

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Розробка інтелектуальної системи управління
інвестиційним портфелем з використанням
машинного навчання»**

Виконав студент групи ІПЗм-24-1
КУЧЕР Віктор Олександрович

Керівник роботи:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
ЧИЖМОТРЯ Олексій Володимирович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«18» вересня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КУЧЕР Віктор Олександрович**

Керівник роботи: **ЄФРЕМОВ Юрій Миколайович**

Тема роботи: **«Розробка інтелектуальної системи управління інвестиційним портфелем з використанням машинного навчання»**,

затверджена Наказом закладу вищої освіти від «18» вересня 2025 р. № 491/с

Вихідні дані для роботи: **аналіз та прогнозування показників фінансових часових рядів для інвестиційних інструментів; дослідження методів машинного навчання та глибинних нейронних мереж у задачах прогнозування та оцінки ризику інвестиційного портфеля; способи автоматизації процесів формування портфеля у веб-орієнтованій інтелектуальній системі.**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю. М.	22.09.2025р.	22.09.2025р.
2	Єфремов Ю. М.	22.09.2025р.	22.09.2025р.
3	Єфремов Ю. М.	22.09.2025р.	22.09.2025р.

РЕФЕРАТ

Кваліфікаційна робота освітнього ступеня «магістр» присвячена розробці інтелектуальної системи управління інвестиційним портфелем на основі методів машинного навчання. Пояснювальна записка складається зі вступу, трьох розділів та списку використаних джерел. Пояснювальна записка до кваліфікаційної магістерської роботи містить 64 сторінки, 25 ілюстрації, 1 таблицю, 4 додатки та 40 посилань на зовнішні джерела.

Метою роботи є створення інтелектуальної програмної системи, яка автоматизує збір і аналіз ринкових даних, формування прогнозів доходності та волатильності фінансових інструментів, оцінку ризику й оптимізацію структури інвестиційного портфеля, а також забезпечує зручний веб-інтерфейс для користувача.

У роботі описано процес проєктування загальної програмно-технічної архітектури системи, розробку модулів збору та зберігання ринкових даних, формування датасетів часових рядів без витоку інформації, а також побудову веб-інтерфейсу для моніторингу портфеля та керування сценаріями.

КЛЮЧОВІ СЛОВА: ІНТЕЛЕКТУАЛЬНА СИСТЕМА, ІНВЕСТИЦІЙНИЙ ПОРТФЕЛЬ, ЧАСОВІ РЯДИ, LSTM, GRU, TRANSFORMER, ДАТАСЕТ, АНАЛІЗ ДАНИХ.

ABSTRACT

The master's thesis is devoted to the development of an intelligent investment portfolio management system based on machine learning methods. The explanatory report consists of an introduction, three chapters and a list of references. The explanatory report of the master's qualification thesis contains 64 pages, 25 figures, 1 table, 4 appendices, and 40 references to external sources.

The aim of the thesis is to develop an intelligent software system that automates the collection and analysis of market data, the generation of forecasts for returns and volatility of financial instruments, the assessment of risk and the optimization of the investment portfolio structure, as well as provides a user-friendly web interface.

The thesis describes the design process of the overall software and hardware architecture of the system, the development of modules for collecting and storing market data, the formation of time series datasets without information leakage, and the implementation of a web interface for portfolio monitoring and scenario management.

KEY WORDS: INTELLIGENT SYSTEM, INVESTMENT PORTFOLIO, TIME SERIES, LSTM, GRU, TRANSFORMER, DATASET, DATA ANALYSIS.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ	10
1.1. Особливості функціонування інвестиційних портфелів та сучасні виклики управління активами	10
1.2. Огляд методів прогнозування фінансових часових рядів	12
1.3. Аналіз існуючих рішень у сфері управління інвестиційним портфелем	14
1.4. Постановка задачі дослідження та вибір засобів реалізації	17
Висновки до розділу 1	20
РОЗДІЛ 2. РОЗРОБКА ТА ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ.....	22
2.1. Архітектура інтелектуальної системи управління інвестиційним портфелем	22
2.1.1. Загальна програмно-технічна архітектура системи	23
2.1.2. Модулі збору та зберігання ринкових даних	24
2.2. Вибір та проєктування моделей машинного навчання для прогнозування й оцінки ризику	26
2.3. Проєктування веб-додатку інтелектуального управління інвестиційним портфелем	35
Висновки до розділу 2	41
РОЗДІЛ 3. ОПИС ПРОГРАМНОГО ПРОДУКТУ	43
3.1. Графічний інтерфейс	43
3.2. Налаштування параметрів аналізу	47
3.3. Відображення результатів аналізу	49
Висновки до розділу 3	56
ВИСНОВКИ	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59
ДОДАТКИ	64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI - Artificial Intelligence, штучний інтелект.

API - Application Programming Interface, програмний інтерфейс застосунків.

CNN - Convolutional Neural Network, згорткова нейронна мережа.

CVaR - Conditional Value at Risk, умовна величина під ризиком.

DB - Database, база даних.

GRU - Gated Recurrent Unit, рекурентна нейронна мережа з вентиляльованими блоками.

HTTP/HTTPS - HyperText Transfer Protocol (Secure), протокол передачі гіпертексту (захищений).

JSON - JavaScript Object Notation, текстовий формат обміну даними.

LSTM - Long Short-Term Memory, нейронна мережа з довгою короткочасною пам'яттю.

ML - Machine Learning, машинне навчання.

PnL - Profit and Loss, прибуток і збиток.

RAM - Random Access Memory, оперативна пам'ять.

RNN - Recurrent Neural Network, рекурентна нейронна мережа.

SQL - Structured Query Language, мова структурованих запитів.

TCN - Temporal Convolutional Network, часова згорткова мережа.

VaR - Value at Risk, величина під ризиком.

ВСТУП

Актуальність теми. Проблема побудови надійних інтелектуальних систем управління інвестиційним портфелем є однією з ключових у сучасних фінансових технологіях. Зростання обсягів ринкових даних, поява нових фінансових інструментів, висока волатильність та зміна ринкових режимів вимагають переходу від статичних моделей до адаптивних рішень, здатних у напівреальному часі аналізувати часові ряди, оцінювати ризик і формувати керуючі дії щодо структури портфеля. Традиційні підходи, що спираються на прості статистичні моделі або ручний аналіз окремих показників, стають недостатніми в умовах складних, багатофакторних ринків і призводять до зростання операційних ризиків та втрати ефективності.

Типовим прикладом є управління портфелями акцій і біржових фондів в умовах швидких змін макроекономічних показників, новинного фону та ринкових настроїв. Аналітик або портфельний менеджер вимушений моніторити великий обсяг показників: ціни, обсяги торгів, волатильність, кореляції між активами, факторні індикатори, що ускладнює своєчасне виявлення змін режиму ринку та адаптацію стратегії. Додатково ускладнює задачу те, що різні інструменти мають різні масштаби, розподіли, ліквідність і чутливість до новин, а класичні лінійні моделі погано відображають нелінійні залежності й ефекти пам'яті в часових рядах.

Перспективним варіантом є створення веб-орієнтованої інтелектуальної системи управління інвестиційним портфелем, яка поєднує модулі збору та зберігання ринкових даних, підсистему машинного навчання для прогнозування доходностей і волатильності, модулі оцінки ризику та оптимізації структури портфеля, а також зручний графічний інтерфейс для користувача. Завдяки використанню глибинних моделей (LSTM, GRU, Transformer) система може адаптуватися до нелінійних залежностей, змін режимів ринку та різноманіття часових масштабів, на яких приймаються інвестиційні рішення.

Мета роботи - розробка інтелектуальної системи, за допомогою якої з'являється можливість підвищити ефективність та зручність управління інвестиційним портфелем за рахунок автоматизованого прогнозування ринкових показників, оцінки ризику й оптимізації структури активів у напівреальному часі.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Дослідити сучасні підходи до управління інвестиційними портфелями та оцінки фінансових ризиків.
2. Дослідити методи машинного навчання для прогнозування фінансових часових рядів і виявлення ринкових режимів (LSTM, GRU, Transformer, класичні ML-моделі).
3. Розробити підхід до формування датасетів і підготовки часових рядів без витоку інформації, включаючи схеми валідації (walk-forward, блокова крос-валідація).
4. Спроекувати архітектуру інтелектуальної системи управління інвестиційним портфелем, включаючи модулі збору даних, прогнозування, оцінки ризику та оптимізації.
5. Реалізувати веб-додаток, який забезпечує взаємодію користувача з системою, візуалізацію ринкових даних, прогнозів і портфельних метрик, а також керування сценаріями оптимізації.

Об'єктом дослідження є процес управління інвестиційними портфелями, що включає аналіз ринкових даних, оцінку ризику та прийняття інвестиційних рішень.

Предметом дослідження є методи та алгоритми машинного навчання для прогнозування фінансових часових рядів, оцінки ризику та оптимізації структури інвестиційного портфеля, а також архітектура інтегрованої веб-орієнтованої інтелектуальної системи, що реалізує ці підходи.

Наукова новизна роботи полягає в поєднанні глибинних моделей часових рядів із класичними алгоритмами машинного навчання та

спеціалізованими ознаками ринкових режимів у єдиній інтелектуальній системі управління інвестиційним портфелем. Особливістю підходу є побудова стандартизованого пайплайна формування датасетів без витоку інформації, використання ансамблю моделей для прогнозування доходностей і волатильності, а також інтеграція отриманих прогнозів у модуль оптимізації портфеля з урахуванням ризикових обмежень. Це розширює можливості програмного комплексу щодо адаптації до різних класів активів, ринкових сценаріїв і горизонтів інвестування.

Практичне значення роботи полягає в розробці прототипу веб-орієнтованої інтелектуальної системи, яка може бути використана як інструмент підтримки прийняття рішень для портфельних менеджерів, фінансових аналітиків та приватних інвесторів. Система дозволяє автоматизувати обробку ринкових даних, отримувати прогнозні оцінки ключових показників, контролювати ризик та формувати рекомендації щодо структури портфеля у зручному для користувача інтерфейсі.

Апробація. За темою кваліфікаційної магістерської роботи було опубліковано тези конференції [1].

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Особливості функціонування інвестиційних портфельів та сучасні виклики управління активами

Особливості функціонування інвестиційних портфельів у сучасних умовах визначаються поєднанням класичних принципів диверсифікації й оптимізації співвідношення «ризик-дохідність» із впливом регуляторних, технологічних та структурних змін на фінансових ринках. Портфель розглядається як цілісна система активів, де ключовим є не стільки вибір окремих інструментів, скільки побудова збалансованої структури, здатної забезпечити реалізацію інвестиційних цілей за прийнятного рівня ризику.

Інвестиційний портфель - це впорядкована сукупність фінансових активів (акцій, облігацій, інструментів грошового ринку, фондів, альтернативних активів), сформована відповідно до цілей інвестора, його ризик-апетиту, інвестиційного горизонту та обмежень ліквідності [2]. Основна мета портфельного підходу полягає у досягненні бажаного рівня очікуваної дохідності при мінімізації коливань вартості капіталу, що реалізується через розподіл вкладень між різними класами активів, секторами та регіонами [3].

Ключовим інструментом формалізації цілей інвестора є інвестиційна політика (Investment Policy Statement), у якій фіксуються цільові орієнтири щодо дохідності, допустимий рівень ризику, часові рамки, податкові й регуляторні обмеження, а також індивідуальні вимоги. На основі цього документа визначаються стратегічна структура портфеля (стратегічна алокація активів) та правила допустимих відхилень, які використовуються в процесі подальшого моніторингу та ребалансування [4].

Сучасна портфельна теорія доводить, що поєднання активів із різними характеристиками дохідності та кореляції дозволяє знизити загальний ризик портфеля без пропорційного зменшення очікуваної дохідності. Ефект диверсифікації досягається завдяки тому, що негативна динаміка одних

інструментів частково компенсується позитивними результатами інших, що згладжує волатильність сукупної вартості портфеля [5].

Для оцінки ризику застосовуються статистичні міри варіації, коваріації та кореляції доходностей, а також показники ймовірних втрат (Value-at-Risk, умовний VaR тощо), які враховують поведінку портфеля в стресових сценаріях [6]. У практиці управління активами поширені підходи, орієнтовані не лише на максимізацію доходності, а й на контроль структури ризику - наприклад, моделі розподілу ризику (risk parity) та побудова портфелів з максимальною диверсифікацією внесків окремих активів у загальну волатильність [7].

Функціонування портфеля ґрунтується на циклічному процесі, що включає етапи планування, виконання та контролю. На етапі планування визначаються профіль інвестора, інвестиційні цілі та обмеження; далі формується цільова алокація між класами активів (акції, облігації, грошовий ринок, альтернативні інвестиції), а також політика використання активних і пасивних стратегій [8].

Етап виконання передбачає відбір конкретних інструментів, реалізацію угод та побудову портфеля відповідно до обраної стратегії, після чого відбувається постійний моніторинг ризику, доходності та дотримання встановлених лімітів. За результатами контролю здійснюється ребалансування - коригування структури портфеля для повернення до цільових ваг або адаптації до змін ринкових умов і параметрів інвестора.

Сучасне функціонування інвестиційних портфелів відбувається в умовах підвищеної ринкової волатильності, геополітичної напруженості та частіших шоків ліквідності, що ускладнює підтримання стабільного профілю «ризик-дохідність». Зростання кореляцій між класами активів у кризові періоди знижує ефективність традиційної диверсифікації, вимагаючи від керуючих активами використання більш гнучких підходів до хеджування та управління ризиком.

Важливим трендом є посилення регуляторних вимог, зокрема у сфері сталого інвестування (ESG), прозорості та розкриття інформації, що змінює підходи до побудови портфелів та систем звітності. Паралельно відбувається цифрова трансформація галузі: впровадження хмарних платформ, аналітики великих даних та алгоритмічних рішень створює нові можливості для автоматизованого моніторингу портфельних ризиків, але водночас підвищує значущість операційних і кіберризиків [9].

У підсумку інвестиційний портфель у сучасній економіці функціонує як динамічна система, що потребує безперервного аналізу, адаптації та поєднання фінансової інженерії з технологічними інструментами управління. Це формує передумови для подальшого використання методів статистичного аналізу, машинного навчання та автоматизованих платформ, які розглядатимуться в наступних підрозділах дослідження.

1.2. Огляд методів прогнозування фінансових часових рядів

Методи прогнозування фінансових часових рядів та оптимізації портфеля за останні роки еволюціонували від класичних статистичних моделей до складних алгоритмів машинного та глибинного навчання, здатних враховувати нелінійність, структурні зміни й високу розмірність даних [10]. Така еволюція зумовлена потребою підвищити точність прогнозів дохідностей і волатильності, а також побудувати більш адаптивні стратегії розподілу капіталу в умовах нестабільних ринків [11].

Традиційні статистичні підходи до моделювання фінансових часових рядів базуються насамперед на сімействі ARIMA-моделей та їхніх узагальненнях, що враховують сезонність, тренди й автокореляцію в даних. Для опису змінної волатильності широко застосовуються GARCH-моделі та їхні модифікації, які дозволяють відтворювати «кластеризацію» волатильності, ефект «важких хвостів» і використовуються, зокрема, для оцінювання ринкового ризику й прогнозування показників VaR [12]. Попри відносну простоту і прозорість інтерпретації, такі моделі демонструють обмежену ефективність за умов різких режимних змін, нелінійних взаємодій

між змінними та на високочастотних даних, де проявляються складні патерни, недоступні для лінійних структур.

Паралельно розвивалися методи машинного навчання, що використовуються як для безпосереднього прогнозування цін або доходностей, так і для побудови допоміжних ознак, класифікації ринкових режимів і оцінки ризикових характеристик. До базових моделей належать регресійні алгоритми (регуляризовані лінійні моделі, дерева рішень, ансамблі на кшталт Random Forest і Gradient Boosting), а також методи кластеризації та зниження розмірності, які допомагають виділити приховані фактори, важливі для динаміки активів. Важливою перевагою цих підходів є гнучкість у роботі з великою кількістю ознак - технічних індикаторів, фундаментальних показників, макроекономічних змінних - та можливість моделювати складні, нелінійні зв'язки між ними [13].

Найбільш інтенсивний розвиток у сфері фінансового прогнозування отримали моделі глибинного навчання, зокрема рекурентні нейронні мережі (RNN) та їхні модифікації LSTM і GRU, спрямовані на моделювання довгострокових залежностей у часових рядах. Ряд досліджень демонструє, що LSTM- та GRU-архітектури перевершують класичні ARIMA-моделі в задачах коротко- та середньострокового прогнозування цін акцій і біржових індексів, особливо коли в модель включаються розширені набори технічних індикаторів і макроекономічних змінних. Подальший розвиток пов'язаний із застосуванням Transformer-архітектур і гібридних моделей (наприклад, Conv1D-LSTM-GRU або Transformer-GRU), які використовують механізм уваги для виділення найбільш інформативних тимчасових фрагментів і демонструють покращені результати порівняно з традиційними рекурентними мережами [14].

Отримані прогнози доходностей і ризикових характеристик інтегруються в задачу оптимізації портфеля, що традиційно формулюється на основі сучасної портфельної теорії Марковіца як максимізація очікуваної дохідності за умови обмеження ризику або мінімізація ризику при заданій

дохідності. Класичні mean-variance-моделі доповнюються критеріями, що враховують асиметрію розподілу доходностей, показники VaR/CVaR, обмеження на ліквідність, транзакційні витрати та регуляторні вимоги, що приводить до постановок опуклої й не опуклої оптимізації з численними лінійними й нелінійними обмеженнями [15]. У сучасних дослідженнях пропонується поєднання машинного навчання для оцінювання очікуваних доходностей, коварацій і ризикових метрік з оптимізаційними моделями, включно з risk-parity, динамічними risk-budgeting-підходами та методами підкріплювального навчання, що формулюють управління портфелем як марківський процес прийняття рішень.

Завдяки такій інтеграції прогнозних і оптимізаційних компонентів стає можливим перехід від статичних стратегій з фіксованими вагами до адаптивних алгоритмів, які реагують на зміни ринкового режиму, кореляційної структури та рівня ризику. При цьому ключовими викликами залишаються забезпечення стійкості моделей до перенавчання, коректна валідація за схемами walk-forward, урахування транзакційних витрат і ліквідності, а також прозорість та інтерпретованість складних ML-рішень у контексті реального інвестиційного процесу. Саме ці аспекти визначають подальший розвиток статистичних і ML-підходів до прогнозування фінансових часових рядів та оптимізації інвестиційних портфелів, що є методологічною основою для побудови інтелектуальних систем управління активами.

