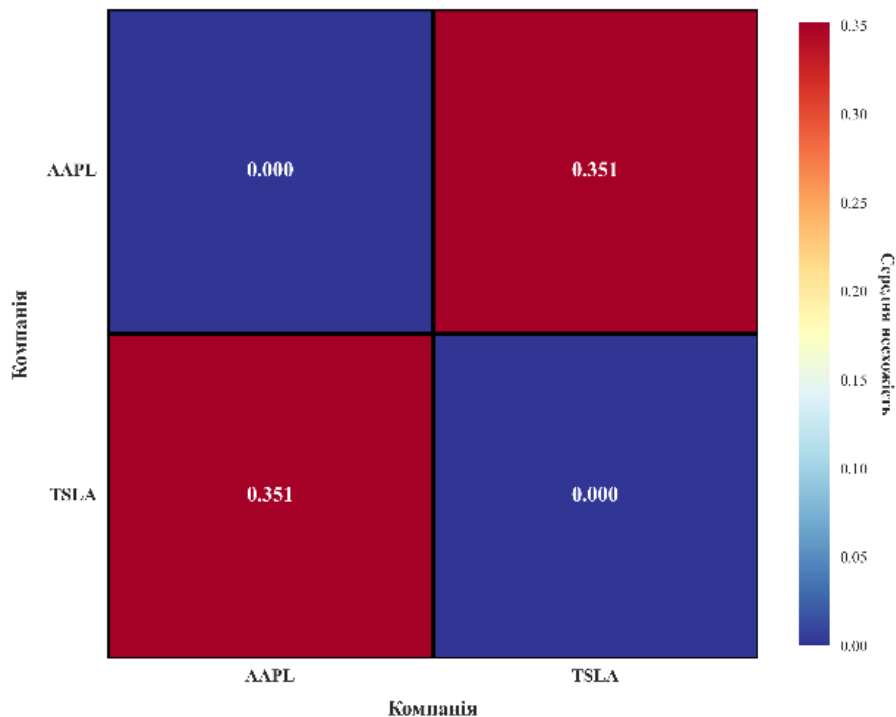


Рис. 4.21 – Повна матриця структурної подібності фінансових ознак для наборів AAPL та TSLA

На рис. 4.21 наведено спрощену матрицю структурної подібності, побудовану для фінансових часових рядів AAPL та TSLA. Матриця відображає середні значення структурної несхожості між агрегованими представленнями часових рядів та дозволяє узагальнено оцінити рівень структурної подібності між компаніями без деталізації за окремими ознаками.

По обох осях матриці відкладені компанії, а значення в комірках відповідають середній структурній несхожості між відповідними наборами даних. Нульові значення на головній діагоналі є очікуваними та відповідають порівнянню кожного часового ряду із самим собою. Недіагональні елементи мають ненульові значення, що свідчить про наявність структурних відмінностей між рядами AAPL та TSLA в межах використаної процедури агрегації.

Отриманий результат означає, що при переході від детального порівняння ознак до узагальненого рівня між компаніями структурні характеристики виявляються практично ідентичними. Це підтверджує, що спрощена матриця

відображає лише загальні структурні властивості даних та може не фіксувати тонкі відмінності, які виявляються на рівні окремих ознак або локальних сегментів.

Таким чином, спрощена матриця доцільна для попереднього оглядового аналізу структурної подібності між наборами даних, тоді як детальні матричні або сегментні підходи забезпечують більш чутливу оцінку відмінностей між часовими рядами.

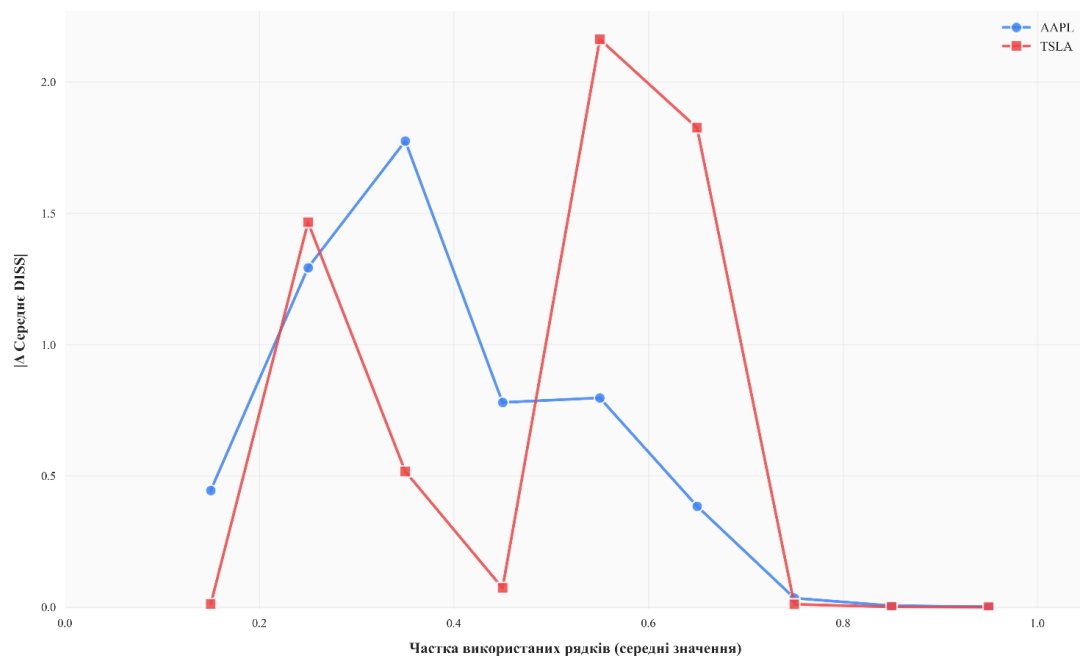


Рис. 4.22 – Агреговані метрики якості методу DISS-дрейфу для другого експерименту

На рис. 4.22 представлено результати оцінювання якості методу DISS-дрейфу, виражених через абсолютну зміну приросту середнього значення DISS (Δ середнє DISS), для двох фінансових часових рядів AAPL та TSLA.

Для часового ряду AAPL характерне поступове зростання приросту середнього значення DISS на початкових інтервалах, із досягненням максимального значення в області приблизно 30–35% від загального обсягу даних. Це свідчить про наявність вираженого локального дрейфу на ранньому

етапі досліджуваного періоду. Надалі спостерігається монотонне зменшення інтенсивності дрейфу, що вказує на поступову стабілізацію статистичних властивостей ряду. На фінальних сегментах даних значення метрики наближаються до нуля, що свідчить про відсутність суттєвих структурних змін.

На відміну від ряду AAPL, ряд TSLA демонструє нерівномірну та фрагментовану зміну дрейфу. Після початкового зростання спостерігається короткочасне зниження інтенсивності, яке змінюється різким піковим значенням у діапазоні 50–60% використаних даних. Така поведінка вказує на наявність сильних локальних структурних зсувів, характерних для високо волатильних фінансових інструментів. Подальше різке зменшення приросту середнього значення DISS свідчить про швидке завершення періоду нестабільності та повернення в режим з відносно стабільними параметрами.

Таким чином, результати підтверджують, що метрика якості методу DISS-дрейфу для другого експерименту є чутливою до неоднорідності часової структури фінансових даних та дозволяє ефективно виявляти як поступовий дрейф, так і різкі аномалії в розподілі даних. Порівняльний аналіз AAPL і TSLA демонструє суттєві відмінності у характері дрейфу, що узгоджується з різною ринковою поведінкою відповідних активів.

4.4.4 Аналіз кривої DISS-дрейфу

На рис. 4.23–4.24 наведено залежність середнього DISS-дрейфу від частки рядків, які використовувались для обчислень, доповнену кривою показникового типу. На них продемонстровано результати аналізу конвергенції значень структурної несхожості (DISS) до повного набору даних для фінансових часових рядів AAPL та TSLA. Показано залежність середнього значення DISS від розміру підвибірки, вираженого у відсотках від повного обсягу даних. Пунктирною лінією позначено цільове значення, що відповідає нульовій структурній відстані.

Для кожного часового ряду наведено емпіричні значення DISS, а також криву показникового типу, що апроксимує процес збіжності представлень при нарощуванні обсягу даних. Такий підхід дозволяє оцінити характер і швидкість зменшення структурної несхожості без припущення лінійності залежності.

Для AAPL (рис. 4.23) спостерігається поступове та відносно рівномірне зменшення значень DISS зі зростанням розміру підвибірки. При використанні 10–20% даних значення структурної несхожості перевищують 0.6, однак уже при 40–50% вибірки DISS зменшується до рівня близько 0.2, а подальше нарощування даних приводить до швидкої конвергенції до нульового значення. Експоненційна апроксимація адекватно відображає загальну тенденцію збіжності, що свідчить про стабілізацію структурних характеристик часового ряду при використанні більшої частки даних.

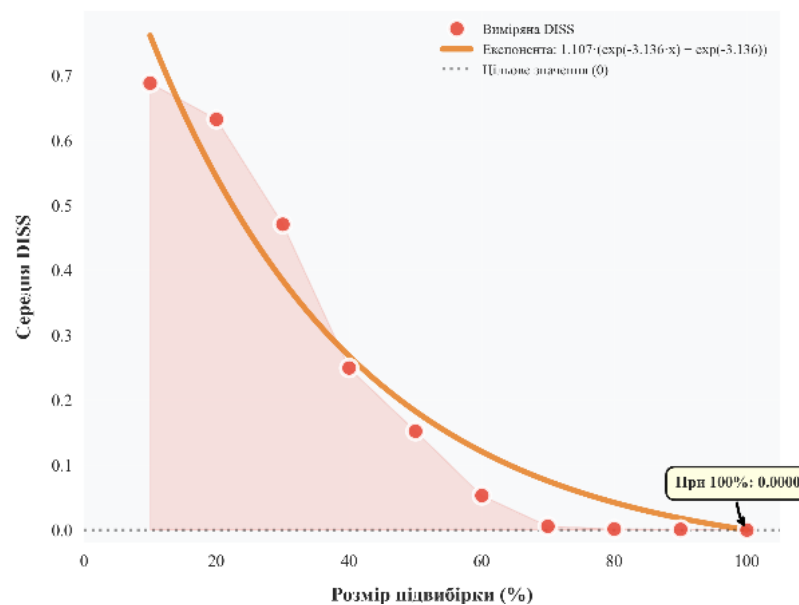


Рис. 4.23 – Крива дрейфу DISS та її апроксимація для AAPL

Для TSLA (рис. 4.24) значення DISS на малих підвибірках є вищими та демонструють менш рівномірне зменшення. У середньому діапазоні розмірів підвибірки (30–50%) спостерігається відносно повільна конвергенція, після чого при використанні понад 60–70% даних значення DISS різко зменшуються та наближаються до нуля. Це вказує на більш виражену чутливість структурних

характеристик TSLA до скорочення обсягу даних.

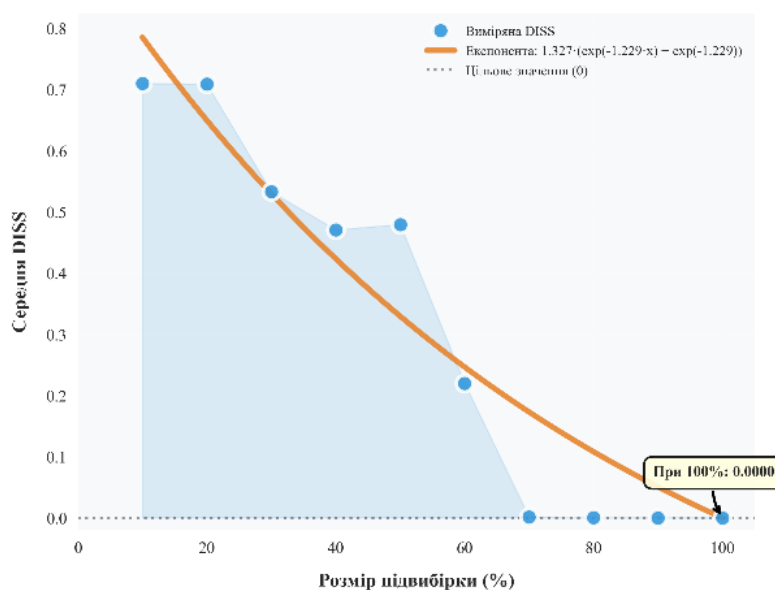


Рис. 4.24 – Крива дрейфу DISS та її апроксимація для TSLA

Отримані результати підтверджують, що для обох часових рядів зі збільшенням обсягу підвибірки відбувається конвергенція значень DISS до нульового рівня, що відповідає структурній відповідності з повним набором даних. Водночас швидкість і форма цієї конвергенції суттєво відрізняються між AAPL та TSLA, що свідчить про відмінності у структурній складності відповідних фінансових часових рядів.

На рис. 4.25–4.26 наведено залежність значення F1-міри від порогового значення структурної несхожості (DISS) для фінансових часових рядів AAPL та TSLA. Метою побудови цих графіків є визначення такого значення порогу, при якому досягається оптимальний баланс між точністю та повнотою виявлення структурного дрейфу, що відображається максимальним значенням F1-міри.

Для часового ряду AAPL (рис. 4.25) максимальне значення F1-міри становить приблизно 0.65 та досягається при пороговому значенні DISS близько 0.27. У діапазоні малих значень порогу ефективність класифікації залишається відносно стабільною, однак подальше збільшення порогу призводить до

поступового, а згодом і різкого зниження F1-міри. Це свідчить про втрату чутливості алгоритму до структурних змін при надмірному збільшенні допустимого порогу.

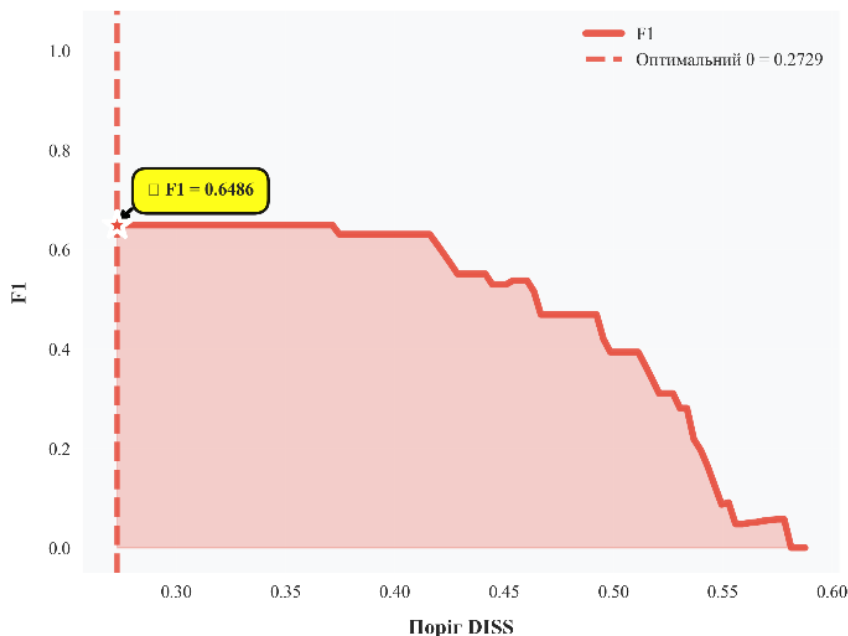


Рис. 4.25 – Локальні прирости середнього DISS-дрейфу для AAPL

Для TSLA (рис. 4.26) максимальне значення F1-міри також становить близько 0.65, проте досягається при значно меншому пороговому значенні 0.02. Зі зростанням порогу значення F1-міри зменшується майже монотонно, що вказує на високу чутливість результатів класифікації до вибору порогового значення для цього часового ряду.

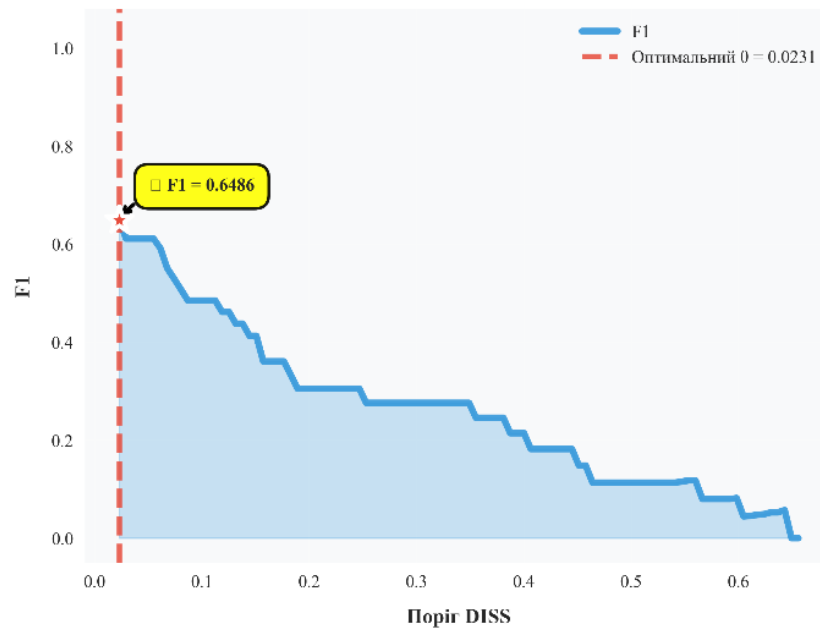


Рис. 4.26 – Локальні прирости середнього DISS-дрейфу для TSLA

Порівняльний аналіз показує, що хоча максимальні значення F1-міри для AAPL та TSLA є близькими, оптимальні порогові значення суттєво відрізняються. Це свідчить про відмінності у характері структурних змін у відповідних часових рядах та підкреслює необхідність індивідуального налаштування порогових значень для методу виявлення дрейфу структурних характеристик різних фінансових активів.

Отримані результати підтверджують, що вибір порогового значення DISS має суттєвий вплив на якість виявлення дрейфу, а універсальний поріг не забезпечує оптимальної ефективності для всіх наборів даних.

4.5 Ідентифікація ключових зв'язків у наборах даних

Побудуємо графи структурної подібності колонок даних для двох експериментальних наборів даних.

На рис. 4.27–4.28 наведено графи структурної подібності колонок даних, побудовані для наборів даних Yahoo з використанням порогового значення подібності не нижче 0.70. Вузли графів відповідають окремим колонкам даних, що представляють різні фінансові інструменти (AAPL – синій, MSFT – зелений,

NVDA – фіолетовий), ознаки та підвибірки, тоді як зважені ребра відображають ступінь їхньої структурної подібності, обчислений за відповідною метрикою.

Повний граф подібності містить 270 вузлів та характеризується кільцеподібним розташуванням із помітними локальними скупченнями вузлів одного кольору, що свідчить про тенденцію колонок одного фінансового інструменту до формування внутрішньо зв'язаних кластерів. Водночас наявність міжколіорових зв'язків вказує на структурну подібність окремих ознак між різними компаніями. Менш зв'язні області графа відображають структурно специфічні або більш варіативні ознаки, зокрема колонки Volume, які за статистичними характеристиками суттєво відрізняються від цінкових рядів.

