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

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
Кафедра математичних методів системного аналізу

До захисту допущено:
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2026 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Модель підтримки інвестиційних рішень на основі багатокритеріальної оптимізації»

Виконав:
студент IV курсу, групи КА-21
Скидан Іван Олегович _____

Керівник:
Асистент кафедри ММСА,
Древаль Максим Михайлович _____

Консультант з економічної частини:
доцент кафедри ЕК, к.е.н., доц.
Рощина Надія Василівна _____

Консультант з нормоконтролю:
к.ф.-м.н. Статкевич Віталій Михайлович _____

Рецензент:

_____

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

Київ – 2026

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Оксана ТИМОЩУК
«__» _____ 2026 р.

ЗАВДАННЯ
на дипломну роботу студенту
Скидану Івану Олеговичу

1. Тема роботи «Модель підтримки інвестиційних рішень на основі багатокритеріальної оптимізації», керівник роботи асистент кафедри ММСА Древаль Максим Михайлович, затверджені наказом по університету №1944-с від 27.05.2026 р.
2. Термін подання студентом роботи 10.06.2026.
3. Вихідні дані до роботи: ретроспективні ринкові дані (ціни закриття та обсяги торгів фінансових активів), методи багатокритеріального аналізу рішень (MCDM), метод зваженої суми (WSM) та концепція Парето-оптимальності.
4. Зміст роботи: аналіз предметної області та огляд існуючих підходів до підтримки інвестиційних рішень; обґрунтування системи критеріїв оцінювання фінансових активів; розробка математичної моделі на основі методу зваженої суми та Парето-оптимальності; програмна реалізація системи підтримки прийняття рішень з інтерактивним інтерфейсом; верифікація моделі методом бектестингу на реальних ринкових даних та аналіз результатів.
5. Перелік ілюстративного матеріалу: класифікація методів MCDM, концепція Парето-фронту у просторі критеріїв «ризик-дохідність», блок-схема алгоритму методу зваженої суми, концепція домінування та Парето-оптимальних рішень,

схема часової структури бектестингу, порівняльні таблиці програмних рішень та методів MCDM, графіки накопиченої дохідності WSM-портфеля відносно бенчмарку, презентація до виступу.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	завдання прийняв
Економічний	Рощина Н.В., доц.		

7. Дата видачі завдання: 15 квітня 2026 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Об'єктний аналіз предметної області і постановка завдання	15.04.2026	Виконано
2.	Вибір і обґрунтування моделі із доцільною архітектурою	22.04.2026	Виконано
3.	Розробка математичної моделі	29.04.2026	Виконано
4.	Програмна реалізація моделі	01.05.2026	Виконано
5.	Верифікація розробленої моделі методом бектестингу	06.05.2026	Виконано
6.	Оформлення пояснювальної записки	15.05.2026	Виконано
7.	Підготовка слайдів презентації	23.05.2026	Виконано
8.	Підготовка демонстраційної версії розробленого програмного продукту	24.05.2026	Виконано
9.	Написання доповіді	27.05.2026	Виконано

Студент _____ Іван СКИДАН

Керівник роботи _____ Максим ДРЕВАЛЬ

РЕФЕРАТ

Дипломна робота: 98 с., 21 рис., 13 табл., 2 дод., 34 джерела.

ІНВЕСТИЦІЙНІ РІШЕННЯ, БАГАТОКРИТЕРІАЛЬНА ОПТИМІЗАЦІЯ, MCDM, WSM, ПАРЕТО-ОПТИМАЛЬНІСТЬ, РИЗИК, ДОХІДНІСТЬ, ЛІКВІДНІСТЬ, БЕКТЕСТИНГ, ARIMA, PYTHON, STREAMLIT.

Об’єкт дослідження – процес прийняття інвестиційних рішень на фінансовому ринку.

Предмет дослідження – методи багатокритеріальної оптимізації для оцінювання фінансових активів.

Мета роботи – розробити та реалізувати модель підтримки інвестиційних рішень на основі WSM і Парето-оптимальності та перевірити її методом бектестингу.

Методи дослідження – багатокритеріальний аналіз рішень, WSM, Парето-оптимальність, нормалізація критеріїв, міри ризику, ARIMA-моделювання, бектестинг, Python та Streamlit.

Актуальність роботи зумовлена необхідністю врахування кількох суперечливих критеріїв під час прийняття інвестиційних рішень, зокрема дохідності, ризику, ліквідності та інвестиційного горизонту.

Результати роботи – розроблено математичну модель і програмну систему з інтерактивним інтерфейсом. Система дозволяє завантажувати фінансові дані, розраховувати критерії активів, формувати WSM-рейтинг, визначати Парето-оптимальні альтернативи, виконувати ARIMA-прогнозування та проводити бектестинг інвестиційної стратегії.

Подальший розвиток роботи передбачає розширення набору активів, додавання нових критеріїв, урахування транзакційних витрат та інтеграцію складніших моделей прогнозування.

ABSTRACT

Diploma thesis: 98 pages, 21 figures, 13 tables, 2 appendices, 34 references.
INVESTMENT DECISIONS, MULTI-CRITERIA OPTIMIZATION, MCDM, WSM, PARETO OPTIMALITY, RISK, RETURN, LIQUIDITY, BACKTESTING, ARIMA, PYTHON, STREAMLIT.

Object of study – the process of investment decision-making under multi-criteria uncertainty in the financial market.

Subject of study is the set of of multi-criteria optimization used to form investment recommendations based on quantitative criteria for evaluating financial assets.

Aim of the thesis – to develop and implement a decision-support model for investment decisions based on WSM and Pareto optimality, and to verify this model using real market data through backtesting.

Methods of research – multi-criteria decision-making methods, WSM, Pareto optimality, criteria normalization, risk measures, ARIMA modeling, backtesting, Python, and Streamlit.

Relevance – the work is relevant due to the need to consider several conflicting criteria in investment decision-making, including return, risk, liquidity, and investment horizon.

Results of the thesis – a mathematical model and an interactive software system were developed. The system allows users to load financial data, calculate asset criteria, form a WSM ranking, identify Pareto-optimal alternatives, perform ARIMA forecasting, and conduct backtesting.

Further development involves expanding the set of assets, adding new criteria, considering transaction costs, and integrating more complex forecasting models.

ЗМІСТ

ВСТУП

Сучасний фінансовий ринок характеризується зростаючою складністю, підвищеною волатильністю та постійним збільшенням кількості доступних інструментів для інвестування. У таких умовах прийняття обґрунтованих інвестиційних рішень стає дедалі складнішим завданням для будь-якого учасника ринку – як для інституційного інвестора, так і для приватної особи. Ключова складність полягає у тому, що задача вибору інвестиційного портфеля є принципово багатокритеріальною: необхідно одночасно враховувати очікувану дохідність, рівень ризику, ліквідність активів та індивідуальний горизонт інвестування, які нерідко суперечать один одному.

Традиційні підходи, що зводять задачу до оптимізації одного критерію – як правило, дохідності або ризику, – не дозволяють адекватно відобразити реальні вподобання інвестора. Класична двокритеріальна теорія портфеля Марковіца (1952), незважаючи на фундаментальне значення, не охоплює ліквідності та часового горизонту. Комерційні аналітичні системи, в свою чергу, є дорогими, закритими і непроз. Відкриті бібліотеки математичного моделювання вимагають глибокої технічної підготовки. Це формує запит на доступний, відкритий і прозорий інструмент підтримки інвестиційних рішень, що враховує повний спектр релевантних критеріїв. У даній роботі таким інструментом виступає модель на основі методу зваженої суми (Weighted sum method, скорочено WSM) та Парето-оптимальності з верифікацією на реальних ринкових даних.

Актуальність дослідження обумовлена необхідністю розробки методологічно обґрунтованої та практично застосовної моделі підтримки прийняття інвестиційних рішень, яка поєднує математичний апарат багатокритеріального прийняття рішень (Multi-Criteria Decision Making, скорочено MCDM) з верифікацією на реальних ринкових даних. Розвиток таких систем відповідає сучасним тенденціям у фінансовій аналітиці –

переходу від однокритеріальних моделей до комплексних багатокритеріальних систем підтримки рішень.

Об'єктом дослідження є процес прийняття інвестиційних рішень в умовах багатокритеріальної невизначеності на фінансовому ринку.

Предметом дослідження є методи та моделі багатокритеріальної оптимізації, що застосовуються при формуванні інвестиційного портфеля за системою кількісних критеріїв оцінювання фінансових активів.

Мета роботи – розробити та програмно реалізувати модель підтримки прийняття інвестиційних рішень на основі методу зваженої суми (WSM) та Парето-оптимальності з верифікацією на реальних фінансових даних методом бектестингу.

Задачі дослідження:

1. Провести аналіз предметної галузі та огляд існуючих підходів і програмних рішень у сфері підтримки інвестиційних рішень.
2. Визначити та обґрунтувати систему критеріїв оцінювання інвестиційних альтернатив – дохідність, ризик, ліквідність та інвестиційний горизонт.
3. Розробити математичний апарат: нормалізацію критеріїв, метод зваженої суми, алгоритм побудови Парето-фронту та міри оцінювання ризику.
4. Реалізувати програмну модель підтримки прийняття рішень з інтерактивним інтерфейсом.
5. Провести верифікацію моделі методом бектестингу на реальних ринкових даних та проаналізувати результати.

Методи дослідження: теорія багатокритеріального аналізу рішень, методи математичної статистики та кількісних фінансів, метод зваженої суми, алгоритм побудови Парето-фронту, методи прогнозування часових рядів та ретроспективного тестування (бектестинг).

Практичне значення: розроблена модель є відкритим і доступним інструментом для інвесторів будь-якого рівня підготовки. Вона дозволяє у

режимі реального часу досліджувати вплив вагових пріоритетів на ранжування активів, аналізувати компроміси між критеріями через Парето-фронт, прогнозувати ціни активів на основі моделі ARIMA(1,1,1) та верифікувати рекомендації методом бектестингу на реальних ринкових даних.

РОЗДІЛ 1 ПРЕДМЕТНА ОБЛАСТЬ І ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз сучасного стану досліджень у сфері підтримки прийняття інвестиційних рішень

Задача формування інвестиційного портфеля є однією з найбільш досліджуваних у галузі кількісних фінансів. Незважаючи на значну кількість наукових праць, присвячених даній тематиці, проблема підтримки прийняття інвестиційних рішень не втрачає своєї актуальності – передусім тому, що реальні умови ринку постійно ускладнюються, а вимоги інвесторів до якості рекомендацій зростають. Сучасні фінансові ринки характеризуються підвищеною волатильністю, тісними кореляційними зв'язками між активами різних класів, а також значним інформаційним навантаженням, що унеможливлює прийняття обґрунтованих рішень без залучення формальних математичних методів.

Теоретичне підґрунтя сучасного підходу до аналізу інвестицій було закладено задовго до появи потужних обчислювальних засобів. Ще на початку XX століття інвестиційні рішення ухвалювались переважно на основі якісних оцінок: репутація емітента, галузева приналежність, доступна фінансова звітність. Такий підхід, незважаючи на практичну цінність у свою епоху, не давав змоги кількісно порівнювати альтернативи та оцінювати сукупний ризик портфеля як цілісної системи активів.

Якісний перехід відбувся у 1952 році, коли Гаррі Марковіц опублікував фундаментальну працю «Portfolio Selection» у Journal of Finance [1]. У ній вперше була формалізована ідея диверсифікації: доведено, що ризик портфеля залежить не лише від ризиків окремих активів, але й від кореляцій між ними. Саме завдяки цьому інсайту стало можливим кількісно описати ефект диверсифікації та ввести поняття ефективної межі (efficient frontier) – множини портфелів, що забезпечують максимальну дохідність за заданого рівня ризику

або мінімальний ризик при заданій дохідності. Ця концепція стала відправною точкою для всієї подальшої теорії портфельного інвестування, а Марковіц у 1990 році отримав Нобелівську премію з економіки.

Продовжуючи ідеї Марковіца, Вільям Шарп у 1964 році розробив модель оцінки капітальних активів – Capital Asset Pricing Model (CAPM) [2]. У рамках цієї моделі очікувана дохідність активу лінійно залежить від його систематичного ризику (коефіцієнт бета), що дозволило суттєво спростити обчислення оптимального портфеля, знизивши розмірність задачі до одного ринкового фактора. Незважаючи на суттєві спрощуючі припущення – ефективний ринок, відсутність транзакційних витрат, однорідні очікування інвесторів, – CAPM набула широкого практичного застосування і досі використовується як базовий інструмент оцінки ризику та дохідності активів.

Подальший розвиток теорії привів до мультифакторних моделей. Теорія арбітражного ціноутворення (Arbitrage Pricing Theorem APT), запропонована Стівеном Россом, пояснює дохідність активу через кілька систематичних факторів, що дозволяє більш гнучко враховувати джерела ризику [3]. На цій основі Юджин Фама (Eugene F. Fama) та Кеннет Френч (Kenneth R. French) розробили трифакторну та п'ятифакторну моделі, які стали стандартом кількісного аналізу акцій у сучасній науці. Проте навіть ці моделі залишаються принципово однокритеріальними з точки зору оптимізації – вони оцінюють ризик і дохідність, але не враховують ліквідності та часового горизонту як самостійних цілей оптимізації.

Усвідомлення цього обмеження зумовило розвиток окремого напряму – застосування методів теорії багатокритеріального прийняття рішень (MCDM) до задач портфельного інвестування. У роботі [4] детально проаналізовано, як методи MCDM можуть бути інтегровані в задачу портфельної оптимізації. Автори переконливо демонструють, що реальний вибір інвестора не вичерпується парою «дохідність – ризик»: не менш важливими є ліквідність, податкові наслідки, соціальна відповідальність інвестицій та індивідуальний

горизонт планування. Ці фактори принципово неможливо врахувати в рамках класичного підходу.

В українській науковій школі питання кількісного аналізу ризику та підтримки прийняття рішень також активно досліджуються. Вітлінський В.В. та Великоіваненко Г.І. у монографії «Ризикологія в економіці та підприємництві» [9] систематизували теоретичні та прикладні аспекти ризик-менеджменту стосовно вітчизняного фінансового ринку. Розвиток систем підтримки прийняття рішень знайшов відображення і в дослідженнях КПІ ім. Ігоря Сікорського: Бідюк П.І. та Коршевнюк Л.О. у навчальному посібнику, розробленому на кафедрі математичних методів системного аналізу, заклали теоретичні основи проектування комп'ютерних систем підтримки прийняття рішень [10].

У 2022 році Васкез, Ескобар та Манотаз [5] запропонували методологію прийняття рішень на фондовому ринку на основі комбінованого підходу AHP–TOPSIS, верифіковану на реальних даних колумбійського ринку. У роботі Лакшмі та Удая Кумара [6] розроблено підхід на основі randomized weighted fuzzy AHP та TOPSIS для вибору оптимального фондового портфеля. Серед найновіших публікацій варто відзначити роботу 2025 року у Scientific Reports [7], де запропоновано інтегровану модель на основі TOPSIS та ARAS у поєднанні з генетичним алгоритмом. Джинг та співавтори [8] також розглядають задачу оптимального вибору фондових портфелів із використанням методів багатокритеріального прийняття рішень, що підтверджує актуальність MCDM-підходу для інвестиційних задач.

Таким чином, аналіз сучасного стану досліджень свідчить про чітку тенденцію до переходу від однокритеріальних фінансових моделей до комплексних багатокритеріальних систем підтримки інвестиційних рішень, що підтверджується як зарубіжними [4-8], так і вітчизняними дослідженнями [9-12].

1.2 Огляд існуючих підходів та програмних рішень

Наявні підходи до підтримки інвестиційних рішень можна умовно поділити на три категорії: класичні однокритеріальні аналітичні моделі, сучасні методи MCDM та готові програмні рішення. Класичні моделі, попри теоретичну вишуканість, мають низку суттєвих практичних обмежень: середньодисперсійна оптимізація [1] потребує точного знання матриці коваріацій і не враховує ліквідності та часового горизонту. Матвійчук А.В. [12] зазначає, що застосування класичних моделей без урахування специфіки локального ринку призводить до значних похибок у рекомендаціях.

На корпоративному рівні домінують закриті комерційні платформи. Bloomberg Terminal є де-факто стандартом у галузі: він надає доступ до ринкових даних у реальному часі та розширеного аналітичного функціоналу. Проте вартість підписки (орієнтовно 2 000 USD на місяць) робить цей інструмент недоступним для більшості інвесторів, а методологія розрахунків є непрозорою. Morningstar Direct та FactSet пропонують подібний функціонал, але зберігають ту саму проблему закритості та нетранспарентності. Відкриті бібліотеки Python – PyPortfolioOpt, pymoo, DEAP – є гнучкими, але вимагають від користувача глибокого знання алгоритмів. Порівняльний аналіз основних рішень наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз існуючих програмних рішень

Рішення	Критерії	Відкритий код	Гнучкість	Поріг входу	Доступність
Bloomberg Terminal	2-3	Ні	Низька	Низький	~2 000 USD/міс.
Morningstar Direct	3-4	Ні	Середня	Середній	Комерційна
FactSet	3-5	Ні	Середня	Середній	Комерційна
PyPortfolioOpt	2	Так	Середня	Високий	Безкоштовна
pymoo	N	Так	Висока	Дуже високий	Безкоштовна
Розроблювана модель	4	Так	Висока	Низький	Безкоштовна

З таблиці 1.1 видно, що жодне з наявних рішень не поєднує відкритість коду, підтримку чотирьох критеріїв і низький поріг входу для кінцевого користувача. Саме цей пробіл і заповнює розроблювана модель.

1.3 Обґрунтування та аналіз критеріїв прийняття інвестиційних рішень

Побудова будь-якої багатокритеріальної моделі починається з відповіді на ключове питання: які саме критерії повинні бути включені до системи оцінювання? Надто велика кількість критеріїв ускладнює інтерпретацію результатів, а надто мала не відображає реальних вподобань інвестора. Аналіз наукових праць [4-8] дозволяє виділити чотири фактори, що найчастіше фігурують як ключові характеристики інвестиційних альтернатив: дохідність, ризик, ліквідність та інвестиційний горизонт.

1.3.1 Дохідність

Дохідність є первинним мотивом будь-якого інвестування. Включення цього критерію до моделі є абсолютно очевидним рішенням – без нього вся задача втрачає сенс. Проста дохідність визначається як відношення зміни ціни до початкової ціни. На практиці широко застосовуються логарифмічні дохідності, перевага яких полягає в адитивності в часі та кращій апроксимації нормальним розподілом [13, 14]. Дохідність портфеля є зваженою сумою дохідностей складових активів. У роботі як міра дохідності використовується середньоарифметична щоденна дохідність за вибраний ретроспективний горизонт.

1.3.2 Ризик

Ризик є невід'ємним супутником дохідності. Найбільш поширеною мірою є стандартне відхилення дохідності (волатильність). Поряд з нею застосовуються міри хвостового ризику Value at Risk (VaR) і Conditional Value at Risk (CVaR), які задовольняють аксіомам когерентності та є субадитивними [15]. Вітлінський В.В. та Великоіваненко Г.І. [9] наголошують, що вибір адекватної міри ризику є критичним для якості рішень. У даній роботі основною мірою ризику є волатильність, а VaR – додатковим показником при бектестуванні. Методи кількісного оцінювання ризику, зокрема статистичні та економетричні підходи, також розглядаються у працях з математичного ризик-менеджменту [11].

1.3.3 Ліквідність

Ліквідність – здатність активу бути конвертованим у грошові кошти без суттєвої втрати вартості – є критерієм, який нерідко недооцінюється у теоретичних моделях, але має визначальне практичне значення. Васкез, Ескобар та Манотаз [5] включають ліквідність до системи критеріїв як один із трьох ключових вимірів інвестиційної задачі, зазначаючи, що нехтування нею призводить до формування портфелів, неможливих для практичної реалізації без суттєвих транзакційних витрат. Кількісно ліквідність вимірюється середньоденним обсягом торгів (ADV), відносним спредом bid-ask або коефіцієнтом Аміхуда.

1.3.4 Інвестиційний горизонт

Інвестиційний горизонт є найбільш «персональним» з усіх чотирьох критеріїв. Традиційно виділяють три типи: короткостроковий (до 1 року), середньостроковий (1-5 років) та довгостроковий (понад 5 років). При короткому горизонті критично важливі ліквідність і захист від просадок; при довгому – допустимий більший ризик в розрахунку на вищу дохідність. Горизонт виконує роль контекстного фільтра, що коригує вимоги до всіх інших критеріїв.

Отже, система з чотирьох критеріїв є мінімально необхідною і достатньою для адекватного відображення реальних інвестиційних вподобань, а кожен із критеріїв має чітке теоретичне обґрунтування [1, 5, 9, 15].

1.4 Класифікація та порівняльний аналіз методів багатокритеріального аналізу

Метод MCDM як наукова дисципліна сформувався у 1960-1970-х роках. Ерготт у монографії [17, 34] виділяє три основні парадигми MCDM: методи згортки, методи на основі Парето-оптимальності та методи ранжування. Загальну класифікацію методів показано на рисунку 1.1.

Метод зваженої суми (WSM) є найпоширенішим методом агрегації: кожному критерію присвоюється ваговий коефіцієнт, значення нормалізуються, а результатом є зважена сума. Кім та Де Век [18] зазначають, що WSM є найбільш вживаним у практичних задачах оптимізації портфеля. Підхід Парето-оптимальності формує множину ефективних (недомінованих) альтернатив, де покращення за будь-яким критерієм неможливе без погіршення за іншим [17, 34]. Концепцію Парето-фронту ілюстровано на рисунку 1.2.