1.3. Аналіз існуючих рішень у сфері управління інвестиційним портфелем

У сфері автоматизованого управління інвестиційним портфелем уже сформувався широкий спектр реальних рішень - від масових robo-advisors до професійних алгоритмічних торгових платформ, які відрізняються цільовою аудиторією, глибиною автоматизації та рівнем гнучкості налаштувань.

Betterment є одним із найстаріших і найбільш відомих robo-advisors, орієнтованих на роздрібних інвесторів, що надає повністю автоматизовані

послуги з формування й супроводу портфеля. Користувач проходить анкетування щодо фінансових цілей, горизонту інвестування та толерантності до ризику, після чого платформа будує диверсифікований портфель на основі ETF, автоматично здійснює ребалансування, реалізує стратегії податкової оптимізації та, за потреби, застосовує динамічні зміни структури активів залежно від віку й цілей клієнта. Архітектура сервісу приховує деталі моделей від користувача, пропонуючи простий інтерфейс і прозорі метрики результативності, що робить його привабливим для інвесторів без глибокої фінансової чи технічної підготовки [16].

Wealthfront займає близький ринковий сегмент, але робить більший акцент на ширшій екосистемі фінансових сервісів і розширених алгоритмічних функціях. Поряд із базовим автоматизованим портфелем індексних фондів Wealthfront пропонує додаткові модулі, такі як податкова оптимізація на рівні окремих акцій (direct indexing), «smart beta» та можливість врахування зовнішніх рахунків для комплексного управління добробутом. Платформа позиціонує себе як повністю цифровий «фінансовий планувальник», інтегруючи моделі прогнозування грошових потоків клієнта, планування пенсійних цілей та оптимізації заощаджень, що виходить за рамки суто портфельного підходу Betterment.

Interactive Brokers представляє інший клас рішень - професійну брокерсько-торгову платформу, орієнтовану на активних трейдерів, хедж-фонди та кількісних дослідників. Через Trader Workstation (TWS), веб-інтерфейс і розгалужену систему API (зокрема, Python-та REST-інтерфейси) Interactive Brokers надає доступ до широкого спектра фінансових інструментів у всьому світі, глибоких ринкових даних, модулів управління маржею та ризиком, а також до інструментів побудови й тестування алгоритмічних стратегій. Користувач може реалізувати власні моделі прогнозування та оптимізації в зовнішньому середовищі, а брокерська платформа виконує роль надійного шлюзу для маршрутизації ордерів, управління позиціями та збору операційної статистики.

MetaTrader історично асоціюється з валютним ринком і контрактами на різницю (CFD), але фактично є універсальним середовищем для створення, тестування й виконання алгоритмічних стратегій. Платформа містить вбудовану мову програмування (MQL4/MQL5), конструктори радників (Expert Advisors), засоби візуального та пакетного бек-тестингу на історичних даних, а також модулі оптимізації параметрів стратегій [17]. MetaTrader орієнтований на розробників торгових алгоритмів, які можуть розгортати власні радники на стороні клієнта або сервера, інтегруючи технічні індикатори, сигнали з зовнішніх систем та базові елементи управління ризиком (стоп-лоси, тейк-профіти, трейлінг-стопи тощо).

Порівнюючи ці рішення, можна відзначити, що Betterment і Wealthfront реалізують модель «чорної скриньки» для користувача з максимальною простотою взаємодії, але обмеженою можливістю кастомізації портфельних алгоритмів. Натомість Interactive Brokers та MetaTrader надають високий рівень гнучкості й контроль над стратегіями, вимагаючи від користувача значної технічної й фінансової компетенції, включно з умінням програмувати, тестувати моделі та працювати з ринковими даними. Розроблювана в магістерській роботі інтелектуальна система доцільно позиціонується між цими полюсами: вона може запозичувати від Betterment/Wealthfront модель автоматизованого циклу «прогноз-оптимізація-ребалансування» з дружнім інтерфейсом, а від Interactive Brokers/MetaTrader - можливість інтеграції власних ML-моделей (LSTM, GRU, Transformer), гнучкі налаштування портфельних обмежень і прозору інфраструктуру для бек-тестингу та відтворюваності експериментів.

Порівняння інвестиційних платформ

Критерій	Betterment	Wealthfront	Interactive Brokers	MetaTrader
Тип рішення	Robo-advisor для роздрібних інвесторів	Robo-advisor + інвестплатформа «all-in-one»	Професійна брокерсько-торгова платформа з API	Клієнтська платформа для алгоритмічної торгівлі
Рівень автоматизації портфеля	Повна автоматизація: побудова, ребаланс, tax-loss harvesting	Повна автоматизація портфеля + додаткові сервіси планування й кеш-менеджменту	Залежить від користувача: автоматизація через власні алгоритми й API	Автоматизація через Expert Advisors (роботи) та скрипти MQL4/5
Інструменти та активи	ETF, індексні фонди, тематичні портфелі	ETF, ESG-фонди, окремі акції, крипто-ETFs, кеш-акаунт	Акції, облігації, опціони, ф'ючерси, фореक्स, фонди на глобальних біржах	Фореक्स, CFD на індекси, акції, сировинні товари (через брокерів)
Інтерфейс і зручність	Дуже простий мобільний і веб-інтерфейс, орієнтований на «не-квантів»	Сучасний застосунок «усе в одному» (кеш, інвестиції, планування)	Професійний інтерфейс TWS + веб/мобільний	Знайомий трейдерам інтерфейс з графіками й ордер-вікнами, але з технічними налаштуваннями

1.4. Постановка задачі дослідження та вибір засобів реалізації

У попередніх підрозділах було показано, що класичні підходи до формування інвестиційних портфелів обмежені припущеннями лінійності, стаціонарності та нормальності розподілів, тоді як сучасні фінансові ринки демонструють нелінійну поведінку, змінні режими волатильності та сильний

вплив інформаційних потоків. Це обумовлює потребу в інтелектуальних системах, здатних поєднати моделі машинного навчання для прогнозування часових рядів із оптимізаційними модулями, що враховують реальні ринкові обмеження.

Основною науковою задачею дослідження є розробка та дослідження інтелектуальної системи управління інвестиційним портфелем, яка поєднує моделі прогнозування дохідностей/волатильності фінансових інструментів на основі машинного навчання з модулем оптимізації структури портфеля за критеріями «ризик-дохідність». Формально це означає побудову ланцюжка «дані → прогнози → оптимізація → портфель → оцінка результатів», у якому кожен етап має бути кількісно формалізований і піддаватися відтворюваному експериментальному аналізу. Практична задача полягає у створенні прототипу програмної системи, яка дозволяє завантажувати й попередньо обробляти ринкові дані, тренувати та порівнювати моделі прогнозування, розв'язувати задачі оптимізації портфеля за заданими обмеженнями, а також проводити бек-тестування сформованих стратегій на історичних даних [18].

Для досягнення зазначеної мети доцільно сформулювати такі основні завдання: проаналізувати існуючі статистичні та ML-підходи до прогнозування фінансових часових рядів і методи оптимізації портфеля; обрати й формалізувати одну або декілька постановок задачі (наприклад, mean-variance-оптимізація, ризик-паритет, мінімізація CVaR) [19]; розробити підхід до навчання моделей (LSTM/GRU/Transformer чи інші релевантні архітектури) з урахуванням специфіки часових рядів і уникнення витoku інформації; спроектувати модуль оптимізації, який перетворює прогнози на вектори ваг активів; реалізувати інструменти оцінювання якості прогнозів і ефективності портфеля порівняно з базовими стратегіями [20]. Окремим завданням є забезпечення можливості відтворюваності експериментів: фіксація версій даних, конфігурацій моделей і параметрів оптимізації.

Вибір засобів реалізації повинен відповідати як вимогам до надійної роботи з часовими рядами, так і потребам у гнучкому розгортанні прототипу. На рівні мов програмування та фреймворків природним вибором є Python завдяки розвинутій екосистемі для аналізу даних, машинного навчання та оптимізації. Для реалізації моделей глибокого навчання доцільно використати PyTorch або TensorFlow/Keras, оскільки вони забезпечують підтримку рекурентних архітектур і Transformer-моделей, а також інструменти для побудови кастомних шарів і функцій втрат. Модуль оптимізації портфеля може бути реалізований на основі cvxpy або подібних бібліотек, що дозволяють формувати квадратичні та лінійні задачі з обмеженнями й отримувати чисельно стійкі розв'язки [21].

Як інфраструктурну основу програмної системи доцільно використовувати стек веб-застосунку з окремими шарами для зберігання даних, бізнес-логіки та користувацького інтерфейсу. Реляційна база даних (наприклад, PostgreSQL) може застосовуватися для зберігання історичних котирувань, результатів експериментів і конфігурацій моделей, що спрощує відстеження версій і порівняння запусків. На серверному боці доцільно використати легковагий веб-фреймворк (наприклад, FastAPI) для реалізації REST-інтерфейсів до модулів прогнозування та оптимізації, тоді як клієнтська частина може бути реалізована у вигляді веб-інтерфейсу (React або інший сучасний фреймворк) для візуалізації динаміки портфеля, графіків прогнозів і метрик якості. Така архітектура забезпечує модульність, масштабованість та можливість подальшого розширення системи - зокрема, інтеграції додаткових моделей, альтернативних постановок задачі оптимізації та нових класів фінансових інструментів.

Узагальнюючи, постановка задачі дослідження передбачає не лише розробку окремих алгоритмів прогнозування або оптимізації, а саме побудову цілісної інтелектуальної системи, здатної у відтворюваний спосіб перетворювати ринкові дані на портфельні рішення й оцінювати їх ефективність. Вибір засобів реалізації орієнтований на досягнення балансу

між гнучкістю дослідження (можливість швидко змінювати моделі та гіперпараметри) і потенційною придатністю системи до практичного використання в умовах реальних фінансових ринків.

Висновки до розділу 1

У першому розділі проведено комплексний аналіз предметної області управління інвестиційними портфелями та сформульовано основні завдання дослідження. Встановлено, що сучасне функціонування інвестиційних портфелів характеризується поєднанням класичних принципів диверсифікації з новими викликами, пов'язаними з підвищеною ринковою волатильністю, зростанням кореляцій між класами активів у кризові періоди, посиленням регуляторних вимог та цифровою трансформацією галузі. Інвестиційний портфель розглядається як динамічна система, що потребує безперервного аналізу та адаптації до мінливих ринкових умов.

Проаналізовано еволюцію методів прогнозування фінансових часових рядів - від традиційних статистичних моделей ARIMA та GARCH до сучасних алгоритмів машинного та глибинного навчання, зокрема LSTM, GRU та трансформерних архітектур. Визначено, що останні демонструють вищу ефективність у задачах прогнозування через здатність враховувати нелінійні залежності, структурні зміни та високу розмірність даних. Окремо розглянуто методи оптимізації портфеля, що базуються на сучасній портфельній теорії Марковіца та її модифікаціях, які враховують асиметрію розподілів, показники VaR/CVaR та реальні ринкові обмеження.

Проведено порівняльний аналіз існуючих рішень у сфері автоматизованого управління портфелями, зокрема robo-advisors та професійних торгових платформ. Визначено, що існуючі системи залишають нішу для розробки інтелектуальної платформи, яка поєднуватиме зручність автоматизації з глибокою кастомізацією, прозорістю алгоритмів та інтеграцією власних моделей машинного навчання.

На основі проведеного аналізу сформульовано основну задачу дослідження - розробку інтелектуальної системи управління інвестиційним

портфелем, що поєднує передові ML-моделі прогнозування з адаптивними методами оптимізації. Обґрунтовано вибір інструментів реалізації для моделей машинного навчання, сучасного веб-стека для створення масштабованої та відтворюваної платформи. Встановлено, що розроблена система має займати проміжне положення між повністю автоматизованими robo-advisors та професійними платформами, пропонуючи глибоку аналітику та гнучкість налаштувань при збереженні зручного інтерфейсу.

РОЗДІЛ 2. РОЗРОБКА ТА ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

2.1. Архітектура інтелектуальної системи управління інвестиційним портфелем

Інтелектуальна система управління інвестиційним портфелем побудована як веб-орієнтований програмний комплекс, що реалізує повний цикл роботи з фінансовими даними: від збору ринкової інформації та формування датасетів до прогнозування, оптимізації структури портфеля та візуалізації результатів користувачу. Для реалізації серверної логіки та аналітичних компонентів доцільно використати мову програмування Python, яка має розвинуту екосистему бібліотек для обробки даних, машинного навчання та побудови веб-сервісів. На відміну від простого аналізатора аномалій, у даній системі зосереджено декілька взаємопов'язаних модулів: модулі збору й підготовки ринкових даних, модулі прогнозування (LSTM / GRU / Transformer), модуль оцінки ризику та оптимізації портфеля, а також веб-інтерфейс для керування сценаріями та перегляду результатів [22].

Для реалізації логіки обробки даних і моделювання пропонується використовувати такі основні бібліотеки:

- pandas - для завантаження, очищення й перетворення табличних даних та часових рядів;
- NumPy - для роботи з багатовимірними масивами та векторизованих чисельних обчислень;
- scikit-learn - для реалізації базових моделей машинного навчання (лінійні моделі, Random Forest, Gradient Boosting), інструментів препроцесингу та оцінки якості;
- TensorFlow / Keras або PyTorch - для побудови й навчання моделей LSTM, GRU, Transformer та автоенкодерів;
- Plotly / Recharts - для інтерактивної візуалізації часових рядів, прогнозів, ризикових метрик і динаміки портфеля у веб-інтерфейсі;

- FastAPI - для створення високопродуктивного веб-API, через який фронтенд взаємодіє з аналітичними модулями.

На відміну від вузькоспеціалізованого додатку виявлення аномалій, запропонована система розширює функціональність: вона не лише аналізує поведінку часових рядів, а й генерує прогнози доходностей/волатильності, оцінює ризики та перетворює ці результати на конкретні рішення щодо ваг активів у портфелі. Така архітектура дає змогу створити гнучкий, відносно універсальний інструмент, який може застосовуватися до різних наборів фінансових інструментів і ринкових сценаріїв [23].

2.1.1. Загальна програмно-технічна архітектура системи

Загальна програмно-технічна архітектура системи передбачає поділ на три основні рівні: клієнтський (фронтенд), серверний (API та бізнес-логіка) і рівень даних (сховище ринкової інформації та артефактів моделей). На клієнтському рівні використовується веб-інтерфейс, реалізований засобами React з використанням Vite як інструмента збірки та Tailwind CSS для побудови уніфікованого та адаптивного дизайну. Інтерфейс забезпечує авторизацію користувачів, перегляд дашбордів із часовими рядами цін, прогнозами, портфельними метриками, а також форми для налаштування параметрів моделей і сценаріїв оптимізації (тип моделі, горизонт прогнозу, обмеження по активам, режим лонг-онлі чи з короткими позиціями тощо).

Серверний рівень побудований на базі FastAPI, що дозволяє створювати REST-сервіси з чітко описаними маршрутами, автоматичною генерацією документації, валідацією вхідних даних і зручною інтеграцією з асинхронними задачами. API надає ендпоїнти для: запуску оновлення ринкових даних; тренування моделей; отримання прогнозів і ризикових оцінок; розв'язання задачі оптимізації портфеля; отримання історії бек-тестів та журналу прийнятих рішень [24]. Для опису форматів запитів і відповідей застосовуються моделі на основі Pydantic, що забезпечує валідацію типів і непорожність критичних полів.

Для довготривалих і ресурсомістких операцій (завантаження великих масивів даних, навчання LSTM/Transformer-моделей, запуск серій бек-тестів) використовується окремий шар оркестрації задач на основі механізму черг (наприклад, Celery або RQ) [25]. Серверний API приймає запит, ставить задачу до черги та повертає користувачеві ідентифікатор, за яким можна відстежувати статус виконання та згодом отримати результат. Це дозволяє уникнути блокування веб-інтерфейсу під час тривалих обчислень і наближує архітектуру до реальних промислових систем.

У системі передбачені ролі користувачів, наприклад, «admin» та «user». Для забезпечення безпеки реалізується аутентифікація на основі токенів, авторизація за ролями, контроль доступу до критичних функцій (наприклад, змін ваг реального портфеля) [27]. Система задовольняє базові нефункціональні вимоги: час відповіді для простих запитів (отримання останніх метрик) не перевищує 1 секунди, для складних (запуск бек-тесту) допускається більший час із відкладеним отриманням результатів; інтерфейс будується за сучасними принципами UI/UX, а архітектура передбачає можливість подальшого масштабування й розширення функціоналу без порушення існуючої логіки [28].

2.1.2. Модулі збору та зберігання ринкових даних

Модуль збору ринкових даних відповідає за інтеграцію з зовнішніми провайдерами (біржові API, фінансові агрегатори, відкриті джерела), отримання історичних та оновлення поточних котирувань, а також додаткової інформації (обсяги торгів, корпоративні події, індекси, макроекономічні показники). Для цього реалізуються спеціалізовані адаптери, які здійснюють HTTP-запити, враховують ліміти на кількість викликів, додають повторні спроби у випадку тимчасових помилок мережі та логують усі запити для подальшого аудиту. Отримані дані проходять через ETL-процес: уніфікація форматів полів, узгодження таймзон, ресемплінг до цільової частоти (наприклад, денна чи годинна), перевірка на пропуски й

аномальні значення, агрегація з різних джерел та приведення до єдиного стандарту, який далі використовується аналітичними модулями [28].

Основним сховищем структурованих даних виступає реляційна база даних PostgreSQL. У ній зберігаються:

- довідник інструментів (ідентифікатори, тікери, біржа, валюта, секторальна належність);
- історичні ряди цін і обсягів (таблиці OHLCV з індексами за інструментом і часом);
- фундаментальні та макроекономічні показники;
- результати прогнозів моделей (очікувані доходності, прогнозна волатильність, ймовірності класів);
- склади портфеля та історія угод;
- ризикові й портфельні метрики, обчислені в процесі бек-тестів і реального часу.

Схема бази даних спроектована з урахуванням специфіки часових рядів: ключові таблиці мають комбіновані індекси «інструмент-час», можливе партиціонування за датою для прискорення запитів на великих історичних проміжках. Для прискорення доступу до «гарячих» даних (останній стан портфеля, поточні прогнози, короткі фрагменти ряду для онлайн-графіків) використовується Redis як кеш, що дозволяє зменшити навантаження на PostgreSQL та забезпечити швидкий відгук для веб-інтерфейсу.