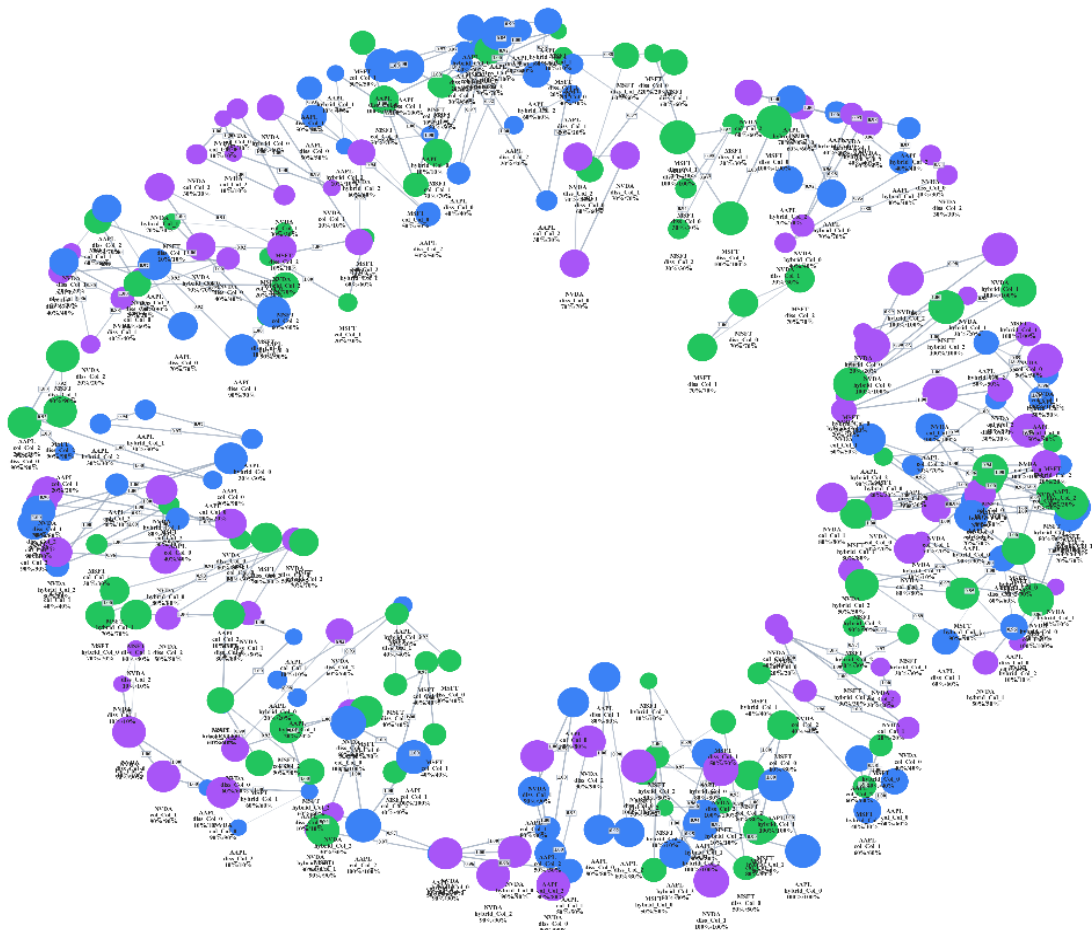


Рис. 4.27 – Повний граф структурної подібності колонок даних (Yahoo, поріг подібності ≥ 0.70 , 270 вузлів)

MST зберігає всі 270 вузлів графа, усуваючи надлишкові зв'язки та

залишаючи лише найбільш інформативні ребра (рис. 4.28). Отримана структура має виражений ієрархічно-радіальний характер із щільним центральним ядром, у якому переважають вузли всіх трьох компаній, що відповідають колонкам із найбільш типовими структурними властивостями, тоді як периферійні вузли відображають більш спеціалізовані або менш узгоджені характеристики.

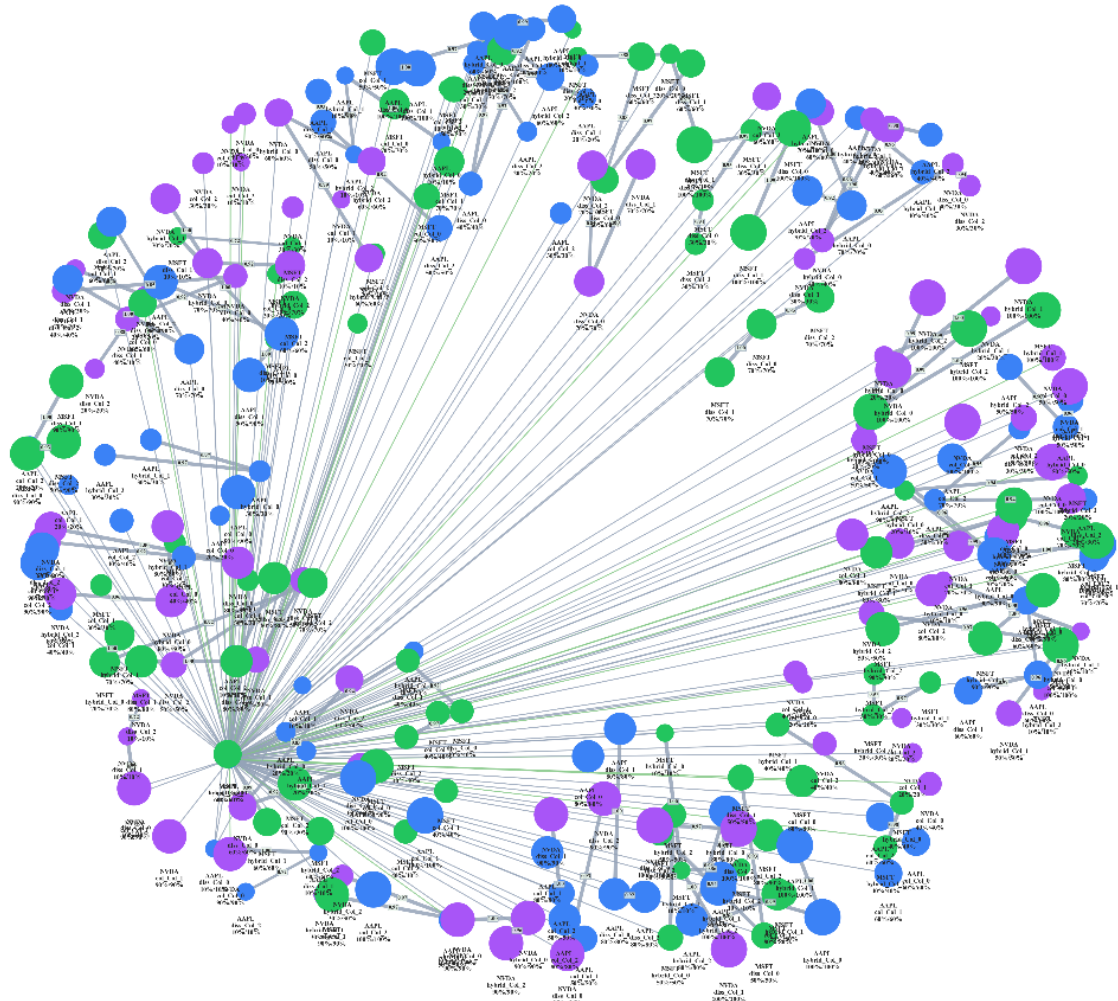


Рис. 4.28 – MST графа структурної подібності підвибірок (10%–100%) для часових рядів AAPL, NVDA та MSFT (Yahoo)

Результати демонструють, що графове представлення є ефективним інструментом для аналізу структурної подібності колонок даних у великомасштабних фінансових наборах. Поєднання повного графа та MST забезпечує як детальне, так і узагальнене уявлення про внутрішню організацію

даних, що є корисним для задач аналізу надлишковості ознак, відбору інформативних колонок та дослідження структурної узгодженості багатовимірних часових рядів.

Таким чином, графове представлення дозволяє виявити закономірності стабілізації структури фінансових часових рядів зі зростанням обсягу даних та підтверджує, що підвибірки, починаючи з певного порогу, наближаються за структурними властивостями до повного набору. Це демонструє ефективність використаного підходу для аналізу репрезентативності підвбірок та візуального дослідження зміни структурної подібності.

На рис. 4.29–4.30 наведено графи структурної подібності колонок даних AAPL та TSLA, побудовані на основі наборів даних Kaggle з використанням порогового значення подібності не нижче 0.70. Вузли графів відповідають окремим колонкам даних двох фінансових інструментів (AAPL – синій, TSLA – червоний), сформованим для різних підвбірок, тоді як зважені ребра відображають ступінь структурної подібності між ними, обчислений за відповідною метрикою.

Повний граф подібності містить 120 вузлів і характеризується розрідженою структурою з чітко відокремленими локальними групами, переважно утвореними вузлами одного кольору. Це свідчить про те, що колонки одного фінансового інструменту мають вищу внутрішню структурну узгодженість, ніж між різними інструментами. На відміну від аналогічного графа для набору Yahoo (рис. 4.27), де 270 вузлів формували щільне кільце зв'язків, граф Kaggle демонструє значно меншу щільність ребер і більшу кількість ізольованих або слабо зв'язних компонентів, що відображає структурну специфічність окремих колонок і відмінності у динаміці часових рядів AAPL і TSLA.

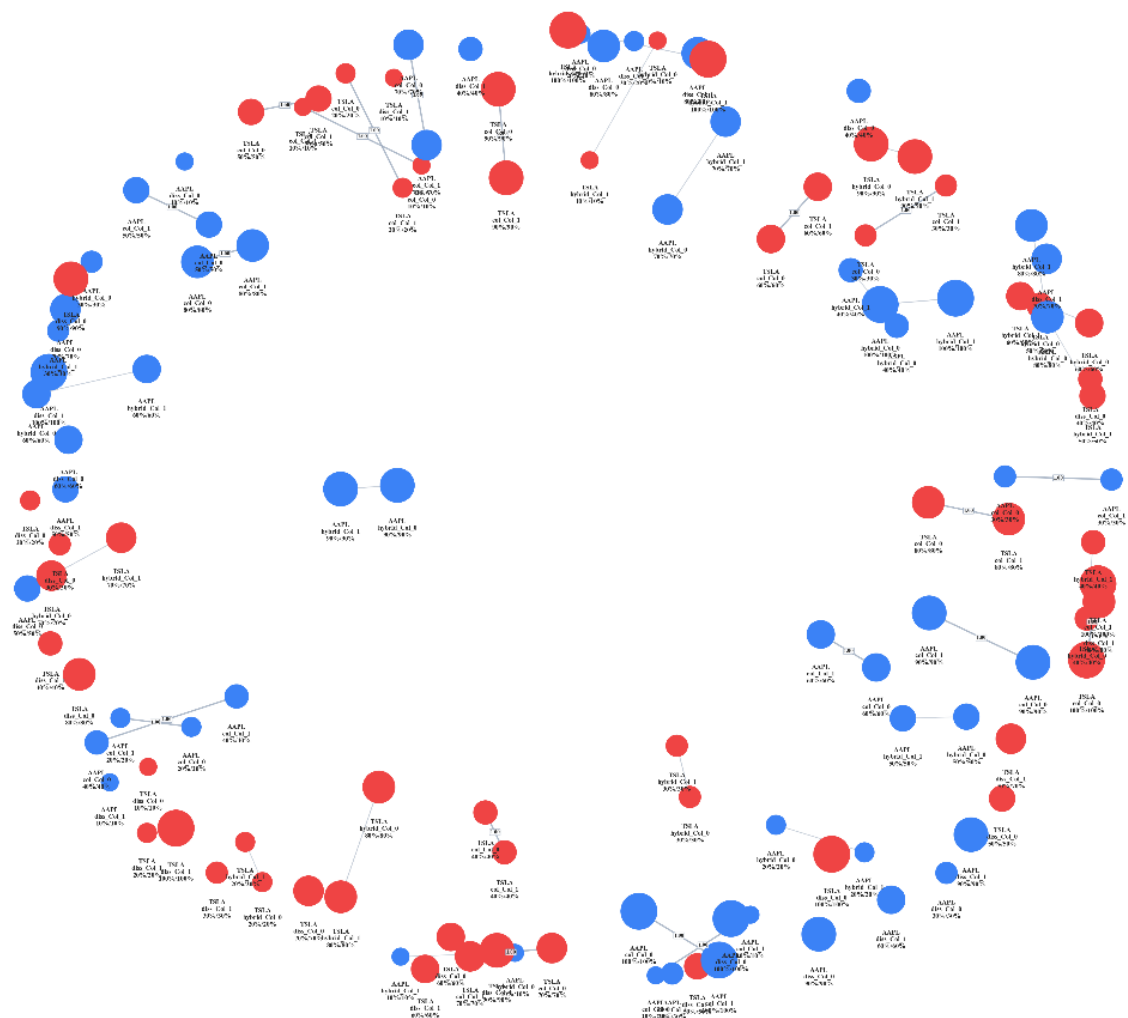


Рис. 4.29 – Повний граф структурної подібності підвибірок (10%–100%) для часових рядів AAPL та TSLA (Kaggle, поріг подібності ≥ 0.70)

MST зберігає всі 120 вузлів, усуваючи надлишкові зв'язки та залишаючи лише найбільш інформативних ребер. Отримана структура має радіально-ієрархічний характер із кількома центральними вузлами високого ступеня, що відповідають колонкам із найбільш типовими або стабільними структурними характеристиками. Периферійні вузли відображають більш специфічні або менш узгоджені ознаки. Зазначимо, що присутність у центральному ядрі MST вузлів обох компаній вказує на наявність спільних структурних шаблонів між AAPL та TSLA незважаючи на загальну тенденцію до групування в середині кожного фінансового інструменту.

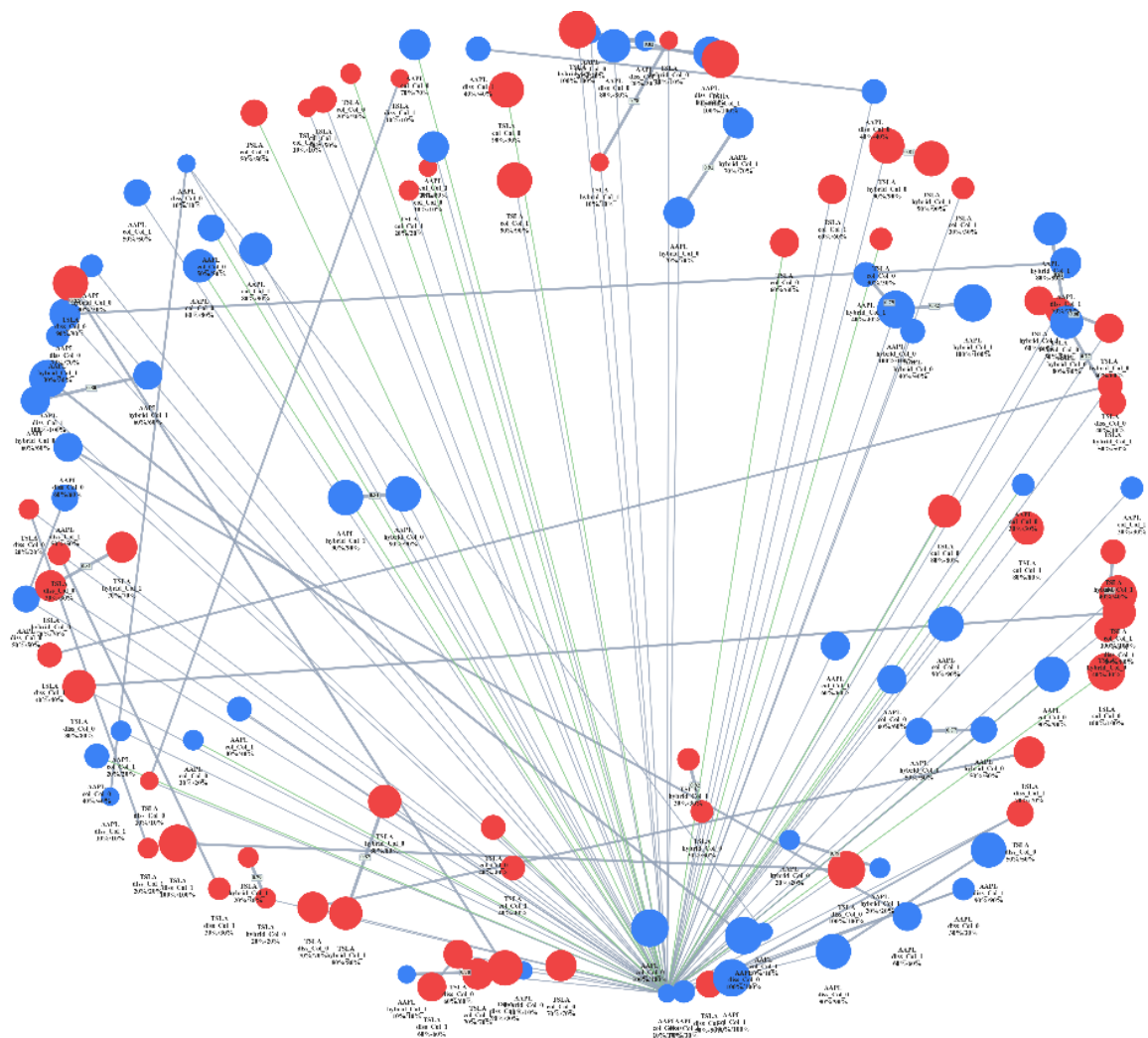


Рис. 4.30 – MST графа структурної подібності колонок даних (Kaggle, 120 вузлів)

Графове представлення колонок даних демонструє ефективність використаного підходу для аналізу структурної подібності на рівні окремих ознак. Поєднання повного графа та MST забезпечує як детальне, так і узагальнене уявлення про внутрішню організацію даних, що є корисним для задач відбору ознак, виявлення надлишковості та дослідження структурної узгодженості багатовимірних фінансових часових рядів.

Таким чином, графове представлення демонструє поступову стабілізацію структурних характеристик фінансових часових рядів AAPL та TSLA зі зростанням обсягу підвибірки та підтверджує, що починаючи з певного порогу,

підвибірки стають репрезентативними щодо повного набору даних. Це свідчить про доцільність використання запропонованого підходу для аналізу репрезентативності вибірок і візуального дослідження структурної еволюції фінансових часових рядів.

Отримані результати демонструють, що розроблена інформаційна система має потенціал до застосування в умовах високої варіативності даних, гарантує стабільне визначення змін і може бути інтегрована у практичні конвеєри моніторингу якості даних. При цьому застосування підходів до візуалізації графових структур у великих даних [232] дозволяє підвищити точність аналізу та зробити результати моніторингу більш наочними.

Висновки до розділу 4

У даному розділі виконано комплексну експериментальну перевірку розробленої інформаційної технології структурного аналізу гетерогенних табличних даних, що базується на методі компактного представлення (CDR) та метриці структурної подібності DISS. Отримані результати підтверджують коректність, стійкість і практичну ефективність запропонованого підходу при аналізі реальних фінансових часових рядів, сформованих із різних відкритих джерел даних.