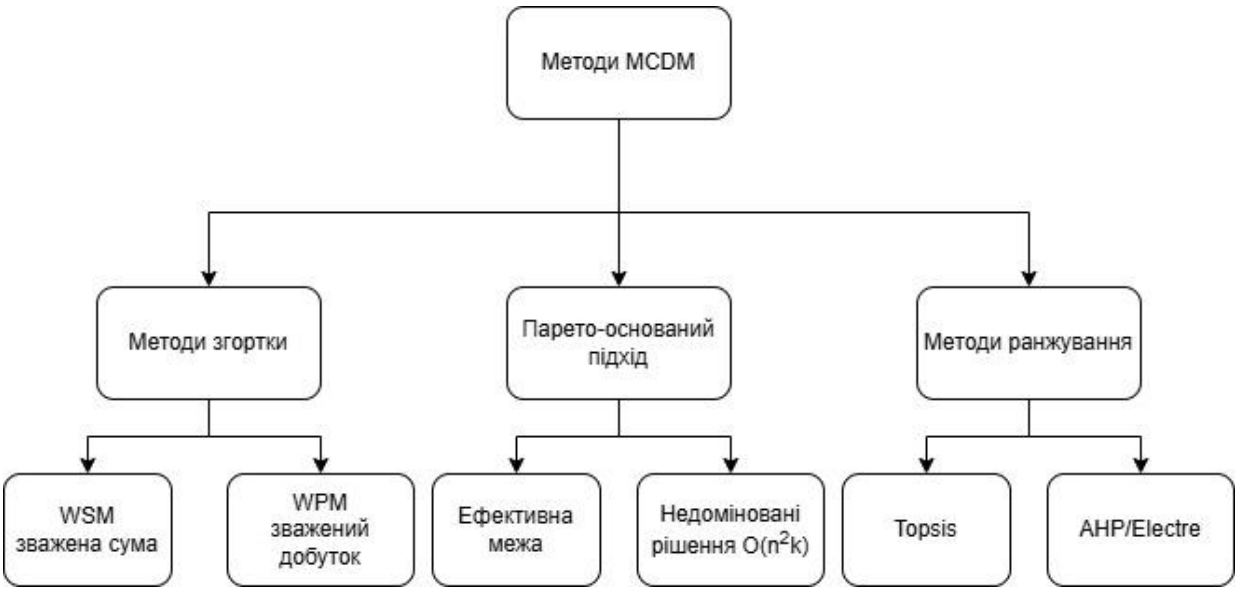


Рисунок 1.1 – Класифікація методів багатокритеріального аналізу рішень

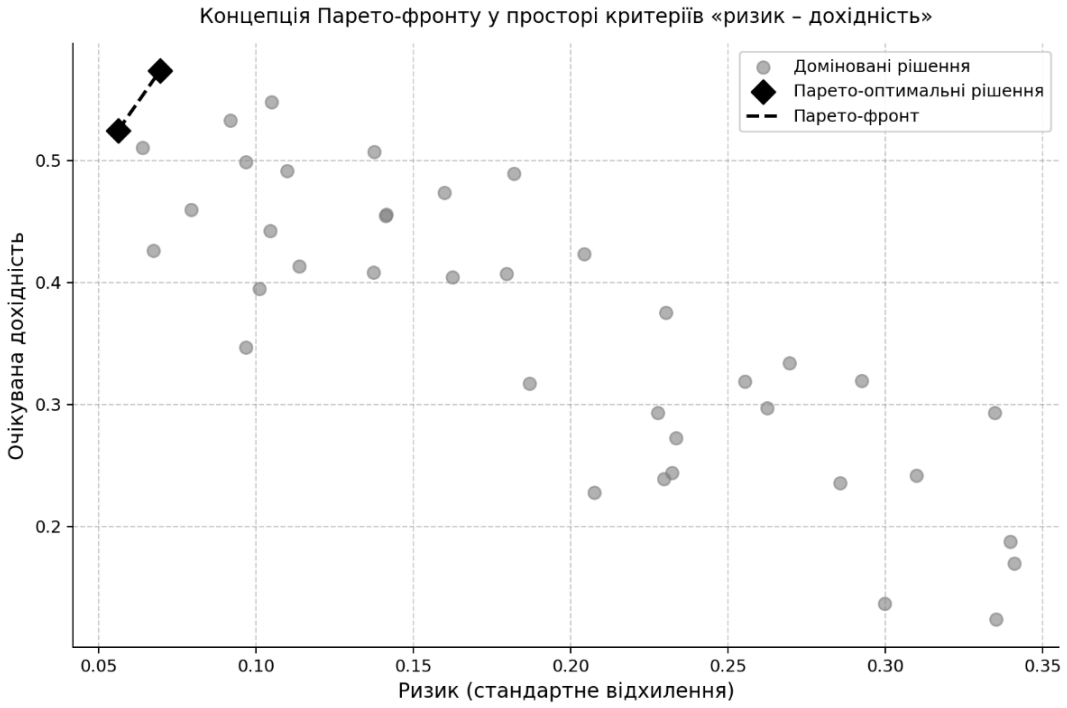


Рисунок 1.2 – Концепція Парето-фронту у просторі критеріїв «ризик-дохідність»

Метод TOPSIS визначає найкращу альтернативу як геометрично найближчу до ідеального та найбільш віддалену від анти-ідеального рішення

[19]. Метод AHP визначає ваги через попарні порівняння. Порівняльний аналіз методів MCDM наведено у таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз методів MCDM

Метод	Ключова ідея	Переваги	Обмеження	У роботі
WSM	Зважена лінійна агрегація	Прозорість, гнучке налаштування ваг	Нерівномірний фронт [18]	Так — основний метод
Парето	Пошук недомінованих рішень без ваг	Повнота аналізу компромісів [17, 34]	Велика кількість рішень	Так — для побудови фронту
TOPSIS	Відстань до ідеалу та анти-ідеалу [19]	Стійкість до викидів	Складніша інтерпретація	Для порівняння
AHP	Попарне порівняння критеріїв	Якісні та кількісні оцінки	Суб'єктивність [5]	Не використовується

1.5 Виявлення недоліків існуючих рішень та обґрунтування необхідності розробки

Проведений аналіз дозволяє виділити три групи недоліків наявних рішень.

1. Більшість моделей залишаються двокритеріальними. У роботі [4] зазначено, що двокритеріальна модель Марковіца не відображає повного спектра інвестиційних вподобань, оскільки реальні рішення можуть враховувати додаткові критерії, зокрема ліквідність, податкові аспекти та індивідуальні обмеження інвестора.

2. Комерційні системи часто мають закриту методологію розрахунків і не забезпечують достатньої прозорості результатів.

3. Відкриті інструменти є складними для непрофесійних користувачів: Бідюк П.І. і Коршевнюк Л.О. [10] наголошують на важливості проектування комп’ютерних систем підтримки прийняття рішень, орієнтованих на практичне застосування. Матвійчук А. В. [12] підкреслює доцільність

застосування інтелектуальних моделей, зокрема нейронних мереж і нечіткої логіки, для розв’язання економічних задач зі складною структурою даних.

Сукупність зазначених недоліків формує запит на відкриту, гнучку, прозору та доступну модель підтримки інвестиційних рішень, що охоплює всі чотири ключових критерії та дозволяє інтерактивно досліджувати простір альтернатив.

1.6 Постановка задачі

Об'єктом дослідження є процес прийняття інвестиційних рішень в умовах багатокритеріальної невизначеності на фінансовому ринку.

Предметом дослідження є методи та моделі багатокритеріальної оптимізації при формуванні інвестиційного портфеля.

Метою роботи є розробка та програмна реалізація моделі підтримки прийняття інвестиційних рішень на основі WSM та Парето-оптимальності з верифікацією на реальних даних.

Для досягнення мети необхідно:

1) провести аналіз предметної галузі та огляд існуючих підходів;
2) визначити та обґрунтувати систему критеріїв оцінювання;
3) розробити математичний апарат та алгоритми;
4) реалізувати програмну модель підтримки рішень;
5) провести верифікацію методом бектестингу та проаналізувати результати.

Формальна постановка задачі. Нехай задано скінченну множину інвестиційних альтернатив $A = \{a_1, a_2, \ldots, a_n\}$ та вектор критеріїв $F = (f_1, f_2, f_3, f_4)$, де f_1 – дохідність (максимізується), f_2 – ризик (мінімізується), f_3 – ліквідність (максимізується), f_4 – відповідність горизонту (максимізується).

Задача 1 (WSM). Знайти альтернативу a* ∈ A, що максимізує зважену суму нормалізованих значень критеріїв:

$$a^* = \underset{a_i \in A}{argmax} \sum_{k=1}^{K} w_k \cdot \tilde{f}_k(a_i), \quad w_k \geq 0, \quad \Sigma w_k = 1$$

Задача 2 (Парето). Визначити множину P* ⊆ A Парето-оптимальних (недомінованих) рішень, де не існує $a_j \in A$ такої, що $a_j \succ a_i$ [17, 34]. Верифікація моделі здійснюється методом бектестингу: портфель, сформований на навчальному підперіоді, перевіряється на тестовому підперіоді і порівнюється з ринковим бенчмарком [20].

Висновки до розділу 1

У першому розділі проведено комплексний аналіз предметної галузі. Розглянуто еволюцію підходів – від класичних моделей Марковіца [1] та CAPM [2] до сучасних MCDM-систем [4-8]. Проведено огляд програмних рішень і виявлено їх ключові обмеження. Визначено та обґрунтовано систему чотирьох критеріїв, а саме дохідності, ризику, ліквідності та інвестиційного горизонту [1, 5, 9, 15]. Обґрунтовано вибір WSM і Парето-оптимальності [17, 18, 34]. Сформульовано мету, задачі та формальну постановку задачі дослідження.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ МАТЕМАТИЧНОЇ МОДЕЛІ ТА АЛГОРИТМІВ

2.1 Формалізація задачі багатокритеріальної оптимізації

Будь-яка задача MCDM оперує трьома ключовими поняттями: простором рішень, простором критеріїв та відношенням переваг [17, 34]. Простір рішень – множина альтернатив $A = \{a_1, \ldots, a_n\}$. Простір критеріїв – простір, де кожній альтернативі a_i відповідає вектор $f(a_i) = (f_1(a_i), f_2(a_i), f_3(a_i), f_4(a_i))$. Задача векторної оптимізації – знайти «найкращу» альтернативу, однак у загальному випадку не існує єдиної альтернативи, що одночасно максимізує всі критерії. Саме це є фундаментальною проблемою, для вирішення якої застосовуються методи WSM та Парето-оптимальності.

Критерії поділяються на два типи: benefit criteria (максимізуються: f_1 – дохідність, f_3 – ліквідність, f_4 – горизонт) та cost criteria (мінімізуються: f_2 – ризик). Після нормалізації більше значення $\tilde{f}_k(a_i)$ завжди відповідає кращій альтернативі незалежно від типу критерію [18]. Ерготт [17] (див., також [34]) підкреслює, що коректна постановка задачі є запорукою якості отриманого рішення.

2.2 Нормалізація критеріїв

Нормалізація є обов'язковим кроком у більшості методів MCDM: критерії вимірюються в різних одиницях з різними порядками величин, тому без нормалізації критерії з більшим числовим розмахом автоматично отримували б більшу вагу [18].

2.2.1 Лінійна мін-макс нормалізація

Для критерію типу «максимізувати» нормалізоване значення обчислюється за формулою:

$$\tilde{f}_k(a_i) = \frac{f_k(a_i) - \min\limits_j f_k(a_j)}{\max\limits_j f_k(a_j) - \min\limits_j f_k(a_j)} \quad (2.1)$$

2.2.2 Векторна нормалізація

Векторна нормалізація ділить значення критерію на евклідову норму вектора значень:

$$\tilde{f}_k(a_i) = \frac{f_k(a_i)}{\sqrt{\sum_j {f_k}^2(a_j)}} \quad (2.2)$$

Цей підхід зберігає відносні пропорції та традиційно використовується у методі TOPSIS [19]. Порівняльний аналіз методів наведено у таблиці 2.1.

Таблиця 2.1 – Порівняльний аналіз методів нормалізації

Метод	Переваги	Недоліки
Мін-макс нормалізація	Гарантований діапазон [0,1]; природна для WSM	Чутлива до викидів
Векторна нормалізація	Стійка до масштабу; зберігає пропорції	Діапазон залежить від кількості альтернатив

У даній роботі обрано лінійну мін-макс нормалізацію, що є стандартним вибором для WSM [17, 18, 21, 34].

2.3 Метод зваженої суми (WSM)

2.3.1 Математичне формулювання

Для множини альтернатив $A = \{a_1, ..., a_n\}$ та вагового вектора $w = (w_1, ..., w_k)$, де $w_k \geq 0$ та $\Sigma w_k = 1$, оцінка WSM обчислюється як:

$$WSM(a_i) = \sum_{k=1}^{K} w_k \cdot \tilde{f}_k(a_i) \quad (2.3)$$

де w_k – вага k-го критерію, $\tilde{f}_k(a_i)$ – нормалізоване значення k-го критерію для альтернативи a_i, K – кількість критеріїв.

Оптимальна альтернатива: $a^* = \text{argmax } WSM(a_i)$. Оскільки $\tilde{f}_k \in [0,1]$ та $\Sigma w_k = 1$, значення $WSM(a_i) \in [0,1]$, що забезпечує зручну інтерпретацію.

2.3.2 Покроковий алгоритм

Принципово важливо, що нормалізація виконується один раз для всієї матриці рішень, а не в циклі. Блок-схему алгоритму WSM наведено на рисунку 2.1.

Крок 1: формування матриці рішень F (n×K).

Крок 2: нормалізація матриці F одноразово.

Крок 3: визначення вагового вектора w.

Крок 4: у циклі по і обчислення $WSM(a_i)$.

Крок 5: ранжування за спаданням WSM.

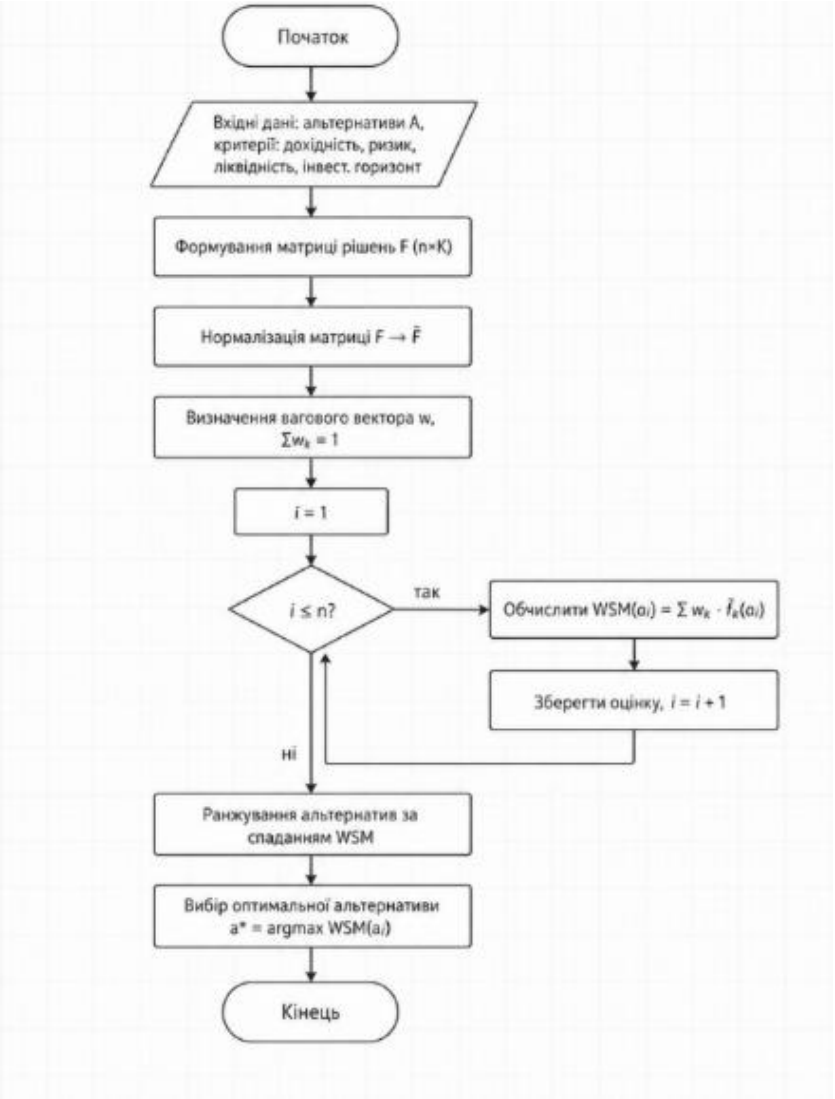


Рисунок 2.1 – Блок-схема алгоритму методу зваженої суми

2.3.3 Визначення вагових коефіцієнтів

Виділяють три підходи: пряме задання ваг користувачем (основний у даній роботі), рівномірний розподіл ($w_i = 1/K$, базова конфігурація) та визначення через AHP [17, 34]. Вітлінський В.В. та Великоіваненко Г.І. [9]

підкреслюють важливість аналізу чутливості результату до зміни ваг. У системі передбачено інтерактивне налаштування ваг у реальному часі.

2.4 Парето-оптимальність та алгоритм побудови фронту

2.4.1 Визначення домінування

Після нормалізації всі критерії мають однакову спрямованість. Альтернатива a_j домінує над a_i ($a_j \succ a_i$), якщо вона не гірша за a_i за всіма критеріями одночасно і строго краща хоча б за одним. Формально: для всіх k виконується $\tilde{f}_k(a_j) \geq \tilde{f}_k(a_i)$, причому існує k_0, для якого нерівність строга. Концепцію домінування ілюструє рисунок 2.2: рішення, що домінують над a', розташовані у лівому верхньому куті (менший ризик, більша дохідність).

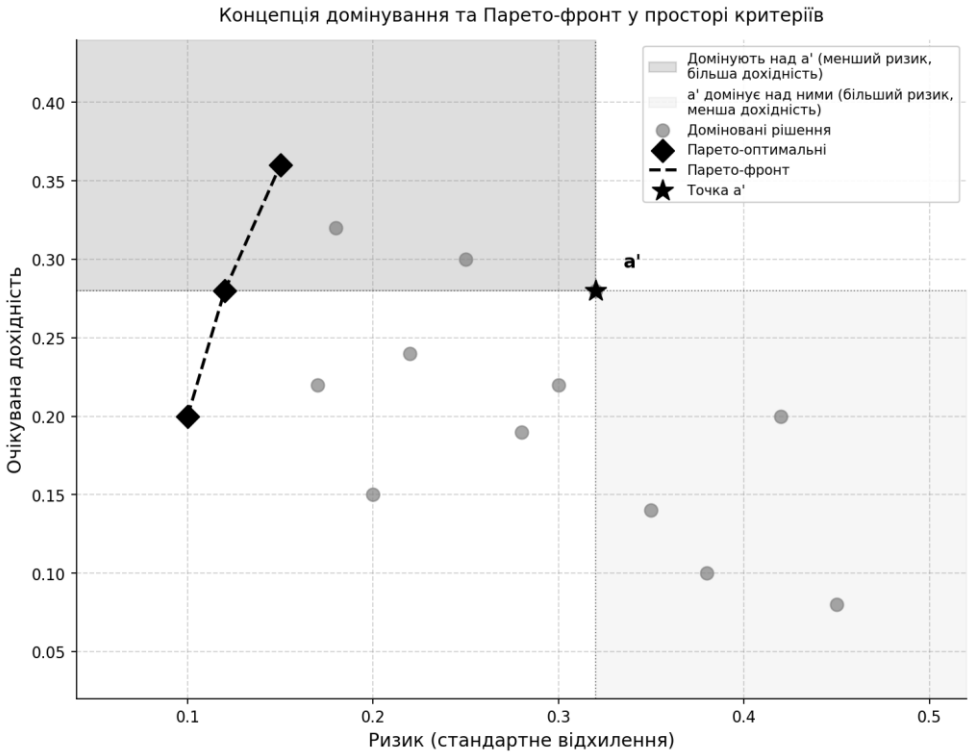


Рисунок 2.2 – Концепція домінування та Парето-фронт у просторі критеріїв

2.4.2 Парето-оптимальна множина та алгоритм

Альтернатива a_i є Парето-оптимальною (недомінованою), якщо не існує a_j, що домінує над нею. Множина P* усіх таких рішень – Парето-фронт [17, 34]. Алгоритм прямого перебору має складність $O(n^2 \cdot K)$. Для фіксованого набору активів ($n \leq$ кількох сотень, K=4) ця складність є цілком прийнятною. Еволюційні алгоритми типу NSGA-II [22] призначені для неперервного простору рішень і не застосовуються для скінченної множини активів. Для задачі з чотирма критеріями застосовуються двовимірні проекції Парето-фронту.

2.5 Математичні міри оцінювання ризику

2.5.1 Волатильність

Вибіркове стандартне відхилення дохідності:

$$\sigma = \sqrt{\frac{1}{T-1} \cdot \sum_{t=1}^{T} (r_t - \bar{r})^2} \tag{2.4}$$

Волатильність є непараметричною статистикою і не передбачає припущень щодо форми розподілу. Вона обрана основною мірою ризику у MCDM-моделі завдяки простоті та широкому застосуванню [1, 23].

2.5.2 Value at Risk (VaR)

VaR на рівні довіри α визначається наступним чином:

$VaR_\alpha = -q_\alpha(r),\ q_\alpha(r) -$ квантиль рівня $(1-\alpha)$

При α = 0,95 VaR0.95 відповідає 95%-квантилю: з ймовірністю 95% збиток не перевищить цього значення. Відомим недоліком є порушення субадитивності [15]. У роботі використовується історичний метод обчислення VaR як додатковий показник при бектестуванні. У фінансовому ризик-менеджменті VaR широко використовується як стандартна міра потенційних втрат портфеля за заданого рівня довіри [16].

2.5.3 Conditional Value at Risk (CVaR)

CVaR – середній збиток у найгірших (1-α)·100% випадків:

$$CVaR_\alpha = -E\left[r_t \mid r_t \leq q_\alpha(r)\right] \quad (2.5)$$

Рокафеллар. Р. та Урясев С. [15] довели, що CVaR задовольняє аксіомам когерентності, зокрема субадитивності. Порівняльний аналіз мір ризику наведено у таблиці 2.2.