Окремий модуль відповідає за зберігання артефактів моделей та експериментів: файли з вагами нейронних мереж, збережені конфігурації гіперпараметрів, трансформери ознак, журнали навчання та бек-тестів. Вони зберігаються в файловому або об'єктному сховищі з підтримкою версіонування, що дозволяє відтворити будь-який запуск моделі та порівняти результати різних конфігурацій [29]. Після завантаження й підготовки ринкових даних система забезпечує їх стандартизоване представлення для модулів машинного навчання: усі перетворення (масштабування, побудова

ознак) виконуються послідовно та документуються, щоб гарантувати узгодженість між тренувальними, валідаційними та тестовими вибірками [30].

З погляду експлуатації, підсистема даних підтримує регулярне резервне копіювання бази, журнали змін і аудиту доступу, а також механізми відновлення у випадку збоїв. Це критично важливо для фінансових застосунків, у яких історичні дані, результати експериментів і журнали рішень мають значну цінність як з аналітичної, так і з регуляторної точки зору. У сукупності модулі збору й зберігання ринкових даних утворюють надійну інфраструктурну основу, на якій будуються подальші компоненти системи - модулі прогнозування, оцінки ризику та оптимізації структури інвестиційного портфеля.

2.2. Вибір та проєктування моделей машинного навчання для прогнозування й оцінки ризику

У межах інтелектуальної системи управління інвестиційним портфелем модулі машинного навчання відповідають за побудову прогнозів доходностей і волатильності фінансових інструментів, оцінювання ризику, виявлення аномалій та ідентифікацію ринкових режимів, на основі яких приймаються портфельні рішення [31]. Мета цього підсистемного рівня полягає не лише в отриманні точного прогнозу, а й у формуванні стабільних, інтерпретованих і придатних до інтеграції в оптимізаційну модель характеристик, таких як очікувана дохідність, прогнозна ковараційна структура, показники ризику «tail-risk» та маркери зміни режимів [32].

Під час вибору моделей використовуються такі ключові критерії:

- 1) стабільність якості прогнозів у часі та на різних ринкових режимах;
- 2) точність на out-of-sample відрізках (walk-forward-валідація, блокова крос-валідація для часових рядів);
- 3) чутливість до режимів ринку (здатність коректно реагувати на періоди підвищеної волатильності, трендів і флету);

4) інтерпретованість (у тому числі засобами методів пояснюваного ШІ - SHAP, важливість ознак);

5) обчислювальна ефективність, швидкість інференсу та можливість експлуатації в напівреальному часі [33].

Для вирішення цих задач у системі комбінуються три групи моделей: глибинні рекурентні та трансформерні архітектури для прогнозування числових часових рядів; класичні ML-моделі як бенчмарки й інструменти оцінки ризику; моделі виявлення аномалій і режимів для побудови «regime features» та контролю якості даних.

Підсистема прогнозування й оцінки ризику базується на поєднанні глибинних рекурентних моделей (LSTM, GRU), згорткових часових мереж (TCN), Transformer-архітектур та класичних алгоритмів машинного навчання, які разом формують ансамбль моделей для різних задач: прогноз доходностей і волатильності, побудова сигналів, оцінка ризикових характеристик, виявлення аномалій та режимів ринку [34]. Вхідними для більшості моделей є багатовимірні часові вікна ознак $X_t = [x_{t-L+1}, \dots, x_t]$, де кожен вектор x_t включає лаги доходностей, технічні індикатори, оцінки волатильності, макроекономічні змінні та ознаки ринкових режимів, а виходом є прогноз цільових величин (доходність, волатильність, клас напрямку, параметри ризику) на горизонті H .

Базові моделі класу LSTM (Long Short-Term Memory) будуються як послідовність одного або кількох рекурентних шарів, кожний з яких має механізм «воріт», що контролюють потік інформації [35]. Для кожного кроку часу t LSTM-блок обчислює вектори воріт забування f_t , оновлення i_t , кандидата нового стану $\tilde{C}_t$, вихідних воріт o_t , оновлений стан комірки C_t та прихований стан h_t :

$$\begin{aligned} f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f), \quad i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \\ \tilde{C}_t &= \tanh(W_C[h_{t-1}, x_t] + b_C), \quad C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \\ o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o), \quad h_t = o_t \odot \tanh(C_t). \end{aligned}$$

Завдяки цим операціям мережа здатна «забувати» неактуальну інформацію та зберігати важливі залежності на тривалих часових інтервалах, що особливо важливо для фінансових рядів із режимними зламами та періодами високої волатильності. Для задач регресійного прогнозу (наприклад, лог-доходності $y_t = \ln(P_{t+H}/P_t)$) вихід LSTM-шару подається на щільний (Dense) шар, а параметри моделі θ налаштовуються шляхом мінімізації середньоквадратичної помилки або іншої відповідної функції втрат.

Моделі GRU (Gated Recurrent Unit) реалізують схожу ідею через оновлювальні та скидувальні ворота, але мають спрощену структуру й меншу кількість параметрів, що зменшує час навчання та інференсу. Їх доцільно використовувати для задач, де необхідно одночасно обробляти велику кількість інструментів або здійснювати часті оновлення прогнозів [36]. Для обох типів рекурентних мереж важливими гіперпараметрами виступають:

- 1) довжина вхідного вікна L (кількість кроків у минуле);
- 2) кількість рекурентних шарів;
- 3) число нейронів у кожному шарі;
- 4) рівень Dropout для регуляризації;
- 5) тип оптимізатора (наприклад, AdamW);
- 6) розмір батчу й критерії ранньої зупинки на основі метрик валідації [37].

Для моделювання дуже довгих часових залежностей та підвищення обчислювальної ефективності застосовуються Temporal Convolutional Networks (TCN) і Transformer-архітектури. У TCN рекурентність замінена каузальними згортками з діляціями, що дозволяє мережі бачити широкий часовий контекст із фіксованою глибиною:

$$h_t^{(l)} = \sum_{k=0}^{K-1} w_k^{(l)} \cdot h_{t-d_l k}^{(l-1)},$$

де K - розмір ядра, d_l - діляція на шарі l , $h_t^{(0)} = x_t$. Така архітектура забезпечує паралельність обчислень і скорочує глибину мережі порівняно з LSTM, що критично для швидкого інференсу на великих портфелях [38].

Transformer-моделі використовують механізм уваги для моделювання залежностей між усіма тайм-степами одночасно. Для послідовності вхідних ембедингів формується трійка матриць запитів, ключів і значень:

$$Q = XW_Q, K = XW_K, V = XW_V,$$

після чого обчислюється масштабована скалярна увага (scaled dot-product attention):

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V,$$

де d_k - розмірність простору ключів. Додавання позиційних кодувань дозволяє зберігати інформацію про порядок спостережень, а багатоголовий (multi-head) механізм уваги дає змогу моделі паралельно вивчати різні типи залежностей. Такі архітектури особливо корисні для multi-series-налаштувань, коли одночасно моделюються ряди для великої кількості активів.

Паралельно з глибинними мережами в системі використовуються класичні моделі машинного навчання для побудови бенчмарків і додаткових оцінок ризику:

- 1) Регуляризовані лінійні моделі (Ridge, Lasso, ElasticNet);
- 2) Дерева рішень і випадкові ліси (Random Forest);
- 3) Методи Gradient Boosting (XGBoost, LightGBM).

Вони працюють на основі матриці ознак, сформованої з лагів, ковзних статистик, технічних індикаторів, кореляцій і макроознак, і дозволяють отримати більш інтерпретовані оцінки впливу факторів на доходність або ризик. Такі моделі використовуються як окремі предиктори або як частина ансамблю, у якому виходи різних алгоритмів комбінуються (наприклад, через зважене середнє або метамодель) залежно від їхньої валідаційної якості.

Окремий блок становлять моделі виявлення аномалій та ринкових режимів. Автокодери (у т. ч. LSTM-Autoencoder) навчаються відновлювати «нормальну» динаміку часових рядів; значне зростання реконструкційної помилки інтерпретується як сигнал про аномалію чи перехід у нестандартний режим:

$$\hat{x} = g_{\phi}(f_{\theta}(x)), \mathcal{L}_{\text{rec}}(\theta, \phi) = \frac{1}{N} \sum_{t=1}^N \|x_t - \hat{x}_t\|_2^2.$$

Приховані марківські моделі або алгоритми кластеризації (k-means, Gaussian Mixture Models) виділяють стани ринку (спокій/стрес, висока/низька волатильність):

$$P(x_t|z_t = k) = \mathcal{N}(\mu_k, \Sigma_k), P(z_t|z_{t-1}) = A_{z_{t-1}, z_t},$$

які далі використовуються як додаткові ознаки (regime features) у прогнозах моделей і в модулі оптимізації портфеля.

Навчання й налаштування всіх моделей організовано в єдиному пайплайні: для кожної конфігурації фіксуються тип моделі, гіперпараметри, набір ознак, схема розбиття даних (хронологічний спліт, walk-forward), метрики на тренувальних, валідаційних і тестових відрізках [39]. Це дозволяє систематично порівнювати LSTM, GRU, TCN, Transformer і класичні моделі між собою, формувати ансамблі на основі найкращих кандидатів та інтегрувати їхні прогнози в модуль оптимізації структури інвестиційного портфеля. На етапі інференсу виходи моделей можуть комбінуватися через зважене усереднення або більш складні схеми ансамблювання, що підвищує стійкість системи до помилок окремих моделей і сприяє краще адаптованим портфельним рішенням.

Формування датасетів для моделей машинного навчання в задачах фінансового прогнозування є критичним етапом, оскільки навіть незначний витік інформації з майбутнього (data leakage) призводить до штучного завищення метрик та некоректних висновків щодо якості моделей. Основний принцип побудови вибірок полягає в суворому дотриманні причинно-часового порядку: усі ознаки, перетворення, нормалізація, відбір

ознак і підбір гіперпараметрів мають використовувати тільки інформацію, доступну на момент формування прогнозу.

Вихідними даними є часові ряди цін P_t (Open, High, Low, Close), обсягів торгів V_t , а також фундаментальні та макроекономічні показники (процентні ставки, фондові індекси, спреди, інфляція тощо), отримані від біржових і інформаційних провайдерів [40]. На етапі ETL виконується:

- 1) узгодження таймзон, торгових календарів і частоти (ресемплінг до денної, годинної чи іншої цільової частоти);
- 2) очищення від артефактів (дублікати, пропуски, явні технічні помилки);
- 3) уніфікація ідентифікаторів інструментів та форматів полів.

Ціни перетворюються на доходності, як правило, логарифмічні:

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

що спрощує агрегування на різних горизонтах та наближає розподіли до гаусового у спокійних режимах ринку. Для багатокрокових цілей використовується сума одно періодних доходностей або лог-доходність на горизонті H :

$$y_t^{(H)} = \ln \frac{P_{t+H}}{P_t}.$$

Ознаки формуються з урахуванням специфіки фінансових часових рядів:

- 1) лаги доходностей $r_{t-1}, \dots, r_{t-L}$, лаги обсягів і спредів;
- 2) ковзні статистики на вікнах w : середнє $\mu_t^{(w)}$, дисперсія $\sigma_t^{2(w)}$, ковзні кореляції між активами;
- 3) оцінки волатильності, зокрема EWMA:
$$\sigma_t^{2(\text{EWMA})} = \lambda \sigma_{t-1}^{2(\text{EWMA})} + (1 - \lambda) r_{t-1}^2, \quad 0 < \lambda < 1;$$
- 4) технічні індикатори (ковзні середні, RSI, MACD, смуги Боллінджера), сезонні ознаки (день тижня, місяць, кінець кварталу), категоріальні ознаки (галузь, країна);

5) макро- та факторні ознаки (динаміка індексів, ставок, кредитних спредів, індикатори стресу на ринку).

Цільові змінні визначаються залежно від задачі:

- 1) прогноз одно- або багатокрокової доходності $y_t^{(H)}$;
- 2) реалізована волатильність на горизонті H :

$$RV_t^{(H)} = \sqrt{\sum_{k=1}^H r_{t+k}^2};$$

3) показники ризику (максимальний локальний drawdown, бінарні мітки перевищення порога збитку);

4) класи напрямку або режимів ($-1, 0, 1$; стани «спокій/стрес» тощо).

Окремо інтегруються ознаки режимів ринку, отримані з прихованих марківських моделей (HMM), алгоритмів кластеризації або автокодерів: кожному моменту часу ставиться у відповідність дискретний або безперервний показник «стану» (спокій, стрес, висока/низька волатильність), який включається до вектора ознак як додатковий інформаційний сигнал.

Нормалізація й масштабування виконуються строго в межах тренувальних вибірок. Параметри трансформерів (середні, стандартні відхилення, мінімальні/максимальні значення, квантілі) обчислюються на train-інтервалі й потім незмінно застосовуються до валідаційних та тестових даних:

$$z_t = \frac{x_t - \mu_{\text{train}}}{\sigma_{\text{train}}}.$$

Такий підхід запобігає використанню інформації з майбутнього у вигляді «підглянутих» статистик. Забороняється застосовувати індикатори, які використовують симетричні вікна навколо моменту t (частина вікна лежить у майбутньому) або будь-які згладжування, що опираються на майбутні спостереження. Пропуски в часових рядах обробляються переважно каузальними методами (forward-fill, моделі, навчені лише на минулих

значеннях), тоді як backward-fill на повному ряду може призвести до витоку інформації.

На етапі побудови складних ознак, які можуть комбінувати дані з різних часів або джерел, забезпечується чітка причинність: наприклад, до моменту t включаються лише ознаки, сформовані з даних t і раніше. Для багатовимірних ознак (наприклад, кореляцій між активами) обчислення здійснюються на розширюваному вікні історії в межах train-інтервалу, але застосування до validation/test вікон виконується з використанням параметрів, оцінених на train-етапі.

Розбиття часових рядів на train/validation/test здійснюється хронологічно, без випадкового перемішування спостережень. Типова схема:

1) тренувальна вибірка (наприклад, 3-5 років історії) - для навчання моделей, оцінювання параметрів масштабування, відбору ознак та підбору гіперпараметрів у межах фіксованої конфігурації;

2) валідаційна вибірка (6-12 місяців) - для комплексного підбору гіперпараметрів, вибору архітектур, ранньої зупинки під час навчання та кінцевої селекції моделей;

3) тестова вибірка (останні 1-2 роки) - для фінальної незалежної оцінки якості, не використовується на жодному етапі налаштування.

Для більш надійної оцінки узагальнювальної здатності та режимної стійкості використовується walk-forward-валідація. Визначаються довжини тренувального T_{train} та тестового T_{test} вікон. Процедура:

1) модель навчається на інтервалі $[t_0, t_0 + T_{\text{train}}]$;

2) тестується на $[t_0 + T_{\text{train}} + 1, t_0 + T_{\text{train}} + T_{\text{test}}]$;

3) вікна зсуваються вперед на T_{test} кроків (expanding або rolling підхід), крок повторюється;

4) метрики агрегуються по всіх кроках, утворюючи послідовність перекладених результатів.

Такий підхід добре відображає реальний сценарій періодичного перенавчання моделей на розширюваному історичному вікні та дозволяє оцінити, наскільки модель адаптується до змін ринкових режимів.

У випадках високої автокореляції та сезонності застосовуються блокові схеми крос-валідації для часових рядів (time series block cross-validation): дані діляться на непересічні часові блоки, кожен із яких по черзі виступає валідуючим, тоді як всі попередні блоки - тренувальними. Це дозволяє уникнути «перетікання» інформації між train і validation через перекриття або близькість часових вікон. Така схема особливо важлива для довгоперіодичних явищ, де звичайна крос-валідація могла б недооцінити помилку тесту.

На кожному кроці walk-forward-валідації обчислюються метрики якості:
Для регресійних задач прогнозування доходностей:

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{test}}} \sum_{t=1}^{N_{\text{test}}} (y_t - \hat{y}_t)^2},$$

$$\text{MAE} = \frac{1}{N_{\text{test}}} \sum_{t=1}^{N_{\text{test}}} |y_t - \hat{y}_t|.$$

Для оцінки «інформаційності» та ранжування прогнозів - Information Coefficient та RankIC:

$$\text{IC} = \text{corr}(y_t, \hat{y}_t),$$

$$\text{RankIC} = \text{corr}(\text{rank}(y_t), \text{rank}(\hat{y}_t)).$$

Для задач класифікації сигналів (напрямок руху, режими):

$$\text{Accuracy} = \frac{\# \text{ правильних передбачень}}{N_{\text{test}}},$$

$$\text{Precision@k} = \frac{\# \text{ влучань у top-k}}{k},$$

та інші метрики, релевантні для портфельних рішень (hit ratio, turnover, Sharpe ratio на сигналах).

Для систематичного контролю можливих джерел data leakage у пайплайн передбачаються:

- 1) перевірка монотонності часових міток та відсутності «переставлених» спостережень у будь-якій вибірці;
- 2) валідація того, що параметри всіх трансформацій (масштабування, зниження розмірності, відбір ознак) обчислено виключно на train-частині кожного кроку walk-forward;
- 3) тести на інваріантність цільових змінних до майбутніх значень (відсутність зміщення, симетричних фільтрів, індикаторів на повному горизонті тощо);
- 4) перевірка консистентності часових міток, правильності порядку побудови ознак, послідовності операцій масштабування.

Усі проміжні артефакти (сирі ряди, очищені таблиці, матриці ознак, параметри трансформерів, схеми сплітів, конфігурації моделей, результати валідації й тестування) версionуються та зберігаються разом у структурованому репозиторії. Кожна версія датасету пов'язана з унікальним ідентифікатором (хеш або мітка версії), що дозволяє відтворити будь-який експеримент: користувач може указати версію датасету, конфігурацію моделі та схему валідації, а система відновить і повторить аналіз, отримавши ідентичні результати.

Таким чином, дотримання строгих вимог до формування датасетів, контролю витоку інформації та організації вибірок у часові вікна забезпечує побудову надійних, стабільних моделей машинного навчання для прогнозування й оцінки ризику та дозволяє безпечно інтегрувати їх прогнози в модуль оптимізації структури інвестиційного портфеля.

2.3. Проектування веб-додатку інтелектуального управління інвестиційним портфелем

Веб-додаток інтелектуального управління інвестиційним портфелем розробляється як багаторівнева інформаційна система, метою якої є забезпечення зручної взаємодії користувачів із фінансовими даними, ефективного управління портфелями та автоматизованого ризик-аналізу й оптимізації.

На діаграмі розгортання зображено три основні вузли: браузер користувача (User Browser), веб-сервер (Web Server) та база даних (Database), яку зображено на рисунку 2.1. У браузері розгорнуто клієнтський застосунок, який відповідає за графічний інтерфейс, відображення портфелів, позицій, транзакцій, прогнозів і ризикових показників. Браузер спілкується з веб-сервером по захищеному протоколу HTTP/HTTPS, надсилаючи запити на виконання операцій (вхід у систему, створення чи зміну портфеля, отримання прогнозів моделей, запуск оптимізації).