Застосування методу компактного представлення забезпечило уніфіковане формалізоване подання змінних різних типів – числових, часових, категоріальних і текстових – у вигляді структурованих дескрипторів, придатних для подальшого порівняння. Це дозволило виконувати зіставлення як окремих ознак, так і наборів даних у цілому незалежно від їхньої внутрішньої гетерогенності та джерела походження. Також встановлено, що використання компактних представлень забезпечує надзвичайно високий рівень зменшення обсягу даних без втрати статистично значущої інформації, що є критично

важливим для масштабованої обробки даних у Big Data-середовищах.

Експериментальне тестування на двох еталонних наборах даних Yahoo Finance (AAPL, NVDA, MSFT) та Kaggle (AAPL, TSLA) продемонструвало універсальність запропонованого підходу та його стійкість до змін джерела даних, часових інтервалів і ринкових режимів. Аналіз чутливості компактних представлень показав, що зі збільшенням обсягу підвибірки структурна відстань DISS зменшується та збігається до нульового значення, що відповідає повній структурній відповідності з базовим представленням.

Встановлено, що швидкість конвергенції значень DISS суттєво залежить від властивостей конкретного фінансового часового ряду. Зокрема, для набору MSFT спостерігається швидка стабілізація структурних характеристик уже при використанні 40% вибірки, тоді як для AAPL аналогічний рівень досягається при 50–60%, а для NVDA та TSLA – тільки при 60–70% даних. Це свідчить про різний рівень структурної складності та волатильності активів, що досліджувалися, і підтверджує необхідність індивідуального підходу до інтерпретації результатів структурного аналізу.

Графове представлення структурної подібності у вигляді повних графів і MST дозволило дослідити внутрішню організацію даних та еволюцію їхніх структурних зв'язків. Показано, що зі збільшенням обсягу підвибірки підграфи стабілізуються та наближаються до структури повного набору даних, а MST-графи ефективно виділяють найбільш інформативні зв'язки між ознаками та підвибірками. Такий підхід є особливо корисним для виявлення надлишкових ознак, аналізу структурної узгодженості та дослідження складних багатовимірних залежностей.

ROC-аналіз якості виявлення структурного дрейфу підтвердив високу ефективність запропонованої технології у задачах бінарної класифікації. Для розглянутих часових рядів значення площі під ROC-кривою (AUC) перевищують 0,98, а досягнуті значення F1-міри становлять близько 0,65, що

свідчить про збалансованість між точністю та повнотою класифікації. Отримані значення точності для всіх активів перевищують 0.94: AAPL – 0.9676, TSLA – 0.9588, NVDA – 0.9412, MSFT – 0.9956. Результати підтверджують високу точність ідентифікації часових сегментів класифікатором, що свідчить як про суттєву статистичну відмінність між сегментами, так і про ефективність застосованого підходу до виявлення дрейфу даних.

Отримані результати для наборів [227, 228] також свідчать, що типи колонок у наборах було визначено коректно та подальший аналіз чутливості, збіжності DISS, підбір порогів за F1 та ROC-криві демонструють узгоджену поведінку для різнотипних ознак без явних ознак помилкової типізації. Отже, застосований підхід адекватно працює як для більш текстово-орієнтованого третього набору Sentiment [227, 228], так і для структуровано-транзакційного набору OnlineRetail [227, 228].

Результати четвертого розділу підтверджують, що розроблена інформаційна технологія забезпечує стійку ідентифікацію структурних змін у гетерогенних табличних даних, є масштабованою, інтерпретованою та придатною для інтеграції в автоматизовані конвеєри моніторингу якості даних. Запропонований спосіб представлення CDR-дескрипторів забезпечує сумісність інформаційної технології із хмарними платформами та її використання у сучасних системах Big Data і промислових аналітичних інфраструктурах.

Основні наукові результати четвертого розділу опубліковані в [70, 165, 187, 232].

ВИСНОВКИ

У дисертаційному дослідженні сформовано новий підхід до стиснення великих і гетерогенних даних, що поєднує теоретичні результати спектрального аналізу випадкових матриць із практичними механізмами компактного подання даних та їх порівняння.

У роботі поставлено і розв'язано актуальну науково-технічну задачу побудови теоретичних засад та створення прикладної інформаційної технології для визначення кластерної структури у великих гетерогенних даних на основі спектрального аналізу стохастичних матриць та компактних представлень табличних наборів даних. Розроблені теоретичні результати, математичні моделі та програмні компоненти утворюють цілісну систему, що забезпечує формальний аналіз структурних властивостей даних, їх порівняння, кластеризацію та виявлення дрейфів у масштабованому обчислювальному середовищі. Отримані результати мають як фундаментальне значення для теорії випадкових матриць і спектральної кластеризації, так і прикладне значення в інженерії даних, машинному навчанні та аналітичних системах обробки великих даних.

Основні результати дисертаційної роботи:

1. Проведено ґрунтовний аналітичний огляд сучасних підходів до кластеризації великих графових структур та ролі випадкових матриць у моделюванні складних мереж. Показано обмеженість класичних підходів у випадку гетерогенних графів із незалежними та мінливими вагами, а також обґрунтовано необхідність побудови нових спектральних критеріїв для визначення кількості кластерів.

2. Набуло подальшого розвитку побудова та дослідження стохастичної моделі графа з незалежними, однаково розподіленими, строго додатними вагами та виконано повний спектральний аналіз відповідної стохастичної матриці. На основі узагальнення теореми Перрона–Фробеніуса та властивостей додатних матриць отримано теоретичні результати щодо розміщення власних значень та умов формування домінуючої частини

спектра.

3. Отримано нові умови, що пов'язують властивості випадкових ваг та поведінку спектра стохастичної матриці з кількістю кластерів у графі. Обґрунтовано критерій оптимальної кількості кластерів

$$k_{opt} = \#\left\{\lambda_i(P): |\lambda_i(P)| > \frac{1}{\alpha\sqrt{N}}\right\}.$$

4. Запропоновано новий підхід побудови компактного представлення даних Compact Data Representation (CDR), яке забезпечує універсальне стисле представлення гетерогенних табличних даних. Запропоновано математичні моделі побудови компактних дескрипторів для числових, категоріальних, текстових і часових змінних.

5. Запропоновано та реалізовано систему метрик структурної подібності DISS, яка дозволяє порівнювати набори даних на рівні їх компактних представлень. До складу системи включено відстані Вассерштейна, Геллінгера, повної варіації, косинусну метрику та спеціальні метрики для текстових та часових змінних. Показано, що використання DISS забезпечує високу чутливість до змін форми розподілу, розподілів з важкими хвостами та залежностей між змінними.

6. Модифіковано підхід до кластеризації табличних даних на основі зваженого графа метрик DISS. Це дозволило поєднати спектральні властивості випадкових матриць, описані у перших розділах роботи, з практичними алгоритмами кластеризації на компактних дескрипторах.

7. Створено повноцінну інформаційну систему для побудови CDR, обчислення DISS, аналізу дрейфу та кластеризації даних. Архітектуру системи побудовано на модульних компонентах: TypeInferencer, CSVAnalyzer, CompactRepresentationComparator, DISS Engine, DriftAnalyzer, ClusterBuilder, CloudComputeLayer. Система підтримує обробку локальних і великих хмарних даних, забезпечує модульність, масштабованість та гнучкість.

8. Розроблено хмарну модель розгортання інформаційної системи

на основі Apache Spark та AWS EMR/EMR Serverless. Показано, що хмарна реалізація забезпечує автоматичне масштабування, зниження витрат на обчислення та ефективність обробки великих наборів даних.

9. Проведено повний комплекс експериментальних досліджень на реальних наборах даних. Виконано прикладний аналіз структурного дрейфу даних та порівняння великих наборів без доступу до повної вибірки. Отримані результати демонструють, що система CDR/DISS дозволяє точно виявляти зміни в структурі даних навіть при значному зменшенні обсягів інформації.

10. Результати дисертаційної роботи впроваджені у ТОВ «Кодерс ПРО», ТОВ «Палетний сервіс» та використовуються в освітньому процесі відділу комп'ютерних технологій Навчально-наукового інституту фізико-технічних та комп'ютерних наук Чернівецького національного університету імені Юрія Федьковича.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Combining structured and unstructured data for predictive models: a deep learning approach / D. Zhang et al. *BMC Medical Informatics and Decision Making*. 2020. Vol. 20, № 1. Art. 280. URL: <https://doi.org/10.1186/s12911-020-01297-1>.
2. Errasti J., Amigo I., Villadangos M. Emotional Uses of Facebook and Twitter: Its Relation With Empathy, Narcissism, and Self-Esteem in Adolescence. *Psychological Reports*. 2017. Vol. 120, № 6. P. 997–1018. URL: <https://doi.org/10.1177/0033294117713496>.
3. Compact and Discriminative Representation of Bag-of-Features / J. Cui et al. *Neurocomputing*. 2015. Vol. 169. P. 389–398. URL: <https://doi.org/10.1016/j.neucom.2014.09.104>.
4. Шаховська Н., Загородний І. Метод пошуку залежностей у великих наборах даних із пропусками на прикладі даних про поширення COVID. *Вісник Хмельницького національного університету. Серія: Технічні науки*. 2024. Т. 333, № 2. С. 11–16. URL: <https://doi.org/10.31891/2307-5732-2024-333-2-1>.
5. Shakhovska N. B. Explainable AI and robust forecasting of global salary trends: Addressing data drift and unseen categories with tree-based models. *Mathematical Modeling and Computing*. 2025. Vol. 12, № 3. P. 993–1004. URL: <https://doi.org/10.23939/mmc2025.03.993>.
6. Goswami M. J. Leveraging AI for Cost Efficiency and Optimized Cloud Resource Management. *International Journal of New Media Studies*. 2020. Vol. 7, № 1. P. 21–27. URL: <https://ijnms.com/index.php/ijnms/article/view/250> (дата звернення: 01.12.2025).
7. Application of Solar Activity Time Series in Machine Learning Predictive Modeling of Precipitation-Induced Floods / S. Malinović-Milićević et al. *Mathematics*. 2023. Vol. 11, № 4. Art. 795. URL: <https://doi.org/10.3390/math11040795>.

8. Applying machine learning in the investigation of the link between the high-velocity streams of charged solar particles and precipitation-induced floods / S. Malinović-Milićević et al. *Environmental Monitoring and Assessment*. 2024. Vol. 196. Art. 400. URL: <https://doi.org/10.1007/s10661-024-12537-x>.
9. Hunt L., Jorgensen M. Mixture model clustering for mixed data with missing information. *Computational Statistics & Data Analysis*. 2003. Vol. 41, № 3–4. P. 429–440. URL: [https://doi.org/10.1016/S0167-9473\(02\)00190-1](https://doi.org/10.1016/S0167-9473(02)00190-1).
10. Ahmad A., Khan S. S. Survey of state-of-the-art mixed data clustering algorithms. *IEEE Access*. 2019. Vol. 7. P. 31883–31902. URL: <https://doi.org/10.1109/ACCESS.2019.2903568>.
11. Guerard G., Djebali S. Mixed Data Clustering Survey and Challenges. *SN Computer Science*. 2025. Vol. 6, Issue 8. Art. 939. URL: <https://doi.org/10.1007/s42979-025-04439-7>.
12. Bianconi G. Multilayer Networks: Structure and Function. Oxford : Oxford University Press, 2018. 402 p. URL: <https://doi.org/10.1093/oso/9780198753919.001.0001>.
13. Holme P., Saramäki J. Temporal Network Theory. Springer Cham, 2019. 375 p. URL: <https://doi.org/10.1007/978-3-030-23495-9>.
14. Ghahramani Z. Probabilistic machine learning and artificial intelligence. *Nature*. 2015. Vol. 521. P. 452–459. URL: <https://doi.org/10.1038/nature14541>.
15. An extensive comparative study of cluster validity indices / O. Arbelaiz et al. *Pattern Recognition*. 2013. Vol. 46, № 1. P. 243–256. URL: <https://doi.org/10.1016/j.patcog.2012.07.021>.
16. von Luxburg U. A tutorial on spectral clustering. *Statistics and Computing*. 2007. Vol. 17, № 4. P. 395–416. URL: <https://doi.org/10.1007/s11222-007-9033-z>.
17. Graph signal processing: Overview, challenges, and applications / Ortega A. et al. *Proceedings of the IEEE*. 2018. Vol. 106, № 5. P. 808–828. URL: <https://doi.org/10.1109/JPROC.2018.2820126>.

18. Fortunato S., Hric D. Community detection in networks: A user guide. *Physics Reports*. 2016. Vol. 659. P. 1–44. URL: <https://doi.org/10.1016/j.physrep.2016.09.002>.
19. Kipf T., Welling M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907v4*. 2016. URL: <https://doi.org/10.48550/arXiv.1609.02907>.
20. Perozzi B., Al-Rfou R., Skiena S. DeepWalk: online learning of social representations. *Knowledge discovery and data mining (KDD '14)* : proceedings of the 20th ACM SIGKDD International Conference, August 24–27, 2014. New York, NY, USA : Association for Computing Machinery, 2014. P. 701–710. URL: <https://doi.org/10.1145/2623330.2623732>.
21. Grover A., Leskovec J. Node2vec: Scalable Feature Learning for Networks. *Knowledge Discovery and Data Mining (KDD '16)* : proceedings of the 22nd ACM SIGKDD International Conference, August 13–17, 2016. New York, NY, USA : Association for Computing Machinery, 2016. P. 855–864. URL: <https://doi.org/10.1145/2939672.2939754>.
22. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains / D. I. Shuman et al. *IEEE Signal Processing Magazine*. 2013. Vol. 30, № 3. P. 83–98. URL: <https://doi.org/10.1109/MSP.2012.2235192>.
23. A Comprehensive Survey on Graph Neural Networks / Z. Wu et al. *IEEE Transactions on Neural Networks and Learning Systems*. 2021. Vol. 32, № 1. P. 4–24. URL: <https://doi.org/10.1109/TNNLS.2020.2978386>.
24. Clauset A., Shalizi C. R., Newman M. E. J. Power-law distributions in empirical data. *SIAM Review*. 2009. Vol. 51, Issue 4. P. 661–703. URL: <https://doi.org/10.1137/070710111>.
25. Barabási A.-L. Network Science. Cambridge : Cambridge University Press, 2016. 475 p.
26. Гнєденко Б. В. Курс теорії ймовірностей. Київ : ВПЦ «Київський університет», 2010. 464 с.

27. Multifractal phases in weighted Erdős–Rényi random graphs / L. F. Cugliandolo et al. *Journal of Physics Review B*. 2024. Vol. 110, Issue 17. P. 174202, 24 pp. URL: <https://doi.org/10.1103/PhysRevB.110.174202>.
28. Tao T. Topics in Random Matrix Theory. American Mathematical Society, 2012. 282 p. URL: <https://doi.org/10.1090/gsm/132>.
29. Vershynin R. High-Dimensional Probability: An Introduction with Applications in Data Science. Cambridge : Cambridge University Press, 2018. 264 p. URL: <https://doi.org/10.1017/9781108231596>.
30. Bordenave C. Lecture notes on random matrix theory. *arXiv preprint* arXiv:2009.05157v1. 2020. URL: <https://doi.org/10.48550/arXiv.2009.05157>.
31. Кириченко О., Кириченко Є. Асимптотичні властивості випадкових матриць. *Вісник Київського національного університету імені Тараса Шевченка. Фізико-математичні науки*. 2024. Т. 78, №1. С. 41–44. URL: <https://doi.org/10.17721/1812-5409.2024/1.7>.
32. Кириченко Є., Малик І. Один результат кластеризації на основі випадкових матриць. *Information Technology: Computer Science, Software Engineering and Cyber Security*. 2025. №3. С. 257-270. URL: <https://doi.org/10.32782/IT/2025-3-29>.
33. Кириченко Є. О., Малик І. В. Використання подвійних стохастичних моделей в BIG DATA. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2023)* : праці XII Міжнародної науково-практичної конференції, 10–12 листопада 2023 р. Чернівці : Черн. нац. ун-т, 2023. С. 33–35. URL: <https://drive.google.com/file/d/10m9pfYczudmIanv0na1MgZqMSoGyFPA>.
34. Кириченко Є. О., Малик І. В. Моделі і процеси при дослідженні подвійних стохастичних систем. *Інформаційні технології: наука, техніка, технологія, освіта, здоров'я* : тези доп. XXXII Міжнар. наук.-практ. конф. MicroCAD-2024, 22–25 травня 2024 р. / за ред. проф. Є. І. Сокола. Харків : НТУ «ХПІ», 2024. С. 1294. URL: <https://ndch.kpi.kharkov.ua/wp->

content/uploads/2025/01/Zbirnik-tez-MicroCAD-2024.pdf (дата звернення: 02.12.25).

35. van Vuren T., Cronk T., Sanders J. Estimating the number of clusters of a Block Markov Chain. *arXiv preprint*. arXiv:2407.18287v1, 2024. URL: <https://doi.org/10.48550/arXiv.2407.18287>.

36. Cape J., Yu X., Liao J. Z. Robust spectral clustering with rank statistics. *Journal of Machine Learning Research*. 2024. Vol. 25, Issue 1. Art. 398. P. 19451 – 19531. URL: <https://dl.acm.org/doi/10.5555/3722577.3722975>.

37. Srivastava P. R., Sarkar P., Hanasusanto G. A. A robust spectral clustering algorithm for sub-Gaussian mixture models with outliers. *Operations Research*. 2023. Vol. 71, № 1. P. 224–244. URL: <https://doi.org/10.1287/opre.2022.2317>.

38. Tastan A. Contributions to robust graph clustering: spectral methods and beyond : Doctoral dissertation / Technische Universität Darmstadt, Darmstadt, 2023. 283 p. URL: <https://doi.org/10.26083/tuprints-00024068>.

39. Newman M. E. J. Networks: An Introduction. Oxford : Oxford University Press, 2018. 789 p.

40. Easley D., Kleinberg J. Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge: Cambridge University Press, 2010. 819 p.

41. Privault N. Understanding Markov Chains. Springer Singapore, 2018. 354 p. URL: <https://doi.org/10.1007/978-981-4451-51-2>.

42. Bapat R. B., Raghavan T. E. S. Nonnegative matrices and applications. Cambridge : Cambridge University Press, 1997. 356 p. URL: <https://doi.org/10.1017/CBO9780511529979>.

43. Кириченко О. Л., Малик І. В., Остапов С. Е. Стохастичні моделі в задачах штучного інтелекту. Вісник Київського національного університету імені Тараса Шевченка. Серія фізико-математичні науки. 2021. №2. С. 53–57. URL: <https://doi.org/10.17721/1812-5409.2021/2.7>.

44. Kullback–Leibler Divergence Metric Learning / S. Jiet al. *IEEE Transactions*

on Cybernetics. Vol. 52, № 4. P. 2047–2058. URL: <https://doi.org/10.1109/TCYB.2020.3008248>.