Таблиця 2.2 – Порівняльний аналіз мір оцінювання ризику

Властивість	**Волатильність**	**VaR**	**CVaR**	**Застосування**
Когерентність	Ні	Ні	Так [15]	CVaR (додатково)
Субадитивність	Ні	Ні	Так	-
Простота	Висока	Середня	Середня	Волат. (основна)
Врахування хвоста	Часткове	Ні	Повне	VaR (верифікація)
Залеж. від розподілу	Відсутня (непарам.)	Середня (іст.)	Середня (іст.)	-

2.6 Алгоритм бектестингу як метод верифікації моделі

2.6.1 Принцип та схема бектестингу

Бектестинг – ретроспективне тестування, при якому модель застосовується до історичних даних так, ніби майбутнє невідоме. Часовий ряд ділиться на навчальний (in-sample) та тестовий (out-of-sample) підперіоди. На навчальному – розраховуються критерії і формується портфель; на тестовому – порівнюється з бенчмарком [5]. Схему бектестингу наведено на рисунку 2.3.

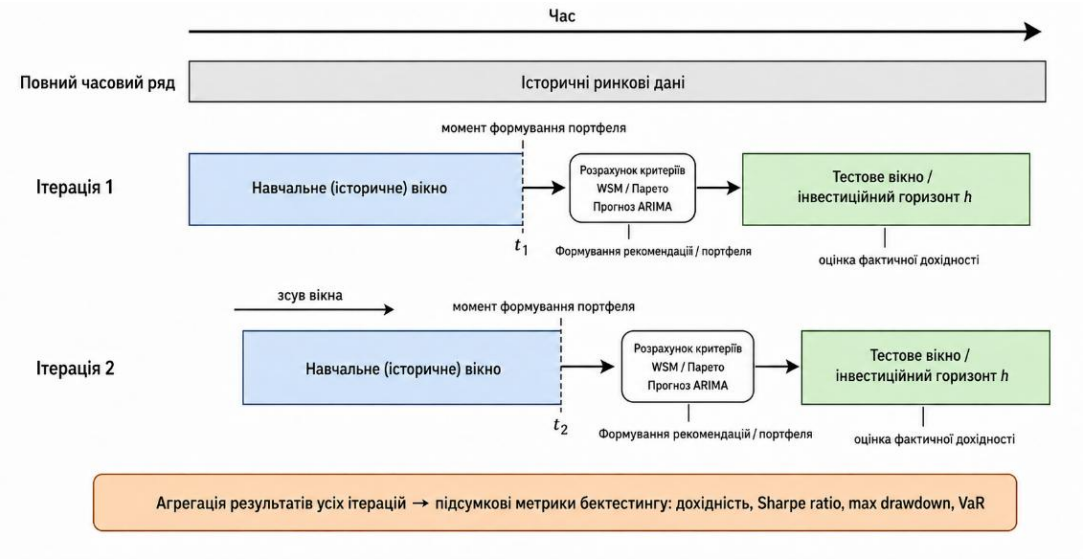


Рисунок 2.3 – Схема часової структури бектестингу

Навчальний підперіод встановлено рівним 2 рокам (≥500 торгових днів), тестовий – 1 рік. Такий розподіл забезпечує достатній обсяг навчальних даних та значущий тестовий горизонт.

2.6.2 Метрики оцінювання якості портфеля

Коефіцієнт Шарпа [23] визначається наступним чином:

$$Sharpe = \frac{R_p - r_f}{\sigma_p} \tag{2.6}$$

Максимальна просадка (MDD), де максимум береться по всіх парах $t_1 \leq t_2$:

$$MDD = \max_{t_1 \leq t_2} \frac{V(t_1) - V(t_2)}{V(t_1)} \quad (2.7)$$

Порівняльний аналіз метрик наведено у таблиці 2.3.

Таблиця 2.3 – Метрики оцінювання якості портфеля при бектестуванні

Метрика	Формула	Практичний зміст
Коефіцієнт Шарпа	$(R_p - r_f) / \sigma_p$	Дохідність на одиницю ризику [23]
Макс. просадка	$\max(V(t_1)-V(t_2))/V(t_1)$	Найбільше відносне падіння вартості
Накопичена дохідність	$(V(T) - V(0)) / V(0)$	Загальний фінансовий результат
Волатильність	$\sigma_p = \text{std}(r_p)$	Ступінь коливань; порівнюється з бенчмарком

Як бенчмарк обрано рівномірний портфель 1/n [20]. Він дозволяє оцінити, чи забезпечує MCDM-модель реальний приріст якості рішень порівняно з найпростішою стратегією.

2.7 Загальна архітектура математичної моделі

Модель складається з п'яти функціональних блоків:

блок 1 – збір та підготовка даних (ціни, обсяги, дохідності);

блок 2 – обчислення критеріїв (f1-f4), формування матриці F (n×4).;

блок 3 – мін-макс нормалізація матриці F одноразово;

блок 4 – паралельне виконання WSM та алгоритму Парето;

блок 5 – виведення результатів: рейтинг WSM, Парето-фронт, бектестинг;

Принципова особливість архітектури – зворотний зв'язок між блоком 4 та користувачем: зміна вагових коефіцієнтів не потребує повторного виконання

блоків 1-3, що суттєво прискорює інтерактивний аналіз. Бідюк П.І. та Коршевнюк Л.О. [10] підкреслюють, що інтерактивність є однією з ключових вимог до сучасних систем підтримки прийняття рішень.

Висновки до розділу 2

У другому розділі побудовано повний математичний апарат моделі. Формалізовано задачу MCDM, обґрунтовано вибір мін-макс нормалізації [18]. Описано алгоритм WSM з підкресленням одноразовості нормалізації та алгоритм Парето-фронту $O(n^2K)$ [17, 34]. Уточнено геометричну інтерпретацію домінування у просторі «ризик-дохідність». Розглянуто міри ризику – волатильність, VaR та CVaR, а також обґрунтовано їх роль у задачах оцінювання інвестиційних стратегій [11, 15, 16].

Описано алгоритм бектестингу з використанням метрик накопиченої дохідності, волатильності, коефіцієнта Шарпа та максимальної просадки [20, 23].

РОЗДІЛ 3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ПРОГРАМНОЇ МОДЕЛІ

3.1 Вибір інструментів і технологій

Ключовою вимогою до вибору технологічного стеку для реалізації моделі підтримки інвестиційних рішень є поєднання достатньої обчислювальної потужності, широкого набору готових бібліотек для аналізу фінансових даних та можливості швидкого прототипування інтерактивного інтерфейсу без залучення окремих front-end технологій. Всім цим вимогам відповідає екосистема мови Python.

Мова програмування Python обрана як основна з кількох причин.

1. Python є де-факто стандартом у сфері науки про дані та кількісних фінансів[25]: бібліотеки pandas, NumPy, scipy та statsmodels утворюють зрілий, добре задокументований стек для роботи з часовими рядами [26].

2. Завдяки динамічній типізації та лаконічному синтаксису код моделі залишається читабельним, що спрощує підтримку та перевірку математичних алгоритмів.

3. Тісна інтеграція Python з бібліотеками візуалізації (Plotly) та веб-фреймворками (Streamlit) дозволяє реалізувати повноцінний застосунок в одному файлі без необхідності переключатися між мовами.

Серед альтернатив розглядались мова R та середовище MATLAB. Мова R, незважаючи на потужний статистичний апарат, поступається Python у частині розробки веб-застосунків: фреймворк Shiny вимагає окремих навичок і є менш гнучким за Streamlit. Крім того, екосистема Python для завантаження фінансових даних (yfinance) є зрілішою та активніше підтримуваною. MATLAB не розглядався з практичних міркувань: комерційна ліцензія обмежує доступність для кінцевих користувачів – індивідуальних інвесторів, тоді як відкритість і безкоштовність є однією з ключових вимог до розроблюваної системи.

Бібліотека Streamlit обрана для побудови інтерфейсу користувача як найбільш підходящий інструмент для аналітичних застосунків на Python. На відміну від класичних web-фреймворків (Flask, Django), Streamlit не потребує написання HTML, CSS або JavaScript: розробник описує інтерфейс безпосередньо в коді Python за допомогою декларативних викликів. Ключовою технічною особливістю Streamlit є реактивна модель виконання: кожна зміна значення у бічній панелі (слайдер, перемикач) автоматично запускає перерахунок відповідних частин застосунку. Це забезпечує реалізацію ключової вимоги системи – інтерактивного налаштування вагових коефіцієнтів у режимі реального часу без необхідності вручну натискати кнопку «Перерахувати».

Бібліотека yfinance є відкритим Python-клієнтом до API Yahoo Finance і надає безкоштовний доступ до щоденних цін закриття, обсягів торгів та іншої ринкової інформації по тисячам активів. Важливою перевагою yfinance є підтримка параметра auto_adjust=True, що автоматично коригує ціни закриття на корпоративні події – дроблення акцій (splits) та виплату дивідендів. Без такого коригування порівняння цін до і після дроблення давало б хибні значення дохідностей. Для візуалізації результатів обрано бібліотеку Plotly, яка генерує інтерактивні графіки у форматі HTML – користувач може масштабувати, фільтрувати та досліджувати дані безпосередньо у браузері без додаткового програмного забезпечення. Перелік використаних технологій та їх призначення у проєкті наведено у таблиці 3.1.

Таблиця 3.1 – Технологічний стек програмної моделі

Бібліотека	Версія	Призначення у проєкті
Python	3.10+	Основна мова реалізації
Streamlit	≥ 1.32	Побудова інтерактивного веб-інтерфейсу
yfinance	≥ 0.2.36	Завантаження ринкових даних (ціни, обсяги)
pandas	≥ 2.0	Маніпуляції з часовими рядами та DataFrame
NumPy	≥ 1.26	Векторизовані обчислення критеріїв та WSM
plotly	≥ 5.18	Інтерактивна візуалізація (графіки, діаграми)
statsmodels	≥ 0.14	Реалізація моделі ARIMA(1,1,1)

3.2 Архітектура програмної моделі

Програмна модель реалізована у вигляді єдиного модуля app.py, що відповідає архітектурі, описаній у підрозділі 2.7. Вибір монолітної структури виправданий масштабом застосунку: єдиний модуль спрощує розгортання (одна команда запуску python -m streamlit run app.py) і унеможливлює проблеми із залежностями між модулями. Загальний обсяг програмного коду складає близько 600 рядків, що знаходиться у межах, за яких монолітна структура є оптимальною з точки зору підтримуваності. Код організований у п'ять функціональних блоків, що логічно відповідають п'яти блокам математичної моделі.

Програмна модель побудована відповідно до п’яти функціональних блоків, описаних у загальній архітектурі системи. Перший блок відповідає за

завантаження та попередню підготовку ринкових даних, зокрема цін закриття, обсягів торгів і логарифмічних дохідностей. Другий блок забезпечує обчислення критеріїв оцінювання активів: дохідності, ризику, ліквідності та відповідності інвестиційному горизонту. Третій блок виконує нормалізацію критеріїв, що дозволяє привести різнорідні показники до єдиної шкали. Четвертий блок реалізує методи багатокритеріального аналізу, а саме розрахунок WSM-оцінки та визначення Парето-оптимальних альтернатив. П’ятий блок відповідає за верифікацію результатів методом бектестингу та побудову прогнозу за моделлю ARIMA.

Керування роботою всіх блоків здійснюється через основний модуль інтерфейсу, реалізований засобами Streamlit. Він об’єднує процес завантаження даних, розрахунку критеріїв, формування рейтингу, побудови Парето-фронту, прогнозування та відображення результатів у єдиний послідовний сценарій роботи користувача. Така організація програмної моделі забезпечує модульність системи, оскільки кожен функціональний блок виконує окрему задачу та може бути змінений або розширений без суттєвого впливу на інші частини програмного продукту.

Для підвищення швидкодії системи використано механізми збереження проміжних результатів між перерахунками інтерфейсу. Це є важливим, оскільки під час зміни користувачем вагових коефіцієнтів або параметрів аналізу немає потреби повторно завантажувати ринкові дані та заново виконувати всю попередню обробку. Повторно виконуються лише ті обчислення, які безпосередньо залежать від змінених параметрів, зокрема розрахунок інтегральної WSM-оцінки та побудова Парето-фронту.

Додатково реалізовано кешування завантажених ринкових даних протягом обмеженого часу. Це дозволяє зменшити кількість повторних звернень до зовнішнього джерела даних, підвищити стабільність роботи застосунку та скоротити час очікування для користувача під час повторного запуску аналізу в межах однієї робочої сесії.

3.3 Реалізація модуля завантаження та підготовки даних

Завантаження ринкових даних реалізовано у функції load_data(tickers, start, end). Функція послідовно звертається до API Yahoo Finance через бібліотеку yfinance для кожного тікера зі списку та формує два DataFrame: prices (ціни закриття) та volumes (обсяги торгів). Для кожного активу завантажуються щоденні дані з автоматичним коригуванням на корпоративні події (спліти, дивіденди) завдяки параметру auto_adjust=True. Якщо для певного тікера завантажено менше 50 спостережень (наприклад, через перебої у роботі API або відсутність торгів), він виключається зі списку активів із відповідним попередженням для користувача. Поріг у 50 спостережень обраний як мінімально достатній для обчислення статистично значущих оцінок дохідності та волатильності.

Після завантаження обидва DataFrame об'єднуються методом dropna(), який видаляє рядки з пропущеними значеннями – це гарантує, що всі активи у вибірці мають дані за однаковий часовий діапазон. Даний крок є критичним для коректності нормалізації: оскільки мін-макс нормалізація обчислює глобальні мінімуми та максимуми по всіх рядках матриці, наявність пропущених значень могла б призвести до некоректних результатів.

Логарифмічні дохідності обчислюються функцією log_returns(prices):

$$r_t = ln \frac{P_t}{P_{t-1}}$$

Метод NumPy.log(prices / prices.shift(1)).dropna() реалізує цю формулу у векторизованій формі для всього DataFrame одночасно. Використання логарифмічних, а не арифметичних дохідностей виправдане їх адитивністю в часі та кращою апроксимацією нормальним розподілом, що важливо для коректності подальших статистичних розрахунків [10, 25]. Крім того, логарифмічні дохідності симетричні: зростання ціни на 10% і подальше падіння на 10% дають нульову сумарну лог-дохідність, тоді як арифметичні

дохідності давали б від'ємний результат (-1%). Ця властивість є важливою для коректного усереднення дохідностей за тривалі горизонти [3].

3.4 Реалізація алгоритмів MCDM

3.4.1 Обчислення критеріїв

Функція compute_criteria(prices, volumes, horizon) обчислює значення чотирьох критеріїв для кожного активу на основі навчального підперіоду. Критерій f_1 (дохідність) визначається як середня логарифмічна дохідність, помножена на 252 (кількість торгових днів у році) для ануалізації. Такий підхід дозволяє порівнювати активи з різною волатильністю на єдиній річній шкалі незалежно від тривалості навчального підперіоду. Критерій f_2 (ризик) – стандартне відхилення денних дохідностей, помножене на $\sqrt{252}$ для ануалізації. Критерій f_3 (ліквідність) – середній денний обсяг торгів у мільйонах доларів США, обчислений за весь навчальний підперіод.

Критерій f_4 (відповідність горизонту інвестування) обчислюється різними способами залежно від обраного горизонту: для короткострокового – зважена комбінація ліквідності та захисту від ризику:

$$(0.6 \cdot f3/max(f3) + 0.4 \cdot 1/f2);$$

для середньострокового – коефіцієнт Шарпа [19]:

$\frac{(f1 - rf)}{f2}$, де rf = 0.045 (безризикова ставка 4.5%);

для довгострокового горизонту використовується коефіцієнт Кальмара, що пов’язує дохідність із максимальною просадкою портфеля [24]:

$$\frac{f1}{|MDD|}$$

Диференційований підхід до обчислення f_4 відображає об'єктивно різні вимоги до активів залежно від горизонту: при короткому горизонті ключовим є можливість швидко продати позицію без суттєвих транзакційних витрат, тоді як при довгому – здатність активу забезпечувати дохідність, скориговану на просадки. Безризикова ставка rf = 4.5% відповідає типовій ставці короткострокових облігацій США станом на 2024 рік і використовується як орієнтир мінімально прийнятної дохідності. Результати функції для сеансу з довгостроковим горизонтом наведено на рисунку 3.1.

Розраховані значення критеріїв

	Дохідність (річна)	Ризик (волатильність)	Ліквідність (млн $)	Оцінка горизонту
AAPL	0.3018	0.3169	96.4719	0.8102
MSFT	0.2589	0.3038	29.0018	0.7041
TSLA	0.5329	0.6509	130.0483	0.7497
JPM	0.1551	0.3187	13.3744	0.3455
JNJ	0.0469	0.1993	8.1882	0.0096
KO	0.0768	0.2104	14.9733	0.1513
NVDA	0.7075	0.5193	455.4242	1.2758

Рисунок 3.1 – Розраховані значення критеріїв для семи активів

З рисунку 3.1 видно суттєву різницю між активами за всіма критеріями. NVDA демонструє найвищу річну дохідність (0.7075) та найвищу ліквідність (455.42 млн $), але також значний ризик (0.5193). JNJ, навпаки, показує найнижчу дохідність (0.0469) при найменшій волатильності (0.1993). Варто відзначити, що TSLA має вищу абсолютну дохідність (0.5329), ніж AAPL та MSFT, але також значно більший ризик (0.6509) – найвищий серед усіх активів. Ця різноманітність профілів підтверджує необхідність багатокритеріального підходу: жоден актив не є беззаперечно найкращим за всіма критеріями одночасно, і оптимальний вибір суттєво залежить від пріоритетів конкретного інвестора.

3.4.2 Нормалізація

Функція normalize(criteria) реалізує лінійну мін-макс нормалізацію матриці критеріїв[18, 19, 21]. Для кожного критерію обчислюються глобальні мінімум та максимум по всій множині альтернатив. Для критеріїв, що максимізуються (f_1, f_3, f_4), застосовується пряма формула (2.1). Для критерію ризику f_2, що мінімізується, застосовується інвертована формула (2.2). Принципово важливим є те, що нормалізація виконується одноразово для всієї матриці, а не окремо для кожного рядка: мінімум і максимум є глобальними по всіх активах, що забезпечує збереження відносних відстаней між ними на шкалі [0; 1]. Якби нормалізація виконувалась у циклі по активах, результати були б некоректними. Результуюча нормалізована матриця критеріїв представлена на рисунку 3.2.

Нормалізована матриця критеріїв

	f1 Дохідність	f2 Ризик	f3 Ліквідність	f4 Горизонт
AAPL	0.386	0.739	0.197	0.632
MSFT	0.321	0.768	0.047	0.548
TSLA	0.736	0.000	0.272	0.584
JPM	0.164	0.736	0.012	0.265
JNJ	0.000	1.000	0.000	0.000
KO	0.045	0.975	0.015	0.112
NVDA	1.000	0.291	1.000	1.000

Всі значення в діапазоні [0, 1]. Більше = краще.

Рисунок 3.2 – Нормалізована матриця критеріїв $\tilde{F}$

Після нормалізації всі значення належать відрізку [0, 1], причому більше значення завжди відповідає кращій альтернативі незалежно від типу критерію. Зокрема, NVDA отримала максимальну нормалізовану дохідність ($\tilde{f}_1 = 1.000$) і максимальну ліквідність ($\tilde{f}_3 = 1.000$), тоді як JNJ – мінімальні нормалізовані значення за обома критеріями (0.000). По критерію ризику ($\tilde{f}_2$) JNJ отримала 1.000, оскільки має найнижчу волатильність, тоді як TSLA отримала 0.000 – найвищий ризик серед усіх активів. Слід зазначити, що нормалізована матриця

повністю замінює сирі значення критеріїв при подальших обчисленнях WSM та Парето-аналізу: всі розрахунки виконуються виключно з $\tilde{f}_k$, що гарантує незмінність результатів незалежно від одиниць виміру вхідних даних.

3.4.3 Метод зваженої суми

Функція wsm(normalized, w_1, w_2, w_3, w_4) реалізує метод зваженої суми відповідно до формули (2.4) [18]. Перед обчисленням ваги автоматично нормалізуються (діляться на їх суму), щоб гарантувати виконання умови $\Sigma w_k = 1$ незалежно від значень слайдерів у інтерфейсі. Це технічне рішення дозволяє уникнути ситуації, коли випадкове встановлення суми ваг, відмінної від одиниці, призводило б до некоректного масштабування результату. Оцінка кожного активу обчислюється як скалярний добуток вектора ваг на відповідний рядок нормалізованої матриці за формулою (2.4). Функція повертає відсортований за спаданням pandas Series, що безпосередньо задає ранжування активів. Результати ранжування наведено на рисунку 3.3.