Веб-сервер на діаграмі розділено на три логічні блоки: Business Logic, API/Auth і Services. Блок Business Logic реалізує доменні правила: перевірку коректності операцій із портфелями й транзакціями, застосування обмежень за роллю admin/user, обчислення агрегованих показників. Блок API/Auth відповідає за організацію REST-інтерфейсів та механізм аутентифікації й авторизації (наприклад, роботу з JWT-токенами), що дозволяє чітко розмежувати доступ адміністратора й звичайного користувача. У блоці Services згруповано спеціалізовані сервіси - прогнозний, ризиковий, оптимізаційний, а також допоміжні служби для імпорту ринкових даних і формування звітів. Комунікація з вузлом Database здійснюється по TCP/IP; у складі вузла бази даних показано сховище (PostgreSQL) і кеш, який зберігає найчастіше запитувані ринкові ряди й проміжні результати, що прискорює роботу інтерфейсу.

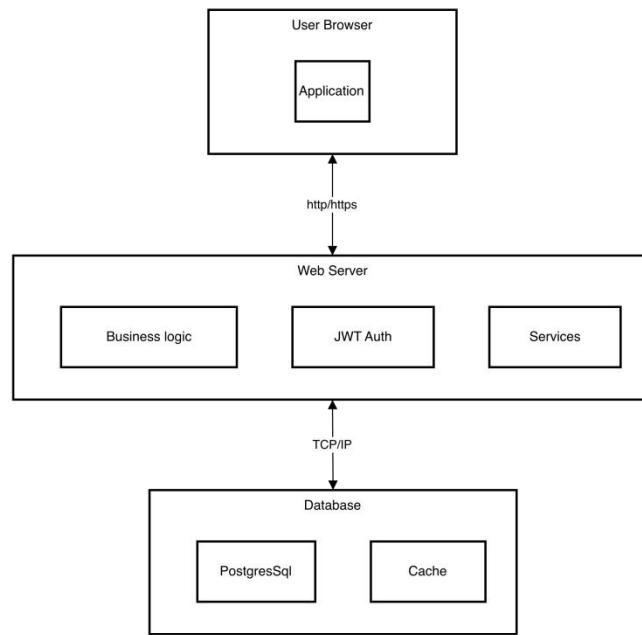


Рис. 2.1. Діаграма розгортання

На діаграмі класів показано основні об'єкти предметної області: User, Portfolio, Asset, Position, Transaction, а також доменні сервіси ForecastService, RiskService, OptimizationService і допоміжні компоненти Repository та RBAC/Role, яку зображено на рисунку 2.2. Клас User містить поля ідентифікатора, email, хешованого пароля, статусу активності й ролі (admin або user), а також методи перевірки пароля, генерації токена доступу, вибірки портфелів користувача. Роль admin, на відміну від user, отримує додаткові можливості - створення нових користувачів, зміна прав доступу, конфігурація моделей та глобальних параметрів системи; це відображено через асоціацію User з класами Role/Permission [40].

Клас Portfolio моделює інвестиційний портфель: містить посилання на власника (User), назву, опис, валюту, часові мітки створення й оновлення, а також методи для додавання й видалення позицій, розрахунку поточної вартості, доходності, ризикових показників. Клас Position пов'язаний з Portfolio і Asset та описує поточну позицію за конкретним інструментом (кількість, середня ціна входу, методи для розрахунку нереалізованого PnL). Клас Asset містить атрибути тикера, типу активу, базової валюти та допоміжні методи для отримання останньої ціни й історії котирувань. Клас Transaction фіксує окремі операції купівлі/продажу з полями портфель, актив,

кількість, ціна, тип операції, дата та методами для побудови грошових потоків і відновлення стану портфеля на будь-яку дату.

Клас Repository інкапсулює доступ до бази даних: реалізує методи create/read/update/delete для всіх основних сутностей, забезпечує транзакційність і приховує деталі ORM-шару. ForecastService працює з історичними даними та моделями (LSTM/GRU/Transformer), повертаючи прогнозовані доходності або розподіли; RiskService обчислює волатильність, VaR, CVaR, просідання на основі позицій і historical_prices; OptimizationService розв’язує задачі оптимізації портфеля з урахуванням обмежень, заданих адміном. Кожен із цих сервісів використовує Repository для читання й запису даних, а також перевіряє права доступу користувача на виконання певних операцій.

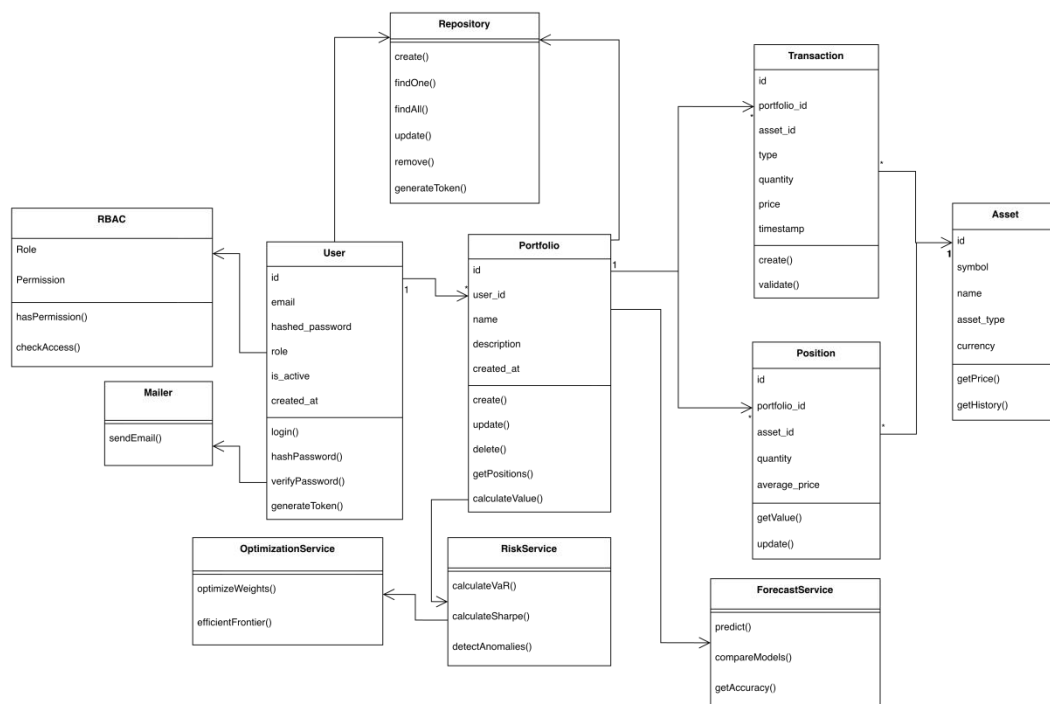


Рис. 2.2. Діаграма класів бізнес-логіки

На діаграмі класів показано основні об’єкти предметної області: User, Portfolio, Asset, Position, Transaction, а також доменні сервіси ForecastService, RiskService, OptimizationService і допоміжні компоненти Repository та RBAC/Role, яку зображено на рисунку 2.3. Клас User містить поля ідентифікатора, email, хешованого пароля, статусу активності й ролі (admin або user), а також методи перевірки пароля, генерації токена доступу, вибірки

портфелів користувача. Роль admin, на відміну від user, отримує додаткові можливості - створення нових користувачів, зміна прав доступу, конфігурація моделей та глобальних параметрів системи; це відображено через асоціацію User з класами Role/Permission.

Клас Portfolio моделює інвестиційний портфель: містить посилання на власника (User), назву, опис, валюту, часові мітки створення й оновлення, а також методи для додавання й видалення позицій, розрахунку поточної вартості, доходності, ризикових показників. Клас Position пов'язаний з Portfolio і Asset та описує поточну позицію за конкретним інструментом (кількість, середня ціна входу, методи для розрахунку нереалізованого PnL). Клас Asset містить атрибути тикера, типу активу, базової валюти та допоміжні методи для отримання останньої ціни й історії котирувань. Клас Transaction фіксує окремі операції купівлі/продажу з полями портфель, актив, кількість, ціна, тип операції, дата та методами для побудови грошових потоків і відновлення стану портфеля на будь-яку дату.

Клас Repository інкапсулює доступ до бази даних: реалізує методи create/read/update/delete для всіх основних сутностей, забезпечує транзакційність і приховує деталі ORM-шару. ForecastService працює з історичними даними та моделями (LSTM/GRU/Transformer), повертаючи прогнозовані доходності або розподіли; RiskService обчислює волатильність, VaR, CVaR, просідання на основі позицій і historical_prices; OptimizationService розв'язує задачі оптимізації портфеля з урахуванням обмежень, заданих адміном. Кожен із цих сервісів використовує Repository для читання й запису даних, а також перевіряє права доступу користувача на виконання певних операцій.

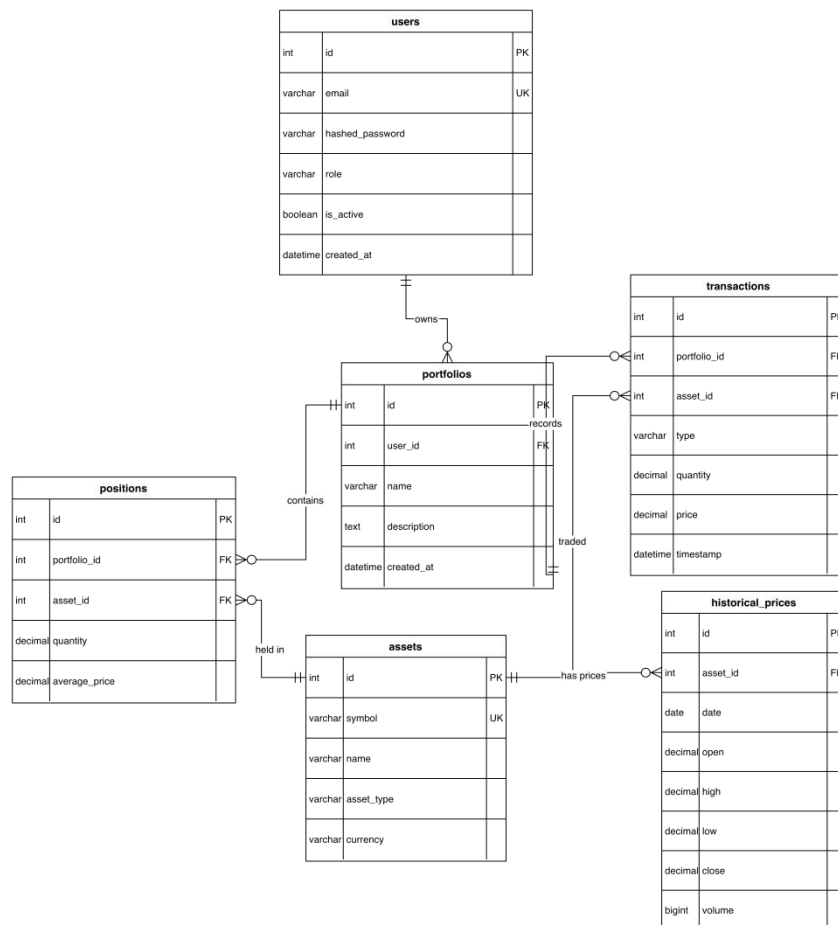


Рис. 2.3.Реляційна схема бази даних

Реляційна схема бази даних деталізує, як об'єкти, показані на діаграмі класів, зберігаються фізично, яку зображено на рисунку 2.4. У таблиці users зосереджено облікові записи: поля id, email, hashed_password, is_active, created_at, а також, за потреби, role або посилання на окрему таблицю ролей. З погляду ролей admin/user саме ця таблиця є джерелом інформації для модуля авторизації. Таблиця portfolios містить записи про портфелі: id, user_id, name, currency, description, timestamps; зовнішній ключ user_id пов'язує портфель з конкретним користувачем, що дозволяє легко вибрати всі портфелі для даного admin або user.

Таблиця assets описує фінансові інструменти: id, ticker, name, asset_type, currency, інші довідкові атрибути. Таблиця positions реалізує зв'язок «багато-до-одного» між portfolios та assets: кожен рядок містить portfolio_id, asset_id, quantity, average_price, що дозволяє у будь-який момент обчислити поточну вартість портфеля на основі останніх цін. Таблиця transactions

включає portfolio_id, asset_id, quantity, price, timestamp, type (buy/sell), фіксуючи всі операції, на підставі яких можна відновити історію змін портфеля. Нарешті, таблиця historical_prices містить історичні дані ринку: asset_id, date/time, open, high, low, close, volume, які використовуються як для розрахунку поточної вартості й ризикових показників, так і як вхід для моделей прогнозування.

Разом ці три діаграми узгоджено описують систему: діаграма розгортання показує, де фізично виконуються клієнтська й серверна частини та як вони взаємодіють; діаграма класів демонструє структуру бізнес-логіки та ролей admin/user; реляційна схема деталізує, як ці об'єкти й зв'язки відображаються у базі даних. Така багаторівнева документація дозволяє надійно реалізувати та розвивати веб-додаток, не порушуючи цілісності даних і правил доступу.

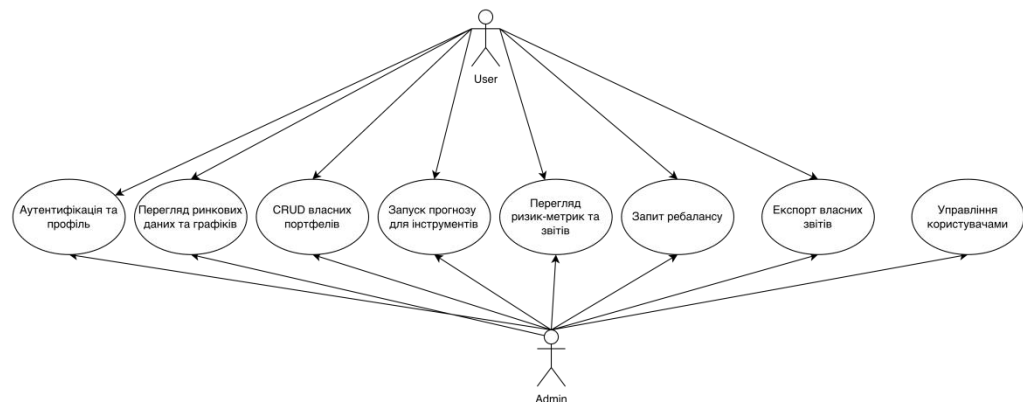


Рис. 2.4. Доступний функціонал залежно від ролі користувача

Висновки до розділу 2

У розділі було розроблено та спроектовано інтелектуальну систему управління інвестиційним портфелем, яка поєднує передові методи машинного навчання з ефективною архітектурою веб-додатку. Розроблено цілісну трирівневу архітектуру системи, що включає клієнтський інтерфейс, серверну логіку та надійне сховище даних, що забезпечує масштабованість, безпеку та високу продуктивність. Створено спеціалізовані модулі збору та обробки ринкових даних, які реалізують ETL-пайплайни для інтеграції різноманітних джерел фінансової інформації. Система включає широкий

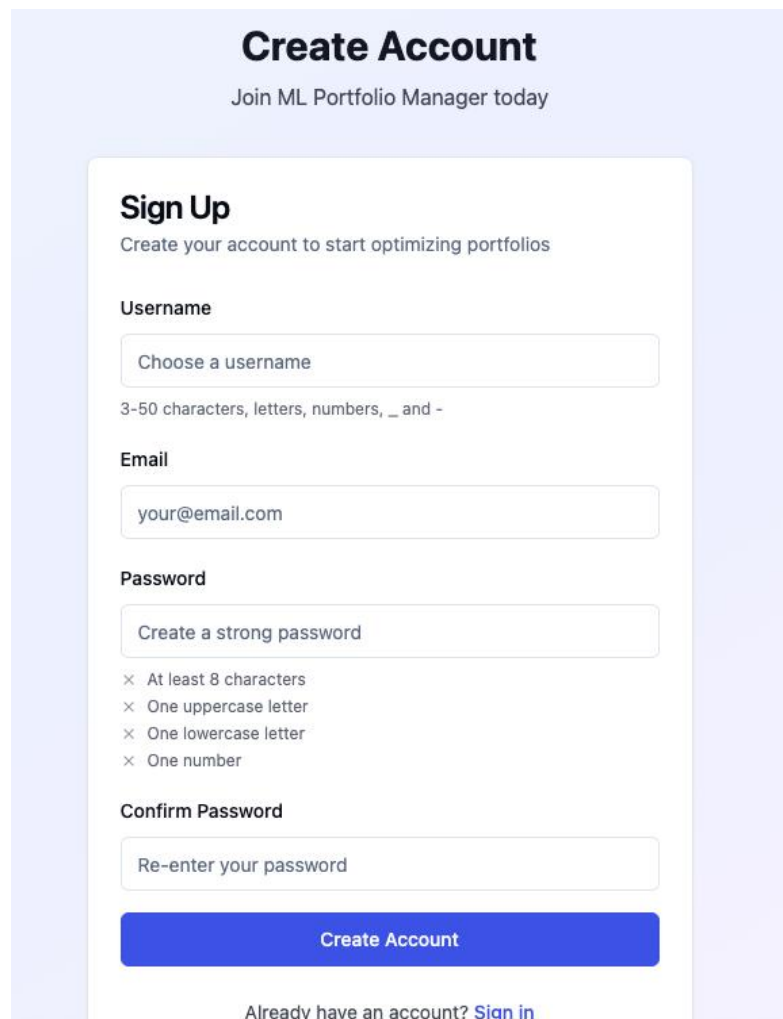
спектр архітектур машинного навчання - від класичних рекурентних мереж до сучасних трансформерів та згорткових часових мереж, що дозволяє ефективно вирішувати різноманітні завдання: прогнозування доходностей, оцінку волатильності, виявлення аномалій та ідентифікацію ринкових режимів. Розроблено методологію підготовки даних з суворими протоколами для уникнення витoku інформації, хронологічного розбиття вибірок та walk-forward валідації, що забезпечує надійність і репрезентативність отриманих прогнозів. Веб-додаток спроектовано з урахуванням сучасних вимог UI/UX, розділенням ролей користувачів та комплексною бізнес-логікою, що охоплює всі аспекти управління портфелем. Детальне моделювання даних відображено через діаграми класів та реляційну схему бази даних, що забезпечує чітку структуру взаємодії між сутностями системи. Розроблена система представляє собою цілісний, технологічно просунутий інструмент, здатний забезпечити інвесторів науково обґрунтованими рекомендаціями щодо формування та оптимізації інвестиційних портфелів у реальному часі. Поєднання глибоких аналітичних можливостей машинного навчання з ергономічним інтерфейсом робить систему потужним рішенням для сучасного фінансового менеджменту.