45. Sharma A., Kundu C. A measure of dispersion based on Kullback–Leibler divergence in residual and past lifetimes. *Journal of Statistical Computation and Simulation*. 2025. P. 1–24. URL: <https://doi.org/10.1080/00949655.2025.2598421>.

46. Basu A., Shioya H., Park, C. Statistical Inference: The Minimum Distance Approach. 1st ed. New York : Chapman and Hall/CRC, 2011. 429 p. URL: <https://doi.org/10.1201/b10956>.

47. Ross S. Simulation. 5th ed. San Diego : Academic Press, 2013. 311 p. URL: <https://doi.org/10.1016/B978-0-12-415825-2.00014-0>.

48. Gil A., Segura J., Temme N. M. Numerical Methods for Special Functions. Society for Industrial and Applied Mathematics (SIAM), 2007. 417 p. URL: <https://doi.org/10.1137/1.9780898717822>.

49. Origins of power-law degree distribution in the heterogeneity of human activity in social networks / L. Muchnik et al. *Scientific Reports*. 2013. Vol. 3. Art. 1783. URL: <https://doi.org/10.1038/srep01783>.

50. Classical information theory of networks / Radicchi F. et al. *Journal of Physics: Complexity*. 2020. Vol. 1, № 2. P. 025001 (12pp). URL: <https://doi.org/10.1088/2632-072X/ab9447>.

51. Bordenave C., Caputo P., Chafaï D. Circular law theorem for random Markov matrices. *Probability Theory and Related Fields*. 2012. Vol. 152. P. 751–779. URL: <https://doi.org/10.1007/s00440-010-0336-1>.

52. A Hybrid Spectral Clustering and Deep Neural Network Ensemble Algorithm for Intrusion Detection in Sensor Networks / T. Ma et al. *Sensors*. 2016. Vol. 16, № 10. Art. 1701. URL: <https://doi.org/10.3390/s16101701>.

53. Power System Zone Partitioning Based on Transmission Congestion Identification Using an Improved Spectral Clustering Algorithm / Y. Hu et al. *Electronics*. 2021. Vol. 10, № 17. Art. 2126. URL: <https://doi.org/10.3390/electronics10172126>.

54. Lin J., Liu M. The search method for key transmission sections based on an improved spectral clustering algorithm. *Frontiers in Energy Research*. 2024. Vol. 12. Art. 1387828. URL: <https://doi.org/10.3389/fenrg.2024.1387828>.
55. Mattera R., Scepi G., Kaur P. Time series clustering for high-dimensional portfolio selection: a comparative study. *Soft Computing*. 2025. Vol. 29. P. 4219–4231. URL: <https://doi.org/10.1007/s00500-025-10656-2>.
56. Zhao W., Roncoli C. A constrained spectral clustering method for lane identification using trajectory data. *Transportation Research Part C: Emerging Technologies*. 2023. Vol. 155. Art. 104270. URL: <https://doi.org/10.1016/j.trc.2023.104270>.
57. Clustering analysis of tumor metabolic networks / I. Manipur et al. *BMC Bioinformatics*. 2020. Vol. 21. Art. 349. URL: <https://doi.org/10.1186/s12859-020-03564-9>.
58. An improved spectral clustering method for accurate detection of brain resting-state networks / J. Barrett et al. *NeuroImage*. 2024. Vol. 299. Art. 120811. URL: <https://doi.org/10.1016/j.neuroimage.2024.120811>.
59. You K., Lee Y., Park H.-J. Scalable geometric learning with correlation-based functional brain networks. *Scientific Reports*. 2025. Vol. 15. Art. 22685. URL: <https://doi.org/10.1038/s41598-025-07703-1>.
60. Load balancing for cloud computing using optimized cluster based federated learning / K. K. Chennam et al. *Scientific Reports*. 2025. Vol. 15. Art. 41328. URL: <https://doi.org/10.1038/s41598-025-25220-z>.
61. A method to recommend cloud manufacturing service based on the spectral clustering and improved Slope one algorithm / Q. Hu et al. *Journal of Cloud Computing*. 2023. Vol. 12. Art. 112. URL: <https://doi.org/10.1186/s13677-023-00489-5>.
62. Meyer C. D. Matrix Analysis and Applied Linear Algebra. Philadelphia : SIAM, 2000. 899 p.
63. Seneta E. Non-negative Matrices and Markov Chains. New York :

Springer New York, 2006. 281 p. URL: <https://doi.org/10.1007/0-387-32792-4>.

64. Rahm E., Do H-H. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*. 2000. Vol. 23, Issue 4. P. 3–13.

65. Kherdekar V., Metkewar P. A Technical Comprehensive Survey of ETL Tools. *International Journal of Applied Engineering Research*. 2016. Vol. 11, №4. P. 2557-2559. URL: <https://doi.org/10.37622/IJAER/11.4.2016.2557-2559>.

66. Ridzuan F., Wan Zainon W. M. N. A Review on Data Cleansing Methods for Big Data. *Procedia Computer Science*. 2019. Vol. 161. P. 731-738. URL: <https://doi.org/10.1016/j.procs.2019.11.177>.

67. Sahay T., Mehta A., Jadon S. Schema Matching using Machine Learning. *Signal Processing and Integrated Networks (SPIN)* : proceedings of the 7th International Conference, 27–28 February, 2020. Noida, India. 2020. P. 359-366.

68. Doan AH., Halevy A., Ives Z. Principles of Data Integration. 1st ed. Morgan Kaufmann, 2012. 520 p.

69. Emmert-Streib F., Moutari S., Dehmer M. Dimension Reduction. In: Elements of Data Science, Machine Learning, and Artificial Intelligence Using R. Cham : Springer, 2023. P. 163–189. URL: https://doi.org/10.1007/978-3-031-13339-8_8.

70. Malyk I. V., Kyrychenko Y., Gorbatenko M., Lukashiv T. Data Optimization through Compression Methods Using Information Technology. *International Journal of Information Technology and Computer Science*. 2025. Vol. 17, №5. P. 84–99. URL: <https://doi.org/10.5815/ijitcs.2025.05.07>.

71. Kannan R., Bayraksan G., Luedtke J. R. Technical Note–Data-Driven Sample Average Approximation with Covariate Information. *Operations Research*. 2025. Vol. 73, № 6. P. 3245–3259. URL: <https://doi.org/10.1287/opre.2020.0533>.

72. Siddharth C. R., Vishwadeepak S. B. Enhancing Cloud Migration Efficiency with Automated Data Pipelines and AI-Driven Insights. *International Journal of Innovative Science and Research Technology (IJISRT)*. 2024. Vol. 9,

Issue 11. P. 3670–3690. URL: <https://doi.org/10.5281/zenodo.14836684>.

73. Sarmah U., Borah P., Bhattacharyya D. K. Ensemble Learning Methods: An Empirical Study. *SN Computer Science*. 2024. Vol. 5, Art. 924. URL: <https://doi.org/10.1007/s42979-024-03252-y>.

74. DANets: Deep Abstract Networks for Tabular Data Classification and Regression / J. Chen et al. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2022. Vol. 36, №4. P. 3930-3938. URL: <https://doi.org/10.1609/aaai.v36i4.20309>.

75. Converting tabular data into images for deep learning with convolutional neural networks / Y. Zhu et al. *Scientific Reports*. 2021. Vol. 11. Art. 11325. URL: <https://doi.org/10.1038/s41598-021-90923-y>.

76. SwitchTab: Switched Autoencoders Are Effective Tabular Learners / J. Wu et al. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2024. Vol. 38, № 14. P. 15924–15933. URL: <https://doi.org/10.1609/aaai.v38i14.29523>.

77. Types of Variables in Research & Statistics. *Scribbr* : website. URL: <https://www.scribbr.com/methodology/types-of-variables/> (дата звернення: 15.12.2025).

78. Dettori J. R., Norvell D. C. The Anatomy of Data. *Global Spine Journal*. 2018. Vol. 8, Issue 3. P. 311–313. URL: <https://doi.org/10.1177/2192568217746998>.

79. Zhu W., Qiu R., Fu Y. Comparative Study on the Performance of Categorical Variable Encoders in Classification and Regression Tasks. *arXiv preprint* arXiv:2401.09682. 2024. URL: <https://doi.org/10.48550/arXiv.2401.09682>.

80. Hancock J. T., Khoshgoftaar T. M. Survey on categorical data for neural networks. *Journal of Big Data*. 2020. Vol. 7, Art. 28. URL: <https://doi.org/10.1186/s40537-020-00305-w>.

81. Wang E., Cook D., Hyndman R. J. A New Tidy Data Structure to Support Exploration and Modeling of Temporal Data. *Journal of Computational and Graphical Statistics*. 2020. Vol. 29, Issue 3. P. 466–478. URL:

<https://doi.org/10.1080/10618600.2019.1695624>.

82. Abrar S., Samad M. D. Perturbation of deep autoencoder weights for model compression and classification of tabular data. *Neural Networks*. 2022. Vol. 156. P. 160–169. URL: <https://doi.org/10.1016/j.neunet.2022.09.020>.

83. A data representation method using distance correlation / X. Liang et al. *Frontiers Computer Science*. 2025. Vol. 19. Art. 191303. URL: <https://doi.org/10.1007/s11704-023-3396-y>.

84. A Compact Representation of Measured BRDFs Using Neural Processes / C. Zheng et al. *ACM Transaction on Graphics (TOG)*. 2021. Vol. 41, Issue 2. Art. 14. 15 p. URL: <https://doi.org/10.1145/3490385>.

85. CLaRe: Compact near-lossless Latent Representations of High-Dimensional Object Data / E. Zohner et al. *arXiv preprint arXiv:2502.07084v1*. 2025. URL: <https://doi.org/10.48550/arXiv.2502.07084>.

86. Compact geometric representation of qualitative directional knowledge / Z. Long et al. *Knowledge-Based Systems*. 2020. Vol. 195. Art. 105616. URL: <https://doi.org/10.1016/j.knosys.2020.105616>.

87. Model comprehension for security risk assessment: an empirical comparison of tabular vs. graphical representations / K. Labunets et al. *Empirical Software Engineering*. 2017. Vol. 22. P. 3017–3056. URL: <https://doi.org/10.1007/s10664-017-9502-8>.

88. Bordenave C., Collins B. Strong asymptotic freeness for independent uniform variables on compact groups associated to nontrivial representations. *Inventiones mathematicae*. 2024. Vol. 237. P. 221–273. URL: <https://doi.org/10.1007/s00222-024-01259-z>.

89. Compact and efficient representation of general graph databases / S. Álvarez-García et al. *Knowledge and Information Systems*. 2019. Vol. 60. P. 1479–1510. URL: <https://doi.org/10.1007/s10115-018-1275-x>.

90. A Method for Discovering Novel Classes in Tabular Data / C. Troisemaine et al. *2022 IEEE International Conference on Knowledge Graph (ICKG)* : proceedings of International Conference, Nov. 30–Dec. 1, 2022. Orlando,

FL, USA. 2022. P. 265–274.

URL: <https://doi.org/10.1109/ICKG55886.2022.00041>.

91. Kim S-K. Compact Data Learning for Machine Learning Classifications. *Axioms*. 2024. Vol. 13, Issue 3. Art. 137. URL: <https://doi.org/10.3390/axioms13030137>.

92. CTAB-GAN+: enhancing tabular data synthesis / Z. Zhao et al. *Front. Big Data*. 2024. Vol. 6. Art. 1296508. URL: <https://doi.org/10.3389/fdata.2023.1296508>.

93. Machine Learning with High-Cardinality Categorical Features in Actuarial Applications / B. Avanzi et al. *ASTIN Bulletin: The Journal of the IAA*. 2024. Vol. 54, Issue 2. P. 213–238. URL: <https://doi.org/10.1017/asb.2024.7>.

94. Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features / F. Pargent et al. *Computational Statistics*. 2022. Vol. 37. № 5. P. 2671–2692. URL: <https://doi.org/10.1007/s00180-022-01207-6>.

95. Richman R., Wüthrich M. V. High-cardinality categorical covariates in network regressions. *Japanese Journal of Statistics and Data Science*. 2024. Vol. 7. P. 921–965. URL: <https://doi.org/10.1007/s42081-024-00243-4>.

96. McKinney W. Pandas: A Foundational Python Library for Data Analysis and Statistics. *Python for High Performance and Scientific Computing*. 2011. Vol. 14. P. 1–9.

97. McNamara A., Horton N. J. Wrangling Categorical Data in R. *The American Statistician*. 2018. Vol. 72, Issue 1: Special Issue on Data Science. P. 97–104. URL: <https://doi.org/10.1080/00031305.2017.1356375>.

98. Contextual semantic type detection in tables / D. Zhanget al. *Proceedings of the VLDB Endowment*. 2020. Vol. 13, Issue 12. P. 1835–1848. URL: <https://doi.org/10.14778/3407790.3407793>.

99. Cerda P., Varoquaux G., Kégl B. Similarity encoding for learning with dirty categorical variables. *Machine Learning*. 2018. Vol. 107. P. 1477–1494. URL: <https://doi.org/10.1007/s10994-018-5724-2>.

100. Cerda P., Varoquaux G. Encoding high-cardinality string categorical variables. *IEEE Transactions on Knowledge and Data Engineering*. 2022. Vol. 34. Issue 3. P. 1164–1176. URL: <https://doi.org/10.1109/TKDE.2020.2992529>.
101. A Memory-Efficient Encoding Method for Processing Mixed-Type Data on Machine Learning / I. Lopez-Arevalo et al. *Entropy*. 2020. Vol. 22, Issue 12. Art. 1391. URL: <https://doi.org/10.3390/e22121391>.
102. Effective Methods of Categorical Data Encoding for Artificial Intelligence Algorithms / F. Bolikulov et al. *Mathematics*. 2024. Vol. 12, Issue 16. Art. 2553. URL: <https://doi.org/10.3390/math12162553>.
103. Tavakol M., Wetzel A. Factor Analysis: a means for theory and instrument development in support of construct validity. *International Journal of Medical Education*. 2020. Vol. 11. P. 245–247. URL: <https://doi.org/10.5116/ijme.5f96.0f4a>.
104. Agresti A. Analysis of Ordinal Categorical Data. 2nd ed. Hoboken : Wiley, 2010. 424 p. URL: <https://doi.org/10.1002/9780470594001>.
105. Ordinal Data: Definition, Examples, Data Collection & Analysis. *Scribbr* : website. URL: <https://www.scribbr.com/statistics/ordinal-data/> (дата звернення: 14.12.2025).
106. Atluri G., Karpatne A., Kumar V. Spatio-Temporal Data Mining: A Survey of Problems and Methods. *ACM Computing Surveys*. 2018. Vol. 51, Issue 4. Art. 83. 41 p. URL: <https://doi.org/10.1145/3161602>.
107. Mining recent temporal patterns for event detection in multivariate time series data / I. Batal et al. *International Conference on Knowledge Discovery and Data Mining (KDD '12)* : proceedings of the 18th ACM SIGKDD, August 12–16, 2012. New York : ACM, 2012. P. 280–288. URL: <https://doi.org/10.1145/2339530.2339578>.
108. Multivariate time series anomaly detection via dynamic graph attention network and Informer / F. Chen et al. *Applied Intelligence*. 2024. Vol. 54. P. 7636–7658. URL: <https://doi.org/10.1007/s10489-024-05575-y>.
109. Modeling long- and short-term temporal patterns with deep neural

networks / G. Lai et al. *Conference on Research & Development in Information Retrieval (SIGIR '18)* : proceedings of the 41st International ACM SIGIR Conference, July 8–12, 2018. New York : ACM, 2018. P. 95–104. URL: <https://doi.org/10.1145/3209978.3210006>.

110. Shih S-Y., Sun F-K., Lee H-y. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*. 2019. Vol. 108. P. 1421–1441. URL: <https://doi.org/10.1007/s10994-019-05815-0>.

111. Rethinking decomposition for time series forecasting: Learning from temporal patterns / D. Ge et al. *Expert Systems with Applications*. 2025. Vol. 287. Art. 128044. URL: <https://doi.org/10.1016/j.eswa.2025.128044>.

112. Mior M., Ororbia A. Column2Vec: Structural Understanding via Distributed Representations of Database Schemas. *arXiv preprint* arXiv: 1903.08621v1. 2019. URL: <https://doi.org/10.48550/arXiv.1903.08621>.

113. Cococcioni M., Fiorini F., Pagano M. Modelling Heavy Tailed Phenomena Using a LogNormal Distribution Having a Numerically Verifiable Infinite Variance. *Mathematics*. 2023. Vol. 11, Issue 7. Art. 1758. URL: <https://doi.org/10.3390/math11071758>.

114. Mitzenmacher M. A Brief History of Generative Models for Power Law and Lognormal Distributions. *Internet Mathematics*. 2004. Vol. 1, Issue 2. P. 226–251. URL: <https://doi.org/10.1080/15427951.2004.10129088>.

115. When Heavy Tails Disrupt Statistical Inference / R. M. Vogel et al. *The American Statistician*. 2025. Vol. 79, Issue 2. P. 221–235. URL: <https://doi.org/10.1080/00031305.2024.2402898>.

116. An empirical analysis of heavy-tails behavior of financial data: The case for power laws / N. Champagnat et al. *HAL open science*. 2013. 19 p. URL: <https://inria.hal.science/hal-00851429v1> (дата звернення: 02.12.2025).

117. Jaroszewicz S., Korzeń M. Arithmetic Operations on Independent Random Variables: A Numerical Approach. *SIAM Journal on Scientific Computing*. 2012. Vol. 34, Issue 3. P. A1241–A1265. URL: <https://doi.org/10.1137/110839680>.

118. Diwakar R. An Evaluation of Normal Versus Lognormal Distribution in Data Description and Empirical Analysis. *Practical Assessment, Research, and Evaluation*. 2017. Vol. 22. Art. 13. 15 p. URL: <https://doi.org/10.7275/zeyh-j468>.

119. Sherlock: A Deep Learning Approach to Semantic Data Type Detection / M. Hulsebos et al. *Knowledge Discovery & Data Mining (KDD '19)* : proceedings of the 25th ACM SIGKDD International Conference, August 4–8, 2019. New York, NY, USA : Association for Computing Machinery, 2019. P. 1500–1508. URL: <https://doi.org/10.1145/3292500.3330993>.

120. Micci-Barreca D. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. SIGKDD Explorations Newsletter. 2001. Vol. 3, №1. P. 27–32. URL: <https://doi.org/10.1145/507533.507538>.

121. Arat M. M. Learning From High-Cardinality Categorical Features in Deep Neural Networks. *Journal of Advanced Research in Natural and Applied Sciences*. 2022. Vol. 8, Issue 2. P. 222–236. URL: <https://doi.org/10.28979/jarnas.1014469>.

122. Razali N. M., Wah Y. B. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*. 2011. Vol. 2, №1. P. 21–33. URL : <https://www.nrc.gov/docs/ml1714/ml17143a100.pdf> (дата звернення: 04.12.2025).

123. Ghasemi A., Zahediasl S. Normality Tests for Statistical Analysis: A Guide for Non-Statisticians. *International journal of Endocrinology and Metabolism*. 2012. Vol. 10, №2. P. 486–489. URL: <https://doi.org/10.5812/ijem.3505>.