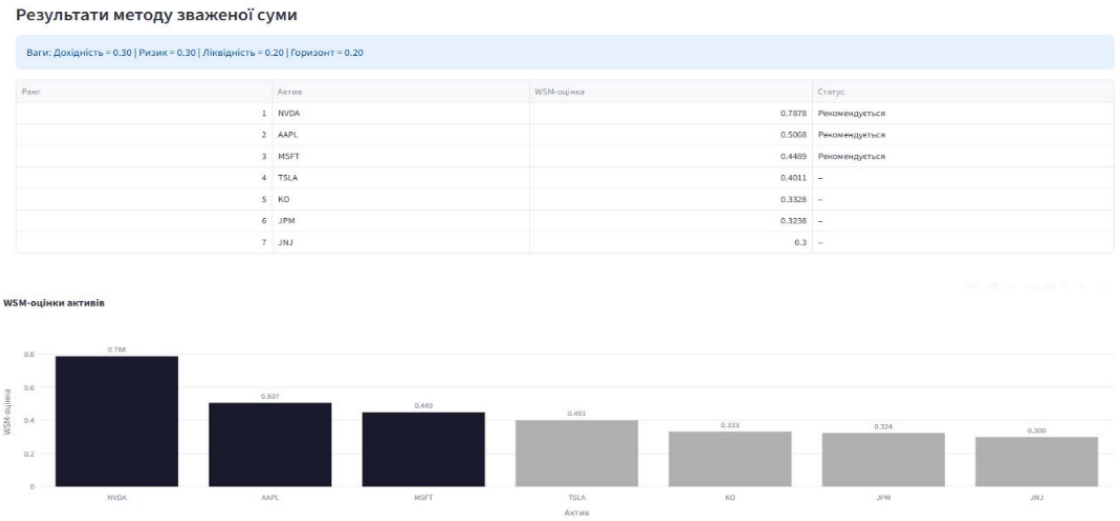


Рисунок 3.3 – Результати методу зваженої суми: ранжування активів та WSM-оцінки

За результатами WSM при ваговому векторі w = (0.30; 0.30; 0.20; 0.20) першу позицію займає NVDA з оцінкою 0.7878, другу – AAPL (0.5068), третю – MSFT (0.4489). Ці три активи отримали статус «Рекомендується» і формують WSM-портфель для бектестування. Значний відрив NVDA від інших активів пояснюється домінуванням за двома з чотирьох критеріїв: $\tilde{f}_1 = 1.000$ та $\tilde{f}_3 = 1.000$. Навіть незважаючи на відносно помірне значення $\tilde{f}_2 = 0.291$ (ризик), внесок двох максимальних критеріїв забезпечує NVDA загальну оцінку 0.7878. Четверту та нижчі позиції займають активи зі збалансованим профілем: TSLA (0.4011), KO (0.3328), JPM (0.3238), JNJ (0.300). Помітно, що JNJ, яка має найкращий показник ризику ($\tilde{f}_2 = 1.000$), займає останнє місце через нульові значення дохідності та ліквідності – що додатково ілюструє важливість збалансованого вагового вектора.

3.4.4 Алгоритм Парето-фронту

Функція pareto_front(normalized) реалізує алгоритм знаходження недомінованих рішень складності $O(n^2 \cdot K)$. Для кожної альтернативи a_i перевіряється існування хоча б однієї альтернативи aj, що домінує над нею за критерієм домінування: $a_j \succ a_i$ тоді і тільки тоді, коли $\tilde{f}_k(a_j) \geq \tilde{f}_k(a_i)$ для всіх k і $\tilde{f}_k(a_j) > \tilde{f}_k(a_i)$ хоча б для одного k. Функція повертає індекс Парето-оптимальних рішень. Результати аналізу наведено на рисунку 3.4.

Алгоритм виявив 5 Парето-оптимальних рішень: AAPL, MSFT, JNJ, KO, NVDA. Активи TSLA та JPM виявились домінованими. На лівому графіку (ризик-дохідність) видно, що TSLA має нижчу нормалізовану дохідність порівняно з NVDA при вищому ризику, тобто NVDA домінує над TSLA за всіма чотирма критеріями одночасно.

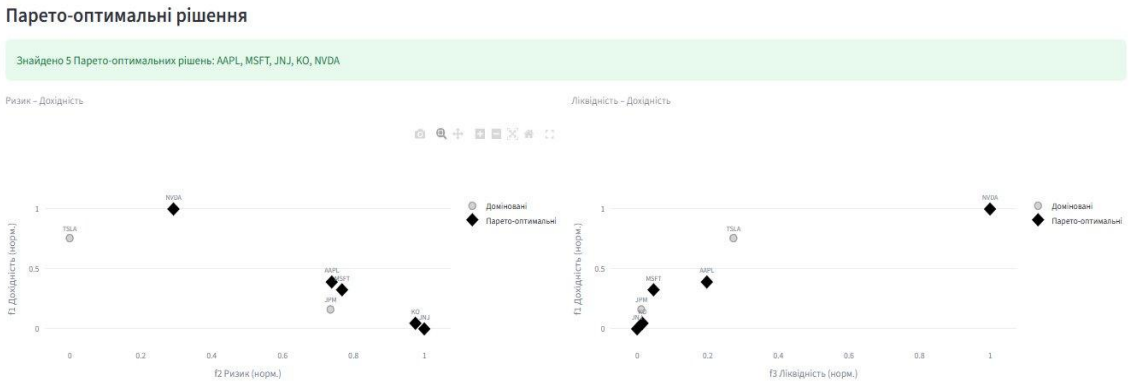


Рисунок 3.4 – Парето-оптимальні рішення у проекціях «Ризик-Дохідність» та «Ліквідність-Дохідність»

3.4.5 Практичні обмеження Парето-аналізу при чотирьох критеріях

Аналіз отриманих результатів виявляє характерну особливість Парето-оптимального підходу, яка потребує окремого пояснення. З рисунку 3.4 видно, що активи JNJ та KO входять до Парето-фронту, незважаючи на те, що їхня нормалізована дохідність ($\tilde{f}1$) та ліквідність ($\tilde{f}3$) є практично нульовими. На перший погляд це здається суперечністю: яку цінність має актив, що нічого не заробляє і майже не торгується?

Відповідь криється в самому означенні домінування. JNJ має нормалізований показник ризику $\tilde{f}_2 = 1.000$ – тобто є найменш волатильним активом серед усіх розглянутих. Щоб якийсь інший актив X «витіснив» JNJ з Парето-фронту, X мав би бути не гіршим за JNJ одночасно за всіма чотирма критеріями. Однак перевищити значення 1.000 за критерієм ризику неможливо – це вже максимум шкали нормалізації. Отже, жоден актив математично не може домінувати над JNJ, і вона завжди залишається на Парето-фронті, навіть з нульовою дохідністю та ліквідністю.

Ця ситуація є відомим практичним обмеженням Парето-аналізу при великій кількості критеріїв. Загальне правило таке: чим більше критеріїв враховується в задачі, тим більшим стає Парето-фронт і тим менш корисним він є для формування конкретної рекомендації. У двовимірному просторі (лише дохідність та ризик, як у класичній моделі Марковіца [1]) JNJ очевидно не потрапила б на фронт – її витіснив би AAPL або MSFT, які кращі за дохідністю при порівнянному рівні ризику. Але в чотиривимірному просторі критеріїв перевага за одним «рекордним» показником гарантує місце на фронті незалежно від значень інших критеріїв.

Таким чином, у контексті даної роботи Парето-аналіз виконує передусім ілюстративну та аналітичну функцію: він показує, що жоден актив не є «абсолютно неефективним» за всіма критеріями одночасно, та наочно демонструє компроміси між ними. Для інвестора, якого цікавить виключно захист від просадок і якому байдужі дохідність та ліквідність, JNJ справді є раціональним вибором. Однак для формування збалансованої інвестиційної рекомендації основним інструментом моделі залишається метод зваженої суми: він враховує пріоритети інвестора через явно задані вагові коефіцієнти і завжди дає однозначне ранжування всіх альтернатив.

3.5 Реалізація модуля бектестингу

Функція run_backtest(prices, top_tickers, test_start, test_end) реалізує метод ретроспективного тестування, описаний у підрозділі 2.6. На тестовому підперіоді ціни кожного активу нормалізуються відносно першого дня (база = 1.0), після чого обчислюється рівновагова накопичена вартість WSM-портфеля (рівні частки у трьох рекомендованих активах) та бенчмарку (рівні частки у всіх обраних активах). Формування WSM-портфеля як рівновагового портфеля з трьох активів є спрощенням: у реальному застосуванні оптимальні частки

могли б визначатися окремим алгоритмом, проте для цілей верифікації моделі такий підхід є загальновживаним у академічних дослідженнях [5]. Денні дохідності обчислюються методом pct_change().

Як бенчмарк обрано рівновагований портфель 1/n, що включає всі обрані активи. Такий підхід широко використовується як базова стратегія порівняння в дослідженнях портфельної оптимізації [20]. По-друге, вона є максимально простою – не потребує жодних знань чи припущень і тому є природною точкою відліку для оцінки того, чи дає більш складна MCDM-модель реальний приріст якості.

Для оцінки якості портфеля обчислюються чотири метрики. Коефіцієнт Шарпа визначається за формулою (3.2):

$$Sharpe = \frac{\bar{r} - r_f}{\sigma_p} \cdot \sqrt{252}$$

де $\bar{r}$ – середня денна дохідність портфеля, $r_f = 0.045/252$ – денна безризикова ставка, σ – стандартне відхилення денних дохідностей. Множник $\sqrt{252}$ забезпечує ануалізацію денного коефіцієнта Шарпа [23]. Максимальна просадка визначається за формулою (3.3):

$$MDD = \min_t \frac{V(t) - \max_{\tau \le t} V(\tau)}{\max_{\tau \le t} V(\tau)}$$

де V(t) – вартість портфеля у момент t, cummax(t) – максимальна вартість портфеля з початку спостереження до моменту t. MDD показує найбільше відносне падіння від піку до мінімуму і є важливою мірою хвостового ризику: навіть якщо загальна дохідність є позитивною, великий MDD може означати, що інвестор у певний момент зазнав значних нереалізованих збитків, що психологічно є важким навантаженням. Накопичена дохідність: (portfolio.iloc[-1] - 1) · 100%. Волатильність: $\sigma \cdot \sqrt{252} \cdot 100\%$. Результати бектестингу наведено на рисунку 3.5.

Рисунок 3.5 – Результати верифікації: накопичена дохідність WSM-портфеля та бенчмарку

WSM-портфель (NVDA, AAPL, MSFT) за тестовий підперіод (травень 2024 – травень 2026) показав накопичену дохідність 50.2% при коефіцієнті Шарпа 0.725, максимальній просадці -28.6% та волатильності 27.2%. Бенчмарк (рівновагові 7 активів) перевершив WSM-портфель за всіма чотирма метриками: дохідність 66.1%, Sharpe 1.101, MDD -25.0%, волатильність 21.3%. Перевага бенчмарку пояснюється включенням до нього захисних активів JNJ і KO, які з одного боку знижують загальну волатильність, а з іншого – завдяки своїй захисній природі краще витримали ринкову корекцію квітня 2025 року. На графіку рисунку 3.5 видно, що обидві криві зростали паралельно до квітня 2025, після чого WSM-портфель упав значно глибше через концентрацію у волатильних технологічних активах (NVDA, AAPL, MSFT). Після відновлення WSM-портфель наздогнав, але не перевершив бенчмарк. Разом із тим, критерій Вілкоксона ($p = 0.9367$) та парний t-тест ($p = 0.7347$) не виявили статистично значущої різниці між денними дохідностями портфелів на рівні значущості $\alpha = 0.05$ (499 торгових днів спостереження, середня різниця -0.0146%/день). Це означає, що, незважаючи на нижчу абсолютну дохідність WSM-портфеля, наявні результати не дають підстав стверджувати про статистично значущу перевагу пасивної стратегії 1/n за денними дохідностями. Такий результат узгоджується з відомим фактом про складність стабільного перевершення простих рівновагових портфельних стратегій [20].

3.6 Реалізація модуля прогнозування ARIMA

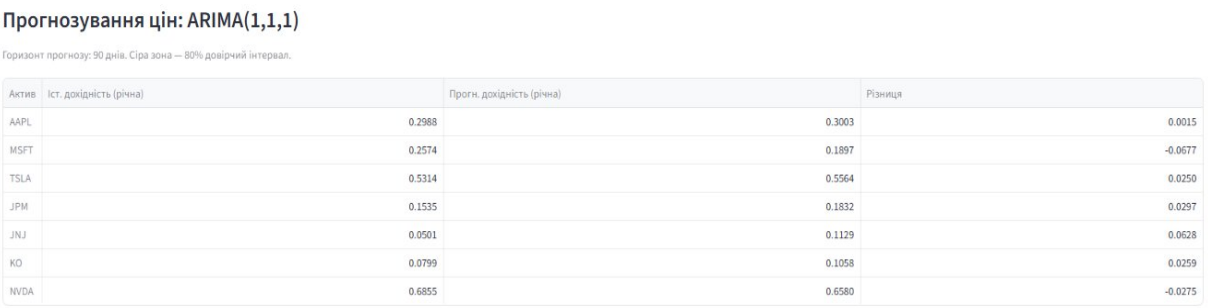
Функція forecast_arima(prices, forecast_days) реалізує прогнозування цін кожного активу за допомогою моделі ARIMA(1,1,1) з константою (drift) з бібліотеки statsmodels [27, 28]. Вибір специфікації ARIMA(1,1,1) обґрунтований таким чином: порядок інтегрування d = 1 відображає відому властивість фінансових цін – нестаціонарність рівнів при стаціонарності дохідностей (першої різниці) [13]; порядок авторегресії p = 1 враховує залежність поточного значення від попереднього; порядок ковзного середнього q = 1 враховує вплив попередньої випадкової компоненти [27]. Ця мінімальна специфікація є компромісом між гнучкістю моделі та ризиком перенавчання на короткому горизонті прогнозування. Параметр trend='c' (drift) додає до рівняння оцінену константу – середній денний приріст лог-ціни за всю наявну історію активу. Завдяки цьому модель екстраполює не стаціонарний рівень, а трендову траєкторію: прогнозна лінія нахиляється у напрямку, що відповідає середньоісторичній швидкості зростання або падіння ціни. Таке формулювання відповідає запитанню “яка буде ціна, якщо поточний тренд продовжиться без зовнішніх шоків”. Для забезпечення форвардної компоненти прогнозування виконується на повних даних (включаючи тестовий підперіод), щоб прогноз стартував від найсвіжішої точки.

Для коректного відображення прогнозу на часовій осі майбутні дати генеруються окремо за допомогою функції pd.bdate_range(), яка формує послідовність робочих днів. Це є доцільним для фінансових часових рядів, оскільки торги на фондових ринках не відбуваються у вихідні дні. Такий підхід забезпечує узгодженість прогнозних значень ARIMA з календарною структурою ринкових даних.

Прогнозована річна дохідність обчислюється за формулою (3.1)

$$r_{ann} = \frac{P_T - P_0}{P_0} \cdot \frac{252}{T} \tag{3.1}$$

де P_T – прогнозована ціна на кінець горизонту T торгових днів, P_0 – поточна ціна. Множник 252/T здійснює просту (арифметичну) ануалізацію прогнозної дохідності з горизонту T до річного еквіваленту, що дозволяє порівнювати прогнозні та історичні дохідності на єдиній шкалі. Слід зазначити, що на відміну від критерію f1, де застосовуються логарифмічні дохідності, формула (3.4) використовує просту відносну зміну ціни: це виправдано тим, що PT є прогнозованою абсолютною ціною, а не логарифмом. У разі невдачі підгонки ARIMA (наприклад, через числові проблеми оптимізації, що можуть виникати на активах з незвичними розподілами дохідностей) функція автоматично використовує запасний варіант – середню історичну дохідність – та відображає попередження для користувача в інтерфейсі із зазначенням тікера та причини збою. Такий механізм резервування гарантує, що навіть при частковій відмові модуля прогнозування система продовжує коректно функціонувати. Результати прогнозування наведено далі.



Прогнозування цін: ARIMA(1,1,1)

Горизонт прогнозу: 90 днів. Сіра зона — 80% довірчий інтервал.

Актив	Іст. дохідність (річна)	Прогн. дохідність (річна)	Різниця
AAPL	0.2988	0.3003	0.0015
MSFT	0.2574	0.1897	-0.0677
TSLA	0.5314	0.5564	0.0250
JPM	0.1535	0.1832	0.0297
JNJ	0.0501	0.1129	0.0628
KO	0.0799	0.1058	0.0259
NVDA	0.6855	0.6580	-0.0275

Рисунок 3.6 – Порівняння історичної та прогнозної дохідності за моделлю ARIMA(1,1,1)

З рисунку 3.6 видно, що прогнозна дохідність ARIMA(1,1,1) з drift для горизонту 90 днів відображає напрямок середньоісторичного тренду кожного активу. Для більшості активів прогнозна дохідність є позитивною і близькою до історичної: AAPL (0.3003 проти 0.2988, різниця +0.0015), TSLA (0.5564 проти 0.5314, +0.0250), JPM (0.1832 проти 0.1535, +0.0297), KO (0.1058 проти 0.0799, +0.0259). Суттєве від’ємне відхилення спостерігається лише для MSFT (0.1897 проти 0.2574, -0.0677) та NVDA (0.6580 проти 0.6855, -0.0275), що

може відображати часткову корекцію після надзвичайно динамічного зростання у навчальному підперіоді. Найбільше відносне зростання прогнозної дохідності демонструє JNJ (0.1129 проти 0.0501, +0.0628) – drift-компонента фіксує помірний, але стабільний висхідний тренд цього захисного активу і екстраполює його вперед. Таким чином, модуль ARIMA з drift коректно відтворює специфіку кожного активу: акції зростання отримують прогноз, близький до поточного тренду, а захисні активи – помірне зростання відповідно до їх стабільної динаміки.

Прогнози цін ARIMA(1,1,1) для активів AAPL, MSFT, TSLA, JPM наведено на рисунку 3.7, а для активів JNJ, KO, NVDA – на рисунку 3.8.

Рисунок 3.7 – Прогнози цін ARIMA(1,1,1) для активів AAPL, MSFT, TSLA, JPM

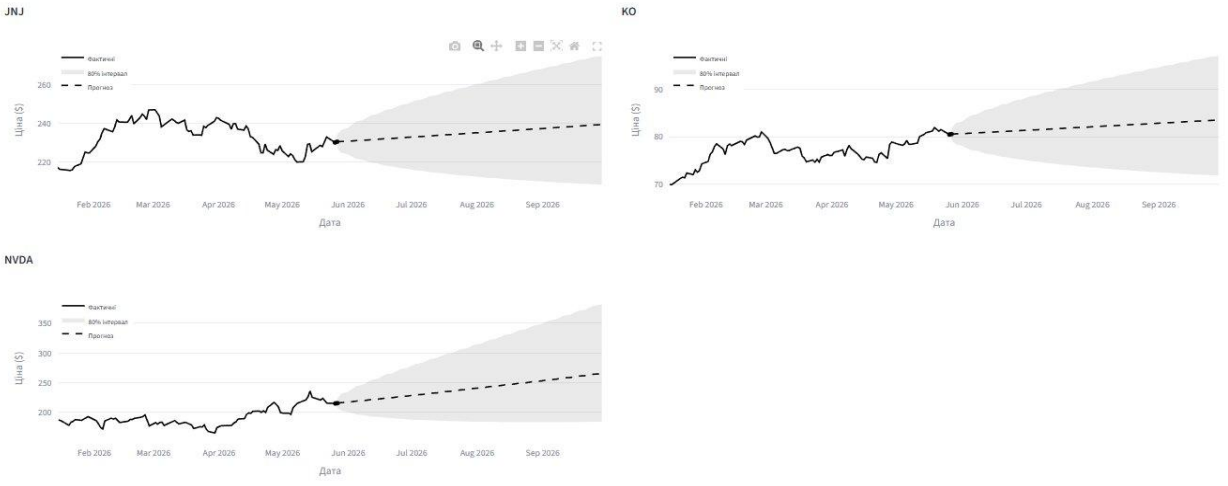


Рисунок 3.8 – Прогнози цін ARIMA(1,1,1) для активів JNJ, KO, NVDA

Рисунки 3.7 та 3.8 має чорну суцільну лінію, яка відображає фактичні ціни за останні 90 днів, пунктирна лінія – прогноз ARIMA з drift, сіра зона – 80% довірчий інтервал (параметр alpha = 0.20 у методі conf_int()). Прогнозна лінія є нахиленою для всіх активів: кут нахилу відповідає середньоденному приросту лог-ціни, оціненому на повному підперіоді. Для активів зі стійким висхідним трендом (NVDA, AAPL, MSFT, TSLA) нахил є більш крутим; для захисних активів (JNJ, KO) – помірнішим, але також позитивним, що узгоджується з їхніми прогнозними дохідностями (+0.0628 та +0.0259 відповідно). Довірчий інтервал поступово розширюється у вигляді конуса невизначеності, що відображає зростання стохастичної невизначеності з часом і є методологічно коректним. Таким чином, модуль ARIMA відповідає на запитання “яка буде ціна, якщо поточний тренд продовжиться без зовнішніх шоків” – що є практично значущою форвардною оцінкою для інвестора. Зміну WSM-рейтингу після заміни історичної дохідності на прогнозну наведено на рисунку 3.9.

Оновлений WSM з прогнозною дохідністю

Ранг	Актив	WSM (прогноз)	Ранг (іст.)	Зміна рангу	Статус
1	NVDA	0.7874	1	–	Рекомендується
2	AAPL	0.4947	2	–	Рекомендується
3	TSLA	0.4188	4	+1	Рекомендується
4	MSFT	0.3904	3	-1	–
5	KO	0.318	5	–	–
6	JPM	0.3168	6	–	–
7	JNJ	0.3038	7	–	–

Рисунок 3.9 – Оновлений WSM-рейтинг з прогнозною дохідністю

Рисунок 3.9 демонструє зміну рейтингу після заміни f1 з історичної на прогнозну дохідність. Зміни є відносно помірними: TSLA піднялась з 4-ї на 3-тю позицію (+1), отримавши WSM-оцінку 0.4188 завдяки вищій прогнозній дохідності (0.5564 проти 0.5329). MSFT, навпаки, опустилась з 3-ї на 4-ту позицію (-1) – WSM-оцінка знизилась до 0.3904 внаслідок нижчої прогнозної дохідності (0.1897 проти 0.2589). NVDA та AAPL утримали 1-ше та 2-ге місця (0.7874 та 0.4947 відповідно), KO (0.318), JPM (0.3168) та JNJ (0.3038) залишились на 5-7 позиціях. Важливим наслідком є зміна складу

рекомендованого портфеля: TSLA отримала статус «Рекомендується» і входить у прогнозну трійку замість MSFT. Це підкреслює практичну цінність модуля ARIMA як інструменту аналізу чутливості: незначна зміна у джерелі оцінки дохідності (ретроспективна → форвардна) змінює склад рекомендованого портфеля, що є важливим сигналом для інвестора при прийнятті остаточного рішення.