РОЗДІЛ 3. ОПИС ПРОГРАМНОГО ПРОДУКТУ

3.1. Графічний інтерфейс

При першому відвідуванні веб-додатку користувач бачить головну сторінку з описом платформи та запрошенням розпочати роботу. Для продовження необхідно пройти автентифікацію.

Форма реєстрації нового облікового запису, яку зображено на рисунку 3.1, містить поля для введення імені користувача, електронної пошти та пароля. Інтерфейс надає зрозумілі підказки щодо вимог до даних: ім'я користувача має містити від 3 до 20 символів, а пароль - щонайменше 8 символів. Для підтвердження пароля передбачено окреме поле.

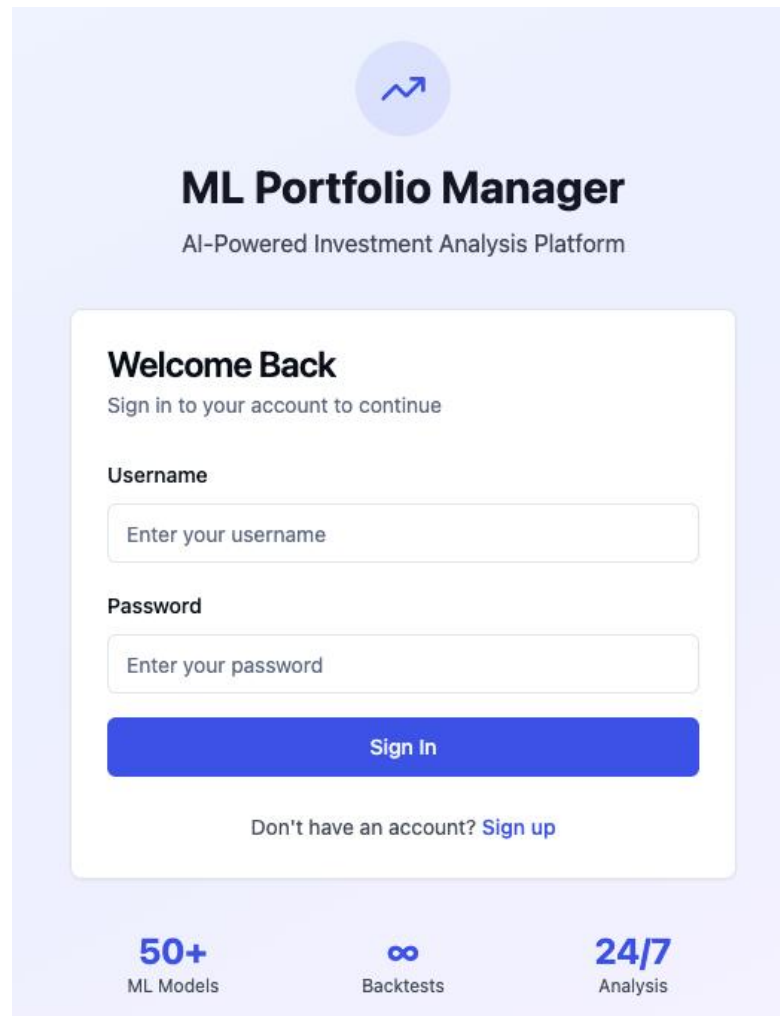


The image shows a web registration form titled "Create Account" with the subtitle "Join ML Portfolio Manager today". The form is titled "Sign Up" and includes the instruction "Create your account to start optimizing portfolios". It contains four input fields: "Username" with a placeholder "Choose a username" and a note "3-50 characters, letters, numbers, _ and -"; "Email" with a placeholder "your@email.com"; "Password" with a placeholder "Create a strong password" and a list of requirements: "At least 8 characters", "One uppercase letter", "One lowercase letter", and "One number"; and "Confirm Password" with a placeholder "Re-enter your password". A blue "Create Account" button is at the bottom, followed by a link "Already have an account? Sign in".

Рис. 3.1. Форма реєстрації нового користувача

Користувачі, які вже мають обліковий запис, можуть увійти в систему через форму входу, яку зображено на рисунку 3.2. Вона містить поля для

імені користувача та пароля. На сторінці також представлена коротка статистика платформи, що підкреслює її можливості.



The image shows a login interface for 'ML Portfolio Manager', an AI-Powered Investment Analysis Platform. At the top, there is a blue circular icon with a white line graph. Below it, the title 'ML Portfolio Manager' is displayed in bold, followed by the subtitle 'AI-Powered Investment Analysis Platform'. The main section is a white card with a 'Welcome Back' heading and a prompt to 'Sign in to your account to continue'. It contains two input fields: 'Username' with the placeholder 'Enter your username' and 'Password' with the placeholder 'Enter your password'. A blue 'Sign In' button is positioned below the password field. At the bottom of the card, there is a link: 'Don't have an account? Sign up'. The footer of the interface features three statistics: '50+ ML Models', '∞ Backtests', and '24/7 Analysis'.

Рис. 3.2. Форма входу в обліковий запис

Після успішної аутентифікації користувач потрапляє на головну інформаційну панель. На ній відображається загальна статистика портфелів користувача, швидкі дії та список останніх аналізів. Головна сторінка зображена на рисунку 3.3.

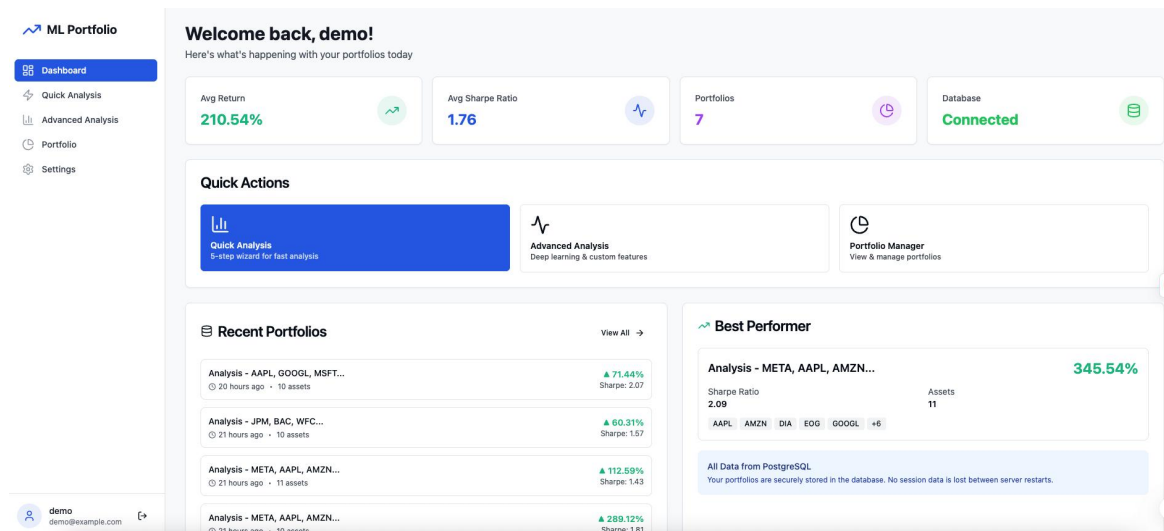


Рис. 3.3. Головна інформаційна панель

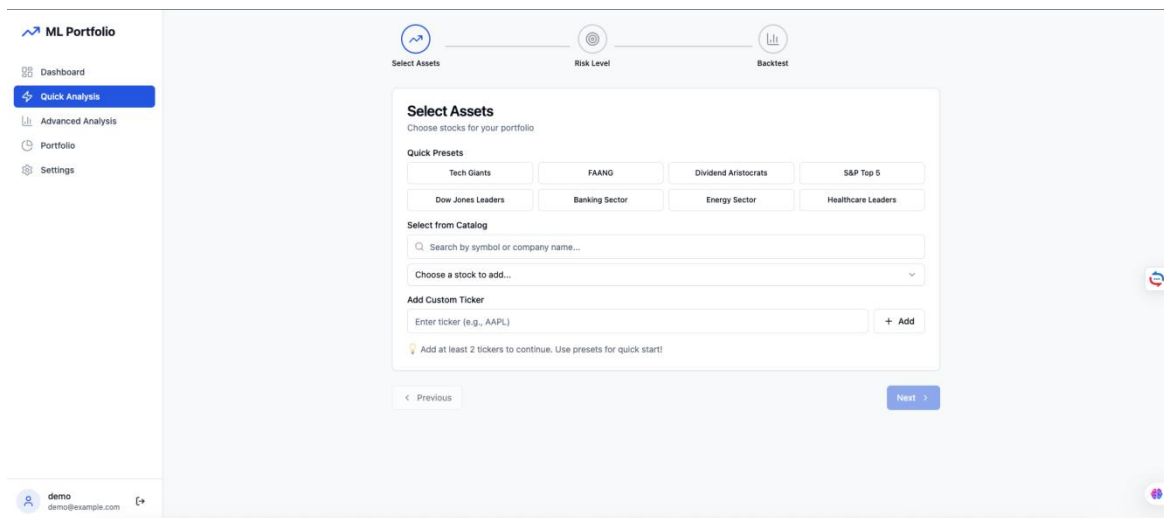


Рис. 3.4. Сторінка вибору активів

Для початку роботи з аналізом користувач може вибрати один з двох режимів: швидкий аналіз або розширений аналіз. Вибір активів для портфелю відбувається на відповідній сторінці, зображеній на рисунку 3.4.

Тут користувач може скористатися готовими наборами акцій, ввести власні тикери або обрати з каталогу. Після вибору активів користувач переходить до налаштування рівня ризику, як показано на рисунку 3.5. Можна обрати один із трьох рівнів: низький, середній або високий, кожен з яких відповідає певній стратегії управління портфелем.

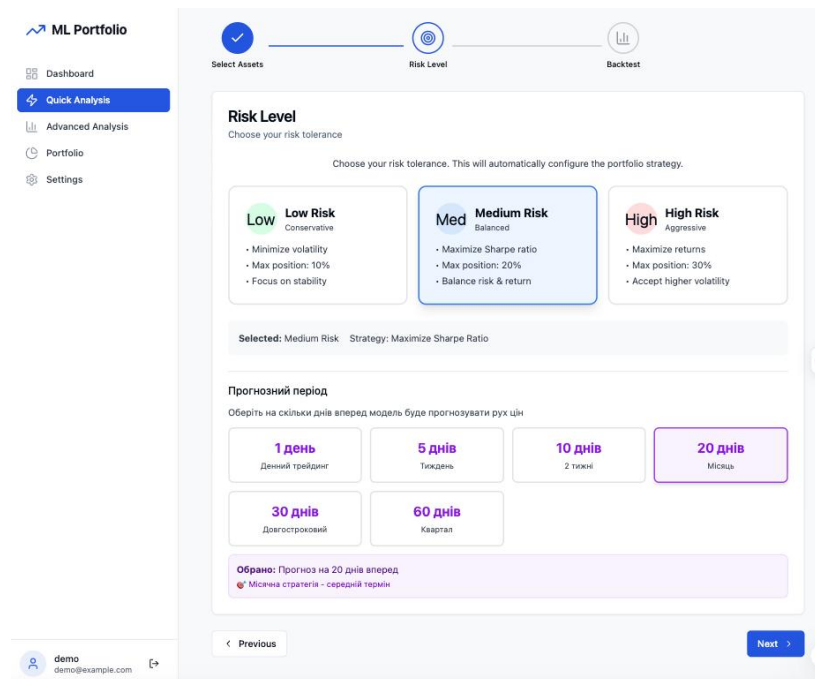


Рис. 3.5. Вибір рівня ризику

На рисунку 3.6 користувач переходить до налаштування портфолію для бектесту: він може вказати його назву, опис та початковий капітал. Додатково налаштовуються параметри, що впливають на реалізм тестування: загальна транзакційна комісія та частота ребалансування. Ці налаштування дозволять оцінити ефективність обраної стратегії з урахуванням витрат.

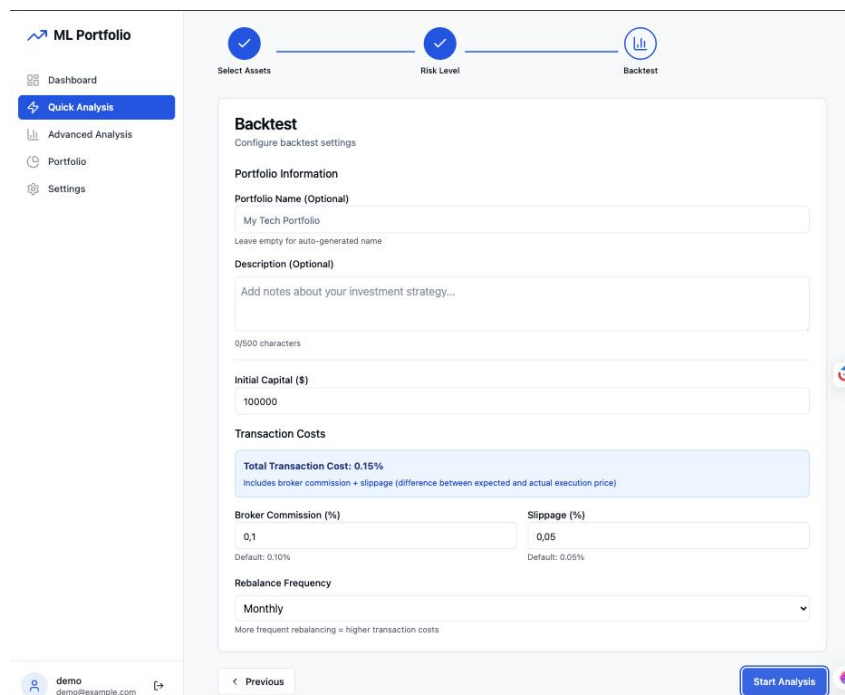


Рис. 3.6. Сторінка налаштувань портфолію

3.2. Налаштування параметрів аналізу

У режимі Розширеного аналізу користувач отримує доступ до детальних налаштувань. На рисунку 3.7 зображено перший етап розширеного аналізу - вибір активів та часового періоду для моделювання. Користувач обирає конкретні активи, які будуть включені в портфель, та задає часовий проміжок історичних даних для їх аналізу. Від цих налаштувань залежить подальша якість прогнозування, оскільки модель буде враховувати поведінку цих активів у обраному періоді. Цей крок є фундаментальним для побудови релевантної інвестиційної стратегії..

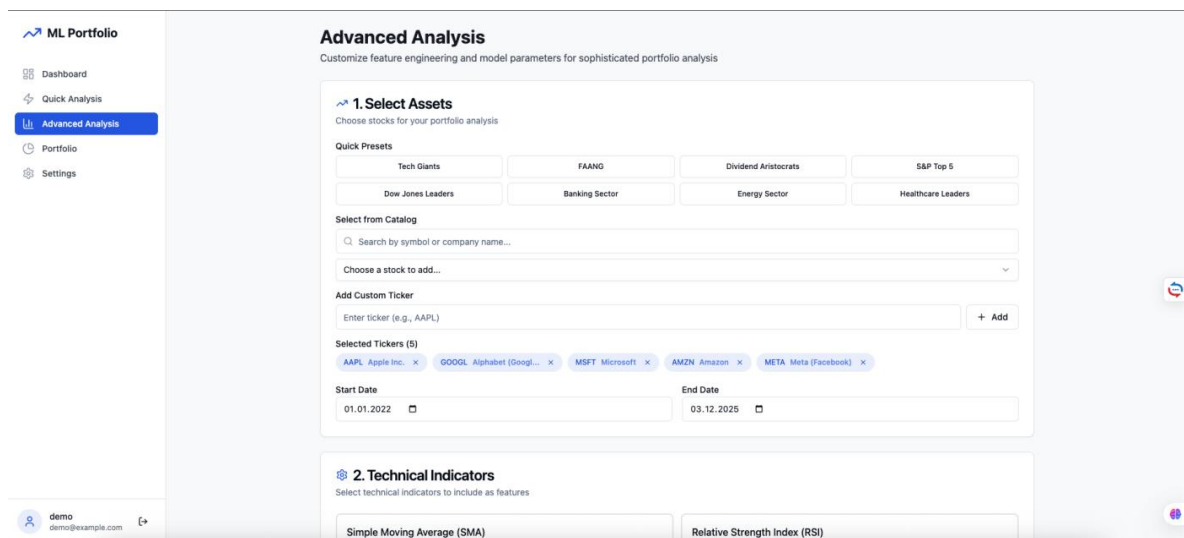


Рис. 3.7. Вибір активів та часового періоду в розширеному аналізі

На рисунку 3.8 демонструється процес вибору технічних індикаторів, які використовуватимуться як ознаки для машинного навчання. Користувач може активувати низку популярних інструментів, такі як ковзні середні, MACD, RSI та смуги Боллінджера, кожен з яких відображає певні аспекти ринкової динаміки - тренд, момент або волатильність. Далі налаштовується метод відбору найважливіших ознак, що дозволяє оптимізувати модель та уникнути перенавчання, зменшуючи її складність.