124. Ogunleye L. I., Oyejola B. A., Obisesan K. O. Comparison of Some Common Tests For Normality. *International Journal of Probability and Statistics*. 2018. Vol. 7, № 5. P. 130–137.

125. Descriptive Statistics And Normality Tests For Statistical Data / P. Mishra et al. *Annals of Cardiac Anaesthesia*. 2019. Vol. 22, № 1. P. 67–72.

URL: https://doi.org/10.4103/aca.ACA_157_18.

126. A Lightweight Learned Cardinality Estimation Model / Y. Zhu et al. *IEEE Transactions on Knowledge and Data Engineering*. 2025. Vol. 37. №10. P. 5719–5734. URL: <https://doi.org/10.1109/TKDE.2025.3591025>.

127. Fayard M. Understanding Cardinality: Challenges and Solutions for Data-Heavy Workflows. *DataCamp Tutorial* : website. URL: <https://www.datacamp.com/tutorial/cardinality> (дата звернення: 17.12.2025).

128. Kudo T., Richardson J. SentencePiece: A simple and language independent approach to subword tokenizer and detokenizer for Neural Text Processing. *Empirical Methods in Natural Language Processing: System Demonstrations* : proceedings of the 2018 Conference, October 31–November 4, 2018. Brussels, Belgium : ACL, 2018. P. 66–71. URL: <https://doi.org/10.18653/v1/D18-2012>.

129. Sennrich R., Haddow B., Birch A. Neural Machine Translation of Rare Words with Subword Units. *ACL 2016* : proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin : ACL, 2016. Vol. 1. P. 1715–1725. URL: <https://doi.org/10.18653/v1/P16-1162>.

130. Data Science Discovery: Data Types. *University of Illinois* : website. URL: <https://discovery.cs.illinois.edu> (дата звернення: 17.12.2025).

131. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge: Cambridge University Press, 2008. 506 p.

132. Amazon EMR Documentation. *Amazon Web Services* : website. URL: <https://docs.aws.amazon.com/emr/> (дата звернення: 11.12.2025).

133. Apache Airflow Documentation. *Apache Airflow: Apache Software Foundation* : website. URL: <https://airflow.apache.org/docs/> (дата звернення: 11.12.2025).

134. AWS Glue Documentation. *Amazon Web Services* : website. URL: <https://docs.aws.amazon.com/glue/> (дата звернення: 11.12.2025).

135. Mercadier Y. Distancia's documentation. *distancia 0.0.1*

documentation : website. URL: <https://distancia.readthedocs.io> (дата звернення: 11.12.2025)

136. Xiang S., Nie F., Zhang C. Learning a Mahalanobis distance metric for data clustering and classification. *Pattern Recognition*. 2008. Vol. 41, Issue 12. P. 3600–3612. URL: <https://doi.org/10.1016/j.patcog.2008.05.018>.

137. Zozoulenko N., Cass T., Gonon L. Infinite-dimensional Mahalanobis Distance with Applications. *Journal of Machine Learning Research*. 2025. Vol. 26. P. 1–47. URL: <https://www.jmlr.org/papers/volume26/24-1126/24-1126.pdf> (дата звернення: 03.12.2025).

138. On Approximating Total Variation Distance / A. Bhattacharyya et al. *IJCAI-2023* : proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, August 19–25, 2023. P. 3479–3487. URL: <https://doi.org/10.24963/ijcai.2023/387>.

139. Gabriel P., Marco C. Computational Optimal Transport with Applications to Data Sciences. *Foundations and Trends in Machine Learning*. 2019. Vol. 11, Issue 5–6. P. 355–607. URL: <https://doi.org/10.1561/22000000073>.

140. A Survey on Time-Series Distance Measures / J. Paparrizos et al. *arXiv preprint*. arXiv:2412.20574v1. 2024. URL: <https://doi.org/10.48550/arXiv.2412.20574>.

141. Hodson T. O. Root-mean-square error (RMSE) or mean absolute error (MAE): When to use them or not. *Geoscientific Model Development*. 2022. Vol. 15, Issue 14. P. 5481–5487. <https://doi.org/10.5194/gmd-15-5481-2022>.

142. Hewamalage H., Ackermann K., Bergmeir C. Forecast Evaluation for Data Scientists: Common Pitfalls and Best Practices. *arXiv preprint*. arXiv:2203.10716v2. 2022. URL: <https://doi.org/10.48550/arXiv.2203.10716>.

143. On the selection of forecasting accuracy measures / D. Koutsandreas et al. *Journal of the Operational Research Society*. 2022. Vol. 73, Issue 5. P. 937–954. URL: <https://doi.org/10.1080/01605682.2021.1892464>.

144. Hester N., Hehman E. Differences in Variance, Skewness, and Kurtosis Can Account for Differences in Binary Outcomes. *Social Psychological*

and Personality Science. 2025. 11 p.

URL: <https://doi.org/10.1177/19485506251385636>.

145. Bhargavi A. D. Comparative Study of Static and Contextual Text Vectorization for Sentiment Analysis. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*. 2025. Vol. 13, Issue VII. P. 484–488. URL: <https://doi.org/10.22214/ijraset.2025.73045>.

146. A Study of Text Vectorization Method Combining Topic Model and Transfer Learning / X. Yang et al. *Processes*. 2022. Vol. 10, Issue 2. Art. 350. 16 p. URL: <https://doi.org/10.3390/pr10020350>.

147. Forecasting: theory and practice / Petropoulos, F. et al. *International Journal of Forecasting* 2022. Vol. 38, Issue 3. P. 705–871. URL: <https://doi.org/10.1016/j.ijforecast.2021.11.001>.

148. Rituraj Saha. Understanding TF-IDF. (Term Frequency-Inverse Document Frequency). *GeeksforGeeks* : website.
URL: <https://www.geeksforgeeks.org/machine-learning/understanding-tf-idf-term-frequency-inverse-document-frequency/> (дата звернення: 08.12.2025)

149. Альперт С. І. Основні міри подібності та нові підходи до їх застосування при класифікуванні гіперспектральних космічних зображень. *Математичні машини і системи*. 2019, № 1. С. 143–151. URL: http://www.immsp.kiev.ua/publications/articles/2019/2019_1/01_2019_Alpert.pdf (дата звернення: 08.12.2025).

150. Li B., Han L. Distance Weighted Cosine Similarity Measure for Text Classification. In: Yin, H., et al. *Intelligent Data Engineering and Automated Learning – IDEAL 2013*. IDEAL 2013. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2013. Vol. 8206. P. 611-618. URL: https://doi.org/10.1007/978-3-642-41278-3_74.

151. Levy A., Shalom B. R., Chalamish M. A guide to similarity measures and their data science applications. *Journal of Big Data*. 2025. Vol. 12. Art. 188. 57 p. URL: <https://doi.org/10.1186/s40537-025-01227-1>.

152. Enhancing query relevance: leveraging SBERT and cosine similarity

for optimal information retrieval / K. Venkatesh Sharma et al. *International Journal of Speech Technology*. 2024. Vol. 27. P. 753–763. URL: <https://doi.org/10.1007/s10772-024-10133-5>.

153. Tian X., Xu Q., Fu C. The Frobenius distances from projections to an idempotent matrix. *Linear Algebra and its Applications*. 2024. Vol. 688. P. 21–43. URL: <https://doi.org/10.1016/j.laa.2024.02.010>.

154. Taha H. A. Operations Research: An Introduction. 10th ed. Harlow : Pearson Education, 2017. 849 p.

155. Learning under Concept Drift: A Review / J. Lu et al. *IEEE Transactions on Knowledge and Data Engineering*. 2019. Vol. 31, №. 12. P. 2346–2363. URL: <https://doi.org/10.1109/TKDE.2018.2876857>.

156. Dataset Shift in Machine Learning / J. Quionero-Candela et al. Cambridge : MIT Press, 2009. 728 p.

157. Widmer G., Kubat M. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning*. 1996. Vol. 23. P. 69–101. URL: <https://doi.org/10.1023/A:1018046501280>.

158. Малик І. В., Кириченко Є. О. Підхід до порівняння гетерогенних табличних даних на основі методів компресії. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2025)* : праці XIV Міжнародної науково-практичної конференції, присвяченої 85-й річниці від дня народження проф. Кириченка М. Ф. 13–15 лист. 2025 р. Чернівці : Технодрук, 2025. С. 124-126.

159. Goodrich M. T., Liu S., Panageas I. Exact Learning of Weighted Graphs Using Composite Queries. *Combinatorial Algorithms. IWOCA 2025* : Lecture Notes in Computer Science / eds. H. Fernau, B. Zhu. Cham : Springer, 2025. Vol. 15885. P. 103–117. URL: https://doi.org/10.1007/978-3-031-98740-3_8.

160. From knowledge graph construction to retrieval-augmented generation: a framework for comprehensive earthquake emergency support / Yao L. et al. *Geo-Spatial Information Science*. 2025. 21 p. URL: <https://doi.org/10.1080/10095020.2025.2514813>.

161. Вступ до алгоритмів / Т. Г. Кормен та ін. ; пер. з англ. Н. Куcssуль, О. Музичук. 3-тє вид. Київ : К.І.С., 2019. 1288 с.
162. Łukaszewski A., Nogal Ł., Januszewski M. The Application of the Modified Prim's Algorithm to Restore the Power System Using Renewable Energy Sources. *Symmetry*. 2022. Vol. 14, Issue 5. Art. 1012. URL: <https://doi.org/10.3390/sym14051012>.
163. Nguyen H. Q., Do M. N. Downsampling of Signals on Graphs Via Maximum Spanning Trees. *IEEE Transactions on Signal Processing*. 2015. Vol. 63, № 1. P. 182–191. URL: <https://doi.org/10.1109/TSP.2014.2369013>.
164. Conceptualizing Similarity Measurement in Data Marketplaces / S. Gupta et al. *UK Academy for Information Systems Conference Proceedings 2025*. 2025. Article 10. 21 p. URL: <https://aisel.aisnet.org/ukais2025/10>.
165. Kyrychenko Y., Malyk I. Information Technology for Data Compression and Transformation by Means of Apache Airflow and Amazon EMR. *Security of Infocommunication Systems and Internet of Things*. 2025. Vol. 3, № 1. P. 01004. URL: <https://doi.org/10.31861/sisiot2025.1.01004>.
166. DeepSqueeze: Deep Semantic Compression for Tabular Data / A. Ilkhechi et al. *SIGMOD '20 : : proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, June 14–19, 2020. New York, NY, USA : Association for Computing Machinery. 2020. P. 1733–1746. URL: <https://doi.org/10.1145/3318464.3389734>.
167. Yet Another Compact Time Series Data Representation Using CBOR Templates (YACTS) / S. Molina Araque et al. *Sensors*. 2023. Vol. 23, Issue 11. Art. 5124. URL: <https://doi.org/10.3390/s23115124>.
168. Gertsy O. Research on graphic data formats for compact representation and comparison of images. *Transport Systems and Technologies*. 2024. №43. P. 173–187. URL: <https://doi.org/10.32703/2617-9059-2024-43-14>.
169. A comprehensive review of model compression techniques in machine learning / P. V. Dantas et al. *Applied Intelligence*. 2024. Vol. 54. P. 11804–11844.

URL: <https://doi.org/10.1007/s10489-024-05747-w>.

170. Graph Neural Networks for Tabular Data Learning: A Survey with Taxonomy and Directions / C.-T. Li et al. *ACM Computing Surveys*. 2025. Vol. 58, Issue 1. Art. 16. 51 p. URL: <https://doi.org/10.1145/3744918>.

171. Graph Neural Machine: A New Model for Learning with Tabular Data / G. Nikolentzos et al. *arXiv preprint* ArXiv: 2402.02862v1. 2024. URL: <https://doi.org/10.48550/arXiv.2402.02862>.

172. Wills P., Meyer F. G. Metrics for graph comparison: A practitioner's guide. *PLoS ONE*. 2020. Vol. 15, №2. 54 p. URL: <https://doi.org/10.1371/journal.pone.0228728>.

173. Wang B., Sun Z., Han Y. A. Path-Based Distribution Measure for Network Comparison. *Entropy*. 2020. Vol. 22, Issue 11. Art. 1287. 15 p. URL: <https://doi.org/10.3390/e22111287>.

174. Chitra K., Maheswari D. A Comparative Study of Various Clustering Algorithms in Data Mining. *International Journal of Computer Science and Mobile Computing*. 2017. Vol. 6, Issue 8. P. 109–115. URL: <https://www.ijcsmc.com/docs/papers/August2017/V6I8201725.pdf> (дата звернення: 09.12.2025).

175. Graph Clustering with Graph Neural Networks / A. Tsitsulin et al. *Journal of Machine Learning Research*. 2024. Vol. 24, № 127. 21 p.

176. Clustering graph data: the roadmap to spectral techniques / R. Mondal et al. *Discover Artificial Intelligence*. 2024. Vol.4. Art. 7. 22 p. URL: <https://doi.org/10.1007/s44163-024-00102-x>

177. Clustering and Classification Algorithms in Data Mining / M. V. Ahamad, et al. *International Journal of Advance Research in Science and Engineering*. 2017. Vol. 6, № 7. 8 p.

178. Anand N., Sharma V., Singh P. ETL and Data Warehousing: Architecture, Vulnerabilities, and Security Mechanisms. *International Journal of Scientific and Research Publications*. 2025. Vol. 15, Issue 10. P. 113 – 121. URL: <https://doi.org/10.29322/IJSRP.15.10.2025.p16612>.

179. Spark: Cluster Computing with Working Sets / M. A. Zaharia et al. *Hot Topics in Cloud Computing* : proceedings of the 2nd USENIX Conference, July 9, 2018. USA : USENIX Association, 2010. 7 p.

180. Lakehouse: A new generation of open platforms for big data analytics / M. Armbrust et al. *CIDR 2021* : proceedings of the 11th Conference on Innovative Data Systems Research, January 11-15, 2021. 8 p. URL: https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf (дата звернення: 12.12.2025).

181. Кириченко Є. О., Кириченко О. Л. Побудова моделей нейронних мереж у хмарному середовищі засобами AWS EMR. *Вчені записки Таврійського національного університету імені В.І. Вернадського*. Серія : Технічні науки. 2024. Т. 35(74), №6. С. 75-79. URL: <https://doi.org/10.32782/2663-5941/2024.6.2/11>.

182. What is IaaS? Infrastructure as a Service. *IBM* : website. URL: <https://www.ibm.com/topics/iaas> (дата звернення: 10.12.2025).

183. What is PaaS? (Platform-as-a-Service). *Red Hat* : website. URL: <https://www.redhat.com/en/topics/cloud-computing/what-is-paas> (дата звернення: 10.12.2025).

184. What is software as a service (SaaS)? *Microsoft Azure* : website. URL: <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-saas/> (дата звернення: 10.12.2025).

185. Patel H., Kansara N. Cloud Computing Deployment Models: A Comparative Study. *International Journal of Innovative Research in Computer Science & Technology* (IJIRCST). 2021. Vol. 9, Issue 2. P. 45–50. URL: <https://doi.org/10.21276/ijircst.2021.9.2.8>.

186. Ghazaryan E. Comparative Analysis of Cloud Deployment Models: Public, Private, Hybrid, and Multi-Cloud. *International Journal of Engineering and Computer Science*. Vol. 14, Issue 07. 2025. P. 27572–27578. URL: <https://doi.org/10.18535/ijecs.v14i07.519>.

187. Kyrychenko Y., Malyk I. Leveraging AWS EMR for Scalable and

Efficient Neural Network Deployment in Cloud Computing. *The 13 th International Conference on Electronics, Communications and Computing's (IC ECCO)* : proceedings of the Intern. Conf., 17–18 October, 2024. Chisinau : Technical University of Moldova, 2024. P. 128–129. URL: <https://ecco.utm.md/wp-content/uploads/2024/12/IC-ECCO-2024-AbstractBookBN.pdf>.

188. Spark SQL: Relational Data Processing in Spark / Armbrust M. et al. *SIGMOD '15* : proceedings of the 2015 International Conference on Management of Data, May 31– June 4, 2015. New York, NY, USA : Association for Computing Machinery, 2015. P. 1383–1394. URL: <https://doi.org/10.1145/2723372.2742797>.

189. Documentation: File Format. *Parquet* : website. URL: <https://parquet.apache.org/docs/file-format/> (дата звернення: 11.12.2025).

190. LanguageManual ORC. *Apache Hive* : website. URL: <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC> (дата звернення: 11.12.2025).

191. Amazon Athena Documentation. *Amazon Web Services* : website. URL: <https://docs.aws.amazon.com/athena/> (дата звернення: 11.12.2025).

192. Ganapathi J. Unified Data Processing with Spark Structured Streaming and Delta CDF on AWS. *International Journal of Scientific and Research Publications (IJSRP)*. 2025. Vol. 15, Issue 6. P. 42–54. URL: <http://dx.doi.org/10.29322/IJSRP.15.06.2025.p16208>.

193. Кириченко Є., Малик І. Інформаційна технологія структурного аналізу табличних наборів даних для виявлення аномалій та невідповідностей. *Science in the Context of Modern Challenges and Prospects* : proceedings of the Intern. Conf., 21 травня 2025 р. (м. Утрехт, Нідерланди). 2025. С. 196–197. URL: <https://researcheurope.org/wp-content/uploads/2025/06/re-21.05.25.pdf>.

194. Amazon Simple Storage Service Documentation. *Amazon Web Services* : website. URL: <https://docs.aws.amazon.com/s3/> (дата звернення: 11.12.2025).

195. Apache Hive: Distributed Data Warehouse at Massive Scale. *Apache Hive* : website. URL: <https://hive.apache.org/> (дата звернення: 11.12.2025).
196. Welcome to Apache Pig! *Apache Hadoop* : website. URL: <https://pig.apache.org/> (дата звернення: 11.12.2025).
197. Apache HBase: The Hadoop Database. *Apache HBase* : website. URL: <https://hbase.apache.org/> (дата звернення: 11.12.2025).
198. Presto Documentation. *Presto Foundation* : website. URL: <https://prestodb.io/docs/current/> (дата звернення: 11.12.2025).
199. Architecture for Amazon EMR on EKS. *Amazon Web Services. Amazon EMR on EKS Development Guide* : website. URL: <https://docs.aws.amazon.com/emr/latest/EMR-on-EKS-DevelopmentGuide/emr-eks-overview.html> (дата звернення: 11.12.2025).
200. Storage Best Practices for Data and Analytics Applications. *Amazon Web Services* : website. URL: <https://docs.aws.amazon.com/whitepapers/latest/building-data-lakes/building-data-lake-aws.html> (дата звернення: 11.12.2025).
201. Durner D., Leis V., Neumann T. Exploiting Cloud Object Storage for High-Performance Analytics. *The VLDB Endowment*. 2023. Vol. 16, Issue 11. P. 2769–2782. URL: <https://doi.org/10.14778/3611479.3611486>.
202. Intorf D., Storey D., van Doorn K. Apache Airflow Best Practices. Birmingham : Packt Publishing Limited, 2024. 188 p.
203. Harenslak B., de Ruiter J. Data Pipelines with Apache Airflow. 2nd ed. Shelter Island : Manning Publications, 2021. 480 p.
204. Reis J., Housley M. Fundamentals of Data Engineering : Plan and Build Robust Data Systems. Sebastopol : O'Reilly Media, 2022. 447 p.
205. Кириченко Є. О., Малик І. В. Порівняльний аналіз економічної ефективності AWS ELASTIC MAPREDUCE та EC2 для розгортання нейронних мереж. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2024)* : праці XIII Міжнар. наук.-практ. конф., 01–03 листопада 2024 р. Чернівці : Черн. нац. ун-т, 2024. С. 40–42. URL:

https://drive.google.com/file/d/1YY-YHwal-apT_TVEOtaGh6rBiT0WJm8x.