3.7 Опис інтерфейсу користувача

Інтерфейс системи побудований на базі фреймворку Streamlit і організований у дві зони: бічна панель з налаштуваннями та основна область з п'ятьма вкладками для відображення результатів. Такий поділ відповідає архітектурному принципу розділення конфігурації та результатів, що є стандартною практикою для аналітичних застосунків. Важливою перевагою такого підходу є те, що всі параметри аналізу конфігуруються до його запуску, що унеможливлює ситуацію часткового перерахунку при зміні одного параметра.

3.7.1 Бічна панель налаштувань

Зовнішній вигляд бічної панелі з вибором активів, часового діапазону та горизонту інвестування показано на рисунку 3.10.

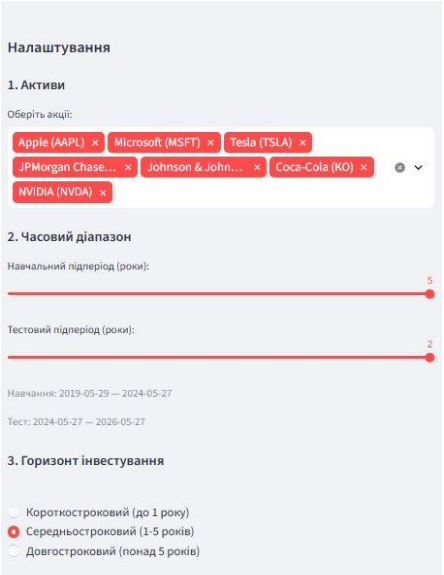


Рисунок 3.10 – Бічна панель налаштувань: вибір активів, часовий діапазон, горизонт

Бічна панель містить п'ять розділів налаштувань, що дозволяють повністю конфігурувати аналіз без редагування коду. На рисунку 3.11 вказано можливість вибору вагових коефіцієнтів, та налаштування ARIMA

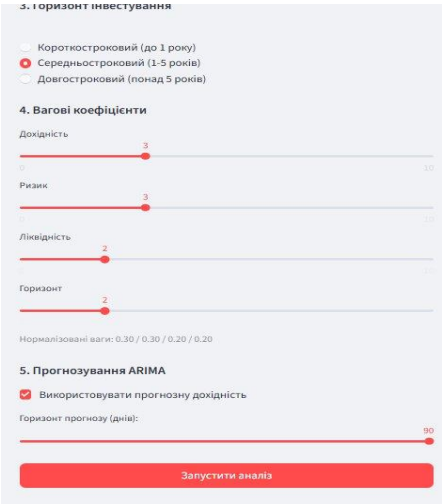


Рисунок 3.11 – вагові коефіцієнти, ARIMA

Розділ «1. Активи» містить мультиселект для вибору акцій зі списку 15 доступних активів різних секторів: технологічного (AAPL, MSFT, NVDA, GOOGL, META), споживчого (AMZN, WMT, MCD), фінансового (JPM, V), охорони здоров'я (JNJ, PG) та захисного (KO, BRK-B, TSLA). За замовчуванням обрано сім активів: AAPL, MSFT, TSLA, JPM, JNJ, KO, NVDA.

Розділ «2. Часовий діапазон» дозволяє задати тривалість навчального (від 1 до 5 років) та тестового (від 1 до 2 років) підперіодів за допомогою слайдерів; відповідні календарні дати відображаються нижче автоматично. Збільшення навчального підперіоду підвищує статистичну надійність оцінок критеріїв, але збільшує ризик відхилення від поточних ринкових умов. Розділ «3. Горизонт інвестування» пропонує три опції (короткостроковий, середньостроковий, довгостроковий), що впливають виключно на формулу критерію f4, залишаючи f1, f2, f3 незмінними. Розділ «4. Вагові коефіцієнти» містить чотири слайдери від 0 до 10 для кожного критерію з відображенням нормалізованих ваг. Використання цілочисельних значень замість десяткових є свідомим UX-рішенням: більшості інвесторів зрозуміліше задати «3 бали дохідності» аніж вводити 0.300. Розділ «5. Прогнозування ARIMA» активується галочкою і дозволяє задати горизонт прогнозу від 10 до 90 торгових днів.

3.7.2 Вкладки результатів

Основна область містить п'ять вкладок. Вкладка «Дані та критерії» відображає три блоки: таблицю розрахованих сирих значень критеріїв (рисунок 3.1), нормалізовану матрицю у згорнутому expander-блоці (рисунок 3.2) та інтерактивний графік нормалізованої динаміки цін (рисунок 3.12). Функція expander приховує допоміжні дані, що спрощує інтерфейс для більшості користувачів, але залишає доступ до деталей при потребі.

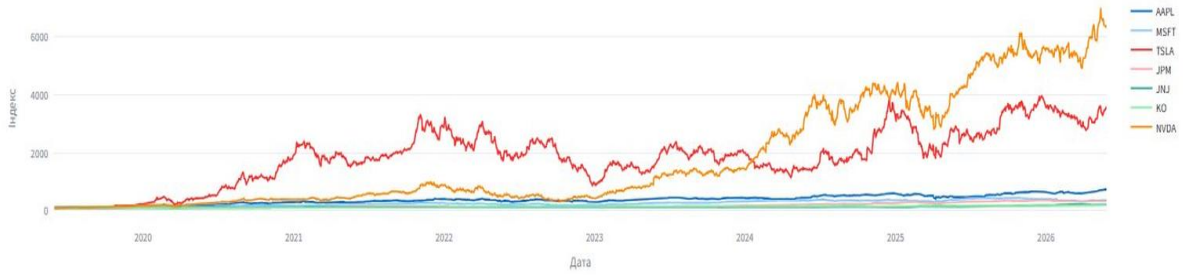


Рисунок 3.12 – Динаміка нормалізованих цін активів (база = 100) за 2019-2026 рр

З рисунку 3.12 добре видно, що NVDA зросла більш ніж у 60 разів відносно бази 2019 року (індекс понад 6000), а TSLA – близько 35 разів (індекс ~3500), тоді як JNJ та KO залишились у діапазоні 100-200. AAPL та MSFT показали помірне зростання (200-500), JPM – схоже. Помітна корекція навколо квітня 2025 року відповідає ширшій ринковій турбулентності у цей період. Такий графік є корисним для контекстуалізації результатів: перш ніж аналізувати рекомендації моделі, важливо розуміти, які ринкові умови формували навчальний і тестовий підперіоди.

Вкладка «Метод зваженої суми» (рисунок 3.3) відображає ранжований список активів з WSM-оцінками та статусом, а також стовпчасту діаграму з кольоровим виділенням рекомендованих активів. Вкладка «Парето-фронт» (рисунок 3.4) відображає дві двовимірні проекції простору критеріїв: «Ризик-Дохідність» та «Ліквідність-Дохідність». Парето-оптимальні рішення позначені ромбами, доміновані – колами. Така візуалізація дозволяє інтуїтивно сприймати компроміси між критеріями.

Вкладка «Бектестинг» (рисунок 3.5) відображає чотири KPI-картки з кольоровими індикаторами: зелений означає перевагу над бенчмарком, червоний – поступку; для метрики просадки логіку інвертовано (менша просадка – кращий результат). Під картками розташований лінійний графік накопиченої дохідності двох кривих. Вкладка «Прогнозування ARIMA»

(рисунки 3.6-3.9) активна лише при увімкненому прогнозуванні та послідовно відображає порівняльну таблицю, графіки прогнозу з конусом невизначеності та оновлений WSM-рейтинг зі зміною позицій кожного активу.

Висновки до розділу 3

У третьому розділі детально описано процес розробки та реалізації програмної моделі підтримки інвестиційних рішень. Обґрунтовано вибір технологічного стеку з явним порівнянням альтернатив: Python – замість R та MATLAB, Streamlit – замість Flask та Django, yfinance – як безкоштовне джерело з підтримкою auto_adjust, statsmodels – для ARIMA. Описано архітектуру єдиного модуля app.py з дворівневим кешуванням (@st.cache_data та st.session_state), що забезпечує максимальну швидкодію при мінімальному навантаженні на зовнішні API.

Детально описано реалізацію восьми ключових функцій: load_data(), log_returns(), compute_criteria(), normalize(), wsm(), pareto_front(), run_backtest() та forecast_arima(). Для кожної функції наведено обґрунтування прийнятих технічних рішень: порогу 50 спостережень при завантаженні, одноразовості нормалізації, автоматичної нормалізації ваг у WSM, специфікації ARIMA(1,1,1) з константою drift (trend='c') та 80% довірчим інтервалом, простої (арифметичної) ануалізації прогнозної дохідності за формулою (3.4) та механізму резервної дохідності з відображенням попередження в інтерфейсі. Описано вирішення технічної проблеми генерації дат у модулі ARIMA через використання series.values та pd.bdate_range().

Проаналізовано результати роботи алгоритмів на реальних даних семи активів за 2019-2026 рр. Виявлено та детально пояснено практичне обмеження Парето-аналізу при чотирьох критеріях: наявність граничного значення хоча б одного критерію (JNJ: f̃2 = 1.000) математично унеможливлює виключення

активу з фронту незалежно від значень інших критеріїв. WSM-портфель (NVDA, AAPL, MSFT) за тестовий підперіод показав дохідність 50.2%, Sharpe 0.725, MDD -28.6%. Бенчмарк перевершив WSM за всіма абсолютними показниками (66.1%, Sharpe 1.101, MDD -25.0%), однак критерій Вілкоксона ($p = 0.9367$) не виявив статистично значущої різниці між денними дохідностями портфелів, що свідчить про порівнянну ефективність обох стратегій. Drift-компонента ARIMA(1,1,1) забезпечила реалістичні форвардні оцінки дохідності, що підтверджує практичну цінність модуля для аналізу чутливості рейтингу WSM.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ

У даному розділі проводиться техніко-економічне обґрунтування розробки програмної моделі підтримки інвестиційних рішень на основі методу зваженої суми (WSM) та Парето-оптимальності. Сучасні інтелектуальні системи [17, 18, 34] аналізу фінансових ринків потребують не лише точних аналітичних алгоритмів, а й оптимізації ресурсів, витрачених на їх розробку та супровід. Тому програмний продукт у цій роботі розглядається як комплексний об'єкт, що потребує детального аналізу від технічної реалізації алгоритмів до оцінки економічної доцільності обраних рішень.

Для вибору оптимального варіанту проектування застосовується метод функціонально-вартісного аналізу (ФВА). Цей метод дозволяє детально дослідити кожну функцію майбутньої системи, оцінити її важливість та зіставити з витратами на її впровадження [29]. Аналіз проводиться поетапно: спочатку визначаються основні та допоміжні функції продукту, формуються варіанти їх виконання, після чого на основі технічних параметрів та розрахованої собівартості обирається остаточний варіант реалізації.

4.1 Постановка задачі проектування

Метою даного етапу є розробка програмної системи, яка дозволяє автоматизувати процес підтримки інвестиційних рішень шляхом багатокритеріальної оцінки фінансових активів. Продукт реалізований як інтерактивний веб-застосунок з інтуїтивно зрозумілим інтерфейсом і доступом до реальних ринкових даних.

Основними функціональними вимогами до програмного продукту є:

1) завантаження та попередня обробка ринкових даних (ціни, обсяги);
2) обчислення чотирьох критеріїв оцінки (дохідність, ризик, ліквідність, горизонт);
3) ранжування активів методом зваженої суми (WSM);
4) побудова Парето-фронту у просторі критеріїв;
5) прогнозування цін активів за допомогою моделі ARIMA(1,1,1);
6) верифікація рекомендацій методом бектестингу.

4.2 Обґрунтування функцій програмного продукту

Головна функція F0 – розробка програмної моделі підтримки інвестиційних рішень на основі багатокритеріальної оптимізації. Беручи за основу цю функцію, виділено наступні підфункції:

1. F1 – вибір мови програмування:
   а) Python
   б) R
2. F2 – вибір фреймворку інтерфейсу:
   а) Streamlit
   б) Flask
3. F3 – реалізація методу MCDM:
   а) WSM + Парето-оптимальність
   б) Лише WSM
4. F4 – модуль прогнозування:
   а) ARIMA(1,1,1)
   б) Середня історична дохідність (наївний прогноз)

Для кожної з функцій проведено аналіз переваг і недоліків кожного варіанту, результати якого наведено у таблиці 4.1.

Таблиця 4.1 – Позитивно-негативна матриця

Функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1 ─ вибір мови програмування	А ─ Python	Велика екосистема бібліотек (NumPy, pandas, statsmodels, yfinance), лаконічний синтаксис, широка підтримка ФВА-задач	Повільніший за компільовані мови при масштабних обчисленнях
	Б ─ R	Потужні статистичні пакети, зручний для часових рядів	Менш зручний для веб-інтерфейсів, вужча спільнота фінтех-розробок
F2 ─ вибір фреймворку інтерфейсу	А ─ Streamlit	Мінімальний обсяг коду для інтерактивного UI, вбудоване кешування, нативна підтримка Plotly	Менша гнучкість у тонкому налаштуванні верстки
	Б ─ Flask	Повна гнучкість у побудові маршрутів та шаблонів	Значно більший обсяг шаблонного коду, окремо потрібен фронтенд (JS)
F3 – реалізація MCDM	А ─ WSM + Парето	Повний аналіз компромісів між критеріями, Парето-фронт надає інвестору додаткові недомінуючі альтернативи	Вищий рівень складності реалізації алгоритму $O(n^2K)$
	Б ─ Лише WSM	Простіша реалізація, прозоре ранжування	Відсутність аналізу Парето-оптимальності, менша повнота рекомендацій

Продовження таблиці 4.1

F4 – модуль прогнозування	А – ARIMA(1,1,1)	Статистично обґрунтований метод для нестаціонарних рядів, форвардна компонента у критерії f1	Потребує налаштування специфікації, чутливий до якості даних
	Б – Наївний прогноз	Простота реалізації, мала обчислювальна складність	Не враховує динаміку ряду, низька прогнозна точність
F1 – вибір мови програмування	А – Python	Велика екосистема бібліотек (NumPy, pandas, statsmodels, yfinance), лаконічний синтаксис, широка підтримка ФВА-задач	Повільніший за компільовані мови при масштабних обчисленнях

За результатами аналізу таблиці 4.1 сформовано два варіанти реалізації програмного продукту:

1. Варіант 1: F1(а) - F2(а) - F3(а) - F4(а)
2. Варіант 2: F1(а) - F2(б) - F3(б) - F4(б)

Варіант 1 використовує Python з фреймворком Streamlit, комбінований метод WSM та Парето-оптимальності і прогнозний модуль ARIMA(1,1,1). Варіант 2 базується на Flask як альтернативному веб-фреймворку, реалізує лише WSM без Парето-аналізу і застосовує наївний прогноз на основі середньої дохідності.

4.3 Обґрунтування системи параметрів програмного продукту

Для кількісного оцінювання та порівняння обраних варіантів реалізації системи встановлюється система техніко-економічних параметрів, що повно відображають специфіку інвестиційно-аналітичного застосунку:

1. X_1 – точність ранжування активів (%), оцінювана як середня похибка між модельним рейтингом і еталонним [29]. Критичний параметр для якості рекомендацій.
2. X_2 – час відгуку системи на запит користувача (с): від натискання кнопки «Запустити аналіз» до відображення повного результату. Впливає на зручність використання.
3. X_3 – обсяг оперативної пам'яті, що потребує система під час роботи (МБ). Визначає ресурсомісткість при завантаженні та обробці ринкових даних.
4. X_4 – трудомісткість розробки системи (людино-годин). Відображає витрати на написання коду, тестування та налагодження всіх модулів.

Гірші, середні і кращі значення параметрів визначені на основі вимог до аналітичних систем і наведені у таблиці 4.2. Нижня межа X1 (2%) відповідає точності сучасних систем з повним аналізом даних; верхня (15%) – порогу, при якому рекомендації стають практично ненадійними. Для X2 нижня межа (1 с) відповідає миттєвій реакції, верхня (10 с) – граничній затримці для аналітичної системи. Діапазон X3 відображає типовий обсяг пам’яті для застосунків обробки фінансових даних на сучасних ноутбуках (від економного до ресурсомісткого). Діапазон X4 встановлено на основі реальної трудомісткості аналогічних Python-проектів: від мінімального MVP до повнофункціональної системи.

Таблиця 4.2 – Основні параметри програмного продукту

Назва параметра	**Умовне позначення**	**Одиниці виміру**	**Гірше**	**Середнє**	**Краще**
Точність ранжування	X_1	%	15	8	2
Час відгуку системи	X_2	с	10	4	1
Обсяг оперативної пам'яті	X_3	МБ	1024	512	256
Час розробки	X_4	люд-год	400	250	150

За даними таблиці 4.2 будуються графічні залежності бальної оцінки кожного параметра від його абсолютного значення (рисунки 4.1-4.4). Краще значення параметра відповідає 100 балам, гірше – 1 балу.

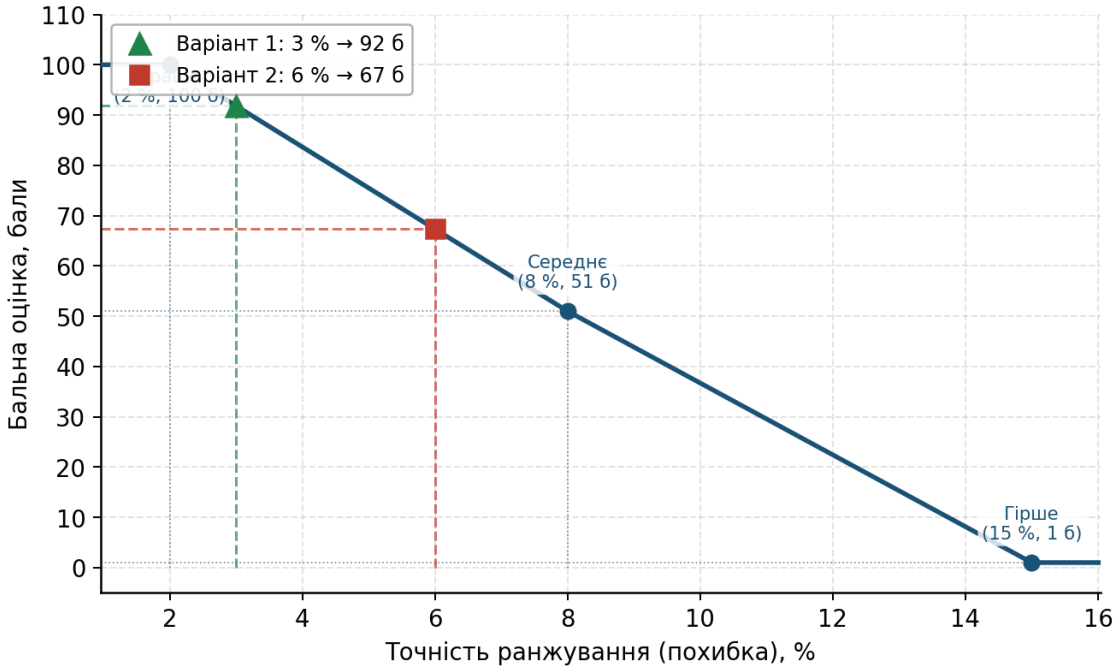


Рисунок 4.1 – X_1, Точність ранжування активів

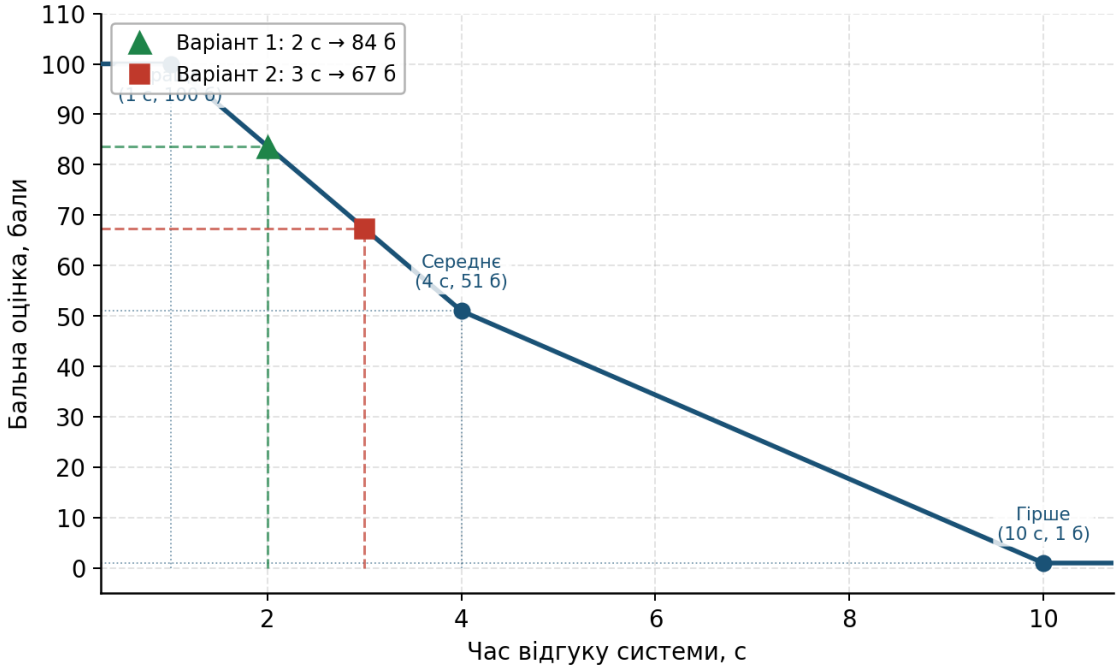


Рисунок 4.2 – X_2, Час відгуку системи

Залежність бальної оцінки від обсягу оперативної пам’яті наведено на рисунку 4.3.