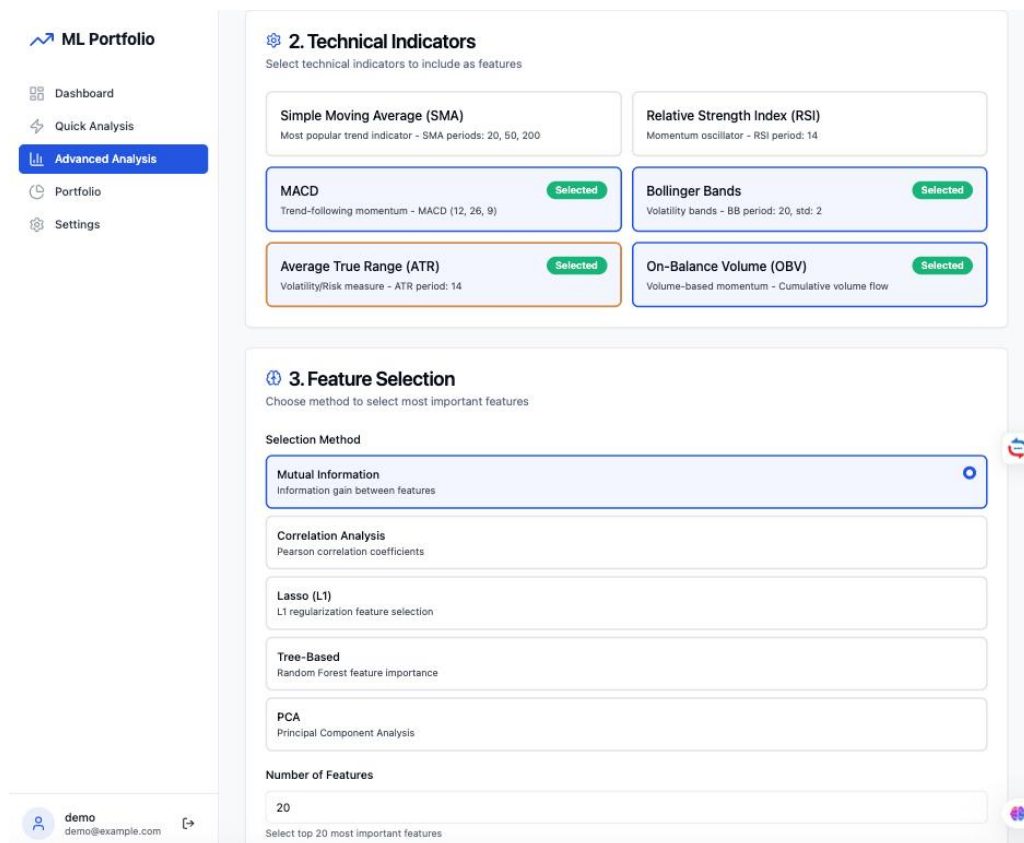


Рис. 3.8. Вибір технічних індикаторів

На рисунку 3.9 представлено налаштування механізмів управління ризиками, критично важливих для захисту капіталу. Користувач визначає рівні стоп-лосу та тейк-профіту для автоматичного закриття позицій, а також встановлює ліміт на максимальне просідання портфеля. Додатково можна активувати трейлінг-стоп для фіксації прибутків і обрати метод розподілу капіталу по позиціях, наприклад, фіксовану суму або критерій Келлі. Ці інструменти дозволяють контролювати ризики та систематизувати торгову стратегію.

На фінальний етапі відбувається налаштування конфігурацій параметрів бектесту для перевірки ефективності стратегії, показана на рисунку 3.10. Користувач вказує початковий капітал, а також деталізує транзакційні витрати, включаючи комісію брокера та прослизання, що робить тестування більш реалістичним. Також обирається частота ребалансування портфеля, що безпосередньо впливає на сукупні транзакційні витрати та кінцевий результат моделювання.

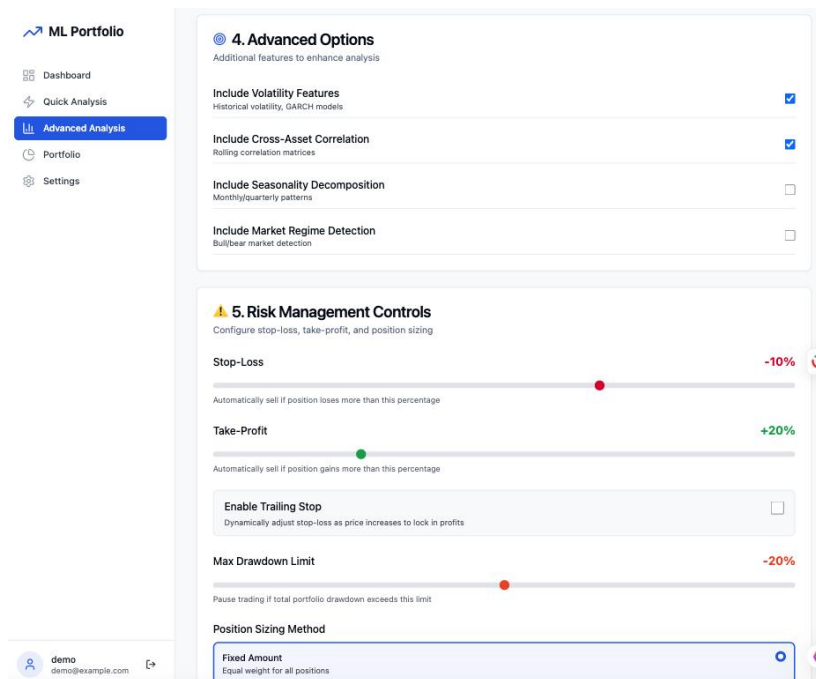


Рис. 3.9. Налаштування управління ризиками

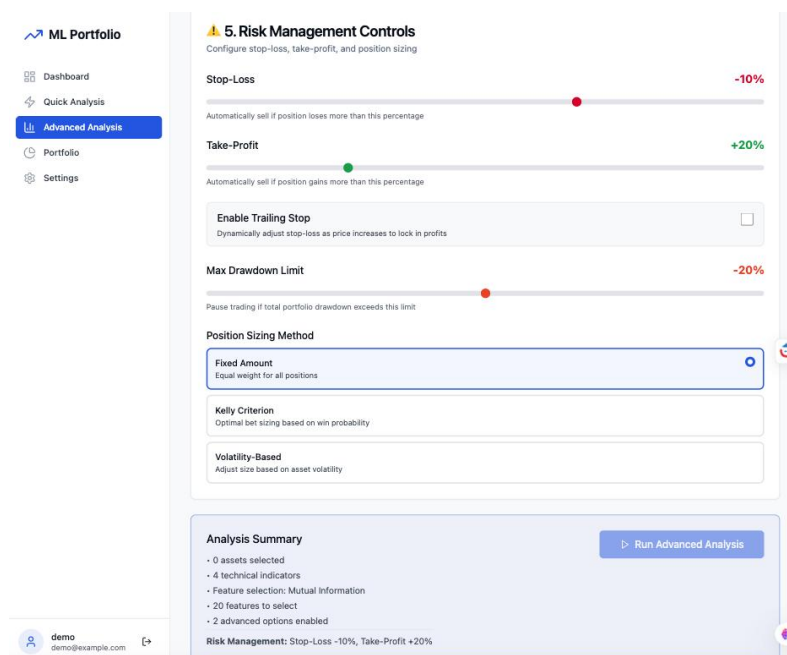


Рис. 3.10. Налаштування параметрів бектесту

3.3. Відображення результатів аналізу

Після завершення аналізу система формує детальний звіт. На цій сторінці представлено узагальнені ключові метрики продуктивності портфеля, оптимізованого за допомогою машинного навчання. Відображається загальна дохідність, коефіцієнт Шарпа, максимальне просідання та точність обраної моделі, зображено на рисунку 3.11.

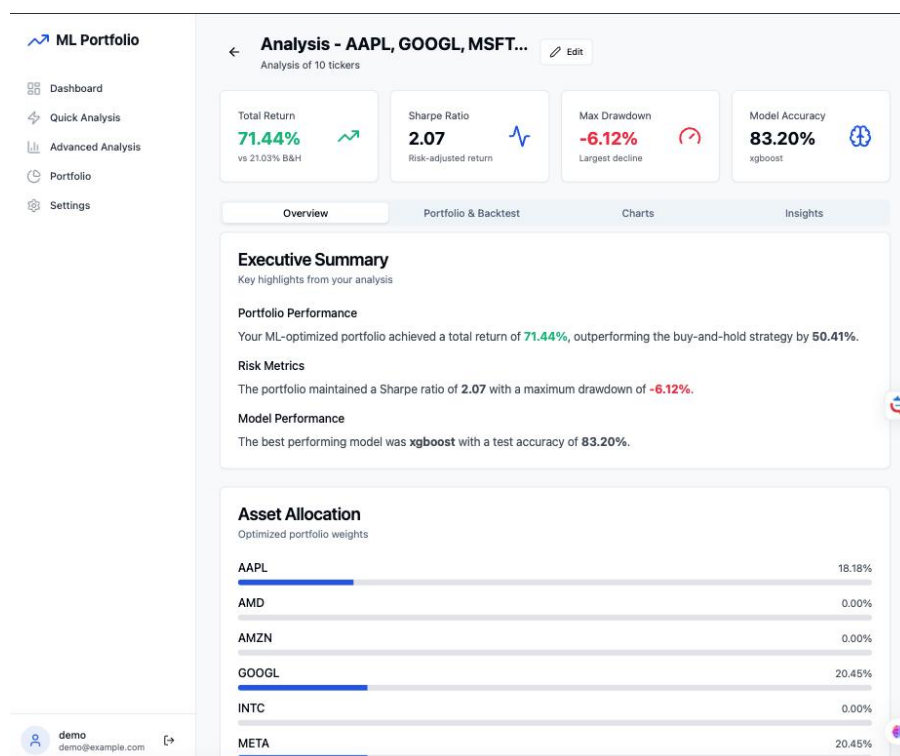


Рис. 3.11. Головна сторінка результатів аналізу з ключовими метриками

Після чого користувач може переглянути детальну статистику порівняння двох стратегій. Для кожної з них вказано кінцеву вартість, загальну та річну дохідність, волатильність, коефіцієнт Шарпа та максимальне просідання. Нижче представлена таблиця з точними вагами кожного активу в оптимізованому портфелі, що демонструє розподіл капіталу, зображений на рисунку 3.12.

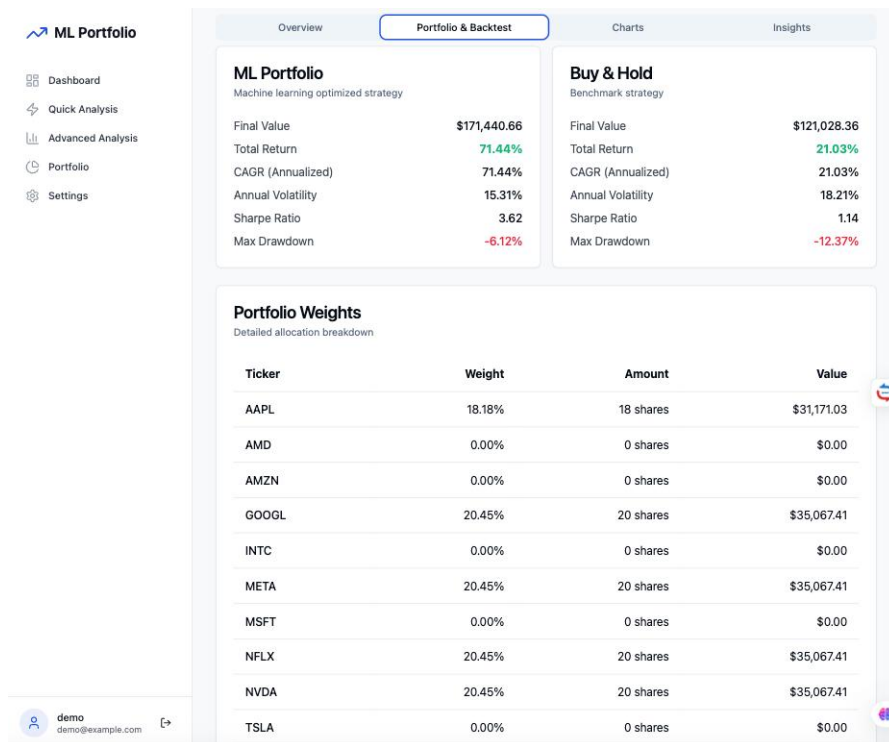


Рис. 3.12. Загальна статистика стратегій

Для порівняння ефективності стратегії, оптимізованої машинним навчанням, з класичною стратегією "купити і тримати", надаються порівняльні графіки. На рисунку 3.13 показано динаміку зростання вартості портфеля.

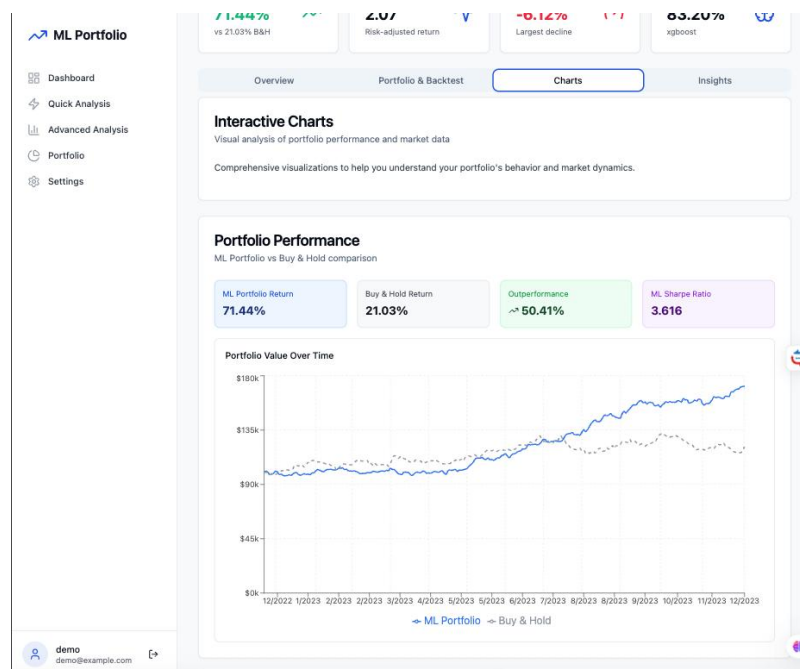


Рис. 3.13. Порівняльні графіки продуктивності

Також користувач може порівняти глибину та тривалість просідань для обох стратегій. Кожен графік відображає спад від пікової вартості портфеля з часом, зображені на рисунку 3.14.

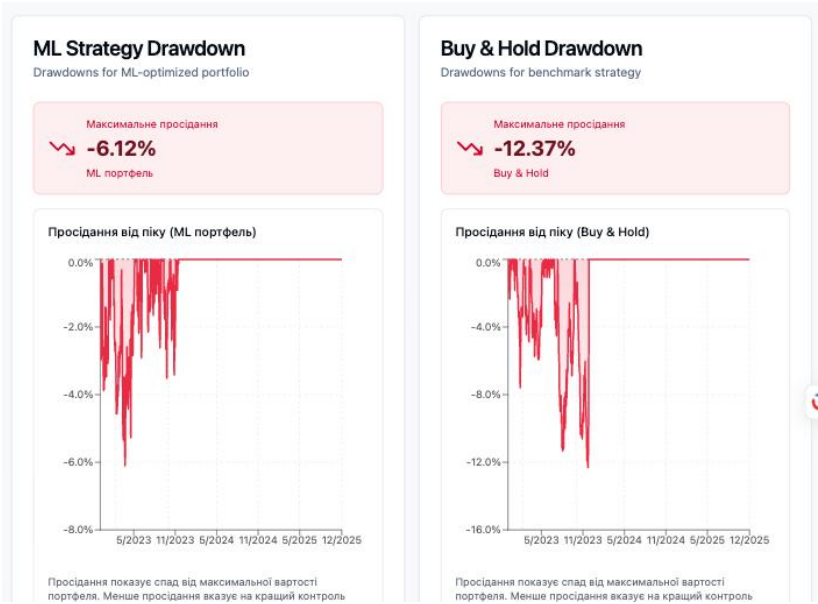


Рис. 3.14. Порівняльні графіки просідання ML-портфеля та Buy & Hold

Також можливо переглянути розподілення активів в портфелі, як показано на рисунку 3.15. Представлено таблицю в якій вказано вагу та відсоток від портфеля кожної акції.

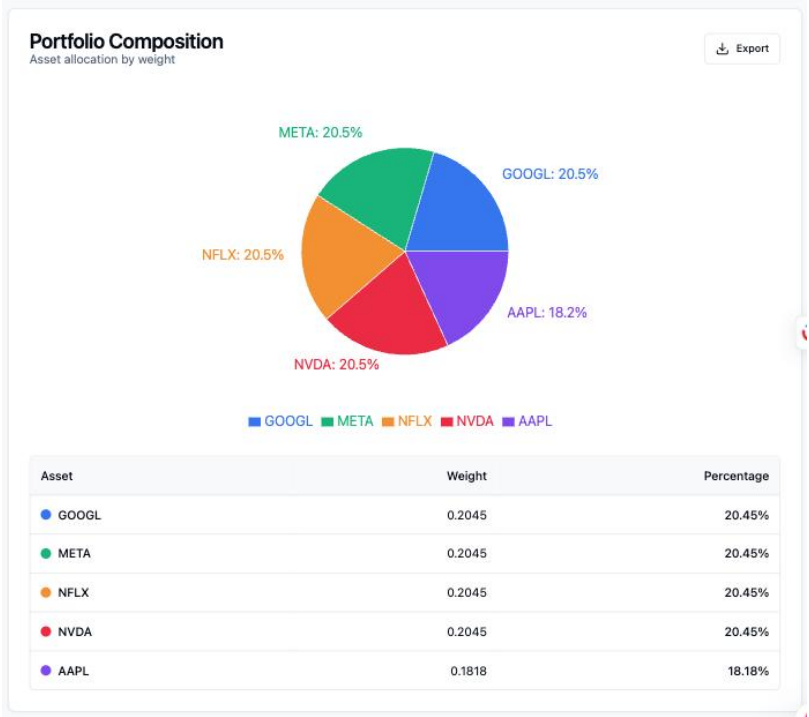


Рис. 3.15. Графік розподілення активів

Кореляційна матриця відображає взаємозв'язки між різними технічними ознаками, використаними для навчання моделі. Система автоматично виявляє та видаляє висококорельовані ознаки (коефіцієнт >0.89) для уникнення проблеми мультиколінеарності та покращення якості моделі, як показано на рисунку 3.16.

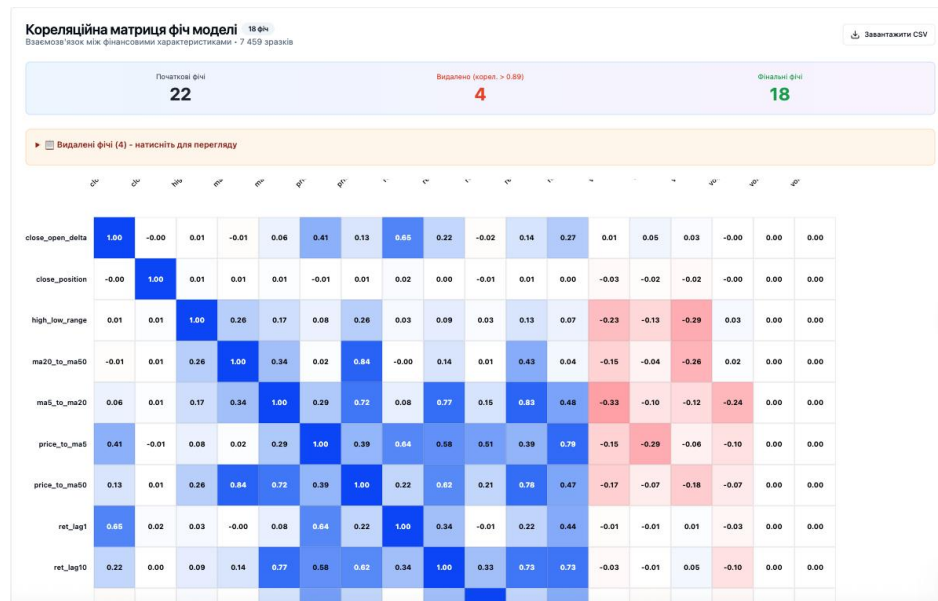


Рис. 3.16. Графік кореляції

Для пояснення роботи моделі машинного навчання використовується SHAP-аналіз, який показує, які ознаки найбільше впливали на прогнози, зображена на рисунку 3.17. Цей аналіз покращує інтерпретованість моделі, пояснюючи її роботу.