206. Ivanov T., Pergolesi M. The impact of columnar file formats on SQL-on-hadoop engine performance: A study on ORC and Parquet. *Concurrency and Computation: Practice and Experience*. 2019. Vol. 32, Issue 5. Art. e5523.

URL: <https://doi.org/10.1002/cpe.5523>.

207. A Deep Dive into Common Open Formats for Analytical DBMSs / C. Liu et al. *The VLDB Endowment*. 2023. Vol. 16, Issue 11. P. 3044–3056. URL: <https://doi.org/10.14778/3611479.3611507>.

208. Himpe C. DatAasee -- A Metadata-Lake as Metadata Catalog for a Virtual Data-Lake. *arXiv preprint ArXiv: 2409.05512v3*. 2025. URL: <https://doi.org/10.48550/arXiv.2409.05512>.

209. Kulkarni A. Amazon Athena: Serverless Architecture and Troubleshooting. *International Journal of Computer Trends and Technology (IJCTT)*. 2023. Vol. 71, Issue 5. P. 57–61. URL: <https://doi.org/10.14445/22312803/IJCTT-V71I5P110>.

210. Enhancing Data Integration and ETL Processes Using AWS Glue / B. Pothineni et al. *International Journal of Research and Analytical Reviews (IJRAR)*. 2024. Vol. 11, Issue 4. P. 728–733. URL: <https://doi.org/10.5281/zenodo.14577615>.

211. Berisha B., Mëziu E., Shabani I. Big data analytics in Cloud computing: an overview. *Journal of Cloud Computing*. 2022. Vol. 11. Art. 24. 10 p. URL: <https://doi.org/10.1186/s13677-022-00301-w>.

212. Kodakandla P. Modernizing Legacy Hadoop Infrastructure through Cloud-Native Migration on AWS. *International Scientific Journal of Engineering and Management*. 2022. Vol 1, Issue 4. 13 p. URL: <https://doi.org/10.55041/ISJEM00064>.

213. Naseemuddin M. Dynamic resource allocation techniques for optimizing cost and performance in multi-cloud environments. *International journal of cloud computing (IJCC)*. 2023. Vol.1, №1. P. 1–12. URL: https://iaeme-library.com/index.php/IJCC/article/view/IJCC_01_01_001 (дата звернення:

12.12.2025).

214. Pandas documentation. *Pandas* : website. URL: <https://pandas.pydata.org/> (дата звернення: 13.12.2025).

215. The fundamental package for scientific computing with Python. *NumPy* : website. URL: <https://numpy.org/> (дата звернення: 13.12.2025).

216. Fundamental algorithms for scientific computing in Python. *SciPy* : website. URL: <https://scipy.org/> (дата звернення: 13.12.2025).

217. scikit-learn: Machine Learning in Python. *scikit-learn* : website. URL: <https://scikit-learn.org/> (дата звернення: 11.13.2025).

218. PySpark Documentation. *Apache Spark* : website. URL: <https://spark.apache.org/docs/latest/api/python/> (дата звернення: 13.12.2025).

219. Large scale performance analysis of distributed deep learning frameworks for convolutional neural networks / M. Aach et al. *Journal of Big Data*. 2023. Vol. 10, Art. 96. 23 p. URL: <https://doi.org/10.1186/s40537-023-00765-w>.

220. Managed Resource Scaling in Amazon EMR / V. Vyas et al. *SIGMOD/PODS '25* : proceedings of the 2025 International Conference on Management of Data, June 22 – 27, 2025. Berlin, Germany. New York : Association for Computing Machinery, 2025. P. 662–674. URL: <https://doi.org/10.1145/3722212.3724443>.

221. Stocks Overview. *Massive Docs* : website. URL: <https://massive.com/docs/rest/stocks/overview> (дата звернення: 02.12.2025).

222. What is Amazon EMR Serverless?. *Amazon Web Services. Amazon EMR Serverless User Guide* : website. URL: <https://docs.aws.amazon.com/emr/latest/EMR-Serverless-UserGuide/emr-serverless.html> (дата звернення: 11.12.2025).

223. Amazon EC2. *Amazon Web Services* : website. URL: <https://aws.amazon.com/ec2> (дата звернення: 11.12.2025).

224. Madabushini P. Optimizing GPU Utilization for AI Workloads on AWS EKS. *World Journal of Advanced Research and Reviews*. 2025. 26(01). P.

1955–1963. URL: <https://doi.org/10.30574/wjarr.2025.26.1.1233>.

225. Shukla T. Apple Inc. (AAPL) Stock Data 1980-2024. Kaggle. URL: <https://www.kaggle.com/datasets/iamtanmayshukla/apple-inc-aapl-stock-data-1980-2024> (дата звернення: 05.12.2025).

226. Shukla T. Tesla Stocks Dataset. Kaggle. URL: <https://www.kaggle.com/datasets/iamtanmayshukla/tesla-stocks-dataset> (дата звернення: 05.12.2025).

227. Parmar K. Social Media Sentiments Analysis Dataset. Kaggle. URL: <https://www.kaggle.com/datasets/kashishparmar02/social-media-sentiments-analysis-dataset> (дата звернення: 05.12.2025).

228. E-Commerce Data / carrie1. Kaggle. URL: <https://www.kaggle.com/datasets/carrie1/ecommerce-data> (дата звернення: 05.12.2025).

229. Dataset-JSON: A data exchange standard for sharing tabular data using JSON. *CDISC (Clinical Data Interchange Standards Consortium)* : website. URL: <https://www.cdisc.org/standards/data-exchange/dataset-json> (дата звернення: 13.12.2025).

230. JSON-stat: A simple lightweight standard for data dissemination. *JSON-stat* : website. URL: <https://json-stat.org/> (дата звернення: 13.12.2025).

231. Tharwat A. Classification assessment methods. *Applied Computing and Informatics*. 2021. Vol. 17. No. 1. P. 168–192. URL: <https://doi.org/10.1016/j.aci.2018.08.003>.

232. Kyrychenko Y. O., Kyrychenko O. L., Filipchuk O. I. The Study Of Complex Networks Using Visualization Tools. *Science and technology today*. «Technics» series. 2024. № 10(38). С. 81–93. URL: [https://doi.org/10.52058/2786-6025-2024-10\(38\)-81-92](https://doi.org/10.52058/2786-6025-2024-10(38)-81-92).

ДОДАТКИ

Додаток А

Список публікацій за темою дисертації

Наукові праці, в яких опубліковані основні наукові результати дисертації:

Наукові праці у виданнях, включених до переліку наукових фахових видань України та проіндексованих у наукометричній базі даних Scopus:

1. Кириченко О., **Кириченко Є.** Асимптотичні властивості випадкових матриць = Asymptotic Properties of Randommatrices. *Вісник Київського національного університету імені Тараса Шевченка. Фізико-математичні науки.* 2024. 78(1). С. 41–44. (**Scopus**) <https://doi.org/10.17721/1812-5409.2024/1.7> (Кириченко О. – концептуалізація, формальний аналіз; Кириченко Є. – аналіз джерел, підготовка огляду літератури).

Наукові праці у періодичних наукових виданнях, проіндексованих у наукометричній базі даних Scopus:

2. Malyk I. V., **Kyrychenko Y.**, Gorbatenko M., Lukashiv T. Data Optimization through Compression Methods Using Information Technology. *International Journal of Information Technology and Computer Science.* 2025. Vol.17, No.5. P. 84-99. (**Scopus, Q3** – <https://www.scimagojr.com/journalsearch.php?q=21101226492&tip=sid&clean=0>) <https://doi.org/10.5815/ijitcs.2025.05.07> (Малик І. – концептуалізація, методологія, формальний аналіз, написання та підготовка оригінальної чернетки, написання, рецензування та редагування, візуалізація, авторський нагляд; Кириченко Є. – концептуалізація, методологія, формальний аналіз, дослідження, написання та підготовка оригінальної чернетки, написання-рецензування та редагування, візуалізація; Горбатенко М. – методологія, формальний аналіз; Лукашів Т. – концептуалізація, написання, рецензування та редагування, авторський нагляд, адміністрування проєкту).

**Наукові праці у виданнях, включених до переліку
наукових фахових видань України:**

3. **Кириченко Є. О.**, Кириченко О. Л. Побудова моделей нейронних мереж у хмарному середовищі засобами AWS EMR. *Вчені записки Таврійського національного університету імені В.І. Вернадського*. Серія : Технічні науки. 2024. 35(74), №6. С. 75-79. <https://doi.org/10.32782/2663-5941/2024.6.2/11> (Кириченко Є. – теоретична та практична розробка положень, програмна реалізація, редагування рукопису; Кириченко О. – постановка задачі, керівництво дослідженням, рецензування та редагування).

4. **Kyrychenko Y.**, Malyk I. Information Technology for Data Compression and Transformation by Means of Apache Airflow and Amazon EMR. *Security of Infocommunication Systems and Internet of Things*. 2025. Vol. 3, no. 1. P. 01004. <https://doi.org/10.31861/sisiot2025.1.01004> (Кириченко Є. – проектування архітектури системи, програмне забезпечення, розробка та тестування програмного забезпечення, написання та редагування; Малик І. – загальне керівництво, концептуалізація).

5. **Кириченко Є.**, Малик, І. Один результат кластеризації на основі випадкових матриць. *Information Technology: Computer Science, Software Engineering and Cyber Security*. 2025. №3. С. 257-270. <https://doi.org/10.32782/IT/2025-3-29> (Кириченко Є. – проведення дослідження, редагування рукопису; Малик І. – постановка задачі, визначення загальної схеми досліджень, обговорення результату).

Наукові праці, які засвідчують апробацію матеріалів дисертації:

6. **Кириченко Є. О.**, Малик І. В. Використання подвійних стохастичних моделей в BIG DATA. *Проблеми інформатики та комп'ютерної техніки (ПІКТ – 2023)* : праці XII Міжнародної науково-практичної конференції, м. Чернівці, 10 – 12 листопада 2023 р. Чернівці : Черн. нац. ун-т, 2023. С. 33–35. (Кириченко Є. – огляд літературних джерел, написання, рецензування та редагування; Малик І. – постановка задачі, рецензування).

7. **Кириченко Є. О.**, Малик І. В. Моделі і процеси при дослідженні подвійних стохастичних систем. *Інформаційні технології: наука, техніка, технологія, освіта, здоров'я* : тези доп. XXXII Міжнар. наук.-практ. конф. MicroCAD-2024, 22–25 травня 2024 р. / за ред. проф. Сокола Є.І. Харків : НТУ «ХПІ». С. 1294. (Кириченко Є. – проведення дослідження, написання рукопису; Малик І. – концептуалізація, методологія, загальне керівництво).

8. **Kyrychenko Y.**, Malyk I. Leveraging AWS EMR for Scalable and Efficient Neural Network Deployment in Cloud Computing. *The 13th International Conference on Electronics, Communications and Computing's (IC ECCO)* : Materials of the Intern. Conf., (Chisinau, Moldova, 17–18 October, 2024.). Chisinau : Technical University of Moldova, 2024. P. 128-129. (Кириченко Є. – проведення досліджень, аналіз та опис результатів; Малик І. – постановка задачі, обговорення результатів).

9. **Кириченко Є. О.**, Малик І. В. Порівняльний аналіз економічної ефективності AWS ELASTIC MAPREDUCE та EC2 для розгортання нейронних мереж. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2024)* : праці XIII Міжнар. наук.-практ. конф., м. Чернівці, 01–03 листопада 2024 р. Чернівці : Черн. нац. ун-т, 2024. С. 40–42. (Кириченко Є. – збір та аналіз даних, обговорення результатів; Малик І. – концептуалізація дослідження, розробка методології, аналіз результатів).

10. **Кириченко Є.**, Малик І. Інформаційна технологія структурного аналізу табличних наборів даних для виявлення аномалій та невідповідностей. *Science in the Context of Modern Challenges and Prospects* : Materials of the Intern. Conf., 21 травня 2025 р. (м. Утрехт, Нідерланди). 2025. С. 196-197. (Кириченко Є. – проведення дослідження, аналіз та опис результату; Малик І. – визначення загальної схеми дослідження).

11. Малик І. В., **Кириченко Є. О.** Підхід до порівняння гетерогенних табличних даних на основі методів компресії. *Проблеми інформатики та комп'ютерної техніки* : праці XIV Міжнародної науково-практичної конференції (ПІКТ – 2025), присвяченої 85-й річниці від дня народження

проф. Кириченка М.Ф. м. Чернівці, 13–15 лист. 2025. Чернівці : Технодрук, 2025. С. 124-126. (Кириченко Є. – концептуалізація, методологія, формальний аналіз, дослідження, написання; Малик І. – постановка задачі, загальне керівництво).

***Наукові праці, які додатково відображають
наукові результати дисертації:***

12. **Kyrychenko Y. O.**, Kyrychenko O. L., Filipchuk O. I. The Study Of Complex Networks Using Visualization Tools. *Science and technology today*. «Technics» series. 2024. № 10(38). С. 81-93. [https://doi.org/10.52058/2786-6025-2024-10\(38\)-81-92](https://doi.org/10.52058/2786-6025-2024-10(38)-81-92) (Кириченко Є. – концептуалізація, методологія, формальний аналіз, дослідження, підготовка ілюстративних матеріалів; Кириченко О. – концептуалізація, обговорення результатів; Філіпчук О. – обговорення результатів, рецензування та редагування)

Відомості про апробацію матеріалів дисертації

Апробація матеріалів дисертації здійснювалася на конференціях та семінарах:

- «Проблеми інформатики та комп'ютерної техніки (ПІКТ)» (Чернівці, 2023-2025),
- «Інформаційні технології: наука, техніка, технологія, освіта, здоров'я (MicroCAD)» (Харків, 2024),
- The 13th International Conference on Electronics, Communications and Computing's (IC ECCO) (м. Кишинів, Молдова, 2024),
- «Science in the Context of Modern Challenges and Prospects» (м. Утрехт, Нідерланди, 2025),
- наукові семінари кафедр математичних проблем управління і кібернетики та програмного забезпечення комп'ютерних систем Чернівецького національного університету імені Юрія Федьковича.

Акти впровадження результатів дисертаційної роботи

Coderse
Development Company

Товариство з обмеженою
відповідальністю «Кодерс ПРО»
Україна, 58002, м. Чернівці,
вул. Головна, буд. 71, оф. 7.
Тел. +380994406032, e-mail: office@coderse.pro

«15» квітня 2025 р.
Вих. № 18/25

Акт
впровадження результатів дисертаційної роботи
Кириченка Євгена Олександровича
на тему «Оптимізація структури гетерогенних даних в Big Data»,
представленої до захисту на здобуття наукового ступеня
доктора філософії з галузі знань 12 – «Інформаційні технології»
за спеціальністю 121 – «Інженерія програмного забезпечення»

Цей акт складено на підтвердження того, що ТОВ «Кодерс ПРО» використовує у своїй діяльності результати, які реалізовано в дисертаційному дослідженні здобувача ступеня доктора філософії Чернівецького національного університету імені Юрія Федьковича Кириченка Євгена Олександровича, а саме:

- запропонований підхід до побудови компактних представлень (CDR) та метрики DISS для порівняння табличних даних дав можливість створити уніфікований механізм кластеризації гетерогенних наборів, що містять числові, категоріальні, часові та текстові змінні;
- розроблений метод багаторівневої кластеризації на основі структурних відстаней дозволив виявляти групи подібних об'єктів у великих датасетах, уникаючи ручної нормалізації та попередньої інженерії ознак;
- запропоновані методи інтеграції кластеризаційних обчислень у аналітичні модулі підприємства забезпечили можливість побудови масштабованих рішень, що працюють у багатопотокових режимах, підтримують контейнеризацію й легко інтегруються в наявні інформаційні системи компанії.

Використання зазначених результатів дозволило підвищити точність сегментації даних, покращити якість аналітичних процесів, автоматизувати виявлення нетипових об'єктів та суттєво зменшити витрати часу на обробку великих масивів інформації у розроблюваних компанією рішеннях.

Директор
ТОВ «Кодерс ПРО»



Валь О. Д.

Акт
впровадження результатів дисертаційної роботи
Кириченко Євгена Олександровича
на тему «Оптимізація структури гетерогенних даних в Big Data»,
представленої до захисту на здобуття наукового ступеня
доктора філософії з галузі знань 12 – «Інформаційні технології»
за спеціальністю 121 – «Інженерія програмного забезпечення»

Цей акт підтверджує, що компанія ТОВ "ПАЛЕТНИЙ СЕРВІС" використовує у своїй роботі результати, які реалізовано в дисертаційному дослідженні здобувача ступеня доктора філософії Чернівецького національного університету імені Юрія Федьковича Кириченка Євгена Олександровича, а саме:

розроблено програмний модуль для автоматизованого аналізу структури даних на основі компактних представлень (Compact Data

- автоматичну ідентифікацію типів змінних у великих CSV-та інших табличних наборах даних;
- побудову компактних структурних репрезентацій для подальшого порівняння та кластеризації;
- виявлення відхилень, аномалій та нерегулярних патернів у гетерогенних даних.

Реалізовано метод порівняння табличних наборів даних на основі метрики структурної подібності DISS (Data Information Structural

- обчислюються відстані між наборами даних з урахуванням числових, категоріальних, часових і текстових атрибутів;
- інтегровано широкий спектр статистичних метрик (MAE, MAPE, Wasserstein, Hellinger, Total Variation) у єдину систему порівняння;
- формуються матриці відстаней і визначаються найбільш подібні та відмінні сценарії даних.

Директор



Худоба П.Р.

«Затверджую»

Директор Навчально-наукового
інституту фізико-технічних та
комп'ютерних наук Чернівецького
національного університету імені Юрія
Федьковича
канд. техн. н., доцент

 П.М. Шпатар
« 08 » грудня 2025 р.

АКТ

**впровадження результатів дисертаційної роботи
Кириченко Євгена Олександровича «Оптимізація структури
гетерогенних даних в Big Data» до захисту на здобуття наукового ступеня
доктора філософії за спеціальністю «Інженерія програмного
забезпечення» в навчальному процесі**

Комісія Чернівецького національного університету імені Юрія
Федьковича в складі:

- заступник директора ННІФТКН з наукової роботи, д.ф.-м.н., професор Дуболазов О.В.;
- завідувача кафедри математичних проблем управління і кібернетики, д.ф.-м.н., професора Малика І.В.;
- завідувача кафедри програмного забезпечення комп'ютерних систем, доктора філософії, доцента Газдюк К.П.