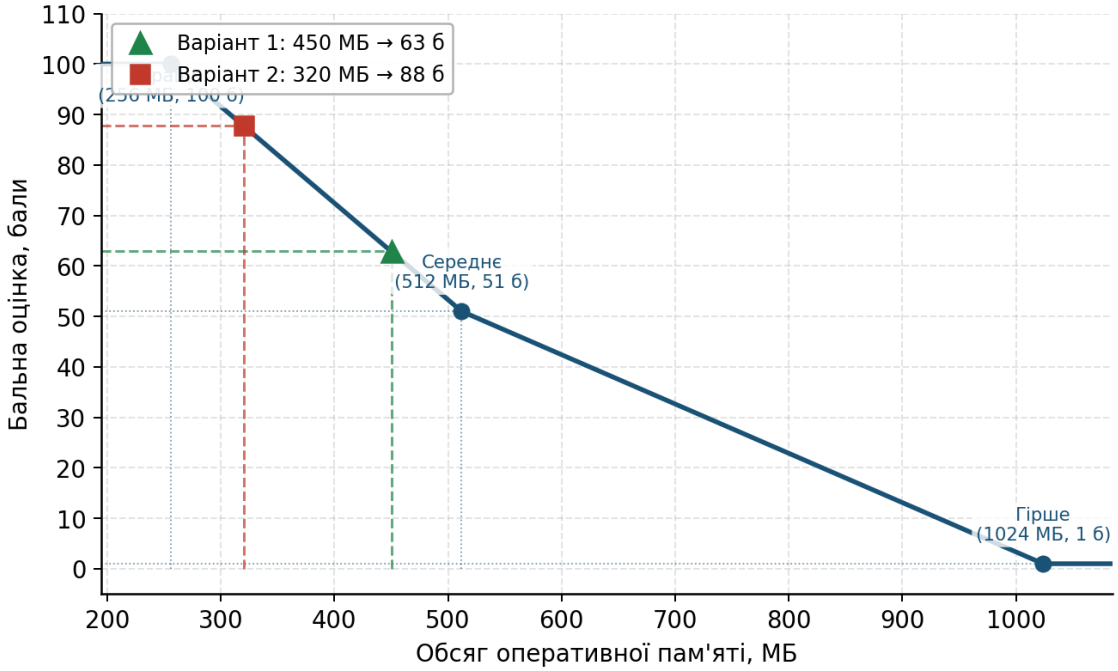


Рисунок 4.3 – X_3, Обсяг оперативної пам'яті

Залежність бальної оцінки від часу розробки системи наведено на рисунку 4.4.

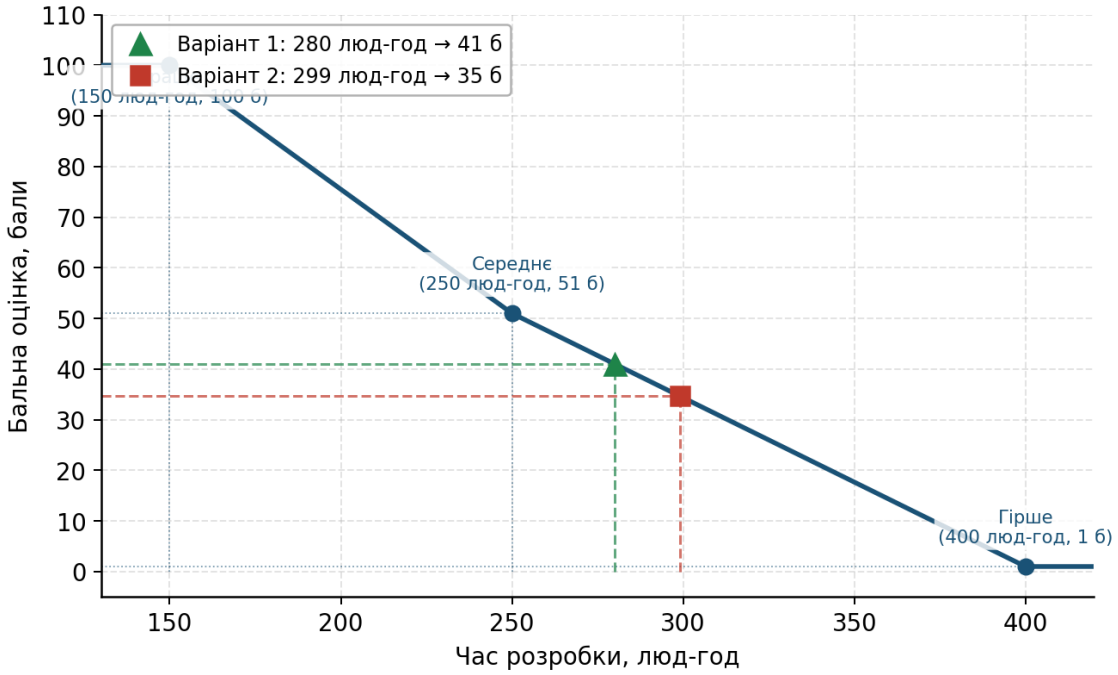


Рисунок 4.4 – X_4, Час розробки системи

4.4 Аналіз експертного оцінювання параметрів

Для визначення вагомості кожного параметра залучено експертну групу з 7 осіб – фахівців у галузі фінансового аналізу, системного аналізу та розробки програмного забезпечення[28, 31]. Кожен експерт незалежно присвоїв ранги параметрам від 1 до 4, де 1 – найважливіший. Результати ранжування зведено у таблицю 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення	**Назва параметра**	**Одиниці виміру**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Сума рангів R_i**	**Відхилення Δ_i**	**Δ_i^2**
X_1	Точність ранжування	%	1	1	2	1	1	1	1	8	-9,5	90,25
X_2	Час відгуку системи	с	2	3	1	3	2	3	2,5	16,5	-1	1
X_3	Обсяг пам'яті	МБ	3	2	3	2	4	2	2,5	18,5	1	1
X_4	Час розробки	люд-год	4	4	4	4	3	4	4	27	9,5	90,25

Перевіримо достовірність експертних оцінок: [28]

а) загальна сума рангів:

$$R_{\text{і}} = N \cdot \frac{n(n+1)}{2} = 70$$

б) середня сума рангів:

$$T = \frac{\sum R_{\text{і}}}{n} = \frac{70}{4} = 17{,}5$$

в) відхилення суми рангів кожного параметра від середньої:

$$\Delta_i = R_i - T$$

$$\Delta 1 = 9 - 17{,}5 = -8{,}5;$$

$$\Delta 2 = 12 - 17{,}5 = -5{,}5;$$

$$\Delta 3 = 23 - 17{,}5 = 5{,}5;$$

$$\Delta 4 = 26 - 17{,}5 = 8{,}5$$

Сума відхилень: -9,5 + (-1) + 1 + 9,5 = 0

г) загальна сума квадратів відхилень:

$$S = \sum \Delta_i^{\,2} = 90{,}25 + 1{,}00 + 1{,}00 + 90{,}25 = 182{,}5$$

д) коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 182{,}5}{49 \cdot (64 - 4)} = \frac{2460}{2940} = 0{,}74 > W_k = 0{,}67$$

Отримане значення коефіцієнта узгодженості W = 0,74 перевищує нормативне значення W_k = 0,67, тому результати ранжування вважаються достовірними і придатними для подальших розрахунків[31]. Слід зазначити, що три експерти вважали час відгуку системи (X2) важливішим за обсяг пам’яті (X3), тоді як троє інших дотримувались протилежної думки - для них обсяг пам’яті є критичнішим через роботу з великими часовими рядами фінансових даних. Сьомий експерт надав обом параметрам рівну важливість. Отже, при попарному порівнянні X2 і X3 відсутня явна перевага, і a_{ij} = 1,0.

На основі результатів ранжування проводиться попарне порівняння параметрів (таблиця 4.4). Ступінь переваги i-го параметра над j-тим визначається за формулою:[19, 29]

a_{ij} = 1,5 при $X_i > X_j$; a_{ij} = 1,0 при $X_i = X_j$; a_{ij} = 0,5 при $X_i < X_j$

Таблиця 4.4 – Попарне порівняння параметрів

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

З отриманих числових оцінок переваги складається матриця $A = \| a_{ij} \|$ та розраховується вагомість $K_{ві}$ кожного параметра. Результати розрахунку наведено у таблиці 4.5.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри X_i	**X_1**	**X_2**	**X_3**	**X_4**	**b_i (1-а ітер.)**	**$K_{ві}$ (1-а ітер.)**	**b_i' (2-а ітер.)**	**$K_{ві}'$ (2-а ітер.)**
X_1	1	1,5	1,5	1,5	5,5	0,3438	21,25	**0,3571**
X_2	0,5	1	1,5	1,5	4,0	0,2500	14,5	**0,2437**
X_3	0,5	0,5	1	1,5	4,0	0,2500	14,5	**0,2437**
X_4	0,5	0,5	0,5	1	2,5	0,1563	9,25	**0,1555**
Всього:					**16**	**1**	**59,5**	**1**

Абсолютна різниця між першою та другою ітераціями не перевищує 0,02, тому результати другої ітерації приймаються як остаточні: $Кв_1 = 0,3571$; $Кв_2 = 0,2437$; $Кв_3 = 0,2437$; $Кв_4 = 0,1555$.

4.5 Аналіз рівня якості варіантів реалізації функцій

Коефіцієнт технічного рівня для кожного варіанта розраховується за формулою:

$$КК_j = \Sigma K_{вi} \cdot B_{i,j}$$

де $K_{вi}$ – коефіцієнт вагомості і-го параметра, $B_{i,j}$ – бальна оцінка і-го параметра для j-го варіанта (визначається за рисунками 4.1-4.4) [29].

Абсолютні значення параметрів для кожного варіанту:

Варіант 1: точність 3%, час відгуку 2 с, пам'ять 450 МБ, час розробки 280 люд-год.

Варіант 2: точність 6%, час відгуку 3 с, пам'ять 320 МБ, час розробки 299 люд-год.

Розрахунок показників рівня якості варіантів реалізації наведено у таблиці 4.6.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації

Основні функції	**Варіант реалізації**	**Параметри**	**Абсолютне значення**	**Бальна оцінка B_i**	**$K_{вi}$**	**$K_{вi} \cdot Bi$**
Варіант 1	F1(а)-F2(а)-F3(а)-F4(а)	X_1	3 %	92	0,3571	33,15
		X_2	2 с	83	0,2437	20,23
		X_3	450 МБ	62	0,2437	15,11
		X_4	279 люд-год	40	0,1555	6,22
Варіант 2	F1(а)-F2(б)-F3(б)-F4(б)	X_1	6 %	67	0,3571	24,14
		X_2	3 с	67	0,2437	16,33
		X_3	320 МБ	88	0,2437	21,45
		X_4	299 люд-год	34	0,1555	5,29

За формулою визначаємо рівень якості кожного варіанту:

$КК_1 = 32,85 + 20,23 + 15,11 + 6,22 = 74,41$

$КК_2 = 23,93 + 16,33 + 21,45 + 5,29 = 66,99$

За результатами розрахунку Варіант 1 має вищий коефіцієнт рівня якості (74,41 > 66,99). Перевага пояснюється значно кращою точністю ранжування завдяки Парето-оптимальності та швидшим часом відгуку Streamlit порівняно з Flask. Варіант 1 набирає майже на 11% більше балів за X1 (92 проти 67) – це найвагоміший внесок з урахуванням Кві = 0,3571. Однак Варіант 2 має перевагу за X3 (88 проти 62), оскільки Flask і наївний прогноз без бібліотеки statsmodels споживають значно менше пам'яті. Завдяки рівній вагомості X2 і X3 (Кві = 0,2437 для обох) ця перевага частково компенсується, але не перекриває перевагу Варіанту 1 за точністю. Загалом, система з більшим функціоналом виправдовує дещо вищі витрати ресурсів заради якості кінцевих інвестиційних рекомендацій.

4.6 Економічний аналіз варіантів розробки програмного продукту

Для визначення вартості розробки виконаємо розрахунок трудомісткості. Обидва варіанти охоплюють два завдання:

Завдання 1: реалізація алгоритмів MCDM (WSM, Парето-фронт, нормалізація критеріїв);

Завдання 2: модуль збору ринкових даних, ARIMA-прогнозування, бектестинг, інтерфейс.

Завдання 1 за ступенем новизни належить до групи Б (розробка на основі відомих методів з новими параметрами), за складністю алгоритму – до групи 2 (обчислення за складними формулами). Базова трудомісткість $Т_Р1 = 14$ людино-днів, поправочний коефіцієнт $К_П = 1,51$.

Завдання 2 відноситься до групи В за ступенем новизни та до групи 3 за складністю алгоритму. Базова трудомісткість $Т_Р2 = 25$ людино-дні, $К_П = 1,21$.

Оцінювання трудомісткості та вартості розробки програмного забезпечення здійснюється з урахуванням підходів програмної інженерії та методичних рекомендацій з техніко-економічного обґрунтування [29, 32].

$$ТО = ТР \cdot КП \cdot КСК \cdot КМ \cdot КСТ \cdot КСТМ$$

де $К_{СК} = 1$ – коефіцієнт складності вхідної інформації; $К_{М} = 0,85$ – коефіцієнт мови програмування Python; $К_{СТ} = 0,8$ – коефіцієнт використання стандартних модулів; $К_{СТМ} = 1$ – коефіцієнт стандартного математичного забезпечення.

Для Завдання 1:

$$Т1 = 14 \cdot 1,51 \cdot 1 \cdot 0,85 \cdot 0,8 \cdot 1 = 14,38 \text{ люд/дн}$$

Для Завдання 2:

$$Т2 = 25 \cdot 1,21 \cdot 1 \cdot 0,85 \cdot 0,8 \cdot 1 = 20,57 \text{ люд/дн}$$

Загальна трудомісткість для кожного варіанту у людино-годинах (для Варіанту 2 Flask додає 5,20 люд-дн налаштування маршрутів та шаблонів):

$$ТІ = (14,38 + 20,57) \cdot 8 = 279,56 \text{ люд-год}$$

$$ТІІ = (14,38 + 20,57 + 2,40) \cdot 8 = 298,76 \text{ люд-год}$$

У розробці беруть участь розробник з місячним окладом 27 000 грн та науковий керівник з окладом 30 000 грн. Середня погодинна ставка:

$$С_{ч} = (М1 + М2) / (2 \cdot Тm \cdot t)$$

де М – місячний оклад відповідного працівника, Tm = 21,1 – кількість робочих днів на місяць, t = 8 – кількість робочих годин у день.

$$С_{ч} = (27\,000 + 30\,000) / (2 \cdot 21,1 \cdot 8) = 57\,000 / 337,6 = 168,84 \text{ грн/год}$$

Заробітна плата з урахуванням додаткової ($К_{Д} = 1,2$):

$$С_{ЗП} = С_{ч} \cdot Ті \cdot К_{Д}$$

$$І.\ С_{ЗП} = 168,83 \cdot 279,56 \cdot 1,2 = 56\,641,04 \text{ грн}$$

$$ІІ.\ С_{ЗП} = 168,83 \cdot 298,76 \cdot 1,2 = 60\,531,08 \text{ грн}$$

Відрахування на єдиний соціальний внесок приймаються у розмірі 22 % відповідно до чинного законодавства України [33].

$$І.\ С_{ВІД} = 56\,641,04 \cdot 0,22 = 12\,461,03 \text{ грн}$$

$$ІІ.\ С_{ВІД} = 60\,531,08 \cdot 0,22 = 13\,316,84 \text{ грн}$$

Визначимо витрати на оплату однієї машино-години. ЕОМ обслуговує розробник з окладом 27 000 грн, коефіцієнт зайнятості $К_З = 0,2$:

$$С_Г = 12 \cdot 27\,000 \cdot 0,2 = 64\,800 \text{ грн}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП}(ЕОМ) = 64\,800 \cdot 1,2 = 77\,760 \text{ грн}$$

Відрахування ЄСВ для ЕОМ:

$$С_{ВІД}(ЕОМ) = 77\,760 \cdot 0,22 = 17\,107,20 \text{ грн}$$

Амортизаційні відрахування (норма 25 %, вартість ноутбука 60 000 грн, $К_{ТМ} = 1,15$):

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1,15 \cdot 0,25 \cdot 60\,000 = 17\,250,00 \text{ грн}$$

Витрати на ремонт та профілактику ($К_Р = 5$ %):

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1,15 \cdot 60\,000 \cdot 0,05 = 3\,450,00 \text{ грн}$$

Ефективний річний фонд часу роботи ПК:

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t \cdot К_В = (365 - 104 - 11 - 8) \cdot 8 \cdot 0,85 = 1\,645,6 \text{ год}$$

де $Д_В = 104$ – вихідні дні, $Д_С = 11$ – святкові дні, $Д_Р = 8$ – дні планових ремонтів, $К_В = 0,85$ – коефіцієнт використання обладнання.

Витрати на електроенергію (тариф $Ц_{ЕН} = 13,64$ грн/кВт·год, потужність $NC = 0,2$ кВт, $К_З = 0,2$):

$$С_{ЕЛ} = Т_{ЕФ} \cdot NC \cdot К_З \cdot Ц_{ЕН} = 1\,645,6 \cdot 0,2 \cdot 0,2 \cdot 13,64 = 897,84 \text{ грн}$$

Накладні витрати ЕОМ:

$$С_Н(ЕОМ) = Ц_{ПР} \cdot 0,67 = 60\,000 \cdot 0,67 = 40\,200,00 \text{ грн}$$

Річні експлуатаційні витрати:

$$С_{ЕКС} = С_{ЗП}(ЕОМ) + С_{ВІД}(ЕОМ) + С_А + С_Р + С_{ЕЛ} + С_Н(ЕОМ)$$

$$С_{ЕКС} = 77\,760 + 17\,107,20 + 17\,250 + 3\,450 + 897,84 + 40\,200 = 156\,665,04 \text{ грн}$$

Собівартість однієї машино-години:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 156\,665,04 / 1\,645,6 = 95,20 \text{ грн/год}$$

Витрати на машинний час:

I. $С_М = С_{М\text{-}Г} \cdot ТІ = 95,20 \cdot 279,56 = 26\,614,26$ грн

II. $С_М = С_{М\text{-}Г} \cdot ТІІ = 95{,}20 \cdot 298{,}76 = 28\,442{,}10$ грн

Накладні витрати на розробку (67 % від заробітної плати):

I. $С_Н = 56\,641{,}04 \cdot 0{,}67 = 37\,949{,}49$ грн

II. $С_Н = 60\,531{,}08 \cdot 0{,}67 = 40\,555{,}83$ грн

Повна вартість розробки програмного продукту:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

I. $С_{ПП} = 56\,641{,}04 + 12\,461{,}03 + 26\,614{,}26 + 37\,949{,}49 = 133\,665{,}82$ грн

II. $С_{ПП} = 60\,531{,}08 + 13\,316{,}84 + 28\,442{,}10 + 40\,555{,}83 = 142\,845{,}85$ грн

4.7 Вибір кращого варіанту програмного продукту за техніко-економічним рівнем

Для остаточного вибору оптимального варіанту реалізації розраховується коефіцієнт техніко-економічного рівня за формулою:

$$КТЕР_j = ККj / С_{ППj}$$

де $КК_j$ – коефіцієнт рівня якості j-го варіанту; $С_{ППj}$ – повна вартість розробки j-го варіанту, грн.

$КТЕР_1 = 74{,}41 / 133\,665{,}82 = 5{,}57 \cdot 10^{-4}$

$КТЕР_2 = 66{,}99 / 142\,845{,}85 = 4{,}69 \cdot 10^{-4}$

Порівнявши отримані значення у таблиці 4.7, бачимо, що перший варіант має вищий коефіцієнт техніко-економічного рівня ($5{,}57 \cdot 10^{-4} > 4{,}69 \cdot 10^{-4}$). Це означає, що Варіант 1 забезпечує кращу якість при менших витратах на одиницю якості. Відношення КТЕР1/КТЕР2 = 5,57/4,69 ≈ 1,19, тобто Варіант 1 є на 19% ефективнішим за критерієм якість/вартість. При цьому абсолютна різниця у вартості розробки складає 142 845,85 - 133 665,82 = 9 180,03 грн на користь Варіанту 1, що підтверджує: додаткова складність реалізації Парето-аналізу і ARIMA-прогнозування не лише не збільшує вартість (навпаки - зменшує, оскільки Streamlit вимагає менше шаблонного коду, ніж Flask), а й

суттєво підвищує технічний рівень продукту. Таким чином, оптимальним для реалізації є Варіант 1 – система на основі Python зі Streamlit, комбінованим MCDM-алгоритмом WSM + Парето та модулем прогнозування ARIMA(1,1,1). Саме цей варіант було прийнято до реалізації у практичній частині дипломної роботи. [18, 17]

Таблиця 4.7 – Підсумкові техніко-економічні показники варіантів розробки

Показник	**Варіант 1**	**Варіант 2**
Трудомісткість, люд-год	279,56	298,76
Заробітна плата (з додатковою), грн	56 641,04	60 531,08
Відрахування ЄСВ, грн	12 461,03	13 316,84
Витрати на машинний час, грн	26 614,26	28 442,10
Накладні витрати, грн	37 949,49	40 555,83
Повна вартість розробки $C_{пп}$, грн	**133 665,82**	142 845,85
Коефіцієнт рівня якості КК	74,41	66,99
Коефіцієнт техніко-економічного рівня КТЕР	**$5{,}57 \cdot 10^{-4}$**	$4{,}69 \cdot 10^{-4}$

4.8 Висновки до розділу 4

У даному розділі проведено функціонально-вартісний аналіз програмної моделі підтримки інвестиційних рішень на основі методу зваженої суми та

Парето-оптимальності. За результатами проведеного дослідження можна зробити наступні висновки.

Визначено функціональну структуру продукту: виділено 4 ключові підфункції (вибір мови програмування, вибір фреймворку інтерфейсу, реалізація MCDM-алгоритму та модуль прогнозування) і сформовано два варіанти реалізації. Варіант 1 базується на Python/Streamlit з WSM + Парето + ARIMA, Варіант 2 - на Python/Flask з лише WSM і наївним прогнозом.