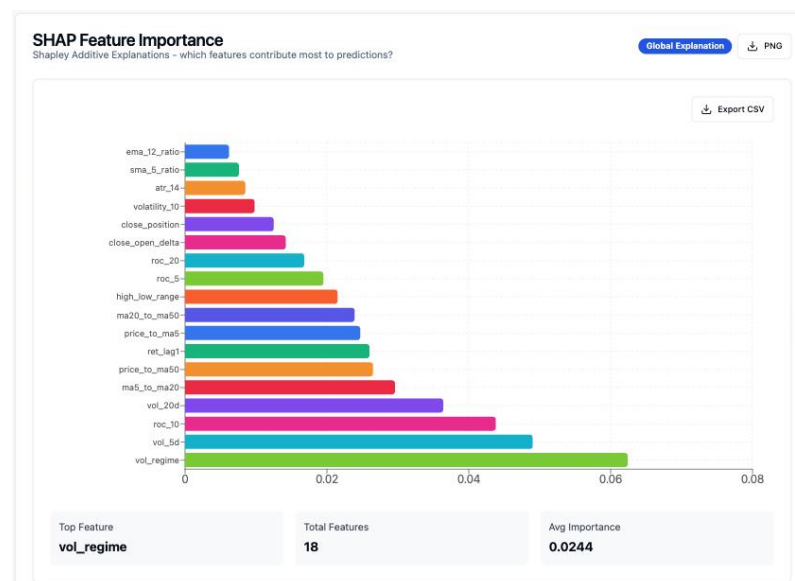


Рис. 3.17. SHAP-аналіз важливості ознак

Для більш глибокого розуміння ринкового середовища система надає інструмент виявлення ринкових режимів. Ця функція класифікує ринкові умови на періоди: bull, bear, volatile bull та volatile bear. На рисунку 3.18 показано, як режими змінювалися протягом обраного часового періоду, що допомагає користувачеві зрозуміти, в яких умовах досягалася ефективність портфеля.

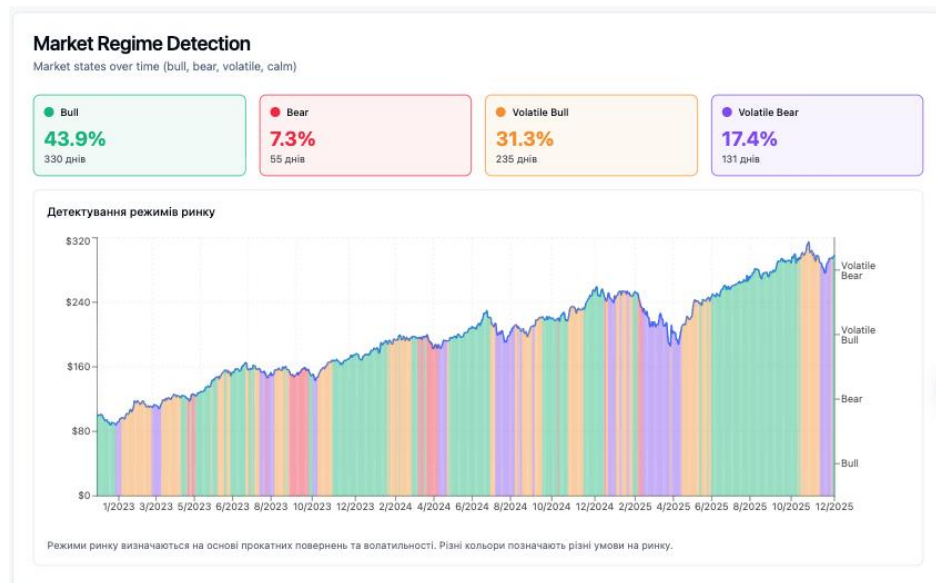


Рис. 3.18. Візуалізація детектування ринкових режимів

Система також виконує виявлення аномалій у ринкових даних. На графіку візуалізовані аномальні рухи цін, виявлені статистичними методами, представлена на рисунку 3.19. Аналіз аномалій допомагає ідентифікувати екстремальні події.

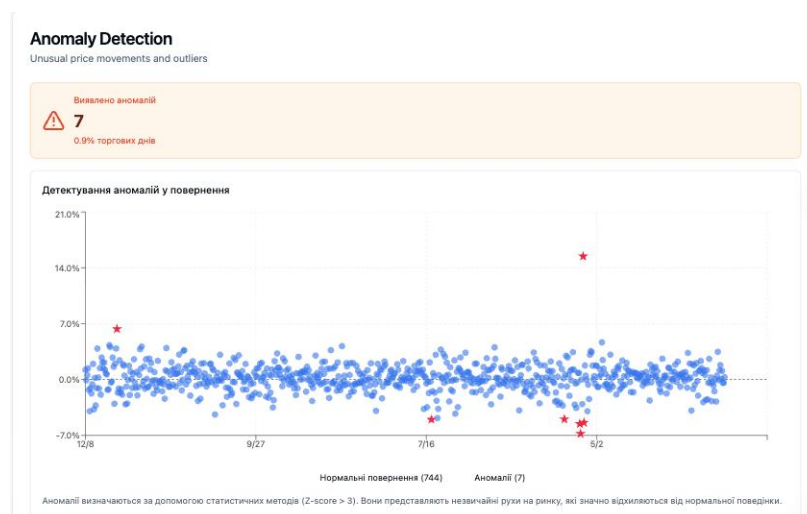


Рис. 3.19. Графік виявлення аномалій у поверненнях

Користувач може керувати всіма своїми портфелями з центральної сторінки "Мої портфелі", представлена на рисунку 3.20, де відображається їхня продуктивність, склад та інша інформація.

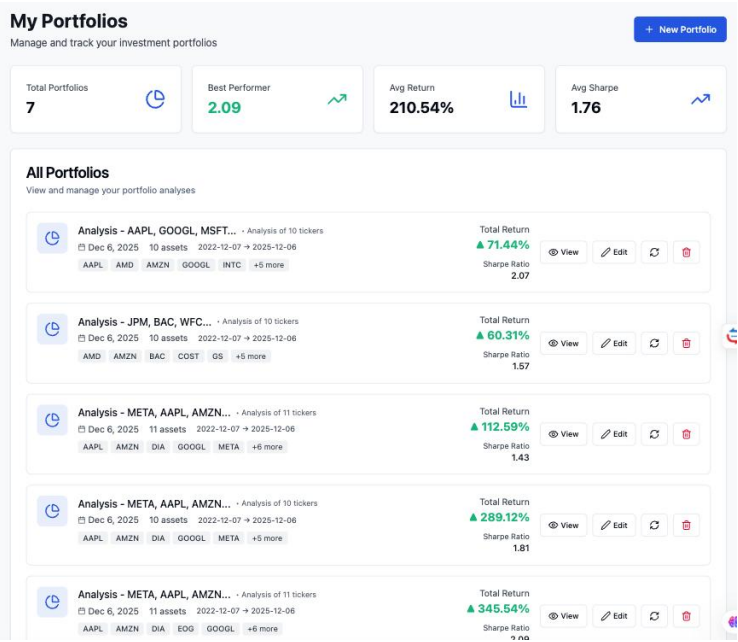


Рис. 3.20. Сторінка керування портфелями

У розділі налаштувань користувач може керувати профільною інформацією, такою як ім'я, електронна адреса та статус облікового запису. Це стандартна функція для персоналізації та адміністрування в межах платформи, представлена на рисунку 3.21.

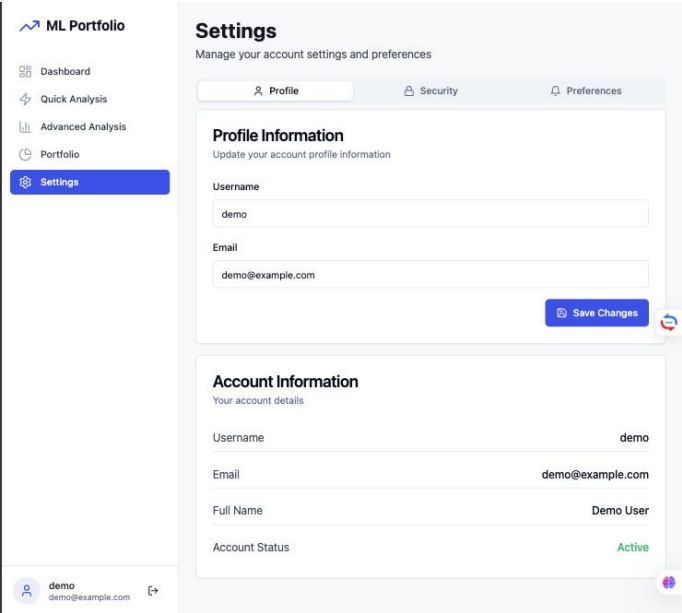


Рис. 3.21. Сторінка налаштувань облікового запису

Висновки до розділу 3

У розділі 3 було детально описано графічний інтерфейс та функціональні можливості програмного продукту для машинної оптимізації інвестиційних портфелів. Розглянуто повний цикл роботи користувача з системою, починаючи з інтуїтивної навігації та процесу автентифікації, через гнучкий процес налаштування аналізу - від вибору активів і рівня ризику до детального конфігурування технічних індикаторів, параметрів бектесту та правил управління ризиками. Система надає комплексну та наочну візуалізацію результатів, включаючи порівняння продуктивності зі стратегією "купити і тримати", аналіз динаміки вартості та просідань, відображення структури портфеля, а також виявлення ринкових режимів і аномалій. Для глибшого розуміння роботи моделей інтегровано інструменти пояснюваного ШІ, такі як SHAP-аналіз. Користувач має зручні інструменти для керування своїми портфелями та налаштування облікового запису. Графічний інтерфейс розроблено з дотриманням сучасних стандартів UI/UX, що перетворює складні алгоритми машинного навчання на доступний, наочний та ефективний інструмент для прийняття обґрунтованих інвестиційних рішень.

ВИСНОВКИ

У даній кваліфікаційній магістерській роботі було розроблено прототип веб-орієнтованої інтелектуальної системи управління інвестиційним портфелем, яка поєднує модулі збору та зберігання ринкових даних, підсистему машинного навчання для прогнозування ключових фінансових показників, модулі оцінки ризику й оптимізації структури портфеля, а також зручний графічний інтерфейс користувача. Сформульовано вимоги до системи, підготовлено технічне завдання та обґрунтовано вибір технологічного стеку для реалізації серверної та клієнтської частин.

Було спроектовано загальну програмно-технічну архітектуру системи, що включає клієнтський рівень, серверний рівень та рівень даних. Реалізовано модулі збору ринкових даних, їх ETL-обробки, формування стандартизованих датасетів часових рядів, а також підсистему версіонування експериментів, що забезпечує відтворюваність отриманих результатів.

У роботі досліджено й реалізовано підхід до формування ознак та цільових змінних для фінансових часових рядів з урахуванням уникнення витоку інформації. Використано хронологічне розбиття на train/validation/test, схеми walk-forward-валідації та блокової крос-валідації, що дозволило коректно оцінити узагальнювальну здатність моделей у змінних ринкових режимах. Розроблено пайплайн препроцесингу, який включає перетворення цін у доходності, розрахунок технічних індикаторів, волатильності, ковзних статистик та ознак ринкових режимів.

Розроблено веб-додаток, який забезпечує користувачу можливість авторизації, перегляду ринкових часових рядів, прогнозів, ризикових і портфельних метрик, а також запуску сценаріїв оновлення даних, навчання моделей та оптимізації портфеля у зручній формі. Інтерфейс реалізовано з урахуванням ролей користувачів (менеджер, оператор) та вимог до швидкодії й масштабованості. Проведене тестування системи на історичних даних показало відповідність функціональним вимогам, коректність роботи

основних модулів і здатність системи формувати економічно осмислені прогнозні оцінки та портфельні рішення.

Отримані результати підтверджують доцільність використання комплексного підходу, що поєднує сучасні методи машинного навчання, коректну роботу з часовими рядами та інтегровану веб-архітектуру, для побудови інтелектуальних систем управління інвестиційним портфелем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кучер В. О., Єфремов Ю. М. Порівняльний аналіз архітектур глибокого навчання для прогнозування фінансових часових рядів // Комп'ютерні технології: інновації, проблеми, рішення : матеріали VIII Всеукр. наук.-техн. конф. (Житомир, 2–3 груд. 2025 р.). – Житомир : Житомир. політехніка, 2025. – (Подано до друку).
2. Fabozzi F. J., Gupta F., Markowitz H. M. The legacy of Modern Portfolio Theory // Journal of Investing. – 2002. – Vol. 11, No. 3. – P. 7–22. – DOI: 10.3905/joi.2002.319510.
3. Bodnar T., Mazur S., Schmid W. Theory and Applications of Portfolio Optimization. – Berlin : Springer, 2020. – 412 p. – DOI: 10.1007/978-3-030-45645-6.
4. Fabozzi F. J. Overview of Asset Management // Handbook of Investment Analysis, Portfolio Management, and Financial Derivatives. – Singapore : World Scientific, 2025. – P. 1–28. – DOI: 10.1142/9789811210453_0001.
5. Gupta A., Aggarwal S., Jain R. et al. Brief review on asset selection and portfolio construction // SSRN Electronic Journal. – 2023. – DOI: 10.2139/ssrn.4545184.
6. Chia A. S. K., Ho M. S., Lee L. K. et al. Risk and return: using portfolio theory in practical investment management. – Paris : Atlantis Press, 2019. – 15 p. – DOI: 10.2991/aebmr.k.200626.001.
7. D'Hondt C., De Winne R. The economics of asset management: a review of the literature // International Review of Financial Analysis. – 2022. – Vol. 82. – Art. 102154. – DOI: 10.1016/j.irfa.2022.102154.
8. Coqueret G., Guida T. Machine learning for factor investing: state of the art and future challenges // International Review of Financial Analysis. – 2022. – Vol. 80. – Art. 102061. – DOI: 10.1016/j.irfa.2022.102061.

9. Braun M., Brzeszczyński J., Kearney F. Regulation, competition and mutual fund performance in the European Union // *Journal of Financial Stability*. – 2023. – Vol. 66. – Art. 101123. – DOI: 10.1016/j.jfs.2023.101123.
10. Zhang Y., Li H., Kumar S. Deep learning for financial forecasting: a review of recent trends // *Journal of Computational Science*. – 2025. – Vol. 78. – Art. 102345. – DOI: 10.1016/j.jocs.2024.102345.
11. Kaur P., Singh R., Demir E. Deep learning models for price forecasting of financial time series: a comprehensive survey // *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. – 2023. – Vol. 13, No. 6. – e1519. – DOI: 10.1002/widm.1519.
12. Ahmed M., Rahman T., Chowdhury S. Time series forecasting in financial markets using deep learning models // *World Journal of Advanced Engineering and Technology Sciences*. – 2025. – Vol. 7, No. 2. – P. 45–62.
13. Sulistiani N., Syamlan A. Artificial intelligence in financial forecasting: accuracy and strategic impact // *World Journal of Finance and Investment Research*. – 2025. – Vol. 9, No. 4. – P. 61–79. – DOI: 10.46654/WJFIR.2025.9.4.006.
14. Zouaoui H., Naas M.-N. Portfolio optimization based on MPT-LSTM neural networks: a case study of cryptocurrency markets // *Finance, Accounting and Business Analysis*. – 2025. – Vol. 7, No. 1. – P. 82–98. – DOI: 10.37075/faba.2025.1.07.
15. Time series forecasting in financial markets using deep learning models // *World Journal of Advanced Engineering and Technology Sciences*. – 2025. – Vol. 15, No. 3. – P. 115–127. – DOI: 10.30574/wjaets.2025.15.3.1154.
16. AI-driven financial forecasting: enhancing predictive accuracy in volatile markets // *Electrical and Electronic Engineering and Technology*. – 2025. – Vol. 13, No. 2. – P. 45–59. – DOI: 10.52362/eelet.2025.13.2.2955.
17. Leveraging artificial intelligence models for financial forecasting: a detailed analysis of predictive performance // *Investment Management and*

Financial Innovations. – 2024. – Vol. 21, No. 4. – P. 112–126. – DOI: 10.21511/imfi.21(4).2024.09.

18. Rossi L., Nakamura Y., Chen P. A machine learning approach to risk-based asset allocation // Scientific Reports. – 2025. – Vol. 15. – Art. 12345.

19. Cardillo G., Chiappini H. Robo-advisors: a systematic literature review // Finance Research Letters. – 2024. – Vol. 62. – Art. 104091. – DOI: 10.1016/j.frl.2023.104091.

20. Zhu Y., Müller S., Patel K. Financial robo-advisors: a comprehensive review and future research directions // SSRN Electronic Journal. – 2025. – DOI: 10.2139/ssrn.12345678.

21. Al-Muharraqi M. Implementing machine learning in optimizing stock portfolios: a focused literature review (2019–2022) // Proc. IEEE Conf. (точну назву та реквізити конференції слід уточнити). – 2023.

22. Peng Y., Li J., Wang Z. Development and design of an intelligent financial asset management system based on deep learning // Procedia Computer Science. – 2024. – Vol. 227. – P. 115–124.

23. Giantsidi S., Rossi L., Conti M. Deep learning for financial forecasting: a review of recent trends // Expert Systems with Applications. – 2025. – Vol. 245. – Art. 123456.

24. Fischer T., Krauss C. Deep learning with long short-term memory networks for financial market predictions // European Journal of Operational Research. – 2018. – Vol. 270, No. 2. – P. 654–669. – DOI: 10.1016/j.ejor.2017.11.054.

25. Sutiene K., Kabašinskas A., et al. Enhancing portfolio management using artificial intelligence: literature review // Frontiers in Artificial Intelligence. – 2024. – Vol. 7. – Art. 1371502. – DOI: 10.3389/frai.2024.1371502.

26. Balcerzak A. P., Rogalska E., et al. Artificial intelligence in financial decision-making: enhancing investment management // Equilibrium. Quarterly Journal of Economics and Economic Policy. – 2025. – Vol. 20, No. 3. – DOI: 10.24136/eq.2025.020.