склали цей акт про те, що результати дисертаційної роботи здобувача наукового ступеня доктора філософії за спеціальністю «Інженерія програмного забезпечення» Кириченко Є.О. впроваджені в навчальний процес кафедр МПУіК та ПЗКС, зокрема, в дисциплінах «Безсерверні обчислення у хмарних середовищах», «Нейронні мережі», «Технології машинного навчання», «Методи та засоби кластерного аналізу», «Проектування інформаційних систем».

При викладанні цих дисциплін використовуються наступні матеріали досліджень, які отримані в дисертаційній роботі:

- Розробка програмного модуля для автоматизованого аналізу структури даних. Реалізація функціоналу для ідентифікації типів змінних у великих наборах даних, створення компактних репрезентацій та виявлення аномалій у гетерогенних даних.
- Імплементация системи порівняння даних на базі метрики DISS. Об'єднання широкого спектру статистичних інструментів для оцінки подібності числових, текстових і часових атрибутів, що дозволяє ефективно формувати матриці відстаней.
- Використання удосконаленого алгоритму спектральної кластеризації для складних структур даних. Застосування модифікованих методів аналізу спектрального розриву та визначення кількості кластерів, що значно підвищує інформативність та точність групування даних.
- Використання хмарної технології AWS EMR (Amazon Web Services Elastic MapReduce) для побудови моделей нейронних мереж.
- Визначення специфіки навчання моделей нейронних мереж на великих масивах даних та здійснення порівняльного аналізу рішень для розгортання нейронних мереж на основі стохастичних моделей за допомогою хмарних сервісів.
- Особливості розгортання моделей машинного навчання, а також визначення підходів до вибору відповідної інфраструктури.

Використання зазначених результатів дозволило підвищити якість навчального процесу із згаданих дисциплін.

Заступник директора ННІФТКН з
наукової роботи д.ф.-м.н., професор



О.В. Дуболазов

Завідувач кафедри МПУіК,
д.ф.-м.н., професор



І.В. Малик

Завідувач кафедри ПЗКС,
доктор філософії, доцент



К.П. Газдюк

Лістинг частини коду програми

// Вміст файлу config.py

Конфігурація та моделі даних

```
import os
from dataclasses import dataclass
from datetime import datetime
from typing import Dict, List, Optional

@dataclass
class AnalysisConfig:

    data_source: str = "kaggle"
    kaggle_paths: Dict[str, str] = None
    yahoo_stocks: List[str] = None
    start_date: str = "2015-01-01"
    end_date: str = "2024-01-01"

    n_sensitivity_points: int = 10
    n_fl_windows: int = 50
    slice_percentages: List[int] = None

    figure_dpi: int = 300
    style: str = 'seaborn-v0_8-darkgrid'

    def __post_init__(self):
        if self.kaggle_paths is None:
            self.kaggle_paths = {
                "AAPL": "data/AAPL.csv",
                "TSLA": "data/TSLA.csv"
            }

        if self.yahoo_stocks is None:
            self.yahoo_stocks = ["AAPL", "NVDA", "MSFT"]

        if self.slice_percentages is None:
            self.slice_percentages = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

class OutputManager:

    def __init__(self, base_name: str = "результати_підвибіркового_аналізу"):
        self.timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        self.output_dir = f"{base_name}_{self.timestamp}"
        os.makedirs(self.output_dir, exist_ok=True)
```

```

        print(f"Створено папку: {self.output_dir}")

    def get_filepath(self, filename: str) -> str:
        return os.path.join(self.output_dir, filename)

    def save_plot(self, fig, filename: str, tight: bool = True):
        filepath = self.get_filepath(filename)
        fig.savefig(
            filepath,
            dpi=300,
            bbox_inches='tight' if tight else None,
            facecolor='white',
            edgecolor='none'
        )
        print(f"Збережено: {filename}")
        import matplotlib.pyplot as plt
        plt.close(fig)

// Вміст файлу data_loader.py
Завантаження та кешування даних

import os
from typing import Dict, Optional

import pandas as pd
import yfinance as yf

class CDRCache:

    def __init__(self):
        self._cache: Dict = {}

    def get_key(self, df: pd.DataFrame) -> Optional[tuple]:
        if df.empty:
            return None
        try:
            return (
                df.shape,
                hash(tuple(df.iloc[0].values)),
                hash(tuple(df.iloc[-1].values))
            )
        except:
            return None

    def get(self, df: pd.DataFrame) -> Optional[Dict]:

```

```

    key = self.get_key(df)
    return self._cache.get(key) if key else None

def set(self, df: pd.DataFrame, cdr: Dict):
    key = self.get_key(df)
    if key:
        self._cache[key] = cdr

def clear(self):
    self._cache.clear()

class KaggleDataLoader:

    def __init__(self, kaggle_paths: Dict[str, str]):
        self._data: Dict[str, pd.DataFrame] = {}
        self.kaggle_paths = kaggle_paths
        self._load_data()

    def _load_data(self):
        print("Завантаження даних Kaggle...")

        col_map = {
            'date': 'Date', 'open': 'Open', 'high': 'High', 'low': 'Low',
            'close': 'Close', 'volume': 'Volume', 'adj_close': 'Adj Close',
            'adjusted_close': 'Adj Close', 'adj close': 'Adj Close'
        }

        for ticker, path in self.kaggle_paths.items():
            if not os.path.exists(path):
                print(f"Файл не знайдено: {path}")
                continue

            try:
                df = pd.read_csv(path)
                df.columns = [col_map.get(c.lower(), c) for c in df.columns]

                if "Date" in df.columns:
                    df["Date"] = pd.to_datetime(df["Date"])
                    df.sort_values("Date", inplace=True)
                    df.reset_index(drop=True, inplace=True)

                self._data[ticker] = df
                print(f"{ticker}: {len(df)} рядків")
            except Exception as e:
                print(f"Помилка {ticker}: {e}")

    def download(self, ticker: str, start: Optional[str] = None,

```

```

        end: Optional[str] = None, progress: bool = False) -> pd.DataFrame:
    if ticker not in self._data:
        return pd.DataFrame()

```

```

    df = self._data[ticker].copy()

```

```

    if "Date" in df.columns and (start or end):
        if start:
            df = df[df["Date"] >= pd.to_datetime(start)]
        if end:
            df = df[df["Date"] <= pd.to_datetime(end)]
        df.set_index("Date", inplace=True)

```

```

    return df

```

```

class DataFetcher:

```

```

    def __init__(self, config, kaggle_loader: Optional[KaggleDataLoader] = None):
        self.config = config
        self.kaggle_loader = kaggle_loader
        self._setup_cache_dirs()

```

```

    def _setup_cache_dirs(self):
        if self.config.data_source == "kaggle":
            self.cache_dir = "kaggle_cache"
        else:
            self.cache_dir = "yh_data"
        os.makedirs(self.cache_dir, exist_ok=True)

```

```

    def fetch(self, ticker: str, start: str, end: str) -> pd.DataFrame:
        print(f"Завантаження {ticker} від {start} до {end}...")

```

```

        cache_path = os.path.join(self.cache_dir, f"{ticker}.csv")

```

```

        if os.path.exists(cache_path):
            try:
                df = pd.read_csv(cache_path)
                if "Date" in df.columns:
                    df["Date"] = pd.to_datetime(df["Date"])
                print(f"Завантажено з кешу ({self.config.data_source}): {len(df)} рядків")
                return df
            except Exception as e:
                print(f"Помилка читання кешу: {e}")

```

```

        try:
            if self.config.data_source == "kaggle" and self.kaggle_loader:
                df = self.kaggle_loader.download(ticker, start=start, end=end)

```

```

        if df.empty:
            print(f'{ticker} не знайдено в Kaggle даних")
            return pd.DataFrame()
        else:
            df = yf.download(ticker, start=start, end=end, progress=False)

            if isinstance(df.columns, pd.MultiIndex):
                df.columns = df.columns.get_level_values(0)

            if isinstance(df.index, pd.DatetimeIndex):
                df.reset_index(inplace=True)

            if not df.empty:
                df.to_csv(cache_path, index=False)
                print(f"Збережено в кеш ( {self.config.data_source})")

            print(f'{len(df)} рядків")
            return df

    except Exception as e:
        print(f"Помилка: {e}")
        return pd.DataFrame()

// Вміст файлу analysis_core.py
Ядро аналізу: CDR, матриці несхожості, MST

from functools import lru_cache
from typing import Dict, Tuple

import numpy as np
import pandas as pd

class CSVAnalyzer:

    @staticmethod
    def _is_date_like_column(series: pd.Series, col_name: str) -> bool:
        date_name_hints = {'date', 'time', 'timestamp', 'datetime', 'дата', 'час'}
        if col_name.lower().strip() in date_name_hints:
            return True

        if series.dtype == object or str(series.dtype) == 'str':
            sample = series.dropna().head(20)
            if len(sample) == 0:
                return False
            try:
                parsed = pd.to_datetime(sample, errors='coerce')
                if parsed.notna().mean() >= 0.8:

```

```

        return True
    except Exception:
        pass

    return False

@staticmethod
@lru_cache(maxsize=128)
def identify_type(series_name: str, nunique: int, length: int, dtype_str: str,
                  is_date_like: bool = False) -> str:
    if dtype_str.startswith('datetime') or is_date_like:
        return "Час"
    if dtype_str in ['int64', 'float64', 'int32', 'float32']:
        return "Числовий"
    if nunique / length < 0.1:
        return "Категоріальний"
    return "Текст"

def get_compact_representation(self, df: pd.DataFrame, cache=None) -> Dict:
    if cache:
        cached = cache.get(df)
        if cached is not None:
            return cached

    summary = {}

    for col in df.columns:
        data = df[col].dropna()
        if data.empty:
            continue

        date_like = self._is_date_like_column(data, col)

        dtype = self.identify_type(
            col,
            data.nunique(),
            len(data),
            str(data.dtype),
            is_date_like=date_like
        )

        desc = {"type": dtype}

        if dtype == "Числовий":
            desc["moments"] = [
                float(data.mean()),
                float(data.var()),
                float(data.skew()),

```

```

        float(data.min()),
        float(data.max())
    ]
    elif dtype == "Чис":
        if str(data.dtype).startswith('datetime'):
            numeric = data.astype(np.int64)
        else:
            parsed = pd.to_datetime(data, errors='coerce').dropna()
            if len(parsed) > 0:
                numeric = parsed.astype(np.int64)
            else:
                numeric = pd.Series(dtype=float)

        if len(numeric) > 0:
            desc["moments"] = [
                float(numeric.mean()),
                float(numeric.var()) if len(numeric) > 1 else 0.0,
                float(numeric.skew()) if len(numeric) > 2 else 0.0,
                float(numeric.min()),
                float(numeric.max())
            ]
        elif dtype == "Категоріальний":
            desc["freq_profile"] = data.value_counts(normalize=True).to_dict()
    elif dtype == "Текст":
        str_data = data.astype(str)
        desc["text_stats"] = [
            float(str_data.str.len().mean()),
            int(data.nunique()),
            float(str_data.str.len().min()),
            float(str_data.str.len().max()),
            int(len(data))
        ]

    summary[col] = desc

    if cache:
        cache.set(df, summary)

    return summary

```

```

class CompactRepresentationComparator:

```

```

    @staticmethod
    def _moments_distance(moments_a: list, moments_b: list) -> float:
        if len(moments_a) != len(moments_b):
            return 1.0

```

```

diffs = []
for va, vb in zip(moments_a, moments_b):
    max_val = max(abs(va), abs(vb))
    if max_val > 0:
        diffs.append(abs(va - vb) / max_val)

if not diffs:
    return 0.0

return min(float(np.mean(diffs)), 1.0)

def calculate_distance(self, desc_a: Dict, desc_b: Dict) -> float:
    if desc_a["type"] != desc_b["type"]:
        return 1.0

    if desc_a["type"] == "Числовий":
        return self._moments_distance(desc_a["moments"], desc_b["moments"])

    if desc_a["type"] == "Час":
        moments_a = desc_a.get("moments")
        moments_b = desc_b.get("moments")
        if moments_a and moments_b:
            return self._moments_distance(moments_a, moments_b)
        return 0.0

    if desc_a["type"] == "Категоріальний":
        keys_a = set(desc_a["freq_profile"].keys())
        keys_b = set(desc_b["freq_profile"].keys())
        union = len(keys_a.union(keys_b))

        if union == 0:
            return 1.0

        return 1.0 - (len(keys_a.intersection(keys_b)) / union)

    if desc_a["type"] == "Текст":
        stats_a = desc_a.get("text_stats")
        stats_b = desc_b.get("text_stats")
        if stats_a and stats_b:
            diffs = []
            for va, vb in zip(stats_a, stats_b):
                max_val = max(abs(va), abs(vb))
                if max_val > 0:
                    diffs.append(abs(va - vb) / max_val)
            return min(float(np.mean(diffs)), 1.0) if diffs else 0.0
        return 0.0

return 0.0

```

```

def build_diss_matrix(
    self,
    summary_a: Dict,
    summary_b: Dict,
    is_self_comparison: bool = False,
) -> Tuple[pd.DataFrame, pd.DataFrame]:
    cols_a = list(summary_a.keys())
    cols_b = list(summary_b.keys())

    n_a, n_b = len(cols_a), len(cols_b)

    if summary_a is summary_b:
        is_self_comparison = True

    matrix = np.zeros((n_a, n_b))
    mask = np.zeros((n_a, n_b), dtype=bool)

    for i, col_a in enumerate(cols_a):
        for j, col_b in enumerate(cols_b):
            desc_a = summary_a[col_a]
            desc_b = summary_b[col_b]

            if is_self_comparison and col_a == col_b:
                matrix[i, j] = 0.0
                mask[i, j] = False

            elif desc_a["type"] != desc_b["type"]:
                mask[i, j] = True

            else:
                matrix[i, j] = self.calculate_distance(desc_a, desc_b)
                mask[i, j] = False

    if is_self_comparison and n_a == n_b and set(cols_a) == set(cols_b):
        col_to_idx_a = {col: i for i, col in enumerate(cols_a)}
        col_to_idx_b = {col: j for j, col in enumerate(cols_b)}

        for i, col_a in enumerate(cols_a):
            for j, col_b in enumerate(cols_b):
                if i < j:
                    idx_b_in_a = col_to_idx_a.get(col_b)
                    idx_a_in_b = col_to_idx_b.get(col_a)

                    if idx_b_in_a is not None and idx_a_in_b is not None:
                        if not mask[i, j] and not mask[idx_b_in_a, idx_a_in_b]:
                            avg_dist = (matrix[i, j] + matrix[idx_b_in_a, idx_a_in_b]) / 2
                            matrix[i, j] = avg_dist

```

```

        matrix[idx_b_in_a, idx_a_in_b] = avg_dist
    if mask[i, j] or mask[idx_b_in_a, idx_a_in_b]:
        mask[i, j] = True
        mask[idx_b_in_a, idx_a_in_b] = True

    return (
        pd.DataFrame(matrix, index=cols_a, columns=cols_b),
        pd.DataFrame(mask, index=cols_a, columns=cols_b)
    )

```

class PairwiseDistanceCalculator:

```

    def __init__(self, analyzer: CSVAnalyzer, comparator: CompactRepresentationComparator):
        self.analyzer = analyzer
        self.comparator = comparator

    def _compute_column_type_distances(self, stocks_data: Dict[str, pd.DataFrame],
                                       cache=None) -> np.ndarray:
        tickers = sorted(stocks_data.keys())
        n = len(tickers)
        distance_matrix = np.zeros((n, n))

        cdrs = {
            ticker: self.analyzer.get_compact_representation(df, cache)
            for ticker, df in stocks_data.items()
        }

        for i in range(n):
            for j in range(i + 1, n):
                ticker_i, ticker_j = tickers[i], tickers[j]

                features_i = set(cdrs[ticker_i].keys())
                features_j = set(cdrs[ticker_j].keys())
                common_features = features_i.intersection(features_j)

                if not common_features:
                    distance_matrix[i, j] = 1.0
                    distance_matrix[j, i] = 1.0
                    continue

                type_mismatches = 0
                for feature in common_features:
                    type_i = cdrs[ticker_i][feature]["type"]
                    type_j = cdrs[ticker_j][feature]["type"]
                    if type_i != type_j:
                        type_mismatches += 1

```

```

        mismatch_ratio = type_mismatches / len(common_features)
        distance_matrix[i, j] = mismatch_ratio
        distance_matrix[j, i] = mismatch_ratio

    return distance_matrix

def _compute_diss_distances(self, stocks_data: Dict[str, pd.DataFrame],
                            cache=None) -> np.ndarray:
    tickers = sorted(stocks_data.keys())
    n = len(tickers)
    distance_matrix = np.zeros((n, n))

    cdrs = {
        ticker: self.analyzer.get_compact_representation(df, cache)
        for ticker, df in stocks_data.items()
    }

    for i in range(n):
        for j in range(i + 1, n):
            ticker_i, ticker_j = tickers[i], tickers[j]

            features_i = set(cdrs[ticker_i].keys())
            features_j = set(cdrs[ticker_j].keys())
            common_features = features_i.intersection(features_j)

            if not common_features:
                distance_matrix[i, j] = 1.0
                distance_matrix[j, i] = 1.0
                continue

            distances = []
            for feature in common_features:
                desc_i = cdrs[ticker_i][feature]
                desc_j = cdrs[ticker_j][feature]

                if desc_i["type"] == desc_j["type"]:
                    dist = self.comparator.calculate_distance(desc_i, desc_j)
                    distances.append(dist)

            if distances:
                avg_dist = np.mean(distances)
                distance_matrix[i, j] = avg_dist
                distance_matrix[j, i] = avg_dist
            else:
                distance_matrix[i, j] = 1.0
                distance_matrix[j, i] = 1.0

    return distance_matrix

```

```

def compute_pairwise_distances(self, stocks_data: Dict[str, pd.DataFrame],
                               cache=None) -> Tuple[np.ndarray, list]:
    tickers = sorted(stocks_data.keys())

    print(f"Обчислення попарних відстаней для {len(tickers)} акцій...")
    distance_matrix = self._compute_diss_distances(stocks_data, cache)

    is_symmetric = np.allclose(distance_matrix, distance_matrix.T, rtol=1e-10)
    if not is_symmetric:
        distance_matrix = (distance_matrix + distance_matrix.T) / 2

    return distance_matrix, tickers

def compute_pairwise_distances_with_slices(
    self,
    stocks_data: Dict[str, pd.DataFrame],
    slice_percentages: list,
    cache=None
) -> list:
    results = []

    for slice_pct in slice_percentages:
        print(f"Обчислення для підвибірки {slice_pct}%...")

        sliced_data = {}
        for ticker, df in stocks_data.items():
            n_rows = int(len(df) * slice_pct / 100)
            if n_rows < 5:
                continue
            sliced_data[f"{ticker}_{slice_pct}%"] = df.iloc[:n_rows]

        if len(sliced_data) < 2:
            continue

        matrices = {}

        print(f"- Column types distance...")
        matrices['col'] = self._compute_column_type_distances(sliced_data, cache)

        print(f"- DISS distance...")
        matrices['diss'] = self._compute_diss_distances(sliced_data, cache)

        print(f"- Hybrid distance...")
        matrices['hybrid'] = (matrices['col'] + matrices['diss']) / 2

        tickers = sorted(sliced_data.keys())
        results.append((slice_pct, matrices, tickers))