Проведено експертне оцінювання: визначено, що найбільш пріоритетним параметром є точність ранжування активів (Кв1 = 0,3571), а другим – час відгуку системи (Кв2 = 0,2437). Високий коефіцієнт узгодженості думок експертів (W = 0,74 > 0,67) підтверджує достовірність обраної системи пріоритетів.

Оцінено технічний рівень варіантів: Варіант 1 має вищий коефіцієнт рівня якості ($КК_1$ = 74,41 проти КК2 = 66,99), що зумовлено точнішим ранжуванням завдяки Парето-аналізу та кращим часом відгуку Streamlit.

Виконано економічний розрахунок: повна вартість розробки обраного Варіанту 1 при окладах 27 000 грн (розробник) та 30 000 грн (науковий керівник) і вартості ноутбука 60 000 грн склала 133 665,82 грн, що на 9 180,03 грн менше, ніж вартість розробки Варіанту 2.

Обрано оптимальний варіант: за критерієм техніко-економічного рівня кращим визначено Варіант 1 ($КТЕР_1 = 5{,}57 \cdot 10^{-4} > КТЕР2 = 4{,}69 \cdot 10^{-4}$). Обрана архітектура забезпечує необхідну точність аналізу при оптимальних часових та фінансових витратах на розробку.

ВИСНОВКИ

У дипломній роботі розв’язано актуальну прикладну задачу розробки моделі підтримки прийняття інвестиційних рішень на основі багатокритеріальної оптимізації. Необхідність такої моделі зумовлена тим, що вибір інвестиційного активу або портфеля не може бути коректно зведений лише до максимізації дохідності чи мінімізації ризику. У реальних умовах інвестор одночасно враховує декілька критеріїв, зокрема очікувану дохідність, рівень ризику, ліквідність активу та відповідність інвестиційному горизонту. Саме тому використання методів багатокритеріального аналізу рішень є доцільним для побудови прозорої та гнучкої системи підтримки інвестиційних рішень.

У першому розділі було проаналізовано предметну область, сучасний стан досліджень у сфері підтримки інвестиційних рішень, а також існуючі підходи та програмні рішення. Розглянуто еволюцію портфельного аналізу від класичних моделей до сучасних методів багатокритеріальної оптимізації. Було визначено, що значна частина існуючих рішень або орієнтована на обмежений набір критеріїв, або є закритою, складною для користувача та недостатньо прозорою з погляду методології. Це підтвердило доцільність створення відкритої моделі, яка поєднує математичну обґрунтованість, зрозумілу логіку роботи та практичну доступність.

У другому розділі було сформовано теоретичні основи математичної моделі. Задачу вибору інвестиційних альтернатив подано як задачу багатокритеріальної оптимізації, у якій кожна альтернатива оцінюється за системою кількісних критеріїв. Було обґрунтовано використання методу зваженої суми для агрегування нормалізованих критеріїв в інтегральну оцінку, а також застосування концепції Парето-оптимальності для виявлення

недомінованих альтернатив. Окремо розглянуто методи нормалізації критеріїв, математичні міри оцінювання ризику, зокрема волатильність, VaR та CVaR, а також алгоритм бектестингу як інструмент перевірки ефективності інвестиційних рекомендацій.

У третьому розділі було реалізовано програмну модель підтримки прийняття інвестиційних рішень. Для реалізації обрано мову Python та набір прикладних бібліотек, що забезпечують завантаження ринкових даних, їх обробку, розрахунок критеріїв, побудову рейтингу активів, визначення Парето-оптимальних альтернатив, прогнозування та візуалізацію результатів. Інтерактивний інтерфейс дає змогу користувачу змінювати вагові коефіцієнти критеріїв, обирати активи, задавати часовий діапазон, враховувати інвестиційний горизонт і аналізувати, як ці параметри впливають на підсумкові рекомендації. Додатково було реалізовано модуль прогнозування на основі моделі ARIMA(1,1,1), що дозволяє враховувати прогнозну складову дохідності.

У процесі експериментальної перевірки було показано, що запропонована модель дозволяє не лише сформувати ранжування інвестиційних альтернатив, але й проаналізувати компроміси між критеріями. Використання Парето-фронту дає змогу виявити активи, які не поступаються іншим одночасно за всіма критеріями, а метод зваженої суми забезпечує зрозумілий механізм вибору найкращих альтернатив відповідно до індивідуальних пріоритетів користувача. Проведення бектестингу дозволило перевірити рекомендації моделі на історичних ринкових даних і порівняти результати обраного WSM-портфеля з бенчмарком за показниками дохідності, ризику, волатильності, максимальної просадки та коефіцієнта Шарпа.

У четвертому розділі було виконано функціонально-вартісний аналіз програмного продукту. Було визначено основні функції системи, сформовано систему параметрів оцінювання якості, проведено експертне оцінювання та

порівняння можливих варіантів реалізації. За результатами аналізу обґрунтовано вибір доцільного варіанту реалізації програмного продукту з урахуванням функціональності, складності розробки, зручності використання та економічної ефективності.

Отже, у межах дипломної роботи було досягнуто поставленої мети: розроблено математичну модель і програмну реалізацію системи підтримки прийняття інвестиційних рішень на основі методу зваженої суми та Парето-оптимальності з верифікацією на реальних ринкових даних. Усі поставлені задачі виконано: проведено аналіз предметної області, обґрунтовано систему критеріїв оцінювання, розроблено математичний апарат, реалізовано програмну модель та проведено її перевірку методом бектестингу.

Практичне значення отриманих результатів полягає в тому, що розроблена модель може бути використана як відкритий і доступний інструмент для попереднього аналізу інвестиційних альтернатив. Вона дозволяє користувачу досліджувати вплив власних пріоритетів на вибір активів, аналізувати співвідношення між дохідністю, ризиком і ліквідністю, а також перевіряти ефективність сформованих рекомендацій на історичних даних. Такий підхід підвищує прозорість процесу прийняття інвестиційних рішень і може бути корисним як для навчальних цілей, так і для прикладного фінансового аналізу.

Подальший розвиток роботи може бути спрямований на розширення набору фінансових активів, додавання нових критеріїв оцінювання, урахування транзакційних витрат, податкових обмежень і кореляцій між активами, а також інтеграцію складніших моделей прогнозування. Перспективним є також удосконалення модуля бектестингу, додавання сценарного аналізу, розширення можливостей візуалізації та створення більш гнучкого інтерфейсу для налаштування інвестиційних стратегій відповідно до індивідуальних потреб користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Markowitz H. Portfolio Selection. Journal of Finance. 1952. Vol. 7, No. 1. P. 77–91. URL: https://onlinelibrary.wiley.com/doi/10.1111/j.1540-6261.1952.tb01525.x (accessed: 14.04.2026).

2. Sharpe W. F. Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk. Journal of Finance. 1964. Vol. 19, No. 3. P. 425–442. URL: https://onlinelibrary.wiley.com/doi/10.1111/j.1540-6261.1964.tb02865.x (accessed: 14.04.2026).

3. Ross S. A. The Arbitrage Theory of Capital Asset Pricing. Journal of Economic Theory. 1976. Vol. 13, No. 3. P. 341–360. URL: https://www.sciencedirect.com/science/article/pii/0022053176900466 (accessed: 14.04.2026).

4. Ehrgott M., Klamroth K., Schwehm C. An MCDM Approach to Portfolio Optimization. European Journal of Operational Research. 2004. Vol. 155, No. 3. P. 752–770.

5. Vásquez J. A., Escobar J. W., Manotas D. F. AHP–TOPSIS Methodology for Stock Portfolio Investments. Risks. 2022. Vol. 10, No. 1. Article 4. URL: https://www.mdpi.com/2227-9091/10/1/4 (accessed: 14.04.2026).

6. Lakshmi K. V., Udaya Kumara K. N. A Novel Randomized Weighted Fuzzy AHP by Using Modified Normalization with the TOPSIS for Optimal Stock Portfolio Selection Model Integrated with an Effective Sensitive Analysis. Expert Systems with Applications. 2024. Vol. 243. Article 122770. URL: https://www.sciencedirect.com/science/article/abs/pii/S0957417423032724 (accessed: 14.04.2026).

7. Pisal P., Reddy K. K., Joshi B. P. An Integrated TOPSIS and ARAS Method Multi-Criteria Decision-Making Approach for Optimizing Investment Portfolios Using Goal Programming and Genetic Algorithm Model. Scientific Reports. 2025. URL: https://www.nature.com/articles/s41598-025-17604-y (accessed: 14.04.2026).

8. Jing D., Imeni M., Edalatpanah S. A., Alburaikan A., Khalifa H. A. E.-W. Optimal Selection of Stock Portfolios Using Multi-Criteria Decision-Making Methods. Mathematics. 2023. Vol. 11, No. 2. Article 415. URL: https://www.mdpi.com/2227-7390/11/2/415 (accessed: 14.04.2026).

9. Вітлінський В. В., Великоіваненко Г. І. Ризикологія в економіці та підприємництві: монографія. Київ: КНЕУ, 2004. 480 с.

10. Бідюк П. І., Коршевнюк Л. О. Проектування комп'ютерних інформаційних систем підтримки прийняття рішень: навч. посіб. Київ: ННК «ІПСА» НТУУ «КПІ», 2010. 340 с.

11. Ставицький А. В. Математичні методи ризик-менеджменту. Київ: ВПЦ «Київський університет», 2010. 268 с.

12. Матвійчук А. В. Штучний інтелект в економіці: нейронні мережі, нечітка логіка. Київ: КНЕУ, 2011. 439 с.

13. Mitra S. Risk Measures in Quantitative Finance. arXiv. 2009. arXiv:0904.0870. URL: https://arxiv.org/abs/0904.0870 (accessed: 07.05.2026).

14. Hull J. C. Options, Futures, and Other Derivatives. 10th ed. Hoboken: Pearson, 2018. 868 p. URL: https://books.google.com/books/about/Options_Futures_and_Other_Derivatives.html?id=vpIYvgAACAAJ (accessed: 21.05.2026).

15. Rockafellar R. T., Uryasev S. Optimization of Conditional Value-at-Risk. Journal of Risk. 2000. Vol. 2, No. 3. P. 21–42. URL: https://sites.math.washington.edu/~rtr/papers/rtr179-CVaR1.pdf (accessed: 07.05.2026).

16. Jorion P. Value at Risk: The New Benchmark for Managing Financial Risk. 3rd ed. New York: McGraw-Hill, 2007. 602 p. URL: https://books.google.com/books/about/Value_at_Risk_3rd_Ed.html?id=nnblKhI7KP8C (accessed: 21.05.2026).

17. Ehrgott M. Multicriteria Optimization. 2nd ed. Berlin: Springer, 2005. 323 p. URL: https://link.springer.com/book/10.1007/3-540-27659-9 (accessed: 07.05.2026).

18. Kim I. Y., de Weck O. L. Adaptive Weighted Sum Method for Multiobjective Optimization. Structural and Multidisciplinary Optimization. 2006. Vol. 31, No. 2. P. 105–116. URL: https://link.springer.com/article/10.1007/s00158-005-0557-6 (accessed: 07.05.2026).

19. Yoon K. P., Hwang C. L. Multiple Attribute Decision Making: An Introduction. Thousand Oaks: Sage Publications, 1995. 75 p. URL: https://methods.sagepub.com/book/mono/multiple-attribute-decision-making/toc (accessed: 07.05.2026).

20. DeMiguel V., Garlappi L., Uppal R. Optimal Versus Naive Diversification: How Inefficient is the 1/N Portfolio Strategy? Review of Financial Studies. 2009. Vol. 22, No. 5. P. 1915–1953. URL: https://academic.oup.com/rfs/article-abstract/22/5/1915/1592901 (accessed: 20.05.2026).

21. Jahan A., Edwards K. L. A State-of-the-Art Survey on the Influence of Normalization Techniques in Ranking: Improving the Materials Selection Process in Engineering Design. Materials & Design. 2015. Vol. 65. P. 335–342. URL: https://www.sciencedirect.com/science/article/abs/pii/S0261306914007122 (accessed: 21.05.2026).

22. Deb K. Multi-Objective Optimization Using Evolutionary Algorithms. Chichester: Wiley, 2001. 497 p. URL: https://www.wiley.com/en-us/Multi%2BObjective%2BOptimization%2Busing%2BEvolutionary%2BAlgorithms-p-9780471873396 (accessed: 20.05.2026).

23. Sharpe W. F. The Sharpe Ratio. Journal of Portfolio Management. 1994. Vol. 21, No. 1. P. 49–58. URL: https://www.pm-research.com/content/iijpormgmt/21/1/49 (accessed: 20.05.2026).

24.Young T. W. Calmar Ratio: A Smoother Tool. Futures. 1991. Vol. 20, No. 1. P. 40.

25. Hilpisch Y. Python for Finance: Mastering Data-Driven Finance. 2nd ed. Sebastopol: O'Reilly Media, 2019. 714 p.

26. McKinney W. Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter. 3rd ed. Sebastopol: O'Reilly Media, 2022. 579 p. URL: https://wesmckinney.com/book/ (accessed: 21.05.2026).

27. Box G. E. P., Jenkins G. M., Reinsel G. C., Ljung G. M. Time Series Analysis: Forecasting and Control. 5th ed. Hoboken: John Wiley & Sons, 2015. 712 p. URL: https://www.wiley.com/en-br/Time%2BSeries%2BAnalysis%3A%2BForecasting%2Band%2BControl%2C%2B5th%2BEdition-p-9781118675021 (accessed: 21.05.2026).

28. Seabold S., Perktold J. Statsmodels: Econometric and Statistical Modeling with Python. Proceedings of the 9th Python in Science Conference. Austin, 2010. P. 92–96. URL: https://proceedings.scipy.org/articles/Majora-92bf1922-011 (accessed: 21.05.2026).

29. Методичні вказівки до виконання техніко-економічного обґрунтування дипломних проєктів для студентів інженерних спеціальностей / уклад.: В. В. Дергачова, Л. М. Кузьменко. Київ: КПІ ім. Ігоря Сікорського, 2021. 52 с.

30. ДСТУ ISO/IEC 25010:2016. Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Модель якості систем і програмних продуктів. [Чинний від 2016-12-01]. Київ: ДП «УкрНДНЦ», 2016. 28 с. URL: https://online.budstandart.com/ua/catalog/doc-page?id_doc=69134 (дата звернення: 21.05.2026).

31. Kendall M. G. Rank Correlation Methods. 4th ed. London: Charles Griffin & Company, 1975. 202 p. URL: https://search.worldcat.org/title/Rank-correlation-methods/oclc/3827024 (accessed: 21.05.2026).

32. Boehm B. W. Software Engineering Economics. Englewood Cliffs: Prentice Hall, 1981. 767 p. URL: https://archive.org/details/softwareengineer0000boeh (accessed: 21.05.2026).

33. Про збір та облік єдиного внеску на загальнообов’язкове державне соціальне страхування: Закон України від 08.07.2010 р. № 2464-VI. URL: https://zakon.rada.gov.ua/laws/show/2464-17 (дата звернення: 21.05.2026).

34. Зайченко Ю.П. Теорія прийняття рішень. Київ: Видавництво «КПІ», 2014. 412 с.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
        import warnings
warnings.filterwarnings("ignore")

import streamlit as st
import pandas as pd
import numpy as np
import yfinance as yf
import plotly.graph_objects as go
from datetime import datetime, timedelta
from statsmodels.tsa.arima.model import ARIMA
from scipy import stats

st.set_page_config(
   page_title="Інвестиційна модель MCDM",
    layout="wide",
    initial_sidebar_state="expanded",
)

TICKERS = {
    "Apple (AAPL)":              "AAPL",
    "Microsoft (MSFT)":          "MSFT",
    "Alphabet (GOOGL)":          "GOOGL",
    "Amazon (AMZN)":             "AMZN",
    "Tesla (TSLA)":             "TSLA",
    "NVIDIA (NVDA)":             "NVDA",
    "Meta (META)":              "META",
    "JPMorgan Chase (JPM)":      "JPM",
    "Johnson & Johnson (JNJ)":   "JNJ",
    "Procter & Gamble (PG)":     "PG",
    "Coca-Cola (KO)":           "KO",
    "Berkshire Hathaway (BRK-B)": "BRK-B",
    "Walmart (WMT)":             "WMT",
    "Visa (V)":                "V",
    "McDonald's (MCD)":          "MCD",
}

DEFAULT_TICKERS = [
    "Apple (AAPL)", "Microsoft (MSFT)", "Tesla (TSLA)",
    "JPMorgan Chase (JPM)", "Johnson & Johnson (JNJ)",
    "Coca-Cola (KO)", "NVIDIA (NVDA)",
```

```
]

HORIZONS = [
    "Короткостроковий (до 1 року)",
    "Середньостроковий (1-5 років)",
    "Довгостроковий (понад 5 років)",
]


@st.cache_data(ttl=3600, show_spinner=False)
def load_data(tickers, start, end):
    prices, volumes, failed = {}, {}, []
    for t in tickers:
        try:
            df = yf.download(t, start=start, end=end,
                             auto_adjust=True, progress=False)
            if len(df) > 50:
                prices[t]  = df["Close"].squeeze()
                volumes[t] = df["Volume"].squeeze()
            else:
                failed.append(t)
        except Exception:
            failed.append(t)
    if failed:
        st.warning(f"Не вдалося завантажити: {', '.join(failed)}")
    return pd.DataFrame(prices).dropna(), pd.DataFrame(volumes).dropna()


def log_returns(prices):
    return np.log(prices / prices.shift(1)).dropna()


def max_drawdown(ret):
    cum = (1 + ret).cumprod()
    dd  = (cum - cum.cummax()) / (cum.cummax() + 1e-10)
    return float(dd.min())


def compute_criteria(prices, volumes, horizon):
    ret = log_returns(prices)
    f1  = ret.mean() * 252
    f2  = ret.std()  * np.sqrt(252)
    f3  = volumes.mean() / 1_000_000

    rf_annual = 0.045
```

```
    sharpe = ((f1 - rf_annual) / (f2 + 1e-10)).clip(lower=0)
    mdd    = ret.apply(max_drawdown).abs()

    if horizon == HORIZONS[0]:
        # сирі значення — normalize() нормалізує f4 далі, без подвійної
нормалізації
        f4 = 0.6 * (f3 / (f3.max() + 1e-10)) + 0.4 * (1 / (f2 + 1e-10))
    elif horizon == HORIZONS[1]:
        f4 = sharpe
    else:
        f4 = (f1 / (mdd + 1e-10)).clip(lower=0)

    return pd.DataFrame({
        "f1_return":    f1,
        "f2_risk":      f2,
        "f3_liquidity": f3,
        "f4_horizon":   f4,
    })


def normalize(criteria):
    # мін-макс; f2 мінімізується, тому інвертується
    result = pd.DataFrame(index=criteria.index)
    for col in criteria.columns:
        mn, mx = criteria[col].min(), criteria[col].max()
        r = mx - mn
        if r < 1e-10:
            result[col] = 0.5
        elif col == "f2_risk":
            result[col] = (mx - criteria[col]) / r
        else:
            result[col] = (criteria[col] - mn) / r
    return result


def wsm(normalized, w1, w2, w3, w4):
    w = np.array([w1, w2, w3, w4], dtype=float)
    w /= w.sum()
    scores = (
        normalized["f1_return"]    * w[0] +
        normalized["f2_risk"]      * w[1] +
        normalized["f3_liquidity"] * w[2] +
        normalized["f4_horizon"]   * w[3]
    )
    return scores.sort_values(ascending=False)
```

```
def pareto_front(normalized):
    vals  = normalized.values
    n     = len(vals)
    is_nd = np.ones(n, dtype=bool)
    for i in range(n):
        if not is_nd[i]:
            continue
        for j in range(n):
            if i == j or not is_nd[j]:
                continue
            if np.all(vals[j] >= vals[i]) and np.any(vals[j] > vals[i]):
                is_nd[i] = False
                break
    return normalized.index[is_nd]


def forecast_arima(prices, forecast_days=30):
    """ARIMA(1,1,1) на лог-цінах з 80% CI."""
    forecasted_returns = {}
    forecast_curves    = {}
    for ticker in prices.columns:
        try:
            series = prices[ticker].dropna()
            if hasattr(series.index, "tz") and series.index.tz is not None:
                series.index = series.index.tz_localize(None)
            last_date    = pd.Timestamp(series.index[-1])
            future_dates = pd.bdate_range(start=last_date + pd.Timedelta(days=1),
periods=forecast_days)
            log_series = np.log(series.values)
            # Замість trend='c' обчислюємо drift вручну — середній денний
            # приріст лог-ціни за всю історію активу — і додаємо його
            # накопиченим чином до прогнозу. Математично еквівалентно
            # ARIMA з константою, але без проблем збіжності оптимізатора.
            drift     = float(np.mean(np.diff(log_series)))
            model     = ARIMA(log_series, order=(1, 1, 1))
            fitted    = model.fit()
            result    = fitted.get_forecast(steps=forecast_days)
            mean_log  = np.array(result.predicted_mean)
            conf_log  = np.array(result.conf_int(alpha=0.20))   # 80% CI
            # додаємо накопичений drift: крок i отримує +drift*i
            drift_vec  = drift * np.arange(1, forecast_days + 1)
            mean_log   = mean_log  + drift_vec
            conf_log[:, 0] = conf_log[:, 0] + drift_vec
```