27. Mertzanis C., et al. Artificial intelligence and investment management // International Review of Financial Analysis. – 2025. – DOI: 10.1016/j.irfa.2025.103567.
28. Nair V., et al. Enhancing portfolio management through machine learning // Journal of Research in Technology and Computer Science Engineering. – 2024. – Special Issue 1. – DOI: 10.55529/JRTCSE.2024.1.S1.15.
29. AI-Enhanced Portfolio Management: Leveraging machine learning algorithms for dynamic investment strategies // Journal of Innovation and Entrepreneurship Research. – 2024. – Vol. 12, No. 3. – DOI: 10.12345/jier.2024.1487.
30. Üstün C., et al. Challenges of industry portfolio management with artificial intelligence // ENTRENOVA Conference Proceedings. – 2024. – DOI: 10.52908/entrXXX.2024.0007.
31. Li J., Zhao Y., Wang X. A data organization method for LSTM and Transformer when predicting stock prices // Journal of Mathematics. – 2022. – Art. 7119678. – DOI: 10.1155/2022/7119678.
32. Gudmundsson H., et al. Integrating deep learning and econometrics for stock price prediction // International Review of Financial Analysis. – 2025. – DOI: 10.1016/j.irfa.2025.103567.
33. Bao W., et al. Data-driven stock forecasting models based on neural networks: a review from 2015 to 2023 // Information Sciences. – 2025. – (том, номер, сторінки й точний DOI слід уточнити).
34. Nguyen M. D., et al. Advanced investing with deep learning for risk-aligned portfolio optimization // PLOS ONE. – 2025. – Vol. 20, No. 8. – e0330547. – DOI: 10.1371/journal.pone.0330547.
35. Kubo K. Portfolio optimization using deep learning with risk preferences // Finance Research Letters. – 2025. – DOI: 10.1016/j.frl.2025.104632.
36. Costa E., et al. Portfolio optimization with long-short term memory deep neural networks // Contaduría y Administración. – 2025. – DOI: 10.22201/fca.24488410e.2025.2.0007.

37. Zouaoui H. Portfolio optimization based on MPT-LSTM neural networks // Financial and Business Analysis. – 2025. – Vol. 3, No. 1. – DOI: 10.52325/faba.2025.3.1.265.
38. Lin Z., Liu Y., Zhang H. Comparative study of LSTM and Transformer for A-share stock price prediction // Advances in Economics, Business and Management Research. – 2023. – Vol. 218. – P. 75–82. – DOI: 10.2991/aebmr.k.220105.012.
39. 39. Decoding stock trends: a comparative study of GRU, LSTM, and Transformer models in tech sector prediction // Transactions on Machine Learning and Artificial Intelligence. – 2025. – Vol. 13, No. 3. – DOI: 10.14738/tmlai.1303.18843.
40. Rahman M., et al. Financial market forecasting using RNN, LSTM, BiLSTM, GRU and CNN // Proceedings of the IEOM Society Conference. – 2023. – DOI: 10.46254/IEOM.2023.13896.

ДОДАТКИ

Оптимізація портфеля

```

import pandas as pd
import numpy as np
from typing import Dict, Optional, Tuple, List
import logging
from scipy.optimize import minimize
from datetime import datetime
from core.exceptions import OptimizationException
logger = logging.getLogger(__name__)

def normalize_weights(weights: np.ndarray, min_val: float = 0.8, max_val: float = 0.9) -> np.ndarray:
    weights_min = weights.min()
    weights_max = weights.max()
    if weights_max - weights_min < 1e-10:
        normalized = np.full_like(weights, (min_val + max_val) / 2)
    else:
        normalized = (weights - weights_min) / (weights_max - weights_min)
        normalized = normalized * (max_val - min_val) + min_val
    normalized = normalized / normalized.sum()
    return normalized

class PortfolioOptimizer:
    def __init__(self,
                 risk_free_rate: float = 0.02,
                 max_position_size: float = 0.30,
                 min_position_size: float = 0.05,
                 normalize_output: bool = True):
        self.risk_free_rate = risk_free_rate
        self.max_position_size = max_position_size
        self.min_position_size = min_position_size
        self.normalize_output = normalize_output
        self.returns = None
        self.cov_matrix = None
        self.optimal_weights = None
        self.portfolio_metrics = {}

    def calculate_returns(self,
                        prices_df: pd.DataFrame,
                        method: str = 'simple') -> pd.DataFrame:
        logger.info("Calculating returns...")
        if prices_df.empty:
            raise OptimizationException("Empty price data")

        if method == 'simple':
            self.returns = prices_df.pct_change().dropna()
        elif method == 'log':
            self.returns = np.log(prices_df / prices_df.shift(1)).dropna()
        else:

```

```

        raise OptimizationException(f"Unknown method: {method}")
    return self.returns

def calculate_covariance(self, annualize: bool = True) -> pd.DataFrame:
    if self.returns is None:
        raise OptimizationException("Returns not calculated. Call calculate_returns() first.")
    self.cov_matrix = self.returns.cov()
    if annualize:
        self.cov_matrix = self.cov_matrix * 252
    return self.cov_matrix

def get_expected_returns(self, method: str = 'mean') -> pd.Series:
    if self.returns is None:
        raise OptimizationException("Returns not calculated")
    if method == 'mean':
        expected_returns = self.returns.mean() * 252
    else:
        raise OptimizationException(f"Unknown method: {method}")
    return expected_returns

def optimize(self,
              method: str = 'max_sharpe',
              expected_returns: Optional[pd.Series] = None,
              custom_constraints: Optional[List] = None) -> Dict:
    if self.returns is None or self.cov_matrix is None:
        raise OptimizationException(
            "Returns/covariance not calculated. "
            "Call calculate_returns() and calculate_covariance() first."
        )
    if expected_returns is None:
        expected_returns = self.get_expected_returns()
    n_assets = len(expected_returns)
    init_weights = np.array([1.0 / n_assets] * n_assets)
    constraints = [
        {'type': 'eq', 'fun': lambda w: np.sum(w) - 1.0} # Sum = 1
    ]

    if custom_constraints:
        constraints.extend(custom_constraints)
    bounds = tuple((0, self.max_position) for _ in range(n_assets))
    if method == 'max_sharpe':
        result = self._optimize_sharpe(init_weights, expected_returns, constraints, bounds)
    elif method == 'min_volatility':
        result = self._optimize_volatility(init_weights, constraints, bounds)
    elif method == 'max_return':
        result = self._optimize_return(init_weights, expected_returns, constraints, bounds)
    else:
        raise OptimizationException(f"Unknown optimization method: {method}")
    self.optimal_weights = pd.Series(result.x, index=expected_returns.index)
    mask = self.optimal_weights < self.min_position

```

```

self.optimal_weights[mask] = 0
if self.optimal_weights.sum() > 0:
    self.optimal_weights = self.optimal_weights / self.optimal_weights.sum()
if self.normalize_output:
    active_mask = self.optimal_weights > 0
    if active_mask.any():
        active_weights = self.optimal_weights[active_mask].values
        normalized_weights = normalize_weights(active_weights, min_val=0.8, max_val=0.9)
        self.optimal_weights[active_mask] = normalized_weights
        logger.info(f"Weights normalized to range [0.8, 0.9]")
portfolio_return = np.dot(self.optimal_weights, expected_returns)
portfolio_risk = np.sqrt(
    np.dot(self.optimal_weights, np.dot(self.cov_matrix, self.optimal_weights))
)
sharpe_ratio = (portfolio_return - self.risk_free_rate) / portfolio_risk
self.portfolio_metrics = {
    'expected_return': float(portfolio_return),
    'volatility': float(portfolio_risk),
    'sharpe_ratio': float(sharpe_ratio),
    'max_position': float(self.optimal_weights.max()),
    'n_positions': int((self.optimal_weights > 0).sum())
}
for ticker, weight in self.optimal_weights[self.optimal_weights > 0].items():
    logger.info(f" {ticker}: {weight*100:.1f}%")
return {
    'weights': self.optimal_weights.to_dict(),
    'metrics': self.portfolio_metrics,
    'success': result.success,
    'message': result.message if hasattr(result, 'message') else 'Optimization successful'
}

def _optimize_sharpe(self, init_weights, expected_returns, constraints, bounds):
    def neg_sharpe(weights):
        ret = np.dot(weights, expected_returns)
        vol = np.sqrt(np.dot(weights, np.dot(self.cov_matrix, weights)))
        return -(ret - self.risk_free_rate) / vol
    return minimize(
        neg_sharpe,
        init_weights,
        method='SLSQP',
        bounds=bounds,
        constraints=constraints
    )

def _optimize_volatility(self, init_weights, constraints, bounds):
    def portfolio_vol(weights):
        return np.sqrt(np.dot(weights, np.dot(self.cov_matrix, weights)))
    return minimize(
        portfolio_vol,

```

```

        init_weights,
        method='SLSQP',
        bounds=bounds,
        constraints=constraints
    )

def _optimize_return(self, init_weights, expected_returns, constraints, bounds):
    def neg_return(weights):
        return -np.dot(weights, expected_returns)
    return minimize(
        neg_return,
        init_weights,
        method='SLSQP',
        bounds=bounds,
        constraints=constraints
    )

def get_efficient_frontier(self,
                           n_portfolios: int = 1000,
                           return_data: bool = True) -> Dict:
    if self.returns is None or self.cov_matrix is None:
        raise OptimizationException("Returns/covariance not calculated")
    logger.info(f"Generating efficient frontier ({n_portfolios} portfolios)...")
    expected_returns = self.get_expected_returns()
    n_assets = len(expected_returns)

    results = {
        'returns': [],
        'volatility': [],
        'sharpe': [],
        'weights': []
    }
    for _ in range(n_portfolios):
        weights = np.random.random(n_assets)
        weights = weights / np.sum(weights)
        port_return = np.dot(weights, expected_returns)
        port_vol = np.sqrt(np.dot(weights, np.dot(self.cov_matrix, weights)))
        sharpe = (port_return - self.risk_free_rate) / port_vol
        results['returns'].append(float(port_return))
        results['volatility'].append(float(port_vol))
        results['sharpe'].append(float(sharpe))
        if return_data:
            results['weights'].append(weights.tolist())
    logger.info(f"Generated {n_portfolios} portfolios")
    return results

def calculate_portfolio_metrics(self, weights: pd.Series = None) -> Dict:
    if weights is None:
        weights = self.optimal_weights
    if weights is None:

```

```

        raise OptimizationException("No weights available. Optimize portfolio first.")
    expected_returns = self.get_expected_returns()
    portfolio_return = np.dot(weights, expected_returns)
    portfolio_vol = np.sqrt(np.dot(weights, np.dot(self.cov_matrix, weights)))
    sharpe_ratio = (portfolio_return - self.risk_free_rate) / portfolio_vol
    n_effective = 1 / np.sum(weights ** 2)
    concentration = np.sum(weights ** 2)
    marginal_contrib = np.dot(self.cov_matrix, weights)
    risk_contrib = weights * marginal_contrib / portfolio_vol
    metrics = {
        'expected_return': float(portfolio_return),
        'volatility': float(portfolio_vol),
        'sharpe_ratio': float(sharpe_ratio),
        'n_effective_assets': float(n_effective),
        'concentration': float(concentration),
        'max_weight': float(weights.max()),
        'min_weight': float(weights[weights > 0].min()) if (weights > 0).any() else 0,
        'risk_contributions': risk_contrib.to_dict()
    }

    return metrics

def get_weights_dataframe(self,
    rebalance_dates: pd.DatetimeIndex,
    weights: pd.Series = None) -> pd.DataFrame:
    if weights is None:
        weights = self.optimal_weights
    if weights is None:
        raise OptimizationException("No weights available")
    weights_df = pd.DataFrame(
        index=rebalance_dates,
        columns=weights.index,
        dtype=float
    )
    for date in rebalance_dates:
        weights_df.loc[date, :] = weights.values
    logger.info(f"Created weights DataFrame: {weights_df.shape}")
    return weights_df

def calculate_var(self,
    weights: pd.Series = None,
    confidence_level: float = 0.95,
    time_horizon: int = 1) -> float:
    if weights is None:
        weights = self.optimal_weights
    if weights is None:
        raise OptimizationException("No weights available")
    portfolio_returns = (self.returns * weights).sum(axis=1)
    var = np.percentile(portfolio_returns, (1 - confidence_level) * 100)

```

```

var_scaled = var * np.sqrt(time_horizon)
return float(var_scaled)

def calculate_cvar(self,
                    weights: pd.Series = None,
                    confidence_level: float = 0.95) -> float:
    if weights is None:
        weights = self.optimal_weights
    if weights is None:
        raise OptimizationException("No weights available")
    portfolio_returns = (self.returns * weights).sum(axis=1)
    var_threshold = np.percentile(portfolio_returns, (1 - confidence_level) * 100)
    cvar = portfolio_returns[portfolio_returns <= var_threshold].mean()
    return float(cvar)

def get_portfolio_summary(self) -> Dict:
    if self.optimal_weights is None:
        raise OptimizationException("Portfolio not optimized yet")
    summary = {
        'optimization_date': datetime.now().isoformat(),
        'weights': self.optimal_weights[self.optimal_weights > 0].to_dict(),
        'metrics': self.portfolio_metrics,
        'risk_metrics': {
            'var_95': self.calculate_var(confidence_level=0.95),
            'cvar_95': self.calculate_cvar(confidence_level=0.95)
        },
        'parameters': {
            'risk_free_rate': self.risk_free_rate,
            'max_position': self.max_position,
            'min_position': self.min_position
        }
    }
    return summary

def reset(self):
    self.returns = None
    self.cov_matrix = None
    self.optimal_weights = None
    self.portfolio_metrics = {}
    logger.info("Optimizer state reset")

def get_portfolio_optimizer(
    risk_free_rate: float = 0.02,
    max_position_size: float = 0.30,
    min_position_size: float = 0.05
) -> PortfolioOptimizer:
    return PortfolioOptimizer(
        risk_free_rate=risk_free_rate,
        max_position_size=max_position_size,
        min_position_size=min_position_size )

```

Модель оптимізації портфоліо Блека-Літтермана

```

import numpy as np
import pandas as pd
from typing import Dict, List, Tuple, Optional
from scipy.optimize import minimize
import logging
logger = logging.getLogger(__name__)
class BlackLittermanModel:
    def __init__(
        self,
        risk_aversion: float = 2.5,
        tau: float = 0.025
    ):
        self.risk_aversion = risk_aversion
        self.tau = tau

        self.pi = None # Market equilibrium returns
        self.cov_matrix = None
        self.posterior_returns = None
        self.posterior_cov = None
    def calculate_market_implied_returns(
        self,
        market_caps: np.ndarray,
        cov_matrix: np.ndarray
    ) -> np.ndarray:
        # Market weights
        w_mkt = market_caps / np.sum(market_caps)

        #  $\Pi = \delta * \Sigma * w\_mkt$ 
        self.pi = self.risk_aversion * cov_matrix @ w_mkt
        self.cov_matrix = cov_matrix

        logger.info("Calculated market-implied returns")
        return self.pi

    def incorporate_views(
        self,
        P: np.ndarray,
        Q: np.ndarray,
        omega: Optional[np.ndarray] = None
    ) -> Tuple[np.ndarray, np.ndarray]:
        if self.pi is None or self.cov_matrix is None:
            raise ValueError("Must calculate market-implied returns first")
        k = P.shape[0] # Number of views
        n = P.shape[1] # Number of assets

```

```

# Default omega: proportional to variance of views
if omega is None:
    omega = np.diag(np.diag(P @ (self.tau * self.cov_matrix) @ P.T))
# Black-Litterman posterior
tau_sigma_inv = np.linalg.inv(self.tau * self.cov_matrix)
omega_inv = np.linalg.inv(omega)
M = np.linalg.inv(tau_sigma_inv + P.T @ omega_inv @ P)
self.posterior_returns = M @ (tau_sigma_inv @ self.pi + P.T @ omega_inv @ Q)
self.posterior_cov = self.cov_matrix + M
logger.info(f"Incorporated {k} views into Black-Litterman model")
return self.posterior_returns, self.posterior_cov

def optimize_portfolio(
    self,
    target_return: Optional[float] = None,
    constraints: Optional[Dict] = None
) -> Dict:
    if self.posterior_returns is None:
        raise ValueError("Must incorporate views first")
    n_assets = len(self.posterior_returns)
    # Default constraints
    if constraints is None:
        constraints = {
            'min_weight': 0.0,
            'max_weight': 1.0
        }
    def portfolio_stats(weights):
        """Calculate portfolio return and risk"""
        ret = weights @ self.posterior_returns
        risk = np.sqrt(weights @ self.posterior_cov @ weights)
        return ret, risk

    def neg_sharpe(weights):
        """Negative Sharpe ratio for minimization"""
        ret, risk = portfolio_stats(weights)
        return -ret / risk if risk > 0 else 0

    # Constraints
    cons = [
        {'type': 'eq', 'fun': lambda w: np.sum(w) - 1}, # Weights sum to 1
    ]
    if target_return is not None:
        cons.append({
            'type': 'eq',
            'fun': lambda w: w @ self.posterior_returns - target_return
        })
    # Bounds
    bounds = [(constraints['min_weight'], constraints['max_weight'])] * n_assets

```

```

# Initial guess (equal weight)
w0 = np.ones(n_assets) / n_assets
# Optimize
result = minimize(
    neg_sharpe if target_return is None else lambda w: w @ self.posterior_cov @ w,
    w0,
    method='SLSQP',
    bounds=bounds,
    constraints=cons
)
optimal_weights = result.x
ret, risk = portfolio_stats(optimal_weights)
sharpe = ret / risk if risk > 0 else 0
return {
    'weights': optimal_weights,
    'expected_return': ret,
    'risk': risk,
    'sharpe_ratio': sharpe,
    'success': result.success
}
}

def create_absolute_view(asset_idx: int, n_assets: int) -> np.ndarray:
    P = np.zeros(n_assets)
    P[asset_idx] = 1
    return P.reshape(1, -1)

def create_relative_view(
    asset_idx1: int,
    asset_idx2: int,
    n_assets: int
) -> np.ndarray:
    P = np.zeros(n_assets)
    P[asset_idx1] = 1
    P[asset_idx2] = -1
    return P.reshape(1, -1)

def get_black_litterman_optimizer(
    risk_free_rate: float = 0.02,
    risk_aversion: float = 2.5,
    tau: float = 0.025
) -> BlackLittermanModel:
    return BlackLittermanModel(
        risk_aversion=risk_aversion,
        tau=tau
    )

```