```

```
return results
```

```
class MSTBuilder:
```

```
    @staticmethod
    def build_mst(distance_matrix: np.ndarray, tickers: list) -> list:
        n = len(tickers)
        edges = []

        for i in range(n):
            for j in range(i + 1, n):
                if distance_matrix[i, j] > 0:
                    edges.append((distance_matrix[i, j], tickers[i], tickers[j]))

        edges.sort()

        parent = {ticker: ticker for ticker in tickers}

        def find(node):
            if parent[node] != node:
                parent[node] = find(parent[node])
            return parent[node]

        def union(node1, node2):
            root1 = find(node1)
            root2 = find(node2)
            if root1 != root2:
                parent[root2] = root1
                return True
            return False

        mst_edges = []
        for weight, ticker1, ticker2 in edges:
            if union(ticker1, ticker2):
                mst_edges.append((ticker1, ticker2, weight))
                if len(mst_edges) == n - 1:
                    break

        return mst_edges
```

```
// Вміст файлу quality_evaluator.py
```

```
Оцінка якості даних та детекція дрейфу
```

```
from typing import Dict, Optional
```

```

import numpy as np
import pandas as pd
from scipy.stats import wasserstein_distance, linregress
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, roc_curve
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

class DataQualityEvaluator:

    @staticmethod
    def calculate_hellinger(p: np.ndarray, q: np.ndarray) -> float:
        return np.sqrt(np.sum((np.sqrt(p) - np.sqrt(q)) ** 2)) / np.sqrt(2)

    def evaluate_drift_metrics(self, df_full: pd.DataFrame,
                              df_subset: pd.DataFrame) -> pd.DataFrame:
        results = []
        numeric_cols = df_full.select_dtypes(include=np.number).columns.intersection(
            df_subset.columns
        )

        for col in numeric_cols:
            full_data = df_full[col].dropna()
            subset_data = df_subset[col].dropna()

            if len(full_data) == 0 or len(subset_data) == 0:
                continue

            w_dist = wasserstein_distance(full_data, subset_data)

            bins = np.linspace(
                min(full_data.min(), subset_data.min()),
                max(full_data.max(), subset_data.max()),
                20
            )
            hist_full, _ = np.histogram(full_data, bins=bins, density=True)
            hist_subset, _ = np.histogram(subset_data, bins=bins, density=True)

            hist_full = hist_full / (np.sum(hist_full) + 1e-10) + 1e-10
            hist_subset = hist_subset / (np.sum(hist_subset) + 1e-10) + 1e-10

            h_dist = self.calculate_hellinger(hist_full, hist_subset)

            psi = np.sum((hist_subset - hist_full) * np.log(hist_subset / hist_full))

            results.append({
                "Колонка": col,

```

```

        "Вассерштейн": w_dist,
        "Геллінгер": h_dist,
        "PSI": psi
    })

    return pd.DataFrame(results)

def compute_domain_classifier_metrics(self, df_full: pd.DataFrame,
                                     df_subset: pd.DataFrame) -> Optional[Dict]:
    common_cols = df_full.columns.intersection(df_subset.columns)
    numeric_cols = df_full[common_cols].select_dtypes(include=np.number).columns

    if len(numeric_cols) == 0:
        return None

    subset_size = len(df_subset)
    df_full_only = df_full.iloc[subset_size:].copy()

    if len(df_full_only) < 10:
        return None

    X_full_only = df_full_only[numeric_cols].assign(target_label=0)
    X_subset = df_subset[numeric_cols].assign(target_label=1)
    X_combined = pd.concat([X_full_only, X_subset], ignore_index=True).dropna()

    if X_combined.empty or len(X_combined) < 20:
        return None

    y = X_combined["target_label"].values
    X = StandardScaler().fit_transform(
        X_combined.drop(columns=["target_label"]).values
    )

    if len(np.unique(y)) < 2 or np.min(np.bincount(y)) < 5:
        return None

    try:
        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=0.3, random_state=42, stratify=y
        )

        clf = RandomForestClassifier(
            n_estimators=100,
            random_state=42,
            max_depth=5,
            n_jobs=-1
        )
        clf.fit(X_train, y_train)

```

```

y_probs = clf.predict_proba(X_test)[:, 1]
y_pred = clf.predict(X_test)
fpr, tpr, _ = roc_curve(y_test, y_probs)

return {
    "Точність": accuracy_score(y_test, y_pred),
    "F1_Оцінка": f1_score(y_test, y_pred),
    "AUC_ROC": roc_auc_score(y_test, y_probs),
    "FPR": fpr,
    "TPR": tpr
}
except Exception as e:
    print(f'Помилка класифікатора: {e}')
    return None

```

class DriftAnalyzer:

```

def __init__(self, analyzer, comparator):
    self.analyzer = analyzer
    self.comparator = comparator

def compute_drift_with_subsampling(self, df_full: pd.DataFrame,
                                   n_points: int = 10, cache=None) -> Dict:
    fractions = np.linspace(0.1, 1.0, n_points)
    avg_diss_values = []

    cdr_full = self.analyzer.get_compact_representation(df_full, cache)

    for frac in fractions:
        if np.isclose(frac, 1.0, atol=0.01):
            avg_diss_values.append(0.0)
            continue

        n_samples = int(len(df_full) * frac)
        if n_samples < 5:
            avg_diss_values.append(np.nan)
            continue

        df_subset = df_full.iloc[:n_samples]
        cdr_subset = self.analyzer.get_compact_representation(df_subset, cache)
        diss_matrix, mask = self.comparator.build_diss_matrix(cdr_subset, cdr_full)

        valid = diss_matrix.values[~mask.values]
        avg_diss_values.append(np.mean(valid) if len(valid) > 0 else np.nan)

    valid_idx = ~np.isnan(avg_diss_values)

```

```
trend = None
```

```
if np.sum(valid_idx) >= 2:  
    x_valid = fractions[valid_idx]  
    y_valid = np.array(avg_diss_values)[valid_idx]  
    slope, intercept, r_value, p_value, _ = linregress(x_valid, y_valid)
```

```
trend = {  
    'нахил': slope,  
    'перетин': intercept,  
    'r_квадрат': r_value ** 2,  
    'p_значення': p_value  
}
```

```
return {  
    'дробові': fractions,  
    'середня_diss': avg_diss_values,  
    'тренд': trend  
}
```

```
def compute_f1_vs_threshold(self, df_full: pd.DataFrame,  
                             n_windows: int = 50, cache=None) -> Optional[Dict]:
```

```
    window_size = max(len(df_full) // n_windows, 10)
```

```
    diss_values = []
```

```
    true_labels = []
```

```
    cdr_full = self.analyzer.get_compact_representation(df_full, cache)
```

```
    for i in range(n_windows):
```

```
        start_idx = i * window_size
```

```
        end_idx = min((i + 1) * window_size, len(df_full))
```

```
        if end_idx - start_idx < 5:
```

```
            break
```

```
        window_data = df_full.iloc[start_idx:end_idx]
```

```
        cdr_window = self.analyzer.get_compact_representation(window_data, cache)
```

```
        diss_matrix, mask = self.comparator.build_diss_matrix(cdr_window, cdr_full)
```

```
        valid = diss_matrix.values[~mask.values]
```

```
        if len(valid) > 0:
```

```
            diss_values.append(np.mean(valid))
```

```
            true_labels.append(1 if i >= n_windows // 2 else 0)
```

```
    if len(diss_values) < 10:
```

```
        return None
```

```

diss_values = np.array(diss_values)
true_labels = np.array(true_labels)
thresholds = np.linspace(diss_values.min(), diss_values.max(), 100)
f1_scores = []

for threshold in thresholds:
    predictions = (diss_values > threshold).astype(int)
    if len(np.unique(predictions)) > 1:
        f1_scores.append(f1_score(true_labels, predictions))
    else:
        f1_scores.append(0.0)

f1_scores = np.array(f1_scores)
optimal_idx = np.argmax(f1_scores)

optimal_predictions = (diss_values > thresholds[optimal_idx]).astype(int)
optimal_accuracy = accuracy_score(true_labels, optimal_predictions)

return {
    'пороги': thresholds,
    'f1_оцінки': f1_scores,
    'оптимальний поріг': thresholds[optimal_idx],
    'оптимальний f1': f1_scores[optimal_idx],
    'оптимальна точність': optimal_accuracy
}

def compute_local_drift_slopes(self, stocks_data: Dict[str, pd.DataFrame],
                               n_points: int = 10, cache=None) -> Dict:
    results = {}

    for ticker, df_full in stocks_data.items():
        print(f"Обчислення локальних нахилів для {ticker}...")

        fractions = np.linspace(0.1, 1.0, n_points)
        diss_values = []

        cdr_full = self.analyzer.get_compact_representation(df_full, cache)

        for frac in fractions:
            if np.isclose(frac, 1.0, atol=0.01):
                diss_values.append(0.0)
                continue

            n_samples = int(len(df_full) * frac)
            if n_samples < 5:
                diss_values.append(np.nan)
                continue

```

```

df_subset = df_full.iloc[:n_samples]
cdr_subset = self.analyzer.get_compact_representation(df_subset, cache)
diss_matrix, mask = self.comparator.build_diss_matrix(cdr_subset, cdr_full)

valid = diss_matrix.values[~mask.values]
diss_values.append(np.mean(valid) if len(valid) > 0 else np.nan)

slopes = []
for i in range(len(diss_values) - 1):
    if not np.isnan(diss_values[i]) and not np.isnan(diss_values[i + 1]):
        slope = abs(diss_values[i + 1] - diss_values[i]) / (fractions[i + 1] - fractions[i])
        slopes.append(slope)
    else:
        slopes.append(np.nan)

midpoints = [(fractions[i] + fractions[i + 1]) / 2 for i in range(len(fractions) - 1)]

results[ticker] = {
    'fractions': fractions,
    'midpoints': midpoints,
    'diss_values': diss_values,
    'slopes': slopes
}

return results

```

// Вміст файлу pipeline.py

Оркестратор аналітичного конвеєра

```

import matplotlib.pyplot as plt
from typing import Dict

from config import AnalysisConfig, OutputManager
from data_loader import CDRCache, KaggleDataLoader, DataFetcher
from analysis_core import (
    CSVAnalyzer, CompactRepresentationComparator,
    PairwiseDistanceCalculator, MSTBuilder
)
from quality_evaluator import DataQualityEvaluator, DriftAnalyzer
from visualization_base import VisualizationSetup
from heatmap_visualizer import HeatmapVisualizer
from drift_visualizer import DriftVisualizer
from classification_visualizer import ClassificationMetricsVisualizer
from graph_visualizer import GraphVisualizer

```

```

class AnalysisPipeline:

```

```

def __init__(self, config: AnalysisConfig):
    self.config = config
    self.output_manager = OutputManager()

    self.cdr_cache = CDRCache()

    if config.data_source == "kaggle":
        self.kaggle_loader = KaggleDataLoader(config.kaggle_paths)
    else:
        self.kaggle_loader = None

    self.data_fetcher = DataFetcher(config, self.kaggle_loader)

    self.analyzer = CSVAnalyzer()
    self.comparator = CompactRepresentationComparator()
    self.evaluator = DataQualityEvaluator()
    self.drift_analyzer = DriftAnalyzer(self.analyzer, self.comparator)
    self.distance_calculator = PairwiseDistanceCalculator(self.analyzer, self.comparator)
    self.mst_builder = MSTBuilder()

    self.heatmap_viz = HeatmapVisualizer(config.data_source)
    self.drift_viz = DriftVisualizer(config.data_source)
    self.classification_viz = ClassificationMetricsVisualizer(config.data_source)
    self.graph_viz = GraphVisualizer(config.data_source)

    self.stocks_data: Dict = {}
    self.results = {
        'heatmaps': {},
        'sensitivity': {},
        'drift_linear': {},
        'fl_threshold': {},
        'roc_curves': {}
    }

def setup(self):
    VisualizationSetup.setup_plot_style()

    print(f'\n{'=' * 100}')

    stocks = (list(self.config.kaggle_paths.keys())
              if self.config.data_source == "kaggle"
              else self.config.yahoo_stocks)

    print(f'Акції: {' '.join(stocks)}')
    print(f'Період: {self.config.start_date} до {self.config.end_date}')
    print(f'\n{'=' * 100}')

```

```

def load_data(self):
    stocks = (list(self.config.kaggle_paths.keys())
              if self.config.data_source == "kaggle"
              else self.config.yahoo_stocks)

    for ticker in stocks:
        print(f"\n{'-' * 60}\n Завантаження {ticker}...\n{'-' * 60}")
        try:
            df_full = self.data_fetcher.fetch(
                ticker,
                self.config.start_date,
                self.config.end_date
            )

            if df_full.empty:
                print(f"Пропуск {ticker}\n")
                continue

            self.stocks_data[ticker] = df_full
            print(f'{ticker}: {len(df_full)} рядків')

        except Exception as e:
            print(f"Помилка {ticker}: {e}")

def analyze_individual_stocks(self):
    for ticker in self.stocks_data.keys():
        print(f"\n{'-' * 60}\n Аналіз {ticker}...\n{'-' * 60}")
        df_full = self.stocks_data[ticker]

        try:
            df_half = df_full.iloc[:len(df_full) // 2]
            cdr_full = self.analyzer.get_compact_representation(df_full, self.cdr_cache)
            cdr_half = self.analyzer.get_compact_representation(df_half, self.cdr_cache)
            diss, mask = self.comparator.build_diss_matrix(cdr_half, cdr_full)
            self.results['heatmaps'][ticker] = {"matrix": diss, "mask": mask}

            self.results['roc_curves'][ticker] = self.evaluator.compute_domain_classifier_metrics(
                df_full, df_half
            )
            roc_result = self.results['roc_curves'][ticker]
            if roc_result:
                print(f"Точність: {roc_result['Точність']:.4f}")
            drift_result = self.drift_analyzer.compute_drift_with_subsampling(
                df_full, n_points=self.config.n_sensitivity_points, cache=self.cdr_cache
            )
            self.results['sensitivity'][ticker] = drift_result

            self.results['drift_linear'][ticker] = drift_result

```

```

        self.results['f1_threshold'][ticker] = self.drift_analyzer.compute_f1_vs_threshold(
            df_full, n_windows=self.config.n_f1_windows, cache=self.cdr_cache
        )
        f1_result = self.results['f1_threshold'][ticker]
        if f1_result:
            print(f"Точність детектора дрейфу: {f1_result['оптимальна_точність']:.4f} (F1:
{f1_result['оптимальний_f1']:.4f})")

        print(f"{ticker} завершено!")

    except Exception as e:
        print(f"Помилка {ticker}: {e}")
        import traceback
        traceback.print_exc()

def create_visualizations(self):
    if self.results['heatmaps']:
        self.heatmap_viz.plot_combined_heatmaps(
            self.results['heatmaps'], self.output_manager
        )

    if self.stocks_data:
        self.heatmap_viz.plot_control_heatmaps_full_vs_full(
            self.stocks_data, self.analyzer, self.comparator, self.output_manager
        )

    if self.stocks_data:
        self.heatmap_viz.plot_unified_heatmap_with_cross_comparisons(
            self.analyzer, self.stocks_data, self.output_manager
        )

    if self.stocks_data:
        self.heatmap_viz.plot_pairwise_heatmaps(
            self.analyzer, self.stocks_data, self.output_manager
        )

    if self.results['roc_curves']:
        self.classification_viz.plot_combined_roc_curves(
            self.results['roc_curves'], self.output_manager
        )

    if self.results['sensitivity']:
        import numpy as np
        slices_pct = np.linspace(10, 100, 10)
        self.drift_viz.plot_combined_sensitivity(
            self.results['sensitivity'], slices_pct, self.output_manager
        )

```

```

if self.results['drift_linear']:
    self.drift_viz.plot_drift_with_parametric_curve(
        self.results['drift_linear'], self.output_manager
    )

if self.results['f1_threshold']:
    self.classification_viz.plot_f1_vs_threshold(
        self.results['f1_threshold'], self.output_manager
    )

if len(self.stocks_data) >= 2:
    drift_slopes = self.drift_analyzer.compute_local_drift_slopes(
        self.stocks_data,
        n_points=self.config.n_sensitivity_points,
        cache=self.cdr_cache
    )
    self.drift_viz.plot_local_drift_slopes(drift_slopes, self.output_manager)

    slice_results = self.distance_calculator.compute_pairwise_distances_with_slices(
        self.stocks_data,
        self.config.slice_percentages,
        self.cdr_cache
    )

    if slice_results:
        self.graph_viz.plot_similarity_graph_and_mst(
            slice_results,
            self.mst_builder,
            self.output_manager,
            show_all_edges_above=0.10,
        )

    if slice_results:
        self.graph_viz.plot_similarity_graph_and_mst(
            slice_results, self.mst_builder, self.output_manager
        )

        self.graph_viz.plot_sliced_graphs_combined(
            slice_results, self.mst_builder, self.output_manager
        )

        self.graph_viz.create_summary_statistics_plot(
            slice_results, self.mst_builder, self.output_manager
        )

def cleanup(self):

```

```

self.cdr_cache.clear()
plt.close('all')

def print_summary(self):
    print(f"\n{'=' * 100}")
    print(f"ОПТИМІЗОВАНИЙ АНАЛІЗ ЗАВЕРШЕНО!")
    print(f"{'=' * 100}")
    print(f"Результати: {self.output_manager.output_dir}/")
    print(f"Час: {self.output_manager.timestamp}")

def run(self):
    try:
        self.setup()
        self.load_data()

        if not self.stocks_data:
            print(" Немає даних для аналізу!")
            return

        self.analyze_individual_stocks()
        self.create_visualizations()
        self.print_summary()

    finally:
        self.cleanup()

def run_full_analysis():
    config = AnalysisConfig(
        data_source="kaggle",
        start_date="2015-01-01",
        end_date="2024-01-01"
    )

    pipeline = AnalysisPipeline(config)
    pipeline.run()

if __name__ == "__main__":
    run_full_analysis()

// Вміст файлу main.py
Точка входу

from config import AnalysisConfig
from pipeline import AnalysisPipeline

```

```

def main():

    config = AnalysisConfig(
        data_source="kaggle",
        kaggle_paths={
            "AAPL": "data/AAPL.csv",
            "TSLA": "data/TSLA.csv"
        },
        yahoo_stocks=["AAPL", "NVDA", "MSFT"],
        start_date="2015-01-01",
        end_date="2024-01-01",
        n_sensitivity_points=10,
        n_fl_windows=50,
        slice_percentages=[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
    )

    pipeline = AnalysisPipeline(config)
    pipeline.run()

if __name__ == "__main__":
    main()

```