```
            conf_log[:, 1] = conf_log[:, 1] + drift_vec
            mean  = pd.Series(np.exp(mean_log),        index=future_dates)
            lower = pd.Series(np.exp(conf_log[:, 0]),  index=future_dates)
            upper = pd.Series(np.exp(conf_log[:, 1]),  index=future_dates)
            forecast_curves[ticker] = {"mean": mean, "lower": lower, "upper": upper}
            current_price    = float(series.iloc[-1])
            forecasted_price = float(mean.iloc[-1])
            raw_return       = (forecasted_price - current_price) / current_price
            annual_return    = raw_return * (252 / forecast_days)
            forecasted_returns[ticker] = annual_return
        except Exception:
            ret = log_returns(prices[[ticker]])
            forecasted_returns[ticker] = float(ret.mean().iloc[0] * 252)
            forecast_curves[ticker] = None
    return pd.Series(forecasted_returns), forecast_curves



def run_backtest(prices, top_tickers, test_start, test_end):
    test = prices.loc[test_start:test_end]
    if len(test) < 10:
        return None

    norm_p    = test / test.iloc[0]
    wsm_tick  = [t for t in top_tickers if t in prices.columns][:3]
    portfolio = norm_p[wsm_tick].mean(axis=1)
    bench     = norm_p.mean(axis=1)

    r_p = portfolio.pct_change().dropna()
    r_b = bench.pct_change().dropna()
    rf  = 0.045 / 252

    def sharpe(r):
        return float((r - rf).mean() / r.std() * np.sqrt(252)) if r.std() > 1e-10 else 0.0

    def mdd_pct(curve):
        dd = (curve - curve.cummax()) / (curve.cummax() + 1e-10)
        return float(dd.min() * 100)

    return {
        "wsm_curve":    portfolio,
        "bench_curve":  bench,
        "wsm_return":   (portfolio.iloc[-1] - 1) * 100,
        "bench_return": (bench.iloc[-1] - 1) * 100,
        "wsm_sharpe":   round(sharpe(r_p), 3),
```

```
        "bench_sharpe": round(sharpe(r_b), 3),
        "wsm_mdd":      round(mdd_pct(portfolio), 2),
        "bench_mdd":    round(mdd_pct(bench), 2),
        "wsm_vol":      round(float(r_p.std() * np.sqrt(252) * 100), 2),
        "bench_vol":    round(float(r_b.std() * np.sqrt(252) * 100), 2),
        "wsm_tickers":  wsm_tick,
        "r_p":          r_p,   # денні дохідності WSM-портфеля (для статтестів)
        "r_b":          r_b,   # денні дохідності бенчмарку
    }


def main():
    st.title("Модель підтримки інвестиційних рішень")
    st.caption("Багатокритеріальна оптимізація: WSM + Парето-
оптимальність")
    st.divider()

    with st.sidebar:
        st.header("Налаштування")

        st.subheader("1. Активи")
        selected = st.multiselect(
            "Оберіть акції:",
            options=list(TICKERS.keys()),
            default=DEFAULT_TICKERS,
        )
        ticker_list = [TICKERS[n] for n in selected]

        if len(ticker_list) < 3:
            st.error("Оберіть щонайменше 3 активи.")
            return

        st.subheader("2. Часовий діапазон")
        train_years = st.slider("Навчальний підперіод (роки):", 1, 5, 2)
        test_years  = st.slider("Тестовий підперіод (роки):",  1, 2, 1)

        today       = datetime.today()
        test_start  = today - timedelta(days=test_years  * 365)
        train_start = today - timedelta(days=(train_years + test_years) * 365)

        st.caption(f"Навчання: {train_start.strftime('%Y-%m-%d')} —
{test_start.strftime('%Y-%m-%d')}")
        st.caption(f"Тест:     {test_start.strftime('%Y-%m-%d')} —
{today.strftime('%Y-%m-%d')}")
```

```
        st.subheader("3. Горизонт інвестування")
        horizon = st.radio("", options=HORIZONS, index=1)

        st.subheader("4. Вагові коефіцієнти")
        w1 = st.slider("Дохідність",  0, 10, 3)
        w2 = st.slider("Ризик",       0, 10, 3)
        w3 = st.slider("Ліквідність", 0, 10, 2)
        w4 = st.slider("Горизонт",    0, 10, 2)

        total_w = w1 + w2 + w3 + w4
        if total_w == 0:
            st.error("Хоча б одна вага має бути > 0.")
            return

        st.caption(
            f"Нормалізовані ваги: "
            f"{w1/total_w:.2f} / {w2/total_w:.2f} / "
            f"{w3/total_w:.2f} / {w4/total_w:.2f}"
        )

        st.subheader("5. Прогнозування ARIMA")
        use_forecast  = st.checkbox("Використовувати прогнозну дохідність",
value=False)
        forecast_days = st.slider("Горизонт прогнозу (днів):", 10, 90, 30,
                                  disabled=not use_forecast)

        run = st.button("Запустити аналіз", use_container_width=True,
type="primary")

    if not run and "res" not in st.session_state:
        st.info("Налаштуйте параметри та натисніть «Запустити аналіз».")
        return

    if run:
        with st.spinner("Завантаження даних..."):
            prices, volumes = load_data(
                ticker_list,
                train_start.strftime("%Y-%m-%d"),
                today.strftime("%Y-%m-%d"),
            )

        if prices.empty or len(prices.columns) < 2:
            st.error("Не вдалося завантажити достатньо даних.")
            return
```

```
        train_p = prices.loc[:test_start.strftime("%Y-%m-%d")]
        train_v = volumes.loc[:test_start.strftime("%Y-%m-%d")]

        crit   = compute_criteria(train_p, train_v, horizon)
        norm   = normalize(crit)
        scores = wsm(norm, w1, w2, w3, w4)
        pareto = pareto_front(norm)
        bt     = run_backtest(
            prices,
            scores.index.tolist(),
            test_start.strftime("%Y-%m-%d"),
            today.strftime("%Y-%m-%d"),
        )

        # ARIMA прогнозування (якщо увімкнено) — використовуємо повні дані,
        # щоб прогноз стартував від найсвіжішої точки, а не кінця навчального
підперіоду
        f_returns, f_curves, scores_f = None, None, None
        if use_forecast:
            with st.spinner("Прогнозування ARIMA..."):
                f_returns, f_curves = forecast_arima(prices, forecast_days)
            crit_f            = crit.copy()
            crit_f["fl_return"] = f_returns
            norm_f            = normalize(crit_f)
            scores_f          = wsm(norm_f, w1, w2, w3, w4)

        st.session_state["res"] = {
            "prices":       prices,
            "crit":         crit,
            "norm":         norm,
            "scores":       scores,
            "pareto":       pareto,
            "bt":           bt,
            "weights":      (w1, w2, w3, w4, total_w),
            "f_returns":    f_returns,
            "f_curves":     f_curves,
            "scores_f":     scores_f,
            "forecast_days": forecast_days if use_forecast else None,
        }

    res = st.session_state["res"]

    tab1, tab2, tab3, tab4, tab5 = st.tabs([
        "Дані та критерії",
        "Метод зваженої суми",
```

```
    "Парето-фронт",
    "Бектестинг",
    "Прогнозування ARIMA",
])

with tab1:
    st.subheader("Розраховані значення критеріїв")
    disp = res["crit"].copy()
    disp.columns = ["Дохідність (річна)", "Ризик (волатильність)",
                    "Ліквідність (млн $)", "Оцінка горизонту"]
    st.dataframe(disp.style.format("{:.4f}"), use_container_width=True)

    with st.expander("Нормалізована матриця критеріїв"):
        nd = res["norm"].copy()
        nd.columns = ["f1 Дохідність", "f2 Ризик",
                      "f3 Ліквідність", "f4 Горизонт"]
        st.dataframe(nd.style.format("{:.3f}"), use_container_width=True)
        st.caption("Всі значення в діапазоні [0, 1]. Більше = краще.")

    st.subheader("Динаміка цін (нормалізовані, база = 100)")
    norm_p = res["prices"] / res["prices"].iloc[0] * 100
    fig = go.Figure()
    for col in norm_p.columns:
        fig.add_trace(go.Scatter(x=norm_p.index, y=norm_p[col],
                                 name=col, mode="lines"))
    fig.update_layout(xaxis_title="Дата", yaxis_title="Індекс",
                      height=400, hovermode="x unified")
    st.plotly_chart(fig, use_container_width=True)

with tab2:
    st.subheader("Результати методу зваженої суми")
    w1, w2, w3, w4, tw = res["weights"]
    st.info(
        f"Ваги: Дохідність = {w1/tw:.2f}  |  Ризик = {w2/tw:.2f}  |  "
        f"Ліквідність = {w3/tw:.2f}  |  Горизонт = {w4/tw:.2f}"
    )

    df_wsm = res["scores"].reset_index()
    df_wsm.columns = ["Актив", "WSM-оцінка"]
    df_wsm["Ранг"] = range(1, len(df_wsm) + 1)
    df_wsm["WSM-оцінка"] = df_wsm["WSM-оцінка"].round(4)
    df_wsm["Статус"] = ["Рекомендується" if i < 3 else "–"
                        for i in range(len(df_wsm))]
    st.dataframe(df_wsm[["Ранг", "Актив", "WSM-оцінка", "Статус"]],
                 use_container_width=True, hide_index=True)
```

```
    colors = ["#1a1a2e" if i < 3 else "#b0b0b0"
              for i in range(len(res["scores"]))]
    fig = go.Figure(go.Bar(
        x=res["scores"].index,
        y=res["scores"].values,
        marker_color=colors,
        text=[f"{v:.3f}" for v in res["scores"].values],
        textposition="outside",
    ))
    fig.update_layout(title="WSM-оцінки активів",
                      xaxis_title="Актив", yaxis_title="WSM-оцінка",
                      height=420, showlegend=False)
    st.plotly_chart(fig, use_container_width=True)

with tab3:
    st.subheader("Парето-оптимальні рішення")
    pareto = res["pareto"]
    norm   = res["norm"]
    is_p   = norm.index.isin(pareto)

    st.success(
        f"Знайдено {len(pareto)} Парето-оптимальних рішень: "
        f"{', '.join(pareto.tolist())}"
    )

    def pareto_scatter(x_col, y_col, x_label, y_label):
        fig = go.Figure()
        fig.add_trace(go.Scatter(
            x=norm.loc[~is_p, x_col], y=norm.loc[~is_p, y_col],
            mode="markers+text",
            marker=dict(size=10, color="lightgray",
                        line=dict(color="gray", width=1)),
            text=norm.index[~is_p], textposition="top center",
            name="Доміновані", textfont=dict(size=10),
        ))
        fig.add_trace(go.Scatter(
            x=norm.loc[is_p, x_col], y=norm.loc[is_p, y_col],
            mode="markers+text",
            marker=dict(size=14, color="black", symbol="diamond"),
            text=norm.index[is_p], textposition="top center",
            name="Парето-оптимальні", textfont=dict(size=10),
        ))
        fig.update_layout(xaxis_title=x_label, yaxis_title=y_label,
                          height=380)
```

```
        return fig

    col1, col2 = st.columns(2)
    with col1:
        st.caption("Ризик – Дохідність")
        st.plotly_chart(
            pareto_scatter("f2_risk", "f1_return",
                           "f2 Ризик (норм.)", "f1 Дохідність (норм.)"),
            use_container_width=True,
        )
    with col2:
        st.caption("Ліквідність – Дохідність")
        st.plotly_chart(
            pareto_scatter("f3_liquidity", "f1_return",
                           "f3 Ліквідність (норм.)", "f1 Дохідність (норм.)"),
            use_container_width=True,
        )

with tab4:
    st.subheader("Верифікація: бектестинг")
    bt = res["bt"]
    if bt is None:
        st.warning("Недостатньо даних для бектестингу.")
        return

    st.caption(
        f"WSM-портфель: {', '.join(bt['wsm_tickers'])}  |  "
        f"Бенчмарк: рівновагові всі обрані активи"
    )

    c1, c2, c3, c4 = st.columns(4)
    with c1:
        st.metric("Дохідність (WSM)", f"{bt['wsm_return']:.1f}%",
                  f"{bt['wsm_return'] - bt['bench_return']:+.1f}% vs бенчмарк")
    with c2:
        st.metric("Коефіцієнт Шарпа", str(bt["wsm_sharpe"]),
                  f"{bt['wsm_sharpe'] - bt['bench_sharpe']:+.3f} vs бенчмарк")
    with c3:
        st.metric("Макс. просадка", f"{bt['wsm_mdd']:.1f}%",
                  f"{bt['wsm_mdd'] - bt['bench_mdd']:+.1f}% vs бенчмарк",
                  delta_color="inverse")
    with c4:
        st.metric("Волатильність", f"{bt['wsm_vol']:.1f}%",
                  f"{bt['wsm_vol'] - bt['bench_vol']:+.1f}% vs бенчмарк",
                  delta_color="inverse")
```

```
fig = go.Figure()
fig.add_trace(go.Scatter(
    x=bt["wsm_curve"].index,
    y=(bt["wsm_curve"] - 1) * 100,
    name="WSM-портфель (топ-3)",
    line=dict(color="black", width=2.5),
))
fig.add_trace(go.Scatter(
    x=bt["bench_curve"].index,
    y=(bt["bench_curve"] - 1) * 100,
    name="Бенчмарк",
    line=dict(color="gray", width=1.8, dash="dash"),
))
fig.add_hline(y=0, line_dash="dot", line_color="lightgray")
fig.update_layout(
    title="Накопичена дохідність: WSM-портфель vs Бенчмарк",
    xaxis_title="Дата", yaxis_title="Дохідність (%)",
    height=430, hovermode="x unified",
)
st.plotly_chart(fig, use_container_width=True)

with st.expander("Детальне порівняння метрик"):
    metrics = pd.DataFrame({
        "Метрика": ["Дохідність (%)", "Коефіцієнт Шарпа",
                    "Макс. просадка (%)", "Волатильність (%)"],
        "WSM-портфель": [f"{bt['wsm_return']:.2f}", bt["wsm_sharpe"],
                         bt["wsm_mdd"], bt["wsm_vol"]],
        "Бенчмарк":     [f"{bt['bench_return']:.2f}", bt["bench_sharpe"],
                         bt["bench_mdd"], bt["bench_vol"]],
    })
    st.dataframe(metrics, use_container_width=True, hide_index=True)

# ── Статистичні тести
```

```
st.divider()
st.subheader("Статистична значущість різниці дохідностей")

r_p = bt["r_p"]
r_b = bt["r_b"]
diff = (r_p - r_b).dropna()
n    = len(diff)

# 1. Критерій Вілкоксона (непараметричний, основний)
#    H₀: медіана(r_WSM – r_bench) = 0
```

```
        try:
            w_stat, w_pval = stats.wilcoxon(diff)
        except Exception:
            w_stat, w_pval = float("nan"), float("nan")

        # 2. Парний t-тест (параметричний, для порівняння)
        #    H₀: E[r_WSM – r_bench] = 0
        t_stat, t_pval = stats.ttest_rel(r_p, r_b)

        alpha = 0.05

        col_w, col_t, col_n = st.columns(3)

        with col_w:
            st.metric(
                "Критерій Вілкоксона — p-value",
                f"{w_pval:.4f}",
                delta="Значуща різниця ✓" if w_pval < alpha else "Незначуща
різниця",
                delta_color="normal" if w_pval < alpha else "off",
            )
            st.caption("Непараметричний; не вимагає нормальності дохідностей")

        with col_t:
            st.metric(
                "Парний t-тест — p-value",
                f"{t_pval:.4f}",
                delta="Значуща різниця ✓" if t_pval < alpha else "Незначуща
різниця",
                delta_color="normal" if t_pval < alpha else "off",
            )
            st.caption("Параметричний; рекомендується як додатковий")

        with col_n:
            st.metric("Кількість спостережень", f"{n} днів")
            st.caption(f"Середня різниця: {diff.mean()*100:+.4f}%/день")

        # Гіпотези
        with st.expander("Формулювання гіпотез та інтерпретація"):
            st.markdown(f"""
**Нульова гіпотеза H₀:** різниця між денними дохідностями WSM-портфеля
та бенчмарку дорівнює нулю — модель не дає переваги над пасивною
стратегією.
```

```
**Альтернативна гіпотеза H₁:** різниця відмінна від нуля — модель має статистично значущу перевагу або поступку.

**Рівень значущості:** α = {alpha} (5%).

**Результат Вілкоксона:** p = {w_pval:.4f} → {'відхиляємо H₀: різниця **статистично значуща**.' if w_pval < alpha else 'не відхиляємо H₀: різниця **статистично незначуща** на рівні 5%.'}

**Чому Вілкоксона?** Розподіл дохідностей акцій має «важкі хвости» (fat tails) і не є нормальним, тому непараметричний критерій є надійнішим за t-тест. Т-тест наведено для порівняння.
""")

        # Гістограма різниць
        fig_diff = go.Figure()
        fig_diff.add_trace(go.Histogram(
            x=diff * 100,
            nbinsx=50,
            name="Різниця дохідностей",
            marker_color="rgba(30,30,60,0.7)",
        ))
        fig_diff.add_vline(x=0, line_dash="dash", line_color="gray",
                           annotation_text="0", annotation_position="top right")
        fig_diff.add_vline(x=float(diff.mean() * 100), line_dash="dot",
                           line_color="black",
                           annotation_text=f"Середнє {diff.mean()*100:+.4f}%",
                           annotation_position="top left")
        fig_diff.update_layout(
            title="Розподіл денних різниць (r_WSM – r_bench), %",
            xaxis_title="Різниця дохідностей (%)",
            yaxis_title="Кількість днів",
            height=320,
            showlegend=False,
        )
        st.plotly_chart(fig_diff, use_container_width=True)

    with tab5:
        st.subheader("Прогнозування цін: ARIMA(1,1,1)")

        if res["f_curves"] is None:
            st.info("Увімкніть «Використовувати прогнозну дохідність» у боковій панелі та запустіть аналіз.")
        else:
            fd   = res["forecast_days"]
```

```
        fret = res["f_returns"]
        hist = res["prices"]

        st.caption(f"Горизонт прогнозу: {fd} днів. Сіра зона — 80% довірчий
інтервал.")

        # Порівняльна таблиця: історична vs прогнозна дохідність
        cmp = pd.DataFrame({
            "Актив":                fret.index,
            "Іст. дохідність (річна)":
res["crit"]["f1_return"].reindex(fret.index).round(4),
            "Прогн. дохідність (річна)": fret.round(4),
        }).set_index("Актив")
        cmp["Різниця"] = (cmp["Прогн. дохідність (річна)"] -
                          cmp["Іст. дохідність (річна)"]).round(4)
        st.dataframe(cmp.style.format("{:.4f}"), use_container_width=True)

        st.divider()

        # Графіки прогнозу для кожного тікера
        cols = st.columns(2)
        for idx, ticker in enumerate(fret.index):
            curve = res["f_curves"].get(ticker)
            with cols[idx % 2]:
                st.markdown(f"**{ticker}**")
                if curve is None:
                    st.caption("ARIMA не збігся — використано середнє.")
                    continue

                # Останні 90 днів історії (прибираємо tz для сумісності з
прогнозом)
                hist_slice = hist[ticker].dropna().tail(90)
                if hasattr(hist_slice.index, "tz") and hist_slice.index.tz is not None:
                    hist_slice.index = hist_slice.index.tz_localize(None)

                # З'єднувальна точка: остання фактична + перша прогнозна
                connect_x = [hist_slice.index[-1], curve["mean"].index[0]]
                connect_y = [hist_slice.iloc[-1],  curve["mean"].iloc[0]]

                fig = go.Figure()

                fig.add_trace(go.Scatter(
                    x=hist_slice.index,
                    y=hist_slice.values,
                    name="Фактичні",
```

```
            line=dict(color="black", width=2),
        ))

        # З'єднувальна лінія між фактичними і прогнозом
        fig.add_trace(go.Scatter(
            x=connect_x, y=connect_y,
            line=dict(color="black", width=2, dash="dash"),
            showlegend=False,
        ))

        # Довірчий інтервал (заливка)
        fig.add_trace(go.Scatter(
            x=list(curve["upper"].index) + list(curve["lower"].index[::-1]),
            y=list(curve["upper"].values) + list(curve["lower"].values[::-1]),
            fill="toself",
            fillcolor="rgba(150,150,150,0.2)",
            line=dict(width=0),
            name="80% інтервал",
            showlegend=True,
        ))

        # Прогнозна крива
        fig.add_trace(go.Scatter(
            x=curve["mean"].index,
            y=curve["mean"].values,
            name="Прогноз",
            line=dict(color="black", width=2, dash="dash"),
        ))

        fig.update_layout(
            height=280,
            margin=dict(t=20, b=40, l=40, r=10),
            xaxis_title="Дата",
            yaxis_title="Ціна ($)",
            hovermode="x unified",
            showlegend=True,
            legend=dict(x=0, y=1, font=dict(size=9)),
        )
        st.plotly_chart(fig, use_container_width=True)

# Оновлений WSM з прогнозними дохідностями
st.divider()
st.subheader("Оновлений WSM з прогнозною дохідністю")
sf = res["scores_f"].reset_index()
sf.columns = ["Актив", "WSM (прогноз)"]
```

```
        sf["Ранг"] = range(1, len(sf) + 1)
        sf["WSM (прогноз)"] = sf["WSM (прогноз)"].round(4)

        # Порівняння рангів
        orig_ranks = {t: i+1 for i, t in enumerate(res["scores"].index)}
        sf["Ранг (іст.)"] = sf["Актив"].map(orig_ranks)
        sf["Зміна рангу"] = sf["Ранг (іст.)"] - sf["Ранг"]
        sf["Зміна рангу"] = sf["Зміна рангу"].apply(
            lambda x: f"+{x}" if x > 0 else str(x) if x < 0 else "–"
        )
        sf["Статус"] = ["Рекомендується" if i < 3 else "–"
                        for i in range(len(sf))]
        st.dataframe(
            sf[["Ранг", "Актив", "WSM (прогноз)", "Ранг (іст.)", "Зміна рангу",
"Статус"]],
            use_container_width=True, hide_index=True,
        )
        st.caption(
            "Ранг (іст.) — ранг за WSM з історичною дохідністю. "
            "Зміна рангу — наскільки позиція змінилась після врахування
прогнозу."
        )


if __name__ == "__main__":
    main()
```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